DEMO_create_run_import_FEBIO_spheres

Table of Contents

Below is a demonstration for: 1) The use of TETgen for meshing based on surface geometry 2) The specification of boundary conditions for FEBio 3) The exporting of .feb files 4) Running an FEBio job with MATLAB 5) Importing FEBio results into MATLAB

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

Plot settings

```
figColor='w'; figColorDef='white';
fontSize=15;
faceAlpha1=0.5;
faceAlpha2=0.5;
edgeColor=0.25*ones(1,3);
edgeWidth=1.5;

% path names
filePath=mfilename('fullpath');
savePath=fullfile(fileparts(filePath),'data','temp');
```

Defining the surface models

The model will consists of two spheres one contained within the other defining two material regions. A stiff core and a soft outer later.

Control parameters for surface models

```
r1=2; %Outer sphere radius
numRefine1=3; %Number of refinement steps from icosahedron
faceBoundMarker1=2; %Face marker for outer sphere

r2=1.3; %Inner sphere radius
numRefine2=2; %Number of refinement steps from icosahedron
faceBoundMarker2=3; %Face marker for inner sphere
```

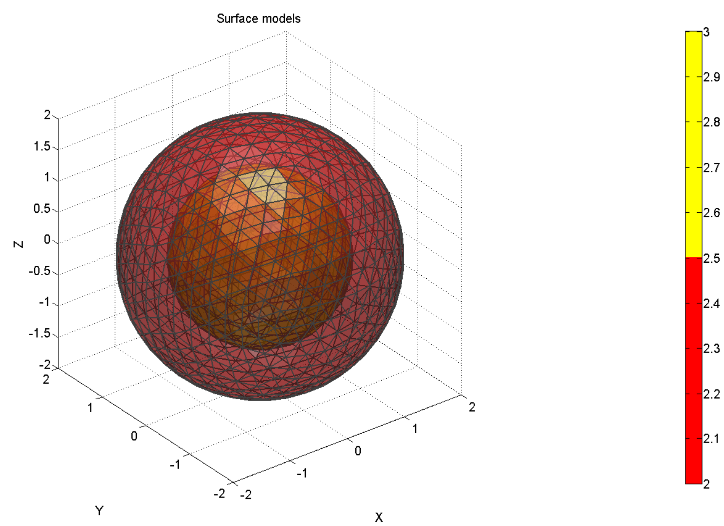
Building the spheres using `geoSphere` function

```
[F1,V1,~]=geoSphere(numRefine1,r1);
```

```
[F2,V2,~]=geoSphere(numRefine2,r2);

% Merging the model geometries into a single set
V=[V1;V2]; %Joining nodes
F=[F1;F2+size(V1,1)]; %Joining faces
faceBoundaryMarker=[faceBoundMarker1*ones(size(F1,1),1); faceBoundMarker2*ones(siz
```

Plotting surface models

```
hf=figuremax(figColor,figColorDef);
title('Surface models','FontSize',fontSize);
xlabel('X','FontSize',fontSize); ylabel('Y','FontSize',fontSize); zlabel('Z','Font
hold on;

patch('Faces',F,'Vertices',V,'FaceColor','flat','CData',faceBoundaryMarker,'FaceAl
colormap(autumn(2));
colorbar;
camlight headlight;
set(gca,'FontSize',fontSize);
view(3); axis tight;  axis equal;  grid on;
```



CREATING A SOLID TETRAHEDRAL MESH USING TETGEN

First region points need to be defined. These represent a list of arbitrary coordinates for points inside the regions. 1 point per region is specified. For the example here the points are easily specified. Sometimes a raytracing algorythm or the use of the `triSurf2Im` function is required to find interior points.

```
V_regions=[0 0 (r1+r2)/2;0 0 0;]; % Define region points
```

Next holes are defined. These are similar to regions. However holes, as the name suggests, are regions that a not meshed and are left empty. This model does not contain holes so the list is empty

```
V_holes=[]; %Define hole points
```

