EEG analysis toolbox

Guido Nolte

June 11, 2007

1 Introduction

This toolbox combines programs MATLAB programs for source analysis of EEG data. The user needs to know only very few of these programs which are explained in this documentation.

It is assumed that the user has some functional data in form of spatial patterns. This could be an averaged potential, a cross-spectrum, or a set of patterns as given e.g. by an ICA analysis, etc. Furthermore, the user needs to know the names of the EEG-channels these patterns refer to.

The programs make forward and inverse calculations and visualize the results on MRI data or surface plots of various brain tissues. Anatomical information is provided by the toolbox using some form of a 'standard head'.

The toolbox contains a demo-program (`demo.m`). Just run this demo to get an impression of the functionalities.

2 Overview

2.1 List of programs

The programs explained here fall into 3 groups which are strictly separated.

The first 'group' is a single program, called `prepare_sourceanalysis`. The input is the list of electrode names, and the output is a complicated structure containing all anatomical information one can possibly need.

The second group contains programs to make any kind of calculation. Right now, there are 4 programs for 1. forward calculations (realistic or 3 spheres), 2. dipole fits of a potential, 3. dipole fits of cross-spectrum, and 4. the RAP-MUSIC scan algorithm.

The third group contains programs for visualization. Right now, there are 4 programs for 1. contour plotting of a potential, 2. contour plots for cross-spectra (and such), 3. surface plots (plus sources), and 4. MRI plots (plus sources).

Here's a list of programs:

Name	Purpose
`prepare_sourceanalysis`	Collect anatomical infos
`forward_general`	makes forward calculations
`dipole_fit_genera`	fits n dipoles to a single pattern
`dipole_fit_cs`	fits n dipoles to a cross-spectrum
`rapmusic`	makes a RAP-MUSIC search
`showfield`	make contour plots of potentials
`showcs`	make 'head-in-head' plots of cross-spectra and such
`showsurface`	make surface plots of tissues (plus sources)
`showmri`	show mri-slices (plus sources)

2.2 General structure of programs

The programs typically have few arguments. Apart from two exceptions, the necessary arguments are always simple numbers, vectors, matrices, tensors or strings. All optional modifications (e.g. the specification of shown MRI-slices) are contained in a single structure, where the fields control the options. The two exceptions are `forward_general` and `showmri` which need complicated but fixed (i.e. pureley anatomical) structures contained in the output of `prepare_sourceanalysis` .

3 Programs

3.1 Preparation

To prepare the source analysis you need a list of the electrode-names you use, e.g. named `clab` with elements `clab{1}='Oz'` ,`clab{2}='C3'` , etc.. Then call
`sa=prepare_sourceanalysis(clab)` ;

The structure `sa` contains all relevant anatomical information for the selected set of electrodes with respect to a fixed head. This head is taken as an example data set from the program CURRY. Right now, there are two models available: this one is called 'curry1', and the second one, called 'montreal' contains the Montreal standard head obtained from an average of many subjects. The curry-head looks nicer but the Montreal scientifically more sound. To choose the Montreal head enter. `sa=prepare_sourceanalysis(clab,'montreal')` ;

The meaning of the fields of the output `sa` is mostly obvious. Perhaps not so obvious is:

1. `sa.fp` is a structure containing all information needed to make a forward calculation in the realistic volume conductor.
2. `sa.fp_sphere` same as `sa.fp` for a 3-shell spherical volume conductor.
3. `sa.locs_2D` contains information about electrode location in a 2D projection in the second and third column, the fourth and fifth column contain slightly shifted coordinates to avoid overlapping sphere in the 'head-in-head-plots'.
4. `sa.mri` is a structure containg all information to show MRI-slices (eventually plus sources).
5. `sa.myinds` appears only if you provided names of electrodes which are not known to the toolbox (it knows 118 names). The `sa.myinds` is the list of channel indices which are known and which will be used. From your patterns you must select these channels by hand, e.g. by
`potential=potential(sa.myinds)` .

