Application of Tensor Computations:Solving Higher Dimensional PDEs and Color Video Compression

Suvendu Kar
Indian Institute of Science
Bengaluru, KA, India
suvendukar@iisc.ac.in

April 22, 2024

Abstract

Matrix-matrix and matrix-vector multiplications became incredibly powerful in the current era of artificial intelligence and deep learning.One example is that when we examine the inner workings of deep neural networks, we find that they are nothing more than a series of matrix and vector computations, even whether we are working with speech, big-data, images, multimodal AI,etc.. It is always highly desired that matrix calculation complexity be reduced.

Since more grid points are needed in the computational domain to resolve every structure that appears in the solution of PDEs, the traditional matrix- and vector-based technique is problematic to address higher dimensional problems with localized structures or sharp transitions.Tensors, which are only multidimensional generalizations of vectors and matrices, make it simple to get around this problem.**In this work, we will focus on doing easy to compute tensor operations for solving higher dimensional PDE(specifically 2D,3D Poisson Equations) in the framework of Einstein Product.Moreover I will compare Pseudoinverse computation based direct solver with Gauss Siedel based iterative solver for solving induced tensor-way system of equations.**

I will also implement PINN for solving 2D Poisson Equation.

In my previous wor$k$[19], I presented t-CUR-based tensor completion for image recovery. In this work, I will provide t-CUR-based color video recovery, which is a broad extension of that work.Moreover I will provide m-CUR based video reconstruction hereby.

Index Terms: PINNs, Data recovery, tensor completion, tensor-CUR decomposition,Einstein-SVD

1 Introduction

Higher dimensional PDEs are frequently solved in Applied Physics, Computational Mathematics, and Fluid Dynamics, and most of the time a closed form solution is not available. In situations like this, numerical techniques are quite dependable. To gather precise data regarding the actual solution Finite Element Based techniques employ a large number of grid points, which typically increases the computational difficulty of the underlying matrix-based calculation for solving system of equations.This work's major goal is to explore high-dimensional Poisson problems within the context of multilinear systems. We now offer a productive method for encoding the Laplacian matrix into a domain with tensor structure. In order to facilitate the integration of boundary conditions into the multilinear system, this will create an effective process for computing solutions on the computational grid.

Additionally, we tackle the problem of missing data recovery in this work as a 3-way tensor completion problem toward the application of video recovery, taking inspiration from other successful tensor pattern implementations.CP based decomposition requires to know (CP rank:The CP rank is defined by the minimum number of rank-one terms in CP decomposition) rank beforehand (calculating rank is NP complete).Moreover, the best rank-K approximation to a tensor may not always exist in the CP approach. The tensor-SVD algorithm is computationally expensive, since it requires computing the SVD decomposition of the approximate matrix at each iteration of the optimization.

Our suggested method(T and M product product bases CUR decomposition i.e. t-CUR, and m-CUR), which computes the low rank approximation of a given tensor using the tensor's actual rows and columns, extends matrix-CUR decomposition

