Table of Contents

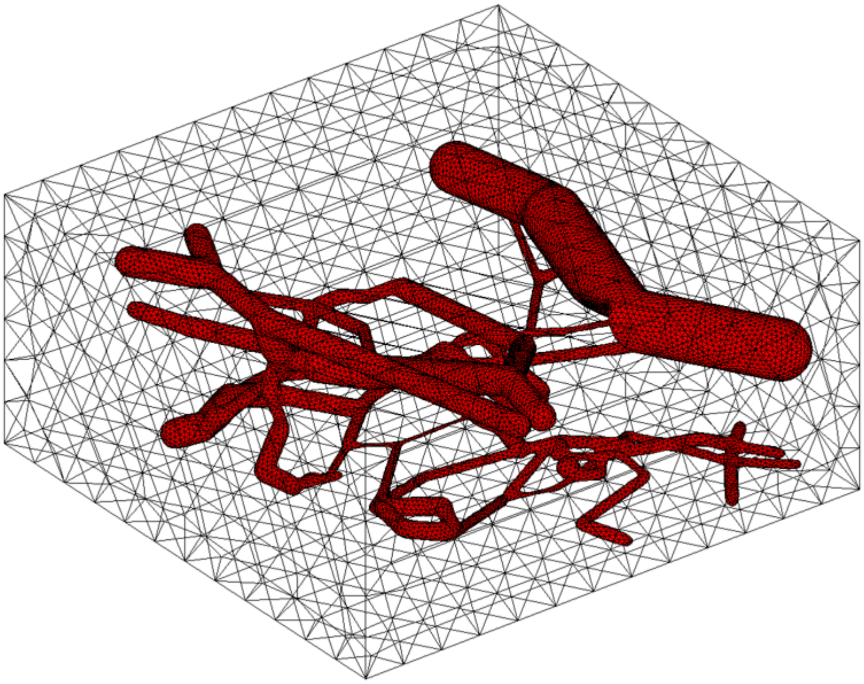
```
clear
clc
close all

fontSize=12;
faceAlpha1=0.8;
markerSize=40;
lineWidth=3;
cMap=blood(250);
```

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);
```

Read .stl file (this file was previously generated in caseStudy 3)

```
fileName = 'Input/fullNetworkOptimised.stl';
[stlStruct] = import_STL(fileName);

Fsurf=stlStruct.solidFaces{1};
Vsurf=stlStruct.solidVertices{1};

% Merging nodes (nodes are not merged in stl)
[Fsurf,Vsurf]=mergeVertices(Fsurf,Vsurf);

% Create Box
boxDim=[650 620 300]; %Width in each direction
pointSpacing=50; %Desired point spacing
[Fbox,Vbox,faceBoundaryMarker]=triBox(boxDim,pointSpacing);

% Move Box
Vx = 300;
Vy = 310;
Vz = 150;
MoveBox = ones(size(Vbox,1),size(Vbox,2));
MoveBox = [Vx*MoveBox(:,1),Vy*MoveBox(:,2),Vz*MoveBox(:,3)];
Vbox = Vbox+MoveBox;

% Visualisation
cFigure;
gpatch(Fsurf,Vsurf,'r','k');
axisGeom;
view([-35 17])
axis off
hold on
gpatch(Fbox,Vbox,'w','k',0.01);
axisGeom(gca,fontSize);
xlabel('X (\mum)')
ylabel('Y (\mum)')
zlabel('Z (\mum)')
zoom(1.1)
%print(gcf,'Surfaces.png','-dpng','-r600');
```



Join Element sets (Network + Box)

multiply by a scaling factor

```
Vbox = Vbox*1e-6;
Vsurf = Vsurf*1e-6;

% Join Vertices and Faces
[F,V,C]=joinElementSets({Fbox,Fsurf},{Vbox,Vsurf});

% Update labels
C(1:size(faceBoundaryMarker(:),1))=faceBoundaryMarker;
C(size(faceBoundaryMarker(:),1)+1:size(C,1)) = 7*ones((size(C,1)- ...
    size(faceBoundaryMarker,1)),1);

% Find interior points
[V_region1]=getInnerPoint({Fbox,Fsurf},{Vbox,Vsurf});
[V_region2]=getInnerPoint(Fsurf,Vsurf);

