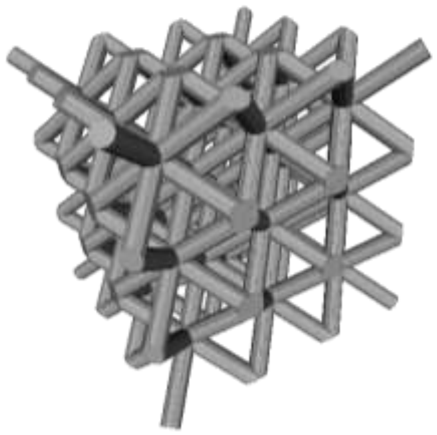
Lattice Generator Program

Draft Documentation



Early release version – check back for updated version

Background

The Lattice Generator is a simple MatLab program that automatically generates various lattice geometries direct to STL format. These periodic structures are intended for use with the metallic additive manufacturing technologies of Selective Laser Melting (SLM) or Electron Beam Melting (EBM), however could be applied to a number of other additive technologies that require the input of an STL file.

Developed out the apparent need for a fast and simple way to generate STL lattice structures for research purposes at RMIT University's Centre for Additive Manufacture, the program is a part of a preliminary work in developing a much larger lattice generation capability. In its current state, the program includes the following uniform lattice types:

- Body Centred Cubic (BCC)
- Face Centred Cubic (FCC)
- Modified FCC variants removing x-y, y-z, and x-z planes
- Combined BCC and FCC cell
- Modified combined BCC and FCC removing x-y plane
- Simple cube (box cell)

In theory, any lattice cell configuration could be added to the program provided that it can occupy a cubic volume and is able to be tessellated.

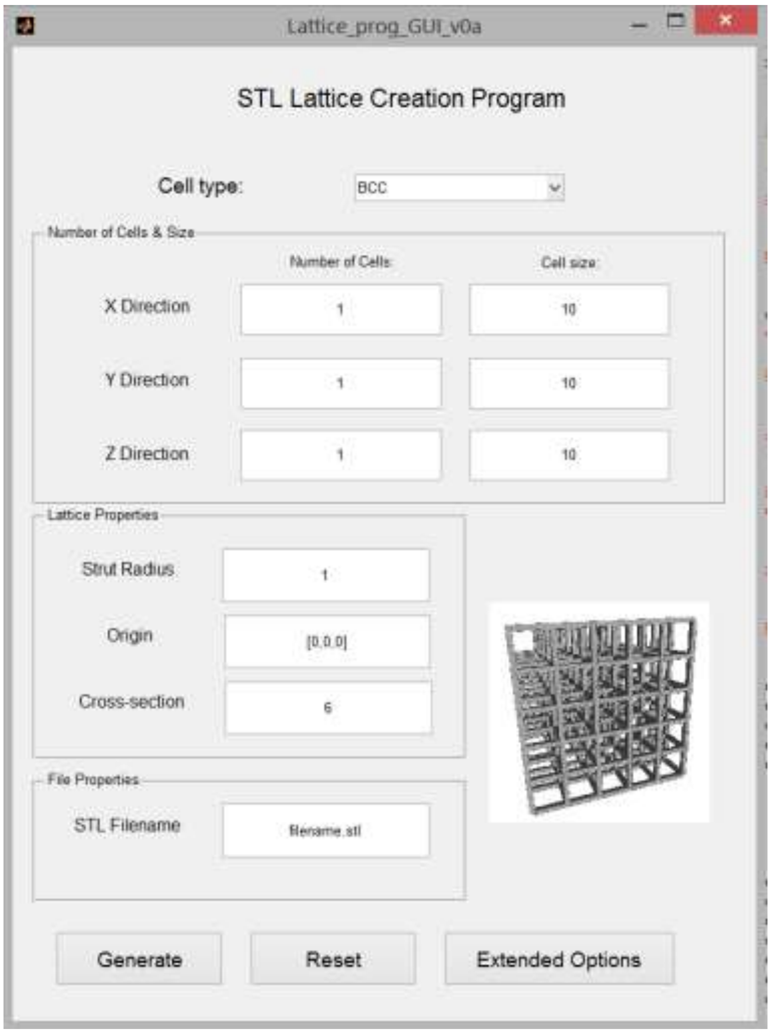
In addition, it was observed that creating Finite Element (FE) meshes was incredibly time consuming via traditional techniques. In response, the lattice generator script has the added functionality of generating FE meshes automatically in NASTRAN format. Other formats are possible, however for the purposes of the development project NASTRAN was all that was needed. In its current form, the lattice FE mesh is a 3D beam element mesh with the properties of the STL lattice applied. Loads or constraints are not generated, and the material properties are only set as a place holder. These aspects need to be modified in a pre-processor (HyperMesh, Patran etc.).

