UNIVERSITY OF DUNDEE

PencilCode and

python

Why Python?

Highly expressive:

```
friends = ['john', 'pat', 'gary', 'michael']
for i, name in enumerate(friends):
    print("iteration {iteration} is {name}".format(iteration=i, name=name))
```

Object oriented

Ease:

Easy to learn, change from IDL or Matlab.

Very active community

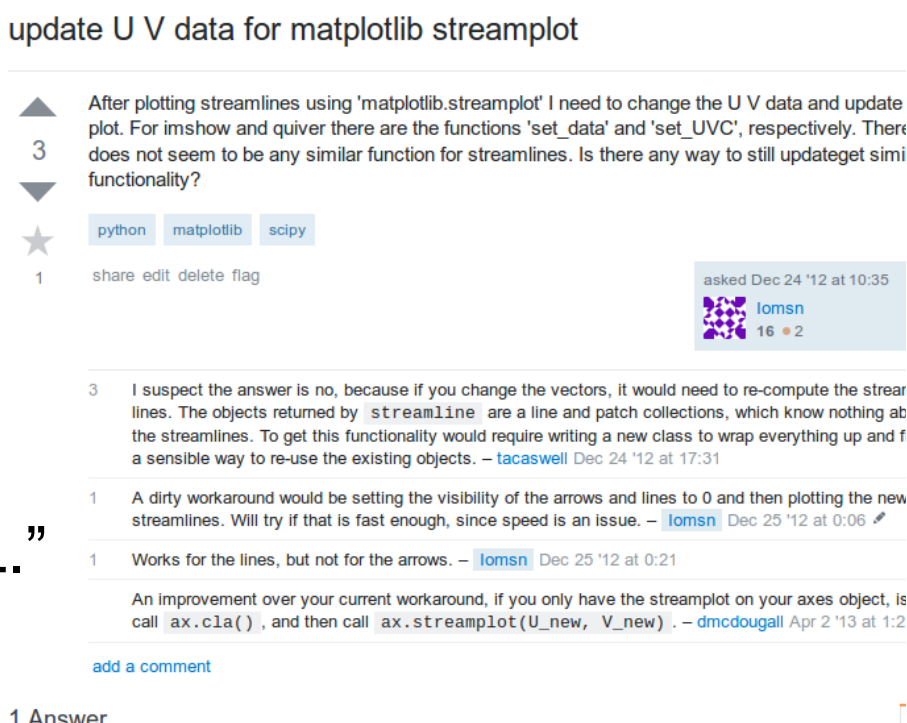
Extendability:

libraries for numerics, data analysis, plotting, financing, ...

Interoperability:

Import IDL and Matlab code.

“But all my routines are written in IDL.”



update U V data for matplotlib streamplot

3

1

After plotting streamlines using 'matplotlib.streamplot' I need to change the U V data and update plot. For imshow and quiver there are the functions 'set_data' and 'set_UVC', respectively. There does not seem to be any similar function for streamlines. Is there any way to still updateget simi functionality?

python matplotlib scipy

share edit delete flag

asked Dec 24 '12 at 10:35

lomsn

16 • 2

3 I suspect the answer is no, because if you change the vectors, it would need to re-compute the strean lines. The objects returned by `streamline` are a line and patch collections, which know nothing ab the streamlines. To get this functionality would require writing a new class to wrap everything up and f a sensible way to re-use the existing objects. – tacaswell Dec 24 '12 at 17:31

1 A dirty workaround would be setting the visibility of the arrows and lines to 0 and then plotting the new streamlines. Will try if that is fast enough, since speed is an issue. – lomsn Dec 25 '12 at 0:06

