DEMO_0013_Mapping_Density_Distribution

Table of Contents

This is a demo for:

- Building a non-uniform infill lattice structure, to map a specific structural properties, e.g. in this demo, it is mapping a density distribution field within the domain.

Name

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

```
Change log:
2023/11/15 MV Created
2024/02/06 MV Edited
--------------------------------------------------------------------------
```

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

Plot settings

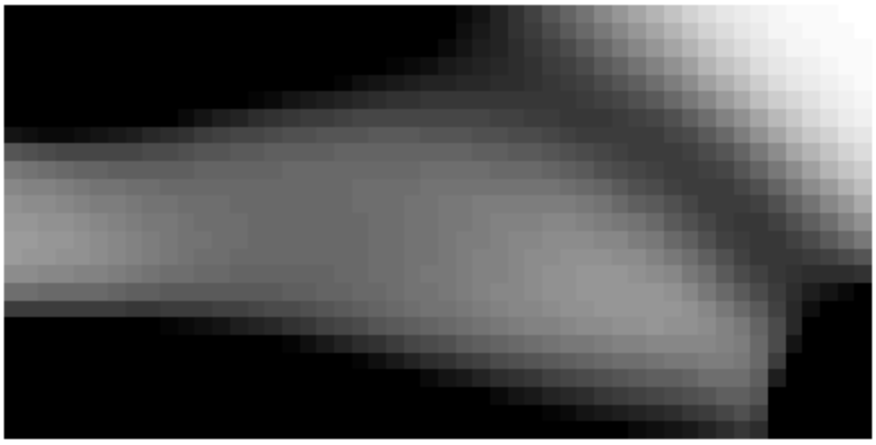
```
cMap=jet(250);
faceAlpha1=0.5;
faceAlpha2=0.65;
edgeColor1='none';
edgeColor2='none';
fontSize=25;
pColors=gjet(6);
```

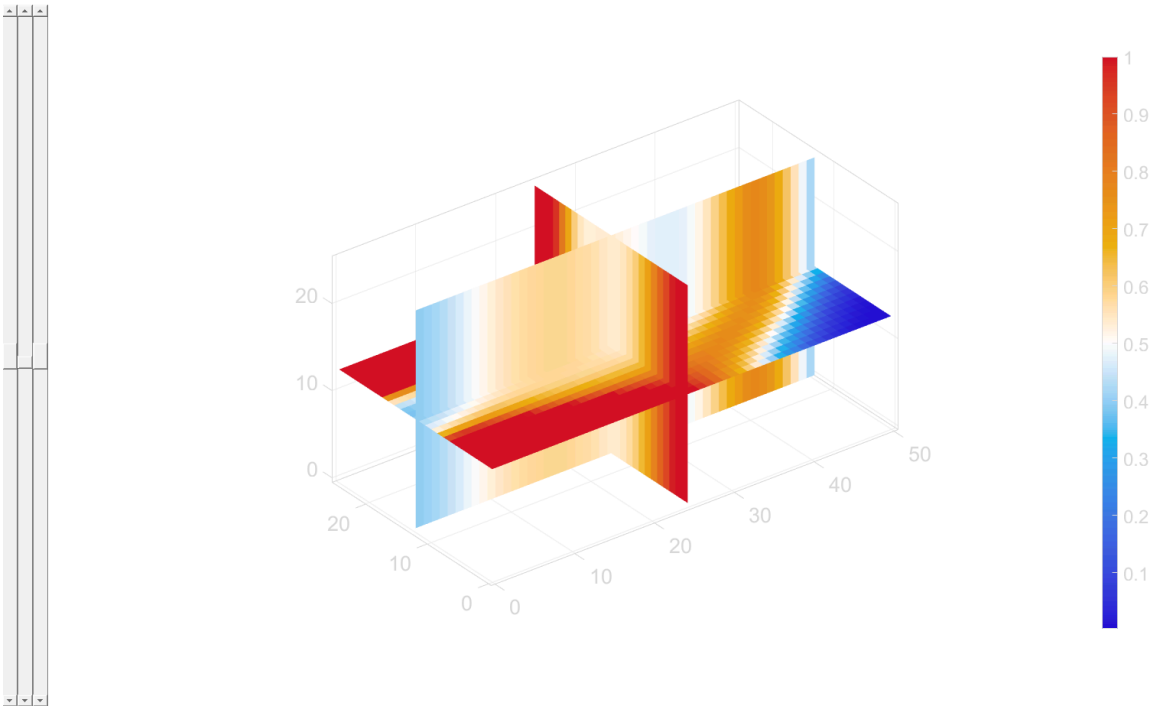
creating density field

```
% define box size, volume fraction as inputs for top.m
```

```
nelx = 50;
nely = 25;
nelz = 25;
volfrac = 0.7;
penal = 2;
rmin = 6;

% calculating the density distribution map
[rho0] = top(nelx,nely,volfrac,penal,rmin);

% extending through the z-direction
ext = [1, 1, nelz];
rho = repmat(rho0, ext);
sv3(rho); colormap warmcold; %Visualize 3D field
```

```
 It.:    1 Obj.:    98.6867 Vol.:  0.700 ch.:  0.200
 It.:    2 Obj.:    73.4967 Vol.:  0.700 ch.:  0.200
 It.:    3 Obj.:    65.6763 Vol.:  0.700 ch.:  0.200
 It.:    4 Obj.:    64.5356 Vol.:  0.700 ch.:  0.068
 It.:    5 Obj.:    64.0940 Vol.:  0.700 ch.:  0.040
 It.:    6 Obj.:    63.8484 Vol.:  0.700 ch.:  0.029
 It.:    7 Obj.:    63.6814 Vol.:  0.700 ch.:  0.022
 It.:    8 Obj.:    63.5642 Vol.:  0.700 ch.:  0.017
 It.:    9 Obj.:    63.5002 Vol.:  0.700 ch.:  0.013
 It.:   10 Obj.:    63.4375 Vol.:  0.700 ch.:  0.010
```