3.2 Forward and Inverse calculations

3.2.1 Forward calculation

Program: `forward_general`
This programs makes all forward calculation essentially by calling the right programs. Usage:
`v=forward_general(dips,fp)`

Input:

- `dips` Nx6 matrix. The first 3 numbers of each row denote the dipole location (in cm) and the next three numbers the dipole amplitude.

- `fp` structure containing all (anatomical) information necessary to make the forward calculation, including the name of the program which does the calculation. `fp` is contained in the output (`sa`) of `prepare_sourceanalysis` as `sa.fp` (for realistic volume conductor) or `sa.fp_sphere` (for a spherical volume conductor). Both models assume 3 shells with conductivity ratios 1:.02:1.

Output:

- `v` is an MxN matrix where M is the number of channels, and the i.th column is the potential of the i.th dipole.

E.g., `v=forward_general(randn(1,6),sa.fp);` calculates the potential for a random dipole in a realistic volume conductor.

Reference: In the pre-defined head models, the reference is taken as a channel on the nose, which itself is not included in the list of channels. When you want to use a different reference already in the forward model, you must change `fp` . Let `A` be an NxN matrix which performs this referencing (N is the number of channels), then set `fp.lintrafo=A`(or `sa.fp.lintrafo=A` if you keep `fp` as part of the bigger structure `sa`). This change of `fp` is necessary if you use the forward model for dipole-fitting in a re-referenced system. If you re-reference, be sure that also the patterns are in the new reference.

3.2.2 Fitting a dipole to a single pattern

Program: `dipole_fit_field`
This program fits N dipoles to a given pattern. Usage:
`[dips,res_out,k,field_theo]=dipole_fit_field(v,fp,ndip);`

Input:

- `v` is an Mx1 vector denoting amplitudes in M channels.

- `fp` is the structure for forward caculations.

- `ndip` is the number of dipoles

Output:

- `dips`is an ndipx6 matrix, where each row denotes dipole location and amplitude

- `res_out` is the relative error

- `k` is the number of iterations needed.

- `field_theo` is an Mx1 vector, denoting the theoretical potential for the estimated dipole(s) in M channels.

The fit is done using the Levenberg-Marquardt algorithm. As initial condition ndip random dipoles are chosen. Since the optimization problem is not convex, it is recommended to repeat the fit several times and take the one with the smallest error.

The program is a simple least-squares optimization. To use a weighted least squares with a weight matrix `W` , this can be incorporated into the forward model and into the data, like `v->W*v`and `fp.lintrafo->W*fp.lintrafo`.

3.2.3 Fitting a dipole to a cross-spectrum

Program: `dipole_fit_cs`
This program fits N dipoles to a given cross-spectrum (or covariance matrix). Usage:
`[dips,c_source,res_out,c_theo]=dipole_fit_cs(cs,fp,ndip,xtype);`

Input:

- `cs` is an MxM (in general complex) matrix vector denoting cross-spectral amplitued in M channels.
- `fp` is the structure for forward caculations.
- `ndip` is the number of dipoles
- `xtype` is a string with possible values 'real', 'imag', and 'comp', depending what part of cs should be used.

Output:

- `dips` is an Nx6 matrix, where each row denotes dipole location and amplitude
- `c_source` estimaed cross-spectrum of source
- `res_out` is the relative error
- `c_theo` cross-spectrum of the model

The fit is done using the Levenberg-Marquardt algorithm. As initial condition ndip random dipoles are chosen. Since the optimization problem is not convex, it is recommended to repeat the fit several times and take the one with the smallest error.

The program is a simple least-squares optimization. To use a weighted least squares with a weight matrix `W` , this can be incorporated into the forward model and into the data, like `cs->W*cs*W'`and `fp.lintrafo->W*fp.lintrafo`. (In this form, make sure that you do not apply the weight matrix several times to fp.lintrafo.)

