FLatt Pack: a research-focussed lattice design program

I. Maskery[a,*], L.A. Parry[a,b], D. Padrão[a], R.J.M. Hague[a], I.A. Ashcroft[a]

[a] *Centre for Additive Manufacturing, Faculty of Engineering, University of Nottingham, Nottingham NG8 1BB, UK*
[b] *Added Scientific Ltd., Isaac Newton Centre, Nottingham NG7 2RH, UK*

Abstract

Lattice structures are an important aspect of design for additive manufacturing (DfAM). They enable significant component light-weighting and the tailoring of a wide range of physical responses; mechanical, thermal, acoustic, etc. In turn, lattice design relies on fundamental research to uncover useful structure-property relationships, such as the influence of cell geometry and volume fraction. A number of commercial computer-aided-design (CAD) programs exist that offer lattice generation, but these tend to prioritise product design. This paper describes the FLatt Pack program (or Functional Lattice Package), which was created to address the paucity of *research-focussed* lattice design software. It possesses a number of features with this in mind, including; (i) it is free to use for research, (ii) it is standalone software with minimal, and also free, dependencies, and (iii) it undergoes frequent and rapid development based on state of the art lattice information and modelling methods. FLatt Pack includes twenty-three lattice cell types covering a broad range of pore connectivity, structural anisotropy, and surface area; a clear GUI presenting the lattice design stages in a sequential manner; and the option to export designs in appropriate formats for AM and finite element (FE) simulation. The program also features conformal lattice generation in arbitrary shapes, arbitrary volume fraction grading, and resource-efficient computation through an adaptive spatial resolution based on the user's design choices. The most recent version of FLatt Pack is freely available at: www.github.com/ian27ax/FLatt_Pack_dist.

Keywords: lattice design, additive manufacturing, triply periodic minimal surfaces (TPMS), Matlab GUI

1. Introduction

The emergence of additive manufacturing (AM), and the enhanced manufacturing freedoms it represents, has led to a surge of interest in the design of cellular, or lattice, structures. (Note that 'lattice' is typically used to refer to cellular structures in an AM context, so will be used throughout this paper.) The reason for this is clear if we consider the processes traditionally used to produce such structures. Planar honeycombs and sandwich panels can be assembled from polymer, metal or composite sheets [1, 2], or through the corrugation [3, 4] or expansion [5, 6] processes. And metal foams can be made using the gas injection and gas-releasing particle decomposition methods, as well as by metal deposition on pre-existing polymer foams [7]. With respect to the production of lattice structures, AM offers three main advantages over these and other conventional processes:

1. Complete control over the cell geometry, and therefore the resulting physical properties.
2. Complete control over the structure's outer *form*; creating net shape components without subsequent machining or assembly steps.
3. AM components are made according to a computer representation of the design, enabling a range of simulation and optimisation methods prior to manufacture.

*Corresponding author
Email address: ian.maskery@nottingham.ac.uk (I. Maskery)

These provide strong motivation for the incorporation of lattices in AM components, as they mean that computer-aided-design (CAD) can be used to create lattice structures with pre-defined properties (e.g., mass, stiffness), therefore providing materially-efficient solutions for particular applications. Consequently, a number of CAD programs have been developed for lattice structure generation.

Among the commercial programs featuring lattice generation, Autodesk offers a range of latticing modules for their Netfabb software [8], with capabilities ranging from simple lattice infill to the simulation of lattices under quasi-static loads. Materialise 3-matic and the Magics Structures Module [9], and nTopology [10], provide considerable lattice design freedom (in their range of available cell geometries, for example), with emphases on optimisation and AM build preparation. The free Meshmixer software from Autodesk [11] features some limited lattice design capability, whilst providing many of the elementary tools for AM build preparation; STL file repair, orientation optimisation, support generation, etc.

Open-source lattice generation software include the Matlab-based STL Lattice Generator [12] and the Rhino-based Crystallon [13]. Special attention should be given to MSLattice from Al-Ketan and Abu Al-Rub [14] (free but not open-source), which provides good lattice visualistion and has some overlap in functionality with the FLatt Pack software presented here. It too focusses on the generation of surface-based lattices and provides extensive control over their geometry. And, like FLatt Pack, MSLattice facilitates fundamental *lattice structure research* based on precise control of lattices at the cellular level. This is an important distinction from the commercial packages listed above, the main purpose of which is generally to offer lattice creation as part of a product design work flow in a conventional CAD environment. Though on this topic, it must also be noted that nToplogy's design software deviates from traditional CAD concepts in some key areas, particularly in its use of mathematically defined fields for the control of geometrical features [10]. NTopology also offers its software with a free educational license for students and educators, so is not solely a 'commercial' package.

Concerning lattice structure research, one of the main objectives in this area is the determination of structure-property relationships that can be used to control the functionality of a structure or component. The physical properties of lattices, typically mechanical but extending to other domains such as fluid flow, thermal, electrical, chemical and biological, are dictated chiefly by their volume fraction (the volume of solid material as a fraction of the total volume of the structure), which is reflected in the well known scaling laws of Gibson and Ashby [15]. The volume fraction has therefore become the primary *design variable* for AM lattices. But AM has enabled, and indeed necessitated, extensive investigations into the influence of other design variables; in particular, the lattice cell geometry. Among the useful findings from this broad field of research are that cell geometry determines the mechanical anisotropy of lattice structures [16–19], their deformation mechanisms and failure modes under quasi-static loading [20–22], their vibration resonance and transmissibility [23–25], their fluid permeability [26, 27] and their suitability as biological cell scaffolds for regenerative medicine [28, 29].

These represent just some of the topics in which lattice structures have provided new and useful results, and they highlight the need for research-focussed lattice design software. To this end, FLatt Pack (short for The Functional Lattice Package) provides complete control over the main lattice design variables; cell type, cell size and volume fraction, with a graphical user interface (GUI) that presents those variables in an clear sequential manner. FLatt Pack also offers:

- Lattice generation within any defined geometrical surface based on user-supplied CAD input.
- 2D and 3D volume fraction grading, again based on user input.
- Access to a new family of surface-based 2D honeycombs and approximations to strut-based lattices.
- The option to directly export the lattice design in the form of an STL file or finite element (FE) mesh.

The latter is extremely useful for lattice research, as it enables numerical simulation of a wide array of physical properties using an FE solver, therefore facilitating both comparison with experiment and the exploration of physical performance beyond that which can be easily measured.

A further distinguishing feature of FLatt Pack is that the programming implementation aims to minimise its memory consumption and run time through the use of an adaptive spatial resolution based on the user's

Table 1: Summary of past results from FLatt Pack.

	Publication year	Summary
Ref [30]	2017	Examined the role of cell size and post-manufacture heat treatment on the failure modes and energy absorption of gyroid lattice structures.
Ref [20]	2018	Examined three surface-based lattice types (primitive, gyroid and diamond) with a combination of compressive mechanical testing and FE simulation.
Ref [31]	2018	FE simulation of six surface-based lattice structures to determine Gibson-Ashby scaling laws along high symmetry loading directions. Also examined hybrid structures comprising two cell types and developed a density correction procedure for such structures.
Ref [23]	2019	Examined phonon dispersion and resonant frequencies in surface-based lattice structures, including the influence of cell size and volume fraction.
Ref [32]	2020	Examined the thermal conductivity of the gyroid lattice using a specially developed thermal testing rig.
Ref [16]	2020	Examined the deformation mode and anisotropy of a new honeycomb structure based on the gyroid surface equation.
Ref [33]	2021	Examined a range of surface-based lattice structures to determine their suitability for bone growth scaffolds, including the influence of pore size on the rate biological cell growth.

lattice design choices (as described in Appendix A.2). This means FLatt Pack, in its default mode of operation, removes the burden of resolution selection from the user, who in general will not be able to make an informed choice on this matter because it requires specific knowledge of the lattice generation method. This, combined with the clear GUI, makes FLatt Pack both a user-friendly and computer resource-efficient lattice design tool.

The main purpose of FLatt Pack is to be a free design tool to enable advanced lattice research across a diverse range of scientific fields. The program installer is available via a public Github repository (www.github.com/ian27ax/FLatt_Pack_dist), which is frequently updated with bug fixes and new features based on state of the art lattice information and modelling methods. Should FLatt Pack users wish to discuss the program or report bugs, they can contact the corresponding author by e-mail at: ian.maskery@nottingham.ac.uk.

1.1. FLatt Pack research summary

Whilst this paper focuses on the development and capabilities of FLatt Pack, the program has already contributed to several lattice studies in the literature. Table 1 summarises these previous works.

The remainder of this paper is organised as follows. In section 2 the main theoretical principles of surface-based lattice design are introduced, while section 3 provides several illustrated examples of FLatt Pack's use. Concluding remarks are given in section 5. Two appendices are attached, which together provide an extensive account of FLatt Pack's design and operation. In Appendix A close attention is paid to the implementation and coding of the FLatt Pack program, while Appendix B describes the organisation and operation of the its GUI.

2. Theory: Surface-based lattice design

Surface-based lattice structures with arbitrary numbers of cells and volume fractions can be constructed by determining isosurfaces from their surface equations. In the field of differential geometry, minimal surfaces are described using the Enneper-Weierstrass parameterisation in the complex plane, but they may also be approximated using Fourier series expansions of trigonometric terms [34–36]. For the purpose of designing, simulating and fabricating lattice structures based on surfaces, these approximations provide reasonable descriptions of their shape (as demonstrated by Gandy *et al* [35]) and, being real functions expressed in a Cartesian frame, are simpler to express and implement programmatically.

A total of twenty-three lattice cell types are available in FLatt Pack. These include those based on the primitive and diamond triply periodic minimal surfaces (TPMSs) discovered by Schwarz [37], the gyroid, I-WP and O,C-TO TPMSs of Schoen [38], and the neovius, lidinoid and split-P TPMSs of Hoffman *et al* [39]. Also included are approximations to simple cubic [40] and body-centred cubic lattice structures, as well as several 2D honeycombs based on reduced-periodicity TPMS equations.

Of course, there are countless other, non-surface-based cell types available for lattice design, with some of the most commonly studied being the octet-truss [41, 42] and the Kelvin cell [43]. The addition of these cell types will be considered for future updates to FLatt Pack, in order to broaden its appeal and relevance across the various fields of lattice research. The interested reader is directed to the recent work of Benedetti *et al* [44], which contains a good overview of the design and mechanical properties of a broad range of lattice types.

To describe the surface-based lattice structures of FLatt Pack, some basic nomenclature must be introduced. The lattice design space is a simple Cartesian frame, with axes x, y and z. k_i are the lattice function periodicities, defined by

$$k_i = 2\pi n_i, \tag{1}$$

where $i = x,\ y,\ z$, and n_i are the numbers of cell repetitions in those directions.

Sine and cosine functions are defined with the shorthand notation

$$S_i = \sin\left(k_i \frac{i}{L_i}\right), \tag{2a}$$

$$S_{2i} = \sin\left(2k_i \frac{i}{L_i}\right), \tag{2b}$$

and

$$C_i = \cos\left(k_i \frac{i}{L_i}\right), \tag{3a}$$

$$C_{2i} = \cos\left(2k_i \frac{i}{L_i}\right), \tag{3b}$$

again with $i = x,\ y,\ z$, and L_i being the absolute sizes of the lattice structure in those directions.

There are several sources for series approximations to TPMSs [35, 36, 39, 45], though they differ somewhat in the number of Fourier expansion terms they include. The inclusion of more terms brings the generated surface closer to the ideal TPMS with zero mean curvature. FLatt Pack uses the level surface approximations

given by Wohlgemuth *et al* [36] and Hoffman *et al* [39]. These are:

$$U_{\mathrm{P}} = \Big(C_x + C_y + C_z\Big)^m - t^m, \tag{4a}$$

$$U_{\mathrm{G}} = \Big(S_xC_y + S_yC_z + S_zC_x\Big)^m - t^m, \tag{4b}$$

$$U_{\mathrm{D}} = \Big(C_xC_yC_z + S_xS_yC_z + S_xC_yS_z + C_xS_yS_z\Big)^m - t^m, \tag{4c}$$

$$U_{\mathrm{I}} = \Big(C_xC_y + C_yC_z + C_zC_x\Big)^m - t^m, \tag{4d}$$

$$U_{\mathrm{O}} = \Big(4\big(C_xC_y + C_yC_z + C_zC_x\big) - 3\big(C_x + C_y + C_z\big)\Big)^m - t^m, \tag{4e}$$

$$U_{\mathrm{N}} = \Big(3\big(C_x + C_y + C_z\big) + 4C_xC_yC_z\Big)^m - t^m, \tag{4f}$$

$$U_{\mathrm{L}} = \Big(\big(S_{2x}C_yS_z + S_{2y}C_zS_x + S_{2z}C_xS_y\big) - \big(C_{2x}C_{2y} + C_{2y}C_{2z} + C_{2z}C_{2x}\big)\Big)^m - t^m, \tag{4g}$$

$$U_{\mathrm{SP}} = \Big(1.1\big(S_{2x}C_yS_z + S_{2y}C_zS_x + S_{2z}C_xS_y\big) - 0.2\big(C_{2x}C_{2y} + C_{2y}C_{2z} + C_{2z}C_{2x}\big) - 0.4\big(C_{2x} + C_{2y} + C_{2z}\big)\Big)^m - t^m, \tag{4h}$$

where $S_{x,y,z}$, etc., are defined in equations 2 and 3, and t is an arbitrary parameter used to control the position of the $U = 0$ isosurface. The subscripts P, G, D, etc., stand for; primitive (P), gyroid (G), diamond (D), I-WP (I), O,C-TO (O), neovius (N), lidinoid (L) and split-P (SP).

In equation(s) 4 the exponent m can take the value of 1, identifying the network phase lattice, or 2, the matrix phase (in the literature these are also known as 'skeleton' and 'sheet', respectively). In the network lattice, space is divided by the $U = 0$ boundary into two continuous regions; one solid, one void. The matrix lattice contains three separate regions; two of equal size and equivalent geometry divided by a third in the form of a continuous solid wall. Examples of network and matrix phase cells based on the same gyroid TPMS (equation 4b) are shown in figure 1.

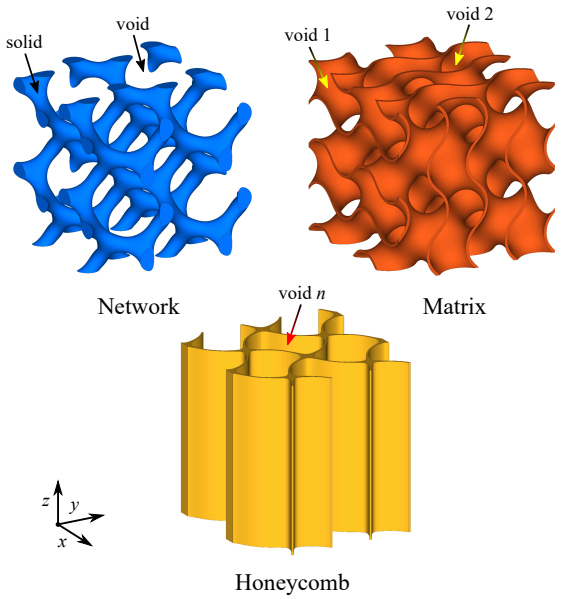


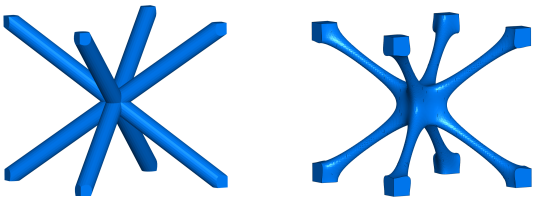
Figure 1: Network, matrix and honeycomb varieties of the gyroid lattice. Reproduced from [16].

Surface-based approximations to simple cubic (SC) [40] and body-centred cubic (BCC) lattices are given by:

$$U_{\mathrm{SC}} = \Big(2(C_x + C_y + C_z) + (C_xC_y + C_yC_z + C_zC_x)\Big) - t, \tag{5a}$$

$$U_{\mathrm{BCC}} = \Big((C_{2x} + C_{2y} + C_{2z}) - 2(C_xC_y + C_yC_z + C_zC_x)\Big) - t, \tag{5b}$$

which produce structures similar to network phase TPMS lattices, in that they comprise only one solid and one void domain, but they differ in that they resemble strut-based lattices. For example, the surface approximation of the BCC cell is shown in figure 2 beside a strut-based version with equivalent volume fraction. The principle differences in geometry are the slight tapering of the structural members and the smooth, filleted intersection for the surface-based cell. Smooth fillets in additively manufactured lattice structures have previously been shown to improve the structure's strength and energy absorption efficiency compared to standard connected struts [46]. Savio *et al* similarly discovered that fillets improve the fatigue life of lattice structures, by reducing their stress concentration factor (K_t), and showed, with FE modelling, that K_t is reduced by increasing the fillet radius [47]. This suggests that user-defined fillet radius may be useful for further lattice investigations, so this feature will be considered for inclusion in future updates to FLatt Pack.



