Scientific Data Interpolation with Low Dimensional Manifold Model[☆]

Wei Zhu[a,1,*], Bao Wang[a], Richard Barnard[b], Cory D. Hauck[b,c,2], Frank Jenko[d], Stanley Osher[a,3]

[a] *Department of Mathematics, University of California Los Angeles, Los Angeles, CA 90095, USA*
[b] *Computational and Applied Mathematics Group, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA*
[c] *Department of Mathematics, University of Tennessee, Knoxville, TN 37996-1320, USA*
[d] *Department of Physics and Astronomy, University of California, Los Angeles, CA 90095, USA*

Abstract

We propose to apply a low dimensional manifold model to scientific data interpolation from regular and irregular samplings with a significant amount of missing information. The low dimensionality of the patch manifold for general scientific data sets has been used as a regularizer in a variational formulation. The problem is solved via alternating minimization with respect to the manifold and the data set, and the Laplace-Beltrami operator in the Euler-Lagrange equation is discretized using the weighted graph Laplacian. Various scientific data sets from different fields of study are used to illustrate the performance of the proposed algorithm on data compression and interpolation from both regular and irregular samplings.

Keywords: Low dimensional manifold model (LDMM), scientific data interpolation, data compression, regular and irregular sampling, weighted

[*]Corresponding author

Email addresses: weizhu731@math.ucla.edu (Wei Zhu), wangbao@math.ucla.edu (Bao Wang), barnardrc@ornl.gov (Richard Barnard), hauckc@ornl.gov (Cory D. Hauck), jenko@physics.ucla.edu (Frank Jenko), sjo@math.ucla.edu (Stanley Osher)

[1]This work is supported by ONR Grant N00014- 14-1-0444.

[2]This material is based, in part, upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing and by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC for the U. S. Department of Energy under Contract No. De-AC05-00OR22725.

[3]This material is based, in part, upon work supported by the U.S. Department of Energy, Office of Science and by National Science Foundation, under Grant Numbers DOE-SC0013838 and DMS-1554564.

1. Introduction

Interpolation and reconstruction of scientific data sets from sparse sampling is of great interest to many researchers from various communities. In many situations, data are only partially sampled due to logistic, economic, or computational constraints: limited number of sensors in seismic data or hyperspectral data acquisition, low-dose radiographs in medical imaging, coarse-grid solutions of partial differential equations due to computational complexity, etc. Moreover, sometimes one may also intentionally sample partial information of the scientific data set as a straightforward data compression technique. As a result, it has become an important topic to reconstruct the original data set from regular or irregular samplings.

There are typically two ways to approach this problem. The first one is to use the underlying physics to infer the missing data [1, 2, 3, 4, 5]. The drawback is that such techniques are usually problem-specific and not generally applicable to similar problems in other fields of study. Signal and data processing techniques, on the other hand, usually do not require too much prior information of the governing physics. These models intend to fill in the missing information by the properties manifested by the sampled data themselves, while implicitly enforcing common structures from physical intuition in the regularization.

Many signal processing approaches to data interpolation have been studied in the context of image inpainting and seismic data interpolation. Popular interpolation models have been proposed through total variation [6, 7], wavelets [8, 9], and curvelets [10, 11, 12, 1]. After the introduction of the nonlocal mean by Buades et al. in [13], patch-based techniques exploiting similarity and redundancy of local patches have been extensively studied for inpainting and reconstruction [14, 15, 16]. This also leads to a wide variety of sparse-signal models which assume that patches can be sparsely represented by atoms in a prefixed or learned dictionary [8, 17]. Patch-based Bayesian models have also been proposed in image and data interpolation [18, 19]. However, as reported in [18], some of the algorithms can only be applied to the interpolation of randomly selected samples, and fail to achieve satisfactory results for uniform grid interpolation. Moreover, most of the methods perform poorly when a significant amount of information ($\geq 95\%$) is missing.

Recently, a low dimensional manifold model (LDMM) has been proposed for general image processing problems [20]. In particular, it achieved state-of-the-art results for image interpolation problems with a significant number of missing pixels. The main idea behind LDMM is that the patch manifold (to be explained in Section 2) of a real-world 2D image has a much lower intrinsic dimension than that of the ambient space. Based on this observation, the authors used the dimension of the patch manifold as a regularizer in the variational formulation, and the optimization problem is solved using alternating minimization with respect to the image and the manifold. The key step in the algorithm, which

involves solving a Laplace-Beltrami equation over an unstructured point cloud sampling the patch manifold, is solved via either the point integral method [21] or the weighted graph Laplacian [22].

In this work, we apply LDMM to the interpolation of 2D and 3D scientific data sets from either regular or irregular samplings, and demonstrate its superiority when compared to other methods. Moreover, we also compare the performance of LDMM as a sampling-based data compression technique to other standard compression methods. Unlike the other compression methods, sampling-based methods do not require access to the full data set. Although the results of sampling-based algorithms are generally inferior to standard compression methods, they have the advantage of easy implementation in the compression step, and they are also faster in the reconstruction step if only the reconstruction of a small portion of the data set is required. A useful by-product of this comparison is that the standard compression methods are implicitly compared against one another on a set of physically meaningful test cases that can be used for future benchmarks.

The rest of the paper is organized as follows. Section 2 reviews the low dimensional manifold model and justifies its application to scientific data interpolation through a dimension analysis. Section 3 outlines the detailed numerical implementaion of LDMM with weighted graph Laplacian which was missing in [22]. A comparison of the numerical results on various scientific data interpolation and compression is reported in Section 4. Finally, we draw our conclusion in Section 5.

2. Low Dimensional Manifold Model

Low dimensional manifold model (LDMM) is a recently proposed mathematical image processing technique which performs particularly well on natural image inpainting [20, 23]. The main observation is that the intrinsic dimension of the patch manifold of a natural image is much smaller than that of the ambient Euclidean space. Therefore it is intuitive to use the dimension of the patch manifold as a regularizer to recover the degraded image. We argue that the same property holds true for scientific data sets. Throughout the entire paper, we present our analysis and algorithm for 3D scientific data sets. The formulation for 2D and higher dimensional data sets follows in a natural way.

2.1. Patch Manifold and Dimension Analysis

Consider a 3D datacube $f \in \mathbb{R}^{m\times n\times r}$. For any voxel $\boldsymbol{x} \in \bar{\Omega} = \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\} \times \{1, 2, \ldots, r\}$ [4], the patch $\mathcal{P}f(\boldsymbol{x})$ is defined as a vector storing the data values in a 3D cube of size $s_1 \times s_2 \times s_3$, with $\boldsymbol{x}$ being the first voxel of the 3D cube in the lexicographic order, i.e. $\boldsymbol{x}$ is in one particular corner of the

[4] The notation Ω is reserved for the sampled subset of $\bar{\Omega}$.

cube[5]. The patch set $\mathcal{P}(f)$ of f is the collection of all patches:

$$\mathcal{P}(f) = \left\{\mathcal{P}f(\boldsymbol{x}) : \boldsymbol{x} \in \bar{\Omega}\right\} \subset \mathbb{R}^d, \quad d = s_1 \times s_2 \times s_3.$$

We assume that the patch set $\mathcal{P}(f)$, which is a point cloud in $\mathbb{R}^d$, samples an underlying structure $\mathcal{M}$, which is refered to as the patch manifold of f. Rigorously speaking, $\mathcal{M}$ is not a smooth manifold, but instead is a collection of manifolds, $(\mathcal{N}_l)_{l=1}^L$, with different dimensions corresponding to various patterns in the data set, $\mathcal{M} = \cup_{l=1}^L \mathcal{N}_l$. For any $\boldsymbol{p} \in \mathcal{M}$, we use the notation $\mathcal{M}(\boldsymbol{p})$ to denote the smooth manifold $\mathcal{N}_l$ to which $\boldsymbol{p}$ belongs, and $\dim(\mathcal{M}(\boldsymbol{p}))$ is the dimension of $\mathcal{M}(\boldsymbol{p})$.

An important assumption is that for scientific data sets, the intrinsic dimension of the patch manifold $\mathcal{M}$ is often much smaller than the dimension of the embedding space $\mathbb{R}^d$. For example, if f is locally smooth at $\boldsymbol{x}$ corresponding to smoothly variant region of the data set, then $\mathcal{P}f(\boldsymbol{x})$ can be approximated by a linear function via Taylor expansion:

$$\mathcal{P}f(\boldsymbol{x})(\boldsymbol{y}) \approx f(\boldsymbol{x}) + \nabla f(\boldsymbol{x}) \cdot (\boldsymbol{y} - \boldsymbol{x}).$$

Therefore, $\mathcal{M}$ can be approximated by a 4D manifold locally at $\mathcal{P}f(\boldsymbol{x})$. If f is a piecewise smooth function with a sharp interface corresponding to a shock wave, then the patches can be parameterized by the location and orientation of the shock, as well as the gradient and voxel value information in the two regions. This implies that $\mathcal{M}$ is locally close to an 11D manifold. If $f = a(\boldsymbol{x})\cos(\theta(\boldsymbol{x}))$ models oscillatory structures, then Taylor expansion with respect to a and θ implies that $\mathcal{M}$ can be locally approximated by a smooth manifold of dimension 8.

When dealing with 3D data sets of size $256 \times 256 \times 32$ in our numerical tests, we typically choose patches of size $6 \times 6 \times 4$. This implies that the dimension d of the ambient space is 144. The dimension analysis above justifies the claim that the patch manifold $\mathcal{M}$ is a low dimensional manifold.

2.2. Variational Formulation

Based on the discussion in the previous section, we use the dimension of the patch manifold $\mathcal{M}$ as a regularizer in the following variational formula:

$$\min_{\substack{f \in \mathbb{R}^{m\times n\times r},\\ \mathcal{M}\subset\mathbb{R}^d}} \quad \int_{\mathcal{M}} \dim(\mathcal{M}(\boldsymbol{p}))d\boldsymbol{p}, \qquad \text{subject to:} \quad b = \Phi_\Omega f, \quad \mathcal{P}(f) \subset \mathcal{M}, \quad (1)$$

where

$$\int_{\mathcal{M}} \dim(\mathcal{M}(\boldsymbol{p}))d\boldsymbol{p} = \sum_{l=1}^{L} \int_{\mathcal{N}_l} \dim(\mathcal{N}_l) d\mu_{\mathcal{N}_l}(\boldsymbol{p}) = \sum_{l=1}^{L} |\mathcal{N}_l| \dim(\mathcal{N}_l),$$

[5]One can also choose $\boldsymbol{x}$ to be the center of the cube, and the result will be similar. The reason is that the reconstruction is performed on patches instead of on voxels. This will be clear in Section 3.

$\mu_{\mathcal{N}_l}$ is the surface measure on $\mathcal{N}_l$, Φ_Ω is the sampling operator on the subset $\Omega \subset \bar{\Omega}$, and b is the partially observed data. It is worth mentioning that $\int_{\mathcal{M}} \dim(\mathcal{M}(\boldsymbol{p}))d\boldsymbol{p}$ can be thought of as the L^1 norm of the local dimension of the manifold $\mathcal{M}$. It has been shown in [20] that the dimension $\dim(\mathcal{N})$ of any smooth manifold $\mathcal{N}$ can be calculated by the following simple formula:

Theorem 1. *Let $\mathcal{N}$ be a smooth submanifold isometrically embedded in $\mathbb{R}^d$. For any $\boldsymbol{p} = (p_1, p_2, \cdots, p_d) \in \mathcal{N}$,*

$$\dim(\mathcal{N}) = \sum_{i=1}^{d} \left|\nabla_{\mathcal{N}} \alpha_i(\boldsymbol{p})\right|^2,$$

where $\alpha_i(\boldsymbol{p}) = p_i$ is the coordinate function, and $\nabla_{\mathcal{N}}$ is the gradient operator on the manifold $\mathcal{N}$. More specifically, $\nabla_{\mathcal{N}}\alpha_i = \sum_{s,t=1}^{k} g^{st}\partial_t \alpha_i \partial_s$, where k is the intrinsic dimension of $\mathcal{N}$, and g^{st} is the inverse of the metric tensor.

The interested reader can refer to [24] for manifold calculus and [20] for the proof. As a result of Theorem 1, (1) can be reformulated as:

$$\min_{\substack{f\in\mathbb{R}^{m\times n\times r},\\ \mathcal{M}\subset\mathbb{R}^d}} \quad \sum_{i=1}^{d} \|\nabla_{\mathcal{M}}\alpha_i\|^2_{L^2(\mathcal{M})}, \quad \text{subject to:} \quad b = \Phi_\Omega f, \quad \mathcal{P}(f) \subset \mathcal{M}, \tag{2}$$

where

$$\begin{aligned}\sum_{i=1}^{d} \|\nabla_{\mathcal{M}}\alpha_i\|^2_{L^2(\mathcal{M})} &= \sum_{l=1}^{L}\sum_{i=1}^{d} \|\nabla_{\mathcal{N}_l}\alpha_i\|^2_{L^2(\mathcal{N}_l)} = \sum_{l=1}^{L}\sum_{i=1}^{d} \int_{\mathcal{N}_l} \left|\nabla_{\mathcal{N}_l}\alpha_i(\boldsymbol{p})\right|^2 d\mu_{\mathcal{N}_l}(\boldsymbol{p}) \\ &= \sum_{l=1}^{L} |\mathcal{N}_l| \dim(\mathcal{N}_l) = \int_{\mathcal{M}} \dim(\mathcal{M}(\boldsymbol{p}))d\boldsymbol{p} \end{aligned} \tag{3}$$

The variational problem (2) can be solved by alternating minimization with respect to $\mathcal{M}$ and f. More specifically, given $\mathcal{M}^k$ and f^k at step k satisfying $\mathcal{P}(f^k) \subset \mathcal{M}^k$:

- With fixed $\mathcal{M}^k$, update the data f^{k+1} by solving:

$$\begin{aligned} \min_{f\in\mathbb{R}^{m\times n\times r}} \quad & \sum_{i=1}^{d} \|\nabla_{\mathcal{M}^k}\alpha_i^f\|^2_{L^2(\mathcal{M}^k)}, \\ \text{subject to:} \quad & \alpha_i^f(\mathcal{P}(f^k)(\boldsymbol{x})) = \mathcal{P}_i f(\boldsymbol{x}), \quad \boldsymbol{x} \in \overline{\Omega}, \quad i = 1, \cdots, d, \\ & f(\boldsymbol{x}) = b(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \end{aligned} \tag{4}$$

  where $\mathcal{P}_i f(\boldsymbol{x})$ is the i-th element in the patch at the voxel $\boldsymbol{x}$.

- Update the manifold $\mathcal{M}^{k+1}$ by setting:

$$\mathcal{M}^{k+1} = \alpha^{f^{k+1}}(\mathcal{M}^k)$$

If f^k converges to a solution f^*, then $\alpha^{f^*} = \alpha$, the identity map, so that $\mathcal{M}^k$ converges to a manifold $\mathcal{M}^*$. f^* is then the LDMM approximation of the unknown data.

The remaining question is how to solve (4). In [20], the authors transformed the Euler-Lagrange equation of (4) into an integral equation, which was solved by the point integral method [21]. This procedure avoids discretizing the manifold gradient operator $\nabla_{\mathcal{M}}$, and is shown to perform very well on image inpainting. However, the point integral method involves solving d linear equations on the patch domain per iteration, which makes the numerical procedure very computationally expensive. In [23], the authors presented an alternative solution procedure by using the weighted graph Laplacian (WGL) [22] to discretize $\nabla_{\mathcal{M}}$ directly. This speeds up the numerical computation significantly because only one linear equation is to be solved every iteration. We hereby briefly introduce for completeness the intuition and implementation of WGL.

