The Pencil Code:

MHD 湍流的高阶 MPI 模拟包

用户参考手册

2018 年 10 月 23 日

http://www.nordita.org/software/pencil-code/

https://github.com/pencil-code/pencil-code

前言 Foreword

这一代码包一开始是在一个暑期学校 Turbulence Summer School of the Helmholtz Institute in Potsdam (2001) 期间发展起来的. 虽然一些流体力学和吃流体力学的 SPH 和 PPM 的代码都是公开的, 但这不代表高阶微分法或谱分析代码也被公开了. 已经有不少对此感兴趣的朋友接触过我们, 基于此, 我们决定让其使用起来更灵活并尽可能地对用户友好一些, 于是这个程序包便被放到了公有 CVS 库上. 自从 2008 年 9 月 21 号起, 它便通过 https://github.com/pencil-code/pencil-code 发布了出来. https://github.com/pencil-code/pencil-code. 这段代码不应该被视作黑箱, 为了用一种较为优化的方式解决新出现的问题, 用户将会需要自己配置合适的参数. 特别的, 你需要小心地对一些参数选择正确的数值, 比如说粘度, 磁扩散率和辐射热传导率等.

Pencil Code 主要用来解决弱可压缩的湍流问题, 这也是为什么我们用了高阶的一次和二次导数. 为了实现较好的并行化预算, 我们使用了显式有限差分法 (与之相对的是紧致的有限差分法). 本程序包典型的科学目标包括周期边界条件长方体内受迫的 MHD 湍流; 平面板上下边界条件非周期性, 研究对流运行; 嵌在完全非周期盒中的对流星 (翻译存疑 a convective star embedded in a fully nonperiodic box); 吸积盘湍流的剪切片近似等等. 此外, 我们还有非局域辐射输运, 惯性粒子, 尘埃凝聚, 自引力, 化学反应网络等多种物理元件, 其数量在稳步增加. 除了笛卡尔坐标系外, 该代码还可以处理球极坐标系和柱极坐标系.

磁场是通过磁矢势来实现的, 这可以保证磁场维持环形, 而不夹杂任何散度. 同时, 用磁矢势还有一个巨大的好处, 例如, 当研究者想要控制磁场螺旋度时, 磁矢势会非常方便. 从而使得本程序包非常适合于各类 dynamo (翻译存疑) 问题.

Pencil Code 一般来说不保证守恒量守恒; 所以, 守恒量的稳定性将会依赖于算法的离散误差 (不是指机器精度). 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.

目前没有计划实现自适应的网格改善算法, 则会让 Pencil Code 技术上变得非常复杂. 鉴于湍流通常会填充到整个空间, 局部的网格尺度缩小常常不是一个有效的解决方法. 另一方面, 在一些几何位型下, 湍流会被约束在空间中的特定区域. 那么用户只需要在外部空间用更少的格点就可以让模拟效果变好.

In order to be cache-efficient, we solve the equations along pencils in the x direction. 为了使得缓存使用便捷高效, Pencil Code 在 x 方向上沿着 pencils 求解方程. One very convenient side-effect is that auxiliary and derived variables use very little memory, as they are only ever defined on one pencil. 这样做的便利是附属变量和衍生变量仅仅占用非常少的内存, 因为他们仅仅定义在一个 pencil 上. The domain can be tiled in the y and z directions. 定义域可以在 y 和 z 方向上堆积起来. On multiprocessor computers, the code can use MPI (Message Passing

Interface) calls to communicate between processors. 在多核计算机上, Pencil Code 可以用 MPI 来实现处理器之间的交互. An easy switching mechanism allows the user to run the code on a machine without MPI libraries (e.g. a notebook computer). 程序中用鬼区来在物理边界和处理器边界上实现边界条件.

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.

输入参数 (在 'start.in', 'run.in' 中配置好的) 修改后无需再编译. 甚至用户还可以在数值模拟过程中修改输出的诊断变量种类, 还有相关程序可以实现运行中重新载入参数或者热退出 (exit gracefully). 你可以通过命令 pc_configtest 来检验这些文件的正确性.

使用 Pencil-MPI 代码的要求不高: 你可以在任何一个带了 F95 和 C 编译器的 Linux 或者 Unix 系统上运行. 比如说 GNU gcc 和 gfortran 编译器就可以. 另外还需要 shell CSH 和 Perl 解释器.

尽管 Pencil Code 主要设计在超级计算机上运行, 但也有超过 50% 的用户直接在 Macs 上运行代码, 另一半用户在他们的 Linux 笔记本电脑系统上运行或者在 Windows 上安装 Ubuntu Linux 虚拟机. 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. 如果你也有 IDL, 你就能够便捷地可视化运算结果 (我们有大量的样例), 但其他的工具比如说也能用, 相关的工具和程序和放在一起.

如果你想要充分地, 富有开创性地利用这个程序包, 这份文档的内容还不充分. 它主要的目标是告诉你程序的组织架构, 让你能够有效率地 *阅读源码*, 源码中包含了更多详尽的注释. 你可能会需要看一遍样例文件夹里面的文件. 选择一个最贴近你应用的样例并着手修改. 如果你希望给 Pencil Code 一些新功能, 你可以把代码中已经实现的相关变量, 函数, 诊断, 方程等拿来当例子. 记住一点: 当你想要理解一些特定的变量或者是函数时, grep 能给你很大的帮助.

我们将会非常开心地接纳来自用户的代码改进, 欢迎任何形式的反馈.

wdobler@gmail.com Potsdam
AxelBrandenburg@gmail.com Stockholm

致谢 Acknowledgments

已经有许许多多的人对 Pencil Code 做出了贡献. 我们在此首先感谢 Åke Nordlund (Copenhagen Observatory) 和 Bob Stein (University of Michigan), 他们首先提出了将高阶差分方法用在可压缩流体里, 并且他们还教导了其他人许多关于模拟的知识.

功率谱, 结构函数, 再生成网格, 改变处理器数量的程序以及剪切片近似和辐射传输的极限扩散近似的工作都是 Nils Erland L. Haugen (University of Trondheim) 完成的. Tobi Heinemann added the long characteristics method for radiative transfer as well as hydrogen ionization. Tobi Heinemann 添加了针对辐射传输的长距离特性方法 (翻译存疑) 以及氢元素电离方法. 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 Dhrubaditya (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 实现了 HDF5 的支持并且为高性能运算多样的数据传输过程撰写了输入输出模块. 他还贡献了日冕模块并且做了 IDL GUI 方面的工作, 包括让 IDL 读入并处理大规模数据的程序. 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).

目录

第一部分　使用代码 Using the Pencil Code

1　系统要求 System requirements

为了能够运行这个程序包, 你需要有一些下面列出的东西:

1. 必需:
   - F95 编译器
   - C 编译器
2. 经常需要用到的 (如果你没有其中之一, 你需要手动修改许多细节):
   - a Unix/Linux-type system with make and csh
   - Perl (remember: if it doesn't run Perl, it's not a computer)
3. 接下来的是一些可有可无的东西, 但或多或少会增强代码功能:
   - 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 **获得源码** Obtaining the code

The code is now distributed via 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/. 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

   ```
   svn checkout https://github.com/pencil-code/pencil-code/trunk/ ...\\
   pencil-code --username MY_GITHUB_USERNAME
   ```

   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

不管你第一次安装程序包通过什么方式 (from tarball or via svn/git), 你都可以在之后通过 svn/git 更新你的程序版本, 如果你对你的程序做了实质上的改变, 那就要小心这个升级过程产生矛盾. 尽管 svn/git , 是一个极好的分布式编程工具. 但因为我们中的每一个都在发展代码的过程中接触大量的源码, 所以依然会时不时的会出现毛病. 尽管这份源码是通过 svn/git 来发展, 你也应该在每次更新之前备份好你说做出的重要的改变.

以下是通过 git 的升级过程:

1. Perform a git update of the tree:

   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:

   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 https://github.com/pencil-code/pencil-code) who can give you push or check-in permission. 如果你已经做出了一些有意义的修改, 那就可以联系贡献者列表中的任何一人了, 他们在这个网址下列出来了, 他们可以给你 push 或者 check-in 的权限. Be sure to have sufficient comments in the code and please follow our standard coding conventions explained in Section 9.1. 请务必在代码中写上足够多的注释, 并遵循我们标注的代码风格. 我们的代码风格请参见. There is also a script to check and fix the most common stylebreaks, pc_codingstyle.

2.3 **取得最新的有效版本** Getting the last validated version

The script 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:

```
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.

```
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 pc_svnup -val option allows all other people to stay away from any trouble.

2.4 **取得更老版本的代码** Getting older versions

You may find that the latest 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 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 **开始吧**! Getting started

如果你要开始的话, 你需要先拿 'samples/' 内的样例下手. 注意你只有将数据用 IDL, DX 或其他工具 (see Sect. 5.15) 可视化, 你才能得到一个对数值计算结果充分展现的效果.

3.1 **安装** Setup

3.1.1 环境设置 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
```

对于 tcsh 或 csh 使用者而言; 对 bash, Bourne shell, or similar shells 的使用者而言, 需要以下语句:

```
sh> cd pencil-code
sh> . ./sourceme.sh
```

你应该得到与下面类似的输出:

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

除了 PATH 变量以外, 环境变量 IDL_PATH 也被设置为类似这样的形式: ./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

屋里的环境已经正确的配置好, 那你现在可以你希望

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.

具体来说, 假设你希望使用样例程序 'samples/conv-slab' 来作为你的 run directory(即运行三层的层状结构来模拟太阳对流).

我们只需要做下面的事情:

```
unix> cd samples/conv-slab
unix> pc_setupsrc
src already exists
2 files already exist in src
```

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 调整 'Makefile.src' Adapting 'Makefile.src'

这一步骤需要一些你的输入, 但它仅仅需要在电脑上看见一次就足够了. 请参看 Sect. 5.2 这里所需要进行的步骤的描述,

Note: 如果你幸运地和我们用的类似的编译器, 那么这一步你很有可能可以跳过去. 但如果不能编译的话, 就需要你自己检查一下了. 如果用的不是类似的编译器的话, 你可以运行 make 并指定一些标记, 这部分请参看 Sect. 5.2 , 具体的标记会在出现在表格 Table 2 内.

3.1.4 执行 make 命令 Running make

下一步, 在你的运行目录的子目录 'src' 里面运行 make. 因为你正在用一个之前定义好的问题来做测试, 在 'src/Makefile.local' 里面的设置和 'src/cparam.local' 都是有效的. 那么你现在只需要运行下面的代码.

```
unix> make
```

如果已经正确的配置好了这个编译器的标记 (flags). 那边编译将会成功完成.

3.1.5 选定存储数据的文件夹 Choosing a data directory

默认情况下程序会将数据结果切片写入到工作目录下的子目录 'data'. 因为这涉及到大量的输入输出操作 (至少对于大尺度网格是这样的), 一般我们会建议接近尽量避免用 NFS 手段写入数据. 推荐的配置数据文件夹 'data' 的方法是生成一个磁盘中对应的目录, 将它软链接到你的工作目录下的 './data' 文件夹. 即使这个链接是 NFS 目录的一部分, 所有的输入输出操作也会是局部的.

例如, 如果你有一个本地磁盘 '/scratch' 你可以运行下面的代码:

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

这是由 pc_mkdatadir 命令自动完成. 而 pc_mkdatadir 又会在比如说新建一个工作文件夹 pc_newrun 的时候被调用.

即使你不想用 NFS 型的文件夹 (比如在你的笔记本电脑上), 我们仍然推荐你将代码和数据通过一个类似上述的模式来分.

符号链接的一个替代选项是设置一个文件 'datadir.in' 在工作目录的根目录下, 这个文件应该包含一行文本说明绝对或相对的数据目录存放的地址. 如果使用者希望将它的工作目录在不同的数据目录下切换的时候, 这将非常有用. 在这样的情况下符号链接将会在不同的工作目录下重复生成, 所以 'datadir.in' 应该只包含一个简短的相对路径.

3.1.6 运行代码 Running the code

你现在可以执行代码了:

```
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
 nxgrid,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. 1, 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 $= \sqrt{\langle \boldsymbol{u}^2 \rangle}$;
5. the maximum velocity, umax $= \max |\boldsymbol{u}|$;
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 $= \delta t / (c_{\rm s0}\, \delta x_{\min})$;
9. the time step in units of the advective time step, dtu $= \delta t / (c_{\delta t}\, \delta x / \max |{\rm u}|)$;
10. the time step in units of viscous time step, dtnu $= \delta t / (c_{\delta t,{\rm v}}\, \delta x^2 / \nu_{\max})$;
11. the time step in units of the conductive time step, dtchi $= \delta t / (c_{\delta t,{\rm v}}\, \delta x^2 / \chi_{\max})$.

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

If you have IDL, you can visualize the stratification with (see Sect. 5.15.4 for details)

```
unix > idl
```

图 1: Stratification of the three-layer convection model in 'samples/conv-slab' after 50 timesteps ($t = 0.428$). Shown are (from left to right) density ρ, 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).

```
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 outdates specific routines that produce results like that shown in Fig. 1.

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 **更多的测试** Further tests

在 ‘samples/’ 文件夹内有大量的其他测试. 你可以用脚本 ‘bin/auto-test’ 来自动运行这些测试程序并且将输出与参考结果作对比.

4 代码架构

4.1 目录分支树

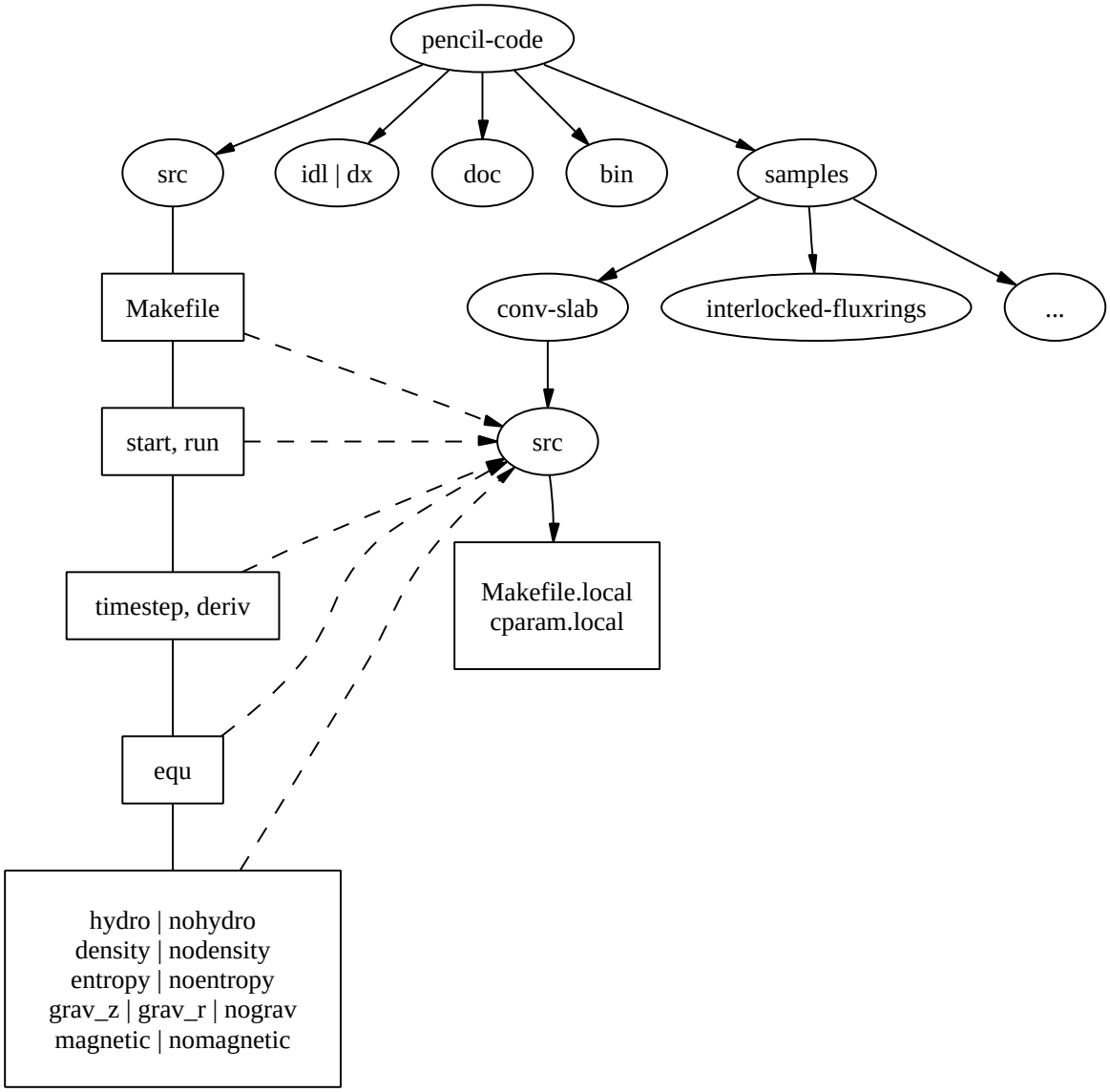


图 2: The basic structure of the code

所有代码的目录结构已经展示在 Fig. 2 内. 在 'pencil-code' 目录下, 现在有以下文件和文件夹:

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

几乎所有的代码都已经包含在文件夹 'src/' 内. 但为了对单独的程序设置依然能方便地处理, 运行的程序代码会复制到运行文件夹下的 'src'内, 如 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 'pencil-code/'. For example, we have our own runs in a repository called 'pencil-runs', so we do

```
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/
buoy_tube/  forced/        pass_only/
convstar/   interstellar/  radiation/
```

目录 'forced/' 包含了一些受迫湍流的运行代码 (有磁化的也有未经磁化的); 'gravz/' 包含了垂直重力场下的运行代码; 'rings/' 包含一些衰减磁流体问题 (如互锁的流环作为初始条件 interlocked flux rings); 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.[1]

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

[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.

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 基本概念

4.2.1 Pencils 中的数据访问 Data access in pencils

不像 80 年代和 90 年代早期作为超级计算领域主角的 CRAY 计算机, 所有的现代计算器都有缓存, 而这原是引起许多程序运行的瓶颈.

这意味着在有缓存的条件下, 如果大型三维数组每一步都会进行计算, 那会带来巨大的处理超长数组和矢量运算的优势. 这一设计也使程序设计更为简化, 扩展了符合 F90 数组设计的应用. 然而, 一个更有效利用缓存的方法是一次沿着一个维度 pencil 的数据计算完一个时步 (或者一个多时步计划中的一小步). 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 模块化

每一个 run 文件夹有一个文件 Each run directory has a file 'src/Makefile.local' in which you choose certain modules[2], which tell the code whether or not entropy, magnetic fields, hydrodynamics, forcing, etc. should be invoked. 例如, 给一个受外力作用的湍流 MHD 做模拟的设置如下

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

MPICOMM   = nompicomm
GLOBAL    = noglobal
```

[2]We stress once more that we are not talking about F90 modules here, although there is some connection, 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.

```
IO      =  io_dist
FOURIER = nofourier
```

This file will be processed byand 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. J in the Appendix lists all currently available modules.

Note that most of these make variables must be set, but they will normally obtain reasonable default values in 'Makefile' (so you only need to set the non-standard ones in 'Makefile.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.

这个文件将由 make 处理, 因此这些设置是 make 所需要的参数.

除了物理模块之外 (以上样例子中, 空行上方的为物理模块), 一些技术模块也可以被选择为使用或停用；在上面的例子中, 这些是 MPI(关闭), 附加的全局变量 (没有), 输入/输出 (分布式) 和 FFT(未使用). 教派的桌子.??? 在附录中

列出所有当前可用的模块.

请注意, 大多数这些 make 变量必须被设置, 但它们通常会在“MaMaFILE”中获得合理的默认值 (因此, 您只需要在“MaxField.Load”中设置非标准的值).

这是通过使用这种切换机制, 使我们获得高灵活性, 而不诉诸过多的密码预处理器指令或其他代码内的开关.

模块的许多可能组合已经被测试, 实例是分布的一部分, 但是您可能感兴趣的是以前从未尝试过并且可能还没有工作的组合, 因为模块不是完全正交的. 在这种情况下, 我们依赖于用户反馈来修复问题并记录他人的更改.

4.3 run 文件夹内的文件

4.3.1 'start.in', 'run.in', 'print.in'

这些文件指定启动和运行时的参数 (参见 Sects. 5.12 和 K.2), 并设置需要诊断的变量输出 (参见 5.5). 这些参数明确了所需模拟的配置并且放在 'samples' 文件夹内并用 svn 同步.

4.3.2 'datadir.in'

如果这个文件存在的话, 它必须包含一个现有文件夹的名字, 而该文件夹将被作为 data directory 文件夹使用, 换句话说, 所有数据输出的地方. 如果 'datadir.in' 不存在, 数据文件夹默认为 'data/'.

4.3.3 'reference.out'

如果存在的话,'reference.out'指定您应该在 run 目录中获得的输出, 只要您没有更改任何参数. 如果想检查你的运行结果是否成功, 可以比较 'time_series.dat' 和 'reference.out':

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

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

这些文件是通往 '$PENCIL_HOME/bin' 的链接. 你会常常用到 'start.csh' 和 'run.csh' 来使程序初始化. 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/ '

'src' 文件夹包含两个本地 (local, 局部？) 文件, 'src/Makefile.local' 和 'src/cparam.local', 让使用者可以方便地选择模块 (see 4.2.2) 并设置参数, 比如设置网格大小和每个方向上用到的处理器这两个文件是一个模拟的设置参数, 他们在 'samples' 文件夹内, 通过 svn 同步.

'src/cparam.inc' 是通过脚本 'mkcparam' 自动生成的, 它包含了一个明确的设置所需要的基本变量数.

所有 'src/' 内的其他文件要不就是到位于 '$PENCIL_HOME/src'内的源文件的链接 (and 'Makefile.src') , 要不就是编译器产生的对象或模块文件.

4.3.6 'data/ '

这个文件夹存储程序的运行结果:(但如果 'datadir.in'存在的话, 该文件夹名会被 'datadir.in'第一行指定, 参见 §4.3.2)

'data/dim.dat' 全局数组的维度 The global array dimensions.

'data/legend.dat' 首行指定了输出在 'time_series.dat'中的诊断变量名字.

'data/time_series.dat' 诊断变量的时序表 (also printed to stdout). 你可以直接使用这个文件绘图, 通过 Gnuplot, IDL, Xmgrace 或其他类似工具 (see also §5.15).

'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 'RELOAD' 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;

'VARN' 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_mpiodist.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 'proc$N$/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

bx.xy bx.xz by.xy by.xz bz.xy bz.xz divu.xy lnrho.xz
ss.xz ux.xy ux.xz uz.xy uz.xz

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

4.4 **开始并运行参数** Start and run parameters

所有的 'start.in' 和 'run.in'里面的输入参数都集合在 Fortran namelists内. 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
  cvsid='$Id:$',                ! 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 [?], rather than the compiler's built-in generator. The adiabatic index γ 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}.

'samples/helical-MHDturb' 文件内的运行参数如下

```
&run_pars
```

```
  cvsid='$Id:$',                ! 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
/
&density_run_pars
/
&forcing_run_pars
  iforce='helical', force=0.07, relhel=1.
/
&magnetic_run_pars
  eta=5e-3
/
&viscosity_run_pars
  nu=5e-3
/
```

这里我们运行 nt= 10 个时间步长, 每隔一步输出一次诊断; 我们要求时间步长能使得对流 Courant 数 ≤ 0.4 和扩散 Courant 数 ≤ 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). 我们会写入 'VARN' 一些需要记录的参数, 每隔 dsnap= 50 个时间单位和可供视频化的二维数据记录每 dvid= 0.5 个时间单位. 并且, 我们使用具有确定性的随机数生成器 (deterministic random number generator). 粘度 ν 和磁场扩散系数 η 设为 5×10^{-3} (所以网孔雷诺数 (mesh Reynolds number) 将近为 $u_{\rm rms}\delta x/\nu = 0.3 \times (2\pi/32)/5 \times 10^{-3} \approx 12$, 这个数值有些偏高). forcing_run_pars 里的参数指明了螺旋的力的幅度.

完整的输入参数在附录 K 内.

4.5 物理单位

许多计算并不依赖于单位, 也就是说所有结果都依赖于你用于解释计算结果的单位制. 例如, 如果你在一个长度 $L = 1$ 箱子里模拟一束简单的流体并且在 $t = 3$ 个时间单位后得到了它的最大速度 $u_{\rm max} = 0.5$. 这可以解释为 $L = 1\,{\rm m}$, $u_{\rm max} = 0.5\,{\rm m/s}$, $t = 3\,{\rm s}$, 也可以是 $L = 1\,{\rm pc}$, $u_{\rm max} = 0.5\,{\rm pc/Myr}$, $t = 3\,{\rm Myr}$, , 取决于你用的单位制. 你所使用的单位需要保持一致, 所以在上面的第二个例子中, 扩散系数的单位应该是 ${\rm pc}^2/{\rm Myr}$.

磁感应强度和温度的单位会通过在内部设置 $\mu_0 = 1$, $c_p = 1$ 来得到[3]. 这意味着如果你的密度和速度单位分别是 $[\rho]$ 和 $[v]$, 则磁场单位则为

$$[B] = \sqrt{\mu_0 [\rho] [v]^2} , \tag{1}$$

而温度单位则是

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

表 1: 通过 Eqs. (1) and (2) 和确定的 $[\rho]$, $[v]$ 得到的磁感应强度和温度单位. Values are for a monatomic gas ($\gamma = 5/3$) of mean atomic weight $\bar{\mu}_g = \bar{\mu}/1\,\mathrm{g}$ in grams.

$[\rho]$	$[v]$	$[B]$	$[T]$
$1\,\mathrm{kg/m^3}$	$1\,\mathrm{m/s}$	$1.12\,\mathrm{mT} = 11.2\,\mathrm{G}$	$\left(\frac{\bar{\mu}_g}{0.6}\right) \times 2.89 \times 10^{-5}\,\mathrm{K}$
$1\,\mathrm{g/cm^3}$	$1\,\mathrm{cm/s}$	$3.54 \times 10^{-4}\,\mathrm{T} = 3.54\,\mathrm{G}$	$\left(\frac{\bar{\mu}_g}{0.6}\right) \times 2.89\,\mathrm{nK}$
$1\,\mathrm{g/cm^3}$	$1\,\mathrm{km/s}$	$35.4\,\mathrm{T} = 354\,\mathrm{kG}$	$\left(\frac{\bar{\mu}_g}{0.6}\right) \times 28.9\,\mathrm{K}$
$1\,\mathrm{g/cm^3}$	$10\,\mathrm{km/s}$	$354\,\mathrm{T} = 3.54\,\mathrm{MG}$	$\left(\frac{\bar{\mu}_g}{0.6}\right) \times 2\,890\,\mathrm{K}$

对一些密度和速度单位, 表 1 已经展示了最后得到的磁感应强度和温度单位.

另一方面, 一旦使用了材料方程, 例如: 辐射损耗的常用参数,Kramers 不透明度, Spitzer 导电性或电离度 (这表征良定义的电离能), 在代码中对应的例程需要知道你正在使用的单位. 这些信息在 'start.in' 或 'run.in' 里面可以通过 unit_system, unit_length, unit_velocity, unit_density 和 unit_temperature[4] 的参数指定. 例如:

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

Note: 默认的单位制是 unit_system='cgs', 它也可以写为 unit_system='Babylonian cubits'.

4.6 粘度存在最小允许值

我们强调, 默认条件下, 这个程序都会将扩散系数视为常数 (粘度 ν, 热扩散系数 χ, 磁扩散系数 η 及被动标量扩散系数 passive scalar diffusivity $\mathcal{D}$). 如果这些量中某些量过小, 那么你需要更多的格点以使之得到合理的数值计算结果, 否则最后程序运算结果就会越来越不稳定. 基于最大速度的网格雷诺数可以给出一个有用的判据,

$$\mathrm{Re}_{\mathrm{mesh}} = \max(|\boldsymbol{u}|) \max(\delta x, \delta y, \delta z)/\nu, \tag{3}$$

[3] Note that $c_p = 1$ is only assumed if you use the module noionization.f90. If you work with ionization.f90, temperature units are specified by unit_temperature as described below.

[4] 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. (2) above.

根据具体的问题, 网格雷诺数不能超过某个具体的值. $\mathrm{Re}_{\mathrm{mesh}}$ 网格雷诺数最大的值常常在 5 左右. 类似的 mesh Péclet 和 mesh magnetic Reynolds numbers 也不能太大.

Note that in some cases, 'wiggles' in $\ln\rho$ will develop despite sufficiently large diffusion coefficients, 本质上是因为连续性方程不包含耗散项. 我们对于包括但不限于对流的情形可以通过 upwinding 来抑制这种现象, 参见 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, 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.

4.7 **时间步长** The time step

4.7.1 常规 RK-2N 时间步长 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. RK-2N 指的是三阶龙格-库塔算法 (Williamson 1980), 它有着两个代码块 (翻译存疑) 的空间复杂度, 所以名字里有 2N. Therefore the 2N in the name.

The time step is normally specified as Courant time step through the coefficients cdt ($c_{\delta t}$), cdtv ($c_{\delta t,\mathrm{v}}$) and cdts ($c_{\delta t,\mathrm{s}}$). 产生的 Courant 时间步长依据下式给出

$$\delta t=\min\left(c_{\delta t}\frac{\delta x_{\min}}{U_{\max}},c_{\delta t,\mathrm{v}}\frac{\delta x_{\min}^2}{D_{\max}},c_{\delta t,\mathrm{s}}\frac{1}{H_{\max}}\right)\ , \tag{4}$$

where

$$\delta x_{\min}\equiv\min(\delta x,\delta y,\delta z)\ ; \tag{5}$$

$$U_{\max}\equiv\max\left(|\boldsymbol{u}|+\sqrt{c_{\mathrm{s}}^2+v_{\mathrm{A}}^2}\right)\ , \tag{6}$$

c_{s} 和 v_{A} 分别表示声速 Alfvén 速度;

$$D_{\max}=\max(\nu,\gamma\chi,\eta,D), \tag{7}$$

公式中 ν 表示动粘度 (翻译存疑) denotes kinematic viscosity, $\chi=K/(c_p\rho)$ 表示热扩散率而 η 表示磁扩散率; and

$$H_{\max}=\max\left(\frac{2\nu\mathrm{S}^2+\zeta_{\mathrm{shock}}(\boldsymbol{\nabla}\cdot\boldsymbol{u})^2+...}{c_vT}\right), \tag{8}$$

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

To fix the time step δt to a value independent of velocities and diffusivities, explicitly set the run parameter dt, rather than cdt or cdtv (see p. ??).

如果时间步长超过了粘滞的时间步长, 那么模拟结果将不一定可靠. 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. 下面有个图片展示的例子 Fig. 3.

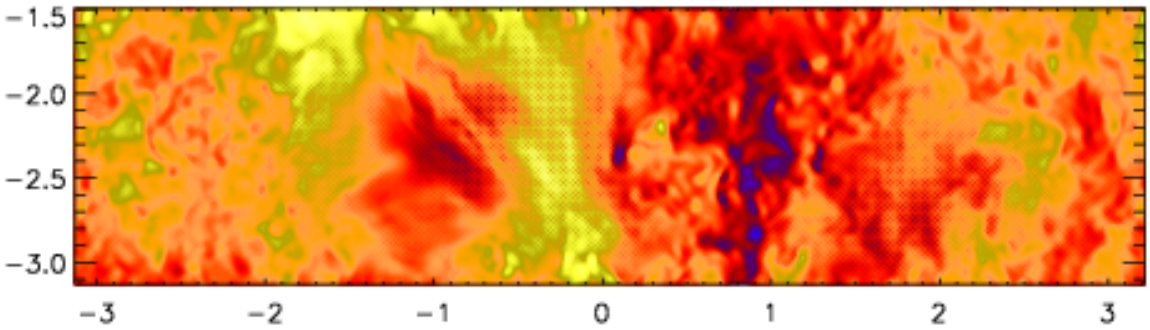


图 3: 时间步长过长的速度数据切片例子. Note the spurious checkerboard modulation in places, for example near $x = -0.5$ and $-2.5 < y < -1.5$. This is an example of a hyperviscous turbulence simulations with 512^3 meshpoints and a third order hyperviscosity of $\nu_3 = 5 \times 10^{-12}$. 流体粘度的解释参见 Appendix E.

通过 Runge-Kutta 2N scheme 时步法得以实现. 细节参见 Sect. H.4.

4.7.2 The Runge-Kutta-Fehlberg time step

五阶的 Runge-Kutta-Fehlberg 时步法是可以使用的. 它通常用来解决化学问题, 常常需要双精度选项. 为了使之运行, 你需要调整 'src/Makefile.local' 里的

```
TIMESTEP = timestep_rkf
```

另外还需要在 'run.in' 里面设置 itorder=5 一个使用它的样例是 'samples/1d-tests/H2_flame-speed'. 但这种方法还在测试中.

4.8 **边界条件**

4.8.1 指定边界条件的文件

自然地, 在大部分 Pencil Code 的测试中, 边界条件常常在 'run.in' 内设定好. 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 (默认的边界条件) in all three directions. 这些 ghost data 在计算中从不会被调用, 而 'run.x' 会在调用 ghost-zone 的数据之前调用边界条件.

为了避免你所设置的初始条件出现在你的输出文档内, 你可以将它们设置在文件 'start.in' 内. 这样, 它们会被 'run.x' 所运行, 除非你又在 'run.in'内进行设置, 否则不会出现问题.

4.8.2 如何指定边值条件

对 MPI 程序来说, 用 ghost zones 来代表相邻单元处理器所模拟区域的数值是很自然的, 于是边界条件可以通过指定每个边界上的三层 ghost points 来实现 (three layers of ghost points on either boundary). 合适的边界条件选择是在 'run.in'里设置 bc{x,y,z} ; 术语使用说明如下. 给 bc{x,y,z} 设置一串字符来设置周期性的边界条件

bcx = 'p','p','p', 'p', 'p'

或者

bcz = 's','s','a','a2','c1:c2'

设置了非周期性的边界条件. 每一个字母对应着 u_x, u_y, u_z, $\ln\rho$, s/c_p, A_x, A_y, A_z, $\ln c$ 里其中之一的变量边界条件 that are actually used in this order. 以下字符是可用的:

'p' 周期边界条件 periodic boundary condition

'a' 关于边界反对称, 即边界数值为零 antisymmetric condition w. r. t. the boundary, i. e. vanishing value

's' 关于边界对称, 即一阶导为零 symmetric condition w. r. t. the boundary, i. e. vanishing first derivative

'a2' 关于边界上任意值的反对称, 即二阶导为零 antisymmetry w. r. t. the arbitrary value on the boundary, i. e. vanishing second derivative

'c1' $\ln\rho$ 和 s 的特殊边界条件: 边界的稳恒热流 special boundary condition for $\ln\rho$ and s: constant heat flux through the boundary

'c2' s 的特别边界条件: 边界上稳定的温度, 该条件需要 $\ln\rho$ 的 a2 边界条件 special boundary condition for s: constant temperature at the boundary — requires boundary condition a2 for $\ln\rho$

'cT' s 或 $\ln\ T$ 的特殊边界条件: 边界常温 (对任意设置的 $\ln\rho$ 成立) special boundary condition for s or $\ln\ T$: constant temperature at the boundary (for arbitrarily set $\ln\rho$)

'ce' s 的特殊边界条件: 在边界的 ghost points 上设置温度 (对任意设置的 $\ln\rho$ 成立) 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' 剪切片边界条件 (这在 Shear模块被使用时是默认的) 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' 对 $\ln \rho$ 和 s 的强制要求使之在垂直边界上满足流体静力学上的平衡 special boundary condition for $\ln \rho$ and s which enforces hydrostatic equilibrium on vertical boundaries

a:b (e. g. 'c1:c2') 这种形式的边界条件意味着: 在左/低边界上使用 a 条件, 在右/上边界使用 b 条件.

如果你从一个老的 'run.in' 文件出发, 修改并另存为了一个新文件, 变量的数目改变了 (比如计算熵和不计熵的时候, (noentropy vs. entropy), 你需要记得调整数组 bcx,bcy,bcz的长度. 这样做的好处是通过在一个方向上设置新的边界条件, 很容易就可以交换所有的边界条件 (例如, 让所有边界都周期化, 或者不要剪切片边界条件切到无压力条件)

5 **使用程序** Using the code

5.1 **配置代码以使之编译并运行**,Configuring the code to compile and run on your computer

Note: 我们建议不要参考 Sects. 5.2 和 4.3.4 中比较老的文档内容, 本章节的内容更合适.

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
```

这行命令会告诉你正在使用的系统的 host-ID：

```
unix> pc_build -i
```

5.1.2 配置文件的结构 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.[5]

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

%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.

[5]However, if the include statement is inside a section, only the file's contents inside that section are inserted.

Thus,

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

is the same as

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

5.1.3 编译源码 Compiling the code

用 pc_build 命令来编译代码:

```
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 跑代码 Running the code

用 pc_run 命令来跑代码:

```
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:

```
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.

The pencil-test script will use pc_auto-test if given the ‘–use-pc_auto-test’ or ‘-b’ option:

```
unix> pencil-test --use-pc_auto-test
unix> pencil-test -b                  # ditto
unix> pencil-test -b                -Wa,--max-level=1,--no-pencil-check  # quick pencil test
```

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

5.2 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:

```
### Begin machine dependent

## IRIX64:
[...]  (leave as it is in the original Makefile)
## OSF1:
[...]  (leave as it is in the original Makefile)

## Linux:
[...]  (leave everything from original Makefile and add:)
```

```
#FC=f95
#FFLAGS=-O2 -u
#FC=f90             #(Janus)
#FFLAGS=-O3         #(Janus)
#LDMPI=-lmpi -llam  #(Janus)

## SunOS:
[...]  (leave as it is in the original Makefile)
## UNICOS/mk:
[...]  (leave as it is in the original Makefile)
## HI-UX/MPP:
[...]  (leave as it is in the original Makefile)
## AIX:
[...]  (leave as it is in the original Makefile)

### 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 2 summarizes flags we use for common compilers.

5.3 **改变分辨率** 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 ..
```

表 2: 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 LDMPI.

