DEMO_0003_Multi_Morphology_Spinodoid

Table of Contents

..... 1
Plot settings 1
Control parameters 1
Figure-3: Cubic Domain-SF Gyroid-Diamond Multi-Morphology 2
Compute individual gyroids 2
Define the central location of each individual spinodoid in space 2
Transition lengthscal and shape 2
Compute weight functions 3
Figure-4(d-f): Cubic Domain-GRBF Graded Spinodoid 3
Compue isosurface 4
Visualize 5

This is a demo for:

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

Name

License: to license

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=[3 1 1]; %Heigh of the sample

res=[300 100 100]; % set the resolution in 3D
```

Figure-3: Cubic Domain-SF Gyroid-Diamond Multi-Morphology

```
inputStruct.isocap=true;
inputStruct_A.domainSize=sampleSize;
inputStruct_A.resolution=res;
inputStruct_A.numWaves=1000;

inputStruct_B = inputStruct_A;

% Set parameters for individual gyroid
inputStruct_A.waveNumber=15*pi;
inputStruct_A.relativeDensity=0.3;
inputStruct_A.thetas=[15 15 0];
levelset_A=inputStruct_A.relativeDensity;

inputStruct_B.waveNumber=20*pi;
inputStruct_B.relativeDensity=0.5;
inputStruct_B.thetas=[90 90 0];
levelset_B=inputStruct_B.relativeDensity;
```

Compute individual gyroids

No need to store faces and vertices, only require underlying S, grid coordinates, and levelset values

```
[~,~,~,S_A,X,Y,Z]=spinodoid(inputStruct_A);
[~,~,~,S_B,~,~,~]=spinodoid(inputStruct_B);
```

Define the central location of each individual spinodoid in space

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

```
center_A = [0.5, 0.5, 0.5];
center_B = [1.5, 0.5, 0.5];
```

Transition lengthscal and shape

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

```
kappa = 15;

%%Transition path (shape)
G1 = X;
G1 = G1/max(G1(:)); % from 0-1
G1 = G1-(max(G1(:))/2); % from -1/2 to 1/2
```

```
G2 = Y;
G2 = G2/max(G2(:)); % from 0-1
G2 = G2-(max(G2(:))/2); % from -1/2 to 1/2

G = G1 - G2;
```

Compute weight 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);
```

Figure-4(d-f): Cubic Domain-GRBF Graded Spinodoid

```
inputStruct.isocap=true;
inputStruct_A.domainSize=sampleSize;
inputStruct_A.resolution=res;
inputStruct_A.numWaves=1000;
% isoLevel=inputStruct_A.levelset;

inputStruct_B = inputStruct_A;
inputStruct_C = inputStruct_A;

% Set parameters for individual gyroid
inputStruct_A.waveNumber=10*pi;
inputStruct_A.relativeDensity=0.3;
inputStruct_A.thetas=[15 15 0];
levelset_A=inputStruct_A.relativeDensity;

inputStruct_B.waveNumber=15*pi;
inputStruct_B.relativeDensity=0.7;
inputStruct_B.thetas=[90 90 0];
levelset_B=inputStruct_B.relativeDensity;

inputStruct_C.waveNumber=20*pi;
inputStruct_C.relativeDensity=0.4;
inputStruct_C.thetas=[0 0 15];
levelset_C=inputStruct_C.relativeDensity;

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

```
% Define the central location of each individual spinodoid in space
% E.g., At center_A, the spinodoid will definitely correspond to input_A.
% As we move away from center_A, it will slowly transition into other
% spinodoids with input_B and input_C.
center_A = [0.5, 0.5, 0.5];
center_B = [1.5, 0.5, 0.5];
center_C = [2.5, 0.5, 0.5];


% kappa controls the lengthscale of transition between spinodoids
% Higher kappa => faster transition
% Lower kappa => slower transition
kappa = 8;

% Using Gaussian (a.k.a. radial basis functions) interpolation.
% One can use any interpolation scheme of choice as long as weights at
% every grid point sum up to 1.
% Computing the weights for each spinodoid evaluated on all grid points.
weights_A = exp(-kappa * Squared_distance_from_point(X,Y,Z,center_A));
weights_B = exp(-kappa * Squared_distance_from_point(X,Y,Z,center_B));
weights_C = exp(-kappa * Squared_distance_from_point(X,Y,Z,center_C));

% Weights must sum up to 1.
sum_weights = weights_A + weights_B + weights_C;

weights_A = weights_A ./ sum_weights;
weights_B = weights_B ./ sum_weights;
weights_C = weights_C ./ sum_weights;

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

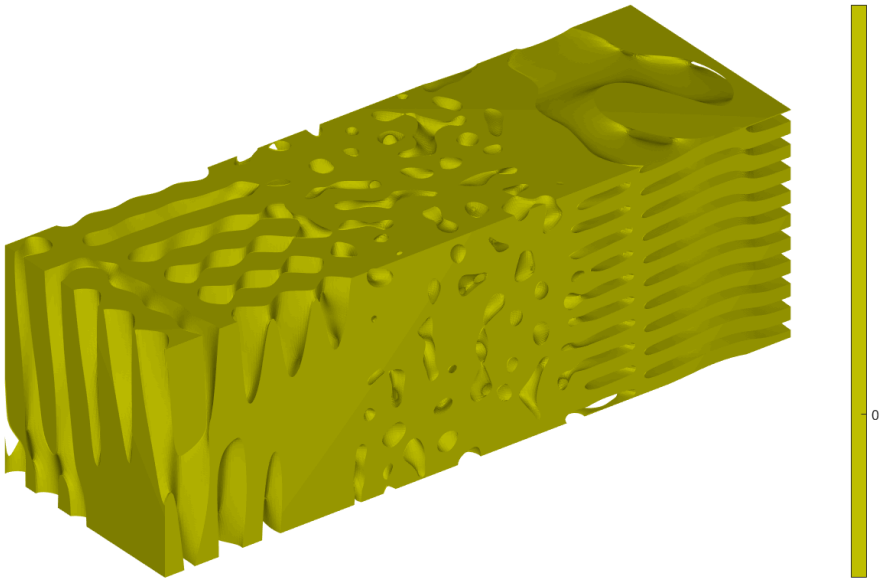
Compue isosurface

```
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