Table of Contents

```
close all force
clear all
clc
```

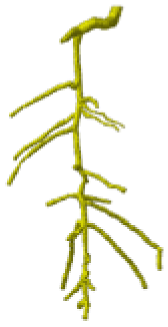
Adding Tube2FEM src directories to path

```
CurrentFolder = pwd;
TopFolder = fileparts(pwd);
TopTopFolder = fileparts(fileparts(pwd));
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 the segmented Root tomography (.tif format)

```
VolTif = tiffreadVolume('Input\RootSoil_tomo.tif');
```

```
% Visulisation using volshow + edit visualisation settings
V = volshow(fliplr(VolTif));
V.Parent.BackgroundColor = [1 1 1];
V.Parent.BackgroundGradient='off';
V.RenderingStyle = 'GradientOpacity';
V.Parent.LightColor = [1 1 0];
V.Parent.LightPositionMode = 'left';
V.Parent.OrientationAxes = 'off';
V.Parent.CameraPosition = [742 -471 263];
```



Get surface mesh (.stl format) from the Root segmented image

```
logicVoxels=(VolTif>0);
[Frs,Vrs]=im2patch(VolTif,logicVoxels);
[Frs,Vrs]=patch2tri(Frs,Vrs);

fig = cFigure; set(fig, 'Units', 'normalized', 'OuterPosition', [0 0 1 1]);
subplot(1,2,1) % Surface mesh view
gpatch(Frs,Vrs,'w','k');
axisGeom;
axis off
view([138,14])

subplot(1,2,2) % Zoomed in view
gpatch(Frs,Vrs,'w','k');
axisGeom;
axis off
view([138,14])
zoom(5)
ax = gca;
camPos = get(ax, 'CameraPosition');
camTgt = get(ax, 'CameraTarget');
```

```
shift = [0, 0, 180];
% Apply pan
set(ax, 'CameraPosition', camPos + shift);
set(ax, 'CameraTarget', camTgt + shift);
```

Remeshing the extracted surface mesh using Geogram library

(using ggremesh function in GIBBON)

```
optionStruct1.nb_pts= 100000; % Number of vertices in the remeshed surface
 mesh
[Frr,Vrr]=ggremesh(Frs,Vrs,optionStruct1);

% Multiplying the vertices matrix by a scaling factor
scaleFactor = 0.0002/0.9;
Vrr = Vrr*scaleFactor;

fig = cFigure; set(fig, 'Units', 'normalized', 'OuterPosition', [0 0 1 1]);
subplot(1,2,1)
gpatch(Frr,Vrr,'y','k')
axisGeom;
view([138,14])
axis off

subplot(1,2,2)
gpatch(Frr,Vrr,'y','k')
axisGeom;
axis off
view([138,14])
zoom(5)
ax = gca;
```

```
camPos = get(ax, 'CameraPosition');
camTgt = get(ax, 'CameraTarget');
shift = [0, 0, 180*scaleFactor];
% Apply pan
set(ax, 'CameraPosition', camPos + shift);
set(ax, 'CameraTarget', camTgt + shift);
```

Smoothing the surface mesh

```
nSteps = 2;
cPar2.Method='HC'; % Smooth method
cPar2.n=nSteps; % Number of iterations
[Vrr]=patchSmooth(Frr,Vrr,[],cPar2);

fig = cFigure; set(fig, 'Units', 'normalized', 'OuterPosition', [0 0 1 1]);
subplot(1,2,1)
gpatch(Frr,Vrr,'y','k')
axisGeom;
view([138,14])
axis off

subplot(1,2,2)
gpatch(Frr,Vrr,'y','k')
axisGeom;
axis off
view([138,14])
zoom(5)
ax = gca;
camPos = get(ax, 'CameraPosition');
camTgt = get(ax, 'CameraTarget');
shift = [0, 0, 180*scaleFactor];
% Apply pan
```

```
set(ax, 'CameraPosition', camPos + shift);
set(ax, 'CameraTarget', camTgt + shift);
```

Save surface mesh as .stl

```
fileName= 'Mesh\smoothMesh.stl';
patch2STL(fileName,Vrr,Frr,[],'SmoothMesh');
```

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

Repair Mesh

