`invertmeeg`: A Unified Python Library and Benchmark for 112 M/EEG Inverse Solvers

Lukas Hecker
lukas.hecker.job@gmail.com

Abstract

Magnetoencephalography (MEG) and electroencephalography (EEG) source imaging requires solving an ill-posed inverse problem for which numerous algorithms have been proposed over the past decades. However, these methods are scattered across different programming languages, software packages, and publications, making systematic comparison under controlled conditions nearly impossible. We present the most comprehensive benchmark of M/EEG inverse solvers to date, evaluating 106 methods from ten algorithmic families—minimum norm estimates, LORETA variants, beamformers, Bayesian methods, sparse recovery, subspace scanning, matching pursuit, dipole fitting, deep learning, and hybrid approaches—across four evaluation scenarios of increasing difficulty using a unified simulation and evaluation framework. All solvers are implemented in `invertmeeg`, an open-source Python package providing 112 inverse solvers through a consistent two-step interface that integrates with the `MNE-Python` ecosystem. We evaluate solver performance on an `ico3` source space (1,284 dipoles, BioSemi-32 montage) using earth mover's distance (EMD) as the primary metric and average precision (AP) as a secondary detection metric, which we argue is more appropriate than the commonly used AUC for the highly imbalanced classification problem inherent to sparse source imaging. Our results reveal that no single method dominates all scenarios: subspace and hybrid methods excel at multi-source and extended-source recovery, while Bayesian methods lead under focal and noisy conditions. We provide practical guidance on solver selection based on the expected source configuration and computational budget. `invertmeeg` is available via `pip install invertmeeg` under the GPL-3.0-only license.

1 Introduction

The reconstruction of neural source activity from M/EEG sensor measurements is a fundamental problem in functional neuroimaging [1, 2]. Given a forward model that describes how source currents produce sensor-level signals, the inverse problem seeks to recover the underlying source distribution. Because the number of potential source locations (typically 10^3–10^4) far exceeds the number of sensors (typically 10^1–10^2), the problem is severely underdetermined and requires regularization or prior assumptions to obtain a unique solution [3].

Over the past four decades, a rich landscape of inverse methods has emerged. Minimum norm approaches [4] minimize the L_2 norm of the source distribution; the LORETA family [5, 6] incorporates spatial smoothness constraints; beamformers [7] construct spatial filters that maximize signal power at a target location while suppressing interference; Bayesian methods [8, 9] cast the problem in a probabilistic framework with structured priors; sparse recovery algorithms [10] exploit the assumption that only a small number of sources are simultaneously active; and subspace scanning methods such as MUSIC [11] exploit the orthogonality between signal and noise subspaces. More

recently, deep learning approaches have been proposed that learn the inverse mapping from data [12, 13].

Despite this wealth of methods, practitioners face a fragmentation problem. Implementations are distributed across different toolboxes—`MNE-Python` [14], Brainstorm [15], FieldTrip [16], SPM [17]—each supporting only a subset of methods, often with incompatible data formats and interfaces. Many published algorithms lack publicly available implementations entirely, or exist only as standalone scripts tied to specific datasets. This fragmentation impedes systematic comparison of methods under controlled conditions and makes it difficult for researchers to evaluate which method is most appropriate for their specific experimental paradigm.

We present `invertmeeg`, an open-source Python package that addresses this gap by providing 112 inverse solvers under a single, consistent application programming interface (API). All solvers inherit from a common `BaseSolver` class and follow a two-step workflow: (1) compute an inverse operator from a forward model, and (2) apply the operator to M/EEG data. The package builds on `MNE-Python` for data structures, forward modeling, and visualization, ensuring compatibility with existing analysis pipelines. Source code, documentation, and tutorials are available at `https://github.com/LukeTheHecker/invertmeeg` and `https://lukethehecker.github.io/invertmeeg/`.

The central contribution of this paper is a systematic benchmark of 106 solvers across four evaluation scenarios that span the range of source configurations encountered in practice: focal dipoles, multiple focal sources, spatially extended patches, and low-SNR conditions. To our knowledge, this constitutes the largest comparative evaluation of M/EEG inverse solvers published to date, and we provide actionable guidance on which solver families are best suited to which experimental conditions. Beyond the benchmark, `invertmeeg` provides:

- A **simulation framework** with contiguous Gaussian patch sources and realistic structured sensor noise for generating diverse synthetic M/EEG data with known ground truth.
- **Automatic regularization** via generalized cross-validation (GCV), L-curve, and product methods, with a shared implementation across all linear solvers.
- A **standardized evaluation suite** with metrics including earth mover's distance (EMD), average precision (AP), mean localization error (MLE), correlation, and spatial dispersion.

The remainder of this paper is organized as follows. Section 2 describes the software architecture and the solver interface. Section 3 provides an overview of the implemented solver categories. Section 4 details the shared regularization framework. Section 5 presents the simulation methodology. Section 6 describes the evaluation metrics, benchmark scenarios, and results. Section 7 provides usage examples. Section 8 discusses solver selection guidelines, related work, and limitations. Section 9 concludes the paper.

2 Software Architecture

A meaningful benchmark requires that all solvers are compared under identical conditions with a shared forward model, simulation pipeline, and evaluation protocol. The design of `invertmeeg` is guided by two principles: (1) every solver, regardless of its algorithmic family, should expose the same interface, and (2) adding a new solver should require minimal boilerplate. This section describes the main architectural components. Figure 2 provides an overview of the architecture and the two-step workflow.

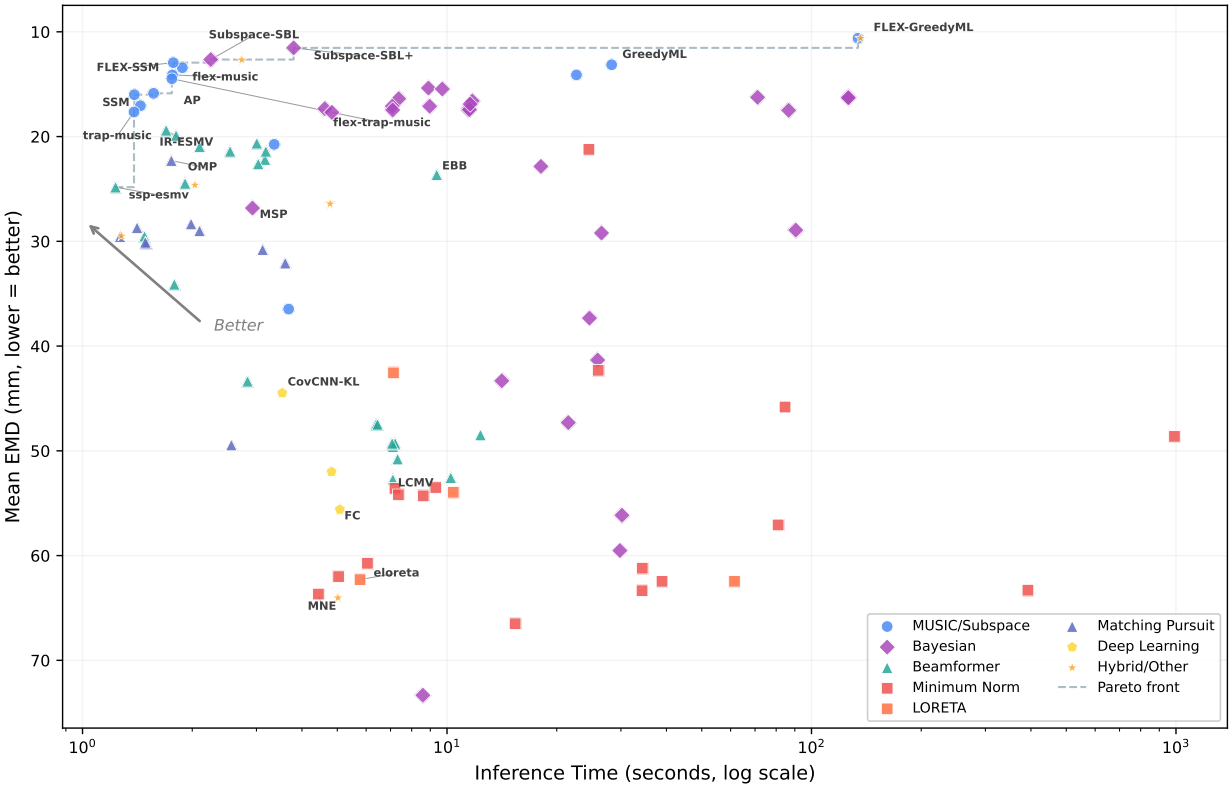

Figure 1: Performance–compute trade-off for all 106 evaluated solvers. Each point shows a solver's mean EMD (across four scenarios, in mm, lower is better) versus its average inference time per dataset (seconds, log scale). For deep learning solvers, only inference time is shown (training excluded). The dashed line marks the Pareto front of non-dominated solvers. Solver families occupy distinct regions: subspace methods cluster at the fast-and-accurate end, Bayesian methods trade speed for quality, and beamformers spread broadly.

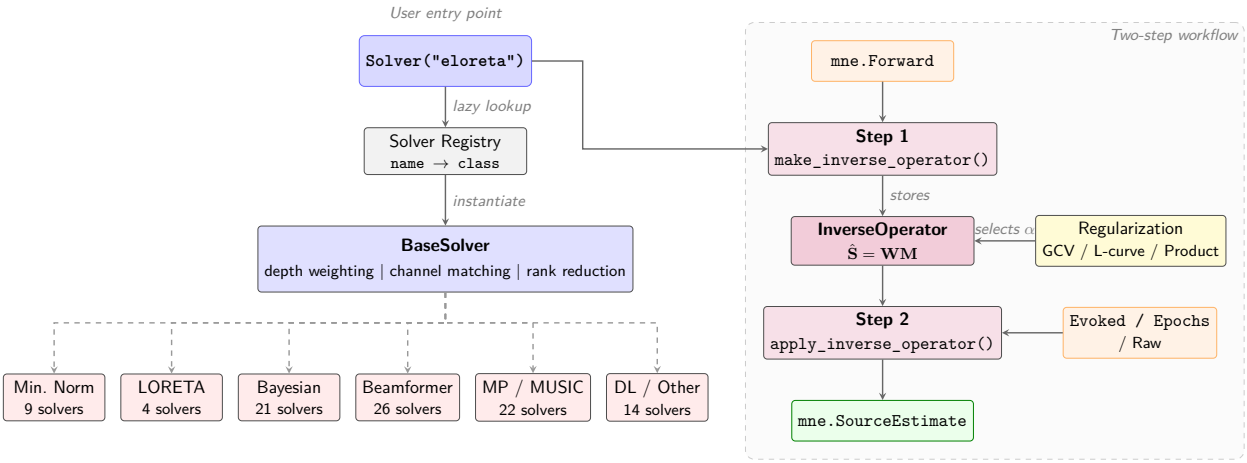

Figure 2: Architecture of `invertmeeg`. The `Solver()` factory lazily resolves a name to a concrete solver class inheriting from `BaseSolver`. The two-step workflow separates inverse operator computation (Step 1, depending only on the forward model) from application to data (Step 2). Regularization parameter selection happens at apply time.

2.1 The Solver Factory

The primary user-facing entry point is the `Solver()` factory function:

```
from invert import Solver

solver = Solver("eloreta")     # eLORETA
solver = Solver("champagne")   # Champagne (default update rule)
solver = Solver("em-champagne") # Champagne with EM update rule
```

The factory maintains a registry that maps solver names and aliases to constructor functions. Many solvers are registered under multiple aliases for convenience (e.g., `"l1"`, `"fista"`, and `"mce"` all resolve to the same minimum current estimate solver). The registry is populated lazily: solver modules are imported only when the factory is first invoked, which keeps the initial import of `invertmeeg` lightweight. For solvers that represent parametric variants of the same algorithm (e.g., the seven Champagne update rules), the registry stores `functools.partial` objects that pre-fill the relevant keyword arguments.

Calling `list_solvers()` returns a sorted list of all registered aliases.

2.2 BaseSolver and the Two-Step Interface

Every solver class inherits from `BaseSolver`, which defines the contract:

1. `make_inverse_operator(forward, alpha="auto", ...)` accepts an `MNE-Python Forward` object, computes the inverse operator, and stores it internally.

2. `apply_inverse_operator(mne_obj)` accepts an `MNE-Python` data object (`Evoked`, `Epochs`, or `Raw`), applies the precomputed inverse operator, and returns an `mne.SourceEstimate`.

This separation has a practical motivation: the inverse operator depends only on the forward model and regularization parameters, not on the data. Once computed, it can be applied to multiple

recordings without recomputation, which is particularly relevant for beamformers and minimum norm methods where the operator is a matrix multiplication.

The base class handles several concerns shared across all solvers:

- **Forward model preparation**: conversion of free-orientation forward models to fixed orientation, optional common average referencing of the leadfield, and depth weighting of leadfield columns.
- **Channel matching**: automatic intersection of channels present in the forward model and the data object.
- **Regularization parameter grid**: construction of a logarithmically spaced grid of regularization parameters, scaled by the largest eigenvalue of the leadfield Gram matrix $\mathbf{L}\mathbf{L}^\top$ (see Section 4).
- **Rank reduction**: optional signal subspace selection via SVD-based rank estimation.
- **Output conversion**: mapping of the source matrix back to an `mne.SourceEstimate` with correct vertex indices, timing, and subject metadata.

2.3 InverseOperator

For solvers that produce a linear inverse operator (a matrix $\mathbf{W} \in \mathbb{R}^{n_{\text{sources}} \times n_{\text{channels}}}$), the precomputed operator is wrapped in an `InverseOperator` object. Applying the operator to a data matrix $\mathbf{M} \in \mathbb{R}^{n_{\text{channels}} \times n_{\text{times}}}$ is then a single matrix multiplication:

$$\hat{\mathbf{S}} = \mathbf{W}\mathbf{M}, \tag{1}$$

where $\hat{\mathbf{S}} \in \mathbb{R}^{n_{\text{sources}} \times n_{\text{times}}}$ is the estimated source activity. When automatic regularization is enabled (`alpha="auto"`), the solver stores one `InverseOperator` per candidate regularization parameter; the optimal one is selected at apply time based on the chosen regularization criterion (Section 4).

2.4 Solver Metadata

Each solver class carries a `SolverMeta` dataclass with fields for the full solver name, acronym, category, a brief description, and literature references. This metadata serves two purposes: it enables programmatic generation of solver catalogs and documentation, and it allows users to inspect a solver's provenance at runtime.

2.5 Depth Weighting

Sources deeper in the brain produce weaker signals at the sensor level, biasing unweighted solutions toward superficial cortex. `invertmeeg` applies column-wise depth weighting to the leadfield matrix $\mathbf{L}$ before computing the inverse operator:

$$\tilde{\mathbf{L}}_{\cdot,j} = \frac{\mathbf{L}_{\cdot,j}}{\|\mathbf{L}_{\cdot,j}\|^d}, \tag{2}$$

where $d \in [0, 1]$ controls the weighting strength (default $d = 0.5$). After source estimation, the weighting is removed to recover physical amplitude units.

