

Development of a data driven approach to identify the trends in mechanical properties of cellular metals

Charlie Rougier supervised by Stefan Szyniszewski

Abstract—The aim of this research project was to develop a data driven approach to visualise the trends in the mechanical properties of cellular metals. This approach would build upon work conducted by Stefan Szyniszewski and Rodrigo Mamone, in partnership with SpringerMaterials, in creating a database for cellular metal properties. The first stage of this project was to supplement this database. In total 57 entries were added bringing the total from 272 to 329. The project then moved to the analysis stage. A MATLAB script was developed to automate the fetching, processing and post-processing of this data and the tool was used to analyse known and unknown relationships in metal foam properties. Given sufficient data points the tool was shown to be able to closely predict models obtained through experimental research. When it came to identifying new trends the tool showed a lack of trend for pore size and mechanical properties but was limited by a lack of data points for more uncommon properties. The tool also proved to be useful in identifying outliers and errors in the database. Future additions to the tool functionality and approaches to the data collection process have been proposed and the advantages and disadvantages discussed.

Index Terms—Data driven, Cellular metals, metal foams, property trends, MATLAB, SQL.

I. INTRODUCTION

DATA driven science is a new approach to scientific research which promises many opportunities for the field of engineering. The traditional experimental approach to research has varied little for hundreds of years. Put simply, a hypothesis is proposed, experiments are conducted to generate new data on the topic and results are concluded from this data. Data driven approaches break this mould by analysing large sets of pre-existing data, often too complex for human analysis, to uncover new trends and relationships. By avoiding the process of setting up and conducting often expensive and time consuming experiments data driven science presents attractive time and money savings. Another issue with hypothesis-driven research is that even when studies are conducted, there are often focused on identifying particular relationships and may overlook seemingly unrelated ones. By analysing a larger data set for many potential relationships at once, with no particular bias, new scientific relationships can be discovered. In engineering, data driven approaches have been particularly useful for the field of material science where the growing demand for stronger and lighter materials has driven efforts to understand what affects these properties. Granta design was founded in 1994 by Mike Ashby and David Cebon and represents one of the first data driven approach in this field. By plotting material property (or 'Ashby') charts like the one

shown in figure 1 they provided a tool to compare materials based on key properties. Originally developed for academic use they have now expanded into the commercial market and their GRANTA selector tool is used extensively in industry for material selection.

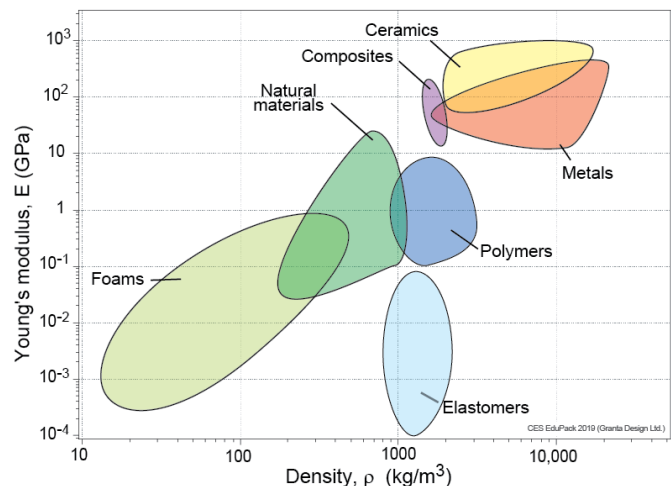


Fig. 1: Example material property chart from GRANTA design

Cellular metals, otherwise known as metal foams are relatively new engineering materials and ideal candidates for data driven research. They are defined simply as solid metals with large volume fractions of gas filled pores and possess a number of mechanical properties such as excellent stiffness-to-weight ratios, vibration damping, high thermal conductivity and high surface area to name a few. Each of these alone are sufficient to make metal foams attractive materials to many industries but the combination of these are where they stand out. For example the combination of their lightweight and kinetic energy absorption properties make them outstanding race car crash absorbers [1], their high bending stiffness-to-weight ratio and vibration damping properties make excellent crane lifting arms [2] and their even stress distribution and improved ability to fuse with bone (due to high surface area) make good titanium implants [3]. As a result there is a high demand to better understand the link between the morphology of cellular metals and their characteristics so they can continue to be optimised.

In 'Metal Foams: A Design Guide' [4], Gibson and Ashby reported basic trends between the porosity and stiffness and a number of other variables based on a limited number of test

data points. Since then metal foams have received a huge and continued scientific attention. This large and growing data set has opened the door for data driven approaches however there are many steps to this. First the data set must be obtained. Data is published in a variety of ways such as industrial data sheets and publications, and in a variety of formats such as tables and graphs and prior to analysis it must be collected and compiled in a database. Next is the processing stage where the data must be standardised to correct differences due to the use of various reporting methods, testing standards and units. Finally in the post processing stage the data is visualised for human consumption, usually in graphs and tables, and conclusions can be made. One of the greatest barriers to entry for data driven research is that usually the processing and post processing stages involve complex manipulation of data, which is often time consuming and usually requires coding knowledge.

The objective of this project is first to supplement an existing database for cellular metals which has been provided by SpringerMaterials and then to develop a tool in MATLAB to automate both the processing and post-processing of data from the database to improve the efficiency and accessibility of data driven research in this field. The aim is then to use the tool to analyse the data and identify new trends in the mechanical properties. Failing this, the tool should provide new insight to the foams and possibly identify outliers or areas of interest which could be investigated further. Compared to the the product offered by Granta Design, this project is focused more specifically on cellular metals and and to provide a open source tool that can be iterated upon as the database evolves.

II. METHODS

A. Data collection

As with all data driven approaches sufficient data is required before meaningful conclusions can be extracted. As a result the first stage of this project was to supplement the existing database provided by Stefan Szyniszewski and SpringerMaterials. This was done by manually searching the existing literature to find papers or journals that had reported relevant properties on new or existing metal foams. Due to the nature of academic research each paper represents a different approach to research into the field and as such there is a high variability in not only the data reported but also the reporting format. Therefore, the data must first be collected and compiled into a standard format on which a script could be run to automate the importing process. In this case the standard format chosen was a google sheet document, also provided by Stefan Szyniszewski, shown in figure 2. Each sheet represented a new metal foam with a different topology, composition or manufacturing method and recorded any mechanical, fluidic or thermal properties reported as well as details about the foam make up and the source of the data. A full screenshot of an example sheet can be found in the appendices. Many papers also reported data in the form of graphs such as stress strain curves. To record this data a plot digitizer tool [5] was used to convert the curves into a series of data points which could then be recorded as shown in figure 3. At the start of the

project there were 271 sheets and over its course 58 were added bringing the total to 329.

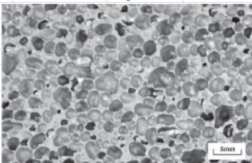
Identifier	0320-AI						Image
Trade name	Alporas						
Base material	Aluminium						
Purity							
Foam type	Closed cell						
Manufacturing Method	batch casting process						
Geometry							
Description	AL-Ca5-Ti3 matrix, closed cell, varying cell size.						
		mean		std			
Porosity		84%					
Bulk density		0.43 g/cm³					
Pores per unit length		pores/cm					
Average pore size		4 mm					
Mechanical							
Elastic properties:		mean		std	Number of measurements	Temperature	
Young modulus		GPa					C
		GPa					C
		GPa					C
		GPa					C
Elastic Poisson ratio							
Plastic properties:		mean		std		Temperature	
Yield stress		MPa					C
		MPa					C
		MPa					C
		MPa					C

Fig. 2: Screenshot of Google sheet format

Stress-Strain	Strain rate	0.001
Engineering strain	Engineering stress	
0.0000	0.0000 MPa	▼
0.0146	2.5263 MPa	▼
0.0417	15.3158 MPa	▼
0.0602	20.3684 MPa	▼
0.0932	23.5263 MPa	▼
0.1544	25.2632 MPa	▼
0.2272	27.3158 MPa	▼
0.2981	29.8421 MPa	▼
0.3660	33.3158 MPa	▼
0.4184	37.7368 MPa	▼
0.4709	42.9474 MPa	▼
0.5155	48.6316 MPa	▼
0.5563	55.7368 MPa	▼
0.5806	61.7368 MPa	▼
0.6097	70.4211 MPa	▼
0.6388	81.4737 MPa	▼
0.6553	89.5263 MPa	▼

Fig. 3: example of recorded stress strain data

Once the data had been collected it was then imported into the SQLite database. This was done using a Perl script which was provided by Stefan Szyniszewski and had been used to input the previous entries. The google sheets were downloaded to CSV files which the script would then be run on, parsing text from the file and saving extracted variables into the SQLite database. The database was organised into six different sections: MetF_General for free text information such as the foam description and manufacturing method, MetF_Index for the foams database identifier and base metal, MetF_Keywords for the database keywords, MetF_References for the information on the source of each data point such as author and link to paper, MetF_StandardTable for all of the metal foam properties and MetF_StressStrain for the stress strain data obtained using

the plot digitizer. In this project the data analysis tool focuses on the properties in the MetF_StandardTable section with background information about each foam coming from the MetF_General and MetF_Index sections.

	id	mf_id	keyword	mean v	unit
1	125	1	average pore size	3.7	mm
2	126	1	bulk density	2.45	g/cm³
3	127	1	densification strain	57	NULL
4	128	1	elastic Poisson ratio	NULL	NULL
5	129	1	Forchheimer factor	NULL	NULL
6	130	1	heat transport	NULL	NULL
7	131	1	permeability	NULL	NULL
8	132	1	plastic Poisson ratio	NULL	NULL
9	133	1	plateau stress	105	MPa
10	134	1	pores per unit length	NULL	NULL
11	135	1	porosity	58	%
12	136	1	shear failure strain	NULL	NULL
13	137	1	tensile failure strain	NULL	NULL
14	138	1	thermal conductivity	NULL	NULL
15	139	1	yield stress	55	MPa
16	140	1	Young modulus	11	GPa
17	141	2	average pore size	1.2	mm
18	142	2	bulk density	0.35	g/cm³

Fig. 4: Standard Table database section within SQLiteStudio

Figure 4 shows the structure of MetF_StandardTable with columns for database entry number (id), metal foam identification number (mf_id), property name (keyword), property value (mean value) and property unit (unit). Also in the table but not shown (and largely unused due to lack of reporting) are columns for standard deviation, number of measurements, temperature, temperature unit, additional parameter and additional parameter unit.

B. MATLAB code

Once sufficient data had been collected the project moved to the processing and post processing stage. Initially the MATLAB app "Database explorer" was used to query the SQL data. This provided an excellent beginners tool as it allowed the querying of SQL data with no SQL knowledge. The import data feature was then used to generate MATLAB scripts which ran the SQL code within them and returned the data as a MATLAB table where it could then be manipulated using MATLAB code. The issue with this approach was that each query had to be manually created and this process could not be automated. Therefore although it provided an excellent starting point eventually the app was bypassed by directly editing the SQL code within MATLAB.

```
query1 = ['SELECT mf_id, ' ...
' mean_value, ' ...
' unit ' ...
'FROM MetF_StandardTable ' ...
'WHERE keyword = '',variable1,' ' ...
' AND mean_value != 'NaN'''];

table1 = fetch(conn,query1);
```

Fig. 5: Query used to obtain the property "variable1" from the database

The first challenge in the automation of this stage was to create a table in MATLAB with all of the required data. The database was queried by table and column and then a where statement was used to specify which property (keyword) was desired. Figure 5 shows a screenshot of a query for variable1 which could be assigned to a chosen property by the user at the input stage. This query was then fetched and the result was assigned to a table called table1. Due to structure of the database, with all properties in one column, only one property could be extracted per query. Therefore to create one large table of both properties, a separate table for each property had to be extracted and then joined using the "innerjoin" function [6]. This function joins tables along a key variable, in this case mf_id, to create a new table. For example if both tables have a row with mf_id of 1 then these rows are combined in the new table. Once the tables of properties were joined, separate tables with extra information about the metal foams including base material, foam type, method of production, description and references to the papers that reported them were also added. Figure 6 shows an example of the resultant table where variable1 is porosity and variable2 is Young's modulus.