(a) (b)

Figure 2: The body-centred cubic (BCC) cell. In (a) is shown the traditional strut-based cell, while the surface-based approximation available in FLatt Pack is shown in (b).

Lastly, several 2D honeycombs (honeycomb primitive (HP), honeycomb gyroid (HG), honeycomb diamond (HD), honeycomb I-WP (HI) and honeycomb lidinoid (HL)) are given by:

$$U_{\mathrm{HP}} = \Big(C_x + C_y\Big)^2 - t^2, \tag{6a}$$

$$U_{\mathrm{HG}} = \Big(S_x C_y + S_y + C_x\Big)^2 - t^2, \tag{6b}$$

$$U_{\mathrm{HD}} = \Big(C_x C_y + S_x S_y + S_x C_y + C_x S_y\Big)^2 - t^2, \tag{6c}$$

$$U_{\mathrm{HI}} = \Big(C_x C_y + C_y + C_x\Big)^2 - t^2, \tag{6d}$$

$$\begin{aligned} U_{\mathrm{HL}} = \Big(&1.1\big(S_{2x} C_y + S_{2y} S_x + C_x S_y\big) \\ &- \big(C_{2x} C_{2y} + C_{2y} + C_{2x}\big)\Big)^2 - t^2, \end{aligned} \tag{6e}$$

which are based on the original TPMS approximations of equation 4, but with the z-periodic terms omitted. Structures based on these equations generally divide space into more than two volumes; one continuous solid wall and a number of unconnected voids. This can be seen in the honeycomb gyroid in figure 1, which is based on equation 6b and was previously investigated by Maskery and Ashcroft [16].

For each of equations 4, 5 and 6, $U = 0$ is treated as a boundary separating space into solid and void volumes, with the solid volume then representing the lattice structure for the purposes of analysis (e.g., with FE modelling) or manufacture. This is illustrated in figure 3, and is the main principle of surface-based lattice design.

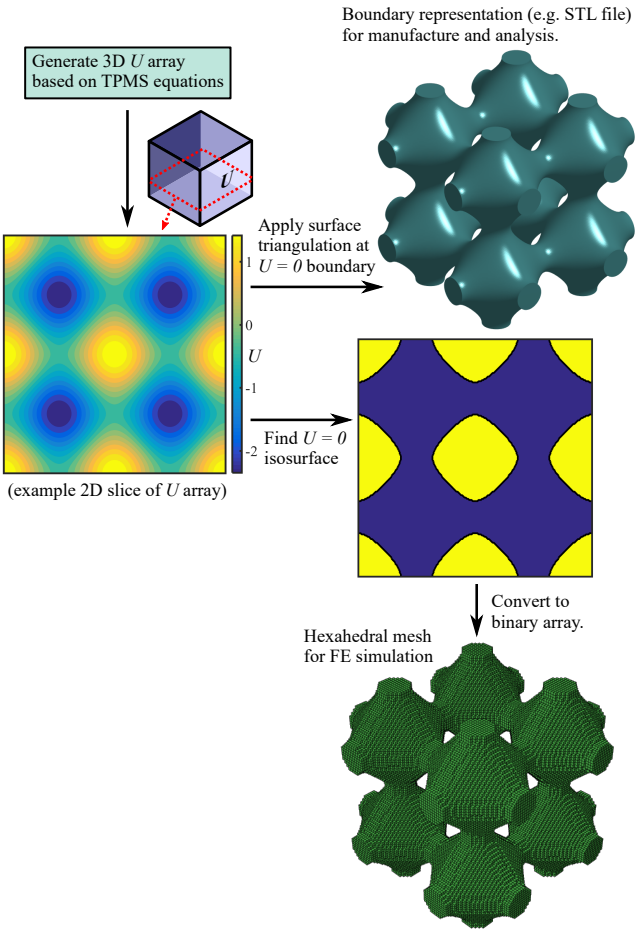


Figure 3: The main principle of surface-based lattice design; the $U = 0$ isosurface separates the design space into solid and void domains.

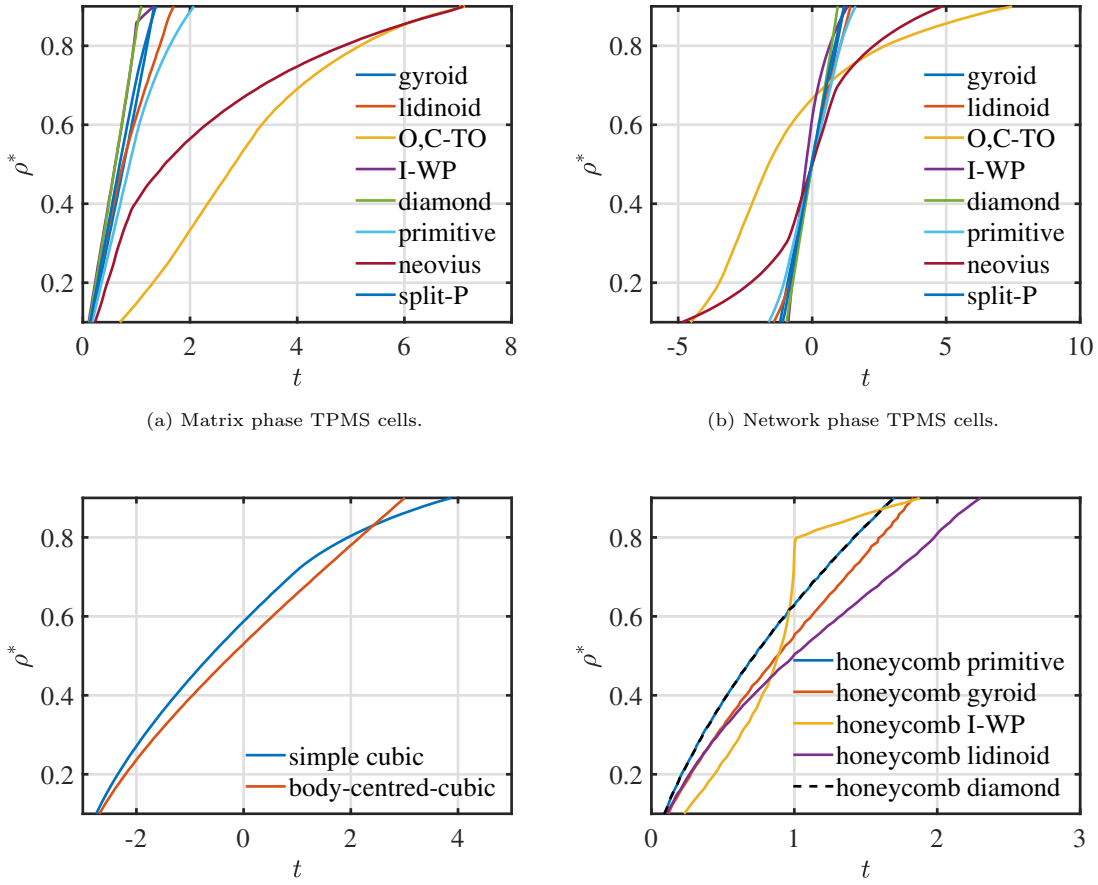
As a final point before the operation of FLatt Pack is described, we should note that it is desirable, from the perspective of the lattice designer, to be able to precisely control the volume of the solid phase of a lattice, since this ultimately determines the mass of the structure, as well as many of its physical properties. This is achieved by modification of the t parameter of equations 4, 5 and 6, and thus it is necessary to know the relationships between t and the solid volume for each cell type.

The volume fraction, ρ^*, of a lattice is the fraction of the design space composed of solid material; it takes values between 0 and 1, where 1 indicates an entirely solid volume and 0 entirely void. $\rho^* - t$ correlations for each of the lattices available in FLatt Pack are shown in figure 4. (We have provided a small number of them in [31] and [16], previously.) The curves of figure 4 were obtained by generating high-resolution models ($150 \times 150 \times 150$ voxels) of each cell over a range of one hundred t values, then summing the solid voxels to determine the total solid volume. The subsequent correlations were obtained through modified Akima interpolation between the $\rho^* - t$ pairs and are included in FLatt Pack, such that when the user designates a value of ρ^*, or a distribution of ρ^* representing a functionally graded lattice, the appropriate t values are looked up in a table and used to generate U. Thus, the FLatt Pack user is able to directly control the lattice volume fraction without explicit knowledge of equations 4, 5 and 6 or the associated $\rho^* - t$ correlations.

Inspection of figure 4 reveals that some $\rho^* - t$ curves exhibit discontinuities; for example, the matrix neovious structure at $\rho^* \sim 0.4$, the simple cubic structure at $\rho^* \sim 0.7$, and the honeycomb I-WP structure at $\rho^* \sim 0.8$. These relate to fundamental changes in the cellular geometry at those volume fractions, with the general cause being the closure of a particular pore. It may be interesting, in future work, to examine the

different deformation mechanisms for lattices that possess such structural transformations with increasing volume fraction.

A further consequence of volume fraction control is the implicit control of the geometry of solid and void regions of the lattice. Thus, if the lattice application requires struts or walls, or internal pores or channels, of specific dimensions, to accommodate fluid flow for example, FLatt Pack can be used to meet this demand too.



(a) Matrix phase TPMS cells.

(b) Network phase TPMS cells.

(c) Surface-based strut type cells.

(d) Surface-based honeycomb cells.

Figure 4: $\rho^* - t$ correlations for the surface-based cells available in FLatt Pack.

3. Results: Lattice generation examples

This section provides several examples of the use of FLatt Pack for lattice research. The relevant input and output files are available as research data associated with this article, or can be requested directly from the corresponding author.

3.1. Lattice specimen for mechanical testing

Here we look at the use of FLatt Pack to investigate the response of a lattice structure under quasi-static compressive loading. For this example, a network phase gyroid lattice is examined, comprising $5 \times 5 \times 5$ cells with dimensions of $50 \times 50 \times 50$ mm and volume fraction of 0.15. (Arrangements of $5 \times 5 \times 5$ cells have

previously been shown to provide stiffness results reflective of homogeneous lattices, unlike structures with fewer cells [31].)

The investigation comprises two parts; (i) AM fabrication and mechanical testing, and (ii) numerical modelling with FE analysis, as illustrated in figure 5. Both routes begin with the creation of appropriate files in FLatt Pack. For manufacture an STL file is required (figure 5(a)), while for FE analysis an Abaqus INP file is required (figure 5(d)).

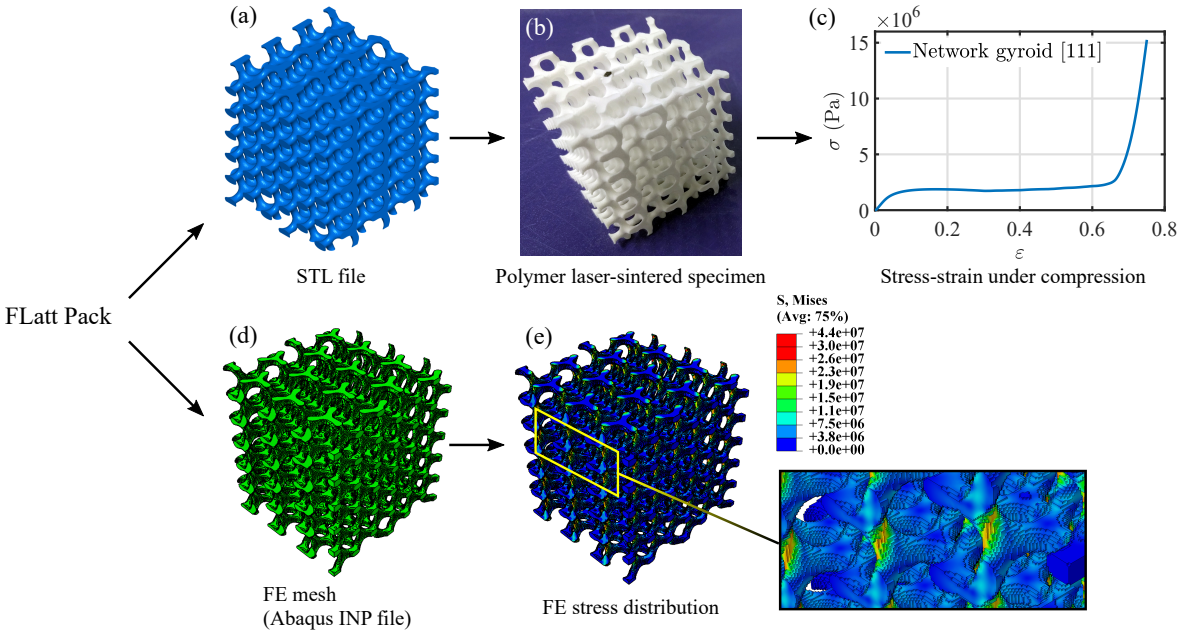


Figure 5: The use of FLatt Pack to investigate the response of a lattice structure under quasi-static loading.

The user proceeds through the 'Geometry' and 'Dimensions and cell repetitions' windows of FLatt Pack, then arrives at the 'Cell type' window. The 'Modulus estimate' subwindow is consulted (see figure B.22), which indicates that aligning the cubic [111] direction of the lattice with the applied load will provide the stiffest solution. This is achieved with rotations of 35.26° and 45° around the x and y axes, respectively. A uniform volume fraction of 0.15 is selected in the 'Volume fraction' window, and the user opts not to include an exterior skin, so that the observed deformation, stiffness, etc., may be ascribed to the lattice type *in general*, and may therefore be useful in making future design choices.

At the 'Output files' window, the user chooses to export STL and Abaqus INP files. A triangle mesh reduction setting of 50% is chosen, leading to an output STL file of 28 MB containing $\sim 600,000$ surface triangles. Following its generation in FLatt Pack, the surface mesh can be reduced further using dedicated AM build preparation software (e.g., Materialise Magics [9]), which is generally required anyway, to perform minor STL repairs, as well as the slicing and hatching operations to create an AM machine input file.

The specimen is fabricated by polymer laser sintering of PA2200 (figure 5(b)) before being subject to quasi-static compression using a universal testing machine with a load transducer. The resulting stress-strain curve (figure 5(c)) exhibits the typical features of low volume fraction cellular structures under compression; an initial elastic region at low strain, a long platic plateau, and a densification region at high strain, in which the stress increases significantly due to the collapse and subsequent contact of the cell struts. The elastic modulus can be extracted from the low strain linear region; in this case it is $E = 39.7 \pm 0.1$ MPa.

On the matter of design and fabrication accuracy, the volume fraction of the FLatt Pack output STL file was checked and was found to correspond to the value of 0.15, as designed, while weighing and measurement of three manufactured specimens yielded a volume fraction of 0.162 ± 0.003. This is $\sim 8\%$ larger than the target of 0.15, and is most likely a consequence of partially-sintered powder adhered to the internal lattice surfaces. Adhesion of partially-sintered powder on laser sintered parts is common and its influence

on the mass of lattice structures, like here, can be significant because of their large surface area. This, and our previous experience with FLatt Pack and its operation with a range of AM processes, supports what is already widely understood in AM; that the accuracy of the production process is a function of several parameters, including the design generation, the AM file preparation (slicing, hatching, etc.), the material and the AM method. In the development of FLatt Pack, we have endeavored to ensure the first of these is not a contributor to production inaccuracy.

Pursuing the numerical modelling route, at the 'Abaqus FEA job creation' window, the user selects the 'Displacement load' option and provides the elastic modulus of the material used for manufacturing. In this example, the modulus was given as 1.59 GPa, based on our previous study of laser sintered PA2200 [20]. The displacement load is set to 1% of the height of the specimen (i.e., 0.5 mm) and the load direction is set as z, mirroring the load applied in physical testing. The output INP file is 92 MB in size, containing ~ 1.1 million nodes and $\sim 830,000$ hexahedral elements. It is generated by FLatt Pack in around 25 seconds. The INP file is run using the Abaqus/Standard solver (Dassault Systemes - Vélizy-Villacoublay, France), providing an output database (ODB) file with node and element information for the FE solution. As an example of such information, the von Mises stress distribution for the network gyroid lattice is shown in figure 5(e), in which it can be seen that the greatest stress is found in those struts which are near-vertically aligned or, more precisely, those which are closely aligned with the direction of the applied load. The effective elastic modulus of the lattice obtained from the FE analysis is 45.6 MPa, which is an overestimate compared to the experimental result. This highlights an important and under-investigated aspect of FE modelling for AM lattice structures; the choice of material properties for the simulation. In the case above, we chose a linear elastic material model with 1.59 GPa for the modulus, based on our own uniaxial compression testing of laser sintered PA2200, and a Poisson's ratio of 0.4 [48]. But we have not yet determined if this approach is appropriate, given that; (i) lattice struts and walls are typically mm-scale, rather than the cm-scale of standard tensile and compressive test geometries, so the possibility of a size effect cannot be ignored as the feature sizes approach the manufacturing resolution, and (ii) in all but the simplest cases, lattice struts do not undergo purely axial deformation, but rather experience a complex load case including bending and shear forces. The FE meshes provided by FLatt Pack may be easily modified to include the user's material model, to more accurately replicate the behaviour of materials made via a range of AM processes.