3 Solver Catalog

`invertmeeg` implements 112 inverse solvers organized into the categories summarized in Table 1. Rather than presenting the mathematical formulation of each solver, we describe the general principle of each category and refer the reader to the original publications. A complete list of all solvers with aliases is provided in the Appendix.

Table 1: Overview of solver categories in `invertmeeg`. The "Benchmarked" column indicates how many solvers from each category were included in the benchmark evaluation (3 deprecated solvers excluded).

Category	**Total**	**Bench.**	**Representative Methods**
Minimum Norm	17	17	MNE, wMNE, dSPM, MCE, MxNE, LAURA, S-MAP, TV
LORETA	4	4	LORETA, sLORETA, eLORETA, SSLOFO
Bayesian / SBL	31	29	Champagne (5 rules × noise variants), Gamma-MAP, MSP, SBL
Beamformers	25	24	LCMV, DICS, SAM, ESMV family, MCMV, EBB
Matching Pursuit	10	10	OMP, CoSaMP, ReMBo, Subspace Pursuit
MUSIC / Subspace	13	13	MUSIC, RAP-MUSIC, FLEX-MUSIC, SSM, AP, GreedyML
Dipole Fitting	2	2	ECD, SESAME
Deep Learning	6	3	FC (ESINet), CovCNN, CovCNN-KL, LSTM, CNN
Hybrid / Other	4	4	Chimera, Hydra, APSE, Random Noise (baseline)
Total	**112**	**106**	

3.1 Minimum Norm Methods

Minimum norm methods seek the source distribution $\hat{\mathbf{s}}$ that minimizes a weighted norm of the source vector subject to the data fit:

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \|\mathbf{L}\mathbf{s} - \mathbf{m}\|_2^2 + \alpha \|\mathbf{R}\mathbf{s}\|_p^p, \tag{3}$$

where $\mathbf{L}$ is the leadfield matrix, $\mathbf{m}$ is the measurement vector, α is the regularization parameter, $\mathbf{R}$ is a weighting matrix, and $p \in \{1, 2\}$ selects the norm.

For $p = 2$ and $\mathbf{R} = \mathbf{I}$, Equation (3) yields the classic minimum norm estimate (MNE) [4] with the closed-form solution:

$$\hat{\mathbf{s}} = \mathbf{L}^\top (\mathbf{L}\mathbf{L}^\top + \alpha \mathbf{I})^{-1} \mathbf{m}. \tag{4}$$

Variants include weighted MNE (wMNE), which uses noise-weighted columns, and dynamic statistical parametric mapping (dSPM) [18], which normalizes source power by noise variance.

For $p = 1$, Equation (3) promotes sparse solutions; `invertmeeg` implements this via the FISTA algorithm [19] as well as combined L_1/L_2 penalties.

3.2 LORETA Family

Low Resolution Electromagnetic Tomography (LORETA) [5] adds a spatial smoothness constraint by incorporating the cortical graph Laplacian $\mathbf{\Delta}$ into the regularization term. The family includes LORETA, standardized LORETA (sLORETA) [6], and exact LORETA (eLORETA) [20], which differ in their normalization strategies and have distinct localization properties.

3.3 Bayesian Methods

Bayesian approaches model the source distribution as a random variable with a structured prior and optimize the model evidence (type-II maximum likelihood). The Champagne algorithm [8] and its variants form the largest subgroup, with seven update rules (MacKay, Convexity/MM, EM, AR-EM, TEM, LowSNR, Adaptive) and optional noise learning. Other Bayesian solvers include Gamma-MAP [21], Multiple Sparse Priors (MSP) [9], coherent maximum entropy on the mean (cMEM) [22], and Sparse Bayesian Learning (SBL) variants.

3.4 Beamformers

Beamformers construct spatial filters that pass signals from a target location while minimizing contributions from elsewhere. The linearly constrained minimum variance (LCMV) beamformer [7] solves:

$$\mathbf{w}_j = \frac{\mathbf{C}^{-1}\mathbf{l}_j}{\mathbf{l}_j^\top \mathbf{C}^{-1}\mathbf{l}_j}, \tag{5}$$

where $\mathbf{C}$ is the data covariance matrix and $\mathbf{l}_j$ is the leadfield column for source j. `invertmeeg` implements 25 beamformer variants, including DICS for frequency-domain analysis [23], synthetic aperture magnetometry (SAM) [24], eigenspace-projected variants (ESMV family), multiple-constraint approaches (MCMV), and the empirical Bayesian beamformer (EBB).

3.5 Matching Pursuit and Sparse Recovery

These methods greedily or iteratively select active sources from the leadfield dictionary. Orthogonal matching pursuit (OMP) [25] iteratively selects the leadfield column most correlated with the residual; CoSaMP [26] and subspace pursuit [27] extend this with two-stage selection and pruning. `invertmeeg` includes 10 matching pursuit variants, including FLEX variants that combine greedy selection with flexible subspace scanning, ReMBo (random embedding), and hierarchical subspace methods (SubSMP, ISubSMP).

3.6 MUSIC and Subspace Methods

Multiple signal classification (MUSIC) [11] localizes sources by scanning for leadfield columns that lie in the signal subspace. RAP-MUSIC [28] extends this with recursive projection to resolve multiple sources. `invertmeeg` provides 13 subspace methods: MUSIC, RAP-MUSIC, TRAP-MUSIC, and their FLEX variants, as well as signal subspace matching (SSM), alternating projections (AP), ExSo-MUSIC, and maximum-likelihood scanners (ExhaustiveML, GreedyML) that search for the source configuration maximizing the data likelihood.

3.7 Dipole Fitting

Equivalent current dipole (ECD) fitting models the source as one or more point dipoles whose positions and moments are optimized to fit the data. SESAME [29] uses a sequential Monte Carlo approach for multi-dipole estimation.

3.8 Deep Learning

`invertmeeg` includes six neural network architectures that learn the inverse mapping from simulated training data. These solvers require PyTorch as an optional dependency and use the simulation framework (Section 5) for on-the-fly training data generation—no pre-recorded data is needed. Three of these solvers (FC, CovCNN, CovCNN-KL) are included in the benchmark.

FC (ESINet) [12] is a two-layer fully connected network that maps sensor time series directly to source amplitudes. Each time point is processed independently through a dense layer ($n_{\text{channels}} \to n_{\text{units}}$, tanh activation), a second dense layer ($n_{\text{units}} \to n_{\text{units}}$), and a linear output layer ($n_{\text{units}} \to n_{\text{dipoles}}$). With 600 hidden units and 1,284 dipoles, this yields $\sim 1.2 \times 10^6$ trainable parameters.

CovCNN [13] operates on the sensor covariance matrix $\mathbf{C} = \mathbf{M}\mathbf{M}^\top \in \mathbb{R}^{n_{\text{ch}} \times n_{\text{ch}}}$ rather than the raw time series, providing implicit temporal averaging and sign invariance. The covariance is passed through a frozen convolutional layer whose weights are derived from the leadfield matrix, projecting the $n_{\text{ch}} \times n_{\text{ch}}$ input to n_{dipoles} feature maps. The resulting features are flattened and processed by a multi-layer perceptron (MLP) with tanh activations, producing per-dipole source estimates. CovCNN is trained with cosine similarity loss.

CovCNN-KL shares the CovCNN architecture but is trained with Kullback-Leibler divergence loss after applying a softmax to the output logits, treating source localization as a distribution-matching problem. This reformulation improves localization accuracy compared to the regression-based CovCNN. In the benchmark configuration (24 hidden units), both CovCNN variants have $\sim 1.0 \times 10^6$ trainable parameters; FC has $\sim 1.2 \times 10^6$. All three are trained with the Adam optimizer for 200 epochs with early stopping (patience 60) and a batch size of 2,048 simulated samples.

The remaining (non-benchmarked) architectures include CovCNN-KL-FlexOMP, which augments CovCNN-KL with iterative sparse post-processing; a bidirectional LSTM for temporal modeling; and a CNN with LSTM encoder for spatiotemporal processing.

3.9 Flexible Extent Estimation (FLEX)

A common limitation of classical inverse methods is the assumption that each source is a point dipole. In practice, cortical generators often extend over patches of varying size. The FLEX (Flexible Extent Estimation) framework [30] addresses this by augmenting any base solver with a multi-order leadfield dictionary that allows simultaneous estimation of source location and spatial extent.

The core idea is to construct smoothed versions of the leadfield matrix $\mathbf{L}$ via iterative diffusion on the cortical adjacency graph. Given the graph Laplacian $\mathbf{\Delta}$ of the source space, a smoothing operator is defined as:

$$\mathbf{S} = \mathbf{I} - \alpha_{\text{diff}} \cdot \mathbf{\Delta}, \tag{6}$$

where α_{diff} controls the diffusion rate. Applying $\mathbf{S}$ to the leadfield produces leadfields for increasingly extended sources:

$$\mathbf{L}^{(k)} = \mathbf{L} \cdot \mathbf{S}^k, \quad k = 0, 1, 2, \ldots, K, \tag{7}$$

where $k = 0$ corresponds to point dipoles and increasing k produces progressively smoother spatial profiles. For each order k, a gradient matrix $\mathbf{B}^{(k)}$ tracks how smoothed activations map back to individual dipoles in the original source space.

The extended leadfield dictionary $\mathbf{L}_{\text{ext}} = [\mathbf{L}^{(0)} \mid \mathbf{L}^{(1)} \mid \cdots \mid \mathbf{L}^{(K)}]$ contains $K \cdot n$ candidate columns (one per dipole per order). The base solver then operates on this extended dictionary and selects the best (location, order) pair for each source using its native selection criterion—subspace projection for MUSIC variants, residual correlation for matching pursuit, likelihood maximization for ML scanners, evidence maximization for Bayesian methods, or beamformer output power for spatial filters. A mutual exclusivity constraint ensures that at most one order is active per dipole location. The selected sources are mapped back to the original source space via their corresponding gradient matrices $\mathbf{B}^{(k)}$.

This mechanism is general: `invertmeeg` applies it to subspace methods (FLEX-MUSIC, FLEX-SSM, FLEX-GreedyML), matching pursuit algorithms (FLEX-OMP, FLEX-CoSaMP, FLEX-SP, FLEX-ReMBo), beamformers (the FLEX-ESMV family), and Bayesian solvers (Flex-Champagne, Flex-NL-Champagne). In all cases, the data-driven order selection allows each source to adopt the spatial extent that best explains the measurements, without requiring the user to specify source size a priori. The FLEX variants consistently outperform their fixed-order counterparts across all benchmark scenarios.

3.10 Hybrid Methods

`invertmeeg` includes three hybrid solvers that combine multiple algorithmic strategies to adapt to the characteristics of the data at hand.

Chimera adaptively selects between a beamformer (FLEX-ESMV) and a subspace scanner (SSM) based on properties estimated from the data. It first estimates the number of active sources $\hat{k}$ using the minimum description length (MDL) criterion [31] applied to the eigenvalues of the data covariance. If exactly one source is detected ($\hat{k} = 1$), Chimera computes an order-score ratio that compares the subspace fit of a point-dipole leadfield against smoothed (extended) leadfields: if the ratio indicates a focal source, the beamformer branch is used; otherwise, SSM handles multi-source or extended configurations. This two-stage decision rule allows Chimera to route each data sample to the solver best suited to its source geometry.

Hydra combines a flexible maximum-likelihood scanner (FLEX-GreedyML) with an adaptive beamformer (Adapt-FLEX-ESMV), switching between them based on a variance-based heuristic. The mean sensor variance of the input data is compared against a calibrated threshold: high-variance data (characteristic of distributed or ongoing activity) is routed to the beamformer, while lower-variance data (characteristic of sparse focal sources) is processed by FLEX-GreedyML. This simple switching mechanism allows Hydra to achieve a top-five overall rank across all benchmark scenarios (mean rank 11.0), as it leverages the complementary strengths of subspace scanning (excellent for sparse multi-source recovery) and beamforming (robust to distributed interference).

APSE (Alternating Projections with Source Enumeration) integrates subspace scanning with automatic model-order selection via information-theoretic criteria.

3.11 Novel Solvers Developed with AI Coding Agents

Beyond re-implementing published algorithms, `invertmeeg` includes over 30 solvers that have not appeared in prior publications. These methods were developed by the first author in collaboration with AI coding agents (large language models used as interactive programming assistants) and represent novel combinations and extensions of existing algorithmic components. Families

of novel solvers include FLEX-ESMV beamformer variants, Flex-Champagne Bayesian solvers, hybrid switching methods (Chimera, Hydra, APSE), subspace matching pursuit variants, and all deep learning solvers (CovCNN, CovCNN-KL, CNN, LSTM). Each novel solver is documented with its component references in the solver source code and in the appendix tables.

As an illustrative example, we describe **Subspace-SBL**—the top-ranked Bayesian solver and overall rank 2 in our benchmark—in detail. Subspace-SBL decomposes the inverse problem into two stages: a combinatorial source detection stage followed by a continuous amplitude estimation stage on the detected support.

Stage 1: Subspace detection. Given sensor data $\mathbf{Y} \in \mathbb{R}^{m \times T}$, we first form a regularized signal-subspace projector via signal subspace matching (SSM):

$$\mathbf{P}_Y = \mathbf{Y}\left(\mathbf{Y}^\top \mathbf{Y} + \lambda_1 \operatorname{tr}(\mathbf{Y}^\top \mathbf{Y})\, \mathbf{I}_T\right)^{-1} \mathbf{Y}^\top, \qquad \mathbf{C} = \mathbf{P}_Y^\top \mathbf{P}_Y, \tag{8}$$

where λ_1 is a small regularization constant. The matrix $\mathbf{C}$ captures the signal subspace robustly even under model errors. Sources are then detected greedily from the multi-order FLEX dictionary $\{\mathbf{L}^{(0)}, \ldots, \mathbf{L}^{(K)}\}$. At each step q, we select the (location, order) pair that maximizes the projected subspace score:

$$(d^*, k^*) = \arg\max_{d,k} \frac{\mathbf{a}_{d,k}^\top \mathbf{C}\, \mathbf{a}_{d,k}}{\mathbf{a}_{d,k}^\top \mathbf{a}_{d,k}}, \qquad \text{where } \mathbf{a}_{d,k} = (\mathbf{I} - \mathbf{P}_A)\, \ell_d^{(k)}, \tag{9}$$

and $\mathbf{P}_A$ is the (regularized) projector onto the subspace spanned by previously selected leadfield columns. After the greedy forward pass, a cyclic refinement procedure revisits each selected source and re-optimizes its (location, order) assignment while holding the others fixed, iterating until convergence.

