MOCASSIN Voronoi-visualisation quickstart guide

David Hubber

December 29, 2016

1 Overview

The `voronoi_visualisation` program is a small python/C++ tool designed to visualise the results of simulations performed with the MOCASSIN code (`http://mocassin.nebulousresearch.org/`), in particular with the new Voronoi version. It has functions to

- Read in data generated from MOCASSIN via a python script
- Generate a Voronoi tessellation of the data for generating images
- Plot slices or column-integrated images from MOCASSIN output

2 Dependencies

If you are not interested in the Python library, you need only a C++ compiler. In order to use all features in GANDALF, the following programs and libraries must be installed :

- C++ compiler
- Python 2.7
- swig compatible with python 2.7
- matplotlib compatible with python 2.7
- numpy
- scipy
- voro++ library

3 Makefile

A small Makefile is included, which may need to be modified in order to successfully compile the python library.

- CPP : C++ compiler.
  g++ : GCC C++ compiler
  clang : CLANG compiler
  icpc : Intel C++ compiler
- PYTHON : name of python command-line executable (e.g. python, python2.7)
- VORO++ : Absolute path of voro++ library `src` directory (same Voronoi library as used in MOCASSIN. This can simply be set to point to the `voro++` library in the MOCASSIN directory.)

3.1 Compilation

To compile the visualisation code, after installing the required packages and setting the correct Makefile variables, simply type :

```
make
```

in the command line. The compiler should run both swig and the C++ compiler before finally linking all object files into a python shared-object file `_Voronoi.so`.

4 Basic usage

The visualisation library can generate images from files generated during a MOCASSIN simulation. In particular, it needs to load in two files (located in the `output` directory) :

- The file containing the positions of all the points that were used in the original MOCASSIN simulation, `grid4.out`
- The file containing all line data for each point from the MOCASSIN simulation, '`plot.out`'.

To use the viewer, a series of python commands must be carried out to create the various objects, load in the input files and generate the plots. The following example (`example1.py` in the main directory) shows the basic usage for loading in the files and generating a rendered plot.

`import numpy as np`	Import numpy library
`from voronoiviewer import *`	Import visualisation library
`pfile = 'grid4.dat'`	Name of file containing points data
`lfile = 'plot.dat'`	Name of file containing line data
`N = 1000000`	No. of points
`Ncol1 = 4`	No. of columns in point data file
`Ncol2 = 8`	No. of columns in line data file
`L = 15.0`	Size of box in pc
`vor = VoronoiViewer()`	Create a view object from the library
`vor.SetBoundingBox(-L, L, -L, L, -L, L)`	Set the bounding box volume for creating the tessellation
`vor.SetScaleFactor(3.08e18)`	Set the length scale in units of cm (e.g. here pc in cm)
`vor.LoadMocassinSnapshot(N, Ncol1, Ncol2, pfile, lfile)`	Read in the input data (both the points and lines)
`columnData = vor.RenderColumn(3, xmin=-L, xmax=L, ymin=-L, ymax=L, logscale=True, autoscale=True)`	Create a render plot of (not normalised) column-integrated line intensities and record the pixels in the numpy array `columnData`
`sliceData = vor.RenderSlice(3, xmin=-L, xmax=L, ymin=-L, ymax=L, zslice = 0.0, logscale=False, autoscale=True)`	Create a slice through the tessellation and save to the numpy array `sliceData`.

For full details of all commands and options (in particular for the various optional parameters for the `RenderColumn` and `RenderSlice` commands), it is recommended to open and read the `voronoiviewer.py` file directly inside the main `voronoi_visualisation` directory.