Structured tensor missing-trace interpolation in the Hierarchical Tucker format

Curt Da Silva and Felix J. Herrmann, University of British Columbia*

SUMMARY

Owing to the large scale and dimensionality of a 3D seismic experiment, acquiring fully-sampled data according to the Nyquist criterion is an exceedingly arduous and cost-prohibitive task. In this paper, we develop tools to interpolate 5D seismic volumes with randomly missing sources or receivers using a relatively novel tensor format known as the *Hierarchical Tucker* (HT) format. By exploiting the underlying smooth structure of HT tensors, specifically its *smooth manifold* structure, we develop solvers which are fast, immediately parallelizable, and SVD-free, making these solvers amenable to large-scale problems where SVD-based projection methods are far too costly. We also build on intuition of multidimensional sampling from the perspective of matrix-completion and demonstrate the ability of our algorithms to recover frequency slices even amidst very high levels of source subsampling on a synthetic large-scale 3D North Sea dataset.

INTRODUCTION

Acquiring seismic data from three-dimensional seismic experiments is a time- and cost-intensive process, which results in an enormous five-dimensional data volume (two source coordinates, two receiver coordinates, and time). Marine surveys involve managing a number of boats, manned by expensive crews, towing an expensive and difficult to maneuver array of air guns and receiver lines for many months, all in an attempt to acquire a finely-sampled data volume which satisfies the Nyquist criterion and avoids aliasing.

The idealized data volume that we are trying to collect, however, exhibits a large amount of structure that is subsequently destroyed by randomized subsampling. By developing algorithms which restore this structure and applying them to subsampled data, we can interpolate a data volume with randomly missing source or receiver positions in an efficient manner. Seismic frequency slices tend to exhibit *low-rank* structure, especially at lower frequencies, which makes them particularly amenable to recovery from rank-increasing measurements such as random subsampling (see, e.g. Candès and Recht (2009)). In the case of a 5D data volume, each fully-sampled frequency slice (at low frequencies) can be well approximated by a tensor in the *Hierarchical Tucker* format, which will be detailed in the next section. Our experiments model a marine seismic experiment using ocean bottom nodes with a source vessel. In our case, randomly subsampling receivers corresponds to placing far *fewer* nodes in *random* locations compared to standard equidistantly-spaced fully-sampled grid. While we work with missing sources in this paper, by the reciprocity underlying the data volume, we can just as easily work with subsampled receivers.

Here we build on largely theoretical considerations in Uschmajew and Vandereycken (2012) to develop an algorithmic framework for solving optimization problems in the Hierarchical Tucker format and apply these algorithms to interpolate seismic frequency slices with randomly missing sources.

Previous work in tensor completion of seismic data includes Kreimer and Sacchi (2012), which uses the Tucker tensor format and a projection onto convex sets (POCS) approach, which has to compute SVDs of the underlying tensor at each iteration, which are expensive for large data volumes, and keeps several copies of the full data set in working memory. A variation on their approach works on small windows of the data to overcome this, but unfortunately limits the distance between subsampled points in the process. The authors in Gao et al. (2011) use a Toeplitz matrix as a skeleton for their interpolant and exploits the relationship between FFTs and Toeplitz matrices to perform matrix-vector products efficiently, again using a POCS approach. This technique may suffer greatly in terms of reconstruction error when the subsampling rate is moderately high. These drawbacks are not present in our current work and our method is much more general than interpolating Toeplitz structure, as it does not rely on a skeleton for the matrix entries.

HIERARCHICAL TUCKER TENSOR FORMAT

Before we can describe Hierarchical Tucker tensors themselves, we give two preliminary non-standard definitions.

Definition 1. The *matricization* of a $d-$dimensional tensor $\mathbf{X}$ of size $n_1 \times n_2 \times \ldots n_d$ along the indices $t \subset \{1,\ldots,d\}$ is the unique matrix $X^{(t)}$ such that the rows of $X^{(t)}$ contain the vectorized dimensions $t_1, t_2, \ldots, t_k$ of $\mathbf{X}$ and the columns of $X^{(t)}$ contain the vectorized dimensions of $\mathbf{X}$ specified by t^c.