1 Works for the lines, but not for the arrows. – lomsn Dec 25 '12 at 0:21

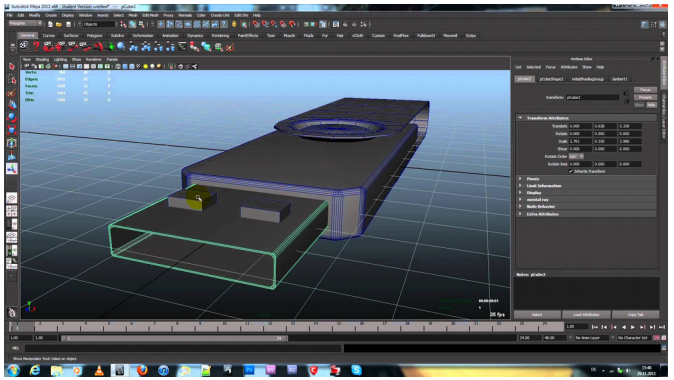
An improvement over your current workaround, if you only have the streamplot on your axes object, is call `ax.cla()`, and then call `ax.streamplot(U_new, V_new)`. – dmcdougall Apr 2 '13 at 1:2

add a comment

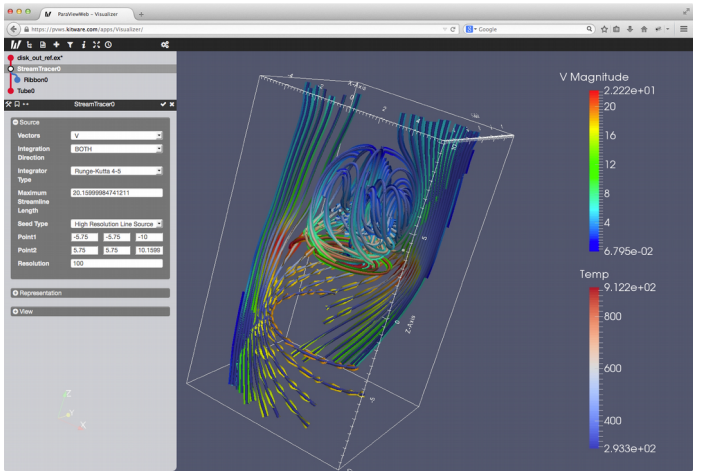
1 Answer

Why Python?

"Python is everywhere, it is all around us, even now in this very room."



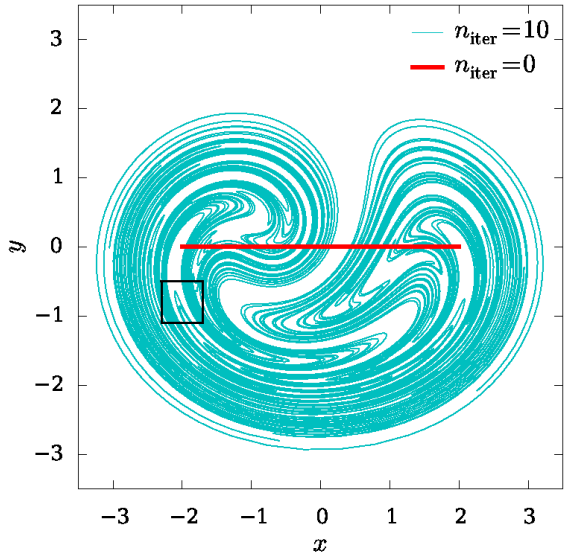
3ds Max
Maya
Blender
Cinema 4D
...

