HAL
open science

OpenMEEG: opensource software for quasistatic bioelectromagnetics

Alexandre Gramfort, Théodore Papadopoulo, Emmanuel Olivi, Maureen Clerc

▶ To cite this version:

Alexandre Gramfort, Théodore Papadopoulo, Emmanuel Olivi, Maureen Clerc. OpenMEEG: opensource software for quasistatic bioelectromagnetics. [Research Report] 2010. inria-00467061v1

HAL Id: inria-00467061
https://inria.hal.science/inria-00467061v1

Submitted on 25 Mar 2010 (v1), last revised 27 May 2010 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

OpenMEEG: opensource software for quasistatic bioelectromagnetics

Alexandre Gramfort, Théodore Papadopoulo, Emmanuel Olivi, Maureen Clerc

***Abstract*—Many devices used in the clinical or the cognitive science domain perform electromagnetic measurements, or stimulation, on the human body. To interpret and control bioelectromagnetic phenomena involved with these devices, realistic physiological modeling is required, and accurate numerical solutions of the governing equations must be computed. We present OpenMEEG, which solves the electromagnetic forward problem in the quasistatic regime, for models with piecewise constant conductivity. The core of OpenMEEG resides in the symmetric Boundary Element Method, based on an extended Green Representation theorem. The accuracy of several BEM solutions has been compared for forward electroencephalography, showing that OpenMEEG outperforms the other BEMs. OpenMEEG is an opensource, multiplatform C++ package, with interfaces in Python and Matlab.**

***Index Terms*—Boundary Element Method, Electromagnetics, quasistatic regime, Electroencephalography, Magnetoencephalography, Forward modeling, opensource software**

INTRODUCTION

Many devices used in the clinical or the cognitive science domain perform electromagnetic measurements, or stimulation, on the human body. Devices measuring electric fields include electro-encephalograms (EEG), -cardiograms (ECG), -myograms (EMG), while magneto-encephalograms (MEG) measure magnetic fields. Among stimulating devices, transcranial magnetic stimulation (TMS) uses magnetic coils to stimulate brain regions, while functional electric stimulation (FES) and electrical impedance tomography (EIT) impose an electric current or potential through contact electrodes.

To interpret and control the bioelectromagnetic phenomena involved with these devices, realistic physiological modeling is required, in terms of geometry and conductivity [26]. Accurate numerical solutions of the governing equations must be computed: obtaining the best accuracy possible for a given numerical model is one of the goals of OpenMEEG.

Electromagnetic propagation is governed by the Maxwell equations, coupling the electrical and magnetic fields. This coupling simplifies when the relevant frequencies are low enough for the quasistatic regime to hold. The electric potential is then governed by the law of electrostatics

$$\nabla \cdot (\sigma \nabla V) = \nabla \cdot \mathbf{J}^{\mathrm{p}} \ , \tag{1}$$

where σ is the conductivity field, and $\mathbf{J}^{\mathrm{p}}$ is a dipolar source distribution within the domain, representing average postsynaptic currents within pyramidal cortical neurons. A boundary condition must be imposed, typically controlling the normal current on the domain boundary:

$$\sigma \nabla V \cdot \mathbf{n} = j \ . \tag{2}$$

The electric potential can be computed independently from the magnetic field, by solving (1) with boundary condition (2). The magnetic field $\mathbf{B}$ depends both on the electric potential V and on the current source distribution $\mathbf{J}^{\mathrm{p}}$, through the Biot and Savart law:

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \int (\mathbf{J}^{\mathrm{p}}(\mathbf{r}') - \sigma \nabla V(\mathbf{r}')) \times \frac{\mathbf{r} - \mathbf{r}'}{\|\mathbf{r} - \mathbf{r}'\|^3} \, d\mathbf{r}' \ , \tag{3}$$

written here in the case where $j = 0$ on the boundary.
A *forward problem* consists in simulating V and/or $\mathbf{B}$ when σ, $\mathbf{J}^{\mathrm{p}}$, and boundary current j are known. The forward electromagnetostatics problem is well-posed, in contrast to the far more difficult, ill-posed *inverse problem* of estimating σ, or $\mathbf{J}^{\mathrm{p}}$, from partial boundary data. Still, obtaining an accurate solution for the forward problem is far from trivial. This paper presents a software package, OpenMEEG, that makes available to the community recent research efforts that improve the accuracy of forward solvers.

Forward solutions provide the relationship between the quantities of interest and the measurement. To obtain a good description of this relationship, one must model the sources, the conductivity, and the sensors.

The choice of conductivity model is especially delicate because it strongly constrains the numerical solutions that can be used to solve the problem. In all generality, the conductivity field σ should be modeled as a tensor field, because composite tissues such as bone and fibrous tissues have an effective conductivity that is anisotropic. Realistic, anisotropic conductivity models can however be difficult to calibrate; simpler, semi-realistic, conductivity models assign a different constant conductivity to each tissue, as depicted in Figure 1. There are three main types of conductivity models, and associated numerical methods:

1) if the conductivity field can be described using simple geometries (with axilinear, planar, cylindrical, spherical or ellipsoidal symmetry), *analytical methods* can be derived, and fast algorithms have been proposed that converge to the analytical solutions for EEG [8, 16] and MEG [25];
2) for piecewise constant conductivity fields as in Figure 1, *Boundary Element Methods* (BEMs) can be applied, resulting in a simplified description of the geometry only on the boundaries;
3) general non-homogeneous and anisotropic conductivity fields are handled by *volumic methods*; Finite Element Methods and Finite Difference Methods belong to this category.

This paper deals with Boundary Element Methods, whose

advantage over volumic methods is to use an economic representation of the conductivity field (a few conductivity parameters - one per tissue, and a few triangular meshes to represent the interfaces). Until recently, all Boundary Element Methods used in bio-electromagnetics were based on a Green representation theorem, involving operators called single- and double-layer potentials. In OpenMEEG the approach is to consider an extended version of the Green representation theorem, involving more operators, and leading to a new BEM formulation, called the Symmetric Boundary Element Method.

The structure of the paper is as follows: the OpenMEEG software is presented in Section I. To motivate the use of OpenMEEG, Section II presents a comparison study between OpenMEEG and four other solvers in the context of EEG forward modeling. The accuracy of the solvers is tested on multiple random sphere models. The accuracy of OpenMEEG is also assessed by numerical experiments for MEG forward modeling. Finally Section III describes briefly how to use OpenMEEG. The mathematical derivation of the symmetric BEM is explained in Appendix A, while Appendix B presents OpenMEEG usage via a command line interface, via Python or via the Fieldtrip Matlab toolbox.

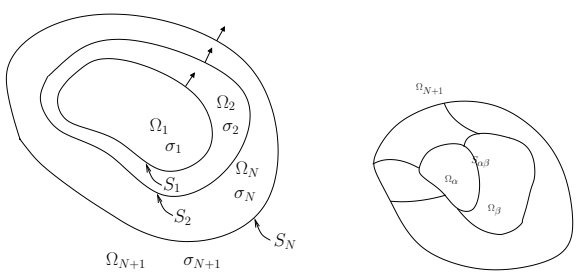


Fig. 1. Boundary Elements are well-suited for piecewise constant isotropic conductivity models. OpenMEEG currently handles nested regions (left), but could in principle be extended to more general, disjoint regions (right) as presented in [18].

I. What is OpenMEEG?

A. Uses of OpenMEEG

The main purpose of OpenMEEG is to solve the forward problems arising in magneto- and electro-encephalography. A geometrical model of nested meshes representing tissue interfaces must be provided to OpenMEEG, which does not perform any segmentation or meshing. The main output of a forward problem is a lead field, i.e., the linear application relating the intensity of a source at a specific location, and the sensor measurements. Although the principal target of OpenMEEG is magneto- and electro-encephalography, other types of bioelectromagnetic problems have also been handled with OpenMEEG: we hereforth describe its current scope.

a) Electroencephalography (EEG): EEG is concerned with the variations of electric potential on the scalp, due to sources within the brain. At frequencies of interest, the quasistatic regime is valid, resulting in the electrostatics relation (1). The air surrounding the scalp is supposed non-conductive, hence the normal current vanishes on the scalp: $j = 0$ in boundary condition (2). The sources within the brain are represented by dipoles: in equation (1), the sources are $\mathbf{J}^{\mathrm{p}} = \vec{q}\delta_p$ where p is a dipole position and $\vec{q}$ the associated dipolar moment. OpenMEEG computes the electric potential and the normal current on each interface between two homogeneous tissues, due to electric sources within the brain. EEG sensors are electrodes, modeled in OpenMEEG as discrete positions on the scalp at which the potential can be measured (infinite impedance assumption). On these sensors, OpenMEEG computes the EEG lead field, which is the linear relationship between source amplitude (for fixed position and orientation) and sensor values.