3.2. Lattice infill for ease of manufacture

In this example, FLatt Pack is used to generate lattice infill for a solid part, providing a design that can be manufactured quicker and with less material than the original. This is the ubiquitous application of lattice generation in AM, since it enables materially-efficient manufacture and light weight products. The demonstration case here is a piston rod from the popular digital design sharing website Thingiverse (www.thingiverse.com), as shown in figure 6(a).

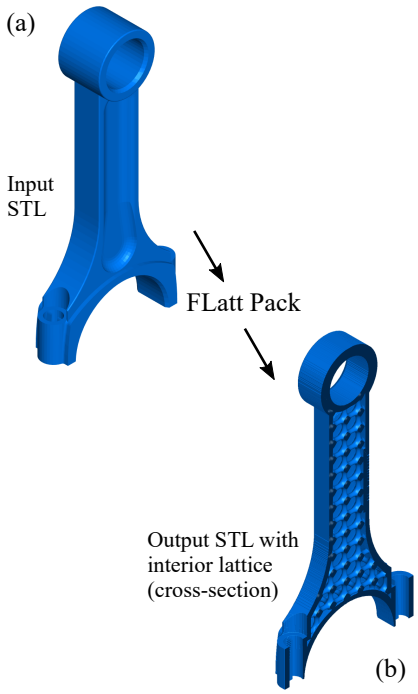


Figure 6: The use of FLatt Pack to provide a supporting lattice infill.

The original model STL was loaded into FLatt Pack using the 'Custom' option in the 'Geometry' window. The 'Dimensions and cell repetitions' window then provides the dimensions of the part, which in this case are approximately $33 \times 11 \times 68$ mm. The user specifies cell repetitions of $9.3 \times 3 \times 18.9$, maintaining the aspect ratios of the part dimensions, and therefore ensuring cubic lattice cells (see figure B.19). The user selects the 'Very fine' input STL resolution option, to preserve as much as possible the quality of the part's outer form. In the following windows the user selects the simple cubic cell type, a uniform volume fraction of 0.2, and a solid skin of thickness 1 mm. At the 'Triangle mesh reduction' window the user opts to export 100% of the surface triangles, providing an output STL file 184 MB in size, containing $\sim$ 3.8 million triangles. The file is then post-processed with AM build preparation software to make minor STL repairs, and for slicing and hatching. The AM build preparation software may also be used to introduce a small hole in the part skin for the removal of excess feedstock if necessitated by the AM process. The final design, with the interior lattice exposed in cross-section, is shown in figure 6(b).

3.3. Lattice generation based on an input volume fraction distribution

This last example demonstrates the use of FLatt Pack to generate a lattice based on an arbitrary volume fraction distribution in an input file. The part in question is a tensile test specimen (or 'dog bone'), which the user wishes to create with solid ends to facilitate better gripping during the tensile test.

The user begins by selecting the 'Custom' option in the 'Geometry' window, then importing the file 'ASTM_type_1_tensile_bar.stl', which is included with the FLatt Pack installation (figure 7(a)). The user selects the 'Fine' input STL resolution option and specifies cell repetitions of $25.8 \times 2.95 \times 0.5$, maintaining the aspect ratio of the part dimensions, as in the previous example.

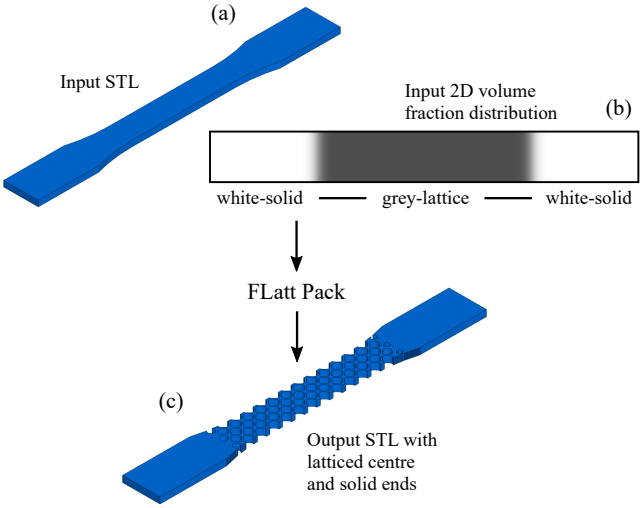


Figure 7: Lattice generation based on an input volume fraction distribution.

In the following windows the user selects the honeycomb primitive cell type and uses the 'Import vol. frac. map' button to import the file 'Density_map_for_ASTM_tensile_bars.bmp' (figure 7(b)). As described in Appendix B.6, this bitmap image contains volume fraction values represented as colours, where black (r g b = 0 0 0) corresponds to void in the lattice design space, and white (r g b = 255 255 255) corresponds to solid. The 'Density_map_for_ASTM_tensile_bars.bmp' image has a central grey region (r g b = 77 77 77) corresponding to $\rho^* = 77/255 = 0.3$, which transitions to the white, or solid, end pieces. The bitmap was created using the free and open-source vector graphics editor, Inkscape (www.inkscape.org), though other image editors could just as easily be used. At the 'Triangle mesh reduction' window the user opts to export 100% of the surface triangles, providing an output STL file 55 MB in size, containing ~ 1.1 million triangles. The final design is shown in figure 7(c).

4. Software limitations and advantages

Here, we summarise what we see to be the main limitations and advantages of the FLatt Pack program in its current form. The program is under ongoing development, with bug reporting and suggestions from users both encouraged and highly valued. FLatt Pack's creators will endeavor to tackle its main limitations in future updates.

Limitations:

1. The current visualistion of the lattice design and the effect of various parameter choices (e.g., volume fraction, cell orientation), is not ideal, and is below the standard of more established and intuitive CAD environments. This is chiefly a consequence of the sequential lattice design process, though is also influenced by the 3D rendering capabilities of Matlab and the speed with which the large numerical arrays underpinning FLatt Pack can be created and updated.
2. Volume fraction grading requires careful design of a 2D or 3D array, which is not intuitive and may be beyond some users, while the current inbuilt grading functions are limited to a small selection of mathematical operations resolved in a Cartesian frame.
3. FE output files are represented only as non-conformal hexahedral meshes, which are unsuitable for contact-analysis problems. Tetrahedral meshes would be desirable, as would meshes of combined solid and void domains for fluid dynamics problems, especially given the motivation to use lattice structures in fluid-based devices like heat exchangers and filters.

4. The method to apply a lattice to an arbitrary input geometry requires the voxelisation of a user-supplied STL file. The STL format is currently the only accepted input file type, which is limiting given the number of more efficient CAD formats available. Additionally, the voxelisation of the input STL file within FLatt Pack is, at present, one of its most resource-intensive operations.

Advantages:

1. FLatt Pack provides control over the main lattice design variables in a clear GUI. It has inbuilt correlations between the volume fractions of many lattice types and their corresponding surface equations, as well as an adaptive spatial resolution to minimise computational demands whilst maintaining geometrical accuracy. These features significantly reduce the challenge and time burden of lattice design. The program may therefore facilitate future lattice research in a diverse range of fields from users who do not possess the knowledge or experience to create manufacturable lattices on their own.
2. Output formats include STL files and FE meshes. Thus, FLatt Pack targets the main file types used for lattice research; those which enable manufacture and computational modelling.
3. FLatt Pack undergoes frequent and rapid development based on state of the art lattice information and modelling methods (e.g., new cell types, quicker lattice generation from the surface equations, more accurate geometry representation in FE meshes), with the newest version being made freely available for researchers via a its Github repository.

5. Conclusions

Here we have presented the FLatt Pack program, or The Functional Lattice Package. It facilitates fundamental lattice research by providing complete control over the main cellular design variables, with the benefits of a clear GUI and the option to export lattice designs that are readily manufacturable by AM processes and useful for FE simulation. The program also features lattice generation in arbitrary shapes based on user-supplied CAD input, and arbitrary 2D and 3D volume fraction grading, again based on user input. The main principle of the program is the generation of lattices from periodic surface equations. The implementation of this in the Matlab programming language is covered here, as is the the organisation and operation of the FLatt Pack GUI.

FLatt Pack includes an automatic and adaptive spatial resolution based on the user's lattice design choices, which, to our knowledge, is not present in other research-focussed lattice software (e.g., MSLattice [14]). This reduces the size of numerical arrays stored in the host PC's memory during operation, making FLatt Pack a resource-efficient research tool. Compared to MSLattice, FLatt Pack also provides a broader choice of inbuilt cell designs, including novel 2D honeycombs and approximations to strut-based cells, thus providing more flexibility for the tailoring of mechanical, thermal and fluid-flow properties to specific applications. Furthermore, FLatt pack offers lattice generation in user-supplied CAD input geometries, while MSLattice does not. But FLatt Pack's reliance on discrete numerical arrays also leads to one of its main shortcomings; the presence of staircase artifacts at the surface of some output STL files, particularly those based on arbitrary input geometries. Methods to eliminate this issue will be pursued for future updates to the FLatt Pack program. Further planned updates include; (i) the addition of more application-specific lattice information such as pore diameter and tortuosity, to complement the existing surface area and elastic modulus estimates, (ii) the option to export tetrahedral FE meshes in addition to the hexahedral meshes already offered, and (iii) a new mode of operation which bypasses the GUI, allowing the user to run FLatt Pack with a text or data input (e.g., a CSV file) to generate lattices automatically. This will enable quicker 'batch' generation of lattice structures, which can be used in physical or numerical experiments to investigate the impact of lattice design variables.

The FLatt Pack installer is available freely via a public Github repository (www.github.com/ian27ax/FLatt_Pack_dist), which is updated periodically with new features and bug fixes. It is hoped that it will be taken up in diverse fields as a useful tool for lattice research.

Acknowledgments

This work was supported by the University of Nottingham (via Nottingham Research Fellowship funding).

References

[1] F. Côté, V. S. Deshpande, N. A. Fleck, and A. G. Evans, "The compressive and shear responses of corrugated and diamond lattice materials," *International Journal of Solids and Structures*, vol. 43, no. 20, pp. 6220–6242, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020768305004919

[2] S. Park, B. P. Russell, V. S. Deshpande, and N. A. Fleck, "Dynamic compressive response of composite square honeycombs," *Composites Part A: Applied Science and Manufacturing*, vol. 43, no. 3, pp. 527–536, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1359835X11003939

[3] M. R. M. Rejab and W. J. Cantwell, "The mechanical behaviour of corrugated-core sandwich panels," *Composites Part B: Engineering*, vol. 47, pp. 267–277, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1359836812007305

[4] C. Kılıçaslan, M. Güden, I. Kutlay Odacı, and A. Taşdemirci, "Experimental and numerical studies on the quasi-static and dynamic crushing responses of multi-layer trapezoidal aluminum corrugated sandwiches," *Thin-Walled Structures*, vol. 78, pp. 70–78, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263823114000275

[5] W.-Y. Jang and S. Kyriakides, "On the buckling and crushing of expanded honeycomb," *International Journal of Mechanical Sciences*, vol. 91, pp. 81–90, 2015, mechanics of Solids & Structures. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020740314000484

[6] T. N. Bitzer, *Honeycomb technology: materials, design, manufacturing, applications and testing*. Springer Science & Business Media, 1997.

[7] M. F. Ashby, A. G. Evans, N. A. Fleck, L. J. Gibson, L. W. Hutchinson, and H. G. Wadley, *Metal Foam: A Design Guide*. Butterworth-Heinemann, 2000.

[8] (2020) Netfabb lattice modules. [Online]. Available: https://www.autodesk.co.uk/

[9] (2020) Materialise 3-matic. [Online]. Available: https://www.materialise.com/en/software/3-matic/modules/lattice-module

[10] (2020) nTopology. [Online]. Available: https://ntopology.com/

[11] (2020) Meshmixer. [Online]. Available: https://www.meshmixer.com/

[12] Marten. (2021) STL Lattice Generator. [Online]. Available: https://uk.mathworks.com/matlabcentral/fileexchange/48373-stl-lattice-generator

[13] A. Porterfield. (2020) Crystallon. [Online]. Available: http://fequalsf.blogspot.com/p/crystallon.html

[14] O. Al-Ketan and R. K. Abu Al-Rub, "MSLattice: A free software for generating uniform and graded lattices based on triply periodic minimal surfaces," *Material Design & Processing Communications*, vol. n/a, no. n/a, p. e205, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mdp2.205

[15] L. J. Gibson and M. J. Ashby, *Cellular Solids: Structure and properties*. Cambridge University Press, 1997.

[16] I. Maskery and I. A. Ashcroft, "The deformation and elastic anisotropy of a new gyroid-based honeycomb made by laser sintering," *Additive Manufacturing*, vol. 36, p. 101548, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214860420309209

[17] Y. Lu, W. Zhao, Z. Cui, H. Zhu, and C. Wu, "The anisotropic elastic behavior of the widely-used triply-periodic minimal surface based scaffolds," *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 99, pp. 56–65, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1751616119301456

[18] C. Soyarslan, V. Blümer, and S. Bargmann, "Tunable auxeticity and elastomechanical symmetry in a class of very low density core-shell cubic crystals," *Acta Materialia*, vol. 177, pp. 280–292, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1359645419304537

[19] S. Xu, J. Shen, S. Zhou, X. Huang, and Y.-. M. Xie, "Design of lattice structures with controlled anisotropy," *Materials and Design*, vol. 93, pp. 443–447, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0264127516300120

[20] I. Maskery, L. Sturm, A. O. Aremu, A. Panesar, C. B. Williams, C. J. Tuck, R. D. Wildman, I. A. Ashcroft, and R. J. M. Hague, "Insights into the mechanical properties of several triply periodic minimal surface lattice structures made by polymer additive manufacturing," *Polymer*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0032386117311175

[21] M. Afshar, A. P. Anaraki, H. Montazerian, and J. Kadkhodapour, "Additive manufacturing and mechanical characterization of graded porosity scaffolds designed based on triply periodic minimal surface architectures," *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 62, pp. 481–494, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1751616116301552

[22] N. Novak, O. Al-Ketan, L. Krstulović-Opara, R. Rowshan, R. K. Abu Al-Rub, M. Vesenjak, and Z. Ren, "Quasi-static and dynamic compressive behaviour of sheet TPMS cellular structures," *Composite Structures*, vol. 266, p. 113801, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263822321002622

[23] W. Elmadih, W. P. Syam, I. Maskery, D. Chronopoulos, and R. Leach, "Mechanical vibration bandgaps in surface-based lattices," *Additive Manufacturing*, vol. 25, pp. 421–429, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214860418302781

[24] W. Jiang, M. Yin, Q. Liao, L. Xie, and G. Yin, "Three-dimensional single-phase elastic metamaterial for low-frequency and broadband vibration mitigation," *International Journal of Mechanical Sciences*, vol. 190, p. 106023, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020740320324280

[25] O. McGee, H. Jiang, F. Qian, Z. Jia, L. Wang, H. Meng, D. Chronopoulos, Y. Chen, and L. Zuo, "3D printed architected hollow sphere foams with low-frequency phononic band gaps," *Additive Manufacturing*, vol. 30, p. 100842, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221486041931098X

[26] Y. Lu, L. Cheng, Z. Yang, J. Li, and H. Zhu, "Relationship between the morphological, mechanical and permeability properties of porous bone scaffolds and the underlying microstructure," *PLOS ONE*, vol. 15, no. 9, pp. 1–19, 09 2020. [Online]. Available: https://doi.org/10.1371/journal.pone.0238471

[27] T. Santos, J.and Pires, B. P. Gouveia, A. P. G. Castro, and P. R. Fernandes, "On the permeability of TPMS scaffolds," *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 110, p. 103932, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1751616120304860

[28] A. Gregor, E. Filová, M. Novák, J. Kronek, H. Chlup, M. Buzgo, V. Blahnová, V. Lukášová, M. Bartoš, A. Nečas, and J. Hošek, "Designing of PLA scaffolds for bone tissue replacement fabricated by ordinary commercial 3D printer," *Journal of Biological Engineering*, vol. 11, no. 31, 2017. [Online]. Available: https://doi.org/10.1186/s13036-017-0074-3