Definition 2. A *dimension tree* for a $d-$dimensional tensor is a nontrivial binary tree such that

- The root node, $\mathrm{t}_{\mathrm{root}}$, has the label $\{1,\ldots,d\}$
- The labels for the children of each non-leaf node form a partition of the parent's label, i.e.

$$t_l \cup t_r = t, t_l \cap t_r = \emptyset, \quad t \notin L$$

  where t_l, t_r are the left and right children of the node t, respectively, and L is the set of all leaves of T.

A dimension tree determines which grouping of dimensions will be separated from each other (in the sense of the SVD) in the HT construction. The proper choice of a dimension tree is critical for ensuring that one's data exhibits low-rank behaviour in the chosen dimension groupings, and we will discuss the implications for our choice of dimension tree later.

Definition 3. Let $\mathbb{R}_*^{n\times p}$ and $\mathbb{R}_*^{p\times q\times r}$ denote the set of all $n \times p$ matrices of full rank and $p \times q \times r$ 3-tensors of full *multilinear* rank (that is, $A^{(t)}$ is full-rank for $t = 1,2,3$), respectively. A $d-$tensor $\mathbf{X}$ is said to be in *Hierarchical Tucker* format with associated dimension tree T and hierarchical ranks $(k_t)_{t\in T}$ if there exist parameter matrices/tensors $x = (U_t, B_t)$ with $U_t \in$

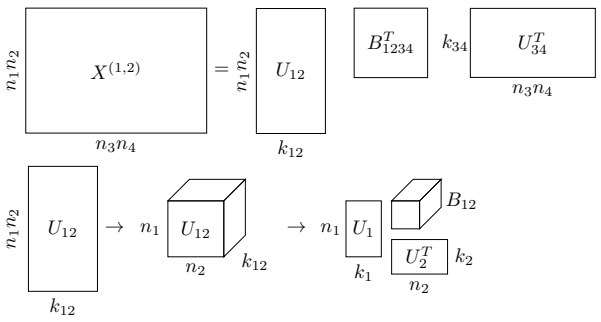


Figure 1: Visualization of the HT format for a four dimensional tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$

$\mathbb{R}_*^{n_t \times k_t}$, $B_t \in \mathbb{R}_*^{k_r \times k_l \times k_t}$ such that $\phi(x) = \mathbf{X}$, where

$$\begin{aligned} \operatorname{vec} \phi(x) &= (U_{t_r} \otimes U_{t_l})(B_t^{(k_l,k_r)}) \qquad t = \mathrm{t}_{\mathrm{root}} \\ U_t &= (U_{t_r} \otimes U_{t_l})(B_t^{(k_l,k_r)}) \qquad t \notin L \cup \mathrm{t}_{\mathrm{root}} \end{aligned} \tag{1}$$

where $C \otimes D$ is the Kronecker product of matrices C and D, k_t is the rank associated to node t, $k_{\mathrm{t_{root}}} = 1$, and k_l, k_r are the ranks associated to nodes t_l, t_r, respectively.

Our definition is somewhat modified from the one given in Uschmajew and Vandereycken (2012), so that $\operatorname{rank}(X^{(t)}) = k_t$ for each $t \in T$. Note that the intermediate matrices U_t for $t \in T \setminus (L \cup \mathrm{t}_{\mathrm{root}})$ do *not* need to be stored– a HT tensor X is completely determined by the matrices at the leaves, $U_t, t \in L$ and the so-called *transfer tensors* $B_t, t \in T \setminus L$. Due to this observation, the number of parameters needed to represent a $d-$dimensional HT tensor is at most $m = dNK + (d-2)K^3 + K^2$, where K is the maximum rank and N is the maximum dimension size. When $d > 3$ and K is small, $m \ll N^d$, the usual storage requirement for a $d-$tensor. E.g. when $\mathbf{X}$ is a $4D$ tensor with size 100 in each dimension, and K is 50, then $m \approx 2.73 \cdot 10^6$ whereas $N^d = 10^8$, a compression of 99.7%.

A visualization of the HT format for a particular choice of dimension tree and a 4-dimensional tensor is shown in Figure 1. The main insight of the HT format is that the "left singular vectors" U_{12} contain dimensions n_1 and n_2 vectorized. By reshaping this matrix in to a 3-dimensional tensor, we can split apart dimensions n_1 and n_2 in the same Kronecker fashion.

