FaIMS: A fast algorithm for the inverse medium problem with multiple frequencies and multiple sources for the scalar Helmholtz equation

S. Chaillat G. Biros

Abstract

We consider the inverse medium problem, for the low-frequency time-harmonic wave equation with broadband and multi-point illumination in the low frequency regime. This model finds many applications in science and engineering (e.g., seismic imaging, non-destructive evaluation, and optical tomography).

We formulate the problem using a Lippmann-Schwinger formulation, which we discretize using a quadrature method. We consider small perturbations of the background medium and we invert the Born approximation. To solve this inverse problem, we use a least squares formulation that is regularized with the truncated Singular Value Decomposition (SVD).

If N_ω is the number of excitation frequencies, N_s the number of incoming waves, N_d the number of detectors, and N the discretization size of the medium perturbation, a dense singular value decomposition of the overall input-output map costs $[\min(N_s N_\omega N_d, N)]^2 \times \max(N_s N_\omega N_d, N)$ operations. We have developed an approximate SVD method that reduces the cost of the factorization to $\mathcal{O}(N N_\omega N_d + N N_\omega N_s)$ thus, providing orders of magnitude improvements over a black-box dense SVD. The storage is proportional to the product of N and the rank of the input-output map. If a fast multipole method is used for the forward scattering problem, the overall work complexity of the inversion scheme can be reduced to $\mathcal{O}(N N_\omega)$. We provide numerical results that demonstrate the scalability of the method.

1 Introduction

We describe a numerical algorithm for the Born approximation formulation of the inverse medium problem in scalar scattering.[10] Given $N_s N_\omega$ incident ("illumination") fields $\{u(\mathbf{r}; s, \omega)\}_{s=1,\omega=1}^{N_s,N_\omega}$ (where s indexes the spacial location of the source of the incident field and ω indexes its frequency), we record the scattered field $\boldsymbol{\phi}(\mathbf{r}_d; s, \omega)$ at N_d detector locations $\{\mathbf{r}_d\}_{d=1}^{N_d}$ and we seek to recover the medium perturbation $\eta(\mathbf{r})$ by solving

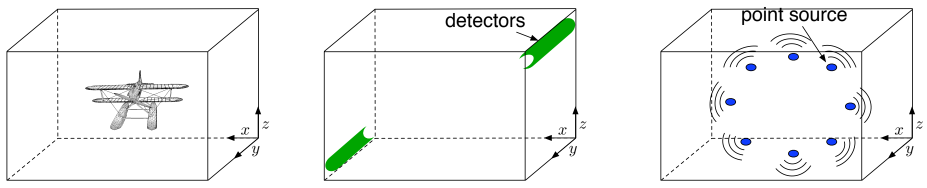
$$\phi(\mathbf{r}_d; s, \omega) = \int_H G(\mathbf{r}_d, \mathbf{r}; \omega)\eta(\mathbf{r})u(\mathbf{r}; s, \omega)\, d\mathbf{r} \tag{1}$$

for η.

This is a Lippmann-Schwinger scattering equation, where $G(\cdot, \cdot; \omega)$ is the Green's function (here, for a homogeneous background medium) at frequency ω, H is the support of η, and $\mathbf{r}$ is a point in H. Upon discretization using N quadrature points, we have

$$\phi(\mathbf{r}_d; s, \omega) = \sum_{j=1}^{N} G(\mathbf{r}_d, \mathbf{r}_j; \omega)\eta(\mathbf{r}_j)u(\mathbf{r}_j; s, \omega), \tag{2}$$

where the quadrature weights have been absorbed in $\eta(\mathbf{r}_j)$ (and by using "=" we ignore the quadrature discretization error)[1]. In the rest of the paper, ϕ will be generated by point scatterers located at the quadrature points $\{\mathbf{r}_j\}_{j=1}^N$ with scattering strengths $\{\,\eta(\mathbf{r}_j)\,\}_{j=1}^N$. (The integrand is smooth if the detectors and sources are not located in H.) The problem is summarized in Figure 1.



Complex unknown object — Detectors placed on arbitrary geometries — Spherical wave sources placed at arbitrary locations

Figure 1: *We propose an algorithm for the Born approximation of the inverse medium problem. For simplicity, we assume that the medium perturbation is represented by a set of point scatterers in a 3-D domain H. The data is measurements of scattered fields due to several different incident fields. In this paper, the incident fields are generated by point sources that illuminate that region of interest—possibly at multiple frequencies. (Of course, one can use plane waves or any other incident field.) Both sources and detectors can be located in arbitrary positions.*

Since (2) is linear on η, we can write

$$\mathbf{M}\boldsymbol{\eta} = \boldsymbol{\phi}.$$

We want to “invert” it using a truncated SVD algorithm for

$$\min_{\boldsymbol{\eta}} \|\mathbf{M}\boldsymbol{\eta} - \boldsymbol{\phi}\|_2.$$

One approach is to form $\mathbf{M}$ and then use a dense SVD factorization [13]. But in our formulation, $\mathbf{M}$ is a dense matrix. A dense SVD is prohibitively expensive[2] because its work complexity is $[\min(N_s N_\omega N_d, N)]^2 \times \max(N_s N_\omega N_d, N)$. An alternative approach is to use a Krylov iterative method like the LSQR and Conjugate Gradient (CG) method for the normal equations [9, 24]. To analyze the cost of such an iterative solver, let us define the cost of the forward scattering solver as $k_f(N, N_d)$. Then, the cost of an iterative method would be $N_s N_\omega k_f(N_d, N)$ per matvec. The number of iterations will depend on the scaling and clustering of M. For example, CG would scale with the condition number (after truncation) of M, and therefore such an approach will be expensive as well. Preconditioned CG or LSQR methods are typically based on limited-memory BFGS or Lanczos preconditioners [2, 14, 18]. However, constructing such preconditioners has similar complexity with inverting $\mathbf{M}$ [22]. Yet another approach is to use spectral preconditioning methods [1] or multigrid [6, 3]. The former requires regular geometries. The latter is a viable alternative, which we will not explore in this paper.

Contributions. Our goal is to design an algorithm that reconstructs $\boldsymbol{\eta}$ and scales “well” with N, N_ω, N_s and N_d, for the low frequency regime. Our main contribution is the construction of an approximate singular value decomposition for $\mathbf{M}$ based on the following algorithmic components:

[1]Assuming that sources and detectors are well separated, the Green’s function is $\mathcal{C}^\infty$, with no singularity. Therefore standard quadratures can be used.

[2]For example, if $N_\omega = 10$, $N_s = 100$, $N_d = 10^2$, and $N = 100^3$, we will need over one month of computation to compute the SVD on a single core 2 Gigaflops/sec machine.

- a rank-revealing randomized factorization (the algorithm proposed in [20]);
- preprocessing of the incident field u using an SVD to transform the incoming field and data and reduce the dimension of N_s;[3]
- and a new recursive SVD that can be used to approximately compute the SVD of $\mathbf{M} = [\mathbf{M}_1\ \mathbf{M}_2]^t$ given the SVDs of $\mathbf{M}_1$ and $\mathbf{M}_2$.

Using these components, we construct an approximate SVD factorization for $\mathbf{M}$ whose, given the incident field, the total work complexity is $\mathcal{O}(N_\omega k_f(N, N_d) R_s$, where R_s depends on the problem geometry and the maximum frequency but is independent of N_s, asymptotically. In our implementation, we use a direct evaluation for the scattered field$k_f(N, N_d) = NN_d$. Using a fast multipole acceleration the complexity can be reduced to $k_f(N, N_d) = N + N_d$ for the low frequency. We test our algorithm on problems in which the size of the scatterer varies from $1/10$ to five wavelengths. Our algorithm supports arbitrary distributions of sources, detectors and frequencies.

The main idea goes as follows. First, we reduce the number of incident fields from N_s to R_s using the randomized SVD. Then, we decompose $\mathbf{M}$ into N_ω smaller submatrices $\mathbf{M}^\omega$ of size $R_s N_d \times N$ $(1 \leq \omega \leq N_\omega)$. We compute the SVD of each small matrix by using the randomized SVD. We apply a low rank approximation whenever possible, leading to a compression of the matrix and a speed-up of the computations. We combine the SVDs of the $\mathbf{M}^\omega$ to approximate the SVD of $\mathbf{M}$ using the recursive SVD. This recursive SVD provides a precise characterization of the inverse problem and allows us to easily apply the pseudo-inverse of $\mathbf{M}$ to the date. We have termed the overall algorithm "*FaIMS*". This algorithm can handle efficiently a large number of sources and frequencies which lead to better resolution (Fig. 2). The storage complexity of FaIMS is $\mathcal{O}(N, R)$, where R is the rank of the approximation. FaIMS can achieve this cost since it does not require the assembly of $\mathbf{M}$. It only requires matrix-vector multiplications with submatrices of $\mathbf{M}$.

Limitations. FaIMS, works well when the detectors and the anomaly are well separated and we are in the low frequency regime, so that the mapping $\boldsymbol{\phi}^\omega = \mathbf{M}^\omega \boldsymbol{\eta}$ is low rank and thus, the storage and computational savings will be significant.

Another limitation is the use of the Born approximation for the reconstruction, which restricts our scheme to small perturbations of the background medium.

In this paper,We do not consider noise. By accounting for the noise, a more aggressive rank approximation can be used. In the presence of significant noise, it is unclear on whether FaIMS would be superior to a preconditioned Conjugate Gradients method for the normal equation. Also, we are considering neither sparse reconstruction ideas for η [7, 17] nor adaptive reconstruction schemes [4, 15]. We assume that the location of the detectors is independent of the source location and frequency. Finally, we commit an "inverse crime", since we use the same forward solver to both generate the data and invert for η.

We assume that we know the Green's function in analytic form so that the scattered field due to N scatterers can be evaluated at N_d detectors in $\mathcal{O}(N + N_d)$ work and storage using a fast multipole scheme [8, 28]. However, this is not a fundamental limitation of the algorithm. Any forward-scattering method with good complexity and accuracy features can be used in FaIMS without changing the behavior of the algorithm. In higher frequencies, such solvers are harder to construct.

Also, we haven't pursued randomization in frequencies. The input-output operator depends nonlinearly on the frequency and randomization techniques for linear operators are not directly applicable.

[3]This preprocessing step, is valid only in the case in which the detectors are the same for all of the sources.

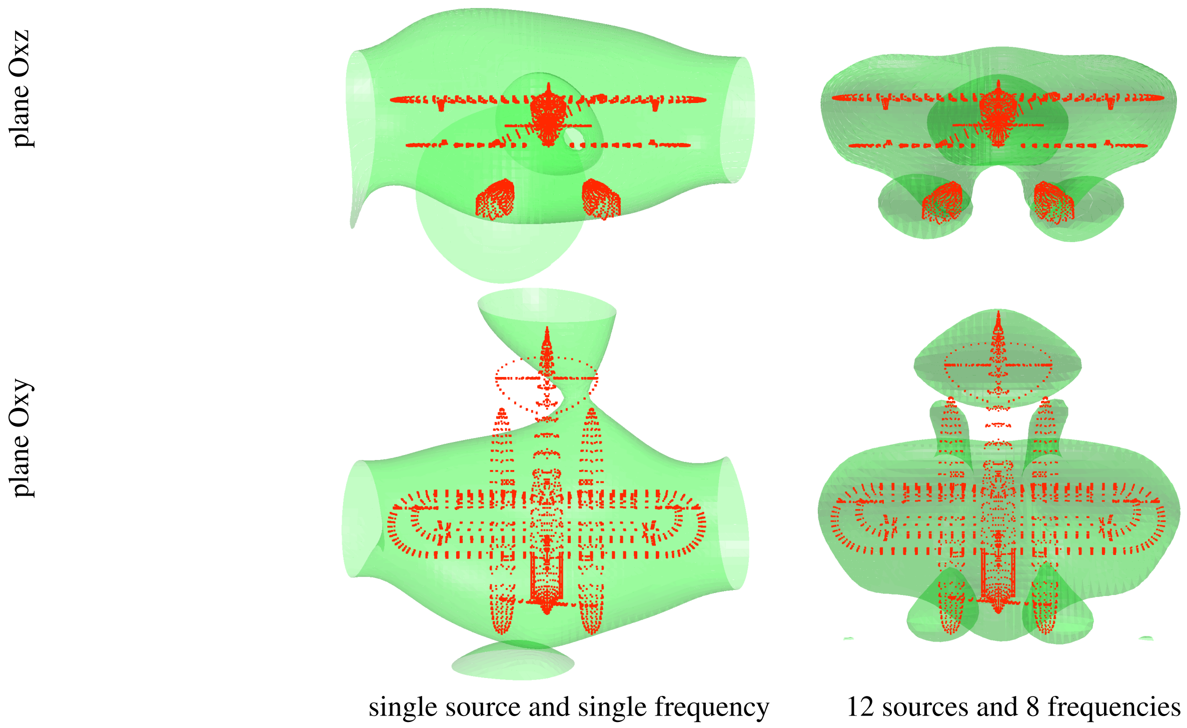

plane Oxz
plane Oxy
single source and single frequency
12 sources and 8 frequencies

Figure 2: *We report the isosurfaces $\boldsymbol{\eta} = 0.25\boldsymbol{\eta}_{max}$ for the scatterer model of a biplane with size one wavelength. The incident field for results on the left column is generated with one point sources excited at a single frequency. For the results of the right columns, we generate the incident field by using*12 *sources and* 8 *frequencies. The mesh size is* $N = [51]^3$ *and the scattered field is measured at* 162 *detectors located on a sphere enclosing the biplane. The results of the inversion with a single source and single frequency already enables to find the object location but the addition of more sources and frequencies permits to obtain better accuracy.*

Related work for problems with multiple sources. Our work has been inspired by the work in [21] and [25], in which a fast analytic SVD based on Fourier analysis is used for the case in which the sources and detectors are uniformly distributed on the boundary of a regular geometry (plane, cylinder, or sphere) and the scatterer is uniformly discretized in the domain of the corresponding regular geometry. The problems considered in [21] were reconstructions of absorption and diffusion coefficients for optical tomography problems formulated in the frequency domain. So the mathematical setup is similar to ours. Overall, the main advantage of FaIMS is that we can consider detectors on arbitrary geometries and point sources at arbitrary locations, only requires a fast forward scattering solver, and can be used with adaptive schemes.

