LA-UR-18-23285

Title: VORONOI

Author(s): Livingston, Daniel Reece

Intended for: Web

Issued: 2019-01-08 (Rev.2) (Draft)

V O R O N O I

A massively parallel Voronoi tessellation generator, built for subsurface flow and transport codes

Download

View on Github

Version 1.0.0 from October 22, 2018.
Zip-archive, includes all source code and documentation

Augment VORONOI with LaGriT , a powerful 2D and 3D unstructured finite element mesh generation suite

Massively Parallel

Built on PETSc and MPI, VORONOI is made to be fast. Compute control volumes for a 10^6 million node mesh in under 1.5 seconds*. And using PETSc's distributed sparse matrices as the core data type for computation, local node memory is fast, cheap, and is only gathered once.

Cross-Solver Compatibility

Use VORONOI with PFLOTRAN, FEHM, or TOUGH to take any Delaunay triangular, tetrahedral, quad or hex mesh and make it ready for use.

LaGriT Integration

With VORONOI, all the power of LaGriT is embedded into the binary - so you can pass in any LaGriT input file, and VORONOI will run the script and generate the control volumes on the CMO. In addition, VORONOI is now available as an output subroutine in LaGriT.

Solver-Specific Features

With FEHM, STOR file size can be reduced by removing any coefficients below a threshold and 'merging' duplicate coefficients.
In TOUGH, the ISOT and ROCK blocks are automatically generated based on mesh characteristics.

Control Volume Visualization

Output your Voronoi or median tessellation in the VTK XML file format, ready for viewing in many popular 3D visualization software programs.

Open Source

VORONOI is licensed under BSD-3 , and is absolutely free for personal or commercial use.

GitHub

License

Contact

EES-16 | Computational Earth and Environmental Sciences

LA-CC 1234-567-890

About

VORONOI was built from the ground up for subsurface flow and transport simulators; with fast, accurate conversion of unstructured finite element meshes into Voronoi and median tessellations.

Overview

VORONOI is a parallel, scalable control volume tessellation generator, built for use in subsurface flow and transport solvers.

The program accepts a Delaunay mesh composed of one of four element types: triangles, tetrahedrals, quads, or hexes.

It then finds the Voronoi dual or median of that mesh, and writes it in a solver-specific format:

1. FEHM (`.stor`)
2. PFLOTRAN (`.uge`)
3. TOUGH (`MESH`)
4. HDF5 (for general use; `.h5`)

Features

- Accurate calculation of Voronoi/Median cell volume and cell face area in 2D and 3D
- Reads AVS and LaGriT input files (.lgi)
- Outputs to FEHM, PFLOTRAN, TOUGH, and HDF5
- Outputs visualization of control volume cells
- Assigns ROCK coefficients based on node attributes (TOUGH)
- Automatic element assignment of isotropic / anisotropic permeability coefficients (TOUGH)
- Removes duplicate and negligable coefficients (FEHM)
- Built with PETSc for parallel execution and sparse matrix data types
- Cross-platform & open source

Building

Steps to compile VORONOI on a POSIX-compliant OS or shell.

Required software

- Fortran 95 compiler (`gfortran > 4.0; ifort >= 14`)
- C/C++ compiler (`gcc; icc`)
- MPI
- git
- Cmake

Build steps

1.0. Ensure that all required dependencies are fulfilled.

Building PETSc

2.0. Clone and build PETsc, using the `xsdk-0.2.0` branch:

```
https://bitbucket.org/petsc/petsc petsc
cd petsc
git checkout xsdk-0.2.0
```

2.1. Run `./configure` to configure the build with environment-specific settings. As an example,

```
./configure --download-mpich=yes --download-metis=yes --download-parmetis=yes
```

to configure your build with requests to download MPICH, METIS, and PARMETIS.

More information can be found on the official PETSc website.

2.2. Export `PETSC_DIR` and `PETSC_ARCH` variables to your environment. These values are printed to stdout at the end of the configure process:

```
xxx=========================================================================xxx
 Configure stage complete. Now build PETSc libraries with (gnumake build):
   make PETSC_DIR=/opt/atlassian/pipelines/agent/build/petsc PETSC_ARCH=arch-linux2-c-opt
all
xxx=========================================================================xxx
```

This is important as VORONOI will need to reference these variables during its build stage.