Basic Operation

The basic guided user interface (GUI) is the best place to start for first time users, as it contains all the elements needed to quickly generate an STL lattice. There are two form of the GUI entitled:

1. Lattice_Prog_GUI_vXX.fig ← Standard interface
2. Lattice_GUI_vXX-extended.fig ← Extended options

The basic version and recommended starting point is the first of these two options. When opened, the basic interface will show:



Cell type: BCC

Number of Cells & Size

	Number of Cells	Cell size
X Direction	1	10
Y Direction	1	10
Z Direction	1	10

Lattice Properties

Strut Radius: 1

Origin: [0,0,0]

Cross-section: 6

File Properties

STL Filename: filename.stl

Working down the page on this simple UI, the user first selects the cell type from the drop-down list. The current list of cell options are as follows:

The cell types are:

- Body Centred Cubic (BCC)
- Face Centred Cubic (FCC)
- Modified FCC variants removing x-y, y-z, and x-z planes
- Combined BCC and FCC cell
- Simple cube (box cell)

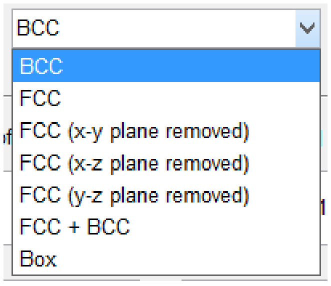


Figure 1: Cell options

Following cell selection the number of cells desired and the cell size in each of the X, Y, and Z axis can be set. The number of cells and size of cell in each axis are independent of each other, so any combination can be set.

Note: Currently the program generates the lattice cells independently from adjoining cells, so it is recommended to not generate more than 1000 cells from this interface. Future versions will remove this limitation with the use of symmetry.

The strut radius, origin of the lattice and cross-section order can then be set. For the purposes of this program, each of the elements connecting the lattice nodes are referred to as 'struts', with many struts making up each cell, and many cells making up the lattice.

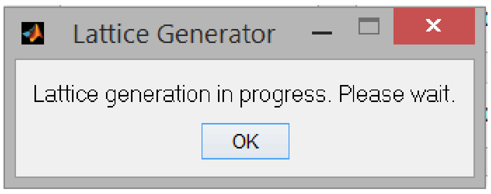
The lattice origin is defined as the location of the first node in the lattice, with every other node in the lattice being placed in the positive direction. For example, a lattice with an origin of [0,0,0] will extend entirely into the positive direction along each of the X, Y and Z axes. If the lattice was to be 100 units across in each axis and it was intended that the global origin be at the centre, then the origin should be set at [-50,-50,-50]. It is also important to note that Matlab syntax be observed here, enclosing the three data points in a comma separated square bracket array.

The strut cross-section is a unique feature of this program, and one of the principle reasons behind developing it. The cross-section parameter sets the cross-sectional polygon order of each of the lattice struts, and in-turn setting the total number of STL facets in the lattice. For example, an order of 6 (the default) will create struts with a hexagonal cross-section.

The final option is the setting of the STL file name, in which the lattice will be saved to. It is important to label the file with the extension ".STL" so that it is recognisable to other programs outside the Matlab environment. This file will be saved in the current Matlab working directory.

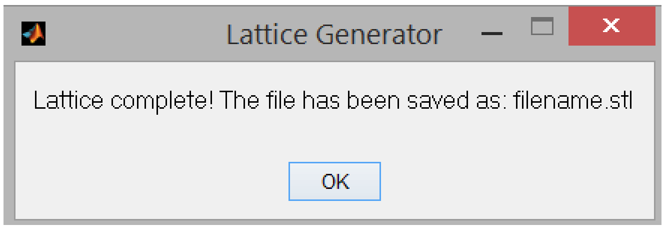
CAUTION: If there is a file with the same filename in the current working directory it will overwrite the file, deleting all previous data.

