Advances In Tensor Decompositions

WAKE FOREST
UNIVERSITY

Department of Computer Science

João Pinheiro

Wednesday, April 16^{th} 2025

Fast Matrix Multiplication

- **A Simple and Famous Algorithm**:
  Every student knows the $O(n^3)$ version, but a lower bound proof exists, and we are nowhere near it with practical algoritms
- **Fundamental Operation in Computation:**
  It is a core building block in numerous algorithms across scientific computing, engineering, and data science
- **Impact on Other Algorithms:**
  Faster matrix multiplication improves the performance of many higher-level algorithms, including those in graph theory and numerical linear algebra.

Two Projects, Two Journeys

Fast Matrix Multiplication

- **A Simple and Famous Algorithm**:
  Every student knows the $O(n^3)$ version, but a lower bound proof exists, and we are nowhere near it with practical algoritms
- **Fundamental Operation in Computation:**
  It is a core building block in numerous algorithms across scientific computing, engineering, and data science
- **Impact on Other Algorithms:**
  Faster matrix multiplication improves the performance of many higher-level algorithms, including those in graph theory and numerical linear algebra.

Scientific Simulations

- **Accelerating Discovery**:
  Scientific computations allow researchers to simulate, model, and analyze complex phenomena without relying solely on physical experiments, which can be costly or impractical.
- **Handling Massive Datasets:**
  Modern scientific research often involves massive datasets, which require significant computational power to process, analyze, and extract meaningful insights.
- **Tensor Decompositions Help:**
  These uncover patterns on the data without requiring the whole dataset to be stored in memory, but rather a small representation of it.

What Is A Tensor?

What Is A Tensor?

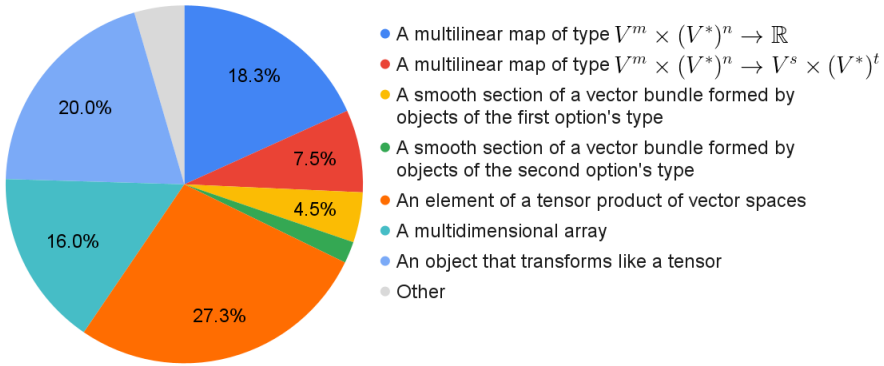
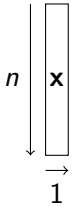


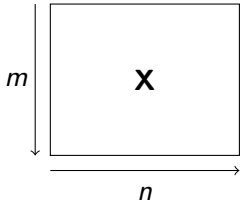
Figure: Question #45 of the *Math Conventions Survey* [1]

Multidimensional Arrays

Figure: Tensors of orders One, Two, and Three



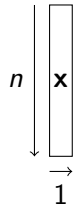
(a) Vector $\mathbf{x} \in \mathbb{R}^n$ is a 1-way tensor



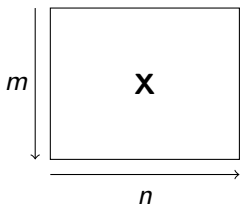
(b) Matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ is a 2-way tensor

Multidimensional Arrays

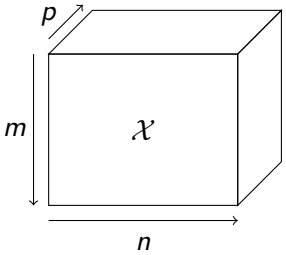
Figure: Tensors of orders One, Two, and Three



(a) Vector $\mathbf{x} \in \mathbb{R}^n$ is a 1-way tensor

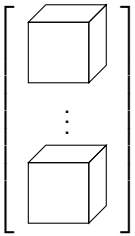


(b) Matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ is a 2-way tensor

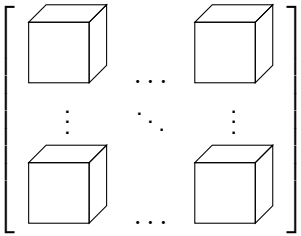


(c) Tensor $\mathcal{X} \in \mathbb{R}^{m \times n \times p}$ is a 3-way tensor

Multidimensional Arrays Cont.



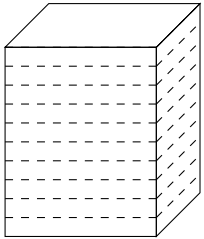
(a) A 4D Tensor
$\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$



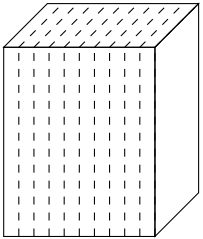
(b) A 5D Tensor
$\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4 \times n_5}$

Figure: Tensors of orders four and five

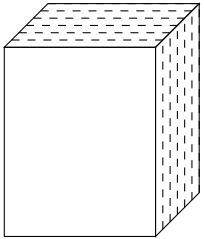
Tensor Slices



(a) Horizontal Slices $\mathcal{X}(i,:,:)$



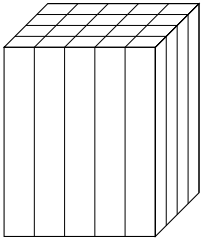
(b) Lateral Slices $\mathcal{X}(:,j,:)$



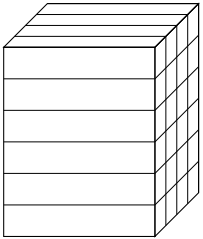
(c) Frontal Slices $\mathcal{X}(:,:,k)$

Figure: Two-way slices of a 3-way tensor

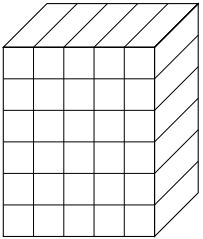
Tensor Fibers



(a) Column Fibers $\mathbf{x}_{:jk}$



(b) Row Fibers $\mathbf{x}_{i:k}$

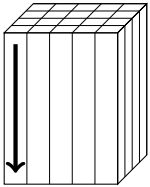


(c) Tube Fibers $\mathbf{x}_{ij:}$

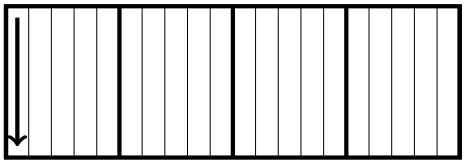
Figure: Fibers of a 3-way tensor

Tensor Unfoldings - Mode 1

Figure: Unfoldings of a 3-way tensor



(a) Mode 1 Fibers



(b) Mode 1 Unfolding $\mathbf{X}_{(1)}$

Vector-Vector Products

Vector Inner Products

$$
\begin{aligned}
c &= \langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^\intercal \mathbf{b} \in \mathbb{R} \\
&= \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \\
&= \sum_{i=1}^{n} a_i b_i
\end{aligned}
$$

Vector Outer Products

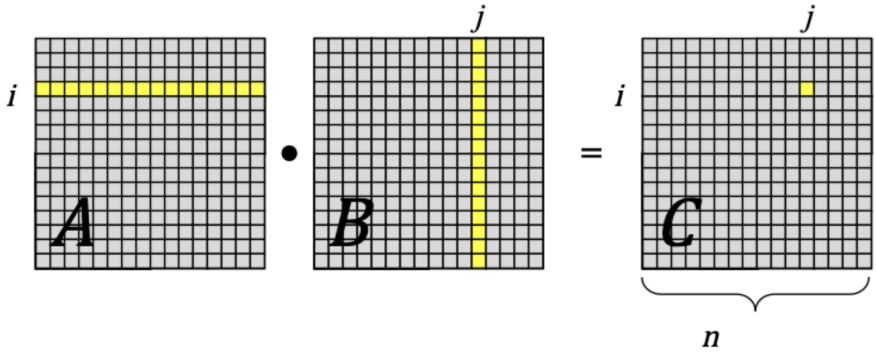
$$
\mathbf{C} = \mathbf{a} \circ \mathbf{b} = \mathbf{a}\mathbf{b}^\intercal \in \mathbb{R}^{m \times n}
$$

$$
\begin{bmatrix} a_1 b_1 & \cdots & a_1 b_n \\ \vdots & \ddots & \vdots \\ a_m b_1 & \cdots & a_m b_n \end{bmatrix} = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} \begin{bmatrix} b_1 & \cdots & b_n \end{bmatrix}
$$

$$
c_{ij} = a_i b_j
$$

Matrix-Matrix Products

Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$. Matrix Multiplication $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$ can be visualized as



Each entry of $\mathbf{C}$ is an inner product between corresponding rows of $\mathbf{A}$ and columns of $\mathbf{B}$

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$$

Tensor-Matrix Multiplication (TTM) - Tensor Format

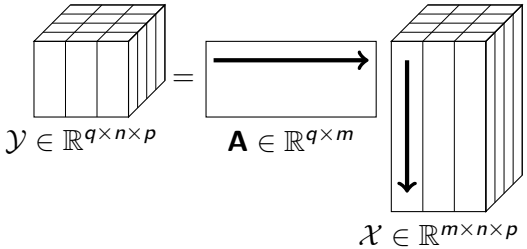


Figure: Tensor form TTM: the first row of $\mathbf{A}$ and first mode-1 fiber of $\mathcal{X}$ are emphasized with arrows.

Tensor-Matrix Multiplication (TTM) - Matrix Format

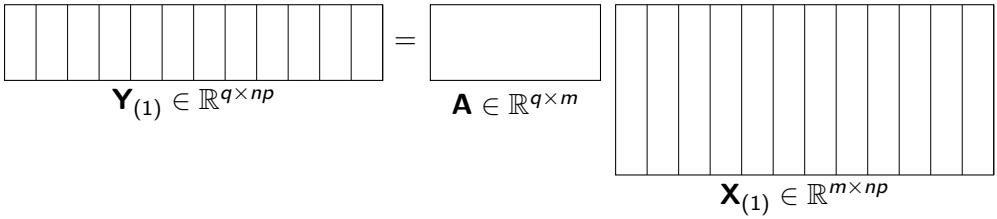


Figure: Matrix Form TTM: Multiplying tensor $\mathcal{X}$ with matrix $\mathbf{A}$ in the 1^{st} mode is the same as multiplying as the matrix-matrix product of $\mathbf{A}$ and $\mathbf{X}_{(1)}$.

Searching for Fast Matrix Multiplication Algorithms with Cyclic Invariance using CP Decompositions

Department of Computer Science and Department of Mathematics

João Pinheiro, Grey Ballard, Frank Moore, Pratyush Mishra

A 2 by 2 example of Matrix Multiplication

Consider a simple example of 2 by 2 matrices

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

$$= \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

A 2 by 2 example of Matrix Multiplication

Consider a simple example of 2 by 2 matrices

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$
$$= \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

We can use the above multiplication for any two matrices whose dimensions are multiples of 2. Each entry can be thought of as submatrices.

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22} \\ \mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22} \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{bmatrix}$$

A Recursive Implementation of the 2 by 2 Matrix Multiplication Algorithm

function $\mathbf{C} = \textsc{MatMul}(\mathbf{A}, \mathbf{B})$

  if $\dim(\mathbf{A}) = \dim(\mathbf{A}) = 1$ **then**

    return $\mathbf{A} \cdot \mathbf{B}$

  end if

  Divide into quadrants: $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$ $\mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}$

  $\mathbf{M}_1 = \text{MatMul}(\mathbf{A_{11}}, \mathbf{B_{11}})$

  $\mathbf{M}_2 = \text{MatMul}(\mathbf{A_{12}}, \mathbf{B_{21}})$

  $\mathbf{M}_3 = \text{MatMul}(\mathbf{A_{11}}, \mathbf{B_{12}})$

  $\mathbf{M}_4 = \text{MatMul}(\mathbf{A_{12}}, \mathbf{B_{22}})$

  $\mathbf{M}_5 = \text{MatMul}(\mathbf{A_{21}}, \mathbf{B_{21}})$

  $\mathbf{M}_6 = \text{MatMul}(\mathbf{A_{22}}, \mathbf{B_{12}})$

  $\mathbf{M}_7 = \text{MatMul}(\mathbf{A_{21}}, \mathbf{B_{12}})$

  $\mathbf{M}_8 = \text{MatMul}(\mathbf{A_{22}}, \mathbf{B_{22}})$

  return $\mathbf{C} = \begin{bmatrix} \mathbf{M}_1 + \mathbf{M}_2 & \mathbf{M}_3 + \mathbf{M}_4 \\ \mathbf{M}_5 + \mathbf{M}_6 & \mathbf{M}_7 + \mathbf{M}_8 \end{bmatrix}$

end function

A Recursive Implementation of the 2 by 2 Matrix Multiplication Algorithm

function $\mathbf{C} = \text{MatMul}(\mathbf{A}, \mathbf{B})$
  if $\dim(\mathbf{A}) = \dim(\mathbf{A}) = 1$ **then**
    return $\mathbf{A} \cdot \mathbf{B}$
  end if
  Divide into quadrants: $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$ $\mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}$
  $\mathbf{M}_1 = \text{MatMul}(\mathbf{A_{11}}, \mathbf{B_{11}})$
  $\mathbf{M}_2 = \text{MatMul}(\mathbf{A_{12}}, \mathbf{B_{21}})$
  $\mathbf{M}_3 = \text{MatMul}(\mathbf{A_{11}}, \mathbf{B_{12}})$
  $\mathbf{M}_4 = \text{MatMul}(\mathbf{A_{12}}, \mathbf{B_{22}})$
  $\mathbf{M}_5 = \text{MatMul}(\mathbf{A_{21}}, \mathbf{B_{21}})$
  $\mathbf{M}_6 = \text{MatMul}(\mathbf{A_{22}}, \mathbf{B_{12}})$
  $\mathbf{M}_7 = \text{MatMul}(\mathbf{A_{21}}, \mathbf{B_{12}})$
  $\mathbf{M}_8 = \text{MatMul}(\mathbf{A_{22}}, \mathbf{B_{22}})$
  return $\mathbf{C} = \begin{bmatrix} \mathbf{M}_1 + \mathbf{M}_2 & \mathbf{M}_3 + \mathbf{M}_4 \\ \mathbf{M}_5 + \mathbf{M}_6 & \mathbf{M}_7 + \mathbf{M}_8 \end{bmatrix}$
end function

$$
\begin{aligned}
\mathbf{M}_1 &= \mathbf{A}_{11} \cdot \mathbf{B}_{11} \\
\mathbf{M}_2 &= \mathbf{A}_{12} \cdot \mathbf{B}_{21} \\
\mathbf{M}_3 &= \mathbf{A}_{11} \cdot \mathbf{B}_{12} \\
\mathbf{M}_4 &= \mathbf{A}_{12} \cdot \mathbf{B}_{22} \\
\mathbf{M}_5 &= \mathbf{A}_{21} \cdot \mathbf{B}_{11} \\
\mathbf{M}_6 &= \mathbf{A}_{22} \cdot \mathbf{B}_{21} \\
\mathbf{M}_7 &= \mathbf{A}_{21} \cdot \mathbf{B}_{12} \\
\mathbf{M}_8 &= \mathbf{A}_{22} \cdot \mathbf{B}_{22} \\
\mathbf{C}_{11} &= \mathbf{M}_1 + \mathbf{M}_2 \\
\mathbf{C}_{12} &= \mathbf{M}_3 + \mathbf{M}_4 \\
\mathbf{C}_{21} &= \mathbf{M}_5 + \mathbf{M}_6 \\
\mathbf{C}_{22} &= \mathbf{M}_7 + \mathbf{M}_8
\end{aligned}
$$

