Table of Contents

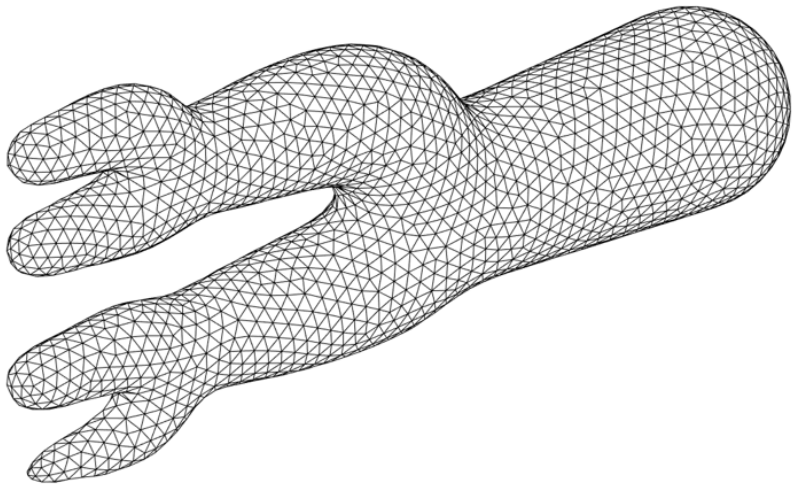

```
clear
close all
clc
```

Adding Tube2FEM src directories to path

```
CurrentFolder = pwd;
TopFolder = fileparts(pwd);
TopTopFolder = fileparts(fileparts(pwd));
FEFolder = strcat(CurrentFolder,'\FiniteElement');
srcFolder = strcat(TopTopFolder,'\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);
```

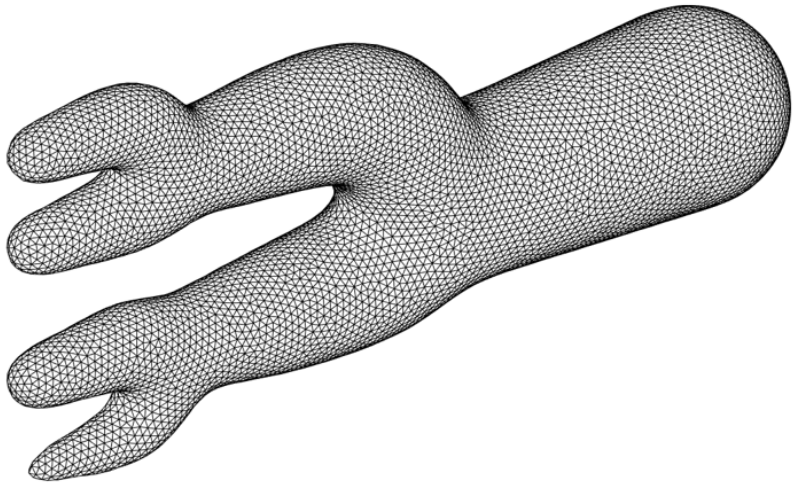
Import Synthetic network (.stl format)

```
[stlStruct] = import_STL('Input\syntheticNetwork.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;
axis off
zoom(1.3)
```



Remeshing using Geogram

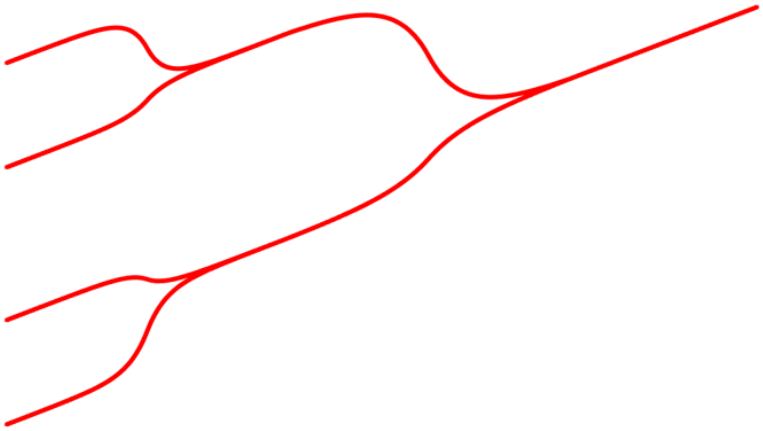
```
optionStruct1.nb_pts= 10000; % number of nodes in the remesh surface mesh
[Fsurf,Vsurf]=ggremesh(Fsurf,Vsurf,optionStruct1);

% Visualisation
cFigure;
gpatch(Fsurf,Vsurf,'w','k');
axisGeom;
axis off
zoom(1.3)
```



Plot Centerline Skeleton (saved in Centreline folder)