Stage 2: Noise-learning SBL refinement. Let $\mathbf{L}_{\text{sel}} \in \mathbb{R}^{m \times \hat{k}}$ denote the leadfield columns corresponding to the $\hat{k}$ detected sources. We model the data as $\mathbf{Y} = \mathbf{L}_{\text{sel}}\mathbf{S} + \mathbf{N}$, with a Gaussian prior $\mathbf{s}_t \sim \mathcal{N}(\mathbf{0}, \operatorname{diag}(\gamma))$ and heteroscedastic sensor noise $\mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, \operatorname{diag}(\lambda))$. The model covariance is $\mathbf{\Sigma}_y = \mathbf{L}_{\text{sel}} \operatorname{diag}(\gamma) \mathbf{L}_{\text{sel}}^\top + \operatorname{diag}(\lambda)$. Source variance and noise parameters are estimated by iterating Type-II ML fixed-point updates:

$$\gamma_i \leftarrow \sqrt{\frac{\sum_t \bar{s}_{i,t}^2}{[\mathbf{L}_{\text{sel}}^\top \mathbf{\Sigma}_y^{-1} \mathbf{L}_{\text{sel}}]_{ii}}}, \qquad \lambda_j \leftarrow \sqrt{\frac{\sum_t (y_{j,t} - \hat{y}_{j,t})^2}{[\mathbf{\Sigma}_y^{-1}]_{jj}}}, \tag{10}$$

where $\bar{\mathbf{S}} = \operatorname{diag}(\gamma) \mathbf{L}_{\text{sel}}^\top \mathbf{\Sigma}_y^{-1} \mathbf{Y}$ is the posterior mean and $\hat{\mathbf{Y}} = \mathbf{L}_{\text{sel}} \bar{\mathbf{S}}$ the predicted data. Sources with $\gamma_i < 10^{-3} \max(\gamma)$ are pruned, and the algorithm converges when the relative change in the Type-II ML objective falls below 10^{-8}. The final inverse operator is $\mathbf{K} = \mathbf{B}^\top \operatorname{diag}(\gamma) \mathbf{L}_{\text{sel}}^\top \mathbf{\Sigma}_y^{-1}$, where $\mathbf{B}$ contains the FLEX gradient vectors that map detected atoms back to the full source space.

Subspace-SBL+ extends this by expanding the SSM-detected candidate set with graph-neighborhood atoms (one hop in the cortical adjacency graph) before running the SBL refinement. This local expansion addresses SSM's sensitivity to vertex-level misalignment: the Bayesian refinement automatically prunes spurious candidates while retaining those that genuinely explain data variance.

The design philosophy behind all novel solvers follows a common pattern: leverage the complementary strengths of fast geometric methods (subspace scanning, beamforming) for discrete

decisions and statistically principled methods (SBL, maximum likelihood) for continuous estimation. A dedicated paper on the agentic development process—describing how AI coding agents were used to systematically explore the combinatorial space of algorithmic components, identify promising combinations, and iteratively refine solver implementations against benchmark feedback—is in preparation.

4 Regularization

Many inverse solvers require a regularization parameter α that balances data fit against solution complexity. Choosing α appropriately is critical: too little regularization amplifies noise, while too much regularization oversmooths the solution. `invertmeeg` provides a shared regularization framework that is used by all solvers that produce linear inverse operators.

4.1 Regularization Parameter Grid

When `alpha="auto"` (the default), the solver constructs a grid of candidate values. A set of dimensionless knobs r_i is logarithmically spaced in $[10^{-10}, 10^{1}]$, and each is scaled by the largest eigenvalue $\lambda_{\max}$ of a reference matrix (by default $\mathbf{L}\mathbf{L}^\top$):

$$\alpha_i = r_i \cdot \lambda_{\max}(\mathbf{L}\mathbf{L}^\top). \tag{11}$$

This scaling ensures that the regularization grid is adapted to the magnitude of the forward model, making the same r-values meaningful across different sensor configurations and source spaces. For each α_i, a separate inverse operator $\mathbf{W}_i$ is precomputed and stored.

4.2 Generalized Cross-Validation (GCV)

GCV [32] selects the α that minimizes a leave-one-out prediction error estimate without explicit knowledge of the noise level. For a linear inverse operator that produces the hat matrix $\mathbf{H}_\alpha = \mathbf{L}\mathbf{W}_\alpha$, the GCV criterion is:

$$\mathrm{GCV}(\alpha) = \frac{\|\mathbf{m} - \mathbf{H}_\alpha \mathbf{m}\|^2}{(n - \gamma \cdot \mathrm{tr}(\mathbf{H}_\alpha))^2}, \tag{12}$$

where n is the number of channels and $\gamma \geq 1$ is a correction factor. Setting $\gamma = 1$ gives standard GCV; values slightly above 1 (modified GCV, MGCV) bias the selection toward stronger regularization, which can be helpful when the forward model is rank-deficient.

For iterative Bayesian solvers such as MacKay-Champagne and Convexity-Champagne, we found that standard GCV frequently selects an insufficiently regularized inverse operator, leading to degraded localization quality. This occurs because the Champagne update rules refine both source priors and the noise model iteratively; when initialized with a poorly regularized inverse, the algorithm can converge to a suboptimal local maximum of the model evidence. To address this, we use the strong robust GCV (R1GCV) variant [33]:

$$\mathrm{R1GCV}(\alpha) = \big[\gamma + (1-\gamma) \cdot \mu_{12}(\alpha)\big] \cdot \mathrm{GCV}(\alpha), \quad \mu_{12}(\alpha) = \frac{\mathrm{tr}(\mathbf{H}_\alpha) - \mathrm{tr}(\mathbf{H}_\alpha^2)}{n \cdot \alpha}, \tag{13}$$

where μ_{12} is the negative derivative of the mean leverage $\mathrm{tr}(\mathbf{H}_\alpha)/n$ with respect to α. The weighting factor penalizes regions of the regularization grid where solution variance amplifies rapidly as α decreases, biasing the selection toward more stable operating points. On real somatosensory data, R1GCV selected the same regularization parameter as the oracle optimum, whereas standard GCV selected a parameter three orders of magnitude too small.

4.3 L-Curve Method

The L-curve method [34] plots the solution norm $\|\hat{\mathbf{s}}\|_2$ against the residual norm $\|\mathbf{L}\hat{\mathbf{s}} - \mathbf{m}\|_2$ for each candidate α. The optimal α is located at the "corner" of the resulting L-shaped curve, which represents the best trade-off between data fit and solution regularity. `invertmeeg` detects the corner using a triangle-area heuristic: the candidate that maximizes the area of the triangle formed with the two endpoints of the curve is selected.

4.4 Product Method

The product method [3] selects the α that minimizes the product of the solution norm and the residual norm:

$$\alpha^* = \arg\min_{\alpha} \|\hat{\mathbf{s}}_\alpha\|_2 \cdot \|\mathbf{L}\hat{\mathbf{s}}_\alpha - \mathbf{m}\|_2. \tag{14}$$

This criterion does not require computing the hat matrix and is therefore computationally cheaper than GCV.

5 Simulation Methodology

A meaningful comparison of inverse solvers requires synthetic data with known ground truth that spans the diversity of source configurations encountered in practice. Existing simulation approaches often rely on simple Gaussian patches centered on fixed cortical locations, which produces unrealistically regular spatial patterns and may favor solvers whose assumptions match these idealized geometries. We designed a simulation framework that generates more diverse and challenging source configurations through two complementary spatial models, structured sensor noise, and flexible temporal dynamics.

5.1 Spatial Source Models

`invertmeeg` provides two spatial source models that produce qualitatively different source geometries.

Diffusion basis patches. The first model constructs source patterns using a graph-based spatial basis derived from the cortical mesh adjacency. Given the adjacency matrix $\mathbf{A}$ of the source space, the diffusion-based gradient operator is:

$$\mathbf{G} = \mathbf{I} - \alpha_{\text{diff}} \cdot \mathbf{\Delta}(\mathbf{A}), \tag{15}$$

where $\mathbf{\Delta}(\mathbf{A})$ is the graph Laplacian and α_{diff} controls the smoothing strength. Multi-order spatial patterns are generated by iteratively applying $\mathbf{G}$ to a seed dipole, where the spatial order parameter k controls source extent: $k = 0$ produces point dipoles, $k = 1$–2 produces cortical patches, and $k \geq 3$ produces broadly distributed activity.

Contiguous Gaussian patches. The second model generates spatially extended sources with irregular, anatomically plausible geometries. For each source, a contiguous region of a specified size is grown from a random seed vertex using randomized breadth-first search (BFS) on the cortical adjacency graph. At each step, a random vertex is selected from the frontier (the set of unvisited neighbors of the current region), producing irregular region shapes that conform to the cortical surface topology rather than forming symmetric discs.

Within each region, one or two spatial components are created with Gaussian-weighted profiles. For each component, a center vertex c is chosen and the graph hop distance d_v from c to every vertex v in the region is computed via BFS restricted to the region. The spatial weights follow a Gaussian kernel:

$$w_v = \exp\left(-\frac{d_v^2}{2\sigma^2}\right), \tag{16}$$

where σ controls the spatial smoothness (in graph-hop units). For rank-2 patches, a second component with a different center produces two overlapping smooth maps within the same region, yielding more complex spatial patterns.

This approach produces sources that are (1) contiguous and connected on the cortical surface, (2) irregularly shaped due to the randomized BFS growth, and (3) smoothly varying within the patch rather than sharply bounded. These properties make contiguous Gaussian patches substantially more diverse than standard Gaussian blobs and more representative of the spatially extended generators observed in electrophysiological recordings of oscillatory or evoked activity.

5.2 Temporal Dynamics

Source time courses are drawn from a pre-generated pool of colored noise signals with power spectral density $S(f) \propto 1/f^\beta$, where β is uniformly sampled from a configured range. This produces time courses ranging from white noise ($\beta = 0$) to smooth, slowly varying signals ($\beta = 3$). Inter-source correlations are introduced via Cholesky decomposition of a correlation matrix, allowing control over the degree of temporal dependence between simultaneously active sources.

5.3 Sensor Noise Model

Sensor-level noise is added at a specified signal-to-noise ratio (SNR in dB). For the benchmark evaluation, we use a "realistic" noise model that combines three components to approximate the structure of empirical M/EEG noise:

$$\mathbf{C}_{\text{noise}} = \mathbf{A}\mathbf{A}^\top + \alpha \cdot \mathbf{C}_{\text{banded}} + \sigma^2\mathbf{I}, \tag{17}$$

where $\mathbf{A} \in \mathbb{R}^{m\times r}$ is a random basis generating r low-rank artifact-like components (simulating environmental interference or physiological artifacts), $\mathbf{C}_{\text{banded}}$ captures spatial correlations between neighboring channels (with elements $[\mathbf{C}_{\text{banded}}]_{ij} = \rho^{|i-j|}$ for a coefficient ρ), and $\sigma^2\mathbf{I}$ adds a white noise floor. All components are trace-normalized before combination to keep their relative contributions interpretable. The low-rank dimension r is randomly drawn from a configured range (typically 2–8), introducing variability in the noise structure across simulations.

This structured noise model is more challenging than i.i.d. Gaussian noise because it introduces spatial correlations and low-rank interference patterns that can bias inverse solutions toward artifactual source configurations, particularly for methods that assume spatially white noise.

5.4 Simulation Modes

Source activity in the benchmark consists of the selected source patches, representing the common assumption that a small number of discrete cortical generators are active. The framework additionally supports a mixture mode that combines sparse patches with smooth spatiotemporal background activity, but this mode is not used in the current benchmark.

6 Evaluation and Benchmarking

`invertmeeg` provides a standardized evaluation suite for comparing solver performance on simulated data with known ground truth.

6.1 Metrics

The following metrics are computed between the true source distribution $\mathbf{y}_{\text{true}}$ and the estimated distribution $\mathbf{y}_{\text{pred}}$. We designate EMD as the primary metric and AP as the secondary metric for ranking solvers, as they capture complementary aspects of reconstruction quality.

Earth Mover's Distance (EMD, primary metric). The Wasserstein-1 distance between the true and predicted source distributions, computed using optimal transport [35]. The EMD quantifies the minimum "work" required to transform one distribution into the other, jointly capturing localization accuracy, amplitude matching, and spatial extent. Unlike point-based metrics (e.g., MLE), the EMD is well-defined for both focal and extended sources and penalizes both missed sources and false positives in a geometrically meaningful way.

Average Precision (AP, secondary metric). Average precision summarizes the precision-recall curve and quantifies how well the solver identifies the correct source locations across all detection thresholds. We prefer AP over the more commonly reported area under the ROC curve (AUC) because source imaging is an inherently imbalanced classification problem: in a source space with $n = 1{,}284$ dipoles, a focal source activates only 1 dipole ($< 0.1\%$ positive rate), and even extended sources rarely exceed 3–5% of the source space. Under such extreme class imbalance, AUC is dominated by the true negative rate and can assign high scores to reconstructions that spread activity broadly, whereas AP focuses exclusively on the precision among predicted positives and is therefore more sensitive to false activations [36].

Mean Localization Error (MLE). The Euclidean distance between the centers of mass of the true and predicted source distributions, averaged over all active sources. MLE is most interpretable for focal sources but can be misleading for extended or multi-source configurations where center-of-mass comparisons lose meaning.

Correlation. The Pearson correlation between the true and predicted source time courses, evaluated at the true source locations. This metric captures temporal fidelity but is insensitive to spatial errors outside the true source region.

Spatial Dispersion (SD). A measure of the spatial spread of the reconstructed source around the true source location. Higher dispersion indicates more spatial blurring in the reconstruction.

6.2 Evaluation Scenarios

Different inverse methods are suited to different source configurations. We define four evaluation scenarios that span the range of conditions encountered in practice (Table 2):

1. **Focal Source**: A single point dipole at moderate-to-high SNR (3–5 dB). This baseline scenario tests pure localization accuracy and favors sparse/dipolar methods.

2. **Multi Source**: 2–4 simultaneously active focal dipoles at low-to-moderate SNR (1–3 dB). Tests the ability to resolve multiple concurrent sources without mutual interference.
3. **Extended Source**: 2–4 spatially extended contiguous Gaussian patches (spatial orders 1–3) at moderate SNR (1–3 dB). Uses the contiguous Gaussian patch model (Section 5.1) with rank-1 or rank-2 temporal dynamics within each patch, producing irregularly shaped cortical activations.
4. **Noisy**: 1–3 sources with variable spatial extent (orders 0–2) under challenging low-SNR conditions (−5 to 0 dB). Tests robustness to noise, which is the primary practical challenge in clinical and cognitive applications.

All four scenarios use the "realistic" noise model (Section 5.3) with a low-rank artifact dimension drawn uniformly from $[2, 8]$.

Table 2: Configuration of the four evaluation scenarios. All scenarios use realistic structured sensor noise and 50 simulations each.