A Recursive Implementation of the 2 by 2 Matrix Multiplication Algorithm

function $\mathbf{C} = \text{MatMul}(\mathbf{A}, \mathbf{B})$
  if $\dim(\mathbf{A}) = \dim(\mathbf{A}) = 1$ **then**
    return $\mathbf{A} \cdot \mathbf{B}$
  end if
  Divide into quadrants: $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$ $\mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}$
  $\mathbf{M}_1 = \text{MatMul}(\mathbf{A_{11}}, \mathbf{B_{11}})$
  $\mathbf{M}_2 = \text{MatMul}(\mathbf{A_{12}}, \mathbf{B_{21}})$
  $\mathbf{M}_3 = \text{MatMul}(\mathbf{A_{11}}, \mathbf{B_{12}})$
  $\mathbf{M}_4 = \text{MatMul}(\mathbf{A_{12}}, \mathbf{B_{22}})$
  $\mathbf{M}_5 = \text{MatMul}(\mathbf{A_{21}}, \mathbf{B_{21}})$
  $\mathbf{M}_6 = \text{MatMul}(\mathbf{A_{22}}, \mathbf{B_{12}})$
  $\mathbf{M}_7 = \text{MatMul}(\mathbf{A_{21}}, \mathbf{B_{12}})$
  $\mathbf{M}_8 = \text{MatMul}(\mathbf{A_{22}}, \mathbf{B_{22}})$
  return $\mathbf{C} = \begin{bmatrix} \mathbf{M}_1 + \mathbf{M}_2 & \mathbf{M}_3 + \mathbf{M}_4 \\ \mathbf{M}_5 + \mathbf{M}_6 & \mathbf{M}_7 + \mathbf{M}_8 \end{bmatrix}$
end function

$$
\begin{aligned}
\mathbf{M}_1 &= \mathbf{A}_{11} \cdot \mathbf{B}_{11} \\
\mathbf{M}_2 &= \mathbf{A}_{12} \cdot \mathbf{B}_{21} \\
\mathbf{M}_3 &= \mathbf{A}_{11} \cdot \mathbf{B}_{12} \\
\mathbf{M}_4 &= \mathbf{A}_{12} \cdot \mathbf{B}_{22} \\
\mathbf{M}_5 &= \mathbf{A}_{21} \cdot \mathbf{B}_{11} \\
\mathbf{M}_6 &= \mathbf{A}_{22} \cdot \mathbf{B}_{21} \\
\mathbf{M}_7 &= \mathbf{A}_{21} \cdot \mathbf{B}_{12} \\
\mathbf{M}_8 &= \mathbf{A}_{22} \cdot \mathbf{B}_{22} \\
\mathbf{C}_{11} &= \mathbf{M}_1 + \mathbf{M}_2 \\
\mathbf{C}_{12} &= \mathbf{M}_3 + \mathbf{M}_4 \\
\mathbf{C}_{21} &= \mathbf{M}_5 + \mathbf{M}_6 \\
\mathbf{C}_{22} &= \mathbf{M}_7 + \mathbf{M}_8
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= 8T(\tfrac{n}{2}) + O(n^2) \\
&= O(n^3)
\end{aligned}
$$

The Man, The Myth, The Legend

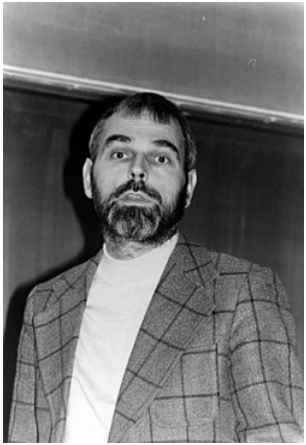


Figure: Volkner Strassen, 1969

The Man, The Myth, The Legend

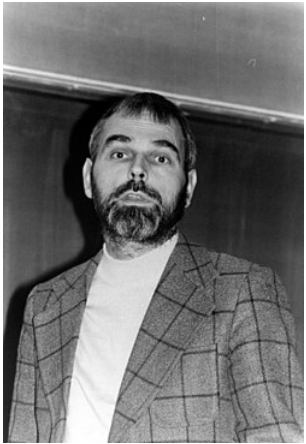


Figure: Volkner Strassen, 1969

$$
\begin{aligned}
\mathbf{M}_1 &= (\mathbf{A}_{11} + \mathbf{A}_{22}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{22}) \\
\mathbf{M}_2 &= (\mathbf{A}_{12} + \mathbf{A}_{22}) \cdot \mathbf{B}_{11} \\
\mathbf{M}_3 &= \mathbf{A}_{11} \cdot (\mathbf{B}_{21} - \mathbf{B}_{22}) \\
\mathbf{M}_4 &= \mathbf{A}_{22} \cdot (\mathbf{B}_{12} - \mathbf{B}_{11}) \\
\mathbf{M}_5 &= (\mathbf{A}_{11} + \mathbf{A}_{21}) \cdot \mathbf{B}_{22} \\
\mathbf{M}_6 &= (\mathbf{A}_{12} - \mathbf{A}_{11}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{21}) \\
\mathbf{M}_7 &= (\mathbf{A}_{21} - \mathbf{A}_{22}) \cdot (\mathbf{B}_{12} + \mathbf{B}_{22}) \\
\mathbf{C}_{11} &= \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{12} &= \mathbf{M}_3 + \mathbf{M}_5 \\
\mathbf{C}_{21} &= \mathbf{M}_2 + \mathbf{M}_4 \\
\mathbf{C}_{22} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6
\end{aligned}
$$

The Man, The Myth, The Legend

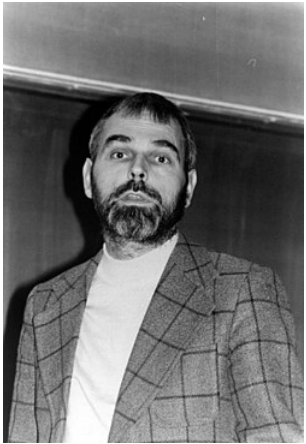


Figure: Volkner Strassen, 1969

$$
\begin{aligned}
\mathbf{M}_1 &= (\mathbf{A}_{11} + \mathbf{A}_{22}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{22}) \\
\mathbf{M}_2 &= (\mathbf{A}_{12} + \mathbf{A}_{22}) \cdot \mathbf{B}_{11} \\
\mathbf{M}_3 &= \mathbf{A}_{11} \cdot (\mathbf{B}_{21} - \mathbf{B}_{22}) \\
\mathbf{M}_4 &= \mathbf{A}_{22} \cdot (\mathbf{B}_{12} - \mathbf{B}_{11}) \\
\mathbf{M}_5 &= (\mathbf{A}_{11} + \mathbf{A}_{21}) \cdot \mathbf{B}_{22} \\
\mathbf{M}_6 &= (\mathbf{A}_{12} - \mathbf{A}_{11}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{21}) \\
\mathbf{M}_7 &= (\mathbf{A}_{21} - \mathbf{A}_{22}) \cdot (\mathbf{B}_{12} + \mathbf{B}_{22}) \\
\mathbf{C}_{11} &= \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{12} &= \mathbf{M}_3 + \mathbf{M}_5 \\
\mathbf{C}_{21} &= \mathbf{M}_2 + \mathbf{M}_4 \\
\mathbf{C}_{22} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= 7T(\tfrac{n}{2}) + O(n^2) \\
&= O(n^{2.81})
\end{aligned}
$$

Permuted Algorithms

Classic Strassen's Algorithm

$$
\begin{aligned}
\mathbf{M}_1 &= (\mathbf{A}_{11} + \mathbf{A}_{22}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{22}) \\
\mathbf{M}_2 &= (\mathbf{A}_{12} + \mathbf{A}_{22}) \cdot \mathbf{B}_{11} \\
\mathbf{M}_3 &= \mathbf{A}_{11} \cdot (\mathbf{B}_{21} - \mathbf{B}_{22}) \\
\mathbf{M}_4 &= \mathbf{A}_{22} \cdot (\mathbf{B}_{12} - \mathbf{B}_{11}) \\
\mathbf{M}_5 &= (\mathbf{A}_{11} + \mathbf{A}_{21}) \cdot \mathbf{B}_{22} \\
\mathbf{M}_6 &= (\mathbf{A}_{12} - \mathbf{A}_{11}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{21}) \\
\mathbf{M}_7 &= (\mathbf{A}_{21} - \mathbf{A}_{22}) \cdot (\mathbf{B}_{12} + \mathbf{B}_{22}) \\
\mathbf{C}_{11} &= \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{12} &= \mathbf{M}_3 + \mathbf{M}_5 \\
\mathbf{C}_{21} &= \mathbf{M}_2 + \mathbf{M}_4 \\
\mathbf{C}_{22} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6
\end{aligned}
$$

Permuted Strassen's Algorithm

$$
\begin{aligned}
\mathbf{M}_1 &= (\mathbf{A}_{11} + \mathbf{A}_{22}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{22}) \\
\mathbf{M}_2 &= (\mathbf{A}_{12} + \mathbf{A}_{22}) \cdot \mathbf{B}_{11} \\
\mathbf{M}_3 &= \mathbf{A}_{22} \cdot (\mathbf{B}_{12} - \mathbf{B}_{11}) \\
\mathbf{M}_4 &= (\mathbf{A}_{21} - \mathbf{A}_{22}) \cdot (\mathbf{B}_{12} + \mathbf{B}_{22}) \\
\mathbf{M}_5 &= (\mathbf{A}_{11} + \mathbf{A}_{21}) \cdot \mathbf{B}_{22} \\
\mathbf{M}_6 &= \mathbf{A}_{11} \cdot (\mathbf{B}_{21} - \mathbf{B}_{22}) \\
\mathbf{M}_7 &= (\mathbf{A}_{12} - \mathbf{A}_{11}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{21}) \\
\mathbf{C}_{11} &= \mathbf{M}_1 + \mathbf{M}_3 - \mathbf{M}_4 - \mathbf{M}_6 \\
\mathbf{C}_{12} &= \mathbf{M}_2 + \mathbf{M}_3 \\
\mathbf{C}_{21} &= \mathbf{M}_5 + \mathbf{M}_6 \\
\mathbf{C}_{22} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_5 + \mathbf{M}_7
\end{aligned}
$$

Notice how the two algorithms are the same, except that $\mathbf{M}_3$ became $\mathbf{M}_6$, $\mathbf{M}_4$ became $\mathbf{M}_3$, $\mathbf{M}_6$ became $\mathbf{M}_7$, and $\mathbf{M}_7$ became $\mathbf{M}_4$ and the additions in $\mathbf{C}$ changed respectively.