	1	2	3	4	5	6	7	8	9	10	11
	mf_id	variable1	unit	variable2	unit	base_material	foam_type	method	description	label	link
1	1	58%		11	GPa	Stainless Steel	Closed cell	Casting ...	steel hollo...	Ra2009-1	'https://ww...
2	2	96%		83	MPa	Stainless Steel	Open cell	Polymer ...	Cells take o...	Ad2004-2	'https://ww...
3	3	93.5000%		196	MPa	Stainless Steel	Open cell	Polymer ...	Cells take o...	Ad2004-3	'https://ww...
4	4	92.4000%		268	MPa	Stainless Steel	Open cell	Polymer ...	Cells take o...	Ad2004-4	'https://ww...
5	5	90.1000%		300	MPa	Stainless Steel	Open cell	Polymer ...	Cells take o...	Ad2004-5	'https://ww...
6	6	61.1000%		5.6000	GPa	Ferrous	Closed cell	PM/HS c...	Powder mal...	Ra2009-6	'https://ww...
7	7	67.6000%		5.6000	GPa	Ferrous	Closed cell	PM/HS c...	Powder mal...	Ra2009-7	'https://ww...
8	8	62.5000%		9.6500	GPa	Stainless Steel	Closed cell	PM/HS c...	Powder mal...	Ra2009-8	'https://ww...
9	9	96%		201	MPa	Stainless Steel	Closed cell	Sintered ...	Two differe...	Fr2007-9	'https://link...
10	10	96%		261	MPa	Stainless Steel	Closed cell	Sintered ...	Two differe...	Fr2007-10	'https://link...
11	11	96%		358	MPa	Stainless Steel	Closed cell	Sintered ...	Two differe...	Fr2007-11	'https://link...
12	12	96%		362	MPa	Stainless Steel	Closed cell	Sintered ...	Two differe...	Fr2007-12	'https://link...
13	13	92%		637	MPa	Stainless Steel	Closed cell	Sintered ...	Two differe...	Fr2007-13	'https://link...
14	14	92%		627	MPa	Stainless Steel	Closed cell	Sintered ...	Two differe...	Fr2007-14	'https://link...
15	15	85%		3.1500	GPa	Stainless Steel	Closed cell	Consolid...	Two differe...	Sz2014-15	'https://ww...

Fig. 6: Data table in MATLAB after using innerjoin function

Once the full table of data had been obtained it was time to move to the plotting stage. The greatest challenge for this stage was unit consistency. Due to the nature of the collected data, different papers had reported properties with different units, as can be seen in the unit_variable2 column of figure 6 with MPa and GPa. To plot these values on the same axis these units had to first be normalised. This is a simply enough task to complete manually however to automate it the code had to recognise a range of possible units for each property and then apply the correct conversion. It was also important that this range of units covered not only the units currently present in the database but also extra in case future entries differed from these. The chosen solution to this problem was to create a table in MATLAB that would be referred to, to find the unit conversion rate. A section of the unit conversion table is shown in figure 7, the full table can be found in the appendices.

Figure 8 shows an overview of how the unit conversion algorithm works. At the input stage of the code the user enters the desired properties and units to be plotted. The code stores this desired unit as well and reads the reported unit from the

	1 Pa	2 MPa	3 GPa	4 percent	5 decimal	6 um	7 mm	8 cm	9 m
1 Pa		1.0000e-03	1.0000e-06	0	0	0	0	0	0
2 MPa	1000		1.0000e-03	0	0	0	0	0	0
3 GPa	1000000	1000		0	0	0	0	0	0
4 percent	0	0	0		1	0.0100	0	0	0
5 decimal	0	0	0	100		1	0	0	0
6 um	0	0	0	0	0		1.0000e-03	1.0000e-04	1.0000e-06
7 mm	0	0	0	0	0	1000		0.1000	1.0000e-03
8 cm	0	0	0	0	0	10000	10		0.0100
9 m	0	0	0	0	0	1000000	1000	100	

Fig. 7: Screenshot of a section of the unit conversion table

first row of the data table. The conversion rate for these units is then found at the intersection of these two units in the unit conversion table (where the row name is the reported unit and the column name is the desired unit). The recorded value (in column variable1) is then multiplied by this conversion rate to give the value in the correct unit. This process is then repeated for each row of the table and then again for the second variable.

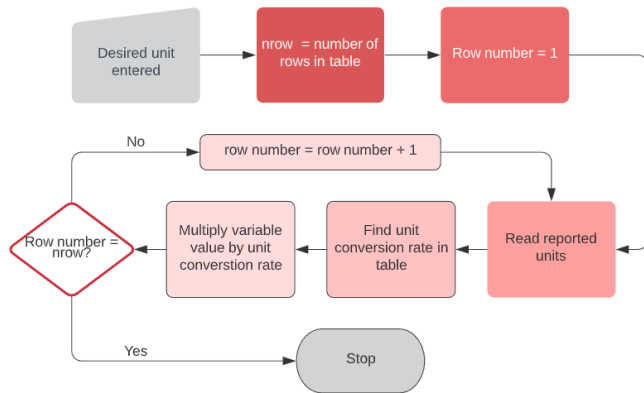


Fig. 8: Flowchart showing unit conversion algorithm

One issue with this method is that, for the unit conversion table to work, both the desired and the reported units had to exactly match the row and column names of the table. MATLAB does not allow the names of rows and columns to be either empty or to include special characters. This meant that unit entries that were empty such as decimals or units that used special characters such as % or g/cm^3 were not able to be indexed properly in the unit conversion table. To get around this problem these rows and columns were given alternative names (e.g. 'percent' and ' g_cm3' ') and prior to the unit conversion algorithm the code would scan each unit column and change any of these units to their alternative names. This is why figure 7 has columns of "percent" and "decimal". At this stage the code performed three other operations as well. The first of which is to filter and remove any rows that still had empty unit column. If the user has specified a property that may be reported in decimal such as porosity or densification strain then the code changes any empty unit entries to "decimal". However if an empty unit column is found for any property that should not be reported in decimal (yield stress for example) then this is the result of an error in data

collection and the row is removed.

The second operation is to alter the "label" column of the data table. When it is imported from the database the "label" column represents a combination of the first two letters of the lead authors surname, the year the paper was written and the metal foam id number. For example 'Ra2009-1' stands for Rabiei A, 2009, metal foam id 1. This column is used at the plotting stage of the code to group the results by study and therefore, to operate correctly, entries from the same study must have the same label. This is achieved by removing the metal foam section of each row so that only the author and year are represented (e.g. Ra2009-1 changes to Ra2009). The final operation is to fix any issues that occurred with with the conversion from CSV files to the database. One example of this issue was that units that included powers would be converted to html form (e.g. m^2 was stored as ' $m²$ ' in the database). This was fixed by scanning each row for ' $m²$ ' and replacing it with ' m^2 '. Once the units had been converted, they no longer had to match the row column names of the unit table and therefore a function called "prettyunit" was created to convert the units back so that they displayed nicely in the final table of data. This can be seen in figure 9 where '%' is shown rather than 'percent'.

When analysing the properties of metal foams the two factors which can have the greatest effect are the the foam cell type (such as open or closed) and the base material however the effect of these are already fairly well understood. A steel foam will usually be stronger than an aluminium foam of similar topology and the same can be said of closed cell foam vs an open cell foam. As a result a researcher may only be interested in comparing foams of the same metal or cell type and therefore it was decided that extra functionality to filter by base material and cell type should be added. This could not be done at the query stage as data about cell type and base metal are present only in the MetF_General and MetF_Index sections of the database. Therefore additional functions to remove unwanted entries from the data table were created. These functions scanned through their respective columns (base_material and foam_type) and marked any rows that didn't match the entry that had been specified by the user at the input stage (i.e. didn't match the metal specified or the cell type specified). These rows were then deleted to give the filtered final data table which was ready to be plot. The reason that rows could not be immediately deleted from the table is that this would cause a change in the table size and rows would consequently be missed in the scan. Figure 9 shows the result of unit conversion and filtering to show only steel alloy and titanium, closed cell entries on the table in figure 6. The final function of code called 'numericalfilter' allows the user to filter the data to be within a certain range of values for a third variable. This function can be switched on or off and runs the same operations as 'jointables', 'convertunit' and 'prettyunit' but only for the selected 'fitvariable'.

At this stage the data is ready to be plotted. This was done using the "gscatter" function which allows data points to be grouped by a chosen variable. Based off of the user input this variable is defined to be the "base_material" column or

the "label" column. Finally the column names of the data table are changed to represent the chosen properties and the table is exported using "exporttable" function. For example variable1 is changed to porosity and unit_variable1 is changed to unit_porosity.

	1	2	3	4	5	6	7	8	9	10	11
	mf_id	porosity	unit_por	Young_mod	unit_Young	base_material	foam_type	method	description	label	link
1	1	58	%	11000	MPa	'Stainless St...	'Closed cell'	'Casting...	'steel hollo...	'Ra2009'	'https://ww...
2	2	96	%	83	MPa	'Stainless St...	'Open cell'	'Polyme...	'Cells take ...	'Ad2004'	'https://ww...
3	3	93.5000	%	196	MPa	'Stainless St...	'Open cell'	'Polyme...	'Cells take ...	'Ad2004'	'https://ww...
4	4	92.4000	%	268	MPa	'Stainless St...	'Open cell'	'Polyme...	'Cells take ...	'Ad2004'	'https://ww...
5	5	90.1000	%	300	MPa	'Stainless St...	'Open cell'	'Polyme...	'Cells take ...	'Ad2004'	'https://ww...
6	6	61.1000	%	5600	MPa	'Ferrous'	'Closed cell'	'PM/HS ...	'Powder m...	'Ra2009'	'https://ww...
7	7	67.6000	%	5600	MPa	'Ferrous'	'Closed cell'	'PM/HS ...	'Powder m...	'Ra2009'	'https://ww...
8	8	62.5000	%	9650	MPa	'Stainless St...	'Closed cell'	'PM/HS ...	'Powder m...	'Ra2009'	'https://ww...
9	9	96	%	201	MPa	'Stainless St...	'Closed cell'	'Sintere...	'Two differ...	'Fr2007'	'https://link...
10	10	96	%	261	MPa	'Stainless St...	'Closed cell'	'Sintere...	'Two differ...	'Fr2007'	'https://link...
11	11	96	%	358	MPa	'Stainless St...	'Closed cell'	'Sintere...	'Two differ...	'Fr2007'	'https://link...

Fig. 9: Section of final data table after unit conversion and filtering

Once the tool had been completed it was used to analyse a variety of relationships within the data. The analysis begun by looking into relationships where there was already significant understanding. This aimed to test the accuracy of the data as well as the applicability of the tool. The analysis then moved to other relationships and aimed to identify previously unknown trends in the data or to identify areas of interest which could support further study.

III. RESULTS

A. Database Analysis Tool

The first and main result of this project is the data analysis tool which has been developed. This tool is in the form of a MATLAB script which automates the fetching, processing and post-processing of property data from the Springer metal foams database based off of user input. The script is fully commented, providing a breakdown of the operation of each segment of code throughout as well as detailed instructions on how to use it. In order to operate this script a MATLAB license is required however as this tool is catered towards academic research this should not be an issue. Also required are two additional files being the metal foams database file (metal-foams_sqlite3.db) and the JDBC SQLite driver file (sqlite-jdbc-3.30.1.jar). As mentioned above complete instructions on how to download these additional files and then set up the connection to the database are included in the script pre-amble. A copy of the full MATLAB script including these instructions is shown in the appendices.