2.3. Weighted Graph Laplacian

The weighted graph Laplacian (WGL) was recently proposed in [22] to smoothly interpolate functions on a point cloud. Let $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n\}$ be a set of points in $\mathbb{R}^d$, and let g be a function defined on a subset $S = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n\} \subset C$. The goal is to extend g to C by finding a smooth function u on $\mathcal{M}$ that agrees with g when restricted to S.

The widely used harmonic extension model [25, 26] seeks to solve the interpolation problem by minimizing the following energy:

$$\mathcal{J}(u) = \|\nabla_{\mathcal{M}} u\|^2_{L^2(\mathcal{M})}, \quad \text{subject to:} \quad u(\boldsymbol{p}) = g(\boldsymbol{p}) \quad \text{on } S. \tag{5}$$

A common way to discretize the manifold gradient $\nabla_{\mathcal{M}} u$ is to use the nonlocal approximation:

$$\nabla_{\mathcal{M}} u(\boldsymbol{p})(\boldsymbol{q}) \approx \sqrt{w(\boldsymbol{p}, \boldsymbol{q})}\,(u(\boldsymbol{p}) - u(\boldsymbol{q}))\,,$$

where w is a positive weight function, e.g. $w(\boldsymbol{p}, \boldsymbol{q}) = \exp\left(-\frac{\|\boldsymbol{p}-\boldsymbol{q}\|^2}{\sigma^2}\right)$. With this approximation

$$\mathcal{J}(u) \approx \sum_{\boldsymbol{p},\boldsymbol{q} \in P} w(\boldsymbol{p}, \boldsymbol{q})\,(u(\boldsymbol{p}) - u(\boldsymbol{q}))^2\,. \tag{6}$$

Such discretization leads to the well-known graph Laplacian method [25, 27, 28].

A closer look into the energy $\mathcal{J}$ in (6) reveals that the model will fail to achieve satisfactory results when the sample rate $|S|/|C|$ is very low. More specifically, after rewriting (6) in the following form:

$$\mathcal{J}(u) = \sum_{\boldsymbol{p} \in S} \sum_{\boldsymbol{q} \in C} w(\boldsymbol{p}, \boldsymbol{q})\,(u(\boldsymbol{p}) - u(\boldsymbol{q}))^2 + \sum_{\boldsymbol{p} \in C \setminus S} \sum_{\boldsymbol{q} \in C} w(\boldsymbol{p}, \boldsymbol{q})\,(u(\boldsymbol{p}) - u(\boldsymbol{q}))^2\,, \tag{7}$$

one can see that the first term in (7) is much smaller than the second term when $|S| \ll |C|$. As a result, the minimizing procedure will prioritize the second term, and therefore sacrifice the continuity of u on the sampled set S. An easy remedy for this scenario is to add a large weight $\mu = |C|/|S|$ in front of the first term in (7) to balance the two terms:

$$\mathcal{J}_{\mathrm{WGL}}(u) = \mu \sum_{\boldsymbol{p}\in S} \sum_{\boldsymbol{q}\in C} w(\boldsymbol{p},\boldsymbol{q}) \left(u(\boldsymbol{p}) - u(\boldsymbol{q})\right)^2 + \sum_{\boldsymbol{p}\in C\setminus S} \sum_{\boldsymbol{q}\in C} w(\boldsymbol{p},\boldsymbol{q}) \left(u(\boldsymbol{p}) - u(\boldsymbol{q})\right)^2. \tag{8}$$

It is readily checked that $\mathcal{J}_{\mathrm{WGL}}$ generalizes the graph Laplacian $\mathcal{J}$ in the sense that $\mathcal{J}_{\mathrm{WGL}} = \mathcal{J}$ when $|S| = |C|$. The generalized energy functional $\mathcal{J}_{\mathrm{WGL}}$ is called the weighted graph Laplacian.

We point out that such intuition can be made precise by deriving (8) through the point integral method. The interested reader can refer to [22] for the details.

3. Numerical Implementation

In this section, we provide a detailed explaination of the numerical implementation of LDMM. Using the terminology introduced in Section 2.3, the functions to be interpolated in (4) are α_i, the point cloud C is $\mathcal{P}(f^k)$, and the sampled set for α_i is $S_i = \left\{\mathcal{P}f^k(\boldsymbol{x}) : \mathcal{P}_i f^k(\boldsymbol{x}) \text{ is sampled}\right\}$. Based on the discussion in Section 2.3, (4) can be discretized into the following problem:

$$\begin{aligned}\min_{f\in\mathbb{R}^{m\times n\times r}} \quad & \sum_{i=1}^{d} \left(\sum_{\boldsymbol{x}\in\overline{\Omega}\setminus\Omega_i} \sum_{\boldsymbol{y}\in\overline{\Omega}} \overline{w}(\boldsymbol{x},\boldsymbol{y})(\mathcal{P}_i f(\boldsymbol{x}) - \mathcal{P}_i f(\boldsymbol{y}))^2 \right. \\ & \left. \qquad +\mu \sum_{\boldsymbol{x}\in\Omega_i} \sum_{\boldsymbol{y}\in\overline{\Omega}} \overline{w}(\boldsymbol{x},\boldsymbol{y})(\mathcal{P}_i f(\boldsymbol{x}) - \mathcal{P}_i f(\boldsymbol{y}))^2 \right), \end{aligned} \tag{9}$$

$$\text{Subject to:} \qquad f(\boldsymbol{x}) = b(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \subset \overline{\Omega},$$

where $\mu = \frac{|\bar{\Omega}|}{|\Omega|}$, $\Omega_i = \left\{\boldsymbol{x} \in \bar{\Omega} : \mathcal{P}_i f^k(\boldsymbol{x}) \text{ is sampled}\right\}$, the values $\bar{w}(\boldsymbol{x},\boldsymbol{y}) = w(\mathcal{P}f(\boldsymbol{x}), \mathcal{P}f(\boldsymbol{y}))$ form the elements of a matrix $\bar{\boldsymbol{W}}$, and w is a symmetric sparse weight function computed from the point cloud $\mathcal{P}f^k$. More specifically,

$$w(\boldsymbol{p},\boldsymbol{q}) = \exp\left(-\frac{\|\boldsymbol{p}-\boldsymbol{q}\|^2}{\sigma(\boldsymbol{p})\sigma(\boldsymbol{q})}\right), \tag{10}$$

where $\sigma(\boldsymbol{p})$ is the normalizing factor. In the numerical experiments, the weight w has been truncated to 20 nearest neighbors using the space-partitioning data structure k-d tree [29]. We employ a randomized and approximate version of the algorithm [30, 31] implemented in the open source VLFeat package[6] [32].

[6] http://ww.vlfeat.org

The normalizing factor is chosen as the distance between $\boldsymbol{p}$ and its 10th nearest neighbor.

In order to derive the Euler-Lagrange equation of (9), we define $\mathcal{P}_i$ as the translation operator that maps f into the shifted data set $\mathcal{P}_i f = (\mathcal{P}_i f(\boldsymbol{x}))_{\boldsymbol{x}\in\bar{\Omega}}$, where $\mathcal{P}_i f(\boldsymbol{x})$ is the i-th element in the patch at the voxel $\boldsymbol{x}$ defined in (4), and a periodic padding is used when patches exceed the domain of the 3D data set. With such padding, the adjoint operator $\mathcal{P}_i^*$ of $\mathcal{P}_i$ is equal to its inverse $\mathcal{P}_i^{-1}$. It is readily checked by standard variational techniques that the Euler-Lagrange equation of (9) is:

$$\begin{cases} \left[\sum_{i=1}^{d} \mathcal{P}_i^*(h_i) + (\mu-1)\sum_{i=1}^{d} \mathcal{P}_i^*(g_i)\right](\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \bar{\Omega}\setminus\Omega \\ f(\boldsymbol{x}) = b(\boldsymbol{x}), \quad \boldsymbol{x}\in\Omega \end{cases} \tag{11}$$

where

$$\begin{cases} h_i(\boldsymbol{x}) = \sum_{\boldsymbol{y}\in\bar{\Omega}} 2\bar{w}(\boldsymbol{x},\boldsymbol{y})(\mathcal{P}_i f(\boldsymbol{x}) - \mathcal{P}_i f(\boldsymbol{y})) \\ g_i(\boldsymbol{x}) = \sum_{\boldsymbol{y}\in\Omega_i} \bar{w}(\boldsymbol{x},\boldsymbol{y})(\mathcal{P}_i f(\boldsymbol{x}) - \mathcal{P}_i f(\boldsymbol{y})). \end{cases} \tag{12}$$

We use the notation $\boldsymbol{x}_{\widehat{j}}$ to denote the j-th element after $\boldsymbol{x}$ in the patch. It is easy to verify that $\mathcal{P}_i f(\boldsymbol{x}) = f(\boldsymbol{x}_{\widehat{i-1}})$, and $\mathcal{P}_i^* f(\boldsymbol{x}) = \mathcal{P}_i^{-1} f(\boldsymbol{x}) = f(\boldsymbol{x}_{\widehat{1-i}})$.

Using such notation, we have:

$$\begin{aligned} \mathcal{P}_i^* h_i(\boldsymbol{x}) = h_i(\boldsymbol{x}_{\widehat{1-i}}) &= \sum_{\boldsymbol{y}\in\bar{\Omega}} 2\bar{w}(\boldsymbol{x}_{\widehat{1-i}},\boldsymbol{y})\left(\mathcal{P}_i f(\boldsymbol{x}_{\widehat{1-i}}) - \mathcal{P}_i f(\boldsymbol{y})\right) \\ &= \sum_{\boldsymbol{y}\in\bar{\Omega}} 2\bar{w}(\boldsymbol{x}_{\widehat{1-i}},\boldsymbol{y})\left(f(\boldsymbol{x}) - f(\boldsymbol{y}_{\widehat{i-1}})\right) \\ &= \sum_{\boldsymbol{y}\in\bar{\Omega}} 2\bar{w}(\boldsymbol{x}_{\widehat{1-i}},\boldsymbol{y}_{\widehat{1-i}})\left(f(\boldsymbol{x}) - f(\boldsymbol{y})\right). \end{aligned}$$

Therefore

$$\sum_{i=1}^{d} \mathcal{P}_i^*(h_i)(\boldsymbol{x}) = \sum_{i=1}^{d}\sum_{\boldsymbol{y}\in\bar{\Omega}} 2\bar{w}(\boldsymbol{x}_{\widehat{1-i}},\boldsymbol{y}_{\widehat{1-i}})\left(f(\boldsymbol{x}) - f(\boldsymbol{y})\right). \tag{13}$$

Similarly,

$$\sum_{i=1}^{d} \mathcal{P}_i^*(g_i)(\boldsymbol{x}) = \sum_{i=1}^{d}\sum_{\boldsymbol{y}\in\Omega} \bar{w}(\boldsymbol{x}_{\widehat{1-i}},\boldsymbol{y}_{\widehat{1-i}})\left(f(\boldsymbol{x}) - f(\boldsymbol{y})\right). \tag{14}$$

When we substitute (13) and (14) into (11), the Euler-Lagrange equation

becomes:

$$\begin{cases} \displaystyle\sum_{\boldsymbol{y}\in\bar{\Omega}} \left(\sum_{i=1}^{d} 2\bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}_{\widehat{1-i}}) \right) (f(\boldsymbol{x}) - f(\boldsymbol{y})) \\ \displaystyle + (\mu-1) \sum_{\boldsymbol{y}\in\Omega} \left(\sum_{i=1}^{d} \bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}_{\widehat{1-i}}) \right) (f(\boldsymbol{x}) - f(\boldsymbol{y})) = 0, \quad \boldsymbol{x} \in \bar{\Omega} \setminus \Omega \\ f(\boldsymbol{x}) = b(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega. \end{cases}$$

Let $\tilde{w}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} \bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}_{\widehat{1-i}})$, i.e. $\tilde{\boldsymbol{W}}$ is assembled from translated versions of the original matrix $\bar{\boldsymbol{W}}$, then

$$\begin{cases} \displaystyle 2\sum_{\boldsymbol{y}\in\bar{\Omega}} \tilde{w}(\boldsymbol{x}, \boldsymbol{y}) (f(\boldsymbol{x}) - f(\boldsymbol{y})) + red(\mu-1) \sum_{\boldsymbol{y}\in\Omega} \tilde{w}(\boldsymbol{x}, \boldsymbol{y}) (f(\boldsymbol{x}) - f(\boldsymbol{y})) = 0, \quad \boldsymbol{x} \in \bar{\Omega} \setminus \Omega \\ f(\boldsymbol{x}) = b(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega. \end{cases} \tag{15}$$

Define the graph Laplacian matrix $\tilde{\boldsymbol{L}}$ associated with the new weight matrix $\tilde{\boldsymbol{W}}$ as $\tilde{\boldsymbol{L}} = \tilde{\boldsymbol{D}} - \tilde{\boldsymbol{W}}$, where $\tilde{\boldsymbol{D}}$ is the diagonal matrix with diagonal entries $\tilde{\boldsymbol{D}}(\boldsymbol{x}, \boldsymbol{x}) = \sum_{\boldsymbol{y}\in\bar{\Omega}} \tilde{w}(\boldsymbol{x}, \boldsymbol{y})$. It is easy to check that (15) can be written in the matrix form:

$$\left(2\tilde{\boldsymbol{L}}_{11} + (\mu-1)\boldsymbol{\Delta}\right) \boldsymbol{v} = (\mu+1)\tilde{\boldsymbol{W}}_{12}\boldsymbol{b} \tag{16}$$

where $\tilde{\boldsymbol{W}}_{ij}$ and $\tilde{\boldsymbol{L}}_{ij}$ are submatrices corresponding to unsampled $(i, j = 1)$ or sampled $(i, j = 2)$ parts of $\tilde{\boldsymbol{W}}$ and $\tilde{\boldsymbol{L}}$, $\boldsymbol{v}$ and $\boldsymbol{b}$ correspond to unsampled and sampled parts of $\boldsymbol{f}$, and $\boldsymbol{\Delta}$ is the diagonal matrix with its diagonal entries equaling the sums of the rows of $\tilde{\boldsymbol{W}}_{12}$. See Figure 1 for a visual illustration of the definitions of the matrices.

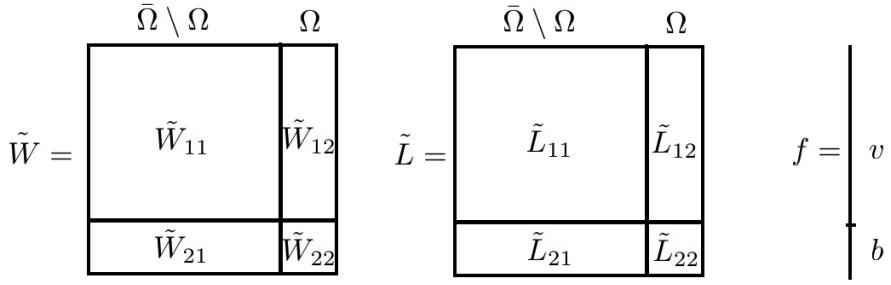


Figure 1: A visual illustration of the matrix/vector definitions. The matrices $\tilde{\boldsymbol{W}}$, $\tilde{\boldsymbol{L}}$ and $\boldsymbol{f}$ are partitioned into sampled (Ω) and unsampled ($\bar{\Omega} \setminus \Omega$) blocks. For example, $\tilde{\boldsymbol{W}}_{12}$ is the matrix corresponding to the weights between unsampled and sampled points.

The final LDMM algorithm for 3D scientific data reconstruction from partial sampling is shown in Algorithm 1. As a remark, we point out that in our

current Matlab and C++ implementation, the most time consuming part of the algorithm is step 3, the assembling of the weight matrices, which involves permutations of sparse weight matrices. We reduce this cost with a parallelization implementation in the matrix assembly step.

Algorithm 1 LDMM for 3D scientific data reconstruction from partial sampling

Require: A subsampled data $f|_\Omega = b$.
Ensure: Reconstructed data f.

