Table of Contents

.... 1
Adding Tube2FEM src directories to path 1
Create Geometry 1
Test skeletonisation submodule 5
Test boundaryConditions submodule (Part 1) 10
Test volumetricMesh submodule 11
Test boundaryConditions submodule (Part 2) 15
Test FiniteElement submodule 15
Test postprocessingParaView submodule 19

```
clear
close all
clc
```

Adding Tube2FEM src directories to path

```
CurrentFolder = pwd;
TopFolder = fileparts(pwd);
TopTopFolder = fileparts(fileparts(pwd));
FEFolder = strcat(CurrentFolder,'\FiniteElement');
srcFolder = strcat(TopFolder,'\src');
finiteElementFolder = strcat(srcFolder,'\FiniteElement');
boundaryConditionsFolder = strcat(srcFolder,'\boundaryConditions');
postprocessingParaViewFolder = strcat(srcFolder,'\postprocessingParaView');
surfaceMeshProcessingFolder = strcat(srcFolder,'\surfaceMeshProcessing');
volumetricMeshFolder = strcat(srcFolder,'\volumetricMesh');
skeletonisationFolder = strcat(srcFolder,'\skeletonisation');

addpath(srcFolder);
addpath(finiteElementFolder);
addpath(boundaryConditionsFolder);
addpath(postprocessingParaViewFolder);
addpath(surfaceMeshProcessingFolder);
addpath(volumetricMeshFolder);
addpath(skeletonisationFolder);
```

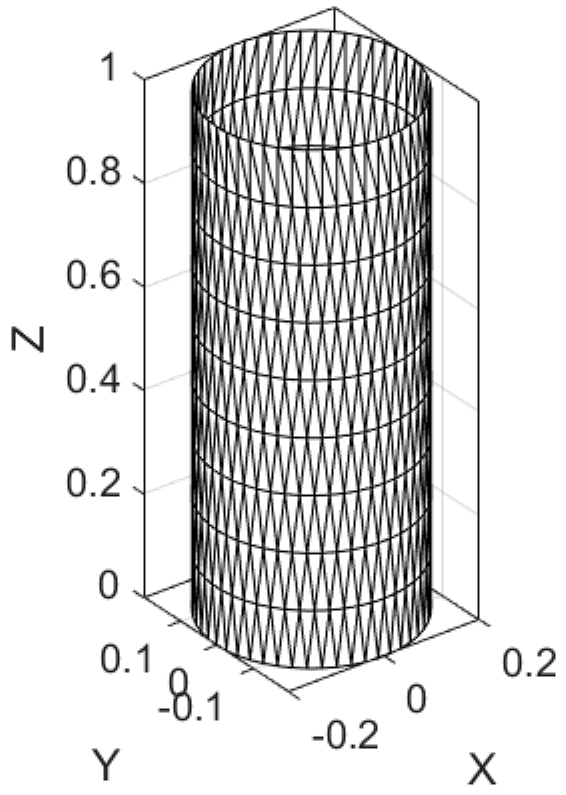
Create Geometry

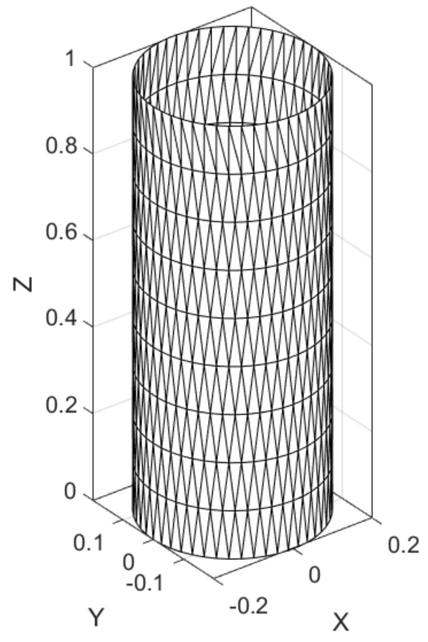
```
inputStruct.cylRadius=0.2;
inputStruct.numRadial=50;
inputStruct.cylHeight=1;
inputStruct.numHeight=10;
inputStruct.meshType='tri';

% Move cylinder in Z direction
movez = inputStruct.cylHeight/2;
```

