DEMO_0002_SF_Multi_Morphology_TPMS

Table of Contents

This is a demo for:

- Building geometry for generating multi-morphology lattice of Gyroid-Diamond in cubic domain using Sigmoid Function (SF)

Name

Author: *Mahtab Vafaee*, mahtab.vafaee@gmail.com

```
Change log:
2023/11/15 MV Created
2024/01/31 MV Sorted for publishing
----------------------------------------------------------------------
```

```
clear; close all; clc;
```

Plot settings

```
fontSize=20;
faceAlpha1=0.8;
markerSize=10;
lineWidth1=3;
lineWidth2=4;
markerSize1=25;
```

Control parameters

```
sampleSize=2; %Heigh of the sample
pointSpacing=sampleSize/100; % Resolution

overSampleRatio=1;
numStepsLevelset=ceil(overSampleRatio.*(sampleSize./pointSpacing)); %Number of
 voxel steps across period for image data (roughly number of points on mesh
 period)
```

Set-up the input parameters for individual lattices

```
inputStruct_A.L=[4 2 2]; % characteristic length
inputStruct_A.Ns=numStepsLevelset; % number of sampling points
inputStruct_A.isocap=1; %Option to cap the isosurface
inputStruct_A.surfaceCase='g'; %Surface type

inputStruct_B = inputStruct_A;

% Set parameters for individual lattices
% Structure-A
inputStruct_A.numPeriods=[5 2 2]; %Number of periods in each direction
inputStruct_A.levelset=-0.1 ; %Isosurface level
inputStruct_A.gradiantF=0 ; %Gradiant Factor within individual structures
levelset_A=inputStruct_A.levelset;

% Structure-B
inputStruct_B.numPeriods=[6 3 3];
inputStruct_B.levelset=-0.4;
inputStruct_B.surfaceCase='d';
inputStruct_B.gradiantF=0 ; %Gradiant Factor within individual structures
levelset_B=inputStruct_B.levelset;
```

Compute individual gyroids

```
% No need to store faces and vertices, only require underlying S,
% grid coordinates, and levelset values
[~,~,~,S_A,X,Y,Z]=triplyPeriodicMinimalSurface(inputStruct_A);
[~,~,~,S_B,~,~,~]=triplyPeriodicMinimalSurface(inputStruct_B);
```

Define the central location of each individual lattices in space

E.g., At center_A, the structure will definitely correspond to input_A. As we move away from center_A, it will slowly transition into other structures with input_B.

```
center_A = [0.75, 0.5, 0.5];
center_B = [1.25, 0.5, 0.5];
```

Transition lengthscal and shape

kappa controls the lengthscale of transition between lattices Higher kappa => faster transition Lower kappa => slower transition

```
kappa = 5;

%Transition path (shape)
G = X;
```

```
G = G/max(G(:));
G = G-(max(G(:))/2);
```

Compute the weitgh functions

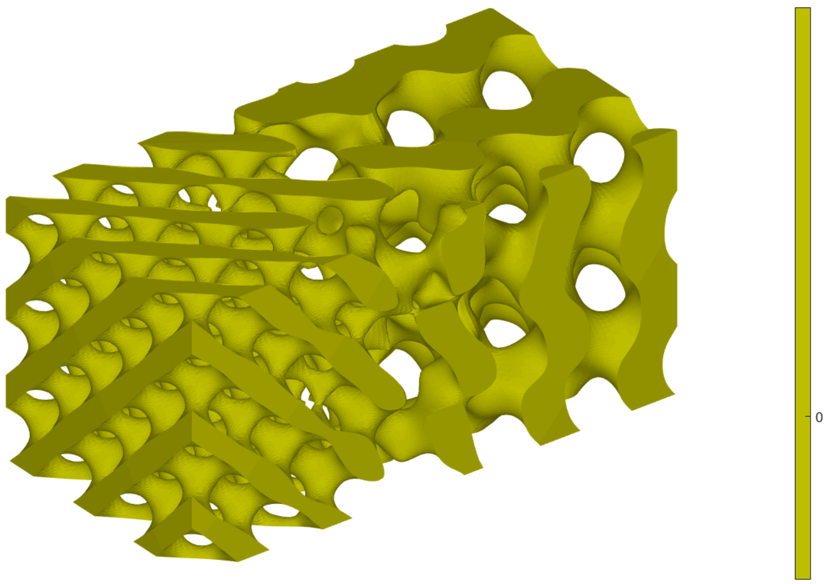
```
%Weight functions of each morphology
weights_A = 1/(1+exp(-kappa * G));
weights_B=(1-weights_A);

% Interpolating using the above weights
graded_S =  weights_A .* (S_A - levelset_A) ...
            + (weights_B).* (S_B - levelset_B);
```

Compue isosurface

```
graded_levelset = 0;

[f,v] = isosurface(X,Y,Z,graded_S,graded_levelset);
c=zeros(size(f,1),1);

% Compute isocaps
[fc,vc] = isocaps(X,Y,Z,graded_S,graded_levelset,'enclose','below');

% Boilerplate code for preparing output for exporting/visualization
nc=patchNormal(fc,vc);
cc=zeros(size(fc,1),1);
cc(nc(:,1)<-0.5)=1;
cc(nc(:,1)>0.5)=2;
cc(nc(:,2)<-0.5)=3;
cc(nc(:,2)>0.5)=4;
cc(nc(:,3)<-0.5)=5;
cc(nc(:,3)>0.5)=6;

% Join sets
[f,v,c]=joinElementSets({f,fc},{v,vc},{c,cc});

% Merge nodes
[f,v]=mergeVertices(f,v);

% Check for unique faces
[~,indUni,~]=unique(sort(f,2),'rows');
f=f(indUni,:); %Keep unique faces
c=c(indUni);

% Remove collapsed faces
[f,logicKeep]=patchRemoveCollapsed(f);
c=c(logicKeep);

% Remove unused points
[f,v]=patchCleanUnused(f,v);

% Invert faces
f=fliplr(f);
```

Visualize

```
Hybrid_vizualize(f,v,c);
```



Published with MATLAB® R2021b