b) Magnetoencephalography (MEG): The origin of MEG is identical to that of EEG, and MEG sources within the brain are also modeled as dipoles [13], MEG is concerned with magnetic fields rather than electric fields, but OpenMEEG makes use of the Biot and Savart relation (3) to compute the magnetic field, and hence requires the electric potential to be computed beforehand on all the interfaces [9]. Magnetometers or gradiometers can be modeled in OpenMEEG. Magnetometers are defined by their position and the direction of the field that they measure. Gradiometers and more general sensors are handled by providing to OpenMEEG the positions, the orientations and the weights of integration points. For example with axial gradiometers present in CTF MEG systems, the forward field for a sensor is obtained by subtracting the two leadfields computed at the locations of two nearby magnetometers.

c) Electrical Impedance Tomography (EIT): Electrical Impedance Tomography infers characteristics of a conductive domain, by analyzing the potential resulting from the application of a current on the boundary. This method has been applied to calibrate the conductivity of EEG head models [11, 4, 3]. OpenMEEG computes the forward problem associated to EIT: given a conductive domain Ω defined by the interfaces between homogeneous regions, and their conductivity, and for a prescribed normal current j, OpenMEEG computes the potential V and the normal current $\sigma\partial_{\mathbf{n}}V$ on each interface by solving (1) for $\mathbf{J}^{\mathrm{p}} = 0$, and with boundary condition (2). Similarly to EEG, the potential is interpolated onto the electrode positions.

d) Intracranial electric potentials: In some clinical settings, electric potentials are measured within the brain, either on the surface (ElectroCorticography) or within the gray matter (intracranial EEG, or stereotaxic EEG). OpenMEEG is able to compute lead fields for such intracranial electrodes, in realistic head models. Since OpenMEEG computes the potential and the normal current on all surfaces of the model, the application of a Green representation theorem is necessary to compute the potential at any prescribed position within the brain volume.

e) Functional Electrical Stimulation (FES): Functional Electrical Stimulation (FES) provides a way to restore movement of paralyzed body regions by activating the efferent somatic axons. For this a current is applied to a nerve sheath, using specially designed electrodes. The precise location and time course of the applied current must be optimized in order to achieve the best selectivity, and to minimize the current intensity for a desired outcome. Optimizing the stimulation parameter in a realistic nerve model requires a forward model

for FES. OpenMEEG provides such a tool, by combination of the concepts of EIT (Section I-A0c) and of internal potential simulation (Section I-A0d) [15, 19].

f) Cortical Mapping: The last application of OpenMEEG presented here is an *inverse problem*. Cortical Mapping aims to recover the potential and the normal current occurring on the surface of the cortex (i.e., under the skull bone), given EEG measurements on electrodes [14]. A particularly elegant solution to this problem has been proposed with the symmetric BEM [6], making it possible e.g. to solve further source localization problems. Cortical Mapping using OpenMEEG has been implemented within the FindSources3D library[1].

B. Implementation

OpenMEEG uses a Galerkin Boundary Element formulation, with the specificity of jointly considering the electric potential and the normal current on each interface as unknowns of the problem. A P1 (piecewise linear) approximation is used for the electric potential, whereas the normal current $\sigma\frac{\partial V}{\partial \mathbf{n}}$ is approximated with P0 (piecewise constant) boundary elements. The most intricate part of the implementation concerns the assembly of the system matrix, requiring singular kernel integration over triangles. The inner integrals are computed with analytical schemes [7], whereas the outer integrals are computed with a 7-point Gauss quadrature scheme [2]. An adaptive integration scheme recursively subdivides the triangles until the required precision is achieved. This adaptive integration has an influence on the accuracy, as demonstrated in Section II-B.

Since the electric potential can only be computed up to a constant, the system matrix is deflated to make it invertible. In practice, it consists in imposing the constraint that the integral of the potential over the external layer is 0.

The magnetic field is computed by using the Biot and Savart equation with the Galerkin piecewise linear approximation for the potential [9].

Given a set of dipole positions and orientations, a set of sensors and a head model defining homogeneous conductivity regions, the M/EEG forward problem produces as output the *lead field*, i.e., the matrix that relates the dipole amplitudes and the sensor data. OpenMEEG computes the lead field with the following procedure:

- assemble the system matrix involving boundary integral operators on the discretized surfaces;
- for a specified set of dipole positions and orientations, assemble a discretized, source-related term;
- solve the linear system relating the two above matrices (providing V and $\sigma\frac{\partial V}{\partial \mathbf{n}}$ on each interface)
- interpolate the scalp potential (for EEG) or apply the discretized Biot and Savart relation (for MEG).

See Appendix A for more mathematical details and Appendix B for practical considerations.

C. Software

Licence - OpenMEEG is distributed under the French opensource license CeCILL-B. It is intended to give users the freedom to modify and redistribute the software. It is therefore compatible with popular opensource licenses such as the GPL and BSD licenses. The CeCILL-B license imposes to anybody distributing a software incorporating OpenMEEG the obligation to give credits (by citing the appropriate publications), in order for all contributions to be properly identified and acknowledged[2].

Source code - OpenMEEG is implemented in C/C++ with limited external dependencies. It uses the Intel MKL libraries on Windows and ATLAS (BLAS/LAPACK) on Unix systems for fast and accurate linear algebra. A dependency with the MATIO library[3] is optional for Matlab compatible IOs. The source code of OpenMEEG is hosted on the INRIA GForge platform and is accessible to an anonymous user via the Subversion[4] version control system. OpenMEEG binaries and source code are both available from http://openmeeg.gforge.inria.fr.

Multiplatform - OpenMEEG is available as precompiled binaries for GNU-Linux systems, Mac OS and Windows. OpenMEEG's build and packaging system is based on *CMake/CPack*[5] allowing easy development and deployment on all architectures.

Parallel processing - Compilation of OpenMEEG can be done using advanced features provided by modern compilers. OpenMP is a technology that enables parallel computation at a limited cost in terms of software design. When OpenMEEG is compiled using OpenMP, the numerical integration, on which most of the computation time is spent, can be run in parallel. On a machine with 8 CPUs a standard EEG leadfield is computed up to 6 times faster.

Documentation - Source code documentation is generated via doxygen[6] and a tutorial is accessible as a PDF file.

Testing - Deployment on multiple architectures with heterogenous hardware and software environments requires testing procedures to assess the stability of the solutions provided by compiled binaries. This testing procedure is made easy for any developer by the *CMake/CTest* testing software. OpenMEEG test suite guarantees the integrity of the results obtained by MEG, EEG and EIT forward solvers. Part of the tests consist in running OpenMEEG on a 3-layer spherical geometry like the one presented in figure 2. Outputs are then compared to analytical results to test that the accuracy is not degraded by a modification of the code.

II. Why use OpenMEEG?

OpenMEEG is a general package for quasistatic bioelectromagnetics. In order to motivate the use of this software, we have conducted a set of numerical experiments that compare the accuracy and the robustness of OpenMEEG with for alternative solvers of the M/EEG forward problems. The comparison in the context of EEG forward modeling is run with the 4 alternative BEM solvers. The precision of the MEG forward

[1] http://www-sop.inria.fr/apics/FindSources3D/

[2] Please cite references: [17], this article, and according to application domain, [6], etc.

[3] http://sourceforge.net/projects/matio/

[4] http://subversion.tigris.org/

[5] http://www.cmake.org

[6] http://www.stack.nl/~dimitri/doxygen/

solver is demonstrated using known analytical properties of the magnetic field when considering sphere models and with two other solvers.

A. Publicly available M/EEG forward solvers

We now present a set of non-commercial software packages that numerically solve the M/EEG forward problem. Some of these packages are completely open source: OpenMEEG, Simbio, Brainstorm, EEGLAB (via the NFT Toolbox) and Fieldtrip that shares with SPM the same code for M/EEG forward modeling.

The Fieldtrip Toolbox and SPM offer two implementations of the BEM. The first one, called *dipoli*, was written by Oostendorp [23] and is not open source (only binary files for UNIX systems are available), while the second one, called *BEMCP*, is opensource and was written by Christophe Phillips during his PhD [24]. The *dipoli* solver, like Simbio, implements a linear collocation method, whereas BEMCP respectively implements a constant collocation method. The *dipoli* implementation details can be found in [23]. Another implementation with constant collocation can be found in the Matlab toolbox called Helsinki BEM and written by Stenroos [27]. The MNE package written in pure C code by Hämäläinen also offers a linear collocation implementation of the BEM. The Brainstorm toolbox in its latest version uses only sphere models for both EEG and MEG. Since recently the EEGLAB toolbox also provides a package called NFT that is based on a BEM called METU [1]. Note that the SimBio code is an opensource version of the M/EEG solvers integrated in the commercial product ASA [29].

After the Geselowitz BEM was discovered to have unacceptable accuracy errors for the forward EEG problems, the Isolated Skull Approach (ISA) was proposed to compensate for these errors [12]. The solvers tested in the following section, other than OpenMEEG, use the ISA.

B. Benchmark: Numerical experiments

The accuracy of forward solvers can be assessed for simple geometries such as nested spheres, by comparison with analytical results. The precision of a forward solution is tested with 2 quantities: the RDM (Relative Difference Measure) and the MAG (Magnitude) [20].