Let us mention that there is work for forward and inverse problems with multiple sources in the geophysics community. However little is related to our work. An exception is the the idea of reducing the number of sources using linear combinations [19, 23, 17].

Finally, let us comment on randomized SVD-like decompositions for high-order tensors. Such decompositions are relevant because the forward operator $\mathbf{M}$ can be viewed as a third-order tensor that maps the sources and the medium perturbation to data. One could explore a randomized tensor decomposition [16], but we have not pursued this approach in this paper.

Outline. In Section 2, we state the problem formulation. In Section 3, we give a summary of the algorithm FaIMS. In Section 4 we present the SVD algorithms that are required in the overall method. Finally, in Section 6, we present numerical results for the reconstruction of various point scatterer locations.

Notation. In Table 1, we summarize all the symbols used throughout this article. We use roman letters to denote continuous scalar fields and operators, bold lower case letters to denote finite dimensional vectors, and bold upper case ones to denote finite dimensional linear operators.

ω	indexes the frequency of the incident field;
λ	wavelength $\lambda = 2\pi/k$;
$k(\mathbf{r})$	wavenumber;
k_0	background medium wavenumber;
$\boldsymbol{\eta}(\mathbf{r})$	perturbation of the background medium; $k^2(\mathbf{r}) = k_0^2 + \boldsymbol{\eta}(\mathbf{r})$;
G	Green's function of the homogeneous infinite medium characterized by k_0;
H	support of the anomaly $\boldsymbol{\eta}$ (typically a cube of L^3);
L	edge size of H;
N	number of points in H;
N_d	number of detectors;
N_ω	number of incident wave frequencies $(\omega_1, \dots, \omega_{N_\omega})$;
N_s	number of spherical wave source locations (point sources);
$\mathbf{M}$	overall input-output operator ($\in \mathbb{R}^{N_s N_\omega N_d \times N}$);
$\mathbf{r}$	position in space $\mathbf{r} = (x, y, z)$;
$\mathbf{r}_d$	detector locations;
s	indexes the location of the source of the incident field;

Table 1: *List of the main symbols used in this Article.*

2 Definition of the inversion formula

The time-harmonic scalar wave equation is given by

$$\nabla^2 v(\mathbf{r}) + k^2(\mathbf{r}) v(\mathbf{r}) = -s(\mathbf{r}), \tag{3}$$

where s is the source term and k is the wavenumber defined by $k^2(\mathbf{r}) = \omega^2 c^{-2}(\mathbf{r})$. We consider the case in which k is defined by $k^2(\mathbf{r}) = k_0^2 + \eta(\mathbf{r})$, where $k_0^2 = \omega^2 c_0^{-2}$ is the parameter characterizing the background and η is the unknown perturbation of the background medium. If we denote the total scattered field $\mathbf{v} = \boldsymbol{\phi} + \mathbf{u}$, eq. (3) becomes

$$\nabla^2 \phi(\mathbf{r}) + k_0^2 \phi(\mathbf{r}) = -\eta(\mathbf{r})(\phi(\mathbf{r}) + u(\mathbf{r})). \tag{4}$$

Using the Born approximation, we neglect $-\eta(\mathbf{r})\phi(\mathbf{r})$ and we obtain

$$\nabla^2 \phi(\mathbf{r}) + k_0^2 \phi(\mathbf{r}) = -\eta(\mathbf{r}) u(\mathbf{r}). \tag{5}$$

We introduce the free-space Green's function G given by

$$G(\mathbf{r}, \mathbf{r}') = \frac{\exp(ik_0|\mathbf{r} - \mathbf{r}'|)}{4\pi|\mathbf{r} - \mathbf{r}'|}. \tag{6}$$

The solution of eq. (5) can be obtained as a convolution with G [11] and is given by

$$\phi(\mathbf{r}_d; s, \omega) = \int_H G(\mathbf{r}_d, \mathbf{r}; \omega)\eta(\mathbf{r})u(\mathbf{r}; s, \omega)d^3\mathbf{r}. \tag{7}$$

Equation (7) is the forward problem, in which given $\boldsymbol{\eta}$ we can compute $\boldsymbol{\phi}$. In the inverse problem, we seek to recover the anomaly $\boldsymbol{\eta}$ ($\in \mathbb{R}^N$) given $\phi(\mathbf{r}_d; s, \omega)$, a set of measurements generated by $N_s N_\omega$ known incident fields and measured at N_d detector locations.

For a given source s, frequency ω and detector d (if the quadrature weights are absorbed in $\boldsymbol{\eta}$ and we ignore the discretization error) equation (7) becomes

$$\boldsymbol{\phi}_{ds}^{\omega} = \sum_{j}^{N} \mathbf{G}_{dj}^{\omega}\mathbf{u}_{js}^{\omega}\boldsymbol{\eta}_j,$$

which we write in a matrix form $\boldsymbol{\phi} = \mathbf{M}\boldsymbol{\eta}$. To invert for $\boldsymbol{\eta}$, we first compute the SVD of $\mathbf{M}$, which we then use to apply the pseudo-inverse of $\mathbf{M}$ on $\boldsymbol{\phi}$. In our experiments the incident field is $u(\mathbf{r}; s, \omega) = G(\mathbf{r}; s, \omega)$, a spherical wave corresponding to a point source.

3 Summary of FaIMS

Before presenting the details of FaIMS (section 5) let us outline the basic steps in the algorithm. Recall that our goal is to avoid the $N_s N_\omega N_d N$ complexity of a single matvec.

We introduce a preprocessing step in which use a principal component analysis type compression (PCA) to reduce the number of incident fields. This step is analogous to source recombination techniques that have appeared in the literature. In the next two steps, we compute the inputs for the recursive SVD (section 4.2): the SVD of $\mathbf{M}^\omega$ for a fixed frequency ω. Then we use the recursive SVD to combine the individual SVDs for each frequency. Overall, FaIMS has four main steps:

Step A: **Incident field SVD.** We preprocess the incident field $\mathbf{u}^\omega$ using the randomized SVD [20] (section 4.1) to compute a PCA-like reduction of the number of incoming fields $\mathbf{u}^\omega$ and data $\boldsymbol{\phi}^\omega$ and reduce the number of sources from N_s to R_s.

Step B: **Forward problem SVD.** For every frequency, we compute the SVD of the Green's function $\mathbf{G}^\omega$ by applying the randomized SVD. Each matrix $\mathbf{G}^\omega \in \mathbb{R}^{N_d \times N}$ is approximated by a matrix of rank lower or equal to R_g^ω. [4]

Step C: **Single frequency-multiple sources SVD**. Once we have computed the SVDs of the Green's functions, we can combine them for a fixed frequency using the algorithm presented in section 5. Each matrix $\mathbf{M}^\omega \in \mathbb{R}^{R_s R_g^\omega \times N}$ is approximated by a matrix of rank lower or equal to R_ω.

Step D: **Overall SVD**. Using the results of step C, we apply the recursive SVD (section 4.2) to obtain the SVD of the complete system matrix $\mathbf{M}$.

[4]This step seems specific to our forward problem formulation; it is not. If the detectors are well separated from the H, then the forward operator will be low rank. So a finite element or finite-difference-based forward solver can be used in place of $\mathbf{G}$.

Once the SVD of $\mathbf{M}$ is computed, we can use it to precondition an iterative method or to apply the pseudoinverse of $\mathbf{M}$ to the data.

In the following section, we present the randomized and recursive algorithms required in the inversion procedure but not dependent of the inverse problem. The randomized SVD allows fast approximation of low-rank matrices, using a matrix-vector multiplication. Then the recursive SVD enables to obtain the SVD of the matrix $\mathbf{M} = [\mathbf{M}_1\mathbf{M}_2]^t$ given the SVDs of $\mathbf{M}_1$ and $\mathbf{M}_2$. This algorithm is faster than a standard SVD when low rank approximations of $\mathbf{M}_1$ and $\mathbf{M}_2$ are available. In Table 3, we summarize the notation for the approximate ranks of the operators that appear in FaIMS.

step	A	B	C	D
approximate rank	R_s	R_g^ω	R_ω	R
size initial matrix	$N_s \times N$	$N_d \times N$	$R_s R_g^\omega \times N$	$N_\omega R^\omega \times N$

Table 2: *Notation for the approximate ranks of operators that appear in the four steps of FaIMS.*

4 Randomized and recursive SVDs

4.1 Randomized SVD

The last two decades, there has been a significant amount of work on randomized algorithms for low rank approximations of matrices. In our work, we use the algorithm proposed in [20]. We briefly summarize its main steps here for completeness. Let $\mathbf{M}$ a matrix of size $m \times n$. Then the randomized SVD method computes r, Φ, Λ and Ψ, with singular values $\sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_{\min(m,n)}$ such that

$$\|\Phi\Lambda\Psi^* - \mathbf{M}\| \leq \varepsilon\sigma_1 \leq \sigma_{r+1}. \tag{8}$$

Here σ_{r+1} is the $r+1$ singular value of $\mathbf{M}$ and * denotes the conjugate transpose. Φ and Ψ are matrices of size respectively $m \times r$ and $n \times r$, where r are the number of singular values greater than a prescribed accuracy ε. Λ is the diagonal matrix of size $r \times r$ containing the corresponding singular values.

It is shown in [20] that (8) is equivalent to find $\mathbf{Q}$ such that

$$\|\mathbf{MQQ}^* - \mathbf{M}\| \leq \sigma_{\mathbf{r+1}}. \tag{9}$$

In [20], the authors showed that $\mathbf{MQQ}^*$, where $\mathbf{Q}$ is orthonormal, is a good approximation to $\mathbf{M}$ if there exist two matrices $\mathbf{G}$ and $\mathbf{S}$ such that the product $\mathbf{QS}$ is a good approximation to $\mathbf{M}^*\mathbf{G}^*$ and there exist a matrix $\mathbf{F}$ such that $\|\mathbf{F}\|$ is small and $\mathbf{FGM}$ is a good approximation to $\mathbf{M}$. In [20], $\mathbf{G}$ is chosen to be a Gaussian random matrix. Using the properties of the SVD, a good approximation $\mathbf{QS}$ to $\mathbf{M}^*\mathbf{G}^*$ can be defined. The algorithm is summarized below in Algorithm 1 (we use MATLAB notation). To avoid the need to precompute the matrix rank, we use an error estimate (Algorithm 1). If the complexity of the matvec is $\mu(m,n)$, then the complexity of the algorithm is $\mathcal{O}(\mu(m,n)\ell + \mu(n,m)r + m\ell^2 + nr^2)$. Assuming $m < n$, the complexity is $\mathcal{O}(\mu(m,n)\ell + nr^2)$. If we have a dense matrix, the complexity is $\mathcal{O}(mn\ell)$.

