The Pencil Code Newsletter

Issue 2026/1

June 21, 2026, Revision: 1.9

Contents

1

1.1 No yz-slices?

During the Pencil Code office hours in xx and April, a problem about missing yz slices was reported.

1.2

In viscosity.f90, we had a piece:

```
if (ldynamical_diffusion .and. lvisc_hyper3_mesh) then
  diffus_nu3 = p%diffus_total3 * sum(abs(dline_1),2)
else
  diffus_nu3 = p%diffus_total3*dxyz_6
endif
```

```
which was not correct, because it makes diffus_nu3 finite in
have nothing to do with hyper3. I noticed this because of it
constraint, and this could have been a problem also in a few
I noticed that this piece of code goes back to 2011-08-02 an
that it was coded by Chao-Chin with the message

  By fixing the mesh Reynolds number, dynamically adjust the
  coefficients for density, magnetic, and viscosity.  Genera
  non-equidistant grids.  cdtv3 may be as high as 0.9, or ev

after earlier work by Wlad. I then disentagled the loop into
realized that diffus_nu3 will be needed in a few other cases
in samples/MRI-turbulence_hyper we have lvisc_hyper3_rho_nu_
so I included this and it worked. I have now added all 14 lo
that seem to be needed.

        if (lvisc_hyper3_mesh .or. lvisc_hyper3_simplified .
            lvisc_hyper3_mesh_residual .or. lvisc_hyper3_mes
            lvisc_hyper3_mu_const_strict .or. lvisc_hyper3_n
            lvisc_hyper3_cmu_const_strt_otf .or. lvisc_hyper
            lvisc_hyper3_rho_nu_const_aniso .or. lvisc_hyper
            lvisc_hyper3_rho_nu_const_bulk .or. lvisc_hyper3

I wanted to explain my changing of the interstellar sample d
With Matthias, we enabled the possibility to output spectra
slices based on another trigger than just the time. Therefor
in general, the time may not be available, which could be un
Therefore we added it to the spectra. In the usual case, the
comes now twice, but the advantage is that we don't need to
complicated queries. Let me know if there are questions.
Axel

PS: here the differences:

<    1.7948403418635799E-008   1.7948403418635799E-008
---
>    1.7948403418635799E-008
3c3
<    3.4513211517011286E-008   3.4513211517011286E-008
---
>    3.4513211517011286E-008
5c5
```

```
<    3.9465918924760758E-008   3.9465918924760758E-008
---
>    3.9465918924760758E-008
7c7
<    5.6922915463575225E-008   5.6922915463575225E-008
---
>    5.6922915463575225E-008
```

```
start_run.csh habe ich angepasst, funktioniert jetzt auch mit HDF5.
```

2 Default changes

2.1 Lorentz force spectra for kinematic flows

Since 6-mar-26 (i.e., since revision r42876), we compute Lorentz spectrum also for lhydro_kinematic.and.iux/=0, i.e., when the velocity is defined as an auxiliary array.

3 Code additions

3.1 Persist variables

src/record_types.h

The relevant routine (e.g., special) must have subroutine input_persist_special_id(id,done)

3.2 Regarding GPUs

Changes for GPU capability. To explain meaning of `!$omp do collapse(2)` and `!$omp workshare` and and

Also, why do we need `m=m_loc;n=n_loc`? We should explain this both in the newsletter and in `src/power_spectrum.f90`.

```
!$omp parallel private(k,k2) num_threads(num_helper_threads) &
!$omp copyin(MPI_COMM_GRID,MPI_COMM_PENCIL,MPI_COMM_XBEAM,MPI_COMM_YBEAM,MPI_COMM_ZBEAM, &
!$omp MPI_COMM_XYPLANE,MPI_COMM_XZPLANE,MPI_COMM_YZPLANE)
!$ thread_id = omp_get_thread_num()+1
```

4 sourceme.sh

–rebase.

5 Signalling NaNs

Initializing the code with signalling NaNs can be helpful in detecting uninitialized variables. To enable this, one can compile the code with with `FFLAGS+= -finit-real=sNaN` (for the gnu compiler) and with `FFLAGS += -finit-integer=-2147483648` for integer. During the last `Pencil Code` office hours in May, Philippe reminded us that such options and similar ones are listed in the directory

```
pencil-code/config/compilers/extensions
```

in the file

```
GNU-GCC_debug.conf
```

```
~/pencil-code/config/compilers/extensions/GNU-GCC_debug.conf
```

6 Excessive NaN run-time output

Put ip>14 (14 is the default)

7 ReadTheDocs: tutorials and postprocessing

ReadTheDocs hosts the central Pencil Code documentation and resources: general documentation, developer guides, API reference, tutorials, and sample workflows. See `https://pencil-code.readthedocs.io/`https://pencil-code.readthedocs.io/ for an overview.

