Orriginal author: Matthew McMillan, DATE: 07/06/18

description

This program takes a variety of inputs and writes a lattice file in various output formats for use in other programs. The use of the plgBatch file simplifies the creation of a large number of lattice(s). Example implementations is present in the complex example file.

Methods

The following is a list of methods and a quick overview of their function, it should obvious from the name what it is they do.

- PLG: Creates a PLG object, 1 input loads a custom file.
- set: used to set properties in the PLG using name value pairs
- defineUnit: used to define the individual unit cell
- cellReplication: replicates a unit cell based on inputs
- cleanLattice: removes duplicate vertices or faces and orders the vertices in Z
- scale: resize the existing structure
- translate: translates the lattice structure.
- rotate: rotates the lattice structure
- plus: combines an existing PLG object into another plg object usefull for generating complex lattice shapes
- plot: displays a rendering of the beam model that represents the lattice.
- save: this method is in its own method group and each sub save method will be named according to its save out type. Eg saveStl saves out stl format.

Properties

properties of the PLG are defiend using the set method to ensure that only trhe correct type can be used.

```
obj = set(obj,'propertyName',value)
```

- resolution - resolution of the struts - scaler integer
- strutDiameter - strut diameter - scaler float
- sphereAddition - determines wheter to add spheres to the structure - logical (true or false)
- sphereResolution - required if sphereAddition is true
- sphereDiameter - required if sphereAddition is true
- baseFlat - flattens the spheres at minimum z height vertices to create a flat base for supports - logical (true or false) - does nothing if sphereAddition is false
- unitSize - specifies the size of a single unit in a lattice - 3x1 vector of floats
- replications - specifies the number of copies of the unit cell - 3x1 vector of integers
- origin starting location for the centre of the initial unit cell - 3x1 vector of floats - default value is [0,0,0]

generating a unit cell

See subfolder unitCell in the PLG code

addSupport

This class is a submethod of the PLG and enables the addition of support pins. it is not intended as a lattice Generating code but instead will load a custom file saved out from the regular PLG. once this is done the following functions can be used:

- addSupport - takes a custom file, support strutDiameter, support sphereDiameter, critical incline and search range(as a percentage from the base up).
- padSupport - extends supports a defined a distance below the minimum z.

Examples

A 3x4x5 BCC lattice with x struts, a 0.3mm strut diameter, 4mm unit cell and 0.5mm ball diameter with its origin moved by 6,7,8 and then saved as a stl (12 facet resolution) and 3mf file with a resolution of 30. See complex example for the use of translation, rotation and plus.

```
obj = PLG();
obj = set(obj,'resolution',12);
obj = set(obj,'strutDiameter',0.3);
obj = set(obj,'unitSize',[4,4,4]);
obj = set(obj,'sphereAddition',true);
obj = set(obj,'baseFlat',true);
obj = set(obj,'sphereResolution',12);
obj = set(obj,'sphereDiameter',0.5);
obj = set(obj,'origin',[6,7,8]);
obj = set(obj,'replications',[3,4,5]);
obj = defineUnit(obj,{'bcc','xRods'});
obj = cellReplication(obj);
obj = cleanLattice(obj);
saveStl(obj,'exampleOut.stl');

obj = set(obj,'resolution',30);
obj = set(obj,'sphereResolution',30);
save3mf(obj,'exampleOut.3mf');
```

SubClass splitStrut

Enables splitting of a bad custom file where beam do interesect but this is not present in the file. splitStruts will identify these and split the beams in two.

how to use plgBatch

1. starting from top to bottom set all properties. The properties under **TestParameter** will undergo a full factorial design.
2. Scroll down to method and move the curser to the desired output style:

- runAllCombinations - runs through every single permutation of everything in **TestParameter**.
- squareUnitCell - uses only unitSizeX for all unit cell dimensions runs through all other **TestParameter**.
- squareLattice - uses only unitSizeX and repsX

3. click on run current test in the menu bar (ctrl+enter). Alternatively run the following command: `results = run(plgBatch,'squareLattice')`
4.

Be patient matlab internally calculates the full factorial before begining to generate files this may take a while depending on the number of outputs

make your own generation function

Generating your own function may be the most usefull for your use case.

1. create a function with the following format: `function functionNameHere(obj,desiredParameters)`

- functionNameHere - can be anything not already used and can not be plgBatch or PLG or any PLG functions you plan to call
- obj - the first input must be the class object itself. this is used to access any constant properties eg: obj.outputFolder
- desiredParameters - as seperate inputs place any variables in **TestParameter** that you wish to use. New parameters can also be added.

2. follow above instructions

note: depending on the number of inputs there can be a big lag between hitting run and the script generating data. Therefore it is recommended that you test a single output with a script before placing in this class.