It follows that the overall complexity of this approximate factorization is $\mathcal{O}(\ell nm)$ for work $\mathcal{O}(rm + rn)$ for storage.

Algorithm 1 *Randomized SVD*

```
1: Inputs: $\mathbf{M} \in \mathbb{R}^{m\times n}$, $\varepsilon$.
2: Outputs: approximate rank $r$, $\mathbf{\Phi}$, $\mathbf{\Lambda}$, $\mathbf{\Psi}$ such that $\mathbf{M} \approx \mathbf{\Phi\Lambda\Psi}^*$.
3: $r = 1$
4: $\ell = r + 20$
5: Define $\mathbf{G}$ (Gaussian random matrix $\mathbf{G} \in \mathbb{R}^{\mathbf{n}\times\ell}$)
6: $\mathbf{R} = \mathbf{MG}$
7: $[\mathbf{U_r}, \mathbf{S_r}, \mathbf{V_r}] = \mathrm{SVD}(\mathbf{R})$
8: $\mathbf{Q}(:, \mathbf{1} : \mathbf{r}) = \mathbf{U_r}(:, \mathbf{1} : \mathbf{r})$
9: error estimate $= \|\mathbf{MG} - \mathbf{Q}^\star\mathbf{QG}\|$
10: **if** error estimate $> \varepsilon \times \mathbf{S_r}(\mathbf{1}, \mathbf{1})$ **then**
11:     Increase $r$ (e.g., $r = r + 0.05m$)
12:     Goto 4
13: **end if**
14: $\mathbf{T} = \mathbf{M}^\star\mathbf{Q}$
15: $[\mathbf{\Phi}, \mathbf{\Lambda}, \mathbf{W}] = \mathrm{SVD}(\mathbf{T}^\star)$
16: $\mathbf{\Psi} = \mathbf{Q\Phi}$
```

4.2 Recursive SVD

Let $\mathbf{M}$ be a matrix of size $2m \times n$ and $\mathbf{M}_1$ and $\mathbf{M}_2$ be two matrices of size $m \times n$. We wish to construct the SVD of $\mathbf{M}$ given

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \end{bmatrix} \tag{10}$$

and we assume that we know the SVDs of $\mathbf{M}_1$ and $\mathbf{M}_2$: $\mathbf{M}_1 = \mathbf{\Phi}_1\mathbf{\Lambda}_1\mathbf{\Psi}_1^*$ and $\mathbf{M}_2 = \mathbf{\Phi}_2\mathbf{\Lambda}_2\mathbf{\Psi}_2^*$. Let also r be the rank of $\mathbf{M}_1$ and $\mathbf{M}_2$ (this need not to be the same, but for simplicity we assume that $\mathbf{M}_1$ and $\mathbf{M}_2$ have the same rank). We seek for $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{\Sigma}$ to be so that $\mathbf{M} = \mathbf{U\Sigma V}^\star$. We show that instead of solving the initial system of size $2m \times n$ we can solve a smaller equivalent system of size $2r \times 2r$. To define the algorithm, we introduce a block decomposition of $\mathbf{U}$ and $\mathbf{\Sigma}$:

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{bmatrix} \text{ and } \mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_2 \end{bmatrix}. \tag{11}$$

$\mathbf{U}_i$ is a matrix of size $m \times r$ and $\mathbf{\Sigma}_i$ is a diagonal matrix of size $r \times r$ ($i = 1, 2$). The SVD of the equivalent system gives $\mathbf{U}$ and $\mathbf{\Sigma}$. Once $\mathbf{U}$ and $\mathbf{\Sigma}$ have been reconstructed, we can compute an approximation to $\mathbf{V}$.

Computation of U and $\mathbf{\Sigma}$. By definition, $\mathbf{MM}^\star\mathbf{U} = \mathbf{U\Sigma}^2$. Furthermore, substituting the SVDs of $\mathbf{M}_1$ and $\mathbf{M}_2$ and (11) into (10), we obtain

$$\begin{bmatrix} \mathbf{\Phi}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Phi}_2 \end{bmatrix} \begin{bmatrix} \mathbf{\Lambda}_1^2 & \mathbf{Z_1}^*\mathbf{Z_2} \\ \mathbf{Z_2}^*\mathbf{Z_1} & \mathbf{\Lambda}_2^2 \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}_1^* & \mathbf{0} \\ \mathbf{0} & \mathbf{\Phi}_2^* \end{bmatrix} \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_1^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_2^2 \end{bmatrix} \tag{12}$$

where $\mathbf{Z_i} = \mathbf{\Psi}_i\mathbf{\Lambda}_i \in \mathbb{R}^{n\times r}$ ($i = 1, 2$). Our key variable change is to introduce $\mathbf{c}_k$ ($k = 1, 2$) such that

$$\mathbf{U}_k = \mathbf{\Phi}_k\mathbf{c}_k. \tag{13}$$

Then using (13), (12) becomes

$$\left[\begin{array}{cc} \mathbf{\Lambda}_1^2 & \mathbf{Z_1}^*\mathbf{Z_2} \\ \mathbf{Z_2}^*\mathbf{Z_1} & \mathbf{\Lambda}_2^2 \end{array}\right]\left[\begin{array}{c} \mathbf{c}_1 \\ \mathbf{c}_2 \end{array}\right] = \left[\begin{array}{c} \mathbf{c}_1 \\ \mathbf{c}_2 \end{array}\right]\left[\begin{array}{cc} \mathbf{\Sigma}_1^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_2^2 \end{array}\right]. \tag{14}$$

Let us define $\mathbf{A}$ as

$$\mathbf{A} = \left[\begin{array}{cc} \mathbf{\Lambda}_1^2 & \mathbf{Z_1}^*\mathbf{Z_2} \\ \mathbf{Z_2}^*\mathbf{Z_1} & \mathbf{\Lambda}_2^2 \end{array}\right], \tag{15}$$

then the eigenvalues and eigenvectors of $\mathbf{A}$ are the vectors $\mathbf{c}_i$ and the coefficients Σ_i $(i = 1, 2)$.

Computation of V. Using the SVD of $\mathbf{M}$ we have $\mathbf{M}^\star\mathbf{U} = \mathbf{V\Sigma}$, which using (10) and (13) is equivalent to

$$\left[\begin{array}{c} \mathbf{Z}_1 \\ \mathbf{Z}_2 \end{array}\right]\left[\begin{array}{c} \mathbf{c}_1 \\ \mathbf{c}_2 \end{array}\right] = \mathbf{V\Sigma}.$$

Finally, the final expression for $\mathbf{V}$ is given by

$$\boxed{\mathbf{V}_j = \frac{1}{\Sigma_j}\sum_{k=1}^{2}\mathbf{Z}_k\mathbf{c}_{k,j}, \qquad 1 \le j \le 2r.} \tag{16}$$

where $\mathbf{V}_j \in \mathbb{R}^n$ denotes the j_{th} column of $\mathbf{V}$. If we expect that there exists a good low rank approximation for $\mathbf{A}$ with rank R, we can use the Algorithm 1 to compute the SVD of $\mathbf{A}$. As a result $1 \le j \le R$. If Σ_j is very small, the above equation may be unstable. We prefer to compute

$$\Sigma_j\mathbf{V}_j = \sum_{k=1}^{2}\mathbf{Z}_k\mathbf{c}_{k,j}, \qquad 1 \le j \le R. \tag{17}$$

Notice that $\Sigma_j\mathbf{V}_j$ are orthogonal but not orthonormal. In Algorithm 2, we summarize the overall recursive SVD.

Algorithm 2 *Recursive SVD*

```
1: Inputs: $\mathbf{M}_1$ and $\mathbf{M}_2$.
2: Outputs: $\mathbf{U}, \mathbf{\Sigma}$ and $\mathbf{V}$ such that $\mathbf{M} = \mathbf{U\Sigma V}^\star$.
3: Compute $\mathbf{\Phi}_k$, $\mathbf{\Lambda}_k$ and $\mathbf{\Psi}_k$ $(k = 1, 2)$ such that $\mathbf{M}_k = \mathbf{\Phi}_k\mathbf{\Lambda}_k\mathbf{\Psi}_k^*$ (with $r$ denoting the rank of the $\mathbf{M}_k$).
4: Compute $\mathbf{A}$ defined by $\mathbf{A}_{k\ell} = \mathbf{Z}_k^\star\mathbf{Z}_\ell, \quad 1 \le k, \ell \le 2$.
5: Compute the eigenvalues $\mathbf{\Sigma}$ and eigenvectors $\mathbf{c}$ of $\mathbf{A}$.
6: for $j = 1 \dots R$ do
7:     Compute $\Sigma_j\mathbf{V}_j = \sum_{\ell=1}^{p}\mathbf{Z}_\ell\mathbf{c}_{\ell,j}$.
8: end for
9: for $\ell = 1 \dots 2$ do
10:    Compute $\mathbf{U}_\ell = \mathbf{\Phi}_\ell\mathbf{c}_\ell$.
11: end for
```

Complexity of the recursive SVD. First, we compute the SVD of the matrices $\mathbf{A}$ of size $2r \times 2r$. The cost of the SVD is reduced to $\mathcal{O}(Rr^2)$ if we expect that there exists a good low rank approximation for $\mathbf{A}$ with rank R. The computation of V costs $\mathcal{O}(Rnr)$. The computation of $\mathbf{U}$ costs $\mathcal{O}(mrR)$. Therefore, the overall complexity of this algorithm is $\mathcal{O}(Rr(r+n+m))$.

The memory requirements for the computation of the pseudo-inverse include storing one matrix $\mathbf{U}$ of size $m \times R$, one matrix $\mathbf{V}$ of size $n \times R$ and one vector $\mathbf{\Sigma}$ of size R. As a result, the memory requirements for the computation of the pseudo-inverse is $\mathcal{O}(R(m+n))$.

Accuracy of the recursive SVD. Here we quantify the error introduced by this recursive SVD on the singular values. In Lemma 4.1 we show that this error can be bounded. We were not able to proof a similar stability for the singular values [26, 27].

Lemma 4.1 *Let $\mathbf{M}_1$ and $\mathbf{M}_2 \in \mathbb{R}^{m\times n}$ be two matrices whose rank is equal to $r+1$. Assume that the $r+1$th singular values of $\mathbf{M}_1$ and $\mathbf{M}_2$ denoted λ^1_{r+1} and λ^2_{r+1} are equal to ε. We denote $\mathbf{M}_\varepsilon$ the approximation of $\mathbf{M}$ obtained by applying the recursive SVD presented in Algorithm 2. If we use the $r+1$ non-zero singular values of $\mathbf{M}_1$ and $\mathbf{M}_2$, then $\mathbf{M} = \mathbf{M}_\varepsilon$. On the other hand, if use only the r first singular values of $\mathbf{M}_1$ and $\mathbf{M}_2$, the error introduced on the singular values λ_k of $\mathbf{M}_\varepsilon$ is bounded by*

$$\boxed{|\lambda_k(\mathbf{M}) - \lambda_k(\mathbf{M}_\varepsilon)| \le 2\varepsilon\Big[2\max(\lambda^1_1, \lambda^2_1) + \varepsilon\Big]}, \quad 1 \le k \le 2m \le n.$$

Proof: We write $\mathbf{M}_1 = \mathbf{\Phi}_1\mathbf{\Lambda}_1(\mathbf{\Psi}_1)^* + \varepsilon\mathbf{\Phi}^\varepsilon_1(\mathbf{\Psi}^\varepsilon_1)^*$ and $\mathbf{M}_2 = \mathbf{\Phi}_2\mathbf{\Lambda}_2(\mathbf{\Psi}_2)^* + \varepsilon\mathbf{\Phi}^\varepsilon_2(\mathbf{\Psi}^\varepsilon_2)^*$, i.e. we separate $\mathbf{M}_1$ and $\mathbf{M}_2$ into two submatrices. It is immediate that

$$\mathbf{M}\mathbf{M}^* = \mathbf{\Phi}\mathbf{B}\mathbf{B}^*\mathbf{\Phi}^* + \varepsilon\mathbf{\Phi}\mathbf{B}\mathbf{\Psi}^\varepsilon(\mathbf{\Phi}^\varepsilon)^* + \varepsilon\mathbf{\Phi}^\varepsilon(\mathbf{\Psi}^\varepsilon)^*\mathbf{B}^*\mathbf{\Phi}^* + \varepsilon^2\mathbf{\Phi}^\varepsilon(\mathbf{\Psi}^\varepsilon)^*\mathbf{\Psi}^\varepsilon(\mathbf{\Phi}^\varepsilon)^* \quad (18)$$

where

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_2 \end{bmatrix}, \mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Phi}_2 \end{bmatrix} \text{ and } \mathbf{\Phi}^\varepsilon = \begin{bmatrix} \mathbf{\Phi}^\varepsilon_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Phi}^\varepsilon_2 \end{bmatrix},$$

$$\mathbf{\Psi} = \begin{bmatrix} \mathbf{\Psi}_1 & \mathbf{\Psi}_2 \end{bmatrix}, \mathbf{\Psi}^\varepsilon = \begin{bmatrix} \mathbf{\Psi}^\varepsilon_1 & \mathbf{\Psi}^\varepsilon_2 \end{bmatrix} \text{ and } \mathbf{B} = \mathbf{\Lambda}\mathbf{\Psi}^*.$$