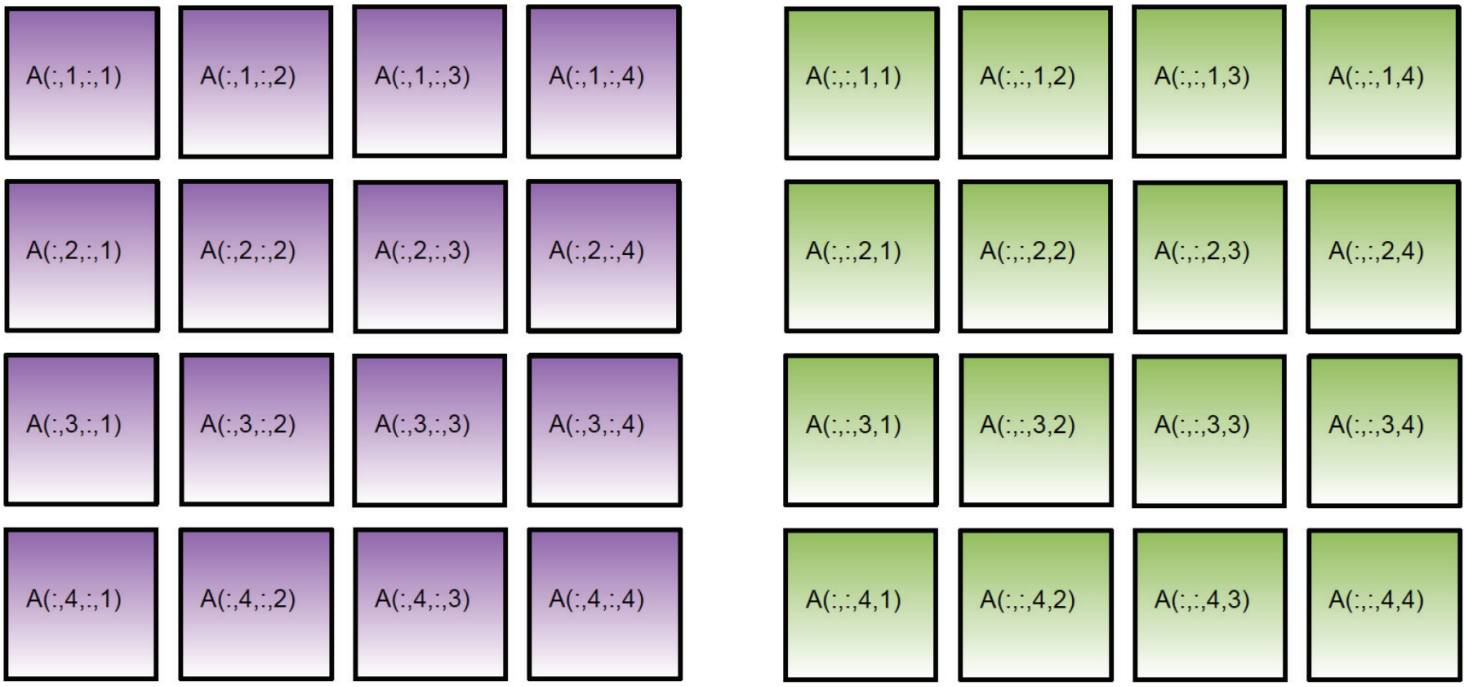
to tensor. Because it simply has to resolve a typical regression problem, it is computationally efficient. Furthermore, our suggested method effectively compute the low rank approximation of a given tensor utilizing the tensor's real rows and columns without requiring knowledge of the rank beforehand.

2 (Answer for Question 1)Tensor Based Solution of Numerical PDE

As our discussion will be based on Einstein product, let me first briefly introduce Einstein product before getting into our actual problem of solving 2D ,and 3D Poisson equation.

Definition 1.1 (tensor blocks of N th order tensor)
Given a tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with $N \geq 4$ with partition $(J_1, J_2, \ldots, J_N)$ on the index sets $I_1, I_2, \ldots, I_N$, a tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is partitioned into $j_1 j_2 \ldots j_N$ number of tensor blocks of size $J_1 \times J_2 \times \cdots \times J_N$.

Here we discuss some relevant tensor blocks in this work. Given a fourth order tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$ with partition $(1, 1, I_3, I_4), \mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$ is partitioned into matrix blocks of size $I_1 \times I_2$. Each block is denoted by $\mathbf{A}^{(3,4)}_{i_3,i_4} = \mathscr{A}(:,:,i_3,i_4) \in \mathbb{R}^{I_1 \times I_2}$ with $i_3 = 1, \ldots, I_3$ and $i_4 = 1, \ldots, I_4$.For visualization below at right-green picture has been drawn with $I_3 = I_4 = 4$ Similarly, given a fourth order tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$ with partition $(I_1, I_2, 1, 1), \mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$ is partitioned into matrix blocks $\mathbf{A}^{(1,2)}_{i_1,i_2}$ of size $I_3 \times I_4$. Now given a sixth order tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5 \times I_6}$ with partition $(1, I_2, 1, I_4, 1, I_6), \mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5 \times I_6}$ is partitioned into tensor blocks $\mathscr{A}^{(2,4,6)}_{i_2 i_4, i_6}$ of size $I_1 \times I_3 \times I_5$.