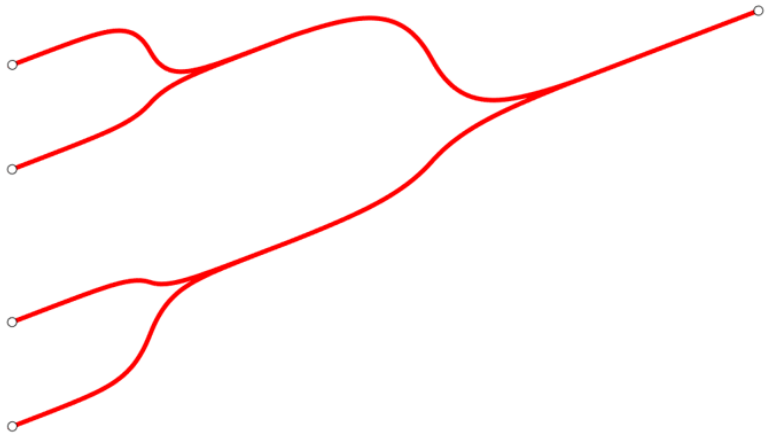
```
cFigure
[data,txt] = xlsread('Centreline\centrelineSyntheticNetwork.csv');

% 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
axis off
zoom(1.3)
```



Detect Edge nodes of the centreline

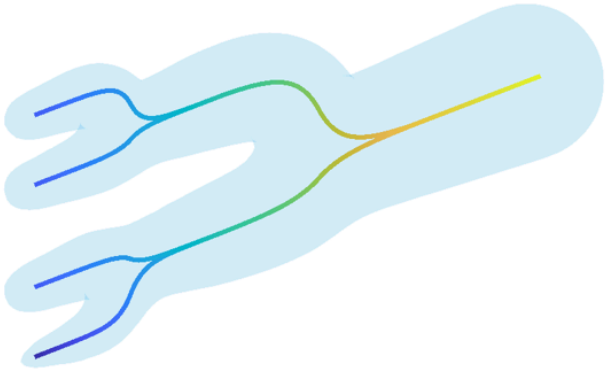
```
% Detect edge input nodes
[EdgeInput,EdgeInputCoor,labelCount] = EdgeInputDetection(data);
% Detect edge output nodes
[EdgeOutput,EdgeOutputCoor] = EdgeOutputDetection(data,labelCount);
EdgePtCoor = [EdgeInputCoor ; EdgeOutputCoor];
EdgePtCoor = EdgePtCoor([1,10,11,13,14],:);
EdgePtCoor(2:5,10) = [3;4;5;6];

% 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 (a radius is assigned for each segment)

```
% 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
axis off
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

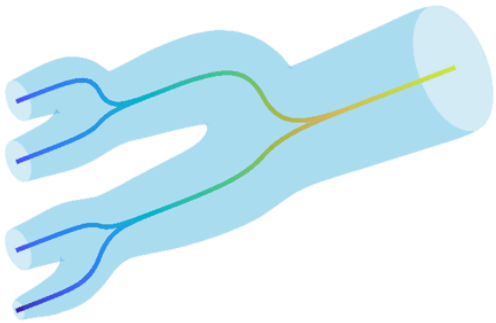
```
F = Fsurf;
V = Vsurf;
dirVec = zeros(size(EdgePtCoor,1),3);
indAll = 1:size(EdgePtCoor,1);
notWorking = [];
flag = ~ismember(indAll,notWorking);
index = find(flag);
cFigure

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);
```

```
axis off
```

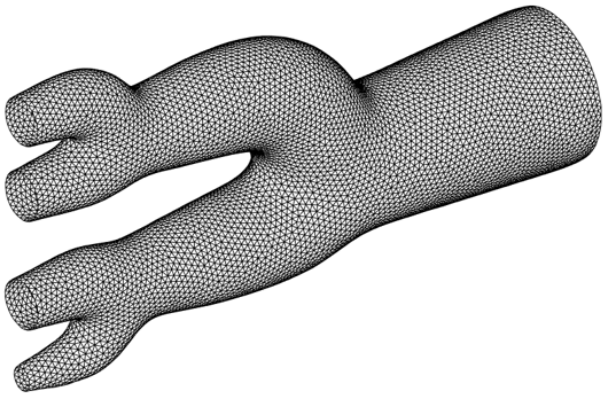
```
maxCount =

     1
```

```
maxCount =

     2
```



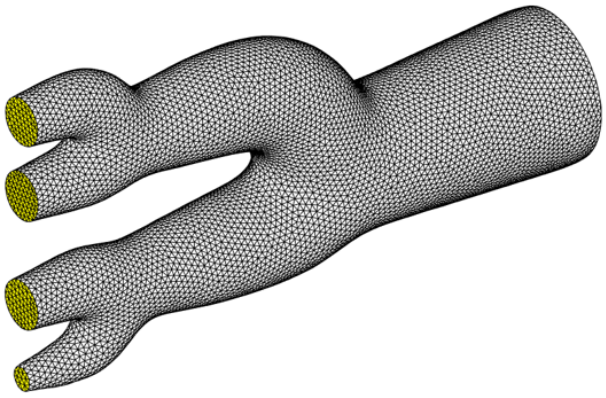
Remesh open surface of the sliced surface mesh

