Contents

How to use computational neuromodeling pipelines?

Introduction

Computational neuromodeling provides insights towards understanding the neural basis of non-invasive electrical stimulation techniques such as transcranial direct current stimulation (tDCS). The major steps in computational modeling include segmentation of structural MRI into different tissue classes, placing virtual electrodes on the models, tessellating this volumetric anatomy into a 3D mesh, and numerically solving for the voltage distribution on this finite element model. Various open source and commercial tools are available to perform each of these steps. SimNIBS (Simulation of Non-Invasive Brain Stimulation) is one such free and open source software package which allows realistic calculations of electric field induced by transcranial magnetic stimulation and transcranial electric stimulation (Saturnino et al., 2018). Simnibs provides three approaches for brain segmentation, mri2mesh, Headreco, Headreco with CAT12 (computational anatomy toolbox, http://www.neuro.uni-jena.de/cat/). 'mri2mesh' performs segmentation using Freesurfer (Dale et al., 1999)and FSL (Smith, 2002; Smith et al., 2004). 'Headreco' uses SPM (statistical parametric mapping, (Friston, 2007)) to perform segmentation, while 'Headreco with CAT12' uses CAT12 to perform segmentation. The user can do electrode placement by using interactive interface of the tool, which allows automatically placing the electrodes as per standard EEG system (Jurcak et al., 2007). SimNIBS uses open source mesher 'GMSH' (Geuzaine and Remacle, 2002) and solver 'GetDP' (Dular, 1998) to model the electric current . Roast is an open source automated pipeline which uses SPM for MRI segmentation and automatically places the electrodes (given the electrode configuration) by registering native MRI to MNI space (Huang et al., 2019). Roast uses iso2mesh (Qianqian Fang and Boas, 2009) to construct the FE model and getDP solver to solve the model. The commercial tools are also available to generate and solve finite element models such as ScanIP (Simpleware Ltd, Exeter, UK), Abaqus (SIMULIA, Providence, RI) etc.

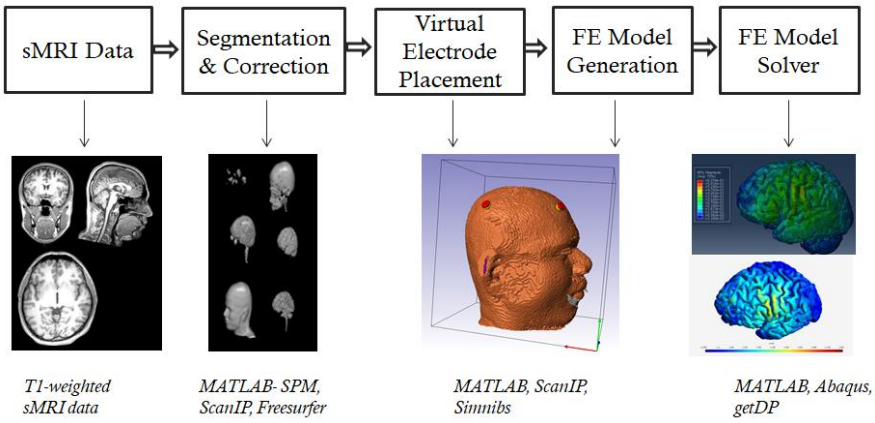

Figure 1. Computational neuromodeling workflow

This article provides guidelines towards using five different neuromodeling pipelines. These pipelines are- ROAST, Simnibs- mri2mesh, headreco with CAT12, headreco with SPM12, ScanIP-Abaqus. Along with the details on processing the data through pipelines, we also provide customized MATLAB functions for batch processing at each step. Please note that, the figures and commands given in this article are designed for conventional tDCS protocol of tDCS used to alleviate the auditory verbal hallucinations symptoms in Schizophrenia, hence we use AF3-CP5 pad electrode montage having 7cm x 5 cm size to target temporoparietal junction and prefrontal cortex with 2 mA current (Bose et al., 2018). The electrodes are positioned so as to match actual clinical settings.

Realistic volumetric-approach to simulate transcranial electric stimulation- ROAST-pipeline

Running ROAST in MATLAB

ROAST is an open source end-to-end neuromodeling pipeline which provides series of steps, brain segmentation, creation of head model, virtual placement of electrodes, solving the finite element model. Roast is an uses SPM for brain MRI segmentation and automatically places the electrodes (given the electrode configuration) by registering native MRI to MNI space (Huang et al., 2019). Roast uses iso2mesh (Qianqian Fang and Boas, 2009) to construct the FE model and getDP (Dular, 1998) solver to solve the model. Command line usage of roast is as follows:

```
roast(subject.nii',{'AF3',2.0,'CP5',-2.0},'elecType','pad','elecSize',[70 50 3],'elecOri',{'si','ap'})
```

ARGUMENTS TO FUNCTION:
roast: roast command to execute entire pipeline

AF3: anode electrode with 2.0 mA current

CP5: cathode electrode name with -2.0 mA current

elecType: type of the electrode, here pad electrode

elecSize: size of the electrode 70x50x30 mm

elecOr: electrode orientation, si- long axis going up (superior) and down (inferior), ap- long axis pointing front (anterior) and back (posterior)

We developed a batch code for running all the subjects at one go. The batch function is given as follows:

```
function batch_roast(roastdir,indir)
cd(indir)
a=dir('*');
cd(roastdir);
for i=3:length(a)
    subj_folder=fullfile(indir,a(i).name);
    subj_name=dir(fullfile(subj_folder,[a(i).name,'.nii']));
    subj=fullfile(subj_folder,subj_name.name);
    roast_forAbaqus(subj,{'AF3',2.0,'CP5',-2.0},'elecType','pad','elecSize',[70 50 3],'elecOri',{'si','ap'});
```

```
end
end
```

ARGUMENTS TO FUNCTION:
roastdir: directory path to roast package
Indir: directory path to all subjects nifti images stored one in each folder

ROAST outputs

Following figure shows ROAST workflow (Figure 1). ROAST provides all the outputs in MATLAB figures. One can also visualize outputs any time after simulation using reviewRes command in MATLAB command line. Roast also outputs voxel wise distribution of electric field and voltage in nifti files.

```
reviewRes('subjectname.nii', 'subjectTag','tissuename')
```

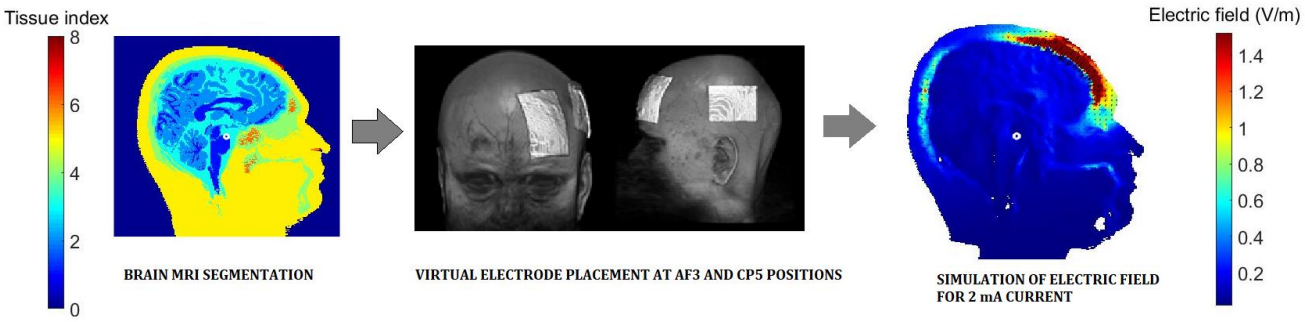

Figure 1. ROAST workflow.

We also performs post-processing of simulation results and exports electric fields into nifti output files ('*_emag.nii').

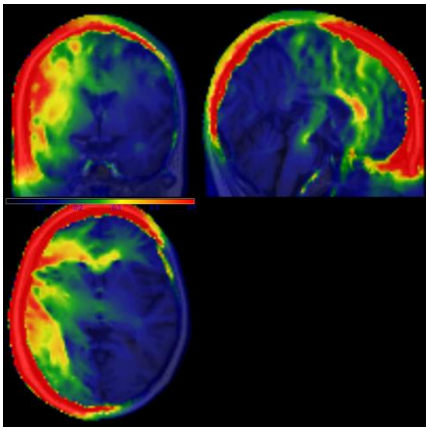

Figure 2. Voxel wise distribution of electric field in a representative subject

Simnibs pipeline

SimNIBS (Simulation of Non-Invasive Brain Stimulation) (version 2.1.2) is free and open source software package which allows realistic calculations of electric field induced by transcranial magnetic stimulation and transcranial electric stimulation (Saturnino et al., 2018). Simnibs provides three approaches for brain segmentation, mri2mesh, Headreco, Headreco with CAT12 (computational anatomy toolbox, http://www.neuro.uni-jena.de/cat/). 'mri2mesh' performs segmentation using Freesurfer (Dale et al., 1999)and FSL (Smith, 2002; Smith et al., 2004). 'Headreco' uses SPM (statistical parametric mapping, (Friston, 2007)) to perform segmentation, while 'Headreco with CAT12' uses CAT12 to perform segmentation. The user can do electrode placement by using interactive interface of the tool, which allows automatically placing the electrodes as per standard EEG system (Jurcak et al., 2007). SimNIBS uses open source mesher 'GMSH' (Geuzaine and Remacle, 2002) and solver 'GetDP' (Dular, 1998) to model the electric current .

Running simnibs in bash

mri2mesh

Simnibs Freesurfer-FSL segmentation (SNF) can be performed using mri2mesh command in bash. mri2mesh reconstructs a tetrahedral head mesh from T1- and T2-weighted structural MR images. It runs also with only a T1w image, but will create better skull segmentations when also a T2w image is available. In our analysis, only T1w is available, so we use only T1w for generating segmentations.

```
mri2mesh –all subjectname subjectname.nii
```

We can also run this function in batch to process all the subjects. Store each subjects T1w image in separate folder. Give similar name to folder and nifti file for simplicity in bash apply following command:

```
For dir in parentdirpath;do(cd "$dir" && mri2mesh –all ${dir##*/} ${dir##*/}.nii);done
```

Headreco

Simnibs uses CAT12 toolbox (SNC) for segmentation to reconstruct a tetrahedral head mesh from T1- and T2-weighted structural MR images. It runs also with only a T1w image, but it will achieve more reliable skull segmentations when a T2w image is supplied. In our analysis, only T1w is available, so we use only T1w for generating segmentations.

```
headreco all --cat subjectname subjectname.nii
```

We can also run this function in batch to process all the subjects. Store each subjects T1w image in separate folder. Give similar name to folder and nifti file for simplicity in bash apply following command

```
For dir in parentdirpath;do(cd "$dir" && headreco all --cat ${dir##*/} ${dir##*/}.nii);done
```

One can also run headreco without using CAT12, in such case simnibs uses SPM12 segmentation (SPM). The command line usage is as follows:

```
headreco all subjectname subjectname.nii
```

We can also run this function in batch to process all the subjects. Store each subjects T1w image in separate folder. Give similar name to folder and nifti file for simplicity in bash apply following command:

```
For dir in parentdirpath;do(cd "$dir" && headreco all ${dir##*/} ${dir##*/}.nii);done
```

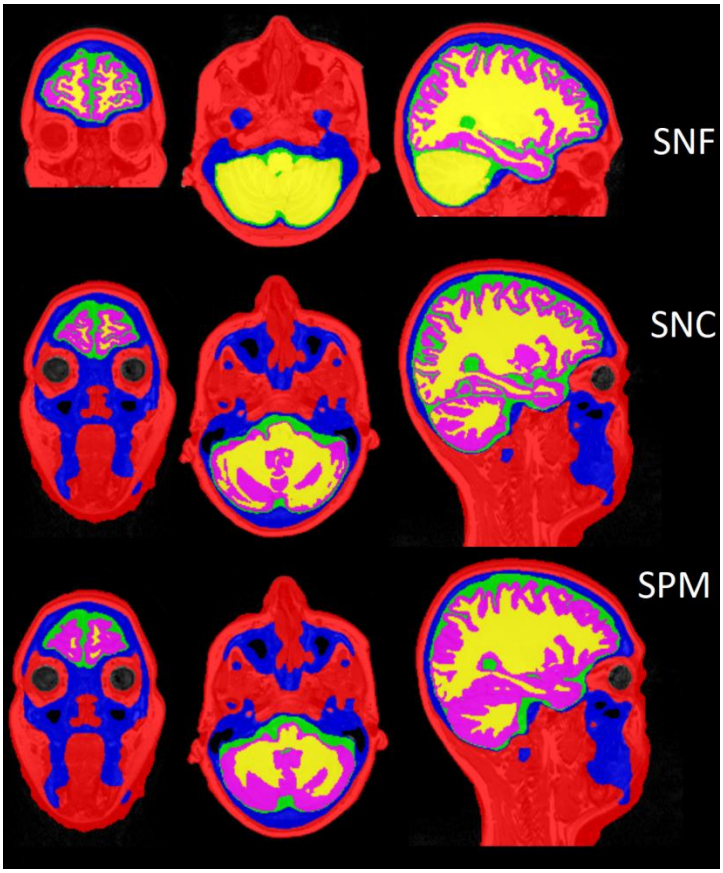

Figure 2. Simnibs segmentation pipelines, SNF: simnibs freesurfer-FSL segmentation, SNC: simnibs CAT segmentation, SPM: simnibs SPM segmentation. pink: GM, yellow: WM, green: CSF, blue: skull, red: skin

Electrode placement

Simnibs offers interactive graphical user interface for placement of electrodes on head meshes. After the head segmentation and head model creation, one has to import the subject's model in simnibs GUI. User only has to browse and enter the folder name with m2m_*. This folder contains all the segmentations, EEG positions of electrodes, prepared masks, and MNI conversions. Simnibs automatically detects head mesh (*.msh format) once we enter m2m_* folder. See the simnibs GUI snapshot below.