Variant Algorithms

Classic Strassen's Algorithm

$$
\begin{aligned}
\mathbf{M}_1 &= (\mathbf{A}_{11} + \mathbf{A}_{22}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{22}) \\
\mathbf{M}_2 &= (\mathbf{A}_{12} + \mathbf{A}_{22}) \cdot \mathbf{B}_{11} \\
\mathbf{M}_3 &= \mathbf{A}_{11} \cdot (\mathbf{B}_{21} - \mathbf{B}_{22}) \\
\mathbf{M}_4 &= \mathbf{A}_{22} \cdot (\mathbf{B}_{12} - \mathbf{B}_{11}) \\
\mathbf{M}_5 &= (\mathbf{A}_{11} + \mathbf{A}_{21}) \cdot \mathbf{B}_{22} \\
\mathbf{M}_6 &= (\mathbf{A}_{12} - \mathbf{A}_{11}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{21}) \\
\mathbf{M}_7 &= (\mathbf{A}_{21} - \mathbf{A}_{22}) \cdot (\mathbf{B}_{12} + \mathbf{B}_{22}) \\
\mathbf{C}_{11} &= \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{12} &= \mathbf{M}_3 + \mathbf{M}_5 \\
\mathbf{C}_{21} &= \mathbf{M}_2 + \mathbf{M}_4 \\
\mathbf{C}_{22} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6
\end{aligned}
$$

Variant Strassen's Algorithm

$$
\begin{aligned}
\mathbf{M}_1 &= \mathbf{A}_{11} \cdot \mathbf{B}_{11} \\
\mathbf{M}_2 &= (\mathbf{A}_{12} + \mathbf{A}_{22}) \cdot (\mathbf{B}_{12} + \mathbf{B}_{22}) \\
\mathbf{M}_3 &= (\mathbf{A}_{22} - \mathbf{A}_{21}) \cdot (\mathbf{B}_{22} - \mathbf{B}_{21}) \\
\mathbf{M}_4 &= (\mathbf{A}_{21} - \mathbf{A}_{12} - \mathbf{A}_{22}) \cdot (\mathbf{B}_{21} - \mathbf{B}_{12} - \mathbf{B}_{22}) \\
\mathbf{M}_5 &= (-\mathbf{A}_{12}) \cdot (-\mathbf{B}_{21}) \\
\mathbf{M}_6 &= (\mathbf{A}_{11} - \mathbf{A}_{12} + \mathbf{A}_{21} - \mathbf{A}_{22}) \cdot (-\mathbf{B}_{12}) \\
\mathbf{M}_7 &= (-\mathbf{A}_{21}) \cdot (\mathbf{B}_{11} - \mathbf{B}_{12} + \mathbf{B}_{21} - \mathbf{B}_{22}) \\
\mathbf{C}_{11} &= \mathbf{M}_1 + \mathbf{M}_5 \\
\mathbf{C}_{22} &= \mathbf{M}_4 - \mathbf{M}_3 + \mathbf{M}_5 + \mathbf{M}_6 \\
\mathbf{C}_{22} &= \mathbf{M}_2 - \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{22} &= \mathbf{M}_2 + \mathbf{M}_3 - \mathbf{M}_4 + \mathbf{M}_5
\end{aligned}
$$

Exhaustive Search of Fast MatMul Algorithms

$$
\begin{aligned}
\mathbf{M}_1 &= (u_{11}^{(1)}\mathbf{A}_{11} + u_{12}^{(1)}\mathbf{A}_{12} + u_{21}^{(1)}\mathbf{A}_{21} + u_{22}^{(1)}\mathbf{A}_{22}) \cdot (v_{11}^{(1)}\mathbf{B}_{11} + v_{12}^{(1)}\mathbf{B}_{12} + v_{21}^{(1)}\mathbf{B}_{21}v_{22}^{(1)} + \mathbf{B}_{22}) \\
\mathbf{M}_2 &= (u_{11}^{(2)}\mathbf{A}_{11} + u_{12}^{(2)}\mathbf{A}_{12} + u_{21}^{(2)}\mathbf{A}_{21} + u_{22}^{(2)}\mathbf{A}_{22}) \cdot (v_{11}^{(2)}\mathbf{B}_{11} + v_{12}^{(2)}\mathbf{B}_{12} + v_{21}^{(2)}\mathbf{B}_{21}v_{22}^{(2)} + \mathbf{B}_{22}) \\
\mathbf{M}_3 &= (u_{11}^{(3)}\mathbf{A}_{11} + u_{12}^{(3)}\mathbf{A}_{12} + u_{21}^{(3)}\mathbf{A}_{21} + u_{22}^{(3)}\mathbf{A}_{22}) \cdot (v_{11}^{(3)}\mathbf{B}_{11} + v_{12}^{(3)}\mathbf{B}_{12} + v_{21}^{(3)}\mathbf{B}_{21}v_{22}^{(3)} + \mathbf{B}_{22}) \\
\mathbf{M}_4 &= (u_{11}^{(4)}\mathbf{A}_{11} + u_{12}^{(4)}\mathbf{A}_{12} + u_{21}^{(4)}\mathbf{A}_{21} + u_{22}^{(4)}\mathbf{A}_{22}) \cdot (v_{11}^{(4)}\mathbf{B}_{11} + v_{12}^{(4)}\mathbf{B}_{12} + v_{21}^{(4)}\mathbf{B}_{21}v_{22}^{(4)} + \mathbf{B}_{22}) \\
\mathbf{M}_5 &= (u_{11}^{(5)}\mathbf{A}_{11} + u_{12}^{(5)}\mathbf{A}_{12} + u_{21}^{(5)}\mathbf{A}_{21} + u_{22}^{(5)}\mathbf{A}_{22}) \cdot (v_{11}^{(5)}\mathbf{B}_{11} + v_{12}^{(5)}\mathbf{B}_{12} + v_{21}^{(5)}\mathbf{B}_{21}v_{22}^{(5)} + \mathbf{B}_{22}) \\
\mathbf{M}_6 &= (u_{11}^{(6)}\mathbf{A}_{11} + u_{12}^{(6)}\mathbf{A}_{12} + u_{21}^{(6)}\mathbf{A}_{21} + u_{22}^{(6)}\mathbf{A}_{22}) \cdot (v_{11}^{(6)}\mathbf{B}_{11} + v_{12}^{(6)}\mathbf{B}_{12} + v_{21}^{(6)}\mathbf{B}_{21}v_{22}^{(6)} + \mathbf{B}_{22}) \\
\mathbf{M}_7 &= (u_{11}^{(7)}\mathbf{A}_{11} + u_{12}^{(7)}\mathbf{A}_{12} + u_{21}^{(7)}\mathbf{A}_{21} + u_{22}^{(7)}\mathbf{A}_{22}) \cdot (v_{11}^{(7)}\mathbf{B}_{11} + v_{12}^{(7)}\mathbf{B}_{12} + v_{21}^{(7)}\mathbf{B}_{21}v_{22}^{(7)} + \mathbf{B}_{22})
\end{aligned}
$$

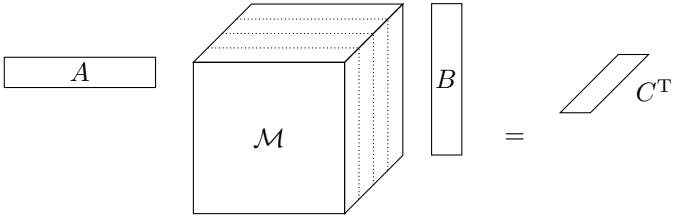
$$
\begin{aligned}
\mathbf{C}_{11} &= w_{11}^{(1)}\mathbf{M}_1 + w_{11}^{(2)}\mathbf{M}_2 + w_{11}^{(3)}\mathbf{M}_3 + w_{11}^{(4)}\mathbf{M}_4 + w_{11}^{(5)}\mathbf{M}_5 + w_{11}^{(6)}\mathbf{M}_6 + w_{11}^{(7)}\mathbf{M}_7 \\
\mathbf{C}_{12} &= w_{12}^{(1)}\mathbf{M}_1 + w_{12}^{(2)}\mathbf{M}_2 + w_{12}^{(3)}\mathbf{M}_3 + w_{12}^{(4)}\mathbf{M}_4 + w_{12}^{(5)}\mathbf{M}_5 + w_{12}^{(6)}\mathbf{M}_6 + w_{12}^{(7)}\mathbf{M}_7 \\
\mathbf{C}_{21} &= w_{21}^{(1)}\mathbf{M}_1 + w_{21}^{(2)}\mathbf{M}_2 + w_{21}^{(3)}\mathbf{M}_3 + w_{21}^{(4)}\mathbf{M}_4 + w_{21}^{(5)}\mathbf{M}_5 + w_{21}^{(6)}\mathbf{M}_6 + w_{21}^{(7)}\mathbf{M}_7 \\
\mathbf{C}_{22} &= w_{22}^{(1)}\mathbf{M}_1 + w_{22}^{(2)}\mathbf{M}_2 + w_{22}^{(3)}\mathbf{M}_3 + w_{22}^{(4)}\mathbf{M}_4 + w_{22}^{(5)}\mathbf{M}_5 + w_{22}^{(6)}\mathbf{M}_6 + w_{22}^{(7)}\mathbf{M}_7
\end{aligned}
$$

Considering only 2 by 2 algotithms of rank 7,

if we are searching for discrete solutions (coeffients -1, 0, 1) then we have 3^{84} possibilities.

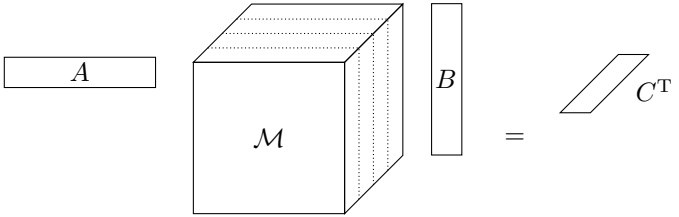
The Matrix Multiplication Tensor

We can visualize matrix multiplication in tensor format:



The Matrix Multiplication Tensor

We can visualize matrix multiplication in tensor format:



Which is equivalent to:

$$\mathcal{M} \times_1 vec(\mathbf{A}) \times_2 vec(\mathbf{B}) = \mathcal{M} \times_1 \begin{bmatrix} \mathbf{A}_{11} \\ \mathbf{A}_{12} \\ \mathbf{A}_{21} \\ \mathbf{A}_{22} \end{bmatrix} \times_2 \begin{bmatrix} \mathbf{B}_{11} \\ \mathbf{B}_{12} \\ \mathbf{B}_{21} \\ \mathbf{B}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{11} \\ \mathbf{C}_{21} \\ \mathbf{C}_{12} \\ \mathbf{C}_{22} \end{bmatrix} = vec(\mathbf{C}^{\mathsf{T}})$$

Matrix Multiplication as Tensor-Vector Products Example

For 2x2 case, these are what the frontal slices of the tensor corresponds to

$$\mathcal{M}(:,:,1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathcal{M}(:,:,2) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathcal{M}(:,:,3) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathcal{M}(:,:,4) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix Multiplication as Tensor-Vector Products Example

For 2x2 case, these are what the frontal slices of the tensor corresponds to

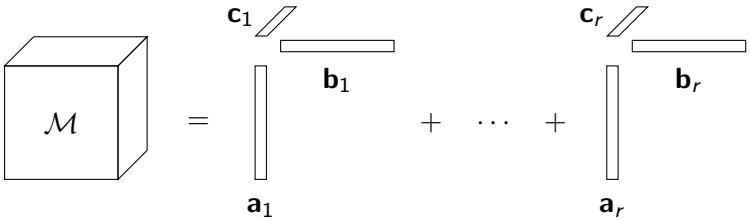
$$\mathcal{M}(:,:,1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathcal{M}(:,:,2) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathcal{M}(:,:,3) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathcal{M}(:,:,4) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

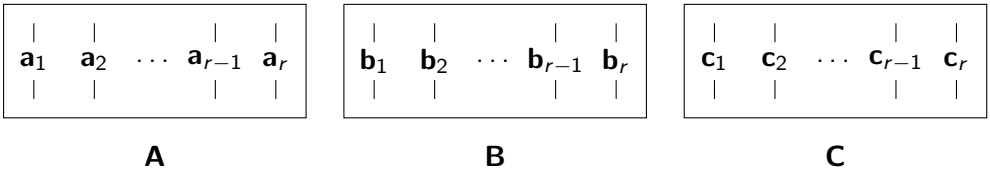
For example:

$$\mathcal{M}(:,:,3) \times_1 \begin{bmatrix} \mathbf{A}_{11} \\ \mathbf{A}_{12} \\ \mathbf{A}_{21} \\ \mathbf{A}_{22} \end{bmatrix} \times_2 \begin{bmatrix} \mathbf{B}_{11} \\ \mathbf{B}_{12} \\ \mathbf{B}_{21} \\ \mathbf{B}_{22} \end{bmatrix} = [\mathbf{A}_{11}\ \mathbf{A}_{12}\ \mathbf{A}_{21}\ \mathbf{A}_{22}] \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11} \\ \mathbf{B}_{12} \\ \mathbf{B}_{21} \\ \mathbf{B}_{22} \end{bmatrix} = \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22} = \mathbf{C}_{12}$$

The CP Decomposition



(a) A 3-way Kruskal Tensor Diagram



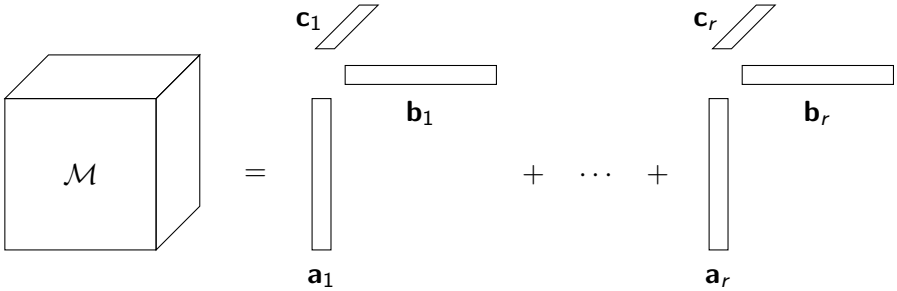
(b) The vectors of the components of the KTensor come together to form factor matrices

The Coolest Fact of All Time

$$
\begin{aligned}
\mathbf{M}_1 &= (\mathbf{A}_{11} + \mathbf{A}_{22}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{22}) \\
\mathbf{M}_2 &= (\mathbf{A}_{12} + \mathbf{A}_{22}) \cdot \mathbf{B}_{11} \\
\mathbf{M}_3 &= \mathbf{A}_{11} \cdot (\mathbf{B}_{21} - \mathbf{B}_{22}) \\
\mathbf{M}_4 &= \mathbf{A}_{22} \cdot (\mathbf{B}_{12} - \mathbf{B}_{11}) \\
\mathbf{M}_5 &= (\mathbf{A}_{11} + \mathbf{A}_{21}) \cdot \mathbf{B}_{22} \\
\mathbf{M}_6 &= (\mathbf{A}_{12} - \mathbf{A}_{11}) \cdot (\mathbf{B}_{11} + \mathbf{B}_{21}) \\
\mathbf{M}_7 &= (\mathbf{A}_{21} - \mathbf{A}_{22}) \cdot (\mathbf{B}_{12} + \mathbf{B}_{22}) \\
\mathbf{C}_{11} &= \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{12} &= \mathbf{M}_3 + \mathbf{M}_5 \\
\mathbf{C}_{21} &= \mathbf{M}_2 + \mathbf{M}_4 \\
\mathbf{C}_{22} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6
\end{aligned}
$$