Initial guess f^0.

while not converge **do**

1. Compute the patch set $\mathcal{P}(f^k)$ from the current iterate f^k.
2. Compute the weight function

$$\overline{w}(\boldsymbol{x}, \boldsymbol{y}) = w(\mathcal{P}f^k(\boldsymbol{x}), \mathcal{P}f^k(\boldsymbol{y})), \quad \boldsymbol{x}, \boldsymbol{y} \in \overline{\Omega}.$$

3. Assemble the new weight function

$$\tilde{w}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} \bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}_{\widehat{1-i}})$$

4. Update the data f^{k+1} by solving for variable v in equation (16).
5. $k \leftarrow k + 1$.

end while

$f = f^k$.

4. Numerical Results

In this section, we present the numerical results of LDMM on various 2D and 3D scientific data interpolation from either regular or irregular samplings. The performance of LDMM is compared to that of the exemplar-based interpolation (EBI) [16] and the piecewise linear estimator (PLE) [18] in the case of random sampling interpolation. As pointed out in [18], PLE fails to work on regular sampling interpolation without a proper initialization (bicubic interpolation in their case). We also noticed in our experiment that the result of EBI on regular sampling interpolation is inferior to that of the simple cubic spline interpolation. Therefore, in the case of regular sampling interpolation, we instead compare the results of LDMM to the standard methods including cubic spline interpolation, discrete Fourier transform (DFT), discrete cosine transform (DCT), and wavelet transform. Moreover, we also examine the effectiveness of LDMM as a data compression technique and compare it to other standard compression methods including DFT, DCT, wavelet transform, and tensor decomposition. As for the

tensor decomposition methods, we use the singular value decomposition (SVD) for 2D data sets, and the Tucker decomposition [33, 34] for 3D data sets. The Tucker decomposition is a form of higher-order SVD, which decomposes a tensor into a core tensor multiplied by a matrix along each mode.

4.1. Description of the Testing Data sets and Parameter Setup

The algorithms are tested on six scientific data sets, three of which are three-dimensional. See Figure 2 and Figure 3 for visual illustrations of the data sets.

- **3D plasma (magnetic field)**: The data set is taken from a gyrokinetic simulation of Alfvénic turbulence in 5D phase space (3D real space plus 2D velocity space, with the fast gyroangle dependence removed) [35], carried out with the GENE code [36]. It represents a snapshot of the magnitude of magnetic field fluctuations in real space during the statistically quasi-stationary state of fully developed turbulence. In this simulation, the focus is on the dissipation range of this weakly collisional turbulent plasma which cannot be described adequately by magnetohydrodynamics (MHD). Gyrokinetics offers an efficient description of the very tail of the MHD cascade. The size of this data is $256 \times 256 \times 32$.

- **3D/2D lattice**: The lattice benchmark problem, originally due to Brunner [37, 38], is a two-dimensional cartoon of a nuclear reactor assembly that has become a common test problem of angular discretization methods for kinetic equations of radiation transport [39, 40, 41, 42].

  A schematic of the problem is shown in Figure 4. It involves a particle source surrounded by a checkerboard array of highly absorbing material (gray) embedded within a lightly scattering material (white). Particles are emitted into the domain through a central source region (red).

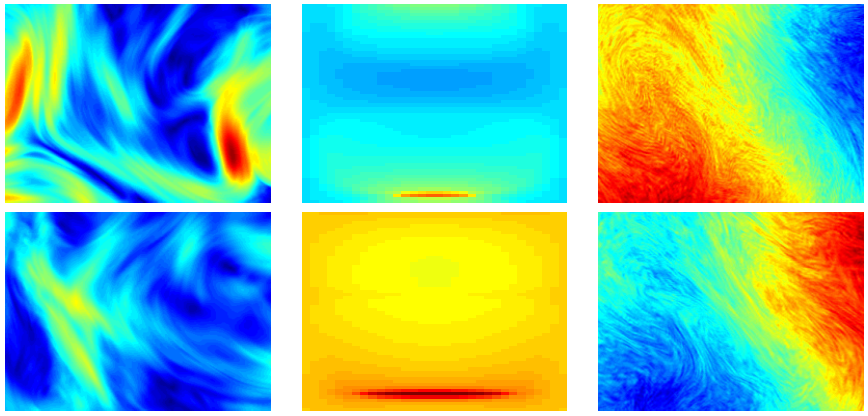
  The simulated quantity is a distribution function that depends on five independent variables: two spatial, two angular, plus time. The data used here was generated using the algorithm described in [43] which combines a third-order space-time discretization (discontinuous Galerkin in space and integral deferred correction in time) and an angular discretization based on a tensor product collocation scheme.

  We consider for this problem two quantities of interest. The first (**2D lattice**) is the angular average of the distribution function at a fixed time; this is a two-dimensional data set of size 896×896. The second is the distribution function at a fixed time and fixed vertical location along the line $y = 4.5$. This is a three dimensional data set of size $188 \times 64 \times 32$. Both sets of data are given in log scale.

- **3D/2D plasma (distribution function)**: This data set is again taken from a gyrokinetic simulation of Alfvénic turbulence in 5D phase space (3D real space plus 2D velocity space, with the fast gyroangle dependence removed) described in [35]. The 3D data set describes the distribution function for the ion species as a function of the two spatial coordinates

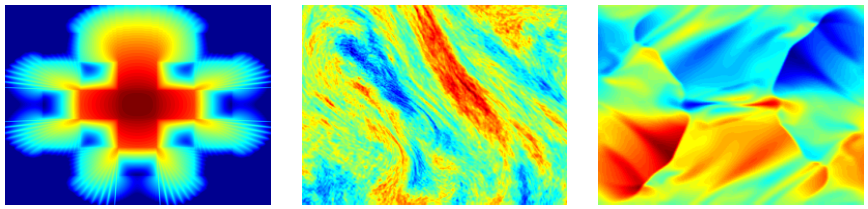
perpendicular to the background magnetic field and of the velocity parallel to this guide field at a given value of perpendicular velocity and time. Meanwhile, the 2D data set describes a snapshot of the same distribution function for the ion species as a function of the two perpendicular spatial coordinates integrated over velocity space. The sizes of the 3D and 2D data sets are $256 \times 256 \times 32$ and 256×256 respectively.

- **2D vortex**: This data set comes from a numerical solution of the Orszag-Tang vortex system [44], which provides a model of complex flow with many features of magnetohydrodynamics systems. Starting from a smooth state, the system evolves into turbulance, generating complex interactions between different shock waves. The data set used in this paper is the numerical solution at time $t = 2$ of the density component obtained with the third order Chebyshev polynomial approximate Osher-Solomon scheme [45] on a 256×256 uniform mesh.



(a) (b) (c)

Figure 2: Visual illustrations of the 3D data sets. The two figures in column (a) are 2D spatial cross sections of the 3D plasma (magnetic field) data set at different z coordinates. The figures in column (b) are 2D cross sections of the 3D lattice data set correspongding to angular flux at $x = 0.24$ and $x = 1.18$. The figures in column (c) are 2D spatial cross sections of the 3D plasma (distribution function) data set.



(a) (b) (c)

Figure 3: Visual illustrations of the 2D data sets. (a) 2D lattice. (b) 2D plasma (distribution function). (c) 2D vortex.

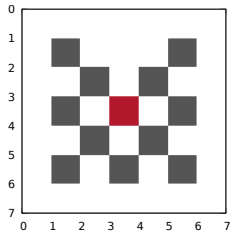


Figure 4: The schematic of the lattice benchmark problem. A central particle source region (red) is surrounded by a checkerboard array of highly absorbing material (gray) embedded within a lightly scattering material (white).

For irregular sampling interpolation, the algorithms are tested to reconstruct the original data sets from 5% and 10% random subsamples. For the regular aliased sampling, the original 2D data sets are decimated by a factor of 4 in both directions; for 3D data sets, we consider two types of sampling procedures: downsampling by a factor of 2 in all directions, or by a factor of 4 in only the first two dimensions.

For all the data sets listed above, the weight matrices in LDMM are truncated to 20 nearest neighbors, and the normalizing factor $\sigma(\boldsymbol{p})$ in (10) is chosen as the distance between $\boldsymbol{x}$ and its 10th nearest neighbor. The patch sizes chosen for different data sets are listed in Table 1. The reason why the 2D plasma (distribution function) data set uses a much larger patch size, 16×16 instead of 6×6, is that the structures in this data set are much more complicated than the other data sets. This complexity implies a much higher intrinsic dimension of the patch manifold. Therefore a larger patch size is chosen so that the manifold dimension can be still smaller than that of the embedding space. Notice also that $6 \times 6 \times 1$ patch size is chosen for the 3D plasma (magnetic field) data set. This is because of the low resolution of the data set in the third dimension. However, $6 \times 6 \times 4$ patches are chosen in the $2 \times 2 \times 2$ regular down sampling. This is because we want to avoid patches that do not contain any sampled voxels.

	5%	10%	4×4	$4 \times 4 \times 1$	$2 \times 2 \times 2$
2D lattice	6×6	6×6	6×6	N/A	N/A
2D plasma (D)	16×16	16×16	16×16	N/A	N/A
2D vortex	6×6	6×6	6×6	N/A	N/A
3D plasma (M)	$6 \times 6 \times 1$	$6 \times 6 \times 1$	N/A	$6 \times 6 \times 1$	$6 \times 6 \times 4$
3D lattice	$4 \times 4 \times 4$	$4 \times 4 \times 4$	N/A	$4 \times 4 \times 4$	$4 \times 4 \times 4$
3D plasma (D)	$6 \times 6 \times 4$	$6 \times 6 \times 4$	N/A	$6 \times 6 \times 4$	$6 \times 6 \times 4$

Table 1: Patch sizes for different datasets. The first row of the table indicates the different types of irregular and regular downsampling procedures. 3D/2D plasma (D) stands for 3D/2D plasma (distribution function), and 3D plasma (M) stands for 3D plasma (magetic field).

The quality of the reconstruction $\hat{f}$ of the original data $f \in \mathbb{R}^{m \times n \times r}$ ($r = 1$ for 2D data sets) is evaluated in the following three norms:

$$\|e\|_1 = \frac{1}{mnr} \sum_{i,j,k} |e_{i,j,k}/R|, \tag{17}$$

$$\|e\|_2 = \left(\frac{1}{mnr} \sum_{i,j,k} |e_{i,j,k}/R|^2 \right)^{\frac{1}{2}}, \tag{18}$$

$$\|e\|_\infty = \max_{i,j,k} |e_{i,j,k}/R|, \tag{19}$$

where $e = f - \hat{f}$ is the error of the reconstruction, $R = \max_{i,j,k} \hat{f}_{i,j,k} - \min_{i,j,k} \hat{f}_{i,j,k}$ is the numerical range of the data set. Moreover, the peak signal-to-noise ratio (PSNR), which is related to (18), is also given to measure the performance of the algorithms:

$$PSNR = 10 \log_{10} \left(\frac{1}{\|f - \hat{f}\|_2^2} \right). \tag{20}$$

4.2. Interpolation with Random Sampling

The visual of the interpolation with 10% and 5% are shown in Figure 5-12. The errors of the reconstruction in different norms are displayed in Table 2-7. It can be observed that LDMM consistently performs at a higher accuracy than EBI and PLE either visually or numerically. The superiority of LDMM is more dramatic when the sample rate is very low (5%), in which case PLE fails to achieve reasonable results. LDMM also manages to yield smoother results, whereas EBI tends to create artificial patchy patterns. We point out that the reconstruction of the 3D data sets with PLE and EBI are obtained by applying the algorithms to 2D cross sections because of a lack of 3D implementations of both algorithms. Therefore it is not entirely fair to compare LDMM to PLE and EBI on the 3D data sets. This is especially clear on the 3D lattice data set, where values change smoothly on each direction. Nonetheless, the vast superiority of LDMM on 2D examples illustrates its advantage over the competing algorithms.

The numerical convergence of LDMM in PSNR is shown in Figure 13. It can be observed that the algorithm converges fairly fast, usually within 10 iterations, and the result does not deteriorate as the iteration goes on.

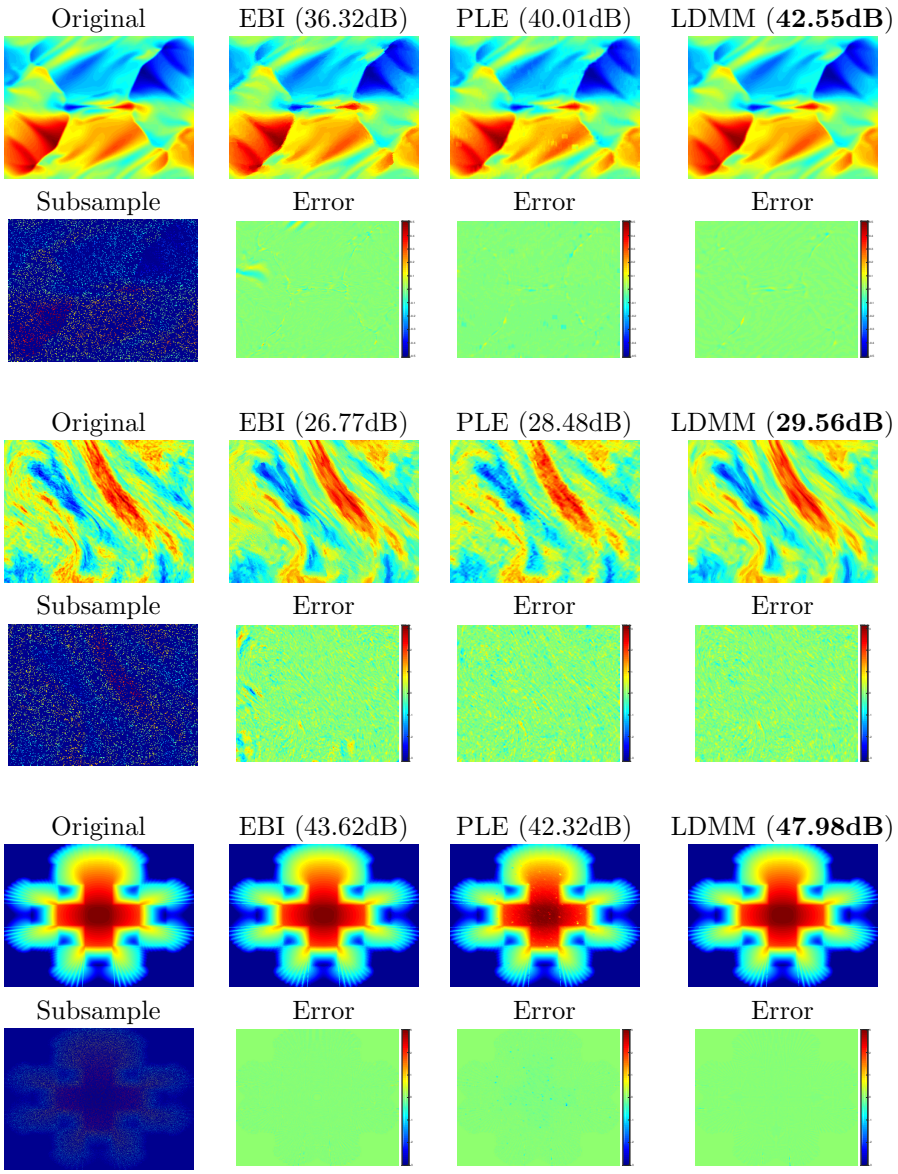


Figure 5: Interpolation of 2D scientific data sets from 10% random sampling. The figures in the first column are the original and subsampled data. The figures in the other three columns are the results and errors of the competing algorithms.

