Polytopal Cell Complex (PCC) Processing Design (CPD code\*)

Technical Manual

Manual version 0.3.0 26-Mar-2024

**The code online source repository:** <https://github.com/PRISBteam/PCC_Processing_Design>

**The code reference example:**

Elijah Borodin (2024) *Polytopal Cell Complex (PCC) Processing Design (CPD code)*, (Version 3.0) [Computer software]. Zenodo. <https://github.com/PRISBteam/PCC_Processing_Design>

Contents:

1. Project Folders

2. Modules and Libraries

3. Functions, Measures and Classes

4. Configuration files

5. Tasks and Tests

CPD code is fully written in C++ as a CMake project (can be launched using CMakeLists.txt file) and consists of several modules-libraries, configuration < \*.ini> files governing the code execution, sources of PCCs as discrete combinatorial spaces, and databases of material parameters. For successful code execution, both the C++ compiler and CMake software must be installed. The code is tested for C++17 and works well with CMake 3.23 , g++ compiler and CLion IDE. The code uses explicitly the Eigen and Spectra libraries which must be downloaded (from <https://spectralib.org/download.html>) and copied to the directory containing all the STL C++ libraries on the local PC. The code works with a pre-created PCC (please see more details on the <http://materia.team/> project page) represented in the form of incidence and adjacency sparse matrices.

There is also parallelised with Open MP libraries version of the code useful for its effective execution simultaneously in several cores within one CPU. The Open MP libraries must be installed only for the execution of this particular version of the CPD code.

1. **Project folders (go above)**

The project directory contains several directories:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Fig. 1: Project directories of the CPD code.

**\*src*** — contains all the source files, <\*.h> libraries of the project and the main.cpp file. In particular, *\src* directory contains several subfolders with project libraries:

**\*lib*** — directory contains all the <\*.h> project libraries Each of the modules is placed in the folder with the same name as its own. These subdirectories contain, in their turn, <\*.h> and <\*.cpp> files with the same name as the corresponding module, and a subdirectory named \functions containing libraries with all the functions used in this particular module. Besides, it contains the library *SupportFunctions.h* with the additional functions used in several modules, *measures.h* with functions for calculations of various structural measures, and another library *Objects.h* as the only place in the project containing all the definitions of the code-specific classes. The subdirectory \ini contains the code-specific readers of the project's \*.ini files which nedds to be amended at their any significant changes. \other — contains additional external libraries such as the simple mINI reader of the \*.ini files.

**\external :: \mINI**

**\ini**

**\PCC\_Processing :: \functions**

**\PCC\_Characterisation :: \functions**

**\PCC\_Writer :: \functions**

**\*task*** — contains all the additional user-defined tasks written as separate functions and included in the *main.cpp*. These functions became active in the mode "TASK" instead of the "LIST" of the [simulation\_mode] option in the main.ini file.

**\*config*** — contains all the <\*.ini > files used for the initial definition of parameters governing the execution of the corresponding CPD modules;

**\*database*** — contains supplementary data files (like *materials.ini*) for each of the project modules containing physical parameters of various materials studied in simulation tasks;

**\*PCC\_sources*** — contains a few PCCs created based on the Voronoi tessellations of the 2D and 3D cubes. A large PCC library is published on the webpage (http://materia.team/) of the Materia software project. Please see also Readme file for the PCC Analyser Python code (<https://github.com/PRISBteam/Voronoi_PCC_Analyser>) which is part of the MATERiA codes software project.

1. **Modules and Libraries (go above)**

Considering as scientific software, CPD code consists of several ***modules***, some of which are mandatory (such as Main, Configuration reader, and Writer) and some are the *core libraries*, (such as PCC\_Processing and PCC\_Characterisation). The using of core modules is problem-dependent. Three *general project libraries*, whose functions are exploited in various project modules: PCC\_Support\_Functions, PCC\_Measures and PCC\_Objects. Such a *division in several modules is primarily logical structure* rather than formal, but it strongly correlates with the code’s architecture.