3.2.4 RAP Music

Program: `dipole_fit_cs`
This program makes a RAP-MUSIC scan. Usage:
`[s,vmax,imax,dip_mom,dip_loc]=rapmusic(vv,V,n,grid)`

Input:

- `vv` is an MxK matrix where each column denotes a spatial pattern,
- `V` MxLx3 tensor V(:,i,j) denotes the potential of a unit dipole at i.th grid location in j.th direction. `V`is given in `sa.V_fine` (for 5mm grid) and `sa.V_coarse` for 1cm grid.
- `n` number of dipoles

- `grid` this is optional. grid is an Lx3 matrix denoting the locations for L grid-points. `grid` is given in `sa.grid_fine` and `sa.grid_coarse` . (for fone and coarse grid, respectively)

Output:

- `s` Lxns matrix where the i.th column corresponds to the Musing scan over L points projecting out the field of the proceeding i-1 dipoles.

- `max` nx1 vector where imax(i) denotes the index of grid point for the i.th dipole.
- `vmax` Mxn matrix where the i.th column is the potential of the i.th dipole.
- `dip_mom` nx3 matrix where the i.th row is the moment of the i.th dipole. The i.th dipole is found as that one which maximises s(:,i). Optimization of dipole direction is included in s(:,i).
- `dip_loc` nx3 matrix denoting the locations of the dipoles. It can only be calculated if a grid was given. Otherwise an empty matrix is returned.

3.3 Visualization

3.3.1 Contour maps of potentials

Program: `showfield`
This makes makes 2D contour-plots of potentials and power-distributions. Usage:
`showfield(v,locs_2D[,para])`

Input:

- `v` Mx1 vector, pattern to be shown for M channels.
- `locs_D` Mx5 matrix containing 2D channel locations (in the 2nd and 3rd column). `locs_2D`is provided in `sa.locs_2D` .
- `para` optional parameter to specify display options w.r.t. resolution and colorbar. Type `help showfield` for detials.

3.3.2 Contour maps of cross-spectra

Program: `showcs`
This makes makes 2D contour-plots of cross-spectra (and such). Usage:
`showcs(cs,locs_2D[,para])`

Input:

- `cs` MxM real valued matrix, which can be e.g. the imaginary part of cross-spectrum, M channels.
- `locs_D` Mx5 matrix containing 2D channel locations (in the 2nd and 3rd column). `locs_2D`is provided in `sa.locs_2D` .
- `para` optional parameter to specify display options w.r.t. resolution, colorbar, and position and size of circles. Type `help showcs` for details.

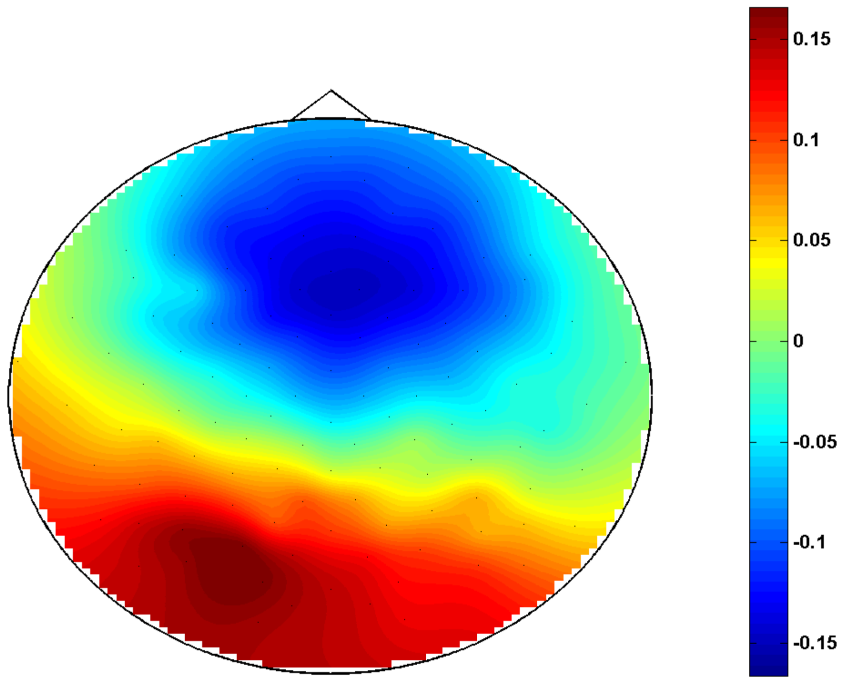

Figure 1: Contour plot of a potential (in default resolution).