Compiler	FC	FFLAGS	CC	CFLAGS
Unix/POSIX:				
GNU	gfortran	-O3	gcc	-O3 -DFUNDERSC=1
Intel	ifort	-O2	icc	-O3 -DFUNDERSC=1
PGI	pgf95	-O3	pgcc	-O3 -DFUNDERSC=1
G95	g95	-O3 -fno-second-underscore	gcc	-O3 -DFUNDERSC=1
Absoft	f95	-O3 -N15	gcc	-O3 -DFUNDERSC=1
IBM XL	xlf95	-qsuffix=f=f90 -O3	xlc	-O3 -DFUNDERSC=1
outdated:				
IRIX Mips	f90	-64 -O3 -mips4	cc	-O3 -64 -DFUNDERSC=1
Compaq	f90	-fast -O3	cc	-O3 -DFUNDERSC=1

```
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 it that with static allocation the compiler can check whether we are out of bounds.

5.4 **使用非等距网格** 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 ζ_i 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

$$\frac{df}{dz} = \frac{df}{d\zeta}\frac{d\zeta}{dz} = \frac{1}{z'} f'(\zeta) \;, \qquad \frac{d^2 f}{dz^2} = \frac{1}{z'^2} f''(\zeta) - \frac{z''}{z'^3} f'(\zeta) \;, \tag{9}$$

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

$$\frac{df}{dz} = \zeta'(z)\, f'(\zeta) \;, \qquad \frac{d^2 f}{dz^2} = \zeta'^2(z)\, f''(\zeta) + \zeta''(z)\zeta'(z) f'(\zeta) \;. \tag{10}$$

Internally, the code uses the quantities dz_1 $\equiv 1/z' = \zeta'(z)$ and dz_tilde $\equiv -z''/z'^2 = \zeta''/\zeta'$, and stores them in 'data/proc$N$/grid.dat'.

The parameters lequidist (a 3-element logical array), grid_func, 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 grid_function='sinh', grid_function='linear' (which produces an equidistant grid for testing purposes), and grid_function='step-linear'.

The sinh profile: For grid_function='sinh', the function $z(\zeta)$ is given by

$$z(\zeta) = z_0 + L_z \frac{\sinh a(\zeta - \zeta_*) + \sinh a(\zeta_* - \zeta_1)}{\sinh a(\zeta_2 - \zeta_*) + \sinh a(\zeta_* - \zeta_1)} \,, \tag{11}$$

where z_0 and $z_0 + L_z$ are the lowest and uppermost levels, ζ_1, ζ_2 are the ζ 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 ζ_* is the ζ value of the inflection point of the sinh function. The z coordinate and ζ value of the inflection point are related via

$$z_* = z_0 + L_z \frac{\sinh a(\zeta_* - \zeta_1)}{\sinh a(\zeta_2 - \zeta_*) + \sinh a(\zeta_* - \zeta_1)} \,, \tag{12}$$

which can be inverted ("after some algebra") to

$$\zeta_* = \frac{\zeta_1 + \zeta_2}{2} + \frac{1}{a} \operatorname{artanh} \left[\left(2 \frac{z_* - z_0}{L_z} - 1 \right) \tanh a \frac{\zeta_2 - \zeta_1}{2} \right] . \tag{13}$$

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

$$z(\zeta) = z_0 + L_z \frac{\psi[a(\zeta - \zeta_*)] + \psi[a(\zeta_* - \zeta_1)]}{\psi[a(\zeta_2 - \zeta_*)] + \psi[a(\zeta_* - \zeta_1)]} \,, \tag{14}$$

and the reference point ζ_* is found by inverting

$$z_* = z_0 + L_z \frac{\psi[a(\zeta_* - \zeta_1)]}{\psi[a(\zeta_2 - \zeta_*)] + \psi[a(\zeta_* - \zeta_1)]} \,, \tag{15}$$

numerically.

管流 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 [?] and use a Chebyshev-type grid with a cosine distribution of the grid points such that in the y direction they are located at

$$y_j = h \cos \theta_j \,, \tag{16}$$

where

$$\theta_j = \frac{(N_y - j)\,\pi}{N_y - 1}, \quad j = 1, 2, \ldots, N_y \tag{17}$$

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

$$l_w = \frac{\nu}{u_\tau}\,, \qquad u_\tau = \sqrt{\frac{\tau_w}{\rho}}\,, \tag{18}$$

and τ_w is the shear wall stress. This is accomplished by using at least

$$N_y \;\geq\; N_y^* \;=\; \frac{\pi}{\arccos\!\left(1 - \dfrac{\delta}{h}\right)} + 1 \tag{19}$$

$$\;=\; \pi\sqrt{\frac{h}{2\,\delta}} + 1 - \frac{\pi}{24}\sqrt{\frac{2\,\delta}{h}} + O\left[\left(\frac{\delta}{h}\right)^{3/2}\right] \tag{20}$$

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

$$N_y \geq \left\lceil \pi\sqrt{\frac{h}{2\,\delta}} \right\rceil + 1 \tag{21}$$

(where $\lceil x \rceil$ 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'):

```
&init_pars
  [...]
  xyz0  = -2., -2., -2.       ! first corner of box
  Lxyz  =  4.,  4.,  4.       ! box size
  lperi =  F ,  F ,  F        ! periodic direction?
  lequidist = F, F, T         ! non-equidistant grid in z
  xyz_star  = , , -2.         ! position of inflection point
  grid_func = , , 'sinh'      ! sinh function: linear for small, but
                              ! exponential for large z
  coeff_grid = , , 0.5
/
```

The parameter array coeff_grid represents z_* and a; the bottom height z_0 and the total box height L_z are taken from xyz0 and 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 xyz_star(3) (i. e. z_*) to -2.

5.5 **诊断输出** Diagnostic output

Every it1 time steps (it1 is a runtime parameter, see Sect. K.2), the code writes monitoring output to 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:

```
t(F10.3)
urms(E13.4)
rhom(F10.5)
oum
```

which means that the output table will contain time t in the first column formatted as (F10.3), followed by the mean squared velocity, urms (i.e. $\langle \boldsymbol{u}^2 \rangle^{1/2}$) in the second column with format (E13.4), the average density rhom ($\langle \rho \rangle$, which allows to monitor mass conservation) formatted (F10.5) and the kinetic helicity oum (that is $\langle \boldsymbol{\omega} \cdot \boldsymbol{u} \rangle$) in the last column with the default format (E10.2).[6] The corresponding diagnostic output will look like

```
----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
```

5.6 **数据文件** 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/proc$N$/'. The parameter isav determines the frequency at which the current snapshot 'data/proc$N$/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

[6] 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.

makes it difficult to follow the evolution interactively during test runs. There is also the ialive parameter. Setting this to 1 or 10 gives an updated 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[7]:

1. variable vector f [mx×my×mz×nvar]
2. time t [1], coordinate vectors x [mx], y [my], z [mz], grid spacings δx [1], δy [1], δz [1], shearing-box shift Δy [1]

All numbers (apart from the record markers) are single precision (4-byte) floating point numbers, unless you use double precision (see §5.17, 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:

```
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 **视频和截断图片** 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)

[7] 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 $\langle N_{\rm bytes}, \text{raw\_data}, N_{\rm bytes}\rangle$

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 § K.4.

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

```
!lwrite_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.

另一个例子 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:

```
&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:

```
&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:

   bb

2. Edit the namelist run_pars in the file 'run.in' to include zbot_slice and ztop_slice.

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): bb3

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

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

   etc.

文件结构 File structure Slice files consist of one Fortran record (see footnote on page 37) 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_1 [nx×ny×nz], time t_1 [1], position_1 [1]

2. data_2 [nx×ny×nz], time t_2 [1], position_2 [1]

3. data_3 [nx×ny×nz], time t_3 [1], position_3 [1]

etc.

5.8 **平均** 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 it1th 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 it1dth 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 'yaver.in', 'zaver.in', and 'phiaver.in' where the first letter indicates the averaging direction. The output is then stored to the files 'data/yaverages.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. K.5. In addition, one needs to set lwrite_phiaverages, lwrite_yaverages, 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 $\boldsymbol{B}^2$.

Azimuthal averages are written every d2davg time units to the files 'data/averages/PHIAVGN'. The file format of azimuthal-average files consists of the following Fortran records:

1. number of radial points $N_{r,\phi-\mathrm{avg}}$ [1], number of vertical points $N_{z,\phi-\mathrm{avg}}$ [1], number of variables $N_{\mathrm{var},\phi-\mathrm{avg}}$[1], number of processors in z direction [1]
2. time t [1], positions of cylindrical radius r_{cyl} [$N_{r,\phi-\mathrm{avg}}$] and z [$N_{z,\phi-\mathrm{avg}}$] for the grid, radial spacing δr_{cyl} [1], vertical spacing δz [1]
3. averaged data [$N_{r,\phi-\mathrm{avg}}\times N_{z,\phi-\mathrm{avg}}$]
4. label length [1], labels of averaged variables [$N_{\mathrm{var},\phi-\mathrm{avg}}$]

All numbers are 4-byte numbers (floating-point numbers or integers), unless you use double precision (see §5.17).

To read and visualize azimuthal averages in IDL, use '$PENCIL_HOME/idl/files/pc_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 时间平均 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 τ_{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,[8]

$$\langle f\rangle_{t+\delta t} = \langle f\rangle_t + \frac{\delta t}{\tau_{\mathrm{avg}}}[f(t+\delta t) - \langle f\rangle_t]\ , \tag{24}$$

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

$$\langle f\rangle_{t+\delta t} = \langle f\rangle_t + \frac{\delta t}{t-t_0+\delta t}[f(t+\delta t) - \langle f\rangle_t]\ , \tag{22}$$

which is equivalent to

$$\langle f\rangle_t = \frac{1}{t-t_0}\int_{t_0}^{t} f(t')\,dt'\ , \tag{23}$$

but we do not expect large differences.

which is equivalent to

$$\langle f \rangle_t = \int_{t_0}^{t} e^{-(t-t')/\tau_{\rm avg}} f(t')\, dt' \ . \tag{25}$$

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 (24) will approximate Eq. (25) only to first-order accuracy in δt. In practice, however, δt is small enough to make this accuracy suffice.

In 'src/ctimeavg.local', you need to set the number of slots used for time averages. Each of these slots uses mx × my × 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*/TAVN'. 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 **帮助脚本** 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 'VARN'. Copies a specified snapshot 'VARN' to 'var.dat' where N and (optionally) the target run directory 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 others' 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 orientation via a slow modem line.

local Materialize a symbolic link

mkcparam 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('proc$N$'/), 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, 'proc$N$/', to a new target tree creating directories where necessary.

nl2idl Transform a Fortran namelist (normally the files 'param.nml', 'param2.nml' written 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.

pc_inspectrun Check the execution of the current run: prints legend and the last few lines of the 'time_series.dat' file. It also appends this result to a file called 'SPEED', which contains also the current wall clock time, so you can work out the speed of the code (without being affected by i/o time).

read_videofiles.csh The script for running read_videofiles.x.

remote-top Create a file 'top.log' in the relevant 'proc$N$/' directory containing the output of top for the appropriate processor. Used in batch scripts for multi-processor runs.

run.csh The script for producing restart files with the initial condition; see § 4.3.4

scpdatadir Make a tarball of data directory, 'data/' and use scp to secure copy to copy it to the specified destination.

pc_setupsrc Link 'start.csh', 'run.csh' and 'getconf.csh' from '$PENCIL_HOME/bin'. Generate 'src/' if necessary and link the source code files from '$PENCIL_HOME/src' to that directory.

start.csh The script for initializing the code; see § 4.3.4

summarize-history Evaluate ‘params.log’ and print a history of changes.

timestr Generate a unique time string that can be appended to file names from shell scripts through the backtick mechanism.

pc_tsnap Extract time information from a snapshot file, ‘VAR*N*’.

There are several additional scripts on ‘pencil-code/utils’. Some are located in separate folders according to users. There could be redundancies, but it is often just as easy to write your own new script than figuring out how something else works.

5.10 RELOAD and STOP files

The code periodically (every it 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

```
unix> 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 sever. 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
  cvsid='$Id:$',                ! 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 [?], rather than the compiler's built-in generator. The adiabatic index γ 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
```

```
  cvsid='$Id:$',                ! 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
/
&density_run_pars
/
&forcing_run_pars
  iforce='helical', force=0.07, relhel=1.
/
&magnetic_run_pars
  eta=5e-3
/
&viscosity_run_pars
  nu=5e-3
/
```

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 ≤ 0.4 and the diffusive Courant number ≤ 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 'VARN' 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 ν and magnetic diffusivity η are set to 5×10^{-3} (so the mesh Reynolds number is about $u_{\rm rms}\delta x/\nu = 0.3 \times (2\pi/32)/5 \times 10^{-3} \approx 12$, which is in fact rather a bit to high). The parameters in forcing_run_pars specify fully helical forcing of a certain amplitude.

A full list of input parameters is given in Appendix K.

5.13 **重启模拟** Restarting a simulation

如果你的运行程序跑到一半中断了，你还可以重新运行，从中断的地方继续，它可以从'data/proc*/var.dat'里你保存的最新帧的数据处再开始. 最新帧的数据保存在一个单独的文件中 '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'N file instead.

如果你要到另一台计算机上接着运行了, 你只需要复制 'data/proc*/var.dat' (为了以防万一加上 'data/proc*/time.dat') 到一个新的目录下面. 你也许还需要 'data/proc*/seed.dat' 作为你的随机数生成器. 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.

5.14 **一维和二维的运行** 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. nygrid=1 or 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, nygrid=1 and 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.

5.15 **可视化** Visualization

5.15.1 Gnuplot

Simple visualization can easily be done using Gnuplot (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 $u_{\rm rms}(t)$. 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 $u_{\rm rms}(t)$ and $u_{\rm max}(t)$, use ASCII graphics:

```
gnuplot> set term dump
gnuplot> set logscale y
```

```
gnuplot> plot "data/time_series.dat" using 2:4 title "urms", \
gnuplot>      "data/time_series.dat" using 2:5 title "umax"
```

5.15.2 Data explorer

DX (data explorer; http://www.opendx.org) is an open-source tool for visualization of three-dimensional data.

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 Ctl-o or Execute → 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 Ctl-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 → 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.

5.15.3 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 expensive IDL package (see S. 5.15.4). For the way we use IDL for the Pencil Code, compatibility 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 simple 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 § 5.15.4.

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:

```
GDL> pc_read_dim, OBJECT=dim
GDL> $$PENCIL_HOME/bin/nl2idl -d ./data/param.nml> ./param.pro
GDL> pc_read_param, OBJECT=par
GDL> pc_read_grid, OBJECT=grid
GDL> pc_read_var, OBJECT=var
```

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[9]

```
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. 5.15.4.

5.15.4 IDL

IDL is a commercial visualization program for two-dimensional and simple three-dimensional 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.

There are also other small GUIs available, e.g. the file 'time-series.dat' can easily be analyzed with the command:

[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.

```
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 [10] 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
```

[10] 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/ (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.

```
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

```
IDL>  .r rall
```

instead of .r r (here, .r is a shorthand for .run). The procedure '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 'r.pro' instead.

If you need the magnetic field or the current density, you can calculate them in IDL by [11]

```
IDL>  bb=curl(aa)
IDL>  jj=curl2(aa)
```

By default one is reading always the current snapshot 'data/proc$N$/var.dat'; if you want to read one of the permanent snapshots, use (for example)

```
IDL>  varfile='VAR2'
IDL>  .r r (or .r rall)
```

See Sect. 5.6.1 for details on permanent snapshots.

With 'r.pro', you can switch the part of the domain by changing the variable datadir:

```
IDL>  datadir='data/proc3'
IDL>  .r r
```

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 .r is not allowed.

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

```
IDL>  pc_read_var_raw, obj=var, tags=tags
```

[11] Keep in mind that jj=curl(bb) would use iterated first derivatives instead of the second derivatives and thus be numerically less accurate than jj=curl2(aa), particularly at small scales.

```
IDL> bb = pc_get_quantity ('B', var, tags)
IDL> jj = pc_get_quantity ('j', var, tags)
```

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

To read a snapshot 'VAR10' into the IDL structure ff, type the following command

```
IDL> pc_read_var, obj=ff, varfile='VAR10', /trimall
```

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

```
IDL> print, tag_names(ff)
T X Y Z DX DY DZ UU LNRHO AA
```

One can access the individual variables by typing ff.varname, e.g.

```
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. 5.15.3.

5.15.5 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/. 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:

```
import_all pencil
```

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

```
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:

```
#!/usr/bin/python
import numpy as np
import pylab as plt
import pencil as pc
import atexit
#import readline
import rlcompleter

# 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:

```
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.:

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.

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

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

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

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

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

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

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:

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). You can save the plot into a file using the GUI or with

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

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

In [1]: var = pc.read_var()

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.

5.16 **在多核处理器上跑吧** 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.)

5.16.1 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 4×4 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'.

5.16.2 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, 8×16 is 3 times slower, and 16×8 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_slowest=F.

5.16.3 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

$$\frac{N_w}{P_w} = \frac{N_x \times N_y \times N_z}{P_x \times P_y \times P_z}. \tag{26}$$

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

$$\frac{N_w^{(\rm eff)}}{P_w} = \left(\frac{N_x}{P_x} + Q\right) \times \left(\frac{N_y}{P_y} + Q\right) \times \left(\frac{N_z}{P_z} + Q\right). \tag{27}$$

There is efficient scaling only when

$$\min\left(\frac{N_x}{P_x}, \frac{N_y}{P_y}, \frac{N_z}{P_z}\right) \gg Q. \tag{28}$$

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

$$\frac{N_w^{(\rm eff)}}{P_w} = (N + Q) \times \left(\frac{N}{P} + Q\right)^2. \tag{29}$$

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

$$\frac{N_w^{(\rm eff)}}{N_w} = (N + Q) \times \left(\frac{N}{P} + Q\right)^2 \times \frac{P_w}{N_w}. \tag{30}$$

These factors are listed in Table 3.

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.

表 3: $N_w^{(\rm eff)}/N_w$ versus N and P.

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

5.17 **双精度运算** Running in double-precision

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

Add the lines

```
# 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 Sect. F.1.

5.18 **功率谱** Power spectrum

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

$$\tilde{u}(k_x,k_y,k_z) \;=\; \mathcal{F}(u(x,y,z)) \;=\; \frac{1}{N_xN_yN_z}\sum_{p=0}^{N_x-1}\sum_{q=0}^{N_y-1}\sum_{r=0}^{N_z-1}u(x_p,y_q,z_r) \times\exp(-ik_xx_p)\exp(-ik_yy_q)\exp(-ik_zz_r), \tag{31}$$

where

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

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

$$P(k)=\frac{1}{2}\tilde{u}\tilde{u}^*, \tag{32}$$

where

$$k=\sqrt{k_x^2+k_y^2+k_z^2}. \tag{33}$$

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:

```
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.

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.

5.19 **结构函数** Structure functions

We define the p-th order longitudinal structure function of $\boldsymbol{u}$ as

$$S^p_{\rm long}(l) = \langle |u_x(x+l,y,z) - u_x(x,y,z)|^p \rangle \ , \tag{34}$$

while the transverse is

$$S^p_{\rm trans}(l) = \langle |u_y(x+l,y,z) - u_y(x,y,z)|^p \rangle + \langle |u_z(x+l,y,z) - u_z(x,y,z)|^p \rangle \ . \tag{35}$$

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

表 4: Examples of mesh layouts for which Fourier transforms with x parallelization is possible.

ny	nprocyx	nprocy	nprocz	ncpus
256	1	16	16	256
256	2	16	16	512
256	4	16	16	1024
256	8	16	16	2048
288	2	16	18	576
512	2	16	32	1024
512	4	16	16	1024
512	4	16	32	2048
576	4	18	32	2304
576	8	18	32	4608
576	16	18	32	9216
1024	4	32	32	4096
1024	4	16	64	4096
1024	8	16	32	4096
1152	4	36	32	4608
1152	4	32	36	4608
2304	2	32	72	4608
2304	4	36	64	9216
2304	4	32	72	9216

```
STRUCT_FUNC = struct_func
```

In ‘src/cparam.local’, define lb_nxgrid and make sure that

```
nxgrid = nygrid = nzgrid = 2**lb_nxgrid
```

E.g.

```
integer, parameter :: lb_nxgrid=5
integer, parameter :: nxgrid=2**lb_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 qmax, such that the largest moment calculated is qmax -1. The next imax numbers represent the first moment structure function for the first snapshot, here

$$\text{imax} = 2\frac{\ln(\text{nxgrid})}{\ln 2} - 2. \tag{36}$$

The next imax numbers contain the second moment structure function, and so on until qmax-1. The following imax numbers then contain the data of the signed third order structure function i.e. $S^3_{\rm long}(l) = \langle [u_x(x+l,y,z) - u_x(x,y,z)]^3 \rangle$.

The following imax $\times$ qmax $\times$ 2 numbers are zero if nr_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 structure and you get the time-average of the first order longitudinal structure function (be sure that 'pencil-runs/forced/idl/' is in IDL_PATH). See head of 'pencil-runs/forced/idl/structure.pro' for options to structure.

5.20 **粒子群** Particles

The Pencil Code has modules for tracer particles and for dust particles (see Sect. 6.15). The particle modules are chosen by setting the value of the variable PARTICLES in Makefile.local to either particles_dust or 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 PARTICLES_RADIUS to particles_radius in Makefile.local.

All particle infrastructure is controlled and organized by the Particles_main module. This module is automatically selected by Makefile.src if PARTICLES is different from noparticles. Particle modules are compiled as a separate library. This way the main part of the Pencil Code only needs to know about the 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 cparam.local. Here is a sample of cparam.local for a parallel run with 2,000,000 particles:

```
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
```

The parameter npar is the number of particles in the simulation, mpar_loc is the number of particles that is allowed on each processor and 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 npar. The number of particle species is set through npar_species (assumed to be one if not set). The particle input parameters are given in start.in and run.in. Here is a sample of the particle part of start.in for dust particles:

```
/
&particles_init_pars
  initxxp='gaussian-z', initvvp='random'
  zp0=0.02, delta_vp0=0.01, eps_dtog=0.01, tausp=0.1
  lparticlemesh_tsc=T
/
```

The initial positions and velocities of the dust particles are set in initxxp and 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 lparticlemesh_cic=T) and TSC (Triangular Shaped Cloud, set lparticlemesh_tsc=T). See Youdin & Johansen (2007) for details.

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.

5.20.1 并行的粒子 Particles in parallel

The particle variables (e.g. $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$) 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 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 mpar_loc (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:

```
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.

5.20.2 大规模粒子群 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,

```
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

```
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

```
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.

5.20.3 随机数生成器 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.

5.21 **非笛卡尔坐标系** Non-cartesian coordinate systems

Spherical coordinates are invoked by adding the following line in the file 'start.in'

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

One can also invoke cylindrical coordinates by saying cylindrical_coords instead. In practice, the names (x, y, z) are still used, but they refer then to (r, θ, ϕ) or (r, ϕ, z) instead.

When working with curvilinear coordinates it is convenient to use differential operators in the non-coordinate basis, so the derivatives are taken with respect to length, and not in a mixed fashion with respect to length for $\partial/\partial r$ and with respect to angle for $\partial/\partial\theta$ and $\partial/\partial\phi$. The components in the non-coordinate basis are denoted by hats, see, e.g., [?], p. 213; see also Appendix B of [?]. For spherical polar coordinates the only nonvanishing Christoffel symbols (or connection coefficients) are

$$\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}} = \Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}} = -\Gamma^{\hat{r}}{}_{\hat{\theta}\hat{\theta}} = -\Gamma^{\hat{r}}{}_{\hat{\phi}\hat{\phi}} = 1/r, \tag{37}$$

$$\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}} = -\Gamma^{\hat{\theta}}{}_{\hat{\phi}\hat{\phi}} = \cot\theta/r. \tag{38}$$

The Christoffel symbols enter as correction terms for the various differential operators in addition to a term calculated straightforwardly in the non-coordinate basis. The derivatives of some relevant Christoffel symbols are

$$\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta},\hat{\theta}} = \Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi},\hat{\phi}} = \Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi},\hat{\phi}} = 0 \tag{39}$$

$$\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta},\hat{r}} = \Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi},\hat{r}} = -r^{-2} \tag{40}$$

$$\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi},\hat{\theta}} = -r^{-2}\sin^{-2}\theta \tag{41}$$

Further details are given in Appendix I.

6 **方程** The equations

Pencil Code 求解的方程基本上都是标准的可压缩磁流体力学方程. 不过模块化的代码结构能够允许用户做出一些磁流体力学方程的改变. 你可以关掉某些方程, 或者将方程中的某些项消掉 (比如说 nomagnetic, noentropy 等等).

在本章节中方程都会以尽可能完整的形式展现出来. Pencil Code 团队希望能够将这些方程尽可能地实现, 或者退而求其次实现其化简版.

6.1 **连续性方程** Continuity equation

在本代码中, 连续性方程 $\partial\rho/\partial t + \boldsymbol{\nabla}\cdot\rho\boldsymbol{u} = 0$, 是以 $\ln\rho$ 代替 ρ 来写的,

$$\frac{\mathrm{D}\ln\rho}{\mathrm{D}t} = -\boldsymbol{\nabla}\cdot\boldsymbol{u}\,. \tag{42}$$

这里 ρ 表示密度, $\boldsymbol{u}$ 是流体速度, t 是时间而 $\mathrm{D}/\mathrm{D}t \equiv \partial/\partial t + \boldsymbol{u}\cdot\boldsymbol{\nabla}$ 是对流导数.(翻译存疑) the convective derivative.

6.2 **动量方程** Equation of motion

在动量方程汇总, 如果是理想气体的话, 压强项可以写作 $-\rho^{-1}\boldsymbol{\nabla}p = -c_{\mathrm{s}}^2(\boldsymbol{\nabla}s/c_p + \boldsymbol{\nabla}\ln\rho)$, 其中声速平方可由下式给出

$$c_{\mathrm{s}}^2 = \gamma\frac{p}{\rho} = c_{\mathrm{s}0}^2\exp\left[\gamma s/c_p + (\gamma{-}1)\ln\frac{\rho}{\rho_0}\right], \tag{43}$$

式中 $\gamma = c_p/c_v$ 是比热比, 或者说 绝热指数. 注意 c_{s}^2 和温度成正比, 因为 $c_{\mathrm{s}}^2 = (\gamma-1)c_pT$.

动量方程如下所示

$$\begin{aligned}\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = \quad & -c_{\mathrm{s}}^2\boldsymbol{\nabla}\left(\frac{s}{c_p} + \ln\rho\right) - \boldsymbol{\nabla}\Phi_{\mathrm{grav}} + \frac{\boldsymbol{j}\times\boldsymbol{B}}{\rho} \\ & +\nu\left(\nabla^2\boldsymbol{u} + \frac{1}{3}\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot\boldsymbol{u} + 2\mathrm{S}\cdot\boldsymbol{\nabla}\ln\rho\right) + \zeta\left(\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot\boldsymbol{u}\right);\end{aligned} \tag{44}$$

式中 Φ_{grav} 为重力势能, $\boldsymbol{j}$ 电流密度, $\boldsymbol{B}$ 磁感应强度, ν 运动粘度, ζ 体粘滞系数. 并且在笛卡尔坐标下

$$\mathrm{S}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\boldsymbol{\nabla}\cdot\boldsymbol{u}\right) \tag{45}$$

是剪切速率张量, 它的矩阵迹等于零, 因为它可以写成广义应变率张量减去其自身迹的形式. 如果是在曲线坐标系下, 这里的偏微分就需要用斜变微分来代替 (用分号表示), 可以写成 $\mathrm{S}_{ij} = \frac{1}{2}(u_{i;j} + u_{j;i}) - \frac{1}{3}\delta_{ij}\boldsymbol{\nabla}\cdot\boldsymbol{u}$.

对于两个粘滞系数项的解释会因为所使用的粘性模型而异, 并依赖于给数值计算对应模块的具体参数. 参见 §6.6.

对于恒温流体力学, 参见 §6.4 .

6.3 **磁感应方程** Induction equation

$$\frac{\partial \boldsymbol{A}}{\partial t} = \boldsymbol{u} \times \boldsymbol{B} - \eta\mu_0 \boldsymbol{j} \ . \tag{46}$$

此处 $\boldsymbol{A}$ 是磁矢势, $\boldsymbol{B} = \nabla \times \boldsymbol{A}$ 是磁感应强度, $\eta = 1/(\mu_0\sigma)$ 磁扩散率 (σ 为电导率), 以及 μ_0 为真空磁导率. 这种形式的磁感应方程能够确保 Weyl gauge $\Phi = 0$, 而 Φ 意味着标量势.

6.4 **熵方程** Entropy equation

目前的热力学模块 entropy 计划用 entropy s 来作为热力学物理量中的基石, 而不是通过读者更熟悉的热能密度 thermal energy e. 于是 $\ln\rho$ 和 s 便成为两个基础的热力学变量. 我们这么做的原因是因为熵 (至少) 于对流过程而言是更自然的物理量; 熵梯度的正负号帮助确定对流的 (不) 稳定性; Rayleigh 数在流体静力学的实例中和熵的梯度成正比等 (翻译存疑 the Rayleigh number is proportional to the entropy gradient of the associated hydrostatic reference solution, etc).

求解的方程如下

$$\rho T \frac{\mathrm{D}s}{\mathrm{D}t} = \mathcal{H} - \mathcal{C} + \boldsymbol{\nabla}\cdot(K\boldsymbol{\nabla}T) + \eta\mu_0\boldsymbol{j}^2 + 2\rho\nu \mathrm{S}\otimes\mathrm{S} + \zeta\rho\left(\boldsymbol{\nabla}\cdot\boldsymbol{u}\right)^2 \ . \tag{47}$$

其中, T 为温度, c_p 恒温比热, $\mathcal{H}$ 和 $\mathcal{C}$ 是显示的加热和降温项, K 为 (热) 辐射传导率, ζ 描述的是体粘滞系数, 以及 S 为迹为零的剪切速率张量 (翻译存疑 traceless).

熵模块用来求解比熵, s(翻译存疑 specific entrop). 右侧的热传导项可以用下面的形式写

$$\begin{aligned}
&\frac{\boldsymbol{\nabla}\cdot(K\boldsymbol{\nabla}T)}{\rho T} && (48)\\
&= c_p\chi\Big[\nabla^2\ln T + \boldsymbol{\nabla}\ln T\cdot\boldsymbol{\nabla}(\ln T + \ln\chi + \ln\rho)\Big] && (49)\\
&= c_p\chi\left[\gamma\nabla^2 s/c_p + (\gamma-1)\nabla^2\ln\rho\right] \\
&\quad + c_p\chi\left[\gamma\boldsymbol{\nabla}s/c_p + (\gamma-1)\boldsymbol{\nabla}\ln\rho\right]\cdot\left[\gamma\left(\boldsymbol{\nabla}s/c_p + \boldsymbol{\nabla}\ln\rho\right) + \boldsymbol{\nabla}\ln\chi\right] \ , && (50)
\end{aligned}$$

$\chi = K/(\rho c_p)$ 是热扩散率. 后一等式表明, s 的扩散率是 $\gamma\chi$, 我们在方程 Eq. (7) 中已经用到了它.

在对常数 K 的另一种表示中, 右侧的热传导率也可以写为

$$\frac{\boldsymbol{\nabla}\cdot(K\boldsymbol{\nabla}T)}{\rho T} = \frac{K}{\rho}\Big[\nabla^2\ln T + (\boldsymbol{\nabla}\ln T)^2\Big] \tag{51}$$

当 K 已知时, 我们常常会用这种形式.

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 $\partial s/\partial t = 0$).

温度不变条件下的数值模拟推荐使用 noentropy 模块.

6.4.1 粘滞加热 Viscous heating

我们将粘滞系数在这里写成一个张量的散度形式 $\tau_{ij,j}$,

$$\rho \frac{\partial u_i}{\partial t} = ... + \tau_{ij,j}\,, \tag{52}$$

其中 $\tau_{ij} = 2\nu\rho \mathrm{S}_{ij}$ 为应力张量. 粘滞加热功率密度为 P

$$\begin{aligned} P &= u_i \tau_{ij,j} && (53)\\ &= \frac{\partial}{\partial x_j}\left(u_i \tau_{ij}\right) - u_{i,j}\tau_{ij} && (54) \end{aligned}$$

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 τ_{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

$$u_{i,j}\tau_{ij} = \frac{1}{2}\nu\rho(u_{i,j} + u_{j,i})(2\mathrm{S}_{ij})\,. \tag{55}$$

但由于 S_{ij} 的迹为零, 我们可以添加正比于 δ_{ij} 的数给它, 特别的一种加法是这样的

$$\begin{aligned} u_{i,j}\tau_{ij} &= \frac{1}{2}(u_{i,j} + u_{j,i})(2\nu\rho\mathrm{S}_{ij}) && (56)\\ &= \frac{1}{2}(u_{i,j} + u_{j,i} - \frac{1}{3}\delta_{ij}\boldsymbol{\nabla}\cdot\boldsymbol{u})(2\nu\rho\mathrm{S}_{ij}) && (57)\\ &= 2\nu\rho\mathrm{S}^2, && (58) \end{aligned}$$

而它是正定的.

6.4.2 可选的描述?? 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 $K\boldsymbol{\nabla}T$) is important. Another alternative is to use another module, i.e. set ENTROPY=temperature_idealgas in 'Makefile.local'.

When pretend_lnTT=T is set, the entropy equation

$$\frac{\partial s}{\partial t} = -\boldsymbol{u}\cdot\boldsymbol{\nabla}s + \frac{1}{\rho T}\mathrm{RHS} \tag{59}$$

is replaced by

$$\frac{\partial \ln T}{\partial t} = -\boldsymbol{u}\cdot\boldsymbol{\nabla}\ln T + \frac{1}{\rho c_v T}\mathrm{RHS} - (\gamma-1)\,\boldsymbol{\nabla}\cdot\boldsymbol{u}, \tag{60}$$

where RHS is the right hand side of equation (47).

6.5 **被动标量的输运方程** Transport equation for a passive scalar

In conservative form, the equation for a passive scalar is

$$\frac{\partial}{\partial t}(\rho c) + \boldsymbol{\nabla} \cdot [\rho c \boldsymbol{u} - \rho \mathcal{D} \nabla c] = 0. \tag{61}$$

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

$$\frac{\mathrm{D} \ln c}{\mathrm{D}t} = \mathcal{D} \left[\nabla^2 \ln c + (\boldsymbol{\nabla} \ln \rho + \boldsymbol{\nabla} \ln c) \cdot \boldsymbol{\nabla} \ln c \right]. \tag{62}$$

Using $\ln c$ instead of c has the advantage that it enforces $c > 0$ for all times. 然而, 这样做的缺点是我们无法确保 $c = 0$. 出于这个原因, 我们最后通过设置 PSCALAR=pscalar_nolog 采用了非对数的形式.

6.6 **体粘滞系数** Bulk viscosity

对单原子气体而言, 它的体粘滞系数作用是可以忽略不计的, 所以我们在大多数运算中并不会用到这个系数. 然而, 对于超声速气流或其他类似情况而言, 研究者可能会想要加入激波粘性 (在最简单的表示中, 它表现为体粘滞系数)

6.6.1 激波粘性 Shock viscosity

激波粘性, 它在本代码和 Åke Nordlund 的 Stagger Code 中都有使用. 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,

$$\zeta_{\mathrm{shock}} = c_{\mathrm{shock}} \left\langle \max_5 [(-\boldsymbol{\nabla} \cdot \boldsymbol{u})_+] \right\rangle (\min(\delta x, \delta y, \delta z))^2, \tag{63}$$

where c_{shock} is a constant defining the strength of the shock viscosity. 在程序包中这个无量纲系数变量名为 nu_shock, 它的取值通常在 1 左右. 假设粘性系数仅作为体粘滞系数而作用, 那么整个应力张量可以表示为

$$\boldsymbol{\tau}_{ij} = 2\rho\nu \mathrm{S}_{ij} + \rho \zeta_{\mathrm{shock}} \delta_{ij} \boldsymbol{\nabla} \cdot \boldsymbol{u}. \tag{64}$$

假设 $\nu = \mathrm{const}$, 但 $\zeta \neq \mathrm{const}$, 于是有

$$\rho^{-1} \boldsymbol{F}_{\mathrm{visc}} = \nu \left(\nabla^2 \boldsymbol{u} + \frac{1}{3} \boldsymbol{\nabla} \boldsymbol{\nabla} \cdot \boldsymbol{u} + 2\mathrm{S} \cdot \boldsymbol{\nabla} \ln \rho \right) + \zeta_{\mathrm{shock}} \left[\boldsymbol{\nabla} \boldsymbol{\nabla} \cdot \boldsymbol{u} + (\boldsymbol{\nabla} \ln \rho + \boldsymbol{\nabla} \ln \zeta_{\mathrm{shock}}) \boldsymbol{\nabla} \cdot \boldsymbol{u} \right]. \tag{65}$$

$$\rho^{-1}\Gamma_{\rm visc} = 2\nu \mathrm{S}^2 + \zeta_{\rm shock}(\boldsymbol{\nabla}\cdot\boldsymbol{u})^2. \tag{66}$$

在特殊的有周期性边界条件的情况下, 有 $2\langle \mathrm{S}^2\rangle = \langle \boldsymbol{\omega}^2\rangle + \frac{4}{3}\langle(\boldsymbol{\nabla}\cdot\boldsymbol{u})^2\rangle$.

6.7 **气体状态方程** Equation of state

在目前的代码构造中, 只有氢的电离程度在代码中实现了. 其他成分 (目前支持 He 和 H_2) 有着固定的电离程度参数. 压强正比于粒子总数, 可以写成

$$p = (n_{\rm HI} + n_{\rm HII} + n_{\rm H_2} + n_{\rm e} + n_{\rm He} + ...)k_{\rm B}T. \tag{67}$$

It is convenient to normalize to the total number of H both in atomic and in molecular hydrogen, $n_{\rm Htot} \equiv n_{\rm H} + 2n_{\rm H_2}$, where $n_{\rm HI} + n_{\rm HII} = n_{\rm H}$, and define $x_{\rm e} \equiv n_{\rm e}/n_{\rm Htot}$, $x_{\rm He} \equiv n_{\rm He}/n_{\rm Htot}$, and $x_{\rm H_2} \equiv n_{\rm H_2}/n_{\rm Htot}$. Substituting $n_{\rm H} = n_{\rm Htot} - 2n_{\rm H_2}$, we have

$$p = (1 - x_{\rm H_2} + x_{\rm e} + x_{\rm He} + ...)n_{\rm Htot}k_{\rm B}T. \tag{68}$$

This can be written in the more familiar form

$$p = \frac{\mathcal{R}}{\mu}\rho T, \tag{69}$$

where $\mathcal{R} = k_{\rm B}/m_{\rm u}$ and $m_{\rm u}$ is the atomic mass unit (which is for all practical purposes the same as $m_{\rm Htot}$) and

$$\mu = \frac{n_{\rm H} + 2n_{\rm H_2} + n_{\rm e} + 4n_{\rm He}}{n_{\rm H} + n_{\rm H_2} + n_{\rm e} + n_{\rm He}} = \frac{1 + 4x_{\rm He}}{1 - x_{\rm H_2} + x_{\rm e} + x_{\rm He}} \tag{70}$$

is the mean molecular weight (which is here dimensionless; see Kippenhahn & Weigert 1990, p. 102). The factor 4 is really to be substituted for 3.97153. Some of the familiar relations take still the usual form, in particular $e = c_v T$ and $h = c_p T$ with $c_v = \frac{3}{2}\mathcal{R}/\mu$ and $c_p = \frac{5}{2}\mathcal{R}/\mu$.

