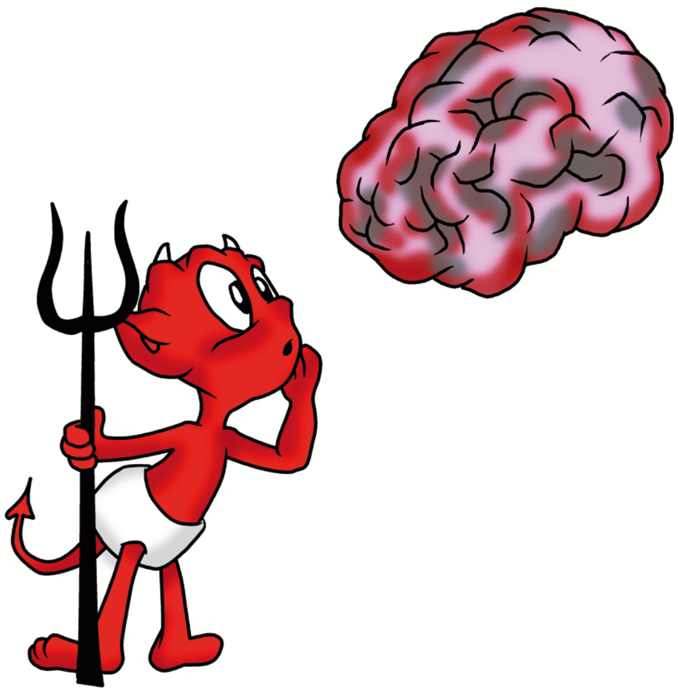
C. Campi[1], A. Pascarella[1], M. Piana[1,2], A. Sorrentino[1]

[1] Dipartimento di Matematica, Università di Genova, Italy
[2] Consiglio Nazionale delle Ricerche - SPIN Genova, Italy



A guide through HADES

The Particle Filter for MEG

Contents

Introduction

During the last years, we have been working at the development, implementation and testing of a Bayesian method, a *particle filter*, for automatic estimation of current dipoles from MEG data. To tell the truth, we did not have the idea personally, but started from the published work of Erkki Somersalo and colleagues [**?**]; nevertheless, during these years we have developed and built so much, both on the theoretical aspects and on the practical usability, that we eventually feel like this is *our* method.

We believe that what makes this method different from other automatic dipole estimation algorithms is its general flavour: not only there are no assumptions on the time correlatedness of the neural sources; in fact, the underlying model is completely dynamical, thus accounting for the non-stationary properties of the neural dynamics. The method is based on the dipole model, but the number of sources is a dynamic variable, free to change across time; even the source locations are not constrained to be fixed during time. The main counterbalance of this generality is that the computational requirements are not cheap, although they are now such that the method can be used in practice.

We are either applied mathematicians or physicists; while we could prove some heuristic results concerning the reliability of our algorithm, we know that only neuroscientists can say something real about the usefulness of this method, and possibly suggest how to improve it. This is why we came up with a Graphical User Interface implementing the particle filter. Since we know many of you will comment "Hell, one more inverse method for MEG", we have just anticipated you and called it HADES: Highly Automated Dipole EStimation.

We have implemented HADES as a black box for those who don't like the mathematical details. Chapter 1 describes how to use the Graphical User Interface and the basic operations to obtain source estimates quickly. Since one must always be careful in using black boxes, and even more in doing things quickly, the Chapter starts with the simplest description of the underlying model, and it ends with a few suggestions for a “safe” use of HADES.

The core of HADES is a multi-dipole particle filter. Roughly speaking, it's a sequential statistical algorithm which solves the inverse problems by trying all possible dipole combinations. It has his theoretical roots in the Bayesian framework, and hence solves the inverse problem of recovering the posterior density for the source configuration; the implementation passes through Monte Carlo approximation because of the non-linear relationship between the unknown parameters and the data. Details about the algorithm can be found in our publications [**? ? ? ?**].

In Chapter 2 we provide a description of the individual Matlab functions implementing the algorithm. These can be useful for those who want to go beyond the possibilities provided by the simple GUI, and use the particle filter for their own purposes.

In case you have questions, comments, suggestions, or anything else, please send an email to hades@dima.unige.it.

Chapter 1

The Graphical User Interface

1.1 Overview: models and methods

Dipolar model. HADES is based on the dipolar model for neural activations: each active area is represented as a single elementary current (*current dipole*), characterized by a location (the active point in the brain) and a dipole moment (direction and strength of the activation). HADES assumes no prior knowledge on the number of active dipoles, which is automatically estimated from the data.