The RDM between the forward field given by a numerical solver g_n and the analytical solution g_a is defined as:

$$RDM(g_\mathrm{n}, g_\mathrm{a}) = \left\| \frac{g_n}{\|g_n\|} - \frac{g_a}{\|g_a\|} \right\| \in [0, 2] \ ,$$

while the MAG between the two forward fields is defined as:

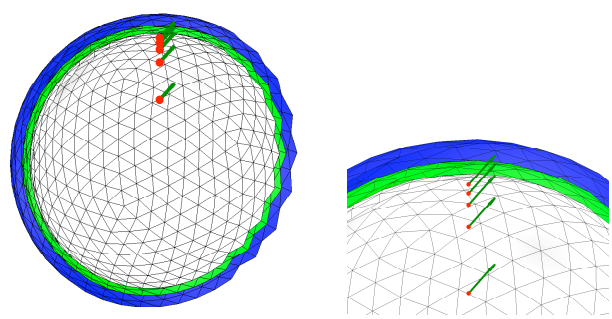
$$MAG(g_n, g_a) = \frac{\|g_n\|}{\|g_a\|} \ .$$

In both of these expression, the norm is a discrete ℓ^2 norm over the set of sensor measurements. The closer to 0 (resp. to 1) the RDM (resp. the MAG), the better it is.

The implementations tested for EEG are: OpenMEEG with and without adaptive numerical integration (abbr. OM and OMNA), SimBio (abbr. SB), Helsinki BEM (abbr. HB), Dipoli (abbr. DP) and BEMCP (abbr. CP). The METU solver was also tested but we were unable to obtain the precisions advertised in [1], so we decided not to include it in the comparison.

The comparisons were made on classic regular sphere meshes as in figure 2, and with random meshes. A random sphere mesh with unit radius and N vertices is obtained by randomly sampling 10N 3D points, normalizing them, meshing their convex hull and decimating the obtained triangular mesh from 10N to N vertices. This process guarantees an irregular meshing while avoiding flat triangles. The BEM solvers are tested with three nested sphere shells which model the inner and outer skull, and the skin. The radii of the 3 layers are set to 88, 92 and 100, while the conductivities of the 3 homogeneous volumes are set to standard values: 1, 1/80 (skull) and 1. For every head model, solvers are tested with the same 5 dipoles positioned on the z-axis with orientation (1,0,1) and various distances to the inner layer (cf. fig. 2). Precision of the solvers decreases as this distance gets small.

The results with regular sphere meshes are presented in fig. 3 on the next page, for 3 different point samplings on each interface. The coarsest sampling has only 42 vertices per interface and 42 EEG electrodes, the intermediate one has 162 points per interface and 162 EEG electrodes, and the finest sampling has 642 points per interface and 642 EEG electrodes.



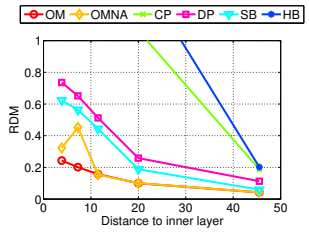
(a) 3-layer spherical head model (b) Zoom

Fig. 2. Head model made of 3 nested regular sphere meshes with 5 dipoles close to the inner layer.

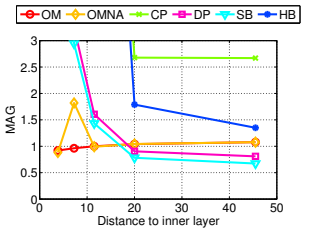
From these simulations it can be observed that:

- HB and CP, that implement a simple constant collocation method, have similar results and are clearly the less precise solvers.
- SB and DP have very similar results, but SB that implements a Galerkin method is slightly more accurate than DP.
- OpenMEEG provides the most precise solutions even when no adaptive integration is used. The adaptive integration significantly improves the results, particularly when the meshes are coarsely sampled (42 and 162 vertices per layer).

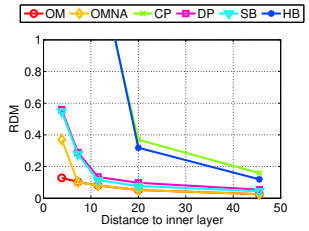
Simulations have also been run for a large number of sphere meshes with random sampling, in order to compare the robustness of the different solvers. Each result is obtained by running all solvers on 100 random head models. The mean accuracy (RDM and MAG) are represented using boxplots, in order to display the variance of the errors. Figure 4 presents



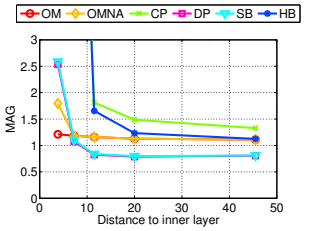
(a) RDM 42 points per interface



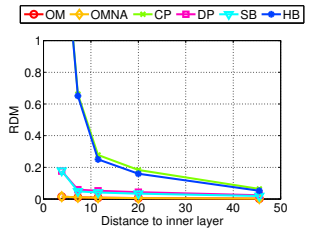
(b) MAG with 42 points per interface



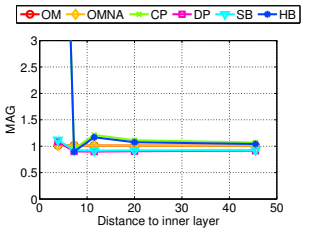
(c) RDM with 162 points per interface



(d) MAG with 162 points per interface

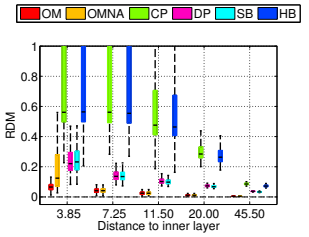


(e) RDM with 642 points per interface

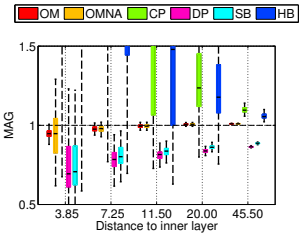


(f) MAG with 642 points per interface

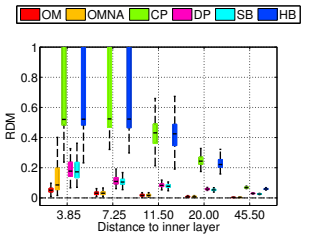
Fig. 3. Forward EEG: accuracy comparison of different BEM solvers with three-layers sphere head models. We observe that the Symmetric BEM outperforms the other methods in term of precision.



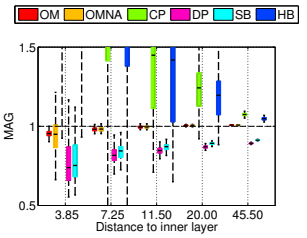
(a) RDM 600 points per interface



(b) MAG 600 points per interface

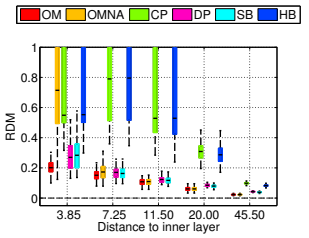


(c) RDM 800 points per interface

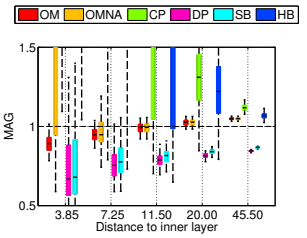


(d) MAG 800 points per interface

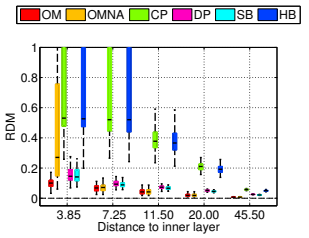
Fig. 4. Forward EEG: RDM and MAG boxplots obtained on 100 random 3-layers sphere models. Each layer contains 600 or 800 random vertices.



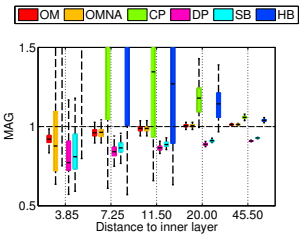
(a) RDM 1500 unknowns



(b) MAG 1500 unknowns



(c) RDM 3000 unknowns



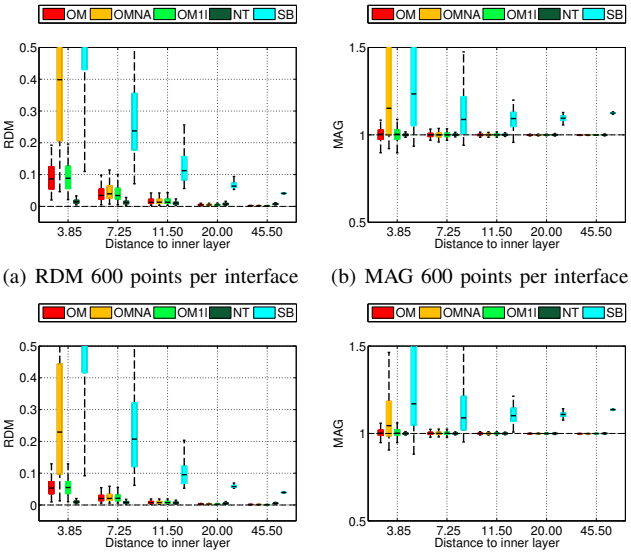
(d) MAG 3000 unknowns

Fig. 5. Forward EEG: RDM and MAG boxplots obtained on 100 random 3-layers sphere models. Each forward solution is obtained taking as constraint that the number of unknowns that is estimated is the same for all BEM solvers. Results are presented for 1500 and 3000 unknowns.

the boxplots obtained by running the solvers on random head models with either 600 or 800 vertices per interface. These results lead to the same ranking of methods as those of Figure 3, if the average accuracy is considered. However the variances are also very informative. It can be observed that OM is very accurate not only on average but also by its very small variance, which is an appreciable feature. The OMNA solver is also accurate but has a larger variance. This demonstrates that the adaptive integration makes the solver more robust to irregular meshing. One can also observe that collocation based methods are very sensitive to irregular meshing, as the variances observed for CP and HB are significantly larger than for the other solvers.

