Tutorial

Simulating and estimating EEG Sources
============

In this tutorial, we will learn how simple cortical current sources can generate complicated patterns of EEG potentials at the scalp. We will then see how, under certain conditions, the underlying neural sources can be effectively localized from the scalp measurements.
However, the tutorial also aims to illustrate how the solution to the of the inverse EEG problem can be easily confounded e.g. by the presence of noisy background EEG activity or the depth of the underlying sources.

This has been implemented for Octave or Matlab.

The current tutorial is based on a tutorial which was prepared for the course [Advanced Human Neurophysiology][ahn],given at the [VU University Amsterdam][vu], and organized by [Klaus Linkenkaer-Hansen][klaus]. The original author of the tutorial is [German Gomez-Herrero][german.gomezherrero@kasku.org]. Modifications and adaptation to study the EEG source problem by A. Koulouri.

Part I: A realistic head model

Script: PartI_SourceImagingProblem.m

In this tutorial we use a 2D realistic electromagnetic model to map intracerebral current sources (modelled as dipoles) to a set of EEG sensors. Our model is based on a real MRI scan from an unknown subject from OASIS database. Below are the main steps that we need to take in order to build our head model. The steps of this part have been already performed for you, but you will need to carry out the other steps to successfully build models that we can play around with.

Step 1: Tissue segmentation

Different brain tissues have different characteristic conductivities. A realistic head model needs to take this fact into account and, therefore, tissue types need to be segmented. Typically, only scalp, skull and cerebral tissues need to be identified in order to build a reasonably accurate electromagnetic model. In the old times, brain tissues had to be manually or semi-automatically segmented. This was not only time consuming but also highly subjective. Fortunately, the standardization of MRI scanning technology has allowed the development of extremely accurate fully-automated tissue segmentation packages. There are multiple freely available alternatives but, arguably, the most robust and widely tested software package is Freesurfer, developed at the Athinoula A. Martinos Center for Biomedical Imaging.

Step 2: Discretization

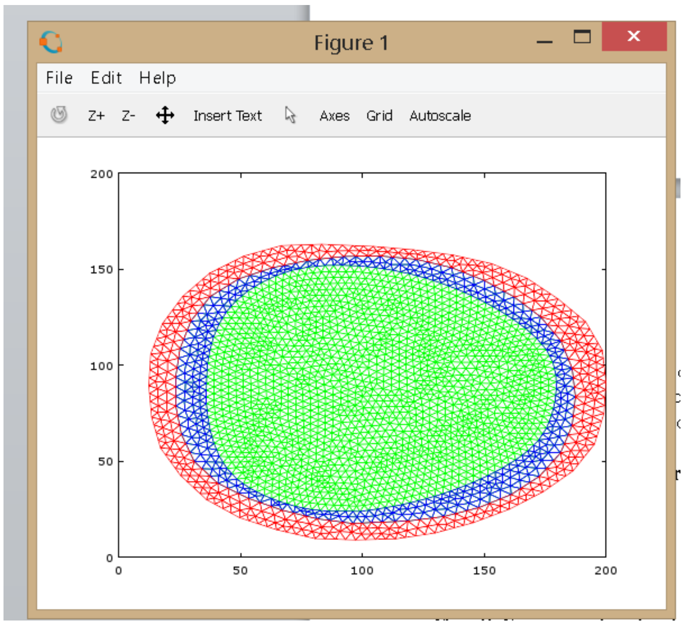
Once different tissues have been identified, we need to build closed triangulated surfaces between all relevant tissue types. This procedure is called discretization and the whole domain is expressed with a set of nodes (or points) and elements (triangles). In our model we have three compartments: the *outer skin* (scalp) surface (on which the EEG sensors rest), the *outer skull (Skull)* surface, and the *inner skull (Brain)* surface (within which the *EEG sources* will be located). These surfaces can be automatically generated using a variety of software packages. In this tutorial we used Comsol, however there are many other packages such as MNE software, which is also maintained by the Athinoula A. Martinos Center.

