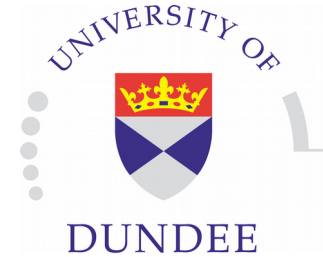


Quo vadis pencilnew?

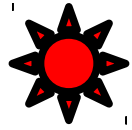
Simon Candelaresi



```
temp.py x div_grad_curl.py x _init_.py x README x TO_DO_LIST.txt x var.py x
pencilnew
├── __pycache__
├── backpack
├── calc
├── diag
├── export
├── io
├── math
├── read
├── __pycache__
│   ├── _init_.py
│   ├── _init_.pyc
│   ├── averages.py
│   ├── averages.pyc
│   ├── dim.py
│   ├── dim.pyc
│   ├── grid.py
│   ├── grid.pyc
│   ├── index.py
│   ├── index.pyc
│   ├── ogdim.py
│   ├── ogdim.pyc
│   ├── ogvar.py
│   ├── ogvar.pyc
│   ├── param.py
│   ├── param.pyc
│   ├── pdim.py
│   ├── pdim.pyc
│   ├── power.py
│   ├── power.pyc
│   ├── pstalk.py
│   ├── pstalk.pyc
│   ├── pvar.py
│   ├── pvar.pyc
│   ├── slices.py
│   ├── slices.pyc
│   ├── ts.py
│   ├── ts.pyc
│   └── var.py
├── var.py
├── var.pyc
└── zprof.py
```

```
80 class DataCube(object):
81     """
82     DataCube -- holds Pencil Code VAR file data.
83     """
84
85     def __init__(self):
86         """
87         Fill members with default values.
88         """
89
90         self.t = 0.
91         self.dx = 1.
92         self.dy = 1.
93         self.dz = 1.
94
95
96     def read(self, var_file='', datadir='data', proc=-1, ivar=-1, quiet=True,
97             trimall=False, magic=None, sim=None, precision='f'):
98         """
99         Read VAR files from Pencil Code. If proc < 0, then load all data
100        and assemble, otherwise load VAR file from specified processor.
101
102        The file format written by output() (and used, e.g. in var.dat)
103        consists of the following Fortran records:
104        1. data(mx, my, mz, nvar)
105        2. t(1), x(mx), y(my), z(mz), dx(1), dy(1), dz(1), deltatay(1)
106        Here nvar denotes the number of slots, i.e. 1 for one scalar field, 3
107        for one vector field, 8 for var.dat in the case of MHD with entropy.
108        but, deltatay(1) is only there if lshear is on! need to know parameters.
```

Motivation



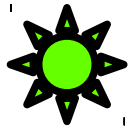
Python3 compatible.



Improved documentation.



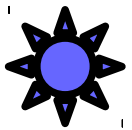
Consistent and clean coding style.



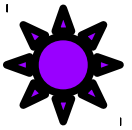
Improved performance.



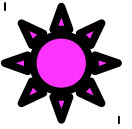
Object oriented approach.



Clean and logical structure.



Tutorials



Coordinate transformations.

Contributors

Andreas-schreiber <andreas.schreiber88@googlemail.com>

PABourdin <git@Bourdin.ch>

Wladimir Lyra <wlyra@caltech.edu>

Jørgen R. Aarnes <jorgen.r.aarnes@ntnu.no>

Luiz Felipe S. Rodrigues <luiz.rodrigues@newcastle.ac.uk>

Wolfgang Dobler <wdobler@gmail.com>

Frederick Gent <fred.gent.ncl@gmail.com>

Mrheinhardt <mreinhardt@nordita.org>

Ewa <ewa.karchniwy@gmail.com>

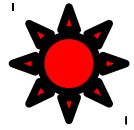
Callum Reid <apollocreid@gmail.com>

lomsn <iomsn1@gmail.com>

Progress

- ☺ Porting of old routines.
- ☺ Python3 compatible.
- ☺ Good documentation.
- ☺ Consistent and clean coding style.
- ☺ Improved performance.
- ☺ Object oriented approach.
- ☺ Clean and logical structure.
- ☺ Tutorials
- ☺ Simulation objects.
- ☺ Coordinate transformations.

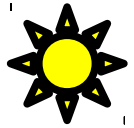
ToDo.



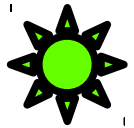
Paralellization.



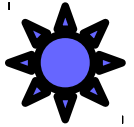
Coordinate transformations.



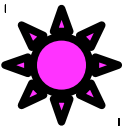
Particles.



Information about array indices.



Example in documentation strings.



Imports from IDL.