If we let M denote the set of all permissible parameters $x = (U_t, B_t)$, then the mapping $\phi : M \to \mathbb{R}^{n_1 \times n_2 \times \ldots n_d}$ is a smooth mapping whose image $\phi(M)$ is the set of all HT tensors of fixed rank $(k_t)_{t \in T}$. Thus if we assume that our fully-sampled tensor is well-approximated in the HT format, then the optimization problem we are interested in solving is

$$\min_{x=(U_t,B_t) \in M} \|A\phi(x) - b\|_2^2 \tag{2}$$

where A is our subsampling operator and b is our subsampled data. In the seismic case, A would randomly remove source positions from the data volume and b would be the full data volume with the same source positions removed. Unfortunately from an optimization point of view, the function ϕ is not injective, so that different parameters x and y can map to the same tensor $\phi(x)$, leading (somewhat artificially) to spurious local minima. The specifics are not of interest here and we address how we deal with this non-uniqueness in the next section.

OPTIMIZATION ALGORITHMS

In order to solve optimization problems in the HT format, we must first be able to compute derivatives of a general objective function evaluated at $\phi(x)$. Differentiating (1) gives that, at node $t \in T$,

$$\delta U_t = \frac{\partial U_t}{\partial U_{t_l}} \delta U_{t_l} + \frac{\partial U_t}{\partial U_{t_r}} \delta U_{t_r} + \frac{\partial U_t}{\partial B_t} \delta B_t$$

and the Jacobian, $D\phi(x)[\delta U_t, \delta B_t] = \delta U_{\mathrm{t_{root}}}$, can be evaluated using the above recursion. Our interest is not in the Jacobian, however, but in its adjoint, $D\phi(x)^*$, and we can obtain an algorithm for applying this operator by taking the adjoint of the recursion above. Each of the partial derivatives can be derived using identities from matrix calculus (omitted here), and, after extensive simplification, result in a short sequence of MATLAB commands, using code from the SPOT framework (van den Berg and Friedlander, 2009) and from the hTucker toolbox (Kressner and Tobler, 2012), requiring only matrix-matrix multiplications and permutations. These operations can be performed efficiently and in a distributed computational environment for large data sets.

To account for the non-uniquness mentioned earlier, we need to perform projections on to the orthogonal complement of each U_t to compute $D\phi(x)^* Z$. In order for these projections to be computed efficiently, we need to restrict our parameters $x = (U_t, B_t)$ to be in *Orthogonal Hierarchical Tucker* format, i.e., so that

$$U_t^T U_t = I_{k_t}, (B_t^{(k_l,k_r)})^T B_t^{(k_l,k_r)} = I_{k_t} \quad t \in T \setminus \mathrm{t}_{\mathrm{root}}$$

This can be done efficiently through means of *reorthogonalization* – specifically, if we are at the point $x = (U_t, B_t)$ and want to move in the direction $Z = (\delta U_t, \delta B_t)$ (for instance, the negative gradient direction), then we ensure that $x + Z$ is orthogonalized via the following mapping R

$$R(x+Z) = \begin{cases} \mathrm{qf}(U_t + \delta U_t) & \text{if } t \in L \\ \mathrm{qf}((R_{t_r} \otimes R_{t_l})(B_t + \delta B_t)) & \text{if } t \notin \mathrm{t}_{\mathrm{root}} \cup L \\ (R_{t_r} \otimes R_{t_l})(B_t + \delta B_t) & \text{if } t = \mathrm{t}_{\mathrm{root}} \end{cases}$$

where $\mathrm{qf}(Y_t)$ and R_t are the $Q-$ and $R-$factors from the QR decomposition of Y_t, respectively. The parameters $R(x+Z)$ are computed very efficiently, as the mapping R only involves QR-factorizations on small matrices and tensors. With these two components in hand, the computation of $D\phi(x)^* Z$ and the reorthogonalization mapping R, the Steepest Descent algorithm with an Armijo line search follows immediately in Figure 2.