% Only |Box-Network| region is considered
% Net region is considered a hole
V_regions=V_region1;

% Volume parameters
[vol1]=tetVolMeanEst(Fbox,Vbox);
[vol2]=tetVolMeanEst(Fsurf,Vsurf);

regionTetVolumes=vol1; %Element volume settings
stringOpt='-pq1.2AaY'; %Tetgen options
modelName = 'test';
```

```
cd Mesh
```

Mesh inputs

```
%Create tetgen input structure
inputStruct.stringOpt=stringOpt; %Tetgen options
inputStruct.Faces=F; %Boundary faces
inputStruct.Nodes=V; %Nodes of boundary
inputStruct.faceBoundaryMarker=C;
inputStruct.regionPoints=V_regions; %Interior points for regions
% the microvascular network consist of 2 detached sub-networks
% therefore, we need 2 hole points
inputStruct.holePoints=[V_region2; [475e-6 288e-6 180e-6]];
inputStruct.regionA=regionTetVolumes; %Desired tet volume for each region
inputStruct.modelName = modelName;

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

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- TETGEN Tetrahedral meshing --- 25-Apr-2025 06:50:19

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- Writing SMESH file --- 25-Apr-2025 06:50:19

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 06:50:19

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 06:50:19
Opening C:\Program Files\MATLAB\GIBBON-master\data\temp\test.smesh.
Delaunizing vertices...
Delaunay seconds:  0.076
Creating surface mesh ...
Surface mesh seconds:  0.02
Recovering boundaries...
Boundary recovery seconds:  0.065
Removing exterior tetrahedra ...
Spreading region attributes.
Exterior tets removal seconds:  0.032
```

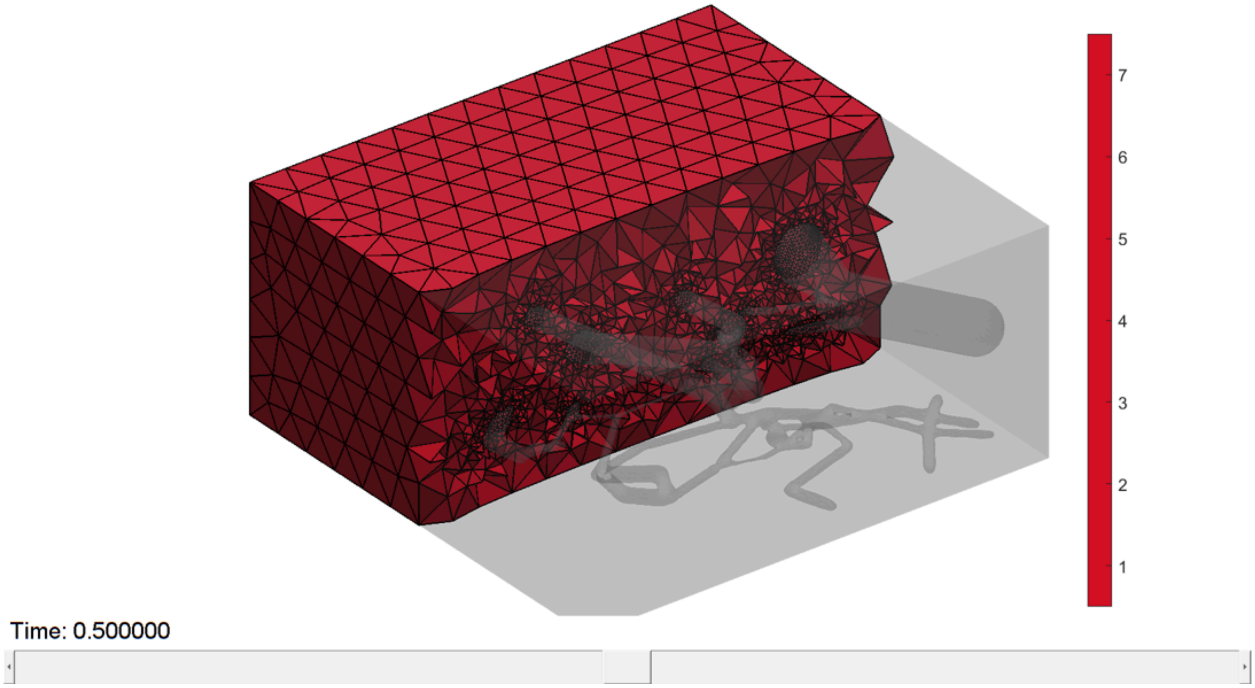
```
Recovering Delaunayness...
Delaunay recovery seconds:  0.023
Refining mesh...
Refinement seconds:  0.688
Optimizing mesh...
Optimization seconds:  0.054