	$\mathbf{M}_1$	$\mathbf{M}_2$	$\mathbf{M}_3$	$\mathbf{M}_4$	$\mathbf{M}_5$	$\mathbf{M}_6$	$\mathbf{M}_7$
$\mathbf{A}_{11}$	1	0	1	0	1	-1	0
$\mathbf{A}_{12}$	0	1	0	0	0	1	0
$\mathbf{A}_{21}$	0	0	0	0	1	0	1
$\mathbf{A}_{22}$	1	1	0	1	0	0	-1
$\mathbf{B}_{11}$	1	1	0	-1	0	1	0
$\mathbf{B}_{12}$	0	0	0	1	0	0	1
$\mathbf{B}_{21}$	0	0	1	0	0	1	0
$\mathbf{B}_{22}$	1	0	-1	0	1	0	1
$\mathbf{C}_{11}$	1	0	0	1	-1	0	1
$\mathbf{C}_{21}$	0	0	1	0	1	0	0
$\mathbf{C}_{12}$	0	1	0	1	0	0	0
$\mathbf{C}_{22}$	1	-1	1	0	0	1	0

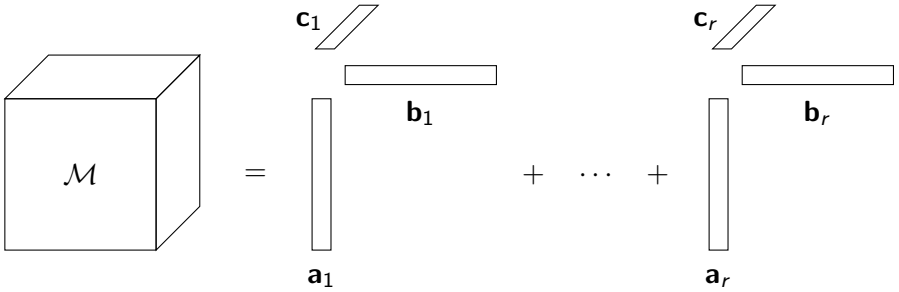
Details of the CP Decomposition



$$\mathcal{M} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_{\ell=1}^{r} a_\ell \circ b_\ell \circ c_\ell$$

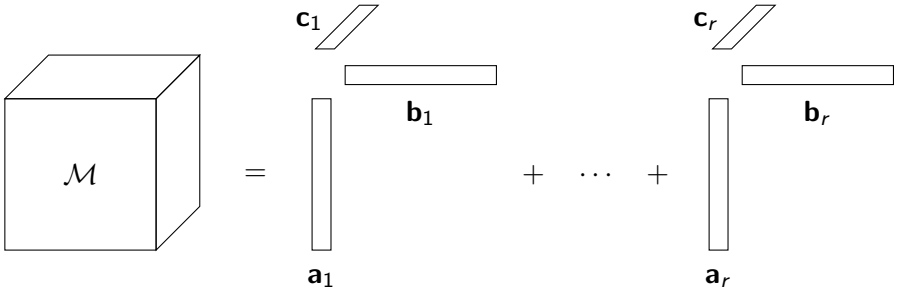
$$m_{ijk} = \sum_{\ell=1}^{r} a_{i\ell} b_{j\ell} c_{k\ell}$$

Details of the CP Decomposition Cont.



$$\|\mathcal{M} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|^2 \equiv \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{p} \left(t_{ijk} - \sum_{\ell=1}^{r} a_{i\ell} b_{j\ell} c_{k\ell} \right)^2$$

Details of the CP Decomposition Cont.



$$\|\mathcal{M} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|^2 \equiv \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{p} \left(t_{ijk} - \sum_{\ell=1}^{r} a_{i\ell} b_{j\ell} c_{k\ell} \right)^2$$

$$\min \|\mathcal{M} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|^2, \text{ subject to } \mathbf{A} \in \mathbb{R}^{m \times r} \mathbf{B} \in \mathbb{R}^{n \times r} \mathbf{C} \in \mathbb{R}^{p \times r}$$

Assumptions for Numerical Optimization

$$\min \|\mathcal{M} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|^2, \text{ subject to } \mathbf{A} \in \mathbb{R}^{m \times r}, \mathbf{B} \in \mathbb{R}^{n \times r}, \mathbf{C} \in \mathbb{R}^{p \times r}$$

Assumptions for Numerical Optimization

$$\min \|\mathcal{M} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|^2, \text{ subject to } \mathbf{A} \in \mathbb{R}^{m \times r}, \mathbf{B} \in \mathbb{R}^{n \times r}, \mathbf{C} \in \mathbb{R}^{p \times r}$$

$$\mathbf{v} = \text{vec}\left(\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{bmatrix}\right) = \begin{bmatrix} \text{vec}(\mathbf{A}) \\ \text{vec}(\mathbf{B}) \\ \text{vec}(\mathbf{C}) \end{bmatrix} \in \mathbb{R}^{3nr}$$

Assumptions for Numerical Optimization

$$\min \|\mathcal{M} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|^2, \text{ subject to } \mathbf{A} \in \mathbb{R}^{m \times r}, \mathbf{B} \in \mathbb{R}^{n \times r}, \mathbf{C} \in \mathbb{R}^{p \times r}$$

$$\mathbf{v} = \text{vec}\left(\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{bmatrix}\right) = \begin{bmatrix} \text{vec}(\mathbf{A}) \\ \text{vec}(\mathbf{B}) \\ \text{vec}(\mathbf{C}) \end{bmatrix} \in \mathbb{R}^{3nr}$$

$$\phi(\mathbf{v}) = \text{vec}(\mathcal{M} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]) : \mathbb{R}^{3nr} \to \mathbb{R}^{n^8}$$

Assumptions for Numerical Optimization

$$\min \|\mathcal{M} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|^2, \text{ subject to } \mathbf{A} \in \mathbb{R}^{m \times r}, \mathbf{B} \in \mathbb{R}^{n \times r}, \mathbf{C} \in \mathbb{R}^{p \times r}$$

$$\mathbf{v} = \text{vec}\left(\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{bmatrix}\right) = \begin{bmatrix} \text{vec}(\mathbf{A}) \\ \text{vec}(\mathbf{B}) \\ \text{vec}(\mathbf{C}) \end{bmatrix} \in \mathbb{R}^{3nr}$$

$$\phi(\mathbf{v}) = \text{vec}(\mathcal{M} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]) : \mathbb{R}^{3nr} \to \mathbb{R}^{n^8}$$

$$f(\mathbf{v}) = \frac{1}{2}\|\phi(\mathbf{v})\|^2 : \mathbb{R}^{3nr} \to \mathbb{R}$$

Assumptions for Numerical Optimization

$$\min \|\mathcal{M} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|^2, \text{ subject to } \mathbf{A} \in \mathbb{R}^{m \times r}, \mathbf{B} \in \mathbb{R}^{n \times r}, \mathbf{C} \in \mathbb{R}^{p \times r}$$

$$\mathbf{v} = \text{vec}\left(\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{bmatrix}\right) = \begin{bmatrix} \text{vec}(\mathbf{A}) \\ \text{vec}(\mathbf{B}) \\ \text{vec}(\mathbf{C}) \end{bmatrix} \in \mathbb{R}^{3nr}$$

$$\phi(\mathbf{v}) = \text{vec}(\mathcal{M} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]) : \mathbb{R}^{3nr} \to \mathbb{R}^{n^8}$$

$$f(\mathbf{v}) = \frac{1}{2}\|\phi(\mathbf{v})\|^2 : \mathbb{R}^{3nr} \to \mathbb{R}$$

$$\mathbf{d}_k = \text{vec}\left(\begin{bmatrix} \bar{\mathbf{A}} \\ \bar{\mathbf{B}} \\ \bar{\mathbf{C}} \end{bmatrix}\right) = \begin{bmatrix} \text{vec}(\bar{\mathbf{A}}) \\ \text{vec}(\bar{\mathbf{B}}) \\ \text{vec}(\bar{\mathbf{C}}) \end{bmatrix} \in \mathbb{R}^{3nr}$$

Damped Gauss Newton

Given our search direction $(\mathbf{J}^\intercal\mathbf{J} + \lambda\mathbf{I})\mathbf{d}_k = -\nabla f(\mathbf{v})$, where $\mathbf{J}$ is the jacobian of $\phi(n)$ and ∇f is the gradient of f, we wish to know how to (1) compute ∇f given a KTensor $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$.

$$\nabla f = \text{vec}\left(\begin{bmatrix} \partial f/\partial \mathbf{A} \\ \partial f/\partial \mathbf{B} \\ \partial f/\partial \mathbf{C} \end{bmatrix}\right) = \begin{bmatrix} \partial f/\partial \text{vec}(\mathbf{A}) \\ \partial f/\partial \text{vec}(\mathbf{B}) \\ \partial f/\partial \text{vec}(\mathbf{C}) \end{bmatrix} \in \mathbb{R}^{3n^2 r}$$

Where each partial derivative is defined as:

$$\begin{aligned} \partial f/\partial \mathbf{A} &= -\mathbf{M}_{(1)}(\mathbf{C} \odot \mathbf{B}) + \mathbf{A}(\mathbf{C}^\intercal\mathbf{C} * \mathbf{B}^\intercal\mathbf{B}) \in \mathbb{R}^{n^2 \times r} \\ \partial f/\partial \mathbf{B} &= -\mathbf{M}_{(2)}(\mathbf{C} \odot \mathbf{A}) + \mathbf{B}(\mathbf{C}^\intercal\mathbf{C} * \mathbf{A}^\intercal\mathbf{A}) \in \mathbb{R}^{n^2 \times r} \\ \partial f/\partial \mathbf{C} &= -\mathbf{M}_{(3)}(\mathbf{B} \odot \mathbf{A}) + \mathbf{C}(\mathbf{B}^\intercal\mathbf{B} * \mathbf{A}^\intercal\mathbf{A}) \in \mathbb{R}^{n^2 \times r} \end{aligned}$$

Damped Gauss Newton Cont.

Given our search direction $(\mathbf{J}^\intercal\mathbf{J} + \lambda\mathbf{I})\mathbf{d}_k = -\nabla f(\mathbf{v})$, where $\mathbf{J}$ is the jacobian of $\phi(n)$ and ∇f is the gradient of f, we wish to know how to (2) apply $(\mathbf{J}^\intercal\mathbf{J} + \lambda\mathbf{I})$ to vector $\mathbf{d}_k$ without forming $\mathbf{J}$ explicitly

$$(\mathbf{J}^\intercal\mathbf{J} + \lambda\mathbf{I})\mathbf{d}_k = \begin{bmatrix} \text{vec}(\bar{\mathbf{A}}(\mathbf{B}^\intercal\mathbf{B} * \mathbf{C}^\intercal\mathbf{C} + \lambda\mathbf{I}) + \mathbf{A}(\bar{\mathbf{B}}^\intercal\mathbf{B} * \mathbf{C}^\intercal\mathbf{C} + \lambda\mathbf{I}) + \mathbf{A}(\mathbf{B}^\intercal\mathbf{B} * \bar{\mathbf{C}}^\intercal\mathbf{C} + \lambda\mathbf{I})) \\ \text{vec}(\mathbf{B}(\bar{\mathbf{A}}^\intercal\mathbf{A} * \mathbf{C}^\intercal\mathbf{C} + \lambda\mathbf{I}) + \bar{\mathbf{B}}(\mathbf{A}^\intercal\mathbf{A} * \mathbf{C}^\intercal\mathbf{C} + \lambda\mathbf{I}) + \mathbf{B}(\mathbf{A}^\intercal\mathbf{A} * \bar{\mathbf{C}}^\intercal\mathbf{C} + \lambda\mathbf{I})) \\ \text{vec}(\mathbf{C}(\bar{\mathbf{A}}^\intercal\mathbf{A} * \mathbf{B}^\intercal\mathbf{B} + \lambda\mathbf{I}) + \mathbf{C}(\mathbf{A}^\intercal\mathbf{A} * \bar{\mathbf{B}}^\intercal\mathbf{B} + \lambda\mathbf{I}) + \bar{\mathbf{C}}(\mathbf{A}^\intercal\mathbf{A} * \mathbf{B}^\intercal\mathbf{B} + \lambda\mathbf{I})) \end{bmatrix}$$

Believe it or not,
computing the above saves us more time than computing $(\mathbf{J}^\intercal\mathbf{J} + \lambda\mathbf{I})\mathbf{d}_k$ directly...

