Analysis of Al6061, Ti-6Al-4V, and SS316L with varying lattice for a stator housing design leveraging generative design engineering

Submitted by:

Adam Michalak

AMFG 521 – Design for Additive Manufacturing

May 11, 2022

Executive Summary

A scaled down model of a stator housing was modeled as a unit cell to evaluate material and lattice type for guiding the design decision making process. Three materials were evaluated, Stainless Steel (316L), Titanium (Ti-6AL-4V), and Aluminum (6061) and for each material six lattice types were evaluated on the unit cell geometry. For each material type the highest performing lattice type was SplitP. The best performing material was Aluminum 6061. The SplitP Aluminum 6061 model is recommended to be studied further as it had the strongest thermal performance with a minimum temperature of 347 Celsius and relatively low mass of 111.3 grams. Further verification and Validation is required before placing this lattice type and material into production for the stator housing.

Introduction

Heat transfer and lattice structure analysis of electric motors can require extensive verification and validation depending on the application and end-user of the motor. Simulating heat transfer on a large part or assembly can be time consuming and computationally expensive. To give quick feedback to a designer or design team, models are often simplified to help guide decisions and allocate time to areas which show promising results. Once initial results have been identified further verification and validation of both the simplified and full-scale model can take place.

In this study an electric motor stator housing has been simplified down to a design domain of a small unit cell of 50 mm^3. This reduction in size was made to provide quick verification of various materials and lattice cell types. Another reason the cell domain is small is that this study utilized Generative Design Engineering (GDE). By utilizing GDE many lattice types and materials can be studied quickly if the computational time is low for each lattice type. The intent of this stator housing is to be additively manufactured due to the housings complex geometry which is explored in this analysis. This study has identified the best material and best lattice type so that a design team can provide further verification and validation before moving forward with an additive manufacturing (AM) process and placing the stator housing into service. The materials that were studied include Aluminum (Al6061), Titanium (Ti-6Al-4V), and Stainless Steel (SS316L). The different lattice types that were studied are Gyroid, Schwarz, Diamond, Lidinoid, SplitP, and Neovius.

Methods

This study utilized nTopology Design and Simulation Software version 3.23. Another software that was utilized was MathWork's MATLAB R2022A. The geometry of the unit cell was provided by Dr. Anthony Petrella from the Colorado School of Mine's Advanced Manufacturing program. The unit cell is comprised of two parts a base and a fin domain. The base is 50 mm wide by 50 mm depth by 5 mm in height. The second part is the fin domain which is a cube that is 50 mm^3. Both parts can be seen below in Figure 1. The fin domain was the area that the various lattice types were applied to. The base represents the stator housing and has a temperature of 673K (400C) applied to it.

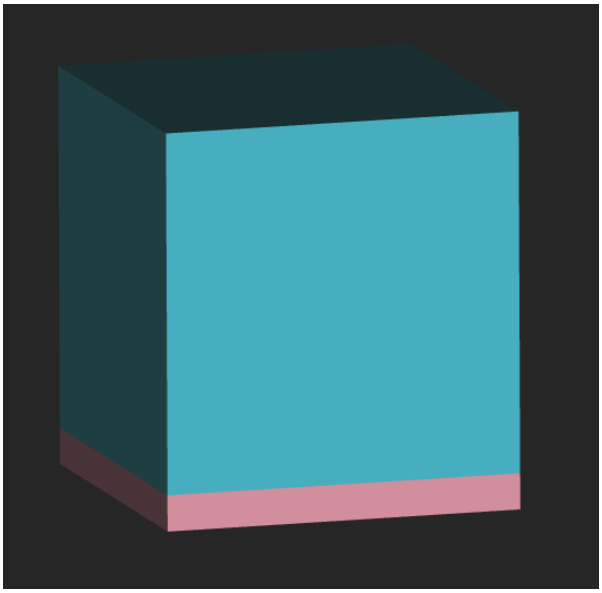

Fig. 1. Fin domain in blue with the base in red.

Generative Design Engineering (GDE) was utilized to manipulate the lattice cell type to make this design exploration study as efficient as possible. Different nTopology (nTop) models were also set up along with respective MATLAB scripts to manipulate and output the three different metals that were studied. The window's command prompt was used to create a input template and output template using nTop's command language prior to utilizing the MATLAB script. Table 1 below shows the different nTop file names and MATLAB script names for each material type. Relevant material properties for each of these materials can be found in the appendix of this report.

Table 1. File names for the nTopology and MATLAB files

	Aluminum (Al6061)	**Stainless Steel (SS316L)**	**Titanium (Ti-6Al-4V)**
nTopology File Name	gde_6061.ntop	gde_316.ntop	gde_ti6.ntop
MATLAB File Name	gde_6061.m	gde_316.m	gde_ti6.m

The nTop software was set up with four inputs as shown in Figure 2 below. These inputs are the following: cell_type, cell_size, thickness, and path. For this GDE study just the cell_type was manipulated using a MATLAB script. The material type and its linear elastic properties, density, and thermal properties were input into nTop as well, special attention had to be paid to density as it was used in multiple objects. The output for the model was a group of seven parameters as seen in Figure 3.

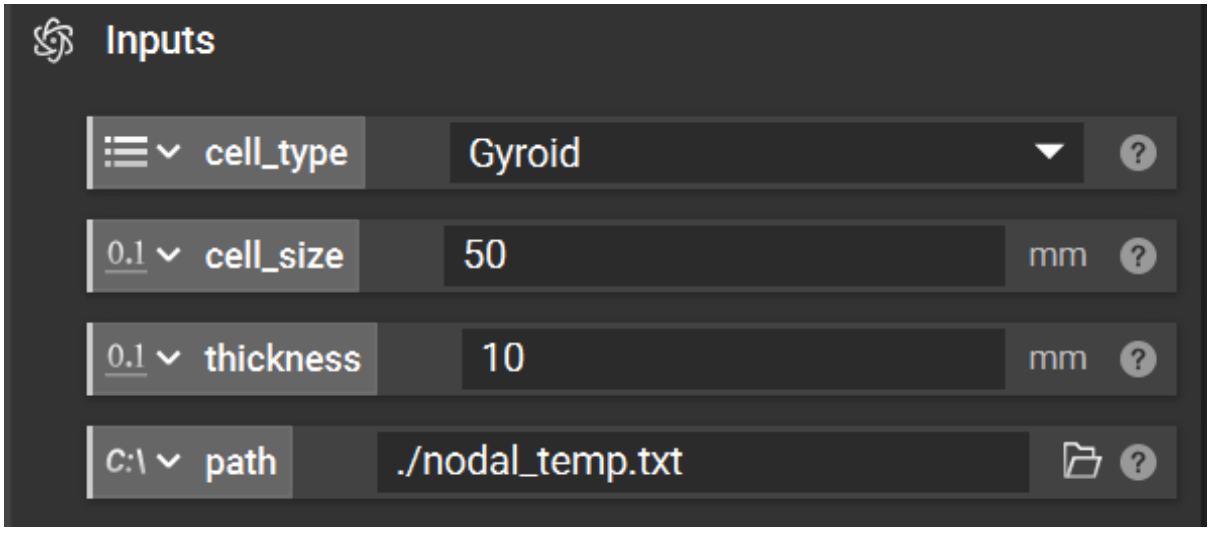

Fig. 2. nTopology inputs for GDE with only the cell_type being manipulated for this study