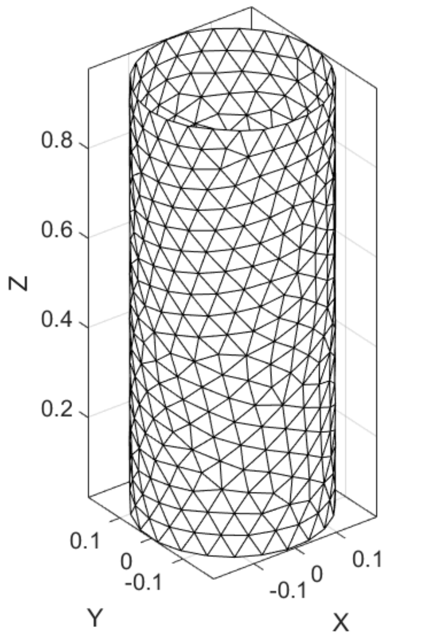
```
[F,V]=patchcylinder(inputStruct);
V(:,3) = V(:,3)+movez*ones(size(V,1),1);

gpatch(F,V,'w','k')
axisGeom

% Save in Input folder
fileName = 'Input/cylinder.stl';
patch2STL(fileName,V,F,[],'cylinder');

% Import  (.stl format)
[stlStruct] = import_STL('Input/cylinder.stl');
F=stlStruct.solidFaces{1}; %Faces
V=stlStruct.solidVertices{1}; %Vertices
[Fsurf,Vsurf]=mergeVertices(F,V); % Merging nodes

% Visualisation
cFigure;
gpatch(Fsurf,Vsurf,'w','k');
axisGeom;


% Remeshing using Geogram
optionStruct1.nb_pts= 500; % number of nodes in the remesh surface mesh
[Fsurf,Vsurf]=ggremesh(Fsurf,Vsurf,optionStruct1);

% Visualisation
cFigure;
gpatch(Fsurf,Vsurf,'w','k');
axisGeom;


% Plot Centerline Skeleton (saved in Centreline folder)
cFigure
[data,txt] = xlsread('Centreline\centrelineCylinder.csv');

% Plot the network (3D network of 1D segments)
k = 1;
m=size(data,1);
for i = 1:m
     startNode1 = data(i,3);
     endNode1   = data(i,4);
     barRadius1 = data(i,5);
     barLength1 = data(i,6);
     startNodeCoor1 = [data(i,7) data(i,8) data(i,9)];
     endNodeCoor1 = [data(i,10) data(i,11) data(i,12)];
     line([startNodeCoor1(1) endNodeCoor1(1)], ...
         [startNodeCoor1(2) endNodeCoor1(2)], ...
         [startNodeCoor1(3) endNodeCoor1(3)])
     hold on
     n=2; % Each segment is formed of 2 nodes
     V = [startNodeCoor1;endNodeCoor1];
     V =evenlySampleCurve(V,n,'linear',0);
```

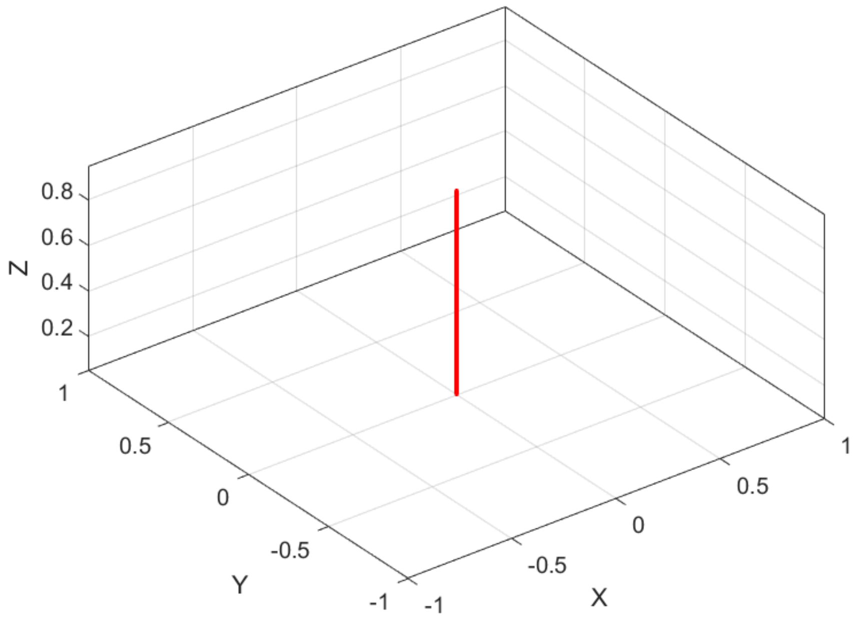
```
    VN_all(1:n,3*i-2:3*i) = V;
    plotV(V,'r.-','lineWidth',3);

end

% Visualisation
axisGeom;
hold on
```

```
Warning: The trinorm function is depricated. Use patchNormal instead.
```