[29] A. Seyedsalehi, L. Daneshmandi, M. Barajaa, J. Riordan, and C. T. Laurencin, "Fabrication and characterization of mechanically competent 3D printed polycaprolactone-reduced graphene oxide scaffolds," *Scientific Reports*, vol. 10, no. 22210, 2020. [Online]. Available: https://doi.org/10.1038/s41598-020-78977-w

[30] I. Maskery, N. T. Aboulkhair, A. O. Aremu, C. J. Tuck, and I. A. Ashcroft, "Compressive failure modes and energy absorption in additively manufactured double gyroid lattices," *Addit. Manuf.*, vol. 16, pp. 24 – 29, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214860417301203

[31] I. Maskery, A. O. Aremu, L. Parry, R. D. Wildman, C. J. Tuck, and I. A. Ashcroft, "Effective design and simulation of surface-based lattice structures featuring volume fraction and cell type grading," *Materials and Design*, vol. 155, pp. 220–232, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S026412751830443X

[32] R. R. J. Sélo, S. Catchpole-Smith, I. Maskery, I. Ashcroft, and C. Tuck, "On the thermal conductivity of alsi10mg and lattice structures made by laser powder bed fusion," *Additive Manufacturing*, vol. 34, p. 101214, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214860420305868

[33] E. F. Lehder, I. A. Ashcroft, R. D. Wildman, L. A. Ruiz-Cantu, and I. Maskery, "A multiscale optimisation method for bone growth scaffolds based on triply periodic minimal surfaces," *Biomechanics and Modeling in Mechanobiology*, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s10237-021-01496-8#citeas

[34] J. Klinowski, A. Mackay, and H. Terrones, "Curved surfaces in chemical structure," *Philos. Trans. Roy. Soc. London A*, vol. 354, no. 1715, pp. 1975–1987, 1996. [Online]. Available: http://rsta.royalsocietypublishing.org/content/354/1715/1975

[35] P. J. F. Gandy, S. Bardhan, A. L. Mackay, and J. Klinowski, "Nodal surface approximations to the P, G, D and I-WP triply periodic minimal surfaces," *Chem. Phys. Lett.*, vol. 336, no. 3, pp. 187 – 195, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0009261400014184

[36] M. Wohlgemuth, N. Yufa, J. Hoffman, and E. L. Thomas, "Triply periodic bicontinuous cubic microdomain morphologies by symmetries," *Macromolecules*, vol. 34, no. 17, pp. 6083–6089, 2001. [Online]. Available: http://dx.doi.org/10.1021/ma0019499

[37] H. Schwarz, *Gesammelte Mathematische Abhandlungen*. Berlin: Springer, 1885, vol. 1 & 2.

[38] A. H. Schoen, "Infinite periodic minimal surfaces without self-intersections. NASA technical report TN D-5541," NASA, Washington, D.C., Tech. Rep., 1970.

[39] D. Hoffman, J. Hoffman, M. Weber, M. Trazet, M. Wohlgemuth, E. Boix, M. Callahan, E. Thayer, and F. Wei., "Table of surfaces," Jan. 2005. [Online]. Available: http://www.msri.org/publications/sgp/jim/papers/morphbysymmetry/table/index.html

[40] M. Maldovan, C. K. Ullal, J.-H. Jang, and E. L. Thomas, "Sub-Micrometer Scale Periodic Porous Cellular Structures: Microframes Prepared by Holographic Interference Lithography," *Advanced Materials*, vol. 19, no. 22, pp. 3809–3813, 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.200700811

[41] V. S. Deshpande, N. A. Fleck, and M. F. Ashby, "Effective properties of the octet-truss lattice material," *J. Mech. Phys. Sol*, vol. 49, no. 8, pp. 1747 – 1769, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022509601000102

[42] C. Ling, A. Cernicchi, M. D. Gilchrist, and P. Cardiff, "Mechanical behaviour of additively-manufactured polymeric octet-truss lattice structures under quasi-static and dynamic compressive loading," *Materials & Design*, vol. 162, pp. 106–118, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0264127518308426

[43] J. A. Hawreliak, J. Lind, B. Maddox, M. Barham, M. Messner, N. Barton, B. J. Jensen, and M. Kumar, "Dynamic behavior of engineered lattice materials," *Scientific Reports*, vol. 6, no. 28094, 2016. [Online]. Available: https://www.nature.com/articles/srep28094#citeas

[44] M. Benedetti, A. du Plessis, R. O. Ritchie, M. Dallago, S. M. J. Razavi, and F. Berto, "Architected cellular materials: A review on their mechanical properties towards fatigue-tolerant design and fabrication," *Materials Science and Engineering: R: Reports*, vol. 144, no. 100606, 2021. [Online]. Available: https://www-sciencedirect-com.nottingham.idm.oclc.org/science/article/pii/S0927796X21000012

[45] D.-J. Yoo, "Computer-aided porous scaffold design for tissue engineering using triply periodic minimal surfaces," *Int. J. Precis. Eng. Mat.*, vol. 12, no. 1, pp. 61–71, Feb 2011. [Online]. Available: https://doi.org/10.1007/s12541-011-0008-9

[46] A. Arshad, A. Nazir, and J.-Y. Jeng, "The effect of fillets and crossbars on mechanical properties of latticestructures fabricated using additive manufacturing," *Int J Adv Manuf Technol*, vol. 111, pp. 931–943, 2020. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s00170-020-06034-x.pdf

[47] G. Savio, S. Rosso, A. Curtarello, R. Meneghello, and G. Concheri, "Implications of modeling approaches on the fatigue behavior of cellular solids," *Additive Manufacturing*, vol. 25, pp. 50–58, 2019. [Online]. Available: https://www-sciencedirect-com.nottingham.idm.oclc.org/science/article/pii/S2214860418305517?via=ihub

[48] D. I. Stoia, E. Linul, and L. Marsavina, "Influence of manufacturing parameters on mechanical properties of porous materials by selective laser sintering." *Materials*, vol. 12, 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6471919/

[49] P. Hammer. (2020) Marching cubes. [Online]. Available: https://uk.mathworks.com/matlabcentral/fileexchange/32506-marching-cubes
[50] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169, aug 1987. [Online]. Available: http://portal.acm.org/citation.cfm?doid=37402.37422
[51] A. Aitkenhead. (2020) Mesh voxelisation. [Online]. Available: https://uk.mathworks.com/matlabcentral/fileexchange/27390-mesh-voxelisation
[52] Matlab. (2021) The PLY file format. [Online]. Available: https://www.mathworks.com/help/vision/ug/the-ply-format.html
[53] A. Aitkenhead. (2021) Write stl. [Online]. Available: https://uk.mathworks.com/matlabcentral/fileexchange/29585-compute-mesh-normals?focused=43c5d28a-9a0c-15f0-0618-d60ccf8d36ac&tab=function
[54] Paraview. (2021) Paraview. [Online]. Available: https://www.paraview.org/
[55] G. Dong, Y. Tang, and Y. F. Zhao, "A 149 line homogenization code for three-dimensional cellular materials written in matlab," *J. Eng. Mater.-T. ASME*, vol. 141, no. 1, p. 011005, 2018.
[56] Z. Y. Dong G, Tang Y. (2021) A 149 line homogenization code for three-dimensional cellular materials written in matlab. [Online]. Available: https://github.com/GuoyingDong/homogenization
[57] Z. Chen, Y. M. Xie, X. Wu, Z. Wang, Q. Li, and S. Zhou, "On hybrid cellular materials based on triply periodic minimal surfaces with extreme mechanical properties," *Mater. Design*, vol. 183, p. 108109, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0264127519305477
[58] D. Li, W. Liao, N. Dai, G. Dong, Y. Tang, and Y. M. Xie, "Optimal design and modeling of gyroid-based functionally graded cellular structures for additive manufacturing," *Computer-Aided Design*, vol. 104, pp. 87–99, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448518300381

Appendix A. Methods: The operation of FLatt Pack

This section describes how the theoretical principles of section 2 are implemented in the FLatt Pack program. The main aim is the efficient creation of lattice output files for manufacture and analysis, with the minimal possible memory consumption. Where relevant, representative extracts of the FLatt Pack code are presented (in the Matlab programming language).

Appendix A.1. Lattice design steps

FLatt Pack enables a user to create lattice structures based on the following design variables; geometry, size, number of repeating cells, cell type, volume fraction, and the addition or omission of an external solid 'skin'. Influenced by early user experience of the FLatt Pack GUI, as well as programming considerations regarding the order of mathematical operations for lattice generation, a sequential design process emerged as the best option.

For any given lattice design, a critical interdependence exists between the cell type and volume fraction, and the *spatial resolution* required to generate that lattice programmatically. Central to this is the fact that a high spatial resolution is required to accurately generate lattice cells with fine struts or walls, whereas a lower spatial resolution, and therefore ultimately a smaller quantity of RAM, can be used when the cells have thicker geometrical features. This means that, for the optimal use of computer resources, choosing cell type *then* volume fraction was established in FLatt Pack, with the user's choices of these variables determining the spatial resolution for lattice generation. The complete order of lattice design steps in FLatt Pack is shown in figure A.8, with the implementation of these being described in the rest of this section.

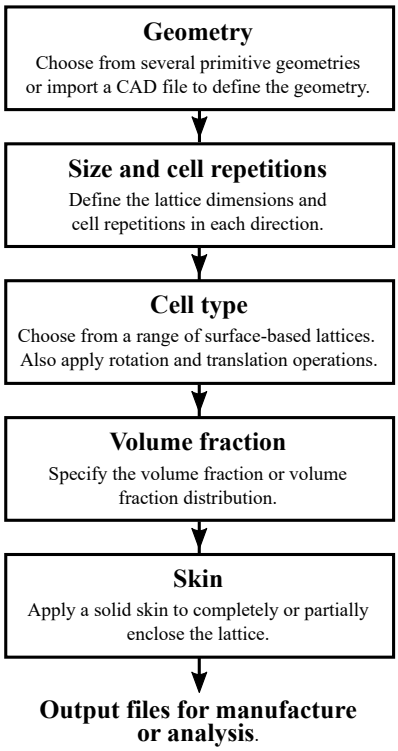


Figure A.8: The order of lattice design variable selection in FLatt Pack.

Appendix A.2. The design space and mesh resolution

Several 3D numerical arrays are created during FLatt Pack's operation. These include three arrays containing x, y and z coordinates for the lattice design space, a *material array* designating which regions of the design space are inside (elements with value 1) and outside (0) the lattice domain, a *density array* specifying the volume fraction at the level of individual elements, and the U array that results from the 3D surface equations of section 2. By way of illustration, and for ease of interpretation compared to 3D illustrations, figure A.9 shows 2D examples of these arrays and the resulting U array.

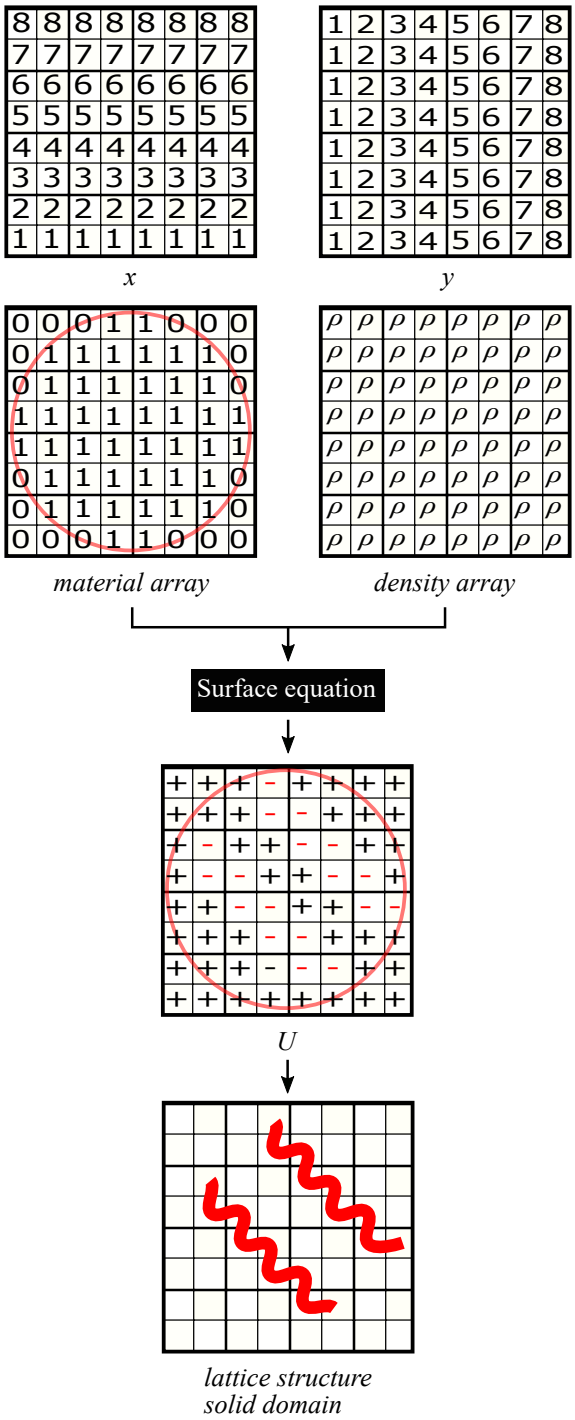
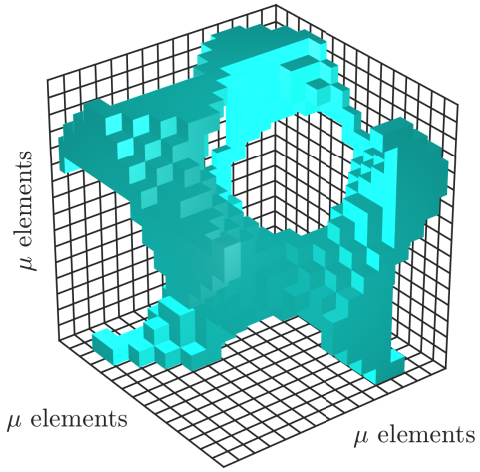


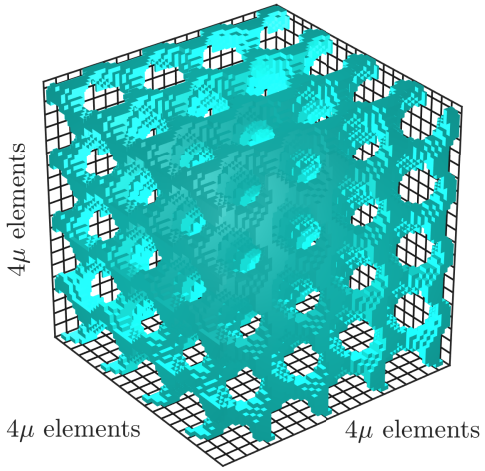
Figure A.9: 2D illustrations of the numerical arrays generated during FLatt Pack's operation. The x and y arrays specify the spatial coordinates, while the *material array* designates the extent of the lattice domain (highlighted here with a shaded red circle). The *density array* contains the volume fraction at every spatial element. The U array comprises positively (+) and negatively (-) valued elements from which the solid-void boundary of the lattice is extracted. The boundary is located at the $U = 0$ isosurface, with the negative values being in the solid domain, as shown in the lowermost panel of the figure.

Each of the six numerical arrays mentioned above is of equivalent size. We may designate that size here as $\mu n_x \times \mu n_y \times \mu n_z$ voxels, where n_x, n_y and n_z are the numbers of cell repetitions in each direction, and μ

is the spatial discretisation, a constant across all directions. Thus, each lattice cell is effectively generated in a cubic array with μ voxels along each edge, as illustrated in figure A.10.



(a) A unit cell in a spatial mesh of $\mu \times \mu \times \mu$ elements.



(b) $4 \times 4 \times 4$ cells in a spatial mesh of $4\mu \times 4\mu \times 4\mu$ elements.

Figure A.10: The spatial discretisation, μ, determines the total number of elements in FLatt Pack's arrays. Shown here is a gyroid unit cell (a) and a lattice based on the same cell type (b). Note that these are voxel models, not the smooth-featured boundary representations used to construct output STL files for manufacture.

μ is a key parameter in FLatt Pack. It determines the fineness of the lattice features (e.g., cell struts or walls) which may be accurately resolved with a given spatial mesh, and plays the dominant role in determining the memory consumption and execution time of FLatt Pack, since these will in general be proportional to μ^3.