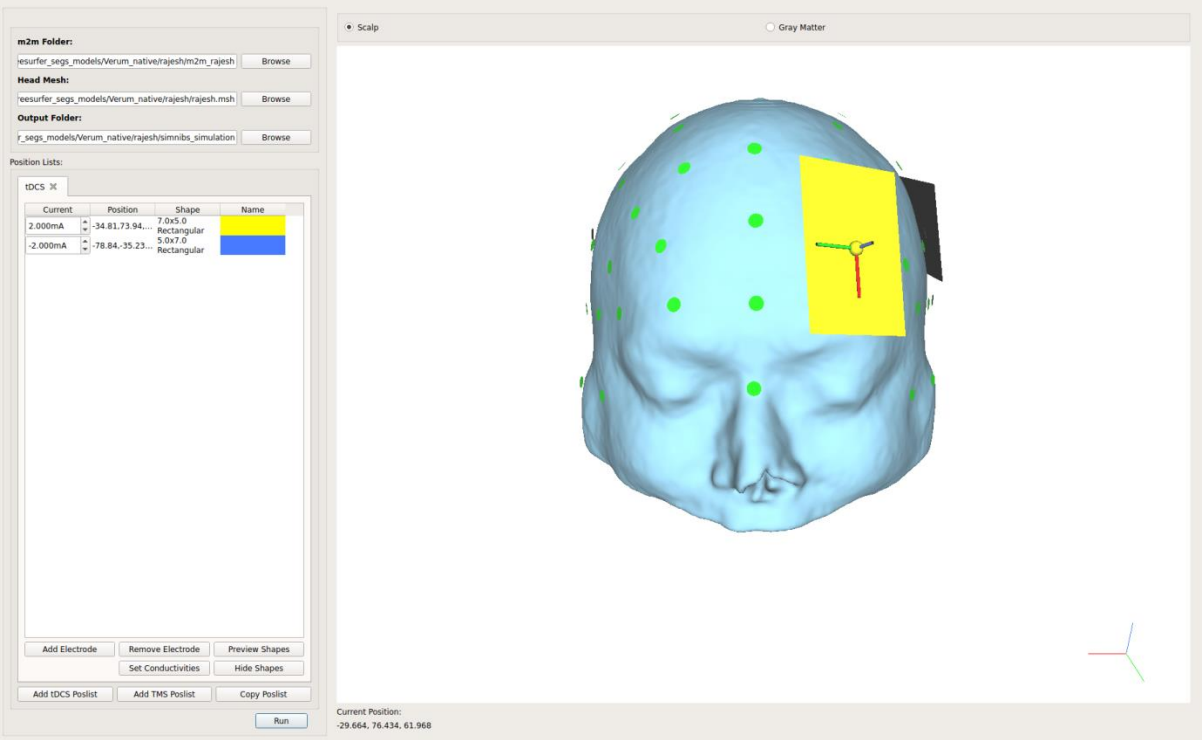

Figure 3. Electrode placement in Simnibs GUI

For tDCS electrode placement, we select ‘Add tDCS Poslist’ and add anode and cathode electrodes. The position and orientation of electrodes can be specified by the user using mouse buttons. The shapes of the electrodes can be specified by clicking ‘Shape’ at each electrode. Once the electrodes are places, they can be previewed by choosing ‘Preview Shapes’ button. Figure 3 shows one representative head model (model created after SNF segmentation) with AF3 anode and CP5 as cathode. We have provided 2 mA current and electrodes are sizes and positions are done as per actual clinical settings.

Simnibs outputs

Once all the settings are made, the simulation can be set to run by clicking ‘Run’ button. This simulation results will be by default stored in ‘simnibs_simulation’ folder. This folder will contain ‘.mat’ file, which will have all the simulation settings that user has made, this can be viewed in MATLAB. The simulated electric field will be stored in this folder with ‘.msh’ extension. Also, it will contain simulations log file to view the processing, and/or see if there are any errors. The simulated .msh file can be visualized over gray matter surface in GMSH. Figure 4 shows the simulated electric field for SNF, SNC and SPM for a representative subject.

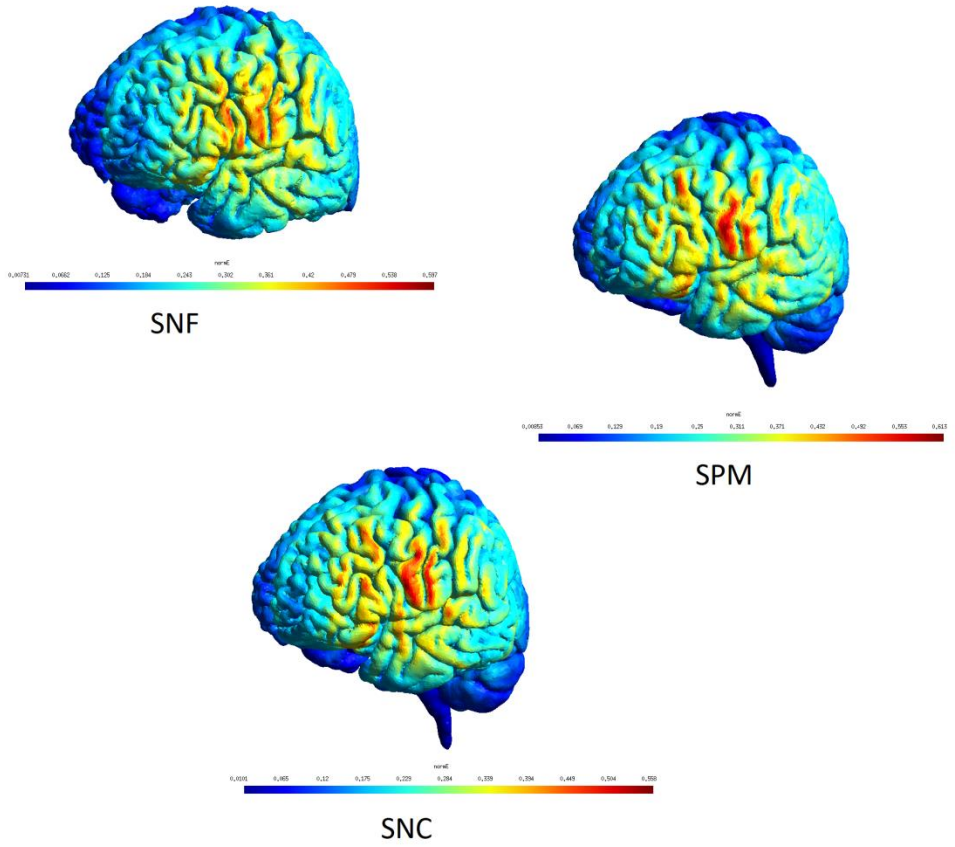

Figure 4. Simnibs simulations viewed in GMSH viewer. The electric field is overlayed on gray matter surface.

It is necessary to export the simulation output .msh into nifti format, in order to visualize as 3D image and use it for other post-processing purposes. Simnibs command-line tool provides a function named 'msh2nii' to perform this task. This function will take input .msh file, T1fs_conform file and output a nifti file which will be a voxel wise map of distribution of electric field (Figure 5). The usage of msh2nii is as follows:

```
msh2nii subjectname_TDCS_1_scalar.msh  ../m2m_subjectname/T1fs_conform.nii.gz subjectname_TDCS_1
```

This function can also output the masks that were used for creating the head model. The usage is as follows:

```
msh2nii --create_masks subjectname_TDCS_1_scalar.msh ../m2m_subjectname/T1fs_conform.nii.gz
subjectname_TDCS_1
```

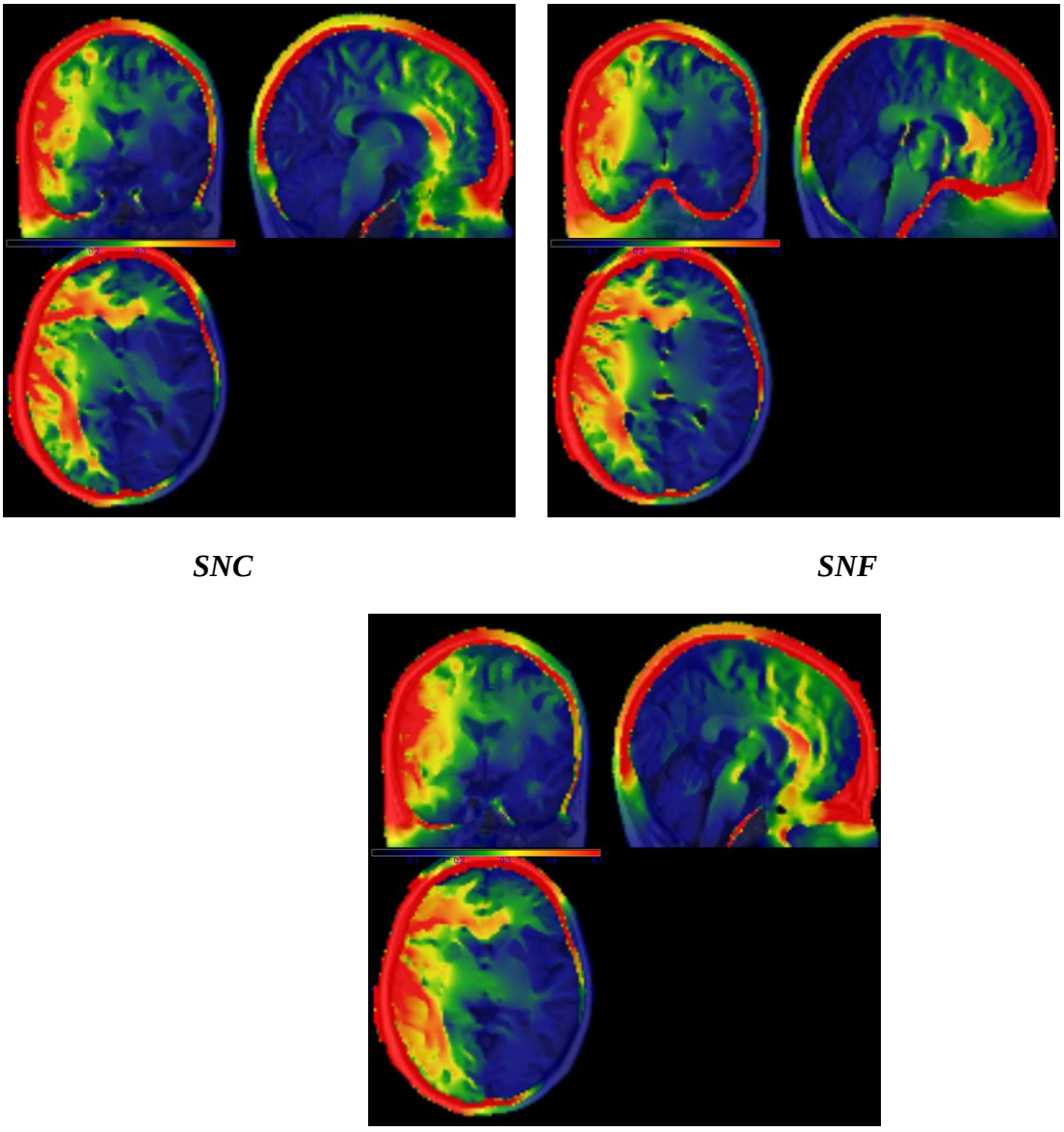

Figure 5. Voxel wise electric field distribution in SNC, SNF and SPM pipelines for same representative subject's data

ScanIP-Abaqus pipeline

Running ScanIP-Abaqus pipeline

The commercial tools are also available to generate and solve finite element models such as ScanIP (Simpleware Ltd, Exeter, UK), Abaqus (SIMULIA, Providence, RI) etc. We have developed a pipeline that uses ScanIP as a mesher and Abaqus as finite element solver. One can export the created model from ScanIP to Abaqus. Once we export the model, the model name

will be identified with extension '.inp' extension. The '.inp' file exported from ScanIP can now be prepared for simulation in Abaqus. We have developed custom MATLAB functions for this task. First of all, we use segmentation procedures of ROAST to create masks but abaqus preparation requires some variable details to process. For that purpose, we have modified ROAST function and created our own function for segmentation (note that this function is just modified in a way so that we can save required variables from ROAST output, by default TOAST do not save them).