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:  0.316
Total running seconds:  1.275

Statistics:

  Input points: 17260
  Input facets: 34572
  Input segments: 51858
  Input holes: 2
  Input regions: 1

  Mesh points: 46345
  Mesh tetrahedra: 241620
  Mesh faces: 500526
  Mesh faces on exterior boundary: 34572
  Mesh faces on input facets: 34572
  Mesh edges on input segments: 51858
  Steiner points inside domain: 29085

--- Done --- 25-Apr-2025 06:50:20

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- Importing TetGen files --- 25-Apr-2025 06:50:20
--- Done --- 25-Apr-2025 06:50:21
```

Mesh Output

```
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

% Update Element Material ID (label)
meshOutput.elementMaterialID = ones(size(CE,1),1);
```

Visualization

```
hf=cFigure; hold on;

% Visualizing using |meshView|
optionStruct.hFig=hf;
meshView(meshOutput,optionStruct);
axisGeom(gca,fontSize);
gdrawnow;
axis off
zoom(1.1)
%print(gcf,'Surfaces.png','-dpng','-r600');
```



Write version 2 ASCII .msh mesh format

```
GmshFileName = 'Secomb3D1DGmshFormat.msh';
[status] = writeGmsh(GmshFileName,meshOutput);
```

```
Gmsh Version 2 ASCII written
```

Convert .msh to .xdmf via meshio

```
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/";

% Get wsl mesh path
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);

% Combine two scripts (mesh info & general conversion script in src)
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');

cd ../simOutput_1D
```

```
Current working directory: /mnt/c/Users/homeuser/Documents/GitHub/Tube2FEM/
caseStudies/caseStudy4_3D-1D/Mesh
```

Read Simulation Results

Reading 1D Finite Volume simulation outputs

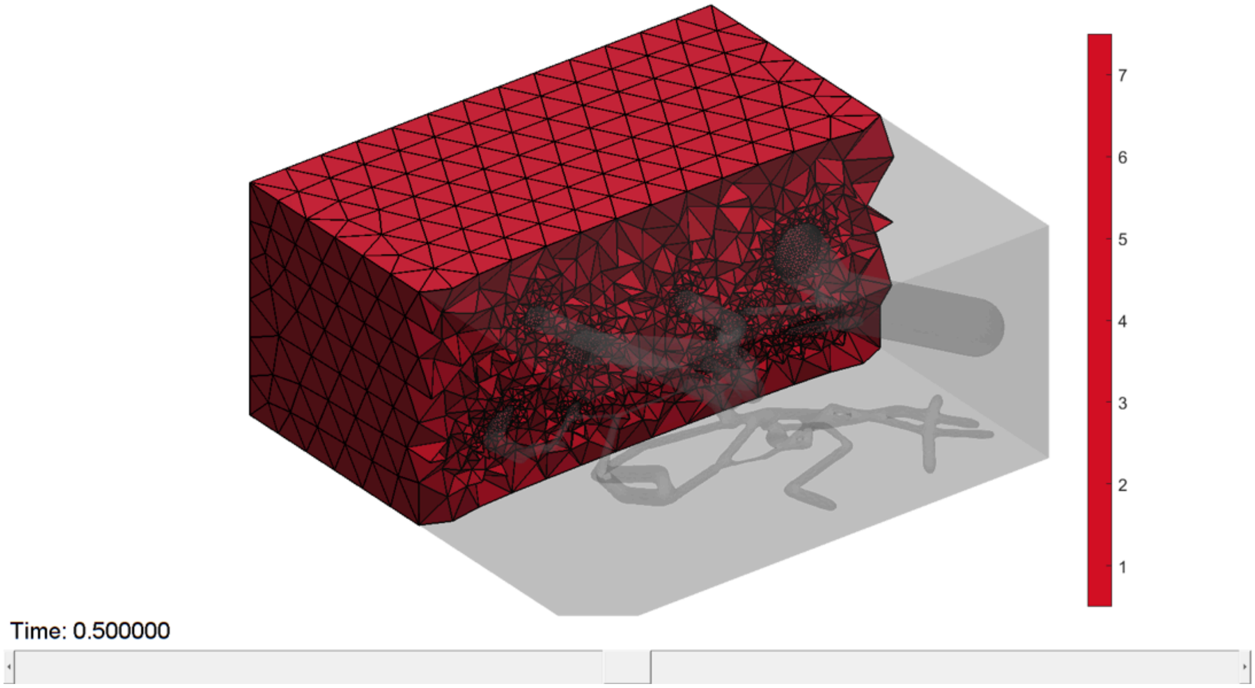
```
field1 = xlsread('@1.xlsx');
field2 = xlsread('@2.xlsx');
field3 = xlsread('@3.xlsx');
field4 = xlsread('@4.xlsx');
field5 = xlsread('@5.xlsx');
field6 = xlsread('@6.xlsx');