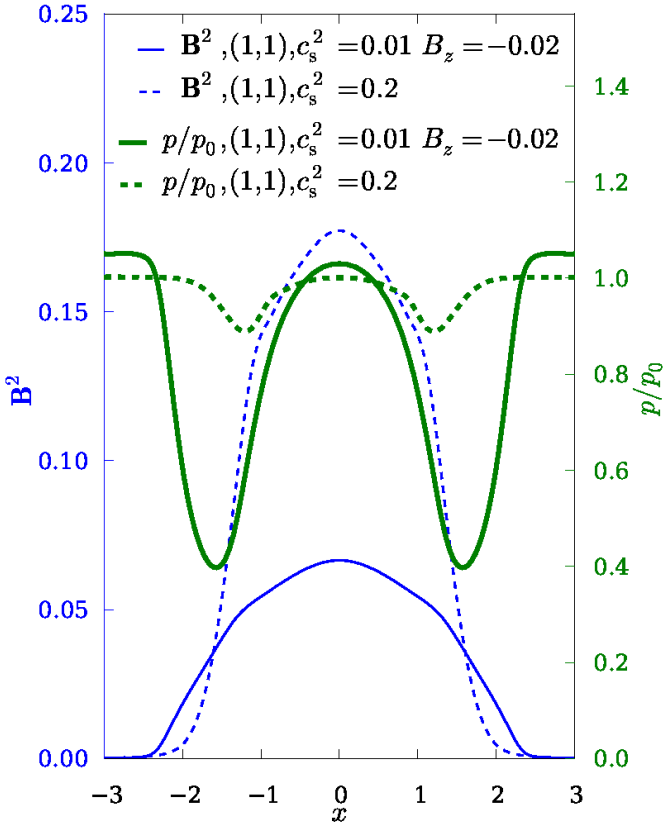


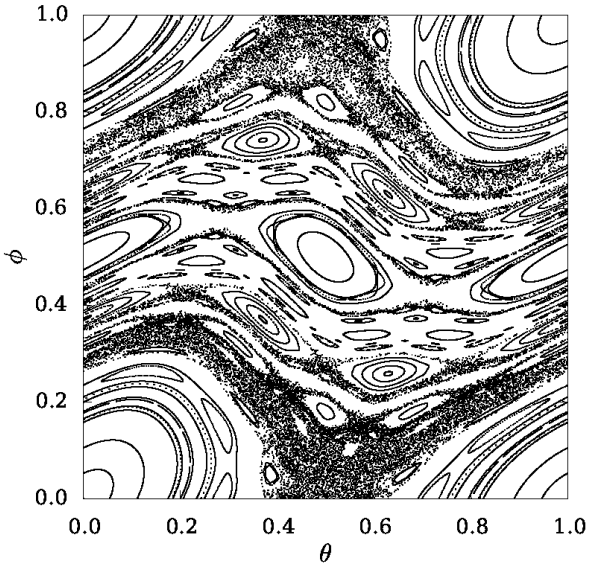
Paraview, Visit, Vapor, ...

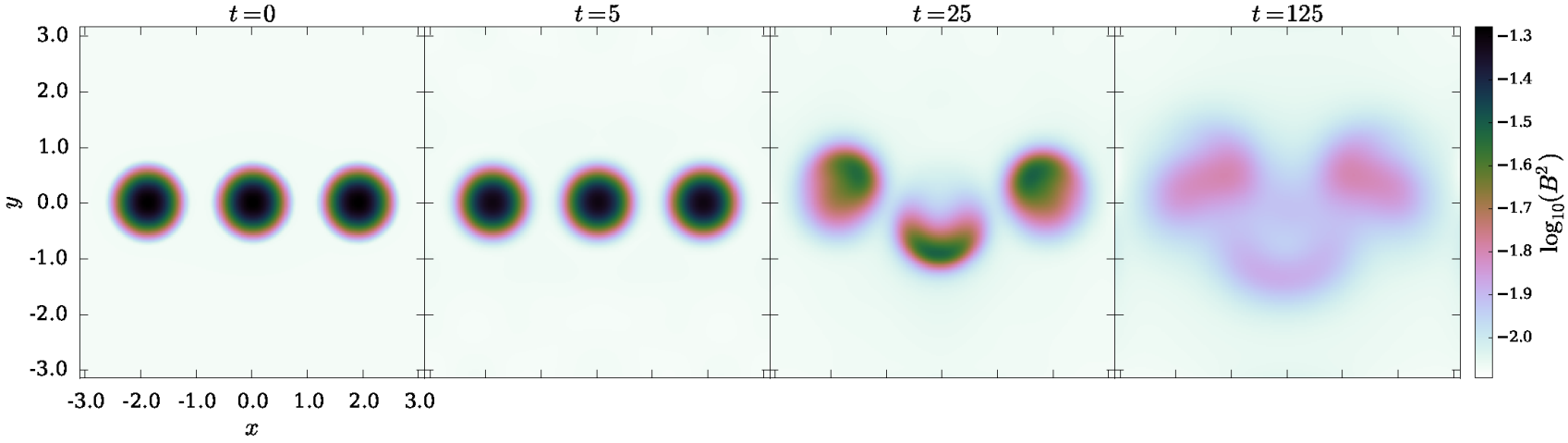
Civilization IV, Battlefield 2, World of Tanks, ...