Parameter	**Focal**	**Multi**	**Extended**	**Noisy**
Sources	1	[2, 4]	[2, 4]	[1, 3]
Source type	Point dipole	Point dipole	Gauss. patch	Diffusion patch
Diffusion order	0	0	[1, 3]	[0, 2]
SNR (dB)	[3.0, 5.0]	[1.0, 3.0]	[1.0, 3.0]	[-5.0, 0.0]

6.3 Experimental Setup

We evaluated 106 solvers—all classical methods plus three deep learning solvers (FC, CovCNN, CovCNN-KL) and the random-noise baseline—on all four scenarios using a BioSemi-32 electrode montage with an `ico3` source space comprising $n = 1{,}284$ dipoles with fixed orientation. Each scenario was tested with 50 independent simulations (random seed 42 for reproducibility), yielding 200 simulation instances per solver.

For each simulation, a noise covariance matrix is estimated from a separate 200-time-point baseline segment drawn independently from the same noise model (spatial covariance and temporal coloring) as the noise added to the signal. This simulates the common experimental practice of estimating noise statistics from an empty-room recording or pre-stimulus baseline. The empirical covariance is regularized with a fixed shrinkage toward a scaled identity ($\gamma = 0.05$). Each solver receives this estimated noise covariance and uses it to spatially whiten both the leadfield and the data via eigendecomposition-based whitening: the noise covariance is decomposed as $\mathbf{C}_n = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, and the whitener $\mathbf{W} = \mathbf{\Lambda}^{-1/2}\mathbf{V}^\top$ is applied, with rank truncation to avoid amplifying near-zero-variance directions. This ensures that all solvers operate on data with approximately identity noise covariance, placing them on equal footing regardless of the original noise structure.

The `BenchmarkRunner` class automates the evaluation pipeline: it generates simulated data for each scenario, applies each solver with its recommended depth-weighting parameters, computes all metrics, and aggregates results.

6.4 Benchmark Results

Table 3 shows the earth mover's distance (EMD) across all four scenarios for all 106 evaluated solvers, sorted by mean EMD. Detailed per-scenario results are provided in the Appendix (Tables 5

to 8).

Overall ranking. The top-performing solver across all scenarios is **FLEX-GreedyML** (mean rank 7.0), followed by **Subspace-SBL+** (7.5), **Subspace-SBL** (8.5), **FLEX-SSM** (11.0), and **Hydra** (11.0). Notably, three of the top five solvers belong to the MUSIC/subspace family, underscoring the strength of subspace methods when the number of sources is moderate. The Bayesian Subspace-SBL variants occupy ranks two and three, demonstrating the effectiveness of structured priors combined with subspace information.

Focal and multi-source scenarios. For focal source localization, 51 solvers achieve a perfect or near-perfect mean EMD below 5 mm, including all subspace methods, most Bayesian solvers, matching pursuit methods, and several beamformers. The differentiation occurs in the multi-source scenario, where **GreedyML** leads (EMD 8.7 mm), followed by **FLEX-GreedyML** and **Hydra** (9.1 mm), **ExhaustiveML** (11.0 mm), and **Subspace-SBL+** (12.0 mm).

Extended source scenario. For spatially extended sources, **Subspace-SBL+** leads (EMD 22.6 mm), followed closely by **FLEX-GreedyML** and **Hydra** (22.9 mm), **FLEX-SSM** (26.2 mm), and **Subspace-SBL** (26.5 mm). MSP—which employs a multiple sparse priors basis set—ranks well (rank 9, AP 0.31), demonstrating that methods with distributed priors can excel when sources are genuinely extended. As expected, sparse methods (OMP, CoSaMP) and methods with strong sparsity assumptions struggle in this scenario.

Noisy scenario. Under low-SNR conditions, **Subspace-SBL** achieves the best EMD (9.3 mm), closely followed by **Chimera** (9.7 mm), **FLEX-GreedyML** and **Hydra** (10.5 mm), and **FLEX-AP** and **FLEX-SSM** (10.9 mm). Bayesian methods generally maintain their performance better than beamformers under noise, consistent with the regularizing effect of their probabilistic priors.

Deep learning solvers. Three neural network solvers were included in the benchmark: CovCNN-KL, CovCNN, and FC. CovCNN-KL achieves the best performance among the deep learning methods (mean rank 70.5, mean EMD 44.5 mm), placing it in the range of mid-tier classical beamformers such as DICS and LCMV. CovCNN (mean rank 79.5, EMD 52.0 mm) and FC (mean rank 90.5, EMD 55.6 mm) perform similarly to minimum norm methods. Notably, all three neural network solvers outperform the random-noise baseline and several classical methods, but fall substantially short of the top-performing classical solvers. The benchmark used reduced model capacity (24 dense units, 200 training epochs) to keep training tractable within the evaluation framework, resulting in approximately 10^6 trainable parameters per model. With longer training and larger networks, performance may improve; however, even with these conservative settings, the computational cost is orders of magnitude higher than classical methods: CovCNN-KL requires $\sim$1,961 s per sample (training included) compared to $\sim$2 s for FLEX-SSM.

Table 3: Earth mover's distance (EMD in mm↓, lower is better) across all four evaluation scenarios for all 106 evaluated solvers, sorted by mean EMD. Average inference time in seconds (excludes training for neural network solvers). Best value per scenario in **bold**.

Solver	Cat.	Focal	Multi	Extended	Noisy	s	Solver	Cat.	Focal	Multi	Extended	Noisy	s
FLEX-GreedyML	MUS	**0.0**	9.1	22.9	10.5	134	Gamma-MAP-MSP	Bay	18.5	42.9	27.9	27.5	27
Hydra	Oth	**0.0**	9.1	22.9	10.5	136	ECD	Oth	**0.0**	45.1	43.8	29.1	1.3
Subspace-SBL+	Bay	**0.0**	12.0	**22.6**	11.5	3.8	LCMV-MVPURE	BF	11.5	38.8	38.3	29.4	1.5
Subspace-SBL	Bay	**0.0**	14.8	26.5	**9.3**	2.2	SP	MP	13.3	36.4	38.2	30.3	1.3
Chimera	Oth	**0.0**	14.4	26.6	9.7	2.7	FLEX-CoSaMP	MP	8.4	42.4	40.1	29.5	1.5
FLEX-SSM	MUS	**0.0**	14.6	26.2	10.9	1.8	CoSaMP	MP	8.4	40.4	41.0	31.0	1.5
GreedyML	MUS	**0.0**	**8.7**	27.2	16.8	28	ReMBo	MP	4.4	39.8	42.2	36.8	3.1
FLEX-AP	MUS	**0.0**	15.2	27.6	10.9	1.9	FLEX-ReMBo	MP	4.7	40.1	45.5	38.1	3.6
ExhaustiveML	MUS	**0.0**	11.0	28.5	17.0	23	ESMV-MVPURE	BF	22.6	41.1	40.4	32.3	1.8
FLEX-MUSIC	MUS	**0.0**	14.4	30.2	11.9	1.8	MUSIC	MUS	31.9	40.3	36.1	37.6	3.7
FLEX-TRAP-MUSIC	MUS	**0.0**	15.6	30.6	11.7	1.8	Source-MAP	Bay	34.2	41.2	34.3	39.7	25
Flex-NL-Champagne	Bay	**0.0**	19.2	27.9	14.4	8.9	Block-Champagne	Bay	36.5	42.7	35.0	51.2	26
Flex-NL-Champ.-Omix	Bay	**0.0**	20.2	27.2	14.5	9.7	L1/L2	MN	25.3	44.3	42.7	56.9	26
AP	MUS	**0.0**	14.4	31.3	17.8	1.6	SR-eLORETA	LOR	46.2	42.1	38.0	43.9	7.1
SSM	MUS	**0.0**	13.9	32.4	17.7	1.4	Champ.-MacKay-NL-F	Bay	18.9	45.2	48.2	61.0	14
Champ.-Conv.	Bay	0.3	16.6	27.9	20.2	126	DICS	BF	21.9	51.0	48.4	52.2	2.8
Champ.-MacKay	Bay	0.4	16.6	27.9	20.1	71	CovCNN-KL	NN	45.6	50.1	41.5	40.7	3.5
Champ.-LowSNR	Bay	0.4	15.8	28.1	20.9	126	MCE	MN	38.5	46.0	42.5	56.3	85
Champ.-MacKay-NL-H	Bay	1.8	14.8	28.2	20.8	7.4	Champ.-Conv.-NL-F	Bay	34.7	44.2	48.6	61.7	22
Champ.-Conv.-NL-H	Bay	2.0	15.2	28.3	20.9	12	HOCMCMV	BF	48.9	45.5	42.0	53.5	6.4
NL-Champagne	Bay	1.4	15.8	27.9	22.5	12	HOCMV	BF	48.9	45.5	42.0	53.5	6.4
RAP-MUSIC	MUS	**0.0**	12.9	35.8	19.5	1.4	ReciPSIICOS	BF	48.4	47.5	43.9	54.2	12
Champ.-MacKay-NL-P	Bay	1.9	16.2	28.4	21.8	7.1	SSLOFO	MN	45.2	46.6	50.7	52.0	990
Champ.-MacKay-NL-D	Bay	1.9	16.2	28.4	21.8	9.0	Unit-Noise-Gain	BF	51.2	47.2	43.8	55.0	7.2
Flex-Champagne	Bay	**0.0**	23.6	29.5	16.2	4.6	WNMV	BF	51.2	47.2	43.8	55.0	7.1
Champ.-Conv.-NL-P	Bay	2.3	16.5	28.7	22.1	12	SAM	BF	52.5	47.4	43.8	53.7	7.2
Champ.-Conv.-NL-D	Bay	2.3	16.5	28.7	22.1	11	SubSMP	MP	48.9	52.6	46.4	49.9	2.6
Omni-Champagne	Bay	**0.0**	22.1	30.7	17.1	7.1	MVAB	BF	51.9	47.6	44.0	55.0	7.1
BCS	Bay	3.7	16.0	28.5	21.8	87	SMV	BF	53.4	48.7	45.3	55.8	7.3
TRAP-MUSIC	MUS	**0.0**	15.0	36.2	19.4	1.4	CovCNN	NN	59.9	52.5	43.6	52.1	4.8
Flex-Champ.-Omix	Bay	**0.0**	24.0	29.9	16.8	4.8	MCMV	BF	56.2	50.1	46.3	57.6	10
IR-ESMV	BF	**0.0**	27.1	29.0	21.5	1.7	LCMV	BF	57.2	49.5	46.3	57.9	7.1
SSP-IR-ESMV	BF	**0.0**	28.4	29.3	21.8	1.8	GFT-MNE	MN	60.2	51.3	46.5	56.1	9.3
Safe-Flex-ESMV	BF	**0.0**	27.2	34.2	21.1	3.0	EPIFOCUS	MN	60.2	51.2	47.1	55.9	7.2
EXSO-MUSIC	MUS	11.4	22.2	29.3	20.2	3.4	sLORETA	LOR	60.4	50.9	47.6	56.9	10
ESMV	BF	**0.0**	29.4	30.3	24.2	2.1	LAURA	MN	60.6	52.5	47.2	56.5	7.4
MxNE-BCD	MN	2.7	23.9	29.5	28.7	25	wMNE	MN	60.9	51.8	47.4	57.1	8.6
Deblur-Flex-ESMV	BF	**0.0**	30.1	35.1	20.4	3.2	FC (ESINet)	NN	61.7	52.2	49.1	59.5	5.1
Flex-ESMV	BF	**0.0**	30.1	35.2	20.4	2.5	Source-MAP-MSP	Bay	68.7	59.3	36.7	59.9	30
Adapt-Flex-ESMV	BF	**0.0**	28.5	34.1	26.2	3.2	GFT-L1	MN	63.1	53.7	50.5	60.9	81
OMP	MP	**0.0**	31.2	36.2	21.7	1.8	cMEM	Bay	61.8	59.1	54.5	62.7	30
Sharp-Flex-ESMV	BF	**0.0**	28.8	35.3	26.3	3.0	dSPM	MN	69.5	56.3	53.5	63.7	6.0
Gamma-MAP	Bay	0.1	35.6	28.8	26.9	18	APE-SISSY	MN	72.8	60.9	42.6	68.5	34
EBB	BF	**0.0**	30.1	38.5	25.9	9.4	GBF	MN	73.3	53.5	57.9	63.4	5.0
Flex-ESMV-MVPURE	BF	**0.0**	29.8	41.9	26.2	1.9	eLORETA	LOR	74.9	53.8	58.5	62.0	5.8
SESAME	Oth	0.4	34.5	40.2	23.5	2.0	LORETA	LOR	71.8	57.8	56.2	64.0	61
SSP-ESMV	BF	0.6	38.7	34.3	25.8	1.2	S-MAP	MN	71.8	57.8	56.2	64.0	39
APSE	Oth	13.2	30.5	35.3	26.5	4.8	TV	MN	69.0	56.9	63.9	63.5	392
MSP	Bay	21.4	28.5	27.3	30.1	2.9	SISSY	MN	76.4	62.4	46.0	68.6	34

7 Usage Examples

7.1 Basic Source Imaging

The following example demonstrates the core workflow:

```
import mne
from invert import Solver

# Load forward model and data
forward = mne.read_forward_solution("sample-fwd.fif")
evoked = mne.read_evokeds("sample-ave.fif")[0]

# Create solver and compute inverse operator
solver = Solver("eloreta")
solver.make_inverse_operator(forward, alpha="auto")

# Apply to data
stc = solver.apply_inverse_operator(evoked)

# Visualize
stc.plot(subject="fsaverage", hemi="both")
```

Figure 3 shows glass brain visualizations of a simulated ground truth and reconstructions from five representative solvers, illustrating the different localization characteristics of each method family.