Dynamical model. All the parameters in the model are assumed to be dynamical parameters. The number of dipoles changes with time, which is consistent with the fact that brain areas can activate and deactivate during the time course. Also dipole locations can change with time; while this is not directly related to the underlying physiology, it serves as a powerful tool for separating closeby sources active in differing temporal windows, and also helps to better explore the multi-dimensional source space.

Source space. HADES is based on a discretized source space: dipoles can take only a finite set of pre-defined possible locations. Across time, dipoles are allowed to jump between neighbouring points of the source space, coherently with the dynamical source assumption. On the contrary, dipole moments can vary with continuity, and the forward calculation is performed by a lead field matrix. The source space and the lead field must be provided as an input, while the neighbouring matrix is computed by HADES functions. Optionally, dipole orientations may be constrained to be orthogonal to the cortical surface, in order to improve the localization accuracy.

Sequential estimation procedure. HADES implements a sequential estimation procedure: for each time sample, it estimates the number of dipoles, the dipole locations and dipole moments based only on the previous data. The estimation procedure (the *particle filter*) works by simulating a large number of possible dipole configurations (*particles*) obeying probabilistic laws. The number of particles may be set by the user: the higher this number, the better the inverse solution; in fact, convergence results for the particle filter require in principle an infinite number of particles. At the same time, the computational demand is linearly related to the number of particles.

Discrepancy: the main parameter. The particle filter works by *promoting* and *discarding* dipole configurations based on how likely they are. Such likelihood weighting depends, in principle, on the amount of noise on the data: in highly noisy environments, the model can only explain a fraction of the measured data; *vice versa* in very clean environments a small discrepancy is expected between¡the model and the experiment. Such noise-dependent weighting is automatically implemented in HADES, through the use of either the pre-stimulus standard deviation, or of the full noise covariance matrix. However, in practical situations (for instance, when the noise estimate is biased) it may be useful to require higher/lower discrepancy. The *discrepancy* parameter plays the role of a correction factor with respect to the noise estimate: using a small value usually produces a higher number of estimated dipoles, because these further currents try to explain finer features of the measured field; on the contrary, high values lead to a reduced number of estimated dipoles, and possibly to slight localization bias towards deeper regions.

Time-varying dipole estimates. The output of the particle filter is a sequence of estimated dipoles. HADES does not implement yet dynamic visualization functions (movies): to observe the dipole dynamics you may export the results to MNE (see later in this chapter). Estimated dipoles can be visualized in HADES by selecting a specific time window. Remarkably, at this stage there is no direct relationship between dipoles estimated from different time samples, even if they represent the same neural source, because the particle filter does not keep track of the "source identity": hence, the activity of each neural source is in fact reconstructed as a sequence of independent dipoles, possibly with (slightly) different locations.

Clustering: getting the global picture. In order to bind estimated dipoles which represent the same neural source at different time samples, HADES implements a clustering procedure. Clustering can be performed in two different configurations: a 3-dimensional clustering, where dipoles are grouped based only on their location; a 5-dimensional clustering, where dipoles are grouped based both on location and orientation. The final number of clusters is estimated automatically with a recursive procedure, which starts from the user-defined (maximum) number of clusters, and decreases this number until all the estimated clusters are significantly different. Optionally, a final step can be applied where each cluster which is active in non-overlapping time windows is split, so that the spatio-temporal features of the neural activity are eventually decomposed into continuous elemetary sources.

Estimated dipoles and estimated sources. In the rest of this guide book, we will call *estimated dipoles* the immediate output of the particle filter, before the clustering procedure is applied; we will call *estimated sources* the output of the clustering procedure. Estimated sources have a source waveform, while estimated dipoles live only for one time sample. Estimated sources are groups of estimated dipoles, and we may use the average dipole as the representative element of the whole group.

External links. HADES results may be exported for visualization purposes onto MNE
(http://www.nmr.mgh.harvard.edu/martinos/userInfo/data/MNE_register/index.php)
and Freesurfer
(http://surfer.nmr.mgh.harvard.edu/)
inflated brains.

1.2 The GUI

Installation and start. HADES needs not to be installed: just copy all the files in a unique folder, and add that folder to the MATLAB path (File→Set Path). To start the HADES GUI, run *hades_main* in the Matlab Command Window: the main window will appear.