grid for Phi-field

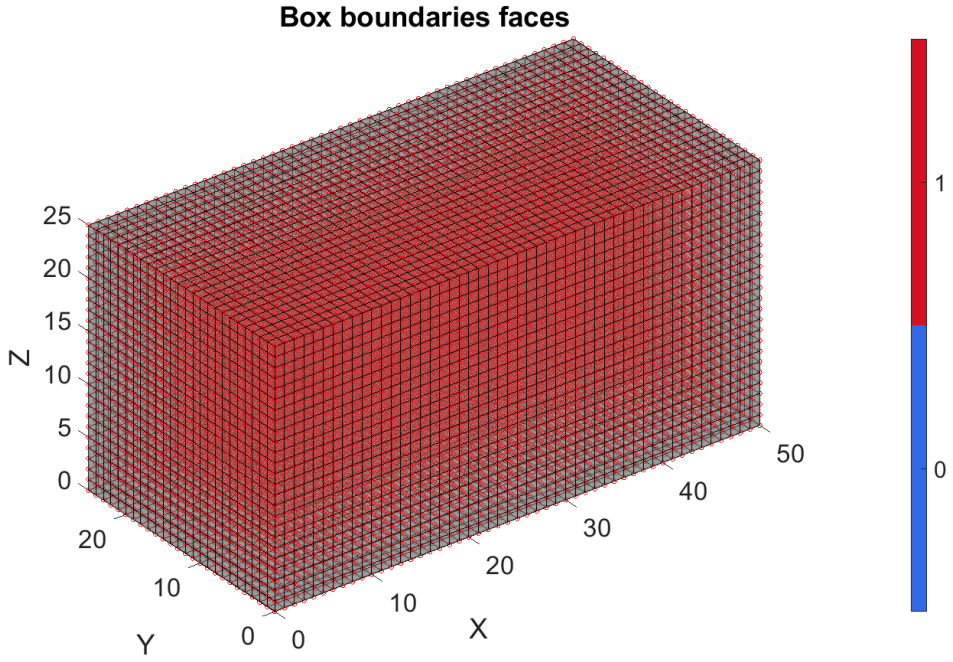
```
n = 5; % resolution scale factor
x = linspace (0,nelx,nelx*n);
y = linspace (0,nely,nely*n);
z = linspace (0,nelz,nelz*n);

[XG, YG, ZG] = meshgrid(x, y, z); %Creating grid

VG= [XG(:), YG(:), ZG(:)];
```

creating hex mesh

```
boxDim=[nelx nely nelz];
boxEl=[nelx nely nelz];

[meshStruct]=hexMeshBox(boxDim,boxEl); %creating elements

%mesh structure
E=meshStruct.E;
V=meshStruct.V;
F=meshStruct.F;
Fb=meshStruct.Fb;
faceBoundaryMarker=meshStruct.faceBoundaryMarker;

V=V-min(V,[],1); %shifting to origin
```