1
0.8
0.6
Z
0.4
0.2
0
0.1
0
-0.1
Y
-0.2
0
0.2
X

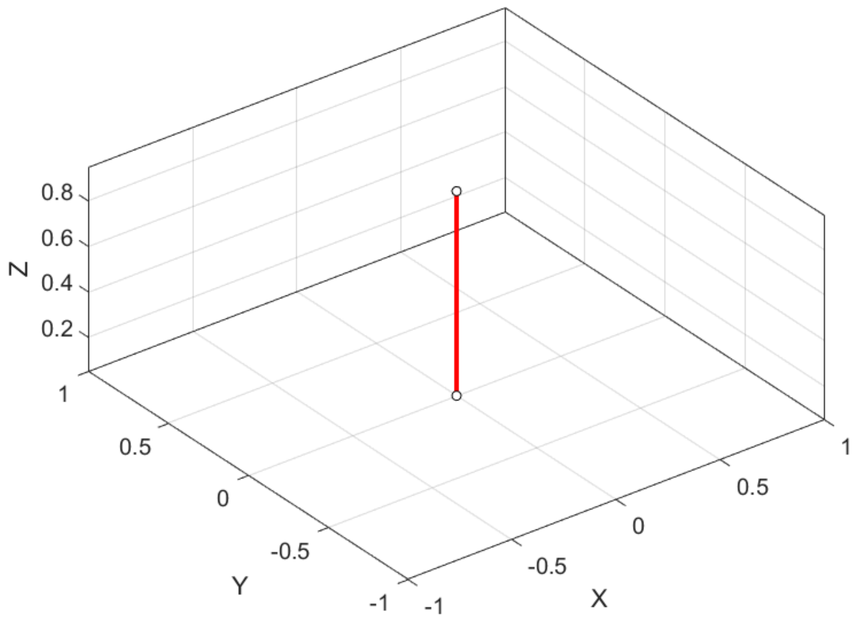

0.8
0.6
Z
0.4
0.2
0.1
0
-0.1
Y
-0.1
0
0.1
X

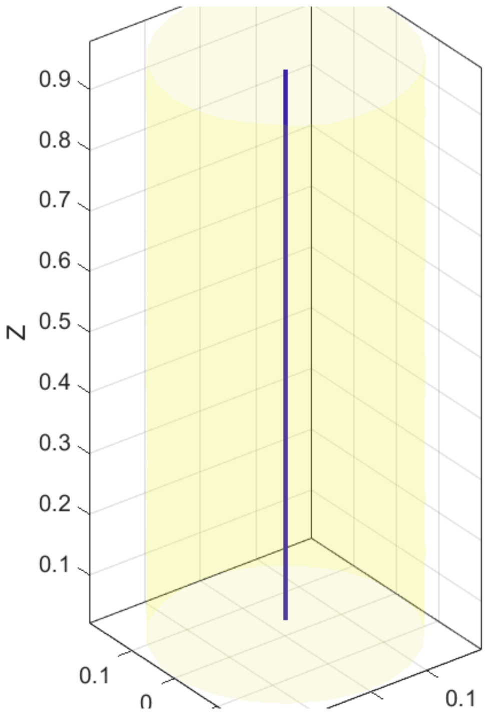


Test skeletonisation submodule

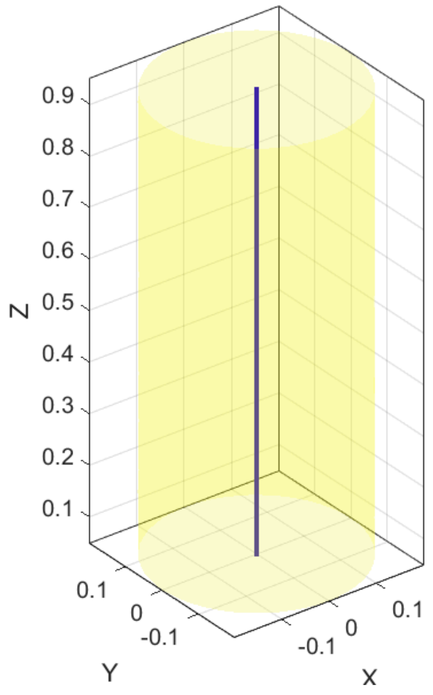
```
% ----------------------------------- %
% Detect Edge nodes of the centreline %
% ----------------------------------- %
% Detect edge input nodes
[EdgeInput,EdgeInputCoor,labelCount] = EdgeInputDetection(data);
% Detect edge output nodes
[EdgeOutput,EdgeOutputCoor] = EdgeOutputDetection(data,labelCount);
EdgePtCoor = [EdgeInputCoor ; EdgeOutputCoor];


% Visualisation
for i=1:size(EdgePtCoor,1)
    scatter3(EdgePtCoor(i,6),EdgePtCoor(i,7), EdgePtCoor(i,8), ...
        'MarkerEdgeColor','k','MarkerFaceColor',[1. 1. 1.]);
    hold on
end

% ---------------------------------- %
% Build Skeleton with 1D radii field %
% ---------------------------------- %
% Define Vertices and Elements Cell Arrays
CellArray_V = cell(1,m);
CellArray_E = cell(1,m);

for i = 1:m
    V = VN_all(1:n,3*i-2:3*i);
    E = [(1:size(V,1)-1)' (2:size(V,1))'];
    barRadius = data(i,5);
    barRadii(((i-1)*n+1):i*n,1) = linspace(barRadius,barRadius,size(V,1))';
    CellArray_V{1,i} = V;
    CellArray_E{1,i} = E;
end

% Join Element Sets (to avoid duplications)
[E,Vske]=joinElementSets(CellArray_E,CellArray_V);
[E,Vske,ind1]=mergeVertices(E,Vske);
barRadii=barRadii(ind1);

% Visualisation
cFigure;
gpatch(E,Vske,'none',barRadii,0,3);
axisGeom;
hold on

Csurf = 0.5*ones(size(Fsurf,1),1);
patch('faces',Fsurf,'vertices',Vsurf,'FaceColor','flat','CData',Csurf, ...
    'FaceAlpha',0.1,'EdgeColor','none');






% ------------------------------------------------------ %
% Slicing the surface mesh to define boundary conditions %
% ------------------------------------------------------ %

F = Fsurf;
V = Vsurf;
dirVec = zeros(size(EdgePtCoor,1),3);
indAll = 1:size(EdgePtCoor,1);
notWorking = [];
flag = ~ismember(indAll,notWorking);
index = find(flag);
cFigure

for i =1:size(EdgePtCoor,1)
    indexi = index(i);