Following the setting of the desired parameters, click "Generate". While processing the user request the following dialog box will show:


Lattice Generator

Lattice generation in progress. Please wait.

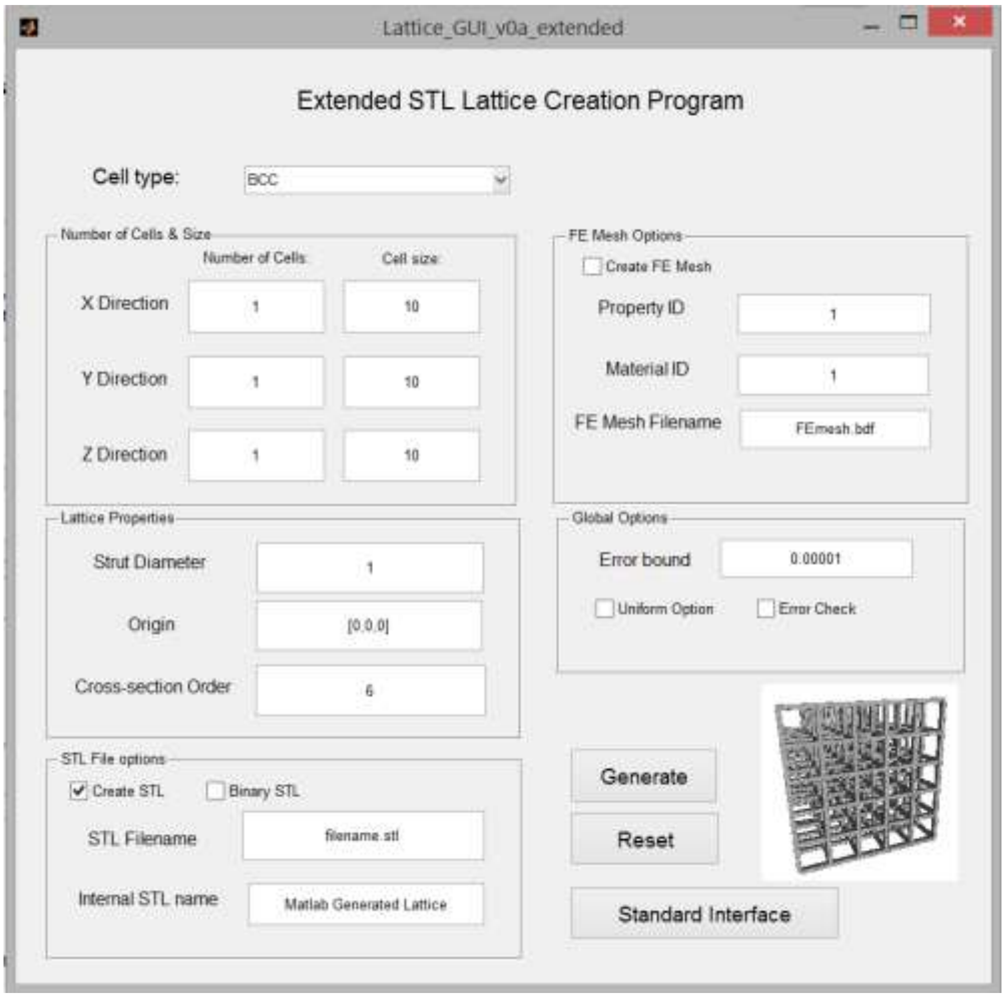
The time of lattice generation is dependent on the size of the lattice to be created and the available processing power of the user's PC. For larger lattices it may take some time (in the present configuration), so feel free to minimise this task. Once completed, the dialog box will show:


Lattice Generator

Lattice complete! The file has been saved as: filename.stl

Extended Version

As noted earlier, the extended version of the UI has additional options not accessible in the normal UI. Most significantly it contains the options to generate the FE mesh alongside or instead of the STL.



Cell type: BCC

Number of Cells & Size

	Number of Cells	Cell size
X Direction	1	10
Y Direction	1	10
Z Direction	1	10

FE Mesh Options

☐ Create FE Mesh

Property ID: 1

Material ID: 1

FE Mesh Filename: FEmesh.bdf

Lattice Properties

Strut Diameter: 1

Origin: [0,0,0]

Cross-section Order: 6

Global Options

Error bound: 0.00001

☐ Uniform Option ☐ Error Check

STL File options

☑ Create STL ☐ Binary STL

STL Filename: filename.stl

Internal STL name: Matlab Generated Lattice

To activate the FE mesh, select the “Create FE Mesh” option. The FE properties can be overwritten when pre-processing the mesh, so the ID numbers are not important. They should, however, be clearly distinguished from other parts of the input deck of a NASTRAN job.