Use following function for segmentation: "roast_forAbaqus.m" (provided in supplementary section)

This function can also be run in batch, where each subject's structural nifti file is stored into a separate folder. We have developed a batch function named, 'batch_roast_forAbaqus.m'.

FUNCTION EXECUTION: IN THE MATLAB SCRIPT WINDOW

```
function batch_roast_forAbaqus(roastdir,indir)
cd(indir)
a=dir('*');
cd(roastdir);
for i=3:length(a)
   subj_folder=fullfile(indir,a(i).name);
   subj_name=dir(fullfile(subj_folder,[a(i).name,'.nii']));
   subj=fullfile(subj_folder,subj_name.name);
   roast_forAbaqus(subj,{'AF3',2.0,'CP5',-2.0},'elecType','pad','elecSize',[70 50 3],'elecOri',{'si','ap'});
end
end
```

ARGUMENTS TO FUNCTION:
roastdir: directory path to roast package
Indir: directory path to all subjects' nifti images stored one in each folder

Also note that, roast_forAbaqus uses 'generateElecMask_edited.m' instead of generateElecMask.m in actual roast.m script. The edited script will store electrode and gel specific variables which will be needed in later part of the scripts. This function 'generateElecMask_edited.m' is also provided in the supplementary sections below. We save these two functions roast_forAbaqus.m and generateelecMask_edited.m in the roast directory.
The segmentation outputs will be one single nifti file containing all masks. To import these masks into ScanIP, we need to separate each mask into single nifti file. We developed a custom MATLAB function to do this task. The function below will pop-up GUI interface to select all your segmentation images and it will separate each tissue into separate nifti file.
Now we import segmentation masks generated from ROAST pipeline into ScanIP to construct the head model. The snapshot of ScanIP model creation is shown in Figure 6 below. The segmentation masks are thresholded and imported one by one using ScanIP interface. These masks are then added to 'active model' and finite element model is created. We choose 'binaries before smoothing' option in setup model tab. The mesh creation algorithm was chosen as '+FE

Free', element type was chosen as 'All tetrahedral (linear)' and coarseness level was chosen as '-25' (range: -50 coarse to +50 fine). The full model for one of the representative subjects is shown in Figure 7 below.

Separate each tissue mask from segmentation output file

```
P=spm_get(Inf,'*.nii','Select images');
P1=cellstr(P);
for i=1:length(P1)
   [dirname,n,~]=fileparts(P1{i});
   vol=load_untouch_nii(P1{i});
   data=vol.img;
   data_backup=data;
   for v=1:6
      data(data~=v)=0;
      data(data==v)=1;
      vol.img=data;
      %save_untouch_nii(vol,sprintf('class_%d_%s',v,n));
      %save_untouch_nii(vol,fullfile(dirname,sprintf('class_%d_%s',v,'tissue')));
      save_untouch_nii(vol,sprintf('%s_%d_%s',n,v,'tissue'));
      data=data_backup;
   end
end
```

Now the model has been created, exported to abaqus and all the necessary variables required for preparing a file to simulate in abaqus are ready. We have created a function named 'prepapreForAbaqus.m', this function creates rnge_elec and rnge_gel files for post ScanIP processing. You can apply this script in batch using batch_prepare_forabaqus.m, where each subject's data is stored in separate folder. Note that this script must be applied after running batch_roast_forAbaqus.m or roast_forAbaqus.m.

```
% Use this batch script to create rnge_elec and rnge_gel mat files for
% multiple subjects at the same time
function batch_prepare_forAbaqus(indir)
cd(indir)
a=dir('*');
%cd(roastdir);
for i=3:length(a)
   Image=fullfile(a(i).folder,a(i).name,[a(i).name,'_T1orT2_masks.nii']);
   elecgellabelsMat=fullfile(a(i).folder,a(i).name,[a(i).name, '_T1orT2_ElecGelLabels.mat']);
   optMat_name=dir(fullfile(a(i).folder,a(i).name,'*_options.mat'));
   optMat=fullfile(a(i).folder,a(i).name,optMat_name(1).name);
   prepareForAbaqus(Image,elecgellabelsMat,optMat);
end
end
```

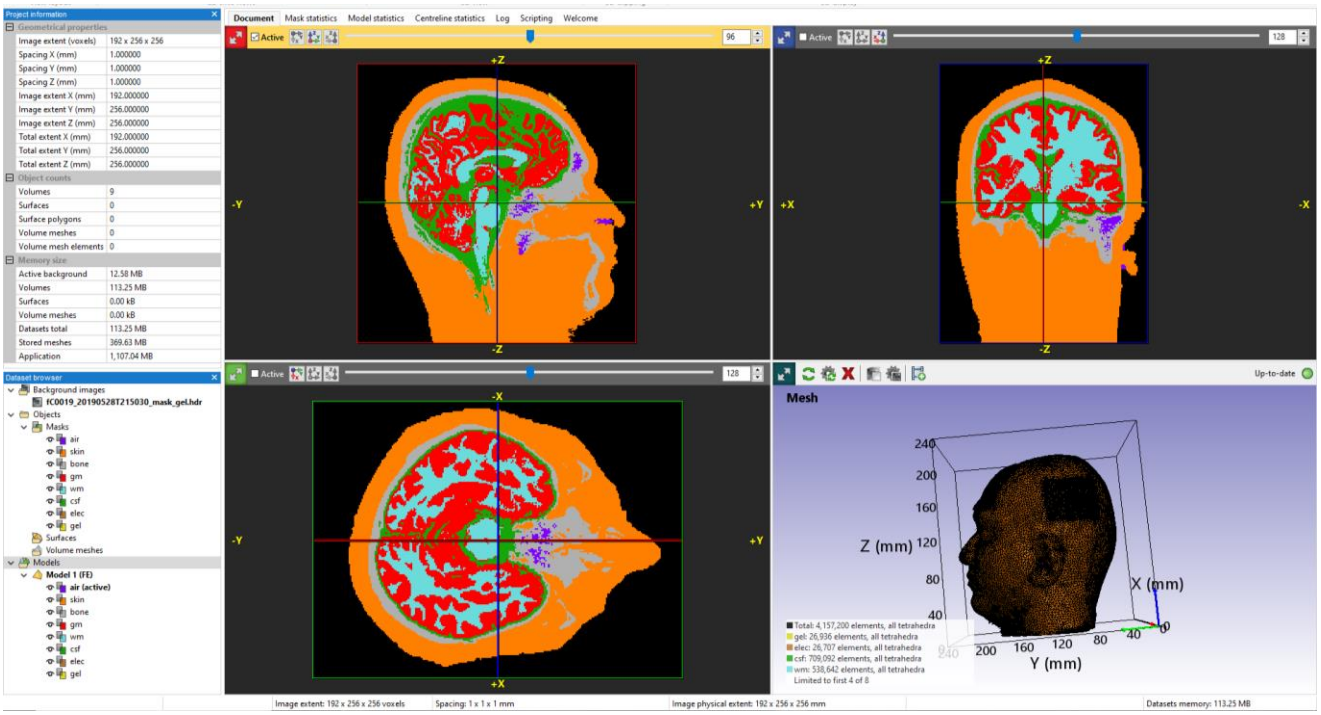

Figure 6. Importing segmentation masks and creation of head model in ScanIP

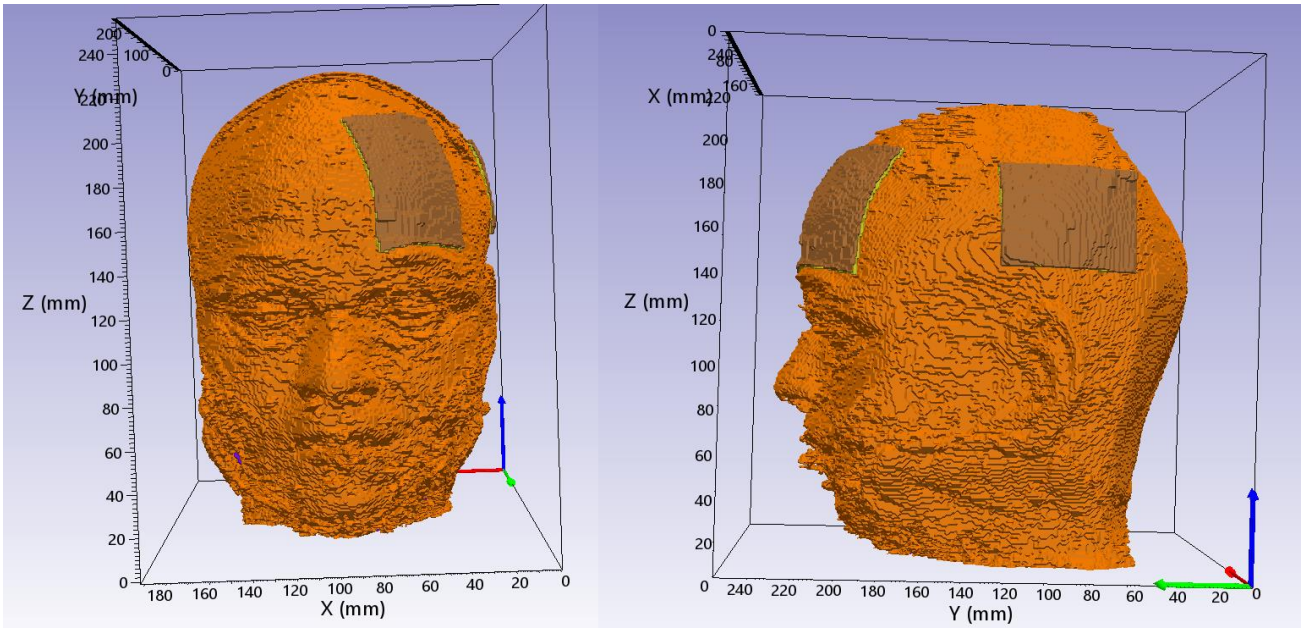

Figure 7. Head models generated from ScanIP, electrodes are placed at location AF3 (anode) and CP5 (cathode)

After preparing electrode and gel specific files, we use following custom script to assign electrode configuration to model. The batch function prepare_forAbaqus2.m will use 'tpsych_bipolar_tDCS.m' function, which is a modified version of a MATLAB script provided earlier study (Huang et al., 2013).

```
function prepare_forAbaqus2(indir,currden)
cd(indir);
list_subj=dir('*');
for subj=3:length(list_subj)
  dirname=fullfile(list_subj(subj).folder,list_subj(subj).name);
  fname=[list_subj(subj).name,'_scanip'];
  tpysch_bipolar_tDCS(dirname,fname,currden);
end
end
```

ScanIP-Abaqus pipeline outputs

Now our head model is ready to be processed into abaqus for current simulation. The final file that will be created after all the processing steps mentioned above, will be '*_abaqus.inp'. We will process this file into abaqus. Create a text file in subject's folder named 'run.txt'. Put modify following example lines and add to this run.txt file. The solved model can be viewed in abaqus along with electric field distribution (Figure 8). We provided number of CPUs as 4 here, job is name of the prepared abaqus file (ending with *_abaqus.inp). Save this text file as '.bat file' and run in windows command prompt.

```
call abaqus job=subjectname_scanip_abaqus.inp int cpus=4
```

The job will run and subsequent files will be created in your working directory. The file that we will need now is the one with extension '.odb', this file will carry same name as that of input .inp file. Open abaqus GUI and import this file into abaqus, export the output coordinates and corresponding electric field values into '.rpt' file. The solutions on the mesh node were read into MATLAB and interpolated onto a regular grid with the same dimensions and resolution as the original MRI. To achieve this we developed two MATLAB functions, 'interpolate_abaqus_field_part1.m' and 'interpolate_abaqus_field_part2.m'. We will be using .rpt file for further processing and creating voxel wise electric field distribution in 3D image. For that purpose we have developed two custom batch functions in MATLAB,

'interpolate_abqus_field_part2.m'
'interpolate_abaqus_field_part2.m'