    C=[];
    snapTolerance=mean(patchEdgeLengths(F,V))/100;
    Pin=[EdgePtCoor(indexi,6),EdgePtCoor(indexi,7),EdgePtCoor(indexi,8)];
    Pi = [EdgePtCoor(indexi,3),EdgePtCoor(indexi,4),EdgePtCoor(indexi,5)];
    n= Pin - Pi;

    [D]=minDist(V,Pi);
    radius = EdgePtCoor(indexi,9);
    logicDist=D<4.5*radius;
    logicCut=any(logicDist(F),2);
    Fa= F(logicCut,:);
    Fb = F(~logicCut,:);
    % cFigure
    % gpatch(Fa,V ,'w','k');
    % hold on
    % gpatch(Fb, V, 'b','k');
    % axisGeom;


    hold on
    axisGeom
    % scatter3(Pi(1),Pi(2),Pi(3),'MarkerEdgeColor','k', ...
    %     'MarkerFaceColor',[1. 1. 1.])
    % scatter3(Pin(1),Pin(2),Pin(3),'MarkerEdgeColor','k', ...
    %     'MarkerFaceColor',[1. 0. 0.])

    % Check if Fa is composed of multiple disconnected surfaces
    a=(120/180)*pi;
    G=patchFeatureDetect(Fa,V,a);

    if max(G)~=1
        uniqueG = unique(G);
        sizeUniqueG = size(uniqueG,1);
        count = zeros(sizeUniqueG,1);
            for k = 1:sizeUniqueG
                count(k) = sum(G==uniqueG(k));
            end
        maxCount = index(count == max(count))
        Fb = [Fb;Fa(G~=maxCount,:)];
        Fa = Fa(G==maxCount,:);
    end

    % Slicing
    Pcut = 0*Pin + 1*Pi;
    [Fa,Va,~,logicSide]=triSurfSlice(Fa,V,C,Pcut,n,snapTolerance);

    % Attaching the processed surface back to the main synthetic
    % network surface mesh
    [Fa,Va]=patchCleanUnused(Fa(~logicSide,:),Va);
    [F1,V1]=joinElementSets({Fa,Fb},{Va,V});
    [F,V]=mergeVertices(F1,V1);
end

a=(120/180)*pi;
G=patchFeatureDetect(F,V,a);
[GC,GR] = groupcounts(G);
ind = find(GC==max(GC));
indval = GR(ind);
F = F(G==indval,:);

% Visualisation
C = 0.5*ones(size(F,1),1);
patch('faces',F,'vertices',V,'FaceColor','flat','CData',C, ...
    'FaceAlpha',0.2,'EdgeColor','none');
axisGeom
hold on
gpatch(E,Vske,'none',barRadii,0,3);
```

```
% Remesh open surface of the sliced surface mesh
nb_pts = size(V,1);
optionStruct2.nb_pts = nb_pts;
[F,V] = ggremesh(F,V,optionStruct2);
cFigure;
gpatch(F,V,'w','k');
axisGeom;
```