For each region the mesh density parameter can be specified

```
regionA=[0.005 0.005]; % Regional mesh parameters
```

CREATING THE SMESH STRUCTURE. TetGen can mesh geometries from various mesh file formats. For the GIBBON toolbox .smesh files have been implemented. Below a structure is created that fully defines such as smesh file and the meshing settings for TetGen.

```
stringOpt='-pq1.2AaYQ';
modelName=fullfile(savePath,'tetGenModel');
smeshName=[modelName,'.smesh'];

smeshStruct.stringOpt=stringOpt;
smeshStruct.Faces=F;
smeshStruct.Nodes=V;
smeshStruct.holePoints=V_holes;
smeshStruct.faceBoundaryMarker=faceBoundaryMarker; %Face boundary markers
smeshStruct.regionPoints=V_regions; %region points
smeshStruct.regionA=regionA;
smeshStruct.minRegionMarker=2; %Minimum region marker
smeshStruct.smeshName=smeshName;
```

Mesh model using tetrahedral elements using tetGen (see: http://wias-berlin.de/software/tetgen/)

```
[meshOutput]=runTetGenSmesh(smeshStruct); %Run tetGen
```

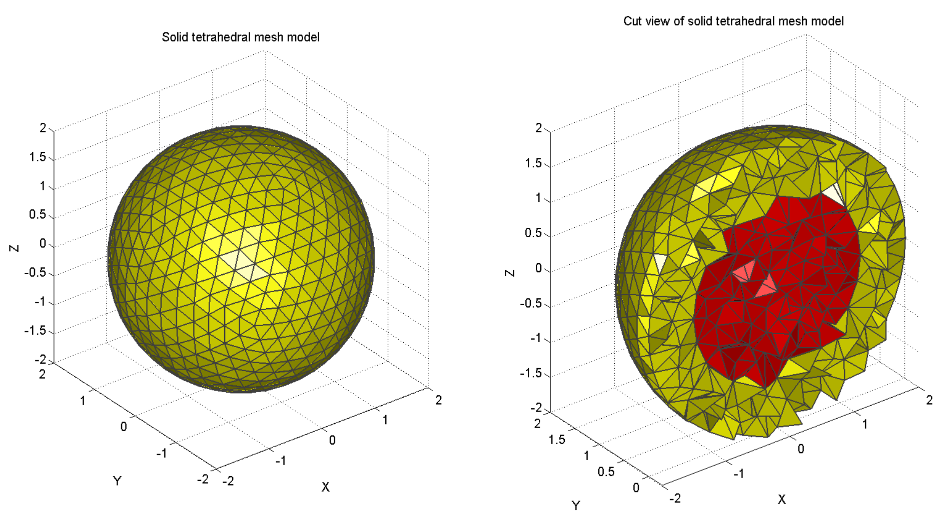
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- TETGEN Tetrahedral meshing --- 25-Mar-2014 13:02:53

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- Writing SMESH file --- 25-Mar-2014 13:02:53
----> Adding node field
----> Adding facet field
----> Adding holes specification
----> Adding region specification
--- Done --- 25-Mar-2014 13:02:53
--- Running TetGen for meshing --- 25-Mar-2014 13:02:53
Opening C:\Users\kmmoerman\00_WORK\05_MATLAB\gibbon\trunk\data\temp\tetGen
--- Done --- 25-Mar-2014 13:02:53

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- Importing TetGen files --- 25-Mar-2014 13:02:53
--- Done --- 25-Mar-2014 13:02:54
```

Accessing the model element and patch data

```
FT=meshOutput.faces;
VT=meshOutput.nodes;
C=meshOutput.faceMaterialID;
E=meshOutput.elements;
elementMaterialIndices=meshOutput.elementMaterialID;
```

Plotting the meshed geometry

```
hf1=figuremax(figColor,figColorDef);
subplot(1,2,1);
title('Solid tetrahedral mesh model','FontSize',fontSize);
xlabel('X','FontSize',fontSize); ylabel('Y','FontSize',fontSize); zlabel('Z','Font
hps=patch('Faces',FT,'Vertices',VT,'FaceColor','flat','CData',C,'lineWidth',edgeWi
view(3); axis tight;  axis equal;  grid on;
colormap(autumn);
camlight headlight;
set(gca,'FontSize',fontSize);