The CP Damped Gauss Newton Algorithm

Input: Matrix Multiplication Tensor $\mathcal{M}$,
CP Tensor Rank r,
Damping Parameter $\lambda \in \mathbb{R}^+$,
Convergence Tolerance $\epsilon > 0$
Output: CP Tensor $\mathcal{K}$
function $\mathrm{CP\_DGN}(\mathcal{M}, r, \lambda, \epsilon)$
Initialize K and K_prev to be a cell of length 3 of $n^2 \times r$ matrices
for $i = 1 : MaxIters$ **do**
f $\longleftarrow \frac{1}{2}\|\mathcal{M} - \mathcal{K}\|^2$ ▷ Compute Function Value
$\nabla \mathbf{f} \longleftarrow [\mathrm{vec}\left(\frac{\partial f}{\partial \mathbf{A}}\right) \mathrm{vec}\left(\frac{\partial f}{\partial \mathbf{B}}\right) \mathrm{vec}\left(\frac{\partial f}{\partial \mathbf{C}}\right)]^\intercal$ ▷ Compute Gradient
$S \longleftarrow$ Solution to$(\mathbf{J}^T\mathbf{J} + \lambda I)K = -\nabla \mathbf{f}$ ▷ Conjugate Gradient Iter. Alg.
while *Goldstein Conditions Are Not Satisfied* **do**
K $\longleftarrow$ K_prev $+ \alpha S$
$\mathrm{f}_{new} \longleftarrow \frac{1}{2}\|\mathcal{M} - \mathcal{K}\|^2$
$\alpha \longleftarrow \alpha/2$
end while
if f - $\mathrm{f}_{new} < \epsilon$ **then**
break
end if
end for
end function

Classic Strassen

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
A_{11}	1	0	1	0	1	−1	0
A_{12}	0	1	0	0	0	1	0
A_{21}	0	0	0	0	1	0	1
A_{22}	1	1	0	1	0	0	−1
B_{11}	1	1	0	−1	0	1	0
B_{12}	0	0	0	1	0	0	1
B_{21}	0	0	1	0	0	1	0
B_{22}	1	0	−1	0	1	0	1
C_{11}	1	0	0	1	−1	0	1
C_{21}	0	0	1	0	1	0	0
C_{12}	0	1	0	1	0	0	0
C_{22}	1	−1	1	0	0	1	0

$r = 7$

Permuted Strassen

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
A_{11}	1	0	0	0	1	1	−1
A_{12}	0	1	0	0	0	0	1
A_{21}	0	0	0	1	1	0	0
A_{22}	1	1	1	−1	0	0	0
B_{11}	1	1	−1	0	0	0	1
B_{12}	0	0	1	1	0	0	0
B_{21}	0	0	0	0	0	1	1
B_{22}	1	0	0	1	1	−1	0
C_{11}	1	0	1	1	−1	0	0
C_{21}	0	0	0	0	1	1	0
C_{12}	0	1	1	0	0	0	0
C_{22}	1	−1	0	0	0	1	1

$r = 7, r_s = 1, r_c = 2$

Different Symmetric Ranks

Pemuted Strassen $r_s = 1, r_c = 2$

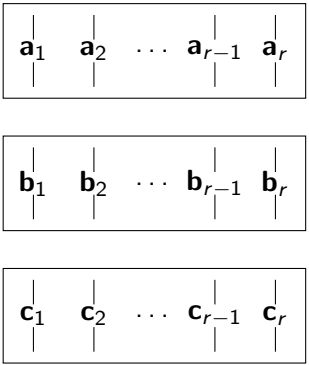
	M_1	M_2	M_3	M_4	M_5	M_6	M_7
A_{11}	1	0	0	0	1	1	−1
A_{12}	0	1	0	0	0	0	1
A_{21}	0	0	0	1	1	0	0
A_{22}	1	1	1	−1	0	0	0
B_{11}	1	1	−1	0	0	0	1
B_{12}	0	0	1	1	0	0	0
B_{21}	0	0	0	0	0	1	1
B_{22}	1	0	0	1	1	−1	0
C_{11}	1	0	1	1	−1	0	0
C_{21}	0	0	0	0	1	1	0
C_{12}	0	1	1	0	0	0	0
C_{22}	1	−1	0	0	0	1	1

Variant Strassen $r_s = 4, r_c = 1$

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
A_{11}	1	0	0	0	0	1	0
A_{12}	0	0	−1	1	0	1	−1
A_{21}	0	1	0	−1	−1	−1	0
A_{22}	0	1	1	−1	0	−1	0
B_{11}	1	0	0	0	0	0	1
B_{12}	0	0	−1	1	−1	0	1
B_{21}	0	1	0	−1	0	−1	−1
B_{22}	0	1	1	−1	0	0	−1
C_{11}	1	0	0	0	1	0	0
C_{21}	0	0	−1	1	1	−1	0
C_{12}	0	1	0	−1	−1	0	−1
C_{22}	0	1	1	−1	−1	0	0

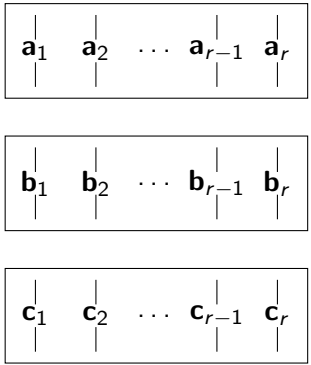
Adapting the CP_DGN Algorithm

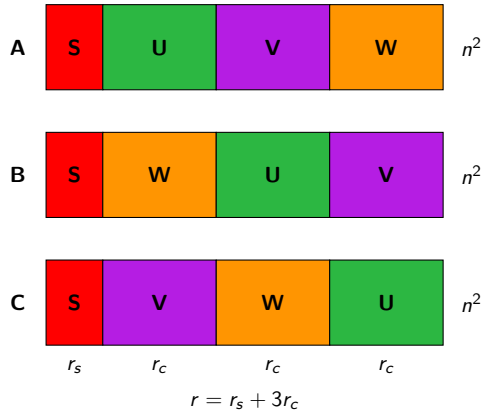
We now wish to search for algorithms with Cyclic Invariant structure



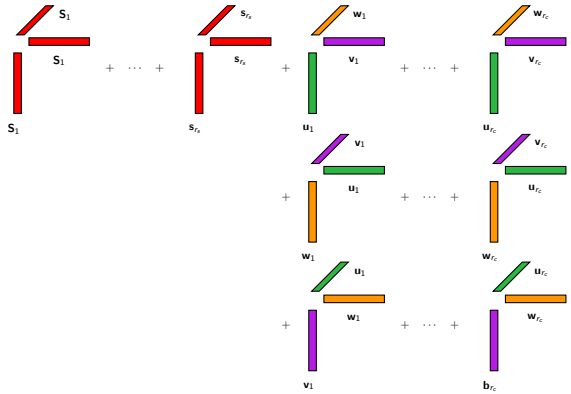
Adapting the CP_DGN Algorithm

We now wish to search for algorithms with Cyclic Invariant structure





The Chickenfeet



$$f(v) = \frac{1}{2}\sum_{i}^{m}\sum_{j}^{n}\sum_{k}^{p}\left(x_{ijk} - \sum_{q}^{r_s} s_{iq}s_{jq}s_{kq} - \sum_{l}^{r_c}\left(u_{il}v_{jl}w_{kl} + w_{il}u_{jl}v_{kl} + v_{il}w_{jl}u_{kl}\right)\right)^2$$

What does that mean Mathematically?

Now we impose the following structure on our algorithm

$$\begin{aligned}\mathbf{A} &= [\ \mathbf{S}\ \ \mathbf{U}\ \ \mathbf{V}\ \ \mathbf{W}\ ]\\ \mathbf{B} &= [\ \mathbf{S}\ \ \mathbf{W}\ \ \mathbf{U}\ \ \mathbf{V}\ ]\\ \mathbf{C} &= [\ \mathbf{S}\ \ \mathbf{V}\ \ \mathbf{W}\ \ \mathbf{U}\ ]\end{aligned}$$

What does that mean Mathematically?

Now we impose the following structure on our algorithm

$$\begin{aligned}\mathbf{A} &= [\ \mathbf{S} \quad \mathbf{U} \quad \mathbf{V} \quad \mathbf{W}\ ] \\ \mathbf{B} &= [\ \mathbf{S} \quad \mathbf{W} \quad \mathbf{U} \quad \mathbf{V}\ ] \\ \mathbf{C} &= [\ \mathbf{S} \quad \mathbf{V} \quad \mathbf{W} \quad \mathbf{U}\ ]\end{aligned}$$

$$\mathbf{v} = \text{vec}\left(\begin{bmatrix}\mathbf{A} \\ \mathbf{B} \\ \mathbf{C}\end{bmatrix}\right) = \begin{bmatrix}\text{vec}(\mathbf{A}) \\ \text{vec}(\mathbf{B}) \\ \text{vec}(\mathbf{C})\end{bmatrix} \in \mathbb{R}^{3nr}$$

What does that mean Mathematically?

Now we impose the following structure on our algorithm

$$\begin{aligned}\mathbf{A} &= [\ \mathbf{S}\ \ \mathbf{U}\ \ \mathbf{V}\ \ \mathbf{W}\ ]\\ \mathbf{B} &= [\ \mathbf{S}\ \ \mathbf{W}\ \ \mathbf{U}\ \ \mathbf{V}\ ]\\ \mathbf{C} &= [\ \mathbf{S}\ \ \mathbf{V}\ \ \mathbf{W}\ \ \mathbf{U}\ ]\end{aligned}$$

$$\mathbf{v} = \operatorname{vec}\left(\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\\ \mathbf{C}\end{bmatrix}\right) = \begin{bmatrix}\operatorname{vec}(\mathbf{A})\\ \operatorname{vec}(\mathbf{B})\\ \operatorname{vec}(\mathbf{C})\end{bmatrix} \in \mathbb{R}^{3nr} \qquad \mathbf{v} = \operatorname{vec}\left(\begin{bmatrix}\mathbf{S}\\ \mathbf{U}\\ \mathbf{V}\\ \mathbf{W}\end{bmatrix}\right) = \begin{bmatrix}\operatorname{vec}(\mathbf{S})\\ \operatorname{vec}(\mathbf{U})\\ \operatorname{vec}(\mathbf{V})\\ \operatorname{vec}(\mathbf{W})\end{bmatrix} \in \mathbb{R}^{nr}$$

Exaustively, this is only 2^{28}. Then we adapt our gradient and how to apply $(\mathbf{J}^{\mathsf{T}}\mathbf{J} + \lambda\mathbf{I})\mathbf{d}_k$ with this structure.

Details of Adapting the CP_DGN Algorithm

The gradients become this:

$$
\begin{aligned}
\frac{\partial f}{\partial \mathbf{S}} &= 3 \cdot \Big(\mathbf{S}(\mathbf{S}^\intercal \mathbf{S} * \mathbf{S}^\intercal \mathbf{S}) + \mathbf{U}(\mathbf{V}^\intercal \mathbf{S} * \mathbf{W}^\intercal \mathbf{S}) + \mathbf{V}(\mathbf{U}^\intercal \mathbf{S} * \mathbf{W}^\intercal \mathbf{S}) + \mathbf{W}(\mathbf{U}^\intercal \mathbf{S} * \mathbf{V}^\intercal \mathbf{S}) \\
&\quad - (\mathbf{X}_{(1)} + \mathbf{X}_{(2)} + \mathbf{X}_{(3)})(\mathbf{S} \odot \mathbf{S}) \Big) \\
\frac{\partial f}{\partial \mathbf{U}} &= 3 \cdot \Big(\mathbf{S}(\mathbf{S}^\intercal \mathbf{V} * \mathbf{S}^\intercal \mathbf{W}) + \mathbf{U}(\mathbf{V}^\intercal \mathbf{V} * \mathbf{W}^\intercal \mathbf{W}) + \mathbf{V}(\mathbf{W}^\intercal \mathbf{V} * \mathbf{U}^\intercal \mathbf{W}) + \mathbf{W}(\mathbf{U}^\intercal \mathbf{V} * \mathbf{V}^\intercal \mathbf{W}) \Big) \\
&\quad - \mathbf{X}_{(1)}(\mathbf{V} \odot \mathbf{W}) - \mathbf{X}_{(2)}(\mathbf{W} \odot \mathbf{V}) - \mathbf{X}_{(3)}(\mathbf{V} \odot \mathbf{W}) \\
\frac{\partial f}{\partial \mathbf{V}} &= 3 \cdot \Big(\mathbf{S}(\mathbf{S}^\intercal \mathbf{U} * \mathbf{S}^\intercal \mathbf{W}) + \mathbf{U}(\mathbf{W}^\intercal \mathbf{U} * \mathbf{V}^\intercal \mathbf{W}) + \mathbf{V}(\mathbf{U}^\intercal \mathbf{U} * \mathbf{W}^\intercal \mathbf{W}) + \mathbf{W}(\mathbf{V}^\intercal \mathbf{U} * \mathbf{U}^\intercal \mathbf{W}) \Big) \\
&\quad - \mathbf{X}_{(1)}(\mathbf{W} \odot \mathbf{U}) - \mathbf{X}_{(2)}(\mathbf{U} \odot \mathbf{W}) - \mathbf{X}_{(3)}(\mathbf{W} \odot \mathbf{U}) \\
\frac{\partial f}{\partial \mathbf{W}} &= 3 \cdot \Big(\mathbf{S}(\mathbf{S}^\intercal \mathbf{U} * \mathbf{S}^\intercal \mathbf{V}) + \mathbf{U}(\mathbf{V}^\intercal \mathbf{U} * \mathbf{W}^\intercal \mathbf{V}) + \mathbf{V}(\mathbf{W}^\intercal \mathbf{U} * \mathbf{U}^\intercal \mathbf{V}) + \mathbf{W}(\mathbf{U}^\intercal \mathbf{U} * \mathbf{V}^\intercal \mathbf{V}) \Big) \\
&\quad - \mathbf{X}_{(1)}(\mathbf{U} \odot \mathbf{V}) - \mathbf{X}_{(2)}(\mathbf{V} \odot \mathbf{U}) - \mathbf{X}_{(3)}(\mathbf{U} \odot \mathbf{V})
\end{aligned}
$$

Details of Adapting the CP_DGN Algorithm Cont.