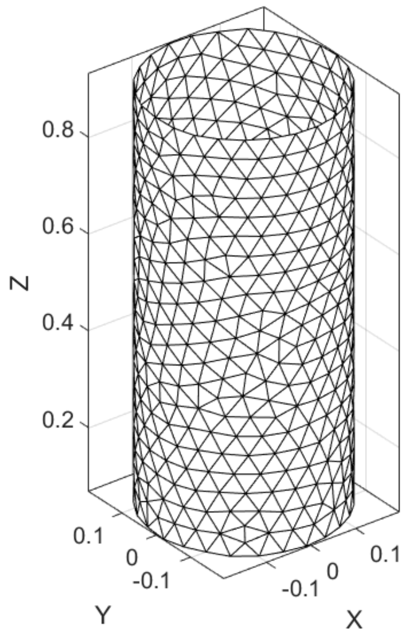




0.9
0.8
0.7
0.6
Z
0.5
0.4
0.3
0.2
0.1
0.1
0
0.1

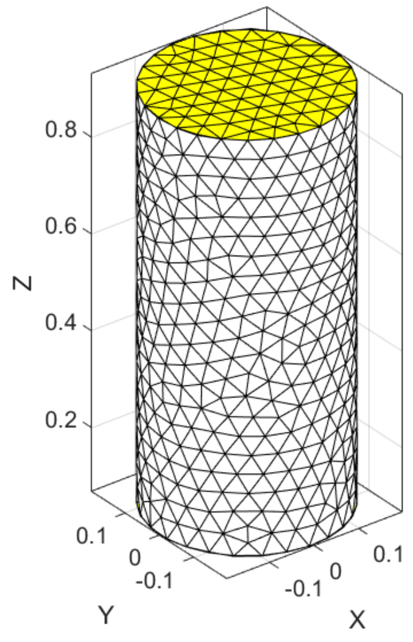


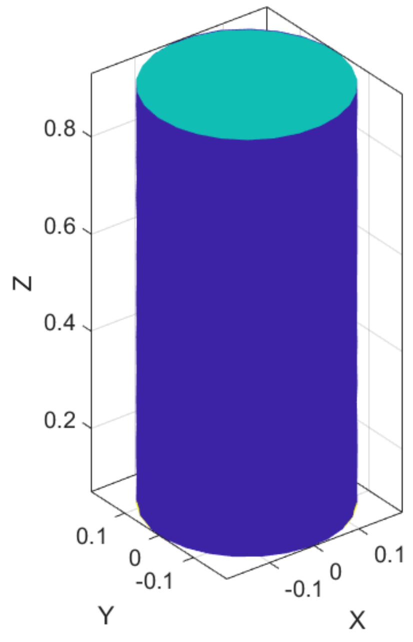
0.9
0.8
0.7
0.6
Z
0.5
0.4
0.3
0.2
0.1
0.1
0
-0.1
Y
-0.1
0
0.1
X



Test boundaryConditions submodule (Part 1)

```
% ------------------------- %
% Close holes and labelling %
% ------------------------- %
C = ones(size(F,1),1);
Cc = C;
Fc = F;
Vc = V;
[Fc,Vc,Cc]=closeHolesAndLabelling(Fc,Vc,Cc);

% Visualisation
logicRegion1=ismember(Cc,1);
logicRegion2=ismember(Cc,2:max(Cc));
V_region2=getInnerPoint(Fc(logicRegion2,:),Vc);
cFigure;
patch('faces',Fc(logicRegion1,:),'vertices',Vc,'FaceColor','flat', ...
    'CData',Cc(logicRegion1),'FaceAlpha',1,'EdgeColor','none');
hold on;
patch('faces',Fc(logicRegion2,:),'vertices',Vc,'FaceColor','flat', ...
    'CData',Cc(logicRegion2),'FaceAlpha',1,'EdgeColor','none');
axisGeom
```