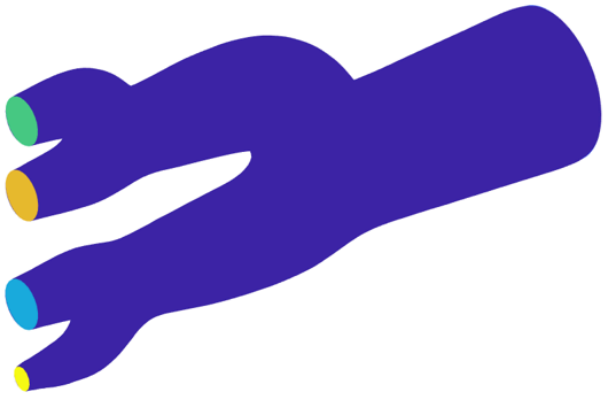
```
nb_pts = size(V,1);
optionStruct2.nb_pts = nb_pts;
[F,V] = ggremesh(F,V,optionStruct2);
cFigure;
gpatch(F,V,'w','k');
axisGeom;
axis off
```



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
axis off
```





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
axis off


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- TETGEN Tetrahedral meshing --- 25-Apr-2025 07:02:27
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- Writing SMESH file --- 25-Apr-2025 07:02:27

smeshName =

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

----> Adding node field
----> Adding facet field
----> Adding holes specification
----> Adding region specification
--- Done --- 25-Apr-2025 07:02:27

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--- 25-Apr-2025 07:02:27
Opening C:\Program Files\MATLAB\GIBBON-master\data\temp\test.smesh.
Delaunizing vertices...
Delaunay seconds:  0.07
Creating surface mesh ...
Surface mesh seconds:  0.012
Recovering boundaries...
Boundary recovery seconds:  0.035
Removing exterior tetrahedra ...
Spreading region attributes.
Exterior tets removal seconds:  0.02
Recovering Delaunayness...
Delaunay recovery seconds:  0.015
Refining mesh...
Refinement seconds:  1.159
Optimizing mesh...
Optimization seconds:  0.141

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:  0.653
Total running seconds:  2.105

Statistics:

  Input points: 11006
  Input facets: 22008
  Input segments: 33012
  Input holes: 0
  Input regions: 1

  Mesh points: 96610
  Mesh tetrahedra: 583633
```

```
  Mesh faces: 1178270
  Mesh faces on exterior boundary: 22008
  Mesh faces on input facets: 22008
  Mesh edges on input segments: 33012
  Steiner points inside domain: 85604

--- Done --- 25-Apr-2025 07:02:30

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- Importing TetGen files --- 25-Apr-2025 07:02:30
--- Done --- 25-Apr-2025 07:02:30
```

Convert Units

```
meshOutput.nodes = meshOutput.nodes*0.01;
```

Convert to (.msh) Gmsh Format (Version 2 ASCII)

```
GmshFileName = 'SyntheticNetwork.msh';
[status] = writeGmsh(GmshFileName,meshOutput);
```

```
Gmsh Version 2 ASCII written
```

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');
```

Current working directory: /mnt/c/Users/homeuser/Documents/GitHub/Tube2FEM/caseStudies/caseStudy6_SyntheticNetwork/Mesh

BC to FEniCS

```
cd ../FiniteElement
BCtoFEniCS(meshOutput,Fc,Vc);
```

BCs are written correctly in BC.txt

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');
```

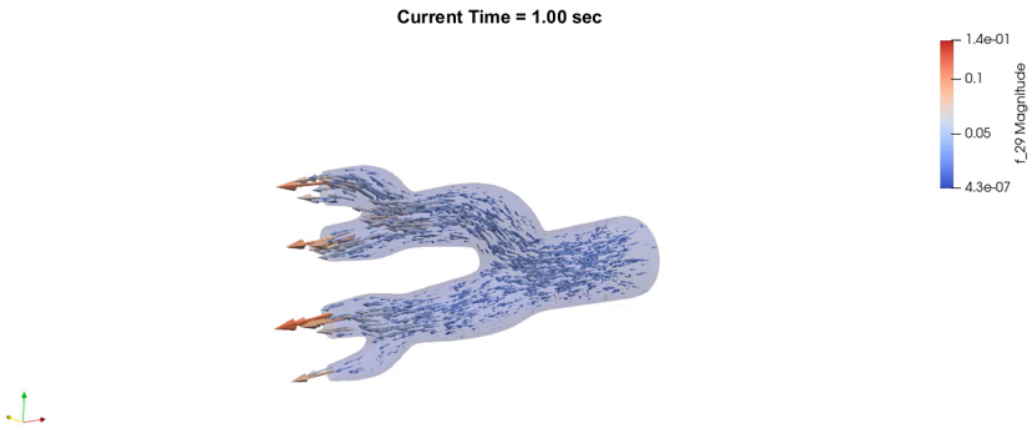
PostProcessing

```
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.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
```



Published with MATLAB® R2023a