Switching to any other solver requires changing only the name string:

```
for method in ["mne", "dspm", "lcmv", "champagne", "music"]:
    solver = Solver(method)
    solver.make_inverse_operator(forward, alpha="auto")
    stc = solver.apply_inverse_operator(evoked)
```

7.2 Simulation and Benchmarking

```
from invert.simulate import SimulationConfig, SimulationGenerator

# Configure simulation
config = SimulationConfig(
    n_sources=(1, 5),
    n_orders=(0, 2),
    snr_range=(-5, 5),
    batch_size=100,
)

# Generate data
gen = SimulationGenerator(forward, config)
x, y, info = next(gen.generate())

# x: sensor data [100, n_channels, n_timepoints]
# y: ground truth [100, n_dipoles, n_timepoints]
# info: metadata DataFrame
```

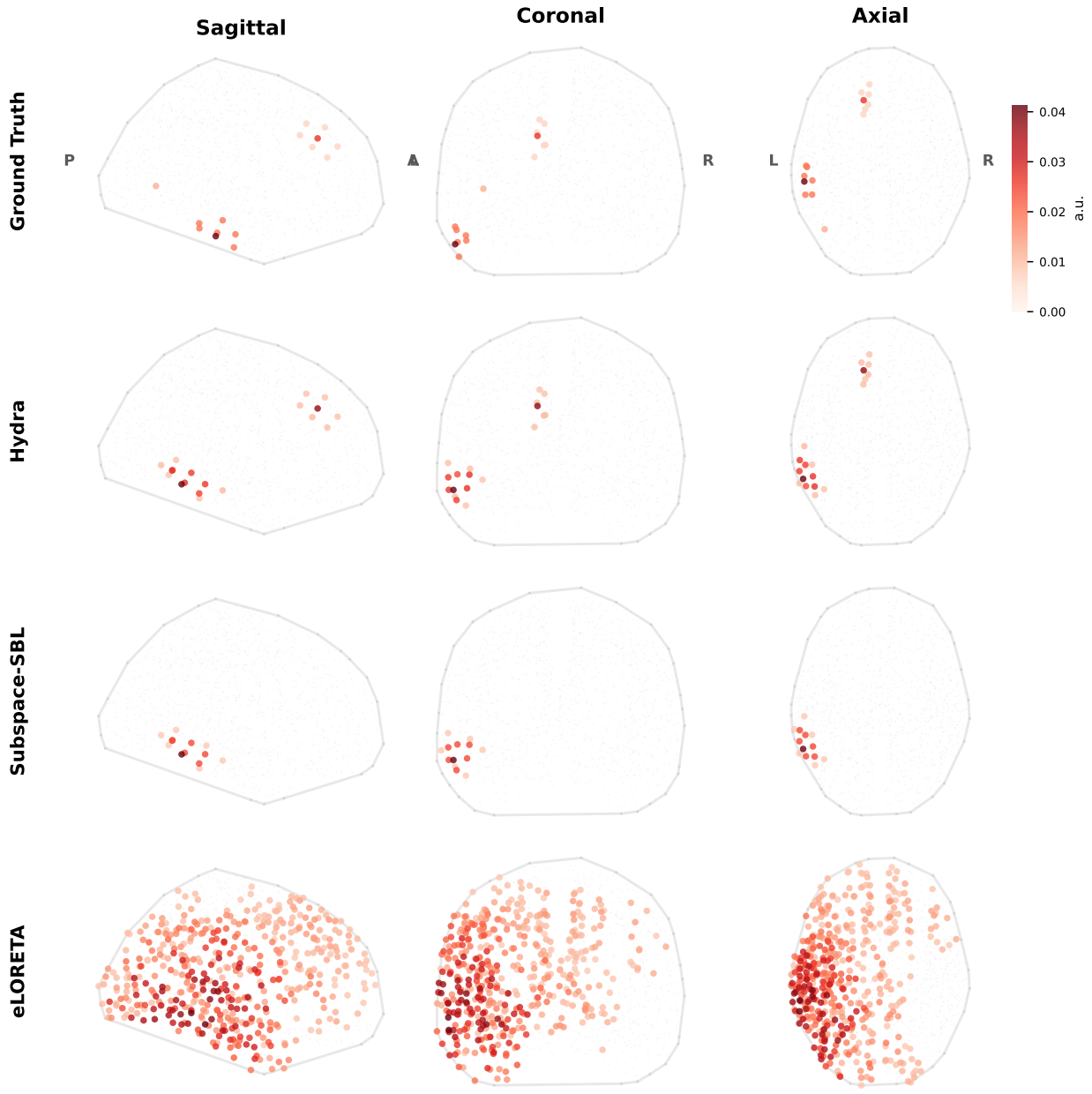

Figure 3: Glass brain projections of simulated source activity (ground truth, top row) and reconstructions by three solvers spanning different method families: Hydra (hybrid), Subspace-SBL (Bayesian), and eLORETA (minimum norm). Green crosses mark ground-truth source locations. The simulation contains three cortical patch sources at 3 dB SNR. Hydra and Subspace-SBL accurately localize all three sources, while eLORETA produces a diffuse, spatially blurred estimate.

8 Discussion

8.1 Practical Guide to Solver Selection

A key goal of this benchmark is to provide actionable guidance for practitioners. Based on the results across all four scenarios, we offer the following recommendations, organized by the expected source configuration. Table 4 summarizes these recommendations.

Table 4: Solver selection guide based on expected source configuration and computational budget. Inference times on a single CPU core (BioSemi-32, ico3 source space).

Scenario	**Recommended (fast, < 5 s)**	**Recommended (best, any time)**
Focal dipole(s)	FLEX-SSM, FLEX-MUSIC, OMP	FLEX-GreedyML, Omni-Champagne
Multiple sources	FLEX-SSM, SSM, Chimera	FLEX-GreedyML, Hydra
Extended patches	FLEX-SSM, FLEX-MUSIC	Subspace-SBL+, FLEX-GreedyML
Low SNR	Subspace-SBL, FLEX-SSM	Subspace-SBL, Chimera
Unknown/general	FLEX-SSM, Chimera	FLEX-GreedyML

When sources are expected to be focal. Many solvers achieve near-perfect performance on focal sources, so the differentiator is computational cost. Subspace methods (FLEX-MUSIC, FLEX-SSM, SSM) and matching pursuit methods (OMP) are fast (∼1–2 s) and highly accurate. Bayesian methods (Omni-Champagne, Flex-Champagne) achieve slightly better correlation but at higher computational cost (∼3–5 s).

When multiple sources are expected. Resolving multiple concurrent sources is the core strength of subspace scanning methods. FLEX-GreedyML and Hydra consistently lead, with SSM and ExhaustiveML close behind. Classical beamformers struggle in this regime due to signal cancellation between correlated sources.

When sources are spatially extended. Extended sources violate the sparsity assumption of many methods. Subspace-SBL+ and Subspace-SBL, which learn flexible source priors, excel here. Gamma-MAP-MSP, which employs a multiple sparse priors basis, also performs well (AP = 0.45). Pure sparse methods (OMP, CoSaMP, MCE) should be avoided.

When SNR is low. Subspace-SBL stands out under noisy conditions, consistently achieving the best rank across all metrics. Bayesian methods with noise learning (Champagne variants with homoscedastic or diagonal noise estimation) maintain reasonable performance. Beamformers and minimum norm methods degrade substantially as they lack the regularizing effect of structured source priors.

When no prior information is available. For a general-purpose solver that performs reasonably across all scenarios, **FLEX-GreedyML** (mean global rank 7.0) offers the best overall performance but at a higher computational cost (∼134 s per sample). For a fast default, **FLEX-SSM** (∼1.8 s) or **Chimera** (∼2.7 s) provide a good balance of performance and speed. We caution

against using minimum norm methods (MNE, eLORETA) as default choices: despite their popularity, they rank in the bottom quarter across most scenarios (global ranks 98–103 out of 106).

Inference time. Figure 1 shows the performance–compute trade-off across all solvers (inference time only; training time for neural networks is excluded). Solver inference times span three orders of magnitude. The fastest solvers (< 2 s) include most subspace scanners (SSM, FLEX-SSM, AP, RAP-MUSIC), matching pursuit methods (OMP, CoSaMP), and simple beamformers (SSP-ESMV, LCMV-MVPURE). Moderate-cost solvers (2–10 s) include most Champagne variants, the ESMV family, and basic minimum norm methods. The most expensive solvers (> 30 s) include iterative Bayesian methods (VB-SBL: 91 s, BCS: 87 s), L_1-based methods (MCE: 85 s), and hybrid methods (Hydra: 136 s, FLEX-GreedyML: 134 s). Total Variation (392 s) and SSLOFO (991 s) are the slowest solvers. Deep learning solvers have modest inference times (FC: $\sim$5 s, CovCNN-KL: $\sim$4 s) but require substantial upfront training ($\sim$250–1,900 s per scenario). The Pareto front (Figure 1) is dominated by subspace methods at the fast end and FLEX-GreedyML at the accurate end, with Subspace-SBL variants offering an attractive middle ground.

8.2 Related Software

Several established toolboxes provide M/EEG source imaging functionality. `MNE-Python` [14] includes MNE, dSPM, sLORETA, eLORETA, LCMV, DICS, MUSIC, and RAP-MUSIC. Brainstorm [15] (MATLAB) provides minimum norm variants, beamformers, and dipole fitting. FieldTrip [16] (MATLAB) focuses on beamforming methods. SPM/DAiSS [17] (MATLAB) offers Bayesian approaches including MSP and variational methods. ESINet [12] introduced deep learning for EEG source imaging but was limited to fully connected architectures.

`invertmeeg` differs from these tools in scope and methodology. With 112 solvers (vs. typically 5–15), it enables the kind of large-scale systematic comparison presented here, which would be infeasible when methods are scattered across different toolboxes with incompatible interfaces and data formats. The bundling of simulation, solving, and evaluation into one package ensures that all methods are compared under identical conditions—same forward model, same noise realizations, same metrics—eliminating confounds that arise from cross-toolbox comparisons.

8.3 Design Decisions and Limitations

All solvers in `invertmeeg` are re-implementations based on published algorithms, not wrappers around original authors' code. While we have validated the implementations against published results where possible, subtle differences may exist. Users who require exact reproduction of a specific paper's results should verify against the original implementation.

The benchmark uses a 32-channel EEG montage with an `ico3` source space (1,284 dipoles). While this configuration is representative of many clinical and research setups, results may differ for high-density arrays (128–256 channels) or finer source spaces, which increase the conditioning of the forward problem.

The package currently supports only fixed-orientation source models. Free-orientation forward models are automatically converted to fixed orientation, which may not be appropriate for all applications.

Computational cost varies substantially across solver categories. Linear solvers (MNE, eLORETA) compute an explicit inverse operator and are fast to apply; iterative solvers (Champagne, BCS) may require many iterations to converge; and deep learning solvers require training, which can

take minutes to hours. `invertmeeg` does not currently provide GPU acceleration for non-neural-network solvers.

The package is licensed under the GPL-3.0-only license.

8.4 Future Work

Planned extensions include GPU-accelerated linear algebra for large-scale source spaces, time-frequency source imaging methods, connectivity-informed inverse solutions, and integration with parcellation-based analyses. We plan to extend the benchmark to MEG sensor configurations and higher-density EEG montages, and to include real-data validation on established phantom and auditory/visual paradigm datasets.

Perhaps the most exciting direction is the systematic use of AI coding agents for inverse solver development. The over 30 novel solvers in `invertmeeg` were developed through an iterative loop in which an AI agent proposes algorithmic combinations, the benchmark provides immediate quantitative feedback, and the agent refines its designs accordingly. This "agentic" workflow collapses the traditional cycle of literature review, implementation, and evaluation from weeks to hours, enabling exploration of a vast combinatorial space of algorithmic building blocks that would be impractical to search manually. Early results are striking: several AI-developed solvers (Subspace-SBL, FLEX-GreedyML, Hydra) rank among the top performers in our benchmark, outperforming methods refined by human experts over decades. A companion paper dedicated to this agentic development methodology—its workflow, failure modes, and implications for scientific software engineering—is forthcoming.

9 Conclusion

We have presented the most comprehensive benchmark of M/EEG inverse solvers to date, evaluating 106 methods—including three deep learning solvers—across four scenarios that span the range of source configurations encountered in practice. Our results demonstrate that solver selection has a profound impact on reconstruction quality, with performance differences of an order of magnitude between the best and worst methods, and that the optimal choice depends strongly on the expected source configuration.

Key findings include: (1) flexible subspace scanners (FLEX-GreedyML, FLEX-SSM) and hybrid methods (Hydra) provide the best overall performance across diverse scenarios; (2) Bayesian methods with structured priors (Subspace-SBL) excel under noisy conditions; (3) popular minimum norm methods (MNE, eLORETA) consistently rank in the bottom quarter, suggesting that their widespread use as defaults may not be justified when better alternatives are available; and (4) deep learning solvers, while promising, currently rank in the lower half of the leaderboard at orders-of-magnitude higher computational cost than leading classical methods.

All 112 solvers are implemented in `invertmeeg`, an open-source Python package that provides a unified interface within the `MNE-Python` ecosystem, together with the simulation framework and evaluation tools needed to reproduce and extend these results. The source code is available at `https://github.com/LukeTheHecker/invertmeeg` under the GPL-3.0-only license and can be installed via `pip install invertmeeg`. Documentation and tutorials are available at `https://lukethehecker.github.io/invertmeeg/`.

References

[1] M. S. Hämäläinen, R. Hari, R. J. Ilmoniemi, J. Knuutila, and O. V. Lounasmaa. Magnetoencephalography—theory, instrumentation, and applications to noninvasive studies of the working human brain. *Reviews of Modern Physics*, 65(2):413, 1993.

[2] S. Baillet, J. C. Mosher, and R. M. Leahy. Electromagnetic brain mapping. *IEEE Signal Processing Magazine*, 18(6):14–30, 2001.

[3] R. Grech, T. Cassar, J. Muscat, K. P. Camilleri, S. G. Fabri, M. Zervakis, P. Xanthopoulos, V. Sakkalis, and B. Vanrumste. Review on solving the inverse problem in EEG source analysis. *Journal of NeuroEngineering and Rehabilitation*, 5(1):25, 2008.

[4] M. S. Hämäläinen and R. J. Ilmoniemi. Interpreting magnetic fields of the brain: minimum norm estimates. *Medical & Biological Engineering & Computing*, 32(1):35–42, 1994.

[5] R. D. Pascual-Marqui, C. M. Michel, and D. Lehmann. Low resolution electromagnetic tomography: a new method for localizing electrical activity in the brain. *International Journal of Psychophysiology*, 18(1):49–65, 1994.

[6] R. D. Pascual-Marqui. Standardized low-resolution brain electromagnetic tomography (sLORETA): technical details. *Methods and Findings in Experimental and Clinical Pharmacology*, 24(Suppl D):5–12, 2002.

[7] B. D. Van Veen, W. Van Drongelen, M. Yuchtman, and A. Suzuki. Localization of brain electrical activity via linearly constrained minimum variance spatial filtering. *IEEE Transactions on Biomedical Engineering*, 44(9):867–880, 1997.

[8] D. P. Wipf, J. P. Owen, H. T. Attias, K. Sekihara, and S. S. Nagarajan. Robust Bayesian estimation of the location, orientation, and time course of multiple correlated neural sources using MEG. *NeuroImage*, 49(1):641–655, 2010.

[9] K. Friston, L. Harrison, J. Daunizeau, S. Kiebel, C. Phillips, N. Trujillo-Barreto, T. Henson, G. Flandin, and J. Mattout. Multiple sparse priors for the M/EEG inverse problem. *NeuroImage*, 39(3):1104–1120, 2008.

[10] I. F. Gorodnitsky, J. S. George, and B. D. Rao. Neuromagnetic source imaging with FOCUSS: a recursive weighted minimum norm algorithm. *Electroencephalography and Clinical Neurophysiology*, 95(4):231–251, 1997.

[11] J. C. Mosher, P. S. Lewis, and R. M. Leahy. Multiple dipole modeling and localization from spatio-temporal MEG data. *IEEE Transactions on Biomedical Engineering*, 39(6):541–557, 1992.

[12] L. Hecker, R. Rupprecht, L. H. Thi, and S. Braun. ConvDip: A convolutional neural network for better EEG source imaging. *Frontiers in Neuroscience*, 15:569918, 2021.

[13] L. Hecker, R. Rupprecht, and S. Braun. Long short-term memory and convolutional neural networks for EEG source imaging with deep learning. 2022.

[14] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7:267, 2013.