3.3.3 Surface plots of cortex, head, etc.

Program: `showsurface`
Makes 3D surface plots plus various kinds of 'sources'. Usage:
`showsurface(mysurface[,para,source1,source2,...])`

Input:

- `mysurface` structure with two fields: mysurface.vc is a list (Nx3 for N surface-nodes) with locations of nodes in cm, and mysurface.tri (Kx3 for K triangles) is a list of location-indices combining to triangles. Surface are provided as `sa.cortex` , `sa.head` ,and `sa.vc{i}` for i=1:3 denoting the shells of the volume conductor.
- `para` optional structure to define viewing options (like viewpoint, dipole-length, etc). Type help showsurface for details. Set `para=[]` if you want to show sources, but use all defaults as viewing options.
- `source1,...` define dipoles as a Kx6 matrix, points as Kx3 matrix, or just values as a Kx1 vector. For Kx6 or Kx6 the first 3 columns are interpreted as locations in cm. For Kx1, the values are interpreted as values on the surface nodes, and K must then be equal N, the number of nodes. An arbitary number of different kinds of sources can added. But there can be only one Kx1 source. Dipoles or points in one source all have the same color, but are different for different sources. For Kx1 the values are shown as color on the surface.

3.3.4 MRI plots (plus sources)

Program: `showmri`
Shows MRI-slices plus various kinds of 'sources'. Usage:

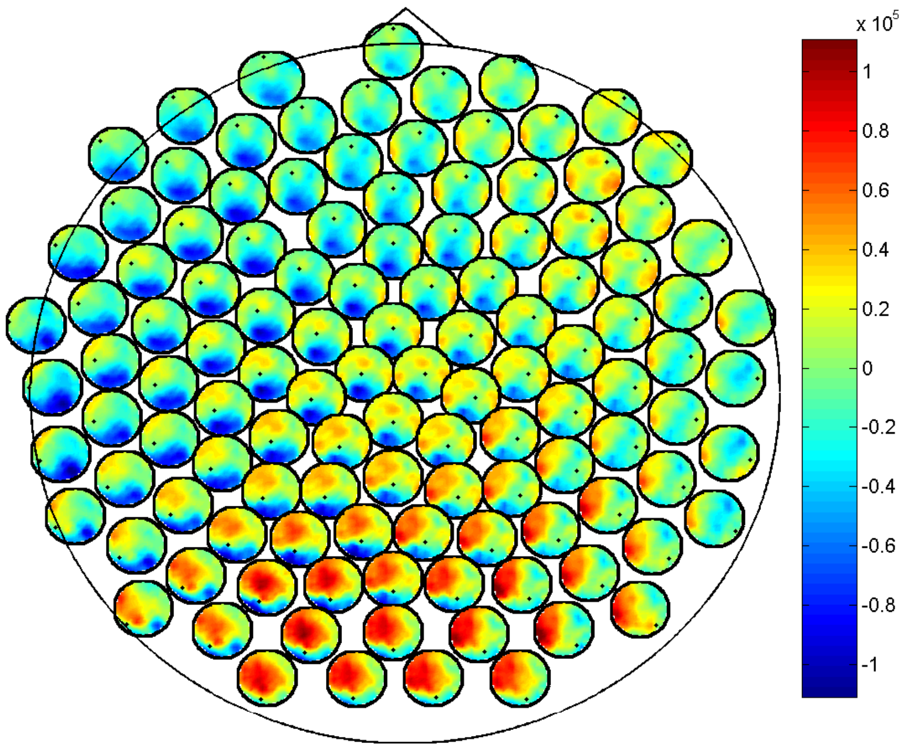

Figure 2: Contour plot of the imaginary part of cross-spectrum at 10Hz. Each small circle shows one row of cs as a contour plot.