The option ‘Error bound’ is a term added to account for a small amount of numerical rounding error inherit in numerical processes. This bounding term acts as an upper and lower variance in the position of nodes and vertices when calculating connecting struts and facets. The presence of this term does not affect the overall accuracy of the final solution, as double precision values are used throughout (the precision of an STL file by design is less than double precision).

Update 27/01/14: Binary file export, uniform cell replication and error checking have been added as an options to the GUI. Binary file export creates the STL in binary instead of the usual ASCII format, saving space and processing time. Uniform cell option is a new form of cell replication, in which one unit cell is created and copied numerous times to create the lattice. The error check option turns on a few checks in the STL write script. These checks do slow performance slightly and are only required in debugging.

Method of Solution

The first step in the lattice creation process is the assignment of a unit cell with the cell properties assigned in the input prompt. The unit cell is selected based on a pre-set type, and is modified in size to match the input parameters in each direction. The unit cell at this point only consists of nodal points, with a node located on each point where two or more struts would intersect. Following the creation of the unit cell, an array of cells is created by replicating the original cell to the total number of cells specified in each direction. Meanwhile, the STL file is created and initialized.

Once the node configuration has been set, the nodes are sorted then processed for connecting nodes based on the cell type. Every node has a unique identifier and a table of connecting nodes is then populated, defining every node that is to connect to another node for that particular lattice.

After establishing the nodal connections, the vertices for every facet that will be forming the individual struts must be defined. This is particular aspect of the program is the most critical and involved as every vertex must be calculated to accommodate all intersection struts at that nodal location. For example in a BCC lattice configuration, where up to six struts intersect at central nodes, each vertex must be adjusted along the direction of the connecting nodal vector so that no facets will overlap. In practice this is completed by first revolving the number of points inputted by the cross-section order around the strut vector, then by calculating the intersection of cylinders the vertices are moved along the direction of the current strut vector so they match the intersecting strut. An example of this can be seen in Figure 2. To calculate the various vector rotations quaternion transformations are used, as they allow for increased computational efficiency over other rotation types and avoids issues such as gimbal lock.

Following the creation of the vertices, they are then defined into facets by selecting two vertices at one end of a strut and one vertex at the opposing end. For the second facet on the strut, one adjoining vertex is selected at the first end, and two vertices are selected at the opposing end, creating a rectangular surface. This process is continued around every strut in the lattice and written to the STL file. After this is complete the STL file is finalised and the program terminates.

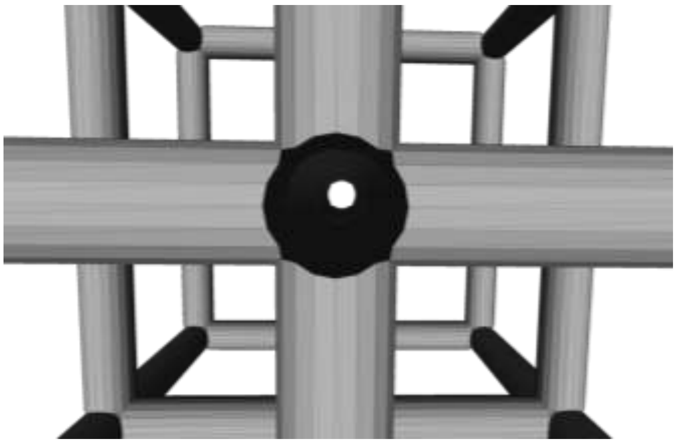


Figure 2 – Calculated intersections for adjoining struts showing circular intersection of intersecting struts

At this point the lattice will require some post processing, as the ends of the lattices will be left open. To close the ends the freely available open-source program 'Meshlab' [9] is used with the simple filter 'close holes'. This filter simply creates additional facets to fill in the open ends in the most efficient manner. As the lattice is generally intended to be added to another structure to act as a stiffener, further operations can be conducted in Meshlab alongside the closing of the ends.