2.3. Finally, `make` PETSc:

```
make all
```

Building LaGriT

3.0. Build LaGriT libraries for use in VORONOI. First, download the LaGriT repo from GitHub:

```
git clone https://github.com/lanl/LaGriT.git
```

Building VORONOI

4.0. Build VORONOI by simply running

```
make voronoi
```

Your executable will be built in the repo `src/` dir; or, in other words, `$VORONOI_SRC_DIR/src/voronoi`.

Contributing

While VORONOI comes with a suite of features and capabilities, everyone's workflow is different. To make VORONOI the best Delaunay conversion framework in its class, contributions and comments are encouraged.

How to Contribute

Pull requests for missing features, bug fixes, and documentation updates are more than welcome. To contribute, please submit a pull request at

```
github.com/lanl/voronoi
```

To ensure the best chance of acceptance for your pull request, please ensure that:

1. All continuous integration tests pass
2. Your code is well documented and commented
3. If adding new functionality, that unit tests have been included in the PR
4. That the purpose and scope of your contributions are well explained in the PR
5. That your code has minimal/no dependencies on extraneous libraries / standards (i.e., outside of the C++11/F95 standard, MPI, LaGriT, or PETSc)

Submitting Feedback

If you notice missing features or bugs but cannot contribute, please submit an issue on the GitHub Issues page. If reporting a bug, include your OS type/version, shell type, FC/CXX compiler name/version, and steps to reproduce.

To directly reach the developers, please email livingston@lanl.gov.

Getting Started

Quick tutorial on getting up and running fast.

Execution Overview

Follow the documentation on building VORONOI and run the test suite to verify build integrity.

General usage for executing VORONOI follows this pattern:

```
mpiexec -np N voronoi [commands...]
```

where `N` is the number of MPI nodes desired.

A list of available commands can be found by running:

```
voronoi -h
```

Examples

To generate a VORONOI tessellation in the FEHM format from an AVS mesh, run

```
mpiexec -np 4 voronoi -avs my_mesh.inp -type fehm -o my_mesh.stor
```

To build a median tessellation in a PFLOTRAN format with a LaGriT infile, run

```
mpiexec -np 4 voronoi -lg infile.lgi -type pflotran -o geom.uge -cv median
```

Release Notes

Development notes for current and past releases.

1.0.0

October 21, 2018

- Initial release

Tools

Supplementary utilities to help visualize and process results.

Visualizing the sparse matrix

Found in: `viewSparseMatrix.py`

This script visualizes a sparse matrix for a given mesh - this is a handy way to quickly view the i,j connections between nodes.

Voronoi cell volumes are represented on the matrix diagonal, while the off-diagonals represent: (Voronoi area_ij/interface_ij length).

In the current implementation, an NxN sparse matrix is converted to an NxN image - be mindful of very large meshes as to not run into memory issues.

References:

[1] https://ieeexplore.ieee.org/document/112361/
[2] https://en.wikipedia.org/wiki/Sparse_matrix

Mesh conversion to AVS

Found in: `createMeshFromData.py`

This script converts a file of node (x,y,z) values and element (i,j,k,...) indices into an AVS-UCD mesh.

Node input file should be in the form:

```
x0 y0 z0
x1 y1 z1
...
xN yN zN
```

Element input file should be in the form:

```
i1 j1 k1 ...
i2 j2 k2 ...
...
iN jN kN ...
```

A delimiter between entries may be specified with the `-d` argument. Defaults to space (`' '`).

It is recommended that the element file reference the node list with 1-based indexing; that is, the element input file should reference the node `x0 y0 z0` as `1`. If you use zero- or N-based indexing, use the `--index` flag to indicate this.

This script will automatically assume that an element file with 3 integers on a line refers to a triangle element, and 4 integers refers to tetrahedrons. If you wish to manually specify what the element type is (i.e., quad) then use `--type ['tet', 'tri', 'quad', 'hex']`. Note that only one element type per file is allowed - mixing of element types in a single mesh is not supported in Voronoi.

If you only have a list of nodes, this script will still write out a file - but no element connectivity will be defined. You can import the mesh into LaGriT and triangulate it, or use the SciPy Delaunay function.