This algorithmic framework allows us to implement both Steepest Descent and Conjugate Gradient, (although we do not delve in to the latter details here), which exhibit linear convergence and can require a large number of iterations to match the data sufficiently well. Since our objective function is the least-squares mismatch, we can exploit this structure to develop

Require: Initial guess $x_0 = (U_t, B_t)$, $0 < c < 1$ sufficient decrease parameter, $0 < \theta < 1$ step size decrease
  for $k = 0, 1, 2, \ldots$ until convergence **do**
    $\mathbf{X}_k \leftarrow \phi(x_k)$
    $f_k \leftarrow f(\mathbf{X}_k)$
    $g_k \leftarrow D\phi(x)^* \nabla_x f(\mathbf{X}_k)$ //Gradient of f at x_k
    $\alpha \leftarrow 1$ //Armijo line search
    while $f(\phi(R_{x_k}(-\alpha g_k))) - f_k > -c\alpha\langle g_k, g_k\rangle$ **do**
      $\alpha \leftarrow \alpha \cdot \theta$
    end while
    $x_{k+1} \leftarrow R_{x_k}(-\alpha g_k)$
  end for

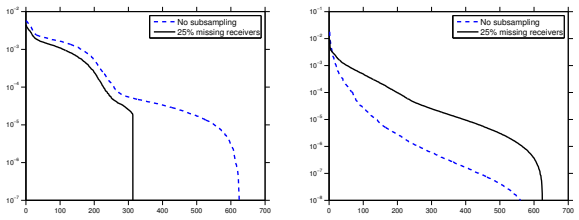
Figure 2: Steepest descent for optimizing a function f over the set of HT tensors

an efficient *Gauss-Newton* method for minimizing (2), which tends to exhibit *superlinear* convergence and fits the data much faster, especially at the early iterations. We note that the Jacobian mapping $D\phi(x)$ maps "small" parameters $(\delta U_t, \delta B_t)$ to "big" parameters of size $n_1 n_2 \ldots n_d$ (the size of the fully-sampled data-space) and the adjoint does the opposite. The Gauss-Newton Hessian, being the composition of the adjoint Jacobian and the forward Jacobian, is an operator which maps "small" parameters to "small" parameters. Given this fact, we can compute the mixed partial derivatives

$$\frac{\partial U_{\mathrm{t_{root}}}}{\partial C}^* \frac{\partial U_{\mathrm{t_{root}}}}{\partial D}$$

explicitly, where $C, D \in \{U_{t_l}, U_{t_r}, B_{\mathrm{t_{root}}}\}$, again using identities from matrix calculus. This allows us to dramatically speed up applications of the GN Hessian $H_{GN} = D\phi(x)^* D\phi(x)$ to arbitrary vectors since we never form intermediate vectors of the size of the full data and thus avoid incurring the computational and memory-based costs associated to such vectors. This fact enables us to use a Conjugate Gradient method to efficiently solve the Gauss-Newton Hessian equation $H_{GN}\delta x = -g$ and dramatically speed up the overall interpolation.

MULTIDIMENSIONAL SUBSAMPLING



(a) (b)

Figure 3: Singular values (y-axis) under the a) (src x, src y) matricization and b) (src x, rec x) matricization for full (black) data and subsampled (blue) data.

Owing to the symmetric nature between sources and receivers, we have essentially two choices of underlying dimension tree, both of which are shown in Figure 3. Namely, we can choose between placing the (src x, src y) dimensions in the rows and (rec x, rec y) dimensions in the columns, or placing the (src x, rec x) dimensions in the rows and (src y, rec y) dimensions in the columns (each choice specifies the rest of the dimension tree). In the case when we are, say, randomly missing receivers, the former organization of data has the effect that the data itself exhibits slowly-decaying singular values and subsampling will tend to remove rows of this matrix. This causes the singular values to not increase and in fact *are set to zero* at the low end (the worst-case scenario for the purposes of rank-minimizing recovery, e.g. see Candès and Recht (2009)). On the other hand, the latter organization of data results in a matrix which has *quickly* decaying singular values (i.e. more compressibility) and a subsampling operator that randomly removes blocks from the underlying matrix, destroying its compressible structure. The latter organization is much more ideal from a matrix-completion perspective, for the same reasons as above. The same discussion holds for matricizations in the singleton dimensions, adding further constraints (and hence regularity) to the computed solution compared to standard matrix completion (which only uses one of the possible matricizations). This insight is supported in Demanet (2006), wherein the author remarks that the solution operator to a general wave equation, when represented in terms of Wave Atoms, exhibits a similar low-rank behaviour when permuting the appropriate indices. This observation makes sense in this context, since our data volume is a sampled Green's function of the wave equation restricted to the surface of the acquisition geometry.