[15] F. Tadel, S. Baillet, J. C. Mosher, D. Pantazis, and R. M. Leahy. Brainstorm: a user-friendly application for MEG/EEG analysis. *Computational Intelligence and Neuroscience*, 2011:879716, 2011.

[16] R. Oostenveld, P. Fries, E. Maris, and J.-M. Schoffelen. FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data. *Computational Intelligence and Neuroscience*, 2011:156869, 2011.

[17] V. Litvak, J. Mattout, S. Kiebel, C. Phillips, R. Henson, J. Kilner, G. Barnes, R. Oostenveld, J. Daunizeau, G. Flandin, W. Penny, and K. Friston. EEG and MEG data analysis in SPM8. *Computational Intelligence and Neuroscience*, 2011:852961, 2011.

[18] A. M. Dale, A. K. Liu, B. R. Fischl, R. L. Buckner, J. W. Belliveau, J. D. Lewine, and E. Halgren. Dynamic statistical parametric mapping: combining fMRI and MEG for high-resolution imaging of cortical activity. *Neuron*, 26(1):55–67, 2000.

[19] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[20] R. D. Pascual-Marqui. Discrete, 3D distributed, linear imaging methods of electric neuronal activity. Part 1: exact, zero error localization. *arXiv preprint arXiv:0710.3341*, 2007.

[21] D. P. Wipf and S. S. Nagarajan. A unified Bayesian framework for MEG/EEG source imaging. *NeuroImage*, 44(3):947–966, 2009.

[22] C. Amblard, E. Lapalme, and J.-M. Lina. Biomagnetic source detection by maximum entropy and graphical models. *IEEE Transactions on Biomedical Engineering*, 51(3):427–442, 2004.

[23] J. Gross, J. Kujala, M. Hämäläinen, L. Timmermann, A. Schnitzler, and R. Salmelin. Dynamic imaging of coherent sources: studying neural interactions in the human brain. *Proceedings of the National Academy of Sciences*, 98(2):694–699, 2001.

[24] J. Vrba and S. E. Robinson. Signal processing in magnetoencephalography. *Methods*, 25(2):249–271, 2001.

[25] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

[26] D. Needell and J. A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[27] W. Dai and O. Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, 2009.

[28] J. C. Mosher and R. M. Leahy. Source localization using recursively applied and projected (RAP) MUSIC. *IEEE Transactions on Signal Processing*, 47(2):332–340, 1999.

[29] S. Sommariva and A. Sorrentino. Sequential Monte Carlo samplers for semi-linear inverse problems and application to magnetoencephalography. *Inverse Problems*, 30(11):114020, 2014.

[30] L. Hecker, L. Tebartz van Elst, and J. Kornmeier. Source localization using recursively applied and projected MUSIC with flexible extent estimation. *Frontiers in Neuroscience*, 17:1170862, 2023.

[31] M. Wax and T. Kailath. Detection of signals by information theoretic criteria. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(2):387–392, 1985.

[32] G. H. Golub, M. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

[33] M. A. Lukas. Strong robust generalized cross-validation for choosing the regularization parameter. *Inverse Problems*, 26(4):045002, 2010.

[34] P. C. Hansen. Analysis of discrete ill-posed problems by means of the L-curve. *SIAM Review*, 34(4):561–580, 1992.

[35] C. Villani. *Optimal Transport: Old and New*. Springer, 2009.

[36] T. Saito and M. Rehmsmeier. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*, 10(3):e0118432, 2015.

[37] S. Yang, M. Jiao, J. Xiang, N. Fotedar, H. Sun, and F. Liu. Rejuvenating classical brain electrophysiology source localization methods with spatial graph Fourier filters for source extents estimation. *Brain Informatics*, 11(1):8, 2024.

[38] I. F. Gorodnitsky and B. D. Rao. Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm. *IEEE Transactions on Signal Processing*, 45(3):600–616, 1997.

[39] A. Gramfort, M. Kowalski, and M. Hämäläinen. Mixed-norm estimates for the M/EEG inverse problem using accelerated gradient methods. *Physics in Medicine and Biology*, 57(7):1937–1961, 2012.

[40] H. Liu, P. H. Schimpf, G. Dong, X. Gao, F. Yang, and S. Gao. Standardized shrinking LORETA-FOCUSS (SSLOFO): A new algorithm for spatio-temporal EEG source reconstruction. *IEEE Transactions on Biomedical Engineering*, 52(10):1681–1691, 2005.

[41] R. Grave de Peralta Menendez, S. Gonzalez Andino, G. Lantz, C. M. Michel, and T. Landis. Noninvasive localization of electromagnetic epileptic activity. I. Method descriptions and simulations. *Brain Topography*, 14(2):131–137, 2001.

[42] G. Backus and F. Gilbert. The resolving power of gross earth data. *Geophysical Journal of the Royal Astronomical Society*, 16(2):169–205, 1968.

[43] S. Baillet and L. Garnero. A Bayesian approach to introducing anatomo-functional priors in the EEG/MEG inverse problem. *IEEE Transactions on Biomedical Engineering*, 44(5):374–385, 1997.

[44] S. Wang, C. Wei, K. Lou, D. Gu, and Q. Liu. Advancing EEG/MEG source imaging with geometric-informed basis functions. *arXiv preprint arXiv:2401.17939*, 2024.

[45] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1–4):259–268, 1992.

[46] M. Aud'hui, A. Kachenoura, L. Albera, I. Merlet, M. Yochum, P. Van Bogaert, and P. Benquet. SISSY: Structured iterative support selection with sparsity for EEG source imaging. *NeuroImage*, 2025.

[47] A. Hashemi and S. Haufe. Improving EEG source localization through spatio-temporal sparse Bayesian learning. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1935–1939. IEEE, 2018.

[48] C. Cai, A. Hashemi, M. Diwakar, S. Haufe, K. Sekihara, and S. S. Nagarajan. Robust estimation of noise for electromagnetic brain imaging with the Champagne algorithm. *NeuroImage*, 225:117411, 2021.

[49] Z. Feng, C. Guan, and Y. Sun. Block-Champagne: a novel Bayesian framework for imaging extended E/MEG source. *IEEE Transactions on Medical Imaging*, 2025.

[50] M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[51] S. Ji, Y. Xue, and L. Carin. Bayesian compressive sensing. *IEEE Transactions on Signal Processing*, 56(6):2346–2356, 2008.

[52] B. D. Van Veen and K. M. Buckley. Beamforming: a versatile approach to spatial filtering. *IEEE ASSP Magazine*, 5(2):4–24, 1988.

[53] J. Jurkowska, J. Dreszer, M. Lewandowska, and T. Piotrowski. Multi-source neural activity indices and spatial filters for EEG/MEG inverse problem: An extension to MNE-Python. *arXiv preprint arXiv:2509.14118*, 2025.

[54] Y. Jonmohamadi, G. Poudel, C. Innes, D. Weiss, R. Krueger, and R. Jones. Comparison of beamformers for EEG source signal reconstruction. *Biomedical Signal Processing and Control*, 14:175–188, 2014.

[55] S. E. Robinson and J. Vrba. Functional neuroimaging by synthetic aperture magnetometry (SAM). In *Recent Advances in Biomagnetism*, pages 302–305, 1999.

[56] K. Sekihara, S. S. Nagarajan, D. Poeppel, and A. Marantz. Asymptotic SNR of scalar and vector minimum-variance beamformers for neuromagnetic source reconstruction. *IEEE Transactions on Biomedical Engineering*, 51(10):1726–1734, 2004.

[57] J. L. Yu and C. C. Yeh. Generalized eigenspace-based beamformers. *IEEE Transactions on Signal Processing*, 43(11):2453–2461, 1995.

[58] A. S. Nunes, A. Moiseev, N. Kozhemiako, T. Cheung, U. Ribary, and S. M. Doesburg. Multiple constrained minimum variance beamformer (MCMV) performance in connectivity analyses. *NeuroImage*, 208:116386, 2020.

[59] A. Kuznetsova, Y. Nurislamova, and A. Ossadtchi. Modified covariance beamformer for solving MEG inverse problem in the environment with correlated sources. *NeuroImage*, 228:117677, 2021.

[60] R. O. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34(3):276–280, 1986.

[61] M. Wax and A. Adler. Direction of arrival estimation in the presence of model errors by signal subspace matching. *Signal Processing*, 181:107900, 2021.

[62] A. Adler, M. Wax, and D. Pantazis. Brain source localization by alternating projection. In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pages 1–5, 2022.

[63] M. Mishali and Y. C. Eldar. Reduce and boost: recovering arbitrary sets of jointly sparse vectors. *IEEE Transactions on Signal Processing*, 56(10):4692–4702, 2008.

[64] M. F. Duarte and Y. C. Eldar. Structured compressed sensing: from theory to applications. *IEEE Transactions on Signal Processing*, 59(9):4053–4085, 2011.

[65] C. Grova, J. Daunizeau, E. Kobayashi, A. P. Bagshaw, J. M. Lina, F. Dubeau, and J. Gotman. Concordance between distributed EEG source localization and simultaneous EEG-fMRI studies of epileptic spikes. *NeuroImage*, 39(2):755–774, 2008.

[66] R. A. Chowdhury, J. M. Lina, E. Kobayashi, and C. Grova. MEG source localization of spatially extended generators of epileptic activity: comparing entropic and hierarchical Bayesian approaches. *PloS One*, 8(2):e55969, 2013.

[67] L. Albera, A. Ferréol, D. Cosandier-Rimélé, I. Merlet, and F. Wendling. Brain source localization using a fourth-order deflation scheme. *IEEE Transactions on Biomedical Engineering*, 55(2):490–501, 2008.

[68] C. Cai, H. Kang, A. Hashemi, D. Chen, M. Diwakar, S. Haufe, K. Sekihara, W. Wu, and S. S. Nagarajan. Bayesian algorithms for joint estimation of brain activity and noise in electromagnetic imaging. *IEEE Transactions on Medical Imaging*, 42(3):762–773, 2023.

[69] C. Cai, A. Hashemi, M. Diwakar, D. Chen, S. Haufe, K. Sekihara, and S. S. Nagarajan. Enhancing M/EEG source imaging with Champagne: model validation and novel contributions. *NeuroImage*, 2024.

[70] R. Grave de Peralta Menendez, M. M. Murray, C. M. Michel, R. Martuzzi, and S. L. Gonzalez Andino. Electrical neuroimaging based on biophysical constraints. *NeuroImage*, 21(2):527–539, 2004.

[71] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[72] F.-H. Lin, T. Witzel, S. P. Ahlfors, S. M. Stufflebeam, J. W. Belliveau, and M. S. Hämäläinen. Assessing and improving the spatial accuracy in MEG source localization by depth-weighted minimum-norm estimates. *NeuroImage*, 31(1):160–171, 2006.

[73] J. P. Owen, D. P. Wipf, H. T. Attias, K. Sekihara, and S. S. Nagarajan. Performance evaluation of the Champagne source reconstruction algorithm on simulated and real M/EEG data. *NeuroImage*, 60(1):305–323, 2012.

[74] K. Uutela, M. Hämäläinen, and E. Somersalo. Visualization of magnetoencephalographic data using minimum current estimates. *NeuroImage*, 10(2):173–180, 1999.

[75] H. Wu, S. Gao, J. Li, and X. Li. A new method for spatio-temporal EEG source reconstruction: standardized shrinking LORETA-FOCUSS (SSLOFO). *IEEE Transactions on Biomedical Engineering*, 52(11):1781–1792, 2005.

[76] Ü. Aydin, S. Rampp, A. Wollbrink, H. Kugel, J.-H. Cho, T. R. Knösche, C. Grova, and C. H. Wolters. Zoomed MRI-guided TRAP-MUSIC for MEG/EEG source analysis in presurgical evaluation of epilepsy patients. *NeuroImage: Clinical*, 17:566–575, 2018.

A Detailed Benchmark Results

The following tables provide per-metric results for each of the four evaluation scenarios. Values are reported as means across 50 independent simulations, sorted by EMD. Metrics: EMD (earth mover's distance in mm, primary), AP (average precision, secondary), MLE (mean localization error in mm), Corr (correlation).

Table 5: Results for the **Focal** scenario (mean across 50 simulations, sorted by EMD). Best value per metric in **bold**.