1.2.1 The Main window

The main window is divided in three panels, plus a bottom row of buttons: in the following we provide a description of the functionalities associated with each panel. Most of the operations are concerned with providing input data: for details about the format of the input files, see section 1.2.3 later on.

Panel *Input files*:

- *subject directory*: opens a user interface for selecting the working folder; this folder will be used as the starting path for all subsequent operations;

- *load lead field*: opens the panel for loading the lead field and the source space; in this panel there are two options: (1) loading a *.fif* file, which contains both the source space and the lead field; (2) loading separately the lead field, the source space and (optionally, if you want the orientation constraint) the orientation file as *.dat* files;

- *load neighbours*: opens a user interface for selecting a *.dat* file containing the matrix of neighbours. If the matrix was not calculated yet, the user may set the radius under which two points are considered as neighbours:

we suggest to use a value close to *2d*, where *d* is the average spacing of the source space. Too small values of the radius would constrain the dipoles in non-communicating regions: the user is alerted whenever such situation occurs. The computation will be performed before the particle filter analysis and the matrix of neighbours will be automatically saved. For more details see the description of the function *hades_compute_neighbours.m* in chapter 2;

- *load ssp*: opens a user interface for selecting a *.dat* file containing the Signal Space Projection matrix;
- *load noise covariance*: opens a user interface for selecting a *.dat* file containing an estimated noise covariance matrix;
- *load data*: opens a panel for loading the data to be analyzed; here, the data file should be loaded first, and it can be either a *.fif* file or a *.dat* ASCII file; the button *show* allows to have a look at the data (superposition of all sensor measurements); in the case a *.dat* file was loaded, one has to set by hand the initial time and the sampling frequency; for all type of input files, the pre-stimulus interval must be set, in order to evaluate the pre-stimulus standard deviation; here, also the time interval to be analyzed by HADES must be set.

Panel *PF Parameters*:

- *number of particles*: this parameter sets how many sample points to use in the particle filter. The larger this number, the better the solution. The computational time increases linearly with this number; our recommendation is to use at least 5,000 particles;
- *discrepancy*: this parameter tunes the sensitivity of the estimation procedure to the input data. By default, this parameter is set to 1, which implies that the sensitivity is determined by the estimated amount of noise on the data. The discrepancy parameter is in fact a correction factor with respect to the noise estimate: values higher than 1 correspond to a higher discrepancy (true noise stronger than estimated), and *vice versa* for values lower than 1.

Panel *Output file*:

- *results directory*: the folder where the results are saved; it is automatically set to be the same folder where the data are, if not set otherwise;
- *subject/experiment*: the name associated to the data; it is used to auto-save the results of the particle filter;
- *auto-save*: if this check box is selected, the output results of the particle filter will be automatically saved in a *.mat* file with name as in *subject/experiment*.

The bottom row:

- *save settings*: saves all the input provided so far, comprising both elements in the *Input files* panel and in the *PF parameters* panel, in a *.mat* file; a message box asks whether the user prefers to save all the data in this file (comprising leadfield, source space and measurements), or keep track of the links to the input files; the first option may require large disk space;

- *load settings*: loads a *.mat* file previously saved with *save settings*;

- *run PF*: starts the particle filter algorithm; in few seconds (depending on the CPU and available RAM) a wait bar will appear;

- *load results*: loads previously saved results, obtained by the particle filter. If a lead field has been loaded already, the *Results* window will appear; please be sure to have the right lead field in memory before loading the results. Otherwise, if no lead field has been loaded yet, the panel *Results* will be opened only after the *Loading the lead field* panel.

1.2.2 The Results window

The *Results* window pops up automatically either when the particle filter has finished or when previously obtained results have been loaded from the *Main Window*. The *Results* window contains three panels.

Cluster the results. The button in this panel enables the clustering controls on the right:

- *clustering parameters*: selects whether to perform the clustering based only on the dipole locations, or both on dipole locations and orientations; in general, the orientation variables may allow to separate spatially close sources;

- *number of clusters*: selects how many clusters to start from; the clustering algorithm is a recursive one, which at each step checks how many significantly different clusters exist, and decreases the total number of clusters if not all clusters are significantly different from each other;

- *compute clusters*: starts the clustering algorithm;

- *save*: saves the results of the clustering algorithm; since the clustering algorithm contains stochastic initialization, running the algorithm twice does not necessarily produce the same results.

Show the results. The *show the results* button enables the visualization controls on the right, described next; estimated dipoles and sources are plotted together with the source space, so as to allow a rough understanding of their locations.