Using that $\mathbf{\Phi}^*\mathbf{U} = \mathbf{c}$, $\mathbf{M}\mathbf{M}^*\mathbf{U} = \mathbf{U}\mathbf{\Sigma}^2$ is equivalent to

$$\mathbf{B}\mathbf{B}^*\mathbf{c} + \varepsilon\mathbf{B}\mathbf{\Psi}^\varepsilon\mathbf{c} + \varepsilon(\mathbf{\Psi}^\varepsilon)^*\mathbf{B}^*\mathbf{c} + \varepsilon^2(\mathbf{\Psi}^\varepsilon)^*\mathbf{\Psi}^\varepsilon\mathbf{c} = \mathbf{c}\mathbf{\Sigma}^2.$$

Finally using (15), we obtain that $\mathbf{\Phi}^*\mathbf{U} = \mathbf{c}$, $\mathbf{M}\mathbf{M}^*\mathbf{U} = \mathbf{U}\mathbf{\Sigma}^2$ is equivalent to

$$(\mathbf{A}+\mathbf{E})\mathbf{c} = \mathbf{c}\mathbf{\Sigma}^2 \text{ where } \mathbf{E} \text{ is given by } \mathbf{E} = \varepsilon\mathbf{B}\mathbf{\Psi}^\varepsilon + \varepsilon(\mathbf{\Psi}^\varepsilon)^*\mathbf{B}^* + \varepsilon^2(\mathbf{\Psi}^\varepsilon)^*\mathbf{\Psi}^\varepsilon.$$

From Theorem 8.6.2 in [13] we know that the error between the singular values of $\mathbf{A}+\mathbf{E}$ and the ones of $\mathbf{A}$ is bounded, i.e.

$$|\lambda_k(\mathbf{A}+\mathbf{E}) - \lambda_k(\mathbf{A})| \le \|\mathbf{E}\|_2, \quad k = 1:2m.$$

We have

$$\|\mathbf{E}\|_2 \le 2\varepsilon\|\mathbf{B}\|_2\|\mathbf{\Psi}^\varepsilon\|_2 + \varepsilon^2\|\mathbf{\Psi}^\varepsilon\|^2_2 \le 2\varepsilon\max(\lambda^1_1, \lambda^1_2)\|\mathbf{\Psi}\|_2\|\mathbf{\Psi}^\varepsilon\|_2 + \varepsilon^2\|\mathbf{\Psi}^\varepsilon\|^2_2.$$

Since Ψ_1, Ψ_2 and $\Psi^\varepsilon_1, \Psi^\varepsilon_2$ are orthonormal matrices, $\|\mathbf{\Psi}\|_2 \le \|\Psi_1\|_2 + \|\Psi_2\|_2$, and $\|\mathbf{\Psi}\|^\varepsilon_2 \le \|\Psi_1\|^\varepsilon_2 + \|\Psi_2\|^\varepsilon_2$, it follows that

$$|\lambda_k(\mathbf{A}+\mathbf{E}) - \lambda_k(\mathbf{A})| \le 2\varepsilon\Big[2\max(\lambda^1_1, \lambda^2_1) + \varepsilon\Big].$$

5 FaIMS

In this section, we present the complete algorithm for the reconstruction of $\boldsymbol{\eta}$. We recall that, for a fixed frequency ω, the forward problem is given by

$$\phi^{\omega}_{ds} = \sum_{j=1}^{N} \mathbf{G}^{\omega}_{dj} \boldsymbol{\eta}_j \mathbf{u}^{\omega}_{js}, \quad d = 1, \ldots, N_d, \quad s = 1, \ldots, N_s, \tag{19}$$

where ϕ^{ω}_{ds} is the set of measurement at the detector locations. We write (19) under a matrix form as $\phi^{\omega}_{s} = \mathbf{M}^{\omega}_{s}\boldsymbol{\eta}$ where $\mathbf{M}^{\omega}_{s} = \mathbf{G}^{\omega}\mathbf{u}^{\omega}_{s}$. The overall algorithm can be stated as follows:

Step A: For each frequency, reduce the number of incident fields.

Step B: For each frequency, compute the SVD of $\mathbf{G}^{\omega}$ using Algorithm 1.

Step C: Combine the SVDs of all sources for a fixed frequency using Algorithm 2.

Step D: Combine the SVDs of all frequencies using several steps of Algorithm 3.

Step E: Apply the pseudo-inverse of $\mathbf{M}$ to recover $\boldsymbol{\eta}$.

Inputs. We specify the domain (unit cube) size L defined in wavelengths λ units (H is the domain $[0; L]^3$), the source locations, the detector locations, the number of points N that will be used to discretize H, and the incident wave frequencies $\omega_{1,\ldots,N_\omega}$.

Step A: Reducing the number of incident fields. This is a preprocessing stage that requires the evaluation of $\phi(\mathbf{r})$ at the scatterer positions. The reduction is done for each frequency ω. If we introduce the SVD of $\mathbf{u}^{\omega} = \mathbf{\Phi}^{\omega}\mathbf{\Lambda}^{\omega}\mathbf{\Psi}^{\omega *}$ (where $\mathbf{u}^{\omega} = [\mathbf{u}^{\omega}_1 \ldots \mathbf{u}^{\omega}_{N_s}]^t \in \mathbb{R}^{N_s \times N}$) and use the orthonormality of the $\mathbf{\Psi}^{\omega}$, (19) becomes

$$\boxed{\phi^{\omega}\mathbf{\Psi}^{\omega}_{\ell} \approx \mathbf{G}^{\omega} \text{ diag } (\mathbf{\Phi}^{\omega}_{\ell}\Lambda^{\omega}_{\ell})\boldsymbol{\eta}}, \; 1 \leq \ell \leq R_s \text{ and } 1 \leq \omega \leq N_\omega. \tag{20}$$

where $\mathbf{\Phi}^{\omega}_{\ell}$ denotes the ℓ-th column of the matrix $\mathbf{\Phi}^{\omega} \in \mathbb{R}^{N \times R_s}$ and $\mathbf{\Psi}^{\omega}_{\ell}$ denotes the ℓ-th column of the matrix $\mathbf{\Psi}^{\omega} \in \mathbb{R}^{N_s \times R_s}$. That is, we have replaced diag $(\mathbf{u}^{\omega}_s) \in \mathbb{R}^{N \times N}$ with $\mathbf{D}^{\omega}_{\ell} =$ diag $(\mathbf{\Phi}^{\omega}_{\ell}\Lambda^{\omega}_{\ell}) \in \mathbb{R}^{N \times N}$; and we have transformed the data accordingly: ϕ^{ω}_{s} is replaced by $\mathbf{d}^{\omega}_{\ell} = \phi^{\omega}\mathbf{\Psi}^{\omega}_{\ell}$. If a low rank approximation is available for ϕ, R_s will be independent of N_s.[5]

Step B: Computation of the randomized SVD of $\mathbf{G}^{\omega}$. For each frequency ω, we compute the SVD of the matrix $\mathbf{G}^{\omega} = \mathbf{\Phi}^{\omega}_{g}\mathbf{\Lambda}^{\omega}_{g}\mathbf{\Psi}^{\omega *}_{g} \in \mathbb{R}^{N_d \times N}$ using the randomized SVD (section 4.1). We denote R^{ω}_{g} the rank of the low rank approximation of $\mathbf{G}^{\omega}$. R^{ω}_{g} depends on the approximation tolerance for the SVD and satisfies $R^{\omega}_{g} \leq N_d, N$.

[5]A similar approach could be followed for ϕ_{ds} to guide numerical rank selection in the building of the SVD of $\mathbf{M}$.

Step C: Combine the SVDs of all sources for a fixed frequency. To combine the SVDs of the R_s sources for a fixed frequency, we compute the SVD of

$$\begin{bmatrix} \mathbf{G}^\omega \mathbf{D}_1^\omega \\ \vdots \\ \mathbf{G}^\omega \mathbf{D}_{R_s}^\omega \end{bmatrix} \in \mathbb{R}^{R_s N_d \times N}, \tag{21}$$

where $\mathbf{G}^\omega \in \mathbb{R}^{N_d \times N}$ and $\mathbf{D}_\ell^\omega = \mathrm{diag}(\mathbf{\Phi}_\ell^\omega \Lambda_\ell^\omega) \in \mathbb{R}^{N \times N}$. Instead of computing directly the SVD of this large matrix, we use the SVD of $\mathbf{G}^\omega$ computed during step B; $\mathbf{G}^\omega = \mathbf{\Phi}_g^\omega \mathbf{\Lambda}_g^\omega \mathbf{\Psi}_g^{\omega *}$. More precisely, we compute the SVD of

$$\mathbf{B}^\omega = \begin{bmatrix} \mathbf{\Lambda}_g^\omega \mathbf{\Psi}_g^{\omega *} \mathbf{D}_1^\omega \\ \vdots \\ \mathbf{\Lambda}_g^\omega \mathbf{\Psi}_g^{\omega *} \mathbf{D}_{R_s}^\omega \end{bmatrix} \in \mathbb{R}^{R_s R_g^\omega \times N}$$

We compute the SVD of $\mathbf{B}^\omega$ with the randomized SVD. R_ω denotes its approximate rank. Finally, we transform the data accordingly, i.e. we replace

$$\begin{bmatrix} \mathbf{d}_1^\omega \\ \vdots \\ \mathbf{d}_{\mathbf{R_s}}^\omega \end{bmatrix} \in \mathbb{R}^{R_s N_d} \text{ by } \begin{bmatrix} \mathbf{\Phi}_g^{\omega *} \mathbf{d}_1^\omega \\ \vdots \\ \mathbf{\Phi}_g^{\omega *} \mathbf{d}_{\mathbf{R_s}}^\omega \end{bmatrix} \in \mathbb{R}^{R_s R_g^\omega}. \tag{22}$$

Step D: Recursion over frequencies. Finally we need to combine the N_ω SVDs of $\mathbf{M}^\omega$ (corresponding to N_ω frequencies) computing in step C. We apply the recursive SVD (section 4.2). R denotes the number of selected singular values (smaller than a prescribed tolerance). We define $\mathbf{T}$, $\mathbf{W}$ and $\mathbf{S}$ such that $\mathbf{M} = \mathbf{T}\mathbf{S}\mathbf{W}^\star$. $\mathbf{T}$ and $\mathbf{W}$ are two matrices of size $N_\omega R_s R_g^\omega \times R$ and $N \times R$ respectively. $\mathbf{S}$ is a diagonal matrix of size R. Instead of applying directly the recursive SVD to combine N_ω frequencies, we apply recursively this algorithm to combine two frequencies at each level of the recursion tree (see Fig. 3).

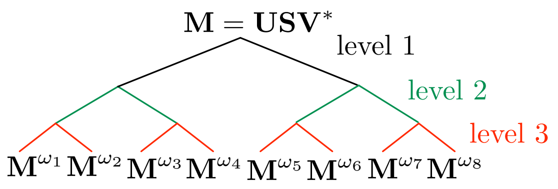

$\mathbf{M} = \mathbf{USV}^*$
level 1
level 2
level 3
$\mathbf{M}^{\omega_1}$ $\mathbf{M}^{\omega_2}$ $\mathbf{M}^{\omega_3}$ $\mathbf{M}^{\omega_4}$ $\mathbf{M}^{\omega_5}$ $\mathbf{M}^{\omega_6}$ $\mathbf{M}^{\omega_7}$ $\mathbf{M}^{\omega_8}$

Figure 3: *Instead of applying directly the recursive SVD (section 4.2)on N_ω frequencies, we apply the algorithm recursively to combine two frequencies at each level of the recursion tree (this example is given for the case $N_\omega = 8$).*

Overall complexity estimate for work and storage.