Solver	EMD (mm)↓	AP↑	MLE (mm)↓	Corr↑	Solver	EMD (mm)↓	AP↑	MLE (mm)↓	Corr↑
FLEX-GreedyML	**0.0**	**1.00**	**0.0**	**1.00**	CoSaMP	8.4	0.88	6.4	0.73
Hydra	**0.0**	**1.00**	**0.0**	**1.00**	FLEX-SP	8.7	0.95	8.3	0.80
Subspace-SBL+	**0.0**	**1.00**	**0.0**	**1.00**	EXSO-MUSIC	11.4	0.81	5.2	0.60
Subspace-SBL	**0.0**	**1.00**	**0.0**	**1.00**	LCMV-MVPURE	11.5	0.65	11.4	0.58
Chimera	**0.0**	**1.00**	**0.0**	0.98	APSE	13.2	**1.00**	6.1	0.53
FLEX-SSM	**0.0**	**1.00**	**0.0**	**1.00**	SP	13.3	0.94	11.1	0.77
GreedyML	**0.0**	**1.00**	**0.0**	**1.00**	Gamma-MAP-MSP	18.5	0.29	15.6	0.19
FLEX-AP	**0.0**	**1.00**	**0.0**	**1.00**	Champ.-MacKay-NL-F	18.9	0.80	19.0	0.72
ExhaustiveML	**0.0**	**1.00**	**0.0**	**1.00**	MSP	21.4	0.16	20.2	0.12
FLEX-MUSIC	**0.0**	**1.00**	**0.0**	**1.00**	DICS	21.9	0.80	16.6	-0.01
FLEX-TRAP-MUSIC	**0.0**	**1.00**	**0.0**	**1.00**	ESMV-MVPURE	22.6	0.29	14.8	-0.03
Flex-NL-Champagne	**0.0**	**1.00**	**0.0**	1.00	L1/L2	25.3	0.51	25.3	0.45
Flex-NL-Champ.-Omix	**0.0**	**1.00**	**0.0**	1.00	MUSIC	31.9	0.18	22.0	0.21
AP	**0.0**	**1.00**	**0.0**	**1.00**	Source-MAP	34.2	0.51	15.4	0.24
SSM	**0.0**	**1.00**	**0.0**	**1.00**	Champ.-Conv.-NL-F	34.7	0.76	31.5	0.61
RAP-MUSIC	**0.0**	**1.00**	**0.0**	**1.00**	Block-Champagne	36.5	0.13	30.1	0.12
Flex-Champagne	**0.0**	**1.00**	**0.0**	**1.00**	MCE	38.5	0.29	34.7	0.25
Omni-Champagne	**0.0**	**1.00**	**0.0**	**1.00**	SSLOFO	45.2	0.41	18.5	0.14
TRAP-MUSIC	**0.0**	**1.00**	**0.0**	**1.00**	CovCNN-KL	45.6	0.41	27.7	0.28
Flex-Champ.-Omix	**0.0**	**1.00**	**0.0**	**1.00**	SR-eLORETA	46.2	0.64	16.2	0.14
IR-ESMV	**0.0**	**1.00**	**0.0**	1.00	ReciPSIICOS	48.4	0.72	13.3	0.18
SSP-IR-ESMV	**0.0**	**1.00**	0.0	0.99	SubSMP	48.9	0.51	24.9	0.49
Safe-Flex-ESMV	**0.0**	**1.00**	0.4	0.94	HOCMCMV	48.9	0.23	22.7	0.09
ESMV	**0.0**	**1.00**	**0.0**	1.00	HOCMV	48.9	0.23	22.7	0.09
Deblur-Flex-ESMV	**0.0**	**1.00**	**0.0**	0.98	Unit-Noise-Gain	51.2	0.38	15.8	0.09
Flex-ESMV	**0.0**	**1.00**	**0.0**	0.98	WNMV	51.2	0.38	15.8	0.09
Adapt-Flex-ESMV	**0.0**	**1.00**	0.5	0.94	MVAB	51.9	0.45	15.0	0.09
OMP	**0.0**	**1.00**	**0.0**	**1.00**	SAM	52.5	0.25	22.9	0.09
Sharp-Flex-ESMV	**0.0**	**1.00**	0.4	0.94	SMV	53.4	0.85	10.9	0.09
EBB	**0.0**	**1.00**	**0.0**	**1.00**	MCMV	56.2	0.06	25.6	0.05
Flex-ESMV-MVPURE	**0.0**	**1.00**	**0.0**	**1.00**	LCMV	57.2	0.18	22.8	0.08
FLEX-OMP	**0.0**	**1.00**	**0.0**	**1.00**	MEM	58.6	0.12	48.2	0.04
ISubSMP	**0.0**	**1.00**	**0.0**	**1.00**	CovCNN	59.9	0.21	39.1	0.10
ECD	**0.0**	**1.00**	**0.0**	**1.00**	EPIFOCUS	60.2	0.11	38.9	0.08
Gamma-MAP	0.1	**1.00**	0.6	0.99	GFT-MNE	60.2	0.05	35.0	0.06
Champ.-Conv.	0.3	**1.00**	0.7	1.00	sLORETA	60.4	0.43	21.5	0.08
Champ.-LowSNR	0.4	**1.00**	1.6	0.99	LAURA	60.6	0.09	41.9	0.07
Champ.-MacKay	0.4	**1.00**	0.9	1.00	wMNE	60.9	0.15	34.1	0.08
SESAME	0.4	0.97	0.4	0.97	FC (ESINet)	61.7	0.16	32.3	0.07
SSP-ESMV	0.6	0.97	0.5	0.97	cMEM	61.8	0.05	42.4	0.04
NL-Champagne	1.4	**1.00**	4.4	0.99	GFT-L1	63.1	0.02	44.1	0.03
Champ.-MacKay-NL-H	1.8	0.95	4.1	0.92	Source-MAP-MSP	68.7	0.07	50.3	0.06
Champ.-MacKay-NL-P	1.9	0.92	5.4	0.91	TV	69.0	0.04	47.8	0.03
Champ.-MacKay-NL-D	1.9	0.92	5.4	0.91	dSPM	69.5	0.03	42.6	0.04
Champ.-Conv.-NL-H	2.0	0.93	4.9	0.92	LORETA	71.8	0.03	50.9	0.02
Champ.-Conv.-NL-P	2.3	0.92	5.6	0.90	S-MAP	71.8	0.03	50.9	0.02
Champ.-Conv.-NL-D	2.3	0.92	5.6	0.90	APE-SISSY	72.8	0.03	57.6	0.03
MxNE-BCD	2.7	0.94	4.4	0.92	GBF	73.3	0.03	56.9	0.04
BCS	3.7	0.93	7.6	0.91	Backus-Gilbert	74.7	0.01	64.9	-0.00
ReMBo	4.4	0.99	12.5	0.97	eLORETA	74.9	0.03	64.0	0.04

Table 6: Results for the **Multi** scenario (mean across 50 simulations, sorted by EMD). Best value per metric in **bold**.

Solver	EMD (mm)↓	AP↑	MLE (mm)↓	Corr↑	Solver	EMD (mm)↓	AP↑	MLE (mm)↓	Corr↑
GreedyML	**8.7**	**0.97**	**1.1**	**0.95**	ReMBo	39.8	0.19	32.6	0.25
FLEX-GreedyML	9.1	0.96	1.4	0.94	FLEX-ReMBo	40.1	0.18	33.5	0.25
Hydra	9.1	0.96	1.4	0.94	MUSIC	40.3	0.08	37.2	0.13
ExhaustiveML	11.0	0.87	2.3	0.87	CoSaMP	40.4	0.16	28.4	0.17
Subspace-SBL+	12.0	0.85	4.5	0.85	ESMV-MVPURE	41.1	0.07	30.0	0.02
RAP-MUSIC	12.9	0.85	4.6	0.85	Source-MAP	41.2	0.18	31.8	0.17
SSM	13.9	0.82	5.1	0.86	SR-eLORETA	42.1	0.14	32.0	0.13
FLEX-MUSIC	14.4	0.81	5.7	0.81	FLEX-CoSaMP	42.4	0.12	30.3	0.13
Chimera	14.4	0.80	5.6	0.83	FLEX-OMP	42.7	0.15	27.7	0.16
AP	14.4	0.75	6.4	0.77	Block-Champagne	42.7	0.05	33.8	0.07
FLEX-SSM	14.6	0.81	5.4	0.84	Gamma-MAP-MSP	42.9	0.09	31.5	0.08
Subspace-SBL	14.8	0.79	5.8	0.81	ISubSMP	43.0	0.12	27.6	0.13
Champ.-MacKay-NL-H	14.8	0.80	8.2	0.81	Champ.-Conv.-NL-F	44.2	0.40	30.1	0.38
TRAP-MUSIC	15.0	0.79	6.3	0.82	L1/L2	44.3	0.15	34.9	0.16
FLEX-AP	15.2	0.73	6.6	0.75	ECD	45.1	0.16	32.3	0.25
Champ.-Conv.-NL-H	15.2	0.80	8.2	0.81	Champ.-MacKay-NL-F	45.2	0.31	30.3	0.32
FLEX-TRAP-MUSIC	15.6	0.77	6.6	0.80	HOCMCMV	45.5	0.05	33.9	0.08
Champ.-LowSNR	15.8	0.79	9.9	0.80	HOCMV	45.5	0.05	33.9	0.08
NL-Champagne	15.8	0.74	10.5	0.78	MCE	46.0	0.08	38.4	0.09
BCS	16.0	0.78	8.8	0.79	SSLOFO	46.6	0.14	33.5	0.11
Champ.-MacKay-NL-P	16.2	0.77	9.7	0.78	Unit-Noise-Gain	47.2	0.09	30.9	0.08
Champ.-MacKay-NL-D	16.2	0.77	9.7	0.78	WNMV	47.2	0.09	30.9	0.08
Champ.-Conv.-NL-P	16.5	0.77	9.5	0.78	SAM	47.4	0.05	36.2	0.08
Champ.-Conv.-NL-D	16.5	0.77	9.5	0.78	ReciPSIICOS	47.5	0.15	27.5	0.09
Champ.-MacKay	16.6	0.76	8.8	0.76	MVAB	47.6	0.10	31.0	0.08
Champ.-Conv.	16.6	0.76	8.8	0.76	SMV	48.7	0.20	29.9	0.08
Flex-NL-Champagne	19.2	0.61	9.6	0.64	LCMV	49.5	0.05	34.6	0.07
Flex-NL-Champ.-Omix	20.2	0.60	9.9	0.63	MCMV	50.1	0.01	34.8	0.03
Omni-Champagne	22.1	0.59	11.4	0.60	CovCNN-KL	50.1	0.06	41.3	0.09
EXSO-MUSIC	22.2	0.60	8.9	0.61	sLORETA	50.9	0.12	35.2	0.08
Flex-Champagne	23.6	0.55	11.9	0.57	DICS	51.0	0.07	39.7	0.01
MxNE-BCD	23.9	0.53	16.3	0.58	EPIFOCUS	51.2	0.06	45.2	0.08
Flex-Champ.-Omix	24.0	0.53	13.0	0.55	GFT-MNE	51.3	0.03	39.0	0.06
IR-ESMV	27.1	0.73	11.4	0.60	wMNE	51.8	0.07	42.2	0.08
Safe-Flex-ESMV	27.2	0.73	12.2	0.54	FC (ESINet)	52.2	0.05	42.9	0.07
SSP-IR-ESMV	28.4	0.69	12.7	0.58	LAURA	52.5	0.04	44.4	0.07
MSP	28.5	0.15	20.6	0.17	CovCNN	52.5	0.05	43.8	0.08
Adapt-Flex-ESMV	28.5	0.68	13.7	0.47	SubSMP	52.6	0.09	35.2	0.07
Sharp-Flex-ESMV	28.8	0.69	13.6	0.47	MNE	53.4	0.04	46.1	0.07
ESMV	29.4	0.66	13.2	0.51	GBF	53.5	0.02	43.7	0.05
Flex-ESMV-MVPURE	29.8	0.85	9.1	0.73	GFT-L1	53.7	0.01	42.2	0.03
Flex-ESMV	30.1	0.77	11.5	0.64	eLORETA	53.8	0.05	44.0	0.07
Deblur-Flex-ESMV	30.1	0.77	11.5	0.64	dSPM	56.3	0.01	42.0	0.04
EBB	30.1	0.54	15.4	0.66	TV	56.9	0.02	45.8	0.03
APSE	30.5	0.52	16.4	0.35	LORETA	57.8	0.01	45.1	0.02
OMP	31.2	0.24	22.2	0.29	S-MAP	57.8	0.01	45.1	0.02
VB-SBL	34.3	0.48	21.0	0.47	Random Noise	58.8	0.00	48.3	0.00
SESAME	34.5	0.19	26.6	0.22	cMEM	59.1	0.02	44.4	0.03
Gamma-MAP	35.6	0.32	21.6	0.28	Source-MAP-MSP	59.3	0.03	47.0	0.04
SP	36.4	0.23	25.3	0.22	APE-SISSY	60.9	0.01	52.3	0.03

Table 7: Results for the **Extended** scenario (mean across 50 simulations, sorted by EMD). Best value per metric in **bold**.

Solver	EMD (mm)↓	AP↑	MLE (mm)↓	Corr↑	Solver	EMD (mm)↓	AP↑	MLE (mm)↓	Corr↑
Subspace-SBL+	**22.6**	0.43	16.9	0.46	FLEX-SP	38.1	0.12	32.0	0.02
FLEX-GreedyML	22.9	**0.45**	**13.8**	**0.53**	SP	38.2	0.05	29.7	0.05
Hydra	22.9	**0.45**	**13.8**	**0.53**	LCMV-MVPURE	38.3	0.05	32.3	0.05
FLEX-SSM	26.2	0.40	17.8	0.47	EBB	38.5	0.08	27.6	0.16
Subspace-SBL	26.5	0.41	18.1	0.46	FLEX-CoSaMP	40.1	0.06	32.9	0.05
Chimera	26.6	0.34	19.4	0.41	SESAME	40.2	0.04	31.6	0.04
GreedyML	27.2	0.08	18.8	0.15	ESMV-MVPURE	40.4	0.04	32.8	0.01
Flex-NL-Champ.-Omix	27.2	0.38	20.9	0.41	CoSaMP	41.0	0.04	32.8	0.03
MSP	27.3	0.31	20.8	0.30	ISubSMP	41.1	0.10	31.0	0.04
FLEX-AP	27.6	0.36	19.6	0.42	CovCNN-KL	41.5	0.11	38.2	0.15
NL-Champagne	27.9	0.12	20.8	0.19	FLEX-OMP	41.6	0.11	30.5	0.04
Champ.-Conv.	27.9	0.10	20.8	0.17	Flex-ESMV-MVPURE	41.9	0.07	27.9	0.11
Flex-NL-Champagne	27.9	0.38	20.8	0.41	HOCMV	42.0	0.08	34.3	0.13
Champ.-MacKay	27.9	0.09	21.0	0.17	HOCMCMV	42.0	0.08	34.3	0.13
Gamma-MAP-MSP	27.9	0.40	18.9	0.32	ReMBo	42.2	0.04	36.1	0.06
Champ.-LowSNR	28.1	0.10	21.0	0.17	MCE	42.5	0.05	38.2	0.08
Champ.-MacKay-NL-H	28.2	0.10	21.5	0.17	APE-SISSY	42.6	0.11	39.5	0.18
Champ.-Conv.-NL-H	28.3	0.10	21.4	0.17	L1/L2	42.7	0.06	35.7	0.09
Champ.-MacKay-NL-P	28.4	0.10	22.1	0.17	CovCNN	43.6	0.09	40.8	0.13
Champ.-MacKay-NL-D	28.4	0.10	22.1	0.17	ECD	43.8	0.04	35.9	0.06
BCS	28.5	0.11	21.9	0.17	Unit-Noise-Gain	43.8	0.09	33.2	0.14
ExhaustiveML	28.5	0.07	19.4	0.13	WNMV	43.8	0.09	33.2	0.14
Champ.-Conv.-NL-P	28.7	0.10	22.4	0.17	SAM	43.8	0.07	36.7	0.13
Champ.-Conv.-NL-D	28.7	0.10	22.4	0.17	ReciPSIICOS	43.9	0.09	30.9	0.10
Gamma-MAP	28.8	0.14	21.4	0.16	MVAB	44.0	0.08	33.1	0.13
IR-ESMV	29.0	0.15	25.6	0.21	SMV	45.3	0.11	32.5	0.14
EXSO-MUSIC	29.3	0.26	23.4	0.33	FLEX-ReMBo	45.5	0.04	38.7	0.04
SSP-IR-ESMV	29.3	0.14	26.7	0.20	SISSY	46.0	0.09	41.2	0.15
Flex-Champagne	29.5	0.37	23.6	0.37	LCMV	46.3	0.07	35.0	0.12
MxNE-BCD	29.5	0.09	23.7	0.16	MCMV	46.3	0.05	34.4	0.08
Flex-Champ.-Omix	29.9	0.36	23.3	0.37	SubSMP	46.4	0.10	30.0	0.02
FLEX-MUSIC	30.2	0.35	20.5	0.41	GFT-MNE	46.5	0.11	38.5	0.17
ESMV	30.3	0.14	26.4	0.18	EPIFOCUS	47.1	0.08	45.5	0.14
FLEX-TRAP-MUSIC	30.6	0.35	21.2	0.40	LAURA	47.2	0.08	42.3	0.14
Omni-Champagne	30.7	0.24	23.5	0.29	wMNE	47.4	0.09	42.3	0.14
AP	31.3	0.06	23.6	0.10	sLORETA	47.6	0.10	36.7	0.14
SSM	32.4	0.06	23.5	0.12	Champ.-MacKay-NL-F	48.2	0.05	36.9	0.07
Adapt-Flex-ESMV	34.1	0.22	25.1	0.25	DICS	48.4	0.04	38.8	-0.00
Safe-Flex-ESMV	34.2	0.22	23.9	0.25	Champ.-Conv.-NL-F	48.6	0.06	37.7	0.08
Source-MAP	34.3	0.13	33.2	0.20	FC (ESINet)	49.1	0.11	40.3	0.18
SSP-ESMV	34.3	0.07	29.4	0.11	GFT-L1	50.5	0.09	40.8	0.14
Block-Champagne	35.0	0.25	29.5	0.34	SSLOFO	50.7	0.08	44.5	0.12
Deblur-Flex-ESMV	35.1	0.19	23.8	0.23	dSPM	53.5	0.04	41.5	0.07
Flex-ESMV	35.2	0.22	23.8	0.25	cMEM	54.5	0.06	43.1	0.08
APSE	35.3	0.15	26.5	0.11	LORETA	56.2	0.07	44.1	0.09
Sharp-Flex-ESMV	35.3	0.20	25.1	0.22	S-MAP	56.2	0.07	44.1	0.09
RAP-MUSIC	35.8	0.06	26.7	0.10	Random Noise	56.8	0.03	48.3	0.00
MUSIC	36.1	0.08	38.0	0.14	GBF	57.9	0.05	51.3	0.10
TRAP-MUSIC	36.2	0.06	27.2	0.10	eLORETA	58.5	0.04	56.6	0.06
OMP	36.2	0.04	28.1	0.07	Backus-Gilbert	59.9	0.04	54.8	0.01

Table 8: Results for the **Noisy** scenario (mean across 50 simulations, sorted by EMD). Best value per metric in **bold**.