- *estimated dipoles*: with this selected, the second row of controls allows to choose a time interval, contained in the analyzed interval; pressing *show selected time points*, a new figure appears, showing all the estimated dipoles in the selected time interval;
- *clustered dipoles*: with this selected, the second row of controls allows to choose which clusters to show; pressing the button *Clustered dipoles* a new figure appears showing the selected clusters;
- *mean dipoles*: with this selected, the second row of controls allows to choose which mean dipoles to show; pressing the button *Mean dipoles* a new figure appears showing the selected mean dipoles;
- *time split*: with this checkbox active, each individual source characterized by activity in N non-overlapping time windows is split into N separate sources.

The *show the model selection* button visualizes a figure showing the probability at each time point of staying in the zero-dipole space, one-dipole space and so on. The number of estimated dipoles at each time point is computed from here, choosing the space with higher probability.

Export the results. The last two buttons write the particle filter results in file with suitable formats for the visualization in MNE and Freesurfer. The *write .stc files* button writes an *.stc* file containing a first "time sample" showing the superposition of all dipoles in the selected time interval, and then the dynamical dipole reconstructions in the same interval; if no interval was selected, the whole time span is used. Saved files are named <subject>_dipoles-lh.stc and <subject>_dipoles-rh.stc, for the left and right hemisphere, respectively. A .fif file containing the lead field is required.
The *write .w files* button write in the results directory a *subject/experiment_dipoles.w* containing the estimated dipoles. A .fif file containing the source space of the subject is required.

1.2.3 File types and data formats

Input/ouput operations in HADES may use one of the following file formats: Matlab *.mat*, Neuromag *.fif*, MNE *.stc*, Freesurfer *.w*, and plain ASCII *.dat*. Please consider that different Matlab version may write non-compatible *.mat* files, hence be careful when loading results saved on a different computer.
The *.dat* input files must be stored in the ASCII format and have specific dimensions. Let us call NVERT the number of points (vertices) in the source space, NSENS the number of MEG sensors and NTIME the number of sampled time points. Here is a list of the input data with associated format:

- lead field matrix: NSENS×NVERT*DIM, with DIM=1 values 1 if the dipole orientations are constrained, 3 otherwise.

- source space: NVERT×3;
- orientation: NVERT×3;
- sensors: NSENS×3;
- versors: NSENS×3;
- ssp: NSENS×NSENS;
- noise covariance: NSENS×NSENS;
- data: NTIME×NSENS.

The MNE Matlab toolbox (http://www.nmr.mgh.harvard.edu/martinos/userInfo/data/MNE_register/index.php) is required for loading *.fif* files and exporting *.stc* or *.w* files.

1.3 On the Safe Use - Final Remarks

A stochastic algorithm. The first remarkable observation concerns the fact that HADES is based on a stochastic algorithm. The whole estimation procedure in the particle filter relies on stochastic simulation of dipole configurations; furthermore, also the final clustering step has a random initialization. As a consequence of these facts, running the algorithm twice, with the very same parameters, on the very same data, does not necessarily produce exactly the same results. Obtaining the same estimated dipoles with different runs is therefore a first assessment of reliability of the reconstructions themselves: we strongly suggest to perform multiple runs in order to check this stability.

The input parameters. The dipole estimation procedure is strongly influenced by the two input parameters, according to the following rules:

- the number of particles determines the accuracy of the reconstruction and is linearly related to the computational time: our recommendation is to keep this number as high as possible for a reliable estimation procedure; a reasonable range is between 5,000 and 100,000;
- the *discrepancy* parameter determines which portion of the measurements may be left "unexplained". Tuning this parameter has then two main consequences: (1) dipole estimates should become increasingly accurate for decreasing values of this parameter; (2) the estimated number of dipoles shall increase for decreasing values of this parameter, because more dipoles are needed to explain finer details of the measurements. Hence, large values will produce rough estimates of the main activations, while small values will produce more complicated (and less stable) source configurations. For those who have knowledge of regularization theory, this parameter clearly plays the role of the regularization parameter, though in a non-linear setting. Due to the higher requirements for exploring spaces with

larger number of dipoles, also this parameter has some influence on the computational time of the algorithm.

Dipole model and ill-posedeness. As a last comment, it may be worth to recall two inherent limitations of the approach HADES is based on. The first limitation is concerned with the dipole model of neural activations: while widely used and certainly valid in a large number of experimental conditions, such model is still simplistic and may not be able to cope with experimental data produced by complicated, extended source configurations. The second limitation is concerned with the mathematical nature of the source estimation problem, which is a so-called ill-posed problem: it is well known that, in principle, it is not possible to recover uniquely the neural currents from their magnetic fields; inverse methods aim at producing reasonable estimates of these currents, but these estimates shall never be considered as the ground truth.