As explained in Section I-B, OpenMEEG considers as unknowns both the potential at vertices, and the normal current at the centers of the triangles. For a fair comparison with respect to numerical complexity, the previous experiments have been repeated with the constraint that each solver should handle an equal number of unknowns. This leads to considering meshes for OpenMEEG with less points than for others solvers. A closed triangular mesh with n vertices contains $2n-4$ triangles and the normal current is not discretized for the outer layer as it is fixed to be 0. For a three-layer BEM, OpenMEEG therefore has to handle $7\,n-8$ unknowns, while for a standard BEM this number is simply $3\,n$. For a fixed number n_u of unknowns, the number of vertices per layer n_{om} for OpenMEEG is $n_{om} = (n_u + 8)/7$ while for a standard BEM the number of vertices n_{std} is $n_{std} = n_u/3$. Results with 1500 and 3000 unknowns are presented in figure 4. It can be observed that OpenMEEG with adaptive integration still outperforms other solvers in term of mean accuracy as well as variance of the results. It is followed by DP and SB, that give also quite accurate solutions. OpenMEEG without adaptive integration is in this simulation behind DP and SB but remains more precise than HB and CP.

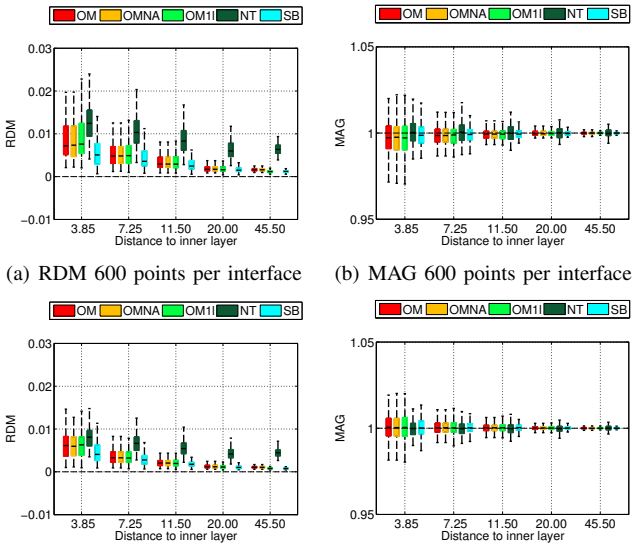
We next present results for MEG forward modeling. MEG manufacturers propose 3 kinds of sensors (magnetometers, axial gradiometers, and planar gradiometers), all of which are oriented radially with respect to the helmet. With a nested sphere model, volume currents do not contribute to the radial component of the magnetic field [25] (the second term in (3)



(a) RDM 600 points per interface (b) MAG 600 points per interface

(c) RDM 800 points per interface (d) MAG 800 points per interface

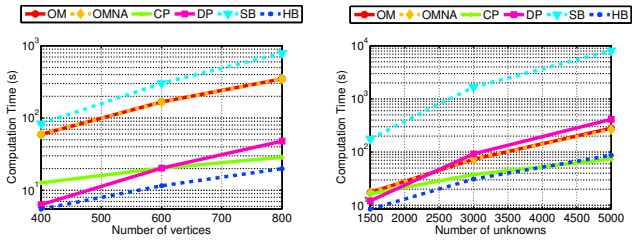
Fig. 6. Forward MEG: RDM and MAG boxplots obtained on 100 random sphere models (1 and 3-layers) using **non-radial magnetometers**. Each layer contains 600 or 800 random vertices.



(a) RDM 600 points per interface (b) MAG 600 points per interface

(c) RDM 800 points per interface (d) MAG 800 points per interface

Fig. 7. Forward MEG: RDM and MAG boxplots obtained on 100 random sphere models (1 and 3-layers) using **radial magnetometers**. Each layer contains 600 or 800 random vertices.



(a) Computation time as a function of the number of vertices. (b) Computation time as a function of the number of unknowns.

Fig. 8. Forward EEG: computation times for the different solvers as a function of the number of vertices per layer or the number of unknowns.

vanishes). The MEG community commonly uses analytical solutions on spheres to compute MEG leadfields although volume currents do have an influence on the magnetic field when considering realistic head models [28]. OpenMEEG (as well as SimBio) uses the previously computed electric potential on all surfaces to compute the contribution of the volume currents to the magnetic field at sensors. These considerations lead to a strategy to validate the MEG forward solutions provided by OpenMEEG. To do so, experiments have been run with two types of sensors: a set of magnetometers all oriented in the Cartesian direction $(1, 0, 1)$ and located at a distance of 120 from the center of the spheres, and a set of magnetometers at the same locations but radially oriented. In these experiments we compared the analytical results with the solutions given by SimBio, by OpenMEEG with and without adaptive integration (abbreviated by OM and OMNA), with a 3-layer model, and also using a single layer (the inner skull) model as commonly done in practice. The single layer solutions is abbreviated OM1L. The Fieldtrip Toolbox provides a solution to the MEG forward problem on realistic volume conductors which is not based on the Biot and Savart law but Helmholtz's reciprocity theorems [22]. This solver proposed by Nolte is abbreviated NT in the comparison results. Figure 6 presents the results with non-radial magnetometers, while fig. 7 presents the results obtained with radial magnetometers. From fig. 6 it can be observed that OpenMEEG with adaptive integration provides precise solutions that are significantly better than the one obtained without adaptive integration. SimBio gives results that are mesh-dependent, whereas Nolte's solver outperforms OpenMEEG and Simbio. In fig. 7, one can notice the correct cancellation of the volume current when the mesh size increases (notice that the y-scale changed). The OpenMEEG and Simbio solvers take the lead for radially oriented sensors with similiar results for both.

Finally we have compared the computation times of EEG forward solvers in the two situations: with a fixed number of vertices per layer or a fixed number of unknowns. When the number of vertices is fixed, the higher number of unknowns in the symmetric BEM makes the problem size larger. This is confirmed by the results presented in fig. 8(a) where is can be observed that OpenMEEG is slower than all solvers except SimBio. One explanation is that SimBio does not use BLAS/LAPACK for efficient linear algebra but implements its own routines in C code. When the number of unknowns is fixed (cf. fig. 8(b)), the computation time of OpenMEEG is comparable with Dipoli and even slightly less for highly sampled models. In all cases the collocation methods (HB and CP) are significantly faster, but their limited accuracy does not make them good candidates for EEG forward modeling.

III. How to use OpenMEEG?

Considerable efforts have been made to facilitate the use of OpenMEEG by the M/EEG community. OpenMEEG can be simply used with a command line interface, or via higher levels languages. OpenMEEG can be used from Python or via the Matlab Fieldtrip Toolbox.

```python
#!/usr/bin/env python
import openmeeg as om

# Load data
condFile='om_demo.cond'
geomFile='om_demo.geom'
dipoleFile='cortex.dip'
squidsFile='meg_squids.txt'
electrodesFile='eeg_electrodes.txt'

geom = om.Geometry()
geom.read(geomFile,condFile)

dipoles = om.Matrix()
dipoles.load(dipoleFile)

squids = om.Sensors()
squids.load(squidsFile)

electrodes = om.Matrix()
electrodes.load(electrodesFile)

# Compute forward problem (Build Gain Matrices)
gaussOrder = 3; # Numerical integration order

hm       = om.HeadMat(geom,gaussOrder)
hminv    = hm.inverse()
dsm      = om.DipSourceMat(geom,dipoles,gaussOrder)
ds2mm    = om.DipSource2MEGMat(dipoles,squids)
h2mm     = om.Head2MEGMat(geom,squids)
h2em     = om.Head2EEGMat(geom,electrodes)
gain_meg = om.GainMEG(hminv,dsm,h2mm,ds2mm)
gain_eeg = om.GainEEG(hminv,dsm,h2em)
```

TABLE I
DEMO SCRIPT FOR COMPUTING MEG AND EEG FORWARD PROBLEMS WITH OPENMEEG IN PYTHON.

OpenMEEG from Python: A demo of the Python scripting interface of OpenMEEG is provided in table I on the next page.