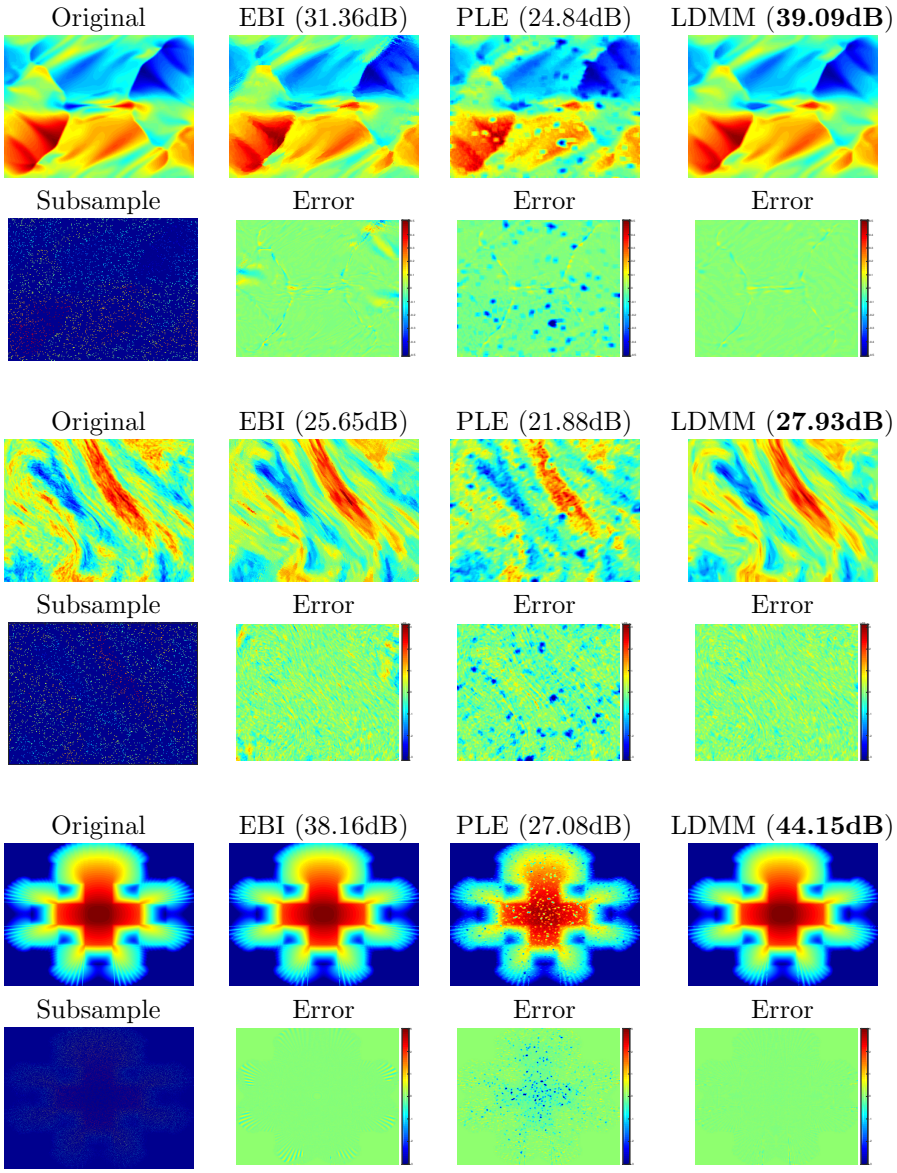


Figure 6: Interpolation of 2D scientific data sets from 5% random sampling. The figures in the first column are the original and subsampled data. The figures in the other three columns are the results and errors of the competing algorithms.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0082	0.0053	**0.0034**	L_1	0.0148	0.0296	**0.0056**
L_2	0.0153	0.0100	**0.0075**	L_2	0.0270	0.0573	**0.0111**
L_∞	0.2280	**0.1232**	0.1376	L_∞	0.3327	0.7872	**0.1102**
PSNR	36.32	40.01	**42.55**	PSNR	31.36	24.84	**39.09**

Table 2: Errors of the interpolation of the 2D vortex data set from 10% and 5% random sampling.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0335	0.0272	**0.0243**	L_1	0.0393	0.0535	**0.0303**
L_2	0.0459	0.0377	**0.0333**	L_2	0.0522	0.0805	**0.0401**
L_∞	0.3782	0.2158	**0.1882**	L_∞	0.2588	0.7148	**0.2063**
PSNR	26.77	28.48	**29.56**	PSNR	25.65	21.88	**27.93**

Table 3: Errors of the interpolation of the 2D plasma (distribution function) data set from 10% and 5% random sampling.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0033	0.0030	**0.0013**	L_1	0.0048	0.0187	**0.0022**
L_2	0.0066	0.0077	**0.0040**	L_2	0.0124	0.0442	**0.0062**
L_∞	0.2172	0.2889	**0.1979**	L_∞	0.8758	0.6156	**0.2097**
PSNR	43.62	42.32	**47.98**	PSNR	38.16	27.08	**44.15**

Table 4: Errors of the interpolation of the 2D lattice data set from 10% and 5% random sampling.

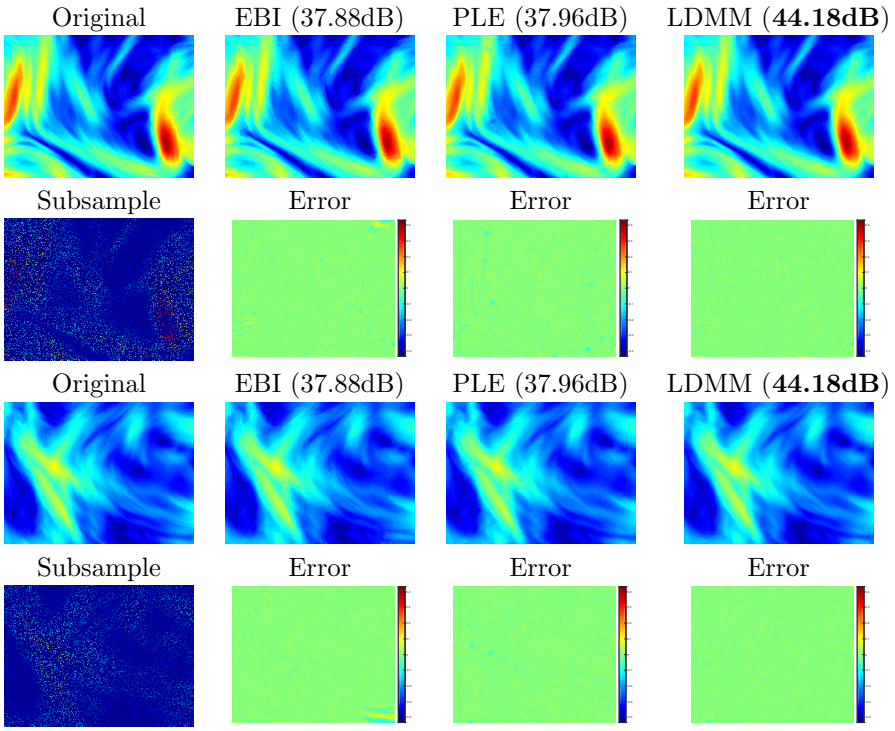


Figure 7: Interpolation of the 3D plasma (magnetic field) data set from 10% random sampling. The figures in the first column are two spatial cross sections of the original and subsampled data. The figures in the other three columns are the results and errors of the competing algorithms.

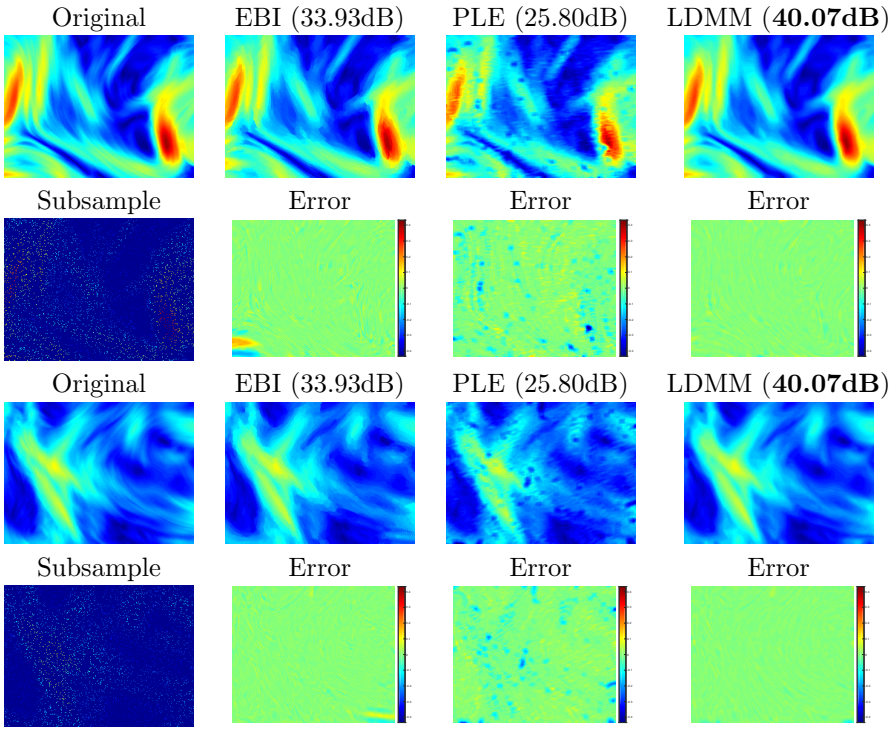


Figure 8: Interpolation of the 3D plasma (magnetic field) data set from 5% random sampling. The figures in the first column are two spatial cross sections of the original and subsampled data. The figures in the other three columns are the results and errors of the competing algorithms.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0075	0.0053	**0.0038**	L_1	0.0115	0.0285	**0.0062**
L_2	0.0128	0.0126	**0.0062**	L_2	0.0201	0.0513	**0.0099**
L_∞	0.3510	0.9432	**0.1330**	L_∞	0.3740	0.7531	**0.2012**
PSNR	37.88	37.96	**44.18**	PSNR	33.93	25.80	**40.07**

Table 5: Errors of the interpolation of the 3D plasma (magnetic field) data set from 10% and 5% random sampling.

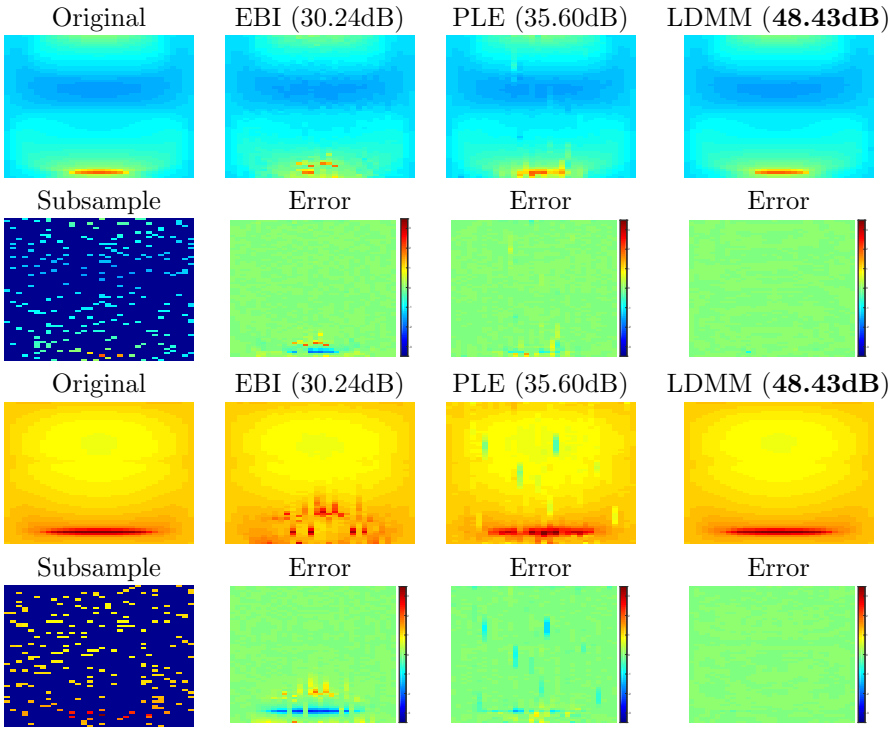


Figure 9: Interpolation of the 3D lattice data set from 10% random sampling. The figures in the first column are the original and subsampled angular flux at $x = 0.24$ and $x = 1.18$. The figures in the other three columns are the results and errors of the competing algorithms.

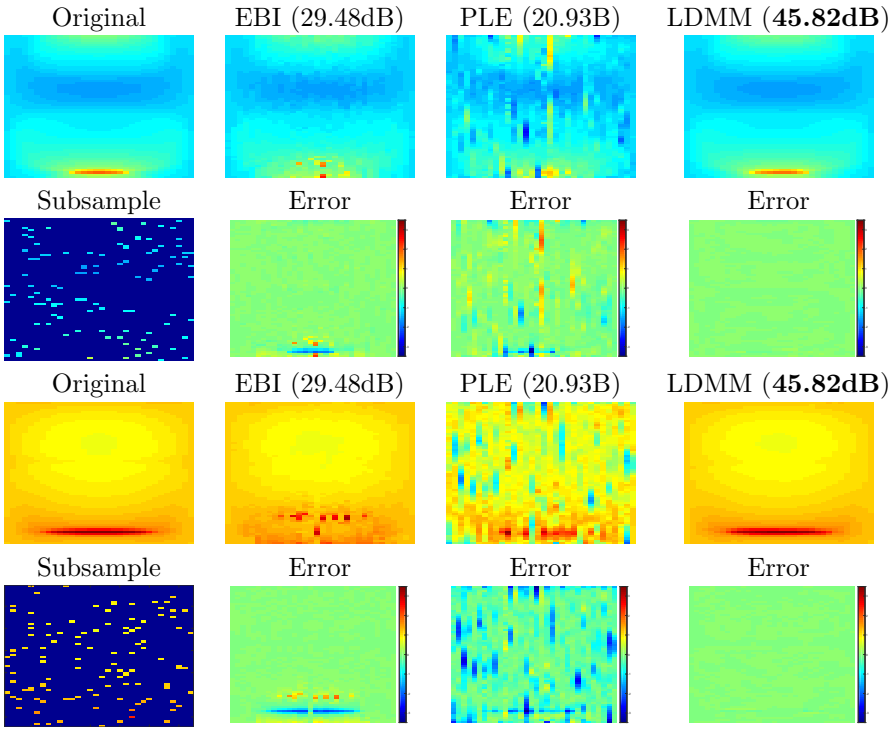


Figure 10: Interpolation of the 3D lattice data set from 5% random sampling. The figures in the first column are the original and subsampled angular flux at $x = 0.24$ and $x = 1.18$. The figures in the other three columns are the results and errors of the competing algorithms.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0094	0.0062	**0.0008**	L_1	0.0112	0.0545	**0.0013**
L_2	0.0308	0.0166	**0.0038**	L_2	0.0336	0.0899	**0.0051**
L_∞	0.5291	0.6635	**0.4262**	L_∞	0.4768	0.0.8595	**0.4530**
PSNR	30.24	35.60	**48.43**	PSNR	29.48	20.93	**45.82**

Table 6: Errors of the interpolation of the 3D lattice data set from 10% and 5% random sampling.