If you wish to view the created mesh, it is recommended that you use ParaView: https://www.paraview.org

Testing strong / weak MPI scaling

Found in: `viewMPIscaling.py`

This script tests against strong or weak scaling, and formats the results into:

1. A set of .PNG plots, generated with Matplotlib, and/or
2. A table of the form

```
=================================
  n        t(n)      E_fehm(n) ...
---------------------------------
  1       9.763s    1.0000
  2       4.901s    0.7540
  4       2.731s    0.7540
  8       1.220s    0.7540
  16      0.813s    0.7540
  32      0.763s    0.7540
  64      0.763s    0.7540
```

Usage

Command line usage for operating VORONOI.

Command Line Usage

Command	Functionality
-avs [INFILE.inp]	AVS-UCD mesh to read
-lg [INFILE.lgi]	LaGriT infile to run
-type [fehm,pflotran,tough2,hdf5]	filetype to write to
-cv [voronoi,median]	control volume type
-d	write mesh statistics to stdout

-o	filepath to save geometric coefficient data
-compress	coefficient compression (FEHM only)
-dedud	epsilon coefficient removal (FEHM only)

More detailed information can be run by calling

```
voronoi -h
```

Examples

1. Converting a Delaunay AVS mesh to an FEHM sparse matrix

As VORONOI has a native AVS reader, use the flag `-avs [file]` to input any AVS mesh. Choose the file output type using the `-type [type]` flag. In this case, we are using FEHM. Finally, the output filename can be chosen using the `-o [output]` argument.

```
mpiexec -np 4 voronoi -avs my_mesh.inp -type fehm -o my_mesh.stor
```

2. Converting a LaGriT input file CMO to a median mesh

Given a LaGriT input file, VORONOI can perform a voronoi/median/hybrid calculation on the current mesh object (CMO) active at the close of file.

Consider the following example, where a tetrahedral Delaunay cube is generated:

`lagrit_test.lgi`:

```
* TEST connect (3d) (lagrit_input_connect)
* LaGriT input file to generate an orthogonal grid on
* a unit cube. Just change nx,ny,nz to change the resolution.
*
define/nx/20
define/ny/20
define/nz/20

cmo / create / cmo / / / tet
rz / xyz / nx, ny, nz / 0. 0. 0. / 1. 1. 1. / 1 1 1
cmo / setatt / cmo / imt / 1 0 0 / 1
connect / noadd
resetpts / itp

* begin compare here
cmo/status
cmo/printatt//-all-/minmax
quality
finish
```

The control volume on the final mesh can then calculated with the `-lg [infile]` flag:

```
mpiexec -np 4 voronoi -lg lagrit_test.lgi -type tough2 -o MESH
```

Publications

Citing VORONOI & Resulting Publications Thereof

BibTeX format

```
@Misc{voronoi-web-page, Author = "Daniel R. Livingston, Carl W. Gable,
Satish Karra, Manuel Sentis, Michael J. Hannon, Rajesh Pawar", Title = "
{VORONOI} {W}eb page", Note = "http://www.github.com/lanl/voronoi", Year =
"2018" }
```

Publications

1. *Hannon, M. et al. Permeability Estimates of Fractured Cylindrical Samples by the Analysis of Pressure Transients. Proceedings of the TOUGH Symposium 2018, Berkeley, CA, Oct 8-10, 2018*
2. *Livingston, D. et al. VORONOI: An Orthogonally-Constrained and Numerically Accurate Processor for Subsurface Codes. Proceedings of the TOUGH Symposium 2018, Berkeley, CA, Oct 8-10, 2018*

Contact

Contact Information

For bug reports and feature requests, submit a ticket at GitHub Issues.

For everything else, contact Daniel Livingston at livingston@lanl.gov

Development Team

- Daniel Livingston, *Los Alamos National Laboratory*
- Carl Gable, *Los Alamos National Laboratory*
- Satish Karra, *Los Alamos National Laboratory*
- Manuel Sentis, *Swiss Federal Safety Inspectorate*
- Michael Hannon, Jr., *Indiana Geological & Water Survey*
- Rajesh Pawar, *Los Alamos National Laboratory*

Related Web Pages

- VORONOI GitHub
- LaGriT Mesh Generation Toolkit
- Meshing for Geological Applications (Los Alamos National Laboratory)
- Computational Earth Science (EES-16) at Los Alamos National Laboratory
- FEHM
- PFLOTRAN
- TOUGH