- Step A: The cost of reducing the number of incident fields is the cost for the construction of the low rank SVDs for matrices of size $N_s \times N$ for each frequency ω. The cost of a single frequency is $\mathcal{O}(R_s N_s N)$ and the overall cost is $\mathcal{O}(R_s N_s N_\omega N)$ (where $R_s \leq \min(N_s, N)$).
- Step B: The cost of the computation of the randomized SVD of $\mathbf{G}^\omega \in \mathbb{R}^{N_d \times N}$ for a fixed frequency is $\mathcal{O}(R_g^\omega N_d N)$ so that the total cost of this step is $\mathcal{O}(R_g^\omega N_\omega N_d N)$ (where $R_g^\omega \leq \min(N, N_d)$).

- Step C: The cost of combining the sources is reduced to the cost of the computation of the randomized SVD of $\mathbf{B}^\omega \in \mathbb{R}^{R_s R_g^\omega \times N}$: $\mathcal{O}(R_\omega R_s R_g^\omega N)$ for each frequency (with $R_\omega \leq R_s R_g^\omega, N$). So that the total cost of this step is $\mathcal{O}(R_\omega R_s N_\omega R_g^\omega N)$.

- Step D: At a fixed level ℓ, we want to combine two matrices of size $2^{\ell-1} R_s R_g^\omega \times N$. The cost of this recursion is thus $\mathcal{O}(R_\ell R_{\ell-1}(R_{\ell-1} + N + 2^\ell R^\omega)_g R_s)$. R_ℓ denotes the rank of the approximate matrix at a fixed level ℓ. As a result, we know that $R_\ell \leq 2^{\ell-1} R^\omega R_s$ (size of the matrix at a fixed level ℓ). So the cost to combine two frequencies at a fixed level ℓ is $\mathcal{O}(R_\ell R_{\ell-1}(N + 2^\ell R^\omega R_s))$. At each level ℓ, we have to compute $N_\omega 2^{-\ell}$ of those recursive SVDs. The total cost at a given level is finally $\mathcal{O}(N_\omega R_\ell R_{\ell-1}(2^{-\ell} N + R_g^\omega R_s))$. The complete algorithm requires $\log_2(N_\omega)$ levels so that the total cost is

$$\sum_{\ell=1}^{\log_2(N_\omega)} N_\omega R_\ell R_{\ell-1}(2^{-\ell} N + R_g^\omega R_s)$$
$$= \log_2(N_\omega) N_\omega R^2 (R_g^\omega R_s + N \sum_{\ell=1}^{\log_2(N_\omega)} 2^{-\ell}), \quad (23)$$

  where we assume that $R_\ell \leq R$ for $1 \leq \ell \leq \log_2(N_\omega)$. Finally the cost of step D is $\mathcal{O}(\log_2(N_\omega) N_\omega R^2 (R_g^\omega R_s + N))$.

The overall complexity of FaIMS is

$$\boxed{\mathcal{O}(R_s N_s N_\omega N + R_g^\omega N_\omega N_d N + R_\omega R_s N_\omega R_g^\omega N + \log_2(N_\omega) N_\omega R^2 N).}$$

At each of the four steps of FaIMS, we perform a low rank approximation of the system matrix. Because this matrix is defined by the values of a function on a given discretization, its rank is constant even though we use a coarser discretization. As a result for large enough numbers of sources, detectors, excitation frequencies and/or discretization of H, the values of R_s, R_g^ω, R_ω and R are constant. As a result the final complexity estimate becomes

$$\mathcal{O}(N_s N_\omega N + N_\omega N_d N).$$

As we have mentioned, if a fast summation method (or a fast forward solver) is available, this complexity reduces to

$$\mathcal{O}(N_\omega (N_s + N + N_d)).$$

Storage. For the step A the storage of the N_ω singular vectors is order $N_\omega R_s(N + N_s)$. For the step B, the storage of the N_ω singular vectors is order $N_\omega R_g^\omega (N + N_d)$ and order $N_\omega R_\omega (R_s R_g^\omega + N)$ for the step C. Finally the cost of the step D in terms of memory is $\mathcal{O}(\log_2(N_\omega) R_g^\omega R_s R + N_\omega N R)$. The storage of the singular vector of the SVD of the complete system matrix is $\mathcal{O}(N_\omega R_g^\omega R_s R + RN)$.

6 Numerical experiments

We present several configurations to illustrate the main advantages of the algorithm: the capability of placing detectors and sources on arbitrary geometries and its overall scalability. We verify the accuracy of the algorithm using two scatterer geometries: a cross-like planar geometry of point scatterers and a biplane-like

3D geometry of point scatterers. We generate the scattered field using single forward problem approximations (Born Approximation). In each example, the wave velocity of the background medium is set to one. We are focused on demonstrating the scalability aspects of our algorithm, so we do not consider noise or regularization. The algorithm has been implemented in Matlab and our experiments took place on a AMD Opteron workstation. We have shown that the accuracy of the recursive SVD depends on the accuracy of the SVDs in input. However, because of the ill-posedness of the problem and to obtain a fast algorithm, we must truncate the smallest singular values. The tolerance ε in the randomized SVD algorithm has been set to 10^{-9} in all of our experiments. The truncation parameter (regularization) we used to invert the approximate factorization of $\mathbf{M}$, was $10^{-8}\sigma_1$.

6.1 Description of the test problems

Definition of the main parameters used in our tests.

- The N_ω *incoming field frequencies* are equispaced in the $[\omega_{\min}, \omega_{\max}]$ interval.
- *Length scales* are measured in terms of the smallest wavelength $\lambda := 2\pi/\omega_{\max}$. (The higher the frequency, the lower the compression of the operators and thus, the higher the computational cost of the inversion.)
- In all of our experiments, we have chosen a cubic domain for the scatterer discretization domain $H := [0, L]^3$ and have selected $L = 12\lambda$. We have used a regular grid to discretize H. [6]

Specification of the target medium perturbations and parameters for the different numerical experiments.

- **Cross-like geometry.** We consider two test problems based on a simple cross-like geometry located at the $z = 6\lambda$ plane.
  - For the Cross A (Fig. 4A), the detectors are regularly spaced on the plane $z = L$ and the sources on the plane $z = 0$.
  - For the Cross B, the detectors and sources are located on arbitrary geometries (Fig. 4B).

  We verify the accuracy of our SVD reconstruction on the Cross A scatterer model with size λ. We use four sources with multiple excitation frequencies to generate the incident fields and measure the scattered fields at $[21]^2$ detectors. The computational domain is discretized with a linear grid with size $[21]^2$. Also on these two test problems, we verify the scalability of the algorithm with increasing N, N_d, N_s and N_ω and demonstrate the effectiveness of a low rank approximation. See Table 3 for a summary of the problem and scatterer model sizes used in this set of experiments.
- **Biplane geometry.** We consider a test problem with a more complex geometry. The sources and detectors are located on a sphere (Fig. 4D). We consider two scatterer model sizes: λ and 5λ. We use 162 detectors, 12 sources, 8 frequencies and we set $N = 51^3$.

[6]As mentioned in the introduction, more sophisticated adaptive schemes can be employed, but here for simplicity we only consider regular grid discretizations.

N	$\omega_{\min}$	$\omega_{\max}$	N_ω	N_s	N_d	scatterer size (λ)
11^2	50	100	8	16	10^2	0.01 and 1
21^2	50	100	16	64	20^2	0.01 and 1
41^2	50	100	32	256	40^2	0.01 and 1
81^2	50	100	64	1024	80^2	0.01 and 1

Table 3: *Cross-like geometry: Summary of the parameters used to verify the scalability of FaIMS. In the following, we will refer to one of those four tests by the mesh size, i.e.,* $N = 11^2$, $N = 21^2$, $N = 41^2$ *or* $N = 81^2$.

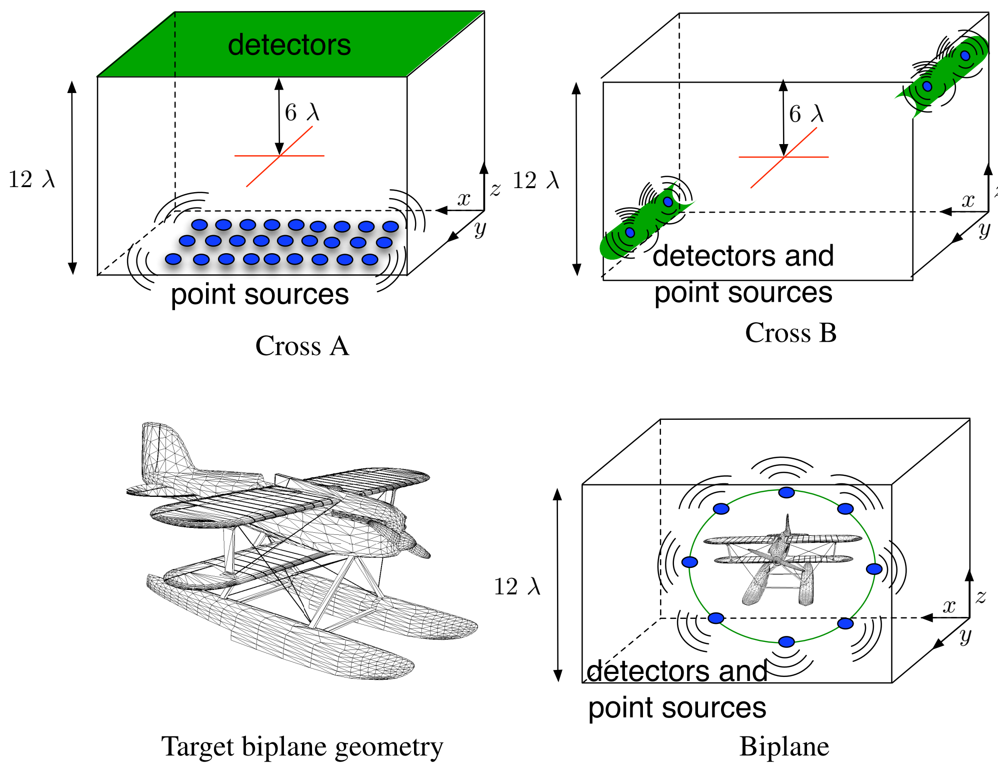


Figure 4: *Definition of the three test problems we have used to test FaIMS. We use two geometries of point scatterers : a cross (*A *and* B*) and a biplane. For the cross* A*, we generate the incident field by sources and detectors regularly spaced on the planes* $z = 0$ *and* $z = 12\lambda$ *respectively. For the Cross* B *test problem, the detectors and point sources are located on two arbitrary geometries. For the biplane-like geometry, the sources and detectors are located on a sphere. The triangulation of the biplane geometry is used for visualization only. To generate the data, we compute the scattered field due to point scatterers located at the vertices of the mesh.*

6.2 Results

Cross A. We first verify the accuracy of our SVD reconstruction. In Table 4, we report the relative error on the approximation normalized by the value of the maximum singular value. We also compute the relative error between FaIMS and the `SVD` MATLAB function on the singular values (normalized by the value of the maximum singular value). Finally, we report the relative error between FaIMS and the `SVD` Matlab function on the approximation of the inverse.

	$N_\omega = 4$	$N_\omega = 8$	$N_\omega = 16$	$N_\omega = 32$
$\|\mathbf{M} - \mathbf{U\Sigma V}^*\|/\sigma_{max}$	3.07 10^{-6}	2.34 10^{-6}	2.10 10^{-6}	2.01 10^{-6}
$\|\mathbf{\Sigma} - \mathbf{\Sigma}_{matlab}\|/\sigma_{max}$	1.00 10^{-7}	3.12 10^{-7}	1.73 10^{-7}	1.90 10^{-7}
$\|\boldsymbol{\eta} - \boldsymbol{\eta}_{matlab}\|/\boldsymbol{\eta}_{matlab}$	1.72 10^{-2}	2.29 10^{-2}	1.19 10^{-2}	2.95 10^{-2}

Table 4: *We report the relative error on the approximation normalized by the value of the maximum singular value. We also compute the relative error between FaIMS and the* SVD *Matlab function on the singular values (normalized by the value of the maximum singular value). Finally, we report the relative error between FaIMS and the* SVD *MATLAB function on the approximation of the inverse. The scatterer model is the Cross* A *with size λ. We use four sources with multiple excitation frequencies to generate the incident fields and measure the scattered fields at* $[21]^2$ *detectors. The computational domain is discretized using a Cartesian grid of size* $[21]^2$.