The first section of the script is the input stage. This stage consists of a series of variables, shown in figure 10 which are set by the user to determine the output. The first variable, "databasename", should be set to whatever the user named the database connection and is required to query the data. The next two set of variables determine the x and y axis of the final graph. Users decide which variables they want to plot on each axis, variable1 and variable2, (out of 15 possible variables) as well as the units they want the data to be in, unit1 and unit2. One condition for these variables is that they must exactly match the property names from the metal foams database (case sensitive) for the script to recognise them. To help with this a

```
databasename = 'Metalfoams';
% X-axis
variable1 = 'porosity';
unit1 = 'percent';
log1 = 0;
setlimit1 = 0;
minmax1 = [0 100];
% Filter Variable
setnumericalfilter = 0;
filtervariable = 'yield stress';
filterunit = 'MPa';
filterrange = [0 5];
% Y-axis
variable2 = 'permeability';
unit2 = 'm2';
log2 = 0;
setlimit2 = 0;
minmax2 = [0 1];
% Options
setcelltype = 0;
metals = {'all'};
groupingvariable = 0;
exporttable = 0;
```

Fig. 10: Inputs of MATLAB script

list of all possible property names and their available units has been provided above the inputs in the comments section of the script. The simplest way to avoid errors here is to simply copy and paste these entries. The "log" variables can be thought of as switches which give the user the option of setting a log scale on either axis, 0 resulting in a normal scale and 1 a log scale. The "setlimit" variables also act as switches but when turned on (with a value of 1) apply the limits specified in "minmax" for the respective axis. This is designed to allow the user to focus on specific areas of a graph and ignore other values. The next set of variables work similarly to the x and y-axis sections except these allow the user to specify a filter on the data. The filter is turned on or off with 'setnumericalfilter' and then the variable to filter by, the unit for this variable and the range for the final data to be with are specified in the next 3 variables. If the example shown in figure 10 was turned on, all of the final plotted values would have yield stresses between 0 and 5 MPa. The "setcelltype" variable allows the user to filter the results by specific cell type, with a 0 showing all cell types, 1 showing only open cell foam, and 2 showing only closed cell foams. The "metals" variable performs a similar function but filters the results by base material. This value is a cell array that can contain any of the metals from the base_materials column of MetF_Index. Like with variable1 and variable2 these entries must match exactly and so to help with this a list of all available metals (7 at the time of writing) is provided above the inputs in the comment section. Setting this value to 'all' removes the filter and shows results from all metals. The "groupingvariable" variable determines what the points in the scatter graph will be grouped by. When set to 0 points are grouped by their base metal and when set to 1 data points are grouped their "label". The final variable "exporttable" gives the user the option of exporting the final table of data from MATLAB into excel. When this value is set to 1 the final table of data (join2) is exported to an excel file with the name of variable1_variable2.xlsx. For example using the inputs from figure 10 the excel file would be named "porosity_permeability.xlsx", a full screenshot of this file is shown in the appendices. Once the user has selected all of the desired inputs the code can then be run. A detailed breakdown on the operation of each function has been provided in the methods section in addition, figure 11 shows an overview of how the full script functions.

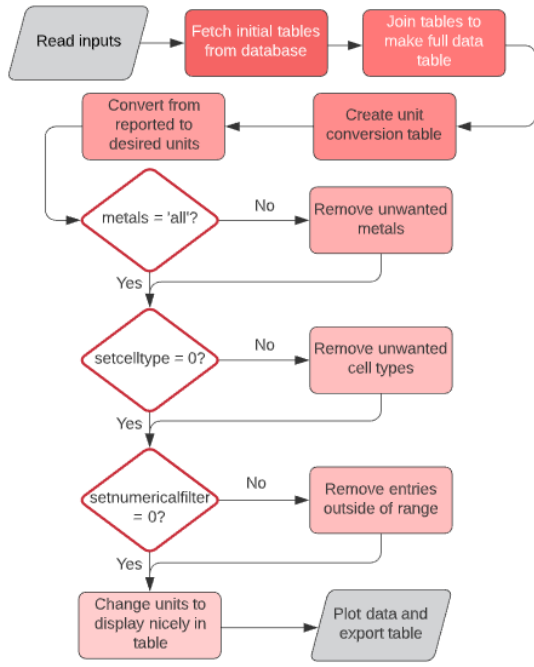


Fig. 11: Flowchart showing operation of MATLAB script

B. Data analysis

The initial results obtained using the database analysis tool were centred around relationships which were already well understood. The aim of these results was to test that the analysis tool performed as expected and to check the validity of the data from the database. One of chosen relationship was between porosity and bulk density and the results are shown in figure 12.

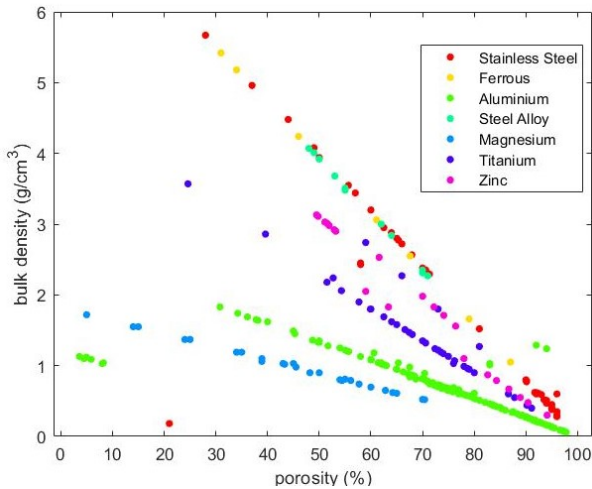


Fig. 12: Scatter plot of porosity against bulk density for all metals

$$\phi = 1 - \frac{\rho_{bulk}}{\rho_{particle}} \quad (1)$$

Porosity is defined as the volume fraction of pores in a foam and can be calculated using equation 1 where ϕ is porosity and ρ_{bulk} is the bulk density and $\rho_{particle}$ is the particle or true

density. As a result we would expect to see a linear relationship between these two properties with a gradient of $-\frac{1}{\rho_{particle}}$. Looking at figure 12 we can observe a separate line for each of base metal with each line converging to 0 g/cm³ at 100% porosity. This agrees with the equation as each metal has a different base metal density and at 100% porosity the foam sample would be completely made up of air and therefore have a close to 0 density. For the case of stainless steel there are also some ferrous and steel alloy foams along the same line and this is likely caused by the base metal densities of each metal type being so similar that the gradient differences are negligible.

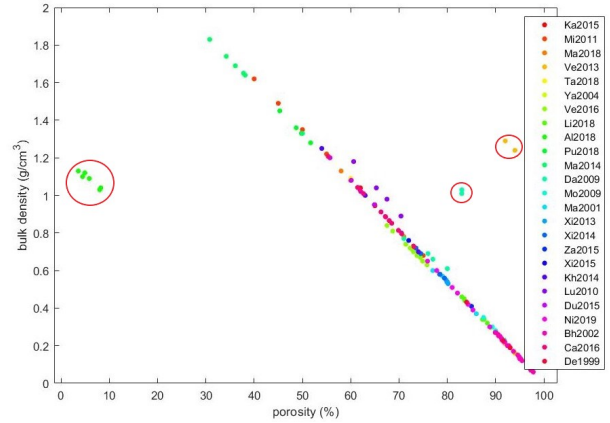


Fig. 13: Scatter plot of bulk density against porosity for Aluminium

Filtering the data by metal and grouping by study allows us to take a closer look at the Aluminium data, shown in figure 13. It can be seen that there are three sets of outliers to the overall trend which have been circled in red and belong to studies Al2018, Da2009 and Ve2013 (from left to right). The data table was then exported to excel where the links to these studies could be found for further investigation. For Al-Sahlani K, 2018 (Al2018) [7] there appears to have been a data collection error and porosity values should be within the 57-61% region. For Daoud A, 2009 (Da2009) [8] the cause of the outliers are that the metal foam samples are actually Zn12Al based rather than pure aluminium. This alloy has a much higher density than Aluminium, at 6.0 g/cm³ which causes the points to have a different gradient to the Aluminium line. This is an error in data reporting as these foams should be classified as 'Zinc'. Finally in the case of Vesenjak M, 2013 (Ve2013) [9] the two samples that were outliers consisted of aluminium foam samples with silicone pore filler, which significantly increased the density of the samples while the porosity remained the same. To determine the reliability of the data, excel's curve fitting function was used to find the y intercept of the graph which should be the base metal density of aluminium. The true density is about 2.71 g/cm³ and our value was 2.69 which is very similar.

Next the permeability was plotted against porosity. This was an interesting relationship because the trend in results should be independent of base material as permeability depends on the pores alone and not on the material around them. The

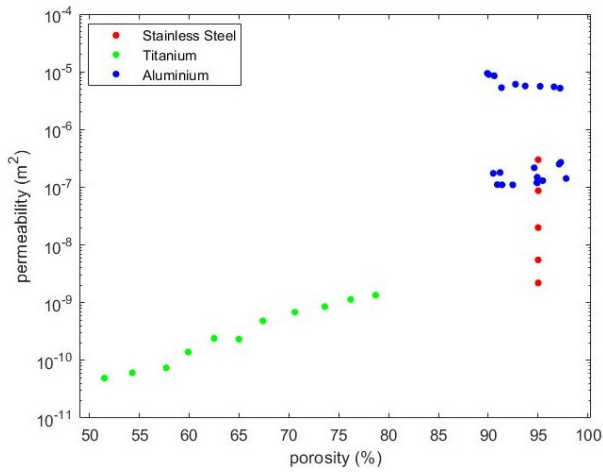


Fig. 14: Scatter plot of permeability (log scale) against porosity for all metals

results are shown in figure 14. It can be seen that there is a general trend for both aluminium and titanium for permeability to increase with porosity. This is as expected as the higher the volume fraction of the open pores the better the flow through the foam should be. For stainless steel however there is an increase in permeability while the porosity remained the same. Upon reviewing the source of this data it can be seen that the reason for this trend is that although the porosity of each sample remained the same the pore diameter varied from 0.45mm for the lowest permeability to 4.8mm for the highest permeability. This finding backs up research such as Ren, 2017 [10] who modelled the effect of pore size on permeability and found that "the larger the pore radius, the larger the apparent permeability". Despite the trend shown it must be noted that data from only three studies is present and as a result no definite conclusions can be made.