```
MeshPath = strcat(CurrentFolder,'\Mesh');
MeshPath = strrep(MeshPath, '\', '/');
% Run the RepairSurfaceMesh.py python script using pyrunf
% input: the previously saved smoothMesh.stl in the Mesh directory
% output: repairedMesh.stl in the Mesh directory
pyrunfile("RepairSurfaceMesh.py",Path=MeshPath)

% Read the repaired surface mesh back in Matlab
cd Mesh
fileName= 'repairedMesh.stl';
[stlStruct] = import_STL(fileName);
F=stlStruct.solidFaces{1}; %Faces
V=stlStruct.solidVertices{1}; %Vertices
[Frr,Vrr]=mergeVertices(F,V); % Merging nodes
```

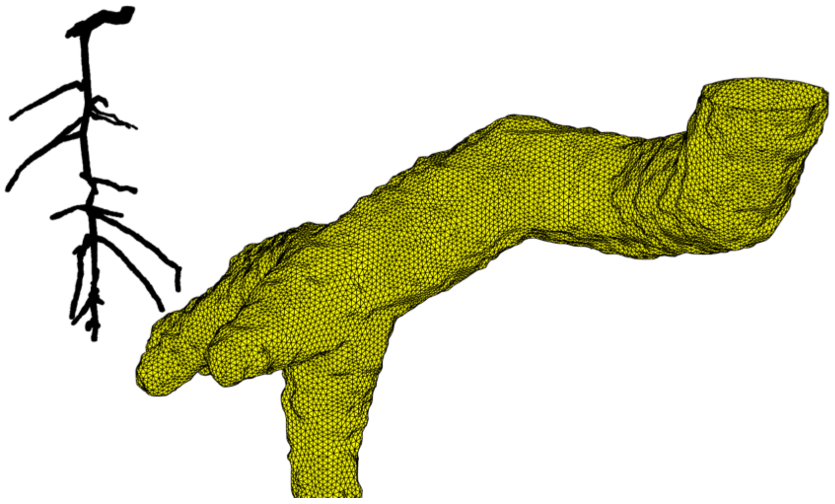
```
Mesh has been fixed and saved in fixed_mesh.stl'
```

Double check if the surface mesh is watertight after repairing

```
mesh = surfaceMesh(Vrr,Frr);
TF = isWatertight(mesh); % TF=1 means it is watertight
```

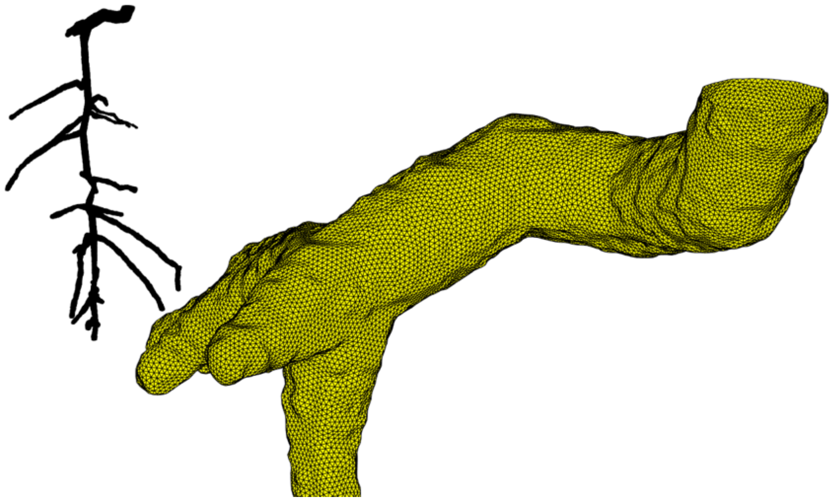
Slicing the surface mesh to define a transpiration boundary condition