Definition 1.2 (Tensor multiplication (Einstein product) [2])
$\mathbb{C}$ refers for field of complex numbers .For $\mathscr{A} \in \mathbb{C}^{I_1 \times \cdots \times I_N \times K_1 \times \cdots \times K_N}$ and $\mathscr{B} \in \mathbb{C}^{K_1 \times \cdots \times K_N \times J_1 \times \cdots \times J_M}$, the Einstein product of tensors $\mathscr{A}$ and $\mathscr{B}$ is denoted by $\mathscr{A} *_N \mathscr{B}$ and defined elementwise as

$$(\mathscr{A} *_N \mathscr{B})_{i_1 \ldots i_N j_1 \ldots j_M} = \sum_{k_1 \ldots k_N} a_{i_1 \ldots i_N k_1 \ldots k_N} b_{k_1 \ldots k_N j_1 \ldots j_M}.$$

Definition 1.3 (Transpose)
The transpose of $\mathscr{A} \in \mathbb{R}^{I \times J \times I \times J}$ is a tensor $\mathscr{B}$ which has entries $b_{ijkl} = a_{klij}$. We denote the transpose of $\mathscr{A}$ as $\mathscr{B} = \mathscr{A}^T$. If $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N \times J_1 \times \ldots \times J_N}$, then $(\mathscr{B})_{i_1, i_2, \ldots, i_n, j_1, j_2, \ldots, j_n} = (\mathscr{A})^T_{j_1, j_2, \ldots, j_n, i_1, i_2, \ldots, i_n}$ is the transpose of $\mathscr{A}$.

Definition 1.4 (Symmetric tensor)
A tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N \times J_1 \times \ldots \times J_N}$ is symmetric if $\mathscr{A} = \mathscr{A}^T$, that is, $a_{i_1, i_2, \ldots, i_n, j_1, j_2, \ldots, j_n} = a_{j_1, j_2, \ldots, j_n, i_1, i_2, \ldots, i_n}$.

Definition 1.5 (Identity tensor)
The identity tensor $\mathscr{I}$ is

$$\mathscr{I}_{i_1 i_2 j_1 j_2} = \delta_{i_1 j_1} \delta_{i_2 j_2}, \text{ where } \delta_{lk} = \begin{cases} 1, & l = k \\ 0, & l \neq k. \end{cases}$$

It generalizes to a $2N$ th order identity tensor,

$$(\mathscr{I})_{i_1 i_2 \ldots i_N j_1 j_2 \ldots j_N} = \prod_{k=1}^{N} \delta_{i_k j_k}.$$

Definition 1.6 Diagonal Tensor
A tensor $\mathscr{D} = \left(d_{\mathbf{i}(N),\mathbf{j}(N)}\right) \in \mathbb{C}^{\mathbf{I}(N)\times\mathbf{I}(N)}$ is called a diagonal tensor if all its entries are zero except for $d_{\mathbf{i}(N),\mathbf{i}(N)}$.

The definition of an upper off-diagonal tensor and lower off-diagonal tensor are defined as follows.
Definition 1.7 (Upper off-diagonal tensor).
A tensor $\mathscr{U} = \left(u_{\mathbf{i}(N),\mathbf{j}(N)}\right) \in \mathbb{C}^{\mathbf{I}(N)\times\mathbf{I}(N)}$ is called an upper off-diagonal tensor if all entries below the main diagonal are zero, i.e., $u_{\mathbf{i}(N),\mathbf{j}(N)} = 0$ for $j_k < i_k$, where $k = 1, 2, \cdots N$.
Definition 1.8 (Lower off-diagonal tensor).
A tensor $\mathscr{L} = \left(l_{\mathbf{i}(N),\mathbf{j}(N)}\right) \in \mathbb{C}^{\mathbf{I}(N)\times\mathbf{I}(N)}$ is called a lower off-diagonal tensor if all entries above the main diagonal are zero, i.e., $l_{\mathbf{i}(N),\mathbf{j}(N)} = 0$ for $i_k < j_k$, where $k = 1, 2, \cdots N$.