To verify the efficiency and accuracy of our algorithm, we solve the inverse problem both using our FaIMS algorithm and the LSQR MATLAB function. The termination tolerance in the LSQR algorithm is set to 10^{-5}. On Fig. 5, we report $\boldsymbol{\eta}$ the results of the inversion for various mesh sizes and for the scatterer model of the cross geometry with size λ. The left column plots represent the results of FaIMS and the right column plots the results with the LSQR. The mesh sizes handled using the LSQR function are limited due to memory constraints.[7] Both methods lead to a good accuracy of the reconstruction. In both cases, the larger the problem is, the better the reconstruction. The accuracy of FaIMS is slightly better. The reason is that we terminated LSQR early. .

In Figure 6, we report the CPU time for the four main steps of FaIMS: the reduction of the number of incident fields, the computation of the SVDs of the Green's functions, the combining of the SVDs for all sources for a fixed frequency and the combining of the SVDs for all frequencies, for the two scatterer model sizes (0.01λ and λ). We normalize the CPU time by the total CPU time required to generate the incident field (which is linear in $N_d,\ N_\omega, N_s$ and N). We note that the smaller the cross size is, the smaller the CPU time. This is due to the effectiveness of the low-rank approximation. For the scatterer model with size λ, the major CPU cost is the combining of the SVDs for all sources whereas, as expected, this step takes a small portion of time for the scatterer model with size 0.01λ. Again, as we increase N the normalized CPU time of each step reduces. This result is in agreement with our complexity estimate.

In Table 5, we report the normalized total CPU time of the inversion both with FaIMS and with the LSQR MATLAB function. Again the scatterer model is the Cross A with size 0.01λ and λ. For the LSQR solver, the total inversion time is independent of the scatterer model size whereas FaIMS benefits from low rank approximations at the low frequency regime. FaIMS is clearly faster than the LSQR MATLAB function. In Figure 7, we illustrate the level of compression according to the scatterer model and problem sizes. For each major step of the algorithm, we report the compression rates against the frequency or the level in the tree for the combining of the SVDs for all frequencies. The ranks are normalized by the full rank. Because the inverse problem is ill-posed, the singular values decay very fast. We use a truncated SVD to regularize the formulation.

[7]We could also use a simple matrix free approximation. In that case, the limitation is the maximum number of iterations allowed.

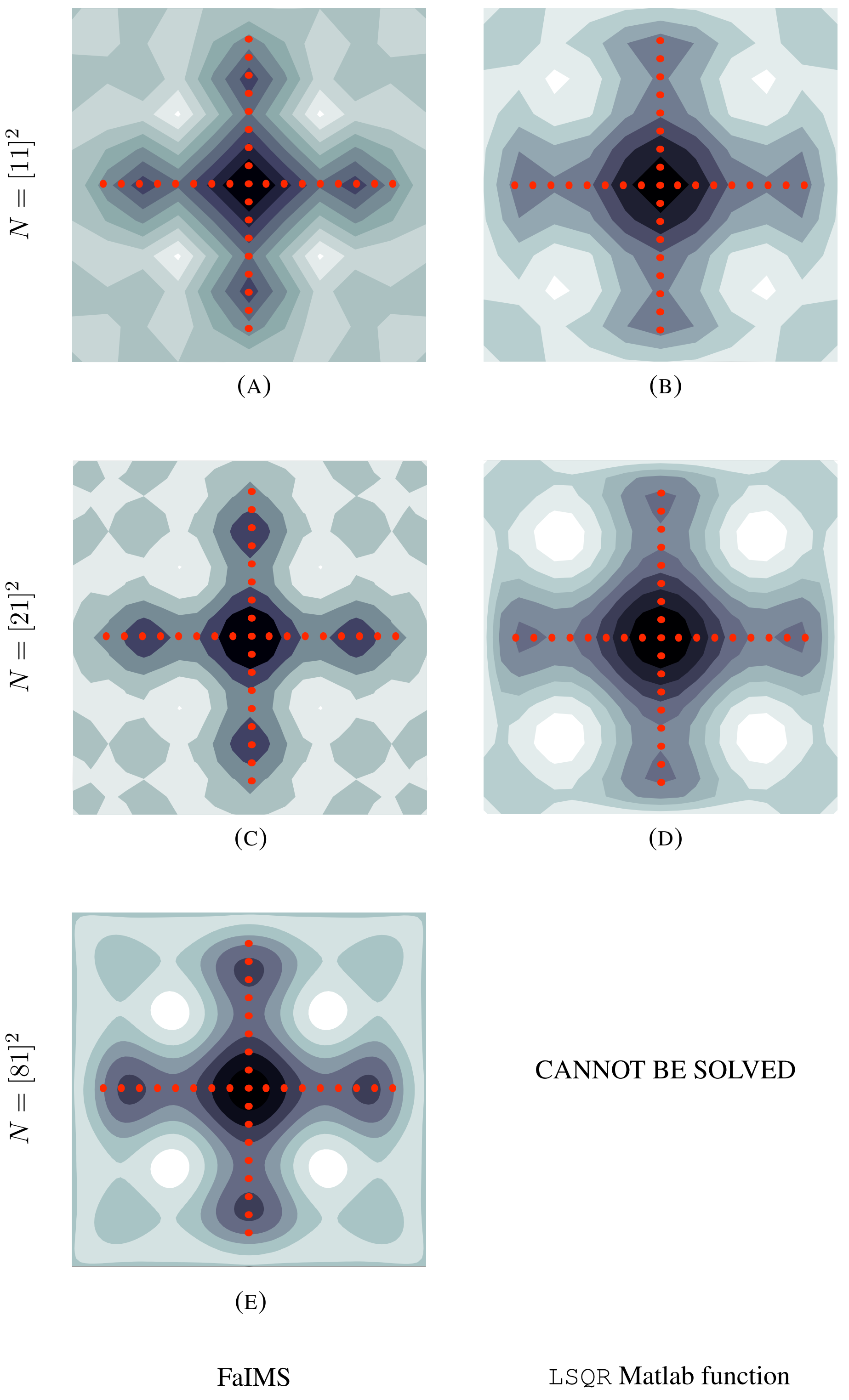
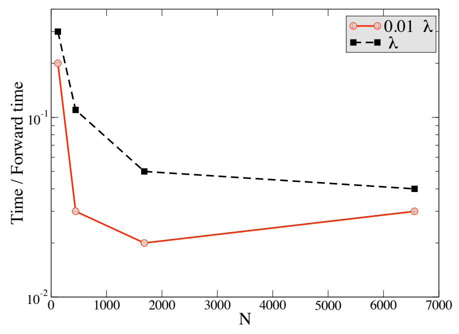
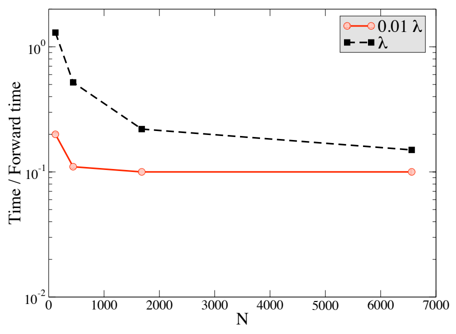


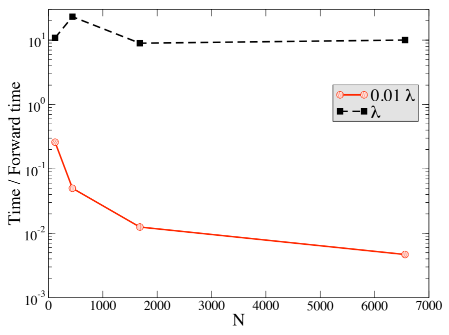
Figure 5: *Cross* A *: We report* $\boldsymbol{\eta}$*, the result of the inversion, at the* $z = 6\ \lambda$ *plane for the scatterer model cross* A *with size* λ*. The left column represents the results using FaIMS (*A,C,E*) and the right column the results using the* LSQR *MATLAB function with tolerance set to* 10^{-5} *(*B,D*). The red points represent the true point scatterer locations. The two methods produce very similar results.*



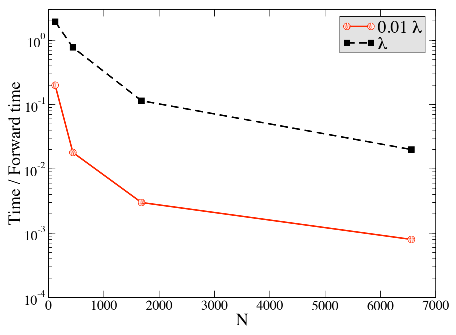
A: Reducing the number of sources



B: SVDs of the Green's functions



C: Combining the SVDs for all sources

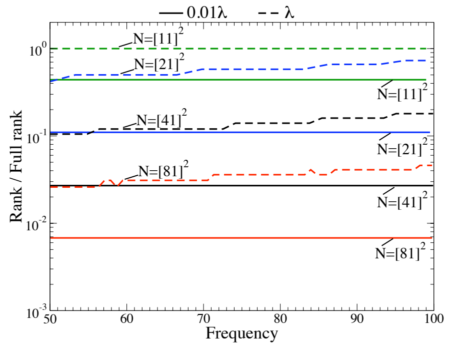


D: Combining the SVDs for all frequencies

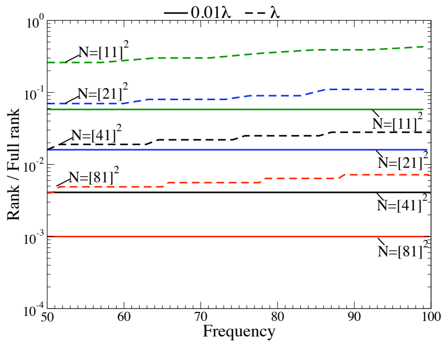
Figure 6: *We report the normalized (by the time to "solve" (evaluate in our case) the forward problem) CPU time for each of the main steps of FaIMS against the mesh size. The scatterer model is the Cross* A *with size* 0.01λ *(plain lines) and* λ *(dashed lines). The smaller is the cross size, the smaller is the normalized CPU time for each step. This is due to the possibility of low rank approximations at the low frequency regime. For the scatterer model with size* λ*, the major CPU consuming step is the combining of the SVDs for all sources whereas, as expected, this step consumes only a small portion of time for the scatterer model with size* 0.01λ*.*

		$N = [11]^2$	$N = [21]^2$	$N = [41]^2$	$N = [81]^2$
0.01λ	**FaIMS**	1.03	0.23	0.14	0.13
	`LSQR`	11.6 (1)	413.6 (1)	X	X
λ	**FaIMS**	15.0	24.7	9.3	10.2
	`LSQR`	13.1 (14)	407.7 (11)	X	X

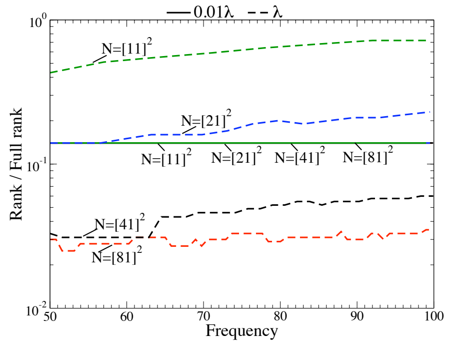
Table 5: *We report the normalized (by the time to solve the forward problem) CPU time against the mesh size using FaIMS and using the* `LSQR` *MATLAB function. For the* `LSQR` *we also report the number of iterations (number in parentheses).*



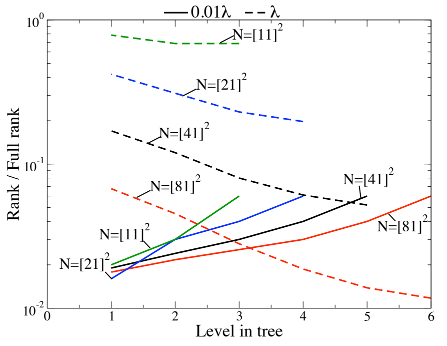
A: Reducing the number of sources



B: SVDs of the Green's functions



C: Combining the SVDs for all sources



C: Combining the SVDs for all frequencies

Figure 7: *Cross* A *: For each major step of the algorithm, we report the compression rates against the frequency or the level in the tree. The ranks are normalized by the full rank. The scatterer model is the Cross* A *with size* 0.01λ *(plain lines) and* λ *(dashed lines). We report the results for the four mesh sizes (red:* $N = 81^2$*, black:* $N = 41^2$*, blue:* $N = 21^2$ *and green:* $N = 11^2$*). Because the useful information is limited, the rank is not dependent of the mesh size. As a result, the larger the mesh is, the larger the compression. Moreover, the larger the scatterer model is, the smaller the compression. Finally, we remark that at each step, the algorithm keeps compressing the information.*