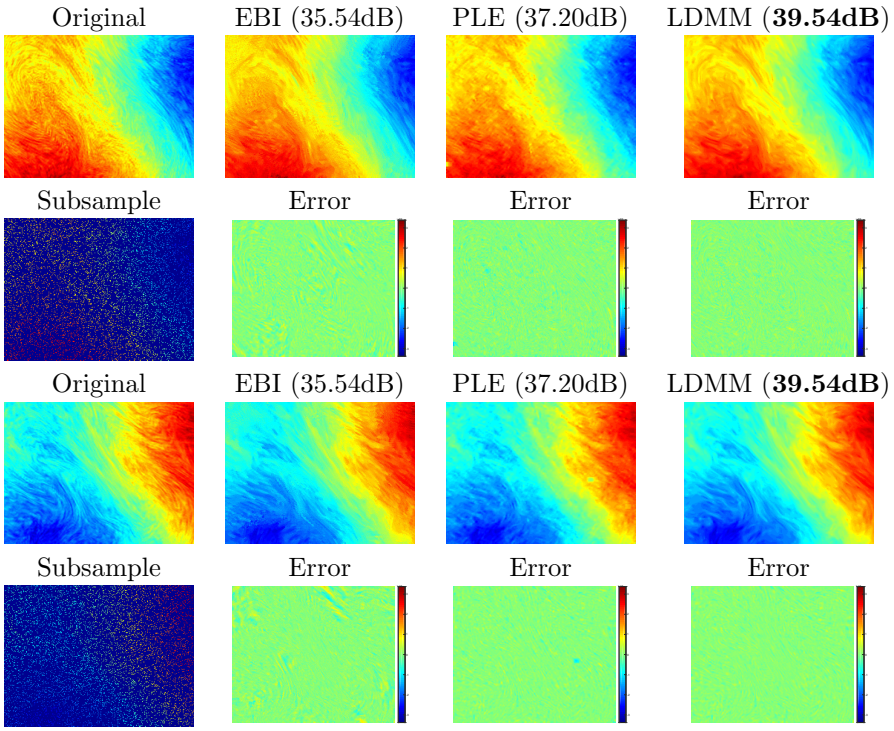


Figure 11: Interpolation of the 3D plasma (distribution function) data set from 10% random sampling. The figures in the first column are two spatial cross sections of the original and subsampled data. The figures in the other three columns are the results and errors of the competing algorithms.

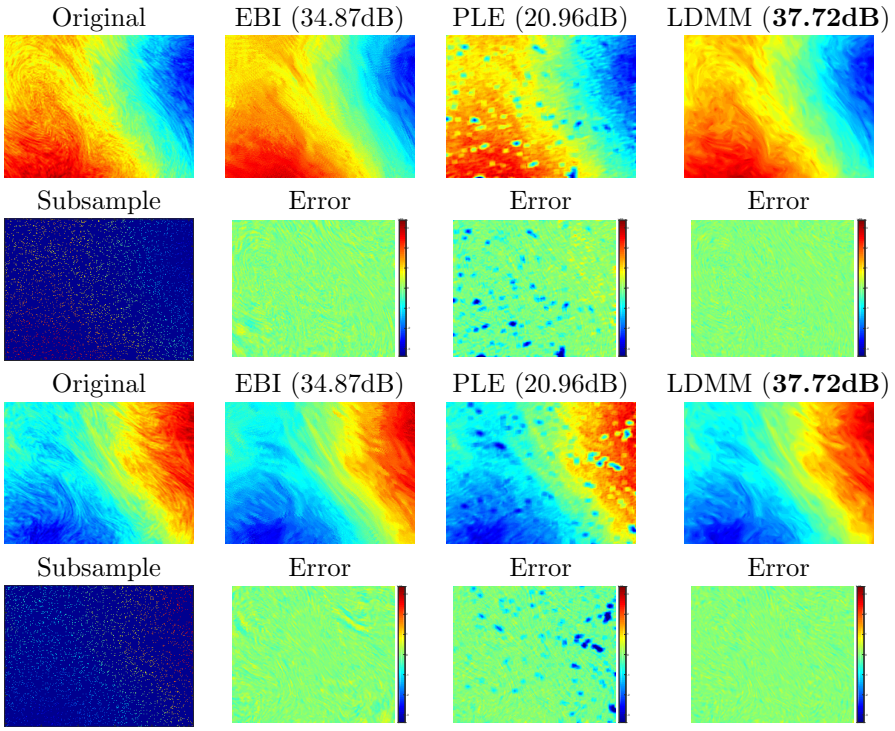


Figure 12: Interpolation of the 3D plasma (distribution function) data set from 5% random sampling. The figures in the first column are two spatial cross sections of the original and subsampled data. The figures in the other three columns are the results and errors of the competing algorithms.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0098	0.0085	**0.0060**	L_1	0.0108	0.0593	**0.0075**
L_2	0.0167	0.0138	**0.0105**	L_2	0.0181	0.0895	**0.0130**
L_∞	0.2005	0.2912	**0.1181**	L_∞	0.1865	0.9093	**0.1793**
PSNR	35.54	37.20	**39.54**	PSNR	34.87	20.96	**37.72**

Table 7: Errors of the interpolation of the 3D plasma (distribution function) data set from 10% and 5% random sampling.

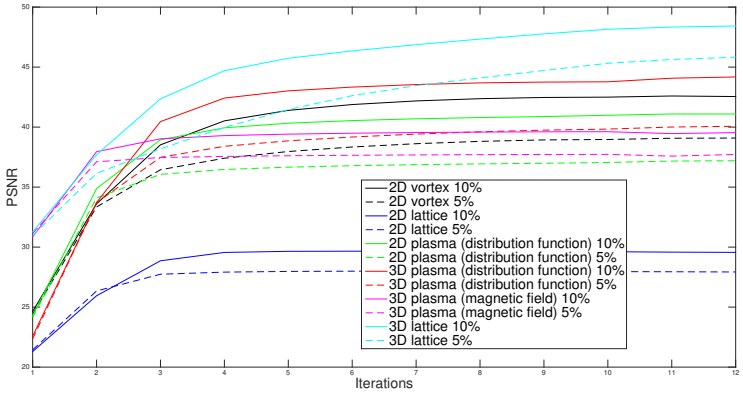


Figure 13: Numerical convergence in PSNR of LDMM on various data sets.

4.3. Interpolation with Regular Sampling

Unlike the random sampling interpolation in the previous section, reasonable initializations of LDMM can be obtained from other standard algorithms for regular sampling interpolation. In the numerical experiments on all the data sets, the results of DCT and cubic spline have been used as the initial iterates for LDMM, and the final results of LDMM initialized with DCT (LDMM (D)) and cubic spline (LDMM (C)) are obtained after three iterations of manifold updates.

The visual of the interpolation with regular sampling (4×4 for 2D data sets, $4 \times 4 \times 1$ and $2 \times 2 \times 2$ for 3D data sets) are shown in Figure 14-20. The errors in different norms are displayed in Table 8-13. It can be observed that the results of LDMM are significantly more accurate than the DCT and cubic spline initializations, and the accuracy of the result does not depend on the choice of the initialization. Moreover, LDMM consistently outperforms all the other competing algorithms on every data set, except for some rare cases where LDMM is inferior in L_1 or L_∞ norms.

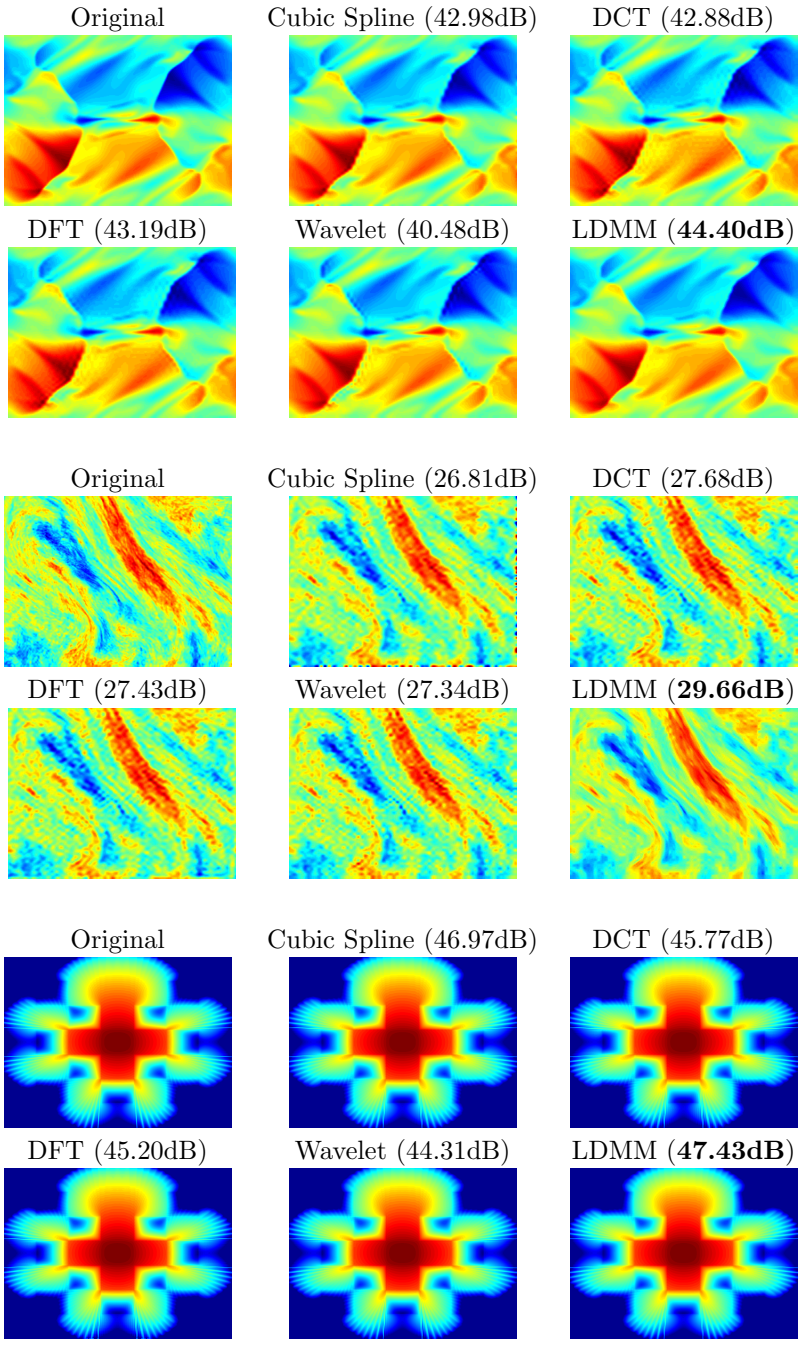


Figure 14: Interpolation of 2D scientific data sets from regular sampling with spacing 4×4. The original data are shown on the upper left corners for each data set. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

4×4	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	**0.0025**	0.0038	0.0035	0.0049	0.0029	0.0028
L_2	0.0071	0.0072	0.0069	0.0095	**0.0060**	0.0061
L_∞	0.1789	**0.0937**	0.0940	0.1122	0.0961	0.1005
PSNR	42.98	42.88	43.19	40.48	**44.40**	44.33

Table 8: Errors of the interpolation of the 2D vortex data set from regular sampling with spacing 4×4.

4×4	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0302	0.0310	0.0314	0.0326	0.0249	**0.0248**
L_2	0.0456	0.0413	0.0425	0.0430	0.0329	**0.0329**
L_∞	0.7629	0.2411	0.3776	0.2514	0.1779	**0.1741**
PSNR	26.81	27.68	27.43	27.34	29.64	**29.66**

Table 9: Errors of the interpolation of the 2D plasma (distribution function) data set from regular sampling with spacing 4×4.

4×4	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	**0.0009**	0.0015	0.0016	0.0020	0.0013	0.0012
L_2	0.0045	0.0051	0.0055	0.0061	0.0044	**0.0041**
L_∞	0.1461	0.1547	0.2202	0.1892	0.1393	**0.1278**
PSNR	46.97	45.77	45.20	44.31	47.18	**47.43**

Table 10: Errors of the interpolation of the 2D lattice data set from regular sampling with spacing 4×4.

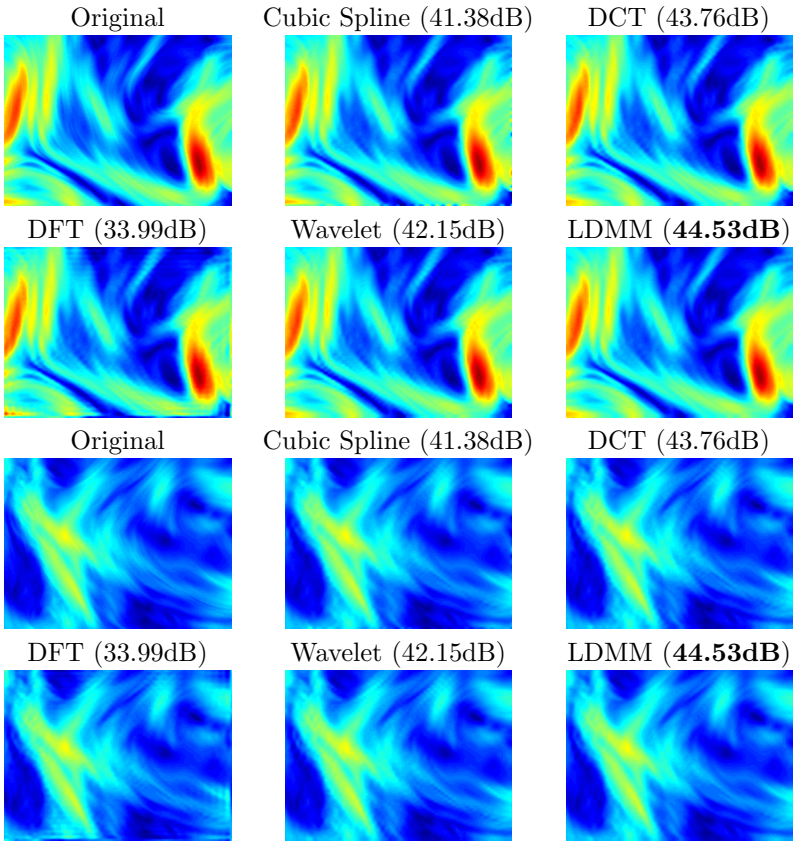


Figure 15: Interpolation of the 3D plasma (magnetic field) data set from regular sampling with spacing $4 \times 4 \times 1$. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

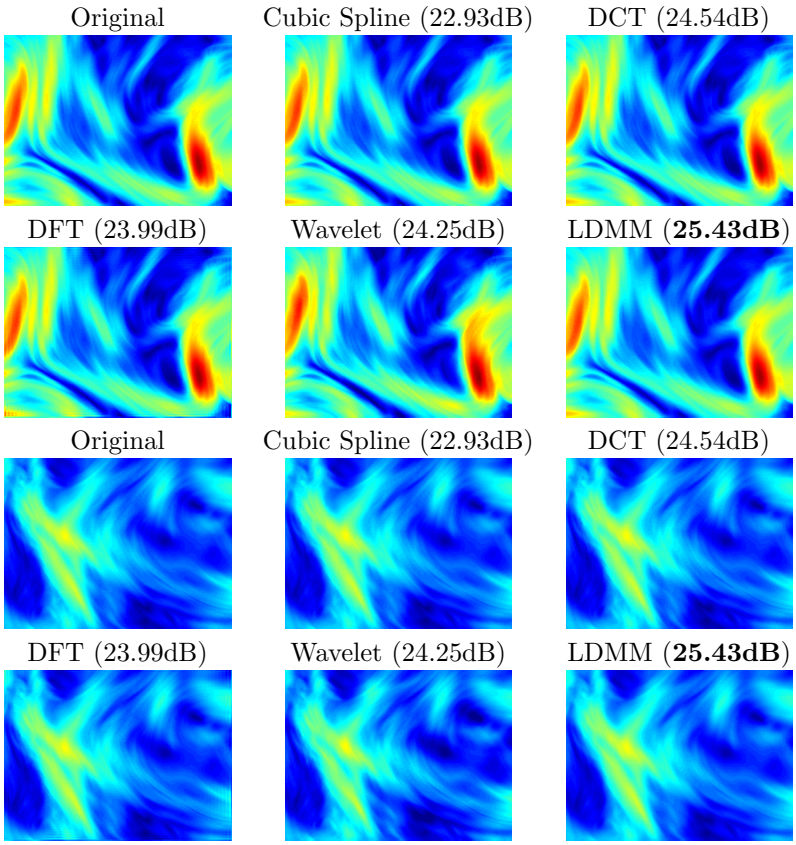


Figure 16: Interpolation of the 3D plasma (magnetic field) data set from regular sampling with spacing $2 \times 2 \times 2$. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

$4 \times 4 \times 1$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0038	0.0040	0.0071	0.0052	0.0037	**0.0036**
L_2	0.0085	0.0065	0.0200	0.0078	**0.0059**	0.0065
L_∞	0.9649	0.1366	0.6449	0.1357	**0.1259**	0.1911
PSNR	41.38	43.76	33.99	42.15	**44.53**	43.73
$2 \times 2 \times 2$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0356	0.0334	0.0352	0.0439	**0.0305**	0.0313
L_2	0.0714	0.0593	0.0632	0.0613	**0.0535**	0.0559
L_∞	0.8770	0.4073	0.5203	0.4283	**0.3711**	0.4060
PSNR	22.93	24.54	23.99	24.25	**25.43**	25.05

Table 11: Errors of the interpolation of the 3D plasma (magnetic field) data set from regular sampling with spacing $4 \times 4 \times 1$ and $2 \times 2 \times 2$.