OpenMEEG from Matlab using Fieldtrip: OpenMEEG is fully integrated into Fieldtrip for M/EEG forward modeling: a sample MATLAB code is presented in table II. The benchmark presented in Section II was run within the Fieldtrip environment and its MATLAB source code can be obtained for noncommercial use from the authors. Since SPM uses the same code as Fieldtrip for forward modeling, SPM can now benefit from integration of OpenMEEG. Thanks to the opensource MATIO library, OpenMEEG also has the possibility to save its results in MATLAB format.

A sample dataset for M/EEG forward modeling is available online at http://openmeeg.gforge.inria.fr. The sample dataset is provided with the different scripts that can be run to compute both MEG and EEG leadfields on a real anatomy with 3 layers.

OpenMEEG is provided with a tutorial available on the web site. This tutorial describes the low level interface of OpenMEEG and details the different steps to be followed when computing M/EEG leadfields. It is briefly summarised in Appendix B.

CONCLUSION

In this paper, the OpenMEEG software project has been detailed, from the mathematical grounds of the symmetric BEM to more practical aspects. OpenMEEG is multiplatform

```cpp
%% The structure for the BEM volume conduction model
%% Each layer mesh is indexed by k
% vol.bnd(k).pnt : contains vertices for mesh "k"
% vol.bnd(k).tri : contains triangles for mesh "k"
%% Set the conductivities of each domain
% vol.cond       : contains conductivities

%% EEG electrodes
% sens.pnt        : contains locations of electrodes

%% Positions of the dipoles
% pos             : contains locations of dipoles

%% Compute the BEM
% choose BEM method (OpenMEEG, BEMCP or Dipoli)
cfg.method = 'openmeeg';
% Compute the BEM matrix
vol = ft_prepare_bemmodel(cfg, vol);
cfg.vol = vol;
cfg.grid.pos = pos;
cfg.elec = sens;
% Compute leadfield (no orientation constraint)
lf_openmeeg = ft_prepare_leadfield(cfg);
```

TABLE II
DEMO SCRIPT FOR COMPUTING AN EEG FORWARD PROBLEM WITH OPENMEEG IN FIELDTRIP.

and is integrated into higher level languages to increase its usability for non experts.

The use of OpenMEEG for quasistatic bioelectromagnetics is motivated by a benchmark of many alternative solvers in the context of M/EEG forward modeling. The results of the simulation study demonstrate that OpenMEEG outperforms all the alternative solvers that have been tested. By providing state-of-the-art solutions for both EEG and MEG forward problems, OpenMEEG enables the joint use of these two complementary modalities.

It should be mentioned that OpenMEEG is also under active development for other problems in the field of quasistatic bioelectromagnetics. OpenMEEG provides solvers for Electrical Impedance Tomography, Intracranial electric potentials, Functional Electrical Stimulation and Cortical Mapping. This wide range of application domains makes the software unique and particularly valuable for research and clinical purposes.

Acknowledgements. The authors acknowledge support from the ANR grant ViMAGINE ANR-08-BLAN-0250-02, and for E.O. from a PhD grant from the Regional Council of Provence Alpes Cote d'Azur. Help is gratefully acknowledged from C. Micheli and R. Oostenveld regarding the integration of OpenMEEG within Fieldtrip, and from A. Anwander, M. Dannhauer and C. Wolters regarding the use of Simbio.

REFERENCES

[1] Zeynep Akalin-Acar and Nevzat G Gençer. An advanced boundary element method (bem) implementation for the forward problem of electromagnetic source imaging. *Physics in medicine and biology*, 49(21):5011–28, 2004.

[2] Marc Bonnet. *Boundary Integral Equations Methods for Solids and Fluids*. John Wiley and Sons, 1999.

[3] M. Clerc, G. Adde, J. Kybic, T. Papadopoulo, and J.-M. Badier. In vivo conductivity estimation with symmetric

boundary elements. In Jaakko Malmivuo, editor, *International Journal of Bioelectromagnetism*, volume 7, pages 307–310, 2005.

[4] M. Clerc, J.-M. Badier, G. Adde, J. Kybic, and T. Papadopoulo. Boundary element formulation for electrical impedance tomography. In *ESAIM: Proceedings*, volume 14, pages 63–71. EDP Sciences, sep 2005.

[5] M. Clerc, A. Dervieux, O. Faugeras, R. Keriven, J. Kybic, and T. Papadopoulo. Comparison of bem and fem methods for the e/meg problem. In *Proceedings of BIOMAG Conference*, aug 2002.

[6] M. Clerc and J. Kybic. Cortical mapping by Laplace-Cauchy transmission using a boundary element method. *Inverse Problems*, 23:2589–2601, 2007.

[7] J. C. de Munck. A linear discretization of the volume conductor boundary integral equation using analytically integrated elements. *IEEE Trans. Biomed. Eng.*, 39(9):986–990, September 1992.

[8] J.C. de Munck and M.J. Peters. A fast method to compute the potential in the multisphere model. *IEEE Trans. on Biomed. Engin.*, 40(11):1163–1174, 1993.

[9] A. S. Ferguson, X. Zhang, and G. Stroink. A complete linear discretization for calculating the magnetic field using the boundary element method. *IEEE Trans. Biomed. Eng.*, 41(5):455–459, May 1994.

[10] D. B. Geselowitz. On bioelectric potentials in an homogeneous volume conductor. *Biophysics Journal*, 7:1–11, 1967.

[11] S. Gonçalves, J.C. de Munck, J.P. Verbunt, R.M. Heethaar, and F.H. Lopes da Silva. In vivo measurement of the brain and skull resistivities using an EIT-based method and the combined analysis of SEF/SEP data. *IEEE Transactions on Biomedical Engineering*, 50(9):1124–8, September 2003.

[12] M. S. Hämäläinen and J. Sarvas. Realistic conductivity geometry model of the human head for interpretation of neuromagnetic data. *IEEE Trans. Biomed. Eng.*, 36(2):165–171, February 1989.

[13] Matti Hämäläinen, Riitta Hari, Risto J. IImoniemi, Jukka Knuutila, and Olli V. Lounasmaa. Magnetoencephalography— theory, instrumentation, and applications to noninvasive studies of the working human brain. *Reviews of Modern Physics*, 65(2):413–497, April 1993.

[14] Bin He, Yunhua Wang, and Dongsheng Wu. Estimating cortical potentials from scalp EEGs in a realistically shaped inhomogeneous head model by means of the boundary element method. *IEEE Transactions on Biomedical Engineering*, 46(10):1264–1268, October 1999.

[15] S. Jacquir, J. Fruitet, D. Guiraud, and M. Clerc. Computation of the electrical potential inside the nerve induced by an electrical stimulus. In *EMBC 2007: Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2007.

[16] F. Kariotou. Electroencephalography in ellipsoidal geometry. *J. Math. Anal. Appl.*, (290):324–42, 2004.

[17] J. Kybic, M. Clerc, T. Abboud, O. Faugeras, R. Keriven, and T. Papadopoulo. A common formalism for the integral formulations of the forward EEG problem. *IEEE Transactions on Medical Imaging*, 24:12–28, jan 2005.

[18] J. Kybic, M. Clerc, O. Faugeras, R. Keriven, and T. Papadopoulo. Generalized head models for MEG/EEG: boundary element method beyond nested volumes. *Physics in Medicine and Biology*, 51:1333–1346, 2006.

[19] J. Laforêt, D. Guiraud, and M. Clerc. A toolchain to simulate and investigate selective stimulation strategies for fes. In *EMBC 2009: Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2009.

[20] J. W. H. Meijs, O. W. Weier, M. J. Peters, and A. van Oosterom. On the numerical accuracy of the boundary element method. *IEEE Trans. Biomed. Eng.*, 36:1038–1049, 1989.

[21] Jean-Claude Nédélec. *Acoustic and Electromagnetic Equations*. Springer Verlag, 2001.

[22] G. Nolte. The magnetic lead field theorem in the quasi-static approximation and its use for magnetoencephalography forward calculation in realistic volume conductors. *Physics in Medicine and Biology*, 48:3637–3652, oct 2003.

[23] T.F. Oostendorp and A. van Oosterom. Source parameter estimation in inhomogeneous volume conductors of arbitrary shape. *IEEE Trans. Bioned. Engin.*, 36:382–391, 1989.

[24] C. Phillips. *Source estimation in EEG*. PhD thesis, Université de Liège, 2000.

[25] Jukka Sarvas. Basic mathematical and electromagnetic concepts of the biomagnetic inverse problem. *Phys. Med. Biol.*, 32(1):11–22, 1987.

[26] M. Soleimani, R.J. Shipley, N. Smith, and C.N. Mitchell. Medical imaging and physiological modelling: linking physics and biology. *BioMedical Engineering OnLine*, 8(1), 2009.

[27] M. Stenroos, V. M"antynen, and J. Nenonen. A matlab library for solving quasi-static volume conduction problems using the boundary element method. *Comput. Methods Prog. Biomed.*, 88(3):256–263, 2007.

[28] R. Van Uitert, C. Johnson, and L. Zhukov. Influence of head tissue conductivity in forward and inverse magnetoencephalographic simulations using realistic head models. *IEEE Trans. on Biomed. Engin.*, 51(12):2129–2137, 2004.