The number ratio, $x_{\rm He}$, is more commonly expressed as the mass ratio, $Y = m_{\rm He}n_{\rm He}/(m_{\rm H}n_{\rm Htot} + m_{\rm He}n_{\rm He})$, or $Y = 4x_{\rm He}/(1 + 4x_{\rm He})$, or $4x_{\rm He} = (1/Y - 1)^{-1}$. For example, $Y = 0.27$ corresponds to $x_{\rm He} = 0.092$ and $Y = 0.25$ to $x_{\rm He} = 0.083$. Note also that for 100% H_2 abundance, $x_{\rm H_2} = 1/2$.

在下面的章节中, 电离度用 $y = n_{\rm e}/n_{\rm H}$ 来表示. 在存在 H_2 的情况下, 它和 $x_{\rm e}$ 略有不同. 如果我们用 $n_{\rm Htot}$ 替换掉 $n_{\rm H}$, 就能得到 $y = x_{\rm e}/(1 - 2x_{\rm H_2})$.

6.8 **电离** Ionization

这一部分的代码可以通过在 'Makefile.local' 中设置 EOS=eos_ionization (或者 EOS=eos_-temperature_ionization) 来调用. The equation of state described below works for variable ionization, and the entropy offset is different from that used in Eq. (43), which is now no longer valid. 作为替代措施, 研究者可以用 EOS=eos_fixed_ionization, 这允许研究者手动设置电离

程度. 它的熵的归一化 (翻译存疑 normalization) 和 EOS=eos_ionization 的是一样的. 下面会将这一情况讲述地更清晰.[12]

这里提到的气体是由部分电离的氢和中性的氦组成的. 其中有四种不同的粒子种类, 均可以被单独视为理想气体.

电离度 y 表示电离的氢在氢的总数 $n_{\rm H}$ 中的占比, 可以从 Saha 方程中求得. Saha 方程可以写成以下形式

$$\frac{y^2}{1-y}=\frac{1}{n_{\rm H}}\left(\frac{m_{\rm e}k_{\rm B}T}{2\pi\hbar^2}\right)^{3/2}\exp\left(-\frac{\chi_{\rm H}}{k_{\rm B}T}\right)\ . \tag{71}$$

温度 T 不可以直接从 Pencil Code 的独立变量 $\ln\rho$ 和 s 中获得, 但它依赖于 y . 所以, y 的计算实质上是一个求根问题.

由 i 粒子构成的理想气体的熵可以通过 Sackur-Tetrode 方程获得

$$S_i=k_{\rm B}N_i\left(\ln\left[\frac{1}{n_{\rm tot}}\left(\frac{m_ik_{\rm B}T}{2\pi\hbar^2}\right)^{3/2}\right]+\frac{5}{2}\right)\ . \tag{72}$$

这里 N_i 表示单一类型的粒子的数目, 而 $n_{\rm tot}$ 表示所有粒子的总的数密度.

In addition to the individual entropies we also have to take the entropy of mixing, $S_{\rm mix}=-N_{\rm tot}k_{\rm B}\sum_i p_i\ln p_i$, 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.

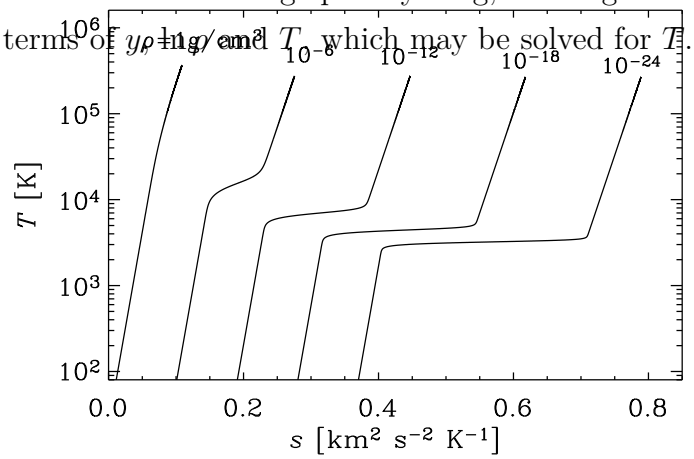


图 4: 在不同的密度下, 温度对熵的依赖关系

只要给定 $\ln\rho$ 和 s , 我们就可以通过找到下列方程的根计算电离百分比 y

$$f(y)=\ln\left[\frac{1-y}{y^2}\frac{1}{n_{\rm H}}\left(\frac{m_{\rm e}k_{\rm B}T(y)}{2\pi\hbar^2}\right)^{3/2}\right]-\frac{\chi_{\rm H}}{k_{\rm B}T(y)}\ . \tag{73}$$

[12]We omit here the contribution of H_2.

在电离的情况下, 一些气体的热力学量会受电离百分比 y 的影响. 比如说压强 $P = (1+y+x_{\rm He})n_{\rm H}k_{\rm B}T$, 内能 $E = \frac{3}{2}(1+y+x_{\rm He})n_{\rm H}k_{\rm B}T + y\chi_{\rm H}$, 其中 $x_{\rm He}$ 给出的是中性氦占氢的总数的百分比. 在不同的密度条件下, 温度对熵的依赖关系已经在 Fig. 4 中刻画出来了.

更多的关于求解熵的信息附在了附录里的 Sect. H.6.

6.8.1 双极性扩散 Ambipolar diffusion

Pencil code 中另外一种处理电离的方法是通过使用 neutrals 模块. 这一模块能够通过双流体模型, 求解中性气体和电离气体的耦合方程

$$\frac{\partial \rho_i}{\partial t} = -\boldsymbol{\nabla}\cdot(\rho_i \boldsymbol{u}_i) + \mathcal{G} \tag{74}$$

$$\frac{\partial \rho_n}{\partial t} = -\boldsymbol{\nabla}\cdot(\rho_n \boldsymbol{u}_n) - \mathcal{G} \tag{75}$$

$$\frac{\partial (\rho_i \boldsymbol{u}_i)}{\partial t} = -\boldsymbol{\nabla}\cdot(\rho_i \boldsymbol{u}_i : \boldsymbol{u}_i) - \boldsymbol{\nabla}\left(p_i + p_e + \frac{B^2}{2\mu_0}\right) + \mathcal{F} \tag{76}$$

$$\frac{\partial (\rho_n \boldsymbol{u}_n)}{\partial t} = -\boldsymbol{\nabla}\cdot(\rho_n \boldsymbol{u}_n : \boldsymbol{u}_n) - \boldsymbol{\nabla} p_n - \mathcal{F} \tag{77}$$

$$\frac{\partial \boldsymbol{A}}{\partial t} = \boldsymbol{u}_i \times \boldsymbol{B} \tag{78}$$

这里的下标 n 和 i 分别表示中性或电离了的粒子. $\mathcal{G}$ 和 $\mathcal{F}$ 是关于两种流体间交换质量和动量的物理量, 通过以下公式给出

$$\mathcal{G} = \zeta\rho_n - \alpha\rho_i^2 \tag{79}$$

$$\mathcal{F} = \zeta\rho_n \boldsymbol{u}_n - \alpha\rho_i^2 \boldsymbol{u}_i + \gamma\rho_i\rho_n(\boldsymbol{u}_n - \boldsymbol{u}_i)\ . \tag{80}$$

在上面的方程中, ζ 是电离系数, α 是复合系数 (翻译存疑 recombination coefficient) 以及 γ 为碰撞阻力强度. 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.

在本程序包中, 两个方程 Eq. (74) 和 Eq. (76) 分别在 'density.f90' 和 'hydro.f90' 中求解. 方程 Equation 75 在 'neutraldensity.f90' 中求解, 而 Eq. (77) 则在文件 'neutralvelocity.f90'内. 样例 '1d-test/ambipolar-diffusion' 已经有了现成的给双流体 (离子和中性粒子) 模型的参数配置.

6.9 **辐射传输** Radiative transfer

这里我们仅列出基础的方程, 更详细的代码实现说明在 Sect. H.7 和 Heinemann et al. (2006) 的初始论文中呈现.

辐射传输的基本方程是

$$\frac{dI}{d\tau} = -I + S \,, \tag{81}$$

其中

$$\tau \equiv \int_0^s \chi(s')\, ds' \tag{82}$$

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 variables 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 **自重力** Self-gravity

The Pencil Code can consider the self-gravity of the fluid in the simulation box by adding the term

$$\frac{\partial \boldsymbol{u}}{\partial t} = \ldots - \boldsymbol{\nabla}\phi_{\rm self} \tag{83}$$

to the equation of motion. The self-potential $\phi_{\rm self}$ (or just ϕ for simplicity) satisfies Poisson's equation

$$\nabla^2\phi = 4\pi G\rho \,. \tag{84}$$

The solution for a single Fourier component at scale $\boldsymbol{k}$ is

$$\phi_{\boldsymbol{k}} = -\frac{4\pi G\rho_{\boldsymbol{k}}}{k^2} \,. \tag{85}$$

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 $\Delta y(t) = \mathrm{mod}[(3/2)\Omega_0 L_x t, L_y]$ in the y-direction. We follow here the method from [?] 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 [?] 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.

[13] We were kindly made aware of the possibility of interpolating in Fourier space by C. McNally on his website.

6.11 **不可压缩流体和非弹性流体方程** Incompressible and anelastic equations

这一部分还没有编写文档, 有待填补.

6.12 **尘埃方程** 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

$$\frac{\mathrm{D}\boldsymbol{u}_\mathrm{d}}{\mathrm{D}t} = \ldots - \frac{1}{\tau_\mathrm{s}}\left(\boldsymbol{u}_\mathrm{d} - \boldsymbol{u}\right) . \tag{86}$$

Here τ_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

$$\tau_\mathrm{s} = \frac{a_\mathrm{d}\rho_\mathrm{s}}{c_\mathrm{s}\rho} , \tag{87}$$

where a_d is the radius of the dust grain and ρ_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

$$\frac{\mathrm{d}n_k}{\mathrm{d}t} = \frac{1}{2}\sum_{i+j=k} A_{ij} n_i n_j - n_k \sum_{i=1}^{\infty} A_{ik} n_i . \tag{88}$$

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

$$\frac{\mathrm{d}N}{\mathrm{d}t} = \frac{1}{\tau_\mathrm{cond}} N^{\frac{d-1}{d}} . \tag{89}$$

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 τ_cond is calculated from

$$\frac{1}{\tau_\mathrm{cond}} = A_1 v_\mathrm{th} \alpha n_\mathrm{mon} \left\{ 1 - \frac{1}{S_\mathrm{mon}} \right\} , \tag{90}$$

where A_1 is the surface area of a monomer, α is the condensation efficiency, n_mon is the number density of monomers in the gas and S_mon is the saturation level of the monomer given by

$$S_\mathrm{mon} = \frac{P_\mathrm{mon}}{P_\mathrm{sat}} . \tag{91}$$

[14]See master's thesis of A. Johansen (can be downloaded from http://www.mpia.de/homes/johansen/research_en.php)

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

所有尘埃粒子满足连续性方程

$$\frac{\partial \rho_{\rm d}}{\partial t} + \boldsymbol{\nabla} \cdot (\rho_{\rm d} \boldsymbol{u}_{\rm d}) = 0. \tag{92}$$

6.13 **扩散近似下的宇宙线压力** Cosmic ray pressure in diffusion approximation

Cosmic rays are treated in the diffusion approximation. The equation of state is $p_{\rm c} = (\gamma_{\rm c}) e_{\rm c}$ where the value of $\gamma_{\rm c}$ is usually somewhere between 14/9 and 4/3. In the momentum equation (44) the cosmic ray pressure force, $-\rho^{-1} \boldsymbol{\nabla} p_{\rm c}$ is added on the right hand side, and $e_{\rm c}$ satisfies the evolution equation

$$\frac{\partial e_{\rm c}}{\partial t} + \boldsymbol{\nabla} \cdot (e_{\rm c} \boldsymbol{u}) + p_{\rm c} \boldsymbol{\nabla} \cdot \boldsymbol{u} = \partial_i (K_{ij} \partial_j e_{\rm c}) + Q_{\rm c}, \tag{93}$$

where $Q_{\rm c}$ is a source term and

$$K_{ij} = K_\perp \delta_{ij} + (K_\parallel - K_\perp) \hat{B}_i \hat{B}_j \tag{94}$$

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.

$$\partial_i (K_{ij} \partial_j e_{\rm c}) = -\boldsymbol{U}_{\rm c} \cdot \boldsymbol{\nabla} e_{\rm c} + K_{ij} \partial_i \partial_j e_{\rm c}. \tag{95}$$

where $U_{{\rm c}\,i} = -\partial K_{ij} / \partial x_j$ acts like an extra velocity trying to straighten magnetic field lines. We can write this term also as $\boldsymbol{U}_{\rm c} = -(K_\parallel - K_\perp) \boldsymbol{\nabla} \cdot (\hat{\boldsymbol{B}} \hat{\boldsymbol{B}})$, where the last term is a divergence of the dyadic product of unit vectors.[15] However, near magnetic nulls, this term can becomes infinite. In order to avoid this problem we are forced to limit $\boldsymbol{\nabla} \cdot (\hat{\boldsymbol{B}} \hat{\boldsymbol{B}})$, and hence $|\boldsymbol{U}_{\rm c}|$, 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. (93) by a divergence of a flux that in turn obeys the equation

$$\frac{\partial \mathcal{F}_{{\rm c}i}}{\partial t} = -\tilde{K}_{ij} \nabla_j e_{\rm c} - \frac{\mathcal{F}_{{\rm c}i}}{\tau} \quad \text{(non-Fickian diffusion)}, \tag{96}$$

where $K_{ij} = \tau \tilde{K}_{ij}$ would be the original diffusion tensor of Eq. (94), if the time derivative were negligible. Further details are described in Snodin et al. (2006).

[15] In practice, we calculate $\partial_j (\hat{B}_i \hat{B}_j) = (\delta_{ij} - 2 \hat{B}_i \hat{B}_k) \hat{B}_j B_{k,j} / |\boldsymbol{B}|$, where derivatives of $\boldsymbol{B}$ are calculated as $B_{i,j} = \epsilon_{ikl} A_{l,jk}$.

6.14 **手性磁流体力学** Chiral MHD

At high energies, $k_{\rm B}T \gtrsim 10$ MeV, a quantum effect called the chiral magnetic effect (CME) can modify the MHD equations. The CME occurs in magnetized relativistic plasmas, in which the number density of left-handed fermions, $n_{\rm L}$, differs from the one of right-handed fermions, $n_{\rm R}$ (see e.g., Kharzeev et al. (2013) for a review). This asymmetry is described by the chiral chemical potential $\mu_5 \equiv 6\,(n_{\rm L} - n_{\rm R})\,(\hbar c)^3/(k_{\rm B}T)^2$, where T is the temperature, $k_{\rm B}$ is the Boltzmann constant, c is the speed of light, and $\hbar$ is the reduced Planck constant. In the presence of a magnetic field, μ_5 leads to the occurrence of the current

$$\boldsymbol{J}_{\rm CME} = \frac{\alpha_{\rm em}}{\pi\hbar}\mu_5\boldsymbol{B}, \tag{97}$$

where $\alpha_{\rm em} \approx 1/137$ is the fine structure constant.

The chiral current (97) adds to the classical Ohmic current, leading to a modification of the Maxwell equations. As a result, the induction equation is extended by one additional term:

$$\frac{\partial \boldsymbol{A}}{\partial t} = \boldsymbol{U} \times \boldsymbol{B} - \eta\,(\boldsymbol{\nabla} \times \boldsymbol{B} - \mu\boldsymbol{B})\,, \tag{98}$$

where the chiral chemical potential μ_5 is normalized such that $\mu = (4\alpha_{\rm em}/\hbar c)\mu_5$. The latter is determined by the evolution equation

$$\frac{D\mu}{Dt} = D_5\,\Delta\mu + \lambda\,\eta\,\left[\boldsymbol{B}\cdot(\boldsymbol{\nabla} \times \boldsymbol{B}) - \mu\boldsymbol{B}^2\right] - \Gamma_{\rm f}\mu, \tag{99}$$

where D_5 is a chiral diffusion coefficient, λ the chiral feedback parameter, and $\Gamma_{\rm f}$ the rate of chiral flipping reactions. All remaining evolution equations are the same as in classical MHD. Details on the derivation of the chiral MHD equations can be found in Boyarsky et al. (2015) and Rogachevskii et al. (2017).

In the Pencil Code, the chiral MHD equations can be solved by adding the line

```
SPECIAL = special/chiral_mhd
```

to src/Makefile.local. Further, for running the chiral MHD module, one needs to add

```
&special_init_pars
initspecial='const', mu5_const=10.
```

to start.in and

```
&special_run_pars
diffmu5=1e-4, lambda5=1e3, cdtchiral=1.0
```

to run.in, where we have chosen exemplary values for the chiral parameters.

Caution should be taken when solving the chiral MHD equations numerically, since the evolution of the plasma can be strongly affected by the CME. In particular, the initial value of μ is related

to a small-scale dynamo instability. In order to resolve this instability in a domain of size $(2\pi)^3$, the minimum number of grid points is given as:

$$N_{\rm grid} \gtrsim \left(\frac{\mu_0}{\lambda}\right)^{1/2} \frac{2\pi}{\nu {\rm Re}_{\rm mesh,crit}}. \tag{100}$$

Also the value of λ should not be choosen too small, since it scales inversely with the saturation magnetic helicity produced by a chiral dynamo. Hence, for a very small λ parameter, the Alfvén time step becomes extremely small in the non-linear stage which can lead to a code crash. More details on the numerical modeling of chiral MHD can be found in Schober et al. (2018).

6.15 **粒子** Particles

粒子是那些有着空间坐标和速度矢量的实体, 而一个流体只有速度矢量场 (流体的连续性方程某种意义上对应着粒子的空间坐标). 本程序包中, 粒子有两种存在形式, 一种是示踪粒子, 另外一种是尘粒.

6.15.1 示踪粒子 Tracer particles

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

$$\frac{\partial \boldsymbol{x}_i}{\partial t} = \boldsymbol{u}\,, \tag{101}$$

where the index i runs over all particles. Here $\boldsymbol{u}$ 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 lquadratic_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.

6.15.2 尘粒 Dust particles

尘粒可以有着和气体不太相似的速度.

$$\frac{{\rm d}\boldsymbol{x}_i}{{\rm d}t} = \boldsymbol{v}_i\,. \tag{102}$$

这些粒子速度遵循着和流体类似的栋梁方程, 只是没有平流项. 尘粒也会受到空气的阻力 (通常正比于尘粒和气体的速度差)

$$\frac{{\rm d}\boldsymbol{v}_i}{{\rm d}t} = \ldots - \frac{1}{\tau_{\rm s}}(\boldsymbol{v}_i - \boldsymbol{u})\,. \tag{103}$$

Here $\tau_{\rm s}$ 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 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 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 $\epsilon_{\rm d}$, and it can be safely ignored when there is much more gas than there is dust, e.g. when $\epsilon_{\rm d} = 0.01$. The friction force on the gas appears in the equation of motion as

$$\frac{\partial \boldsymbol{u}}{\partial t} = \ldots - \frac{\rho_{\rm p}^{(i)}}{\rho} \left(\frac{\partial \boldsymbol{v}^{(i)}}{\partial t} \right)_{\rm drag} \tag{104}$$

Here $\rho_{\rm p}^{(i)}$ is the dust density that particle i represents. This can be set through the parameter eps_todt in &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).

6.16 N 体求解器 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 $a_\bullet$ is comparable to the mean free path λ 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 ρ_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

$$\frac{d\boldsymbol{v}_{p_i}}{dt} = \boldsymbol{F}_{g_i} - \sum_{j\neq i}^{N} \frac{GM_j}{\mathcal{R}_{ij}^2}\hat{\boldsymbol{\mathcal{R}}}_{ij} \tag{105}$$

where $\mathcal{R}_{ij} = |\boldsymbol{r}_{p_i} - \boldsymbol{r}_{p_j}|$ is the distance between particles i and j, and $\hat{\boldsymbol{\mathcal{R}}}_{ij}$ 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

$$\boldsymbol{F}_{g_i} = -G\int_V \frac{[\rho_g(\boldsymbol{r}) + \rho_p(\boldsymbol{r})]\boldsymbol{\mathcal{R}}_i}{(\mathcal{R}_i^2 + b_i^2)^{3/2}}\mathrm{d}V, \tag{106}$$

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 $t_p \ll 2\pi\Omega_{\rm fp}^{-1}$ where $\Omega_{\rm fp}$ 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.

表 5: Scale factor and conformal Hubble parameter for different values of n.

n	a	$\mathcal{H}$	H
0	1	0	0
1/2	$\eta/2$	$1/\eta$	$1/\eta$
2/3	$\eta^2/3$	$2/\eta$	$6/\eta^2$

6.17 **测试场方程** 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. F.3.

6.18 **引力波方程** Gravitational wave equations

The expansion of the universe with time is described by the scale factor $a(\tau)$, where τ is the physical time. Using conformal time, $t(\tau) = \int_0^\tau \mathrm{d}\tau'/a(\tau')$, and dependent variables that are appropriately scaled with powers of a, the hydromagnetic equations can be expressed completely without scale factor [?, ?]. This is not true, however, for the gravitational wave (GW) equations, where a dependence on a remains [?]. 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 × 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 [?]

$$\frac{\partial^2 h_{ij}}{\partial t^2} = -2\mathcal{H}\frac{\partial h_{ij}}{\partial t} + c^2\nabla^2 h_{ij} + \frac{16\pi G}{a^2 c^2} T_{ij} \tag{107}$$

for the six components $1 \le i \le j \le 3$, where t is comoving time, a is the scale factor, $\mathcal{H} = \dot{a}/a$ 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 $\mathcal{H} = 0$. We use the Pencil Code; for the numerical treatment of Eq. (107) and equations (109)–(111). 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,

$$T_{ij}(\boldsymbol{x},t) = \rho u_i u_j - B_i B_j - \tfrac{1}{3}\delta_{ij}(\rho \boldsymbol{u}^2 - \boldsymbol{B}^2). \tag{108}$$

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 ultrarelativistic gas in the form [?, ?]

$$\frac{\partial \ln \rho}{\partial t} = -\frac{4}{3}\left(\boldsymbol{\nabla}\cdot\boldsymbol{u} + \boldsymbol{u}\cdot\boldsymbol{\nabla}\ln\rho\right) + \frac{1}{\rho}\left[\boldsymbol{u}\cdot(\boldsymbol{J}\times\boldsymbol{B}) + \eta\boldsymbol{J}^2\right], \tag{109}$$

$$\begin{aligned}\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{Dt}} &= \frac{\boldsymbol{u}}{3}\left(\boldsymbol{\nabla}\cdot\boldsymbol{u} + \boldsymbol{u}\cdot\boldsymbol{\nabla}\ln\rho\right) - \frac{\boldsymbol{u}}{\rho}\left[\boldsymbol{u}\cdot(\boldsymbol{J}\times\boldsymbol{B}) + \eta\boldsymbol{J}^2\right] \\ &\quad -\frac{1}{4}\boldsymbol{\nabla}\ln\rho + \frac{3}{4\rho}\boldsymbol{J}\times\boldsymbol{B} + \frac{2}{\rho}\boldsymbol{\nabla}\cdot(\rho\nu\mathrm{S}) + \boldsymbol{f},\end{aligned} \tag{110}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} = \nabla\times(\boldsymbol{u}\times\boldsymbol{B} - \eta\boldsymbol{J}), \tag{111}$$

where $\boldsymbol{B} = \nabla\times\boldsymbol{A}$ is the magnetic field expressed in terms of the magnetic vector potential to ensure that $\boldsymbol{\nabla}\cdot\boldsymbol{B} = 0$, $\boldsymbol{J} = \boldsymbol{\nabla}\times\boldsymbol{B}$ is the current density, $\mathrm{D}/\mathrm{Dt} = \partial/\partial\mathrm{t} + \mathrm{u}\cdot\boldsymbol{\nabla}$ is the advective derivative, $\mathrm{S}_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) - \frac{1}{3}\delta_{ij}u_{k,k}$ is the trace-free rate of strain tensor, and $p = \rho c_\mathrm{s}^2$ is the pressure, where $c_\mathrm{s} = c/\sqrt{3}$ 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 $\dot{h}_{ij}$,

$$\tilde{h}_{ij}(\boldsymbol{k},t) = \int h_{ij}(\boldsymbol{x},t)\, e^{-\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{v}} d^3x \quad \text{for } 1\le i\le j\le 3 \tag{112}$$

and compute the components in the TT gauge as

$$\tilde{h}_{ij}^{\mathrm{TT}}(\boldsymbol{k},t) = (P_{il}P_{jm} - \tfrac{1}{2}P_{ij}P_{lm})\,\tilde{h}_{lm}(\boldsymbol{k},t), \tag{113}$$

where $P_{ij} = \delta_{ij} - \hat{k}_i\hat{k}_j$ is the projection operator, and $\hat{\boldsymbol{k}} = \boldsymbol{k}/k$ is the unit vector of $\boldsymbol{k}$, with $k = |\boldsymbol{k}|$ being the modulus. Next, we compute the linear polarization bases

$$e_{ij}^{+} = e_i^1 e_j^1 - e_i^2 e_j^2, \quad e_{ij}^{\times} = e_i^1 e_j^2 + e_i^2 e_j^1, \tag{114}$$

where e^1 and e^2 are unit vectors perpendicular to $\boldsymbol{k}$. Thus

$$\tilde{h}_{+}(\boldsymbol{k},t) = \tfrac{1}{2}e_{ij}^{+}(\boldsymbol{k})\,\tilde{h}_{ij}(\boldsymbol{k},t), \tag{115}$$

$$\tilde{h_{\times}}(\boldsymbol{k},t) = \tfrac{1}{2}e_{ij}^{\times}(\boldsymbol{k})\,\tilde{h}_{ij}(\boldsymbol{k},t). \tag{116}$$

We then return into real space and compute

$$h_{+/\times}(\boldsymbol{x},t) = \int \tilde{h}_{+/\times}(\boldsymbol{k},t)\, e^{\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{x}}\, d^3k/(2\pi)^3. \tag{117}$$

Analogous calculations are performed for $\dot{h}_{+/\times}(\boldsymbol{x},t)$, which are used to compute the GW energy via

$$\mathcal{E}_{\rm GW}(t) = \frac{c^2}{32\pi G}\left(\langle \dot{h}_+^2\rangle + \langle \dot{h}_\times^2\rangle\right), \tag{118}$$

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 $\int_{4\pi} k^2 \mathrm{d}\Omega_{\boldsymbol{k}}$ in $\boldsymbol{k}$ space, defined by

$$S_{\dot{h}}(k) \;=\; \int_{4\pi} \left(|\dot{\tilde{h}}_+|^2 + |\dot{\tilde{h}}_\times|^2\right)\, k^2 \mathrm{d}\Omega_{\boldsymbol{k}}, \tag{119}$$

$$A_{\dot{h}}(k) \;=\; \int_{4\pi} 2\,\mathrm{Im}\left(\dot{\tilde{h}}_+ \dot{\tilde{h}}_\times{}^*\right)\, k^2 \mathrm{d}\Omega_{\boldsymbol{k}}, \tag{120}$$

and normalized such that $\int_0^\infty S_{\dot{h}}(k)\,\mathrm{d}k = \langle \dot{h}_+^2\rangle + \langle \dot{h}_\times^2\rangle$ is proportional to the energy density and $\int_0^\infty A_{\dot{h}}(k)\,\mathrm{d}k$ is proportional to the polarized energy density. The $A_{\dot{h}}(k)$ spectra are not to be confused with the magnetic vector potential $\boldsymbol{A}(\boldsymbol{x},t)$. The corresponding GW energy spectra are noted by

$$E_{\rm GW}(k) \;=\; (c^2/32\pi G)\, S_{\dot{h}}(k), \tag{121}$$

$$H_{\rm GW}(k) \;=\; (c^2/32\pi G)\, A_{\dot{h}}(k). \tag{122}$$

We also define spectra for the metric tensor perturbation,

$$S_h(k) \;=\; \int_{4\pi} \left(|\tilde{h_+}|^2 + |\tilde{h_\times}|^2\right)\, k^2 \mathrm{d}\Omega_{\boldsymbol{k}}, \tag{123}$$

$$A_h(k) \;=\; \int_{4\pi} 2\,\mathrm{Im}\left(\tilde{h_+}\tilde{h_\times}{}^*\right)\, k^2 \mathrm{d}\Omega_{\boldsymbol{k}}, \tag{124}$$

which are normalized such that $\int_0^\infty S_h(k)\,\mathrm{d}k = h_{\rm rms}^2$ is the mean squared metric tensor perturbation.

7 **问题解决方案/经常提到的问题** Troubleshooting / Frequently Asked Questions

7.1 **下载和安装** Download and setup

7.1.1 禁止下载 Download forbidden

A: Both GitHub and SourceForge are banned from countries on the United States Office of Foreign Assets Control sanction list, including Cuba, Iran, Libya, North Korea, Sudan and Syria; see http://de.wikipedia.org/wiki/GitHub and http://en.wikipedia.org/wiki/SourceForge. As a remedy, you might download a tarball from http://pencil-code.nordita.org/; see also Section 2.

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:

   for csh and similar: unsetenv PENCIL_HOME

   for bash and similar: unexport PENCIL_HOME; unset PENCIL_HOME

   (b) switch your shell in verbose mode,

   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:

/bin/csh -v -x $PENCIL_HOME/bin/pc_setupsrc

7.2 **编译** 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 2.

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_setupsrc

If this does not help, say first make clean and then pc_setupsrc.

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).

You can find the ifort compiler at ftp://download.intel.com/software/products/compilers/downloads.

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.

   ```
   limit stacksize 256m
   ulimit -s 256000
   ```

3. Set the environment variable KMP_STACKSIZE to a large value (like 100M)

See also http://softwareforums.intel.com/ids/board/message?board.id=11&message.id=1375

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

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:

```
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.

```
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 t
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':

```
make start.x run.x read_videofiles.x
f90  -fast -O3 -tune ev6 -arch ev6  -c cparam.f90
[...]
f90  -fast -O3 -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 -O3 -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 '5.0 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

```
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 **检查** 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_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. 4.8.1. 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.
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 LANG=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.

```
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 :

```
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 im 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_xyz_parallel_3D: nygrid needs to be an integer multiple...

Q: I just got an:

fft_xyz_parallel_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. 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) [Brms] = Gauss (as output by unit_magnetic, before the run begins)
2) [t] = s (since the default unit system is left as CGS)
3) [urms] = 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 cs0=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 cs0=1 actually means [U]=1 cm/s. Your unit of length is equivalently 1 cm, and therefore the unit of time is [t] = [L]/[U]=1 s. The unit of

density is [rho] = 1 g/cm^3. Since in cgs vA=B/sqrt(4*pi * rho), your unit of magnetic field is [B] = [U] * sqrt([rho] * 4*pi) ~= 3.5 sqrt(g/cm) / s = 3.5 Gauss.

If instead you are assuming SI, you have cs0=1 assuming that means [U]=1 m/s and rho0=1 assuming that to mean [rho]=1 kg/m^3. Using [L]=1 m, you have still [t]=1 s, but now what appears as B=1 in your output is actually [B] = [U] * sqrt (mu * [rho]) = 1 m/s * sqrt(4*pi * 1e-7 N*A-2 1 kg/m^3) ~= 0.0011210 kg/(s^2*A) ~ 11 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 cs0=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 km/ 10 km/s = 10^2 s, i.e., roughly 1.5 minute. For density, assume rho=2x10-4 kg/m^3, typical of the solar photosphere. Your unit of magnetic field is therefore [B] = [U] * sqrt([rho] * 4*pi) = 1e6 cm/s * sqrt(4*pi * 2e-7 g/cm^3) ~ 1585.33 Gauss.

Notice that for mu0=1 and rho0=1 you simply have vA=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 **可视化** 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/
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,WCOOL= 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,WCOOL= 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 **一般问题** 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

```
  epsilon(x) = 1.1920929e-07
  tiny(x)    = 1.1754944e-38
```

(and then there is huge(x) = 3.4028235e+38)
for a single-precision number x.

epsilon(x) is the smallest number that satisfies
1+epsilon(1.) /= 1 ,
while tiny(x) is the smallest number that can be represented without
precision loss.

In the code we have variants hereof,

```
   epsi=5*epsilon(1.0)
   tini=5*tiny(1.0)
   huge1=0.2*huge(1.0)
```

that have added safety margins, so we don't have to think about doing
things like 1/tini.

So in sub.f90,

```
  -      evr = evr / spread(r_mn+epsi,2,3)
```

did (minimally) affect the result for r_mn=O(1), while the correct version

```
  +      evr = evr / spread(r_mn+tini,2,3)
```

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"

```
  ! 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!
```

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

```
logical :: ldensity ! INCLUDE_IN_LPHYSICS
```

could later generate code (in some param_io_extra.inc file) that looks like this:

```
write(unit, *) 'ldensity = ', ldensity
```