A diagram of a computer code

Description automatically generated

Fig. 2: Modular architecture of the CPD code.

*General project libraries:*

* **Main module**

*main.cpp* file is considered a separate module, setting the simulation environment and managing the implementation of all the other project modules. All the global variables and paths are defined here and then it subsequently calls all the other project modules marked “ON” in the config/*main.ini* file.

* **Configuration reader**

The module includes a single *ini\_readers* library whose functions read all the variables listed in the <\*.ini > configuration files (see project folder **\*config*** ) and save them into the corresponding variables defined inside the *main* module.

* **PCC\_Writer**

The module performed formatted output from each module in the core libraries (PCC\_Processing, PCC\_Characterisation, etc.) to the “output” directory specified in the *config/main.ini* file with the pre-defined (described in the module’s code) names.

* **PCC\_Measures**

It is a library of different mathematical measures of sets, vectors, graphs and other combinatorial objects used in the core project libraries. In particular, they used for comparison and finding max/min values of configuration entropies associated to different state vectors.

* **PCC\_Objects**

The model contains the definitions of all classes (structures and methods associated with them) used in the project. As the project priority is the achievement of the maximal possible computational efficiency (without too much harm on clarity and simplicity), the classes defined in this module rarely employed in the core libraries and primarily used for storage, processing and analysis of the obtain results. A typical example is the *Processed Complex* class as an output of the PCC\_Characterisation module and containing diverse set of vectors of different types.

* **PCC\_Support\_Functions**

Contains a pretty large list of various supplementary functions useful or necessary for different project modules. Most of them serve purely technical tasks such as checking “if the file <file\_name> exists in the directory or finding barycentre coordinates of a polyhedron.

*Core libraries:*

* **PCC\_Processing**

The module performs a classification of cells of different dimensions in a PCC by their labelling which is codified in the corresponding *State Vectors*. It can perform this both randomly or according to some governing principles such as maximum or minimum of a specific measure defined in the PCC\_Measures module. As its **output**, the module provides two vectors for each dimension *k* of the PCC’s k-cells: ***1.*** a *special cell sequence* where all the k-cells possessing *special* IDs are listed in the order of their appearance during the considered *process*, and ***2.*** their *Design Vectors* corresponding to the state vectors and containing *cell types* (labels) assigned for each of the k-cells. Please see the project Theoretical Manual for more details.

* **PCC\_Characterisation**

This module performs several complex analyses such as *Entropic* or *Topological* which output vectors of vectors with specific structural characteristics describing their changes during the simulated *process* and the final output of the module is the object of the *Processed Complex* class described in the PCC\_Objects. It contains all the calculated vectors ready for their writing to files by the PCC\_Writer module. The module takes as its input *Design Vectors* provided by PCC\_Processing module or reads from the file (as specified in the *config/characterisation.ini file*). Some of the module libraries (like *algebraic.h*) contain the corresponding functions which can be used in several types of analysis, while other (\*\_analysis) implement only one specific type of a structure characterisation.

1. **Configuration files and the Initial\_configuration (go above)**

All the initial settings are defined in the Main module as a single object ***initial\_configuration*** of a class (structure and methods) *Config* described in the Objects.h and Objects.cpp modules. This class contained:

1. Dimension of the problem (*dim* from 1 to 3);
2. Path to the source directory containing PCC;
3. Paths to the PCC files (*PCCpaths*) relative to the source directory;
4. Vector of integers (*ConfVector*) containing *dim* and exsecution keys (ON/OFF) for all the project modules;
5. Initial State vectors of inclusions (Configuration\_State) and “cracks” (Configuration\_cState).

***main.ini***

[simulation\_mode]

There are three different simulation modes of the Main module:

‘LIST’ - by default, launch all the modules one after another strictly according to the data from \*.ini files.

‘TASK’ - assumes tailored execution of the code using the functions, \*.cpp and \*.h files included explicitly inside the else if(task) {..} statement in the main.cpp module INSTEAD of the

‘TEST’ - …

‘TEST’ ‘LIST’ mode sequence of modules. The ”task” mode is supposed to provide scientific freedom of the code execution and can ignore any instructions listed in the \*.ini files.