7.1 The Pencil Code: samples

The documentation now includes a new set of tutorial samples that walk users through complete workflows for different physical processes. Each sample is designed to cover:

- Scientific background and motivation
- Simulation configuration and setup
- Useful runtime checks and diagnostics
- Post-processing with Python for analysis and visualization

This makes it easier for new users to reproduce results and learn standard workflows.

The first published example is a simple convection simulation, with a step-by-step guide from problem setup through execution and Python-based post-processing: Convection simulation

This is the beginning of a larger tutorial collection, and we encourage contributors to add new samples for other physics cases. If you are working with existing example directories under `pencil-code/samples`, your material would be a great addition.

7.2 IDL to Python

A new documentation page provides an expanded translation guide from IDL to Python, tailored to the Pencil Code ecosystem. It explains how to convert legacy IDL analysis scripts into modern Python code, including recommended Pencil Code utilities, common syntax differences, and practical examples for arrays, plotting, and file handling.

This guide is especially useful for users who want to migrate existing IDL workflows while preserving the same data-processing logic in Python. It helps users modernize legacy analysis without losing existing workflow logic. See the new page here: IDL to Python

8 Pencil Code Python: package, tests, and recent meetings

The idea of these meetings is to share developments, discuss ideas, present new work, and help the community.

This year we held two meetings (2026 Jan 29 and 2026 Mar 30). The Python postprocessing group has made rapid progress on packaging, testing, and tutorials; the meetings concentrated on three priority areas: making the Python toolkit an installable package, establishing automated Python tests, and organizing tutorial samples linked from the documentation.

Key outcomes and ongoing work:

- Packaging: the project is moving toward an official pip-installable package (`pypencil`); this is a high-priority effort led by Alberto and will make installation as simple as "pip install pypencil".

- Testing: Kishore has developed a suite of Python tests; the next step is to host these on an automated server (options such as Nordita machines are being explored) so tests run on new commits.

- Tutorials and docs: the group agreed to develop a set of tutorial samples; the first published example is the convection simulation described in section ??. Samples live in subdirectories of `pencil-code/samples` (for example `pencil-code/samples/conv-slab`); each sample may be a Jupyter notebook or an RST file and will be linked from ReadTheDocs.

- API and usability improvements: work is underway to standardize array-order conventions, unify error handling (raise informative exceptions), harmonize debug output defaults ("quiet" flag), and provide example usage in docstrings to aid both users and tests.

- Interoperability: routines to export var files to other formats (e.g., Athena-compatible output) are being modularized and integrated into the Python toolkit.

The `pypencil` package is a high-priority effort and will make it straightforward to install the postprocessing toolkit via pip.

We warmly encourage the community to join the Python working-group meetings, propose agenda items, and contribute tests or tutorial samples (place them under `pencil-code/samples`). Your participation directly shapes priorities and improves the tools for everyone.

9 Papers since January 2026

As usual, we look here at new papers that make use of the Pencil Code. Since the last newsletter of August 2025, six new papers have appeared on the arXiv, plus nine others, some of which had been just preprints and now have been published with a journal reference on ADS. We list both here, altogether 15. A browsable ADS list of all Pencil Code papers can be found on: `https://ui.adsabs.harvard.edu/public-libraries/iGR7N570Sy6AlhDMQRTe_A`. If something is missing in those entries, you can also include it yourself in: `https://github.com/pencil-code/pencil-code/blob/master/doc/citations/ref.bib`, or otherwise just email brandenb@nordita.org. A compiled version of this file is available as `https://github.com/pencil-code/website/blob/master/doc/citations.pdf`, where we also list a total of now 136 code comparison papers in the last section "Code comparison & reference". Those are not included in our list below, nor among the now total number of 764 research papers that use the Pencil Code.

References

Brandenburg, A., Ghosh, O., Vazza, F. and Neronov, A., Magnetic Field Spreading from Stellar and

Galactic Dynamos into the Exterior. *Astrophys. J.*, 2026, **1002**, 165.

Sharma, R., Majumdar, S. and Sachdeva, D., Suppressed Magnetogenesis from Ultralight Dark Matter due to Finite Conductivity. *arXiv e-prints*, 2026, arXiv:2604.17230.

Vemareddy, P., Nair, S. and Gosain, S., Understanding the Formation and Eruption of Sigmoidal Structure through Data-driven Modeling of Magnetic Evolution in Solar Active Region 13500. *Astrophys. J.*, 2026, **1001**, 16.

This Pencil Code Newsletter was edited by Axel Brandenburg `<brandenb@nordita.org>`, Nordita, KTH Royal Institute of Technology and Stockholm University, SE-10691 Stockholm, Sweden; and Matthias Rheinhardt `<matthias.rheinhardt@aalto.fi>`, Department of Computer Science, Aalto University, PO Box 15400, FI-00076 Aalto, Finland. See `http://www.nordita.org/~brandenb/pencil-code/newsletter` or `https://github.com/pencil-code/website/tree/master/NewsLetters` for the online version as well as back issues.