```
% The slicing plane is be defined by a point (Pcut) and a normal vector (n)
Pcut = [0.04,0.04,0.15];
C=[];
snapTolerance=mean(patchEdgeLengths(F,V))/100;
n = [0,0,1];
% Use triSurfSlice GIBBON function
[Frr,Vrr,~,logicSide]=triSurfSlice(Frr,Vrr,C,Pcut,n,snapTolerance);
Frr = Frr(logicSide,:); % keep on the faces that are below the slicing plane


fig = cFigure; set(fig, 'Units', 'normalized', 'OuterPosition', [0 0 1 1]);
subplot(1,2,1)
gpatch(Frr,Vrr,'y','k')
axisGeom;
view([138,14])
axis off

subplot(1,2,2)
gpatch(Frr,Vrr,'y','k')
axisGeom;
axis off
view([138,14])
zoom(10)
ax = gca;
camPos = get(ax, 'CameraPosition');
camTgt = get(ax, 'CameraTarget');
shift = [-50*scaleFactor, 0, 220*scaleFactor];
% Apply pan
set(ax, 'CameraPosition', camPos + shift);
set(ax, 'CameraTarget', camTgt + shift);
```



Remeshing open surface mesh

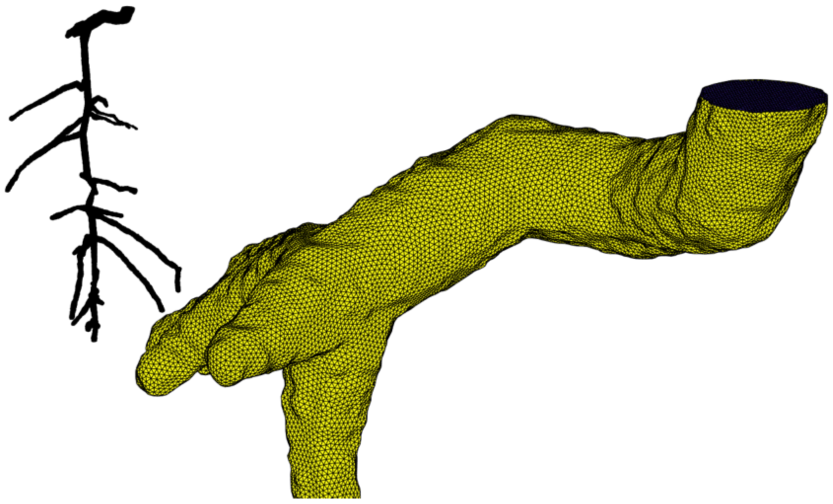
```
nb_pts = size(Vrr,1);
optionStruct2.nb_pts = nb_pts;
[Frr,Vrr] = ggremesh(Frr,Vrr,optionStruct2);

fig = cFigure; set(fig, 'Units', 'normalized', 'OuterPosition', [0 0 1 1]);
subplot(1,2,1)
gpatch(Frr,Vrr,'w','k');
axisGeom;
view([138,14])
axis off

subplot(1,2,2)
gpatch(Frr,Vrr,'y','k')
axisGeom;
axis off
view([138,14])
zoom(10)
ax = gca;
camPos = get(ax, 'CameraPosition');
camTgt = get(ax, 'CameraTarget');
shift = [-50*scaleFactor, 0, 220*scaleFactor];
% Apply pan
set(ax, 'CameraPosition', camPos + shift);
set(ax, 'CameraTarget', camTgt + shift);
```



Close holes and labelling Using closeHolesAndLabelling Tube2FEM function

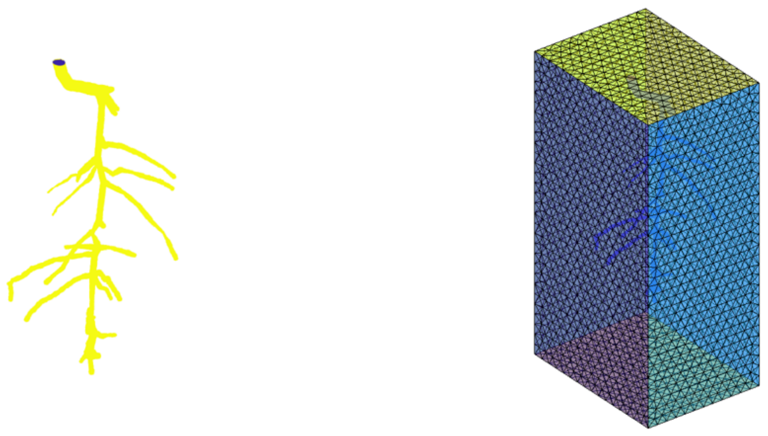
```
fig = cFigure; set(fig, 'Units', 'normalized', 'OuterPosition', [0 0 1 1]);
Crr = ones(size(Frr,1),1);
[Frr,Vrr,Crr]=closeHolesAndLabelling(Frr,Vrr,Crr);