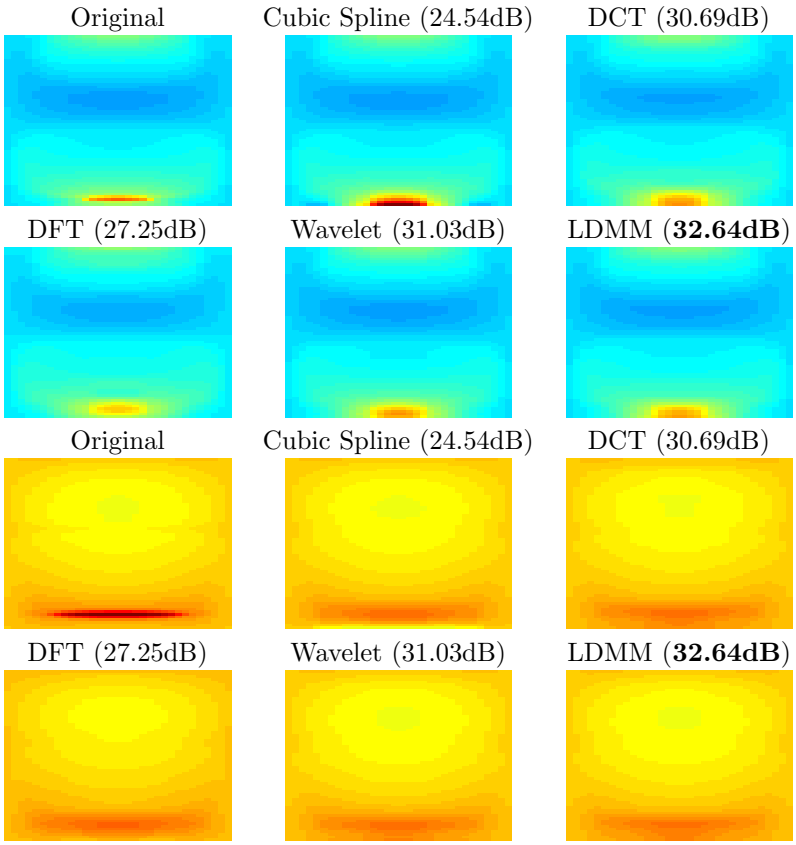


Figure 17: Interpolation of the 3D lattice data set from regular sampling with spacing $4 \times 4 \times 1$. The original angular flux at $x = 0.24$ and $x = 1.18$ are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

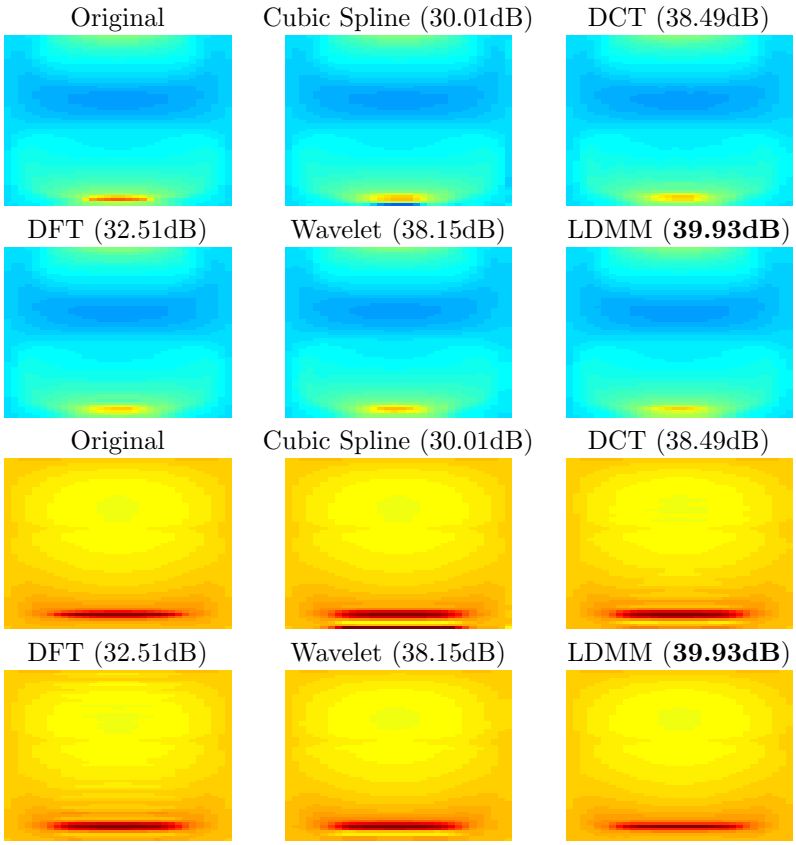


Figure 18: Interpolation of the 3D lattice data set from regular sampling with spacing $2\times2\times2$. The original angular flux at $x = 0.24$ and $x = 1.18$ are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

$4 \times 4 \times 1$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0094	0.0072	0.0168	0.0066	0.0058	**0.0056**
L_2	0.0593	0.0292	0.0434	0.0281	**0.0233**	0.0254
L_∞	1.1890	0.4223	0.5405	0.4245	**0.4164**	0.4362
PSNR	24.54	30.69	27.25	31.03	**32.64**	31.90
$2 \times 2 \times 2$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0039	0.0027	0.0069	0.0045	0.0017	**0.0015**
L_2	0.0316	0.0119	0.0237	0.0124	**0.0101**	0.0101
L_∞	0.7459	0.4109	0.4282	0.4233	**0.4078**	0.4096
PSNR	30.01	38.49	32.51	38.15	**39.93**	39.92

Table 12: Errors of the interpolation of the 3D lattice data set from regular sampling with spacing $4 \times 4 \times 1$ and $2 \times 2 \times 2$.

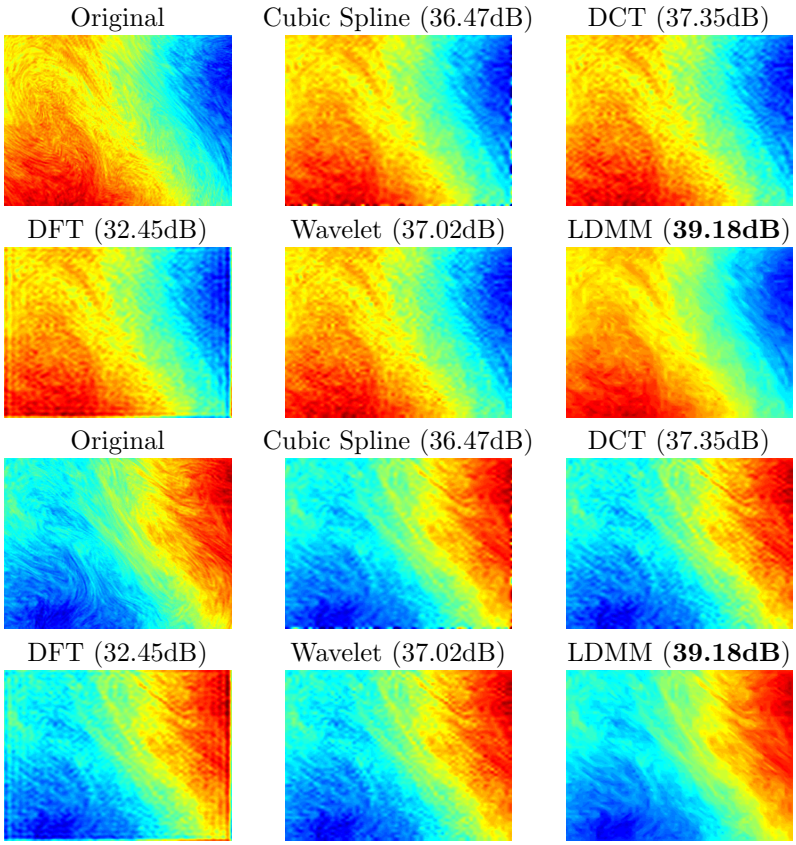


Figure 19: Interpolation of the 3D plasma (distribution function) data set from regular sampling with spacing $4 \times 4 \times 1$. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

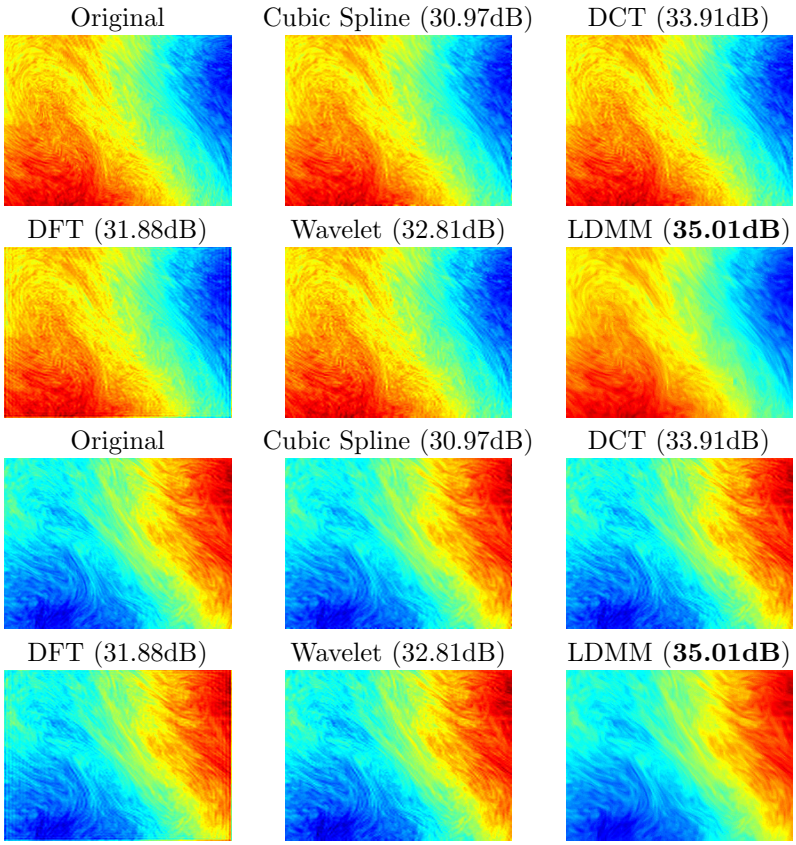


Figure 20: Interpolation of the 3D plasma (distribution function) data set from regular sampling with spacing $2 \times 2 \times 2$. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

$4 \times 4 \times 1$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0076	0.0078	0.0103	0.0083	**0.0064**	0.0064
L_2	0.0150	0.0136	0.0238	0.0141	**0.0110**	0.0111
L_∞	0.8851	0.1551	0.4805	0.1469	**0.1093**	0.1417
PSNR	36.47	37.35	32.45	37.02	**39.18**	39.13
$2 \times 2 \times 2$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0109	0.0098	0.0127	0.0139	**0.0089**	0.0092
L_2	0.0283	0.0202	0.0255	0.0229	**0.0178**	0.0181
L_∞	0.7388	0.2976	0.3438	0.2993	**0.2088**	0.2097
PSNR	30.97	33.91	31.88	32.81	**35.01**	34.85

Table 13: Errors of the interpolation of the 3D plasma (distribution function) data set from regular sampling with spacing $4 \times 4 \times 1$ and $2 \times 2 \times 2$.

4.4. Data Compression

Finally, we compare the performance of LDMM as a sampling-based data compression technique to other standard compression methods including singular value/ Tucker Decomposition, DFT, DCT, and the wavelet transformations. We point out that, unlike the other testing methods which usually involve hard thresholding of the expansion coefficients with respect to a particular basis, LDMM does not require access to the original full data set. Therefore we do not expect LDMM to perform equally well compared to other data compression methods. However, using the sampling-based method as a data compression technique has its own advantages:

- During the data compression step, sampling-based algorithms like LDMM are very easy to implement compared to other standard compression methods. Moreover, in a parallel setting, sampling based methods can be implemented independently on each node without communication, while other methods involving global transforms cannot.

- It is also faster for sampling-based methods to reconstruct a small portion of the data set if only that part of the data set is required.

In the numerical experiments, LDMM with random sampling has been used for each data set. The storage of SVD involves thresholded singular values along with the correponding singular vectors, and the storage of Tucker Decomposition involves a 3D core tensor with reduced size and three matrices for three different modes. For the other methods using global transforms, we store the coefficients with the largest magnitudes with constraint to the given budget. We mention that the results of Tucker Decomposition on 3D data sets are quite sensitive to the dimension of the core tensor along each direction. In our experiments, we choose the best result among all the possible decompositions satisfying the budget. This typically causes Tucker Decomposition to run for about two days on the 3D data sets reported in this paper. The visual and numerical results of the competing methods are reported in Figure 21-28 and Table 14-19. As expected, the performance of LDMM in data compression is usually inferior compared to the other competing methods. However, it does outperform SVD in two of the more complicated 2D data sets (2D vortex and 2D plasma (distribution)) and the wavelet transform in the 3D plasma (magnetic field) data set. DCT almost consistently yields the best result among all the methods, and it can also be observed that tensor decomposition methods tend to achieve better results when the dimension of the data set becomes larger. Therefore, we can conclude that, at least at current stage, LDMM is a viable choice for data compression if the data set is complicated to begin with, and the user is willing to sacrifice accuracy for easy implementation in the compression step.

We point out that although LDMM does not perform equally well in data compression when compared to other methods that assume full access to the entire data set, there is still much room for improvement for LDMM. For instance, instead of randomly sampling the data set in the physical domain, we

may strategically choosing pixels to sample if certain prior information is available. Moreover, if the original data set is known to the user, we can also modify the LDMM algorithm by sampling gradient values or certain entries in the weight matrices. Modifying LDMM for it to work as a data compression method will be the focus of our future work.