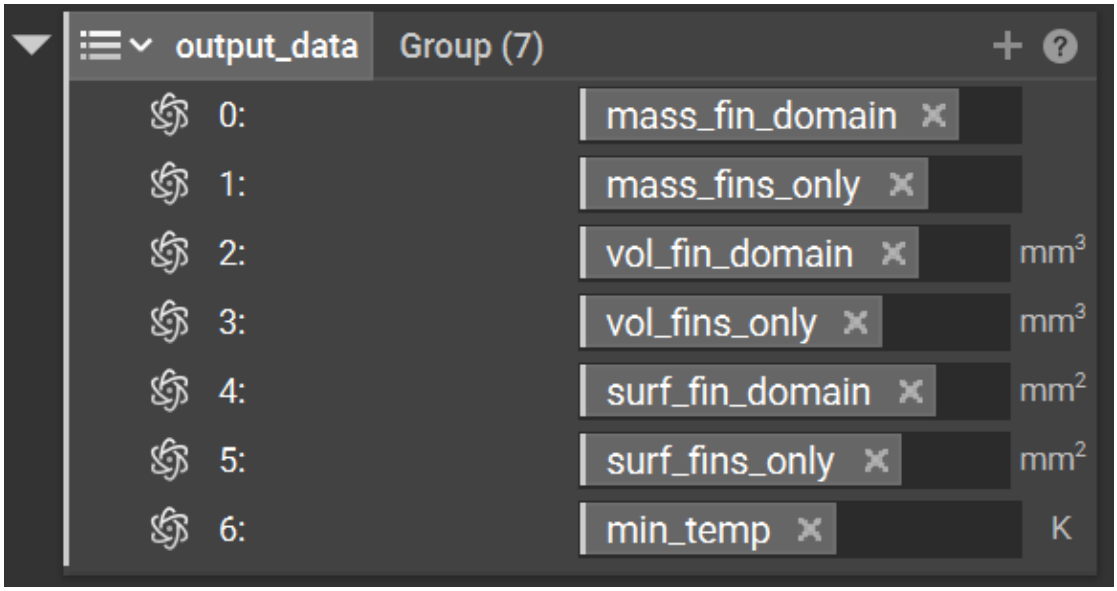

Fig. 3. nTopology output group for GDE

Although the nTopology trees varied slightly based on material type, an example of the nTop tree structure can be seen in Figure 4 below. The foundation for the nTopology file and MATLAB file were provided by Colorado School of Mines.

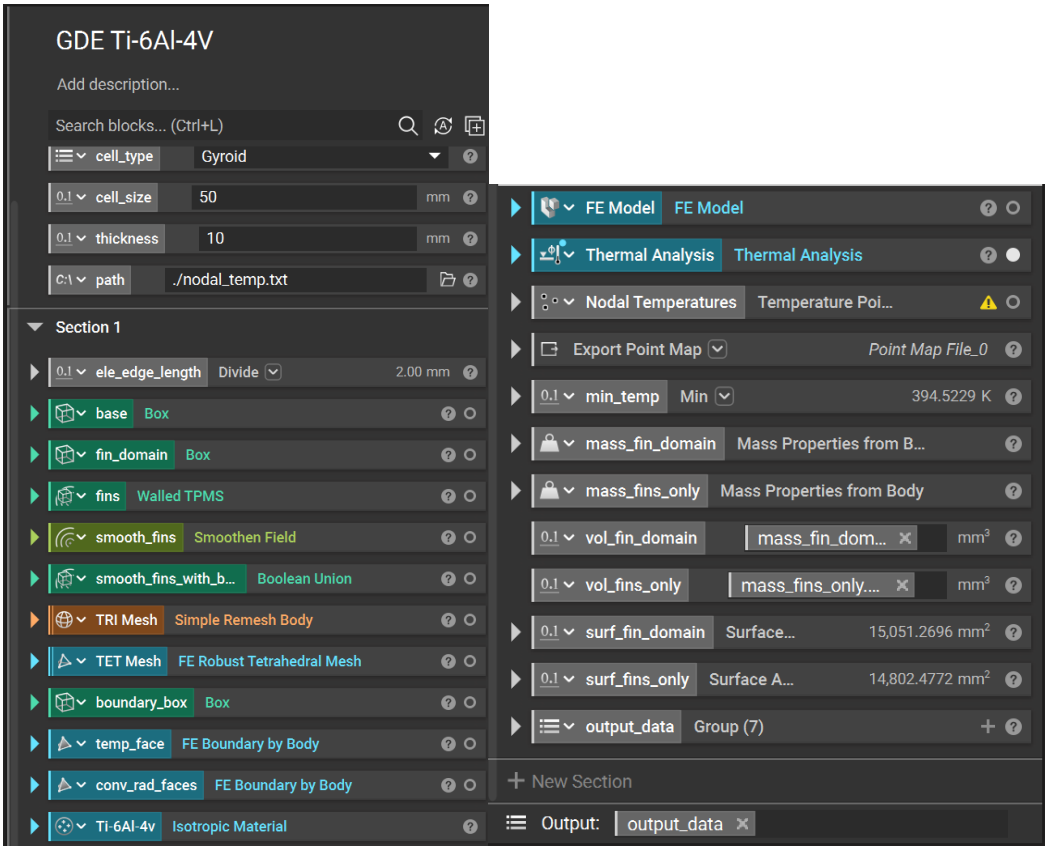

Fig. 4. The complete nTop tree with inputs, an output, and objects required to build and simulate the model

The MATLAB script was setup to output data in a text file and a MATLAB script was ran for each material type that was analyzed. The three output data files' names can be seen in Table 2 below. The output data for each analysis will be in the appendix of this report as well as one of the scripts used for this analysis.

Table 2. Output data file names from running the MATLAB scripts for each material

	Aluminum (Al6061)	**Stainless Steel (SS316L)**	**Titanium (Ti-6Al-4V)**
File Name	outputdata_6061.txt	outputdata_316.txt	outputdata_ti6.txt

Six different lattice cell types were explored using the MATLAB script and referred to by their integer and manipulated using nToplogy's command language 'ntopcl'. The following lattice cell types were analyzed for each material and had an assigned value of 0-5 respectively, Gyroid, Schwarz, Diamond, Lidinoid, SplitP, Neovius. These lattice cell types will be further analyzed in the results section of this study. The thickness of the lattice was chosen to be 10mm. There was a Continuous Boolean Union put in place to smooth the topology from the base of the unit cell to the fin domain.

A tetrahedral mesh with a quadratic geometric order was applied to each of the different lattice cell types with varying nodes and elements. A mesh convergence study was not performed for this study, but it is recommended that if a material and lattice cell type is chosen to move forward that a mesh convergence study is performed. The mesh will be further analyzed in the results section of this report.

Results

The first results that will be presented are the mesh results for each of the different lattice types. Below in Figures 5 – 10 there will be each of the lattice types with the mesh plotted on the geometry. In Table 3 below, the information regarding each lattice type's mesh will be presented which is independent of the material type.