Visualization

```
cFigure; hold on;
title('Box boundaries faces','FontSize',fontSize);

gpatch(Fb,V,'kw','k',faceAlpha1, 0.5);
scatter3(V(:,1),V(:,2),V(:,3),20,'red');
% scatter3(VG(:,1),VG(:,2),VG(:,3),3,'y');

axisGeom(gca,fontSize);
colormap(gjet(6)); icolorbar;
drawnow;
```



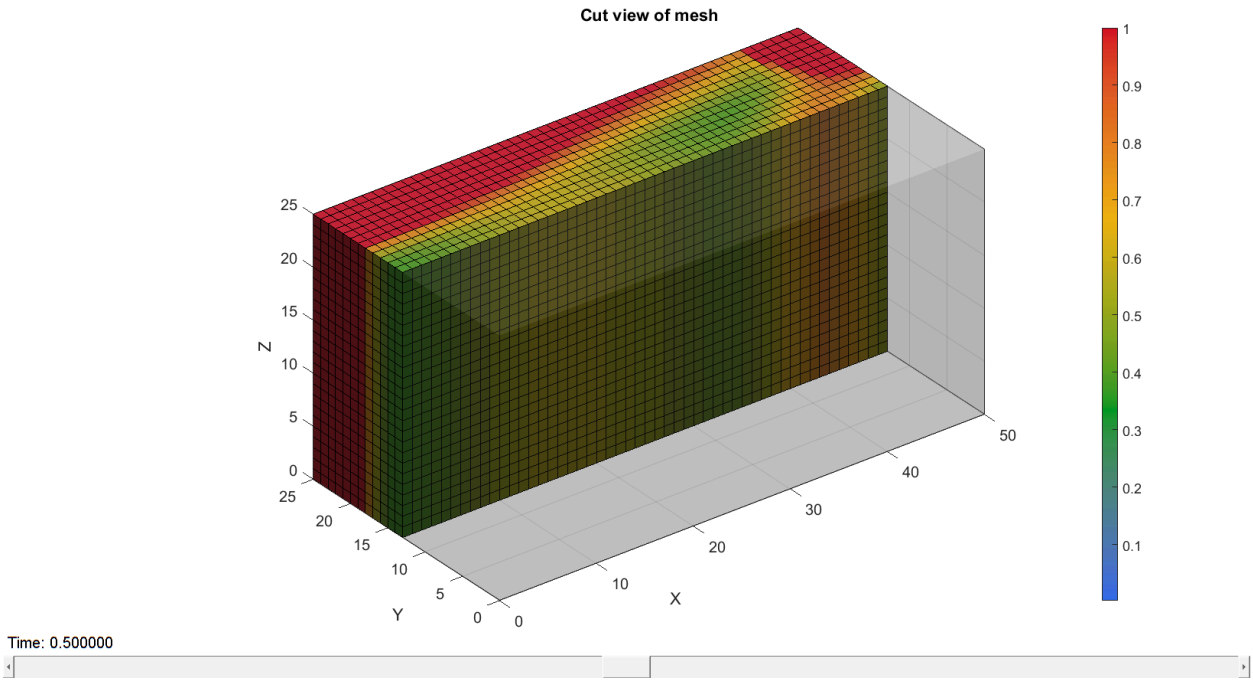
Access model element and patch data

```
% Fb=boundary(Vd(:,1), Vd(:,2), Vd(:,3),1); % boundary with shrink-factor
 1(not the same as faceBoundary)
indBoundary=tesBoundary(Fb);
Fb=Fb(indBoundary,:);
Cb=-1*ones(size(Fb,1),1);
CE=-2*ones(size(E,1),1);

meshStructure.elementData=rho; % Copy density as the element data

% Access model element and patch data
meshStructure.facesBoundary=Fb;
meshStructure.boundaryMarker=Cb;
meshStructure.nodes=V;
meshStructure.elementMaterialID=CE;
```

```
meshStructure.elements=E;
```