[general]

* ‘dim = 3’ or 2 - is the problem dimension for 3D or 2D space tessellations, respectively.
* ‘source = … \’ set the path to the directory containing PCC in its algebraic representation as a set of all adjacency and incidence matrices with some additional data about the corresponding space tessellation such as polyhedra volumes, face areas, face normals, etc. It is important to use “\” symbol at the end of the source path!
* ‘output=…\’ set the output directory for the Writer module - where all the calculation results will be written. It is important to use “\” symbol at the end of the output path!

[modules]

All the rest in the main.ini file is only the list of all MODULES with the two variants: “ON” - for switching on the module execution, and “OFF” - for switching off the module execution.

[execution\_type]

‘e\_type = ..’ set ‘tutorial’ (with the education tour, tips and suggestions during the execution of the code – is intended only for the first acquaintance with the code) or normal ‘scientific’ execution mode.

Each of the following \*.ini files contain an almost similar list of settings for every type of cell in the tessellation and the corresponding PCC: polyhedrons (3-cells), faces (2-cells), edges (1-cells), nodes (0-cells). In the Processing module, any algorithm calculates as its output the lists (vectors) of “special” cells of different types described in the corresponding “state vectors”. The are three distinct sub-modules: (1) assigned structures: the algorithm picks cells and assigns them some type ID (label), writing the cell number in the corresponding special\_sequence (s\_sequence) vector (example: the random assignment of “special” type for some number of faces); (2) imposed structures: assigned types for low-dimensional (k-1)-cells or higher- dimensional (k+1)-cells according to some specific rule based on the already created assigned structures for k-cells (example: classification of face junctions according to the number of special faces incident to each junction); (3) induced structures: assigned types for the k-cells of the same dimension based on the already created assigned structures for k-cells (example: introducing fractured or cracked faces based on the initially assigned structure of faces containing inclusions).

***processing.ini***

The file is divided in several parts reflecting the dimensions of the cells:

[polyhedrons]

Contains only instructions for the assignment of the polyhedrons (3-cells) types. The set of parameters for polyhedra is in full analogy (possibly less in their number) with one for [faces]. Please read the detailed description below.

[faces]

Contains only instructions for the assignment of the faces (2-cells) types. Several parameters below define the settings for the Assignment type of Processing module.

* ‘face\_types\_number =..’ — the number of distinct face types (normally from 0 to 3) where 0 means that there are no special faces and the module does nothing here.
* ‘pf\_mode = ..’ — (if face\_types\_number >0) choose the specific processing type from the list of functions in the /src/lib/PCC\_Processing/functions directory.

S — reading from source \*.txt file (s\_sequence.txt ) the list of special faces (for this particular PCC) created before by some of the processing modes listed below;

* ‘source = /…/s\_cells\_sequence.txt’ — the path to the \*.txt file containing a list of numbers of faces of special types. This “source” affects only S processing mode and does not affect any other parts of the code.

R — simple random choice of new special faces during the assignment process; F — choice of new special faces governing by the maximum configuration entropy production principle (MEPP); D — choice of new special faces governing by the minimum configuration entropy production principle; Cr — determination of new special faces by effective random rotations of grains (applicable only for crystallography-related problems); Cm — determination of new special faces by effective rotations of grains governing by the minimum configuration entropy production principle (MEPP) (applicable only for crystallography-related problems); L — the random choice of new special faces with some restrictions that allow to the creation of elongated chains of special cells, whose lengths are normally distributed with the average “mu” and dispersion “sigma”.

* ‘pf\_index = 0’ — supplementary index for more flexibility in the code execution, it does not affect anything in the default mode;

The fractions from 0 to 1 for three possible face types specified above in the face\_types\_number parameter in their order. It is the fractions of special faces which will be assigned by the PCC\_Processing module. If there is only one special type, only fmax\_fraction1 should be above 0, face\_types\_number =1, and all the rest fractions will be ignored.