There is no single value for μ which would be suitable for all the cell types in FLatt Pack. This is because

the lattice cells differ greatly in their geometry; some cells possess long thin struts (e.g., the BCC and simple cubic cells), others comprise continuous walls of almost constant thickness (e.g., the matrix phase TPMS cells) and others have nodes and connecting regions of varying morphology and thickness (e.g., the network phases TPMS cells). It is therefore a difficult task for the designer of a surface-based lattice to choose an appropriate spatial discretisation for its generation. Choosing too low a value for μ will result in the lattice cell geometry being inaccurately reproduced, since the spatial discretisation will be insufficient to capture fine geometrical features. Choosing too high a value for μ will result in a the generation of much larger numerical arrays than are necessary, leading to greater RAM usage, longer execution time and larger output files.

Sensible default μ values are provided for each lattice type in FLatt Pack. This is implemented by first setting a default spatial discretisation value, μ_0, then modifying this according to the cell type chosen by the user. For example;

```
mu_0 = 40;

switch cell_type
    case 'Gyroid'
        mu = mu_0.*sqrt(2.6);
    case 'Primitive'
        mu = mu_0.*sqrt(0.9);
    case 'Diamond'
        mu = mu_0.*sqrt(3.1);
    case 'Lidinoid'
        mu = mu_0.*sqrt(3.6);
end
```

where $\mu_0 = 40$ is the default value, and μ for each cell type is simply μ_0 multiplied by a scaling factor. Thus, for example, a lidinoid cell will be generated with a finer spatial mesh than a gyroid cell. Inspection of their respective geometries reveals why this is necessary, as the lidnoid cell is far more tortuous and possesses thinner solid struts than a gyroid cell of equivalent volume. The cell type μ scaling parameters ($\sqrt{3.6}$ for the lidinoid cell, etc.) emerged from an initial comparison of the surface-to-volume ratios of each lattice type, followed by iterative refinement to provide a balance between geometrical accuracy and RAM usage.

Further scaling of μ is done with respect to the volume fraction, ρ^*. This is implemented with;

```
reduction_factor = lattice_dens./lower_limit;

% Round up to convert to integer
mu_final = ceil(mu./(reduction_factor.^(1/2)));
```

where *lattice_dens* is the volume fraction chosen by the user and *lower_limit* is the lowest volume fraction accessible for the chosen lattice type. *lower_limit* is pre-defined in FLatt Pack (it is not a parameter the user can modify); its main purpose is to ensure that generated lattices do not possess struts or walls that diminish to zero thickness.

From the code extract above, the value of μ ultimately used to generate the user-chosen lattice is proportional to $1/\sqrt{\rho^*}$. As with the cell type scaling described above, this $\mu \propto 1/\sqrt{\rho^*}$ scaling was the result of iterative refinement, and has been found to provide good geometrical accuracy over a wide range of volume fraction, without increasing the RAM demands beyond those generally available on a modern workstation or laptop computer (i.e., around 16 GB).

In summary, the default spatial resolution in FLatt Pack is not single-valued across all lattice designs, but rather depends on both the chosen cell type and volume fraction. This removes the burden of selecting the resolution from the user and helps minimise FLatt Pack's computational demands by limiting the size of numerical arrays stored in the RAM. One important consequence of this is that, for a given choice of lattice type and volume fraction, FLatt Pack's processing time and RAM usage are mainly dependent on the number of unit cells in the lattice. For example, table A.2 below provides some peak FLatt Pack RAM usage figures and run times for a selection of lattice designs, using default resolution settings and a volume fraction of 0.15. These were obtained using a laptop computer with an Intel i7-6600U CPU, 16 GB of RAM and a 64-bit Windows operating system. The same tasks on a higher specification machine will of course

Table A.2: FLatt Pack peak RAM and run time for a selection of lattice designs, with a volume fraction of 0.15.

Cell type	Cell numbers	Peak RAM usage	STL file creation time
Network gyroid	$5 \times 5 \times 5$	0.8 GB	12 s
Network gyroid	$10 \times 10 \times 10$	3.1 GB	45 s
Matrix gyroid	$5 \times 5 \times 5$	2.6 GB	40 s
Matrix gyroid	$10 \times 10 \times 10$	13 GB	360 s

require less time.

Appendix A.3. The 3D surface array, U

The 3D surface equations in section 2 comprise sine and cosine functions of x, y and z. In FLatt Pack these are given the shorthand *sx,* etc., and are pre-calculated prior to the generation of the 3D surface, U. For example;

```
cx = cos(kx*(x/Lx + x_offset));
cy = cos(ky*(y/Ly + y_offset));
cz = cos(kz*(z/Lz + z_offset));
```

where the notation is the same as given in section 2 and *x_offset*, etc., are included to modify the lattice by spatial translation if requested. Pre-calculation of sine and cosine functions in this manner reduces the code execution time and memory consumption, especially in cases such as the I-WP cell type (equation 4d), where the same trigonometric terms appear more than once.

Further memory and execution time gains are made by only pre-calculating trigonometric terms if they are required for the cell type chosen by the user. For example, only the lidinoid, split P and BCC cell types require doubly periodic cosine terms, and only the lidinoid and split-P types require doubly periodic sine terms. This is implemented as;

```
% Define trig terms only if needed for
% the chosen cell type
if any(strcmp(cell_type,{'Lidinoid', ...
        'Split_P', 'BCC'}))
    c2x = cos(2*kx*(x/Lx + x_offset));
    c2y = cos(2*ky*(y/Ly + y_offset));
    c2z = cos(2*kz*(z/Lz + z_offset));

    if ~strcmp(cell_type, 'BCC')
        s2x = sin(2*kx*(x/Lx + x_offset));
        s2y = sin(2*ky*(y/Ly + y_offset));
        s2z = sin(2*kz*(z/Lx + z_offset));
    end
end
```

The same logic is applied for all the trigonometric terms expressed in equations 4, 5 and 6. The pre-calculated trigonometric terms are then called to generate the 3D surface, U, using, for example;

```
% Specify exponent of surface equation
if any(strcmp(network_matrix,...
        {'Matrix', 'Honeycomb'}))
    m = 2;
else
    m = 1;
end

% Specify surface equation based on
% cell type selection
switch cell_type
    case 'Gyroid'
        U = (sx.*cy + sy.*cz + ...
            sz.*cx).^m ...
            - t.^m;
    case 'Primitive'
        U = (cx + cy + cz).^m ...
            - t.^m;
    case 'Diamond'
        U = (cx.*cy.*cz + sx.*sy.*cz + ...
            sx.*cy.*sz + cx.*sy.*sz).^m ...
            - t.^m;
    case 'Lidinoid'
        U = ((s2x.*sz.*cy + ...
            s2y.*sx.*cz + ...
            s2z.*sy.*cx) - ...
            (c2x.*c2y + c2y.*c2z + ...
            c2z.*c2x)).^m ...
            - t.^m;
end
```

The generation of the network, matrix or honeycomb phases of equivalently named cell types is achieved by setting the constant $m = 1$ or $m = 2$ (as described in section 2), thus reducing the number of separate function definitions for a more efficient representation in the FLatt Pack code.

Appendix A.4. Conformal skin generation

If requested by the FLatt Pack user, a solid conformal skin is added to the outside surface of the lattice. The skin is generated by performing a 3D morphological binary erosion (the *imerode* Matlab function) on the *material array*, as shown here;

```
% Round up to an integer number of voxels
skin_vox = ceil(skin_mm./spatial_res);

material_array_padded  = ...
    padarray(material_array, [1 1 1]);

cuboid_element = strel('cube', skin_vox);
eroded_array = ...
    imerode(material_array_padded,...
    cuboid_element, 'same');

eroded_array = ...
     eroded_array(2:end-1, 2:end-1, 2:end-1);

% Boolean subtraction to yield the skin array
skin_array = logical(material_array - eroded_array);
```

The result is an additional logical array, called the *skin array*, in which voxels of value 1 represent the conformal skin. This is then added to the 3D surface array, U, in a boolean operation to yield the final array for surface mesh generation, as illustrated in 2D in figure A.11.

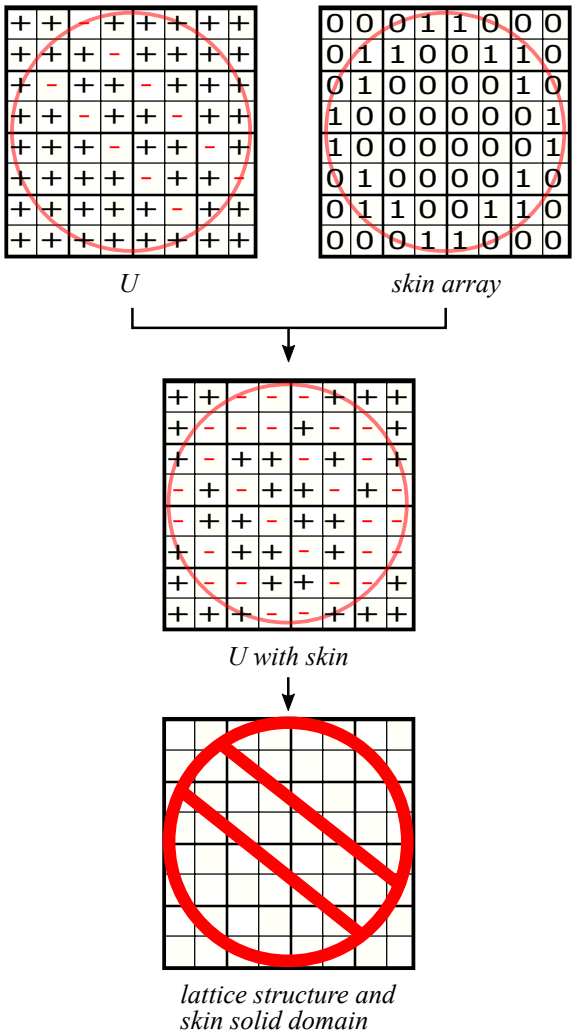


Figure A.11: 2D illustrations of the FLatt Pack skin generation method. (The (+) and (-) symbols indicate positively and negatively valued elements of U, as in figure A.9.)

The requested skin thickness is provided by the user in units of mm, which must be converted in FLatt Pack to an integer number of voxels for the binary erosion operation. This is done in the first line of the code extract above, where *skin_mm* is the user-specified skin thickness in mm, *spatial_res* is the resolution of the underlying spatial mesh in units of mm/voxel and *skin_vox* is the resulting skin thickness expressed as an integer number of voxels. Thus, the realised skin thickness is an approximation of the value requested by the user, with a maximum deviation of 1 voxel $\times$ *spatial_res*. For lattices in the size range $1-30$ cm, which spans the dimensions of most commercial AM platforms, this deviation could be up to around 0.5 mm, but on average is only half of that; as is demonstrated in figure A.12. In figure A.12, the skin thickness requested by the user is 2 mm, while the average realised thickness is 1.76 mm. Requested conformal skin thicknesses for decimetre-scale lattices are typically > 2 mm, so an *absolute* deviation of ~ 0.25 mm represents a relatively small *fractional* deviation from the requested design. More importantly, STL slicing and hatching operations *and* AM processes in general introduce inherent manufacturing resolution restrictions, meaning that small skin thickness errors introduced by FLatt Pack at the STL stage are unlikely to directly result in over- or under-thick lattice skins in manufactured parts.

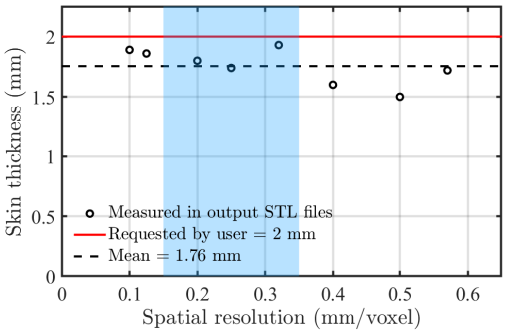


Figure A.12: Skin thickness measured in STL files from FLatt Pack, with a user-requested value of 2 mm. The central shaded region indicates the range of spatial resolution typically used when the user opts for FLatt Pack's default resolution settings (see Appendix A.2). Note that the maximum deviation between requested and realised skin thickness depends on the spatial resolution, which in turn is determined by the user's choice of lattice dimensions, cell type and volume fraction. Therefore, for a particular volume fraction and lattice size, the results in this plot apply across all the available cell types in FLatt Pack.

Appendix A.5. Surface mesh creation

The lattice structure is defined by the $U = 0$ isosurface of the 3D surface array. If requested by the user, the lattice surface is discretised into a triangular mesh, so that it may be exported in a useful format for manufacture (i.e., an STL file). This is done in FLatt Pack using the *MarchingCubes* function from Matlab Central File Exchange [49], which is based on the Marching Cubes algorithm [50]. This is implemented with;

```
[faces, vertices] = MarchingCubes(x, y, z, U, 0);
```

where *faces* and *vertices* are lists of the triangular surface elements and their connecting vertices which describe the solid-void boundary. The number of triangles generated by the *MarchingCubes* function depends on the user's selection of lattice type, volume fraction and, most critically, the resolution of the 3D surface array. The *MarchingCubes* function is cell-centered, meaning the geometric accuracy of the surface element vertices is $\pm\frac{1}{2}$ of the size of the voxels in the 3D surface array.

Surface-based lattice structures possess high surface-to-volume ratio, so a representation of a lattice as a triangular surface mesh can result in large files. To address this, FLatt Pack includes the option to reduce the number of surface triangles by replacing groups of small surface elements with larger ones. This is achieved using the Matlab *reducepatch* function;

```
[faces_reduced, vertices_reduced] = ...
    reducepatch(faces, vertices, ...
     triangle_red_factor);
```

where *triangle_reduction_factor* is provided by the user in the FLatt Pack GUI. It is a number between 0 and 1 (or, as actually expressed in the GUI, 0% and 100%) specifying the fraction of triangles originally generated by the *MarchingCubes* function to include in the output surface mesh. Further information about how this mesh is stored as an output file is given in Appendix A.7.

Appendix A.6. Lattice generation from custom input geometries

As well as several 'primitive' geometries including tailorable cuboid, cylinder and spheroid (which are useful as lattice specimens for experimental testing), FLatt Pack allows the user to generate a lattice based on a 'Custom' input geometry. This is provided by the user as an STL file, which is then voxelised in FLatt Pack to define the lattice design space and *material array*. This is done using the *READ_stl* and *VOXELISE* functions from Matlab Central File Exchange [51], courtesy of A. Aitkenhead. The *VOXELISE* function uses a ray intersection method in which rays are emitted in each Cartesian direction (positive and negative x, y and z) from each point in space. Wherever these rays intersect the surface mesh, this is regarded as a solid-void boundary, with the number of times the mesh is intersected determining whether the originating point of the rays is inside or outside the solid space. This is done in FLatt Pack with;

```
material_array = ...
    logical(VOXELISE(b, a, c, stl_vertices));
```

where a, b and c are the numbers of elements in the *material array* in the x, y and z directions, respectively, and *stl_vertices* is a structured list of the vertices in the input STL mesh. The output of this voxelisation is a *material array* with 0's and 1's in the void and solid domains, exactly as depicted in figure A.9, which is then used to generate the 3D surface array, U.

It is worth noting that this method for processing user-supplied surface meshes can lead to 'staircase' features at the outer boundaries of subsequently exported lattice designs, with the size of these features, and therefore the overall impact on part definition, being dictated by the spatial resolution, μ, of the FLatt Pack arrays. This is an inevitable consequence of the voxelisation process, but nonetheless represents a drawback of the FLatt Pack implementation. At present, a partial correction is achieved with a 3D Gaussian smoothing kernel, which is automatically applied whenever the user opts for the 'Custom' input geometry route. The result is the reduction of surface staircase artifacts to levels unlikely to be observed in manufactured parts, i.e., considering the effects of slicing and hatching operations for AM build preparation, as well as the resolution of the AM process itself. A better solution, one which preserves the original boundaries of the user-supplied geometry as far as possible, will be implemented in future versions of FLatt Pack.

Appendix A.7. Output file formats

FLatt Pack offers a small selection of output file types chosen to be useful for lattice structure research. They are based on either the triangular surface mesh originating from the *MarchingCubes* algorithm (see Appendix A.5), or a binarised version of U in which 0's and 1's represent the void and solid domains of the lattice, respectively. The former category includes the STL file format, which is used throughout AM, and the PLY (or Polygon) file format [52], which is the same triangular mesh represented in a more compact form through the inclusion of shared vertices.

The PLY data is extracted from the original *faces* and *vertices* lists with;

```
[vertices_ply, I] = sortrows(vertices);
M = [true; any(diff(vertices_ply), 2)];
vertices_ply = vertices_ply(M, :);
I(I) = cumsum(M);
faces_ply = I(faces) - 1;
```

and the resulting *faces_ply* and *vertices_ply* lists are then passed to;

```
WRITE_plybinary(filename_PLY, vertices_ply, ...
faces_ply)
```

which simply writes them in the correct structure, along with the required header information, in a file designated by *filename_PLY*.