Mesh output

```
meshView(meshStructure);
```



Use barycentric mapping to figure out the elements voxel centres are found in

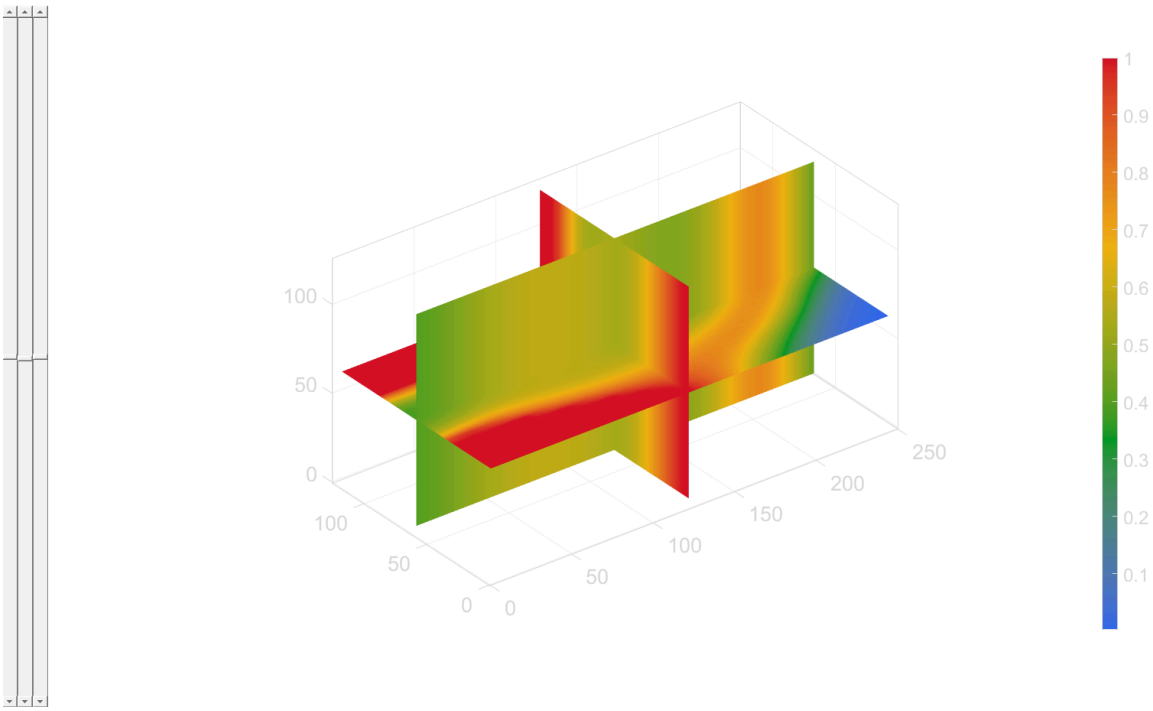
In this exampe, the 8-node elements are generated to hold the grids inside

```
% TR = triangulation(E,V); %Conver to "triangulation" type
% [elementIndexFound,baryCentricCoordinate]=pointLocationTR(TR,VG,1,1,1); %
 Compute
% logicInside=~isnan(elementIndexFound); %if nan then the voxels are outside
 shape
```