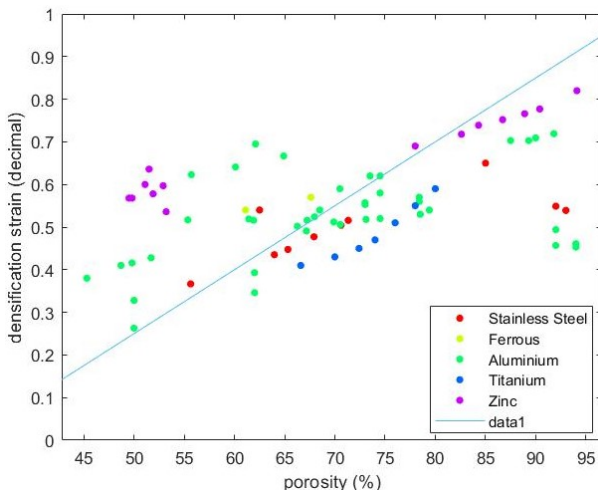


Fig. 15: Filtered densification strain against porosity

When plotting the densification strain against porosity the initial graphs shows little trend in the data with two outliers of 28.9 and 57 and the rest below 1. Engineering strain should always be below 1 so it is clear an error has occurred. Upon

inspection it can be seen that the cause of the outliers are data collection errors. Checking the source of these data points it can be seen that these values should be reported as percentages however in the database they lack any unit. As a result the unit conversion algorithm has read these values to be decimals and therefore not divided them by the unit conversion rate (100). The data was then filtered to include only densification values between 0 and 1 and can be seen in figure 15. Plotted as the blue line (data1) is a basic relationship between densification strain and porosity shown in equation 2 which was reported in Gibson and Ashby [4]. $\frac{\rho}{\rho_s}$ represents the relative density which is the same as 1-porosity and α_1 is a constant that ranges from 1.4-2 but, as a rule of thumb, can be assumed to be $\alpha_1 \approx 1.5$.

$$\epsilon_D = \left(1 - \alpha_1 \frac{\rho}{\rho_s}\right) \quad (2)$$

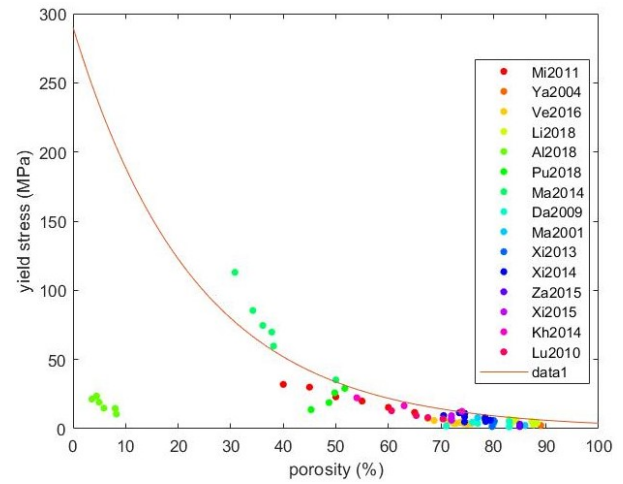


Fig. 16: Yield stress against porosity for Aluminium foams

Figure 16 shows the results of yield stress against porosity. Yield stress data for each base metal was present however each of these followed a different trend curve and so the data was filtered to show only Aluminium foam samples (as it was the base metal with the most data points). The results were then plotted against the model proposed by Park J, 2015 [11] shown in equation 3 where σ_t is the yield stress, σ_0 is the tensile strength that of the base metal and η is the porosity.

$$\sigma_t = \sigma_0 \exp(-4.3\eta) \quad (3)$$

The analysis then moves towards attempting to uncover new relationships in the data. The effect of the average pore size on the mechanical properties of the metal foams was investigated. Figure 17 and 18 shows the effect on the densification strain and Young's modulus respectively. In both cases there seems to be no noticeable trend between the two variables for any metal however this does not necessarily mean that there is no relationship between them. When looking at each graph there can be seen many groups of results that trend vertically with no change in pore size, a few examples have been circled in red. In these cases, as the pore size remains the same, the changes in densification strain and Young's modulus must be the result of other variables changing. Even when this vertical trend is

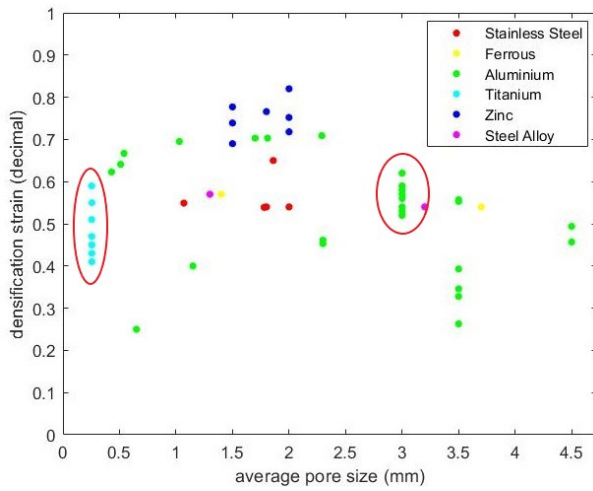


Fig. 17: Densification strain against average pore size

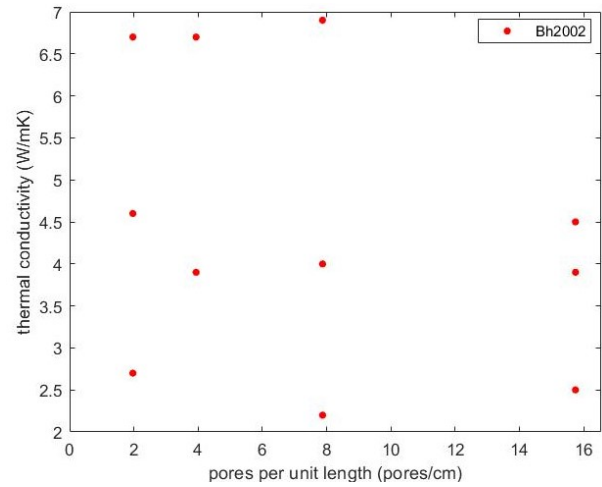


Fig. 19: Thermal conductivity against pores per unit length

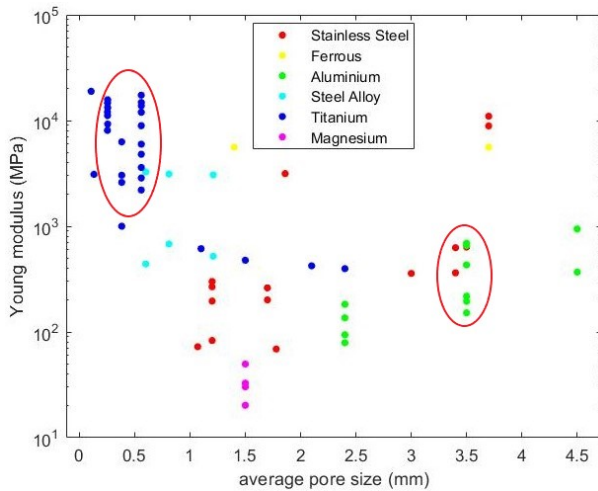


Fig. 18: Young's modulus against average pore size

not seen there is no way of knowing whether other properties of the foams are affecting the result without checking the research paper for each and as a result the results are not conclusive. Both of these properties were also tested against pores per unit length but, as this is a much rarer property to be reported, there was far less data and therefore the results were even more inconclusive.

Finally the effect of pores per unit length on thermal conductivity was tested for Aluminium foams, shown in figure 19. The results show no noticeable trend. It is again difficult to draw a definite conclusion from this analysis as there is only one study present and upon inspection of the paper it can be seen that the foam samples varied in porosity as well as pore distribution. In reality, both pores per unit length and pore size are widely used by industry and manufacturers to classify metal foams, but are fairly meaningless for mechanical and fluidic properties as they don't account for each other. For example a foam with very large pore size and low pores per inch would likely be far weaker than a similar foam with low pore size and large pores per inch. In order to properly compare analyse these properties the porosity of the foam

would have to be controlled so that the only differences in sample were the properties being tested. This was attempted using the 'numericalfilter' function but this resulted in too few results to analyse.

IV. DISCUSSION

The first four results provided a good test of both the analysis tool and the data. Results were plotted on a variety of axis, filtered by metal and cell type, grouped by metal and study and exported for further investigation proving the functionality of the analysis tool. These results were then analysed to test the validity of the data. In the case of Bulk density against porosity there can be seen a distinct gradient for each metal with some overlap on the more dense base metals. The results for Aluminium were then analysed and once the outliers had been removed the result was found to closely predict the base metal density of Aluminium (0.02 g/cm^3 off). For porosity against permeability the data shows a positive correlation as would be expected. More data points would allow a more detailed conclusion here and without having a constant cell size no trend can be plotted for comparison.

For porosity against densification strain a positive trend can be seen, however, the gradient of this trend is lower than that suggested by equation 2. One cause of this might be that, when plotting equation 2, an average value for α_a was used. This value varies for structure and base metal and in figure 15 the data is not filtered by either of these factors. For the yield stress against porosity a curved trend with yield stress increasing as porosity decreases can be seen. This is again to be expected as sample strength increases with bulk density. The trend is very similar to that proposed by Park J, 2015 [11] with only one group of outliers however upon investigation these results can be seen to be from Al-Sahlan K, 2018 (Al2018) which was highlighted in figure 15 identified as a data collection error. These results show that when sufficient data is available the database provides a good match with published models.

When it comes to investigating new relationships the tool proved to be limited by the lack of data for some properties. Most papers report common foam properties such as porosity,

Young's modulus and bulk density and as a result there is plenty of data to analyse for these properties. Rarer properties such as shear failure strain, Poisson ratio and heat transport are more difficult to reliably investigate due to the lack of data point. This is compounded by the fact that the most common properties tend to be the best understood and therefore research interest lies elsewhere. As a result relationships such as the effect of pore size on Poisson ratio and porosity on heat transport could not be studied. It is important to remember that the database is still in its early days with only 329 entries and as this number grows some of these issue could disappear. For example when investigating the effects of pore size. For example, when investigating the effects of pore size the tool was able to demonstrate a lack of trend (negative result) but one possible issue with this result is that there was no way to know if other properties such as porosity or pores per unit length were constant throughout the data. Therefore other factors could have influenced the result. At the time writing there was not enough points to filter the data to avoid these differences however in the future, with a larger database, there might be. Another possibility is that once the population size is large enough these differences could average out allowing overall trends to be seen.

One additional use that the tool proved very useful for was data visualisation and consequently identifying outliers in the data, as seen in figure 15. This application can be used to identify foams with special characteristics due to their manufacturing process or structure. One example of this are the foam samples from Vesnjak M, 2013 [9] which contained silicone pore filler. In this example the property was high bulk density which is usually an unattractive property. However, it still serves a proof of concept for this application.

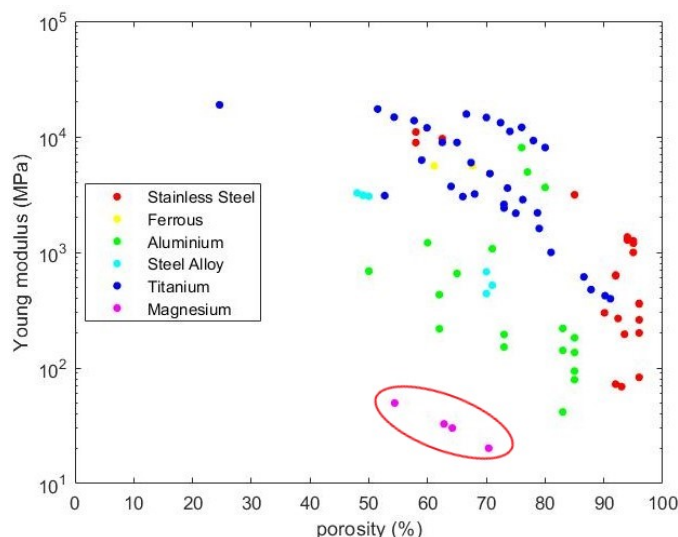


Fig. 20: Young's modulus against porosity

Another application for the identification of outliers is to find errors within the database. Due to the collection process there are many stages at which these errors can occur. To start with, errors can be made even in peer reviewed research papers. Two examples of this were found during the research process. In the first example (Bhattacharya A, 2002) [12]

permeability values were reported as 1E+07 rather than 1E-07. This was obvious to see when plotting the results and the author of the paper was contacted and confirmed the error. The second example (Wang X, 2014) [13], was less obvious and was investigated as values for Young's modulus, shown circled in red in figure 20 seemed lower than normal values. It was determined that this was due to incorrect testing procedures. In this case the initial loading slope was used to determine Young's modulus. This value is incorrect as it includes the deformations of the crushed region in the proximity of the loading plates (see Szyniszewski S, 2014 [14]). In order to properly test the Young's modulus the an unloading stage should be added prior to this. The other and more common stage for errors to occur is in the data collection stage. Here there are many mistakes which can be made such as adding incorrect units, misreading data from papers or entering data to the wrong cell in the google sheet file. In total 26 errors of this kind were found and corrected over the course of this project with many of these being examples of data being copy and pasted from one sheet to another incorrectly. When using the data base for any application it is essential that the data is correct. In a research scenario errors in the data could lead to incorrect conclusions being drawn and if the database was used for material selection purposes errors could lead to serious safety issues. As a result this is an valuable application of the tool.