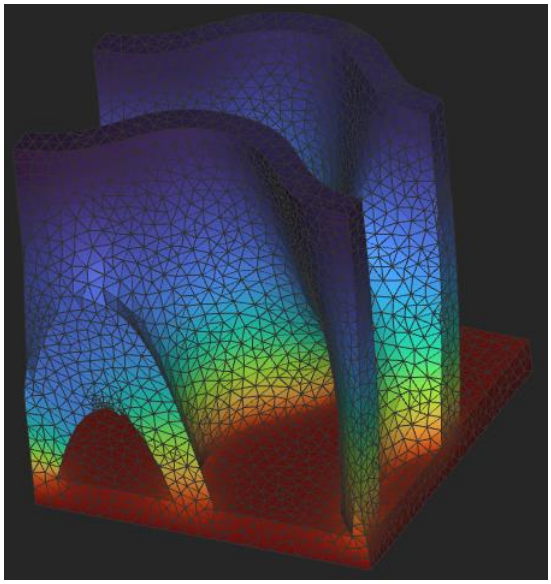

Fig. 5. Gyroid lattice cell type shown with a mesh and a color contour plotted representing temperature

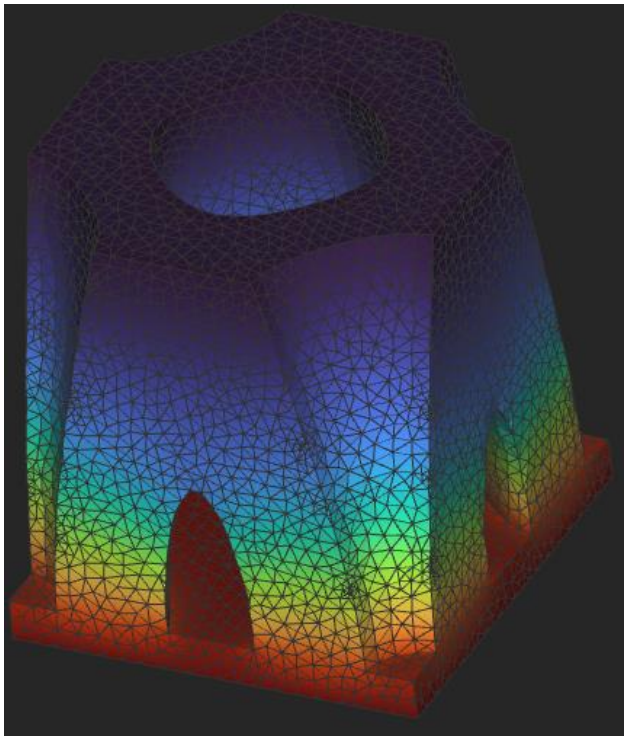

Fig. 6. Schwarz lattice cell type shown with a mesh and a color contour plotted representing temperature

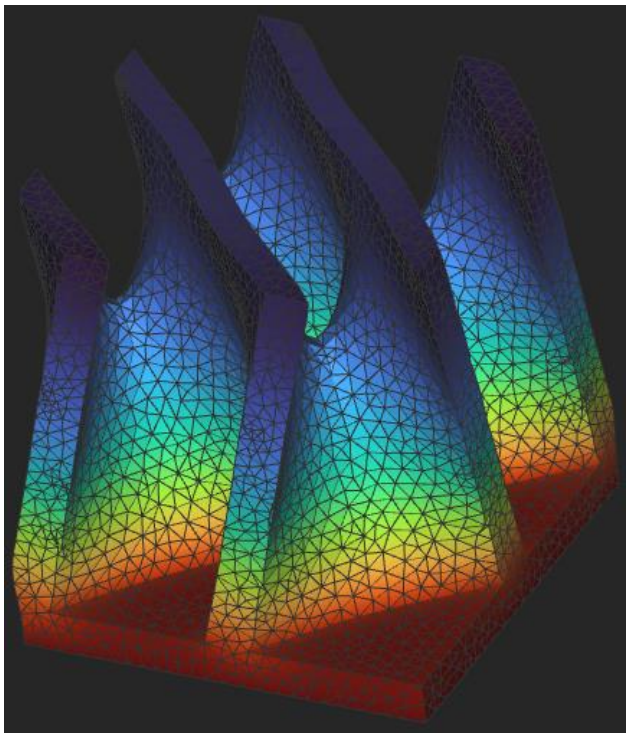

Fig. 7. Diamond lattice cell type shown with a mesh and a color contour plotted representing temperature

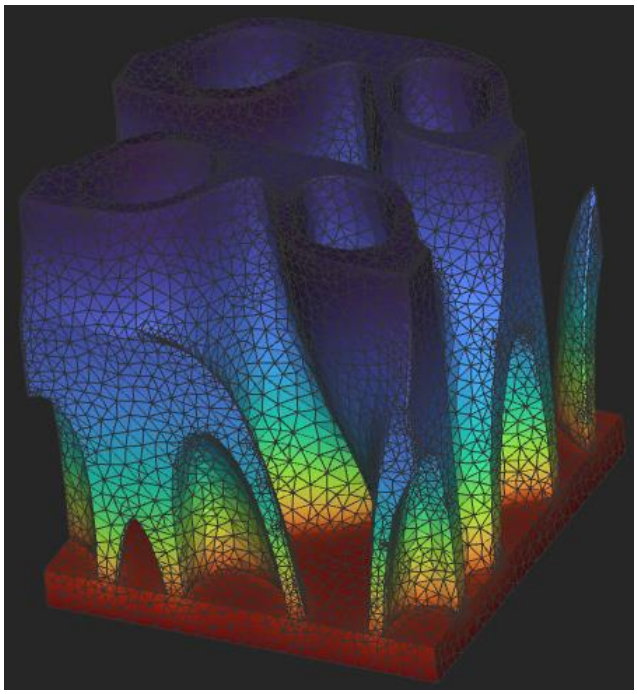

Fig. 8. Lidinoid lattice cell type shown with a mesh and a color contour plotted representing temperature

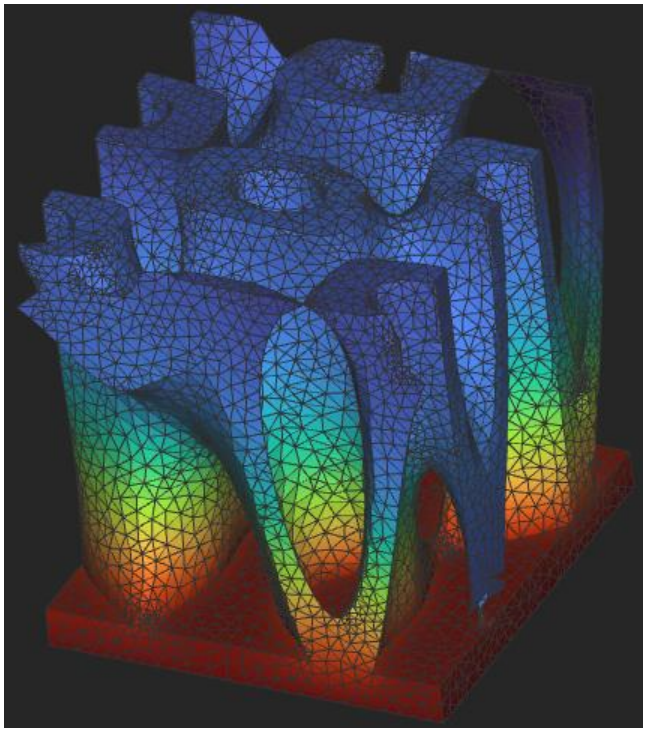

Fig. 9. Split P lattice cell type shown with a mesh and a color contour plotted representing temperature