Applying $(\mathbf{J}^\intercal\mathbf{J} + \lambda\mathbf{I}d)_k$ become this (there are 12 mode of these):

$$
\begin{aligned}
\mathbf{J}_{\mathbf{U}}^\intercal\mathbf{J}_{\mathbf{S}}\text{vec}(\mathbf{K_s}) &= 3\cdot\text{vec}\Big(\mathbf{K_S}(\mathbf{S}^\intercal\mathbf{V})*(\mathbf{S}^\intercal\mathbf{W})+\mathbf{S}\Big((\mathbf{K}_{\mathbf{S}}^\intercal\mathbf{W})*(\mathbf{S}^\intercal\mathbf{V})+(\mathbf{K}_{\mathbf{S}}^\intercal\mathbf{V})*(\mathbf{S}^\intercal\mathbf{W})\Big)\Big)\\
\mathbf{J}_{\mathbf{U}}^\intercal\mathbf{J}_{\mathbf{U}}\text{vec}(\mathbf{K_u}) &= 3\cdot\text{vec}\Big(\mathbf{K_U}(\mathbf{V}^\intercal\mathbf{V})*(\mathbf{W}^\intercal\mathbf{W})+\mathbf{V}(\mathbf{K}_{\mathbf{U}}^\intercal\mathbf{W})*(\mathbf{W}^\intercal\mathbf{V})+\mathbf{W}(\mathbf{K}_{\mathbf{U}}^\intercal\mathbf{V})*(\mathbf{V}^\intercal\mathbf{W})\Big)\\
\mathbf{J}_{\mathbf{U}}^\intercal\mathbf{J}_{\mathbf{V}}\text{vec}(\mathbf{K_v}) &= 3\cdot\text{vec}\Big(\mathbf{K_V}(\mathbf{W}^\intercal\mathbf{V})*(\mathbf{U}^\intercal\mathbf{W})+\mathbf{W}(\mathbf{K}_{\mathbf{V}}^\intercal\mathbf{W})*(\mathbf{U}^\intercal\mathbf{V})+\mathbf{U}(\mathbf{K}_{\mathbf{V}}^\intercal\mathbf{V})*(\mathbf{W}^\intercal\mathbf{W})\Big)\\
\mathbf{J}_{\mathbf{U}}^\intercal\mathbf{J}_{\mathbf{W}}\text{vec}(\mathbf{K_w}) &= 3\cdot\text{vec}\Big(\mathbf{K_W}(\mathbf{U}^\intercal\mathbf{V})*(\mathbf{V}^\intercal\mathbf{W})+\mathbf{U}(\mathbf{K}_{\mathbf{W}}^\intercal\mathbf{W})*(\mathbf{V}^\intercal\mathbf{V})+\mathbf{V}(\mathbf{K}_{\mathbf{W}}^\intercal\mathbf{V})*(\mathbf{U}^\intercal\mathbf{W})\Big)
\end{aligned}
$$

With all of these modifications we implement our new algorithm, called it `CI_CP_DGN`

Our Solutions

$n = 2, r = 4$	S	$n = 3, r = 23$	S	$n = 4, r = 49$	S	$n = 5$	S
$r_s = 1, r_c = 2$	1000s	$r_s = 2, r_c = 7$	100s	$r_s = 1, r_c = 16$	6	$r = 109$	2
$r_s = 4, r_c = 1$	1000s	$r_s = 5, r_c = 6$	100s	$r_s = 4, r_c = 15$	0	$r = 93$	-
		$r_s = 8, r_c = 5$	-	$r_s = 7, r_c = 14$	-	$r = 91$	0
		$r_s = 11, r_c = 4$	100s	$r_s = 10, r_c = 13$	-	$\cdots$	-
		$r_s = 14, r_c = 3$	-	$r_s = 13, r_c = 12$	2		
		$r_s = 17, r_c = 2$	-	$r_s = 16, r_c = 11$	32		
		$r_s = 20, r_c = 1$	-	$\cdots$	-		

Conclusions - Project 2

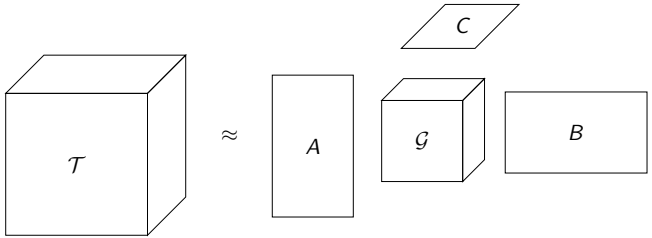
- These algorithms are hard to find, we make it easier
- Current work is done to impose further structure in the algorithms we have.
- Once that is automated, we want to modify `CI_CP_DGN` to have these Structures

Parallel Higher-Order Orthogonal Iteration for Tucker Decomposition with Rank Adaptivity

Department of Computer Science

João Pinheiro, Grey Ballard, Aditya Devarakonda

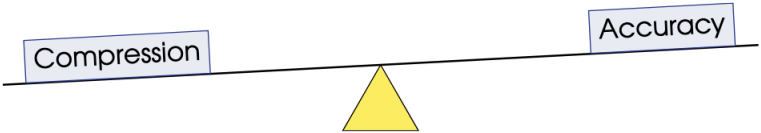
The Tucker Decomposition



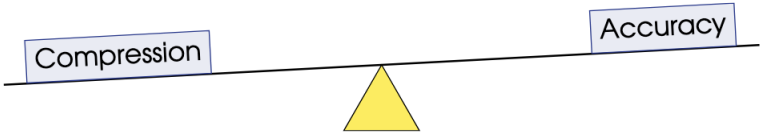
$$\mathcal{T} \approx [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$$

$$t_{ijk} \approx \sum_{\alpha=1}^{q} \sum_{\beta=1}^{r} \sum_{\gamma=1}^{s} g_{\alpha\beta\gamma} \cdot a_{i\alpha} b_{j\beta} c_{i\gamma}, \ \forall (i, j, k) \in [m] \otimes [n] \otimes [p]$$

The Tucker Decomposition Trade-Off



The Tucker Decomposition Trade-Off

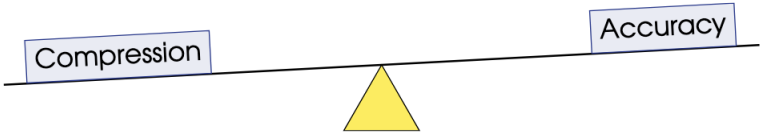


To know compression beforehand, we specify the size of the core tensor $\mathcal{G}$.

$$\begin{matrix}\text{compression}\\ \text{ratio}\end{matrix} = \frac{mnp}{qrs + qm + nr + sp} \approx \frac{mnp}{qrs}$$

This is the rank-specified formulation, where we cannot say in advance what the error will be

The Tucker Decomposition Trade-Off



To know compression beforehand, we specify the size of the core tensor $\mathcal{G}$.

$$\text{compression ratio} = \frac{mnp}{qrs + qm + nr + sp} \approx \frac{mnp}{qrs}$$

This is the rank-specified formulation, where we cannot say in advance what the error will be

To know accuracy beforehand, we specify the maximum relative error threshold

$$\frac{||\mathcal{X} - [\![\mathcal{G}; \mathbf{U}, \mathbf{V}, \mathbf{W}]\!]||}{||\mathcal{X}||} \leq \epsilon$$

This is the error-specified formulation, where we cannot say in advance what the compression will be

The Two Protagonists

function $\mathrm{STHOSVD}(\mathcal{X}$, **r** or $\epsilon)$
  $\mathbf{A} \leftarrow \mathit{LLSV}(\mathbf{X}_{(1)}, r_1 \text{ or } \epsilon)$
  $\mathcal{G} \leftarrow \mathcal{X} \times_1 A^\intercal$

  $\mathbf{B} \leftarrow \mathit{LLSV}(\mathbf{G}_{(2)}, r_2 \text{ or } \epsilon)$
  $\mathcal{G} \leftarrow \mathcal{G} \times_1 B^\intercal$

  $\mathbf{C} \leftarrow \mathit{LLSV}(\mathbf{G}_{(3)}, r_3 \text{ or } \epsilon)$
  $\mathcal{G} \leftarrow \mathcal{G} \times_3 C^\intercal$
  return $[\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$
end function

The Two Protagonists

function $\mathrm{STHOSVD}(\mathcal{X}, \mathbf{r} \text{ or } \epsilon)$
  $\mathbf{A} \leftarrow \mathit{LLSV}(\mathbf{X}_{(1)}, r_1 \text{ or } \epsilon)$
  $\mathcal{G} \leftarrow \mathcal{X} \times_1 A^\intercal$

  $\mathbf{B} \leftarrow \mathit{LLSV}(\mathbf{G}_{(2)}, r_2 \text{ or } \epsilon)$
  $\mathcal{G} \leftarrow \mathcal{G} \times_1 B^\intercal$

  $\mathbf{C} \leftarrow \mathit{LLSV}(\mathbf{G}_{(3)}, r_3 \text{ or } \epsilon)$
  $\mathcal{G} \leftarrow \mathcal{G} \times_3 C^\intercal$
  return $[\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$
end function
function $U = \mathrm{LLSV}([\![Y, r \text{ or } \epsilon)$
  $S = \mathbf{Y} \cdot \mathbf{Y}^\intercal$
  $[\mathbf{U}, \Lambda] = \mathrm{eig}(\mathbf{S})$
  return $\mathbf{U}(:, 1:r)$
end function

LLSV $\rightarrow$ Left Leading Singular Vectors

The Two Protagonists

function STHOSVD($\mathcal{X}$, **r** or ϵ)
  $\mathbf{A} \leftarrow \mathit{LLSV}(\mathbf{X}_{(1)}, r_1 \text{ or } \epsilon)$
  $\mathcal{G} \leftarrow \mathcal{X} \times_1 A^\intercal$

  $\mathbf{B} \leftarrow \mathit{LLSV}(\mathbf{G}_{(2)}, r_2 \text{ or } \epsilon)$
  $\mathcal{G} \leftarrow \mathcal{G} \times_1 B^\intercal$

  $\mathbf{C} \leftarrow \mathit{LLSV}(\mathbf{G}_{(3)}, r_3 \text{ or } \epsilon)$
  $\mathcal{G} \leftarrow \mathcal{G} \times_3 C^\intercal$
  return $[\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$
end function

function $U = \mathrm{LLSV}([\![Y, r \text{ or } \epsilon)$
  $S = \mathbf{Y} \cdot \mathbf{Y}^\intercal$
  $[\mathbf{U}, \Lambda] = \mathrm{eig}(\mathbf{S})$
  return $\mathbf{U}(:, 1:r)$
end function

LLSV $\rightarrow$ Left Leading Singular Vectors

function HOOI($\mathcal{X}$, **r**)
  Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$ randomly
  for Max Iterations **do**
    $\mathcal{Y} = \mathcal{X} \times_2 \mathbf{B}^\intercal \times_3 \mathbf{C}^\intercal$
    $\mathbf{A} \leftarrow \mathit{LLSV}(\mathbf{Y}_{(1)}, r_1)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_3 \mathbf{C}^\intercal$
    $\mathbf{B} \leftarrow \mathit{LLSV}(\mathbf{Y}_{(2)}, r_2)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_2 \mathbf{B}^\intercal$
    $\mathbf{C} \leftarrow \mathit{LLSV}(\mathbf{Y}_{(3)}, r_3)$
  end for
  $\mathcal{G} \leftarrow \mathcal{Y} \times_3 \mathbf{C}^\intercal$
  return $[\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$
end function

Optimization 1: Dimension Tree Memoization

function $\mathrm{HOOI}(\mathcal{X}, \mathbf{r})$
  Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$ randomly
  for Max Iterations **do**
    $\mathcal{Y} = \mathcal{X} \times_2 \mathbf{B}^\intercal \times_3 \mathbf{C}^\intercal$
    $A \leftarrow \mathit{LLSV}(Y_{(1)}, r_1)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_3 \mathbf{C}^\intercal$
    $B \leftarrow \mathit{LLSV}(Y_{(2)}, r_2)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_2 \mathbf{B}^\intercal$
    $C \leftarrow \mathit{LLSV}(Y_{(3)}, r_3)$
  end for
  $\mathcal{G} \leftarrow \mathcal{Y} \times_3 \mathbf{C}^\intercal$
  return $[\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$
end function

Optimization 1: Dimension Tree Memoization

function $\mathrm{HOOI}(\mathcal{X}, \mathbf{r})$
  Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$ randomly
  for Max Iterations **do**
    $\mathcal{Y} = \mathcal{X} \times_2 \mathbf{B}^\intercal \times_3 \mathbf{C}^\intercal$
    $A \leftarrow \mathit{LLSV}(Y_{(1)}, r_1)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_3 \mathbf{C}^\intercal$
    $B \leftarrow \mathit{LLSV}(Y_{(2)}, r_2)$

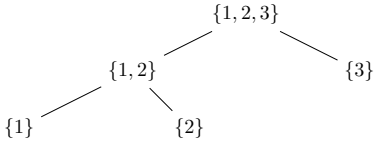
    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_2 \mathbf{B}^\intercal$
    $C \leftarrow \mathit{LLSV}(Y_{(3)}, r_3)$
  end for
  $\mathcal{G} \leftarrow \mathcal{Y} \times_3 \mathbf{C}^\intercal$
  return $[\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$
end function

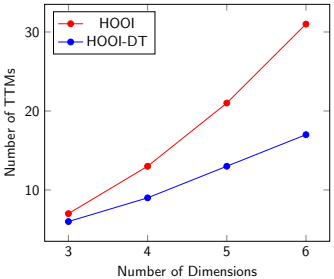
function $\mathrm{HOOI\text{-}DT}(\mathcal{X}, \mathbf{r})$
  Initialize A, B, C randomly
  for Max Iterations **do**
    $\mathcal{Y}_{\text{temp}} = \mathcal{X} \times_3 \mathbf{C}^\intercal$
    $\mathcal{Y} = \mathcal{Y}_{\text{temp}} \times_2 \mathbf{B}^\intercal$
    $\mathbf{A} \leftarrow \mathit{LLSV}(\mathbf{Y}_{(1)}, r_1)$

    $\mathcal{Y} = \mathcal{Y}_{\text{temp}} \times_1 \mathbf{A}^\intercal$
    $\mathbf{B} \leftarrow \mathit{LLSV}(\mathbf{Y}_{(2)}, r_2)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_2 \mathbf{B}^\intercal$
    $\mathbf{C} \leftarrow \mathit{LLSV}(\mathbf{Y}_{(3)}, r_3)$
  end for
  $\mathcal{G} \leftarrow \mathcal{Y} \times_3 \mathbf{C}^\intercal$
  return $[\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$
end function

Optimization 1: Dimension Tree Memoization





function HOOI-DT($\mathcal{X}$, **r**)
  Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$ randomly
  for Max Iterations **do**
    $\mathcal{Y}_{\text{temp}} = \mathcal{X} \times_3 \mathbf{C}^\intercal$
    $\mathcal{Y} = \mathcal{Y}_{\text{temp}} \times_2 \mathbf{B}^\intercal$
    $\mathbf{A} \leftarrow \mathit{LLSV}(\mathbf{Y}_{(1)}, r_1)$