A. Future direction

The data driven approach to metal foam research shows a lot of potential however it is clear that the applications of the data analysis tool are currently limited by the lack of data available. Currently the data collection process is by far the most time consuming stage of this data driven method. One solution to this problem, which could be explored in the future is to open the data collection process to more people. There is currently no standard reporting format for experimental results and therefore data collection for the database involves manually searching through available literature to find relevant data and then adding it the google sheets document. A far more efficient process would be to open the data collection process to researchers with the hope that they would upload their research as and when they conducted it. By sharing this process between many researchers the rate of data collection would be greatly increased at very little time cost. This rapid uploading of data would also facilitate the analysis of new foams and prototype materials much earlier than the current method would allow. This approach could be implemented in a variety of ways. There are two ways this could be implemented. The first would be to collect the data in the pre-submission archiving stage using platforms such as EngXiv [15]. This would be a more polished and robust system than the google sheets method and would be completely free for anyone to access. This method would raise data validity concerns as there is no screening process to publishing papers and therefore it might be necessary to have a small team of curators checking data before it is uploaded. The major advantage of hosting the database on a website such as EngXiv is that it

would be free to use for anybody. The second method would be to collect the data during the uploading of a manuscript for journal consideration. Already for many journals when submitting a paper for consideration there is an option to upload collected data however this is usually unstructured and therefore rarely used. If this option was developed into a structured format with sections, for example: Cellular metals → Mechanical properties → Young's modulus, it could then be uploaded straight to a database without the need for formal collection. As data is peer reviewed prior to publication this method would have fewer data validation issues however being owned by a company such as SpringerMaterials would put access behind a pay wall.

It is similarly important to allow open access to the data analysis tool. Not only would allowing open access would incentivize researchers to add their own data to the database to improve the insight and reliability of data but it would also allow the analysis tool to be iterated upon to allow new features to be added as the database evolves. One obvious addition to the tool would be to add a GUI for improved usability. This would remove the need to copy and paste properties, units and metals when altering variables as drop down boxes could be used instead. Another development, which would greatly improve quantity of available data, is to add functionality to analyse stress strain data. As discussed earlier many variables are rarely reported in research papers which makes them difficult to analyse, however almost all papers reporting mechanical properties include stress strain graphs from which many variables can be extracted. Currently in the database there is data from around 300 stress strain curves, none of which is being used and therefore a feature to process this data to fill the gaps in reported properties would be a great addition.

In the future the uses of the code could change entirely with one example being material selection. This is again currently limited by the lack of available data however with more data points the tool could be used to plot two desirable properties against each other to identify metal foams which suit certain requirements, say below a certain bulk density while also having a Young's modulus of above a certain value. In this way the analysis tool would act very similarly to the GRANTA selector tool [16] produced by GRANTA design with the advantage of being open source and therefore free to use while also having newer and more experimental foam types available due to the rapid open source data collection.

V. CONCLUSIONS

It is clear that a data driven approach shows a lot of potential for the field of cellular metal research. The data analysis tool developed in this project has proved very useful for identifying outliers and errors in the data and given enough data points has been able to find known relationships between properties such as porosity, bulk density, densification strain and Young's modulus.

When it comes to uncovering new relationships, the tool was able to identify a lack of trend in mechanical properties and pore size but the lack of data for certain properties has

prevented their analysis. Even when data has been present variation in variables other than those that were being investigated made it difficult to come to solid conclusions on the trend. Looking to the future there are many areas in which this data driven method could be developed. Evidently there is a large demand for more data and this highlights the fact that data driven research not a replacement for experimental research. It is, in fact, highly dependant on experimental results and should be thought of more as supplementation than replacement. Arguably the most effective method of growing the available data would be to allow open access to the database and encourage researchers to upload their own research as it is conducted. As this pool of data grows, so too will the applications and reliability of the data analysis tool will grow which hopefully will incentivize further progress.

ACKNOWLEDGMENT

The author would like to thank Stefan Szyniszewski for his help and supervision throughout this project

REFERENCES

- [1] O. B. Cardoso E, "Study of the use of metallic foam in a vehicle for an energy economy racing circuit," *Materialwiss. Werkstofftech.*, vol. 41, p. 257–64, 2010.
- [2] S. H. Banhart J, "Aluminium foam sandwich panels: manufacture, metallurgy and application," *Adv Eng Mater*, vol. 10, pp. 793–802, 2008.
- [3] Y. Oshida, "12 - advanced materials, technologies, and processes," in *Bioscience and Bioengineering of Titanium Materials (Second Edition)*, second edition ed., Y. Oshida, Ed. Oxford: Elsevier, 2013, pp. 457 – 497.
- [4] "Chapter 4 - properties of metal foams," in *Metal Foams*, M. Ashby, A. Evans, N. Fleck, L. Gibson, J. Hutchinson, and H. Wadley, Eds. Burlington: Butterworth-Heinemann, 2000, pp. 40 – 54.
- [5] "Plot digitizer," <http://plotdigitizer.sourceforge.net/>.
- [6] "innerjoin function," <https://uk.mathworks.com/help/matlab/ref/innerjoin.html>.
- [7] K. Al-Sahlani, S. Broxtermann, D. Lell, and T. Fiedler, "Effects of particle size on the microstructure and mechanical properties of expanded glass-metal syntactic foams," *Materials Science and Engineering: A*, vol. 728, pp. 80 – 87, 2018.
- [8] A. Daoud, "Effect of strain rate on compressive properties of novel zn12al based composite foams containing hybrid pores," *Materials Science and Engineering: A*, vol. 525, no. 1, pp. 7 – 17, 2009.
- [9] M. Vesenjak, L. Krstulović-Opara, and Z. Ren, "Characterization of irregular open-cell cellular structure with silicone pore filler," *Polymer Testing*, vol. 32, no. 8, pp. 1538 – 1544, 2013.
- [10] R. Y. T. W. Wenxi Ren, Gensheng Li, "A theoretical analysis of pore size distribution effects on shale apparent permeability," *Geofluids*, 2017.
- [11] J. Park, S. Lee, S. Kang, J. Jeon, S. H. Lee, H. kyu Kim, and H. Choi, "Complex effects of alloy composition and porosity on the phase transformations and mechanical properties of powder metallurgy steels," *Powder Technology*, vol. 284, pp. 459 – 466, 2015.
- [12] A. Bhattacharya, V. Calmide, and R. Mahajan, "Thermophysical properties of high porosity metal foams," *International Journal of Heat and Mass Transfer*, vol. 45, no. 5, pp. 1017 – 1031, 2002.
- [13] X. Wang, Z. Li, Y. Huang, K. Wang, X. Wang, and F. Han, "Processing of magnesium foams by weakly corrosive and highly flexible space holder materials," *Materials & Design*, vol. 64, pp. 324 – 329, 2014.
- [14] S. Szyniszewski, B. Smith, J. Hajjar, B. Schafer, and S. Arwade, "The mechanical properties and modeling of a sintered hollow sphere steel foam," *Materials & Design (1980-2015)*, vol. 54, pp. 1083 – 1094, 2014.
- [15] "Engxiv," <https://engrxiv.org/>.
- [16] "Granta selector," <https://grantadesign.com/industry/products/ces-selector/>.

APPENDIX

16/04/20 16:18 ...\\Metal foam database analysis tool.m 1 of 18

```
%% Metal foam database processor
% This code automates the collection and processing of data from the
% "metalfoams_sqlite3" database (MetF_StandardTable) to produce a table
% of data and graph of two metal foam properties

% Before running the code the database data source must be set up
% First ensure that the metalfoams_sqlite3.db file is downloaded in the
% same folder as this Matlab file
% Then go to https://bitbucket.org/xerial/sqlite-jdbc/downloads/ to download the
% latest JDBC driver
% Once the JDBC driver is installed enter "configureJDBCDataSource" into
% the command line
% In the JDBC Data Source Configuration pop up enter the following:

% Name: Provide a name for the database (e.g. Metalfoams). Note this name
%       as it will be used later (databasename)
% Vendor: Other
% Driver location: Enter the full path to the JDBC driver file (or Select the
%                  location of the JDBC driver using the button to the right)
% Driver: org.sqlite.JDBC
% URL: jdbc:sqlite:DBPATH (where dbpath is the full path to your SQLite
%       database on your computer, including the database file name!)
%       Example: jdbc:sqlite:C:\Database\metalfoams_sqlite3.db

% Click test
% In the pop up box leave Username and Password blank, click test
% If connection is successful a box will pop up saying "Connection
% successful!"
% Click save

% Database is now connected!
% For more information on setting up the database connection please visit
% uk.mathworks.com/help/database/ug/sqlite-jdbc-windows.html

%% Choose variables you wish to compare and their units
% Below is a list of available properties keywords and their available units:
% Variable names (variable1, variable2, filtervariable) and unit names
% (unit1, unit2, unit_filter) must match corresponding item from the list
% exactly (case sensitive)
% Easiest way to ensure no errors is to copy and paste items from list
% If empty tables are produced then there is no available data
% -----
% 'average pore size'           unit: 'um', 'mm', 'cm', 'm'
% 'bulk density'                unit: 'g_cm3', 'kg_m3'
% 'densification strain'        unit: 'decimal', 'percent'
% 'elastic Poisson ratio'       unit: 'decimal', 'percent'
% 'Forchheimer factor'          unit: 'one_m', 'one_ft'
% 'heat transport'              unit: 'W', 'kW'
% 'porosity'                    unit: 'percent', 'decimal'
```

Fig. 21: Full MATLAB script (1/15)

16/04/20 16:18 ...\\Metal foam database analysis tool.m 2 of 18

```
% 'permeability'                unit: 'm2'
% 'plastic Poisson ratio'       unit: 'decimal','percent'
% 'plateau stress'              unit: 'Pa','MPa','GPa'
% 'pores per unit length'       unit: 'pores_cm','pores_inch'
% 'shear failure strain'        unit: 'decimal','percent'
% 'tensile failure strain'      unit: 'decimal','percent'
% 'thermal conductivity'       unit: 'W_mK'
% 'yield stress'                unit: 'Pa','MPa','GPa'
% 'Young modulus'              unit: 'Pa','MPa','GPa'
% =====

% Below is a list of available base metals:
% Metal names (metals) must match corresponding item from the list exactly
% (case sensitive)
% Easiest way to ensure no errors is to copy and paste items from list
% -----
% 'all'                          All base metals are included
% 'Aluminium'
% 'Ferrous'
% 'Magnesium'
% 'Stainless Steel'
% 'Steel Alloy'
% 'Titanium'
% 'Zinc'
% =====

close all;
clear all functions;

databasename = 'Metalfoams';          % Enter whatever you named the database
while setting up the connection

% X-axis
variable1 = 'porosity';
unit1 = 'percent';
log1 = 0;                             % scale: 0 for normal, 1 for log
setlimit1 = 0;                         % 0 for no limits, 1 for limits
minmax1 = [0 100];                    % Specify limits

% Y-axis
variable2 = 'Young modulus';
unit2 = 'MPa';
log2 = 0;                             % Y-axis scale: 0 for normal, 1 for log
setlimit2 = 0;                         % 0 for no limits, 1 for limits
minmax2 = [0 50];                     % Specify limits

setcelltype = 0;                      % Specify which cell type to compare
% 0 for all cell types, 1 for open cell, 2 for closed cell

for closed cell
```

Fig. 22: Full MATLAB script (2/15)