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 讨论论坛 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"
```

第二部分　编写 Pencil Code 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 5.

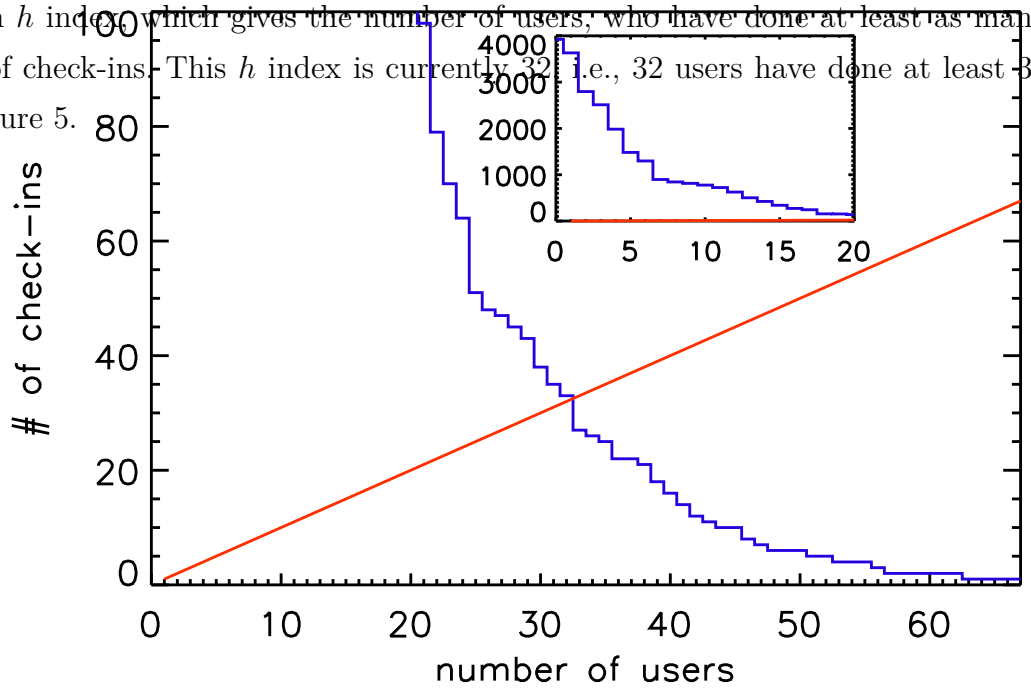


图 5: The h index of Pencil Code check-ins.

The Pencil Code has expanded approximately linearly in the number of lines of code and the number of subroutines (Fig. 6). The increase in the functionality of the code is documented by the rise in the number of sample problems (Fig. 7). It is important to monitor the performance of the code as well. Figure 8 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.

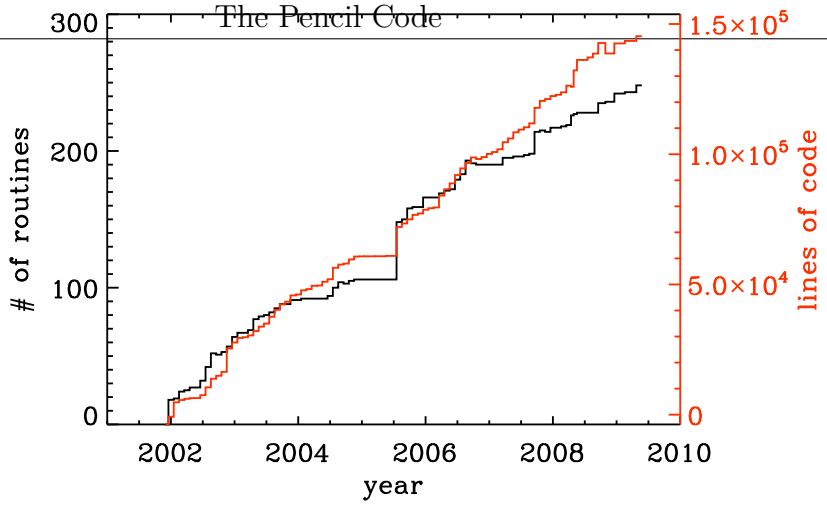


图 6: 自从 2001 年末以来, 代码函数和子程序数目的变化. 2005 年的突增是由于写在另外一个分支 (eos branch) 上的改进合并到了主程序中. 注意代码的增长和时间近似呈现线性增长关系.

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 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. Major code changes are nowadays being discussed by the Pencil Code Steering Committee (https://www.nordita.org/~brandenb/pencil-code/PCSC/). The increase of the revision number with time is depicted

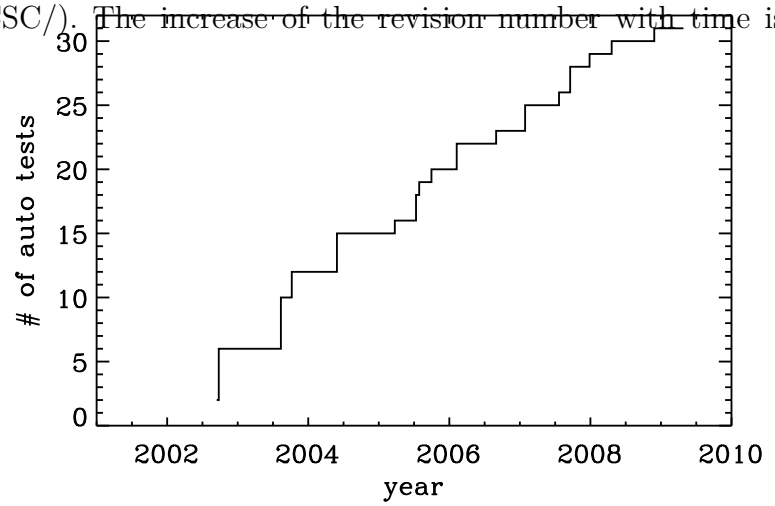


图 7: Number of tests in the sample directory that are used in the nightly auto tests. Note again the approximately linear scaling with time.

in Figure 9. The number of Pencil Code developers increases too (Figure 10), 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.

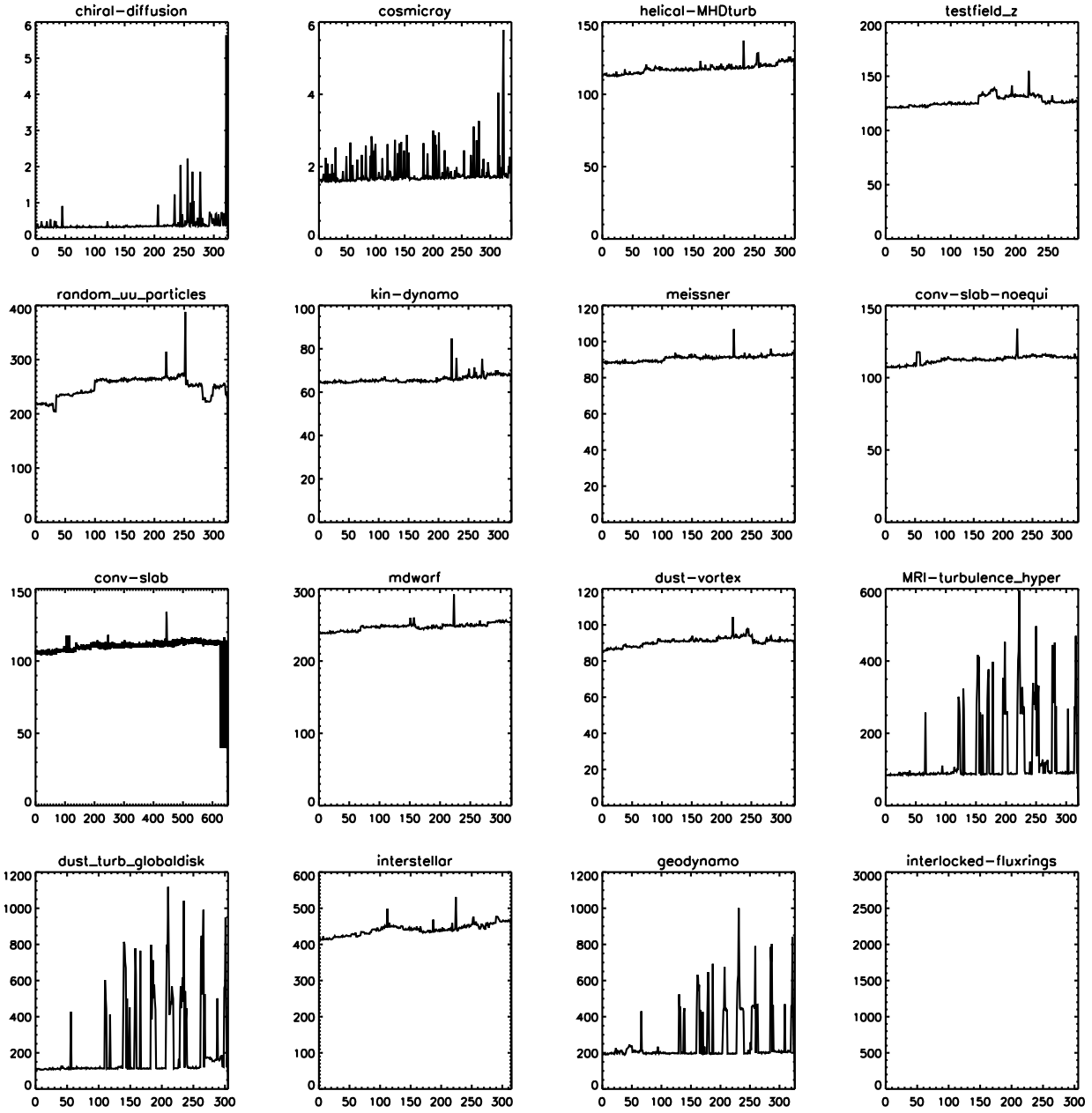


图 8: Run time of the daily auto-tests since August 17, 2008. For most of the runs the run time has not changed much. The occasional spikes are the results of additional load on the machine.

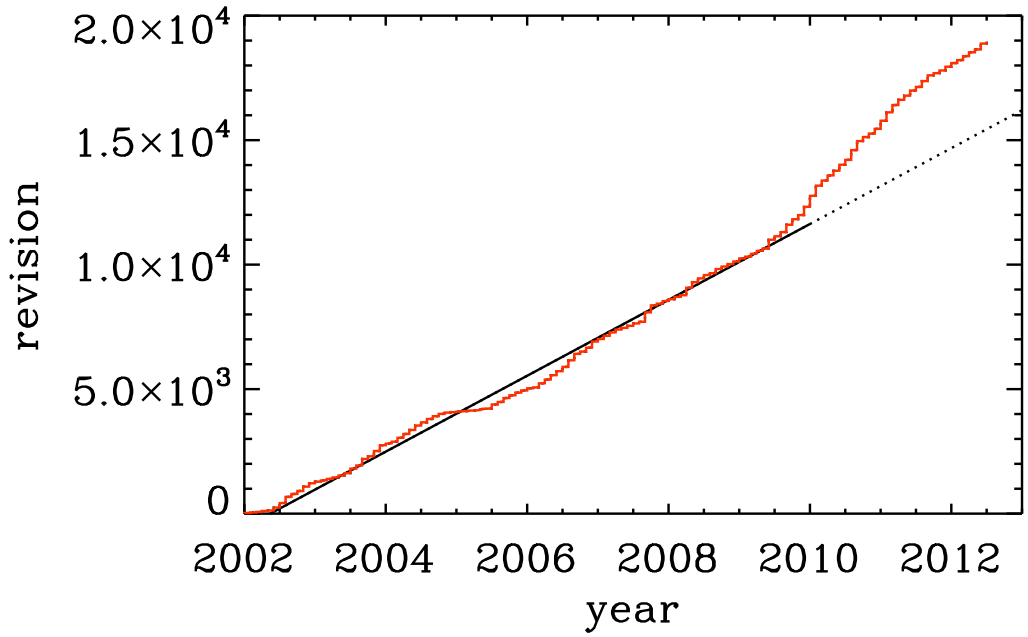


图 9: Number of check-ins since 2002. Note again the linear increase with time, although in the last part of the time series there is a notable speed-up.

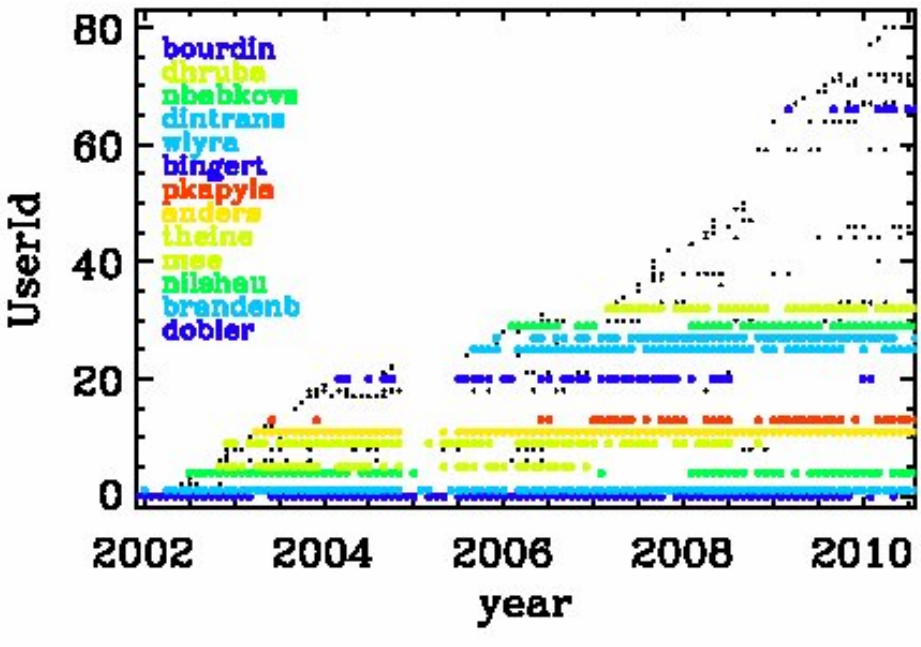


图 10: Check-ins since 2002 per user. Users with more than 100 check-ins are color coded.

8 **理解代码** 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

```
call dlnrho_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,

```
!
!  Continuity equation.
!
      if (lcontinuity_gas) then
        if (ldensity_nolog) then
          df(l1:l2,m,n,irho)   = df(l1:l2,m,n,irho)   - p%ugrho - 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 $\boldsymbol{u} \cdot \nabla \ln \rho$ and divu refers to $\nabla \cdot \boldsymbol{u}$. 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

```
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 $\boldsymbol{u}\cdot\nabla$ 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

```
!
!  Calculate uij and divu, if requested.
!
      if (lpencil(i_uij)) call gij(f,iuu,p%uij,1)
      if (lpencil(i_divu)) call div_mn(p%uij,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 gij routine, which calls the der routine, e.g., like so

```
      k1=k-1
      do i=1,3
        do j=1,3
          if (nder==1) then
            call der(f,k1+i,tmp,j)
```

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

```
!***********************************************************************
    subroutine daa_dt(f,df,p)
!
!  Magnetic field evolution.
!
!  Calculate dA/dt=uxB+3/2 Omega_0 A_y x_dir -eta mu_0 J.
!  For mean field calculations one can also add dA/dt=...+alpha*bb+delta*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
```

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 **改进代码** Adapting the code

9.1 **编程规范** The Pencil Code coding standard

就如同任何超过聊聊几行的代码一样, 代码的书写风格和布局对本程序包非常重要. 布局优秀的代码更加容易阅读和理解, 能够减少犯错的机会.

Pencil Code 已经发展出了统一的书写风格, 并且我们也要求其他贡献者能够尽可能地保持一致, 这对所有人都有好处. 特别的, 如果贡献者对现有的代码进行风格上的统一, 那将是值得赞赏的.

我们没有太多的规则, 简单地遵循已有代码的风格通常来说就已经足够了. 简单来说, 最重要的规则就是:

- 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.

```
if (lcylindrical) 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)

```
if (lcylindrical) &
    call fatal_error('del2fjv','del2fjv not implemented')
[...]
```

- There is always one space separation between 'if' and the criterion following in parenthesis:

```
if (ldensity_nolog) then
  rho=f(l1:l2,m,n,irho)
endif
```

  This is wrong:

```
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. 总而言之, 尽量

模仿程序包内常用的代码风格. For example, in the code fragment above there are no empty spaces within the mathematical expressions programmed in the code. 比如说, 在上面的程序段中, 数学表达式内并没有空格. A unique convention helps in finding certain expressions and patterns in the code. 这样的惯例可以帮助我们找到程序包中特定的表达式和模式. However, empty spaces are often used after commas and semicolons, for examples in name lists. 然而, 空的空格经常在逗号或者分号后面加上, 例如名字列表中.

- 关系运算符通常用符号 (==, / =, <, <=, >, >=), 而不用字符串 (.eq., .ne., .lt., .le., .gt., .ge.).

- In general all comments are placed on their own lines with the '!' appearing in the first column. 所有的注释都放在他们自己的那些行里面,'!' 放在第一列.

- 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. 所有的子程序/函数都开头加上一段标准的注释描述这个玩意是干什么的, 什么时候需要调用, 它的创造者是谁. 任何有意义的修改都需要在后面注释上他们是什么时候被改动的, 以及改动人是谁.

- Lines longer that 78 characters should be explicitly wrapped using the & character, unless there is a block of longer lines that can only be read easily when they are not wrapped. Always add one whitespace before the & character. 如果一行超过了 78 个字符, 那他就要用 & 来换行, 除非不换行的时候更加有助于理解. 记得在 & 前加上空格.

这些事项在 Appendix B 中有着更多的叙述和例子, 特别是在 Sect. B.2 内.

9.2 **增加诊断输出** Adding new output diagnostics

随着新物理规律的加入和新机制的发展, 输出一些新的诊断变量来对其进行观察将会是非常有必要的. 在下文中. 我们将会描述一些设置新的诊断变量所需要的步骤.

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 fmax, and those that need to be summed up over all processors (mostly for calculating mean quantities) in fsum (see subroutine diagnostic in file 'src/equ.f90').

As a sample variable, let us consider jbm (the volume average $\langle \boldsymbol{j} \cdot \boldsymbol{B} \rangle$). Only the module 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 idiag_jbm to the module variables of Magnetic in both 'magnetic.f90' and 'nomagnetic.f90':

   ```
   integer :: idiag_jbm=0
   ```

   The variable idiag_jbm is needed for matching the position of jbm with the list of diagnostic variables specified in 'print.in'.

2. in the subroutine daa_dt in 'magnetic.f90', declare and calculate the quantity jb (the average of which will be jbm), and call sum_mn_name

   ```
   real, dimension (nx) :: jb  ! jj ⋅ 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 rprint_magnetic in both 'magnetic.f90', add the following:

   ```
   !
   !  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?
     [...]
     idiag_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, i_XYZ, where our variable XYZ is stored
!
[...]
write(3,*) 'i_jbm=',idiag_jbm
[...]
```

4. in the subroutine rprint_magnetic in 'nomagnetic.f90', add the following (newer versions of the code may not require this any more):

```
!
!  write column, i_jbm, where our variable jbm is stored
!  idl needs this even if everything is zero
!
[...]
write(3,*) 'i_jbm=',idiag_jbm
[...]
```

5. and don't forget to add your new variable to 'print.in':

```
jbm(f10.5)
```

如果你不想要平均值, 而是想要最大值, 那就需要用 max_mn_name() 去替代 sum_mn_name().

Sect. 5.8.1 讲述了如何输出磁场和速度场的水平平均值. 通过借鉴现有的平均值计算流程 calc_bmz() 或 calc_jmz(), 类似的平均值也可以实现.

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. 'f' 数组是 Pencil Code 里面最大的数组, 它主要的作用是储存当前阶段的时进 (timestepped) 偏微分方程变量. f 数组和稍微比它小一点的类似的 df 数组是本数据包内仅有的三维数组.

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 × my × mz × 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:

```
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,

```
!                       -*-f90-*-     (for Emacs)
!  cparam.local
!
!** AUTOMATIC CPARAM.INC GENERATION ************************************
! Declare (for generation of cparam.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 cparam.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]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

'df' Pencil Code 里面第二大的数据就是 'df' 数组了. By using a 2N storage scheme (see H.4) after Williamson [?] 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 × my × mz × 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. 和 'f' 数组类似的是, 'fp' 数组保存了所有粒子的当前状态. Like the f-array the fp-array also has a time derivative part, the dfp-array. 就好像 f 数组一样, fp 数组也有时间微分的对应数组 dfp. The dimension of the fp-array is $mpar_{l}ocal \times mpvar$ where $mpar_{l}ocal$ is the number of particles in the local processor (for serial runs this is the total number of particles) and $mpvar$ depends on the problem at hand. For example if we are solving for only tracer particles then $mpvar = 3$, for dust particles $mpvar = 6$ 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.

```
!***********************************************************************
    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 resister_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 Pencil **样例** 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 lnrho.

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

- 所有的 pencil 只计算一次
- Pencil 总是由合适的物理模块来计算

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,ilnrho)

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 → requested, df not changed
  The pencil is not requested and is not needed.
- not requested → requested, df changed
  The pencil is not requested, but is needed. The code stops.
- requested → not requested, df not changed
  The pencil is requested, but is not needed. The code gives a warning.
- requested → not requested, df changed
  The pencil is requested and is needed.

9.6.2 添加新的 pencils, 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. Pencils 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 **加入新物理机制**: **特殊模块** 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 straight forward 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 'pencil-code/doc/manual.tex' file.

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. 尽管程序包已经设置了很多的初始条件, 但尽量不要再让它的数量增长了. We aim to eventually removed most of them. 我们希望最后能够将大部分初始条件移除. The recommended course of action is to make use of the InitialCondition module. 推荐的做法是利用好 InitialCondition 模块.

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 variables. 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 **测试代码** 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 (auto-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 \
   --use-pc_auto-test \
   -N15 -Uc -rs \
   -T $HOME/public_html/pencil-code/tests/timings.txt \
   -t 15m
   -m <email1@inter.net,email2@inter.net,...> \
   <work-host.inter.net> \
   -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.

> $HOME/public_html/pencil-code/tests/nightly-tests.html: Write output to the given file.

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 **全局变量** Global variables

The following variables are defined in 'cdata.f90' and are available in any routine that uses the module Cdata. 以下的变量都定义在 'cdata.f90' 内, 并且可以在所有的调用了 use Cdata模块的程序中都可以使用.

Variable	Meaning
	real
t	simulated time t.
	integer
n[xyz]grid	global number of grid points (excluding ghost cells) in x, y and z direction.
nx, ny, nz	number of grid points (excluding ghost cells) as seen by the current processor, i. e. ny=nygrid/nprocy, etc.
mx, my, mz	number of grid points seen by the current processor, but including ghost cells. Thus, the total box for the ivarth variable (on the given processor) is given by f(1:mx,1:my,1:mz,ivar).
l1, l2	smallest and largest x-index for the physical domain (i. e. excluding ghost cells) on the given processor.
m1, m2	smallest and largest y-index for physical domain.
n1, n2	smallest and largest z-index for physical domain, i. e. the physical part of the ivarth variable is given by f(l1:l2,m1:m2,n1:n2,ivar)
m, n	pencil indexing variables: During each time-substep the box is traversed in x-pencils of length mx such that the current pencil of the ivarth variable is f(l1:l2,m,n,ivar).
	logical
lroot	true only for MPI root processor.
lfirst	true only during first time-substep of each time step.
headt	true only for very first full time step (comprising 3 substeps for the 3rd-order Runge–Kutta scheme) on root processor.
headtt	= (lfirst .and. lroot): true only during very first time-substep on root processor.

lfirstpoint	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) .
lout	true when diagnostic output is about to be written.

11.2 **子程序和函数** Subroutines and functions

output(file,a,nv) (module IO): Write (in each 'proc$N$' directory) the content of the global array a to a file called file, where a has dimensions mx×my×mz×nv, or mx×my×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 nontrivial 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×my×mz×3 (global arrays), or of size nx×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×my×mz×3 (a and b) and mx×my×mz (c), or of size nx×3 (a and b) and nx (c).

dot2(a,c) (module Sub): Same as dot(a,a,c).

第三部分　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.[16]

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> 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.

proc	machine	$\frac{\mu s}{\text{pt step}}$	resol.	what	mem/proc	when	who
1	Nl3	19	64^3	kinematic	10 MB	20-may-02	AB
1	Nl3	30	64^3	magn/noentro	20 MB	20-may-02	AB
1	Nq1	10	64^3	magn/noentro		30-may-02	AB
1	Ukaff	9.2	64^3	magn/noentro		20-may-02	AB
1	Nl6	6.8	64^3	magn/noentro		10-mar-03	AB
1	Nl6	36.3	64×128×64	nomag/entro/dust		19-sep-03	AB
1	Nl6	42.7	16^2×256	nomag/entro/rad6/ion		22-oct-03	AB
1	Nl6	37.6	16^2×256	nomag/entro/rad2/ion		22-oct-03	AB
1	Nl6	19.6	16^2×256	nomag/entro/ion		22-oct-03	AB
1	Nl6	8.7	16^2×256	nomag/entro		22-oct-03	AB
1	Nl6n	9.8	32^3	magn/noentro/pscalar		17-mar-06	AB
1	Mhd	7.8	64^3	magn/noentro		20-may-02	AB
1	Nq4	14.4	128^3	magn/noentro		8-oct-02	AB
1	Nq5	6.7	128^3	magn/noentro		8-oct-02	AB
1	fe1	5.1	128^3	magn/noentro		9-oct-02	AB
1	Kabul	4.4	128^3	magn/noentro	130 MB	20-jun-02	WD

[16] 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.

1	Hwwsx5	3.4	256^3	convstar	7.8 GB	29-jan-03	WD
1	Mac/g95	7.7	32^3	magn/noentro		14-jan-07	BD
1	Mac/ifc	4.5	32^3	magn/noentro		14-jan-07	BD
2	Kabul	2.5	128^3	magn/noentro	80 MB	20-jun-02	WD
2	Nq3+4	7.4	128^3	magn/noentro		8-oct-02	AB
2	Nq4+4	8.9	128^3	magn/noentro		8-oct-02	AB
2	Nq4+5	7.3	128^3	magn/noentro		8-oct-02	AB
2	Nq5+5	3.7	128^3	magn/noentro		8-oct-02	AB
2	fe1	3.45	128^3	magn/noentro		9-oct-02	AB
2	Nq2	9.3	64^3	magn/noentro		11-sep-02	AB
2	Nq1+2	8.3	64^3	magn/noentro		11-sep-02	AB
2	Hwwsx5	1.8	256^3	convstar	7.9 GB	29-jan-03	WD
4	Nq1+2	5.4	64^3	magn/noentro		11-sep-02	AB
4	Nq1235	4.1	128^3	magn/noentro		11-sep-02	AB
4	Nq0-3	6.8	256^3	magn/noentro	294 MB	10-jun-02	AB
4	Mhd	2.76	64^3	magn/noentro		30-may-02	AB
4	fe1	3.39	32^3	magn/noentro		16-aug-02	AB
4	Rasm.	2.02	64^3	magn/noentro	2x2	8-sep-02	AB
4	Mhd	8.2	$64^2\times16$	nomag/entro		23-jul-02	AB
4	fe1	6.35	$64\times128\times64$	nomag/entro/dust		19-sep-03	AB
4	fe1	2.09	128^3	magn/noentro		9-oct-02	AB
4	fe1	1.45	128^3	magn/noentro	giga	9-oct-02	AB
4	fe1	7.55	$16^2\times512$	nomag/entro/rad2/ion	4x1	1-nov-03	AB
4	fe1	5.48	$16^2\times512$	nomag/entro/rad2/ion	1x4	1-nov-03	AB
4	Luci	1.77	64^3	magn/noentro		27-feb-07	AB
4	Lenn	0.65	64^3	nomag/noentro		13-jan-07	AB
4	Lenn	1.21	64^3	magn/noentro		7-nov-06	AB
4	Kabul	1.5	128^3	magn/noentro	47 MB	20-jun-02	WD
4	Hwwsx5	1.8	256^3	convstar	8.2 GB	29-jan-03	WD
8	Nqall	3.0	128^3	magn/noentro		8-oct-02	AB
8	fe1	3.15	64^3	magn/noentro	1x8	8-sep-02	AB
8	fe1	2.36	64^3	magn/noentro	2x4	8-sep-02	AB
8	Ukaff	1.24	64^3	magn/noentro		20-may-02	AB
8	Kabul	1.25	$64^2\times128$	nomag/entro		11-jul-02	WD
8	fe1	1.68	128^3	magn/noentro	1x8	8-sep-02	AB
8	fe1	1.50	128^3	magn/noentro	2x4	8-sep-02	AB
8	fe1	1.44	128^3	magn/noentro	4x2	8-sep-02	AB
8	Kabul	0.83	128^3	magn/noentro	28 MB	20-jun-02	WD
8	Gridur	1.46	128^3	magn/noentro		19-aug-02	NE
8	Kabul	0.87	256^3	magn/noentro	160 MB	20-jun-02	WD
8	fe1	0.99	256^3	magn/noentro	2x4	8-sep-02	AB
8	fe1	0.98	256^3	magn/noentro	4x2	8-sep-02	AB
8	cetus	0.58	64^3	magn/noentro	4x2	19-aug-07	SS
8	cetus	0.73	256^3	magn/noentro	4x2,156M	19-aug-07	SS
8	Neolith	0.82	64^3	magn/noentro	4x2	5-dec-07	AB
8	Mhd	1.46	$160^2\times40$	nomag/entro	46 MB	7-oct-02	AB
8	Hwwsx5	0.50	256^3	convstar	8.6 GB	29-jan-03	WD

8	Neolith	0.444	128^3	magn/noentro		6-dec-07	AB
8	Ferlin	0.450	64^3	1test/noentro		21-jun-09	AB
8	Ferlin	0.269	64^3	magn/noentro		2-apr-10	AB
8	Ferlin	0.245	128^3	magn/noentro		2-feb-11	AB
8	nor52	2.00	32^3	magn/noentro		2-dec-09	AB
9	hydra(2)	0.317	72^3	magn/noentro	1x3x3	8-may-16	AB
9	charybdis	0.169	72^3	magn/noentro	1x3x3	8-may-16	AB
9	scylla	0.150	72^3	magn/noentro	1x3x3	8-may-16	AB
12	scylla	0.151	72^3	magn/noentro	1x4x3	8-may-16	AB
12	janus	6.02	$72^2 \times 22$	coag/noentro		17-dec-15	AB
16	fe1	1.77	64^3	convstar		9-feb-03	AB
16	copson	0.596	128^3	geodynamo/ks95		21-nov-03	DM
16	fe1	0.94	128^3	magn/noentro	4x4	8-sep-02	AB
16	fe1	0.75	128^3	magn/noentro	4x4/ifc6	9-may-03	AB
16	workq	0.88	128^3	magn/noentro	4x4/ifc6	21-aug-04	AB
16	giga	0.76	128^3	magn/noentro	4x4/ifc6	21-aug-04	AB
16	giga2	0.39	128^3	magn/noentro	4x4/ifc6	20-aug-04	AB
16	giga	0.47	128^3	chiral	4x4/ifc6	29-may-04	AB
16	giga	0.43	128^3	nomag/noentro	4x4/ifc6	28-apr-03	AB
16	Mhd	2.03	128^3	magn/noentro		26-nov-02	AB
16	Mhd	0.64	256^3	magn/noentro	60 MB	22-may-02	AB
16	fe1	0.56	256^3	magn/noentro	4x4	16-aug-02	AB
16	fe1	6.30	128×256×128	nomag/entro/dust		19-sep-03	AB
16	fe1	1.31	128^2×512	nomag/entro/rad2/ion	4x4	1-nov-03	AB
16	Ukaff	0.61	128^3	magn/noentro		22-may-02	AB
16	Ukaff	0.64	256^3	magn/noentro		20-may-02	AB
16	Kabul	0.80	128^3	magn/noentro	16 MB	20-jun-02	WD
16	Kabul	0.51	256^3	magn/noentro	9 MB	20-jun-02	WD
16	Gridur	0.81	128^3	magn/noentro		19-aug-02	NE
16	Gridur	0.66	256^3	magn/noentro		19-aug-02	NE
16	Sander	0.53	256^3	magn/noentro		8-sep-02	AB
16	Luci	0.375	128^3	magn/noentro		28-oct-06	AB
16	Lenn	0.284	128^3	magn/noentro		8-nov-06	AB
16	Neolith	0.180	256^3	magn/noentro		6-dec-07	AB
16	Triolith	0.075	128^3	magn/noentro	2x2x4	1-mar-14	AB
16	Triolith	0.065	128^3	magn/noentro	1x4x4	1-mar-14	AB
16	Triolith	0.054	256^3	magn/noentro	1x4x4	1-mar-14	AB
16	Coma	0.603	128^3	GW/magn/noentro	1x4x4	27-jul-17	SM
24	Gardar	0.44	$128^2 \times 48$	magn/noentro		6-nov-13	AB
24	Summit	0.041	144^3	magn/noentro		28-jul-17	AB
32	giga?	0.32	256^3	magn/noentro		13-sep-03	AB
32	Ukaff	0.34	256^3	magn/noentro		20-may-02	AB
32	Ukaff	0.32	512^3	magn/noentro		20-may-02	AB
32	Hermit	0.200	256×512×256	spherical conv/magn	1x8x4	22-aug-13	PJK
32	fe1	0.168	512^3	nomag/noentro		9-oct-02	AB
32	fe1	1.26	64^2×256	nomag/entro/rad/ion		7-sep-03	AB
32	Luci	0.182	256^3	magn/noentro		26-feb-07	AB

32	Lenn	0.147	256^3	nomag/entro/cool/fo	4x8	8-nov-06	AB
32	Steno	0.076	256^3	nomag/entro/cool/fo	4x8	20-jun-06	AB
32	Steno	0.081	256^3	nomag/entro/cool	4x8	20-jun-06	AB
32	Steno	0.085	256^3	nomag/entro/cool/sh	4x8	20-jun-06	AB
32	Steno	0.235	$512^2 \times 256$	mag/entro	4x8	9-jul-06	AB
32	Sanss	0.273	128×256^2	nomag	4x8	3-jul-07	AB
32	Neolith	0.275	128^3	testfield4		24-oct-08	AB
32	Ferlin	0.556	128^3	testscalar		7-jan-09	AB
36	Kraken	0.177	$192 \times 384 \times 64$	magn/noentro	3x6x2	12-jan-12	WL
36	scylla	0.096	72^3	magn/noentro	1x6x6	8-may-16	AB
48	janus	0.028	$72^2 * 216$	magn/noentro	4x12	28-mar-16	AB
64	Coma	0.573	128^3	GW/magn/noentro	1x8x8	7-aug-17	SM
64	fe1	0.24	256^3	magn/noentro	8x8	2-sep-02	AB
64	giga	0.11	256^3	nomag/noentro	4x16	29-apr-03	AB
64	giga	0.23	256^3	nomag/noentro/hyp	4x16	8-dec-03	AB
64	fe1	0.164	512^3	nomag/noentro/hyp	4x16	17-dec-03	AB
64	giga	0.091	512^3	nomag/noentro/hyp	4x16	17-dec-03	AB
64	giga	0.150	256^3	magn/noentro	4x16	1-jul-03	AB
64	giga	0.166	512^3	magn/noentro	64*173MB	10-jul-03	AB
64	Gridur	0.25	256^3	magn/noentro		19-aug-02	NE
64	Ukaff	0.17	512^3	magn/noentro		21-may-02	AB
64	Steno	0.075	512^3	magn/noentro	8x16	19-oct-06	AB
64	Neolith	0.0695	256^3	magn/noentro		6-dec-07	AB
64	Ferlin	8.51	150×128^2	Li mechanism	8x8	21-jun-09	AB
64	Ferlin	0.156	256^3	magn/noentro	8x8	14-jun-09	AB
64	Akka	0.038	$256^2 \times 512$	magn/noentro	8x8	27-dec-12	AB
64	Triolith	0.0146	256^3	magn/noentro	1x8x8	1-mar-14	AB
64	Triolith	0.0164	256^3	magn/noentro	2x4x8	1-mar-14	AB
64	Hermit	0.101	$256 \times 512 \times 256$	spherical conv/magn	1x8x8	22-aug-13	PJK
64	Sisu	0.00205	$256 \times 512 \times 256$	spherical conv/magn	1x8x8	22-aug-13	PJK
72	Kraken	0.093	$192 \times 384 \times 64$	magn/noentro	3x12x2	12-jan-12	WL
72	Kraken	0.151	$96 \times 192 \times 16$	magn/noentro	6x12	17-jan-12	WL
72	Kraken	0.091	$192 \times 384 \times 32$	magn/noentro	6x12	17-jan-12	WL
72	Kraken	0.071	$384 \times 768 \times 64$	magn/noentro	6x12	17-jan-12	WL
72	Summit	0.0128	576^3	magn/noentro		7-aug-17	AB
128	fe1	0.44	256^3	nomag/entro/rad8/ion	4x32	10-mar-04	TH
128	fe1	2.8	512^3	magn/noentro	16x8	5-sep-02	AB
128	fe1	0.51	512^3	magn/noentro	8x16	5-sep-02	AB
128	fe1	0.27	512^3	magn/noentro	4x32	5-sep-02	AB
128	fe1	0.108	512^3	magn/noentro	4x32/ifc6	5-jan-02	AB
64+64	giga2	0.0600	512^3	magn/noentro	4x32/ifc6	21-aug-04	AB
128l	giga2	0.0605	512^3	magn/noentro	4x32/ifc6	21-aug-04	AB
128	fe1	0.35	512^3	magn/noentro	2x64	9-sep-02	AB
128	fe1	0.094	786^3	magn/noentro	4x32/ifc6	9-sep-02	AB
128	Hermit	0.0532	$256 \times 512 \times 256$	spherical conv/magn	1x16x8	22-aug-13	PJK
128	Hermit	0.0493	$256 \times 512 \times 256$	spherical conv/magn	2x8x8	22-aug-13	PJK
128	Sisu	0.00108	$256 \times 512 \times 256$	spherical conv/magn	1x16x8	22-aug-13	PJK

144	Kraken	0.080	96×192×32	magn/noentro	6x12x2	13-jan-12	WL
144	Kraken	0.058	192×384×64	magn/noentro	6x12x2	17-jan-12	WL
144	Kraken	0.044	384×768×128	magn/noentro	6x12x2	18-jan-12	WL
144	Gardar	2.19	288×1×288	coag43	8x1x18	13-sep-15	AB
144	Summit	0.0064	576^3	magn/noentro		7-aug-17	AB
192	Janus	0.0123	144×288×72	magn/noentro/sph	1x24x32	24-jul-16	AB
256	Hermit	0.0328	512×1024×512	spherical conv/magn	1x16x16	22-aug-13	PJK
256	Hermit	0.0285	256×512×256	spherical conv/magn	1x16x16	22-aug-13	PJK
256	giga2	0.028	1024^3	magn/noentro	4x64/ifc6	20-aug-04	AB
256	Hermit	0.0262	256×512×256	spherical conv/magn	2x16x8	22-aug-13	PJK
256	Hermit	0.0254	512×1024×512	spherical conv/magn	2x16x8	22-aug-13	PJK
256	Hermit	0.0226	512×1024×512	spherical conv/magn	4x8x8	22-aug-13	PJK
256	Akka	0.0113	512^3	magn/noentro	16x16	12-jun-11	AB
256	Sisu	0.00618	256×512×256	spherical conv/magn	1x16x16	22-aug-13	PJK
256	Sisu	0.00500	512×1024×512	spherical conv/magn	1x16x16	22-aug-13	PJK
256	Triolith	0.030	$256^2 \times 512$	magn/rad	1x16x16	17-mar-14	AB
256	Triolith	0.0049	256^3	magn/noentro	1x16x16	1-mar-14	AB
256	Beskow	3.36	1×1×1024	coag43	1x1x256	3-mar-15	AB
288	Gardar	0.042	576^2×288	magn/rad	1x18x16	17-mar-14	AB
288	Kraken	0.0432	192×384×64	magn/noentro	6x12x4	12-jan-12	WL
288	Kraken	0.0447	96×192×64	magn/noentro	6x12x4	13-jan-12	WL
288	Kraken	0.0201	384×768×256	magn/noentro	6x12x4	18-jan-12	WL
288	Janus	0.0360	288^3	magn/entro/rad	1x16x18	22-feb-16	AB
288	Summit	0.0033	576^3	magn/noentro	1x16x18	7-aug-17	AB
512	Hermit	0.01717	512×1024×512	spherical conv/magn	1x32x16	22-aug-13	PJK
512	Hermit	0.0166	256×512×256	spherical conv/magn	1x32x16	22-aug-13	PJK
512	Hermit	0.0142	256×512×256	spherical conv/magn	2x16x16	22-aug-13	PJK
512	Hermit	0.01340	512×1024×512	spherical conv/magn	2x16x16	22-aug-13	PJK
512	Hermit	0.01189	512×1024×512	spherical conv/magn	8x8x8	22-aug-13	PJK
512	Hermit	0.01165	512×1024×512	spherical conv/magn	4x16x8	22-aug-13	PJK
512	Akka	0.0081	512^3	magn/noentro	16x32	10-sep-11	AB
512	Neolith	0.0073	256^3	magn/noentro		20-nov-09	AB
512	Gardar	0.0035	512^3	magn/noentro		14-jan-13	AB
512	Lindgren	0.0040	512^2×1024	magn/noentro	16x32	8-jul-12	AB
512	Sisu	0.00446	256×512×256	spherical conv/magn	4x16x8	22-aug-13	PJK
512	Sisu	0.00435	1024×2048×1024	spherical conv/magn		22-aug-13	PJK
512	Sisu	0.00268	512×1024×512	spherical conv/magn	1x32x16	22-aug-13	PJK
576	Kraken	0.0257	192×384×64	magn/noentro	6x24x4	12-jan-12	WL
576	Kraken	0.0317	192^2×64	magn/noentro	12^2x4	13-jan-12	WL
576	Kraken	0.0116	768^2×256	magn/noentro	12^2x4	18-jan-12	WL
576	Summit	0.00183	576^3	magn/noentro	1x24x48	29-jul-17	AB
576	Beskow	0.00174	576^3	magn/noentro	1x24x48	23-may-16	AB
768	Lindgren	0.0049	256×1152^2	magn/noentro/sph	1x24x32	17-oct-14	SJ
1024	Hermit	0.00943	512×1024×512	spherical conv/magn	1x32x32	22-aug-13	PJK
1024	Hermit	0.00707	512×1024×512	spherical conv/magn	2x32x16	22-aug-13	PJK
1024	Hermit	0.00698	1024×2048×1024	spherical conv/magn	4x16x16	22-aug-13	PJK
1024	Hermit	0.00630	512×1024×512	spherical conv/magn	4x16x16	22-aug-13	PJK

1024	Triolith	0.00236	256^3	magn/noentro	4x16x16	1-mar-14	AB
1024	Triolith	0.00126	512^3	magn/noentro	2x16x32	1-mar-14	AB
1024	Triolith	0.00129	512^3	magn/noentro	4x16x16	1-mar-14	AB
1024	Sisu	0.00225	1024×2048×1024	spherical conv/magn		22-aug-13	PJK
1024	Sisu	0.00148	512×1024×512	spherical conv/magn	2x32x16	22-aug-13	PJK
1152	Kraken	0.0212	192×384×64	magn/noentro	12x24x4	13-jan-12	WL
1152	Kraken	0.00856	384×768×128	magn/noentro	12x24x4	17-jan-12	WL
1152	Kraken	0.00549	768×1536×256	magn/noentro	12x24x4	17-jan-12	WL
1152	Lindgren	0.016	512^2×512	magn/rad	1x36x32	17-mar-14	AB
1152	Lindgren	0.0066	1152^3	magn/noentro	1x32x36	25-nov-14	AB
1152	Beskow	0.0055	1152^3	magn/noentro/GW	1x32x36	27-aug-17	AB
1152	Beskow	0.0024	1152^3	magn/noentro	1x32x36	20-jan-15	AB
1152	Beskow	0.00098	1152^3	magn/noentro	1x32x36	18-jan-16	AB-gnu
1152	Beskow	0.00090	1152^3	magn/noentro	1x32x36	30-mar-17	AB
1152	Beskow	0.0060	1152^3	magn/noentro/GW	1x32x36	31-mar-18	AB
1152	Beskow	0.0063	576^3	magn/entro/rad	1x32x36	17-feb-18	AB
1536	Lindgren	0.00171	512^2×384	magn/noentro	2x32x24	15-jul-13	AB
2048	Hermit	0.00451	1024×2048×1024	spherical conv/magn	2x32x32	22-aug-13	PJK
2048	Hermit	0.00380	512×1024×512	spherical conv/magn	8x16x16	22-aug-13	PJK
2048	Hermit	0.00355	512×1024×512	spherical conv/magn	4x32x16	22-aug-13	PJK
2048	Hermit	0.00350	1024×2048×1024	spherical conv/magn	4x32x16	22-aug-13	PJK
2048	Lindgren	0.00129	512^2×1024	magn/noentro	32x64	20-apr-13	AB
2048	Lindgren	0.00129	1024^2×2048	magn/noentro	32x64	31-jul-12	AB
2048	Triolith	9.3×10^{-4}	512^3	magn/noentro	4x16x32	1-mar-14	AB
2048	Sisu	0.00120	1024×2048×1024	spherical conv/magn		22-aug-13	PJK
2048	Sisu	9.2×10^{-4}	512×1024×512	spherical conv/magn	4x32x16	22-aug-13	PJK
2304	Triolith	1.07×10^{-3}	576^3	magn/noentro	4x18x32	1-mar-14	AB
2304	Kraken	0.02267	192×384×64	magn/noentro	12x24x8	13-jan-12	WL
2304	Kraken	0.01233	192×768×64	magn/noentro	12x48x4	13-jan-12	WL
2304	Kraken	0.00300	768×3072×256	magn/noentro	12x48x4	18-jan-12	WL
4096	Hermit	0.00193	1024×2048×1024	spherical conv/magn	4x32x32	22-aug-13	PJK
4096	Triolith	3.6×10^{-4}	1024^3	magn/noentro	4x32x32	1-mar-14	AB
4096	Triolith	3.8×10^{-4}	1024^3	magn/noentro	8x16x32	1-mar-14	AB
4096	Triolith	4.2×10^{-4}	1024^3	magn/noentro	4x16x64	1-mar-14	AB
4096	Lindgren	4.6×10^{-4}	2048^3	magn/noentro	4x16x64	26-mar-13	AB
4096	Sisu	6.7×10^{-4}	1024×2048×1024	spherical conv/magn		22-aug-13	PJK
4608	Triolith	7.4×10^{-4}	576^3	magn/noentro	8x18x32	1-mar-14	AB
4608	Triolith	2.7×10^{-4}	1152^3	magn/noentro	4x32x36	1-mar-14	AB
4608	Triolith	3.0×10^{-4}	1152^3	magn/noentro	4x36x32	1-mar-14	AB
4608	Triolith	3.7×10^{-4}	1152^3	magn/noentro	4x18x64	1-mar-14	AB
4608	Triolith	2.36×10^{-4}	2304^3	magn/noentro	2x32x72	1-mar-14	AB
4608	Kraken	0.00764	192×768×128	magn/noentro	12x48x8	13-jan-12	WL
4608	Kraken	0.00144	768×3072×512	magn/noentro	12x48x8	18-jan-12	WL
6144	Lindgren	4.2×10^{-4}	$1024^3 \times 1536$	magn/noentro	4x16x64	21-oct-13	AB
6144	Lindgren	8.9×10^{-4}	256^2	magn/noentro/sph	2x48x64	6-jan-15	SJ
8192	Hermit	0.00101	1024×2048×1024	spherical conv/magn	8x32x32	22-aug-13	PJK
8192	Sisu	4.1×10^{-4}	1024×2048×1024	spherical conv/magn		22-aug-13	PJK

8192	Triolith	1.48×10^{-4}	2048^3	magn/noentro	4x32x64	1-mar-14	AB
9216	Kraken	0.00485	192×768×256	magn/noentro	24x48x8	13-jan-12	WL
9216	Kraken	0.00158	768×1536×256	magn/noentro	24x48x8	17-jan-12	WL
9216	Kraken	8.0×10^{-4}	1536×3072×512	magn/noentro	24x48x8	18-jan-12	WL
9216	Lindgren	2.36×10^{-4}	2304^3	magn/noentro	4x48x48	15-feb-14	AB
9216	Triolith	1.04×10^{-3}	576^3	magn/noentro	16x18x32	1-mar-14	AB
9216	Triolith	1.28×10^{-4}	2304^3	magn/noentro	4x36x64	1-mar-14	AB
9216	Triolith	1.30×10^{-4}	2304^3	magn/noentro	4x32x72	1-mar-14	AB
16384	Hermit	6.4×10^{-4}	1024×2048×1024	spherical conv/magn	16x32x32	22-aug-13	PJK
18432	Kraken	0.00316	384×768×256	magn/noentro	24x48x16	13-jan-12	WL
18432	Kraken	8.8×10^{-4}	768×1536×512	magn/noentro	24x48x16	17-jan-12	WL
18432	Kraken	4.0×10^{-4}	1536×3072×1024	magn/noentro	24x48x16	18-jan-12	WL
36864	Kraken	0.0020	384×768×512	magn/noentro	48^2x16	14-jan-12	WL
36864	Kraken	4.9×10^{-4}	1536^2×512	magn/noentro	48^2x16	17-jan-12	WL
36864	Kraken	2.2×10^{-4}	1536×3072×2048	magn/noentro	24x48x32	18-jan-12	WL
73728	Kraken	0.00121	768^2×512	magn/noentro	48^2x32	19-jan-12	WL
73728	Kraken	2.9×10^{-4}	1536^2×1024	magn/noentro	48^2x32	26-jan-12	WL
73728	Kraken	1.2×10^{-4}	3072^2×2048	magn/noentro	48^2x32	26-jan-12	WL

The machines we have used can be characterized as follows:

Nl3: 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 Lenngren) 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 ??.

A.1 Test case

In the following test samples, we run isothermal magnetohydrodynamics in a periodic domain[17]. 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 should not be so large that

[17]Run directories are available on http://norlx51.nordita.org/~brandenb/pencil-code/timings/bforced/

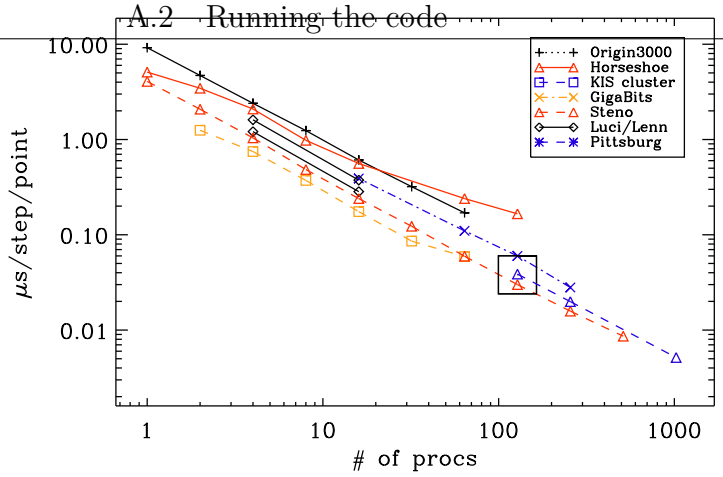


图 11: Scaling results on three different machines. The thin straight line denotes perfectly linear scaling.

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., http://norlx51.nordita.org/~brandenb/pencil-code/timings/bforced/512_4x16x32. The relevant file to be changed is src/cparam.local

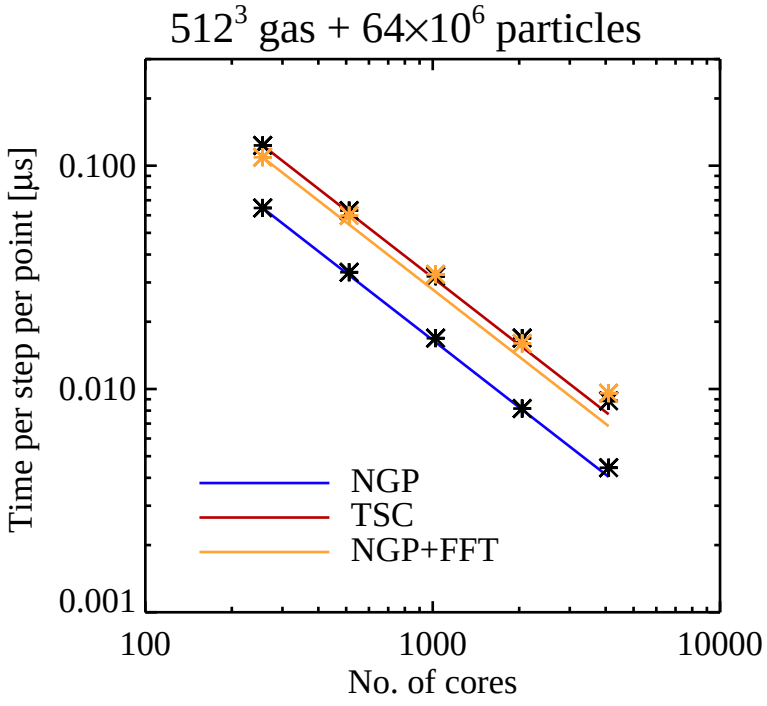


图 12: Scaling results of particle-mesh problem on Blue Gene/P on up to 4096 cores. The different lines denote different particle-mesh schemes (NGP=Nearest Grid Point, TSC=Triangular Shaped Cloud) and whether self-gravity is included (FFT).

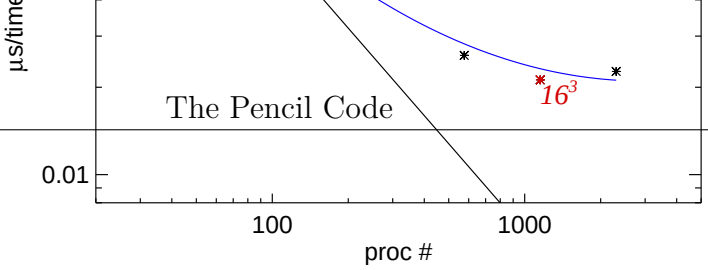
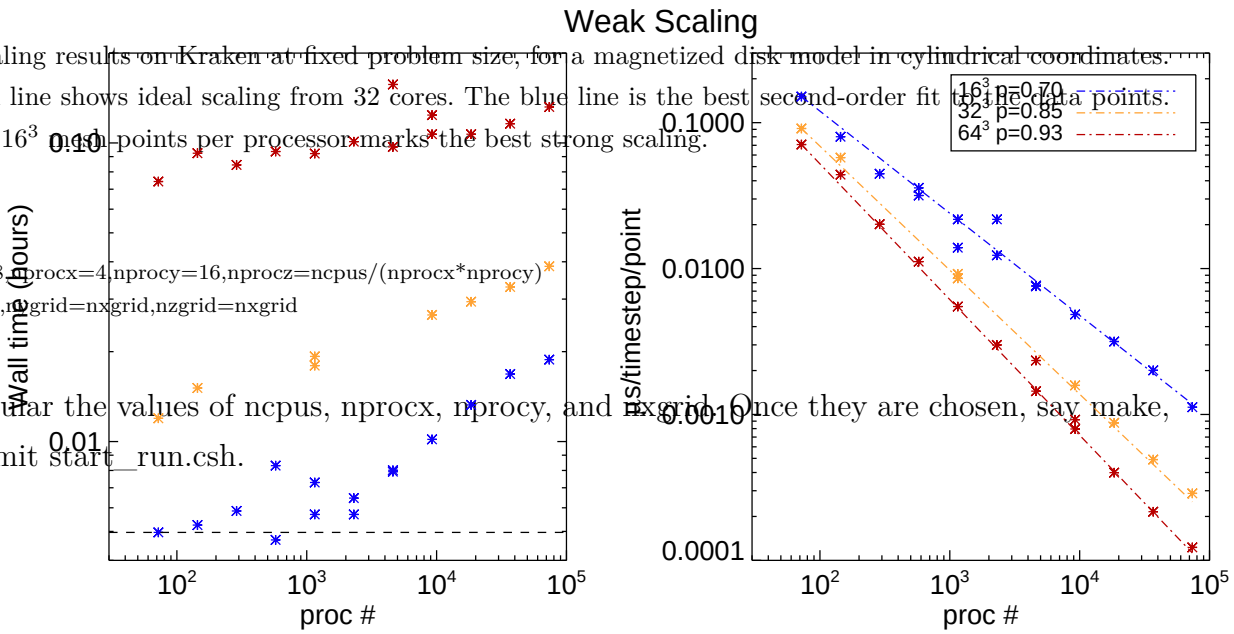


图 13: Scaling results on Kraken at fixed problem size, for a magnetized disk model in cylindrical coordinates. The black line shows ideal scaling from 32 cores. The blue line is the best second-order fit to the data points. A load of 16^3 mesh points per processor marks the best strong scaling.



ncpus=2048,nprocx=4,nprocy=16,nprocz=ncpus/(nprocx*nprocy)
nxgrid=512,nygrid=nxgrid,nzgrid=nxgrid

in particular the values of ncpus, nprocx, nprocy, and nxgrid. Once they are chosen, say make, and submit start_run.csh.

图 14: Scaling results on Kraken at fixed load per processor, for a magnetized disk model in cylindrical coordinates. The figure shows, after determining that 16^3 is the best load per processor for strong scaling, how far one can push with weak scaling. The scaling index is found to be 0.7 for 16^3 and 0.93 for 64^3, up to 73 728 processors.

表 8: Like previous table, but for the versions from the 'samples' directory.

proc(s)	machine	$\frac{\mu s}{\text{pt step}}$	resol.	mem./proc	when	who
			conv-slab			
1	Mhd	6.45	32^3	4 MB	23-jul-02	wd
1	Cincinnatus	4.82	32^3	3 MB	23-jul-02	wd
1	Cincinnatus	11.6	64^3	14 MB	23-jul-02	wd
1	Cincinnatus	20.8	128^3	93 MB	23-jul-02	wd
1	Kabul	3.91	32^3		23-jul-02	wd
1	Kabul	3.88	64^3		23-jul-02	wd
1	Kabul	4.16	128^3	93 MB	23-jul-02	wd
			conv-slab-flat			
1	Kabul	3.02	$128^2\times32$	29 MB	23-jul-02	wd
2	Kabul	1.81	$128^2\times32$	18 MB	23-jul-02	wd
4	Kabul	1.03	$128^2\times32$	11 MB	23-jul-02	wd
8	Kabul	0.87	$128^2\times32$	9 MB	23-jul-02	wd

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.

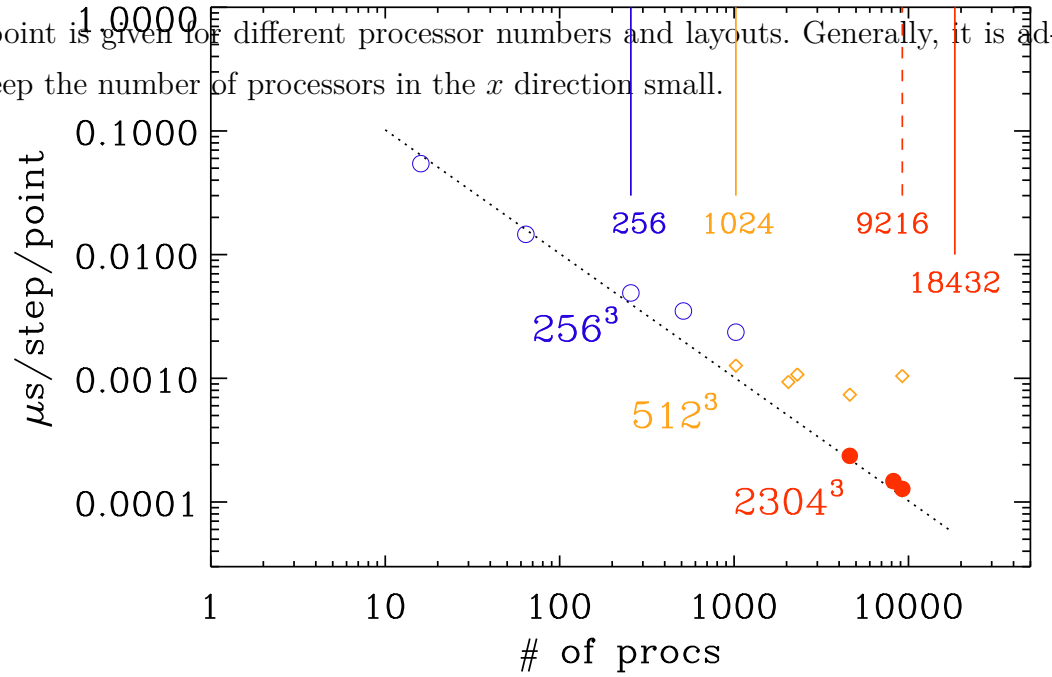


图 15: Strong scaling on Triolith (2014).

表 9: Triolith timings

proc	$\frac{\mu s}{\text{pt step}}$	resol.	layout
16	0.075	128^3	2x2x4
16	0.065	128^3	1x4x4
16	0.0544	256^3	1x4x4
64	0.0146	256^3	1x8x8
64	0.0164	256^3	2x4x8
256	0.0049	256^3	1x16x16
512	0.0035	256^3	2x16x16
1024	0.00236	256^3	2x16x32
1024	0.00127	512^3	2x16x32
1024	0.00129	512^3	4x16x16
2048	9.34×10^{-4}	512^3	4x16x32
2304	0.00107	576^3	4x18x32
4096	3.6×10^{-4}	1024^3	4x32x32
4096	3.8×10^{-4}	1024^3	8x16x32
4096	4.2×10^{-4}	1024^3	4x16x64
4608	7.38×10^{-4}	576^3	8x18x32
4608	2.66×10^{-4}	1152^3	4x32x36
4608	3.03×10^{-4}	1152^3	4x36x32
4608	3.12×10^{-4}	1152^3	4x18x64
4608	2.36×10^{-4}	2304^3	2x32x72
8192	1.475×10^{-4}	2048^3	4x32x64
9216	0.00104	576^3	16x18x32
9216	1.276×10^{-4}	2304^3	4x36x64
9216	1.30×10^{-4}	2304^3	4x32x72

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.

表 10: Lindgren timings

proc	$\frac{\mu s}{\text{pt step}}$	resol.	layout
1536	0.00171	$512^2 \times 384$	2x32x24
2048	0.00129	$512^2 \times 1024$	1x32x64
2048	0.00129	$1024^2 \times 2048$	1x32x64
4096	4.6×10^{-4}	2048^3	4x16x64
9216	2.36×10^{-4}	2304^3	4x48x48

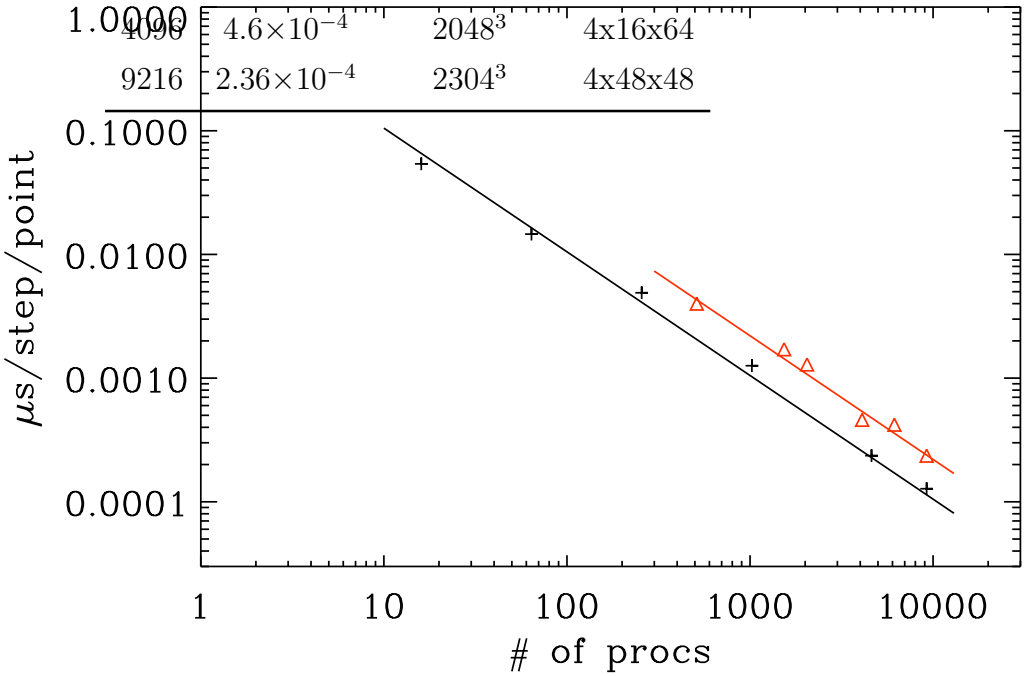


图 16: Comparison Triolith (black, plus signs) and Lindgren (red, triangles). Weak scaling (2014).

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 an underscore ad 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,