Using the notation of diagonal tensor, we define below the diagonal dominant tensor.
Definition 1.9 (Diagonally dominant tensor).
A tensor $\mathscr{A} = \left(a_{\mathbf{i}(N),\mathbf{j}(N)}\right) \in \mathbb{C}^{\mathbf{I}(N)\times\mathbf{I}(N)}$ is called diagonally dominant if

$$\left|a_{\mathbf{i}(N),\mathbf{i}(N)}\right| \geq \sum_{\substack{\mathbf{j}(N)\neq\mathbf{i}(N)\\ \mathbf{j}(N)}} \left|a_{\mathbf{i}(N),\mathbf{j}(N)}\right|.$$

If we replace the condition ' $\geq$ ' by ' $>$ ', then we call strictly diagonally dominant.
Definition 1.10 (Eigen Value and Eigen Vectors)
Let $\mathscr{A} \in \mathbb{C}^{\mathbf{I}(N)\times\mathbf{I}(N)}$. A complex number $\lambda \in \mathbb{C}$ is called an eigenvalue of $\mathscr{A}$ if there exist some nonzero tensor $\mathscr{X} \in \mathbb{C}^{\mathbf{I}(N)}$ such that $\mathscr{A} *_N \mathscr{X} = \lambda\mathscr{X}$.

The nonzero tensor $\mathscr{X}$ is called eigenvector of $\mathscr{A}$ and we define the spectral radius $\rho(\mathscr{A})$ of $\mathscr{A}$, be the largest absolute value of the eigenvalues of $\mathscr{A}$.
As a consequence of the above Definition 1.10, the following lemma easily holds.
Lemma 1.11 Let $\mathscr{A} \in \mathbb{C}^{I(N)\times I(N)}$. If λ is an eigenvalue of $\mathscr{A}$, then for $m \in \mathbb{N}, \lambda^m$ is an eigenvalue of $\mathscr{A}^m$. For a tensor $\mathscr{A} = \left(a_{\mathbf{i}(N),\mathbf{j}(N)}\right) \in \mathbb{C}^{\mathbf{I}(N)\times\mathbf{I}(N)}$, we define $\lim_{k\to\infty} \mathscr{A}^k = \lim_{k\to\infty} \left[\left(\mathscr{A}^k\right)_{\mathbf{i}(N),\mathbf{j}(N)}\right]$.
In view of this, we next present the definition of convergent tensor.
Definition 1.12 (Convergent tensor). A tensor $\mathscr{A} \in \mathbb{C}^{\mathbf{I}(N)\times\mathbf{I}(N)}$ is called convergent tensor if $\mathscr{A}^k \to \mathscr{O}$ as $k \to \infty$.

We now introduce the definition of convergence of a power series of tensors, which is a generalization of the power series in matrices.
Lemma 1.13 Let $\mathscr{A} \in \mathbb{C}^{\mathbf{I}(N)\times\mathbf{I}(N)}$. The series $\sum_{k=0}^{\infty} c_k\mathscr{A}^k$ is convergent if $\sum_{k=0}^{\infty} c_k \left(\mathscr{A}^k\right)_{\mathbf{i}(N),\mathbf{j}(N)}$ is convergent for every $\mathbf{i}(N)$ and $\mathbf{j}(N)$.