NUMERICAL EXPERIMENTS

We use the aforementioned techniques to interpolate a synthetic data set provided to us by BG. The data set consists of 68 x 68 sources, each having 401 x 401 receivers and is generated from an unknown model. We extract a frequency slice at 7.34 Hz from this data, randomly remove 75% of the 4624 sources from this slice, and recover each slice using the aforementioned GN method for 100 iterations. Owing to the low-frequency content of this slice, the data is first subsampled to 101 x 101 receivers and the data volume is recovered to a 68 x 68 x 101 x 101 tensor. We display several common receiver gathers in Figure 4 and common shot gathers in Figure 5. Each shot gather in the latter figure is produced from the interpolated volume by Fourier interpolating each slice to 401 x 401 receivers. Despite the high amount of missing sources, we are still able to leverage the multidimensional dependencies of the data with the HT format and interpolate these low-frequency slices successfully.

CONCLUSION

In this work, we have extended the largely theoretical results of Uschmajew and Vandereycken (2012) to a practical algorithmic framework for solving optimization problems whose solutions are well approximated in the Hierarchical Tucker format. Our methods allow us to interpolate tensors exhibiting this hierarchical low-rank structure from a subset of their entries. There is an open question as to how one can formulate precise recovery results for this problem to the comprehensive level of the recovery results present in the matrix-completion literature, a question that we leave for future research.

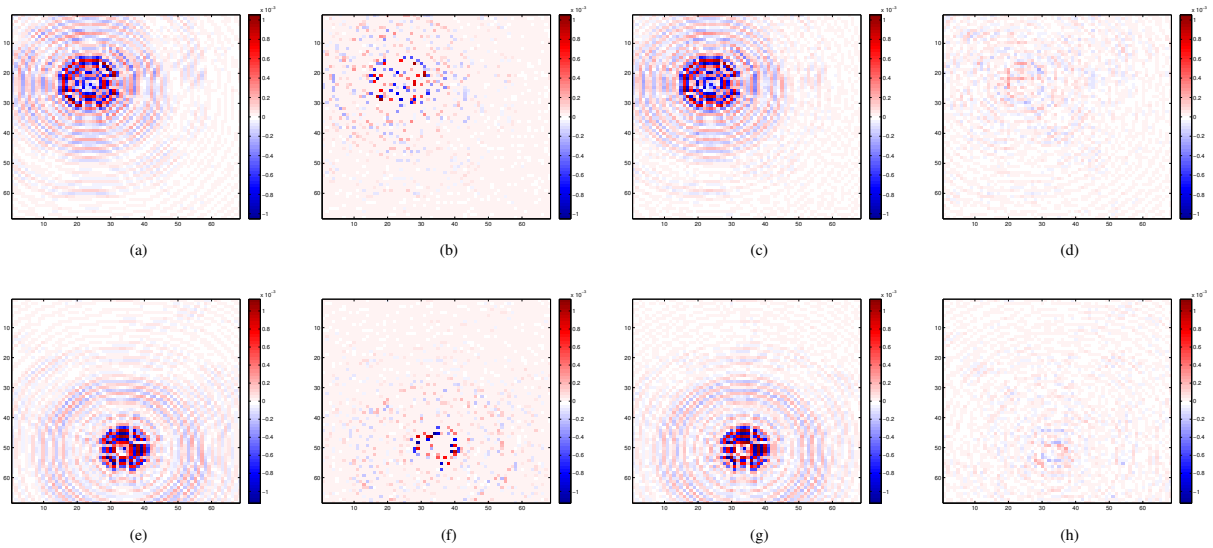


Figure 4: Common receiver gathers at 7.34 Hz a, e) True data, b, f) 75% missing sources, Interpolated data – SNR : c) 12.5 dB, g) 13.3 dB. d, h) Difference