    $\mathcal{Y} = \mathcal{Y}_{\text{temp}} \times_1 \mathbf{A}^\intercal$
    $\mathbf{B} \leftarrow \mathit{LLSV}(\mathbf{Y}_{(2)}, r_2)$

    $\mathcal{Y} = \mathcal{X} \times_1 A^\intercal \times_2 \mathbf{B}^\intercal$
    $\mathbf{C} \leftarrow \mathit{LLSV}(\mathbf{Y}_{(3)}, r_3)$
  end for
  $\mathcal{G} \leftarrow \mathcal{Y} \times_3 C^\intercal$
  return $[\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$
end function

Optimization 2: Subspace Iterations

function $\mathrm{HOSI}(\mathcal{X}, \mathbf{r})$
  Initialize A, B, C randomly
  for Max Iterations **do**
    $\mathcal{Y} = \mathcal{X} \times_2 \mathbf{B}^\intercal \times_3 \mathbf{C}^\intercal$
    $\mathbf{A} \leftarrow \mathit{LLSV}(\mathcal{Y}, \mathbf{A}, 1)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_3 \mathbf{C}^\intercal$
    $\mathbf{B} \leftarrow \mathit{LLSV}(\mathcal{Y}, \mathbf{B}, 2)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_2 \mathbf{B}^\intercal$
    $\mathbf{C} \leftarrow \mathit{LLSV}(\mathcal{Y}, \mathbf{C}, 3)$
  end for
  $\mathcal{G} \leftarrow \mathcal{Y} \times_3 C^\intercal$
  return $[\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$
end function

Optimization 2: Subspace Iterations

function HOSI($\mathcal{X}$, **r**)
  Initialize A, B, C randomly
  for Max Iterations **do**
    $\mathcal{Y} = \mathcal{X} \times_2 \mathbf{B}^\intercal \times_3 \mathbf{C}^\intercal$
    $\mathbf{A} \leftarrow \mathit{LLSV}(\mathcal{Y}, \mathbf{A}, 1)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_3 \mathbf{C}^\intercal$
    $\mathbf{B} \leftarrow \mathit{LLSV}(\mathcal{Y}, \mathbf{B}, 2)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_2 \mathbf{B}^\intercal$
    $\mathbf{C} \leftarrow \mathit{LLSV}(\mathcal{Y}, \mathbf{C}, 3)$
  end for
  $\mathcal{G} \leftarrow \mathcal{Y} \times_3 C^\intercal$
  return $[\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$
end function

function $U = \mathrm{LLSV}(\mathcal{Y}, \mathbf{U}, n)$
  $\mathcal{G} = \mathcal{Y} \times_n \mathbf{U}^\intercal$
  $\mathbf{U} = \mathrm{Contract}(\mathcal{Y}, \mathcal{G}, n)$ ▷ TTT
  $[\mathbf{U}, \sim] = \mathrm{qr}(\mathbf{U})$
end function

function $U = \mathrm{LLSV}(\mathbf{Y}, r \text{ or } \epsilon)$
  $\mathbf{S} = \mathbf{Y} \cdot \mathbf{Y}^\intercal$
  $[\mathbf{U}, \Lambda] = \mathrm{eig}(\mathbf{S})$
  return $\mathbf{U}(:, 1 : r)$
end function

Higher Order Subspace Iteration with Dimension Tree (HOSI-DT)

Input: Tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$
Ranks $\mathbf{r} = r_1, \ldots, r_d$
Output: TTensor $\mathcal{T}$ of ranks $\mathbf{r}$ with $\mathcal{T} \approx \mathcal{X}$
function HOSI-DT($\mathcal{X}$, $\mathbf{U}$, $\mathbf{m}$, $\mathbf{r}$)
  if length($\mathbf{m}$) == 1 **then**
    $\mathcal{G} = \mathcal{X} \times_m \mathbf{U}_m^\intercal$ ▷ Update Core
    $\mathbf{U}_m = \mathbf{Y}_{(m)} \cdot \mathbf{G}_{(m)}^\intercal$ ▷ Contract Two Tensors on Mode k
    $[\mathbf{U}_m, \tilde{}] = \text{qr}(\mathbf{U}_m)$ ▷ Orthogonalize Factor Matrix
  else
    Equally partition m $= [\mu, \eta]$
    $\mathcal{X}_{\text{left}} = \mathcal{X} \times_i \mathbf{U}_i, \forall i \in \eta$
    $[\mathcal{G}, \mathbf{U}] = \text{HOSI-DT}(\mathcal{X}_{\text{left}}, \mathbf{U}, \mu, \mathbf{r})$ ▷ Left Recursion
    $\mathcal{X}_{\text{right}} = \mathcal{X} \times_i \mathbf{U}_i, \forall i \in \mu$
    $[\mathcal{G}, \mathbf{U}] = \text{HOSI-DT}(\mathcal{X}_{\text{right}}, \mathbf{U}, \eta, \mathbf{r})$ ▷ Right Recursion
  end if
end function

Transforming HOOI into an error-specified algorithm

$$\min ||\mathcal{X} - \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}_3||$$
$$\text{subject to } \mathcal{G} \in \mathbb{R}^{q \times r \times s}, \mathbf{A} \in \mathbb{R}^{m \times q}, \mathbf{B} \in \mathbb{R}^{n \times r}, \mathbf{C} \in \mathbb{R}^{p \times s}$$

Now suppose we have a relative error tolerance of how accurate we want our approximation to be. Then the approximation must satisfy:

$$\frac{||\mathcal{X} - \mathcal{T}||}{||\mathcal{X}||} \leq \epsilon$$
$$||\mathcal{X} - \mathcal{T}||^2 \leq \epsilon^2 \cdot ||\mathcal{X}||^2$$
$$||\mathcal{X}||^2 - ||\mathcal{G}||^2 \leq \epsilon^2 \cdot ||\mathcal{X}||^2$$
$$(1 - \epsilon^2) \cdot ||\mathcal{X}||^2 \leq ||\mathcal{G}||^2$$

Transforming HOOI into an error-specified algorithm

$$\min ||\mathcal{X} - \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}_3|| \tag{1}$$
$$\text{subject to } \mathcal{G} \in \mathbb{R}^{q \times r \times s}, \mathbf{A} \in \mathbb{R}^{m \times q}, \mathbf{B} \in \mathbb{R}^{n \times r}, \mathbf{C} \in \mathbb{R}^{p \times s}$$

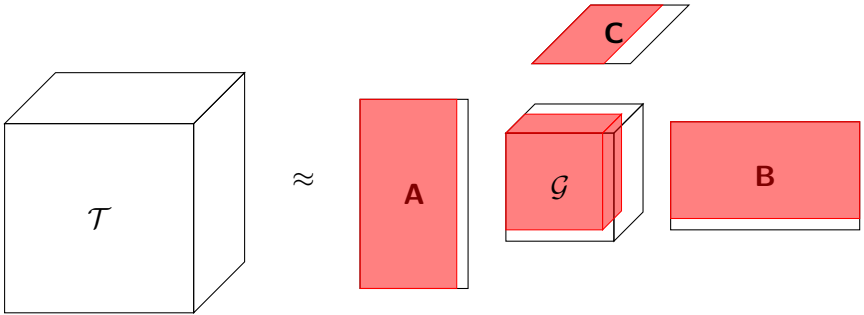
Now suppose we have a relative error tolerance of how accurate we want our approximation to be. Then the approximation must satisfy:

$$(1 - \epsilon^2) \cdot ||\mathcal{X}||^2 \leq ||\mathcal{G}||^2 \tag{2}$$

Then we know if our tucker tensor satisfies the error tolerance solely on $||\mathcal{G}||^2$. If the error tolerance is satisfied, we can find a smaller tucker representation that still satisfies the error by analyzing the core

Visualizing Adaptive HOOI

$$\min_{\mathbf{r}} ||\mathcal{G}(1:\mathbf{r})||^2 \tag{3}$$
$$\text{subject to } ||\mathcal{G}(1:\mathbf{r})||^2 \geq (1-\epsilon^2)||\mathcal{X}||^2$$



Adaptive HOOI

function AdaptiveHOOI($\mathcal{X}$, $\mathbf{r}$, ϵ)
  Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$ randomly
  for Maximum Number of Iterations **do**
    $\mathcal{Y} = \mathcal{X} \times_2 \mathbf{B}^\intercal \times_3 \mathbf{C}^\intercal$
    $A \leftarrow \text{LLSV}(Y_{(1)}, r_1)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_3 \mathbf{C}^\intercal$
    $B \leftarrow \text{LLSV}(Y_{(2)}, r_2)$

    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^\intercal \times_2 \mathbf{B}^\intercal$
    $C \leftarrow \text{LLSV}(Y_{(3)}, r_3)$
    $\mathcal{G} \leftarrow \mathcal{Y} \times_3 C^\intercal$
    $\mathbf{r} = \text{performCoreAnalysis}(\mathcal{G}, \epsilon, \mathbf{r})$
  end for
  return $[\![\mathcal{G}, \mathbf{U}_{1:d}]\!]$
end function

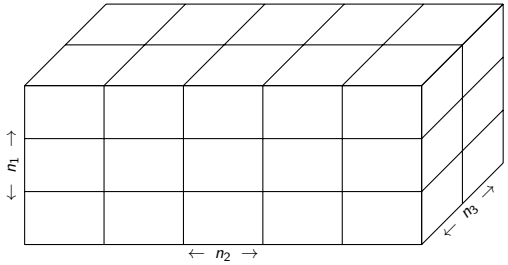
function performCoreAnalysis($\mathcal{G}, \epsilon, \mathbf{r}$)
  if $||\mathcal{G}||^2 \geq (1-\epsilon^2)||\mathcal{X}||^2$ **then**
    Find $\mathbf{r} = \arg\min ||\mathcal{G}(1:\mathbf{r})||^2$
      subject to $||\mathcal{G}(1:\mathbf{r})||^2 \geq (1-\epsilon^2)||\mathcal{X}||^2$

    Truncate $\mathcal{G}, \mathbf{A}, \mathbf{B}, \mathbf{C}$ according to $\mathbf{r}$
  else
    $\mathbf{r} = \alpha\, \mathbf{r}$
    Increase columns of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ according to $\mathbf{r}$
  end if
  return $\mathbf{r}$
end function

TuckerMPI and Parallel Tensor Distribution

- Existing C++/MPI library
- Implements deterministic STHOSVD
- Has efficient sequential and parallel kernels for SVD and TTM

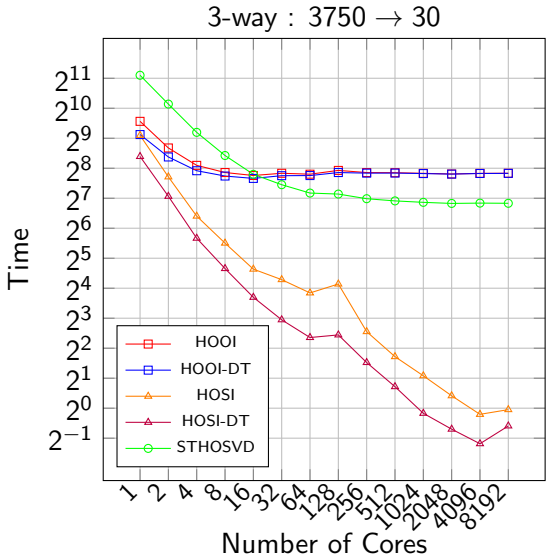
For d-way tensor, we use d-way processor grid with Cartesian block distribution

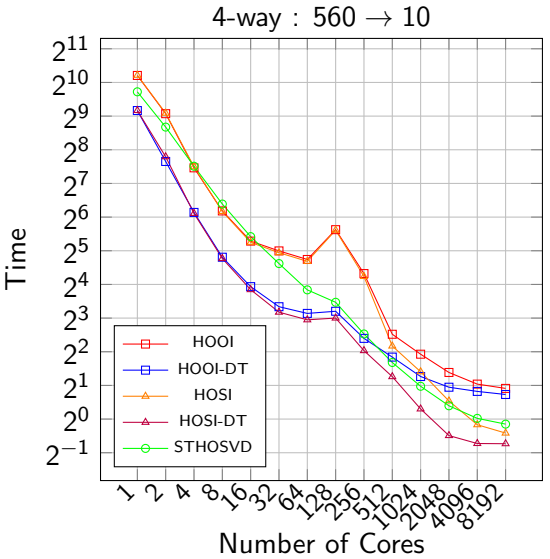


Example: $p_1 \times p_2 \times p_3 = 3 \times 5 \times 2$

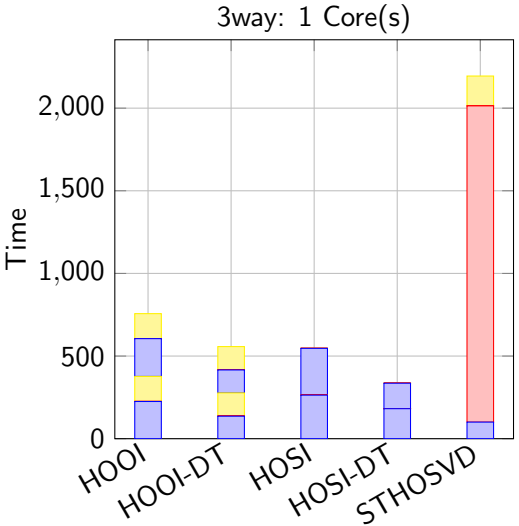
Local tensor size: $\frac{n_1}{p_1} \times \frac{n_2}{p_2} \times \frac{n_3}{p_3}$