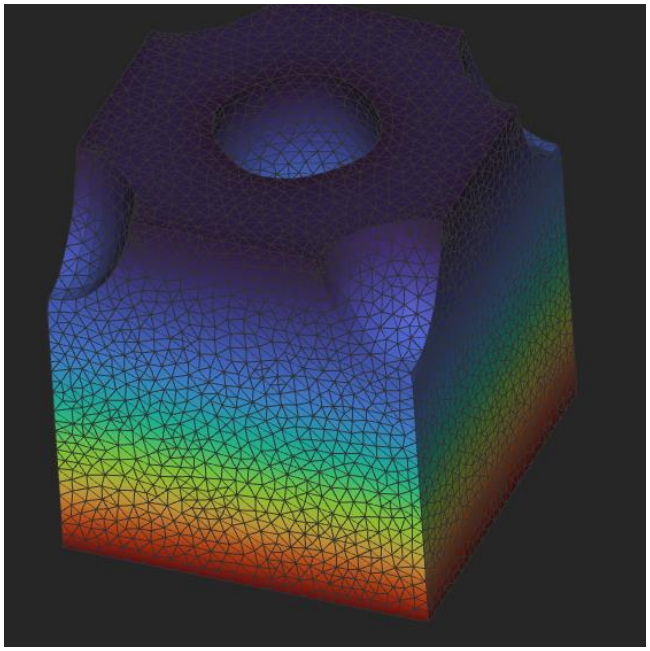

Fig. 10. Neovius lattice cell type shown with a mesh and a color contour plotted representing temperature

Table 3. Mesh information for each lattice cell type

Lattice Cell Type	Number of Nodes	Number of Elements
Gyroid	90610	57857
Schwarz	133544	89043
Diamond	103323	65651
Lidinoid	170762	108167
Split P	132690	81896
Neovius	180214	124419

The following results are for each material type. These results were output from the three MATLAB scripts to the specific output files as described in the methods section of this report. The mass of the fin domain, volume of the fin domain, and surface area of the fin domain are the same for each lattice cell type as this is for the $50mm^3$ domain for the unit cell. The mass of the lattice cell type and minimum temperature will be presented in Tables 4, 5, and 6 along with other results for each material and their respective lattice type.

Table 4. Output results for Aluminum (Al6061)

Lattice Cell Type	Mass of Fin Domain (g)	Mass of Lattice Cell Type (g)	Volume of Fin Domain (mm^3)	Volume of Lattice (mm^3)	Surface Area of Fin Domain (mm^2)	Surface Area of Lattice (mm^2)	Minimum Temperature (C)
Gyroid	337.500	82.045	125000.000	30387.156	15051.270	14802.477	371.263
Schwarz	337.500	148.629	125000.000	55047.653	15051.270	16063.792	380.708
Diamond	337.500	98.333	125000.000	36419.811	15051.270	17838.167	369.818
Lidinoid	337.500	143.323	125000.000	53082.594	15051.270	23614.154	373.502
Split P	337.500	111.327	125000.000	41232.172	15051.270	23495.644	347.134
Neovius	337.500	234.056	125000.000	86687.321	15051.270	17770.512	388.039

Table 5. Output results for Stainless Steel (SS316L)

Lattice Cell Type	Mass of Fin Domain (g)	Mass of Lattice Cell Type (g)	Volume of Fin Domain (mm^3)	Volume of Lattice (mm^3)	Surface Area of Fin Domain (mm^2)	Surface Area of Lattice (mm^2)	Minimum Temperature (C)
Gyroid	1000.000	243.097	125000.000	30387.156	15051.270	14802.477	229.214
Schwarz	1000.000	440.381	125000.000	55047.653	15051.270	16063.792	272.019
Diamond	1000.000	291.358	125000.000	36419.811	15051.270	17838.167	223.005
Lidinoid	1000.000	424.661	125000.000	53082.594	15051.270	23624.072	238.283
Split P	1000.000	329.857	125000.000	41232.172	15051.270	23495.644	152.199
Neovius	1000.000	693.499	125000.000	86687.321	15051.270	17770.512	311.556

Table 6. Output results for Titanium (Ti-6Al-4V)

Lattice Cell Type	Mass of Fin Domain (g)	Mass of Lattice Cell Type (g)	Volume of Fin Domain (mm^3)	Volume of Lattice (mm^3)	Surface Area of Fin Domain (mm^2)	Surface Area of Lattice (mm^2)	Minimum Temperature (C)
Gyroid	550.000	133.703	125000.000	30387.156	15051.270	14802.477	121.523
Schwarz	550.000	242.210	125000.000	55047.653	15051.270	16063.792	168.044
Diamond	550.000	160.247	125000.000	36419.811	15051.270	17838.167	115.141
Lidinoid	550.000	233.563	125000.000	53082.594	15051.270	23618.664	129.951
Split P	550.000	181.422	125000.000	41232.172	15051.270	23495.644	62.820
Neovius	550.000	381.424	125000.000	86687.321	15051.270	17770.512	219.203

In Figures 11-13 a 4D plot for each material type is shown with the fourth dimension being color which is related to surface area of the lattice, these are the results from Tables 4-6 visualized.