[29] F. Zanow and T.R. Knosche. Asa-advanced source analysis of continuous and event-related eeg/meg signals. *Brain Topography*, 16:287–290(4), 2004.

[30] F. Zanow and M. J. Peters. Individually shaped volume conductor models of the head in eeg source localisation. *Medical and biological engineering and computing*, 33(4):582–8, 1995.

APPENDIX

A. The symmetric BEM

Comparing a Boundary Element and a Finite Element Method (FEM) for forward electroencephalography, in our

early numerical experiments, we found a superior precision to the FEM [5]. This triggered a quest to improve the precision of Boundary Element Methods and led us to study the extended Green representation theorem [21]. We proposed a common formalism for the integral formulations of the forward EEG problem, and we derived three different Boundary Element Methods within the same framework [17]. In this section we recall the mathematical background of Boundary Element Methods, and we present both the double-layer BEM, which is the most widespread method, and the symmetric BEM, which is a new formulation.

1) Green Representation: A fundamental result in potential theory shows that a harmonic function (i.e., such that $\Delta u = 0$) is uniquely determined within a domain Ω from its value on the boundary $\partial\Omega$ (Dirichlet condition), or the value of its normal derivative (Neumann condition). The Green Representation Theorem gives an explicit representation of a piecewise-harmonic function as a combination of boundary integrals of its jumps and the jumps of its normal derivative across interfaces. Before stating this theorem, some notation must be defined:

- The restriction of a function f to a surface S_j is indicated by f_{S_j}.
- The functions $f^-_{S_j}$ and $f^+_{S_j}$ represent the interior and exterior limits of f on S_j:
$$\text{for } \mathbf{r} \in S_j, \qquad f^{\pm}_{S_j}(\mathbf{r}) = \lim_{\alpha \to 0^{\pm}} f(\mathbf{r} + \alpha \mathbf{n}).$$
- The jump of a function f across S_j is denoted by:
$$[f]_{S_j} = f^-_{S_j} - f^+_{S_j},$$
- $\partial_{\mathbf{n}} V = \mathbf{n} \cdot \nabla V$ denotes the partial derivative of V in the direction of a unit vector $\mathbf{n}$,
- The function $G(\mathbf{r}) = \frac{1}{4\pi\|\mathbf{r}\|}$ is the fundamental solution of the Laplacian in $\mathbb{R}^3$, such that $-\Delta G = \delta_0$.

Consider an open region Ω and a function u such that $\Delta u = 0$ in Ω and in $\mathbb{R}^3 \backslash \Omega$ (but not necessarily continuous across $\partial\Omega$). The Green Representation Theorem states that, for a point $\mathbf{r}$ belonging to $\partial\Omega$,

$$\frac{u^-(\mathbf{r}) + u^+(\mathbf{r})}{2} = -\int_{\partial\Omega} [u]\, \partial_{\mathbf{n}'} G(\mathbf{r} - \mathbf{r}') ds(\mathbf{r}') + \int_{\partial\Omega} [\partial_{\mathbf{n}'} u]\, G(\mathbf{r} - \mathbf{r}') ds(\mathbf{r}') \ . \quad (4)$$

This representation also holds for the head model in Figure 1, when Ω is the union of disjoint open sets: $\Omega = \Omega_1 \cup \Omega_2 \cup \ldots \Omega_N$, with $\partial\Omega = S_1 \cup S_2 \cup \ldots S_N$. If u is harmonic in each Ω_i, for $\mathbf{r} \in S_i$,

$$\frac{u^-(\mathbf{r}) + u^+(\mathbf{r})}{2} = -\sum_{j=1}^{N} \int_{S_j} [u]_{S_j}\, \partial_{\mathbf{n}'} G(\mathbf{r} - \mathbf{r}') ds(\mathbf{r}') + \int_{S_j} [\partial_{\mathbf{n}'} u]_{S_j}\, G(\mathbf{r} - \mathbf{r}') ds(\mathbf{r}') \quad (5)$$

The notation is simplified by introducing two integral operators, which map a scalar function f on $\partial\Omega$ to another scalar function on $\partial\Omega$: the "double-layer" operator $\big(\mathcal{D} f\big)(\mathbf{r}) = \int_{\partial\Omega} \partial_{\mathbf{n}'} G(\mathbf{r} - \mathbf{r}') f(\mathbf{r}')\, \mathrm{d}s(\mathbf{r}')$, and the "single-layer" operators $\big(\mathcal{S} f\big)(\mathbf{r}) = \int_{\partial\Omega} G(\mathbf{r} - \mathbf{r}') f(\mathbf{r}')\, \mathrm{d}s(\mathbf{r}')$. For an operator $\mathcal{D}$, its restriction $\mathcal{D}_{ij}$ maps a function of S_j to a function of S_i.

An extension of the Green Representation Theorem represents the directional derivative of a harmonic function as a combination of boundary integrals of higher order. This requires two more integral operators: the adjoint $\mathcal{D}^*$ of the double-layer operator, and a hyper-singular operator $\mathcal{N}$ defined by $\big(\mathcal{N} f\big)(\mathbf{r}) = \int_{\partial\Omega} \partial_{\mathbf{n},\mathbf{n}'} G(\mathbf{r} - \mathbf{r}') f(\mathbf{r}')\, \mathrm{d}s(\mathbf{r}')$. If $\mathbf{r}$ is a point of S_i,

$$-\frac{\partial_{\mathbf{n}} u^-(\mathbf{r}) + \partial_{\mathbf{n}} u^+(\mathbf{r})}{2} = +\mathcal{N}[u] - \mathcal{D}^*[\partial_{\mathbf{n}} u] \quad (6)$$

The Geselowitz formula exploits only the first boundary integral representation equation (5), while it is possible to exploit both (5) and (6). Thus three Boundary Element Methods can be derived within a unified setting: a BEM involving only single-layer potentials, a BEM involving only double-layer potentials, and a symmetric BEM combining single- and double-layer potentials [17]. We concentrate hereforth on the double-layer and on the symmetric BEMs.

2) The double-layer BEM: To apply the representation theorem to the forward problem of EEG, a harmonic function must be produced, which relates the potential and the sources. Decomposing the source term as $f = \sum_i f_i$ where the support of each f_i lies inside Ω_i, consider v_{Ω_i} such that $\Delta v_{\Omega_i} = f_i$ holds in all $\mathbb{R}^3$. The function $v_d = \sum_{i=1}^{N} v_{\Omega_i}$ satisfies $\Delta v_d = f$ and is continuous across each surface S_i, as well as its normal derivative $\partial_{\mathbf{n}} v_d$. The function $u = \sigma V - v_d$ is a harmonic function in Ω, to which (5) can be applied. Since $[u]_{S_i} = (\sigma_i - \sigma_{i+1}) V_j$ and $[\partial_{\mathbf{n}} u] = 0$, we obtain, on each surface S_i,

$$\frac{\sigma_i + \sigma_{i+1}}{2} V_j + \sum_{j=1}^{N} (\sigma_j - \sigma_{j+1})\, \mathcal{D}_{ij}\, V_j = v_d \ . \quad (7)$$

The above formula was established by Geselowitz [10], and was the only one used to model electroencephalography or electrocardiography, until recently, when [17] showed the diversity of BEMs that can be derived. This classical BEM is called a double-layer BEM because it only involves the double-layer operator $\mathcal{D}$.

3) The symmetric BEM: The originality of the symmetric Boundary Element Method is to consider a different piecewise harmonic function for each domain: the function u_{Ω_i} equal to $V - \frac{v_{\Omega_i}}{\sigma_i}$ within Ω_i and to $-\frac{v_{\Omega_i}}{\sigma_i}$ outside of Ω_i. This u_{Ω_i} is indeed harmonic in $\mathbb{R}^3 \backslash \partial\Omega_i$, and the representation equations (5) and (6) can be applied, leading to a system of integral equations involving two types of unknowns: the potential V_i and the normal current $(\sigma \partial_{\mathbf{n}} V)_i$ on each interface.

The surfaces are represented by triangular meshes. To fix ideas, consider a three-layer geometrical model for the head. Conductivities of each domain are respectively denoted σ_1, σ_2 and σ_3. The surfaces enclosing these homogeneous conductivity regions are denoted S_1 (inner skull boundary), S_2 (skull-scalp interface) and S_3 (scalp-air interface). Denoting $\psi_i^{(k)}$ the P0 function associated to triangle i on surface S_k, and $\phi_j^{(l)}$ the

P1 function associated to node j on surface S_l, the potential V on surface S_k is approximated as $V_{S_k}(\mathbf{r}) = \sum_i x_i^{(k)} \phi_i^{(k)}(\mathbf{r})$, while $p = \sigma \partial_{\mathbf{n}} V$ on surface S_k is approximated by $p_{S_k}(\mathbf{r}) = \sum_i y_i^{(k)} \psi_i^{(k)}(\mathbf{r})$.