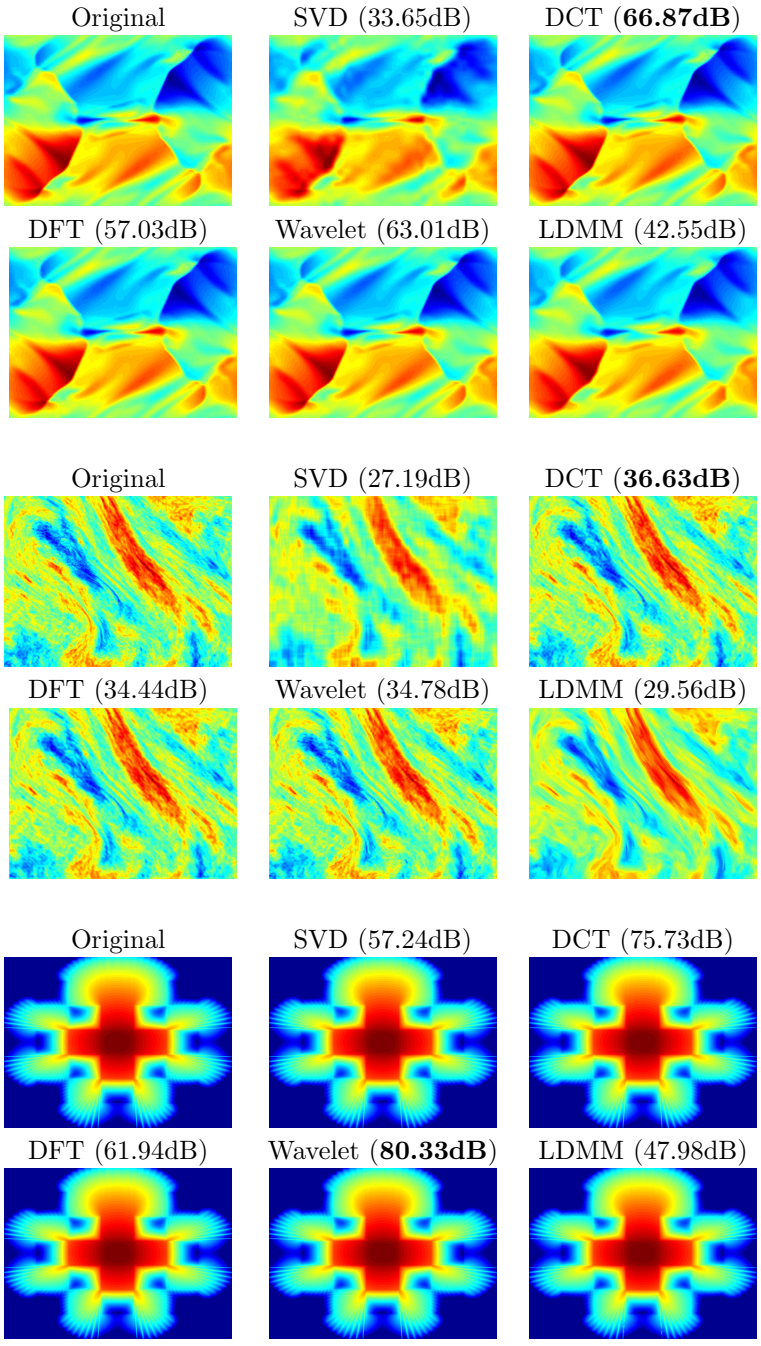


Figure 21: Compression of 2D scientific data sets with a 10% data compression rate. The original data are shown on the upper left corners for each data set. The results of SVD, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

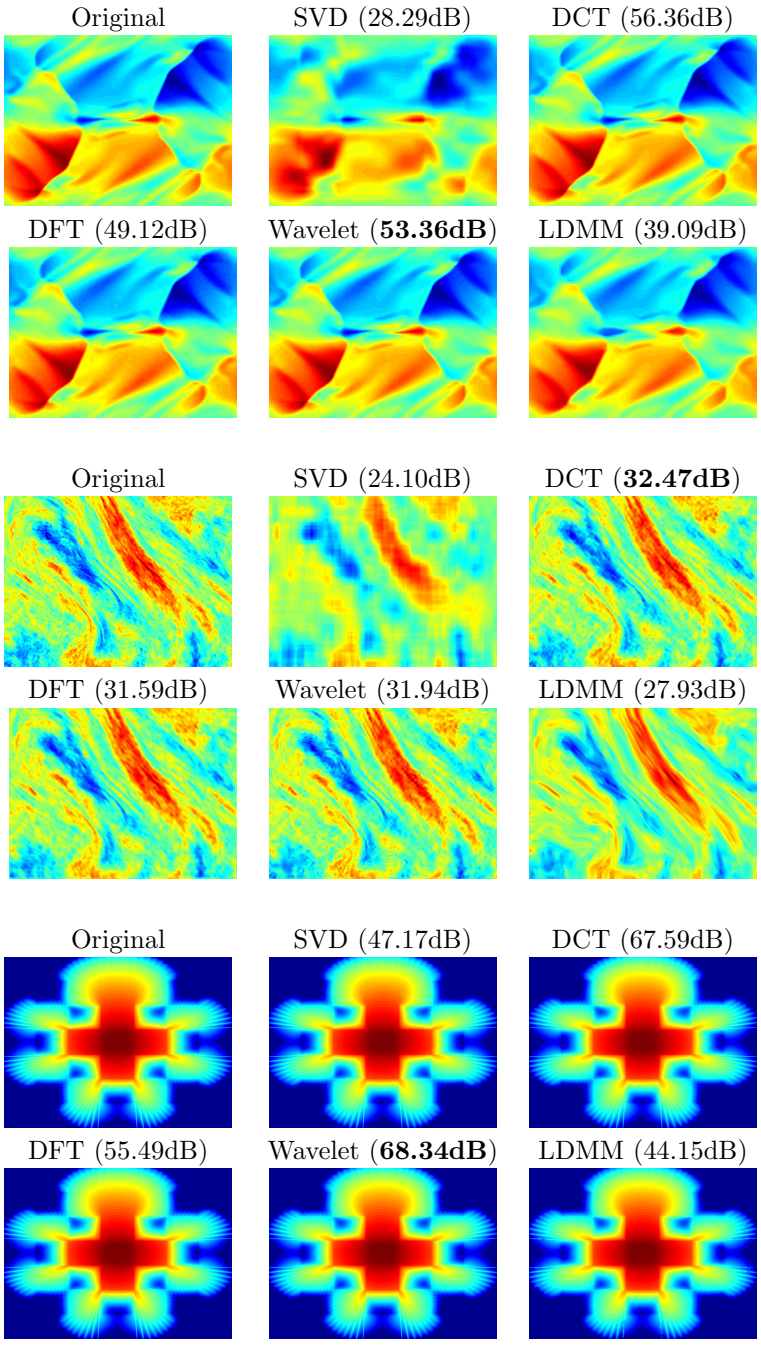


Figure 22: Compression of 2D scientific data sets with a 5% data compression rate. The original data are shown on the upper left corners for each data set. The results of SVD, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

10%	SVD	DCT	DFT	Wavelet	LDMM
L_1	0.0152	**0.0003**	0.0010	0.0005	0.0034
L_2	0.0208	**0.0005**	0.0014	0.0007	0.0075
L_∞	0.1357	**0.0056**	0.0132	0.0067	0.1376
PSNR	33.65	**66.87**	57.03	63.01	42.55
5%	SVD	DCT	DFT	Wavelet	LDMM
L_1	0.0295	**0.0011**	0.0024	0.0016	0.0056
L_2	0.0385	**0.0015**	0.0035	0.0021	0.0111
L_∞	0.1964	0.0154	0.0314	**0.0149**	0.1102
PSNR	28.29	**56.36**	49.12	53.36	39.09

Table 14: Errors of the compression of the 2D vortex data set.

10%	SVD	DCT	DFT	Wavelet	LDMM
L_1	0.0345	**0.0131**	0.0147	0.0145	0.0243
L_2	0.0437	**0.0165**	0.0190	0.0182	0.0333
L_∞	0.2597	**0.0844**	0.1499	0.0861	0.1882
PSNR	27.19	**35.63**	34.44	34.78	29.56
5%	SVD	DCT	DFT	Wavelet	LDMM
L_1	0.0494	**0.0189**	0.0206	0.0202	0.0303
L_2	0.0624	**0.0238**	0.0263	0.0253	0.0401
L_∞	0.2794	0.1121	0.1920	**0.1057**	0.2063
PSNR	24.10	**32.47**	31.59	31.94	27.93

Table 15: Errors of the compression of the 2D plasma (distribution) data set.

10%	SVD	DCT	FFT	Wavelet	LDMM
L_1	0.0009	**0.0001**	0.0004	0.0006	0.0013
L_2	0.0014	0.0002	0.0008	**0.0001**	0.0040
L_∞	0.0186	0.0101	0.0603	**0.0011**	0.1979
PSNR	57.24	75.73	61.94	**80.33**	47.98
5%	SVD	DCT	DFT	Wavelet	LDMM
L_1	0.0029	0.0003	0.0010	**0.0002**	0.0022
L_2	0.0044	0.0004	0.0017	**0.0004**	0.0062
L_∞	0.0539	0.0244	0.0743	**0.0049**	0.2097
PSNR	47.17	67.59	55.49	**68.34**	44.15

Table 16: Errors of the compression of the 2D lattice data set.

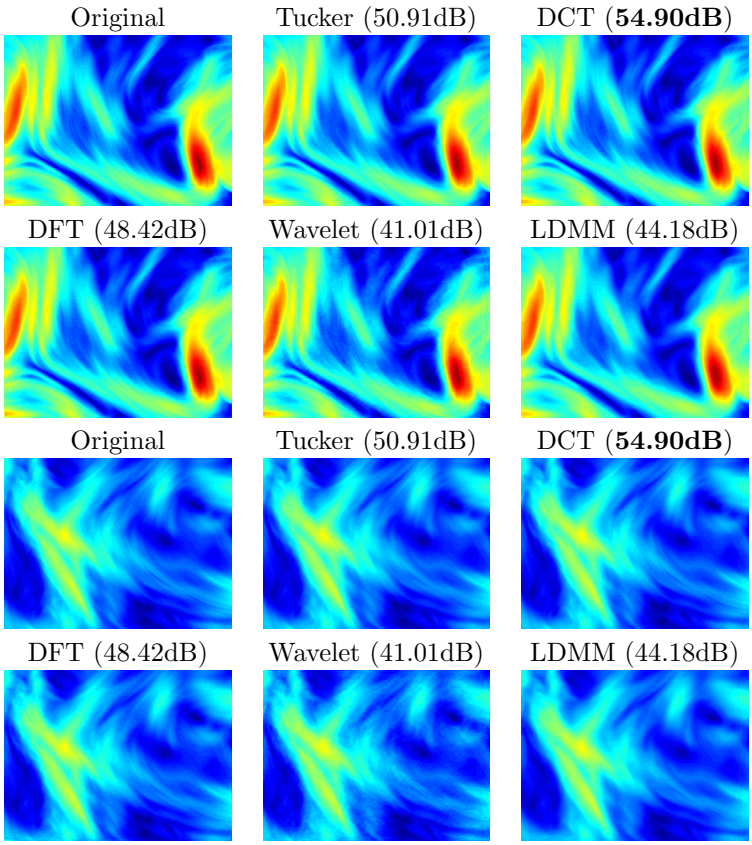


Figure 23: Compression of the 3D plasma (magnetic field) data set with a 10% data compression rate. Two spatial cross sections of the original data set are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

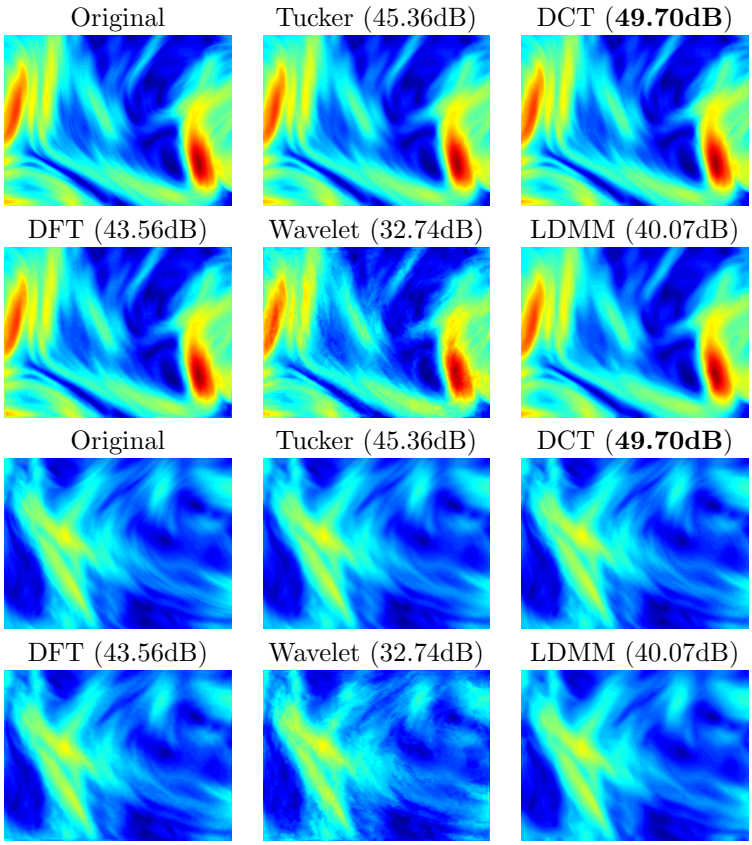


Figure 24: Compression of the 3D plasma (magnetic field) data set with a 5% data compression rate. Two spatial cross sections of the original data set are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

10%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	0.0021	**0.0014**	0.0024	0.0068	0.0038
L_2	0.0028	**0.0018**	0.0038	0.0089	0.0062
L_∞	0.0613	**0.0433**	0.1757	0.0739	0.1330
PSNR	50.91	**54.90**	48.42	41.01	44.18
5%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	0.0040	**0.0025**	0.0043	0.0183	0.0062
L_2	0.0054	**0.0033**	0.0066	0.0231	0.0099
L_∞	0.0911	**0.0698**	0.2141	0.1558	0.2012
PSNR	45.36	**49.70**	43.56	32.74	40.07

Table 17: Errors of the compression of the 3D plasma (magnetic field) data set.

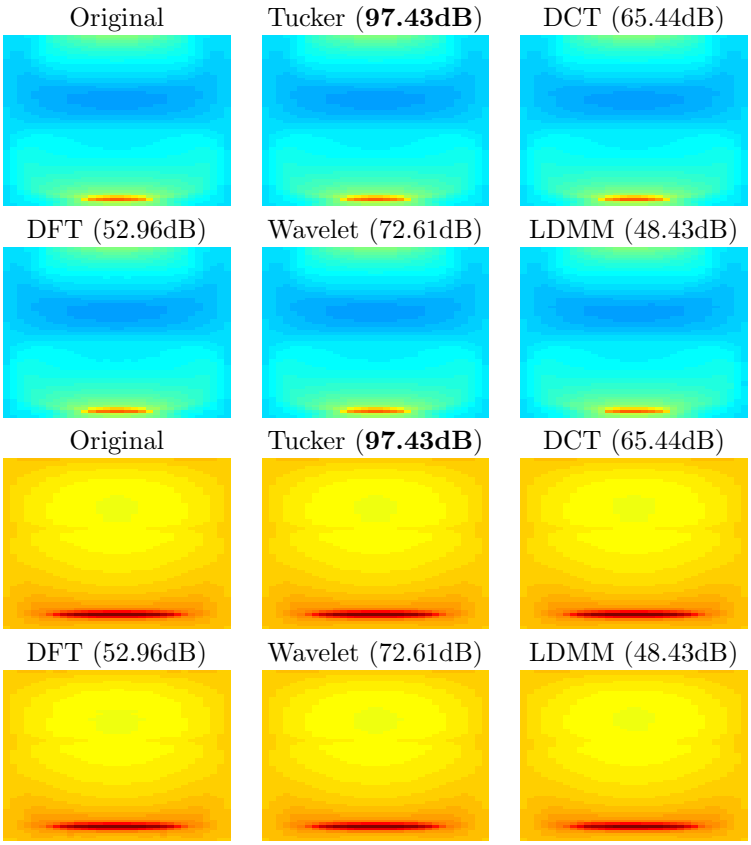


Figure 25: Compression of the 3D lattice data set with a 10% data compression rate. The original angular flux at $x = 0.24$ and $x = 1.18$ are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