Cross B. For this second example, the sources and detectors are located on arbitrary geometries. We report the results of the inversion $\boldsymbol{\eta}$ for the case of $N = 81^2$ on Fig. 8 (left: scatterer model with size 0.01λ;

right: scatterer model with size λ). This example illustrates the ill-posedness of the problem. For the lowest frequency, only a small number of singular values are selected such that the algorithm is very fast. Of course, for this case the reconstruction is not so accurate. We get better reconstructions with increasing frequency. In Figure 9, we report the CPU time for the four main steps of FaIMS: the reduction of the number of

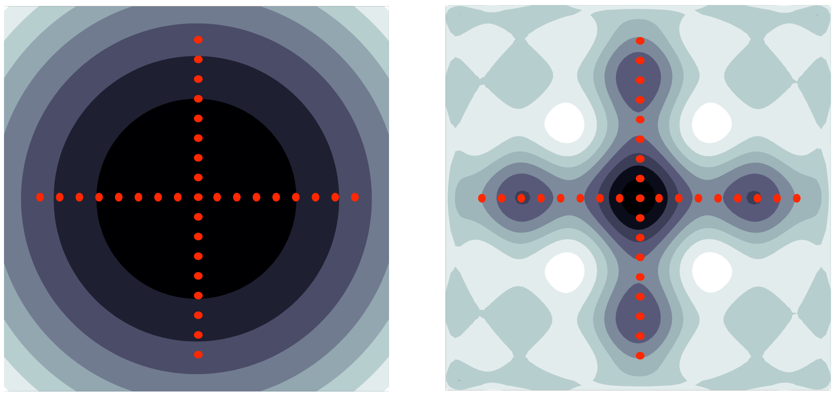
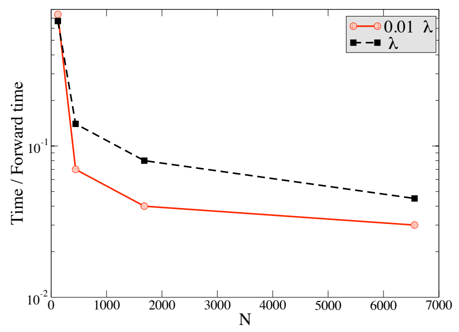


Figure 8: *We report $\boldsymbol{\eta}$ at the $z = 6\,\lambda$ plane. The scatterer model is the Cross* B *with size* 0.01λ *(left) and* λ *(right). The* 6561 *detectors are located on the geometry presented on Fig. 4*B. *We generate the data using incident fields generated by* 1024 *sources on the geometry presented on Fig. 4*B *and* 64 *frequencies.*

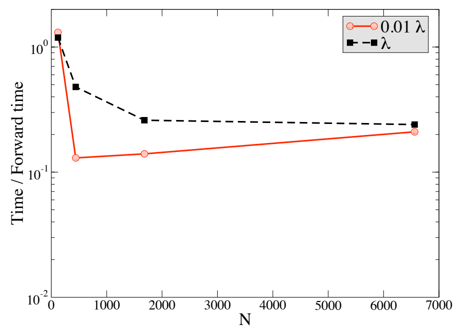
incident fields, the computation of the SVDs of the Green's functions, the combining of the SVDs for all sources for a fixed frequency and the combining of the SVDs for all frequencies, for the two scatterer model sizes (0.01λ and λ). We normalize the CPU time by the total CPU time to generate the incident field (which is linear in $N_d,\ N_\omega, N_s$ and N). The smaller the cross size is, the smaller the CPU time since low-rank approximations are effective. For the scatterer model with size λ, the major CPU consuming step is the combining of the SVDs for all sources whereas, as expected, this step consumes only a small portion of time for the scatterer model with size 0.01λ. In Table 6, we report the normalized total CPU time of the inversion both with FaIMS and with the `LSQR` function. Again the scatterer model is the Cross B with size 0.01λ and λ. For the `LSQR` solver, the total inversion time is independent of the scatterer model size whereas FaIMS benefits from low rank approximations at the low frequency regime. FaIMS is clearly faster than the `LSQR` Matlab function and than the forward solver at low frequencies.

In Figure 10, we report the level of compression according to the scatterer model and problem sizes.

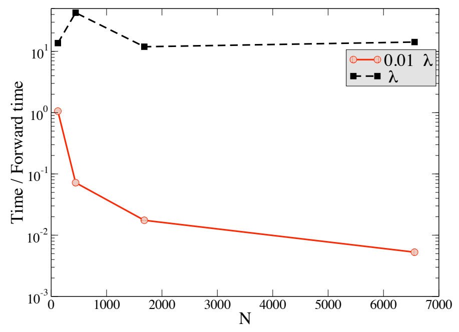
Biplane. This last example demonstrates the ability of the algorithm to recover complex geometries. The scatterer model is a biplane-like geometry with size λ or 5λ. On Figures 11 (size λ) and 12 (size 5λ), we report the results of the inversion $\boldsymbol{\eta}$ under a tabular form. Each row corresponds to a particular point of view (plane Oxy, Oxz or Oyz). The left column corresponds to a 3-D view of the isosurfaces ($\eta = 0.25 * \eta_{max}$) and the right column corresponds to a 2-D slice at the median plane. For the scatterer model with size λ, the inversion is as accurate. We can already distinguish the main components of the biplane. For example, on the Oxz view, we can separate the two floats of the plane. However, we cannot see the two wings or say that the shape is a biplane. Again, this is due to the ill-posedness of the problem at low frequencies. On the other hand, for the scatterer model with size 5λ, the system matrix is nearly full rank. As a result, the inversion is more accurate. We can clearly see the two wings, the two floats and the fin of the biplane. Again, as expected, the higher the frequency is, the more accurate the reconstruction.



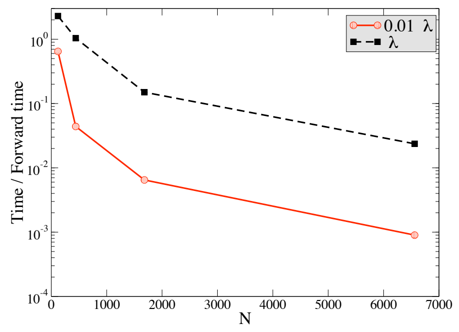
A: Reducing the number of sources



B: SVDs of the Green's functions

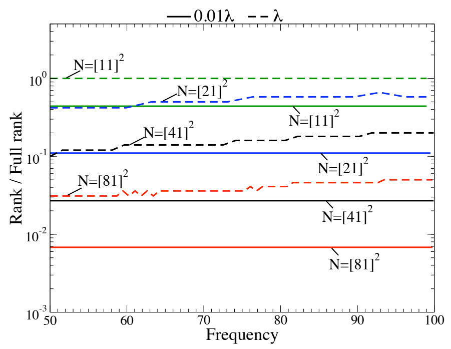


C: Combining the SVDs for all sources

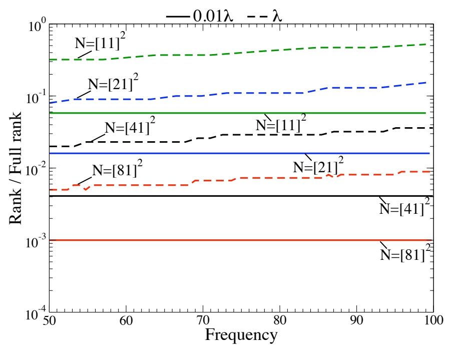


D: Combining the SVDs for all frequencies

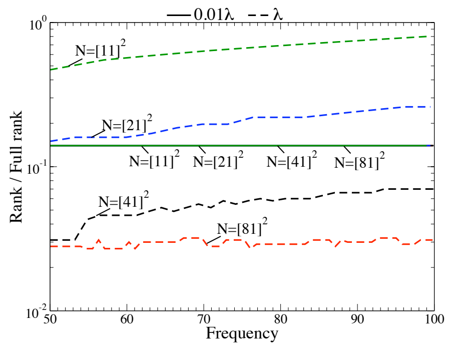
Figure 9: *We report the normalized (by the time to solve the given forward problem) CPU time for each main step of our inversion algorithm against the mesh size. The scatterer model is the Cross* B *with size* 0.01λ *(plain lines) and* λ *(dashed lines).*



Step A: Reducing the number of sources



Step B: SVDs of the Green's functions



Step C: Combining the SVDs for all sources

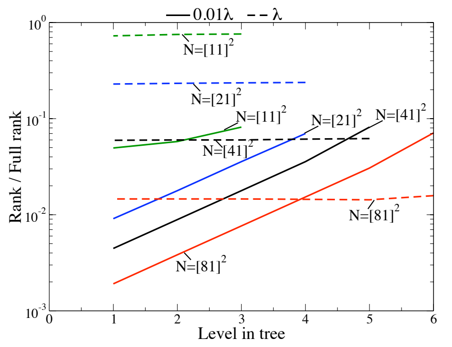


Figure 10: *Cross B : For each major step of the algorithm, we report the compression rates against the frequency. The ranks are normalized by the full rank. The scatterer model is the Cross B with size 0.01λ (plain lines) and λ (dashed lines). We report the results for the four mesh sizes (red: $N = 81^2$, black: $N = 41^2$, blue: $N = 21^2$ and green: $N = 11^2$).*

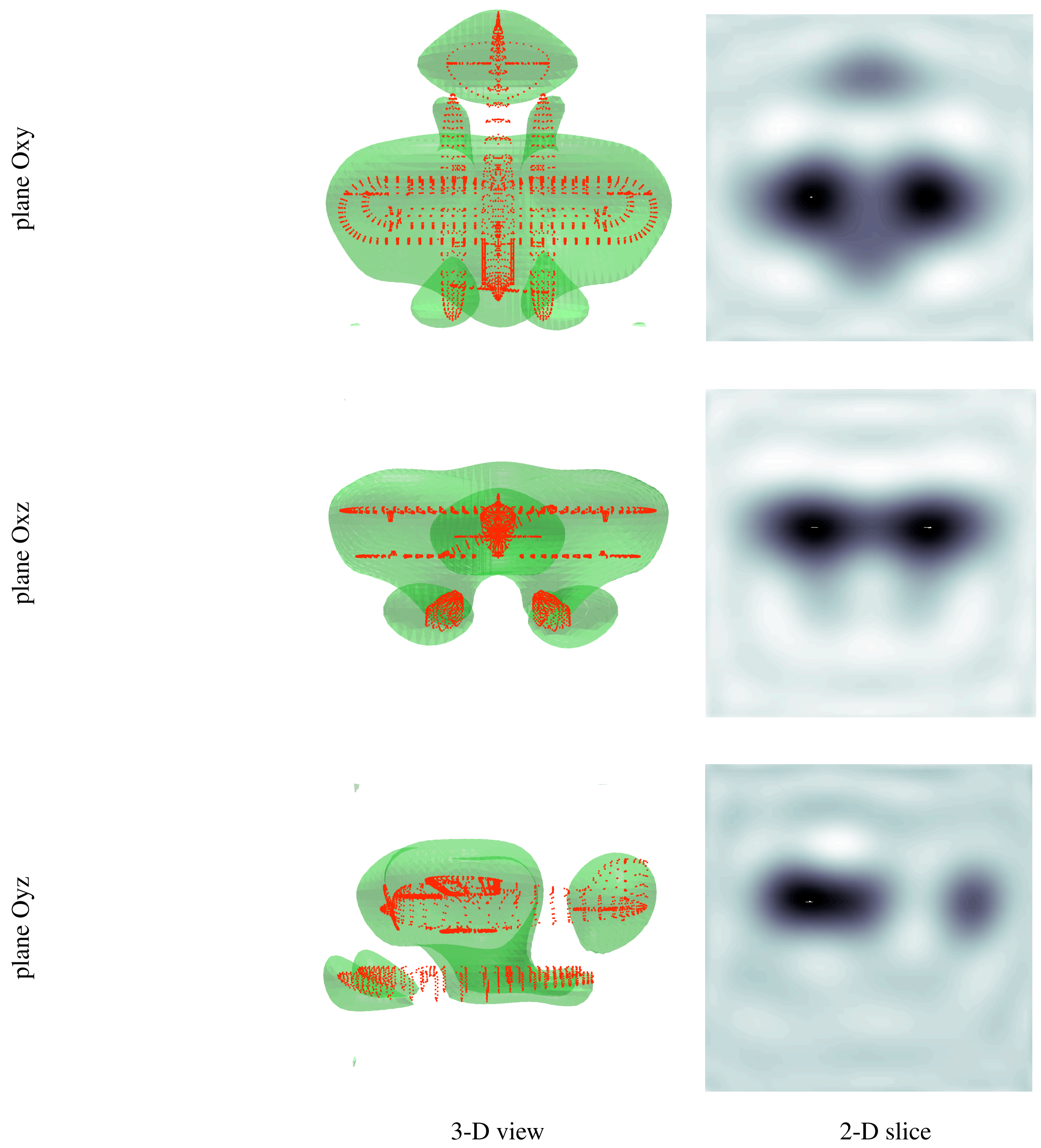


Figure 11: *Results of the inversion for the scatterer model of the biplane with size* λ. *Each row corresponds to a particular point of view (plane Oxy, Oxz or Oyz). The left column corresponds to a 3-D view of the isosurfaces* ($\eta = 0.25 * \eta_{max}$) *and the right column corresponds to a 2-D slice, respectively from top to bottom at the planes* $z = 0$, $y = 0$ *and* $x = 0$.