% Organise data differently
vertexCoor = field1;
connectivity = field2;
m = size(connectivity,1);

CellArray_V = cell(1,m);
CellArray_E = cell(1,m);
CellArray_O = cell(1,m);

for i = 1:m-1
    startNode = connectivity(i,1)+1;
    startNodeAfter = connectivity(i+1,1)+1;
    endNode = connectivity(i,2)+1;
    startNodeCoor = [vertexCoor(startNode,1) vertexCoor(startNode,2)
 vertexCoor(startNode,3)] ;
```

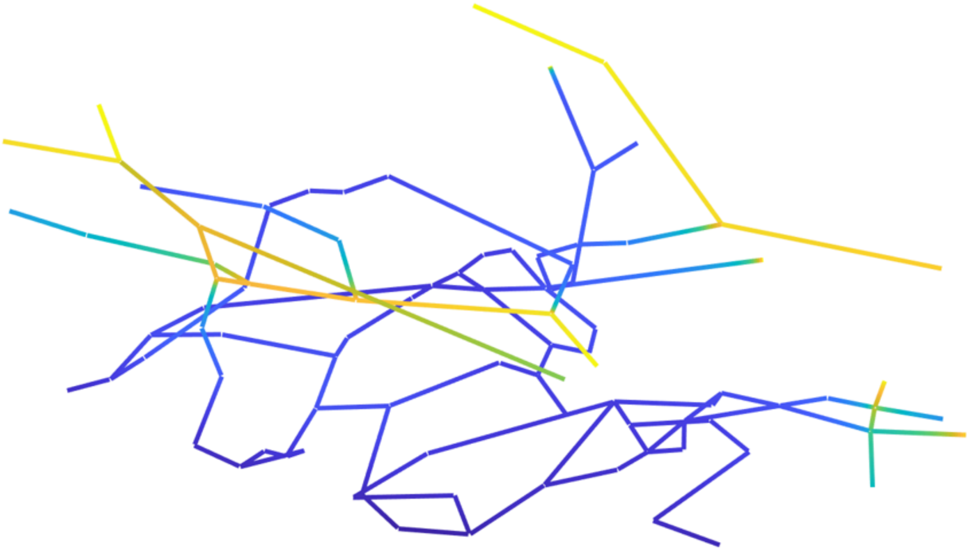
```
    startNodeCoorAfter = [vertexCoor(startNodeAfter,1)
 vertexCoor(startNodeAfter,2) vertexCoor(startNodeAfter,3)];

    rowIndexStart= find(ismember(field4,startNodeCoor,'rows'));
    rowIndexStartAfter=find(ismember(field4,startNodeCoorAfter,'rows'));

        if rowIndexStartAfter(1) == rowIndexStart(1)
            rowIndexStart = rowIndexStart(1);
            rowIndexStartAfter = rowIndexStartAfter(2);
        else
            rowIndexStart = rowIndexStart(1);
            rowIndexStartAfter = rowIndexStartAfter(1);
        end