Interpolation of rho onto grid

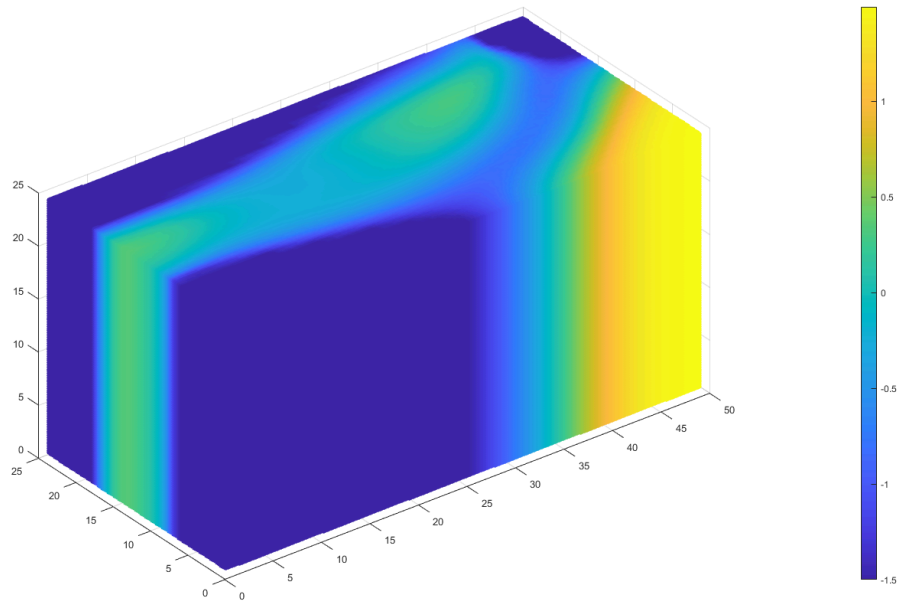
```
interpolationMethod = 2;

switch interpolationMethod
    case 1 % Neirest
        rho_VG=nan(size(VG,1),1);
        rho_VG(logicInside)=rho(elementIndexFound(logicInside));
        rho_VG=reshape(rho_VG, size(XG));
```

```
    case 2 % Linear
        Vm=patchCentre(E,V); % center of the elements

rho_VG=griddata(Vm(:,1),Vm(:,2),Vm(:,3),rho,XG(:),YG(:),ZG(:),'linear'); %Interpolate
on grid

    case 3 % Element nodal average tri-linear
        rho_V = faceToVertexMeasure(E,V,rho); % Average from elements to nodes
        % Shape function (=barycentric coordinate) based within element
        % tri-linear interpolation
        rho_VG=nan(size(VG,1),1);
        for indPoint = find(logicInside)'
            indElement =  elementIndexFound(indPoint);
            indNodes = E(indElement,:);
            rho_VG(indPoint) = sum(baryCentricCoordinate(indPoint,:) .*
rho_V(indNodes)');
        end

    case 4 % Natural
        Vm=patchCentre(E,Vi); % center of the elements
        interpFunc_rho =
scatteredInterpolant(Vm,rho,'natural','none'); %Create interpolator
        rho_VG=nan(size(VG,1),1);
        rho_VG(logicInside) = interpFunc_rho(VG(logicInside,:));
end

rho_VG=reshape(rho_VG, size(XG)); %reshape to the density size
sv3(rho_VG); colormap gjet;
```



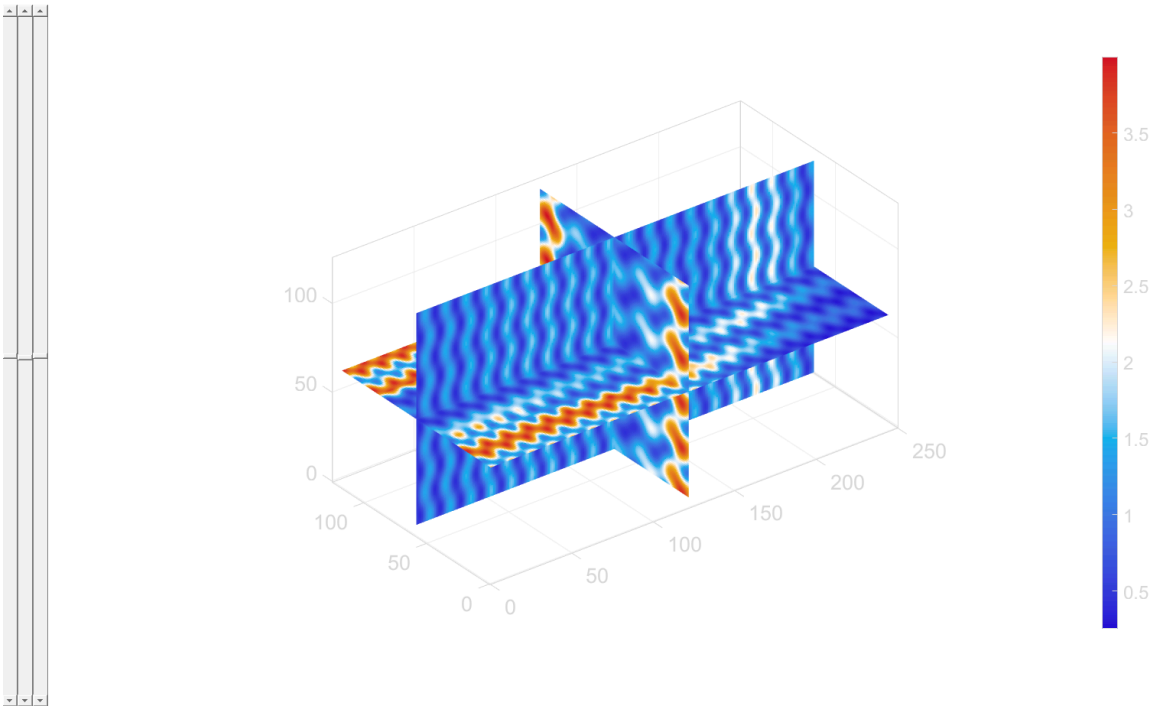
Mapping the density field to the equivalent gyroid levelSet field

```
l=(rho_VG-0.5)/-(1/3); % [-1.5, 0.75] levelset range
freq=0.05; %gyroid cell size
k = freq*boxDim; % number of periods

cFigure;
scatter3(VG(:,1),VG(:,2),VG(:,3),25,l(:),'filled');
axis tight; axis equal;
colorbar;
```