```
do n=n1,n2
do m=m1,m2
  [...]
enddo
enddo
```

Alternatively nested loops may be written on a single line, i.e.

```
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:

```
      some fortran code
      some more fortran code
!
!  A descriptive comment explaining what the following few lines
!  of code do.
!
      the fortran code being described
      the fortran code being described
      ...
!
!  A final detail described here.
!
      the final fortran code
      the final fortran code
      ...
```

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:

```
!***********************************************************************
    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:

```
;;; ~/.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$"   . f90-mode)
         ("\\.inc$"      . 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 $\boldsymbol{u}$ (or $\boldsymbol{A}$) is set to normally distributed, uncorrelated random numbers in all meshpoints for all three components. The power spectrum of $\boldsymbol{u}$ ($\boldsymbol{A}$) increases then quadratically with wavenumber k (without cutoff) and the power spectrum of $\boldsymbol{\omega}$ (or $\boldsymbol{B}$) increases like k^4.

Note that a random initial condition contains significant power at the Nyquist frequency ($k_{\rm Ny} = \pi/N$, 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. 17, 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 $k_{\rm Ny}$. 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. 17, for $k < k_{\rm Ny}/2$ the power spectra for filtered and unfiltered initial conditions is almost the same.

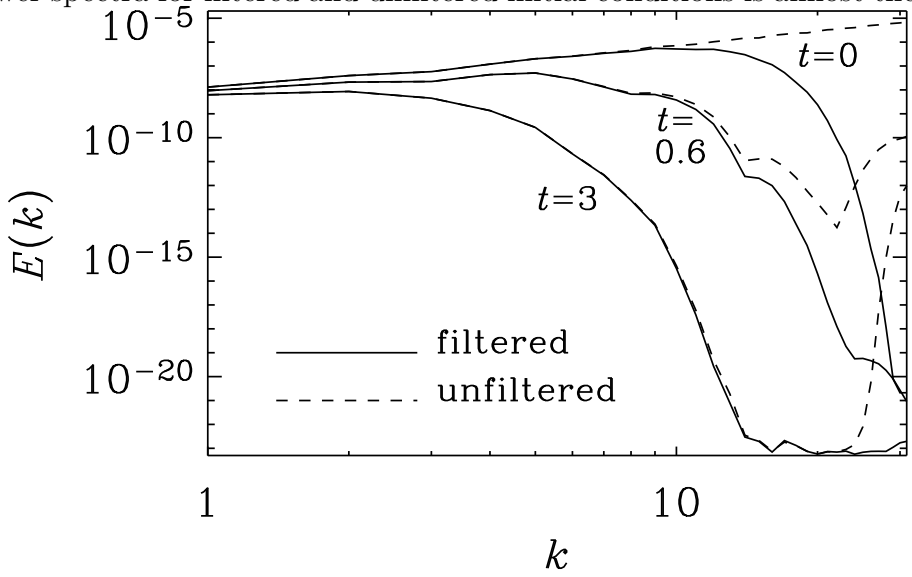


图 17: Velocity power spectra at three different times with and without filtering of the initial condition.