* fmax\_fraction1 = 0.9
* fmax\_fraction2 = 0.0
* fmax\_fraction3 = 0.0

The following statements set parameters for the Induced type of Processing module and for “historical” reasons called cracked faces. By default, there is a possibility to set only one type of such induced face. This part of the Processing module is always following (in execution time) after assignment one, and, by definition, uses the assignment face types for calculation of the corresponding list of induced face types.

* ‘crack\_types\_number = ..’ — similarly to the face\_types\_number set the number of types and currently only two options are allowed: 0 - there are no induced faces, and 1 - means that the induced part of the Processing module is “on”.
* ‘cf\_mode = Km’ — similarly to the pf\_mode set the specific mode of the choice of induced faces.

Km — currently only one mode of the “kinematic fracture” is allowed.

* ‘cfmax\_fraction = …’ — similarly to the fmax\_fraction parameters for assigned face types it sets the fraction of induced faces (in the range from 0 to 1).

[edges]

Containing only instructions for the assignment of the edges (1-cells) types. The set of parameters for edges is in full analogy (possibly less in their number) with one for [faces]. Please read the detailed description above.

[nodes]

Containing only instructions for the assignment of the nodes (0-cells) types. The set of parameters for nodes is in full analogy (possibly less in their number) with one for [faces]. Please read the detailed description above.

Finally, the [distribution] is a very special category relevant for the only case of the elongated chains of special cells, whose lengths are normally distributed with the average “mu = ..” and dispersion “sigma = ..”. It does not affect any other processing modes.

***characterisation.ini***

The characterisation module is divided into several "labs" [polyhedrons\_lab], [faces\_lab], [edges\_lab], [nodes\_lab], corresponding to the each type of k-cells in a 3-complex (PCC) with the similar set of structural characteristics:

* ‘l\_active = ..’ - switch on/off the calculations of the characteristics related to the corresponding k-cells (polyhedra, feaces, edges, nodes);
* ‘config\_entropy=..’ - calculation of the configuration entropy with its mean (if "S\_mean = 1") and deviatoric (if "S\_skew = 1") parts.

For the [edges\_lab] in addition the calculation of the imposed by faces characteristics of "j\_fractions = .." (special edge fractions), ‘d\_fractions = ..’ (special edge degree fractions), and "analytical = .." (analytical solutions) is posdible. Finally, [spectra\_lab] contains parameters of the corresponding Laplacian spectra. Here

* ‘calc\_steps\_numb = ..’ - set the number of points where the spectrum will be calculated; "laplacians = .." - switch on/off the calculation of the calculation of the corresponding matrix of the combinatorial Laplacian; "laplacians\_spectra = .." - switch on/off the calculation of the Laplacian's spectrum (the list of all its eigenvalues); "laplacians\_betti = .." - switch on/off the Laplacian's Betti numbers as the dimensions of its null-space (the number of zero eigenvalues).

***writer.ini***

The *writer.ini* file contains only Boolean-type parameters with a value equal to 1 means writing these characteristics to the corresponding file, and 0 means that the parameter will not be written.

[sequences]

* "isSequencesOutput = .."
* "isDesignvectorsOutput = .."

[entropic\_polyhedrons]

* "isPolyhedronFractions = .."

[entropic\_faces]

* "isFaceFractions = .."
* "isConfEntropy = .."

[entropic\_edges]

* "isConfEntropy = .."
* "isFractions = .."
* "isDegreeFractions = .."

[entropic\_nodes]

* "isNodeFractions = .."

[entropic\_analytical]

* "isEdgeFractions = .."
* "isEdgeConfEntropies = .."

[component\_analysis]

* "isBetti = .."

1. **Tasks and Tests (go above)**

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: a new analysis of modern engineering materials" (GEMS).

Distributed under the MIT License. See LICENSE.txt for more information. https://github.com/PRISBteam/

Please feel free e-mail to Dr Elijah Borodin (Lecturer in Solid Mechanics at the University of Manchester, Department of Solids and Structures) any queries relating with the code.