Table of Contents


```
function MRIhexmesh_interpv7p3_Windows2(default)

%MRIhexmesh_interp Generates required files for MREv7 from MRE data.
%   Interpolates data to an approximate number of nodes per wavlength,
%   based on prior estiamtes of the stiffness. Builds Hex27 mesh manually,
%   i.e. does not require a 'template mesh'.
%   If called using MRIhexmesh('default'), uses the default values without
%   prompting for inputs.
%   INPUTS:
%    default(optional): Set to 'default' to disable propmts for inputs. Any
%                       other value, or if not supplied will propmt for
%                       inputs. Using default values allows automated
%                       meshing.
%
%   IMPORTANT FEATURES
%
%   - DISPLACEMENT DATA IS SCALED!! I chose to scale everything so that the
%     average displacement amplitude is the same for all datasets
%     interpolated with this code.
%   - Interpolation is an option. Cubic spline interpoaltion is used, a
%     custom function is used which does not use values outside of the mask
%     for the interpolated values.
%   - Different meshing strategies are possible: (MRE-Zonev7.05 now takes
%         zone dimensions rather than dividing the extents into an integer
%         number of zones.)
%      meshstrat=1 uses one FE node for each MR voxel, as has been done in
%                  the past.
%      meshstrat=2 allows specification of the number of FE nodes per
%                  wavelength, based on an estiamte of the shear modulus. A
%                  warning will be produced if the data is being
%                  interpolated to a resolution lower than the data.
%      meshstrat=3 allows specification of a target resolution.
%      Both meshstrat=2 and meshstrat=3 make slight modifications to the
%      resolution to fit an integer number of 27 node elements across the
```

```
%      domain defined by the extents of the mask, to avoid wasting planes
%      of data near the edges.
%   - Different zone sizing strategies are possible:
%      zonestrat=1 Matches a number of FE nodes per zone, as has been done
%                  in the past.
%      zonestrat=2 matches a number of wavelengths per zone, based on an
%                  estiamte of the shear modulus
%      Both of these strategies are only approximate, the way that the
%      zoning proscess works (dividing the mesh extents into an integer
%      number of zones) makes it difficult to exactly match specified
%      values.
%   - Input data which may be useful when analyzing results or looking at
%     a reconstruction later on is saved as outstm.InterpLocations.mat,
%     and outstm.InterpData.mat outstm.meshinput.mat. The idea of saving
%     this data is to attach it to reconstruction results so it is clear
%     what parameters were used to produce it.
%
```

Prallel ToolBox and default mode

```
par=false;          % If you have the matlab parallel processing toolbox set this
nlabs=7;            % Number of labs for matlab to use (max 7)
%[usedef]=ParDefault(nlabs,par,nargin,default);
disp('MRE-Zone v7.3 Data Converter')
if(par)
    disp(['Using Parallelized version with ' int2str(nlabs) ' labs'])
else
    disp('Using non-Parallelized version - modify value of ''par'' to change')
end

if(nargin==0)       % Default value is to prompt for inputs.
    default='no';
end
usedef=strcmp(default,'default'); % If no => false;else=>true
if(usedef)
    disp('Using Default values, no input prompts')
end
[default]=SetDefault;
```

```
MRE-Zone v7.3 Data Converter
Using non-Parallelized version - modify value of 'par' to change
```

Get Inputs

```
[A,vox,mridim,MagIm,P,freqHz,msk,mask]=DataExtraction(usedef,default);
```

```
Error using input
Cannot call INPUT from EVALC.

Error in DataExtraction (line 3)
inputtype=input(['MR Data Source <D = Dartmouth, I = Illinois> (default is

Error in MRIhexmesh_v7p3_multiinput (line 72)
```

```
[A,vox,mridim,MagIm,P,freqHz,msk,mask]=DataExtraction(usedef,default);
```

Build Mesh

```
%Get Soft Prior Segmentations if Needed
[regs,noreg,dim]=PriorSegment(usedef,MagIm);

% Meshing Approach: one node per voxel, or interpolate to give nodes per wavelengt
% Zone Sizing Approach: nodes per zone or wavelengths per zone
[strat]=Strategies(usedef,default);

% Meshing Properties
[meshprop]=MeshingProp(usedef,default,freqHz,strat);

% output file stem
[outstm]=Output_File_Sem(usedef,strat,meshprop);

% Displacement Scaling - Many MRE regularization techniques are sensitive
% to the size of the displacements. Either the regularization weights can
% be altered for each case, or the displacements can be scaled so that they
% are always almost the same size.
dispscale = true; % Switch to turn on displacement scaling
dispscalar = 1e-3; % Average displacement amplitude is scaled to this size.
t0=tic;
```