16/04/20 16:18 ...\\Metal foam database analysis tool.m 3 of 18

```

metals = {'Aluminium';'Ferrous'};      % Specify which metals to include use 'all'
to include all metals                    % Metals must be separated by semicolons

setnumericalfilter = 0;                 % Adds a third variable to filter the results
by: 0 for off, 1 for on
filtervariable = 'yield stress';        % Specify variable
filterunit = 'MPa';                     % Specify unit
filterrange = [0 5];                    % Specify numerical range to include in
results

groupingvariable = 0;                    % Specify which variable to group data by in
scatter graph                            % 0 for base metal (base_material), 1 for the
study that the data came from (label)    % For example 'Ra2009' (= Rabiei A, 2009)

exporttable = 0;                         % 0 for no, 1 for yes
                                          % Previous tables of the same file name
                                          % must be deleted

%% Runs functions

[table1,table2,index,foamtype,method,description,references,filtertable] = fetchdata(
(variable1,variable2,filtervariable,databasename);
[join1,join2,join3,join4,join5,join6,nrow] = jointables(table1,table2,index,
foamtype,method,description,references);
[unit_table] = makeunittable;
[join2,nrow] = convertunit(join2,variable1,variable2,unit1,unit2,unit_table,nrow);
if strcmpi('all',metals) == 0
    [join2,nrow] = removemetals(join2,nrow,metals);
end
if setcelltype == 1 || setcelltype == 2
    [join2,nrow] = removecells(join2,nrow,setcelltype);
end
if setnumericalfilter == 1
    [join2,filtertable,nrow] = numericalfilter(join2,unit_table,filterrange,
filtertable,filtervariable,filterunit);
end
[join2,printunit1,printunit2] = prettyunit(join2,unit1,unit2,nrow);
[graph1,join2] = drawgraph(join2,variable1,variable2,printunit1,printunit2,log1,
log2,exporttable,setlimit1,setlimit2,minmax1,minmax2,groupingvariable);

%% Makes connection to database and queries relevant data

function [table1,table2,index,foamtype,method,description,references,filtertable] =
fetchdata(variable1,variable2,filtervariable,databasename)

% Make connection to database

```

Fig. 23: Full MATLAB script (3/15)

16/04/20 16:18 ... \Metal foam database analysis tool.m 4 of 18

```

conn = database(databasename, '', '');

% Defines SQL queries
query1 = ['SELECT mf_id, ' ...
' mean_value, ' ...
' unit ' ...
'FROM MetF_StandardTable ' ...
'WHERE keyword = '',variable1,''' ' ...
' AND mean_value != 'NaN'''];

query2 = ['SELECT mf_id, ' ...
' mean_value, ' ...
' unit ' ...
'FROM MetF_StandardTable ' ...
'WHERE keyword = '',variable2,''' ' ...
' AND mean_value != 'NaN'''];

query3 = ['SELECT mf_id, ' ...
' base_material ' ...
'FROM MetF_Index'];

query4 = ['SELECT mf_id, ' ...
' entry ' ...
'FROM MetF_General ' ...
'WHERE keyword = 'foam type''];

query5 = ['SELECT mf_id, ' ...
' entry ' ...
'FROM MetF_General ' ...
'WHERE keyword = 'method''];

query6 = ['SELECT mf_id, ' ...
' entry ' ...
'FROM MetF_General ' ...
'WHERE keyword = 'description''];

query7 = ['SELECT mf_id, ' ...
' label, ' ...
' link ' ...
'FROM MetF_References'];

query8 = ['SELECT mf_id, ' ...
' mean_value, ' ...
' unit ' ...
'FROM MetF_StandardTable ' ...
'WHERE keyword = '',filtervariable,''' ' ...
' AND mean_value != 'NaN'''];

% Fetches queries

```

Fig. 24: Full MATLAB script (4/15)

16/04/20 16:18 ... \Metal foam database analysis tool.m 5 of 18

```

table1 = fetch(conn,query1);

table2 = fetch(conn,query2);

index = fetch(conn,query3);

foamtype = fetch(conn,query4);

method = fetch(conn,query5);

description = fetch(conn,query6);

references = fetch(conn,query7);

filtertable = fetch(conn,query8);

% Changes coumn names in table to allow for "innerjoin"
table1.Properties.VariableNames{2} = 'variable1';
table1.Properties.VariableNames{3} = 'unit_variable1';

table2.Properties.VariableNames{2} = 'variable2';
table2.Properties.VariableNames{3} = 'unit_variable2';

filtertable.Properties.VariableNames{2} = 'filter_variable';
filtertable.Properties.VariableNames{3} = 'unit_filter';

foamtype.Properties.VariableNames{2} = 'foam_type';

method.Properties.VariableNames{2} = 'method';

description.Properties.VariableNames{2} = 'description';

% Closes connection
close(conn)

clear conn query
end

%% Creates full table of data Creates unit table for conversion of units

function [join1,join2,join3,join4,join5,join6,nrow] = jointables(table1,table2,index, \
foamtype,method,description,references)

%Joins idividual tables to create full data table (join2)
join1 = innerjoin(table1,table2);
join3 = innerjoin(index,foamtype);
join4 = innerjoin(join3,method);
join5 = innerjoin(join4,description);
join6 = innerjoin(join5,references);

```

Fig. 25: Full MATLAB script (5/15)

16/04/20 16:18 ... \Metal foam database analysis tool.m 6 of 18

```

join2 = innerjoin(join1,join6) %prints initial table in command window

nrow = size(join2,1); % Depth of table
end

%% Creates unit conversion table for conversion of units

function [unit_table] = makeunittable
%matlab does not allow use of special characters in an array name therefore
%alternatives have been chosen
%original vales are shown commented

rownames = ["Pa";"MPa";"GPa";"percent";"decimal";"um";"mm";"cm";"m";"g_cm3";"kg_m3";"W";
"kW";"m2";"pores_cm";"pores_inch";"W_mK";"one_m";"one_ft"];

n = 19; %number of units in the unit table

% Makes arrays for table
Pa = zeros(n,1);
MPa = zeros(n,1);
GPa = zeros(n,1);
percent = zeros(n,1); %'
decimal = zeros(n,1); %'' (empty)
um = zeros(n,1);
mm = zeros(n,1);
cm = zeros(n,1);
m = zeros(n,1);
g_cm3 = zeros(n,1); %g/cm^3
kg_m3 = zeros(n,1); %kg/m^3
W = zeros(n,1);
kW = zeros(n,1);
m2 = zeros(n,1); %m^2
pores_cm = zeros(n,1); %pores/cm
pores_inch = zeros(n,1); %pores/inch
W_mK = zeros(n,1); %W/mK
one_m = zeros(n,1); %1/m
one_ft = zeros(n,1); %1/ft

%Joins arrays into table of zeros
unit_table = table(Pa,MPa,GPa,percent,decimal,um,mm,cm,m,g_cm3,kg_m3,W,kW,m2,
pores_cm,pores_inch,W_mK,one_m,one_ft);
unit_table.Properties.RowNames = rownames;

%Assigns conversion values at the intersection of units
%The unit on the row name is the recorded unit and column name is the
%desired unit
unit_table.Pa(1) = 1;
unit_table.Pa(2) = 1000;
unit_table.Pa(3) = 1e06;

```

Fig. 26: Full MATLAB script (6/15)

16/04/20 16:18 ...\\Metal foam database analysis tool.m 7 of 18

```

unit_table.MPa(1) = 0.001;
unit_table.MPa(2) = 1;
unit_table.MPa(3) = 1000;
unit_table.GPa(1) = 0.000001;
unit_table.GPa(2) = 0.001;
unit_table.GPa(3) = 1;
unit_table.percent(4) = 1;
unit_table.percent(5) = 100;
unit_table.decimal(4) = 0.01;
unit_table.decimal(5) = 1;
unit_table.um(6) = 1;
unit_table.um(7) = 1000;
unit_table.um(8) = 10000;
unit_table.um(9) = 1e06;
unit_table.mm(6) = 0.001;
unit_table.mm(7) = 1;
unit_table.mm(8) = 10;
unit_table.mm(9) = 1000;
unit_table.cm(6) = 1e-04;
unit_table.cm(7) = 0.1;
unit_table.cm(8) = 1;
unit_table.cm(9) = 100;
unit_table.m(6) = 1e-6;
unit_table.m(7) = 0.001;
unit_table.m(8) = 0.01;
unit_table.m(9) = 1;
unit_table.g_cm3(10) = 1;
unit_table.g_cm3(11) = 0.001;
unit_table.kg_m3(10) = 1000;
unit_table.kg_m3(11) = 1;
unit_table.W(12) = 1;
unit_table.W(13) = 1000;
unit_table.kW(12) = 0.001;
unit_table.kW(13) = 1;
unit_table.m2(14) = 1;
unit_table.pores_cm(15) = 1;
unit_table.pores_cm(16) = 0.393701;
unit_table.pores_inch(15) = 2.54;
unit_table.pores_inch(16) = 1;
unit_table.W_mK(17) = 1;
unit_table.one_m(18) = 1;
unit_table.one_m(19) = 3.28084;
unit_table.one_ft(18) = 0.30480;
unit_table.one_ft(19) = 1;

end

%% Converts units from reported unit to desired unit

```

Fig. 27: Full MATLAB script (7/15)

16/04/20 16:18 ...\\Metal foam database analysis tool.m 8 of 18

```
function [join2,nrow] = convertunit(join2,variable1,variable2,unit1,unit2,unit_table, nrow)

    %Changes recorded unit to match unittable
    if strcmpi('densification strain',variable1) || strcmpi('porosity',variable1) || strcmpi('elastic Poisson ratio',variable1) || strcmpi('plastic Poisson ratio',variable1)
        for i = 1:nrow
            if strcmpi('%',join2.unit_variable1{i})
                join2.unit_variable1{i} = 'percent';
            else
                join2.unit_variable1{i} = 'decimal';
            end
        end
    end
    if strcmpi('densification strain',variable2) || strcmpi('porosity',variable2) || strcmpi('elastic Poisson ratio',variable2) || strcmpi('plastic Poisson ratio',variable2)
        for i = 1:nrow
            if strcmpi('%',join2.unit_variable2{i})
                join2.unit_variable2{i} = 'percent';
            else
                join2.unit_variable2{i} = 'decimal';
            end
        end
    end

    %Fixes error in recording unit from CSV file to SQL database
    if strcmpi('bulk density',variable1)
        for i = 1:nrow
            if strcmpi('g/cm<sup>3</sup>',join2.unit_variable1{i})
                join2.unit_variable1{i} = 'g_cm3';
            elseif strcmpi('kg/m<sup>3</sup>',join2.unit_variable1{i})
                join2.unit_variable1{i} = 'kg_m3';
            end
        end
    end
    if strcmpi('bulk density',variable2)
        for i = 1:nrow
            if strcmpi('g/cm<sup>3</sup>',join2.unit_variable2{i})
                join2.unit_variable2{i} = 'g_cm3';
            elseif strcmpi('kg/m<sup>3</sup>',join2.unit_variable2{i})
                join2.unit_variable2{i} = 'kg_m3';
            end
        end
    end

    %Fixes error in recording unit from google sheets to SQL database
    if strcmpi('permeability',variable1)
```

Fig. 28: Full MATLAB script (8/15)

