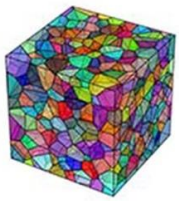
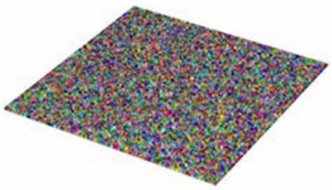


DCC Generator Tool



Manual

The code *DCC Generator Tool* addressed the practical needs of creating discrete (combinatorial) cell complexes (DCCs) based on the both regular (cubic or octahedron) and Laguerre-Voronoi tessellations of space provided by the Neper software. In particular, Voronoi tessellations with the corresponding DCCs provide a very close representation of the real material microstructures and are widely used in molecular dynamics and other types of simulations. Such complexes arise from the tessellations of spatial domains around arbitrary sets of points, which ensure that each 1-cell is in the boundary of exactly three 2-cells and three 3-cells, and each 0-cell is in the boundary of exactly four 1-cells, six 2-cells and four 3-cells.

1. DCC definition and algebraic representation

An excellent simple introduction to the area of DCCs with their various applications is given in the book of Leo Grady and Jonathan Polimeni "*Discrete Calculus. Applied Analysis on Graphs for Computational Science. (2010)*. Below are just a few notes necessary for understanding the output of the code.

- In algebraic topology, a discrete topological n-complex is a collection of cells of dimensions $k \leq n$, where every k-cell for any $0 < k \leq n$ has a boundary formed by (k–1)-cells belonging to the complex. The co-boundary of every k-cell for $0 \leq k < n$ is the set of (k+1)-cells whose boundaries contain the k-cell. In this terminology, 1-complex is a graph. Polyhedral complexes are a special class of regular quasi-convex discrete topological complexes, in the geometric realisation of which 0-cells are identified with points or vertices, 1-cells with line segments or edges, 2-cells with planar polygons or faces, 3-cells with polyhedra or simply cells, etc. We restrict our consideration to the polyhedral 3-complexes whose 3-cells are convex polyhedra with 2-cells in the boundary of exactly two 3-cells. An assembly of polyhedrons is a geometric realisation of a combinatorial structure referred to as a cell complex in algebraic topology.

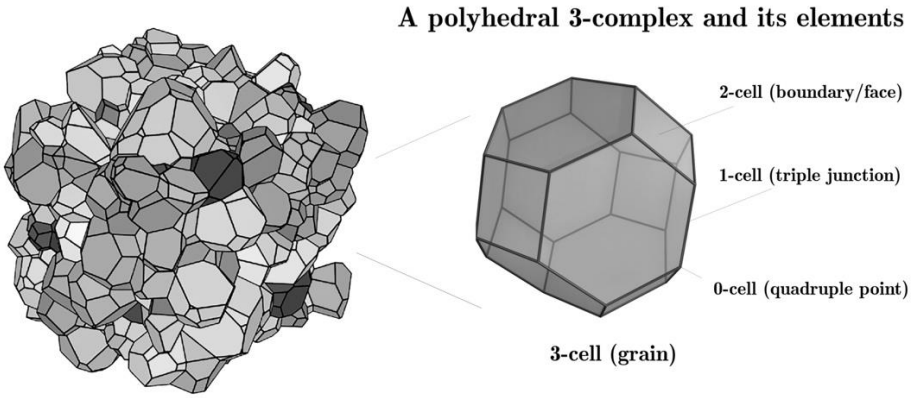


Figure 1: DCC complex and its structural elements [3].

- The geometric properties of the DCC are encoded in the volumes of different cells: 1 for 0-cells, length for 1-cells, area for 2-cells, and volume for 3-cells. The topological properties of the DCC are encoded in the boundary operator B_k, which maps all (k+1)-cells to the k-cells in their boundaries, taking into account cell orientations. The algebraic realisation of the operator for the {k, k+1} pair of cells is referred to as the k-th incidence matrix, B_k, which has N_k rows (where N_k denote the number of k-cells in a complex) and N_{k+1} columns and contains 0, 1, -1, indicating non-adjacency, adjacency with agreeing and with opposite orientations, respectively, between k-cells and (k+1)-cells. The transpose of the k-th incidence matrix, $b_k = (B_k)^T$, is a matrix representing the k-th co-boundary operator, which maps all k-cells to the (k+1)-cells in their co-boundaries.