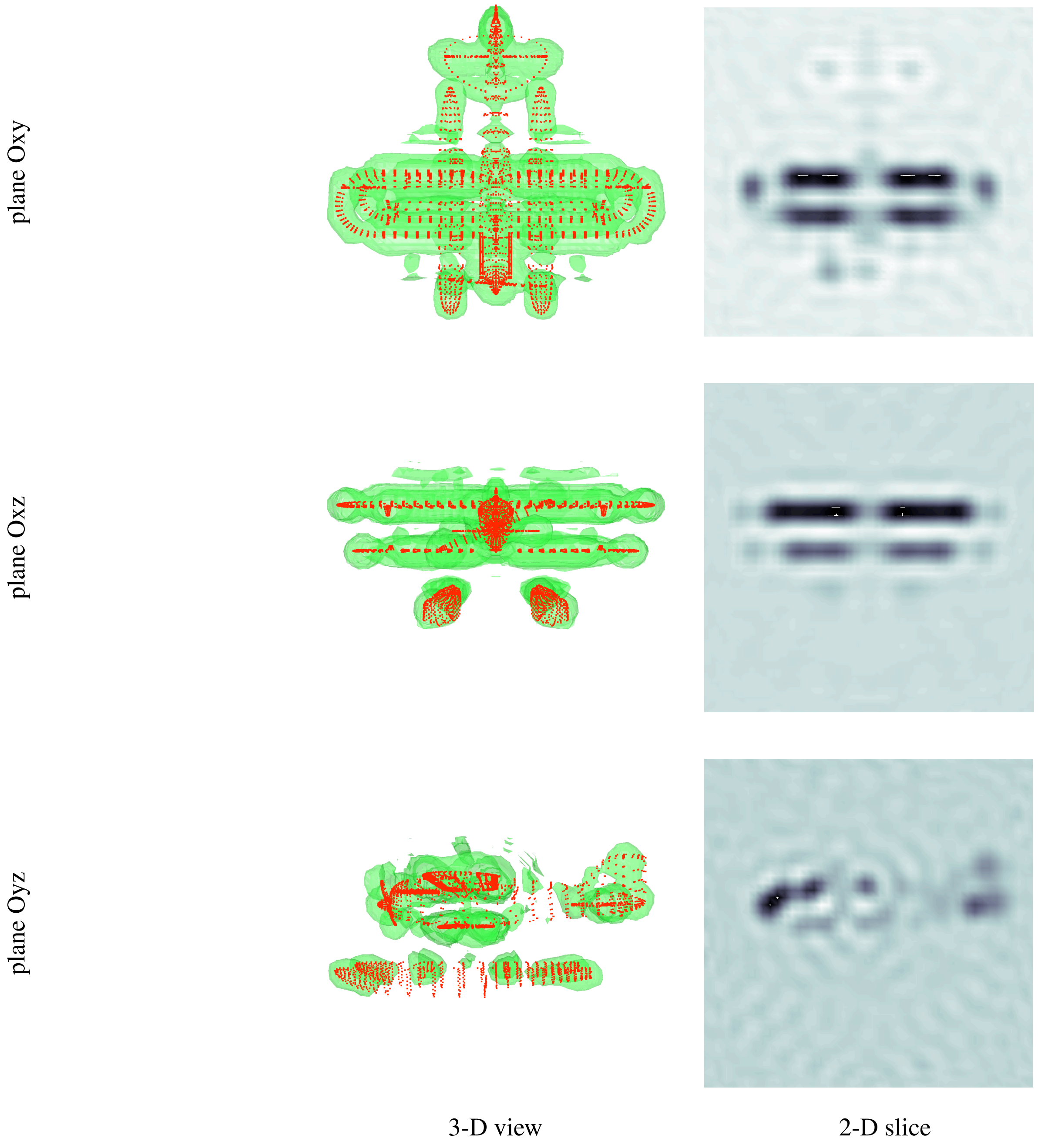


Figure 12: *Results of the inversion for the scatterer model of the biplane with size 5λ. Each row corresponds to a particular point of view (plane Oxy, Oxz or Oyz). The left column corresponds to a 3-D view of the isosurfaces ($\eta = 0.25 * \eta_{max}$) and the right column corresponds to a 2-D slice, respectively from top to bottom at the planes $z = 0$, $y = 0$ and $x = 0$.*

		$N = [11]^2$	$N = [21]^2$	$N = [41]^2$	$N = [81]^2$
0.01λ	FaIMS	4.14	0.35	0.21	0.25
	`LSQR`	6.4 (1)	295.6 (2)	X	X
λ	FaIMS	18.2	44.5	12.4	14.5
	`LSQR`	9.5 (49)	309.8 (71)	X	X

Table 6: *We report the normalized (by the time to solve the forward problem) CPU time against the mesh size using FaIMS and using the* `LSQR` *Matlab function. For the* `LSQR` *Matlab function we also report the number of iterations (number in parentheses).*

7 Conclusions

In this paper, we have presented FaIMS, a method for the inverse medium problem for the time-harmonic scalar wave equation. FaIMS uses a randomized SVD algorithm to compute SVDs of small submatrices and then applies a recursive SVD algorithm to reconstruct the overall factorization. Its complexity estimate is orders-of-magnitude smaller than the standard SVD factorization. The method is matrix-free, it only requires matrix-vector multiplication for the forward and adjoint problems. In our experiments, FaIMS outperformed the LSQR Krylov iterative method. We showed that the factorization error in the singular values is bounded by the smallest largest singular value that we truncate in the rank approximation. The numerical efficiency and accuracy of the method is demonstrated in several numerical experiments in the low frequency (0-10 wavelengths) regime for the case of point scatterers. FaIMS can handle detectors and sources located on arbitrary geometries. In future work, we intend using our approximate SVD factorization as a preconditioner with a Newton-Krylov-Multigrid iterative method for full nonlinear inversion method (for example, for problems in which G is not analytically available [5]). Also, we ongoing work includes adaptive algorithms and parallelization of the method. For higher frequencies, ideas discussed in [12] can be explored to construct directional low-rank approximations.

References

[1] S. S. Adavani and G. Biros, *Fast algorithms for source identification problems with parabolic PDE constraints*, SIAM Journal on Imaging Sciences, (2010). in press.

[2] V. Akcelik, G. Biros, and O. Ghattas, *Parallel multiscale Gauss-Newton-Krylov methods for inverse wave propagation*, in Proceedings of the IEEE/ACM SC2002 Conference, The SCxy Conference series, Baltimore, Maryland, November 2002, ACM/IEEE.

[3] U. Ascher and E. Haber, *A multigrid method for distributed parameter estimation problems*, ETNA, 15 (2003).

[4] W. Bangerth, *A framework for the adaptive finite element solution of large-scale inverse problems*, SIAM Journal on Scientific Computing, 30 (2008), pp. 2965–2989.

[5] G. Biros and G. Doğan, *A multilevel algorithm for inverse problems with elliptic PDE constraints*, Inverse Problems, 24 (2008), pp. 1–19.

[6] A. BORZÍ, *High-order discretization and multigrid solution of elliptic nonlinear constrained optimal control problems*, Journal Of Computational And Applied Mathematics, 200 (2007), pp. 67–85.

[7] E. CANDÈS AND J. ROMBERG, *Sparsity and incoherence in compressive sampling*, Inverse Problems, 23 (2007), pp. 969–985.

[8] H. CHENG, L. GREENGARD, AND V. ROKHLIN, *A fast adaptive multipole algorithm in three dimensions*, Journal of Computational Physics, 155 (1999), pp. 468–498.

[9] J. CHUNG, J. NAGY, AND D. O'LEARY, *A Weighted-GCV Method for Lanczos-Hybrid Regularization*, Electronic Transactions on Numerical Analysis, 28 (2008), pp. 149–167.

[10] D. COLTON AND R. KRESS, *Inverse Acoustic and Electromagnetic Scattering Theory, 2nd Edition*, Applied Mathematical Sciences, Springer, 1998.

[11] R. COURANT AND D. HILBERT, *Methods of Mathematical Physics, vol. II*, Interscience Publishers, 1961.

[12] B. ENGQUIST AND L. YING, *Fast directional multilevel algorithms for oscillatory kernels*, SIAM Journal on Scientific Computing, 29 (2008), pp. 1710–1737.

[13] G. H. GOLUB AND C. H. V. LOAN, *Matrix Computations*, Johns Hopkins, third ed., 1996.

[14] E. HABER AND U. ASCHER, *Preconditioned all-at-one methods for large, sparse parameter estimation problems*, Inverse Problems, 17 (2001), pp. 1847–1864.

[15] E. HABER, S. HELDMANN, AND U. ASCHER, *Adaptive finite volume method for distributed non-smooth parameter identification*, Inverse Problems, 23 (2007), pp. 1659–1676.

[16] N. HALKO, P. MARTINSSON, AND J. TROPP, *Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions*, arXiv, 909 (2009).

[17] F. HERRMANN, Y. ERLANGGA, AND T. LIN, *Compressive simultaneous full-waveform simulation*, Geophysics, 74 (2009), p. A35.

[18] T. HOHAGE, *On the numerical solution of a three-dimensional inverse medium scattering problem*, Inverse Problems, 17 (2001), p. 1743.

[19] J. KREBS, J. ANDERSON, D. HINKLEY, R. NEELAMANI, S. LEE, A. BAUMSTEIN, AND M. LACASSE, *Fast full-wavefield seismic inversion using encoded sources*, (2009).

[20] E. LIBERTY, F. WOOLFE, P. MARTINSSON, V. ROKHLIN, AND M. TYGERT, *Randomized algorithms for the low-rank approximation of matrices*, Proceedings of the National Academy of Sciences, 104 (2007), p. 20167.

[21] V. A. MARKEL, V. MITAL, AND J. C. SCHOTLAND, *Inverse problem in optical diffusion tomography. iii. inversion formulas and singular-value decomposition*, Journal of the Optical Society of America A, 20 (2003), pp. 890–902.

[22] J. L. MORALES AND J. NOCEDAL, *Automatic preconditioning by limited memory quasi-Newton updating*, SIAM Journal on Optimization, 10 (2000), pp. 1079–1096.

[23] R. NEELAMANI, C. KROHN, J. KREBS, M. DEFFENBAUGH, J. ANDERSON, AND J. ROMBERG, *Efficient seismic forward modeling using simultaneous random sources and sparsity*, submitted to Geophysics, (2009).

[24] C. PAIGE AND M. SAUNDERS, *Algorithm 583: LSQR: Sparse linear equations and least squares problems*, ACM Transactions on Mathematical Software (TOMS), 8 (1982), pp. 195–209.

[25] J. SCHOTLAND AND V. MARKEL, *Inverse scattering with diffusing waves*, J. Opt. Soc. Am. A., 18 (2001), pp. 2767–2777.

[26] G. STEWART, *On the perturbation of pseudo-inverses, projections and linear least squares problems*, SIAM review, 19 (1977), pp. 634–662.

[27] ——, *Perturbation theory for the singular value decomposition*, Computer Science Technical Report Series; Vol. CS-TR-2539, (1990), p. 13.

[28] L. YING, G. BIROS, AND D. ZORIN, *A kernel-independent adaptive fast multipole method in two and three dimensions*, Journal of Computational Physics, 196 (2004), pp. 591–626.