Seismic Data Interpolation

- With each 3D seismic experiment, we acquire a massive 5D data volume
- Volumes easily have dimensions $100 \times 100$ (srcx, srcy) $\times$ $400 \times 400$ (recx, recy) $\times$ 1024 (time)
- Physical and budgetary constraints limit the number of available sources and receivers
Seismic Data Interpolation

- Knowledge of the low-rank structure of certain matricizations of the underlying (full) tensor allows us to recover unknown entries from the subsampled tensor
- Matrix Completion - Jellyfish
- Hierarchical Tucker Tensor Completion
Matricization

- The *matricization* of a tensor $X$ with dimensions $1, \ldots, d$ along the dimensions $t = (t_1, \ldots, t_r)$ is the matrix formed by placing the dimensions $t$ along the rows and dimensions $t^c$ along the columns.
- Denoted $X^{(t)}$.
Hierarchical Tucker Format

- A *dimension tree* $T$ for dimensions $\{1, \ldots, d\}$ is a non-trivial binary tree such that
- the root, $t_{\text{root}}$, is has the label $\{1, \ldots, d\}$
- each non-leaf node, $t$, can be written as $t = t_l \cup t_r$ where $t_l$ is the left child of $t$, $t_r$ is the right child of $t$
Hierarchical Tucker Format

• A tensor $X$ can be written in the **Hierarchical Tucker format** corresponding to a dimension tree $T$ and a vector of hierarchical ranks

$$ (k_t)_{t \in T}, k_{\text{root}} = 1 $$

if it can be written as
Hierarchical Tucker Format

$$\text{vec}(X) = \left( U_{t_l} \otimes U_{t_r} \right) B_{t_{\text{root}}}^{(1,2)} \quad t = t_{\text{root}}$$

$$U_t = \left( U_{t_l} \otimes U_{t_r} \right) B_t^{(1,2)} \quad t \text{ not a leaf}$$

$$U_t \in \mathbb{C}^{n_t \times k_t} \quad B_t \in \mathbb{C}^{k_l \times k_r \times k_t}$$
Hierarchical Tucker Format

\[ X - n_1 \times n_2 \times n_3 \times n_4 \text{ tensor} \]

\[ X^{(1,2)} = U_{12} B_{1234}^T k_{34} U_{34}^T \]

\[ U_{12} = (U_1 \otimes U_2) B_{12} \]
Hierarchical Tucker Format

- We don’t need to store the matrices $U_t$ when $t$ is not a leaf node
- We only need to store $U_t$ for the leaves, $B_t$ for the internal nodes
- We don’t need to store the (full) tensor
Hierarchical Tucker Format

• Storage \( \leq dNK + (d - 2)K^3 + K^2 \)
  where \( K = \max_{t \in T} k_t \), \( N = \max_{i=1, \ldots, d} n_i \)

• Compare to \( N^d \) storage for the full tensor
• Effectively breaking the curse of dimensionality when \( K \ll N, d > 3 \)
Hierarchical Tucker Format

- Truncation from a $N^d$ array to the Hierarchical Tucker format can be performed in $O(dN^{d+1})$ time by using SVDs + a hierarchical construction
- More details in the hTucker toolbox
- Cannot be applied when missing entries of the tensor
Quotient Manifold Structure

• The authors in [1] study the differential geometric structure of the non-orthogonal HTucker format
• We go beyond their analysis with the orthogonalized HTucker format + analysis to derive a computationally efficient optimization scheme for interpolation

Quotient Manifold Structure

- When we constrain $U_t, B_t$ to be orthogonal (except at the root), the group action
  \[ \theta : M \times A \to M \]
  \[ \theta \{ A_t \}_{t \in T} (U_t, B_t) = (U_t A_t, (A_{t_l}^T \otimes A_{t_r}^T) B_t A_t) \]
  induces a quotient manifold structure on the parameter matrices $M$
Optimization

• By characterizing the horizontal and vertical spaces of $M$ induced by this action, and by our choice of orthogonality constraints, we have a Riemannian manifold over which we can optimize.

• The Riemannian metric is the standard inner product.
Currently we are only trying to minimize the energy misfit between our model + the data we have (as well as choosing a rank for our underlying model)

Our assumption is that the underlying tensor has *quickly*-decaying singular values in different matricizations
Singular Value Regularization

- The **Gramian** matrix $G_t$ associated to the node $t$ is the $k_t \times k_t$ symmetric positive semidefinite matrix which satisfies

$$\lambda_i(G_t) = \sigma_i(X^{(t)})^2$$
Singular Value Regularization

- The Gramian matrices for every node can be computed recursively via

\[
G_{t_\text{root}} = 1
\]

\[
G_{t_r} = (B_t^{(k,k_l)})^H (I_{k_l} \otimes G_t) B_t^{(k,k_l)}
\]

\[
G_{t_l} = (B_t^{(k,k_r)})^H (I_{k_r} \otimes G_t) B_t^{(k,k_r)}
\]
Regularized Problem

\[
\min_{x=(U_t, B_t)} \|A\phi(x) - D\|^2_2 + \sum_t \alpha_t \|G_t\|_*
\]

\[
\alpha_t \geq 0 \quad U_t^T U_t = I_{k_t}, (B^{(1,2)})^H B^{(1,2)} = I_{k_t}
\]

Since \(G_t\) is spd, \(\|G_t\|_* = \text{tr}(G_t)\), this simplifies to

\[
\min_{x=(U_t, B_t)} \|A\phi(x) - D\|^2_2 + \sum_t \alpha_t \text{tr}(G_t)
\]
Regularization

- This objective is differentiable
- We already worked out the derivative of the tensor expansion function previously
- It is straightforward to derive the total derivative $\frac{\partial G_t}{\partial B_{t'}}$ and compute it efficiently
Hierarchical Tucker Interpolant

Reconstruction from 200 shots -> 6400 shots

Known data
(Src x, Src y) = (63,66)

Interpolated data -
SNR 13.2 dB
HTuck Interpolant - Regularized

Reconstruction from 200 shots -> 6400 shots

Known data
(Src x, Src y) = (63, 66)

Interpolated data - SNR 15 dB
Hierarchical Tucker Interpolant

Reconstruction from 200 shots -> 6400 shots

Interpolated data
(Src x, Src y) = (22, 18)

Interpolated data
(Src x, Src y) = (73, 18)
Hierarchical Tucker Interpolant

Reconstruction from 200 shots -> 6400 shots

Interpolated data
(Src x, Src y) = (64, 51)

Interpolated data
(Src x, Src y) = (13, 3)
Hierarchical Tucker Interpolant

Reconstruction from 200 shots -> 6400 shots

No Regularization

Regularization

(Src x, Src y) = (81, 51)
HTucker & Jellyfish
Shot Reconstruction

3 shots removed + reconstructed

True data
$(\text{Src } x, \text{ Src } y) = (45, 73)$

HTucker - SNR 8.46 dB
HTucker & Jellyfish
Shot Reconstruction

3 shots removed + reconstructed

True data
(Src x, Src y) = (45, 73)

Jellyfish - SNR 3.86 dB
HTucker & Jellyfish
Shot Reconstruction

3 shots removed + reconstructed

True data
(Src x, Src y) = (28,81)

HTucker - SNR 7.98 dB
HTucker & Jellyfish
Shot Reconstruction

3 shots removed + reconstructed

True data
(Src x, Src y) = (28, 81)

Jellyfish - SNR 3.89 dB