Why Python?

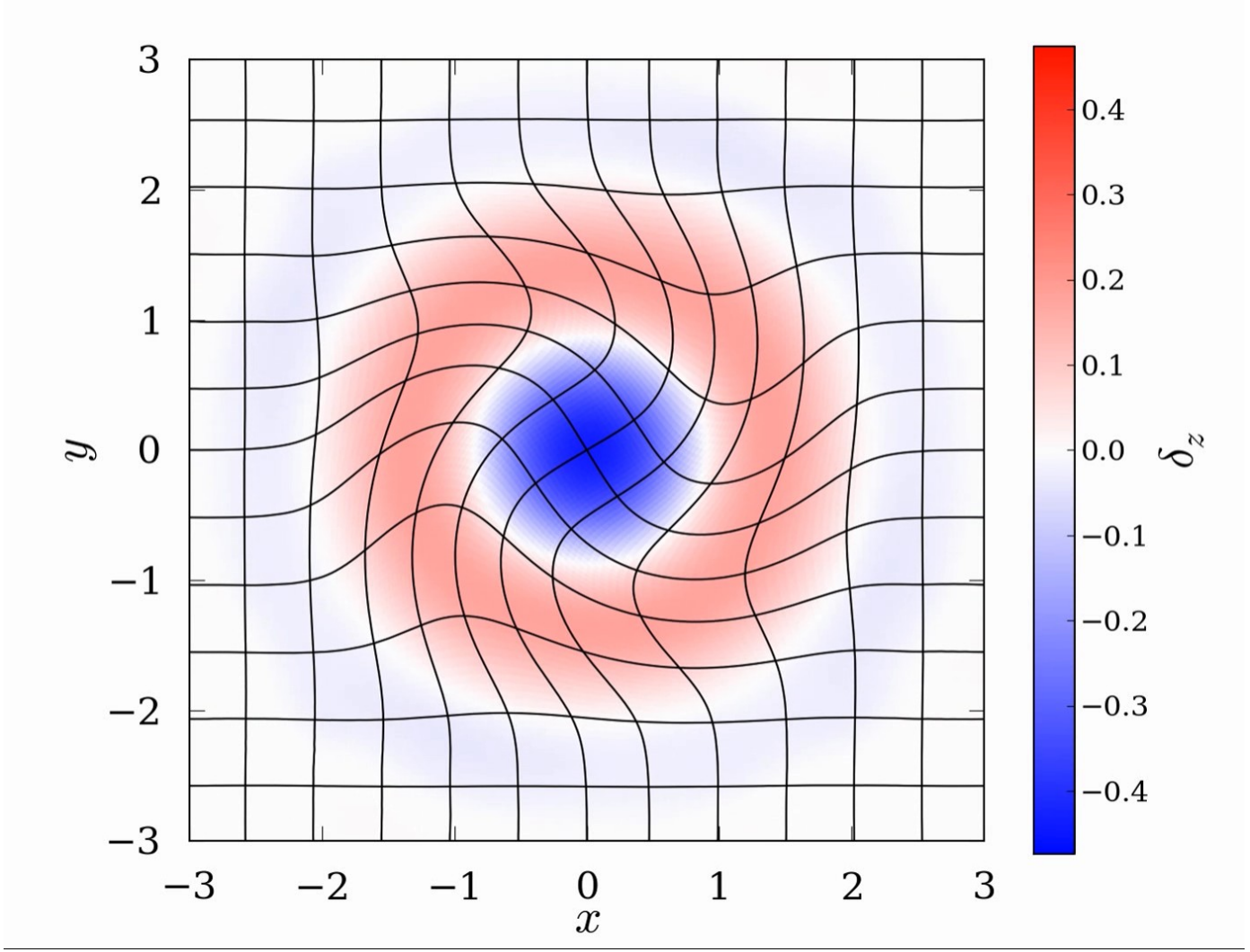

$n_{\rm iter}=10$
$n_{\rm iter}=0$
x
y


$\mathbf{B}^2,(1,1),c_s^2=0.01\ B_z=-0.02$
$\mathbf{B}^2,(1,1),c_s^2=0.2$
$p/p_0,(1,1),c_s^2=0.01\ B_z=-0.02$
$p/p_0,(1,1),c_s^2=0.2$
$\mathbf{B}^2$
p/p_0
x


ϕ
θ

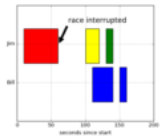

$t=0$
$t=5$
$t=25$
$t=125$
x
y
$\log_{10}(B^2)$

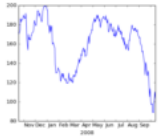
Why Python?

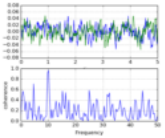


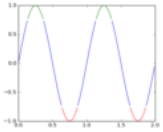
Libraries

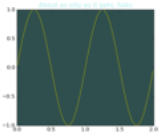
Matplotlib Gallery

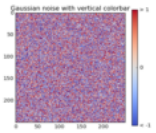

broken_barh

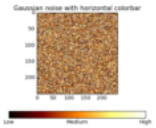

centered_ticklabels

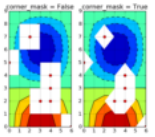

cohere_demo

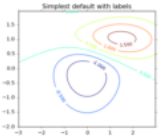

color_by_yvalue

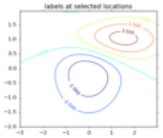

color_demo

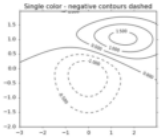

colorbar_tick_labelling_demo