    CellArray_V{1,i} = field4(rowIndexStart:rowIndexStartAfter-1,1:3);
    CellArray_O{1,i} = field6(rowIndexStart:rowIndexStartAfter-1,1);
    n = size(field4,1);
    field4 = field4(rowIndexStartAfter:n,1:3);
    field6 = field6(rowIndexStartAfter:n,1);
end

CellArray_V{1,m} = field4;
CellArray_O{1,m} = field6;

for i=1:m
    sizei = size(CellArray_V{1,i},1);
    E1 = 1:sizei-1;
    E1 = E1';
    E2 = 2:sizei;
    E2 = E2';
    CellArray_E{1,i} = [E1 E2];
end

[E,Vske]=joinElementSets(CellArray_E,CellArray_V);
[E,Vske,ind1]=mergeVertices(E,Vske);

% Shifting network in the (x,y) plane by 30 units in each direction
Vske = Vske +
 30*[ones(size(Vske,1),1),ones(size(Vske,1),1),zeros(size(Vske,1),1)];
% Multiplying by a scaling factor
Vske = Vske*10^-6;

% Reading Oxygen value from 1D simulation
oxygenSke = CellArray_O{1,1};
for i = 2:m
    oxygenSke = [oxygenSke; CellArray_O{1,i}];
end
oxygenSke=oxygenSke(ind1);

% Unit conversion
oxygenSke=oxygenSke*4.2e-6/80;
```

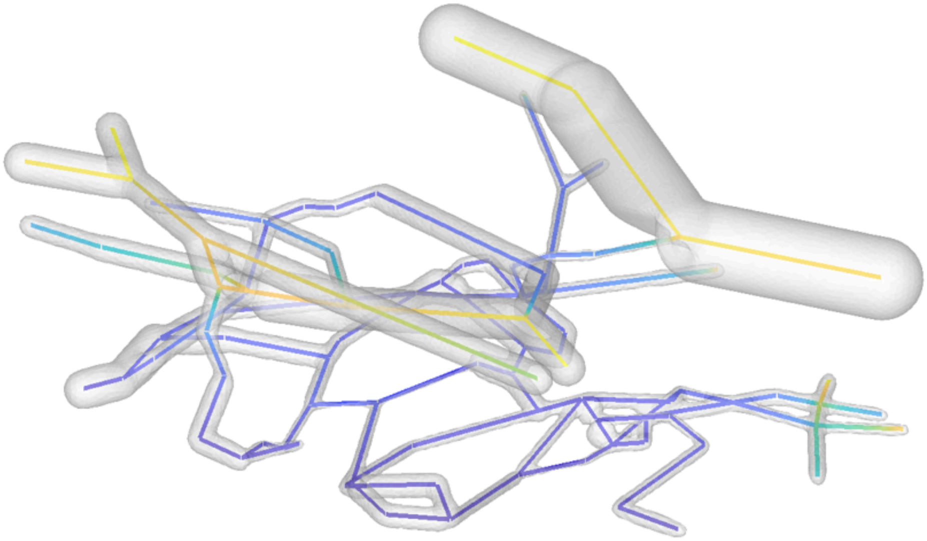
Visualisation

Visualtion of the 1D oxygen Field from Simulation