Scale Data if appropriate

```
meanA = mean(A(repmat(mask,[1 1 1 3])==1));
disp(['Mean Displacement Amplitude all directions ' sprintf('%10.3e',meanA)])
if(dispscale)
    disp(['Scaling Displacements to an average size of ' sprintf('%10.3e',dispscal
    A=A./meanA.*dispscalar;
end

t1=tic;
% Generate Real and Imag Displacements and apply mask
[deplacement,interpo,coord,meshprop]...
    =Gen_Displacements(par,A,P,mask,MagIm,strat,vox,regs,dim,meshprop,noreg);

disp('Displacement Processing Complete, Beginning FE Meshing Process')
tdisp=toc(t1);

% Meshing process:
[temps,node,deplacement]=MeshingProcess(interpo,deplacement,coord);
```

Calcualte Zone sizes

```
touti=tic;

[zoneprop]=ZoneSize(node,strat,meshprop);

disp(['Meshing Complete, ' int2str(node.nn) ' Nodes and ' int2str(node.nel) ' elem
```

Output mesh

```
[namefile]=OutputMesh(outstm,mask,node,deplacement,mridim,coord,regs);

SaveInputs(outstm,strat,interpo,node,coord,dispscale,dispscalar,default,meshprop,d
```

Output Runfiles and submit files

```
% Make sure mu and rho estimates are there
if(strat.mesh~=2)&&(strat.zone~=2)
    meshprop.muest=3300;         % Default shear modulus
    default.rhoest=1000;        % Default density
end

% Create Directories if they dont exist
direc=pwd;
inpath=[direc '\hex\' outstm '\'];
outpath=['inv\'];

% Make hex directory
if (exist('hex','dir')~=7)
    disp('Creating hex directory')
    mkdir('hex')
end

% Make subdirectiories
if (exist(inpath,'dir')~=7)
    disp('Creating hex directory')
    mkdir(inpath)
end
if (exist(outpath,'dir')~=7)
    disp('Creating hex directory')
    mkdir(outpath)
end


% Initialize list of files to move
mvfiles(1).name=[outstm '*']; % Mesh files
```

MREv7.3 runfile

Iso incompressible run file - viscoelastic - Soft Prior On

```
if(~noreg) % do not output soft prior runfiles without a supplied segmentation.
    [mvfiles]=RunfileSPon(outstm,default,meshprop,mvfiles,namefile,freqHz,outpath,
end
```

MREv7.3 runfile

Iso incompressible run file - viscoelastic - No soft Prior

```
[mvfiles]=RunfileSPoff(outstm,meshprop,mvfiles,namefile,freqHz,outpath,default,vox
```

Move all files to where they should be.

```
for ii=1:length(mvfiles)
    [success,message,messageid]=movefile(mvfiles(ii).name,inpath);
    if success==0
        disp(' ');
        disp(['File Transfer Unsuccessful, ii=',num2str(ii)]);
        disp(mvfiles(ii).name);
        disp('MESSAGE: ');
        disp(message);
        disp('MESSAGE ID: ');
        disp(messageid);
    end
end
tout=toc(touti);
ttotal=toc(t0);
```

Display time for each part:

```
disp(['Time for Displacment processing: ' sprintf('%6.2f',tdisp) ' seconds'])
disp(['Time for FE meshing Process: ' sprintf('%6.2f',temps.tfemesh) ' seconds'])
%disp(['  Time to build incidence list: ' sprintf('%6.2f',tin) ' seconds'])
%disp(['  Time to renumber nodes: ' sprintf('%6.2f',tnodrenum) ' seconds'])
%disp(['  Time to find boundary nodes: ' sprintf('%6.2f',tbnod) ' seconds'])
disp(['Time to output files: ' sprintf('%6.2f',tout) ' seconds'])
disp(' ')
disp(['Total processing time: ' sprintf('%6.2f',ttotal) ' seconds'])

end
```

Thresholded Median filter

function [stackout]=selectivemedianfilter(stackin,mask,thresh)

s=size(stackin); stackout=stackin; fsz=1;

for ii=1:s(1) %disp(['ii = ' int2str(ii)]) for jj=1:s(2) for kk=1:s(3) if(mask(ii,jj,kk)==1) nmask=mask(max(ii-fsz,1):min(ii+fsz,s(1)),max(jj-fsz,1):min(jj+fsz,s(2)),max(kk-fsz,1):min(kk+fsz,s(3)))==1; nstack=stackin(max(ii-fsz,1):min(ii+fsz,s(1)),max(jj-fsz,1):min(jj+fsz,s(2)),max(kk-fsz,1):min(kk+fsz,s(3))); medv=median(nstack(nmask)); if(abs((medv-stackin(ii,jj,kk))/medv)>thresh) stackout(ii,jj,kk)=medv; end end end end end

end

Published with MATLAB® R2013a