Test volumetricMesh submodule

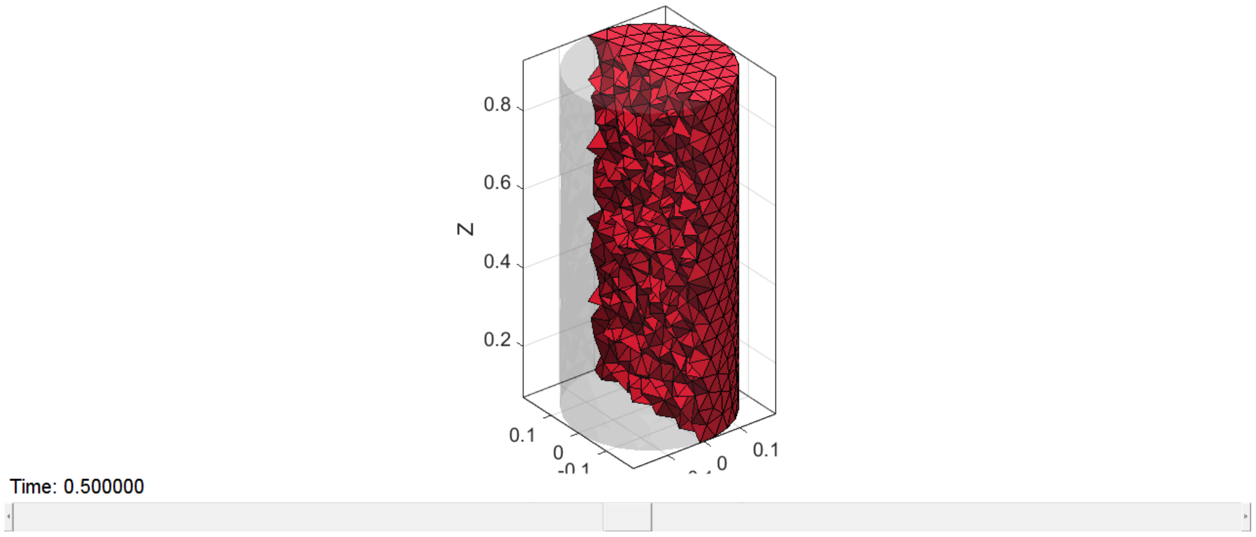
```
% ---------------------------- %
% Volumetric Mesh using TetGen %
% ---------------------------- %
cd Mesh
V_regions=getInnerPoint(Fc,Vc); %Define region points
```

```
V_holes=[]; %Define hole points
[regionTetVolumes]=tetVolMeanEst(Fc,Vc); %Volume estimate for regular tets
stringOpt='-pq1.2AaY'; %Options for tetgen
modelName = 'test';

%Create tetgen input structure
inputStruct.stringOpt=stringOpt; %Tetgen options
inputStruct.Faces=Fc; %Boundary faces
inputStruct.Nodes=Vc; %Nodes of boundary
inputStruct.faceBoundaryMarker=Cc;
inputStruct.regionPoints=V_regions; %Interior points for regions
inputStruct.holePoints=V_holes; %Interior points for holes
inputStruct.regionA=regionTetVolumes; %Desired tet volume for each region
inputStruct.modelName = modelName;

% Mesh model using tetrahedral elements using tetGen
[meshOutput]=runTetGen(inputStruct); %Run tetGen

% Access mesh output structure
E=meshOutput.elements; %The elements
V=meshOutput.nodes; %The vertices or nodes
CE=meshOutput.elementMaterialID; %Element material or region id
Fb=meshOutput.facesBoundary; %The boundary faces
Cb=meshOutput.boundaryMarker; %The boundary markers

% Visualisation
faceAlpha1 = 0.8;
markerSize = 1.5;
lineWidth=3;
cMap=blood(250);

hf=cFigure;
hold on;

% Creating single-domain volumetric mesh
meshOutput.elementMaterialID = ones(size(meshOutput.elementMaterialID,1),1);

% Visualisation using meshView
optionStruct.hFig=[hf];
optionStruct.hFig.WindowState = "maximized";
optionStruct.edgeColor='k';
optionStruct.cutDir = 1;
optionStruct.cutSide = 1;
optionStruct.numSliceSteps=25;
meshView(meshOutput,optionStruct);
% title('Tetrahedral mesh','FontSize',fontSize);
fontSize=15;
axisGeom(gca,fontSize);
gdrawnow;
camlight('right')
colorbar off

% ------------------------------------------------ %
% Convert to (.msh) Gmsh Format (Version 2 ASCII) %
% ---------------------------------------------- %
GmshFileName = 'cylinder.msh';
[status] = writeGmsh(GmshFileName,meshOutput);

% ------------------------- %
% Convert to (.xdmf) Format %
% --------------------------%
meshPath = strcat(CurrentFolder,'\Mesh');
meshPath = strrep(meshPath, '\', '/');
% Split the path into parts
pathParts = strsplit(meshPath, '/');
% Reconstruct the path without the first directory
newMeshPath = strjoin(pathParts(2:end), '/');
rootPath = "../../../../../../../../../../../../../../../../../mnt/c/";
wslMeshPath = strcat(rootPath,newMeshPath,"/meshioConvertMesh.py");
wslMeshPath= sprintf('"%s"', wslMeshPath);

fid1 = fopen('meshInfo.txt','wt');
meshPath1 = sprintf('"%s"', strcat(rootPath,newMeshPath));
fprintf(fid1,strcat('meshPath=',meshPath1));
fprintf(fid1,'\n');
GmshFileName= sprintf('"%s"', GmshFileName);
fprintf(fid1,strcat('GmshFileName=',GmshFileName));
fclose(fid1);

script1 = fileread('meshInfo.txt');
script2 = fileread('../../src/volumetricMesh/ConvertGmshToXdmf.py');

fid2 = fopen('meshioConvertMesh.py','wt');
fprintf(fid2,script1);
fprintf(fid2,'\n');
fprintf(fid2,script2);
fprintf(fid2,'\n');
fclose(fid2);

wslPath = '"C:\Windows\System32\wsl.exe"';
runString = strcat(wslPath,' python3',append(' ',wslMeshPath));
[runStatus,runOut]=system(runString,'-echo');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- TETGEN Tetrahedral meshing --- 15-May-2025 18:38:53

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- Writing SMESH file --- 15-May-2025 18:38:53

smeshName =

    'C:\Program Files\MATLAB\GIBBON-master\data\temp\test.smesh'

----> Adding node field
----> Adding facet field
----> Adding holes specification
----> Adding region specification
```

```
--- Done --- 15-May-2025 18:38:53

runString =

    '"C:\Program Files\MATLAB\GIBBON-master\lib_ext\tetGen\win64\tetgen.exe" -
pq1.2AaY "C:\Program Files\MATLAB\GIBBON-master\data\temp\test.smesh"'

--- Running TetGen to mesh input boundary--- 15-May-2025 18:38:53
Opening C:\Program Files\MATLAB\GIBBON-master\data\temp\test.smesh.
Delaunizing vertices...
Delaunay seconds:  0.006
Creating surface mesh ...
Surface mesh seconds:  0.001
Recovering boundaries...
Boundary recovery seconds:  0.001
Removing exterior tetrahedra ...
Spreading region attributes.
Exterior tets removal seconds:  0.001
Recovering Delaunayness...
Delaunay recovery seconds:  0.001
Refining mesh...
Refinement seconds:  0.027
Optimizing mesh...
Optimization seconds:  0.004

Writing C:\Program Files\MATLAB\GIBBON-master\data\temp\test.1.node.
Writing C:\Program Files\MATLAB\GIBBON-master\data\temp\test.1.ele.
Writing C:\Program Files\MATLAB\GIBBON-master\data\temp\test.1.face.
Writing C:\Program Files\MATLAB\GIBBON-master\data\temp\test.1.edge.

Output seconds:  0.024
Total running seconds:  0.065

Statistics:

  Input points: 710
  Input facets: 1416
  Input segments: 2124
  Input holes: 0
  Input regions: 1

  Mesh points: 2696
  Mesh tetrahedra: 14811
  Mesh faces: 30330
  Mesh faces on exterior boundary: 1416
  Mesh faces on input facets: 1416
  Mesh edges on input segments: 2124
  Steiner points inside domain: 1986

--- Done --- 15-May-2025 18:38:53

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- Importing TetGen files --- 15-May-2025 18:38:53
--- Done --- 15-May-2025 18:38:53
```

```
Gmsh Version 2 ASCII written
Current working directory: /mnt/c/Users/homeuser/Documents/GitHub/Tube2FEM/
tests/Mesh
```