If you have already run the script then if you want to see the different compartment you can write the following piece of code in the command window

```
figure; hold on
triplot(Geometry.Brain.tri',Geometry.Pix.pnt(:,1),Geometry.Pix.pnt(:,2),'g')
triplot(Geometry.skull.tri',Geometry.Pix.pnt(:,1),Geometry.Pix.pnt(:,2),'b')
triplot(Geometry.scalp.tri',Geometry.Pix.pnt(:,1),Geometry.Pix.pnt(:,2),'r')
```

field .tri is the Delaunay triangulation over the grid points .pnt

The result of the previous code gives

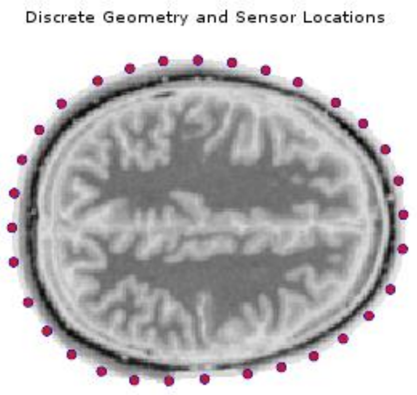


Step 3: Sensor co-localization

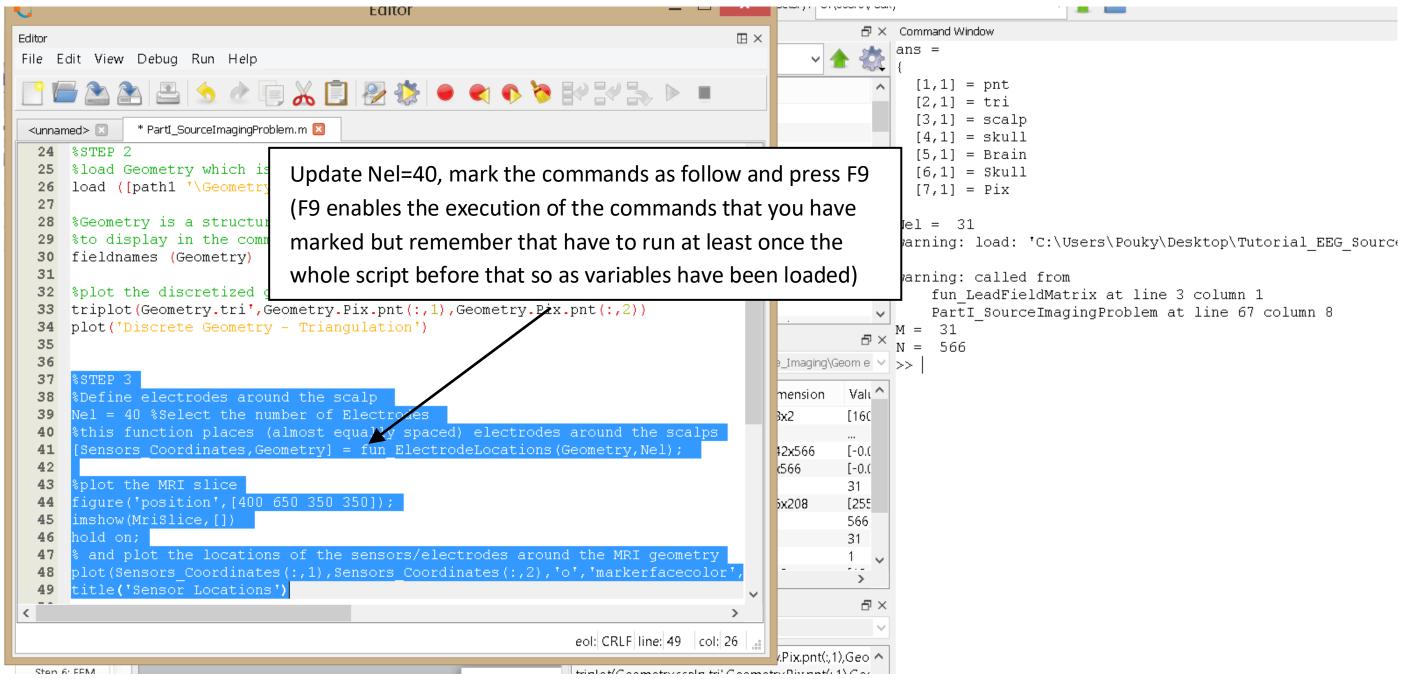
The following piece of code places 31 sensors around the surface of the scalp.

```
%STEP 3
%Define electrodes around the scalp
Nel = 31 %Select the number of Electrodes
%this function places (almost equally spaced) electrodes around the scalps
[Sensors_Coordinates,Geometry] = fun_ElectrodeLocations(Geometry,Nel);
```

If you plot the sensors around the domain you have the following figure



Also if you want to change the number of electrodes (e.g. Nel=20 or Nel=40).