colorbar_tick_labelling_demo


contour_corner_mask


contour_demo


contour_demo


contour_demo

Scipy Functionalities

Tutorial

Tutorials with worked examples and background information for most SciPy submodule

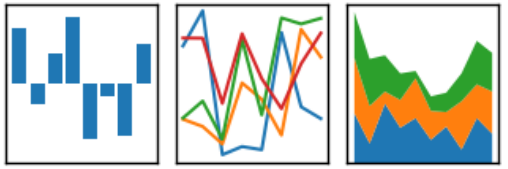
- SciPy Tutorial
  - Introduction
  - Basic functions
  - Special functions (**scipy.special**)
  - Integration (**scipy.integrate**)
  - Optimization (**scipy.optimize**)
  - Interpolation (**scipy.interpolate**)
  - Fourier Transforms (**scipy.fftpack**)
  - Signal Processing (**scipy.signal**)
  - Linear Algebra (**scipy.linalg**)
  - Sparse Eigenvalue Problems with ARPACK
  - Compressed Sparse Graph Routines (**scipy.sparse.csgraph**)
  - Spatial data structures and algorithms (**scipy.spatial**)
  - Statistics (**scipy.stats**)
  - Multidimensional image processing (**scipy.ndimage**)
  - File IO (**scipy.io**)
  - Weave (**scipy.weave**)