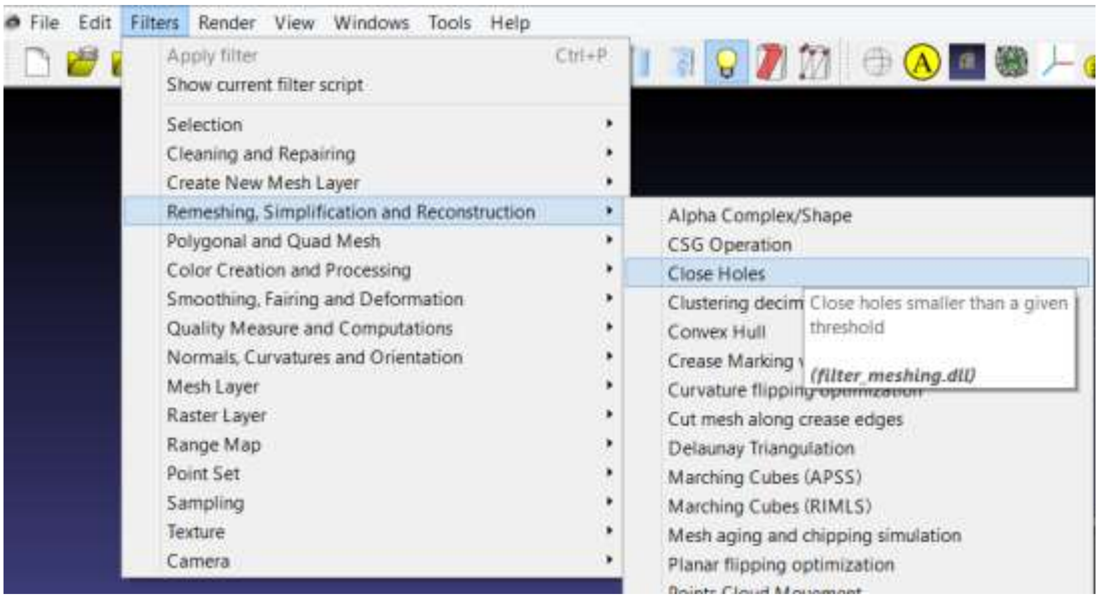
Limitations

Currently, the lattice generator program is limited to uniform lattices of cubic volumes. In future it is intended the program be extended into conformal lattice shapes with variable geometry cell types and region based modifications.

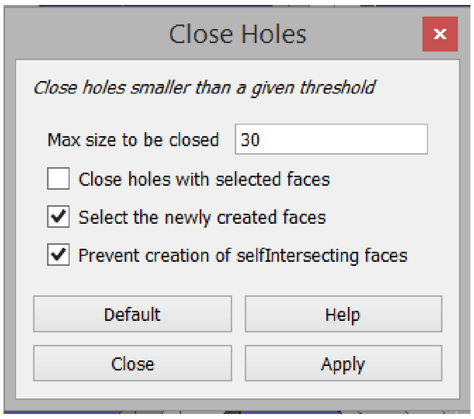
Open Lattice Ends – Non-Manifold

Currently, the program only generates the lattice struts in series, with calculated intersection at each of strut junctions. In this way the facets mesh evenly with adjoining facets and do not overlap. At this time the program does not close the ends of the struts that are on the lattice exterior, leaving open holes in the STL and making it non-manifold.

Closing the holes is to be a feature of future versions, however for now there is a simple fix. Using the open-source program "MeshLab" (available: http://meshlab.sourceforge.net) the holes can easily be filled with the filter "Close holes". The filter is listed under "Filters"→"Remeshing, Simplification and Reconstruction"→"Close Holes".



Once the filter has been selected, the following dialog box will be loaded:


Close Holes

Close holes smaller than a given threshold

Max size to be closed: 30

☐ Close holes with selected faces

☑ Select the newly created faces

☑ Prevent creation of selfIntersecting faces

Default parameters will work fine for most lattices, although the parameter 'Max size' must be reduced when working with finer lattices to not connect across multiple struts. Also, in some cases the option preventing the creation of self-intersecting faces must be deselected to ensure a manifold STL in regions of high complexity.

If there is a problem in applying the filter, simply click the reload button to try again.

Debugging

While there has been an effort to remove bugs from the program, it is likely that there still are a problems hiding somewhere in the code. If you find a bug please leave a message on FileExchange or email: marten@rmitspace.com

Thank you for using the lattice program; I hope you found it useful!