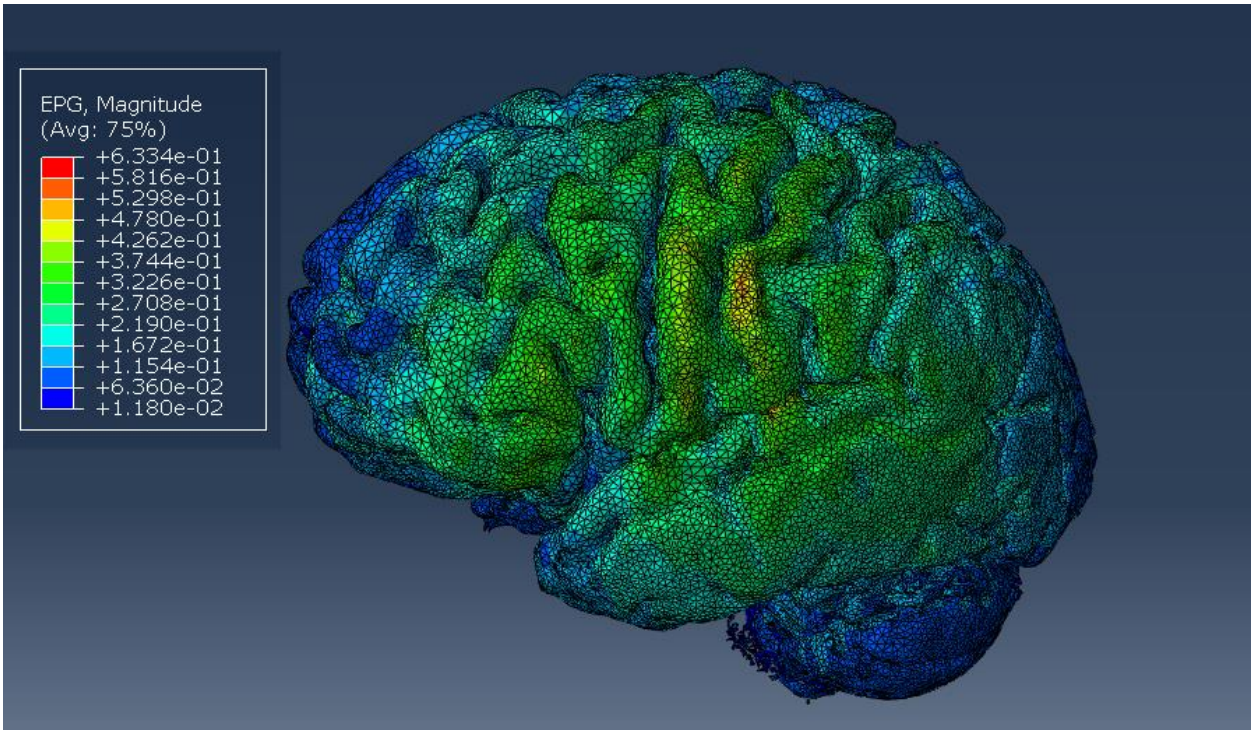

Figure 8. Simulated head model in abaqus (AF3-CP5 electrode montage, 2 mA current)

References

- Bose, A., Shivakumar, V., Agarwal, S.M., Kalmady, S.V., Shenoy, S., Sreeraj, V.S., Narayanaswamy, J.C., Venkatasubramanian, G., 2018. Efficacy of fronto-temporal transcranial direct current stimulation for refractory auditory verbal hallucinations in schizophrenia: A randomized, double-blind, sham-controlled study. Schizophr. Res. 195, 475–480. https://doi.org/10.1016/j.schres.2017.08.047
- Dale, A.M., Fischl, B., Sereno, M.I., 1999. Cortical Surface-Based Analysis. NeuroImage 9, 179–194. https://doi.org/10.1006/nimg.1998.0395
- Dular, P., 1998. Local and Global Constraints in Finite Element Modeling and the Benefits of Nodal and Edge Elements Coupling 4.
- Friston, 2007. Statistical Parametric Mapping: The Analysis of Functional Brain Images - 1st Edition [WWW Document]. URL https://www.elsevier.com/books/statistical-parametric-mapping-the-analysis-of-functional-brain-images/penny/978-0-12-372560-8 (accessed 9.24.19).
- Geuzaine, C., Remacle, J.-F., 2002. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. Int. J. Numer. METHODS Eng. 1–24.
- Huang, Y., Datta, A., Bikson, M., Parra, L.C., 2019. Realistic volumetric-approach to simulate transcranial electric stimulation—ROAST— a fully automated open-source pipeline. J Neural Eng 16.
- Huang, Y., Dmochowski, J.P., Su, Y., Datta, A., Rorden, C., Parra, L.C., 2013. Automated MRI segmentation for individualized modeling of current flow in the human head. J. Neural Eng. 10, 066004. https://doi.org/10.1088/1741-2560/10/6/066004
- Jurcak, V., Tsuzuki, D., Dan, I., 2007. 10/20, 10/10, and 10/5 systems revisited: Their validity as relative head-surface-based positioning systems. NeuroImage 34, 1600–1611. https://doi.org/10.1016/j.neuroimage.2006.09.024
- Qianqian Fang, Boas, D.A., 2009. Tetrahedral mesh generation from volumetric binary and grayscale images, in: 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro. Presented at the 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI), IEEE, Boston, MA, USA, pp. 1142–1145. https://doi.org/10.1109/ISBI.2009.5193259
- Saturnino, G.B., Puonti, O., Nielsen, J.D., Antonenko, D., Madsen, K.H., Thielscher, A., 2018. SimNIBS 2.1: A Comprehensive Pipeline for Individualized Electric Field Modelling for Transcranial Brain Stimulation. bioRxiv 500314. https://doi.org/10.1101/500314
- Smith, S.M., 2002. Fast robust automated brain extraction. Hum. Brain Mapp. 17, 143–155. https://doi.org/10.1002/hbm.10062
- Smith, S.M., Jenkinson, M., Woolrich, M.W., Beckmann, C.F., Behrens, T.E.J., Johansen-Berg, H., Bannister, P.R., De Luca, M., Drobnjak, I., Flitney, D.E., Niazy, R.K., Saunders, J., Vickers, J., Zhang, Y., De Stefano, N., Brady, J.M., Matthews, P.M., 2004. Advances in functional and structural MR image analysis and implementation as FSL. NeuroImage 23, S208–S219. https://doi.org/10.1016/j.neuroimage.2004.07.051