Libraries

Python Data Analysis Library

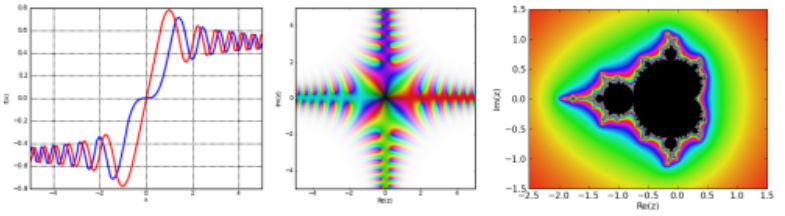
pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



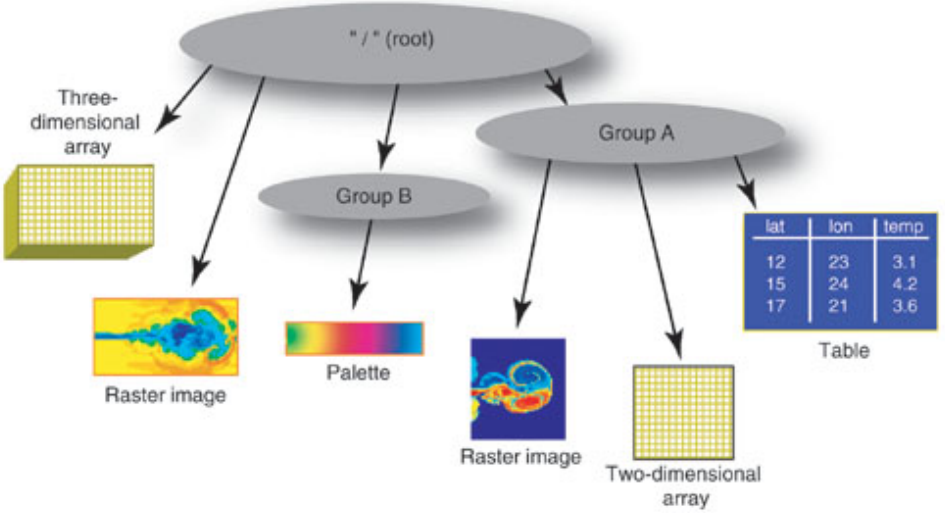
http://pandas.pydata.org

mpmath
floating-point arithmetic with arbitrary precision

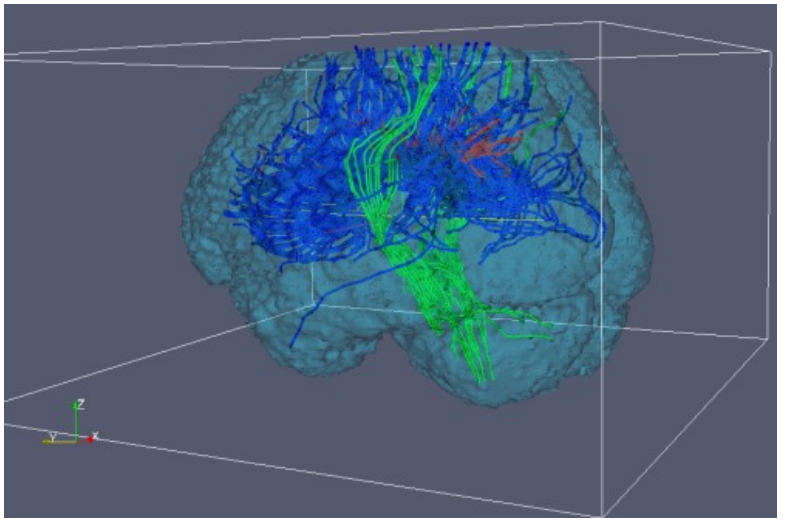


http://mpmath.org

Hierarchical Data Format



vtk Data Format



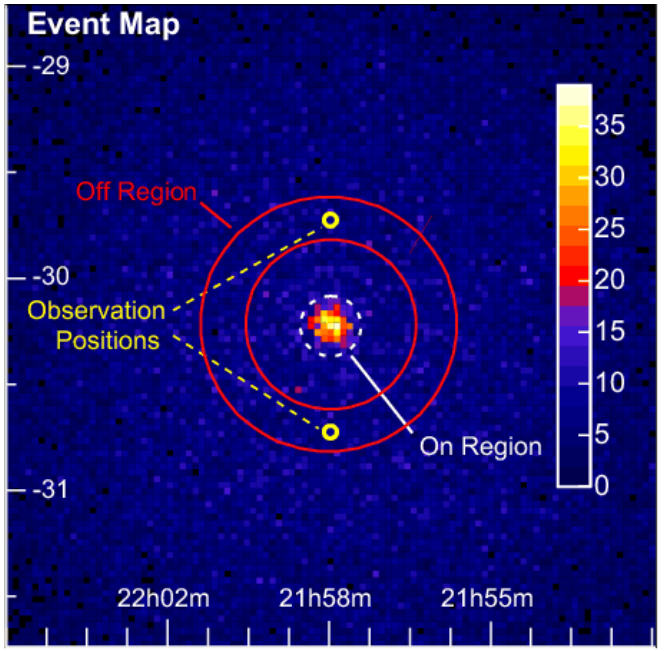
Libraries

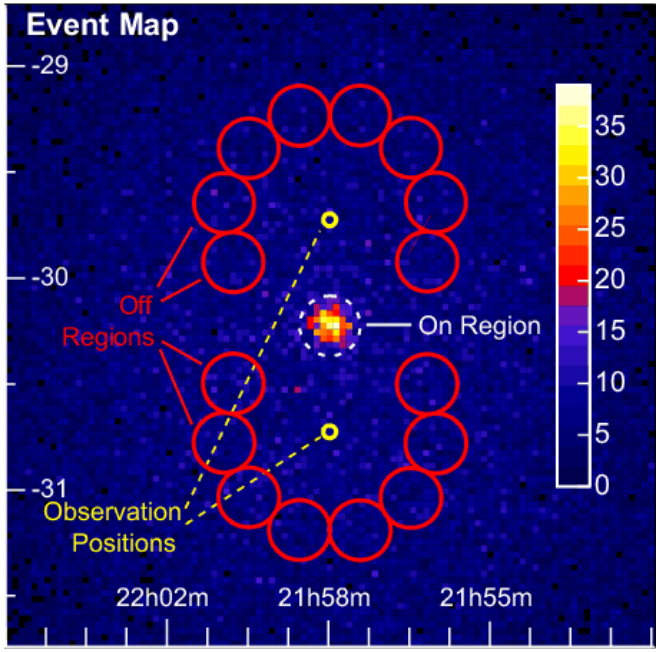
SunPy

http://sunpy.org

astropy

http://www.astropy.org



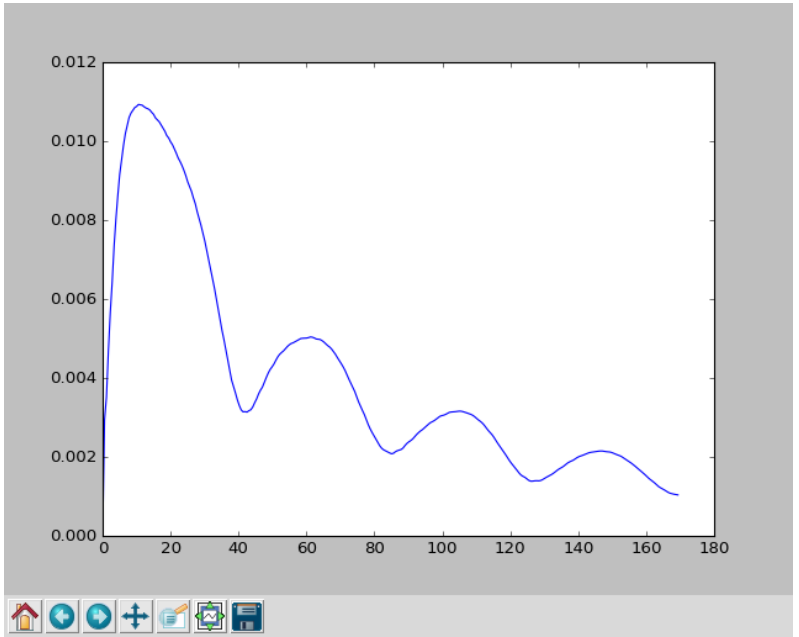


Python and Pencil

```
animate_interactive.py
dim.py
fixed_points.py
get_format.py
grid.py
hdf5.py
index.py
__init__.py
kf.py
movie.py
mtTkinter.py
multi_slices.py
npfile.py
paramdict.py
param.py
particles_removed.py
particles_to_density.py
pc2vtk.py
pdim.py
post_processing.py
power2vtk.py
power.py
read_pvar.py
read_qvar.py
remesh.py
rrmv_par.py
slices.py
sn.py
streamlines.py
tensors.py
tracers.py
ts.py
var.py
xyaver.py
xzaver.py
yaver.py
yzaver.py
zaverages.py
zaver.py
zprof.py
```

```
import pylab as plt
import pencil as pc

ts = pc.read_ts()
plt.plot(ts.t, ts.urms)