- A standard way is to decide on a consistent orientation of all top-dimensional cells, e.g., to select the positive orientation to be from interior to exterior of the 3-cells and assign arbitrary orientations for all lower-dimensional cells. There are exactly three options for the relation between k-cell and (k+1)-cell in an oriented complex: they are not coincident - encoded by 0; the k-cell is on the boundary (k+1)-cell, and they have consistent orientations, encoded by 1; the k-cell is on the boundary (k+1)-cell and they have opposite orientations, encoded by -1. The transpose of the k-th incidence matrix is a matrix representing the k-th co-boundary operator, which maps all k-cells to the (k+1)-cells in their co-boundaries.

- The k-th combinatorial *Laplacian* (Laplace–de Rham operator) can be written as $L_k = b_{k-1} B_{k-1} + B_k b_k$ and it maps all k-cells to themselves, collecting local connectivity information. One important application of the combinatorial Laplacians is in calculating combinatorial curvatures. Since the Laplacians are symmetric positive semi-definite matrices, their eigenvalues are real. The spectra of eigenvalues can be used to classify discrete topologies, with two topologies considered equivalent when they have the same Laplacians' spectra.

2. Terminal commands

The usual terminal app can launch the code on MAC, Windows or Linux operational systems. The first two commands create the needed environment specified in the file requirements.txt

```
conda create --name neper-env --file requirements.txt
conda activate neper-env
python sparsemat.py --file <filename.tess> --dir <my_dir>
```

Here <filename.tess> is the full path (including the file name) to the **.tess* file generated by Neper software, and <my_dir> is the full path to the chosen output directory. Finally, all output files will be written to the <my_dir> output directory.

The required Conda packages can be downloaded and installed directly from the Anaconda and Minicoda projects webpages.

The code has been tested for the Neper output **.tess* files generated by Neper's versions 4.3.1 and 4.5.0.

3. Output files

The code generates a sparse representation of matrices: for any matrix element a(i,j) = c the files of the matrices contain the list of triplets in the form (i,j,c). The enumeration of indices starts from 0, and, for instance, the line "5, 7, 1" in the adjacency matrix A_k means that the *k*-cell #6 is the neighbour of the *k*-cell #8. For any incidence matrices B_k, the same triplet "5, 7, 1" means that the (*k*-1)-cell #6 is on the boundary of the *k*-cell #8, and their orientations coincide (c = -1 for the opposite orientations).

The Voronoi tesselation provided by Neper supposed to be a *dual* complex in terms of algebraic topology and so all the other tessellations provided by the Neper output with the morphology option -morpho <morphology> like *cube*, *square*, *tocta*, *lamellar*, etc. different from *voronoi*.

For the dual (voronoi, cube, tocta, etc.) complex:
All sparse matrices are stored in **.txt* files.

A0.txt - 0-adjacency matrix for 0-cells (vertices)
A1.txt - 1-adjacency matrix for 1-cells (edges)
A2.txt - 2-adjacency matrix for 2-cells (faces)
A3.txt - 3-adjacency matrix for 3-cells (polyhedra)

B1.txt - incidence matrix or boundary operator (0-cells are row indexes, 1-cells are columns)
B2.txt - incidence matrix or boundary operator (1-cells are row indexes, 2-cells are columns)
B3.txt - incidence matrix or boundary operator (2-cells are row indexes, 3-cells are columns)

In the 2D case, there are no A_3 and B_3 matrices.

voro_Ncells.txt - each row in the file corresponds to the numbers of different *k*-cells: the first row is the number of 0-cells, second - is the number of 1-cells and so on.

voro_normals.txt - components of a normal vector for each face written in the format: (face_id a b c), where *face_id* coincide with the numeration of faces in A_2 and B_2 matrices; a, b and c - are the components of the normal vector of a face in a 3D cartesian coordinate system.

voro_seeds.txt - coordinates of the seed points of 3-cells used for Voronoi tessellation of space.