Supplementary Section

roast_forAbaqus.m

```
function roast_forAbaqus(subj,recipe,varargin)

addpath(genpath([fileparts(which(mfilename)) filesep 'lib/']));

fprintf('\n\n');
disp('=========================================================')
disp('CHECKING INPUTS...')
disp('=========================================================')
fprintf('\n');

% warning('on');

% check subject name
if nargin<1 || isempty(subj)
   subj = 'example/MNI152_T1_1mm.nii';
end

if strcmpi(subj,'nyhead')
   subj = 'example/nyhead.nii';
end

if ~strcmpi(subj,'example/nyhead.nii') && ~exist(subj,'file')
   error(['The subject MRI you provided ' subj ' does not exist.']);
end

if ~strcmpi(subj,'example/nyhead.nii')
   t1Data = load_untouch_nii(subj);
   if t1Data.hdr.hist.qoffset_x == 0 && t1Data.hdr.hist.srow_x(4)==0
      error('The MRI has a bad header. SPM cannot generate the segmentation properly for MRI with bad header.
You can manually align the MRI in SPM Display function to fix the header.');
   end
   % check if bad MRI header
end

% check recipe syntax
if nargin<2 || isempty(recipe)
   recipe = {'Fp1',1,'P4',-1};
end

if mod(length(recipe),2)~=0
   error('Unrecognized format of your recipe. Please enter as electrodeName-injectedCurrent pair.');
end

elecName = (recipe(1:2:end-1))';

% take in user-specified options
if mod(length(varargin),2)~=0
   error('Unrecognized format of options. Please enter as property-value pair.');
```

```
end

indArg = 1;
while indArg <= length(varargin)
   switch lower(varargin{indArg})
      case 'captype'
         capType = varargin{indArg+1};
         indArg = indArg+2;
      case 'electype'
         elecType = varargin{indArg+1};
         indArg = indArg+2;
      case 'elecsize'
         elecSize = varargin{indArg+1};
         indArg = indArg+2;
      case 'elecori'
         elecOri = varargin{indArg+1};
         indArg = indArg+2;
      case 't2'
         T2 = varargin{indArg+1};
         indArg = indArg+2;
      case 'meshoptions'
         meshOpt = varargin{indArg+1};
         indArg = indArg+2;
      case 'simulationtag'
         simTag = varargin{indArg+1};
         indArg = indArg+2;
      case 'resampling'
         doResamp = varargin{indArg+1};
         indArg = indArg+2;
      case 'zeropadding'
         paddingAmt = varargin{indArg+1};
         indArg = indArg+2;
      case 'conductivities'
         conductivities = varargin{indArg+1};
         indArg = indArg+2;
      otherwise
         error('Supported options are: "capType", "elecType", "elecSize", "elecOri", "T2", "meshOptions",
"conductivities", "simulationTag", "resampling", and "zeroPadding".');
   end
end

% set up defaults and check on option conflicts
if ~exist('capType','var')
   capType = '1010';
else
   if ~any(strcmpi(capType,{'1020','1010','1005','biosemi'}))
      error('Supported cap types are: "1020", "1010", "1005" and "BioSemi".');
   end
end

if ~exist('elecType','var')
   elecType = 'disc';
else
   if ~iscellstr(elecType)
      if ~any(strcmpi(elecType,{'disc','pad','ring'}))
```

```
            error('Supported electrodes are: "disc", "pad" and "ring".');
        end
    else
        if length(elecType)~=length(elecName)
            error('You want to place more than 1 type of electrodes, but did not tell ROAST which type for each electrode. Please provide the type for each electrode respectively, as the value for option "elecType", in a cell array of length equals to the number of electrodes to be placed.');
        end
        for i=1:length(elecType)
          if ~any(strcmpi(elecType{i},{'disc','pad','ring'}))
              error('Supported electrodes are: "disc", "pad" and "ring".');
          end
        end
    end
end

if ~exist('elecSize','var')
    if ~iscellstr(elecType)
        switch lower(elecType)
            case {'disc'}
                elecSize = [6 2];
            case {'pad'}
                elecSize = [50 30 3];
            case {'ring'}
                elecSize = [4 6 2];
        end
    else
        elecSize = cell(1,length(elecType));
        for i=1:length(elecSize)
            switch lower(elecType{i})
                case {'disc'}
                    elecSize{i} = [6 2];
                case {'pad'}
                    elecSize{i} = [50 30 3];
                case {'ring'}
                    elecSize{i} = [4 6 2];
            end
        end
    end
else
    if ~iscellstr(elecType)
        if iscell(elecSize)
            warning('Looks like you''re placing only 1 type of electrodes. ROAST will only use the 1st entry of the cell array of "elecSize". If this is not what you want and you meant differect sizes for different electrodes of the same type, just enter "elecSize" option as an N-by-2 or N-by-3 matrix, where N is number of electrodes to be placed.');
            elecSize = elecSize{1};
        end
        if any(elecSize(:)<=0)
            error('Please enter non-negative values for electrode size.');
        end
        if size(elecSize,2)~=2 && size(elecSize,2)~=3
            error('Unrecognized electrode sizes. Please specify as [radius height] for disc, [length width height] for pad, and [innerRadius outterRadius height] for ring electrode.');
        end
        if size(elecSize,1)>1 && size(elecSize,1)~=length(elecName)
```

```
            error('You want different sizes for each electrode. Please tell ROAST the size for each electrode respectively, in a N-row matrix, where N is the number of electrodes to be placed.');
        end
        if strcmpi(elecType,'disc') && size(elecSize,2)==3
            error('Redundant size info for Disc electrodes. Please enter as [radius height]');
%           elecSize = elecSize(:,1:2);
        end
        if any(strcmpi(elecType,{'pad','ring'})) && size(elecSize,2)==2
            error('Insufficient size info for Pad or Ring electrodes. Please specify as [length width height] for pad, and [innerRadius outterRadius height] for ring electrode.');
        end
        if strcmpi(elecType,'pad') && any(elecSize(:,1) < elecSize(:,2))
            error('For Pad electrodes, the width of the pad should not be bigger than its length. Please enter as [length width height]');
        end
        if strcmpi(elecType,'pad') && any(elecSize(:,3) < 3)
            error('For Pad electrodes, the thickness should at least be 3 mm.');
        end
        if strcmpi(elecType,'pad') && any(elecSize(:) > 80)
            warning('You''re placing large pad electrodes (one of its dimensions is bigger than 8 cm). For large pads, the size will not be exact in the model because they will be bent to fit the scalp surface.');
        end
        if strcmpi(elecType,'ring') && any(elecSize(:,1) >= elecSize(:,2))
            error('For Ring electrodes, the inner radius should be smaller than outter radius. Please enter as [innerRadius outterRadius height]');
        end
    else
        if ~iscell(elecSize)
            error('You want to place at least 2 types of electrodes, but only provided size info for 1 type. Please provide complete size info for all types of electrodes in a cell array as the value for option "elecSize", or just use defaults by not specifying "elecSize" option.');
        end
        if length(elecSize)~=length(elecType)
            error('You want to place more than 1 type of electrodes. Please tell ROAST the size for each electrode respectively, as the value for option "elecSize", in a cell array of length equals to the number of electrodes to be placed.');
        end
        for i=1:length(elecSize)
            if isempty(elecSize{i})
                switch lower(elecType{i})
                    case {'disc'}
                        elecSize{i} = [6 2];
                    case {'pad'}
                        elecSize{i} = [50 30 3];
                    case {'ring'}
                        elecSize{i} = [4 6 2];
                end
            else
                if any(elecSize{i}(:)<=0)
                    error('Please enter non-negative values for electrode size.');
                end
                if size(elecSize{i},2)~=2 && size(elecSize{i},2)~=3
                    error('Unrecognized electrode sizes. Please specify as [radius height] for disc, [length width height] for pad, and [innerRadius outterRadius height] for ring electrode.');
                end
                if size(elecSize{i},1)>1
```

```
                error('You''re placing more than 1 type of electrodes. Please put size info for each electrode as a 1-row vector in a cell array for option "elecSize".');
            end
            if strcmpi(elecType{i},'disc') && size(elecSize{i},2)==3
                error('Redundant size info for Disc electrodes. Please enter as [radius height]');
%                 elecSize{i} = elecSize{i}(:,1:2);
            end
            if any(strcmpi(elecType{i},{'pad','ring'})) && size(elecSize{i},2)==2
                error('Insufficient size info for Pad or Ring electrodes. Please specify as [length width height] for pad, and [innerRadius outterRadius height] for ring electrode.');
            end
            if strcmpi(elecType{i},'pad') && any(elecSize{i}(:,1) < elecSize{i}(:,2))
                error('For Pad electrodes, the width of the pad should not be bigger than its length. Please enter as [length width height]');
            end
            if strcmpi(elecType{i},'pad') && any(elecSize{i}(:,3) < 3)
                error('For Pad electrodes, the thickness should at least be 3 mm.');
            end
            if strcmpi(elecType{i},'pad') && any(elecSize{i}(:) > 80)
                warning('You''re placing large pad electrodes (one of its dimensions is bigger than 8 cm). For large pads, the size will not be exact in the model because they will be bent to fit the scalp surface.');
            end
            if strcmpi(elecType{i},'ring') && any(elecSize{i}(:,1) >= elecSize{i}(:,2))
                error('For Ring electrodes, the inner radius should be smaller than outter radius. Please enter as [innerRadius outterRadius height]');
            end
        end
    end
end

if ~exist('elecOri','var')
    if ~iscellstr(elecType)
        if strcmpi(elecType,'pad')
            elecOri = 'lr';
        else
            elecOri = [];
        end
    else
        elecOri = cell(1,length(elecType));
        for i=1:length(elecOri)
            if strcmpi(elecType{i},'pad')
                elecOri{i} = 'lr';
            else
                elecOri{i} = [];
            end
        end
    end
else
    if ~iscellstr(elecType)
        if ~strcmpi(elecType,'pad')
            warning('You''re not placing pad electrodes; customized orientation options will be ignored.');
            elecOri = [];
        else
            if iscell(elecOri)
                allChar = 1;
```

```
            for i=1:length(elecOri)
                if ~ischar(elecOri{i})
                    warning('Looks like you''re only placing pad electrodes. ROAST will only use the 1st entry of the cell array of "elecOri". If this is not what you want and you meant differect orientations for different pad electrodes, just enter "elecOri" option as an N-by-3 matrix, or as a cell array of length N (put "lr", "ap", or "si" into the cell element), where N is number of pad electrodes to be placed.');
                    elecOri = elecOri{1};
                    allChar = 0;
                    break;
                end
            end
            if allChar && length(elecOri)~=length(elecName)
                error('You want different orientations for each pad electrode by using pre-defined keywords in a cell array. Please make sure the cell array has a length equal to the number of pad electrodes.');
            end
        end
        if ~iscell(elecOri)
            if ischar(elecOri)
                if ~any(strcmpi(elecOri,{'lr','ap','si'}))
                    error('Unrecognized pad orientation. Please enter "lr", "ap", or "si" for pad orientation; or just enter the direction vector of the long axis of the pad');
                end
            else
                if size(elecOri,2)~=3
                    error('Unrecognized pad orientation. Please enter "lr", "ap", or "si" for pad orientation; or just enter the direction vector of the long axis of the pad');
                end
                if size(elecOri,1)>1 && size(elecOri,1)~=length(elecName)
                    error('You want different orientations for each pad electrode. Please tell ROAST the orientation for each pad respectively, in a N-by-3 matrix, where N is the number of pads to be placed.');
                end
            end
        end
    end
else
    if ~iscell(elecOri)
        elecOri0 = elecOri;
        elecOri = cell(1,length(elecType));
        if ischar(elecOri0)
            if ~any(strcmpi(elecOri0,{'lr','ap','si'}))
                error('Unrecognized pad orientation. Please enter "lr", "ap", or "si" for pad orientation; or just enter the direction vector of the long axis of the pad');
            end
            for i=1:length(elecType)
                if strcmpi(elecType{i},'pad')
                    elecOri{i} = elecOri0;
                else
                    elecOri{i} = [];
                end
            end
        else
            if size(elecOri0,2)~=3
                error('Unrecognized pad orientation. Please enter "lr", "ap", or "si" for pad orientation; or just enter the direction vector of the long axis of the pad');
            end
            numPad = 0;
```

```
            for i=1:length(elecType)
                if strcmpi(elecType{i},'pad')
                    numPad = numPad+1;
                end
            end
            if size(elecOri0,1)>1
                if size(elecOri0,1)~=numPad
                    error('You want different orientations for each pad electrode. Please tell ROAST the orientation for each pad respectively, in a N-by-3 matrix, where N is the number of pads to be placed.');
                end
            else
                elecOri0 = repmat(elecOri0,numPad,1);
            end
            i0=1;
            for i=1:length(elecType)
                if strcmpi(elecType{i},'pad')
                    elecOri{i} = elecOri0(i0,:);
                    i0 = i0+1;
                else
                    elecOri{i} = [];
                end
            end
        end
    else
        if length(elecOri)~=length(elecType)
            error('You want to place another type of electrodes aside from pad. Please tell ROAST the orienation for each electrode respectively, as the value for option "elecOri", in a cell array of length equals to the number of electrodes to be placed (put [] for non-pad electrodes).');
        end
        for i=1:length(elecOri)
            if strcmpi(elecType{i},'pad')
                if isempty(elecOri{i})
                    elecOri{i} = 'lr';
                else
                    if ischar(elecOri{i})
                        if ~any(strcmpi(elecOri{i},{'lr','ap','si'}))
                            error('Unrecognized pad orientation. Please enter "lr", "ap", or "si" for pad orientation; or just enter the direction vector of the long axis of the pad');
                        end
                    else
                        if size(elecOri{i},2)~=3
                            error('Unrecognized pad orientation. Please enter "lr", "ap", or "si" for pad orientation; or just enter the direction vector of the long axis of the pad');
                        end
                        if size(elecOri{i},1)>1
                            error('You''re placing more than 1 type of electrodes. Please put orientation info for each pad electrode as a 1-by-3 vector or one of the three keywords "lr", "ap", or "si" in a cell array for option "elecOri".');
                        end
                    end
                end
            else
                %                 warning('You''re not placing pad electrodes; customized orientation options will be ignored.');
                elecOri{i} = [];
            end
        end
```

```
        end
    end
end

elecPara = struct('capType',capType,'elecType',elecType,...
        'elecSize',elecSize,'elecOri',elecOri);

if ~exist('T2','var')
    T2 = [];
else
    if ~exist(T2,'file'), error(['The T2 MRI you provided ' T2 ' does not exist.']); end

    t2Data = load_untouch_nii(T2);
    if t2Data.hdr.hist.qoffset_x == 0 && t2Data.hdr.hist.srow_x(4)==0
        error('The MRI has a bad header. SPM cannot generate the segmentation properly for MRI with bad header.
You can manually align the MRI in SPM Display function to fix the header.');
    end
    % check if bad MRI header
end

if ~exist('meshOpt','var')
    meshOpt = struct('radbound',5,'angbound',30,'distbound',0.4,'reratio',3,'maxvol',10);
else
    if ~isstruct(meshOpt), error('Unrecognized format of mesh options. Please enter as a structure, with field names as
"radbound", "angbound", "distbound", "reratio", and "maxvol". Please refer to the iso2mesh documentation for more
details.'); end
    meshOptNam = fieldnames(meshOpt);
    if isempty(meshOptNam) || ~all(ismember(meshOptNam,{'radbound';'angbound';'distbound';'reratio';'maxvol'}))
        error('Unrecognized mesh options detected. Supported mesh options are "radbound", "angbound", "distbound",
"reratio", and "maxvol". Please refer to the iso2mesh documentation for more details.');
    end
    if ~isfield(meshOpt,'radbound')
        meshOpt.radbound = 5;
    else
        if ~isnumeric(meshOpt.radbound) || meshOpt.radbound<=0
            error('Please enter a positive number for the mesh option "radbound".');
        end
    end
    if ~isfield(meshOpt,'angbound')
        meshOpt.angbound = 30;
    else
        if ~isnumeric(meshOpt.angbound) || meshOpt.angbound<=0
            error('Please enter a positive number for the mesh option "angbound".');
        end
    end
    if ~isfield(meshOpt,'distbound')
        meshOpt.distbound = 0.4;
    else
        if ~isnumeric(meshOpt.distbound) || meshOpt.distbound<=0
            error('Please enter a positive number for the mesh option "distbound".');
        end
    end
    if ~isfield(meshOpt,'reratio')
        meshOpt.reratio = 3;
    else
```

```
        if ~isnumeric(meshOpt.reratio) || meshOpt.reratio<=0
            error('Please enter a positive number for the mesh option "reratio".');
        end
    end
    if ~isfield(meshOpt,'maxvol')
        meshOpt.maxvol = 10;
    else
        if ~isnumeric(meshOpt.maxvol) || meshOpt.maxvol<=0
            error('Please enter a positive number for the mesh option "maxvol".');
        end
    end
    warning('You"re changing the advanced options of ROAST. Unless you know what you"re doing, please keep
mesh options default.');
end

if ~exist('simTag','var'), simTag = []; end

if ~exist('doResamp','var')
    doResamp = 0;
else
    if ~ischar(doResamp), error('Unrecognized option value. Please enter "on" or "off" for option "resampling".'); end
    if strcmpi(doResamp,'off')
        doResamp = 0;
    elseif strcmpi(doResamp,'on')
        doResamp = 1;
    else
        error('Unrecognized option value. Please enter "on" or "off" for option "resampling".');
    end
end

if ~exist('paddingAmt','var')
    paddingAmt = 0;
else
    if paddingAmt<=0 || mod(paddingAmt,1)~=0
        error('Unrecognized option value. Please enter positive integer value for option "zeroPadding". A recommended
value is 10.');
    end
end

if ~exist('conductivities','var')
    conductivities = struct('white',0.126,'gray',0.276,'csf',1.65,'bone',0.01,...
                    'skin',0.465,'air',2.5e-14,'gel',0.3,'electrode',5.9e7); % literature values
else
    if ~isstruct(conductivities), error('Unrecognized format of conductivity values. Please enter as a structure, with
field names as "white", "gray", "csf", "bone", "skin", "air", "gel" and "electrode".'); end
    conductivitiesNam = fieldnames(conductivities);
    if isempty(conductivitiesNam) ||
~all(ismember(conductivitiesNam,{'white';'gray';'csf';'bone';'skin';'air';'gel';'electrode'}))
        error('Unrecognized tissue names detected. Supported tissue names in the conductivity option are "white",
"gray", "csf", "bone", "skin", "air", "gel" and "electrode".');
    end
    if ~isfield(conductivities,'white')
        conductivities.white = 0.126;
    else
        if ~isnumeric(conductivities.white) || any(conductivities.white(:)<=0)
```

```
        error('Please enter a positive number for the white matter conductivity.');
      end
      if length(conductivities.white(:))>1, error('Tensor conductivity not supported by ROAST. Please enter a scalar value for conductivity.'); end
   end
   if ~isfield(conductivities,'gray')
      conductivities.gray = 0.276;
   else
      if ~isnumeric(conductivities.gray) || any(conductivities.gray(:)<=0)
         error('Please enter a positive number for the gray matter conductivity.');
      end
      if length(conductivities.gray(:))>1, error('Tensor conductivity not supported by ROAST. Please enter a scalar value for conductivity.'); end
   end
   if ~isfield(conductivities,'csf')
      conductivities.csf = 1.65;
   else
      if ~isnumeric(conductivities.csf) || any(conductivities.csf(:)<=0)
         error('Please enter a positive number for the CSF conductivity.');
      end
      if length(conductivities.csf(:))>1, error('Tensor conductivity not supported by ROAST. Please enter a scalar value for conductivity.'); end
   end
   if ~isfield(conductivities,'bone')
      conductivities.bone = 0.01;
   else
      if ~isnumeric(conductivities.bone) || any(conductivities.bone(:)<=0)
         error('Please enter a positive number for the bone conductivity.');
      end
      if length(conductivities.bone(:))>1, error('Tensor conductivity not supported by ROAST. Please enter a scalar value for conductivity.'); end
   end
   if ~isfield(conductivities,'skin')
      conductivities.skin = 0.465;
   else
      if ~isnumeric(conductivities.skin) || any(conductivities.skin(:)<=0)
         error('Please enter a positive number for the skin conductivity.');
      end
      if length(conductivities.skin(:))>1, error('Tensor conductivity not supported by ROAST. Please enter a scalar value for conductivity.'); end
   end
   if ~isfield(conductivities,'air')
      conductivities.air = 2.5e-14;
   else
      if ~isnumeric(conductivities.air) || any(conductivities.air(:)<=0)
         error('Please enter a positive number for the air conductivity.');
      end
      if length(conductivities.air(:))>1, error('Tensor conductivity not supported by ROAST. Please enter a scalar value for conductivity.'); end
   end
   if ~isfield(conductivities,'gel')
      conductivities.gel = 0.3;
   else
      if ~isnumeric(conductivities.gel) || any(conductivities.gel(:)<=0)
         error('Please enter a positive number for the gel conductivity.');
      end
```

```
        if length(conductivities.gel(:))>1 && length(conductivities.gel(:))~=length(elecName)
            error('You want to assign different conductivities to the conducting media under different electrodes, but didn''t tell ROAST clearly which conductivity each electrode should use. Please follow the order of electrodes you put in "recipe" to give each of them the corresponding conductivity in a vector as the value for the "gel" field in option "conductivities".');
        end
    end
    if ~isfield(conductivities,'electrode')
        conductivities.electrode = 5.9e7;
    else
        if ~isnumeric(conductivities.electrode) || any(conductivities.electrode(:)<=0)
            error('Please enter a positive number for the electrode conductivity.');
        end
        if length(conductivities.electrode(:))>1 && length(conductivities.electrode(:))~=length(elecName)
            error('You want to assign different conductivities to different electrodes, but didn''t tell ROAST clearly which conductivity each electrode should use. Please follow the order of electrodes you put in "recipe" to give each of them the corresponding conductivity in a vector as the value for the "electrode" field in option "conductivities".');
        end
    end
    warning('You''re changing the advanced options of ROAST. Unless you know what you''re doing, please keep conductivity values default.');
end

if length(conductivities.gel(:))==1
    conductivities.gel = repmat(conductivities.gel,1,length(elecName));
end
if length(conductivities.electrode(:))==1
    conductivities.electrode = repmat(conductivities.electrode,1,length(elecName));
end

% preprocess MRI data
if ~strcmpi(subj,'example/nyhead.nii') % only when it's not NY head

    if any(t1Data.hdr.dime.pixdim(2:4)<0.8) && ~doResamp
        warning('The MRI has higher resolution (<0.8mm) in at least one direction. This will make the modeling process more computationally expensive and thus slower. If you wish to run faster using just 1-mm model, you can ask ROAST to re-sample the MRI into 1 mm first, by turning on the "resampling" option.');
    end
    % check if high-resolution MRI (< 0.8 mm in any direction)

    if length(unique(t1Data.hdr.dime.pixdim(2:4)))>1 && ~doResamp
        warning('The MRI has anisotropic resolution. It is highly recommended that you turn on the "resampling" option, as the electrode size will not be exact if the model is built from an MRI with anisotropic resolution.');
    end
    % check if anisotropic resolution MRI

    if doResamp
        subjRS = resampToOneMM(subj);
    else
        subjRS = subj;
    end

    if paddingAmt>0
        subjRSPD = zeroPadding(subjRS,paddingAmt);
    else
```

```
        subjRSPD = subjRS;
    end

    if ~isempty(T2)
        T2 = realignT2(T2,subjRSPD);
    end
    % check if T2 is aligned with T1

else

    if ~exist('example/nyhead_T1orT2_masks.nii','file')
        unzip('example/nyhead_T1orT2_masks.nii.zip','example')
    end

    if doResamp
        error('The beauty of New York head is its 0.5 mm resolution. It''s a bad practice to resample it into 1 mm. Use another head "example/MNI152_T1_1mm.nii" for 1 mm model.');
    end

    if paddingAmt>0
        zeroPadding('example/nyhead_T1orT2_masks.nii',paddingAmt);
        subjRSPD = ['example/nyhead_padded' num2str(paddingAmt) '.nii'];
        if ~exist(['example/nyhead_padded' num2str(paddingAmt) '_T1orT2_seg8.mat'],'file')
            load('example/nyhead_T1orT2_seg8.mat','image','tpm','Affine');
            origin = inv(image.mat)*[0;0;0;1];
            origin = origin(1:3) + paddingAmt;
            image.mat(1:3,4) = [-dot(origin,image.mat(1,1:3));-dot(origin,image.mat(2,1:3));-dot(origin,image.mat(3,1:3))];
            save(['example/nyhead_padded' num2str(paddingAmt) '_T1orT2_seg8.mat'],'image','tpm','Affine');
        end
    else
        subjRSPD = subj;
    end

    if ~isempty(T2)
       warning('New York head selected. Any specified T2 image will be ignored.');
       T2 = [];
    end

end

% preprocess electrodes
[elecPara,ind2usrInput] = elecPreproc(subj,elecName,elecPara);

injectCurrent = (cell2mat(recipe(2:2:end)))';
if abs(sum(injectCurrent))>eps
    error('Electric currents going in and out of the head not balanced. Please make sure they sum to 0.');
end
elecName = elecName(ind2usrInput);
injectCurrent = injectCurrent(ind2usrInput);
conductivities.gel = conductivities.gel(ind2usrInput);
conductivities.electrode = conductivities.electrode(ind2usrInput);

% sort elec options
```

```
if length(elecPara)==1
   if size(elecSize,1)>1, elecPara.elecSize = elecPara.elecSize(ind2usrInput,:); end
   if ~ischar(elecOri) && size(elecOri,1)>1
      elecPara.elecOri = elecPara.elecOri(ind2usrInput,:);
   end
elseif length(elecPara)==length(elecName)
   elecPara = elecPara(ind2usrInput);
else
   error('Something is wrong!');
end

configTxt = [];
for i=1:length(elecName)
   configTxt = [configTxt elecName{i} ' (' num2str(injectCurrent(i)) ' mA), '];
end
configTxt = configTxt(1:end-2);

options =
struct('configTxt',configTxt,'elecPara',elecPara,'T2',T2,'meshOpt',meshOpt,'conductivities',conductivities,'uniqueTag
',simTag,'resamp',doResamp,'zeroPad',paddingAmt);

% log tracking
[dirname,baseFilename] = fileparts(subj);
if isempty(dirname), dirname = pwd; end

Sopt = dir([dirname filesep baseFilename '_*_options.mat']);
if isempty(Sopt)
   options = writeRoastLog(subj,options);
else
   isNew = zeros(length(Sopt),1);
   for i=1:length(Sopt)
      load([dirname filesep Sopt(i).name],'opt');
      isNew(i) = isNewOptions(options,opt);
   end
   if all(isNew)
      options = writeRoastLog(subj,options);
   else
      load([dirname filesep Sopt(find(~isNew)).name],'opt');
      options.uniqueTag = opt.uniqueTag;
   end
end
uniqueTag = options.uniqueTag;

fprintf('\n\n');
disp('=======================================================')
if ~strcmp(baseFilename,'nyhead')
   disp(['ROAST ' subj])
else
   disp('ROAST New York head')
end
disp('USING RECIPE:')
disp(configTxt)
disp('...and simulation options saved in:')
disp([dirname filesep baseFilename '_log,'])
disp(['under tag: ' uniqueTag])
```

```
disp('=====================================================')
fprintf('\n\n');

if ~strcmp(baseFilename,'nyhead')

    [~,baseFilenameRSPD] = fileparts(subjRSPD);

    if (isempty(T2) && ~exist([dirname filesep 'c1' baseFilenameRSPD '_T1orT2.nii'],'file')) ||...
            (~isempty(T2) && ~exist([dirname filesep 'c1' baseFilenameRSPD '_T1andT2.nii'],'file'))
        disp('=====================================================')
        disp('      STEP 1 (out of 6): SEGMENT THE MRI...         ')
        disp('=====================================================')
        start_seg(subjRSPD,T2);
    else
        disp('=====================================================')
        disp('         MRI ALREADY SEGMENTED, SKIP STEP 1         ')
        disp('=====================================================')
    end

    if (isempty(T2) && ~exist([dirname filesep baseFilenameRSPD '_T1orT2_masks.nii'],'file')) ||...
            (~isempty(T2) && ~exist([dirname filesep baseFilenameRSPD '_T1andT2_masks.nii'],'file'))
        disp('=====================================================')
        disp('    STEP 2 (out of 6): SEGMENTATION TOUCHUP...       ')
        disp('=====================================================')
        segTouchup(subjRSPD,T2);
    else
        disp('=====================================================')
        disp('   SEGMENTATION TOUCHUP ALREADY DONE, SKIP STEP 2    ')
        disp('=====================================================')
    end

else

    disp('=====================================================')
    disp(' NEW YORK HEAD SELECTED, GOING TO STEP 3 DIRECTLY...  ')
    disp('=====================================================')
    warning('New York head is a 0.5 mm model so is more computationally expensive. Make sure you have a decent
machine (>32GB memory) to run ROAST with New York head.')
    [~,baseFilenameRSPD] = fileparts(subjRSPD);

end

if ~exist([dirname filesep baseFilename '_' uniqueTag '_mask_elec.nii'],'file')
    disp('=====================================================')
    disp('     STEP 3 (out of 6): ELECTRODE PLACEMENT...       ')
    disp('=====================================================')
    hdrInfo = electrodePlacement(subj,subjRSPD,T2,elecName,options,uniqueTag);
else
    disp('=====================================================')
    disp('        ELECTRODE ALREADY PLACED, SKIP STEP 3        ')
    disp('=====================================================')
%     load([dirname filesep baseFilename '_' uniqueTag '_labelVol.mat'],'volume_elecLabel','volume_gelLabel');
    load([dirname filesep baseFilenameRSPD '_header.mat'],'hdrInfo');
end
```

```
end
```