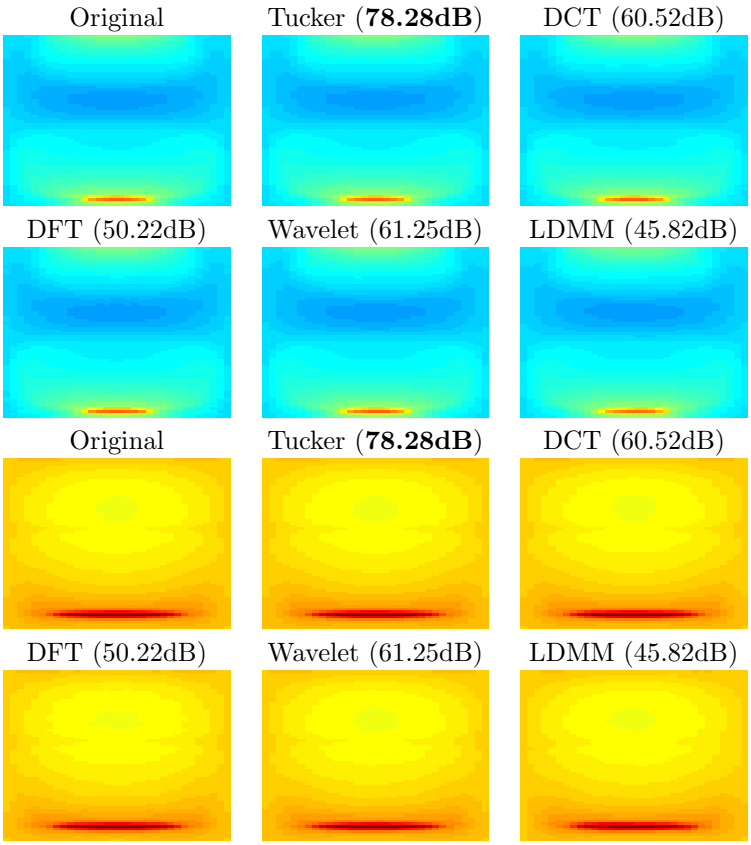


Figure 26: Compression of the 3D lattice data set with a 5% data compression rate. The original angular flux at $x = 0.24$ and $x = 1.18$ are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

10%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	$\mathbf{9 \times 10^{-6}}$	0.0002	0.0007	0.0002	0.0008
L_2	$\mathbf{1 \times 10^{-5}}$	0.0005	0.0022	0.0002	0.0038
L_∞	**0.0002**	0.1338	0.2843	0.0020	0.4262
PSNR	**97.43**	65.44	52.96	72.61	48.43
5%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	**0.0001**	0.0004	0.0010	0.0006	0.0013
L_2	**0.0001**	0.0009	0.0031	0.0008	0.0051
L_∞	**0.0042**	0.2053	0.4266	0.0095	0.4530
PSNR	**78.28**	60.52	50.22	61.25	45.82

Table 18: Errors of the compression of the 3D lattice data set.

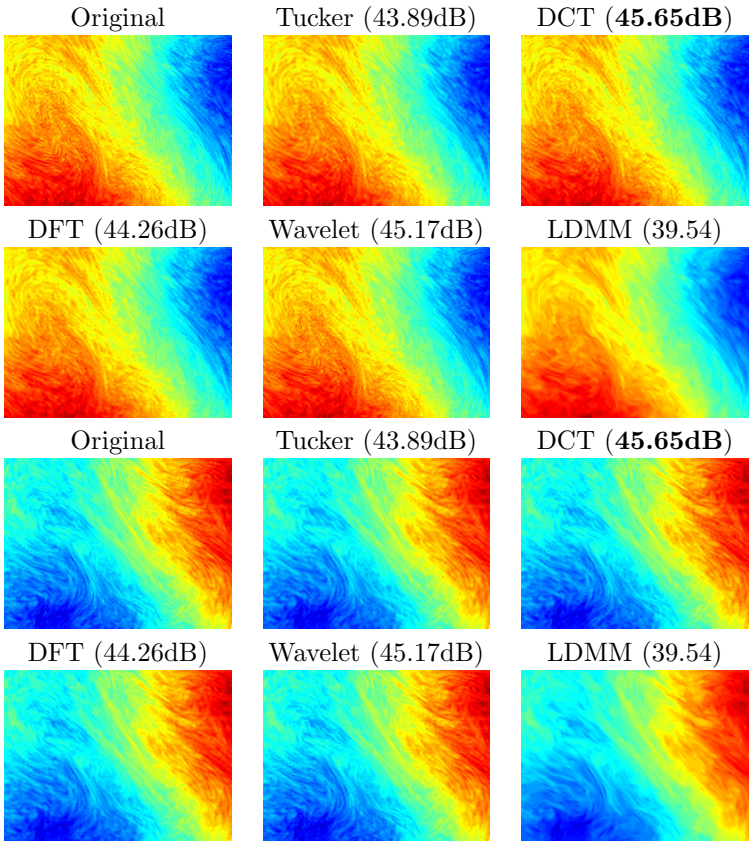


Figure 27: Compression of the 3D plasma (distribution function) data set with a 10% data compression rate. Two spatial cross sections of the original data set are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

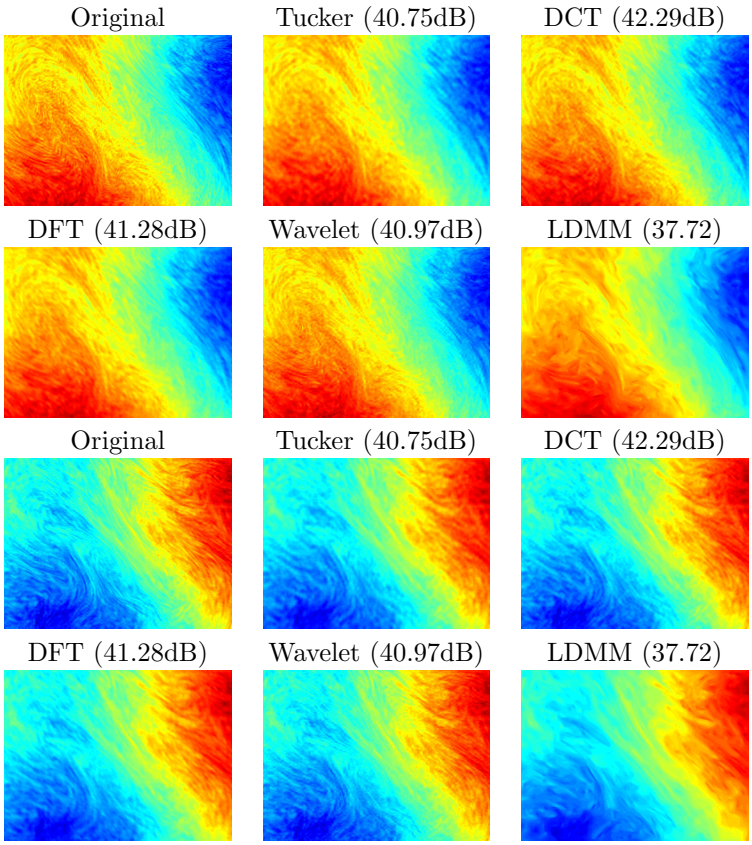


Figure 28: Compression of the 3D plasma (distribution function) data set with a 5% data compression rate. Two spatial cross sections of the original data set are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

10%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	0.0042	**0.0039**	0.0045	0.0042	0.0060
L_2	0.0064	**0.0052**	0.0061	0.0055	0.0105
L_∞	0.0637	0.0644	0.0837	**0.0373**	0.1181
PSNR	43.89	**45.65**	44.26	45.17	39.54
5%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	0.0060	**0.0057**	0.0063	0.0067	0.0075
L_2	0.0092	**0.0077**	0.0086	0.0089	0.0130
L_∞	0.0890	0.0766	0.1018	**0.0660**	0.1793
PSNR	40.75	**42.29**	41.28	40.97	37.72

Table 19: Errors of the compression of the 3D plasma (distribution function) data set.

5. Conclusion

In this paper, we propose a low dimensional manifold model for scientific data reconstruction from regular or irregular samplings. The low dimensionality of the patch manifold is used as a regularizer, and this assumption is justified through a dimension analysis of common patterns in various scientific data sets. The variational problem is solved via alternating direction of minimization, and the corresponding Laplace-Beltrami equation is discretized by weighted graph Laplacian. The proposed algorithm consistently outperforms all the competing algorithms in both regular and irregular sampling cases. The current LDMM algorithm as a data compression method does not perform as well as other standard compression algorithms that assume access to the full data set. But LDMM as a data compression method is easy to implement in the compression step, and it is also faster in the reconstruction step if only a subset of the original data set is required. Modifying LDMM for it to achieve its full potential as a data compression method will be the focus of our future work.

6. Acknowledgment

The authors would like to thank Michael Crockatt and Professor Antonio Marquina for providing the neutron transport data set and the Orszag-Tang vortex data set.

References

[1] M. Naghizadeh, M. D. Sacchi, Beyond alias hierarchical scale curvelet interpolation of regularly and irregularly sampled seismic data, GEOPHYSICS 75 (6) (2010) WB189–WB202.

[2] J. Ronen, Wave-equation trace interpolation, GEOPHYSICS 52 (7) (1987) 973–984.

[3] C. Bagaini, U. Spagnolini, 2-d continuation operators and their applications, GEOPHYSICS 61 (6) (1996) 1846–1858.

[4] R. H. Stolt, Seismic data mapping and reconstruction, GEOPHYSICS 67 (3) (2002) 890–908.

[5] S. Fomel, Seismic reflection data interpolation with differential offset and shot continuation, GEOPHYSICS 68 (2) (2003) 733–744.

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

[7] T. F. Chan, J. Shen, Nontexture inpainting by curvature-driven diffusions, Journal of Visual Communication and Image Representation 12 (4) (2001) 436 – 449.

[8] S. Mallat, A wavelet tour of signal processing: the sparse way, Academic press, 2008.

[9] R. H. Chan, Y. W. Wen, A. M. Yip, A fast optimization transfer algorithm for image inpainting in wavelet domains, IEEE Transactions on Image Processing 18 (7) (2009) 1467–1476.

[10] E. J. Candes, D. L. Donoho, Curvelets: A surprisingly effective nonadaptive representation for objects with edges, Tech. rep., DTIC Document (2000).

[11] M. Elad, J.-L. Starck, P. Querre, D. Donoho, Simultaneous cartoon and texture image inpainting using morphological component analysis (mca), Applied and Computational Harmonic Analysis 19 (3) (2005) 340 – 358.

[12] M. Fadili, J.-L. Starck, F. Murtagh, Inpainting and zooming using sparse representations, The Computer Journal 52 (1) (2009) 64.

[13] A. Buades, B. Coll, J. M. Morel, A review of image denoising algorithms, with a new one, Multiscale Modeling & Simulation 4 (2) (2005) 490–530.

[14] G. Gilboa, S. Osher, Nonlocal operators with applications to image processing, Multiscale Modeling & Simulation 7 (3) (2009) 1005–1028.

[15] G. Peyré, S. Bougleux, L. Cohen, Non-local Regularization of Inverse Problems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 57–68.

[16] G. Facciolo, P. Arias, V. Caselles, G. Sapiro, Exemplar-Based Interpolation of Sparsely Sampled Images, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 331–344.

[17] J. Mairal, M. Elad, G. Sapiro, Sparse representation for color image restoration, IEEE Transactions on Image Processing 17 (1) (2008) 53–69.

[18] G. Yu, G. Sapiro, S. Mallat, Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity, IEEE Transactions on Image Processing 21 (5) (2012) 2481–2499.

[19] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, L. Carin, Nonparametric bayesian dictionary learning for analysis of noisy and incomplete images, IEEE Transactions on Image Processing 21 (1) (2012) 130–144.

[20] S. Osher, Z. Shi, W. Zhu, Low dimensional manifold model for image processing, Tech. rep., Technical Report, CAM report 16-04, UCLA (2016).

[21] Z. Li, Z. Shi, J. Sun, Point integral method for solving poisson-type equations on manifolds from point clouds with convergence guarantees, arXiv preprint arXiv:1409.2623.

[22] Z. Shi, S. Osher, W. Zhu, Weighted nonlocal laplacian on interpolation from sparse data, Journal of Scientific Computing (2017) 1–14.

[23] Z. Shi, S. Osher, W. Zhu, Low dimensional manifold model with semi-local patches, Tech. rep., Technical Report, CAM report 16-63, UCLA (2016).

[24] J. Lee, Introduction to smooth manifolds, Vol. 218, Springer Science & Business Media, 2012.

[25] X. Zhu, Z. Ghahramani, J. D. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proceedings of the 20th International conference on Machine learning (ICML-03), 2003, pp. 912–919.

[26] F. R. Chung, Spectral graph theory, Vol. 92, American Mathematical Soc., 1997.

[27] T. Bühler, M. Hein, Spectral clustering based on the graph p-laplacian, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, ACM, New York, NY, USA, 2009, pp. 81–88.

[28] A. L. Bertozzi, A. Flenner, Diffuse interface models on graphs for classification of high dimensional data, Multiscale Modeling & Simulation 10 (3) (2012) 1090–1118.

[29] J. H. Friedman, J. L. Bentley, R. A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Trans. Math. Softw. 3 (3) (1977) 209–226.

[30] M. Muja, D. G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration., VISAPP (1) 2 (331-340) (2009) 2.

[31] W. Zhu, V. Chayes, A. Tiard, S. Sanchez, D. Dahlberg, A. L. Bertozzi, S. Osher, D. Zosso, D. Kuang, Unsupervised classification in hyperspectral imagery with nonlocal total variation and primal-dual hybrid gradient algorithm, IEEE Transactions on Geoscience and Remote Sensing 55 (5) (2017) 2786–2798.

[32] A. Vedaldi, B. Fulkerson, VLFeat: An open and portable library of computer vision algorithms (2008).

[33] B. W. Bader, T. G. Kolda, et al., Matlab tensor toolbox version 2.6, Available online (February 2015).

[34] B. W. Bader, T. G. Kolda, Algorithm 862: MATLAB tensor classes for fast algorithm prototyping, ACM Transactions on Mathematical Software 32 (4) (2006) 635–653.

[35] D. Told, F. Jenko, J. M. TenBarge, G. G. Howes, G. W. Hammett, Multiscale nature of the dissipation range in gyrokinetic simulations of alfvénic turbulence, Phys. Rev. Lett. 115 (2015) 025003.

[36] F. Jenko, W. Dorland, M. Kotschenreuther, B. Rogers, Electron temperature gradient driven turbulence, Physics of Plasmas 7 (5) (2000) 1904–1910.

[37] T. A. Brunner, Forms of approximate radiation transport, Tech. Rep. SAND2002-1778, Sandia National Laboratories (2002).

[38] T. A. Brunner, J. P. Holloway, Two-dimensional time dependent Riemann solvers for neutron transport, Journal of Computational Physics 210 (2005) 386–399.

[39] C. D. Hauck, R. G. McClarren, A collision-based hybrid method for time-dependent, linear, kinetic transport equations, Multiscale Modeling & Simulation 11 (4) (2013) 1197–1227.

[40] R. G. McClarren, C. D. Hauck, Robust and accurate filtered spherical harmonics expansions for radiative transfer, Journal of Computational Physics 229 (2010) 5597–5614.

[41] R. G. McClarren, C. D. Hauck, Simulating radiative transfer with filtered spherical harmonics, Physics Letters A 374 (2010) 2290–2296.

[42] M. Schaefer, M. Frank, C. D. Levermore, Diffusive corrections to P_N approximations, Multiscale Model. Simul. 9 (2009) 1–28.

[43] M. M. Crockatt, A. J. Christlieb, C. D. Hauck, C. K. Garrett, An arbitrary-order, fully implicit, hybrid kinetic solver for linear radiative transport using integral deferred correction, Journal of Computational Physics.

[44] S. A. Orszag, C.-M. Tang, Small-scale structure of two-dimensional magnetohydrodynamic turbulence, Journal of Fluid Mechanics 90 (01) (1979) 129–143.

[45] M. J. Castro, J. M. Gallardo, A. Marquina, Approximate oshersolomon schemes for hyperbolic systems, Applied Mathematics and Computation 272, Part 2 (2016) 347 – 368, recent Advances in Numerical Methods for Hyperbolic Partial Differential Equations.