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 relhel_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 initpower=2, corresponding to a k^2 energy spectrum for kinetic energy. This would be suitable for the subinertial range from $k = 1$ to $k = k_{\rm p}$ (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 $n =$ '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.,

```
&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'.

$$\boldsymbol{A} = (\cos z,\ \sin z,\ 0), \quad \text{or} \quad \boldsymbol{u} = (\cos z,\ \sin z,\ 0) \tag{125}$$

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, φ, z), by the vector potential

$$\boldsymbol{A} = \Phi_{\rm m} \begin{pmatrix} 0 \\ 0 \\ -\theta(r-R)\delta(z) \end{pmatrix} , \tag{126}$$

resulting in a magnetic field

$$\boldsymbol{B} = \Phi_{\rm m} \begin{pmatrix} 0 \\ \delta(r-R)\delta(z) \\ 0 \end{pmatrix} . \tag{127}$$

Here $\Phi_{\rm m}$ is the magnetic flux through the tube, $\theta(x)$ denotes the Heaviside function, and

$$\delta(x) = \theta'(x) \tag{128}$$

is Dirac's delta function.

Any smoothed versions of $\theta(x)$ and $\delta(x)$ will do, as long as the consistency condition (133) is satisfied. E. g. the pairs

$$\delta_\varepsilon(x) = \frac{1}{\sqrt{2\pi\varepsilon^2}} e^{-\frac{x^2}{2\varepsilon^2}} , \quad \theta_\varepsilon(x) = \frac{1}{2}\left(1 + \operatorname{erf}\frac{x}{\sqrt{2}\varepsilon}\right) \tag{129}$$

or

$$\delta_\varepsilon(x) = \frac{1}{2\varepsilon}\frac{1}{\cosh^2\frac{x}{\varepsilon}} , \quad \theta_\varepsilon(x) = \frac{1}{2}\left(1 + \tanh\frac{x}{\varepsilon}\right) \tag{130}$$

are quite popular. Another possibility is a constant or box-like profile with

$$\delta_\varepsilon(x) = \frac{1}{2\varepsilon}\theta(|x| - \varepsilon) , \quad \theta_\varepsilon(x) = \frac{1}{2}\left\{1 + \max[-1, \min(x/\varepsilon), 1]\right\} \tag{131}$$

Note, however, that the Gaussian profile (134) is the only one that yields a radially symmetric (with respect to the distance from the central line of the torus) magnetic field profile $B_\varphi = B_\phi(\sqrt{(r-R)^2+z^2})$ if ε is sufficiently small.

In Cartesian coordinates, the vector potential (126) takes the form

$$\boldsymbol{A} = \Phi_{\rm m} \begin{pmatrix} 0 \\ 0 \\ -\theta\left(\sqrt{x^2+y^2}-R\right)\delta(z) \end{pmatrix} . \tag{132}$$

Here $\Phi_{\rm m}$ is the magnetic flux through the tube, $\theta(x)$ denotes the Heaviside function, and

$$\delta(x) = \theta'(x) \tag{133}$$

is Dirac's delta function.

Any smoothed versions of $\theta(x)$ and $\delta(x)$ will do, as long as the consistency condition (133) is satisfied. E. g. the pairs

$$\delta_\varepsilon(x) = \frac{1}{\sqrt{2\pi\varepsilon^2}} e^{-\frac{x^2}{2\varepsilon^2}} , \quad \theta_\varepsilon(x) = \frac{1}{2}\left(1 + \operatorname{erf}\frac{x}{\sqrt{2}\varepsilon}\right) \tag{134}$$

or

$$\delta_\varepsilon(x) = \frac{1}{2\varepsilon}\frac{1}{\cosh^2\frac{x}{\varepsilon}} , \quad \theta_\varepsilon(x) = \frac{1}{2}\left(1 + \tanh\frac{x}{\varepsilon}\right) \tag{135}$$

are quite popular. Another possibility is a constant or box-like profile with

$$\delta_\varepsilon(x) = \frac{1}{2\varepsilon}\theta(|x| - \varepsilon) , \quad \theta_\varepsilon(x) = \frac{1}{2}\left\{1 + \max[-1, \min(x/\varepsilon), 1]\right\} \tag{136}$$

Note, however, that the Gaussian profile (134) is the only one that yields a radially symmetric (with respect to the distance from the central line of the torus) magnetic field profile $B_\varphi = B_\phi(\sqrt{(r-R)^2+z^2})$ if ε is sufficiently small.

In Cartesian coordinates, the vector potential (126) takes the form

$$\boldsymbol{A} = \Phi_{\rm m} \begin{pmatrix} 0 \\ 0 \\ -\theta\left(\sqrt{x^2+y^2}-R\right)\delta(z) \end{pmatrix} . \tag{137}$$

C.5 Vertical stratification

Gravity, $\mathrm{g} = -\boldsymbol{\nabla}$, is specified in terms of a potential Φ. In slab geometry, $\Phi = \Phi(z)$, we have $\mathrm{g} = (0, 0, g_z)$ and $g_z = -\mathrm{d}\Phi/\mathrm{d}z$.

Use grav_profile='const' together with gravz= -1 to get

$$\Phi = (z - z_\infty)(-g_z), \quad (-g_z) > 0. \tag{138}$$

Use grav_profile='linear' to get

$$\Phi = \tfrac{1}{2}(z^2 - z_\infty^2)\nu_{\rm g}^2, \quad g_z = -\nu_{\rm g}^2 z \tag{139}$$

where $\nu_{\rm g}$ is the vertical epicyclic frequency. For a Keplerian accretion disc, $\nu_{\rm g} = \Omega$. For galactic discs, $\nu_{\rm g} = 0.5\Omega$ is representative of the solar neighborhood.

The value of z_∞ is determined such that $\rho = \rho_0$ and $c_{\rm s}^2 = c_{\rm s0}^2$ at $z = z_{\rm ref}$. This depends on the values of γ and the polytropic index m (see below).

C.5.1 Isothermal atmosphere

Here we want $c_{\rm s} = c_{\rm s0} = {\rm const}$. Using initlnrho='isothermal' means

$$\ln\frac{\rho}{\rho_0} = -\gamma\frac{\Phi}{c_{\rm s0}^2}\ . \tag{140}$$

The entropy is then initialized to

$$\frac{s}{c_p} = (\gamma{-}1)\frac{\Phi}{c_{\rm s0}^2}\ . \tag{141}$$

In order that $\rho = \rho_0$ and $c_{\rm s}^2 = c_{\rm s0}^2$ at $z = z_{\rm ref}$, we have to choose $z_\infty = z_{\rm ref}$.

C.5.2 Polytropic atmosphere

For a polytropic equation of state, $p = K\rho^\Gamma$, where generally $\Gamma \neq \gamma$, we can write

$$-\boldsymbol{\nabla}h + T\boldsymbol{\nabla}s = -\frac{1}{\rho}\boldsymbol{\nabla}p = -\boldsymbol{\nabla}\left(\frac{\Gamma K}{\Gamma{-}1}\rho^{\Gamma-1}\right) \equiv -\boldsymbol{\nabla}\tilde{h}, \tag{142}$$

where we have introduced a pseudo enthalpy $\tilde{h}$ as

$$\tilde{h} = \frac{\Gamma K}{\Gamma{-}1}\rho^{\Gamma-1} = \left[\left(1-\frac{1}{\gamma}\right)\Big/\left(1-\frac{1}{\Gamma}\right)\right]\ h\ . \tag{143}$$

Obviously, for $\Gamma = \gamma$, the pseudo enthalpy $\tilde{h}$ is identical to h itself. Instead of specifying Γ, one usually defines the polytropic index $m = 1/(\Gamma{-}1)$. Thus, $\Gamma = 1 + 1/m$, and

$$\tilde{h} = (m{+}1)\left(1-\frac{1}{\gamma}\right)\ h \tag{144}$$

This is consistent with a fixed entropy dependence, where s only depends on ρ like

$$\frac{s}{c_p} = \left(\frac{\Gamma}{\gamma}-1\right)\ln\frac{\rho}{\rho_0}\ , \tag{145}$$

and implies that

$$\ln\frac{c_{\rm s}^2}{c_{\rm s0}^2} = (\Gamma{-}1)\ln\frac{\rho}{\rho_0}\ . \tag{146}$$

For hydrostatic equilibrium we require $\tilde{h} + \Phi = \tilde{h}_0 = {\rm const}$. 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 $c_{\rm s}^2 = (\gamma{-}1)h$ together with (144) we find

$$c_{\rm s}^2 = -\frac{\gamma}{m{+}1}\,\Phi. \tag{147}$$

In order that $\rho = \rho_0$ and $c_{\rm s}^2 = c_{\rm s0}^2$ at $z = z_{\rm ref}$, we have to choose (remember that g_z is normally negative!)

$$z_\infty = z_{\rm ref} + (m{+}1)\frac{c_{\rm s0}^2}{\gamma(-g_z)} \quad \text{for grav\_profile='const'}, \tag{148}$$

and

$$z_\infty^2 = z_{\rm ref}^2 + (m{+}1)\frac{c_{\rm s0}^2}{\frac{1}{2}\gamma\nu_{\rm g}^2} \quad \text{for grav\_profile='linear'.} \tag{149}$$

Thus, when using initlnrho='polytropic_simple' we calculate

$$\ln\frac{c_{\rm s}^2}{c_{\rm s0}^2} = \ln\left[-\frac{\gamma\Phi}{(m{+}1)c_{\rm s0}^2}\right] \tag{150}$$

and so the stratification is given by

$$\ln\frac{\rho}{\rho_0} = m\,\ln\frac{c_{\rm s}^2}{c_{\rm s0}^2}\,, \quad \frac{s}{c_p} = \left(\frac{\Gamma}{\gamma}-1\right) m\,\ln\frac{c_{\rm s}^2}{c_{\rm s0}^2}\,. \tag{151}$$

C.5.3 Changing the stratification

Natural: measure length in units of $c_{\rm s0}^2/g_z$. Can increase stratification by moving $z_{\rm top}$ close to z_∞ or, better still, keeping $z_{\rm top}=0$ and moving $z_{\rm bot}\to-\infty$. 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 $g_{\rm max}$, which can be obtained by putting $z_{\rm top}=z_\infty$ in (148), so

$$g_{\rm max} = \frac{m{+}1}{\gamma}\,\frac{c_{\rm s0}^2}{z_{\rm top}-z_{\rm ref}}. \tag{152}$$

For $m=1$, $\gamma=5/3$, $z_{\rm top}=1$, and $z_{\rm ref}=0$, for example, we have $g_{\rm max}=6/5=1.2$.

Gravitational box units: measure speed in units of $\sqrt{g_z d}$. The limit of vanishing stratification corresponds to $c_{\rm s0}\to\infty$. 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 $z=-2$ and the bottom at $z=-3$. In addition, the reference height, $z_{\rm ref}$, is chosen to be at the top of the box, i.e. $z_{\rm ref}=-2$. From Eq. (148) we have $c_{\rm s0}^2=\gamma(-g_z)(-z_{\rm ref})/(m+1)$. Using $(-g_z)=1$ and $m=1$ we find $c_{\rm s0}^2=\gamma$, so $c_{\rm s0}=1.291$ (for $\gamma=5/3$). Values for other combinations are listed in Table 11.

C.5.4 The Rayleigh number

In Ref. [?] the Rayleigh number is defined as

$$\mathrm{Ra} = \frac{g d^4}{\overline{\nu}\,\overline{\chi}}\left(-\frac{\mathrm{d}s/c_p}{\mathrm{d}z}\right)_{\rm hydrostat}, \tag{153}$$

表 11: Correspondence between density contrast, top and bottom values of z, and $c_{\rm s0}$ for $(-g_z) = 1$, $m = 1$, and $\gamma = 5/3$.

$\rho_{\rm bot}/\rho_{\rm top}$	$z_{\rm bot}$	$z_{\rm top}$	$c_{\rm s0}$
1.5	3	2	1.291
3	1.5	0.5	0.645
6	1.2	0.2	0.408
11	1.1	0.1	0.289
21	1.05	0.05	0.204

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 ν was assumed constant) or $\overline{\nu} = \mu/\overline{\rho}$ (if μ was assumed constant). Note that $\overline{\rho}$ is the average mass in the box per volume, which is conserved. For a polytrope we have

$$\left(-\frac{{\rm d}s/c_p}{{\rm d}z}\right)_{\rm hydrostat} = \left[1-(m+1)\left(1-\frac{1}{\gamma}\right)\right]\frac{1}{z_\infty - z_{\rm m}}, \tag{154}$$

where $z_{\rm m} = (z_1+z_2)/2$. This factor was also present in the definition of Hurlburt et al. [?], but their definition differs slightly from Eq. (153), 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 ρ^2, their definition included the extra factor $[(z_\infty - z_{\rm m})/d]^2$. Therefore

$${\rm Ra}_{\rm HTM} = \left(\frac{z_\infty - z_{\rm m}}{d}\right)^{2m}\left(\frac{\rho_{\rm top}}{\overline{\rho}}\right)^2 {\rm Ra} \tag{155}$$

In the first model of Hurlburt et al. (1984), the Rayleigh number, ${\rm Ra}_{\rm HTM}$, was chosen to be 310 times supercritical, and the critical Rayleigh number was around 400, so ${\rm Ra}_{\rm HTM} = 1.25\times10^5$. In their model the density contrast was 1:1.5 and $m = 1$. This turns out to correspond to ${\rm Ra} = 4.9\times10^4$, $F_{\rm bot} = 0.0025$, and $K = 0.002$.

Another model that was considered by Hurlburt & Toomre (1988) had ${\rm Ra}_{\rm HTM} = 10^5$, a density contrast of 11, and had a vertical imposed magnetic field (Chandrasekhar number $Q = 72$). This corresponds to ${\rm Ra} = 3.6\times10^8$, $K = 0.0011$, $F_{\rm bot} = 0.0014$.

C.5.5 熵边界条件 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. $F_{\rm bot} = -K{\rm d}T/{\rm d}z$. Assuming that the density in the ghost zones has already been updated, we can calculate the entropy gradient from

$$F_{\rm bot} = -\frac{K}{c_p}\frac{c_{\rm s}^2}{\gamma-1}\left((\gamma-1)\frac{{\rm d}\ln\rho}{{\rm d}z} + \gamma\frac{{\rm d}s/c_p}{{\rm d}z}\right), \tag{156}$$

which gives

$$\frac{\mathrm{d}s/c_p}{\mathrm{d}z} = -\frac{\gamma-1}{\gamma}\left(c_p\frac{F_{\rm bot}}{Kc_{\rm s}^2}+\frac{\mathrm{d}\ln\rho}{\mathrm{d}z}\right) \tag{157}$$

for the derivative of the entropy at the bottom. This is implemented as the 'c1' boundary condition at the bottom.

C.5.6 温度边界条件 (翻译存疑 at the top)Temperature boundary condition at the top

In earlier papers the temperature at the top was set in terms of the quantity ξ_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

$$c_{\rm s,top}^2 = \gamma\xi_0 gd. \tag{158}$$

$$c_{\rm s,bot}^2 = \left(\xi_0+\frac{1}{m+1}\right)\gamma gd. \tag{159}$$

表 12: Correspondence between ξ_0 and $c_{\rm s,bot}^2$ in single layer polytropes.

ξ_0	$c_{\rm s,bot}^2$
10.00	17.500
0.20	1.167
0.10	1.000
0.05	0.917
0.02	0.867

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.

If the top boundary is at $z=0$, the relevant potential field for $z>0$ is given by

$$\tilde{A}_x(k_x,k_y,z) = C_x(\boldsymbol{k}_{xy})\,e^{-\kappa z}\,, \tag{160}$$

$$\tilde{A}_y(k_x,k_y,z) = C_y(\boldsymbol{k}_{xy})\,e^{-\kappa z}\,, \tag{161}$$

$$\tilde{A}_x(k_x,k_y,z) = C_z(\boldsymbol{k}_{xy})\,e^{-\kappa z}\,, \tag{162}$$

where

$$\tilde{A}_i(k_x,k_y,z) \equiv \int e^{-i\boldsymbol{k}_{xy}\cdot\boldsymbol{x}}A_i(x,y,z)\,dx\,dy \tag{163}$$

is the horizontal Fourier transform with $\boldsymbol{k}_{xy} \equiv (k_x, k_y, 0)$, and $k \equiv |\boldsymbol{k}_{xy}|$. Note that this implies a certain gauge and generally speaking the z dependence in Eq. (162) is completely arbitrary, but the form used here works well in terms of numerical stability.

At the very boundary, the potential field (160)–(162) implies

$$\frac{\partial \tilde{\boldsymbol{A}}}{\partial z} + \kappa \tilde{\boldsymbol{A}} \;=\; 0 \;, \tag{164}$$

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

$$\frac{df}{dz} + \kappa f = 0 \tag{165}$$

using ghost points, we first write it as

$$\frac{d}{dz}\left(f\, e^{\kappa z}\right) = 0 \tag{166}$$

and implement this as symmetry condition for the variable $\phi(z) \equiv f(z)\, e^{\kappa z}$:

$$\phi_{N-j} = \phi_{N+j} \;, \qquad j = 1, 2, 3 \tag{167}$$

(where z_N is the position of the top boundary and z_{N+1}, …are the boundary points). In terms of f, this becomes

$$f_{N+j} = f_{N-j}\, e^{-\kappa(z_{N+j} - z_{N-j})} \;. \tag{168}$$

Note that although the exponential term in Eq. (168) looks very much like the exterior potential field (160)–(162), 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 [?].

Assume $s = 0$ (isentropy), so the equations in 2-D are

$$u_x u_{x,x} + (u_y^{(0)} + u_y) u_{x,y} = 2\Omega u_y - h_{,x} \tag{169}$$

$$u_x u_{y,x} + (u_y^{(0)} + u_y) u_{y,y} = -(2 - q)\Omega u_x - h_{,y} \tag{170}$$

where $u_y^{(0)} = -q\Omega x$. Express u in terms of a stream function, so u $= \nabla \times (\psi \hat{\mathrm{z}})$, or

$$u_x = \psi_{,y}, \quad u_y = -\psi_{,x}. \tag{171}$$

Ansatz for enthalpy

$$h = \tfrac{1}{2}\delta^2\Omega^2(R^2 - x^2 - \epsilon^2 y^2 - z^2/\delta^2) \tag{172}$$

$$\psi = -\tfrac{1}{2}\sigma\Omega(R^2 - x^2 - \epsilon^2 y^2) - \tfrac{1}{2}q\Omega x^2 \tag{173}$$

This implies

$$u_x = \sigma\Omega\epsilon^2 y, \quad u_y = (q - \sigma)\Omega x \tag{174}$$

and $u_{x,x} = u_{y,y} = 0$. Inserting into Eqs (169) and (170) yields

$$(-q + q - \sigma)\sigma\epsilon^2 = 2(q - \sigma) + \delta^2 \tag{175}$$

$$\sigma(q - \sigma) = -(2 - q)\sigma + \delta^2 \tag{176}$$

where we have already canceled common Ω^2 factors in both equations and common ϵ^2 factors in the last equation. Simplifying both equations yields

$$-\sigma^2\epsilon^2 = 2(q - \sigma) + \delta^2 \tag{177}$$

$$-\sigma^2 = -2\sigma + \delta^2 \tag{178}$$

The second equation yields

$$\delta^2 = (2 - \sigma)\sigma \tag{179}$$

and subtracting the two yields

$$\sigma^2 = 2q/(1 - \epsilon^2) \tag{180}$$

表 13: Dependence of ϵ and δ on ϵ.

ϵ	σ	δ
0.1	1.74	0.67
0.2	1.77	0.64
0.3	1.82	0.58
0.4	1.89	0.46
0.48	1.97	0.22
0.5	2	0

D **一些特定的边界条件** Some specific boundary conditions

在这一章节中, 我们讨论了一些常用的球坐标系或柱坐标系中的边界条件.

D.1 **理想导体边界条件** Perfect-conductor boundary condition

这是一个常用的磁场的边界分布条件, 它意味着在边界上

$$B_n = 0 \tag{181}$$

$$\boldsymbol{E}_t = 0 \tag{182}$$

, 下标 n 表示法向分量, 而 $\boldsymbol{E}_t$ 表示切向的电场分量.

In Cartesian geometry, these conditions can be implemented by setting the two tangential components of the vector potential $\boldsymbol{A}$ 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

$$r \sin\theta \, B_r = \partial_\theta(\sin\theta \, A_\phi) - \partial_\phi A_\theta = 0 \; . \tag{183}$$

This can be achieved by setting

$$A_\phi = A_\theta = 0 \tag{184}$$

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 θ, we must have

$$B_\theta = \frac{1}{r \sin\theta} \partial_\phi A_r - \frac{1}{r} \partial_r (r A_\phi) = 0 \; . \tag{185}$$

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

$$u_n = 0 \; , \tag{186}$$

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:

$$S_{r\theta} = \frac{1}{r}\partial_\theta u_r + r\partial_r\left(\frac{u_\theta}{r}\right) , \tag{187}$$

$$S_{r\phi} = \frac{1}{r\sin\theta}\partial_\phi u_r + r\partial_r\left(\frac{u_\phi}{r}\right) . \tag{188}$$

Both of them vanish if we require

$$u_r = 0 , \qquad \partial_r(u_\theta/r) = 0 , \qquad \partial_r(u_\phi/r) = 0 . \tag{189}$$

We implement this by requiring u_r to be antisymmetric and u_θ/r and u_ϕ/r to be symmetric with respect to the boundary.

The more general condition

$$r^\alpha\partial_r(u_\theta/r^\alpha) = \partial_r u_\theta - \frac{\alpha}{r}u_\theta = 0 \tag{190}$$

(where α is a constant) can be implemented by requiring u_θ/r^α to be symmetric.

At a boundary $\theta = \text{const}$, the stress-free boundary condition will take the form

$$S_{r\theta} = \frac{1}{r}\partial_\theta u_r + r\partial_r\left(\frac{u_\theta}{r}\right) = 0 , \tag{191}$$

$$S_{\theta\phi} = \frac{1}{r\sin\theta}\partial_\phi u_\theta + \sin\theta\,\partial_\theta\left(\frac{u_\phi}{r\sin\theta}\right) = 0 . \tag{192}$$

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

$$\frac{\sin\theta}{r}\partial_\theta\left(\frac{u_\phi}{\sin\theta}\right) = 0 \tag{193}$$

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:

$$B_\theta = \frac{1}{r\sin\theta}\partial_\phi A_r - \frac{1}{r}\partial_r(rA_\phi) = 0 , \tag{194}$$

$$B_\phi = \frac{1}{r}\partial_r(rA_\theta) - \frac{1}{r}\partial_\theta A_r = 0 . \tag{195}$$

Which are satisfied by setting

$$A_r = 0 , \qquad \partial_r(rA_\theta) = 0 , \qquad \partial_r(rA_\phi) = 0 , \tag{196}$$

and these are implemented by requiring A_r to be antisymmetric, and rA_θ and rA_ϕ to be symmetric.

On a boundary $\theta = \text{const}$, we have

$$r \sin\theta \, B_r = \partial_\theta(\sin\theta \, A_\phi) - \partial_\phi A_\theta = 0 \, , \tag{197}$$

$$r B_\phi = \partial_r(r A_\theta) - \partial_\theta A_r = 0 \tag{198}$$

which can be achieved by setting

$$\partial_\theta A_r = 0 \, , \qquad A_\theta = 0 \, , \qquad \partial_\theta(\sin\theta \, A_\phi) = 0 \, . \tag{199}$$

We thus require A_r and $\sin\theta \, A_\phi$ to be symmetric, and A_θ 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 $\nu\nabla^2\boldsymbol{u}$ 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 ∇^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 ∇^6, or k^6 is Fourier space. Figure 18 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, hyper-resistivity, 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 ψ, so that Gauss theorem applies and we have

$$\frac{\partial\psi}{\partial t}+\boldsymbol{\nabla}\cdot\boldsymbol{\mathcal{J}}=0 \tag{200}$$

For density, the flow due to mass diffusion is usually taken as the phenomenological Fick's Law

$$\boldsymbol{\mathcal{J}}=-D\boldsymbol{\nabla}\rho \tag{201}$$

i.e., proportional to the density gradient, in the opposite direction. This leads to the usual Laplacian diffusion

$$\frac{\partial \rho}{\partial t} = D\nabla^2 \rho \tag{202}$$

under the assumption that the diffusion coefficient D is isotropic. Higher order hyperdiffusion of order $2n$ involves a generalization of Eq. (201), to

$$\mathcal{F}^{(n)} = (-1)^n D^{(n)} \boldsymbol{\nabla}^{2n-1} \rho \,. \tag{203}$$

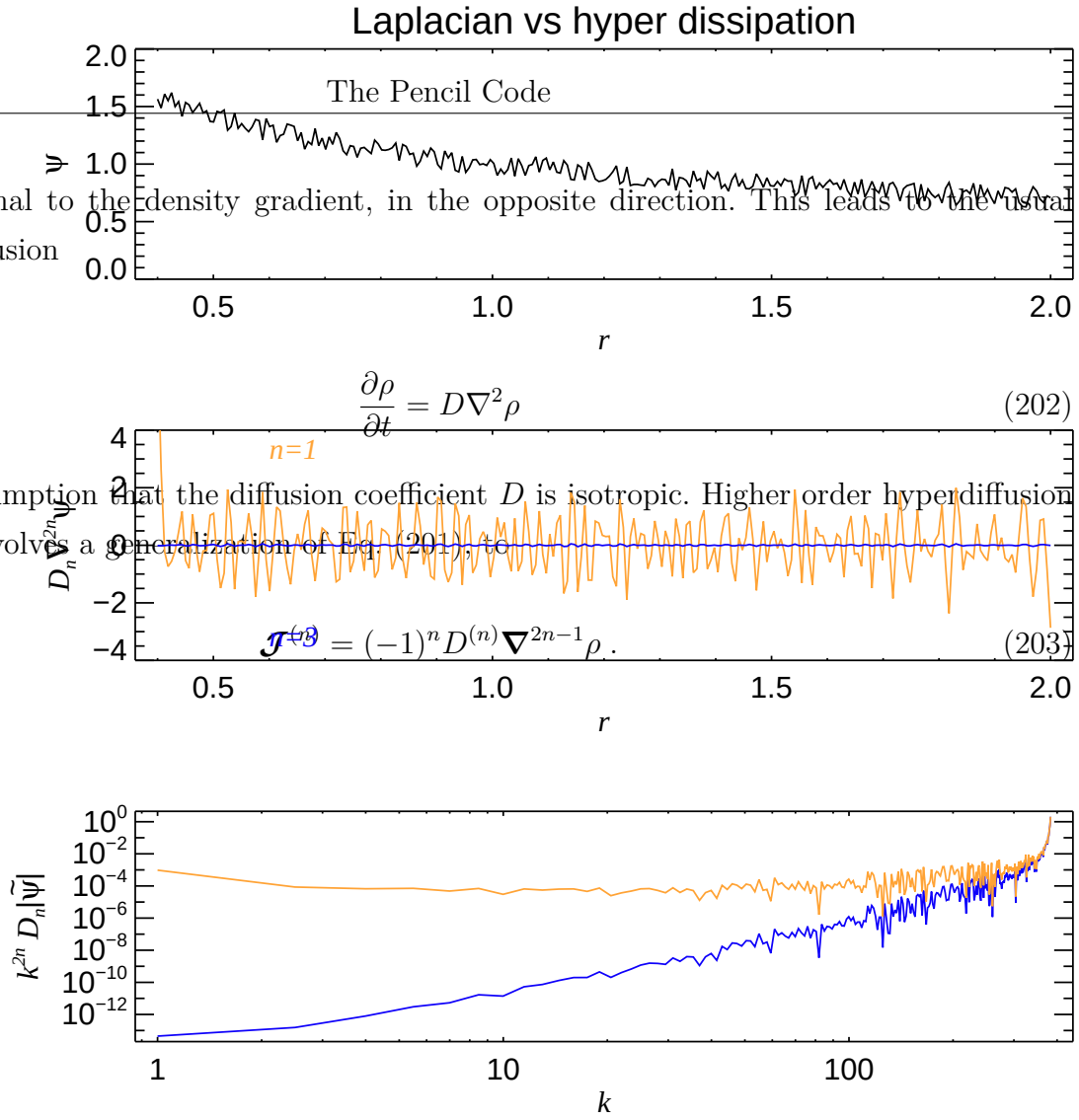


图 18: Dissipation acting on a scalar field ψ, for n=1 (Laplacian dissipation) and n=3 (third-order hyperdissipation). The field is initially seeded with noise (upper panel). For n=3 the large scale is not affected as much as in the n=1 case, which is seen by the larger wiggling of the latter in the middle 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 ($k \simeq 1$), the effect of n=3 is indeed negligible.

In our case, we are interested in the case $n = 3$, so that the hyperdiffusion term is

$$\frac{\partial \rho}{\partial t} = D^{(3)} \nabla^6 \rho. \tag{204}$$

The hyperdiffusion coefficient $D^{(3)}$ can be calculated from D assuming that at the Nyquist frequency the two formulations (202) and (204) yield the same quenching. Considering a wave as a Fourier series in one dimension (x), one element of the series is expressed as

$$\psi_k = A \mathrm{e}^{i(kx+\omega t)} \tag{205}$$

Plugging it into the second order diffusion equation (202) we have the dispersion condition $i\omega = -Dk^2$. The sixth order version (204) 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

$$D^{(3)} = D \left(\frac{\Delta x}{\pi} \right)^4 \tag{206}$$

For the magnetic potential, resistivity has the same formulation as mass diffusion

$$\frac{\partial \boldsymbol{A}}{\partial t} = -\eta \nabla \times \boldsymbol{B} = \eta \nabla^2 \boldsymbol{A}, \tag{207}$$

where we used the Coulomb gauge $\boldsymbol{\nabla} \cdot \boldsymbol{A} = 0$. The algebra is the same as above, also yielding $\eta^{(3)} = \eta(\Delta x/\pi)^4$. For entropy, the heat conduction term is

$$\frac{\partial S}{\partial t} = \frac{1}{\rho T} \boldsymbol{\nabla} \cdot (K \boldsymbol{\nabla} T) \,, \tag{208}$$

and requiring that K be constant, we substitute it by

$$\frac{\partial S}{\partial t} = \frac{K^{(3)}}{\rho T} \nabla^6 T. \tag{209}$$

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

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \boldsymbol{\nabla} \cdot \boldsymbol{u} \right) \,, \tag{210}$$

which only allows the viscous acceleration to be reduced to the simple formulation $\nu \nabla^2 \boldsymbol{u}$ 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

$$f_{\rm visc} = \rho^{-1}\boldsymbol{\nabla}\cdot(2\rho\nu \mathrm{S}) \tag{211}$$

So, for the hyperviscous force, we must replace the rate-of-strain tensor by a high order version

$$f_{\rm visc}^{\rm (hyper)} = \rho^{-1}\boldsymbol{\nabla}\cdot\left(2\rho\nu_n \mathrm{S}^{(n)}\right) \tag{212}$$

where the $n^{\rm th}$-order rate of strain tensor is

$$\mathrm{S}^{(n)} = (-\nabla^2)^{n-1}\mathrm{S}. \tag{213}$$

For the $n=3$ case it is

$$S_{ij}^{(3)} = \frac{1}{2}\left(\frac{\partial^5 u_j}{\partial {x_i}^5} + \frac{\partial^4}{\partial {x_i}^4}\left(\frac{\partial u_i}{\partial x_j}\right) - \frac{1}{3}\frac{\partial^4}{\partial {x_i}^4}\left(\boldsymbol{\nabla}\cdot\boldsymbol{u}\right)\right) . \tag{214}$$

Plugging it into Eq. (212), and assuming $\mu_3 = \rho\nu_3 = \text{const}$

$$f_{\rm visc}^{\rm (hyper)} = \nu_3\left(\nabla^6\boldsymbol{u} + \frac{1}{3}\nabla^4(\boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\boldsymbol{u}))\right) . \tag{215}$$

For $\nu_3 = \text{const}$, we have to take derivatives of density as well

$$f_{\rm visc}^{\rm (hyper)} = \nu_3\left(\nabla^6\boldsymbol{u} + \frac{1}{3}\nabla^4(\boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\boldsymbol{u})) + 2\mathrm{S}^{(3)}\cdot\boldsymbol{\nabla}\ln\rho\right) \tag{216}$$

E.2.2 Non-conservative cases

Equations (215) and (216) explicitly conserve linear and angular momentum. Although desirable properties, such expressions are cumbersome and numerically expensive, due to the fourth order derivatives of $\boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\boldsymbol{u})$.

This term, however, is only important when high compressibility is present (since it depends on the divergence of $\boldsymbol{u}$). In practice we drop this term and use a simple hyperviscosity

$$f_{\rm visc} = \begin{cases} \nu_3\nabla^6\boldsymbol{u} & \text{if } \mu = \text{const} \\ \nu_3\left(\nabla^6\boldsymbol{u} + 2\mathrm{S}^{(3)}\cdot\boldsymbol{\nabla}\ln\rho\right) & \text{if } \nu = \text{const} \end{cases} \tag{217}$$

Notice that this can indeed be expressed as the divergence of a simple rate-of-strain tensor

$$S_{ij}^{(3)} = \frac{\partial^5 u_i}{\partial {x_j}^5} , \tag{218}$$

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

$$S_{ij} = \frac{1}{2}\left(\frac{\partial^5 u_i}{\partial {x_j}^5} + \frac{\partial^5 u_j}{\partial {x_i}^5}\right) \tag{219}$$

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

$$\mathrm{Re}_{\mathrm{grid}} = u_{\mathrm{rms}} \big/ \left(\nu_n k_{\mathrm{Ny}}^{2n-1}\right) \tag{220}$$

approximately constant. Here, $k_{\mathrm{Ny}} = \pi/\delta x$ is the Nyquist wavenumber and δx 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 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 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.

$$\mathrm{Re}_{\mathrm{half-grid}} = u_{\mathrm{rms}} \big/ \left[\nu_n \left(k_{\mathrm{Ny}}/2\right)^{2n-1}\right] \approx 5\text{–}7 \tag{221}$$

The following table gives some typical values used in simulations with forcing wavenumber $k_{\mathrm{f}} = 1.5$ 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 ν_3; see Table 14.

表 14: Empirical values of viscosity and hyperviscosity, as well as hyperdiffusion for density, at different numerical resolution, for simulations with forcing wavenumber $k_{\rm f}=1.5$ and a forcing amplitude of $f_0=0.02$ in a 2π periodic domain. In all cases the half-mesh Reynolds number is about 5–7. For comparison, estimates of the numerical 4th order hyperdiffusion resulting from a third order time step are give for two values of the CFL parameter.

N	ν_1	ν_2	ν_3	D_3	$\kappa_2^{\rm CFL=0.4}$	$\kappa_2^{\rm CFL=0.9}$
16	1×10^{-2}	3×10^{-4}	2×10^{-5}	6×10^{-7}	7×10^{-4}	1×10^{-4}
32	5×10^{-3}	4×10^{-5}	6×10^{-7}	2×10^{-8}	1×10^{-6}	2×10^{-5}
64	2×10^{-3}	5×10^{-6}	2×10^{-8}	6×10^{-10}	2×10^{-7}	3×10^{-6}
128	1×10^{-3}	6×10^{-7}	6×10^{-10}	2×10^{-11}	3×10^{-8}	4×10^{-7}
256	5×10^{-4}	8×10^{-8}	2×10^{-11}	6×10^{-13}	4×10^{-9}	5×10^{-8}
512	2×10^{-4}	1×10^{-8}	6×10^{-13}	2×10^{-14}	5×10^{-10}	6×10^{-9}
1024	1×10^{-4}	1×10^{-9}	2×10^{-14}	6×10^{-16}	6×10^{-11}	8×10^{-10}

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

$$\kappa_2^{\rm CFL}=\frac{1}{24}u_{\rm rms}\left(C_{\rm CFL}\delta x\right)^3 \tag{222}$$

where $C_{\rm CFL}$ 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. (206). 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. (203) by

$$\boldsymbol{\mathcal{J}}=\left(D_x\frac{\partial^5\rho}{\partial x^5},D_y\frac{\partial^5\rho}{\partial y^5},D_z\frac{\partial^5\rho}{\partial z^5}\right), \tag{223}$$

so that different diffusion operates in different directions. Since D_x, D_y and D_z are constants, the divergence of this vector is

$$\boldsymbol{\nabla}\cdot\boldsymbol{\mathcal{J}}=D_x\frac{\partial^6\rho}{\partial x^6}+D_y\frac{\partial^6\rho}{\partial y^6}+D_z\frac{\partial^6\rho}{\partial z^6}. \tag{224}$$

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 (218).

Mathematically, these operations can be written compactly by noticing that the coefficients in Eq. (224) transform like diagonal tensors $\chi_{ij}^{(3)} = \chi_k^{(3)}\delta_{ijk}$, where δ_{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 ψ (density, any of the three components of the velocity or magnetic potential), we can write

$$\frac{\partial\psi}{\partial t} = -\chi_{ij}^{(3)}\partial_i\partial_j^5\psi = -\sum_q \chi_q^{(3)}\frac{\partial^6}{\partial x_q^6}\psi. \tag{225}$$

E.4 Hyperviscosity in Burgers shock

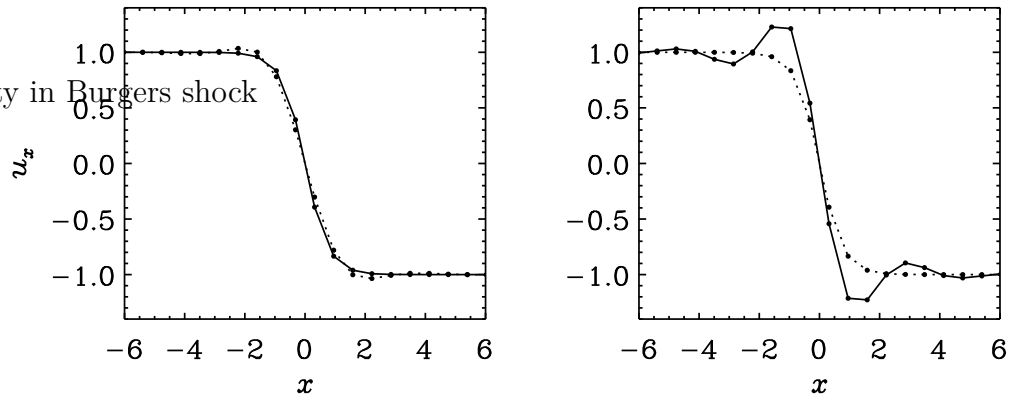


图 19: Left: Burgers shock from teach/PencilCode/material/BurgersShock (in the teaching material) with $-20 \leq x \leq 20$, $n_x = 64$ mesh points, $u_x = \mp 1$ on the two ends, $\nu = 0.4$ and either $\nu_3 = 0$ (solid line) or $\nu_3 = 0.05$ (dashed line). Right: similar to the left hand side, but 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. 19. 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 64^3 run (running on N_0 =ny×nz=2×1 CPUs) that you want to continue from 'VAR1' at 128^3 (on ny×nz=4×4 CPUs).

1. The remeshing code is part of the Pencil Code repository.
2. Create another run directory with current 'VAR1' as 'var.dat' (remesh.csh so far only works with 'var.dat'):

   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/; crtmp)
   run_64> (cd ../tmp_64/data ; mkproc-tree N_0)
   run_64> restart-new-dir-VAR ../tmp_64 1

3. Create the new run directory (linking the executables with -s):

   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; crtmp; pc_setupsrc; make)

4. Setup and do remeshing

   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 us 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.

```
&testfield_run_pars
  !linit_aatest=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:

   forrtl: severe (24): end-of-file during read, unit 1, file /wkspace/brandenb/pencil-code/axel/Poiseuille/2d/32c/data/param.nml, line 258, position 0

   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

```
&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

```
&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_aatest=T. Note that daainit=100 sets the reset interval to 100.

F.4 Restarting with particles from a run without them

If you want to restart from a run without particles to a run with them, you need to

(1) Compile a run with particles,

RunWithParticles$ pc_build

(2) Copy a VARN or var.dat into this new run directory. Say the old run is at the directory OldRun and you want to copy the var.dat of that run and restart with particles. You do

RunWithParticles$ pc_copyvar v v ../OldRun . -e

(3) In the start.in init_pars of the new run, add the lines

lread_oldsnap = T
ireset_tstart = 0

(4) Remove all calls to initial conditions (make all initlnrho, inituu, initss, initaa, etc 'nothing'), and

5) Run pc_start.

The variable lread_oldsnap makes the code on start time read from the f-array of the snapshop, instead of the default, which is to initialize it with zeros. The necessity in step (4) to remove all calls to initial conditions is because otherwise the code would rewrite the content of the f-array with these initial conditions, or add them on top of the existing values of the snapshot.

The variable ireset_tstart when set to zero makes it read the timestamp of the old snapshot and restart from that time. The default is 2, which sets the timestamp back to zero.

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

表 15: Combinations of ρ, 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.

ρ	p	s
1.0	1.0	0.3065
0.1	0.1	1.2275
0.01	0.01	2.1486

G.1.2 Temperature jump

表 16: Combinations of $c_{\rm s}^2$, 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.

$c_{\rm s}^2$	s
1.0	0.0
0.1	-2.3
0.01	-4.6
10^{-4}	-9.2

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 $\boldsymbol{f}$ of the form

$$\boldsymbol{f}(\boldsymbol{x},t) = \mathrm{Re}\{N\boldsymbol{f}_{\boldsymbol{k}(t)}\exp[i\boldsymbol{k}(t)\cdot\boldsymbol{x} + i\phi(t)]\}, \tag{226}$$

where $\boldsymbol{k}(t) = (k_x, k_y, k_z)$ is a time dependent wave vector, $\boldsymbol{x} = (x, y, z)$ is position, and $\phi(t)$ with $|\phi| < \pi$ is a random phase. On dimensional grounds the normalization factor is chosen to be $N = f_0 c_{\rm s}(kc_{\rm s}/\delta t)^{1/2}$, where f_0 is a nondimensional factor, $k = |\boldsymbol{k}|$, and δt 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, δt. We focus on the case where $|\boldsymbol{k}|$ is around 5, and select at each timestep randomly one of the 350 possible vectors in $4.5 < |\boldsymbol{k}| < 5.5$. We force the system with eigenfunctions of the curl operator,

$$\boldsymbol{f}_{\boldsymbol{k}} = \frac{i\boldsymbol{k}\times(\boldsymbol{k}\times\boldsymbol{e}) - \sigma|\boldsymbol{k}|(\boldsymbol{k}\times\boldsymbol{e})}{\sqrt{1+\sigma^2}\,\boldsymbol{k}^2\sqrt{1-(\boldsymbol{k}\cdot\boldsymbol{e})^2/\boldsymbol{k}^2}}, \tag{227}$$

where $\boldsymbol{e}$ is an arbitrary unit vector needed in order to generate a vector $\boldsymbol{k}\times\boldsymbol{e}$ that is perpendicular to $\boldsymbol{k}$. Note that $|\boldsymbol{f}_{\boldsymbol{k}}|^2 = 1$ and, for $\sigma = 1$, $i\boldsymbol{k}\times\boldsymbol{f}_{\boldsymbol{k}} = |\boldsymbol{k}|\boldsymbol{f}_{\boldsymbol{k}}$, so the helicity density of this forcing function satisfies

$$\boldsymbol{f}\cdot\boldsymbol{\nabla}\times\boldsymbol{f} = |\boldsymbol{k}|\boldsymbol{f}^2 > 0 \quad (\text{for } \sigma = 1) \tag{228}$$

at each point in space. We note that since the forcing function is like a delta-function in $\boldsymbol{k}$-space, this means that all points of $\boldsymbol{f}$ 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

$$\boldsymbol{f}_{\boldsymbol{k}} = (\boldsymbol{k}\times\boldsymbol{e})\,/\sqrt{\boldsymbol{k}^2-(\boldsymbol{k}\cdot\boldsymbol{e})^2}. \tag{229}$$

For $0 < |\sigma| < 1$, the forcing function has fractional helicity, where $\sigma \approx \langle\boldsymbol{\omega}\cdot\boldsymbol{u}\rangle\,/(k_{\rm f}\,\langle\boldsymbol{u}^2\rangle)$; see Sect. 4.5 of Ref. [?]. In the code and the forcing_run_pars namelist, σ 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 $k_{\rm f}$, you need to change the file. In the directory '$PENCIL_HOME/samples/helical-MHDturb/K_-VECTORS/' there are several such files prepared:

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 code for many more time

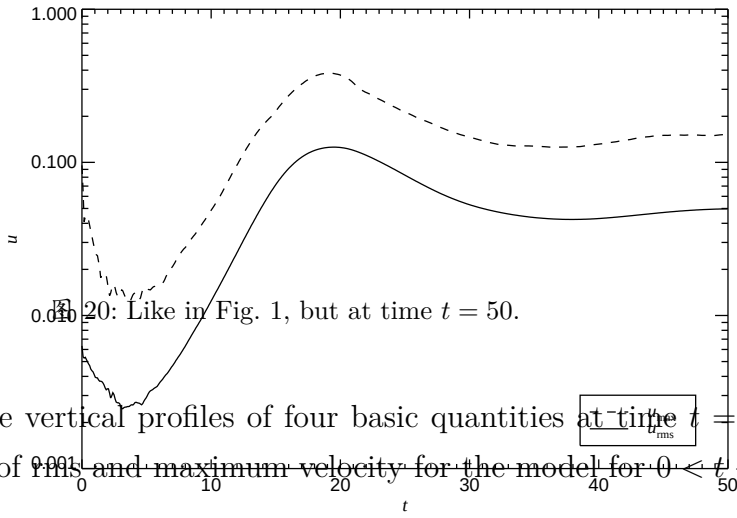


图 20: Like in Fig. 1, but at time $t = 50$.

steps. Figure 20 shows the vertical profiles of four basic quantities at time $t = 50$. Figure 21 shows the time evolution of rms and maximum velocity for the model for $0 < t < 50$.

图 21: 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'.

Figures 22 and 23 show vertical and horizontal sections for time $t = 50$.

G.4 Magnetic helicity in the shearing sheet

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.

Thus, in 'src/Makefile.local' we just use

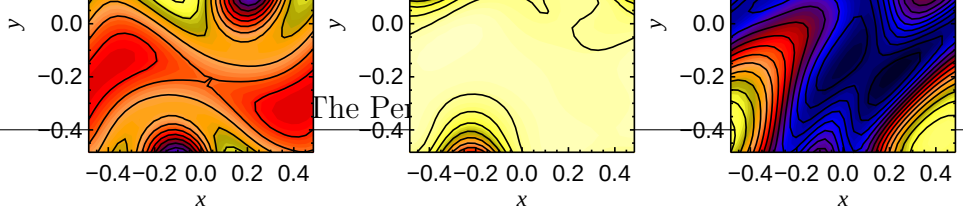


图 22: Horizontal sections for $t = 50$. Top: velocity field. Bottom: entropy (color coded) and density (isocontours). Plots of this type can be produced by running the IDL script 'hsections.pro')

```
  MAGNETIC=magnetic
  HYDRO=nohydro
  EOS=noeos
  DENSITY=nodensity
  SHEAR=shear
  VISCOSITY=noviscosity
```

and put

```
&init_pars
  cvsid='$Id$',
/
&magnetic_init_pars
  initaa='Beltrami-y', amplaa=1.
/
&shear_init_pars
/
```

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
```

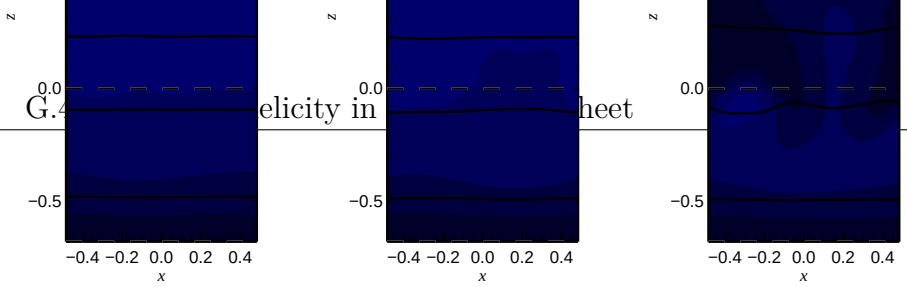


图 23: Vertical section $y = 0.516$ at $t = 50$. Top: velocity field. Bottom: entropy (color coded) and density (isocontours). Plots of this type can be produced by running the IDL scripts 'vsections.pro') or 'vsections2.pro').

```
/
&magnetic_run_pars
  eta=0.
/
&shear_run_pars
  Sshear=-1.
/
```

in 'run.in'. The output includes, among other things

```
  arms(f10.7)
  brms(f12.7)
  jrms(f14.7)
  abm(f14.11)
```

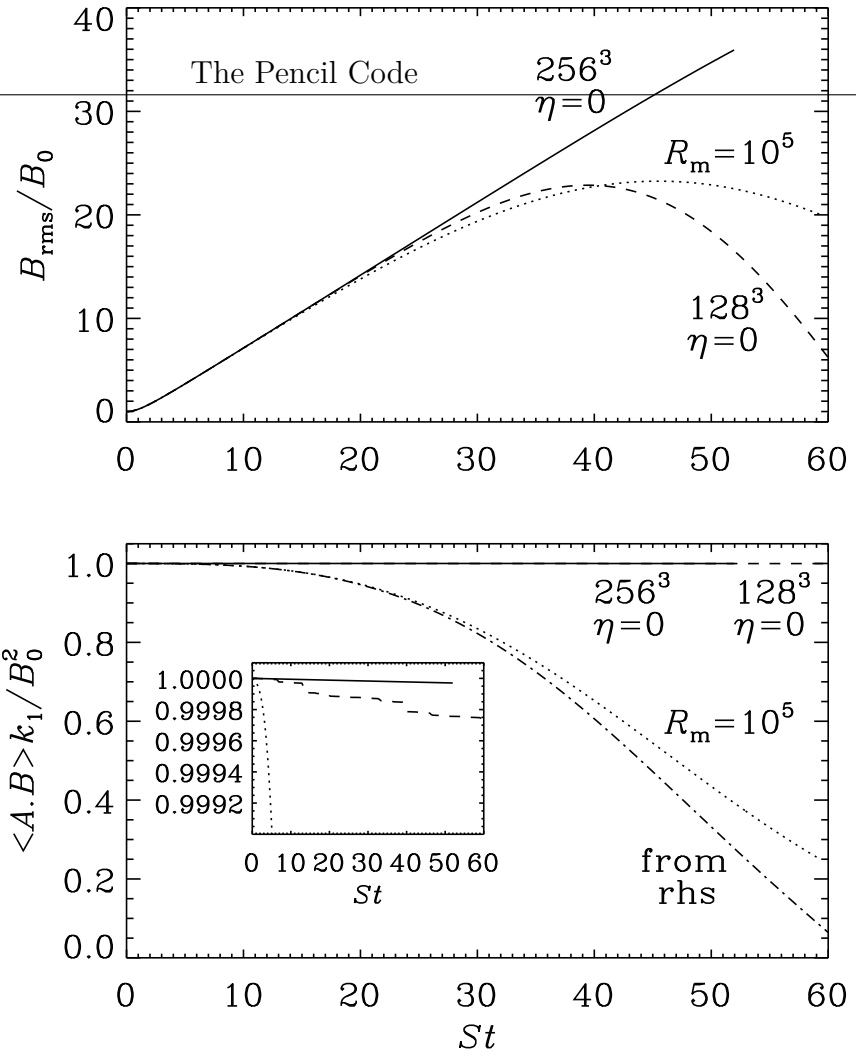


图 24: Wind-up of the magnetic field leads to a linear increase in the rms magnetic field strength until Ohmic diffusion begins to become important (top panel). During this time the magnetic helicity is conserved. With Ohmic diffusion, the decay of $\langle \boldsymbol{A} \cdot \boldsymbol{B} \rangle$ is well described by integrating $-2\eta \langle \boldsymbol{J} \cdot \boldsymbol{B} \rangle$ (indicated by "from rhs" in the second panel).

jbm(f14.7)

The result is shown in Figure 24, 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 $\langle \boldsymbol{A} \cdot \boldsymbol{B} \rangle$ is well described by integrating $-2\eta \langle \boldsymbol{J} \cdot \boldsymbol{B} \rangle$ (indicated by "from rhs" in the second panel).

H 数值方法 Numerical methods

H.1 六阶精度的空间导数 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 [?] and Brandenburg et al. [?] use high order finite difference methods, for example fourth and sixth order compact schemes [?].[18]

六阶精度的一阶, 二阶空间导数由下式给出

$$f'_i = (-f_{i-3} + 9f_{i-2} - 45f_{i-1} + 45f_{i+1} - 9f_{i+2} + f_{i+3})/(60\delta x), \tag{230}$$

$$f''_i = (2f_{i-3} - 27f_{i-2} + 270f_{i-1} - 490f_i + 270f_{i+1} - 27f_{i+2} + 2f_{i+3})/(180\delta x^2), \tag{231}$$

In Fig. 25 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

$$\tfrac{1}{3}f'_{i-1} + f'_i + \tfrac{1}{3}f'_{i+1} = (f_{i-2} - 28f_{i-1} + 28f_{i+1} - f_{i+2})/(36\delta x), \tag{232}$$

for the first derivative, and

$$\tfrac{2}{11}f''_{i-1} + f''_i + \tfrac{2}{11}f''_{i+1} = (3f_{i-2} + 48f_{i-1} - 102f_i + 48f_{i+1} + 3f_{i+2})/(44\delta x^2). \tag{233}$$

for the second derivative. As we have already mentioned in the introduction, this scheme involves obviously solving tridiagonal matrix equations and is therefore effectively nonlocal.

In the second panel of Fig. 25 we have plotted effective wavenumbers for second derivatives, which were calculated as

$$(\cos kx)''_{\rm num} = -k^2_{\rm eff}\cos kx. \tag{234}$$

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 $k^2_{\rm eff}$ 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

[18]The fourth order compact scheme is really identical to calculating derivatives from a cubic spline, as was done in Ref. [?]. In the book by Collatz [?] 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.

图 25: 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.

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 $\boldsymbol{J}$ in the Lorentz force $\boldsymbol{J}\times\boldsymbol{B}$ from a vector potential $\boldsymbol{A}$ using $-\mu_0\boldsymbol{J}=\nabla^2\boldsymbol{A}-\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot\boldsymbol{A}$. This will be important in the MHD calculations presented below.

H.2 **避免振荡的逆风导数** 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.[19] 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

[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 $u_z\propto z_{\rm X}-z$.

derivative operator where the point furthest downwind is excluded from the stencil. For $u > 0$, that means that instead of

$$f'_0 = \frac{-f_{-3} + 9f_{-2} - 45f_{-1} \qquad + 45f_1 - 9f_2 + f_3}{60\,\delta x} - \frac{\delta x^6\, f^{(7)}}{140} = D^{(\mathrm{cent},6)} + O\left(\delta x^6\right) \ , \tag{235}$$

one takes

$$f'_0 = \frac{-2f_{-3} + 15f_{-2} - 60f_{-1} + 20f_0 + 30f_1 - 3f_2}{60\,\delta x} + \frac{\delta x^5\, f^{(6)}}{60} = D^{(\mathrm{up},5)} + O\left(\delta x^5\right) \ . \tag{236}$$

A fourth-order upwind scheme (excluding two downwind points) would be

$$f'_0 = \frac{-f_{-3} + 6f_{-2} - 18f_{-1} + 10f_0 + 3f_1}{12\,\delta x} - \frac{\delta x^4\, f^{(5)}}{20} = D^{(\mathrm{up},4)} + O\left(\delta x^4\right) \ . \tag{237}$$

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

$$[D^{(\mathrm{up},5)} - D^{(\mathrm{cent},6)}]f_0 = \frac{-f_{-3} + 6f_{-2} - 15f_{-1} + 20f_0 - 15f_1 + 6f_2 - f_3}{60\,\delta x} = -\frac{\delta x^5}{60} f_0^{(6)} \ . \tag{238}$$

In other words, 5th-order upwinding can be represented for any sign of u as hyperdiffusion (Dobler et al. 2006):

$$-\,u f'_{(\mathrm{up,5th})} = -u f'_{(\mathrm{centr,6th})} + \frac{|u|\,\delta x^5}{60} f^{(6)} \ . \tag{239}$$

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 (239) 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 α:

$$-\,u f'_{(\mathrm{up,5th},\alpha)} = -u f'_{(\mathrm{centr,6th})} + \alpha\,|u|\,\delta x^5 f^{(6)} \ . \tag{240}$$

If α 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=fac*( &
        270.*( f(l1+1:l2+1,m+1,n,k)-f(l1-1:l2-1,m+1,n,k)  &
```

```
          +f(l1-1:l2-1,m-1,n,k)-f(l1+1:l2+1,m-1,n,k)) &
      - 27.*( f(l1+2:l2+2,m+2,n,k)-f(l1-2:l2-2,m+2,n,k)  &
          +f(l1-2:l2-2,m-2,n,k)-f(l1+2:l2+2,m-2,n,k)) &
      +  2.*( f(l1+3:l2+3,m+3,n,k)-f(l1-3:l2-3,m+3,n,k)  &
          +f(l1-3:l2-3,m-3,n,k)-f(l1+3:l2+3,m-3,n,k)) &
    )
```

and is "visualized" in the left part of Fig. 26. It is way more efficient than the straight-forward approach of first taking the x and the y derivative consecutively. (shown in the right part of Fig. 26).