generateElecMask_edited.m used inside roast_forAbaqus.m

```
function [vol_elec,vol_elecLabel] = generateElecMask_edited(elec_allCoord,coordRange,elec,doWarn)
% vol_elec = generateElecMask(elec_allCoord,coordRange,elec,doWarn)
%
% Convert point clouds of electrodes into 3D masks.
%
% (c) Yu (Andy) Huang, Parra Lab at CCNY
% yhuang16@citymail.cuny.edu
% April 2018

vol_elec = zeros(coordRange);
vol_isElec = zeros(coordRange); % for detecting electrodes overlap
vol_elecLabel = zeros(coordRange);
rnge = cell(length(elec_allCoord),1);

for i = 1:length(elec_allCoord)
   temp = elec_allCoord{i};
   if isempty(temp)
      error(['Electrode ' elec{i} ' goes out of image boundary. ROAST cannot proceed without a properly placed electrode. Please expand the input MRI by specifying the "zeroPadding" option.']);
   else
      ind = find(temp(:,1)>0 & temp(:,1)<=coordRange(1)...
         & temp(:,2)>0 & temp(:,2)<=coordRange(2)...
         & temp(:,3)>0 & temp(:,3)<=coordRange(3));
      if isempty(ind)
         error(['Electrode ' elec{i} ' goes out of image boundary. ROAST cannot proceed without a properly placed electrode. Please expand the input MRI by specifying the "zeroPadding" option.']);
        elec_allCoord{i} = [];
        rnge{i} = [];
      else
         if length(ind)<size(temp,1)
            if doWarn
               warning(['Part of the electrode ' elec{i} ' goes out of image boundary. ROAST can continue but results may not be accurate. It is recommended that you expand the input MRI by specifying the "zeroPadding" option.']);
            end
         end
         temp = temp(ind,:);
         indElec = sub2ind(size(vol_elec),temp(:,1),temp(:,2),temp(:,3));
         if any(vol_isElec(indElec))
            error(['Electrode ' elec{i} ' overlaps with Electrode ' elec{i-1} '. ROAST cannot continue as overlapping electrodes will confuse ROAST when setting up the boundary conditions for the model. To avoid overlapping, please do not place two electrodes too close to each other, or reduce the size of any neighboring electrodes.']);
         end
         vol_elec(indElec) = 1;
         vol_elec(indElec) = i; % individual mask/color for each elec & gel
         vol_isElec(indElec) = 1;
         vol_elecLabel(indElec) = i;
         elec_allCoord{i} = temp;
         rnge{i} = [max(temp);min(temp)];
      end
   end
end
```

```
% allCoord = cell2mat(elec_allCoord);
% vol_elec(sub2ind(size(vol_elec),allCoord(:,1),allCoord(:,2),allCoord(:,3)))=1;
```