% Edit the labels
Crr(Crr==1)=-1;
Crr(Crr==2)=1;
Crr(Crr==-1)=2;

subplot(1,2,1)
gpatch(Frr,Vrr,Crr,'k');
axisGeom
view([138,14])
axis off

subplot(1,2,2)
gpatch(Frr,Vrr,Crr,'k');
axisGeom;
axis off
view([138,14])
zoom(10)
ax = gca;
campos = get(ax, 'CameraPosition');
camTgt = get(ax, 'CameraTarget');
shift = [-50*scaleFactor, 0, 220*scaleFactor];
% Apply pan
set(ax, 'CameraPosition', camPos + shift);
set(ax, 'CameraTarget', camTgt + shift);
```

```
i =

     0


i =

    81
```



Create Box that surrounds the Root surface mesh

```
boxDim=[0.06 0.08 0.15]; % Width in each direction
pointSpacing=0.005; % Desired point spacing
[Fbox,Vbox,Cbox]=triBox(boxDim,pointSpacing);

% Move Box
Vx = 0.05;
Vy = 0.04;
Vz = 0.078;
MoveBox = ones(size(Vbox,1),size(Vbox,2));
MoveBox = [Vx*MoveBox(:,1),Vy*MoveBox(:,2),Vz*MoveBox(:,3)];
Vbox = Vbox+MoveBox;

% Visualisation
fig = cFigure; set(fig, 'Units', 'normalized', 'OuterPosition', [0 0 1 1]);
% Crr = ones(size(Frr,1),1);
subplot(1,2,1)
patch('faces',Frr,'vertices',Vrr,'FaceColor','flat','CData',Crr, ...
    'FaceAlpha',1,'EdgeColor','none');
axisGeom;
axis off

subplot(1,2,2)
patch('faces',Frr,'vertices',Vrr,'FaceColor','flat','CData',Crr, ...
    'FaceAlpha',1,'EdgeColor','none');
axisGeom;
% Cbox = 2*ones(size(Fbox,1),1);
patch('faces',Fbox,'vertices',Vbox,'FaceColor','flat','CData',Cbox, ...
    'FaceAlpha',0.5,'EdgeColor','k');
axis off
```



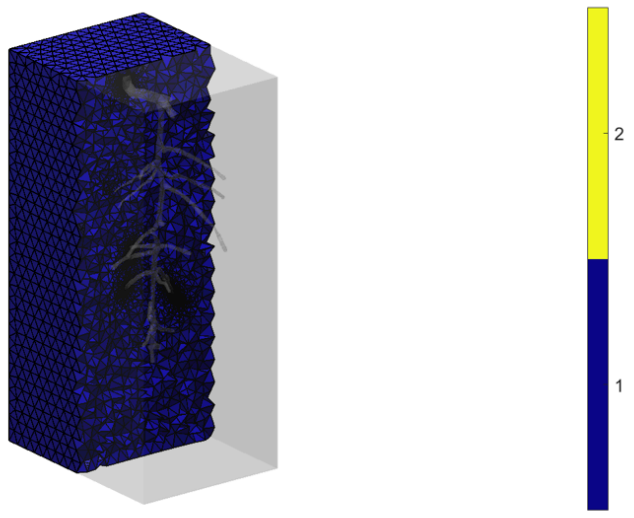
Generate Volumetric mesh of the Root-Soil system

```
Vnet = Vrr; % Vertices (net stands for network, root in this case)
Fnet = Frr; % Faces
Cnet = Crr; % Labels

% Join Element sets between network and box
[F,V,C]=joinElementSets({Fbox,Fnet},{Vbox,Vnet},{Cbox,Cnet});