```
cFigure;
gpatch(E,Vske,'none',oxygenSke,0,3);
axisGeom;
camlight headlight;
drawnow;
gpatch(E,Vske,'none',oxygenSke,0,3);
axis off
zoom(1.3)
%print(gcf,'1Dsimulation.png','-dpng','-r600');
% Visualisation of the surface mesh and the 1D network
cFigure;
gpatch(Fsurf,Vsurf,'w','none',0.4);
axisGeom;
camlight headlight;
hold on
gpatch(E,Vske,'none',oxygenSke,0,3);
drawnow
axis off
zoom(1.3)
% print(gcf,'1DsimulationInSTL.png','-dpng','-r600');
```



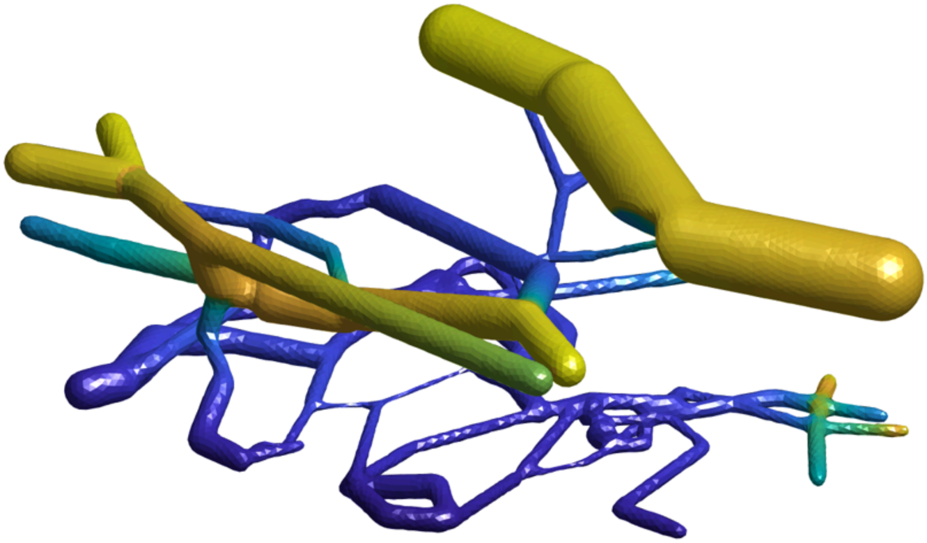




Projection back from 1D segments to 3D Surface Mesh

```
%minInd is the index of the vertices in the skeleton
[D,minInd]=minDist(Vsurf,Vske);
oxygenSurf = oxygenSke(minInd);
```

```
%Visualisation
[oxygenSurfFaces]=vertexToFaceMeasure(Fsurf,oxygenSurf);
faceAlpha=1;
cFigure;
patch('faces',Fsurf,'vertices',Vsurf,'FaceColor','flat', ...
    'CData',oxygenSurfFaces,'FaceAlpha',faceAlpha,'EdgeColor','none');
axisGeom;
camlight ;
drawnow;
axis off
zoom(1.3)
% print(gcf,'BC.png','-dpng','-r600');
```



Exporting Vsurf and oxygenSurf

```
ExportBCs(FEFolder,Vsurf,oxygenSurf);
```

Run FEniCS 3D-1D FEniCS script

```
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,"/FEniCS_3D-1D.py");
wslFEPath= sprintf('"%s"', wslFEPath);

% ----Create Info.txt -----%
fid1 = fopen('Info.txt','wt');
meshPath1 = sprintf('"%s"', strcat(rootPath,newMeshPath,"/Tetra.xdmf"));
meshPath2 = sprintf('"%s"', strcat(rootPath,newMeshPath,"/Tri.xdmf"));
outputPath = sprintf('''%s''',strcat(rootPath,newOutPath,"/solution.pvd"));
bcpPath = sprintf('''%s''',strcat(rootPath,newFEPath,"/BC_points.txt"));
bcvPath = sprintf('''%s''',strcat(rootPath,newFEPath,"/BC_values.txt"));

fprintf(fid1,strcat('tetraFileName=',meshPath1));
fprintf(fid1,'\n');
fprintf(fid1,strcat('triFileName=',meshPath2));
fprintf(fid1,'\n');
fprintf(fid1,strcat('outputFileName=',outputPath));
fprintf(fid1,'\n');
fprintf(fid1,strcat('BCpoints=',bcpPath));
fprintf(fid1,'\n');
fprintf(fid1,strcat('BCvalues=',bcvPath));
fprintf(fid1,'\n');
fclose(fid1);

script1 = fileread('Info.txt');
script2 = fileread('../../../src/FiniteElement/FEniCS_3D-1D.py');

% Join Info.txt with 3D-1D FEniCS file in the src/FiniteElement
fid2 = fopen('FEniCS_3D-1D.py','wt');
fprintf(fid2,script1);
fprintf(fid2,'\n');
fprintf(fid2,script2);
fprintf(fid2,'\n');
fclose(fid2);
```

```
% Run FEniCS_3D-1D in the case Study directory
wslPath = '"C:\Windows\System32\wsl.exe"';
runString = strcat(wslPath,' python3',append(' ',wslFEPath));
[runStatus,runOut]=system(runString,'-echo');
```

```
Solving linear variational problem.
```

Published with MATLAB® R2023a