-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

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

图 26: 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.

Off-diagonal terms enter not only the diffusion terms through $\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot\boldsymbol{u}$ and $\boldsymbol{\nabla}\boldsymbol{\nabla}\cdot\boldsymbol{A}$ terms, but also through the $\boldsymbol{J}=\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\boldsymbol{A}$ operator. The general formula is $J_i=A_{j,ij}-A_{i,jj}$, so in 2-D in the xy-plane we have

$$J_x = A_{x,xx}+A_{y,xy}-A_{x,xx}-A_{x,yy}=A_{y,xy}-A_{x,yy}\ , \tag{241}$$

$$J_y = A_{x,yx}+A_{y,yy}-A_{y,xx}-A_{y,yy}=A_{x,yx}-A_{y,xx} \tag{242}$$

Figure 27 shows how the two schemes perform for the propagation of Alfvén waves,

$$\dot{u}_z = J_xB_{0y}-J_yB_{0x}\ , \tag{243}$$

$$\dot{A}_x = -u_zB_{0y}\ , \tag{244}$$

$$\dot{A}_y = +u_zB_{0x}\ . \tag{245}$$

The initial condition (as implemented in subroutine alfven_xy) is

$$u_z \sim \cos(k_xx+k_yy-\omega t)\ , \tag{246}$$

$$A_x \sim +B_{0y}\sin(k_xx+k_yy-\omega t)/\omega\ , \tag{247}$$

$$A_y \sim -B_{0x}\sin(k_xx+k_yy-\omega t)/\omega\ , \tag{248}$$

where $\omega=\boldsymbol{k}\cdot\boldsymbol{B}_0$. The figure shows that there is no clear advantage of either scheme, so the code uses the more efficient bidiagonal one.

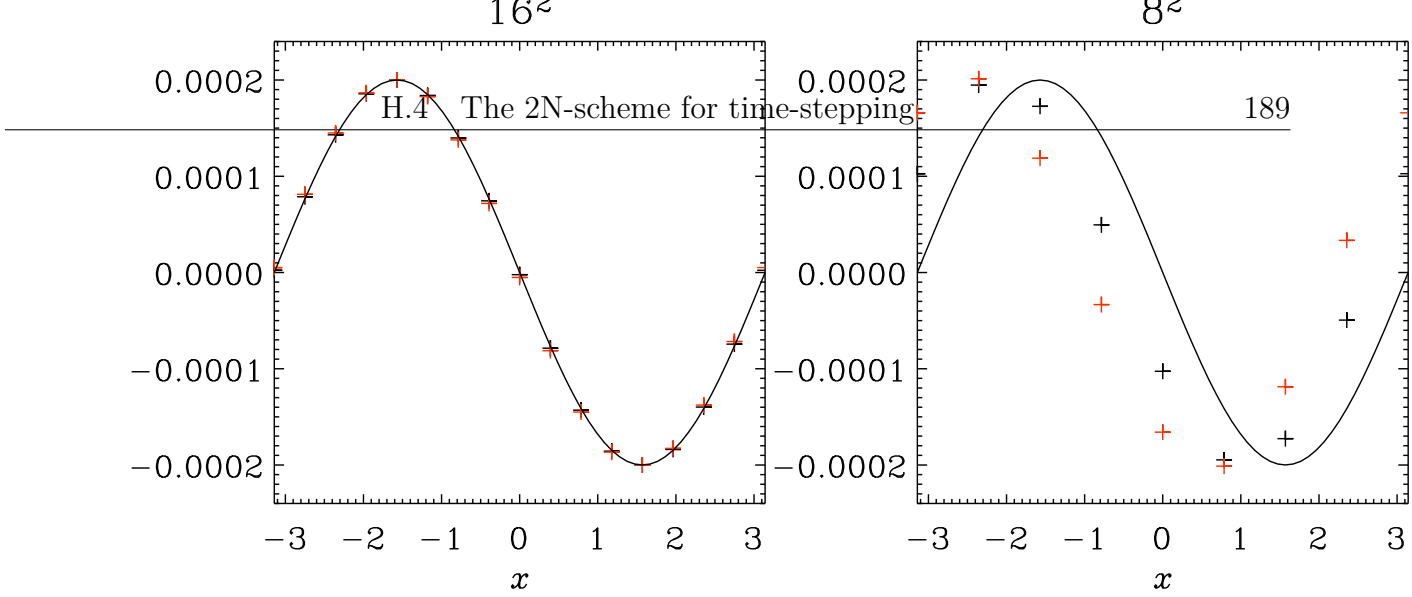


图 27: Alfvén wave for $\boldsymbol{B}_0 = (1, 2, 0)$ and $\boldsymbol{k} = (1, 2, 0)$ after $t = 2\pi$. The wave travels in the direction of $\boldsymbol{k}$. Red symbols are for the bidiagonal scheme, black symbols show results obtained using the consecutive scheme. Already for 16^2 mesh points there 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).

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 [?, ?]. Consider the ordinary differential equation (ODE)

$$\dot{u} = F(u, t) \ , \tag{249}$$

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

$$u^{(n)} \equiv u(t_n) \tag{250}$$

according to the iterative procedure

$$w_i = \alpha_i w_{i-1} + \delta t \, F(u_{i-1}, t_{i-1}) \ , \tag{251}$$

$$u_i = u_{i-1} + \beta_i w_i \ . \tag{252}$$

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. (252)

$$u_0 = u^{(n)} \quad \text{and, after the last step,} \quad u^{(n+1)} = u_3, \tag{253}$$

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, α_2, α_3, β_1, β_2, and β_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 [?]. 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. [?] the accuracy of some nonlinear equations is tested.

表 17: Coefficients for different $2N$-type third-order Runge-Kutta schemes. The coefficients c_i (which are determined by the α_i, β_i) give the time for each substep, $t_i = t_0 + c_i \delta t$

scheme	c_1	c_2	c_3	α_2	α_3	β_1	β_2	β_3
symmetric	0	1/3	2/3	−2/3	−1	1/3	1	1/2
[predictor/corrector]	0	1/2	1	−1/4	−4/3	1/2	2/3	1/2
inhomogeneous	0	15/32	4/9	−17/32	−32/27	1/4	8/9	3/4
Williamson (1980)	0	4/9	15/32	−5/9	−153/128	1/3	15/16	8/15

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

$$\left(\frac{\mathrm{d}f}{\mathrm{d}t}\right)_{n\text{th order}} = \left(\frac{\mathrm{d}f}{\mathrm{d}t}\right)_{\text{exact}} + a_n \delta t^n \left(\frac{\mathrm{d}^{n+1} f}{\mathrm{d}t^{n+1}}\right) + ..., \tag{254}$$

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 $a1\delta t(\boldsymbol{u} \cdot \boldsymbol{\nabla})^2$ for $n = 1$ and a hyperdiffusive error equal to $a3\delta t^3(\boldsymbol{u} \cdot \boldsymbol{\nabla})^4$ for $n = 3$. Substituting $\delta t = c_{\rm CFL}\delta x/|\boldsymbol{u}|$, where $c_{\rm CFL}$ is Courant-Friedrich-Lewy constant, we have a diffusive error $\nu\nabla^2$ with negative $\nu = -a1c_{\rm CFL}|\boldsymbol{u}|\delta x$ for $n = 1$, and a hyperdiffusive error $-\nu_{\rm hyp}\nabla^4$ with positive $\nu_{\rm hyp} = a_3 c_{\rm CFL}^3 |\boldsymbol{u}|\delta x^3$ 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 $|\boldsymbol{u}|\delta x/12$ that is $1/(12a_3c_{\rm CFL}^3) \approx 5.8$ 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 ∇^2. Therefore we define an effective viscosity as $\nu_{\rm eff} = \nu_{\rm hyp} k_{\rm Ny}^2$ with $k_{\rm Ny} = \pi/\delta x$ 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 ${\rm Re}_{\rm Ny} = |\boldsymbol{u}|/\nu_{\rm eff} k_{\rm Ny}$, and find ${\rm Re} = -24/(\pi c_{\rm CFL})^3 \approx 2.3$ for our favorite choice $c_{\rm CFL} = 0.7$. 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 $k_{\rm Ny}$, 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

$$\frac{s_i}{s_0} = x_i \left(\ln \left[\frac{1}{x_{\rm tot}} \frac{\rho_i}{\rho} \left(\frac{T}{T_0} \right)^{3/2} \right] + \frac{5}{2} \right) \ , \tag{255}$$

where

$$x_{\rm H} = 1 - y \ , \quad x_{\rm e} = x_{\rm p} = y \ , \quad x_{\rm tot} = 1 + y + x_{\rm He} \tag{256}$$

$$s_0 = \frac{k_{\rm B}}{\mu m_{\rm H}} \ , \quad T_0 = \frac{\chi_{\rm H}}{k_{\rm B}} \ , \tag{257}$$

and

$$\rho_i = \mu m_{\rm H} \left(\frac{m_i \chi_{\rm H}}{2\pi\hbar^2} \right)^{3/2} \ . \tag{258}$$

The specific entropy of mixing is

$$\frac{s_{\rm mix}}{s_0} = -\sum_i x_i \ln \frac{x_i}{x_{\rm tot}} \ . \tag{259}$$

Summing up everything, we get the total specific entropy

$$\frac{s}{s_0} = \sum_i \frac{s_i}{s_0} + \frac{s_{\rm mix}}{s_0} = \sum_i x_i \left(\ln \left[\frac{1}{x_i} \frac{\rho_i}{\rho} \left(\frac{T}{T_0} \right)^{3/2} \right] + \frac{5}{2} \right) \tag{260}$$

$$= \sum_i x_i \ln \frac{\rho_i}{x_i} + x_{\rm tot} \left(\ln \left[\frac{1}{\rho} \left(\frac{T}{T_0} \right)^{3/2} \right] + \frac{5}{2} \right) \ . \tag{261}$$

Solving for T gives

$$\frac{3}{2} \ln \frac{T}{T_0} = \frac{s/s_0 + \sum_i x_i \ln x_i/\rho_i}{x_{\rm tot}} + \ln \rho - \frac{5}{2} \ . \tag{262}$$

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

$$F = \ln\left[\frac{\rho_{\rm e}}{\rho}\left(\frac{T}{T_0}\right)^{3/2}\frac{1-y}{y^2}\right] - \frac{T_0}{T} \ . \tag{263}$$

The derivative with respect to y for Newton-Raphson is

$$\frac{\partial F}{\partial y} = \left(\frac{3}{2}+\frac{T_0}{T}\right)\frac{\partial \ln T/T_0}{\partial y} - \frac{1}{1-y} - \frac{2}{y} \ , \tag{264}$$

where

$$\frac{\partial \ln T/T_0}{\partial y} = \frac{\frac{2}{3}\left(\ln\rho_{\rm H}/\rho_{\rm p} - F - T_0/T\right) - 1}{1+y+x_{\rm He}} \ . \tag{265}$$

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:

$$\frac{\partial \ln P}{\partial \ln\rho} = \frac{1}{1+y+x_{\rm He}}\frac{\partial y}{\partial \ln\rho} + \frac{\partial \ln T}{\partial \ln\rho} + \frac{\partial \ln T}{\partial y}\frac{\partial y}{\partial \ln\rho} + 1, \tag{266}$$

$$\frac{\partial \ln P}{\partial s} = \frac{1}{1+y+x_{\rm He}}\frac{\partial y}{\partial s} + \frac{\partial \ln T}{\partial s} + \frac{\partial \ln T}{\partial y}\frac{\partial y}{\partial s}. \tag{267}$$

Since $F=0$ for all desired solutions $(y, \ln\rho, s)$ we also have

$$dF = \frac{\partial F}{\partial \ln\rho}{\rm d}\ln\rho + \frac{\partial F}{\partial s}{\rm d}s + \frac{\partial F}{\partial y}{\rm d}y = 0 \ , \tag{268}$$

and thus

$$\frac{\partial y}{\partial \ln\rho} = \left(\frac{{\rm d}y}{{\rm d}\ln\rho}\right)_{{\rm d}s=0} = -\frac{\partial F/\partial \ln\rho}{\partial F/\partial y} \tag{269}$$

and

$$\frac{\partial y}{\partial s} = \left(\frac{{\rm d}y}{{\rm d}s}\right)_{{\rm d}\ln\rho=0} = -\frac{\partial F/\partial s}{\partial F/\partial y}. \tag{270}$$

H.7 Radiative transfer

H.7.1 Solving the radiative transfer equation

A formal solution of Eq. (81) is given by

$$I(\tau) = I(\tau_0)e^{-(\tau-\tau_0)} + \int_{\tau_0}^{\tau} e^{-(\tau-\tau')}S(\tau')\,d\tau' \ . \tag{271}$$

Using a generalization of the trapezoidal rule,

$$\int_{\tau_0}^{\tau} e^{-(\tau-\tau')}f(\tau')\,d\tau' \approx \int_{\tau_0}^{\tau} e^{-(\tau-\tau')}\left[f(\tau_0) + \frac{f(\tau)-f(\tau_0)}{\tau-\tau_0}(\tau'-\tau_0)\right]\,d\tau' \tag{272}$$

$$= \left[1-e^{-(\tau-\tau_0)}\right]f(\tau) - \frac{1-e^{-(\tau-\tau_0)}(1+\tau-\tau_0)}{\tau-\tau_0}[f(\tau)-f(\tau_0)] \ , \tag{273}$$

which is exact for linear functions $S(\tau)$, we discretize this as

$$I_{k+1} = I_k e^{-\delta\tau} + (1-e^{-\delta\tau})S_{k+1} - \frac{1-e^{-\delta\tau}(1+\delta\tau)}{\delta\tau}(S_{k+1}-S_k)\ , \tag{274}$$

$$= I_k e^{-\delta\tau} + (1-e^{-\delta\tau})S_k + \frac{e^{-\delta\tau}-1+\delta\tau}{\delta\tau}(S_{k+1}-S_k)\ . \tag{275}$$

Here the simplest way to calculate $\delta\tau$ is

$$\delta\tau = \frac{\chi_k+\chi_{k+1}}{2}\,\delta x\ ; \tag{276}$$

more accurate alternatives are

$$\delta\tau = \sqrt{\chi_k\chi_{k+1}}\,\delta x \tag{277}$$

or

$$\delta\tau = \frac{\chi_{k+1}-\chi_k}{\ln\frac{\chi_{k+1}}{\ln\chi_k}}\,\delta x \tag{278}$$

H.7.2 Angular integration

表 18: Sums $\sqrt{4\pi}Y_l^m(\theta_i,\phi_i)$ 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.

Directions	Y_0^0	Y_2^0	Y_4^0	$Y_4^{\pm4}$	Y_6^0	$Y_6^{\pm4}$	Y_8^0	$Y_8^{\pm4}$	$Y_8^{\pm8}$
Coord.	6	0	$\frac{21}{2}$	$\frac{3}{4}\sqrt{70}$	$\frac{3}{4}\sqrt{13}$	$-\frac{3}{8}\sqrt{182}$	$\frac{99}{32}\sqrt{17}$	$\frac{3}{32}\sqrt{2618}$	$\frac{3}{64}\sqrt{24310}$
Face diag.	12	0	$-\frac{21}{4}$	$-\frac{3}{8}\sqrt{70}$	$-\frac{39}{16}\sqrt{13}$	$\frac{39}{32}\sqrt{182}$	$\frac{891}{256}\sqrt{17}$	$\frac{27}{256}\sqrt{2618}$	$\frac{27}{512}\sqrt{24310}$
Space diag.	8	0	$-\frac{28}{3}$	$-\frac{2}{3}\sqrt{70}$	$\frac{16}{9}\sqrt{13}$	$-\frac{8}{9}\sqrt{182}$	$\frac{11}{9}\sqrt{17}$	$\frac{1}{27}\sqrt{2618}$	$\frac{1}{54}\sqrt{24310}$
Non-h. f-d.	8	$2\sqrt{5}$	$-\frac{39}{4}$	$\frac{3}{8}\sqrt{70}$	$-\frac{19}{16}\sqrt{13}$	$\frac{27}{32}\sqrt{182}$	$\frac{611}{256}\sqrt{17}$	$\frac{51}{256}\sqrt{2618}$	$\frac{3}{512}\sqrt{24310}$
Coord. x, y	4	$-2\sqrt{5}$	$\frac{9}{2}$	$\frac{4}{3}\sqrt{70}$	$-\frac{5}{4}\sqrt{13}$	$-\frac{3}{8}\sqrt{182}$	$\frac{35}{32}\sqrt{17}$	$\frac{3}{32}\sqrt{2618}$	$\frac{3}{64}\sqrt{24310}$
Coord. z	2	$2\sqrt{5}$	6	0	$2\sqrt{13}$	0	$2\sqrt{17}$	0	0

For angular integration over the full solid angle, we make the ansatz

$$\int_{4\pi} f(\theta,\phi)\,\frac{d\omega}{4\pi} = \sum_{i=1}^{N} w_i f(\theta_i,\phi_i) + R_N\ . \tag{279}$$

Table 18 shows the sums $\sqrt{4\pi}Y_l^m(\theta_i,\phi_i)$ over special sets of directions (θ_i,ϕ_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 [?], §25.4.65).[20]

Cooling times have been determined numerically in [?]. Comparing with analytic expressions obtained in the Eddington approximation, the proposed suitable switches in 1-D and 2-D problems.

20

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

Coordinate axes x, y:	12/215
Coordinate axes z:	10/129
Non-hor. face diagonals:	-14/645
Space diagonals:	171/1720

$l_{\max} = 5$

I Curvilinear coordinates

The use and implementation of non-cartesian coordinate systems is briefly explained in Section 5.21. All differential operators look like their cartesian counterparts, except that all derivatives are now replaced by covariant derivatives. The relevant reference for the Pencil Code is [?]; see their Appendix B. Here some details.

I.1 Covariant derivatives

$$A_{\hat{\alpha};\hat{\beta}} = A_{\hat{\alpha},\hat{\beta}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\beta}}\, A_{\hat{\sigma}}, \tag{280}$$

$$A_{\hat{\alpha}\hat{\beta};\hat{\gamma}} = A_{\hat{\alpha}\hat{\beta},\hat{\gamma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\gamma}}\, A_{\hat{\sigma}\hat{\beta}} - \Gamma^{\hat{\sigma}}{}_{\hat{\beta}\hat{\gamma}}\, A_{\hat{\alpha}\hat{\sigma}}. \tag{281}$$

Second derivative tensor

$$\begin{aligned} A_{\hat{\alpha};\hat{\beta}\hat{\gamma}} &= A_{\hat{\alpha};\hat{\beta},\hat{\gamma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\gamma}}\, A_{\hat{\sigma};\hat{\beta}} - \Gamma^{\hat{\sigma}}{}_{\hat{\beta}\hat{\gamma}}\, A_{\hat{\alpha};\hat{\sigma}} \\ &= A_{\hat{\alpha},\hat{\beta}\hat{\gamma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\beta}}\, A_{\hat{\sigma},\hat{\gamma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\beta},\hat{\gamma}}\, A_{\hat{\sigma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\gamma}}\, A_{\hat{\sigma},\hat{\beta}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\gamma}}\Gamma^{\hat{\nu}}{}_{\hat{\sigma}\hat{\beta}}\, A_{\hat{\nu}} - \Gamma^{\hat{\sigma}}{}_{\hat{\beta}\hat{\gamma}}\, A_{\hat{\alpha},\hat{\sigma}} + \Gamma^{\hat{\sigma}}{}_{\hat{\beta}\hat{\gamma}}\Gamma^{\hat{\nu}}{}_{\hat{\alpha}\hat{\sigma}}\, A_{\hat{\nu}}. \end{aligned}$$

Elements of the first derivative tensor

$$A_{\hat{r};\hat{r}} = A_{\hat{r},\hat{r}}, \qquad A_{\hat{\theta};\hat{\theta}} = A_{\hat{\theta},\hat{\theta}} + r^{-1}\, A_{\hat{r}}, \qquad A_{\hat{\phi};\hat{\phi}} = A_{\hat{\phi},\hat{\phi}} + r^{-1}\, A_{\hat{r}} + r^{-1}\cot\theta\, A_{\hat{\theta}}. \tag{282}$$

$$\begin{aligned} A_{\hat{\phi};\hat{\theta}} &= A_{\hat{\phi},\hat{\theta}} & A_{\hat{\theta};\hat{\phi}} &= A_{\hat{\theta},\hat{\phi}} - r^{-1}\cot\theta\, A_{\hat{\phi}} \\ A_{\hat{r};\hat{\phi}} &= A_{\hat{r},\hat{\phi}} - r^{-1}\, A_{\hat{\phi}} & A_{\hat{\phi};\hat{r}} &= A_{\hat{\phi},\hat{r}} \\ A_{\hat{\theta};\hat{r}} &= A_{\hat{\theta},\hat{r}} & A_{\hat{r};\hat{\theta}} &= A_{\hat{r},\hat{\theta}} - r^{-1}\, A_{\hat{\theta}} \end{aligned} \tag{283}$$

I.2 Differential operators

All differential operators look like their cartesian counterparts, except that all derivatives are now replaced by covariant derivatives.

I.2.1 Gradient

For the gradient operator the covariant and partial derivatives are the same, i.e.

$$\boldsymbol{\nabla}\Psi = \Psi_{;\hat{\alpha}} = \Psi_{,\hat{\alpha}} = \begin{pmatrix} \partial_{\hat{r}}\Psi \\ \partial_{\hat{\theta}}\Psi \\ \partial_{\hat{\phi}}\Psi \end{pmatrix} \tag{284}$$

where

$$\partial_{\hat{r}} = \partial_r \tag{285}$$

$$\partial_{\hat{\theta}} = r^{-1}\partial_\theta \tag{286}$$

$$\partial_{\hat{\phi}} = \varpi^{-1}\partial_\phi \tag{287}$$

and $\varpi = r\sin\theta$ is the cylindrical radius. Thus,

$$\boldsymbol{\nabla}\Psi = (\boldsymbol{\nabla}\Psi)^{(0)}\,, \tag{288}$$

where the superscript (0) indicated the straightforward calculation in the non-coordinate basis. The coordinate and non-coordinate bases are related to each other via

$$(\boldsymbol{\nabla}\Psi)^0 \equiv \begin{pmatrix} \Psi_{,\hat{r}} \\ \Psi_{,\hat{\theta}} \\ \Psi_{,\hat{\phi}} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & r^{-1} & 0 \\ 0 & 0 & \varpi^{-1} \end{pmatrix} (\boldsymbol{\nabla}\Psi)^{(\mathrm{coord})}\,. \tag{289}$$

Here, the result in the coordinate basis is just what one would get by computing as if one had cartesian coordinates. In the Pencil Code the output or the subroutine der is now in this non-coordinate basis.

I.2.2 Divergence

For the divergence operator we have a 'correction term', i.e.

$$\boldsymbol{\nabla}\cdot\boldsymbol{A} = A^{\hat{\alpha}}{}_{;\hat{\alpha}} \tag{290}$$

$$= A^{\hat{\alpha}}{}_{,\hat{\alpha}} + \Gamma^{\hat{\alpha}}{}_{\hat{\beta}\hat{\alpha}} A^{\hat{\beta}}, \tag{291}$$

where the only non-vanishing contributions to the last term are

$$\Gamma^{\hat{\alpha}}{}_{\hat{\beta}\hat{\alpha}} A^{\hat{\beta}} = \Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}} A^{\hat{r}} + \Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}} A^{\hat{r}} + \Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}} A^{\hat{\theta}} \tag{292}$$

$$= 2r^{-1}A^{\hat{r}} + r^{-1}\cot\theta\; A^{\hat{\theta}}. \tag{293}$$

Thus,

$$\boldsymbol{\nabla}\cdot\boldsymbol{A} = (\boldsymbol{\nabla}\cdot\boldsymbol{A})^{(0)} + M^{(\mathrm{div})}_{\hat{\alpha}} A_{\hat{\alpha}}, \tag{294}$$

where

$$M^{(\mathrm{div})}_{\hat{\alpha}} = \begin{pmatrix} 2r^{-1} \\ r^{-1}\cot\theta \\ 0 \end{pmatrix} \tag{295}$$

represents the correction term.

Alternatively:

$$A_{\hat{\alpha};\hat{\alpha}} = A_{\hat{\alpha},\hat{\alpha}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha}}\, A_{\hat{\sigma}} \tag{296}$$

$$= A_{\hat{\alpha},\hat{\alpha}} + 2r^{-1}A_{\hat{r}} + r^{-1}\cot\theta\; A_{\hat{\theta}}. \tag{297}$$

I.2.3 Curl

The curl operator is given by

$$(\boldsymbol{\nabla}\times\boldsymbol{B})^{\hat{\alpha}}=\epsilon^{\hat{\alpha}\hat{\beta}\hat{\gamma}}B_{\hat{\gamma},\hat{\beta}}. \tag{298}$$

So

$$\boldsymbol{\nabla}\times\boldsymbol{A}=\begin{pmatrix}A_{\hat{\phi};\hat{\theta}}-A_{\hat{\theta};\hat{\phi}}\\ A_{\hat{r};\hat{\phi}}-A_{\hat{\phi};\hat{r}}\\ A_{\hat{\theta};\hat{r}}-A_{\hat{r};\hat{\theta}}\end{pmatrix} \tag{299}$$

$$\boldsymbol{\nabla}\times\boldsymbol{A}=\begin{pmatrix}A_{\hat{\phi},\hat{\theta}}-A_{\hat{\theta},\hat{\phi}}+\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}A^{\hat{\phi}}\\ A_{\hat{r},\hat{\phi}}-A_{\hat{\phi},\hat{r}}-\Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}A^{\hat{\phi}}\\ A_{\hat{\theta},\hat{r}}-A_{\hat{r},\hat{\theta}}+\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}}A^{\hat{\theta}}\end{pmatrix} \tag{300}$$

Thus,

$$\boldsymbol{\nabla}\times\boldsymbol{A}=(\boldsymbol{\nabla}\times\boldsymbol{A})^{(0)}+M^{(\mathrm{curl})}_{\hat{\alpha}\hat{\beta}}A_{\hat{\beta}}, \tag{301}$$

where

$$M^{(\mathrm{curl})}_{\hat{\alpha}\hat{\beta}}=\begin{pmatrix}0 & 0 & r^{-1}\cot\theta\\ 0 & 0 & -r^{-1}\\ 0 & r^{-1} & 0\end{pmatrix} \tag{302}$$

is the correction term. In the Pencil Code we use the subroutine curl_mn(aij,bb,aa), which uses aij= $A_{\hat{\alpha},\hat{\beta}}$ and aa= $A_{\hat{\alpha}}$ as input pencils and produces bb= $B_{\hat{\alpha}}$ as output.

I.2.4 Advection operator

[The usage of roman indices here is insignificant.]

$$\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}=\boldsymbol{\omega}\times\boldsymbol{u}+\tfrac{1}{2}\boldsymbol{\nabla}\boldsymbol{u}^2=-\boldsymbol{u}\times\boldsymbol{\omega}+\tfrac{1}{2}\boldsymbol{\nabla}\boldsymbol{u}^2 \tag{303}$$

$$\begin{aligned}(\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u})_i &= -\epsilon_{ijk}\epsilon_{klm}u_ju_{m;l}+u_ju_{j,i}\\ &= (-\delta_{il}\delta_{jm}+\delta_{im}\delta_{jl})u_ju_{m;l}+u_ju_{j,i}\\ &= -u_ju_{j;i}+u_ju_{i;j}+u_ju_{j,i}\\ &= u_ju_{i;j}+\Gamma^{k}{}_{ji}u_ju_k\\ &= u_ju_{i,j}+(-\Gamma^{k}{}_{ij}+\Gamma^{k}{}_{ji})u_ju_k\end{aligned} \tag{304}$$

Note that the terms with $\Gamma^{\hat{r}}{}_{\hat{\theta}\hat{\theta}}$, $\Gamma^{\hat{r}}{}_{\hat{\phi}\hat{\phi}}$, and $\Gamma^{\hat{\theta}}{}_{\hat{\phi}\hat{\phi}}$ drop out. Thus, we have

$$\begin{aligned}(\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u})_{\hat{r}} &= u_ju_{\hat{r},j}+(-\Gamma^{k}{}_{\hat{r}j}+\Gamma^{k}{}_{j\hat{r}})\,u_ju_k\\ &= u_ju_{\hat{r},j}-\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}}\,u^2_{\hat{\theta}}-\Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}\,u^2_{\hat{\phi}}\\ &= u_ju_{\hat{r},j}-r^{-1}(u^2_{\hat{\theta}}+u^2_{\hat{\phi}})\end{aligned} \tag{305}$$

$$\begin{aligned}(\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u})_{\hat\theta} &= u_j u_{\hat\theta,j} + (-\Gamma^k{}_{\hat\theta j} + \Gamma^k{}_{j\hat\theta})\, u_j u_k \\ &= u_j u_{\hat\theta,j} - \Gamma^{\hat\phi}{}_{\hat\theta\hat\phi}\, u^2_{\hat\phi} + \Gamma^{\hat\theta}{}_{\hat r\hat\theta}\, u_{\hat r} u_{\hat\theta} \\ &= u_j u_{\hat\theta,j} - r^{-1}\cot\theta\, u^2_{\hat\phi} + r^{-1} u_{\hat r} u_{\hat\theta} \end{aligned} \tag{306}$$

$$\begin{aligned}(\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u})_{\hat\phi} &= u_j u_{\hat\phi,j} + (-\Gamma^k{}_{\hat\phi j} + \Gamma^k{}_{j\hat\phi})\, u_j u_k \\ &= u_j u_{\hat\phi,j} + \Gamma^{\hat\phi}{}_{\hat r\hat\phi}\, u_{\hat r} u_{\hat\phi} + \Gamma^{\hat\phi}{}_{\hat\theta\hat\phi}\, u_{\hat\theta} u_{\hat\phi} \\ &= u_j u_{\hat\phi,j} + r^{-1} u_{\hat r} u_{\hat\phi} + r^{-1}\cot\theta\, u_{\hat\theta} u_{\hat\phi} \end{aligned} \tag{307}$$

Note that the formulation above is slightly misleading, because

$$\Gamma^k{}_{j\hat\theta}\, u_j u_k = \Gamma^{\hat r}{}_{\hat\theta\hat\theta}\, u_{\hat\theta} u_{\hat r} + \Gamma^{\hat\theta}{}_{\hat r\hat\theta}\, u_{\hat r} u_{\hat\theta} = 0 \tag{308}$$

$$\Gamma^k{}_{j\hat\phi}\, u_j u_k = \Gamma^{\hat r}{}_{\hat\phi\hat\phi}\, u_{\hat\phi} u_{\hat r} + \Gamma^{\hat\phi}{}_{\hat r\hat\phi}\, u_{\hat r} u_{\hat\phi} + \Gamma^{\hat\theta}{}_{\hat\phi\hat\phi}\, u_{\hat\phi} u_{\hat\theta} + \Gamma^{\hat\phi}{}_{\hat\theta\hat\phi}\, u_{\hat\theta} u_{\hat\phi} = 0 \tag{309}$$

I.2.5 Mixed advection operator

$$\boldsymbol{u}\times\boldsymbol{B} = \boldsymbol{u}\times\boldsymbol{\nabla}\boldsymbol{B} = u_j \epsilon_{ijk}\epsilon_{klm} A_{m;l} = u_j(\delta_{il}\delta_{jm} - \delta_{im}\delta_{jl}) A_{m;l} = u_j A_{j;i} - u_j A_{i;j} \tag{310}$$

I.2.6 Shear term

$$u_j A_{j;i} = (u_j A_j)_{;i} - u_{j;i} A_j = (u_j A_j)_{,i} - u_{j;i} A_j \tag{311}$$

$$u_{j;i} A_j = u_{j,i} A_j - \Gamma^k_{ji} u_k A_j \tag{312}$$

So

$$u_{\hat\phi;\hat r} A_{\hat\phi} = u_{\hat\phi,\hat r} A_{\hat\phi} - \Gamma^k_{\hat\phi\hat r} u_k A_{\hat\phi} = u_{\hat\phi,\hat r} A_{\hat\phi} \tag{313}$$

$$u_{\hat r;\hat\phi} A_{\hat r} = u_{\hat r,\hat\phi} A_{\hat r} - \Gamma^{\hat\phi}_{\hat r\hat\phi} u_{\hat\phi} A_{\hat r} \tag{314}$$

I.2.7 Another mixed advection operator

$$u_{\hat\beta} A_{\hat\alpha;\hat\beta} = u_{\hat\beta} A_{\hat\alpha,\hat\beta} - \Gamma^{\hat\sigma}{}_{\hat\alpha\hat\beta}\, u_{\hat\beta} A_{\hat\sigma}, \tag{315}$$

Write out

$$\begin{aligned}(\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{A})_{\hat r} = u_{\hat r} A_{\hat r;\hat r} + u_{\hat\theta} A_{\hat r;\hat\theta} + u_{\hat\phi} A_{\hat r;\hat\phi} &= u_{\hat r} A_{\hat r,\hat r}, + u_{\hat\theta} A_{\hat r,\hat\theta} - r^{-1}\, u_{\hat\theta} A_{\hat\theta} + u_{\hat\phi} A_{\hat r,\hat\phi} - r^{-1}\, u_{\hat\phi} A_{\hat\phi} \\ (\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{A})_{\hat\theta} = u_{\hat r} A_{\hat\theta;\hat r} + u_{\hat\theta} A_{\hat\theta;\hat\theta} + u_{\hat\phi} A_{\hat\theta;\hat\phi} &= u_{\hat r} A_{\hat\theta,\hat r} + u_{\hat\theta} A_{\hat\theta,\hat\theta} + r^{-1}\, u_{\hat\theta} A_{\hat r}, + u_{\hat\phi} A_{\hat\theta,\hat\phi} - r^{-1}\cot\theta\, u_{\hat\phi} A_{\hat\phi} \\ (\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{A})_{\hat\phi} = u_{\hat r} A_{\hat\phi;\hat r} + u_{\hat\theta} A_{\hat\phi;\hat\theta} + u_{\hat\phi} A_{\hat\phi;\hat\phi} &= u_{\hat r} A_{\hat\phi,\hat r} + u_{\hat\theta} A_{\hat\phi,\hat\theta} + u_{\hat\phi} A_{\hat\phi,\hat\phi} + r^{-1}\, u_{\hat\phi} A_{\hat r} + r^{-1}\cot\theta\, u_{\hat\phi} A_{\hat\theta}. \end{aligned}$$