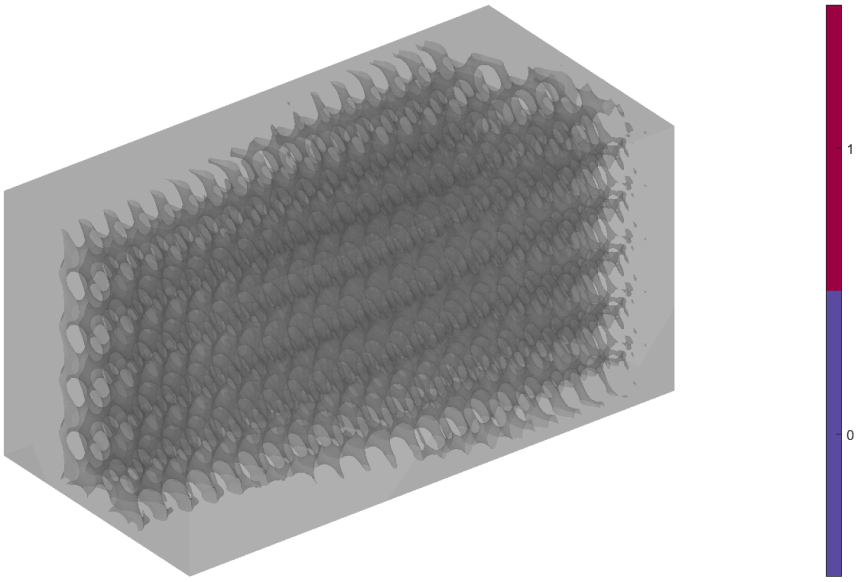
Evaluate triply periodic function

```
S=(sin(k(1,1).*XG).*cos(k(1,2).*YG))+(sin(k(1,2).*YG).*cos(k(1,3).*ZG))+(cos(k(1,1).*XG).*

i=2.5; % Leads to a minimu of 1 later
S=S+i; % S=-S-i;
l=l+i; % l=-l-i; % i:i+s
S=S./l;

%visualize gyroid function field
sv3(S); colormap warmcold;
```



Construct iso-surface

```
[Fg,Vg] = isosurface(XG,YG,ZG,S,1);
[Ft,Vt]=quad2tri(Fb,V,'f'); %Quad patch to triangular patch data

[Fsn,Vsn,Csn]=joinElementSets({Ft,Fg},{Vt,Vg});
[Fsn,Vsn]=patchCleanUnused(Fsn,Vsn); %Remove unused nodes
[Fsn,Vsn]=mergeVertices(Fsn,Vsn); %Merge nodes

% Visualize surface
cFigure;
gpatch(Fsn,Vsn,'kw','none',0.3);
axisGeom; colormap spectral; icolorbar;
camlight headlight; axis off;
drawnow;
```



Slicer view

```
cutViewAnim(Fsn,Vsn);
```

Published with MATLAB® R2021b