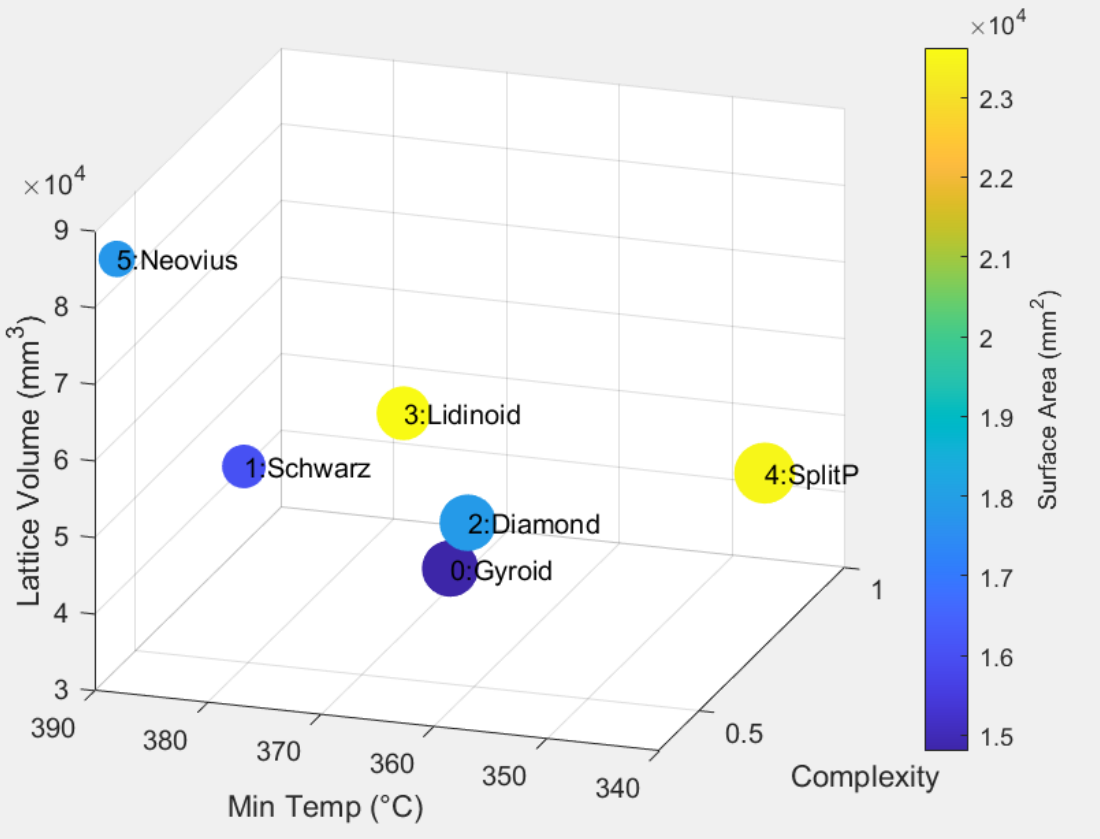

Fig. 11. 4D Plot for Aluminum (Al6061)

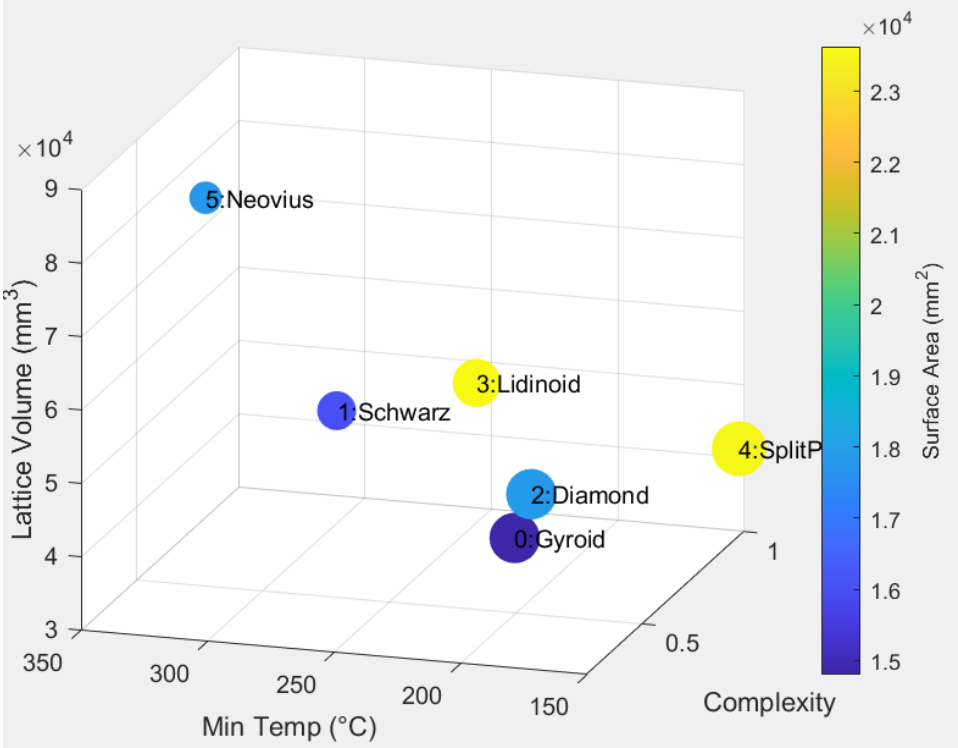

Fig. 12. 4D Plot for Stainless Steel (SS316L)

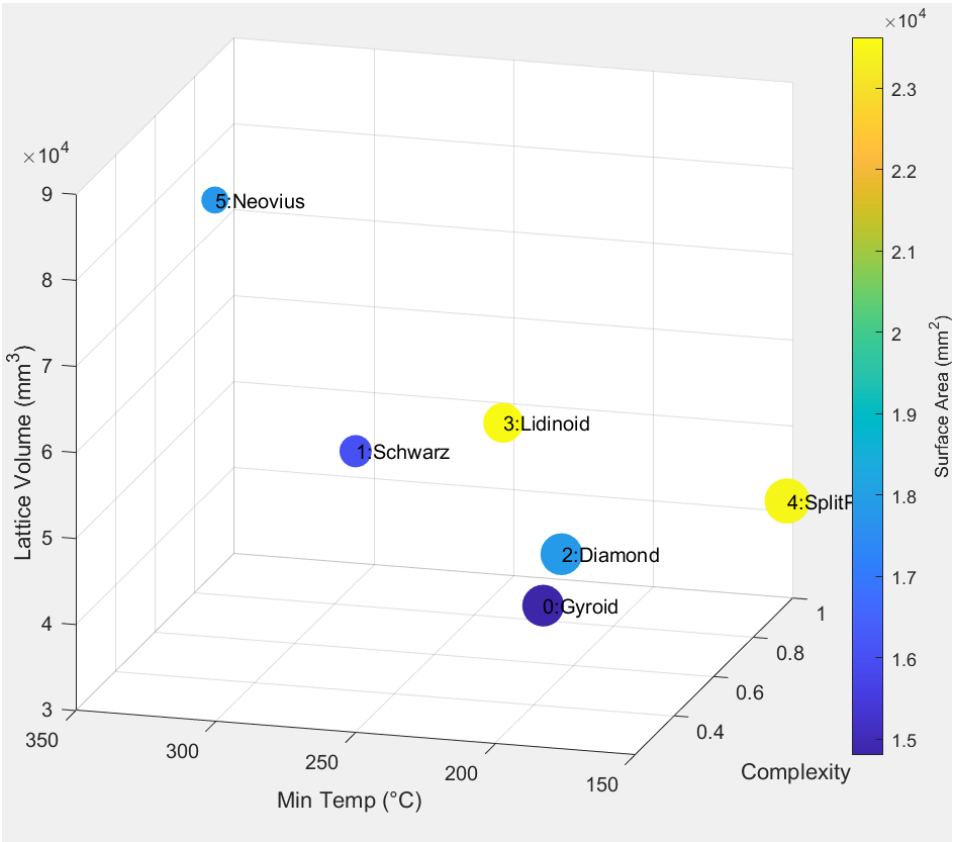

Fig. 13. 4D Plot for Titanium (Ti-6Al-4V)

Figure 14-16 are nodal temperature scatter plots, however, because there are so many data points, they resemble a color contour plot.