As an illustration, considering the source term to reside in the brain compartment Ω_1, the variables $\left(\mathbf{x}_k\right)_i = x_i^{(k)}$ and $\left(\mathbf{y}_k\right)_i = y_i^{(k)}$ satisfy the linear system:

$$\begin{bmatrix} \hat{\sigma}_{12}\mathsf{N}_{11} & -2\mathsf{D}_{11}^* & -\sigma_2\mathsf{N}_{12} & \mathsf{D}_{12}^* & 0 \\ -2\mathsf{D}_{11} & \check{\sigma}_{12}\mathsf{S}_{11} & \mathsf{D}_{12} & -\sigma_2^{-1}\mathsf{S}_{12} & 0 \\ -\sigma_2\mathsf{N}_{21} & \mathsf{D}_{21}^* & \hat{\sigma}_{23}\mathsf{N}_{22} & -2\mathsf{D}_{22}^* & -\sigma_3\mathsf{N}_{23} \\ \mathsf{D}_{21} & -\sigma_2^{-1}\mathsf{S}_{21} & -2\mathsf{D}_{22} & \check{\sigma}_{23}\mathsf{S}_{22} & \mathsf{D}_{23} \\ 0 & 0 & -\sigma_3\mathsf{N}_{32} & \mathsf{D}_{32}^* & \sigma_3\mathsf{N}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{y}_1 \\ \mathbf{x}_2 \\ \mathbf{y}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{c}_1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where $\hat{\sigma}_{ij}$ (resp. $\check{\sigma}_{ij}$) is defined as $\sigma_i + \sigma_j$ (resp. $\sigma_i^{-1} + \sigma_j^{-1}$) and where $\mathbf{b}_1$ and $\mathbf{c}_1$ are the coefficients of the P0 (resp. P1) boundary element decomposition of the source term $\partial_{\mathbf{n}} v_{\Omega_1}$ (resp. $-\sigma_1^{-1} v_{\Omega_1}$).

Blocks N_{ij} and D_{ij} map a potential V_j on S_j to a function defined on S_i. Block S_{ij} maps a normal current p_j on S_j to a function defined on S_i. The resulting matrix is block-diagonal, and symmetric, hence the name "symmetric BEM".

The magnetic field is computed from the electric field and the primary source distribution using the Biot and Savart equation (3), as proposed by Ferguson, Zhang and Stroink [9].

In summary, the symmetric BEM introduces an additional unknown into the problem: the normal current, and uses an additional set of representation equations linking the normal current and the potential. The symmetric BEM departs from the double-layer BEM in several ways:

- the normal current to each surface is explicitely modeled;
- only the surfaces which bound a common compartment have an interaction (whence the blocks of zeros in the matrix);
- only the surfaces which bound a compartment containing sources have a source term;
- the matrix is symmetric;
- the matrix is larger for a given head model.

B. OpenMEEG: Hands-on tutorial

1) OpenMEEG file formats: OpenMEEG handles several file formats corresponding to several type of objects: vectors, matrices, head geometries, meshes, dipoles, conductivities.

a) Vectors and matrices: By default matrices and vectors are stored on disk using a MATLAB file format. Symmetric matrices which are not directly representable in the MATLAB format are represented as a MATLAB struct. Other vector/matrices file formats are also supported. Forcing a specific file format is achieved by specifying the proper file extension. Matlab extension is `.mat`. Other useful file formats are ASCII (extension `.txt`) which generates human readable files, BrainVisa texture file format (extension `.tex`) and OpenMEEG's own binary file format (extension `.bin`) which is available solely for backward compatibility and should be considered as deprecated (as it is subsumed by the MATLAB file format).

b) Geometrical model, mesh and conductivity files: **OpenMEEG geometrical models** are provided through several files. The toplevel file (generally ending with the extension `.geom`) assembles various interface descriptions to build *Domains* corresponding to head tissues. Empty lines or lines beginning with # are non-significant. The file must start with a special comment line which allows its identification (see example in Figure 9). Geometrical models globally contain 2 sections.

The first section specifies the mesh descriptions of the interfaces between tissues. It is introduced by the keyword `Interfaces` followed by the number of such meshes followed by the keyword `Mesh` (which gives the type of the mesh description, `Mesh` being currently the only possibility). This specification is then followed by the names of – mesh – files containing the description of each of the interfaces. Each interface is assigned an index from its order in the list. These indices start at 1.

The second section describes the head tissues and is introduced by the keyword `Domains` followed by the number of such domains. Each domain is then depicted, one domain per line, by the keyword `Domain` followed by the domain name (which will serve for identification and will also appear in the conductivity description) followed by a list of integers. These integer are the indices of the mesh files (as depicted in previous paragraph). They are affected by a + or - sign depending on whether the domain is outside or inside the corresponding interface (as defined by the outward normal of the interface). See Figure 9 for a detailed example.

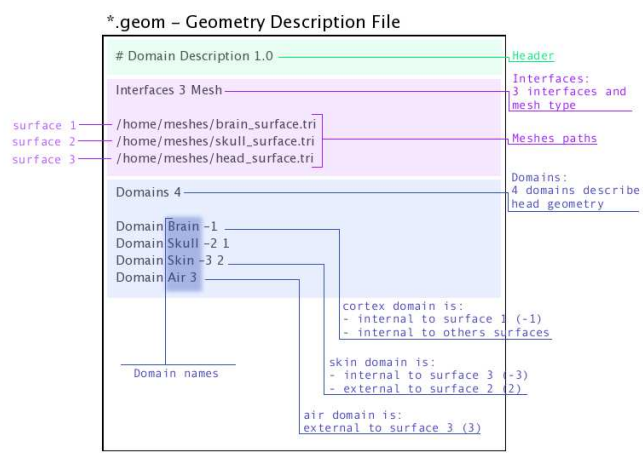


Fig. 9. Sample geometry file.

Mesh files (generally ending with the `.tri` extension) follow the BrainVisa file format for meshes. These files contain two sections. Each section is introduced by the character – appearing at the beginning of the line followed by a space followed by either one number (first section) or three times the same number (second section).

The first section contains a list of points with associated normals. The number on the line introducing the section is the number of points. Each following line corresponds to a single point. Its coordinates are the three first numbers appearing on the line. The normal corresponds to the following

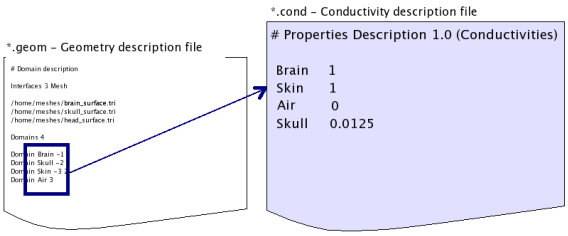


Fig. 10. Sample conductivity file and its correspondance with the geometry file.

three numbers. Each point is assigned an index (starting at 0) corresponding to its order of appearance in the list.

The second section contains the triangles of the mesh. The number (repeated three times) in the section delimiter corresponds to the number of triangles. Each triangle is depicted by a sequence of three integers corresponding to the indices of the points assigned as described in the previous paragraph.

The following small example describes a very simple mesh containing 4 points and 4 triangles.

```
- 4
0 0 0 -0.5773 -0.5773 -0.5773
1 0 0 1 0 0
0 1 0 0 1 0
0 0 1 0 0 1
- 4 4 4
0 1 2
0 1 3
0 2 3
1 2 3
```

The interface meshes are required to be closed in order for the Boundary Element Method to function correctly. This is also necessary for the source meshes when computing forward solutions using surfacic source models (see below). Moreover, the interface meshes must not intersect each other. Non-intersection can be checked with the command `om_check_geom`. The command `om_mesh_info` applied to a mesh provides its number of points, of triangles, minimum and maximum triangle area, and also its Euler characteristic. The Euler characteristic of a closed mesh of genus 0 (homotopic to a sphere) is equal to 2. The Euler characteristic gives an indication if a mesh is likely to be closed or not.

A **conductivity file** (generally ending with the extension `.cond`) is a simple ASCII file that contains associations between tissue names and conductivity values. Associations are provided one per line. Empty lines or lines beginning with # are non-significant. The file must start with a special comment line which allows its identification. Figure 10 provides an example conductivity file corresponding to the geometry file of figure 9.

Note that the tissue names are the ones appearing in the Domains descriptions of the file depicting the geometrical model.

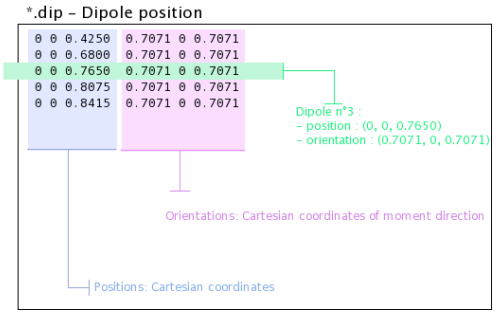


Fig. 11. Sample source description for isolated dipoles.

c) Source descriptions:
Sources may be represented either by a surfacic distribution of dipoles, or by isolated dipoles.
A **surfacic distribution** can be defined by a mesh that supports the dipoles. The dipole orientations are then constrained to the normal direction to the mesh and the moment amplitude is modelled as continuous across the mesh (piecewise linear). Source values are defined at the mesh vertices.
Isolated dipoles are defined by a simple ASCII file as shown in Figure 11.