Update Nel=40, mark the commands as follow and press F9 (F9 enables the execution of the commands that you have marked but remember that have to run at least once the whole script before that so as variables have been loaded)

```
%STEP 2
%load Geometry which is
load ([path1 '\Geometry

%Geometry is a structur
%to display in the comm
fieldnames (Geometry)

%plot the discretized g
triplot(Geometry.tri',Geometry.Pix.pnt(:,1),Geometry.Pix.pnt(:,2))
plot('Discrete Geometry - Triangulation')


%STEP 3
%Define electrodes around the scalp
Nel = 40 %Select the number of Electrodes
%this function places (almost equally spaced) electrodes around the scalps
[Sensors_Coordinates,Geometry] = fun_ElectrodeLocations(Geometry,Nel);

%plot the MRI slice
figure('position',[400 650 350 350]);
imshow(MriSlice,[])
hold on;
% and plot the locations of the sensors/electrodes around the MRI geometry
plot(Sensors_Coordinates(:,1),Sensors_Coordinates(:,2),'o','markerfacecolor',
title('Sensor Locations')
```

```
ans =
{
  [1,1] = pnt
  [2,1] = tri
  [3,1] = scalp
  [4,1] = skull
  [5,1] = Brain
  [6,1] = Skull
  [7,1] = Pix

Nel =  31
warning: load: 'C:\Users\Pouky\Desktop\Tutorial_EEG_Sourc

warning: called from
    fun_LeadFieldMatrix at line 3 column 1
    PartI_SourceImagingProblem at line 67 column 8
M =  31
N =  566
>>
```

Step 5: Intracerebral source space

In order to build an electromagnetic model we need to specify the spatial locations where EEG sources may be located. In this tutorial, this is done by finding the points of the triangulation which belong to the grey matter:

```
%STEP 4
%Define canditate source locations (only in the cortical area)
%Options
%Opt = 1 (superficial locations only)
%Opt = 2 (the whole Gray matter)
Opt = 1;
[Cand_Source_Coordinates,Geometry]= CandSourceLocations(Opt,Geometry);
figure('position',[750 650 350 350]);
imshow(MriSlice,[])
hold on
plot(Cand_Source_Coordinates(:,1),Cand_Source_Coordinates(:,2),'xr');
title('Canditate Source Locations - GM')
```

This piece of code has two options

Opt=1 consider as candidate EEG source only the superficial area of the grey matter while Opt = 2 the whole grey matter (more candidate source locations both deep and shallow locations)

In the currect code, I have select Opt=1 but If you change to Opt = 2 then you can run only this piece of code by marking and pressing F9 . (this code may take few extra msec to be executed.)