Test boundaryConditions submodule (Part 2)

```
% ------------ %
% BC to FEniCS %
% ------------ %
cd ../FiniteElement
BCtoFEniCS(meshOutput,Fc,Vc);
```

```
BCs are written correctly in BC.txt
```

Test FiniteElement submodule

```
% --------------- %
% Combine Scripts %
% --------------- %
combineScripts;

% -------------------------- %
% Run FEniCS - Navier-Stokes %
% -------------------------- %
cd ../FiniteElement

%------ Mesh Path -------%
meshPath = strcat(CurrentFolder,'\Mesh');
meshPath = strrep(meshPath, '\', '/');
% Split the path into parts
pathParts = strsplit(meshPath, '/');
% Reconstruct the path without the first directory
newMeshPath = strjoin(pathParts(2:end), '/');
```

```
%------- Output Path -----%
outPath = strcat(CurrentFolder,'\simOutput_3D');
outPath = strrep(outPath, '\', '/');
% Split the path into parts
pathParts = strsplit(outPath, '/');
% Reconstruct the path without the first directory
newOutPath = strjoin(pathParts(2:end), '/');

% ------- FE Path ---------%
FEPath = strrep(FEFolder, '\', '/');
% Split the path into parts
pathParts = strsplit(FEPath, '/');
% Reconstruct the path without the first directory
newFEPath = strjoin(pathParts(2:end), '/');

% ------ Root Path ---------%
rootPath = "../../../../../../../../../../../../../../../../../mnt/c/";

% ------ WSL FE Path -------%
wslFEPath = strcat(rootPath,newFEPath,"/finalNavierStokes.py");
wslFEPath= sprintf('"%s"', wslFEPath);

% ----Create Infor.txt -----%
fid1 = fopen('Info.txt','wt');
meshPath1 = sprintf('"%s"', strcat(rootPath,newMeshPath,"/Tetra.xdmf"));
meshPath2 = sprintf('"%s"', strcat(rootPath,newMeshPath,"/Tri.xdmf"));
fileName = sprintf('''%s''',strcat(rootPath,newOutPath,"/velocity.pvd"));
fileName1 = sprintf('''%s''',strcat(rootPath,newOutPath,"/pressure.pvd"));
fileName2 = sprintf('''%s''',strcat(rootPath,newOutPath,"/wss.pvd"));
timeSeriesFileName = sprintf('''%s''',strcat(rootPath,newOutPath, ...
    "/velocity_series"));

fprintf(fid1,strcat('tetraFileName=',meshPath1));
fprintf(fid1,'\n');
fprintf(fid1,strcat('triFileName=',meshPath2));
fprintf(fid1,'\n');
fprintf(fid1,strcat('fileName=',fileName));
fprintf(fid1,'\n');
fprintf(fid1,strcat('fileName1=',fileName1));
fprintf(fid1,'\n');
fprintf(fid1,strcat('fileName2=',fileName2));
fprintf(fid1,'\n');
fprintf(fid1,strcat('timeSeriesFileName=',timeSeriesFileName));
fprintf(fid1,'\n');
fclose(fid1);

script1 = fileread('Info.txt');
script2 = fileread('combinedNavierStokes.py');

% Join Info.txt with FEniCS file in the FiniteElement directory
fid2 = fopen('FinalNavierStokes.py','wt');
fprintf(fid2,script1);
fprintf(fid2,'\n');
fprintf(fid2,script2);
fprintf(fid2,'\n');
fclose(fid2);


% Run Navier-Stokes in the case Study directory
% Uncomment to run the FEniCS script on WSL

% wslPath = '"C:\Windows\System32\wsl.exe"';
runString = strcat(wslPath,' python3',append(' ',wslFEPath));
[runStatus,runOut]=system(runString,'-echo');

0.01
0.02
0.03
0.04
0.05
0.060000000000000005
0.07
0.08
0.09
0.09999999999999999
0.10999999999999999
0.11999999999999998
0.12999999999999998
0.13999999999999999
0.15
0.16
0.17
0.18000000000000002
0.19000000000000003
0.20000000000000004
0.21000000000000005
0.22000000000000006
0.23000000000000007
0.24000000000000007
0.25000000000000006
0.26000000000000006
0.2700000000000001
0.2800000000000001
0.2900000000000001
0.3000000000000001
0.3100000000000001
0.3200000000000001
0.3300000000000001
0.34000000000000014
0.35000000000000014
0.36000000000000015
0.37000000000000016
0.38000000000000017
0.3900000000000002
0.4000000000000002
0.4100000000000002
0.4200000000000002
0.4300000000000002
```

```
0.4400000000000002
0.45000000000000023
0.46000000000000024
0.47000000000000025
0.48000000000000026
0.49000000000000027
0.5000000000000002
0.5100000000000002
0.5200000000000002
0.5300000000000002
0.5400000000000003
0.5500000000000003
0.5600000000000003
0.5700000000000003
0.5800000000000003
0.5900000000000003
0.6000000000000003
0.6100000000000003
0.6200000000000003
0.6300000000000003
0.6400000000000003
0.6500000000000004
0.6600000000000004
0.6700000000000004
0.6800000000000004
0.6900000000000004
0.7000000000000004
0.7100000000000004
0.7200000000000004
0.7300000000000004
0.7400000000000004
0.7500000000000004
0.7600000000000005
0.7700000000000005
0.7800000000000005
0.7900000000000005
0.8000000000000005
0.8100000000000005
0.8200000000000005
0.8300000000000005
0.8400000000000005
0.8500000000000005
0.8600000000000005
0.8700000000000006
0.8800000000000006
0.8900000000000006
0.9000000000000006
0.9100000000000006
0.9200000000000006
0.9300000000000006
0.9400000000000006
0.9500000000000006
0.9600000000000006
0.9700000000000006
```