Reorder

$$\begin{aligned}(\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{A})_{\hat r} &= u_{\hat r} A_{\hat r,\hat r} + u_{\hat\theta} A_{\hat r,\hat\theta} + u_{\hat\phi} A_{\hat r,\hat\phi} - r^{-1}\, u_{\hat\theta} A_{\hat\theta} - r^{-1}\, u_{\hat\phi} A_{\hat\phi} \\ (\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{A})_{\hat\theta} &= u_{\hat r} A_{\hat\theta,\hat r} + u_{\hat\theta} A_{\hat\theta,\hat\theta} + u_{\hat\phi} A_{\hat\theta,\hat\phi} + r^{-1}\, u_{\hat\theta} A_{\hat r} - r^{-1}\cot\theta\, u_{\hat\phi} A_{\hat\phi} \\ (\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{A})_{\hat\phi} &= u_{\hat r} A_{\hat\phi,\hat r} + u_{\hat\theta} A_{\hat\phi,\hat\theta} + u_{\hat\phi} A_{\hat\phi,\hat\phi} + r^{-1}\, u_{\hat\phi} A_{\hat r} + r^{-1}\cot\theta\, u_{\hat\phi} A_{\hat\theta}. \end{aligned} \tag{316}$$

I.2.8 Strain Matrix

The strain matrix takes the form $2\mathrm{s}_{\hat{\alpha}\hat{\beta}} = u_{\hat{\alpha};\hat{\beta}} + u_{\hat{\beta};\hat{\alpha}}$. The components of the covariant derivative tensor were given in Eqs. (282) and (283), so the final result reads

$$2\mathrm{s} \equiv \begin{pmatrix} 2u_{\hat{r},\hat{r}} & u_{\hat{r},\hat{\theta}} + u_{\hat{\theta},\hat{r}} - u_{\hat{\theta}}/r & u_{\hat{r},\hat{\phi}} + u_{\hat{\phi},\hat{r}} - u_{\hat{\phi}}/r \\ u_{\hat{r},\hat{\theta}} + u_{\hat{\theta},\hat{r}} - u_{\hat{\theta}}/r & 2u_{\hat{\theta},\hat{\theta}} + 2u_{\hat{r}}/r & u_{\hat{\theta},\hat{\phi}} + u_{\hat{\phi},\hat{\theta}} - u_{\hat{\phi}}\frac{\cot\theta}{r} \\ u_{\hat{r},\hat{\phi}} + u_{\hat{\phi},\hat{r}} - u_{\hat{\phi}}/r & u_{\hat{\theta},\hat{\phi}} + u_{\hat{\phi},\hat{\theta}} - u_{\hat{\phi}}\frac{\cot\theta}{r} & 2u_{\hat{\phi},\hat{\phi}} + 2u_{\hat{r}}/r + 2u_{\hat{\theta}}\frac{\cot\theta}{r} \end{pmatrix} \tag{317}$$

The trace-less rate of strain matrix is given by

$$\mathrm{S}_{ij} = \mathrm{s}_{ij} - \tfrac{1}{3}\delta_{ij}\boldsymbol{\nabla}\cdot\boldsymbol{u}. \tag{318}$$

I.2.9 Lambda effect

$$Q_{ij} = \begin{pmatrix} 0 & 0 & \Lambda_V\Omega \\ 0 & 0 & \Lambda_H\Omega \\ \Lambda_V\Omega & \Lambda_H\Omega & 0 \end{pmatrix} \tag{319}$$

where $\Omega = u_{\hat{\phi}}/r\sin\theta$. Next, compute $Q_{ij;j}$.

$$Q_{\hat{\alpha}\hat{\beta};\hat{\gamma}} = Q_{\hat{\alpha}\hat{\beta},\hat{\gamma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\gamma}}\,Q_{\hat{\sigma}\hat{\beta}} - \Gamma^{\hat{\sigma}}{}_{\hat{\beta}\hat{\gamma}}\,Q_{\hat{\alpha}\hat{\sigma}}. \tag{320}$$

$$\begin{aligned} Q_{\hat{\theta}\hat{r};\hat{r}} &= Q_{\hat{\theta}\hat{r},\hat{r}} - \Gamma^{\hat{\sigma}}{}_{\hat{\theta}\hat{r}}\,Q_{\hat{\sigma}\hat{r}} - \Gamma^{\hat{\sigma}}{}_{\hat{r}\hat{r}}\,Q_{\hat{\theta}\hat{\sigma}}. \\ &= 0 \end{aligned} \tag{321}$$

$$\begin{aligned} Q_{\hat{\theta}\hat{\theta};\hat{\theta}} &= Q_{\hat{\theta}\hat{\theta},\hat{\theta}} - \Gamma^{\hat{\sigma}}{}_{\hat{\theta}\hat{\theta}}\,Q_{\hat{\sigma}\hat{\theta}} - \Gamma^{\hat{\sigma}}{}_{\hat{\theta}\hat{\theta}}\,Q_{\hat{\theta}\hat{\sigma}}. \\ &= -\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\theta}}\,Q_{\hat{\phi}\hat{\theta}} - \Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\theta}}\,Q_{\hat{\theta}\hat{\phi}}. \\ &= 0 \end{aligned} \tag{322}$$

$$\begin{aligned} Q_{\hat{\theta}\hat{\phi};\hat{\phi}} &= Q_{\hat{\theta}\hat{\phi},\hat{\phi}} - \Gamma^{\hat{\sigma}}{}_{\hat{\theta}\hat{\phi}}\,Q_{\hat{\sigma}\hat{\phi}} - \Gamma^{\hat{\sigma}}{}_{\hat{\phi}\hat{\phi}}\,Q_{\hat{\theta}\hat{\sigma}}. \\ &= Q_{\hat{\theta}\hat{\phi},\hat{\phi}} - \Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}\,Q_{\hat{\phi}\hat{\phi}} - \Gamma^{\hat{r}}{}_{\hat{\phi}\hat{\phi}}\,Q_{\hat{\theta}\hat{r}} - \Gamma^{\hat{\theta}}{}_{\hat{\phi}\hat{\phi}}\,Q_{\hat{\theta}\hat{\theta}}. \\ &= 0 \end{aligned} \tag{323}$$

ϕ terms

$$\begin{aligned} Q_{\hat{\phi}\hat{r};\hat{r}} &= Q_{\hat{\phi}\hat{r},\hat{r}} - \Gamma^{\hat{\sigma}}{}_{\hat{\phi}\hat{r}}\,Q_{\hat{\sigma}\hat{r}} - \Gamma^{\hat{\sigma}}{}_{\hat{r}\hat{r}}\,Q_{\hat{\phi}\hat{\sigma}}. \\ &= Q_{\hat{\phi}\hat{r},\hat{r}} \end{aligned} \tag{324}$$

$$\begin{aligned} Q_{\hat{\phi}\hat{\theta};\hat{\theta}} &= Q_{\hat{\phi}\hat{\theta},\hat{\theta}} - \Gamma^{\hat{\sigma}}{}_{\hat{\phi}\hat{\theta}}\,Q_{\hat{\sigma}\hat{\theta}} - \Gamma^{\hat{\sigma}}{}_{\hat{\theta}\hat{\theta}}\,Q_{\hat{\phi}\hat{\sigma}} \\ &= Q_{\hat{\phi}\hat{\theta},\hat{\theta}} - \Gamma^{\hat{r}}{}_{\hat{\theta}\hat{\theta}}\,Q_{\hat{\phi}\hat{r}} \end{aligned}$$

$$= \; Q_{\hat{\phi}\hat{\theta},\hat{\theta}} + r^{-1}\, Q_{\hat{\phi}\hat{r}} \tag{325}$$

$$\begin{aligned} Q_{\hat{\phi}\hat{\phi};\hat{\phi}} &= Q_{\hat{\phi}\hat{\phi},\hat{\phi}} - \Gamma^{\hat{\sigma}}{}_{\hat{\phi}\hat{\phi}}\, Q_{\hat{\sigma}\hat{\phi}} - \Gamma^{\hat{\sigma}}{}_{\hat{\phi}\hat{\phi}}\, Q_{\hat{\phi}\hat{\sigma}} \\ &= -\Gamma^{\hat{\sigma}}{}_{\hat{\phi}\hat{\phi}}\, Q_{\hat{\sigma}\hat{\phi}} - \Gamma^{\hat{\sigma}}{}_{\hat{\phi}\hat{\phi}}\, Q_{\hat{\phi}\hat{\sigma}} \\ &= -\Gamma^{\hat{r}}{}_{\hat{\phi}\hat{\phi}}\, Q_{\hat{r}\hat{\phi}} - \Gamma^{\hat{\theta}}{}_{\hat{\phi}\hat{\phi}}\, Q_{\hat{\theta}\hat{\phi}} - \Gamma^{\hat{r}}{}_{\hat{\phi}\hat{\phi}}\, Q_{\hat{\phi}\hat{r}} - \Gamma^{\hat{\theta}}{}_{\hat{\phi}\hat{\phi}}\, Q_{\hat{\phi}\hat{\theta}} \\ &= +r^{-1}\, Q_{\hat{r}\hat{\phi}} + \cot\theta\, r^{-1}\, Q_{\hat{\theta}\hat{\phi}} + r^{-1}\, Q_{\hat{\phi}\hat{r}} + \cot\theta\, r^{-1}\, Q_{\hat{\phi}\hat{\theta}} \\ &= +2r^{-1}\, \Lambda_{\mathrm{V}}\Omega + 2\cot\theta\, r^{-1}\, \Lambda_{\mathrm{H}}\Omega \end{aligned} \tag{326}$$

So, the ϕ component of the divergence of the Lambda tensor is then

$$Q_{\hat{\phi}\hat{\sigma};\hat{\sigma}} = Q_{\hat{\phi}\hat{r},\hat{r}} + Q_{\hat{\phi}\hat{\theta},\hat{\theta}} + 3r^{-1}\, \Lambda_{\mathrm{V}}\Omega + 2\cot\theta\, r^{-1}\, \Lambda_{\mathrm{H}}\Omega \tag{327}$$

I.2.10 Laplacian of a scalar

Let $E_{\hat{\alpha}} \equiv (\boldsymbol{\nabla}\Psi)_{\hat{\alpha}} = \partial_{\hat{\alpha}}\Psi$ Then

$$\Delta\Psi = E_{\hat{\beta};\hat{\beta}} = (\partial_{\hat{\beta}}\Psi)_{,\hat{\beta}} + \frac{2}{r}\Psi_{,\hat{r}} + \frac{\cot\theta}{r}\Psi_{,\hat{\theta}} \tag{328}$$

I.2.11 Hessian of a scalar

$$s_{;\hat{\alpha}} = s_{,\hat{\alpha}} \tag{329}$$

$$s_{;\hat{r}\hat{r}} = s_{,\hat{r}\hat{r}} - \Gamma^{\hat{\sigma}}{}_{\hat{r}\hat{r}}\, s_{,\hat{\sigma}}, = s_{,\hat{r}\hat{r}} \tag{330}$$

$$s_{;\hat{\theta}\hat{\theta}} = s_{,\hat{\theta}\hat{\theta}} - \Gamma^{\hat{r}}{}_{\hat{\theta}\hat{\theta}}\, s_{,\hat{r}}, = s_{,\hat{\theta}\hat{\theta}} + r^{-1}\, s_{,\hat{r}}, \tag{331}$$

$$s_{;\hat{\phi}\hat{\phi}} = s_{,\hat{\phi}\hat{\phi}} - \Gamma^{\hat{\sigma}}{}_{\hat{\phi}\hat{\phi}}\, s_{,\hat{\sigma}}, = s_{,\hat{\phi}\hat{\phi}} - \Gamma^{\hat{r}}{}_{\hat{\phi}\hat{\phi}}\, s_{,\hat{r}}, - \Gamma^{\hat{\theta}}{}_{\hat{\phi}\hat{\phi}}\, s_{,\hat{\theta}}, = s_{,\hat{\phi}\hat{\phi}} + r^{-1}\, s_{,\hat{r}}, + r^{-1}\cot\theta\, s_{,\hat{\theta}}, \tag{332}$$

$$s_{;\hat{r}\hat{\theta}} = s_{,\hat{r}\hat{\theta}} - \Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}}\, s_{,\hat{\theta}}, = s_{,\hat{r}\hat{\theta}} - r^{-1}\, s_{,\hat{\theta}}, \tag{333}$$

$$s_{;\hat{r}\hat{\phi}} = s_{,\hat{r}\hat{\phi}} - \Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}\, s_{,\hat{\phi}}, = s_{,\hat{r}\hat{\phi}} - r^{-1}\, s_{,\hat{\phi}} \tag{334}$$

$$s_{;\hat{\theta}\hat{r}} = s_{,\hat{\theta}\hat{r}} - \Gamma^{\hat{\sigma}}{}_{\hat{\theta}\hat{r}}\, s_{,\hat{\sigma}}, = s_{,\hat{\theta}\hat{r}} \tag{335}$$

$$s_{;\hat{\phi}\hat{r}} = s_{,\hat{\phi}\hat{r}} - \Gamma^{\hat{\sigma}}{}_{\hat{\phi}\hat{r}}\, s_{,\hat{\sigma}}, = s_{,\hat{\phi}\hat{r}} \tag{336}$$

$$s_{;\hat{\theta}\hat{\phi}} = s_{,\hat{\theta}\hat{\phi}} - \Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}\, s_{,\hat{\phi}}, = s_{,\hat{\theta}\hat{\phi}} - r^{-1}\cot\theta\, s_{,\hat{\phi}} \tag{337}$$

$$s_{;\hat{\phi}\hat{\theta}} = s_{,\hat{\phi}\hat{\theta}} - \Gamma^{\hat{\sigma}}{}_{\hat{\phi}\hat{\theta}}\, s_{,\hat{\sigma}}, = s_{,\hat{\phi}\hat{\theta}} \tag{338}$$

So,

$$s_{;\hat{\alpha}\hat{\beta}} = s_{,\hat{\alpha}\hat{\beta}} + \begin{pmatrix} 0 & -r^{-1}\, s_{,\hat{\theta}} & -r^{-1}\, s_{,\hat{\phi}} \\ 0 & +r^{-1}\, s_{,\hat{r}} & -r^{-1}\cot\theta\, s_{,\hat{\phi}} \\ 0 & 0 & -\Gamma^{\hat{r}}{}_{\hat{\phi}\hat{\phi}}\, s_{,\hat{r}} - \Gamma^{\hat{\theta}}{}_{\hat{\phi}\hat{\phi}}\, s_{,\hat{\theta}} \end{pmatrix} \tag{339}$$

I.2.12 Double curl

For the calculation of $\boldsymbol{J}=\boldsymbol{\nabla}\times\boldsymbol{B}$ we use the same curl routine, but with different arguments, bij= $B_{\hat{\alpha},\hat{\beta}}$ and bb= $B_{\hat{\alpha}}$ as input pencils and jj= $J_{\hat{\alpha}}$ as output, i.e. curl_mn(bij,jj,bb), so that

$$\boldsymbol{J}=\begin{pmatrix}B_{\hat{\phi},\hat{\theta}}-B_{\hat{\theta},\hat{\phi}}+\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}B^{\hat{\phi}}\\ B_{\hat{r},\hat{\phi}}-B_{\hat{\phi},\hat{r}}-\Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}B^{\hat{\phi}}\\ B_{\hat{\theta},\hat{r}}-B_{\hat{r},\hat{\theta}}+\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}}B^{\hat{\theta}}\end{pmatrix}=(\boldsymbol{\nabla}\times\boldsymbol{B})^{(0)}+M^{(\mathrm{curl})}_{\hat{\alpha}\hat{\beta}}B_{\hat{\beta}},\tag{340}$$

where

$$(\boldsymbol{\nabla}\times\boldsymbol{B})^{(0)\hat{\alpha}}=\epsilon^{\hat{\alpha}\hat{\beta}\hat{\gamma}}B_{\hat{\gamma},\hat{\beta}}.\tag{341}$$

Expressing the components of $(\boldsymbol{\nabla}\times\boldsymbol{B})^{(0)}$ in terms of $A_{\hat{\alpha}}$ we have

$$\begin{array}{ll}B_{\hat{\phi},\hat{\theta}}=(A_{\hat{\theta},\hat{r}}-A_{\hat{r},\hat{\theta}}+\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}}A^{\hat{\theta}})_{,\hat{\theta}} & B_{\hat{\theta},\hat{\phi}}=(A_{\hat{r},\hat{\phi}}-A_{\hat{\phi},\hat{r}}-\Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}A^{\hat{\phi}})_{,\hat{\phi}}\\ B_{\hat{r},\hat{\phi}}=(A_{\hat{\phi},\hat{\theta}}-A_{\hat{\theta},\hat{\phi}}+\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}A^{\hat{\phi}})_{,\hat{\phi}} & B_{\hat{\phi},\hat{r}}=(A_{\hat{\theta},\hat{r}}-A_{\hat{r},\hat{\theta}}+\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}}A^{\hat{\theta}})_{,\hat{r}}\\ B_{\hat{\theta},\hat{r}}=(A_{\hat{r},\hat{\phi}}-A_{\hat{\phi},\hat{r}}-\Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}A^{\hat{\phi}})_{,\hat{r}} & B_{\hat{r},\hat{\theta}}=(A_{\hat{\phi},\hat{\theta}}-A_{\hat{\theta},\hat{\phi}}+\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}A^{\hat{\phi}})_{,\hat{\theta}}\end{array}\tag{342}$$

or

$$\begin{array}{ll}B_{\hat{\phi},\hat{\theta}}=A_{\hat{\theta},\hat{r}\hat{\theta}}-A_{\hat{r},\hat{\theta}\hat{\theta}}+(\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}}A^{\hat{\theta}})_{,\hat{\theta}} & B_{\hat{\theta},\hat{\phi}}=A_{\hat{r},\hat{\phi}\hat{\phi}}-A_{\hat{\phi},\hat{r}\hat{\phi}}-(\Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}A^{\hat{\phi}})_{,\hat{\phi}}\\ B_{\hat{r},\hat{\phi}}=A_{\hat{\phi},\hat{\theta}\hat{\phi}}-A_{\hat{\theta},\hat{\phi}\hat{\phi}}+(\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}A^{\hat{\phi}})_{,\hat{\phi}} & B_{\hat{\phi},\hat{r}}=A_{\hat{\theta},\hat{r}\hat{r}}-A_{\hat{r},\hat{\theta}\hat{r}}+(\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}}A^{\hat{\theta}})_{,\hat{r}}\\ B_{\hat{\theta},\hat{r}}=A_{\hat{r},\hat{\phi}\hat{r}}-A_{\hat{\phi},\hat{r}\hat{r}}-(\Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}A^{\hat{\phi}})_{,\hat{r}} & B_{\hat{r},\hat{\theta}}=A_{\hat{\phi},\hat{\theta}\hat{\theta}}-A_{\hat{\theta},\hat{\phi}\hat{\theta}}+(\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}A^{\hat{\phi}})_{,\hat{\theta}}\end{array}\tag{343}$$

Thus,

$$B_{\hat{\alpha},\hat{\beta}}=(B_{\hat{\alpha},\hat{\beta}})^{(0)}+M^{(2\mathrm{curl2})}_{\hat{\alpha}\hat{\beta}\hat{\gamma}\hat{\delta}}A_{\hat{\gamma},\hat{\delta}}+M^{(1\mathrm{curl2})}_{\hat{\alpha}\hat{\beta}\hat{\gamma}}A_{\hat{\gamma}},\tag{344}$$

where

$$(B_{\hat{\alpha},\hat{\delta}})^{(0)}=\epsilon_{\hat{\alpha}\hat{\beta}\hat{\gamma}}A_{\hat{\gamma},\hat{\beta}\hat{\delta}}\tag{345}$$

with

$$\Psi_{,\hat{r}\hat{r}}=\Psi_{,rr}\tag{346}$$

$$\Psi_{,\hat{\theta}\hat{\theta}}=\Psi_{,\theta\theta}\,r^{-2}\tag{347}$$

$$\Psi_{,\hat{\phi}\hat{\phi}}=\Psi_{,\phi\phi}\,r^{-2}\sin^{-2}\theta\tag{348}$$

$$\Psi_{,\hat{r}\hat{\theta}}=\Psi_{,r\theta}\,r^{-1}\tag{349}$$

$$\Psi_{,\hat{\theta}\hat{r}}=\Psi_{,\theta r}\,r^{-1}-\Psi_{,\hat{\theta}}\,r^{-1}\tag{350}$$

$$\Psi_{,\hat{r}\hat{\phi}}=\Psi_{,r\phi}\,r^{-1}\sin^{-1}\theta\tag{351}$$

$$\Psi_{,\hat{\phi}\hat{r}}=\Psi_{,\phi r}\,r^{-1}\sin^{-1}\theta-\Psi_{,\hat{\phi}}\,r^{-1}\tag{352}$$

$$\Psi_{,\hat{\theta}\hat{\phi}}=\Psi_{,\theta\phi}\,r^{-2}\sin^{-1}\theta\tag{353}$$

and

$$\begin{array}{ll}B_{\hat{\phi},\hat{\theta}}=...+\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}}A^{\hat{\theta}}{}_{,\hat{\theta}}+\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta},\hat{\theta}}A^{\hat{\theta}} & B_{\hat{\theta},\hat{\phi}}=...-\Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}A^{\hat{\phi}}{}_{,\hat{\phi}}-\Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi},\hat{\phi}}A^{\hat{\phi}}\\ B_{\hat{r},\hat{\phi}}=...+\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}A^{\hat{\phi}}{}_{,\hat{\phi}}+\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi},\hat{\phi}}A^{\hat{\phi}} & B_{\hat{\phi},\hat{r}}=...+\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta}}A^{\hat{\theta}}{}_{,\hat{r}}+\Gamma^{\hat{\theta}}{}_{\hat{r}\hat{\theta},\hat{r}}A^{\hat{\theta}}\\ B_{\hat{\theta},\hat{r}}=...-\Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}A^{\hat{\phi}}{}_{,\hat{r}}-\Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi},\hat{r}}A^{\hat{\phi}} & B_{\hat{r},\hat{\theta}}=...+\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}A^{\hat{\phi}}{}_{,\hat{\theta}}+\Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi},\hat{\theta}}A^{\hat{\phi}}\end{array}\tag{354}$$

Note that some derivatives of Christoffel symbols vanish, so we are left with

$$\begin{aligned} B_{\hat{\phi},\hat{\theta}} &= ... + r^{-1} A^{\hat{\theta}}{}_{,\hat{\theta}} & B_{\hat{\theta},\hat{\phi}} &= ... - r^{-1} A^{\hat{\phi}}{}_{,\hat{\phi}} \\ B_{\hat{r},\hat{\phi}} &= ... + r^{-1}\cot\theta A^{\hat{\phi}}{}_{,\hat{\phi}} & B_{\hat{\phi},\hat{r}} &= ... + r^{-1} A^{\hat{\theta}}{}_{,\hat{r}} - r^{-2} A^{\hat{\theta}} \\ B_{\hat{\theta},\hat{r}} &= ... - r^{-1} A^{\hat{\phi}}{}_{,\hat{r}} + r^{-2} A^{\hat{\phi}} & B_{\hat{r},\hat{\theta}} &= ... + r^{-1}\cot\theta\, A^{\hat{\phi}}{}_{,\hat{\theta}} - r^{-2}\sin^{-2}\theta\, A^{\hat{\phi}} \end{aligned} \tag{355}$$

I.2.13 Gradient of a divergence

$$A_{\hat{\alpha};\hat{\alpha}\hat{\gamma}} = A_{\hat{\alpha},\hat{\alpha}\hat{\gamma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha}} A_{\hat{\sigma},\hat{\gamma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha},\hat{\gamma}} A_{\hat{\sigma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\gamma}} A_{\hat{\sigma},\hat{\alpha}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\gamma}}\Gamma^{\hat{\nu}}{}_{\hat{\sigma}\hat{\alpha}} A_{\hat{\nu}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\gamma}} A_{\hat{\alpha},\hat{\sigma}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\gamma}}\Gamma^{\hat{\nu}}{}_{\hat{\alpha}\hat{\sigma}} A_{\hat{\nu}}.$$

$\hat{r}$ component:

$$\begin{aligned} A_{\hat{\alpha};\hat{\alpha}\hat{r}} &= A_{\hat{\alpha},\hat{\alpha}\hat{r}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha}} A_{\hat{\sigma},\hat{r}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha},\hat{r}} A_{\hat{\sigma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{r}} A_{\hat{\sigma},\hat{\alpha}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{r}}\Gamma^{\hat{\nu}}{}_{\hat{\sigma}\hat{\alpha}} A_{\hat{\nu}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{r}} A_{\hat{\alpha},\hat{\sigma}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{r}}\Gamma^{\hat{\nu}}{}_{\hat{\alpha}\hat{\sigma}} A_{\hat{\nu}} \\ &= A_{\hat{\alpha},\hat{\alpha}\hat{r}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha}} A_{\hat{\sigma},\hat{r}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha},\hat{r}} A_{\hat{\sigma}} \\ &= A_{\hat{\alpha},\hat{\alpha}\hat{r}} + 2r^{-1} A_{\hat{r},\hat{r}} + r^{-1}\cot\theta A_{\hat{\theta},\hat{r}} - 2r^{-2} A_{\hat{r}} - r^{-2}\cot\theta A_{\hat{\theta}} \end{aligned} \tag{356}$$

$\hat{\theta}$ component:

$$\begin{aligned} A_{\hat{\alpha};\hat{\alpha}\hat{\theta}} &= A_{\hat{\alpha},\hat{\alpha}\hat{\theta}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha}} A_{\hat{\sigma},\hat{\theta}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha},\hat{\theta}} A_{\hat{\sigma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\theta}} A_{\hat{\sigma},\hat{\alpha}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\theta}}\Gamma^{\hat{\nu}}{}_{\hat{\sigma}\hat{\alpha}} A_{\hat{\nu}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\theta}} A_{\hat{\alpha},\hat{\sigma}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\theta}}\Gamma^{\hat{\nu}}{}_{\hat{\alpha}\hat{\sigma}} A_{\hat{\nu}} \\ &= A_{\hat{\alpha},\hat{\alpha}\hat{\theta}} + 2r^{-1} A_{\hat{r},\hat{\theta}} + r^{-1}\cot\theta A_{\hat{\theta},\hat{\theta}} - r^{-2}\sin^{-2}\theta A_{\hat{\theta}} \end{aligned} \tag{357}$$

Note that the last four terms in the above expression canceled, because

$$\begin{aligned} & -\Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\theta}} A_{\hat{\sigma},\hat{\alpha}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\theta}}\Gamma^{\hat{\nu}}{}_{\hat{\sigma}\hat{\alpha}} A_{\hat{\nu}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\theta}} A_{\hat{\alpha},\hat{\sigma}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\theta}}\Gamma^{\hat{\nu}}{}_{\hat{\alpha}\hat{\sigma}} A_{\hat{\nu}} \\ = \; & -r^{-1} A_{\hat{\theta},\hat{r}} + r^{-1} A_{\hat{r},\hat{\theta}} - r^{-2} A_{\hat{\theta}} - r^{-1} A_{\hat{r},\hat{\theta}} + r^{-1} A_{\hat{\theta},\hat{r}} + r^{-2} A_{\hat{\theta}} = 0 \end{aligned} \tag{358}$$

$\hat{\phi}$ component:

$$\begin{aligned} A_{\hat{\alpha};\hat{\alpha}\hat{\phi}} &= A_{\hat{\alpha},\hat{\alpha}\hat{\phi}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha}} A_{\hat{\sigma},\hat{\phi}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha},\hat{\phi}} A_{\hat{\sigma}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\phi}} A_{\hat{\sigma},\hat{\alpha}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\phi}}\Gamma^{\hat{\nu}}{}_{\hat{\sigma}\hat{\alpha}} A_{\hat{\nu}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\phi}} A_{\hat{\alpha},\hat{\sigma}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\phi}}\Gamma^{\hat{\nu}}{}_{\hat{\alpha}\hat{\sigma}} A_{\hat{\nu}}. \\ &= A_{\hat{\alpha},\hat{\alpha}\hat{\phi}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha}} A_{\hat{\sigma},\hat{\phi}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha},\hat{\phi}} A_{\hat{\sigma}} \\ &= A_{\hat{\alpha},\hat{\alpha}\hat{\phi}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\alpha}} A_{\hat{\sigma},\hat{\phi}} \\ &= A_{\hat{\alpha},\hat{\alpha}\hat{\phi}} - \Gamma^{\hat{r}}{}_{\hat{\theta}\hat{\theta}} A_{\hat{r},\hat{\phi}} - \Gamma^{\hat{r}}{}_{\hat{\phi}\hat{\phi}} A_{\hat{r},\hat{\phi}} - \Gamma^{\hat{\theta}}{}_{\hat{\phi}\hat{\phi}} A_{\hat{\theta},\hat{\phi}} \\ &= A_{\hat{\alpha},\hat{\alpha}\hat{\phi}} + r^{-1} A_{\hat{r},\hat{\phi}} + r^{-1} A_{\hat{r},\hat{\phi}} + r^{-1}\cot\theta A_{\hat{\theta},\hat{\phi}} \\ &= A_{\hat{\alpha},\hat{\alpha}\hat{\phi}} + 2r^{-1} A_{\hat{r},\hat{\phi}} + r^{-1}\cot\theta A_{\hat{\theta},\hat{\phi}} \end{aligned} \tag{359}$$

In the first line of the expression above, the last four terms vanish[21] and the ϕ derivative of any Christoffel symbol (term before that) also vanishes.

[21] The following four terms vanish because

$$\begin{aligned} & -\Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\phi}} A_{\hat{\sigma},\hat{\alpha}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\phi}}\Gamma^{\hat{\nu}}{}_{\hat{\sigma}\hat{\alpha}} A_{\hat{\nu}} - \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\phi}} A_{\hat{\alpha},\hat{\sigma}} + \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\phi}}\Gamma^{\hat{\nu}}{}_{\hat{\alpha}\hat{\sigma}} A_{\hat{\nu}} \\ = \; & \Gamma^{\hat{\sigma}}{}_{\hat{\alpha}\hat{\phi}}(-A_{\hat{\sigma},\hat{\alpha}} + \Gamma^{\hat{\nu}}{}_{\hat{\sigma}\hat{\alpha}} A_{\hat{\nu}} - A_{\hat{\alpha},\hat{\sigma}} + \Gamma^{\hat{\nu}}{}_{\hat{\alpha}\hat{\sigma}} A_{\hat{\nu}}) \\ = \; & \Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}(-A_{\hat{\phi},\hat{r}} + \Gamma^{\hat{\nu}}{}_{\hat{\phi}\hat{r}} A_{\hat{\nu}} - A_{\hat{r},\hat{\phi}} + \Gamma^{\hat{\nu}}{}_{\hat{r}\hat{\phi}} A_{\hat{\nu}}) \\ + \; & \Gamma^{\hat{r}}{}_{\hat{\phi}\hat{\phi}}(-A_{\hat{r},\hat{\phi}} + \Gamma^{\hat{\nu}}{}_{\hat{r}\hat{\phi}} A_{\hat{\nu}} - A_{\hat{\phi},\hat{r}} + \Gamma^{\hat{\nu}}{}_{\hat{\phi}\hat{r}} A_{\hat{\nu}}) \\ + \; & \Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}(-A_{\hat{\phi},\hat{\theta}} + \Gamma^{\hat{\nu}}{}_{\hat{\phi}\hat{\theta}} A_{\hat{\nu}} - A_{\hat{\theta},\hat{\phi}} + \Gamma^{\hat{\nu}}{}_{\hat{\theta}\hat{\phi}} A_{\hat{\nu}}) \end{aligned}$$

$$
\begin{aligned}
&+ \quad \Gamma^{\hat{\theta}}{}_{\hat{\phi}\hat{\phi}}(-A_{\hat{\theta},\hat{\phi}} + \Gamma^{\hat{\nu}}{}_{\hat{\theta}\hat{\phi}}\, A_{\hat{\nu}} - A_{\hat{\phi},\hat{\theta}} + \Gamma^{\hat{\nu}}{}_{\hat{\phi}\hat{\theta}}\, A_{\hat{\nu}}) \\
&= \quad \Gamma^{\hat{\phi}}{}_{\hat{r}\hat{\phi}}(-A_{\hat{\phi},\hat{r}} - A_{\hat{r},\hat{\phi}} + \Gamma^{\hat{\nu}}{}_{\hat{r}\hat{\phi}}\, A_{\hat{\nu}}) \\
&+ \quad \Gamma^{\hat{r}}{}_{\hat{\phi}\hat{\phi}}(-A_{\hat{r},\hat{\phi}} + \Gamma^{\hat{\nu}}{}_{\hat{r}\hat{\phi}}\, A_{\hat{\nu}} - A_{\hat{\phi},\hat{r}}) \\
&+ \quad \Gamma^{\hat{\phi}}{}_{\hat{\theta}\hat{\phi}}(-A_{\hat{\phi},\hat{\theta}} - A_{\hat{\theta},\hat{\phi}} + \Gamma^{\hat{\nu}}{}_{\hat{\theta}\hat{\phi}}\, A_{\hat{\nu}}) \\
&+ \quad \Gamma^{\hat{\theta}}{}_{\hat{\phi}\hat{\phi}}(-A_{\hat{\theta},\hat{\phi}} + \Gamma^{\hat{\nu}}{}_{\hat{\theta}\hat{\phi}}\, A_{\hat{\nu}} - A_{\hat{\phi},\hat{\theta}}) \\
&= \quad r^{-1}(-A_{\hat{\phi},\hat{r}} - A_{\hat{r},\hat{\phi}} + \Gamma^{\hat{\nu}}{}_{\hat{r}\hat{\phi}}\, A_{\hat{\nu}}) \\
&- \quad r^{-1}(-A_{\hat{r},\hat{\phi}} - A_{\hat{\phi},\hat{r}} + \Gamma^{\hat{\nu}}{}_{\hat{r}\hat{\phi}}\, A_{\hat{\nu}}) \\
&+ \quad r^{-1}\cot\theta(-A_{\hat{\phi},\hat{\theta}} - A_{\hat{\theta},\hat{\phi}} + \Gamma^{\hat{\nu}}{}_{\hat{\theta}\hat{\phi}}\, A_{\hat{\nu}}) \\
&- \quad r^{-1}\cot\theta(-A_{\hat{\theta},\hat{\phi}} - A_{\hat{\phi},\hat{\theta}} + \Gamma^{\hat{\nu}}{}_{\hat{\theta}\hat{\phi}}\, A_{\hat{\nu}}) = 0
\end{aligned}
\tag{360}
$$

J Switchable modules

The material in this section is being assembled from the file 'inlinedoc-modules.tex', which is automatically assembled. However, it is currently incomplete and contains only a small number of the available modules.

Module	Description
hydro.f90	This module takes care of most of the things related to velocity. Pressure, for example, is added in the energy (entropy) module.
chemistry.f90	This modules adds chemical species and reactions. The units used in the chem.in files are cm3,mole,sec,kcal and K
gpu_astaroth.f90	This module contains GPU related types and functions to be used with TAROTH nucleus.
noentropy.f90	Calculates pressure gradient term for polytropic equation of state $p = \text{const}\rho^{\Gamma}$.
nogpu.f90	This module contains GPU related dummy types and functions.
nohydro.f90	no variable $\boldsymbol{u}$: useful for kinematic dynamo runs.
nopower_spectrum.f90	reads in full snapshot and calculates power spetrum of u
noyinyang.f90	This module contains Yin-Yang related dummy types and functions.
noyinyang_mpi.f90	This module contains Yin-Yang related dummy types and functions.
particles_adsorbed.f90	This module takes care of the evolution of adsorbed species on the particle surface for reactive particles
particles_chemistry.f90	This module implements reactive particles.
particles_surfspec.f90	immediate vicinity of reactive particles.
power_spectrum.f90	reads in full snapshot and calculates power spetrum of u
timestep.f90	Runge-Kutta time advance, accurate to order itorder. At the moment, itorder can be 1, 2, or 3.
timestep_strang.f90	Runge-Kutta time advance, accurate to order itorder. At the moment, itorder can be 1, 2, or 3.
timestep_subcycle.f90	This is a highly specified timestep module currently only working together with the special module coronae.f90.
yinyang.f90	This module contains Yin-Yang related types and functions which are incompatible with FORTRAN 95.

yinyang_mpi.f90	This module contains Yin-Yang related types and functions which are incompatible with FORTRAN 95.

K 启动选项和运行中参数 Startup and run-time parameters

Chinese translation team plans to make the parameters file standalone, but with low priority.
中文翻译的团队计划将这一部分单独列做出一个文件, 这一步还在计划中.

K.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 ε 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, $\varepsilon \approx 10^{-15}$.

K.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.

K.3 List of parameters for 'print.in'

The following table lists all possible inputs to the file 'print.in' that are documented.

K.4 List of parameters for 'video.in'

The following table lists all (at the time of writing, 2018 年 10 月 23 日) possible inputs to the file ‘video.in’.

K.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’.

K.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.

K.7 List of parameters for ‘xzaver.in’

The following table lists possible inputs to the file ‘xzaver.in’. This list is not complete and maybe outdated.

K.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.

K.9 List of parameters for ‘yaver.in’

The following table lists possible inputs to the file ‘yaver.in’. This list is not complete and maybe outdated.

K.10 List of parameters for ‘zaver.in’

The following table lists possible inputs to the file ‘zaver.in’. This list is not complete and maybe outdated.

K.11 Boundary conditions

The following tables list all possible boundary condition labels that are documented.

K.11.1 Boundary condition bcx

K.11.2 Boundary condition bcy

K.11.3 Boundary condition bcz

K.12 Initial condition parameter dependence

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.

inituu	ampluu	widthuu	urand	uu_left	uu_right	uu_upper	uu_lower	uy_left	uy_right	kx_uu	ky_uu	kz_uu
zero												
gaussian-noise	●											
gaussian-noise-x	●											
xjump		◇		●	●			●	●			
Beltrami-x	●											
Beltrami-y	●											
Beltrami-z	●											
trilinear-x	●											
trilinear-y	●											
trilinear-z	●											
cos-cos-sin-uz	●											
tor_pert	●											
trilinear-x	●											
sound-wave	●									●		
shock-tube		◇		●	●							
bullets	●	◇										
Alfven-circ-x	●									◇		
const-ux	●											
const-uy	●											
tang-discont-z	◇	●				●	●					
Fourier-trunc	●	◇								●	●	
up-down	●	◇										

initss	ampl_ss	radius_ss	widthss	epsilon_ss	grads0	pertss	ss_left	ss_right	ss_const	mpoly0	mpoly1	mpoly2	isothtop	khor_ss	center1_x	center1_y	center1_z	center2_x	center2_y	center2_z	thermal_background
zero																					
const_ss									•												
blob	•	•																			
isothermal																					
Ferrière																					
xjump			•				•	•													
hor-fluxtube	•	•		•																	
hor-tube	•	•		•																	
sedov		•													•	•	•				•
sedov-dual		•													•	•	•	•	•	•	•
isobaric																					
isentropic																					
linprof																					
piecew-poly																					
polytropic																					