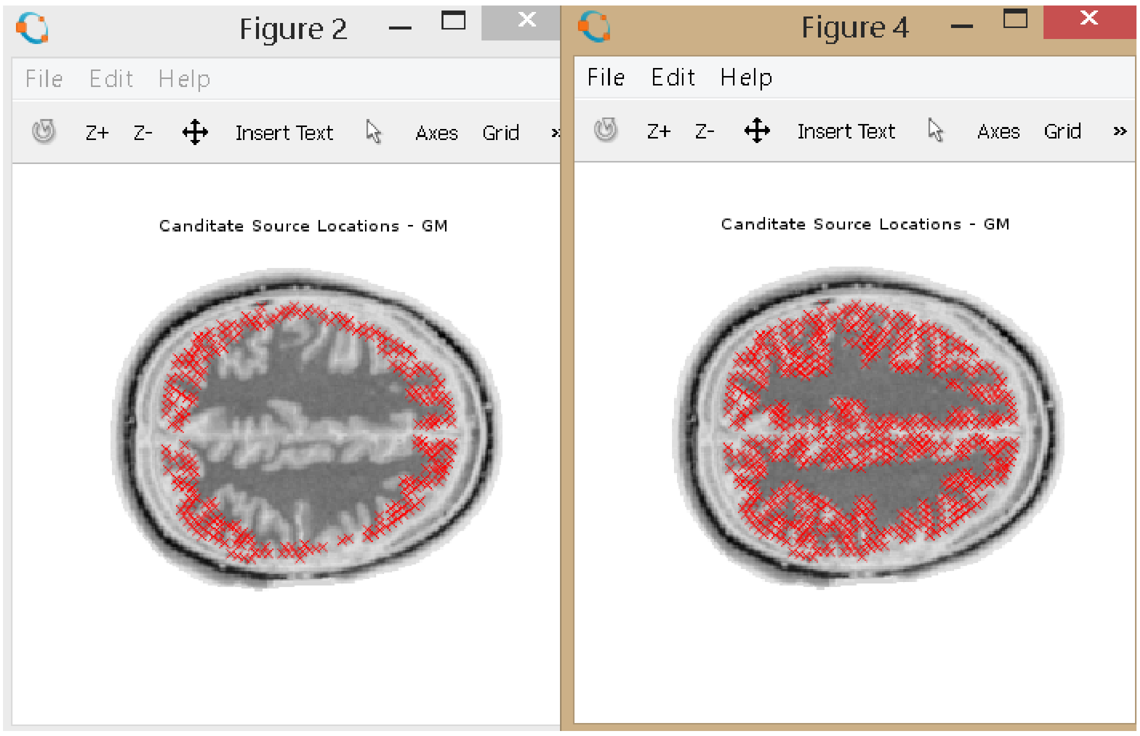


Figure 2
File Edit Help
Z+ Z- Insert Text Axes Grid
Canditate Source Locations - GM
Figure 4
File Edit Help
Z+ Z- Insert Text Axes Grid
Canditate Source Locations - GM

The size of the source space (candidate source locations) affects the size of the lead-field matrix and therefore the inverse solution. Usually based on prior information (e.g. that the activity is only in superficial locations) we try to restrict the source space in order to improve the efficiency of our calculations.

To estimate the source space you can just write in the command window

Length(Geometry.SourceInd)

Step 5: FEM computation and Lead field matrix L

The steps above have fully described the anatomical aspects of our head model. It is now time to compute the electromagnetic model. That is, to describe the mapping that relates activity at the source grid locations to activity at the scalp sensor locations. This mapping can be computed using a Finite element method. Again, there are multiple freely available software packages implementing FEM or BEM. EEGLAB, Fieldtrip, and the MNE software are just three examples. However you need to have Matlab to use them. Moreover, each of these software packages include several algorithms for computing the FEM coefficients.

In this tutorial we will use the following function

```
%STEP 5
%Lead field matrix Estimation
%This function returns the full lead field L matrix which relates every single candidate source location
%with potential values everywhere inside the domain
%also it returns L_el (lead-field matrix of the electrodes) which relates every single candidate source location
%with potential values at the electrodes (This L_el will be used in the inverse problem)
[L,L_el]= fun_LeadFieldMatrix(Geometry);

%Estimate the size of L_el (lead-field matrix for our modeling)
[M,N] = size(L_el) %M-> number of electrodes, N->number of sources
%Is it over-determined or under-determined matrix?
```

Part II: Forward Problem

By now our model should be fully operational so we can start placing artificial EEG sources within the brain volume. An *EEG source* is just one or more source grid locations having a common temporal activation pattern. From a neurophysiological perspective, you can intuitively understand an EEG source as a compact cluster of neurons whose activity is well represented by their average post-synaptic potentials. The time course of such average post-synaptic potentials is what we call the *temporal activation* of the EEG source.

For the second part of the tutorial please open PartII_SourceImagingProblem.m (Make sure that OCTAVE is in the correct directory). For every time instant you can estimate the potential distribution that a dipole source generates.