% Update the labels for the joined system
C(1:size(Cbox(:),1))=Cbox;
C(size(Cbox(:),1)+1:size(C,1)) = 6*ones((size(C,1)-size(Cbox,1)),1)+Cnet;

% Find interior points
[V_region1] = [0.03 0.01 0.03]; % A random point inside the box region
[V_region2] = [0.042 0.04 0.148]; % A random point inside the root region
V_regions= [V_region1;V_region2];
```

```
% Volume parameters
[vol1]=tetVolMeanEst(Fbox,Vbox);
[vol2]=tetVolMeanEst(Fnet,Vnet);

regionTetVolumes=[vol1;vol2]; % Element volume settings
stringOpt='-pq1.2AaY'; % Tetgen options
modelName = 'test';

% 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
inputStruct.regionA=regionTetVolumes; % Desired tet volume for each region
inputStruct.modelName = modelName;

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

% 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

% Change the volumetric mesh elementMaterialID or labels
for i=1:size(meshOutput.elementMaterialID,1)
    if meshOutput.elementMaterialID(i)==-2
        meshOutput.elementMaterialID(i) = 1;
    elseif meshOutput.elementMaterialID(i)==-3
        meshOutput.elementMaterialID(i) = 2;
    end
end

% Visualization
hf=cFigure; set(hf, 'Units', 'normalized', 'OuterPosition', [0 0 1 1]);
hold on;

% Visualizing using meshView
optionStruct.hFig=hf;
meshView(meshOutput,optionStruct);
colormap(matplotlibColormap(3,2))
axis off
view([-37,21])
camlight('left')
% print(gcf,'RootSoilEmbeddedMesh.png','-dpng','-r900');
% print(gcf,'RootSoilEmbeddedMeshZoom2.png','-dpng','-r1500');
```

%%%

```
--- TETGEN Tetrahedral meshing --- 25-Apr-2025 05:51:58

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- Writing SMESH file --- 25-Apr-2025 05:51:58

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 05:51:59

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 05:51:59
Opening C:\Program Files\MATLAB\GIBBON-master\data\temp\test.smesh.
Delaunizing vertices...
Delaunay seconds:  0.626
Creating surface mesh ...
Surface mesh seconds:  0.232
Recovering boundaries...
Boundary recovery seconds:  0.853
Removing exterior tetrahedra ...
Spreading region attributes.
Exterior tets removal seconds:  0.104
Recovering Delaunayness...
Delaunay recovery seconds:  0.356
Refining mesh...
Refinement seconds:  20.272
Optimizing mesh...
Optimization seconds:  2.031

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:  7.839
Total running seconds:  32.317

Statistics:

  Input points: 102740
  Input facets: 205472
  Input segments: 308208
  Input holes: 0
  Input regions: 2
  Mesh points: 734968
  Mesh tetrahedra: 4660632
  Mesh faces: 9323726
  Mesh faces on exterior boundary: 4924
  Mesh faces on input facets: 205472
  Mesh edges on input segments: 308208
  Steiner points inside domain: 632228

--- Done --- 25-Apr-2025 05:52:32

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- Importing TetGen files --- 25-Apr-2025 05:52:32
--- Done --- 25-Apr-2025 05:52:38
```



Write version 2 ASCII .msh mesh format

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

```
Gmsh Version 2 ASCII written
```

Convert .msh to .xdmf via meshio (using python installed on WSL)

```
meshPath = strcat(CurrentFolder,'\Mesh');
meshPath = strrep(meshPath, '\', '/');

% Split the path into parts
pathParts = strsplit(meshPath, '/');

% Reconstruct the path without the first directory
% (!) WSL and Command Prompt reads paths differently (!)
newMeshPath = strjoin(pathParts(2:end), '/');
rootPath = "../../../../../../../../../../../../../../../../../../mnt/c/";
wslMeshPath = strcat(rootPath,newMeshPath,"/meshioConvertMesh.py");
wslMeshPath= sprintf('"%s"', wslMeshPath);

% Create a meshInfo txt file that serves to pass the input to the py script
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 the two scripts
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);

% Run python script on WSL
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/caseStudy2_RootSoil/Mesh
```

Published with MATLAB® R2023a