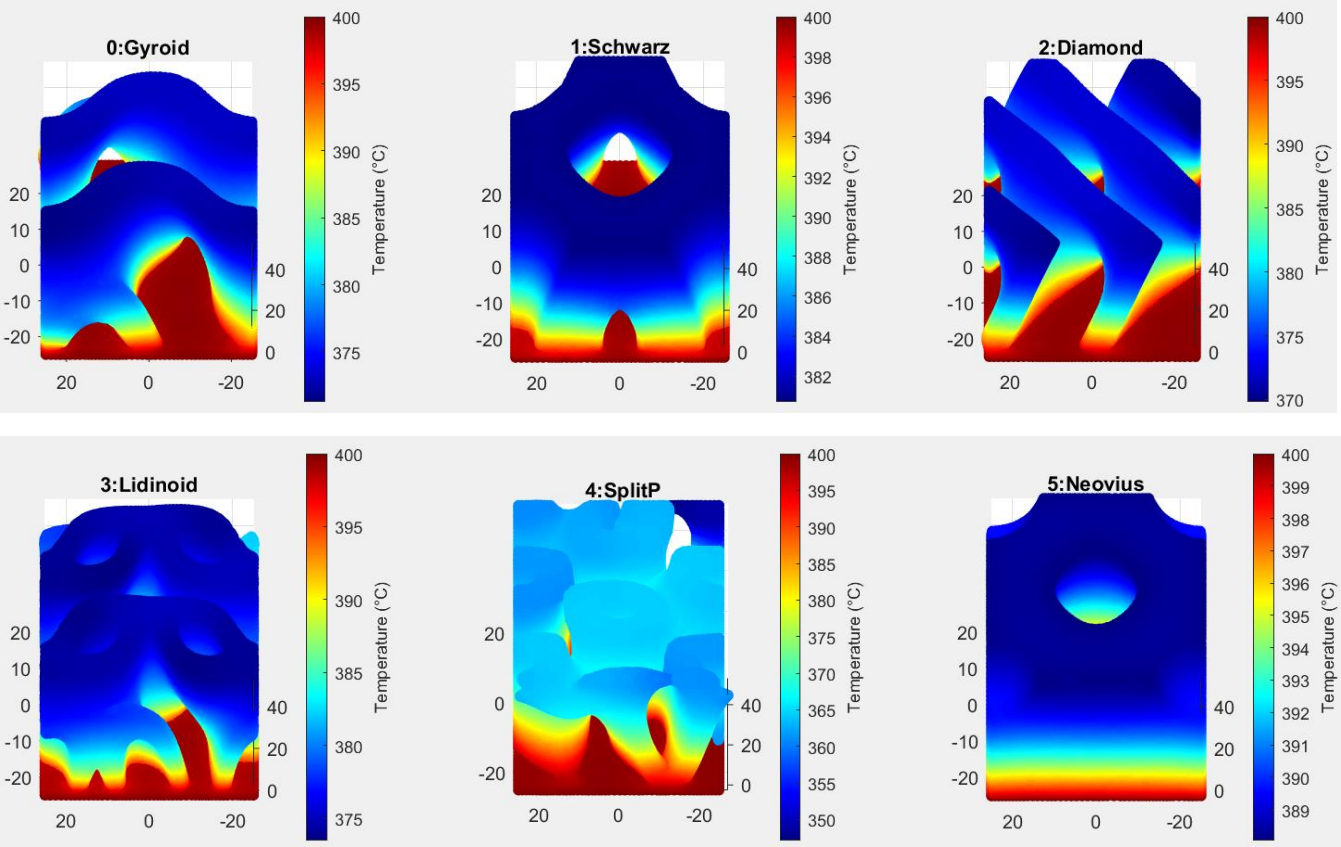

Fig. 14. Scatter plot of Aluminum (Al6061) lattice types

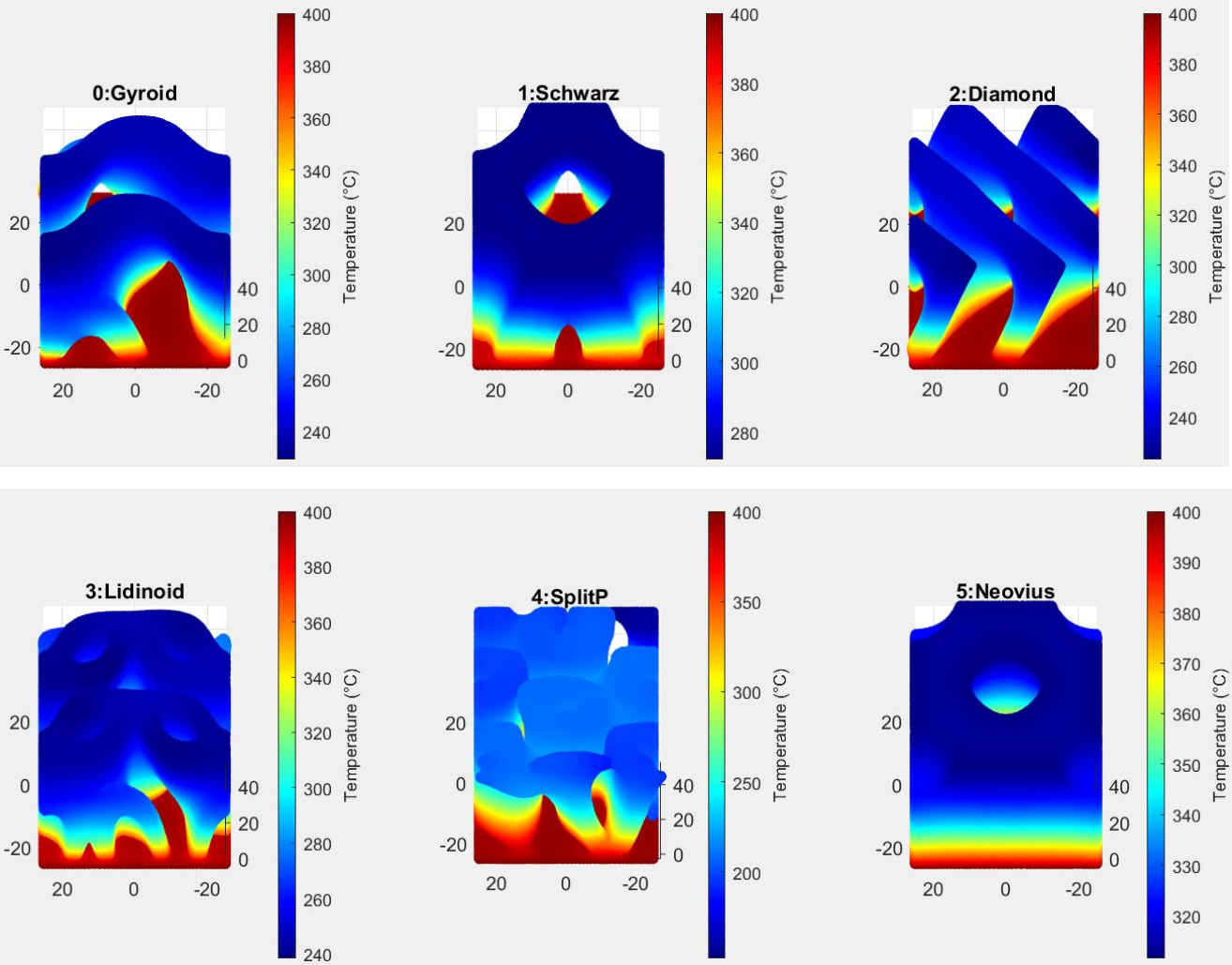

Fig. 15. Scatter plot of Stainless Steel (SS316L) lattice types

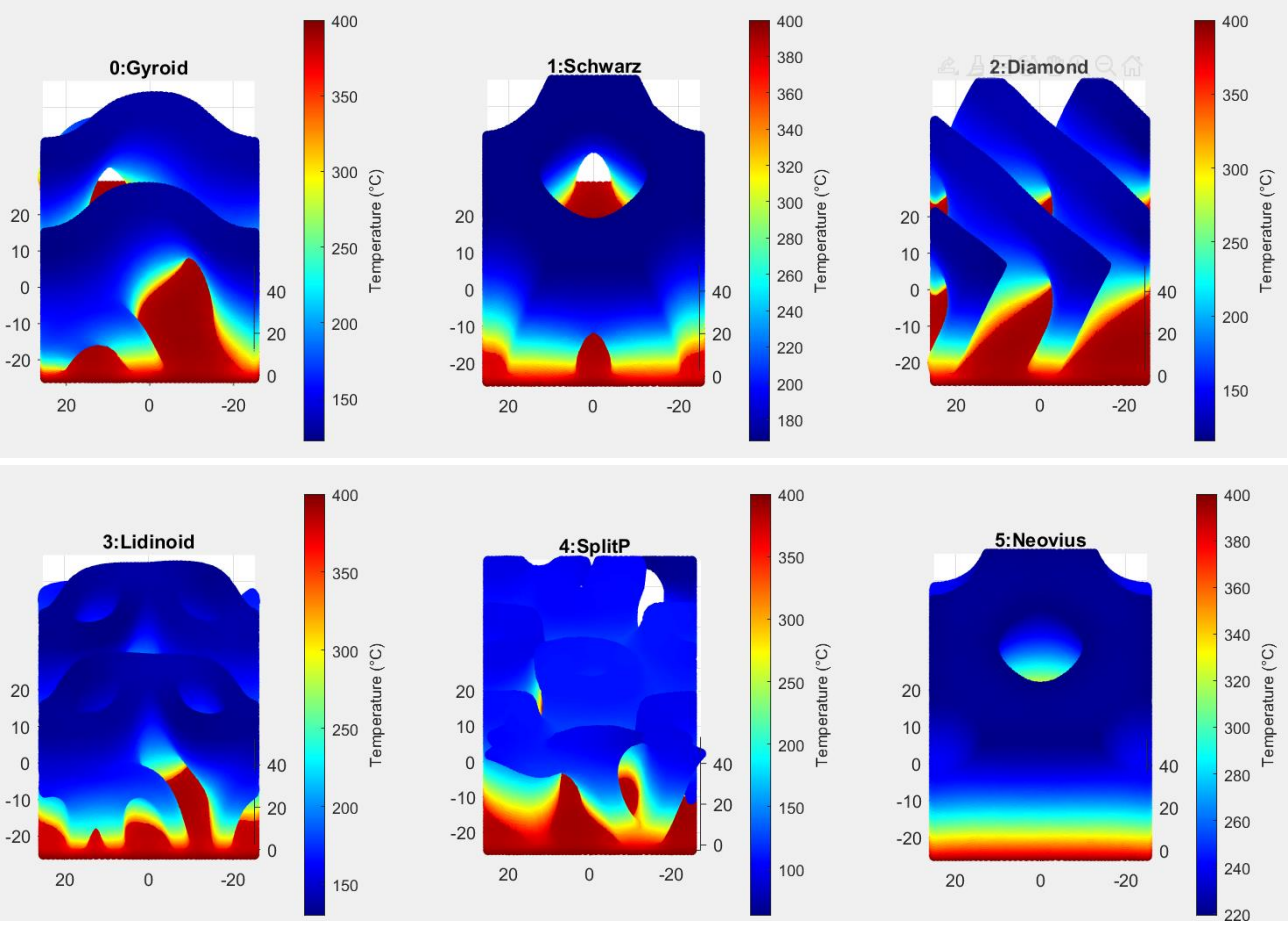

Fig. 16. Scatter plot of Titanium (Ti-6Al-4V) lattice types

Discussion

The purpose of this analysis was to explore various lattice types and materials using GDE to guide a design decision on which material and lattice type would be best to focus design and simulation time on. The volume of the fin domain and surface area of the fin domain are the same for each lattice and material type. The mass for each lattice type and material varies with Aluminum strongest performance in the mass category. For all materials SplitP was the best lattice structure in terms of thermal performance because of the low minimum temperature it was able to achieve which indicates it would perform well as a heat exchanger. The best performing material in terms of thermal performance was Aluminum which had the highest minimum temperatures for each lattice type which is indicative of Aluminum's thermal conductivity performance when compared to Stainless Steel and Titanium. As expected, the mass of the Aluminum lattice types were also the lowest when compared to Stainless Steel and Titanium. The lattice type with the lowest mass was Gyroid for each material.