16/04/20 16:18 ... \Metal foam database analysis tool.m 9 of 18

```

    for i = 1:nrow
        if strcmpi('m<sup>2</sup>',join2.unit_variable1{i})
            join2.unit_variable1{i} = 'm2';
        end
    end
end
if strcmpi('permeability',variable2)
    for i = 1:nrow
        if strcmpi('m<sup>2</sup>',join2.unit_variable2{i})
            join2.unit_variable2{i} = 'm2';
        end
    end
end

%Changes recorded unit to match unittable
if strcmpi('thermal conductivity',variable1)
    for i = 1:nrow
        if strcmpi('W/mK',join2.unit_variable1{i})
            join2.unit_variable1{i} = 'W_mK';
        end
    end
end
if strcmpi('thermal conductivity',variable2)
    for i = 1:nrow
        if strcmpi('W/mK',join2.unit_variable2{i})
            join2.unit_variable2{i} = 'W_mK';
        end
    end
end

%Changes recorded unit to match unittable
if strcmpi('Forchheimer factor',variable1)
    for i = 1:nrow
        if strcmpi('1/m',join2.unit_variable1{i})
            join2.unit_variable1{i} = 'one_m';
        elseif strcmpi('1/ft',join2.unit_variable1{i})
            join2.unit_variable1{i} = 'one_ft';
        end
    end
end
if strcmpi('Forchheimer factor',variable2)
    for i = 1:nrow
        if strcmpi('1/m',join2.unit_variable2{i})
            join2.unit_variable2{i} = 'one_m';
        elseif strcmpi('1/ft',join2.unit_variable2{i})
            join2.unit_variable2{i} = 'one_ft';
        end
    end
end
end

```

Fig. 29: Full MATLAB script (9/15)

16/04/20 16:18 ...\\Metal foam database analysis tool.m 10 of 18

```

%Changes recorded unit to match unittable
if strcmpi('pores per unit length',variable1)
    for i = 1:nrow
        if strcmpi('pores/cm',join2.unit_variable1{i})
            join2.unit_variable1{i} = 'pores_cm';
        elseif strcmpi('pores/inch',join2.unit_variable1{i})
            join2.unit_variable1{i} = 'pores_inch';
        end
    end
end
if strcmpi('pores per unit length',variable2)
    for i = 1:nrow
        if strcmpi('pores/cm',join2.unit_variable2{i})
            join2.unit_variable2{i} = 'pores_cm';
        elseif strcmpi('pores/inch',join2.unit_variable2{i})
            join2.unit_variable2{i} = 'pores_inch';
        end
    end
end

%Deletes any rows with no units (due to error in reporting)
unit_list = [];
for i = 1:nrow % iterate for every row of table
    k = strcmpi('',join2.unit_variable1{i}); % check if unit_variable1 is empty
    l = strcmpi('',join2.unit_variable2{i}); % check if unit_variable2 is empty
    if k == 1 || l == 1 % If either is empty note row number in
delete list
        unit_list = [unit_list,i];
    end
end
join2([unit_list],:) = []; % Delete all rows on deletelist
nrow = size(join2,1); % Find new table depth

%Converts from recorded unit to desired units (unit1,unit2) using
%unittable
for i = 1:nrow
    %Variable1
    recorded_unit1 = join2.unit_variable1{i}; %Reads reported unit
    scalar1 = unit_table{recorded_unit1,unit1}; %Finds unit conversion rate
    join2.variable1(i) = join2.variable1(i)*scalar1; %Multiplies mean value by
unit conversion rate
    join2.unit_variable1{i} = unit1; %Updates unit column

    %Variable2
    recorded_unit2 = join2.unit_variable2{i};
    scalar2 = unit_table{recorded_unit2,unit2};

```

Fig. 30: Full MATLAB script (10/15)

16/04/20 16:18 ...\\Metal foam database analysis tool.m 11 of 18

```

        join2.variable2(i) = join2.variable2(i)*scalar2;
        join2.unit_variable2{i} = unit2;
    end

    %Removes last two characters from label column so that values are the
    %same for the same study
    for i = 1:nrow
        join2.label{i}=join2.label{i}(1:6);
    end
end

%% Removes unwanted metals from the table

function [join2,nrow] = removemetals(join2,nrow,metals)

    %Finds number of metals to filter by
    metalsize = size(metals,1);

    metal_list = [];
    for i = 1:nrow % iterate for every row of table
        trigger = 0;
        for t = 1:metalsize % iterate for each metal
            k = strcmpi(metals{t},join2.base_material{i}); % check if base material
            matches item(t) on list
            if k == 1 % If it does set trigger to 1
                trigger = 1;
            end % Else trigger remains the same
        end
        if trigger == 0 % If metal matched nothing of list note row
            number in deletelist
            metal_list = [metal_list,i];
        end
    end

    join2([metal_list],:) = []; % Delete all rows on deletelist

    nrow = size(join2,1); % Find new table depth

end

%% Removes unwanted cell types from the table

function [join2,nrow] = removecells(join2,nrow,setcelltype)

    %Assigns desired cell type based off "setcelltype"
    if setcelltype == 1
        celltype = 'Open cell';
    elseif setcelltype == 2
        celltype = 'Closed cell';
    end

```

Fig. 31: Full MATLAB script (11/15)

16/04/20 16:18 ... \Metal foam database analysis tool.m 12 of 18

```

end

cell_list = [];
for i = 1:nrow % iterate for every row of table
    k = strcmpi(celltype,join2.foam_type{i}); % check if metal matches item(t)
on list
    if k == 0 % If metal matched nothing of list note row number
in deletelist
        cell_list = [cell_list,i];
    end
end

join2([cell_list],:) = []; % Delete all rows on deletelist

nrow = size(join2,1); % Find new table depth

end

%% Runs each function for the filter variable

function [join2,filtertable,nrow] = numericalfilter(join2,unit_table,filterrange,
filtertable,filtervariable,filterunit)
    %Performs operation from jointables, convertunit, and pretty unit but
    %for the filter variable
    %For more in depth comments refer to earlier functions

    frow = size(filtertable,1); % Depth of filtertable

    %Changes recorded unit to match unittable
    if strcmpi('densification strain',filtervariable) || strcmpi('porosity',
filtervariable) || strcmpi('elastic Poisson ratio',filtervariable) || strcmpi(
('plastic Poisson ratio',filtervariable)
        for i = 1:frow
            if strcmpi('%',filtertable.unit_filter{i})
                filtertable.unit_filter{i} = 'percent';
            else
                filtertable.unit_filter{i} = 'decimal';
            end
        end
    end

    %Fixes error in recording unit from CSV file to SQL database
    if strcmpi('bulk density',filtervariable)
        for i = 1:frow
            if strcmpi('g/cm<sup>3</sup>',filtertable.unit_filter{i})
                filtertable.unit_filter{i} = 'g_cm3';
            elseif strcmpi('kg/m<sup>3</sup>',filtertable.unit_filter{i})
                filtertable.unit_filter{i} = 'kg_m3';
            end
        end
    end
end

```

Fig. 32: Full MATLAB script (12/15)

16/04/20 16:18 ... \Metal foam database analysis tool.m 13 of 18

```

        end
    end

    %Fixes error in recording unit from google sheets to SQL database
    if strcmpi('permeability',filtervariable)
        for i = 1:frow
            if strcmpi('m<sup>2</sup>',filtertable.unit_filter{i})
                filtertable.unit_filter{i} = 'm2';
            end
        end
    end

    %Changes recorded unit to match unittable
    if strcmpi('thermal conductivity',filtervariable)
        for i = 1:frow
            if strcmpi('W/mK',filtertable.unit_filter{i})
                filtertable.unit_filter{i} = 'W_mK';
            end
        end
    end

    %Changes recorded unit to match unittable
    if strcmpi('Forchheimer factor',filtervariable)
        for i = 1:frow
            if strcmpi('1/m',filtertable.unit_filter{i})
                filtertable.unit_filter{i} = 'one_m';
            elseif strcmpi('1/ft',filtertable.unit_filter{i})
                filtertable.unit_filter{i} = 'one_ft';
            end
        end
    end

    %Changes recorded unit to match unittable
    if strcmpi('pores per unit length',filtervariable)
        for i = 1:frow
            if strcmpi('pores/cm',filtertable.unit_filter{i})
                filtertable.unit_filter{i} = 'pores_cm';
            elseif strcmpi('pores/inch',filtertable.unit_filter{i})
                filtertable.unit_filter{i} = 'pores_inch';
            end
        end
    end

    %Deletes any rows with no units (due to error in reporting)
    unit_list = [];
    for i = 1:frow % iterate for every row of table
        k = strcmpi('',filtertable.unit_filter{i}); % check if filter_unit is empty
        (== '')
        if k == 1 % If empty note row number in delete list
    end
end

```

Fig. 33: Full MATLAB script (13/15)

16/04/20 16:18 ... \Metal foam database analysis tool.m 14 of 18

```

        unit_list = [unit_list,i];
    end
end
filtertable([unit_list,:]) = [];      % Delete all rows on deletelist
frow = size(filtertable,1);           % Find new table depth

%Converts from recorded unit to desired units using unittable
for i = 1:frow
    %filtervariable
    recorded_unit3 = filtertable.unit_filter{i};
    scalar3 = unit_table{recorded_unit3,filterunit};
    filtertable.filter_variable(i) = filtertable.filter_variable(i)*scalar3;
    filtertable.unit_filter{i} = filterunit;
end

%Adds filtertable to join2
join2 = innerjoin(join2,filtertable);

nrow = size(join2,1);                 % Find new table depth

%Remove rows outside of filter range
num_list = [];
for i = 1:nrow                        % iterate for every row of table
    if join2.filter_variable(i) <= filterrange(1) || join2.filter_variable(i) >= filterrange(2)
        num_list = [num_list,i];
    end
end
join2([num_list,:]) = [];             % Delete all rows on deletelist

nrow = size(join2,1);                 % Find new table depth

%Fixes units of filtervariable to look nice
if strcmpi('percent',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = '%';
    end
elseif strcmpi('g_cm3',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'g/cm^3';
    end
elseif strcmpi('kg_m3',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'kg/m^3';
    end
elseif strcmpi('m2',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'm^2';
    end
end

```

Fig. 34: Full MATLAB script (14/15)

16/04/20 16:18 ...\\Metal foam database analysis tool.m 15 of 18

```

elseif strcmpi('pores_cm',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'pores/cm';
    end
elseif strcmpi('pores_inch',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'pores/in';
    end
elseif strcmpi('W_mK',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'W/mK';
    end
elseif strcmpi('one_m',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = '1/m';
    end
elseif strcmpi('one_ft',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = '1/ft';
    end
end

filtername = regexp(filtervariable, ' ', '_');
filterunit = ['unit_',filtername];

join2.Properties.VariableNames{12} = filtername;
join2.Properties.VariableNames{13} = filterunit;

end

%% Changes units back to display nicely in table

function [join2,printunit1,printunit2] = prettyunit(join2,unit1,unit2,nrow)

%Fixes unit1
if strcmpi('percent',unit1)
    printunit1 = '%';
    for i = 1:nrow
        join2.unit_variable1{i} = '%';
    end
elseif strcmpi('g_cm3',unit1)
    printunit1 = 'g/cm^3';
    for i = 1:nrow
        join2.unit_variable1{i} = 'g/cm^3';
    end
elseif strcmpi('kg_m3',unit1)
    printunit1 = 'kg/m^3';
    for i = 1:nrow
        join2.unit_variable1{i} = 'kg/m^3';
    end

```

Fig. 35: Full MATLAB script (15/15)

16/04/20 16:18 ...\\Metal foam database analysis tool.m 15 of 18

```

elseif strcmpi('pores_cm',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'pores/cm';
    end
elseif strcmpi('pores_inch',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'pores/in';
    end
elseif strcmpi('W_mK',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'W/mK';
    end
elseif strcmpi('one_m',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = '1/m';
    end
elseif strcmpi('one_ft',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = '1/ft';
    end
end

filtername = regexp(filtervariable, ' ', '_');
filterunit = ['unit_',filtername];

join2.Properties.VariableNames{12} = filtername;
join2.Properties.VariableNames{13} = filterunit;

end

%% Changes units back to display nicely in table

function [join2,printunit1,printunit2] = prettyunit(join2,unit1,unit2,nrow)

%Fixes unit1
if strcmpi('percent',unit1)
    printunit1 = '%';
    for i = 1:nrow
        join2.unit_variable1{i} = '%';
    end
elseif strcmpi('g_cm3',unit1)
    printunit1 = 'g/cm^3';
    for i = 1:nrow
        join2.unit_variable1{i} = 'g/cm^3';
    end
elseif strcmpi('kg_m3',unit1)
    printunit1 = 'kg/m^3';
    for i = 1:nrow
        join2.unit_variable1{i} = 'kg/m^3';
    end