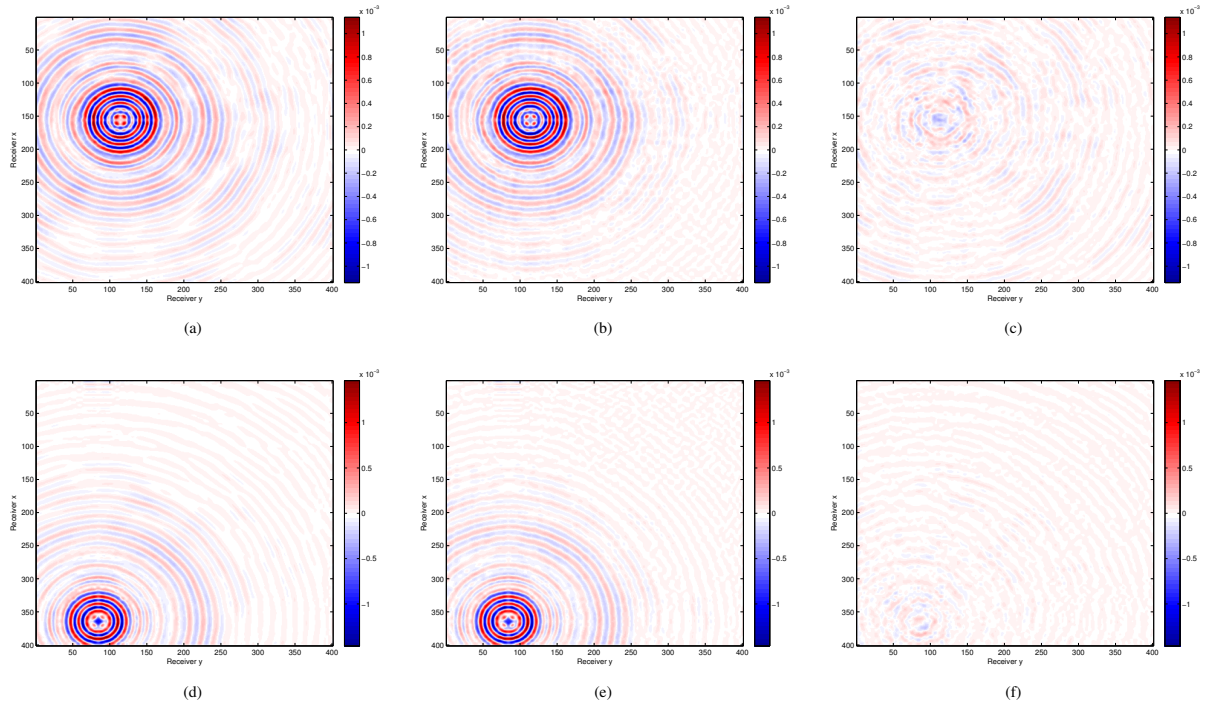


Figure 5: Common source gathers - 75% missing sources at 7.34Hz a, d) True data, Interpolated data – SNR : b) 11.5 dB, e) 15.4 dB. c, f) Difference

REFERENCES

Candès, E., and B. Recht, 2009, Exact matrix completion via convex optimization: Foundations of Computational mathematics, **9**, 717–772.

Demanet, L., 2006, Curvelets, wave atoms, and wave equations: PhD thesis, California Institute of Technology.

Gao, J., M. Sacchi, and X. Chen, 2011, A fast rank reduction method for the reconstruction of 5d seismic volumes: Presented at the 2011 SEG Annual Meeting.

Kreimer, N., and M. Sacchi, 2012, A tensor higher-order singular value decomposition for prestack seismic data noise reduction and interpolation: Geophysics, **77**, V113–V122.

Kressner, D., and C. Tobler, 2012, htucker–a matlab toolbox for tensors in hierarchical tucker format: MATHICSE, EPF Lausanne (Preprint 2012), available at `http://sma.epfl.ch/~anchpcommon/publications/htucker.pdf`.

Uschmajew, A., and B. Vandereycken, 2012, The geometry of algorithms using hierarchical tensors: preprint, http://sma.epfl.ch/v̄anderey/papers/geom_htucker.pdf.

van den Berg, E., and M. Friedlander, 2009, `http://www.cs.ubc.ca/labs/scl/spot/index.html`.