prepareForAbaqus.m

```
function prepareForAbaqus(Image,elecgellabelsMat,optMat)

[dirname,~,~]=fileparts(Image);
%%

vol=load_untouch_nii(Image);
data=vol.img;
data_backup=data;
   for v=1:6
      data(data~=v)=0;
      data(data==v)=1;
      vol.img=data;
      %save_untouch_nii(vol,sprintf('class_%d_%s',v,n));
      save_untouch_nii(vol,fullfile(dirname,sprintf('class_%d',v)));
      data=data_backup;
   end
%%
load(elecgellabelsMat)
load(optMat)
%hdrInfo= electrodePlacement(Image,Image,[],elecConfig,opt,[]);

%%
template = load_untouch_nii(fullfile(dirname,'class_5.nii'));
scalp = template.img;
nasion=landmarks_original(1,:);inion=landmarks_original(2,:); right=landmarks_original(3,:);
left=landmarks_original(4,:);front_neck=landmarks_original(5,:);back_neck=landmarks_original(6,:);
e1 = right-left; e1 = e1/norm(e1);
e2 = nasion-inion; e2 = e2/norm(e2);
e3 = nasion-front_neck; e3 = e3/norm(e3); % detect the orientation of the head based on the anatomical landmarks

[~,perm1] = max(abs(e1)); [~,perm2] = max(abs(e2)); [~,perm3] = max(abs(e3));
isFlip = [sign(e1(perm1)) sign(e2(perm2)) sign(e3(perm3))]; % detect if the head is flipped or not in each direction
compared to RAS system

perm = [perm1,perm2,perm3]; % permutation order into RAS
[~,iperm] = sort(perm); % inverse permutation order back to original orientation of the head
scalp = permute(scalp,perm); % permute the head into RAS
[Nx, Ny, Nz]=size(scalp); % size of head in RAS orientation

%% for elec rnge mat
elec_C_final = cell(1,size(electrode_coord,1));
rnge = cell(1,size(electrode_coord,1));
isOut = zeros(length(rnge),1);
for i = 1:size(electrode_coord,1)
   elec_C_final{i} = zeros(size(elec_C{i},1),3);
   rng = zeros(2,3);
   for j=1:size(elec_C{i},1)
      if elec_C{i}(j,1)>0 && elec_C{i}(j,1)<=Nx && elec_C{i}(j,2)>0 && elec_C{i}(j,2)<=Ny &&
elec_C{i}(j,3)>0 && elec_C{i}(j,3)<=Nz
         elec_C_final{i}(j,1) = elec_C{i}(j,iperm(1));
         elec_C_final{i}(j,2) = elec_C{i}(j,iperm(2));
         elec_C_final{i}(j,3) = elec_C{i}(j,iperm(3));
```

```
        end % permute back electrode coordinates to match the original orientation of the head
    end
    if all(sum(elec_C_final{i},2))
        rng(1,:) = max(elec_C_final{i});
        rng(2,:) = min(elec_C_final{i});
        rnge{i} = rng;
    else if any(sum(elec_C_final{i},2)) % in case of some electrode voxels go out of image boundary then
elec_C_final has 0
            elec_C_temp = elec_C_final{i}(sum(elec_C_final{i},2)>0,:);
            rng(1,:) = max(elec_C_temp);
            rng(2,:) = min(elec_C_temp);
            rnge{i} = rng;
        else rnge{i} = []; % this is the case that the electrode completely goes out of image boundary, thus this
electrode actually does not exist
            warning(['Electrode #' num2str(i) ' goes out of image boundary!']);
            isOut(i) = 1;
        end
    end
end
if any(isOut)
    warning('Some of the electrodes go out of image boundary, the program can continue, but it is highly
recommended that you expand the image by adding empty slices on the boundaries.');
end
for i=1:length(rnge)
    if ~isempty(rnge{i})
        rnge{i} = rnge{i}.*repmat(template.hdr.dime.pixdim(2:4),2,1);
    end
end % use NIFTI header info to convert range info into world coordinates for subsequent electrode labeling ANDY
2014-08-12
    % It's in fact a hack, i.e., only applies the scaling to the range
    % information, to match the pseudo-world space of the mesh coordinates
    % (generated by ScanIP)
% Get the range of coordinates for each electrode for subsequent electrode labelling
% (so that we can use a script to automatically specify anode and cathode
% when solving current flow in Abaqus, and also calculate EXACT energized area for
% each electrode)
save(fullfile(dirname,'rnge_elec.mat'),'rnge');


%% for gel rnge mat
gel_C_final = cell(1,size(electrode_coord,1));
for i = 1:size(electrode_coord,1)
    gel_C_final{i} = zeros(size(gel_C{i},1),3);
    for j=1:size(gel_C{i},1)
        if gel_C{i}(j,1)>0 && gel_C{i}(j,1)<=Nx && gel_C{i}(j,2)>0 && gel_C{i}(j,2)<=Ny && gel_C{i}(j,3)>0
&& gel_C{i}(j,3)<=Nz
            gel_C_final{i}(j,1) = gel_C{i}(j,iperm(1));
            gel_C_final{i}(j,2) = gel_C{i}(j,iperm(2));
            gel_C_final{i}(j,3) = gel_C{i}(j,iperm(3));
        end % permute back gel coordinates to match the original orientation of the head
    end
end % Get ready for the generation of the range of coordinates for gel in the following codes
% ANDY 2014-03-04

disp('constructing electrode and gel volume to be exported...')
```

```
for i = 1:size(electrode_coord,1)
   for j=1:size(gel_C{i},1)
      if gel_C{i}(j,1)>0 && gel_C{i}(j,1)<=Nx && gel_C{i}(j,2)>0 && gel_C{i}(j,2)<=Ny && gel_C{i}(j,3)>0
&& gel_C{i}(j,3)<=Nz
         volume_gel(gel_C{i}(j,1), gel_C{i}(j,2), gel_C{i}(j,3)) = 1;
      end
   end
   for j=1:size(elec_C{i},1)
      if elec_C{i}(j,1)>0 && elec_C{i}(j,1)<=Nx && elec_C{i}(j,2)>0 && elec_C{i}(j,2)<=Ny &&
elec_C{i}(j,3)>0 && elec_C{i}(j,3)<=Nz
         volume_elec(elec_C{i}(j,1), elec_C{i}(j,2), elec_C{i}(j,3)) = 1;
      end
   end
end

rnge = cell(1,size(electrode_coord,1));
for i = 1:size(electrode_coord,1)
   isGel = zeros(size(gel_C_final{i},1),1);
   for j=1:length(isGel)
      if all(gel_C_final{i}(j,:)) && volume_gel(gel_C_final{i}(j,1),gel_C_final{i}(j,2),gel_C_final{i}(j,3))==1
        isGel(j)=1;
      end
   end
   gel_C_final{i} = gel_C_final{i}(isGel==1,:); % remove those gel coordinates that go into scalp/electrode/bone
   rng = zeros(2,3);
   if all(sum(gel_C_final{i},2))
      rng(1,:) = max(gel_C_final{i});
      rng(2,:) = min(gel_C_final{i});
      rnge{i} = rng;
   else if any(sum(gel_C_final{i},2)) % in case of some gel voxels go out of image boundary then gel_C_final has 0
         gel_C_temp = gel_C_final{i}(sum(gel_C_final{i},2)>0,:);
         rng(1,:) = max(gel_C_temp);
         rng(2,:) = min(gel_C_temp);
         rnge{i} = rng;
      else rnge{i} = []; % this is the case that the gel completely goes out of image boundary, thus this gel actually
does not exist
         warning(['Gel of electrode #' num2str(i) ' goes out of image boundary!']);
      end
   end
end
for i=1:length(rnge)
   if ~isempty(rnge{i})
      rnge{i} = rnge{i}.*repmat(template.hdr.dime.pixdim(2:4),2,1);
   end
end % use NIFTI header info to convert range info into world coordinates for subsequent gel labeling ANDY 2014-
08-12
   % It's in fact a hack, i.e., only applies the scaling to the range
   % information, to match the pseudo-world space of the mesh coordinates
   % (generated by ScanIP)
% Get the range of coordinates for each gel for subsequent gel labelling
% (so that we can use a script to automatically calculate EXACT energized area for each electrode)
save(fullfile(dirname,'rnge_gel.mat'),'rnge');
end
```

tpsych_bipolar_tDCS.m

```
function tpysch_bipolar_tDCS(dirname,fname,currden)
% This script will process the mesh file and prepare Abaqus inp file for bipolar electrode configuration
% Input:
% dirname: directory of the inp file;
% fname: name of the inp file without inp extension
% currden: current density (A/m^2) to inject
%
% You will need function meshProcess.m
% Place rnge_gel.mat, rnge_elec.mat in same directory
%
% Modified Code: Sunil Kalmady, June 2016
% Tpsych Lab, NIMHANS, Bangalore
% Original Code
% (c) Jacek Dmochowski, March 2011
% (c) Yu Huang (Andy), February 2015
% (c) Yu Huang (Andy), March 2014
% (c) Yu Huang (Andy), March 2011
% The Neural Engineering Lab, Dept. of Biomedical Engineering, City College of New York

cd(dirname)
filename_in=[fname '.inp']; % input mesh file name
filename_out=[fname '_abaqus.inp']; % mesh file submitted to Abaqus for solving

% Process the mesh file (.inp) generated from Simpleware.
% Read the mesh file, label the electrodes and gel for solving in Abaqus.
meshProcess(dirname,filename_in,2,1)
fprintf('Generating Abaqus input file...\n');

% PARAMETERS
load en;  % cell array of nodes belonging to each electrode
load ee;  % cell array of elements belonging to each electrode
mkdir('./scratch') % temporary directory used by Abaqus

is = 1; % labels of anodes
ib = 2; % label of grounded electrode (cathode)

sigma=[   ...
   2.5e-14; ...  % air
   0.465; ... % skin
   0.01; ...     % bone
   0.276; ...    % gray
   0.126; ...    % white
   1.65; ...     % csf
   5.9e7; ... % elec
   0.3 ; ... % gel
   ];

% conductivities for all tissue types, coming in the priority order in ScanIP
sigma=sigma(end:-1:1); cc=1;
% because in the mesh file (inp), the 8 tissues come in an inversed order of priority order in ScanIP

%  STEPS 1 & 2 -- add material properties + change all element types to thermal electric
```

```
fin = fopen(filename_in);
fout = fopen(filename_out,'w');
while ~feof(fin)
   s = fgetl(fin);
   s = strrep(s, 'C3D4', 'DC3D4E'); % Solving type: thermal electric element
   fprintf(fout,'%s \n',[s]);
   if length(s)>=9
      if strcmp(s(1:9),'*MATERIAL')
         s2='*ELECTRICAL CONDUCTIVITY';
         s3=num2str(sigma(cc)); cc=cc+1;
         fprintf(fout,'%s \n %s \n',s2,s3); % write conductivity values to each tissue type in .inp file
      end
   end
end
fclose(fin);
fclose(fout);

% STEP 3: add boundary condition
% define set of nodes in grounded electrode
M='*NSET, NSET=GELEC';
dlmwrite(filename_out, M, 'delimiter', '','-append');

dlmwrite(filename_out, en{ib}, 'delimiter', ',','-append','precision','%7d');

M='*BOUNDARY';
dlmwrite(filename_out, M, 'delimiter', '','-append');

M='GELEC,9,9';
dlmwrite(filename_out, M, 'delimiter', '','-append');

% STEP 4: add analysis step
% create set of elements that will be energized
M='*ELSET, ELSET=SELEC';
dlmwrite(filename_out, M, 'delimiter', '','-append');
dlmwrite(filename_out, ee{is}, 'delimiter', ',','-append','precision','%7d');

% create surface
M='*SURFACE, TYPE=ELEMENT, NAME=SSURF';
dlmwrite(filename_out, M, 'delimiter', '','-append');

M='SELEC,';
dlmwrite(filename_out, M, 'delimiter', '','-append');

% add step with load
M='*STEP,NAME=STEP-1,AMPLITUDE=STEP,UNSYMM=NO'; % quasi-static
dlmwrite(filename_out, M, 'delimiter', '','-append');
M='*Coupled Thermal-Electrical, steady state, deltmx=0.';
dlmwrite(filename_out, M, 'delimiter', '','-append')
M='1., 1., 1e-05, 1.,  ';
dlmwrite(filename_out, M, 'delimiter', '','-append')

M='*SOLUTION TECHNIQUE,TYPE=SEPARATED';
dlmwrite(filename_out, M, 'delimiter', '','-append')
M='*Dsecurrent';
```

```
dlmwrite(filename_out, M, 'delimiter', '','-append')
M=[' SSURF, CS, ' num2str(currden)]; % injecting current
dlmwrite(filename_out, M, 'delimiter', '','-append')
M='*Restart, write, frequency=0';
dlmwrite(filename_out, M, 'delimiter', '','-append')
M='*Output, field, frequency=99999';
dlmwrite(filename_out, M, 'delimiter', '','-append')
M='*Node Output';
dlmwrite(filename_out, M, 'delimiter', '','-append')
M='COORD,';
dlmwrite(filename_out, M, 'delimiter', '','-append')
M='*Element Output, directions=YES';
dlmwrite(filename_out, M, 'delimiter', '','-append')
M='EPG,'; % output EPG (electric potential gradient, i.e., electric field)
dlmwrite(filename_out, M, 'delimiter', '','-append')
M='*Output, history, variable=PRESELECT';
dlmwrite(filename_out, M, 'delimiter', '','-append')
M='*END STEP';
dlmwrite(filename_out, M, 'delimiter', '','-append')

fprintf('Abaqus input file has been generated...\n');

% STEP 5: call Abaqus to solve the model
% system(['abaqus job=' filename_out ' scratch=./scratch memory="80 %"'])
% you can specify how many CPU cores can be used for the computation

%fprintf('waiting until solution completes...\n');
%fprintf('Check abaqus log file for completion of analysis...\n');
```