For a primal (delaunay triangle, cubic, etc.) complex:
All sparse matrices are stored in **.txt* files.

AC0.txt - adjacency matrix for 0-cells (vertices)
AC1.txt - adjacency matrix for 1-cells (edges)
AC2.txt - adjacency matrix for 2-cells (faces)
AC3.txt - adjacency matrix for 3-cells (polyhedra)

BC1.txt - incidence matrix or co-boundary operator (0-cells are row indexes, 1-cells are columns)
BC2.txt - incidence matrix or co-boundary operator (1-cells are row indexes, 2-cells are columns)
BC3.txt - incidence matrix or co-boundary operator (2-cells are row indexes, 3-cells are columns)

In the 2D case, there are no AC_3 and BC_3 matrices.

delau_Ncells.txt - each row in the file corresponds to the numbers of different k-cells: the first row is the number of 0-cells, second - is the number of 1-cells and so on.

delau_seeds.txt - coordinates of the seed points of 3-cells used for Delaunay tessellation of space.

For the whole complex:
L0.txt - 0-Laplacian matrix with the dimension of 0-cells (vertices)
L1.txt - 1-Laplacian matrix with the dimension of 1-cells (edges)
L2.txt - 2-Laplacian matrix with the dimension of 2-cells (faces)

4. Tips and tricks

- The metric information like the volumes of all 3-cells and areas of all 2-cells can be obtain directly from the Neper output statcell and *statface* options with the corresponding keys like -statcell vol -statface area or providing the corresponding values for every *k*-cell in the complex. In this case, the terminal command may look like

```
neper -T -n 300 -id 1 -dim 3 -statcell vol -statface area
```

  Please, see more examples on the Neper webpage.

- Using the file seeds.txt with some specific set of seed points a new Neper tessellation can be performed. The terminal command creating a complex with coordinates of the seed points as the centres of 3-cells may looks like

```
neper -T -n <number of seeds> -id 1 -statcell vol -statface area -domain "cube(1.0,1.0,1.0)" -morphooptiini
"coo:file(seeds.txt)"
```

  You must call Neper from the folder (cd <path to the directory containing the file "seeds.txt">) containing the seeds.txt file, or write the whole path instead of the file name in the coo:file(<path to seeds.txt>) command.

- More flexibility in the tesselation provide the transformation options of the Neper. In particular, for the creation of a 2D complex as a plane cut of the 3D one, the slice(d,a,b,c) function can be used as it is shown below for the half-cut of the Voronoi complex containing 1000 grains:

```
neper -T -n 1000 -id 1 -domain "cube(1.0,1.0,1.0)" -transform "slice(0.5,0,0,1)" -dim 3 -statcell area; \
neper -V n1000-id1.tess -datacelltrs 0.5  -print DCC_slice
```

  Here *d*, *a*, *b* and *c* are parameters in the corresponding equation of a plane $ax + by + cz = d$ and it is worth to be mentioned here that the normal vector of this plane is $n = (a,b,c)$.

5. Examples

The folder with several examples contains discrete cell complexes already created by Neper and processed with *DCC Generator Tool*. The command below shows three terminal commands launching the creation of the Vorinoi dual complex containing 5000 3-cells and its further processing using DCC Generator Tool:

```
neper -T -n 5000 -dim 3 -id 1 -ori uniform -statcell vol -statface area; \
neper -V n5000-id1.tess -datacelltrs 0.5  -print DCC_voronoi_5000; \
python <path/to/the/directory>/Voronoi_DCC_Analyser/sparsemat.py --file <path/to/the/directory>/n5000-id1.tess –dir
<path/to/right/directory>
```

All the amounts of *k*-cells in the complex can be taken directly from the voro_Ncells.txt or delau_Ncells.txt files.