Similarly, the STL file is written using;

```
WRITE_stlbinary(file_path_stl, ...
    faces_vertices, normals, filename_stl);
```

where *faces_vertices* is a structure containing the face and vertex data, and the list of *normals* identifies which side of each triangular element is inward- or outward-facing. The *WRITE_stlbinary* function is based on *WRITE_stl* from the Matlab Central File Exchange [53], again courtesy of A. Aitkenhead. The FLatt Pack implementation of *WRITE_stlbinary* includes a modest speed-up modification using the Matlab *typecast* function to convert the face, vertex and normal data to unsigned 16-bit integer arrays before writing the file.

With respect to the non-surface-mesh output file types, the MAT option is simply a 3D Matlab array with 0's and 1's representing the void and solid domains of the lattice. This is useful to the lattice structure researcher as it enables further modification and analysis operations in Matlab, which might include, for example, assessment of pore dimensions and connectivity. The BMP output option is a series, or 'stack', of bitmap images representing each layer in the structure's z direction, with black pixels being the solid domain and white pixels being void. Again, this output type lends itself to structural analysis, but the bitmap format

is also a suitable input for some AM platforms, most typically those based on discrete droplet deposition. The VTK option produces a structured XML-based file output, where each layer in z is represented as a separate data array of void (0) and solid (1) voxels. This file type is included as it is directly compatible with the popular open-source data analysis and visualisation software Paraview [54].

Lastly, FLatt Pack provides the option to export a lattice structure in the form of an FE mesh. In this mesh, solid regions are represented as hexahedral elements, which map directly to the binarised U array; i.e., each solid voxel in the lattice corresponds to a hexahedral element in the FE mesh. This element transformation results in list of nodes (an ID number and set of x, y, z coordinates for each) and a list of elements which are defined by those nodes. This could be exported in a number of file formats for use in various FE solvers; the specific file created by FLatt Pack is the Abaqus input file, INP. In its simplest form, this is done with;

```
% Open the file
fid2 = fopen(filename, 'w');

element_ID = (1:size(node_ID, 2));
element_list = [element_ID; node_ID];

node_list = [(1:size(node_coords,1))', ...
    node_coords]';

% Write the header information
fprintf(fid2,'*HEADING \n');
fprintf(fid2, ...
    'HEX MESH GENERATED FROM VOXEL IMAGE \n');
fprintf(fid2,'**\n');

% Write node information
fprintf(fid2,'**NODE DEFINITION \n');
fprintf(fid2,'*NODE, NSET=ALL \n');
fprintf(fid2,'%u, %f, %f, %f \n', node_list);

% Write element information
fprintf(fid2,'*ELEMENT, ELSET=ALL, TYPE=C3D8R \n');
fprintf(fid2, ...
    '%u, %u, %u, %u, %u, %u, %u, %u, %u \n', ...
    element_list);

% Close the file
fclose(fid2);
```

where *element_ID* and *node_ID* are integers labelling each element and node, and *node_coords* are the spatial coordinates of each node, expressed in units of metres. This produces, for example;

```
*HEADING
HEX MESH GENERATED FROM VOXEL IMAGE
**
**NODE DEFINITION
*NODE, NSET=ALL
1, 0.000000, 0.000909, 0.005455
2, 0.000000, 0.000909, 0.006364
3, 0.000000, 0.000909, 0.007273
4, 0.000000, 0.001818, 0.005455
etc.

*ELEMENT, ELSET=ALL, TYPE=C3D8R
1, 52, 98, 107, 60, 53, 99, 108, 61
2, 60, 107, 115, 67, 61, 108, 116, 68
3, 98, 143, 155, 107, 99, 144, 156, 108
4, 107, 155, 163, 115, 108, 156, 164, 116
etc.
```

which contains only the most elementary information required for an FE mesh. Further information may

then be added to the INP file by the researcher to create a processable job for the Abaqus FE solver.

FLatt Pack provides two complementary options for INP file creation, which extend its usefulness in lattice research by adding material properties, boundary conditions and a history output request. For example;

```
fprintf(fid2,'*STEP \n');
fprintf(fid2,'*STATIC \n');
fprintf(fid2,'*BOUNDARY \n');
fprintf(fid2,'TOP_PLANE, %0.f, %0.f, %d \n', ...
    3, 3,  displacement);

% Specify output to dat file
fprintf(fid2, ...
    '*NODE PRINT, NSET=TOP_PLANE, TOTALS=YES \n');
fprintf(fid2,'U3, RF3 \n');
```

which designates a uniaxial displacement load at the top surface of the lattice (*TOP_PLANE*), followed by a request to print the resulting z displacements (*U3*) and reaction forces (*RF3*) to an output file when the FE job is complete. The FLatt Pack user may specify a host of similar FE jobs based on compressive or tensile loading along x, y or z, which is useful for the examination of lattice stiffness and stress distributions under different loading conditions. The other FE job directly prescribable in FLatt Pack concerns thermal conductivity, in which the top and bottom surfaces of the lattice are assigned different temperatures and the solver is requested to determine the temperature distribution through the solid region. A number of material and loading parameters can be adjusted by the FLatt Pack user (see Appendix B.10), which are then written to the appropriate sections of the INP file. The user can of course edit the resulting INP file before submitting it to the FE solver, which is useful for requesting different node and element output information, and modifying the loading conditions or material properties.

Appendix A.8. Estimation of elastic modulus

At the same step in the FLatt Pack GUI in which the user is presented with a range of unit cells from which to construct the lattice, they are also able to view an estimate of the resulting elastic modulus (see Appendix B.5 for illustrations). This takes the form of a separate GUI subwindow in which curves of E^* are plotted as a function of the volume fraction, ρ^*, where E^* is the *relative modulus* of the lattice. E^* is simply the modulus of the lattice normalised by the modulus of the solid material from which it is made, i.e., E_{latt}/E_{sol}, and it takes values between 0 and 1.

$E^* - \rho^*$ correlations are a useful lattice design tool, as they aid the designer in their choice of cell type and volume fraction in the pursuit of a particular structural stiffness. This is the motivation for the inclusion of such correlations in FLatt Pack. As the user selects between different cell types, the appropriate $E^* - \rho^*$ curves are displayed, thus enabling quick comparisons between the predicted moduli of lattice cells under consideration.

Furthermore, the moduli of regular ordered lattices, such as those in FLatt Pack, are generally not isotropic (unlike, for example, stochastic cellular foams, which possess greater elastic isotropy), so it is necessary to give the user some indication of the elastic response under different loading conditions. This is done by presenting $E^* - \rho^*$ curves along the high symmetry [001], [111] and [110] directions, borrowing from crystallographic notation for cubic lattices. These are illustrated in figure A.13 and are sufficient, thanks to the cubic symmetry of the lattices in FLatt Pack, to ensure that the maximal and minimal moduli for each cell type is always provided.

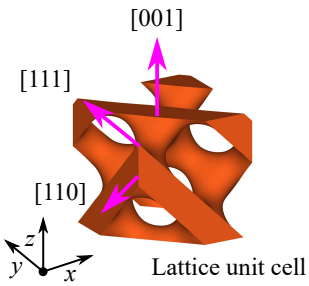


Figure A.13: Loading directions in which FLatt Pack provides estimated elastic moduli for each cell type

The elastic moduli of the FLatt Pack cell types, for a range of volume fractions, were obtained using the finite element homogenisation method of Dong *et al* [55] (which we have described previously in a study of anisotropy in the gyroid lattice [16]). Dong *et al* provided a Matlab implementation of their method [56], which computes the homogenised stiffness matrix, C^H, for a given lattice unit cell. Once C^H is obtained, its inverse provides the homogenised compliance matrix, S^H. The elastic moduli of the lattice along the [001], [111] and [110] directions are then found from [57]:

$$E_{[001]} = \frac{1}{S^H_{11}}, \tag{A.1a}$$

$$\frac{1}{E_{[110]}} = S^H_{11} - \frac{1}{2}\left[\left(S^H_{11} - S^H_{12}\right) - \frac{1}{2}S^H_{44}\right], \tag{A.1b}$$

$$\frac{1}{E_{[111]}} = S^H_{11} - \frac{2}{3}\left[\left(S^H_{11} - S^H_{12}\right) - \frac{1}{2}S^H_{44}\right]. \tag{A.1c}$$

For Flatt Pack, $E^*_{[001]}$, etc., for all cell types and a range of volume fractions, were pre-calculated using Dong *et al*'s code and fit with Gibson-Ashby scaling laws of the form [15]

$$E^*(d) = C_1(d)\ \rho^{*q(d)} + E^*_0(d), \tag{A.2}$$

where d indicates that these properties are directionally dependent. The set of prefactors C_1, exponents q and intercepts E^*_0, are stored in a data file and looked up whenever FLatt Pack is required to plot $E^* - \rho^*$ curves for the modulus estimate subwindow.

Appendix A.9. Estimation of surface-to-volume ratio

In a similar fashion to that described above for elastic modulus estimation, FLatt Pack provides estimates of surface-to-volume ratio for each cell type over a range of volume fractions and cell sizes. This can be a useful design aid when the intended application of the lattice involves the interaction of its surface with the environment; for example, heat exchangers, the performance of which is governed by the transfer of heat across the surface to a fluid medium, and biological scaffolds, where the quantity of surface (as well as properties such as local curvature, etc.) dictates the number of biological cells that can attach and grow there. FLatt Pack provides graphical representations of surface-to-volume ratio, so the user can select the best cell type for these applications (see Appendix B.5 for appropriate illustrations). The user can simultaneously examine the elastic modulus estimate curves, allowing a design judgment based on both surface area and stiffness.

The lattice surface-to-volume ratios are pre-calculated using the Matlab *stlVolume* function;

```
[Enclosed_volume, Surface_area] = ...
    stlVolume(vertices, faces);

Surf_vol_ratio = Surface_area/Enclosed_volume;
```

and are stored in a data file and looked up whenever FLatt Pack is required to provide surface-to-volume

plots. To enure the calculated surface-to-volume ratios are not artificially increased through the inclusion of the planar edges which bound the lattice, the calculation in the code extract above is performed on a $4 \times 4 \times 4$ arrangement of lattice cells, rather than a single unit cell. In this $4 \times 4 \times 4$ arrangement of cells, the relative contribution of the planar edges to the total surface area is minimal, so the resulting surface-to-volume ratios can be seen as broadly representative of the chosen cell type.

Appendix B. Methods: The FLatt Pack GUI

This section provides a guide to the use of FLatt Pack, beginning with program installation and ending with the creation of output files representing a lattice structure. Where relevant, and where this aids the efficient use of FLatt Pack, explanations for the inclusion of certain GUI elements are given.

After installation (described in Appendix B.1) FLatt Pack presents the user with a sequence of windows corresponding to the lattice design steps. These include:

1. Designating the lattice geometry.
2. Choosing the lattice dimensions and cell repetitions.
3. Choosing the cell type and orientation.
4. Specifying the lattice volume fraction (with options for volume fraction grading).
5. Choosing to add an exterior 'skin' to the lattice.
6. Specifying the output filename and file types.
7. Inspecting the lattice design with a 3D graphical preview.
8. Exporting the lattice geometry in the specified file types.

The FLatt Pack GUI windows relating to the above steps are illustrated and described in the following sections, as are several optional subwindows and two preliminary windows which convey general information on the program's operation. The user is able to move back and forth through these windows, and therefore revise previous design choices, using the '< Back' and 'Next >' buttons found in the lower right of each window. Each window also features a 'Cancel' button which, like the '×' in the upper right corner of the window, closes the program.

Appendix B.1. Installation and system requirements

FLatt Pack is written in Matlab and compiled into a standalone application using the Matlab Compiler. As such, neither a Matlab installation, nor a Matlab license, are required to run FLatt Pack on a host PC. Its only dependency is Matlab Runtime, a set of shared libraries which is automatically installed by the FLatt Pack installer or can be obtained freely from the Mathworks website (www.mathworks.com/products/compiler/matlab-runtime.html).

The FLatt Pack installer is freely available via a public Github repository (www.github.com/ian27ax/FLatt_Pack_dist). This provides access to the most recent version of the software, which is frequently updated with new features and bug fixes. Users may report issues and feature requests through the Github tracker. The installer (figure B.14) guides the user through several steps (selecting the installation folder, etc.), including the installation of Matlab Runtime if it is not detected on the host PC. An internet connection is therefore required for the installation of FLatt Pack; if this is not available, the user can contact the corresponding author to obtain a copy of the FLatt Pack installer with Runtime included.

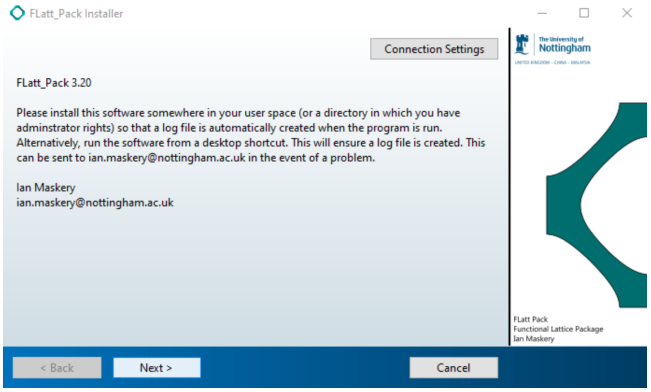


Figure B.14: FLatt Pack Installer.

The host PC should have a 64-bit Windows operating system. There are no specific system requirements regarding memory, or CPU type or speed, but the performance of FLatt Pack will be adversely affected if insufficient memory is available to store the numerical arrays generated during its operation. It is recommended that the host PC has at least 16 GB of RAM.

Appendix B.2. User agreement and program outline

The first window of the FLatt Pack GUI is a user agreement notification (figure B.15), which the user should read before proceeding. The 'User agreement' window also contains buttons to open a new email to the corresponding author, for general queries and bug reporting, and a browser window to the corresponding author's website, where publications related to lattice design are listed. There is also hover-over text which informs the user how to cite the software, should it be used in future work.

Figure B.15: FLatt Pack GUI - User agreement.

FLatt Pack is usually distributed with a time-limited license. The end date of the license is given in the 'User agreement' window, after which FLatt Pack will cease to work and the user will be prompted to contact the corresponding author or obtain a new version of the program from the Github repository. This is principally to ensure that users have access to the most recent version of FLatt Pack, including any bug fixes or performance improvements made since the user's version was compiled.

Following the user agreement, the user is shown the FLatt Pack 'Program outline' (figure B.16), which provides a simple description of the lattice design steps. These are broadly equivalent to those listed above, but with wording which emphasises the creation of STL files for AM.

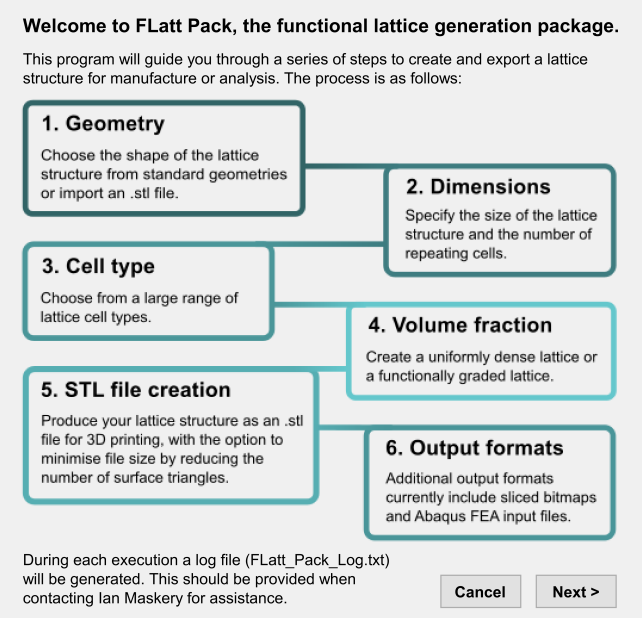


Figure B.16: FLatt Pack GUI - Program outline.

Appendix B.3. Geometry selection

The user is next presented with the 'Geometry' window (figure B.17). Using either the clickable figures toward the bottom of the window, or the radio buttons in the upper left, the user can select between three primitive geometries for the lattice; a cuboid, a cylinder or a spheroid, or can select the 'Custom' option to import a 3D geometry in the form of an STL file. This STL mesh will then define the design space in which the lattice is generated.