Dipole simulation (FORWARD PROBLEM)

The simulation of a dipole source is performed by the following function:

```
Loc = 4;  magnitude = 1; theta = 0; %dipole parameters
mi = fun_SingleDipoleSource(Geometry,Loc,theta,magnitude);
```

Plotting the dipole

The plotting of the location and orientation of the dipole is done based on the following piece of code

```
%To plot/draw the location and orientation of the simulated dipole
 PlotDipoleSource
```

Solve the forward problem

```
%%Estimate the potential values for the previous dipole source (Solve the forward problem as following)
%estimate  the potentials on the electrodes around the scalp
u_el = L_el* mi;
```

Each EEG source mi generates a characteristic distribution of scalp potentials. In the following I will loosely refer to such pattern as the *source leadfield*. The *source leadfield* (or potential distribution) is what we would see at the scalp (sensors) if only the source of interest would be active and all other sources would be silent. You can plot the *source leadfield or potential distribution* with the command:

```
%plot potential on the electrodes
%(these are the observation which we will use to solve the inverse problem i.e.
% the reconstruction of the sources based on the potentials on the scalp)
PlotPotElectr
```

TASK 1 (ROTATING DIPOLE)

Open PartII_SourceImagingProblem_Task1.m

Solve the forward problem at t1, t2, t3, t4 time steps where the dipole source changes orientation. Complete the missing part of the following code:

```
%please complete the code to estimate the different dipole vectors mi and the corresponding %potential
values u_el – remember only the orientation of the dipole changes
for i = 1:length(theta_r)
    theta = theta_r(i);
    %Insert the missing code here: estimate the dipole vector mi
     %Insert the missing code here: estimate the potential values at the electrodes
    figure; PlotPotElectr
 end
```

How the pattern of the potential values changes with respect to the dipole orientation?

TASK 2 (DIPOLE Super-position)

Open PartII_SourceImagingProblem_Task2.m

Consider that there is more than one dipole, estimate the potential patterns of each one separately and verify that the superposition principle is valid. To do so, complete the missing parts in the following code:

```
%Consider three dipole sources with the following characteristics
Loc = [4 153 71         %three locations
theta = [0 100 50];   %three Orientations in degrees
magnitude = [1 1 1]; %magnitudes

%plot dipole sources
figure
PlotDipoleSource

%Estimate separately each potential pattern and after that estimate the total potential distribution
u_i = zeros(size(L_el,1),3); %set size of u_i
for i = 1:3
   mi = fun_SingleDipoleSource(Geometry,Loc(i),theta(i),magnitude(i));
   u_i(:,i) = %insert the missing code;
end
u1 = sum(u_i,2); %total potential distribution

%Now estimate the total dipole source and subsequently the pattern that this complex source creates
m = zeros(size(L_el,2),1); %set a zero vector
for i = 1:3
   m = m + fun_SingleDipoleSource(Geometry,Loc(i),theta(i),magnitude(i));
end
u2 = %insert the missing code ; %Estimate potentials

%Is this potential pattern u2 equal to the one estimated as the sum of the individual patterns?
%to do so compare the u1 with the u2 using norm().
er = 1/length(u2) *norm(u2-u1);  %mean square error https://en.wikipedia.org/wiki/Mean_squared_error
```

Part III: Inverse Problem

In this last part of the tutorial we will attempt to solve the *EEG inverse problem*. That is, given a distribution of scalp potentials we will attempt to localize the underlying EEG sources. In fact this is just one part of the inverse solution. The full solution involves also the reconstruction of the temporal activation of the underlying EEG sources. However, we will leave that for a future tutorial where our EEG sources will have time-varying activation patterns.

In this last part of the current tutorial we will try to estimate a single source using potential measurements at the electrodes.

Minimun Norm Estimate (MNE)

Please open script PartIII_SourceImagingProblem.m

The code for the minimum norm is given by

m_hat = inv(L_el'*L_el) *L_el'*u; %this piece of code will have a warning due to the bad conditioning of the matrix in the inv().

Recall from the slides on linear algebra that for a non-square matrix like the leadfield matrix (as the number of columns is not equal to the number of rows) we use the pseudoinverse to solve the previous system. However, due to the fact that some columns of L being very close to linearly dependent, (**L_el'*L_el**) is severely ill-condition and thus we cannot use just a simple inversion.

TASK 3: Please check the conditioning of a matrix (L'*L) using command cond().