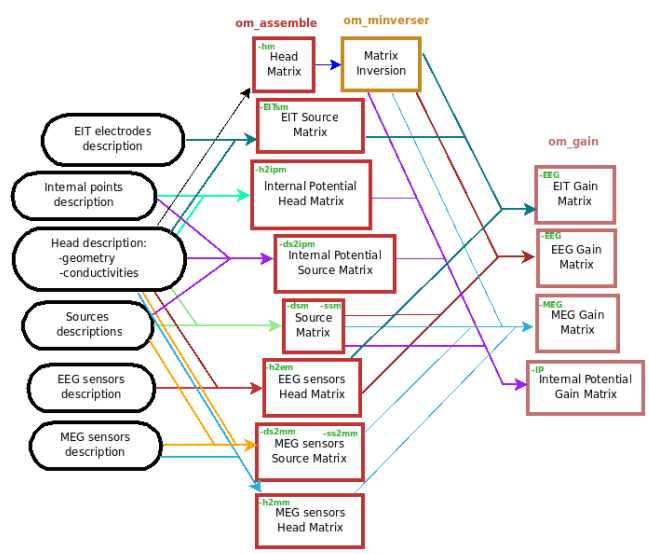


Fig. 12. Diagram for the low level pipeline for computing MEG and EEG leadfields (a.k.a., gain matrices) using OpenMEEG.

2) OpenMEEG from the command line: This section reviews the main OpenMEEG command line tools. The general syntax and main options of each command is briefly provided. Full details are available in OpenMEEG documentation. *In this section, command names are in* `red`*, options are in* `green` *and produced files are shown in* `blue`*.*

om_assemble: General syntax:

om_assemble Option Parameters Matrix

This program assembles the different matrices that will be used in later stages. It uses the head description, the sources and the sensors information. Option selects the type of matrice to assemble. Parameters depends on the specific option Option. Except if otherwise noted, it takes the form:

subject.geom subject.cond OptParam

where subject.geom and subject.cond are files describing respectively the geometrical model and the conductivities of the head (see Appendix B1 for a short description of these files). OptParam depends on the actual Option. Matrix is the name of the output file containing the computed matrix.

We now detail the possible Options (with their abbreviated versions given in parentheses), allowing to define various matrices to assemble:

General options for om_assemble*:*

-help (-h,--help): summarizes all possible options.

Head modelling options for om_assemble*:* produce matrices linked to the propagation of electrical signals in the head.

-HeadMat (-HM, -hm): om_assemble computes the Head matrix for Symmetric BEM (left-hand side of the linear system presented in Section A3). This matrix corresponds to the propagation of electrical signals within the head. There is no OptParam in this case.

Source modelling options for om_assemble*:* compute the source matrix for Symmetric BEM (right-hand side of the linear system). This matrix maps the representation of $\mathbf{J}^{\mathrm{p}}$ to its associated electric potential in an infinite medium (v_{Ω_1} in Appendix A). Different options exist for the 2 types of source models:

-SurfSourceMat (-SSM, -ssm): should be used for continuous surfacic distributions of dipoles. OptParam is a file containing a mesh that describes the surface.

-DipSourceMat (-DSM, -dsm): should be used when considering several isolated dipoles. This model is the most commonly used and should be used by default even if the dipoles correspond to the vertices of a cortical mesh. OptParam is a file containing the dipole descriptions.

Sensor modelling options for om_assemble*:* compute matrices that integrate source information and computed potentials to provide the actual solution of the forward problem. The situation is slightly different for EEG, which only needs to compute the electric potential, and for MEG, which depends both on the electric potential and on the sources:

-Head2EEGMat (-H2EM, -h2em): om_assemble computes the interpolation matrix that maps potentials computed on the scalp to EEG sensors. OptParam is a file describing the EEG sensor positions.

-Head2MEGMat (-H2MM, -h2mm): om_assemble computes the contribution of Ohmic currents to the MEG sensors. OptParam is a file describing the SQUIDS geometries and characteristics.

-Head2InternalPotMat (-H2IPM, -h2ipm): om_assemble computes the matrix that allows the computation of potentials at internal positions from potentials and normal currents on head interfaces, as computed by the symmetric BEM.

-SurfSource2MEGMat (-SS2MM, -ss2mm): om_assemble computes the source contribution to the MEG sensors using the same source model as the one used for the option -SurfSourceMat, i.e. surfacic distribution of dipoles. For this option, OptParam takes the form:

mesh squids

where mesh contains a mesh describing the source surface and squids is a file describing the SQUIDS geometries and characteristics.

-DipSource2MEGMat (-DS2MM, -ds2mm): om_assemble computes the source contribution to the MEG sensors using the same source model as the one used for the option -DipSourceMat, i.e. isolated dipoles. For this option, OptParam takes the form:

dipoles squids

where dipoles contains the dipole description and squids is a file describing the SQUIDS geometries and characteristics.

-DipSource2InternalPotMat (-DS2IPM, -ds2ipm): om_assemble computes the source contribution to the chosen internal points. It gives the potential due to isolated dipoles, as if the medium were infinite. For this option, OptParam takes the form:

dipoles internalPoints

where dipoles contains the dipole description and internalPoints is a file describing the points locations.

EIT options for om_assemble*:*

-EITSourceMat (-EITSM, -EITsm,): om_assemble computes the right-hand side for scalp current injection. This usage of om_assemble outputs the right-hand side vector for a given set of EIT electrode. For this option, OptParam is a file describing the EIT electrode positions.

om_minverser: General syntax:

om_minverser HeadMat HeadMatInv

This program is used to invert the symmetric matrix as provided by the command om_assemble with the option -HeadMat. This command has only one option.

-help (-h,--help): summarizes the usage of om_minverser.

om_gain: General syntax:

om_gain Option HeadMatInv Parameters SourceMat Head2EEGMat GainMatrix

This command computes the gain matrix by multiplying together matrices obtained previously (e.g. HeadMatInv is the matrix computed using om_minverser). The resulting

gain matrix is stored in the file GainMatrix. Option selects the type of matrice to build. Parameters depend on the specific option Option. Possible options are:

General options:

-help (-h,--help): summarizes the usage of om_gain for all its possible options.

Gain matrix type options: select the type of gain matrix to be computed by om_gain.

-EEG : allows to compute an EEG gain matrix.
Parameters are then:
HeadMatInv SourceMat Head2EEGMat

SourceMat is the matrix obtained using om_assemble with either of the options -SurfSourceMat or -DipSourceMat, depending on the source model. Head2EEGMat is the matrix obtained using om_assemble with the option -Head2EEGMat.

The -EEG option is also used to compute an EIT gain matrix: in this case, SourceMat should contain the output of the -EITsource option of om_assemble. Multiplying the EIT gain matrix by the vector of applied currents at each EIT electrode yields the simulated potential on the EEG electrodes. The applied current on the EIT electrodes should sum to zero.

-MEG : allows to compute an MEG gain matrix.
Parameters are then:
HeadMatInv SourceMat Head2MEGMat Source2MEGMat
SourceMat is the matrix obtained using om_assemble with either of the options -SurfSourceMat or -DipSourceMat, depending on the source model. Head2MEGMat is the matrix obtained using om_assemble with the option -HeadMEEGMat. Source2MEGMat is the matrix obtained using om_assemble with either of the options -SurfSource2MEGMat or -DipSource2MEGMat, depending on the source model.

-InternalPotential : allows to compute an internal potential gain matrix for sensors within the volume.
Parameters are then:
HeadMatInv SourceMat Head2InternalPotMat Source2InternalPotMat
Head2InternalPotMat and Source2InternalPotMat are respectivelly obtained using om_assemble with option -Head2InternalPotMat and -DipSource2InternalPotMat.

Examples: Assuming a head model represented by the geometry file head.geom and the conductivity file head.cond and EEG sensors detailed in a file head.eegsensors. Computing the EEG gain matrix for sources distributed on the surface represented by the file sources.tri is done via the following set of commands:

```
om_assemble -HeadMat head.geom head.cond head.hm
om_assemble -SSM head.geom head.cond sources.tri head.ssm
om_assemble -h2em head.geom head.cond head.eegsensors head.h2em
om_minverser head.hm head.hm_inv
om_gain -EEG head.hm_inv head.ssm head.h2em head.gain
```

Considering now isolated dipolar sources detailed in the file sources.dip with MEG sensors depicted in the file head.squids. Using the same head model, the MEG gain matrix is obtained via the following set of commands:

```
om_assemble -HeadMat head.geom head.cond head.hm
om_assemble -DSM head.geom head.cond sources.dip head.dsm
om_assemble -h2mm head.geom head.cond head.squids head.h2mm
om_assemble -ds2mm sources.dip head.squids head.ds2mm
om_minverser head.hm head.hm_inv
om_gain -MEG head.hm_inv head.dsm head.h2mm head.ds2mm head.gain
```