The best choice for choosing a strong performing material and lattice type for a stator housing is dependent on the application. For this application we will focus on the lightest mass and best thermal performance which would be the SplitP lattice type made from Aluminum, this material and lattice offers a relatively low mass and strong thermal performance as a heat exchanger due to its ability to conductively transfer heat. One downside is the complexity of the lattice type as it may pose difficulties if it was chosen to be used in a regulated industry. The reason this lattice

type would be difficult is because the geometry would be hard to inspect and certify which ties in with the validation of the part.

Overall, it is recommended that further verification and validation is carried out if a lattice and material type is chosen. For example, one short fall of this study is that convective heat transfer for each lattice type was not studied. Another reason further verification should be carried out before moving forward with a prototype is that a mesh convergence study was not performed. The design team should feel confident that Aluminum is a strong material choice but more specific attention should be paid to the lattice type for reasons previously mentioned.

References

1. *The Online Materials Information Resource*. MatWeb. (n.d.). Retrieved May 11, 2022, from https://www.matweb.com/search/DataSheet.aspx

Appendix

Material Properties

	Density ($mm^{-3}*g$)	Coefficient of Thermal Expansion (K^{-1})	Young's Modulus (Pa)	Poisson's Ratio	Conductivity ($mm*s^{-3}*g*K$)	Specific Heat ($mm^2*s^{-2}*K^{-1}$)
Al6061	0.0027	2.15 e-5	6.89 e+10	0.33	1.67 e+08	8.96 e+08
SS316L	0.008	1.3 e-5	1.93 e+11	0.28	1.63 e+07	5 e+08
Ti-6Al-4V	0.0044	0.7 e-5	1.138 e+11	0.342	6.7 e+06	5.263 e+08