%Selecting half of the model to see interior
Y=VT(:,2); YE=mean(Y(E),2);
L=YE>mean(Y);
[Fs,Cs]=element2patch(E(L,:),C(L));

subplot(1,2,2);
title('Cut view of solid tetrahedral mesh model','FontSize',fontSize);
xlabel('X','FontSize',fontSize); ylabel('Y','FontSize',fontSize); zlabel('Z','Font
hps=patch('Faces',Fs,'Vertices',VT,'FaceColor','flat','CData',Cs,'lineWidth',edgeW
view(3); axis tight;  axis equal;  grid on;
colormap(autumn);
camlight headlight;
set(gca,'FontSize',fontSize);
drawnow;
```



DEFINE PRESCRIBED DISPLACEMENTS

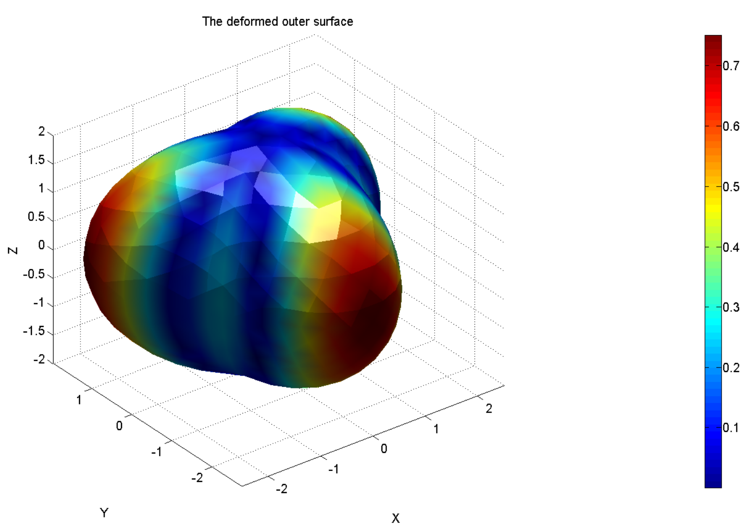
For this example the outer sphere nodes are subjected to a prescrived displacement

```
% Defining deformed boundary coordinates
deformationCase=2;
switch deformationCase
    case 1 %an ellipsoid in z-direction.
        stretchMagnitude=1.5;
        V1_def=V1;
```

```
        V1_def(:,3)=V1_def(:,3)*stretchMagnitude;
    case 2 %Star shaped in cross-section
        [PHI,THETA,R] = cart2sph(V1(:,1),V1(:,2),V1(:,3));
        freqDef=3;
        ampDef=0.5;
        ampDefDiff=0.25;
        R=R+(ampDef-ampDefDiff)+ampDef*sin(freqDef*PHI);
        V1_def=V1;
        [V1_def(:,1),V1_def(:,2),~]=sph2cart(PHI,THETA,R);
end

% Define boundary displacement values
bcPrescribedMagnitudes=(V1_def-V1);

% Define indices (node numbers) for the prescribed displacement
bcIndicesPrescribed=1:1:size(V1,1);
```

Plotting deformed outer surface

```
Cd=sqrt(sum(bcPrescribedMagnitudes.^2,2)); %Color towards displacement magnitude
hf=figuremax(figColor,figColorDef);
title('The deformed outer surface','FontSize',fontSize);
xlabel('X','FontSize',fontSize); ylabel('Y','FontSize',fontSize); zlabel('Z','Font
hold on;

patch('Faces',F1,'Vertices',V1_def,'FaceColor','flat','CData',Cd,'FaceAlpha',1);
colormap jet; colorbar; shading interp;
camlight headlight;
set(gca,'FontSize',fontSize);
view(3); axis tight;  axis equal;  grid on;
```



CONSTRUCTING FEB MODEL