Chapter 2

Hades Matlab functions

This chapter contains a summary of the HADES Matlab scripts and functions. For each function/script, the calling function(s) are indicated in brackets.

hades_check.m (callback in hades_main.m): checks, before running the *particle filter*, if all the input files and the parameters are set.

hades_compute_neighbours.m (callback in hades_check.m): computes the neighbours for the points of the source space. It can be employed for precomputing the neighbours matrix, running in the Matlab Command Window the following command:

$$[neighbours] = hades\_compute\_neighbours(radius, vertices, dir, subject)$$

where

- neighbours output neighbours matrix;
- radius radius [cm] within compute the neighbours;
- vertices the coordinates of the points of the source space, stored in a NVERT×3 matrix, NVERT number of points of the source space;
- dir the directory where save the matrix
- subject name of the subject.

The output neighbours matrix is automatically saved in a *.dat* file with name <subject>_neigbours_<radius>.dat.

hades_load_fwd_dat.m (callback in hades_main.m): loads the lead field and the source space when, in the panel *Loading the lead field* (Fig. **??**) a *.dat* file is selected.

hades_load_fwd_fif.m (callback in hades_main.m): loads the lead field and the source space when in the panel *Loading the lead field* (Fig. **??**) a *.fif* file is selected.

hades_load_parameters.m (callback in hades_main.m): loads a *.mat* file containing settings previously saved.

hades_main.m: starts the HADES GUI and manages the Main window and the callback to the other functions.

hades_pf.m (callback in hades_main.m): runs the particle filter algorithm.

hades_plot_clusters.m (callback in hades_visualization.m): plots the clustered dipoles selected in the Results Window.

hades_plot_seeds.m (callback in hades_visualization.m): plots the mean dipoles selected in the Results window.

hades_script_cluster.m (callback in hades_visualization.m): clusters the estimated dipoles.

hades_statistical_test_cluster.m (callback in hades_visualization.m): performs a statistical test on the clusters computed in hades_script_cluster.

hades_visualization.m (callback in hades_pf.m and in hedes_main.pf): opens the Results window and manages the functions for visualizing the results.

hades_write_stc.m (callback in hades_visualization.m): write the particle filter results in .stc files.

hades_write_w.m (callback in hades_visualization.m): write the estimated dipoles in .w files.

hades_load_data

The data structure. In order to use the particle filter script (*hades_pf.m*) out of the GUI, a struct must be created containing the following fields:

- *pf.subjects_dir* **string** containing the path of the subjects directory
- *pf.results_dir* **string** containing the path of the result directory
- *pf.subject* **string** containing the name of the experiment
- *pf.g_matrix* lead field matrix
- *pf.g_matrix_dir* **string** containing the lead field matrix directory
- *pf.vertices* vertices matrix

- *pf.vertices_dir* **string** containing the vertices matrix directory
- *pf.orient* orientations matrix (**optional**)
- *pf.orient_dir* **string** containing the orientations matrix directory
- *pf.evol* evolution matrix
- *pf.evol_dir* **string** containing the evolution matrix directory
- *pf.ssp* ssp matrix (if not avalaible, set a identity matrix with size NSENS)
- *pf.ssp_dir* **string** containing the ssp matrix directory
- *pf.cov_matrix* noise covariance matrix (if not avalaible, set a identity matrix with size NSENS)
- *pf.cov_matrix_dir* **string** containing the noise covariance matrix directory
- *pf.sigma* mean variance of noise during the pre-stimulus
- *pf.sigma_par* discrepancy parameter
- *pf.np* number of particles
- *pf.zero_time* first time point [ms] of the data
- *pf.final_time* final time point [ms] of the data
- *pf.freq* frequency [Hz]
- *pf.time_interval* = *pf.zero_time*/1000:1/*pf.freq*pf.final_time/1000;
- *pf.t1* the first time point of the analysis in time samples
- *pf.t2* the last time point of the analysis in time samples
- *pf.data* data matrix
- *pf.data_dir* **string** containing the data matrix directory
- *pf.autosave* for saving the result set to 1, otherwise 0

The string fields may be left empty, while the other fields must be filled with appropriate data, except where otherwise explicitely stated.

Bibliography