Solver	EMD (mm)↓	AP↑	MLE (mm)↓	Corr↑	Solver	EMD (mm)↓	AP↑	MLE (mm)↓	Corr↑
Subspace-SBL	**9.3**	0.88	**3.9**	**0.90**	FLEX-CoSaMP	29.5	0.22	22.4	0.19
Chimera	9.7	0.88	4.2	0.90	MSP	30.1	0.27	24.7	0.23
FLEX-GreedyML	10.5	0.88	4.0	0.90	SP	30.3	0.21	19.9	0.24
Hydra	10.5	0.88	4.0	0.90	CoSaMP	31.0	0.19	22.9	0.19
FLEX-AP	10.9	0.86	4.3	0.87	ISubSMP	31.9	0.43	21.1	0.20
FLEX-SSM	10.9	**0.88**	4.0	0.90	ESMV-MVPURE	32.3	0.10	23.8	0.03
Subspace-SBL+	11.5	0.87	5.7	0.87	VB-SBL	36.4	0.25	22.4	0.28
FLEX-TRAP-MUSIC	11.7	0.86	5.1	0.87	ReMBo	36.8	0.17	33.9	0.24
FLEX-MUSIC	11.9	0.87	4.8	0.87	MUSIC	37.6	0.07	33.6	0.14
Flex-NL-Champagne	14.4	0.81	8.0	0.80	FLEX-ReMBo	38.1	0.16	37.0	0.22
Flex-NL-Champ.-Omix	14.5	0.81	8.4	0.81	Source-MAP	39.7	0.13	30.7	0.17
Flex-Champagne	16.2	0.80	8.7	0.78	CovCNN-KL	40.7	0.10	31.4	0.13
GreedyML	16.8	0.36	9.1	0.54	SR-eLORETA	43.9	0.10	32.8	0.14
Flex-Champ.-Omix	16.8	0.79	9.0	0.78	SubSMP	49.9	0.27	32.8	0.06
ExhaustiveML	17.0	0.36	9.2	0.54	Block-Champagne	51.2	0.17	38.4	0.17
Omni-Champagne	17.1	0.66	10.1	0.68	SSLOFO	52.0	0.12	29.7	0.11
SSM	17.7	0.34	9.3	0.52	CovCNN	52.1	0.05	40.8	0.09
AP	17.8	0.34	9.6	0.52	DICS	52.2	0.09	38.0	-0.01
TRAP-MUSIC	19.4	0.33	11.4	0.52	HOCMCMV	53.5	0.07	33.1	0.09
RAP-MUSIC	19.5	0.34	11.3	0.52	HOCMV	53.5	0.07	33.1	0.09
Champ.-MacKay	20.1	0.35	12.2	0.49	SAM	53.7	0.05	38.0	0.09
Champ.-Conv.	20.2	0.35	12.3	0.49	ReciPSIICOS	54.2	0.11	28.3	0.07
EXSO-MUSIC	20.2	0.41	14.3	0.41	Unit-Noise-Gain	55.0	0.10	28.4	0.09
Flex-ESMV	20.4	0.67	10.0	0.69	WNMV	55.0	0.10	28.4	0.09
Deblur-Flex-ESMV	20.4	0.51	9.9	0.68	MVAB	55.0	0.11	27.8	0.09
Champ.-MacKay-NL-H	20.8	0.34	16.8	0.47	SMV	55.8	0.16	27.2	0.09
Champ.-Conv.-NL-H	20.9	0.34	17.1	0.47	EPIFOCUS	55.9	0.05	46.7	0.09
Champ.-LowSNR	20.9	0.36	14.0	0.49	GFT-MNE	56.1	0.07	38.9	0.09
Safe-Flex-ESMV	21.1	0.62	15.6	0.56	MCE	56.3	0.02	45.9	0.04
IR-ESMV	21.5	0.40	14.3	0.46	LAURA	56.5	0.05	45.5	0.08
OMP	21.7	0.26	15.4	0.40	L1/L2	56.9	0.06	42.8	0.10
SSP-IR-ESMV	21.8	0.39	15.2	0.45	sLORETA	56.9	0.08	35.0	0.09
BCS	21.8	0.34	17.3	0.47	wMNE	57.1	0.07	39.7	0.09
Champ.-MacKay-NL-P	21.8	0.33	17.9	0.46	MCMV	57.6	0.02	33.3	0.04
Champ.-MacKay-NL-D	21.8	0.33	17.9	0.46	LCMV	57.9	0.04	37.0	0.08
Champ.-Conv.-NL-P	22.1	0.33	18.8	0.46	FC (ESINet)	59.5	0.06	45.7	0.08
Champ.-Conv.-NL-D	22.1	0.33	18.8	0.46	Source-MAP-MSP	59.9	0.03	46.6	0.05
NL-Champagne	22.5	0.28	20.8	0.42	GFT-L1	60.9	0.03	45.5	0.05
SESAME	23.5	0.24	17.0	0.36	Champ.-MacKay-NL-F	61.0	0.07	42.6	0.10
ESMV	24.2	0.37	16.4	0.37	MNE	61.1	0.04	48.4	0.07
SSP-ESMV	25.8	0.29	19.8	0.40	Champ.-Conv.-NL-F	61.7	0.10	42.7	0.13
EBB	25.9	0.28	16.6	0.42	eLORETA	62.0	0.03	49.6	0.06
Adapt-Flex-ESMV	26.2	0.58	17.6	0.44	cMEM	62.7	0.04	45.2	0.05
Flex-ESMV-MVPURE	26.2	0.34	14.7	0.45	GBF	63.4	0.04	50.4	0.07
Sharp-Flex-ESMV	26.3	0.58	17.7	0.44	TV	63.5	0.04	50.9	0.05
APSE	26.5	0.38	17.9	0.29	dSPM	63.7	0.01	48.0	0.04
Gamma-MAP	26.9	0.29	19.4	0.30	LORETA	64.0	0.03	48.0	0.03
Gamma-MAP-MSP	27.5	0.41	21.6	0.26	S-MAP	64.0	0.03	48.0	0.03
MxNE-BCD	28.7	0.29	23.0	0.40	Random Noise	65.3	0.01	52.5	0.00
FLEX-OMP	29.1	0.45	20.4	0.21	APE-SISSY	68.5	0.02	59.8	0.04

B Complete Solver Reference

Table 9: Complete list of all 112 solvers in `invertmeeg`, organized by category (Part 1: Minimum Norm, LORETA, Bayesian).

Category	Solver	Ref.	Primary Aliases
Minimum Norm (21)	MNE	[4]	`mne`
	GFT-MNE	[37]	`gft-mne`
	wMNE	[4]	`wmne`
	dSPM	[18]	`dspm`
	MCE (L1/FISTA)	[38]	`mce`, `l1`, `fista`
	L1/L2	[38]	`l1l2`
	MxNE-BCD	[39]	`mxne-bcd`, `mxne`
	GFT-L1	[37]	`gft-l1`
	Self-Reg. eLORETA	[20]	`sr-eloreta`
	LORETA	[5]	`loreta`
	sLORETA	[6]	`sloreta`
	eLORETA	[20]	`eloreta`
	SSLOFO	[40]	`sslofo`
	LAURA	[41]	`laura`
	Backus-Gilbert	[42]	`backus-gilbert`, `bg`
	S-MAP	[43]	`s-map`
	GBF	[44]	`gbf`
	EPIFOCUS	[41]	`epifocus`
	Total Variation	[45]	`tv`
	SISSY	[46]	`sissy`
	APE-SISSY	[46]	`apesissy`
Bayesian / SBL (31)	MacKay-Champagne	[8]	`champagne`, `mcc`
	Convexity-Champagne	[8]	`champagne-convexity`, `coc`
	TEM-Champagne	[8, 47]	`champagne-tem`
	AR-EM-Champagne	[8, 47]	`champagne-ar-em`
	LowSNR-Champagne	[8, 47]	`champagne-low-snr`
	MacKay-Champ. (homo. noise)	[8, 48]	`champagne-mackay-nl-homoscedastic`
	MacKay-Champ. (diag. noise)	[8, 48]	`champagne-mackay-nl-diag`
	MacKay-Champ. (prec. noise)	[8, 48]	`champagne-mackay-nl-prec`
	MacKay-Champ. (full noise)	[8, 48]	`champagne-mackay-nl-full`
	Conv.-Champ. (homo. noise)	[8, 48]	`champagne-convexity-nl-homoscedastic`
	Conv.-Champ. (diag. noise)	[8, 48]	`champagne-convexity-nl-diag`
	Conv.-Champ. (prec. noise)	[8, 48]	`champagne-convexity-nl-prec`
	Conv.-Champ. (full noise)	[8, 48]	`champagne-convexity-nl-full`
	NL-Champagne	[8, 48]	`nl-champagne`
	Omni-Champagne	[8]	`omni-champagne`
	Flex-Champagne	[8]	`flex-champagne`
	Flex-Champagne (omix)	[8]	`flex-champagne-omix`
	Flex-NL-Champagne	[8, 48]	`flex-nl-champagne`
	Flex-NL-Champagne (omix)	[8, 48]	`flex-nl-champagne-omix`
	Block-Champagne	[8, 49]	`block-champagne`
	Gamma-MAP	[21]	`gamma-map`
	Source-MAP	[21]	`source-map`
	Gamma-MAP-MSP	[21]	`gamma-map-msp`
	Source-MAP-MSP	[21]	`source-map-msp`
	MSP	[9]	`msp`
	MEM	[22, 65]	`mem`
	cMEM	[22, 65, 66]	`cmem`
	Subspace-SBL	[21]	`subspace-sbl`
	Subspace-SBL+	[21]	`subspace-sbl-plus`
	VB-SBL	[50]	`vb-sbl`
	BCS	[51]	`bcs`

Table 10: Complete list of solvers (Part 2: Beamformers, MUSIC, Matching Pursuit, Dipole Fitting, Deep Learning, Hybrid).

Category	Solver	Ref.	Primary Aliases
Beamformers (25)	MVAB	[52]	`mvab`
	LCMV	[7]	`lcmv`
	LCMV-MVPURE	[53]	`lcmv-mvpure`
	DICS	[23]	`dics`
	SMV	[54]	`smv`
	WNMV	[54]	`wnmv`
	HOCMV	[54]	`hocmv`
	SAM	[55]	`sam`
	EBB	[21]	`ebb`
	Unit Noise Gain	[56]	`unit-noise-gain`
	ESMV	[54, 57]	`esmv`
	ESMV-MVPURE	[53, 57]	`esmv-mvpure`
	Flex-ESMV	[30, 57]	`flex-esmv`
	Flex-ESMV-MVPURE	[53, 57]	`flex-esmv-mvpure`
	Adapt-Flex-ESMV	[30, 57]	`adapt-flex-esmv`
	Deblur-Flex-ESMV	[30, 57]	`deblur-flex-esmv`
	Safe-Flex-ESMV	[30, 57]	`safe-flex-esmv`
	Sharp-Flex-ESMV	[30, 57]	`sharp-flex-esmv`
	SSP-ESMV	[54, 57]	`ssp-esmv`
	IR-ESMV	[54, 57]	`ir-esmv`
	SSP-IR-ESMV	[54, 57]	`ssp-ir-esmv`
	MCMV	[58]	`mcmv`
	HOCMCMV	[58]	`hocmcmv`
	ReciPSIICOS	[59]	`recipsiicos`
	ReciPSIICOS-Whitened	[59]	`recipsiicos-whitened`
MUSIC / Subspace (13)	MUSIC	[60]	`music`
	RAP-MUSIC	[28]	`rap-music`
	TRAP-MUSIC	[28]	`trap-music`
	FLEX-MUSIC	[30]	`flex-music`
	FLEX-TRAP-MUSIC	[30]	`flex-trap-music`
	ExSo-MUSIC	[30]	`exso-music`
	SSM	[61]	`ssm`
	FLEX-SSM	[61]	`flex-ssm`
	AP	[62]	`ap`
	FLEX-AP	[62]	`flex-ap`
	ExhaustiveML	[31]	`exhaustive-ml`
	GreedyML	[31]	`greedy-ml`
	FLEX-GreedyML	[30]	`flex-greedy-ml`
Match. Pursuit (10)	OMP	[25]	`omp`
	FLEX-OMP	[30]	`flex-omp`
	CoSaMP	[26]	`cosamp`
	FLEX-CoSaMP	[30]	`flex-cosamp`
	ReMBo	[63]	`rembo`
	FLEX-ReMBo	[30]	`flex-rembo`
	Subspace Pursuit	[27]	`sp`
	FLEX-SP	[30]	`flex-sp`
	SubSMP	[64]	`subsmp`
	ISubSMP	[64]	`isubsmp`
Dipole Fit.	ECD	[1]	`ecd`
	SESAME	[29]	`sesame`
Deep Learn.†	FC (ESINet)	[12]	`fc`, `esinet`
	CovCNN	[13]	`covcnn`
	CovCNN-KL	[13]	`covcnn-kl`
	CovCNN-KL-FlexOMP	[13]	`covcnn-kl-flexomp`
	LSTM	[12]	`lstm`
	CNN	[12]	`cnn`
Hybrid	Chimera	[30]	`chimera`
	Hydra	[30]	`hydra`
	APSE	[30]	`apse`
Baseline	Random Noise	—	`random-noise`

†Requires PyTorch.