`showmri(mri[,para,source1,source2,...])`

Input:

- `mri` structure containing all information about the MRI rawdata, scales, and coordinate transformation to go from head system to MRI system. This structure is provided as `sa.mri`.
- para optional structure to define viewing options (like slice-thickness, number of slices, orientation (axial, sagittal (default), and coronal), dipole-length, etc). Type 'help showmri' for details. Set `para=[]` if you want to show sources, but use all defaults as viewing options.
- `source1,...` define dipoles as an Kx6 matrix, valued points (shown as colored squares) as a Kx4 matrix and points as a Kx3 matrix. x, The first 3 columns are interpreted as locations in cm. An arbitary number of different kinds of sources can added. But there can be only one Kx1 source. Dipoles or points in one source have all the same color, but are different for different sources. For Kx1 the values are shown as color on the surface.

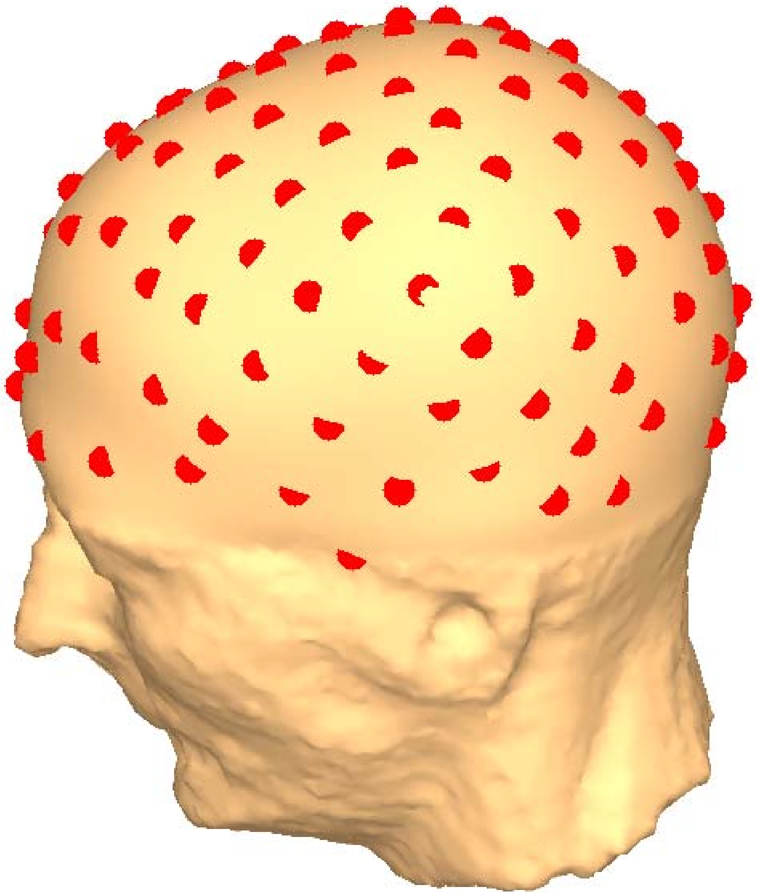

Figure 3: Channel locations plotted on head. The call

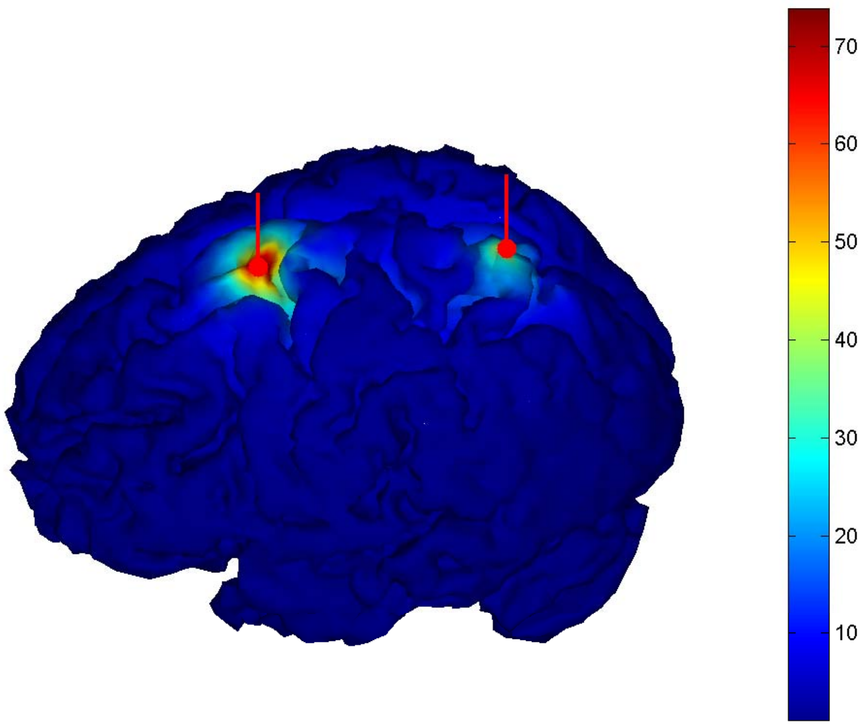

Figure 4: Two simulated dipoles, plus MUSIC-scan on cortex obtained from potential patterns with noise added.

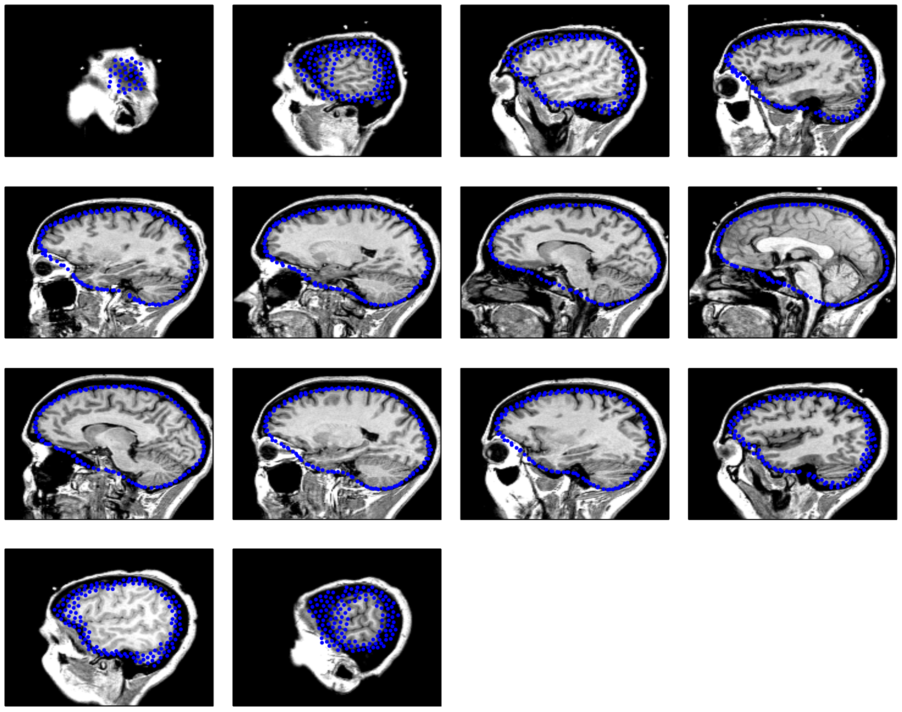

Figure 5: Points on innermost shell of the volume conductor plotted on MRI-slices. The call was showmri(sa.mri,[],sa.vc{1}.vc(:,1:3)); .