Note L_el' is the transpose of L_el -> transpose of L_el is a matrix which has as columns the rows of L_el and as rows the columns of the matrix.

Ill- conditioning of the lead-field matrix L_el!

Instead use the Minimum Norm Estimate:

m_hat = L_el'* inv(L_el*L_el') *u

Alternatively we can use **Tikhonov regularization using λ parameter**

To overcome the ill-conditioning problem and to estimate a solution we use Tikhonov Regularization.

$$\hat{\mathrm{m}} = (\mathbf{L}^{\mathrm{T}}\mathbf{L} + \lambda\mathbf{I})^{-1}\mathbf{L}^{\mathrm{T}}\mathrm{v}$$

TASK 4: Estimate the solution using a selected regularization parameter λ

(e.g. try with lambda = 0.00001 first)

Open PartIII_SourceImagingProblem_tik_Task4.m and write the missing part of the code

```
%Estimate the Tikhonov regularization problem
%solve the problem for different lambda parameters and check the reconstructions
lambda =  %insert a value e.g. 0.0001
I = eye(size(L_el,2)); %Identity matrix
%Write the inverse solution(the transpose of a matrix is written with L_el')
m_hat = %insert the missing code
```

After the estimation of m_hat you can see the estimated dipole locations using the
the following command

```
%plot the true dipole and estimated location with the highest estimated magnitudes (at least 90% of the maximum value)
PlotFrwInvSol
```

To compare the true dipole m with the estimated m_hat write the following code in the command window

```
figure
plot(m,'rx');
hold on
plot(m_hat,'.')
```

What do you observe?

*This minimization gives a solution (m_hat) which is scattered and most of the highest values are close to the superficial source locations.

Explanation: linear system u=Lm has infinite number of solution because it is an under-determined/ill –condition system. In practice, this means that different combinations of dipole sources give exactly the same observation u. From all these combinations, the solver picks the one which gives the lowest dipole energy. Now think a simple example where you have two cases a dipole source which is deep in the domain and another one which is in a superficial layer. The one which is deeper needs to have higher magnitude that the one which is closer to the surface in order to produce the same potential measurements. The solver will pick the dipole which is close to the surface exactly because that one has lower magnitude (or energy).

Please change parameter λ (increase the value of lambda e.g. 0.01), run the script and check the estimated dipole distribution m_hat again.

$$\min_{m} \underbrace{\|v - \mathrm{L}m\|_2^2}_{\text{Fidelity term}} + \lambda \underbrace{\|Wm\|_2^2}_{\text{Regularization term}}$$

As we increase parameter λ we give more trust to the regularization term and not to the fidelity term which includes the observation data. Thus, progressively for large values of λ we observe more scattered solutions, as more dipoles with lower magnitude and closer to the surface locations give lower values to the minimization functional. On the other hand, if we reduce λ a lot then we can see that the condition number of the matrix **(L_el'*L_el+lambda*I)** deteriorates so our result is not robust. If you want you can check for lambda =0.0001 and lambda = 0.000000001 what is the value of the condition number of this matrix using cond(). Always there is a trade-off and careful selection of parameter λ needs to be considered. However this is not a very trivial task.

Can you check in the literature for meaningful ways to estimate **lambda (regularization parameter)?**

Task 5 Additive noise:

Open PartIII_SourceImagingProblem_Task5.m

Using the following command we add some noise in our observations

```
%Add noise and estimate the dipole location
v_n = v+0.01*norm(v)*randn(length(v),1);
```

Solve the inverse problem for different values of λ and observe the estimations.

Before you run the script insert a value for λ!

Task 6 sLoreta:

Check also the so-called Resolution Matrix: R= **= L_el'* inv(L_el*L_el')L_el**

For the MNE to get good estimates, R has to be invertible to recover m since m_hat = R*m (considering very low measurement noise). However this is not the case.

In sLoreta, they proposed to use the diagonal elements of R to recover the source i.e.

m_sLoreta = m_hat./diag(R) = 1./diag(R)*R *m

You can plot the sLoreta result.

What is happening with the mean squared error between the true and estimate, i.e. e=(m_sLoreta-m)_2^2 ? is it the lowest possible to achieve for the MNE optimization?