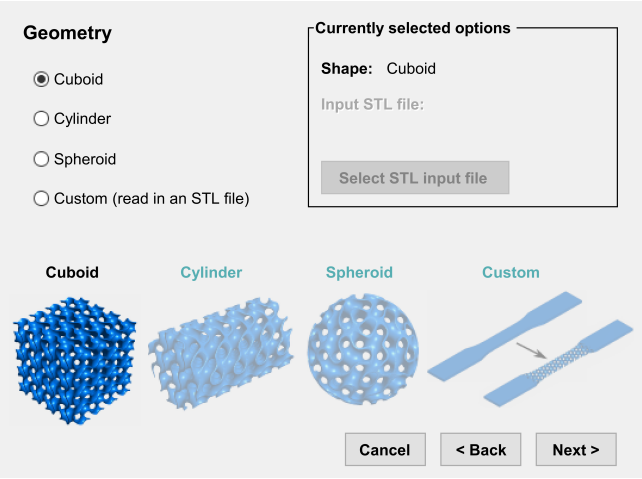


Figure B.17: FLatt Pack GUI - Geometry.

As the user selects between the 'Cuboid', 'Cylinder', 'Spheroid' and 'Custom' options, the relevant figure is highlighted and the others are dimmed to illustrate the current choice. This feature is present in several of the FLatt Pack GUI windows, and is used to make the lattice design process as clear as possible.

Appendix B.4. Dimensions and cell repetitions

After choosing the lattice geometry, the user is asked to specify the dimensions and cell repetitions along each direction. The 'Dimensions and cell repetitions' window (figure B.18) provides editable text boxes for dimensions (in mm) along the x, y and z directions, as well as numbers of cells along those directions. The user can ensure the generation of cubic lattice cells by maintaining an equivalent ratio between the dimension and cell numbers in each direction; e.g., in figure B.18, 30:3 in x, 40:4 in y and 50:5 in z. However, lattice cells of any aspect ratio are permitted; these generally result in cells which appear 'stretched' or 'squashed' compared to their conventional cubic forms; see figure B.19 for example.

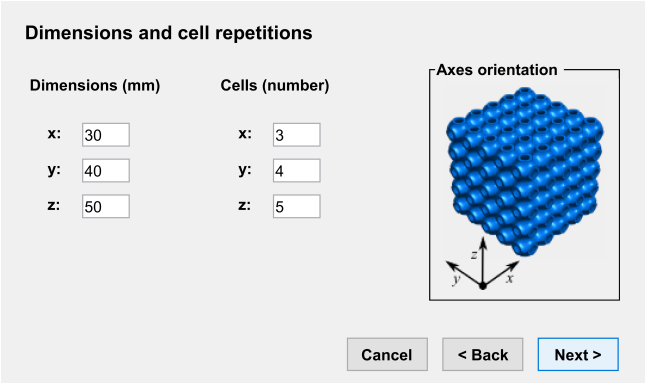


Figure B.18: FLatt Pack GUI - Dimensions and cell repetitions.

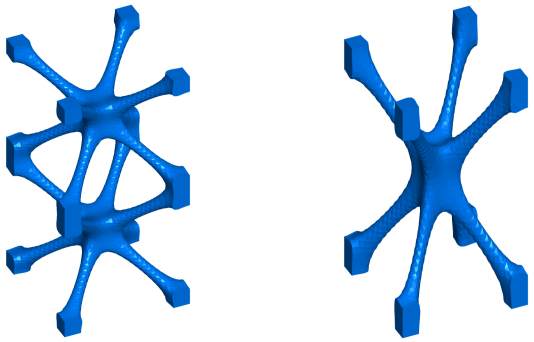


Figure B.19: A lattice of $1 \times 1 \times 2$ body-centred-cubic cells is shown in (a), while (b) shows a single unit cell with the same bounding dimensions. The result is a modified cell geometry which is 'stretched' in the z direction compared to its conventional form.

If the user opted to import an STL mesh as the basis for the lattice geometry, the dimensions in the 'Dimensions and cell repetitions' window will be non-editable; they will be determined directly from the coordinates of the input STL mesh vertices. However, the user still has control over the numbers of lattice cells.

Appendix B.5. Cell type selection

In the 'Cell type' window (figure B.20), the user is presented in the upper left with a choice of four 'classes' (or 'phases'). These represent distinct types of lattice structure; selecting between them alters the range of cell types shown in the central section of the window. There are eight cell types available for both the 'Network' and 'Matrix' class; these are the TPMS gyroid, lidinoid, O,C-TO, I-WP, diamond, primitive, neovius and split-P. Five cell types are available for the 'Honeycomb' class; these are honeycomb versions of

the gyroid, primitive, I-WP, lidinoid and diamond cell types. In the 'Strut' class there are two cell types; body-centred-cubic and simple cubic.

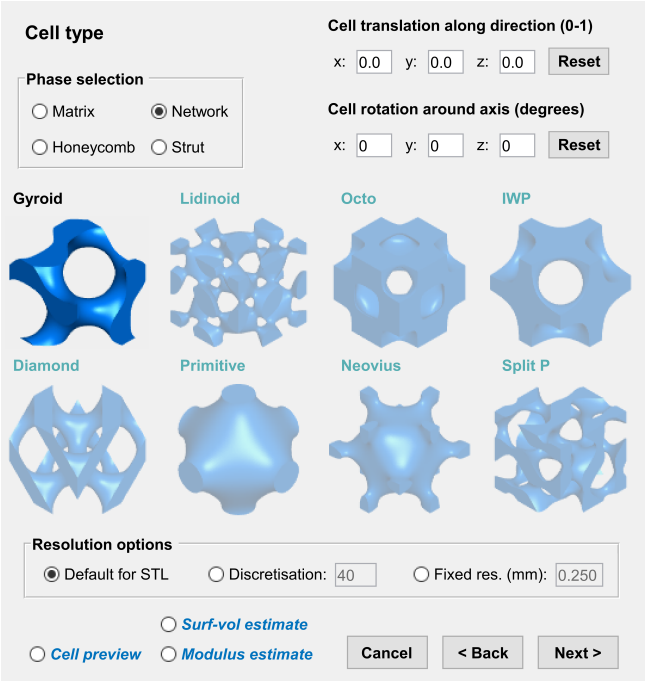


Figure B.20: FLatt Pack GUI - Cell type.

Once the class and cell type are chosen, the user may translate the unit cell in the x, y and z directions and rotate it about each axis (applied in the order z, y, x) using the options in the upper right of the 'Cell type' window. The facility to orient the struts or walls of the cell is included so that the designer may align the high-stiffness direction of a lattice with the direction of an applied load, resulting in minimal deformation.

The 'Cell type' window also contains 'Reset' buttons for cell translation and rotation. These set the editable text box values for translation and rotation to '0', thus restoring the *unmodified* version of the chosen cell type.

The effects of translation and rotation are best represented using the 'Cell preview' radio button in the lower left of the the 'Cell type' window. This opens the 'Cell preview' subwindow (figure B.21), featuring a 3D preview of the chosen cell type. The preview shows a small lattice of 2 or 4 cell tessellations. Cell translations and rotations are applied in this subwindow in real-time, allowing the user to see the effect of their choices before proceeding. Furthermore, the vertical slider to the right of this subwindow allows the user to see the effect of modifying the volume fraction of their chosen cell type.

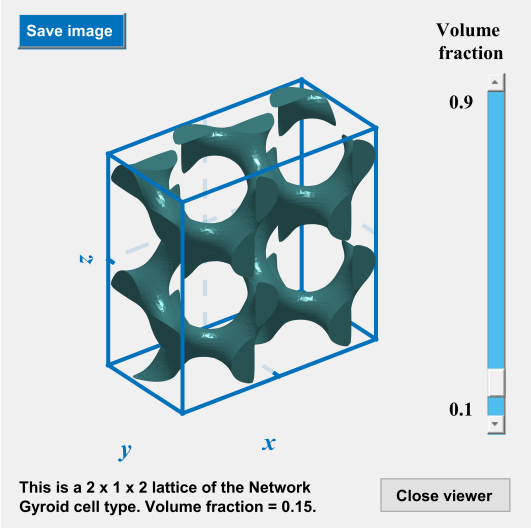


Figure B.21: FLatt Pack GUI - Cell preview.

Three further elements of the 'Cell type' window merit attention. The first is the 'Modulus estimate' radio button, which opens a subwindow of the same name (figure B.22). This provides a plot of relative elastic modulus against volume fraction for the currently selected cell type. Three curves are given, corresponding to modulus estimates along the crystallographic [001], [111] and [110] directions. These are obtained using the numerical homogenisation method outlined in Appendix A.8. The 'Modulus estimate' subwindow therefore provides an estimate of the lattice elastic modulus and anisotropy, through comparison of the [001], [111] and [110] curves. This information allows the user to select the cell type, volume fraction and orientation most suitable for the intended application of the lattice specimen or component.

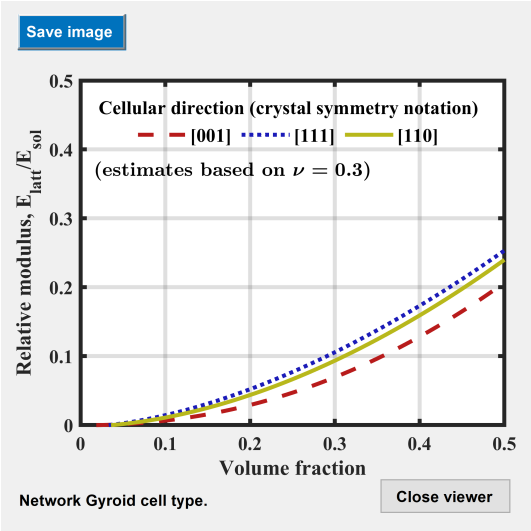


Figure B.22: FLatt Pack GUI - Modulus estimate.

Second, the user may select the 'Surf-vol estimate' radio button, which opens a separate subwindow with a contour plot of surface-to-volume ratio for the currently selected cell type, across a range of cell sizes and volume fractions (figure B.23). This is designed to provide an easily interpretable guide to how the surface-to-volume ratio is affected by the user's lattice design choices. For example, the user can easily

compare the contour plots of several cell types, and proceed to choose the cell type with maximal or minimal surface area, according to the demands of the lattice application. The calculation and design relevance of lattice surface-to-volume ratio is described in Appendix A.9.

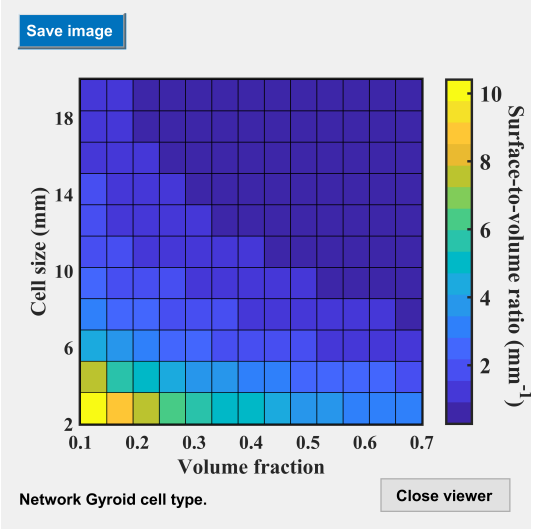


Figure B.23: FLatt Pack GUI - Surface-to-volume ratio estimate.

The last element of note in the 'Cell type' window is the box situated below the cell illustrations which contains three 'Resolution options'. These define the resolution of the design space mesh, as described in Appendix A.2. Editable text boxes allow the user to specify the resolution, either in distance units using the 'Fixed res. (mm)' option, or by prescribing the number of increments, μ, along each axis in a unit cell using the 'Discretisation' option. For example, a discretisation setting of 40 means a single unit cell is generated in a numerical array of $40 \times 40 \times 40$ elements; a lattice of $n_x \times n_y \times n_z$ cells is generated in an array of size $40n_x \times 40n_y \times 40n_z$.

The 'Fixed res. (mm)' and 'Discretisation' options are included to allow the user to precisely control the resolution of voxel meshes and hexahedral FE meshes made using FLatt Pack. For the creation of STL files, the user is encouraged to select the 'Default for STL' option, which provides a spatial resolution sufficient to render the geometry of the chosen cell accurately in the output STL file, without the resolution being *overspecified*. The 'Default for STL' option aims to avoid the generation of unnecessarily large numerical arrays representing the lattice design space, therefore reducing FLatt Pack's memory consumption and execution time compared to a 'one resolution for all lattices' approach. This is described in more detail in Appendix A.2.

Appendix B.6. Volume fraction selection

The 'Volume fraction' window (figure B.24) presents options to create a lattice of uniform volume fraction, graded volume fraction (with some inbuilt mathematical functions for grading) or custom volume fraction, using an input file to describe the volume fraction distribution. The choice is made using the 'Uniform' and 'Graded' radio buttons, as well as the 'Import vol. frac. map' button, in the lower left of the window. A box in the upper right of the 'Volume fraction' window shows the user's currently selected options.

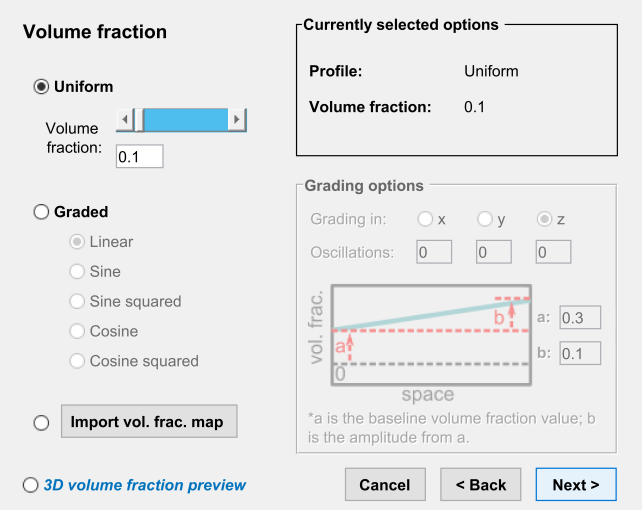


Figure B.24: FLatt Pack GUI - Volume fraction.

Selecting the 'Uniform' option, a slider and editable text box are highlighted, allowing the user to specify a volume fraction for the whole lattice structure. Volume fraction takes values between 0 and 1, representing an entirely void or solid design space, respectively. Internally in FLatt Pack, a volume fraction threshold of 0.9 is used to prevent the creation of almost completely solid structures with separate enclosed voids. These should be avoided in general due to the inability to remove trapped excess material following many AM processes.

A limit is also placed on the lowest accessible volume fraction for each cell type. This is done for two reasons:

1. Low volume fraction cells possess small geometrical features; i.e., thin walls or struts. This necessitates the generation of very large Matlab arrays in order to render the cell geometry accurately. Therefore, limiting the lowest volume fraction available to the user helps minimise FLatt Pack's memory consumption and run time, whilst not overly restricting the user's design choices.
2. Some cell types, for example the network phase primitive cell, cease to have structural connectivity below a certain volume fraction. Limiting the lowest available volume fraction in these cases ensures that the lattice remains structurally intact, can be manufactured and is capable of supporting an applied load. (Pursuing a different approach to this problem in surface-based lattice design, Li *et al* proposed the use of a penalty function which modifies the surface equation to ensure structural connectivity at low volume fraction [58].)

The 'Graded' volume fraction option activates a range of grading function radio buttons ('Linear', 'Sine', 'Sine squared', 'Cosine' and 'Cosine squared') and the 'Grading options' box on the right of the window. These allow the user to choose between linear volume fraction grading, or grading based on sine or cosine functions. The 'Grading options' box has editable text boxes to specify the minimum and maximum volume fractions. The user must also specify in which direction, or directions, the grading should occur and, for the trigonometric functions, the number of oscillations (complete sine or cosine waves) in each direction. At present, volume fraction grading using inbuilt mathematical operations in the FLatt Pack GUI is limited to the Cartesian directions x, y and z (though this limitation does not apply when the user specifies the volume fraction distribution with an input file - see below). For future versions of FLatt Pack, we will consider adding more grading options to the GUI; for example, along the radial direction in cylindrical geometries.

Figures B.25 and B.26 illustrate FLatt Pack's inbuilt volume fraction grading options. In figure B.25, the volume fraction is given by a linear gradient in the z direction between values of 0.1 at the base of the lattice design space, and 0.9 at the top. In figure B.26, the grading is given by one complete sine squared function in both the x and z directions, again between values of 0.1 and 0.9.

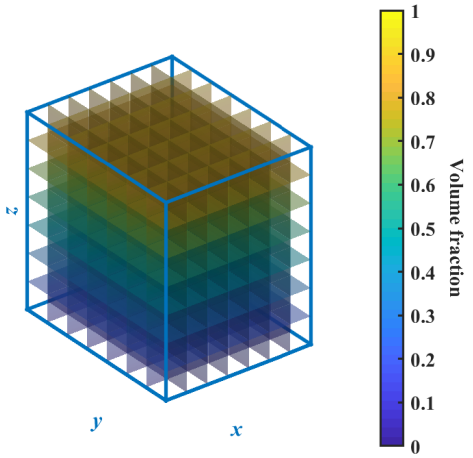


Figure B.25: Volume fraction preview (linear grading).