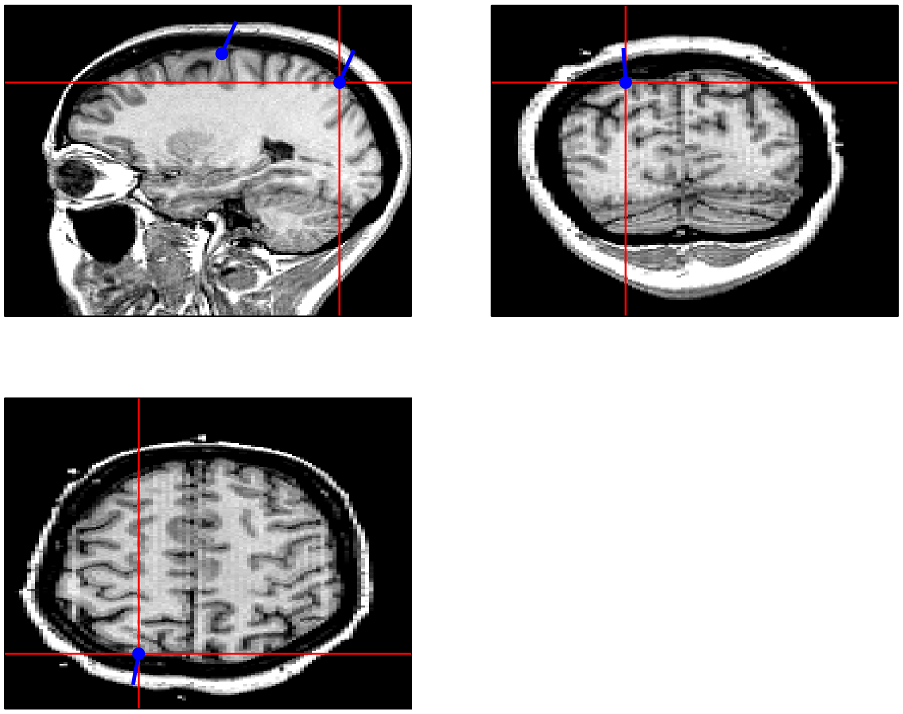

Figure 6: Two dipoles shown in selected slices in all directions. The call was para.orientation='all'; para.mricenter=dips(1,1:3); showmri(sa.mri,para,dips);

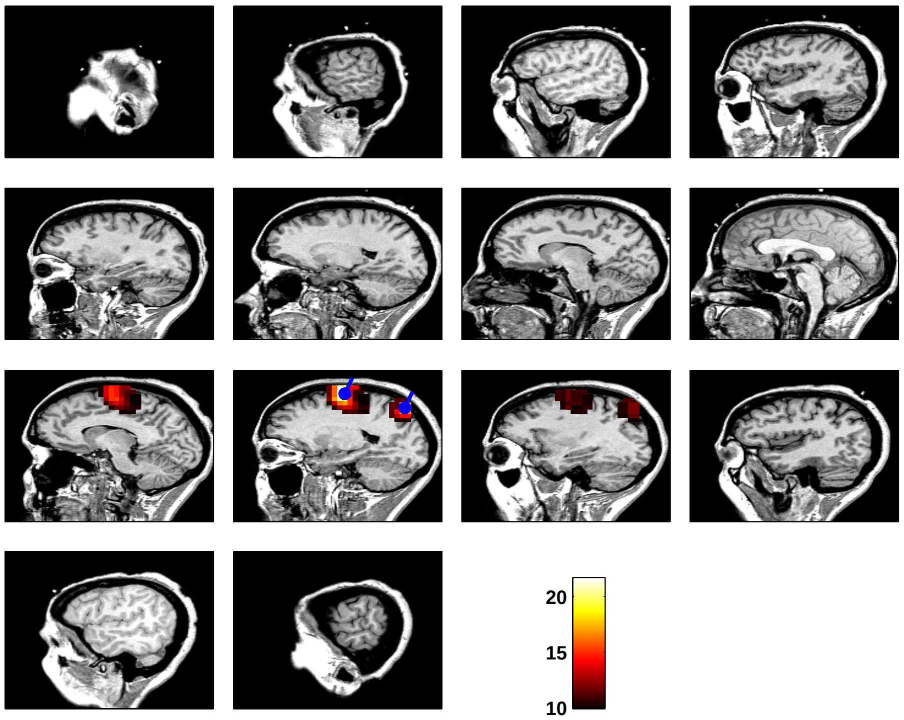

Figure 7: MUSIC scan of potentials of two dipoles plus true dipoles on MRI slices. The call was showmri(sa.mri,[],musicres,dips), where musicres is an Lx4 matrix. First 3 columns are locations and 4th column is equal to $1./(1-s.^2)$ where s is the output of RAPMUSIC for 1 dipole. Only values above 10 are shown.