```
% Defining file names
FEB_struct.run_filename=[modelName,'.feb']; %FEB file name
```

```
FEB_struct.run_logname=[modelName,'.txt']; %FEBio log file name

febMatID=elementMaterialIndices;
febMatID(elementMaterialIndices==-2)=1;
febMatID(elementMaterialIndices==-3)=2;

%Creating FEB_struct
FEB_struct.Geometry.Nodes=VT;
FEB_struct.Geometry.Elements={E}; %The element sets
FEB_struct.Geometry.ElementType={'tet4'}; %The element types
FEB_struct.Geometry.ElementMat={febMatID};

% DEFINING MATERIALS
k_factor=1000;

%Material 1
c1=1e-3;
k=c1*k_factor;
Mat1.type='Mooney-Rivlin';
Mat1.props={'c1','c2','k'};
Mat1.vals={c1,0,k};
Mat1.aniso_type='none';

%Material 2
c1=2e-3;
k=c1*k_factor;
Mat2.type='Mooney-Rivlin';
Mat2.props={'c1','c2','k'};
Mat2.vals={c1,0,k};
Mat2.aniso_type='none';

FEB_struct.Materials{1}=Mat1;
FEB_struct.Materials{2}=Mat2;

% %Adding BC information
% FEB_struct.Boundary.FixList={boundaryConditionNodeList};
% FEB_struct.Boundary.FixType={'xyz'};

FEB_struct.Boundary.PrescribeList={bcIndicesPrescribed,bcIndicesPrescribed,bcIndic
FEB_struct.Boundary.PrescribeType={'x','y','z'};

FEB_struct.Boundary.PrescribeValues={bcPrescribedMagnitudes(:,1),bcPrescribedMagni
FEB_struct.Boundary.LoadCurveIds=[1 1 1];

%Adding output requests
FEB_struct.Output.VarTypes={'displacement','stress','relative volume','shell thick

%Specify log file output
run_node_output_name=[FEB_struct.run_filename(1:end-4),'_node_out.txt'];
FEB_struct.run_output_names={run_node_output_name};
FEB_struct.output_types={'node_data'};
FEB_struct.data_types={'ux;uy;uz'};

%Control section
```

```
FEB_struct.Control.AnalysisType='static';
FEB_struct.Control.Properties={'time_steps','step_size',...
    'max_refs','max_ups',...
    'dtol','etol','rtol','lstol'};
FEB_struct.Control.Values={10,0.1,...
    25,0,...
    0.001,0.01,0,0.9};
FEB_struct.Control.TimeStepperProperties={'dtmin','dtmax','max_retries','opt_iter'
FEB_struct.Control.TimeStepperValues={1e-5, 0.1, 5, 5, 1};

%Load curves
FEB_struct.LoadData.LoadCurves.id=[1];
FEB_struct.LoadData.LoadCurves.type={'smooth'};
FEB_struct.LoadData.LoadCurves.loadPoints={[0 0;1 1]};
```

SAVING .FEB FILE

```
%CREATE XML DOCNODE
docNode=write_FEB_input(FEB_struct);
docNode=add_control_section_FEB(docNode,FEB_struct);
docNode=add_boundary_conditions_support_FEB(docNode,FEB_struct);
%     docNode=add_constraint_section_FEB(docNode,FEB_struct);
write_XML_no_extra_lines(FEB_struct.run_filename,docNode)% Saving XML file
set_output_request_FEBIO(FEB_struct.run_filename,FEB_struct.run_filename,FEB_struc
```

```
        Adding Material level
        Adding Geometry level
        ----> Adding node field
        ----> Adding element field
        ----> Adding tet4 element entries....
        Adding Output level
        ----> Adding Plotfile field
        Boundary condition level
        ----> Defining boundary conditions
```

RUNNING FEBIO JOB

```
%Set-up run-string for "external" (matlab will only start FEBio and not
%wait untill it is finished)
run_string=['start /min febio -i ',FEB_struct.run_filename,' -o ',FEB_struct.run_l
run_string_quit='taskkill /F /IM FEBio.exe /T';