```



PencilOld Issues

1. Documentation not consequent:

```
read_ts(*args, **kwargs)
    Read Pencil Code time series data.
    params:
    string: filename  ='time_series.dat'
    string: datadir   = 'data'
    logical: plot_data = False
    logical: quiet     = False
```

```
Assemble a 2D animation from a 3D array. *data* has to be a
3D array who's time index has the same dimension as *t*. The
time index of *data* as well as its x and y indices can be
changed via *dimOrder*.

Keyword arguments:

  *dimOrder*: [ (i,j,k) ]
    Ordering of the dimensions in the data array (t,x,y).

  *fps*:
    Frames per second of the animation.
```

PencilOld Issues

2. Not much object oriented:

```
slices, t = pc.read_slices()
```

3. High refraction potential:

```
xyaver.py  xzaver.py  yaver.py  yzaver.py  zaverages.py
zaver.py
```

4. Python3 compatibility issues.

5. Some reading routines are slow (slides).

6. Parallelizability potential.

7. Tried to use IDL structure which is not suited for Python.

PencilNew

pencil-code/python/pencilnew

```
backpack   __init__.py       math     STYLEGUIDE.txt
calc       __init__.py.bkp   read     TO_DO_LIST.txt
diag       __init__.pyc      README   tool_kit
export     io                sim      visu
```

```
slices = pn.read.slices(field='uu1', extension='xy')
slices.t
slices.xy.uu1

# Read multiple slices and extensions.
slices = pn.read.slices(field=['uu1', 'bb2’], extension=['xy',
'yz’])
slices.xy.bb2
slices.yz.uu1
```

Sim Objects

Sim objects are very powerful simulation objects.
Capabilities:

1. Read the simulation structure.
2. Modify simulation parameter (*.in files).
3. Modify the cparam.local.
4. Compile.
5. Start, stop and submit (SLURM) a simulation.
6. Create a new simulation and copy it.

```
import pencilnew as pn
sim = pn.sim.get()
ts = sim.get_ts()
sim.change_value_in_file('start.in', 'ampl', 2.0)
sim.change_value_in_file('cparam.local', 'nxgrid', 128)
sim.compile()
sim.bash('start.csh')
```

Sim Objects

Sim object work flow:

```
import pencilnew as pn
sim = pcn.get_sim('./DUMMY')
for vari in [1, 2, 3]:
    new_sim = sim.copy('new_sim' + str(vari))
    new_sim.change_value_in_file('start.in', 'variable', vari)
    new_sim.compile()
    new_sim.bash('mkdir data; pc_run')
```