MATLAB Script provided by Colorado School of Mines and edited by the author of this report

```
clc;clear all;close all;

% ntop and ntopcl commaands are in the Windows path by default, so you can
% just type the command without any other path info
exe = 'ntopcl';
% NOTE: all files should be in the same folder... this is the nTop model
nTopFile='gde_ti6.ntop';
% provide an input JSON file... this is generated in a separate operation by
% running ntopcl with the -t flag to create an input template
input_file='input_template.json';
% use the built-in Matlab function to decode the JSON file into a struct
in = jsondecode(fileread(input_file));

% let's look at six different sets of input values using the range 0...5
% for now, we're only changing the lattice unit cell type in this example
for i=0:5
    % modify input variables
    % this is the lattice cell type (Gyroid,Shwarz,etc)
    in(1).inputs{1,1}.value = i;
    % this is the cell size (mm)
    in(1).inputs{2,1}.value = 50.0;
    % this is the TPMS wall thickness (mm)
```

```
    in(1).inputs{3,1}.value = 10.0;
    % this adds an index to the output filename so we can keep many STL
    in(1).inputs{4,1}.value = ['./nodal_temp_',num2str(i),'.txt'];

    % write JSON file format with the updated input values
    InputFile = ['input_',num2str(i),'.json'];
    OutputFile = ['output_',num2str(i),'.json'];
    JsonStr = jsonencode(in);
    JsonStr = strrep(JsonStr, ',', sprintf(',\r'));
    JsonStr = strrep(JsonStr, '[{', sprintf('[\r{\r'));
    JsonStr = strrep(JsonStr, '}]', sprintf('\r}\r]'));
    fid = fopen(InputFile, 'w');
    if fid == -1, error('Cannot create JSON file'); end
    fwrite(fid, JsonStr, 'char');
    fclose(fid);

    % compose a string containing the execution command
    Arguments=sprintf('%s -j %s -o %s %s',exe,InputFile,OutputFile,nTopFile);

    % run the ntopcl command
    system(Arguments);

    % parse the output data in the order mass (g), volume (mm^3), surf_area (mm^2),
minimum temperature (C)
    mass_box(i+1,1) = jsondecode(fileread(OutputFile)).components(1).value.mass*1e3;
% mass box (g)
    mass_fin(i+1,1) = jsondecode(fileread(OutputFile)).components(2).value.mass*1e3;
% mass fin only (g)
    volm_box(i+1,1) = jsondecode(fileread(OutputFile)).components(3).value.val*1e9;
% vol box (mm^3)
    volm_fin(i+1,1) = jsondecode(fileread(OutputFile)).components(4).value.val*1e9;
% vol fin only (mm^3)
    surf_box(i+1,1) = jsondecode(fileread(OutputFile)).components(5).value.val*1e6;
% surf box (mm^2)
    surf_fin(i+1,1) = jsondecode(fileread(OutputFile)).components(6).value.val*1e6;
% surf fin only (mm^2)
    temp_min(i+1,1) = jsondecode(fileread(OutputFile)).components(7).value.val-273;
% min temp fin (C)

end

% assemble the output data in columns following the same order as shown
% above and clearly written below (mass, volume, surf area, printing time)
output_data = [mass_box mass_fin volm_box volm_fin surf_box surf_fin temp_min];
% save all the output data in an tab-delimited text file
save 'outputdata_ti6.txt' output_data -ascii
% ************************************************************************
% FOR PLOTTING Outcome Metrics
% ************************************************************************
% if you choose to run the above analysis with Python, then you just need
% the remaining lines below here to plot the data in Matlab... you can
% highlight all the lines below, right-click, and choose "Evaluate Selection"
% load the output_data.txt file into a data array
data = load('outputdata_ti6.txt');
% save each column of the data array in its appropriate variable
```

```
mass_box = data(:,1); mass_fin = data(:,2); volm_box = data(:,3); volm_fin =
data(:,4);
surf_box = data(:,5); surf_fin = data(:,6); temp_min = data(:,7);
% compute the ratio of surface area to volume for each lattice type
sa_v = surf_fin./volm_fin; % this is the surface area to volume ratio
% normalize the sa_v data so the max value across lattice types is 1.0
sa_v_norm = sa_v/max(sa_v); % this is a normalized sa_v vector (0,1)
% use scatter plot to visualize the data... the first three arguments
% correspond to the x, y, and z axes, the fourth argument controls size of
% each marker based on the value of sa_v_norm, the fifth argument controls
% the color of each marker based on the value of surface area
h = scatter3(sa_v_norm,temp_min,volm_fin,500*sa_v_norm,surf_fin,'filled');
% label the x, y, and z axes on the scatter plot
xlabel('Complexity'); ylabel('Min Temp (°C)'); zlabel('Lattice Volume (mm^3)');
% this makes a color legend to interpret the magnitude of surface area based on color
c = colorbar; c.Label.String = 'Surface Area (mm^2)';
% these lines apply text labels to we can remember which lattice type is which
text(sa_v_norm(1),temp_min(1),volm_fin(1),'0:Gyroid');
text(sa_v_norm(2),temp_min(2),volm_fin(2),'1:Schwarz');
text(sa_v_norm(3),temp_min(3),volm_fin(3),'2:Diamond');
text(sa_v_norm(4),temp_min(4),volm_fin(4),'3:Lidinoid');
text(sa_v_norm(5),temp_min(5),volm_fin(5),'4:SplitP');
text(sa_v_norm(6),temp_min(6),volm_fin(6),'5:Neovius');

% ************************************************************************
% FOR PLOTTING Nodal Temperatures
% ************************************************************************
figure; colormap(jet);
subplot(2,3,1)
t = load('nodal_temp_0.txt');
h = scatter3(t(:,1),t(:,2),t(:,3),[],t(:,4)-273,'filled');
c = colorbar; c.Label.String = 'Temperature (°C)'; axis equal; view(-180,-30);
title('0:Gyroid');
subplot(2,3,2)
t = load('nodal_temp_1.txt');
h = scatter3(t(:,1),t(:,2),t(:,3),[],t(:,4)-273,'filled');
c = colorbar; c.Label.String = 'Temperature (°C)'; axis equal; view(-180,-30);
title('1:Schwarz');
subplot(2,3,3)
t = load('nodal_temp_2.txt');
h = scatter3(t(:,1),t(:,2),t(:,3),[],t(:,4)-273,'filled');
c = colorbar; c.Label.String = 'Temperature (°C)'; axis equal; view(-180,-30);
title('2:Diamond');
subplot(2,3,4)
t = load('nodal_temp_3.txt');
h = scatter3(t(:,1),t(:,2),t(:,3),[],t(:,4)-273,'filled');
c = colorbar; c.Label.String = 'Temperature (°C)'; axis equal; view(-180,-30);
title('3:Lidinoid');
subplot(2,3,5)
t = load('nodal_temp_4.txt');
h = scatter3(t(:,1),t(:,2),t(:,3),[],t(:,4)-273,'filled');
c = colorbar; c.Label.String = 'Temperature (°C)'; axis equal; view(-180,-30);
title('4:SplitP');
subplot(2,3,6)
t = load('nodal_temp_5.txt');
```

```
h = scatter3(t(:,1),t(:,2),t(:,3),[],t(:,4)-273,'filled');
c = colorbar; c.Label.String = 'Temperature (°C)'; axis equal; view(-180,-30);
title('5:Neovius');
```