```

Fig. 36: Full MATLAB script (16/18)

16/04/20 16:18 ...\\Metal foam database analysis tool.m 15 of 18

```

elseif strcmpi('pores_cm',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'pores/cm';
    end
elseif strcmpi('pores_inch',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'pores/in';
    end
elseif strcmpi('W_mK',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'W/mK';
    end
elseif strcmpi('one_m',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = '1/m';
    end
elseif strcmpi('one_ft',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = '1/ft';
    end
end

filtername = regexp(filtervariable, ' ', '_');
filterunit = ['unit_',filtername];

join2.Properties.VariableNames{12} = filtername;
join2.Properties.VariableNames{13} = filterunit;

end

%% Changes units back to display nicely in table

function [join2,printunit1,printunit2] = prettyunit(join2,unit1,unit2,nrow)

%Fixes unit1
if strcmpi('percent',unit1)
    printunit1 = '%';
    for i = 1:nrow
        join2.unit_variable1{i} = '%';
    end
elseif strcmpi('g_cm3',unit1)
    printunit1 = 'g/cm^3';
    for i = 1:nrow
        join2.unit_variable1{i} = 'g/cm^3';
    end
elseif strcmpi('kg_m3',unit1)
    printunit1 = 'kg/m^3';
    for i = 1:nrow
        join2.unit_variable1{i} = 'kg/m^3';
    end

```

Fig. 37: Full MATLAB script (17/18)

16/04/20 16:18 ...\\Metal foam database analysis tool.m 15 of 18

```

elseif strcmpi('pores_cm',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'pores/cm';
    end
elseif strcmpi('pores_inch',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'pores/in';
    end
elseif strcmpi('W_mK',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = 'W/mK';
    end
elseif strcmpi('one_m',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = '1/m';
    end
elseif strcmpi('one_ft',filtervariable)
    for i = 1:nrow
        join2.unit_filter{i} = '1/ft';
    end
end

filtername = regexp(filtervariable, ' ', '_');
filterunit = ['unit_',filtername];

join2.Properties.VariableNames{12} = filtername;
join2.Properties.VariableNames{13} = filterunit;

end

%% Changes units back to display nicely in table

function [join2,printunit1,printunit2] = prettyunit(join2,unit1,unit2,nrow)

%Fixes unit1
if strcmpi('percent',unit1)
    printunit1 = '%';
    for i = 1:nrow
        join2.unit_variable1{i} = '%';
    end
elseif strcmpi('g_cm3',unit1)
    printunit1 = 'g/cm^3';
    for i = 1:nrow
        join2.unit_variable1{i} = 'g/cm^3';
    end
elseif strcmpi('kg_m3',unit1)
    printunit1 = 'kg/m^3';
    for i = 1:nrow
        join2.unit_variable1{i} = 'kg/m^3';
    end

```

Fig. 38: Full MATLAB script (18/18)

	A	B	C	D	E	F	G	H	I	J	K
mf_id	porosity	unit_porosity	permeability	unit_permeability	base_material	foam_type	method	description	label	link	
1	80	95 %	0.0000003	m ^{√2}	Stainless Steel	Open cell	Polymer precursor - 4.9	Cells take on whatever c	Ad2004	https://www.sciencedirect.com/science/article/pii/S0950268804000087	
2	81	95 %	0.000000087	m ^{√2}	Stainless Steel	Open cell	Polymer precursor - 4.9	Cells take on whatever c	Ad2004	https://www.sciencedirect.com/science/article/pii/S0950268804000087	
3	82	95 %	0.000000002	m ^{√2}	Stainless Steel	Open cell	Polymer precursor - 4.9	Cells take on whatever c	Ad2004	https://www.sciencedirect.com/science/article/pii/S0950268804000087	
4	83	95 %	5.5E-09	m ^{√2}	Stainless Steel	Open cell	Polymer precursor - 4.9	Cells take on whatever c	Ad2004	https://www.sciencedirect.com/science/article/pii/S0950268804000087	
5	84	95 %	2.2E-09	m ^{√2}	Stainless Steel	Open cell	Polymer precursor - 4.9	Cells take on whatever c	Ad2004	https://www.sciencedirect.com/science/article/pii/S0950268804000087	
6	100	51.5 %	4.93E-11	m ^{√2}	Titanium	Open cell	Powder metallurgical process using the space hol	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1		
7	101	54.3 %	6.11E-11	m ^{√2}	Titanium	Open cell	Powder metallurgical pr	Titanium foam produced	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1	
8	102	57.7 %	7.42E-11	m ^{√2}	Titanium	Open cell	Powder metallurgical pr	Titanium foam produced	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1	
9	103	59.9 %	1.39E-10	m ^{√2}	Titanium	Open cell	Powder metallurgical process using the space hol	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1		
10	104	62.5 %	2.4E-10	m ^{√2}	Titanium	Open cell	Powder metallurgical pr	Titanium foam produced	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1	
11	104	62.5 %	2.4E-10	m ^{√2}	Titanium	Open cell	Powder metallurgical pr	Titanium foam produced	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1	
12	105	65 %	2.33E-10	m ^{√2}	Titanium	Open cell	Powder metallurgical pr	Titanium foam produced	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1	
13	106	67.4 %	4.84E-10	m ^{√2}	Titanium	Open cell	Powder metallurgical pr	Titanium foam produced	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1	
14	107	70.6 %	6.84E-10	m ^{√2}	Titanium	Open cell	Powder metallurgical pr	Titanium foam produced	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1	
15	108	73.6 %	8.55E-10	m ^{√2}	Titanium	Open cell	Powder metallurgical pr	Titanium foam produced	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1	
16	109	76.2 %	1.14E-09	m ^{√2}	Titanium	Open cell	Powder metallurgical pr	Titanium foam produced	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1	
17	110	78.7 %	1.35E-09	m ^{√2}	Titanium	Open cell	Powder metallurgical pr	Titanium foam produced	Im2007	<a href="https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1">https://onlinelibrary.wiley.com/doi/10.1002/1522-2675(200705)10:5<104::AID-MATL104>3.0.CO;2-1	
18	281	97.26 %	0.000000027	m ^{√2}	Aluminium	Open cell	open-cell structure comf	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
19	282	91.18 %	0.000000018	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
20	283	94.86 %	0.000000012	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
21	284	91.38 %	0.000000011	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
22	285	89.91 %	9.40E-08	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
23	286	95.46 %	0.000000013	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
24	287	92.45 %	0.000000011	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
25	288	90.05 %	9.00E-08	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
26	289	96.59 %	5.50E-08	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
27	290	92.72 %	6.10E-08	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
28	291	91.32 %	5.30E-08	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
29	292	97.1 %	0.0000000252	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
30	293	94.6 %	0.0000000217	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
31	294	90.5 %	0.0000000174	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
32	295	94.9 %	0.0000000149	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
33	296	90.9 %	0.0000000111	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		
34	297	97.8 %	0.0000000142	m ^{√2}	Aluminium	Open cell	open-celled structure co	Bh2002	https://doi.org/10.1016/j.matpr.2020.04.010		

Fig. 39: Screenshot of exported excel table

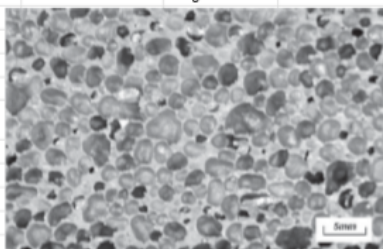
				Image			
Identifier	0320-AI						
Trade name	Alporas						
Base material	Aluminium						
Purity							
Foam type	Closed cell						
Manufacturing							
Method	batch casting process						
Geometry							
Description	AL-Ca5-Ti3 matrix. closed cell, varying cell size.						
	mean	std					
Porosity	84%						
Bulk density	0.43 g/cm^3						
Pores per unit length	pores/cm						
Average pore size	4 mm						
Mechanical							
Elastic properties:	mean	std	Number of measurements	Temperature			
Young modulus	GPa				C		
	GPa				C		
	GPa				C		
	GPa				C		
Elastic Poisson ratio							
Plastic properties:	mean	std			Temperature		
Yield stress	MPa				C		
	MPa				C		
	MPa				C		
	MPa				C		
	mean	std	Number of measurements				
Plastic Poisson ratio	0.22						
Plateau stress	MPa						
Densification strain							
Tensile failure strain							
Shear failure strain							
Fluidic							

Fig. 40: Screenshot of Full Google sheet format (1/2)

	mean	std	Number of measurements						
Permeability	m ²								
Forchheimer factor									
Thermal									
	mean	std	Number of measurements						
Thermal conductivity	W/mK								
Heat transport									
Stress-Strain	Strain rate	0.001 1/s							
Engineering strain	Engineering stress								
0	0 MPa								
0.0016	0.858 MPa								
0.0116	1.18 MPa								
0.025	1.23 MPa								
0.0464	1.06 MPa								
0.0625	1.12 MPa								
0.0705	1.19 MPa								
0.0954	1.4 MPa								
0.117	1.45 MPa								
0.136	1.32 MPa								
0.179	1.57 MPa								
0.225	1.41 MPa								
0.265	1.5 MPa								
0.299	1.8 MPa								
0.348	1.62 MPa								
0.428	2.03 MPa								
0.478	2.19 MPa								
0.526	2.67 MPa								
0.574	3.45 MPa								
0.615	4.37 MPa								
0.657	5.67 MPa								
0.686	7.39 MPa								
References									
Authors - (last name and initials)	Title	Year	Journal				Link		
Deshpande V.S., Fleck N.A.	Isotropic constitutive models for metallic foams	1999	Journal of the Mechanics and Physics of Solids				https://doi.org/10.1016/S0022-5096(99)00082-4		

Fig. 41: Screenshot of Full Google sheet format (2/2)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	Pa	MPa	GPa	percent	decimal	um	mm	cm	m	g_cm3	kg_m3	W	kW	m2	pores_cm	pores_inch	W_mK	one_m	one_ft
1 Pa	1	1.0000e-03	1.0000e-06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2 MPa	1000	1	1.0000e-03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3 GPa	1000000	1000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4 percent	0	0	0	1	0.0100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5 decimal	0	0	0	100	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6 um	0	0	0	0	0	1	1.0000e-03	1.0000e-04	1.0000e-06	0	0	0	0	0	0	0	0	0	0
7 mm	0	0	0	0	0	1000	1	0.1000	1.0000e-03	0	0	0	0	0	0	0	0	0	0
8 cm	0	0	0	0	0	10000	10	1	0.0100	0	0	0	0	0	0	0	0	0	0
9 m	0	0	0	0	0	1000000	1000	100	1	0	0	0	0	0	0	0	0	0	0
10 g_cm3	0	0	0	0	0	0	0	0	0	1	1000	0	0	0	0	0	0	0	0
11 kg_m3	0	0	0	0	0	0	0	0	0	1.0000e-03	1	0	0	0	0	0	0	0	0
12 W	0	0	0	0	0	0	0	0	0	0	0	1	1.0000e-03	0	0	0	0	0	0
13 kW	0	0	0	0	0	0	0	0	0	0	0	1000	1	0	0	0	0	0	0
14 m2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
15 pores_cm	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2.5400	0	0	0
16 pores_inch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3937	1	0	0	0
17 W_mK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
18 one_m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.3048
19 one_ft	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3.2808	1

Fig. 42: Full unit conversion table