FEBioRunStruct.run_string=run_string;
FEBioRunStruct.run_log_name=FEB_struct.run_logname;
FEBioRunStruct.disp_on=1;
FEBioRunStruct.disp_log_on=1;
FEBioRunStruct.run_string_quit=run_string_quit;
FEBioRunStruct.t_check=0.25; %Time for checking log file (dont set too small)
FEBioRunStruct.maxtpi=1e99; %Max analysis time

%--------------------------------------------------------------------
[rundFlag]=runMonitorFEBio(FEBioRunStruct);%START FEBio NOW!!!!!!!!
```

```
%-------------------------------------------------------------------
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--- STARTING FEBIO JOB --- 25-Mar-2014 13:03:06
Waiting for log file...
Proceeding to check log file...25-Mar-2014 13:03:06
------- converged at time : 0.1
------- converged at time : 0.2
------- converged at time : 0.3
------- converged at time : 0.4
------- converged at time : 0.5
------- converged at time : 0.6
------- converged at time : 0.7
------- converged at time : 0.8
------- converged at time : 0.9
------- converged at time : 1
--- Done --- 25-Mar-2014 13:03:15
```

IMPORTING NODAL DISPLACEMENT RESULTS

Importing nodal displacements from a log file

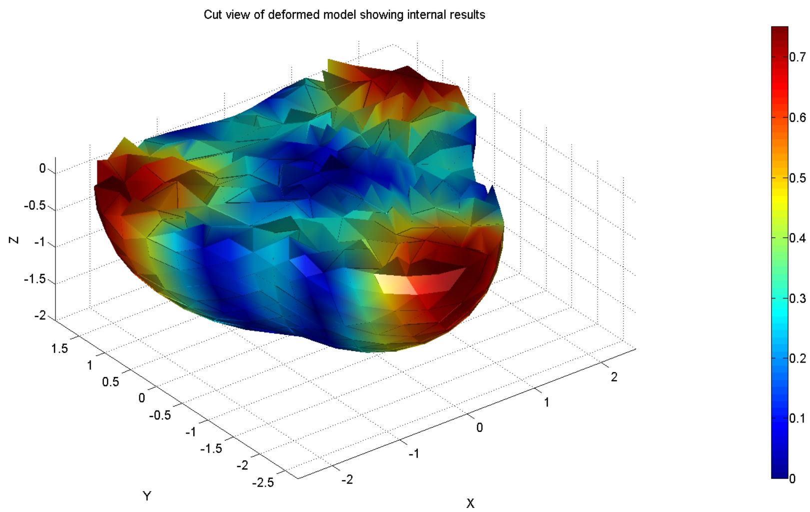
```
[~, N_disp_mat,~]=importFEBio_logfile(FEB_struct.run_output_names{1}); %Nodal disp

DN=N_disp_mat(:,2:end,end); %Final nodal displacements
```

CREATING NODE SET IN DEFORMED STATE

```
VT_def=VT+DN;
```

Plotting the meshed geometry

```
%Selecting half of the model to see interior
Z=VT(:,3); ZE=mean(Z(E),2);
L=ZE<mean(Z);
[Fs,~]=element2patch(E(L,:),[]);

Cs=sqrt(sum(DN.^2,2)); %Color towards displacement magnitude

hf1=figuremax(figColor,figColorDef);
title('Cut view of deformed model showing internal results','FontSize',fontSize);
xlabel('X','FontSize',fontSize); ylabel('Y','FontSize',fontSize); zlabel('Z','Font

hps=patch('Faces',Fs,'Vertices',VT_def,'FaceColor','flat','FaceVertexCData',Cs);

view(3); axis tight;  axis equal;  grid on;
colormap jet; colorbar; shading interp;
camlight headlight;
set(gca,'FontSize',fontSize);
drawnow;
```



GIBBON

Kevin M. Moerman (kevinmoerman@hotmail.com)

Published with MATLAB® R2014a