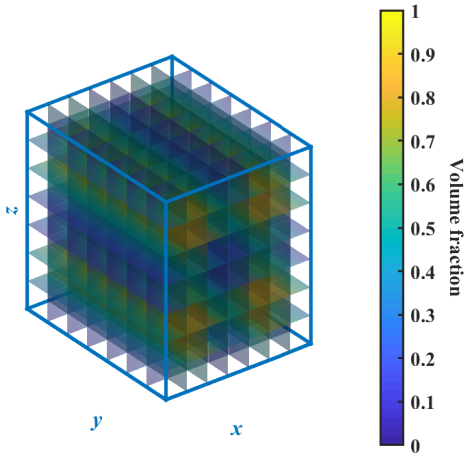


Figure B.26: Volume fraction preview (sine squared grading).

The 'Import vol. frac. map' button allows the user to specify the volume fraction in the lattice design space by importing a file. FLatt Pack currently supports two input file types for this purpose:

1. A Matlab array (.mat) representing the 3D lattice design space, where each element is taken to represent a small volume of space and has a value between 0 (void) and 1 (solid).
2. A 24-bit or monochrome bitmap image (.bmp), where each pixel has a colour between black (r g b = 0 0 0), corresponding to void in the lattice design space, and white (r g b = 255 255 255), corresponding to solid. The bitmap is taken to represent the volume fraction distribution in the xy plane, with the full 3D lattice design space then being constructed by replicating that plane throughout the z direction.

Finally, the 'Volume fraction' window contains a radio button in the lower left corner to provide a '3D volume fraction preview'. This opens a subwindow (figure B.27) showing a 3D preview of the volume fraction distribution in the lattice design space, and is updated in real-time according to the user's choices.

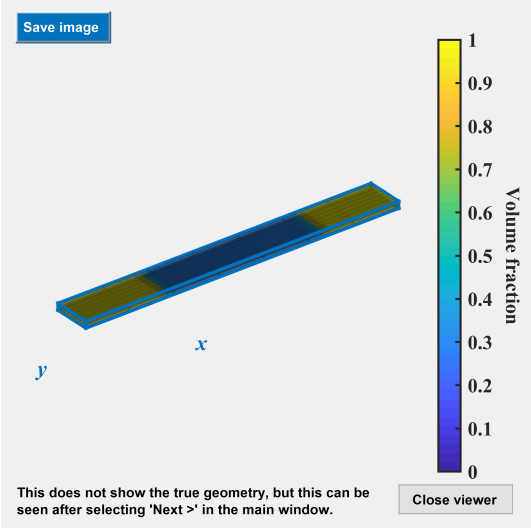


Figure B.27: FLatt Pack GUI - Volume fraction preview (custom volume fraction map).

Proceeding from the 'Volume fraction' window the user is given another opportunity to inspect the volume fraction distribution. The 'Volume fraction and geometry preview' window (figure B.28) again shows the volume fraction distribution in the lattice design space, but this time it is modified by the true lattice geometry, giving the user a clearer impression of their choices at this stage in the lattice design process. Note that figures B.27 and B.28 show the same volume fraction distribution, but the latter also reflects the geometry of the component, in this case a tensile test specimen with solid end pieces.

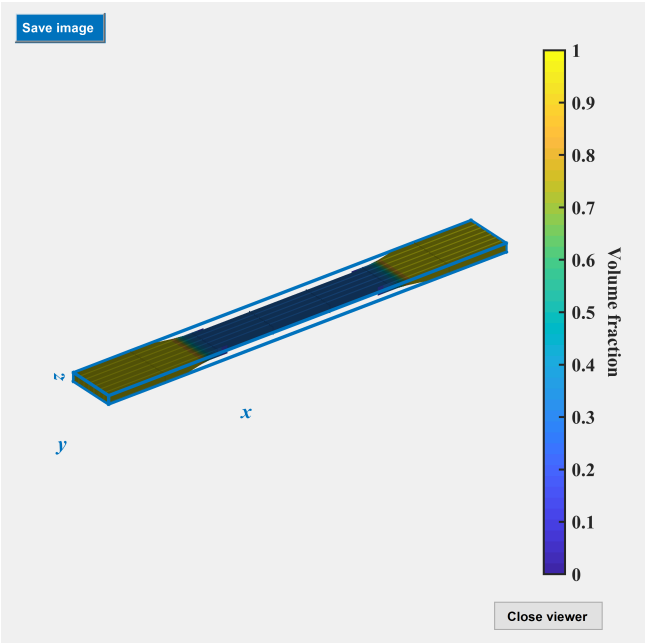


Figure B.28: FLatt Pack GUI - Volume fraction and geometry preview.

Appendix B.7. Solid skin inclusion

The 'Lattice skin' window (figure B.29) provides three skin options. Selecting between them changes the figure on the right of the window, which illustrates their effect on the lattice design. The 'No skin'

option will leave the lattice structure unmodified. Depending on the chosen geometry and cell type, this may lead to exposed cell walls or struts which could be vulnerable to damage. If the user's objective is to determine some bulk property of a lattice (e.g., mechanical, thermal or acoustic performance) through manufacture and testing or simulation, it is recommended not to include an exterior skin, because the skin tends to alter the performance through additional mass, material connectivity and stiffness. However, it must be acknowledged in such studies that a lattice without a solid exterior skin will also exhibit edge effects where the periodicity and connectivity of the structure is terminated. For this reason, it is important to use appropriate boundary conditions in simulations and ensure sufficient numbers of repeating cells to approximate a homogeneous material distribution (as we briefly covered previously [31]).

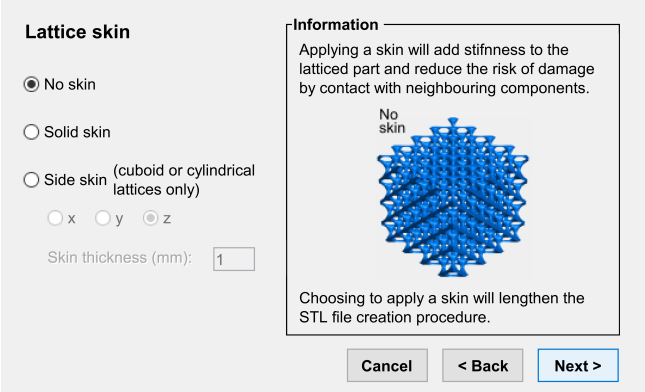


Figure B.29: FLatt Pack GUI - Lattice skin.

The 'Solid skin' option is used to completely enclose the lattice with a conformal skin, which provides protection for the cell walls or struts as well as additional structural stiffness. The skin thickness is designated in an editable text box on the left of the window. The user must note that the inclusion of a solid skin will lead to completely enclosed voids. Such structures are not recommended for some AM processes, such as powder bed fusion and vat polymerisation, because they do not permit the removal of feedstock (powder, photopolymer resin, etc.) after manufacture. AM processes such as material extrusion and jetting do not have the same manufacturing restriction.

Lastly, the 'Side skin' option allows the user to add a solid skin to one opposing pair of faces of a cuboid lattice (e.g., the top and bottom faces), or to the ends or circumference of a cylindrical lattice. This option exists principally for the design of compressive test specimens, which are commonly used for the determination of structure-property relationships for AM lattices. For these specimens, it is sometimes preferable to apply loading to solid, flat surfaces rather than directly to the cell walls or struts.

Appendix B.8. Filename and output file selection

Following the addition or omission of a lattice skin, the user proceeds to the 'Output files' window (figure B.30). Using the 'Save As . . . ' button in the lower left of the window, the user can specify the filename and folder for their files. The default filename consists of the user's lattice design choices; the class, cell type, dimensions, volume fraction, etc., for convenient referencing.

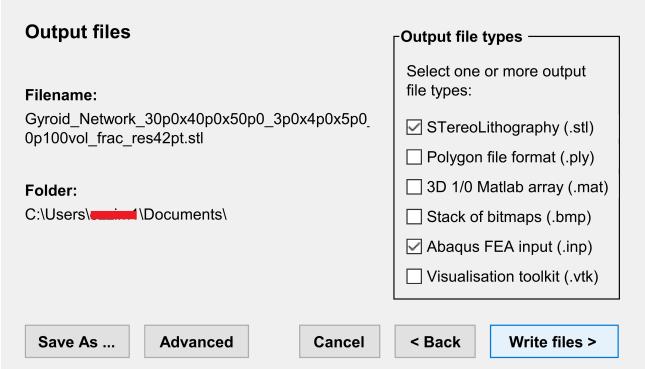


Figure B.30: FLatt Pack GUI - Output files.

On the right of the 'Output files' window are checkboxes corresponding to a range of output file types. These are:

- 'STereoLithography (.stl)' - The format used extensively throughout AM and 3D printing. The binary version of the STL file, rather than the ASCII, is generated.
- 'Polygon file format (.ply)' - Like the STL, this provides a boundary representation of the 3D geometry, but with smaller file sizes. The binary version is generated.
- '3D 1/0 Matlab array (.mat)' - A Matlab logical array. The solid and void domains of the lattice are represented by '1' and '0' elements, respectively.
- 'Stack of bitmaps (.bmp)' - A folder containing a set of bitmap images. These are 'slices' of the lattice structure in the xy plane, with each image filename appended with a number according to its position in the z direction. In each bitmap image, the solid and void domains of the lattice are designated with black and white pixels, respectively.
- 'Abaqus FEA input (.inp)' - A hexahedral finite element mesh composed of hexahedral elements written in the Abaqus input file format. More information on this output option is given in Appendix B.10.
- 'Visualisation toolkit (.vtk)' - A structured dataset representing the lattice with VTK's XML syntax. The solid and void domains of the lattice are designated with '1' and '0', respectively.

Both the STL and PLY output formats are boundary representations. These are meshes of connected triangles marking the boundary between the solid lattice domain and the surrounding void space. The other output formats (MAT, BMP, INP and VTK) are based on a binarised version of the 3D surface array; elements or pixels in these files directly represent solid or void regions of the discretised lattice desin space. The 'Advanced' button in the lower left of the 'Output files' window allows the user to specify the element or pixel size in each of the MAT, BMP, INP and VTK output formats.

Appendix B.9. Triangle mesh reduction

If the user chooses to export an STL or PLY file, they will next be presented with the 'Triangle mesh reduction' window (figure B.31). Using the slider or editable text box, the user specifies the degree of triangle reduction applied to the boundary mesh, which can significantly reduce the size of the output file. A value of 100% indicates that no reduction should be applied; the output mesh will comprise a relatively large number of small triangles. Values lower than 100% apply triangle reduction by replacing groups of small triangles with fewer larger ones. An 'Information' box on the right of the window provides notes to help the user make their choice.

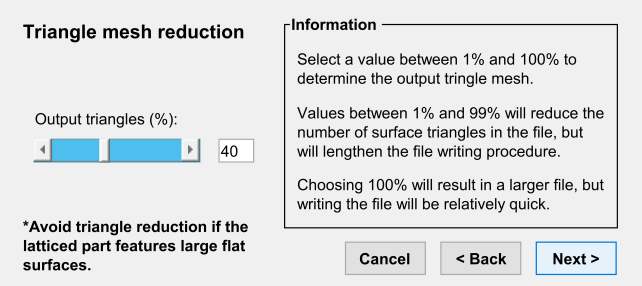


Figure B.31: FLatt Pack GUI - Triangle mesh reduction.

Appendix B.10. FEA input file selection

If the user chooses to export an Abaqus FEA input (.inp) file, they will next encounter the 'Abaqus FEA job creation' window (figure B.32). Radio buttons on the left allow the user to select between exporting the just the hexahedral mesh or an analysis input file, which additionally contains material properties, boundary conditions and a history output request.

The first option exports the lattice geometry as a 'Hexahedral mesh only', where the output file contains just the node coordinates and element information. The second option, 'Displacement load', is used to define a simulation with displacement loading. The elements are assigned a linear elastic material model, with the elastic modulus specified by the user on the right of the window. The elements are of type *C3D8R* (3D eight-node linear isoparametric element with reduced integration). The boundary conditions include a tensile or compressive displacement applied at the nodes of one side of the lattice structure, chosen using the 'Load direction' x, y or z radio buttons, while the nodes of the other side are fixed with either the *XSYMM*, *YSYMM* or *ZSYMM* condition. The magnitude of the applied displacement is specified as a percentage of the lattice structure size along the chosen loading direction. For example, if the user specifies a 2% magnitude for the displacement and the lattice has dimensions $40 \times 40 \times 40$ mm, the applied displacement will be 0.8 mm. The 'Displacement load' Abaqus FEA job is suited to simulating a compression or tension test, which can be used to determine the stiffness of the lattice structure and examine the stress distribution in the lattice walls or struts.

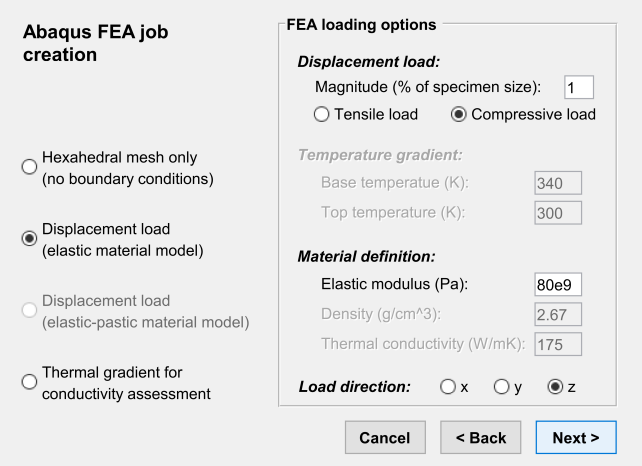


Figure B.32: FLatt Pack GUI - Abaqus FEA job creation.

The third option, 'Thermal gradient for conductivity assessment', is used to define a steady-state heat transfer analysis. The temperatures of two opposing sides of the lattice (again, in x, y or z) are set by the user, as are the relevant material properties; mass density (in g/cm^3) and thermal conductivity (in W/mK). The elements are of type *DC3D8* (similar to *C3D8R*, but used for heat transfer and without

reduced integration). This Abaqus FEA job allows the user to examine the local heat flux and temperature distributions resulting from the thermal gradient across the lattice.

Appendix B.11. Lattice preview and finish screen

If the user chooses to export an STL or PLY file, they are given the option to inspect a 3D preview of the entire lattice structure before the file is exported. This is presented in the 'Lattice preview' subwindow (figure B.33), along with some information about the structure, including a note of the lattice design choices (cell type, dimensions, etc.) as well as the volume and surface area. This preview and structural information are intended to give the user a final opportunity to revise their design choices if the lattice does not correspond to their initial concept. This is done by closing the 'Lattice preview' subwindow and returning to the relevant section of the FLatt Pack GUI with the '< Back' buttons in each window.

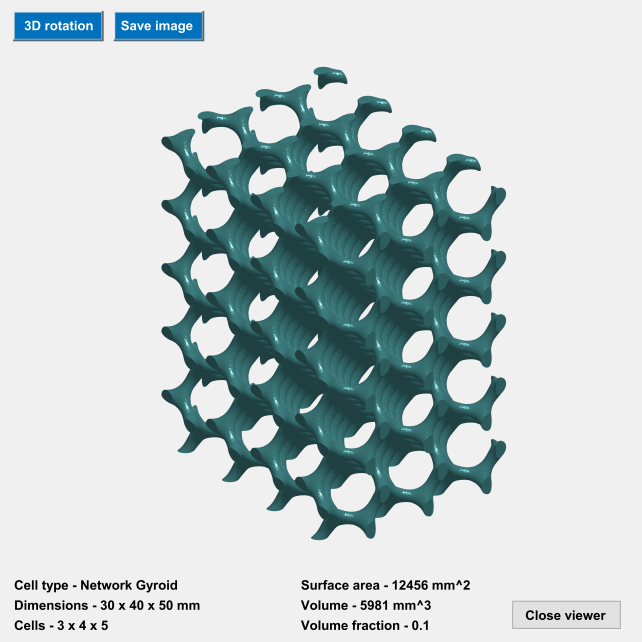


Figure B.33: FLatt Pack GUI - Lattice preview.

Once the user's specified files are saved, they are shown the 'File creation complete' window (figure B.34), which simply allows them to close FLatt Pack with the 'Finish' button or restart the lattice design process to create another structure.

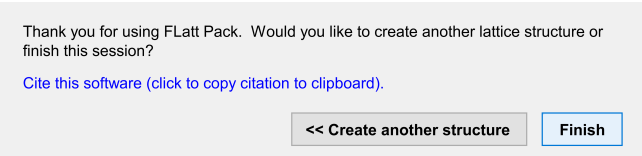


Figure B.34: FLatt Pack GUI - File creation complete.