meshprocess.m used inside tpsych_bipolar_tDCS.m

```
function meshProcess(dirname,fname,elecLoc,gelLoc)

% meshProcess(dirname,fname,elecLoc,gelLoc)
%
% Process the mesh file (.inp) generated from Simpleware.
% It reads the mesh file, labels the electrodes for solving in Abaqus, and
% labels the gel to calculate EXACT energized area for each electrode.
%
% Input:
% dirname: directory of the mesh file;
% fname: name of the mesh file;
% elecLoc: the location of electrode in the .inp file;
% gelLoc: the location of gel in the .inp file;
%
% Output:
% en.mat: nodes belonging to each electrode, used to define grounded
% electrode (cathode);
% ee.mat: elements belonging to each electrode, used to define stimulating
% electrode (anode);
% elecArea.mat: EXACT energized area for each electrode, used to PRECISELY
% scale the Abaqus solution of each electrode pair;
%
% The 'en' and 'ee' will be used in the script 'main.m' for automatically
% solving the FEM for all possible bipolar electrode configurations.
% The 'elecArea' will be used in the optimization script for computation of
% optimal electrode montage for tDCS.
%
% (c) Yu Huang (Andy), February 2015
% (c) Yu Huang (Andy), March 2014
% (c) Yu Huang (Andy), March 2011
% (c) Jacek Dmochowski, March 2011
% The Neural Engineering Lab, Dept. of Biomedical Engineering, City College of New York
% Send bugs to yhuang16@citymail.cuny.edu

% extract mesh from .inp file
cd(dirname)
fid=fopen(fname);
subjname=strsplit(fname,'_');
subjname1=subjname{1};
rnge_elec_filename=[subjname1,'_rnge_elec.mat'];
rnge_gel_filename=[subjname1,'_rnge_gel.mat'];



% find beginning of node listing, and meanwhile count how many tissue masks
% are there in the model
numOfTissue = 0;
mystr='';
while strcmp(mystr,'*NODE')==0
    C = textscan(fid, '%5c', 1);
    %C = fopen(fid, '%5c', 1);
```

```
    mystr=C{1};
    if strcmp(mystr,'*MATE')==1
        numOfTissue = numOfTissue+1;
    end
    fgetl(fid);
end
fprintf('Found beginning of node listings.\n');
% now read in nodes
fprintf('reading in nodes...\n');
fgetl(fid);
fgetl(fid);
fgetl(fid);
N = textscan(fid, '%f %f %f %f', 'Delimiter', ',');

E = cell(numOfTissue,1);
for k = 1:numOfTissue
    mystr='';
    while strcmp(mystr,'*ELEMENT')==0
        C = textscan(fid, '%8c', 1);
        mystr=C{1};
        fgetl(fid);
    end
    fprintf('Found beginning of element listings for Tissue #%d.\n',k);
    % now read in elements
    fprintf('reading in elements for Tissue #%d...\n',k);
    E{k} = cell2mat(textscan(fid, '%d %d %d %d %d', 'Delimiter', ','));
end

fclose(fid);

nodes=cell2mat(N);
nodes(:,2:4) = nodes(:,2:4)+1; % for 1 mm model

element_electrode = E{elecLoc};

% generate coordinates for each electrode element
X = zeros(size(element_electrode,1),3);
for e = 1:size(element_electrode,1)
    X(e,:) = mean ( nodes(element_electrode(e,2:5),2:4) )';  % centroid of element
end

% load the range of coordinates for each electrode
load(rnge_elec_filename)
% labelling electrodes
disp('labeling electrodes...')
label_elec = zeros(size(X,1),1);
for i = 1:size(X,1)
    offset = 0;
    while label_elec(i)==0
        for k = 1:length(rnge)
            if ~isempty(rnge{k}) && X(i,1)>rnge{k}(2,1)-offset && X(i,1)<rnge{k}(1,1)+offset &&
X(i,2)>rnge{k}(2,2)-offset && X(i,2)<rnge{k}(1,2)+offset && X(i,3)>rnge{k}(2,3)-offset &&
X(i,3)<rnge{k}(1,3)+offset
                label_elec(i) = k;
            end
```

```
        end
        offset = offset+0.1;
    end
    if mod(i,100) == 0
        fprintf('%f%% completed...\n',i*100/length(X))
    end
end
disp('labeling electrodes DONE!')

save label_elec.mat label_elec

% data preparation
en = cell(1,length(rnge));
ee = cell(1,length(rnge));
for n = 1:length(rnge)
    if ~isempty(find(label_elec==n))
        en{n}=nodes(element_electrode(label_elec==n,2:5),1); % nodes belonging to each electrode, used to define
grounded electrode
        ee{n}=element_electrode(label_elec==n,1); % elements belonging to each electrode, used to define stimulating
electrode
    end
end
save en.mat en
save ee.mat ee

element_gel = E{gelLoc};

% generate coordinates for each gel element
X = zeros(size(element_gel,1),3);
for g = 1:size(element_gel,1)
    X(g,:) = mean ( nodes(element_gel(g,2:5),2:4) )';  % centroid of element
end

% load the range of coordinates for each gel
load (rnge_gel_filename);
% labelling gel
disp('labeling gel...')
label_gel = zeros(size(X,1),1);
for i = 1:size(X,1)
    offset = 0;
    while label_gel(i)==0
        for k = 1:length(rnge)
            if ~isempty(rnge{k}) && X(i,1)>rnge{k}(2,1)-offset && X(i,1)<rnge{k}(1,1)+offset &&
X(i,2)>rnge{k}(2,2)-offset && X(i,2)<rnge{k}(1,2)+offset && X(i,3)>rnge{k}(2,3)-offset &&
X(i,3)<rnge{k}(1,3)+offset
                label_gel(i) = k;
            end
        end
        offset = offset+0.1;
    end
    if mod(i,100) == 0
        fprintf('%f%% completed...\n',i*100/length(X))
    end
end
disp('labeling gel DONE!')
```

```
save label_gel.mat label_gel

% area calculation
disp('calculating energized area for each electrode...')
elecArea = zeros(length(rnge),1);
element_electrode = double(element_electrode);
element_gel = double(element_gel);
for n = 1:length(elecArea)
    % compute surface faces and vertices for each electrode and gel
    tri = element_electrode(label_elec==n,2:5);
    trep = triangulation(tri,nodes(:,2:4));
    [faces_elec,verts_elec]=freeBoundary(trep);

    tri = element_gel(label_gel==n,2:5);
    trep = triangulation(tri,nodes(:,2:4));
    [faces_gel,verts_gel]=freeBoundary(trep);

    [~,iE,iG] = intersect(verts_elec,verts_gel,'rows');
    tempTag = ismember(faces_elec,iE);

    faces_overlap = faces_elec(sum(tempTag,2)==3,:); % find the overlap faces

    % calculate the total surface area of each electrode
    a = verts_elec(faces_elec(:, 2), :) - verts_elec(faces_elec(:, 1), :);
    b = verts_elec(faces_elec(:, 3), :) - verts_elec(faces_elec(:, 1), :);
    c = cross(a, b, 2);
    totalElecArea = sum(0.5*sqrt(sum(c.^2, 2)));
    % calculate the area of the overlap between electrode and gel
    a = verts_elec(faces_overlap(:, 2), :) - verts_elec(faces_overlap(:, 1), :);
    b = verts_elec(faces_overlap(:, 3), :) - verts_elec(faces_overlap(:, 1), :);
    c = cross(a, b, 2);
    elecGelOverlapArea = sum(0.5*sqrt(sum(c.^2, 2)));

    elecArea(n) = totalElecArea-elecGelOverlapArea;
    % the EXACT energized area is the total electrode surface area minus the area of the overlap between electrode
and gel
end
save elecArea.mat elecArea
disp(rnge_gel_filename)
disp(rnge_elec_filename)
fprintf('Mesh Processing done for %s!',fname)
```

interpolate_abaqus_field_part1.m

```
% Create all_epg.mat for further processing in part 2 script
dirname='E:\Gaurav\tDCS_data_for_scanIP\Abaqus_work_tDCS\00_interplote_efield\Remaining';
cd(dirname)
%%
subj=dir('*');
b=9:16; %dlelete duplicate files
for subjs=3:length(subj)
   subjdir=fullfile(subj(subjs).folder,subj(subjs).name);
   subjfile=[subj(subjs).name,'_efield_abaqus.rpt'];
   getEPG(subjdir,subjfile);
   delete(fullfile(subj(subjs).folder,subj(subjs).name,sprintf('Field_Output_%d.mat',b(1))));
   delete(fullfile(subj(subjs).folder,subj(subjs).name,sprintf('Field_Output_%d.mat',b(2))));
   delete(fullfile(subj(subjs).folder,subj(subjs).name,sprintf('Field_Output_%d.mat',b(3))));
   delete(fullfile(subj(subjs).folder,subj(subjs).name,sprintf('Field_Output_%d.mat',b(4))));
   delete(fullfile(subj(subjs).folder,subj(subjs).name,sprintf('Field_Output_%d.mat',b(5))));
   delete(fullfile(subj(subjs).folder,subj(subjs).name,sprintf('Field_Output_%d.mat',b(6))));
   delete(fullfile(subj(subjs).folder,subj(subjs).name,sprintf('Field_Output_%d.mat',b(7))));
   delete(fullfile(subj(subjs).folder,subj(subjs).name,sprintf('Field_Output_%d.mat',b(8))));
   matfiles=dir(fullfile(subj(subjs).folder,subj(subjs).name,'Field_Output_*.mat'));
   epg=0;
   epg=cell(epg);
      for files=1:length(matfiles)
         load(fullfile(matfiles(files).folder,matfiles(files).name));
         epg{files}=EPGData;
      end
   epg_all=vertcat(epg{1,1},epg{1,2},epg{1,2},epg{1,3},epg{1,5},epg{1,6},epg{1,7},epg{1,8});
   save(fullfile(subj(subjs).folder,subj(subjs).name,'all_epg.mat'),'epg_all');
end
%%
```

getEPG.m used inside interpolate_abaqus_field_part1.m

```
function getEPG(dirname,FileName)
% Splits Abaqus RPT file into separate field outputs / regions (saved as txt) and gets Nodelabel,
% Nodal Coordinates (Coord1, Coord2, Coord3) and averaged EPG values (saved as mat)

% FileName is the name of the RPT file generated by Abaqus/CAE.
%   To generate an RPT field variable file, do the following in ABAQUS/CAE:
%   1. In the Visualizaiton Module, go to "Report -> Field Output...".
%   2. In the "Variable" tab, choose "Unique Nodal" as the position type.
%   3. Make sure that these are the only boxes checked - COORD1,
%   COORD2,COORD3, EPG Magnitude
%   4. In the "Setup" tab under "Output Format", make sure the layout style
%   is set to "Single table for all field output variables".
%   5. Under "Data", uncheck "Column totals" and "Column
%   min/max" as these are not required.
cd(dirname)
Str   = fileread(FileName);
CStr  = regexp(Str, '\n', 'split');
```

```
Index = [find(strncmp(CStr, 'Field Output reported', 21)), length(CStr) + 1];
for i = 1:(length(Index) - 1)
  Foid = fopen(sprintf('Field_Output_%d.txt', i), 'w');
  if Foid == - 1, error('Less than one instance of field output'); end
  fprintf(Foid, '%s\n', CStr{Index(i):Index(i + 1) - 1});
  fclose(Foid);
end

for i = 1:(length(Index) - 1)
    fid = fopen(sprintf('Field_Output_%d.txt', i), 'r');

    while ~feof(fid)
       fgetl(fid);
       cellData = textscan(fid, '%f %f %f %f %f %f %f');
    end

    EPGData = cell2mat(cellData);
    save(sprintf('Field_Output_%d.mat', i),'EPGData')
    fclose(fid);
end
end
```

interpolate_abaqus_field_part2.m

```
clc
dirname='E:\Gaurav\tDCS_data_for_scanIP\Abaqus_work_tDCS\00_interplote_efield';
cd(dirname)
%%
subj=dir('*');
for subjs=3:length(subj)
    subjdir=fullfile(subj(subjs).folder,subj(subjs).name);
    P1=load_untouch_nii(fullfile(subjdir,[subj(subjs).name,'.nii']));
    load(fullfile(subjdir,'all_epg.mat'));
    EPGdata=epg_all;
    [xi,yi,zi] = ndgrid(1:192,1:256,1:256);
    ef_all = zeros([192 256 256 3]);
    F = scatteredInterpolant(EPGdata(:,2:4), EPGdata(:,5));
    ef_all(:,:,:,1) = F(xi,yi,zi);
    F = scatteredInterpolant(EPGdata(:,2:4), EPGdata(:,6));
    ef_all(:,:,:,2) = F(xi,yi,zi);
    F = scatteredInterpolant(EPGdata(:,2:4), EPGdata(:,7));
    ef_all(:,:,:,3) = F(xi,yi,zi);
    ef_mag = sqrt(sum(ef_all.^2,4));
    img=spm_vol(fullfile(subjdir,[subj(subjs).name,'_T1orT2_masks_allmask.nii']));
    img1=spm_read_vols(img);
    ef_mag1=ef_mag.*img1;
    template = load_untouch_nii(fullfile(subjdir,[subj(subjs).name,'_T1orT2_masks.nii']));
    template.hdr.dime.datatype = 16;
    template.hdr.dime.bitpix = 32;
    template.hdr.dime.scl_slope = 1; % so that display of NIFTI will not alter the data
    template.hdr.dime.cal_max = 0;
    template.hdr.dime.cal_min = 0;
    template.img = single(ef_mag1);
    template.hdr.dime.glmax = max(ef_mag1(:));
    template.hdr.dime.glmin = min(ef_mag1(:));
    template.hdr.hist.descrip = 'EF mag';
    save_untouch_nii(template,fullfile(subjdir,[subj(subjs).name,'_interpolatedEfield.nii']));
end
```