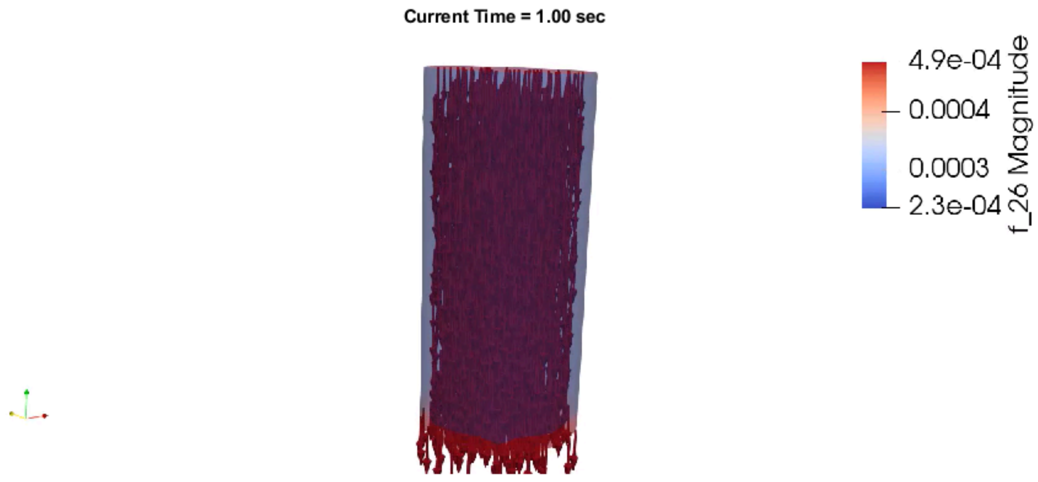
```
0.9800000000000006
0.9900000000000007
1.0000000000000007
```

Test postprocessingParaView submodule

```
% -------------- %
% PostProcessing %
% -------------- %
cd(CurrentFolder)
cd Postprocessing\

%------- velocity Path -----%
outPath = strcat(CurrentFolder,'\simOutput_3D\velocity.pvd');
outPath = strrep(outPath, '\', '/');
%------- Postprocessing Path -----%
postPath = strcat(CurrentFolder,'\Postprocessing\velocity.mp4');
postPath = strrep(postPath, '\', '/');

fid1 = fopen('postprocessingPathes.txt','wt');
outPath = sprintf('"%s"',outPath);
postPath = sprintf('''%s''',postPath);
fprintf(fid1,strcat('velPath=',outPath));
fprintf(fid1,'\n');
fprintf(fid1,strcat('postPath=',postPath));
fprintf(fid1,'\n');
fclose(fid1);

script1 = fileread('postprocessingPathes.txt');
script2 = fileread('macroParaviewNavierStokes.py');

% Join postprocessingPathes.txt with macroParaviewNavierStokes.py file
% in the FiniteElement directory
fid2 = fopen('macroNavierStokes.py','wt');
fprintf(fid2,script1);
fprintf(fid2,'\n');
fprintf(fid2,script2);
fprintf(fid2,'\n');
fclose(fid2);

% Change according to the User ParaView Directory!
paraviewPath = '"C:\Users\homeuser\Downloads\ParaView-5.10.1-Windows-
Python3.9-msvc2017-AMD64\bin\pvbatch.exe"';

% ParaView macro path
macroPath = strcat(CurrentFolder,'\Postprocessing
\macroNavierStokes_Cylinder.py');
macroPath= sprintf('"%s"', macroPath);

% Run ParaView Macro using |pvbatch|
runString = strcat(paraviewPath,append(' ',macroPath));
[runStatus,runOut]=system(runString,'-echo');
```

```
% Read .mp4 file generated by ParaView Macro (CFD)
cFigure;
vidObj = VideoReader("velocity.mp4");
timeStep = 0;
simDuration=1;
while(hasFrame(vidObj))
    frame = readFrame(vidObj);
    imshow(frame)
    timeStep = timeStep+(simDuration/vidObj.NumFrames);
    title(sprintf("Current Time = %.2f sec",timeStep))
    pause(10/vidObj.FrameRate)
end
```

```
C:/Users/homeuser/Downloads/ParaView-5.10.1-Windows-Python3.9-msvc2017-AMD64/
bin\vtkpython: can't open file 'C:\Users\homeuser\Documents\GitHub\Tube2FEM
\tests\Postprocessing\macroNavierStokes_Cylinder.py': [Errno 2] No such file
 or directory
```



Published with MATLAB® R2023a