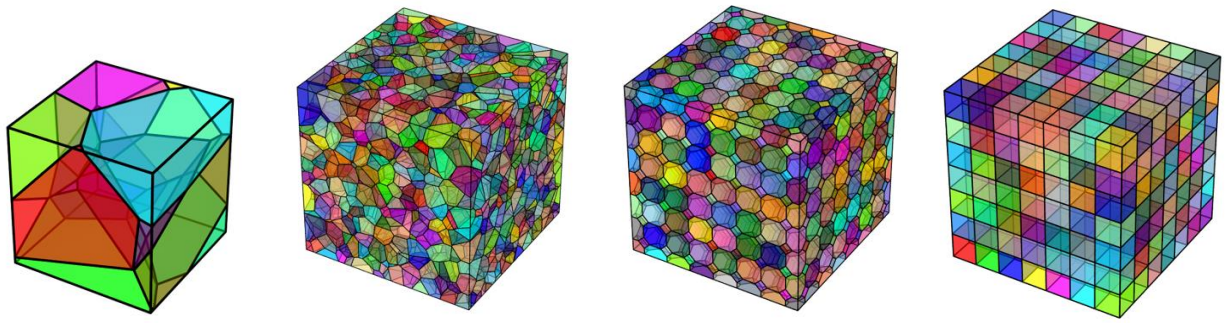


Figure 2: Examples of some tessellations provided by Neper.

6. Applications of DCCs

[1] K. Berbatov, P.D. Boom, A.L. Hazel, A.P. Jivkov, 2022. Diffusion in multi-dimensional solids using Forman's combinatorial differential forms. Applied Mathematical Modelling 110, 172-192. *doi: 10.1016/j.apm.2022.05.043*

[2] P.D. Boom, O. Kosmas, L. Margetts, A.P. Jivkov, 2022. A geometric formulation of linear elasticity based on discrete exterior calculus. International Journal of Solids and Structures 236–237, 111345. *doi: 10.1016/j.ijsolstr.2021.111345*

[3] E.N. Borodin, A.P. Jivkov, A.G. Sheinerman, M.Yu. Gutkin, 2021. Optimisation of rGO-enriched nanoceramics by combinatorial analysis. Materials & Design 212, 110191. *doi: 10.1016/j.matdes.2021.110191*

[4] S. Zhu, E.N. Borodin, A.P. Jivkov, 2021. Triple junctions network as the key structure for characterisation of SPD processed copper alloys. Materials & Design 198(24), 109352. *doi: 10.1016/j.matdes.2020.109352*

[5] D. Šeruga, O. Kosmas, A.P. Jivkov, 2020. Geometric modelling of elastic and elastic-plastic solids by separation of deformation energy and Prandtl operators. International Journal of Solids and Structures 198, 136–148. *doi: 10.1016/j.ijsolstr.2020.04.019*

[6] E. N. Borodin, A. P. Jivkov, 2019. Evolution of triple junctions' network during severe plastic deformation of copper alloys – a discrete stochastic modelling. Philosophical Magazine 100(4), 467-485. *doi: 10.1080/14786435.2019.1695071*

[7] I. Dassios, G. O'Keeffe, A.P. Jivkov, 2018. A mathematical model for elasticity using calculus on discrete manifolds. Math. Methods Appl. Sci. 41(18), 9057– 9070. *doi: 10.1002/mma.4892*

Acknowledgements
This code has been created as a part of the EPSRC funded projects EP/V022687/1 "*Patterns recognition inside shear bands: tailoring microstructure against localisation*" (PRISB) and EP/N026136/1 "*Geometric Mechanics of Solids: new analysis of modern engineering materials*" (GEMS).

Contacts
You are very welcome send e-mail to Dr Oleg Bushuev for technical support or Dr Elijah Borodin for any other questions.