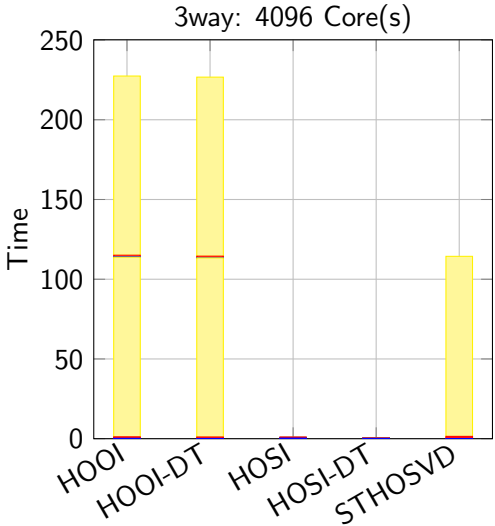
Parallel Scaling of Synthetic Data in Single Precision



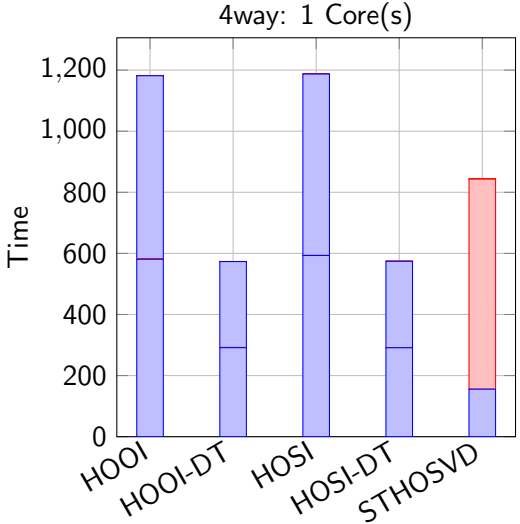


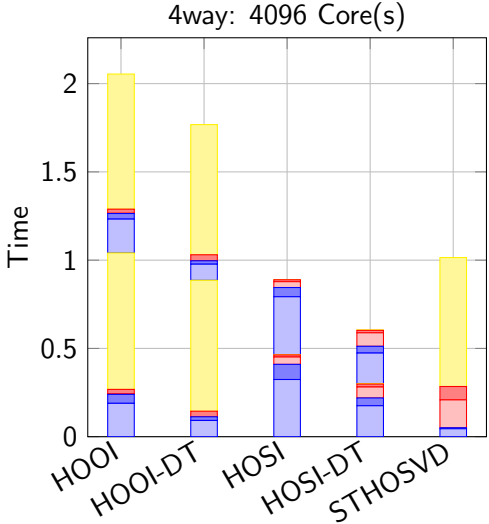
Breakdown of 3-way Parallel Scaling of Synthetic Data in Single Precision



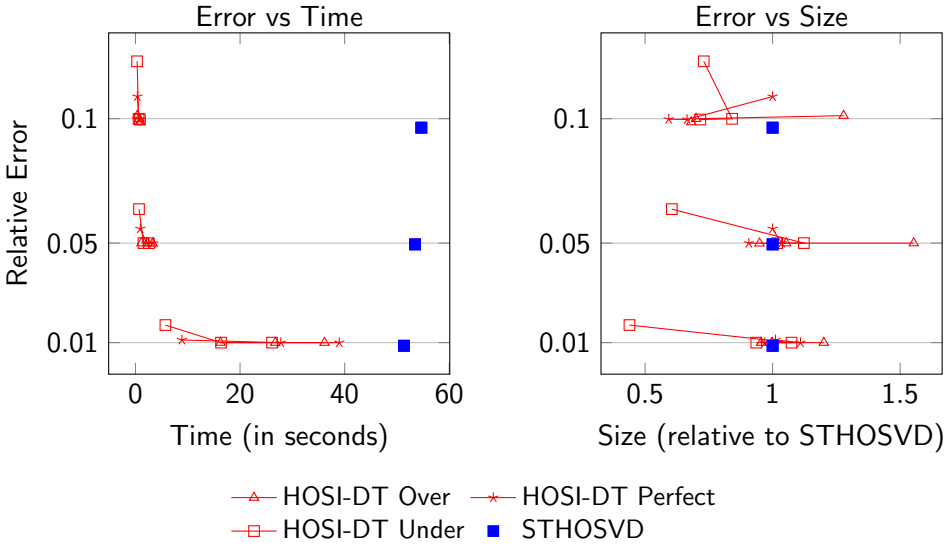


Breakdown of 4-way Parallel Scaling of Synthetic Data in Single Precision

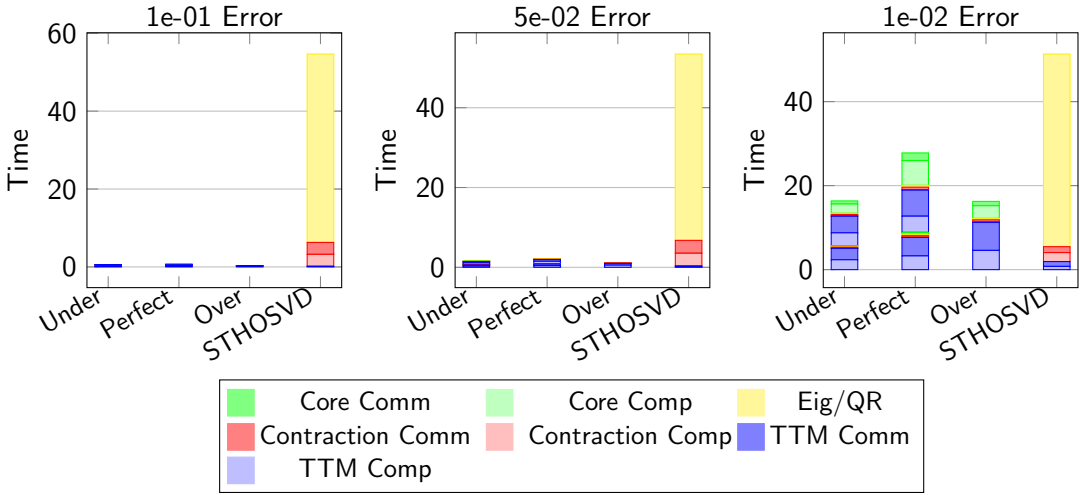




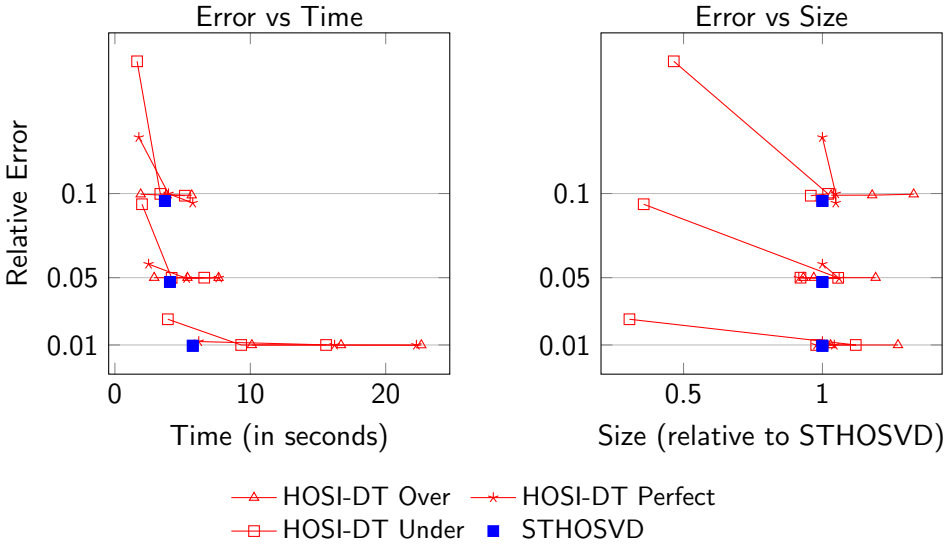
Miranda Data Set ($3072 \times 3072 \times 3072$) - 1024 Cores - 115 GB



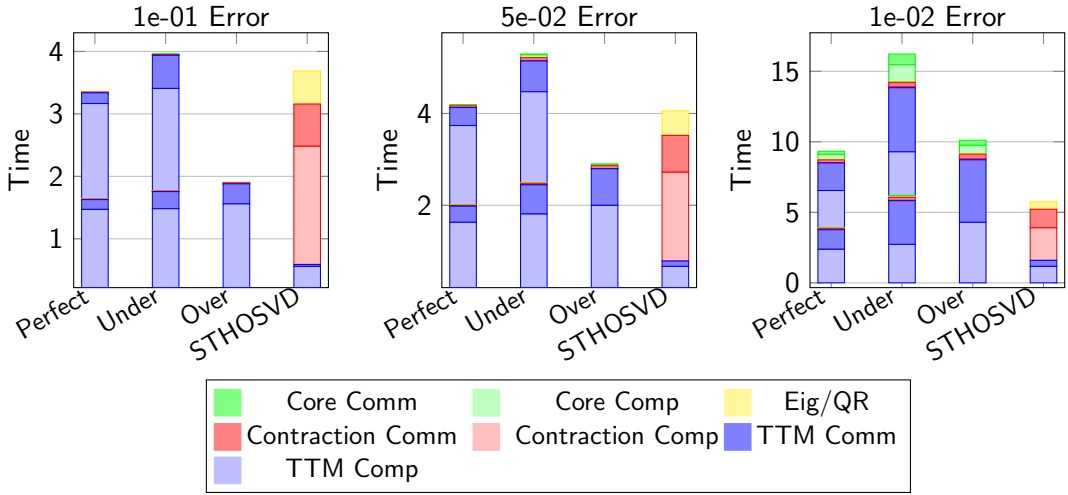
Miranda Data Set (3072 × 3072 × 3072) - 1024 Cores - 115 GB



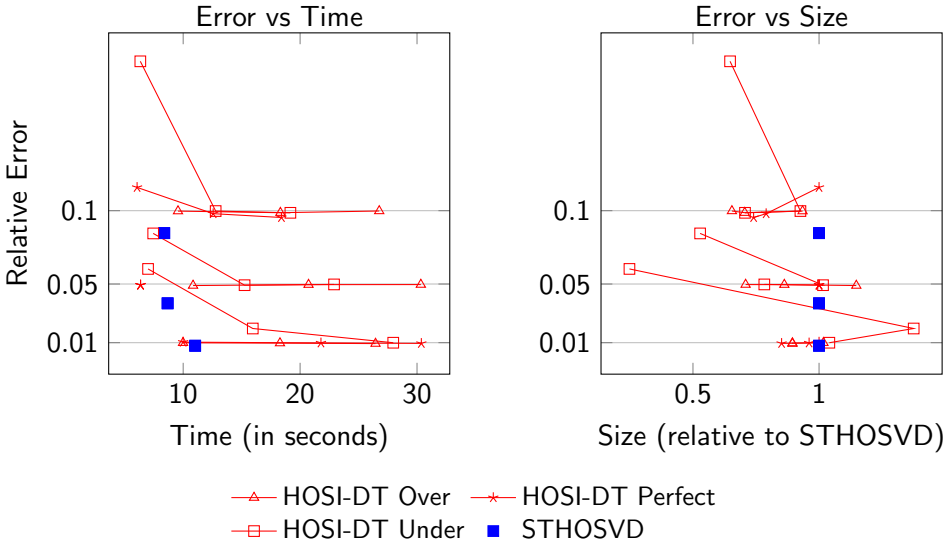
HCCI Data Set ($672 \times 672 \times 33 \times 626$) - 128 Cores - 75 GB



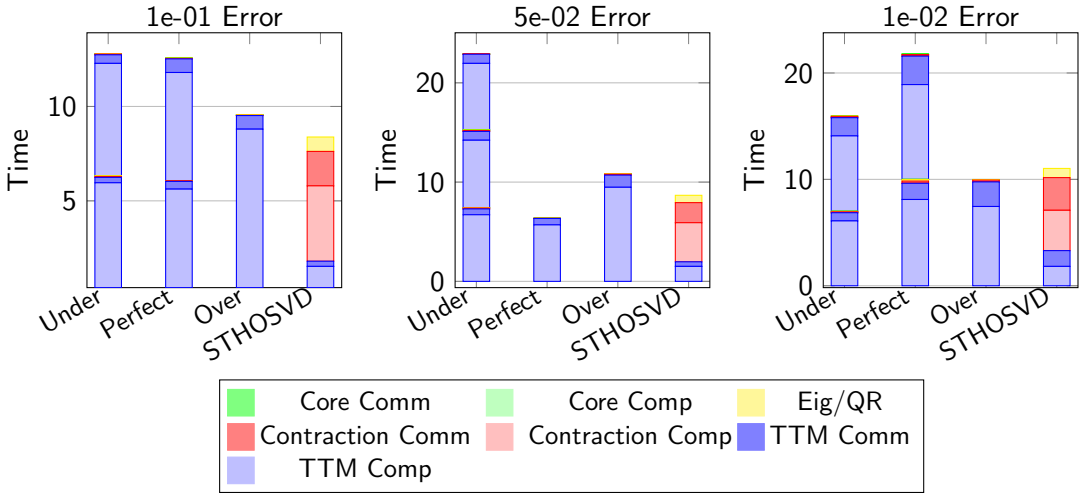
HCCI Data Set ($672 \times 672 \times 33 \times 626$) - 128 Cores - 75 GB



SP Data Set ($500 \times 500 \times 500 \times 33 \times 400$) - 2048 Cores - 4.4 TB



SP Data Set ($500 \times 500 \times 500 \times 33 \times 400$) - 2048 Cores - 4.4 TB



Conclusions - Project 2

- Dimension Tree optimizes TTM operations on HOOI
- Subspace Iteration optimizes LLSV operations on HOOI
- AdaptiveHOOI removes rank-specified constraint on HOOI
- `https://gitlab.com/tensors/TuckerMPI`
- This work will be soon be published to SC25 two days ago


MSC


CSC 112


MST 117


CS Dept.


CSC 355
CSC 753


Research


MST 121


Hero


CSC 250ish


STA 312


CSC 301
CSC 790
CSC 726


CSC 721
Committee


CSC 201
CSC 643


UKE 790


MST 311


MST 225
Committee

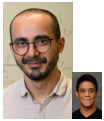

Committee


CSC 673


MST 331


CSC 111


CSC 250
CSC 641


MST 333


MTH 351


CSC 326


MST 113


MTH 624