Let us now focus on Tensor group set-up.
Definition 1.14 (Transformation).
Define the transformation $f : \mathbb{T}_{I_1,I_2,I_1,I_2}(\mathbb{R}) \longrightarrow \mathbb{M}_{I_1I_2,I_1I_2}(\mathbb{R})$ with $f(\mathscr{A}) = \mathbf{A}$ is defined component-wise as

$$(\mathscr{A})_{i_1 i_2 i_1 i_2} \xrightarrow{f} (\mathbf{A})_{[i_1+(i_2-1)I_1][i_1+(i_2-1)I_1]}$$

where $\mathscr{A} \in \mathbb{T}_{I_1,I_2,I_1,I_2}(\mathbb{R})$ and $\mathbf{A} \in \mathbb{M}_{I_1I_2,I_1I_2}(\mathbb{R})$.
Lemma 1.15
Let f be the map defined in (1.14). Then the following properties hold:

i) The map f is a bijection. Moreover, there exists a bijective inverse map $f^{-1} : \mathbb{M}_{I_1I_2,I_1I_2}(\mathbb{R}) \to \mathbb{T}_{I_1,I_2,I_1,I_2}(\mathbb{R})$.

ii) The map satisfies $f(\mathscr{A} *_2 \mathscr{B}) = f(\mathscr{A}) \cdot f(\mathscr{B})$, where $\cdot$ refers to the usual matrix multiplication.
Proof of i). According to the definition of f, we can define a map $h : I_1 \times I_2 \to I_1I_2$ by $h(i_1, i_2) = i_1 + (i_2 - 1) I_1$ where $I_k = \{1, \ldots, I_k\}$ and $I_kI_l = \{1, \ldots, I_kI_l\}$. Clearly, the map h is a bijection so it follows f is a bijection.
Proof of ii). Since f is a bijection, for some $1 \leq i, j \leq I_1I_2$, there exists unique i_1, i_2, j_1, j_2, for $1 \leq i_1, j_1 \leq I_1, 1 \leq i_2, j_2 \leq I_2$ such

that $(i_2-1)\,I_1+i_1=i$, and $(j_2-1)\,I_1+j_1=j$. So,

$$[f\,(\mathscr{A}*_2\mathscr{B})]_{ij}=(\mathscr{A}*_2\mathscr{B})_{i_1i_2j_1j_2}=\sum_{u,v}a_{i_1i_2uv}b_{uvj_1j_2}$$

$$[f(\mathscr{A})\cdot f(\mathscr{B})]_{ij}=\sum_{r=1}^{I_1I_2}[f(\mathscr{A})]_{ir}[f(\mathscr{B})]_{rj}.$$

For every $1\le r\le I_1I_2$, there exists unique u,v such that $(u-1)I_1+v=r$. So,

$$\sum_{u,v}a_{i_1i_2uv}b_{uvj_1j_2}=\sum_{r=1}^{I_1I_2}[f(\mathscr{A})]_{ir}[f(\mathscr{B})]_{rj}.$$

It follows from the properties of f that the Einstein product can be defined through the transformation:

$$\mathscr{A}*_2\mathscr{B}=f^{-1}\left[f\,(\mathscr{A}*_2\mathscr{B})\right]=f^{-1}[f(\mathscr{A})\cdot f(\mathscr{B})].$$

Consequently, the inverse map f^{-1} satisfies

$$f^{-1}(\mathbf{A}\cdot\mathbf{B})=f^{-1}(\mathbf{A})*_2\,f^{-1}(\mathbf{B}).$$

Recall that a subset of $\mathbb{M}_{N,N}(\mathbb{R})$ consisting of all invertible $N\times N$ matrices with the matrix multiplication is a group. Let the subset $\mathbb{M}\subset\mathbb{M}_{I_1I_2,I_1I_2}(\mathbb{R})$ contain all invertible $I_1I_2\times I_1I_2$. Define $\mathbb{T}=\{\mathscr{T}\in\mathbb{T}_{I_1,I_2,I_1,I_2}(\mathbb{R}):\det(f(\mathscr{T}))\neq 0\}$ where $\mathbf{T}\in\mathbb{M}$ with $\mathbf{T}=f(\mathscr{T})$

Theorem 1.16.

Suppose $(\mathbb{M},\cdot)$ is a group. Let $f:\mathbb{T}\to\mathbb{M}$ be any bijection. Then we can define a group structure on $\mathbb{T}$ by defining

$$\mathscr{A}*_2\mathscr{B}=f^{-1}[f(\mathscr{A})\cdot f(\mathscr{B})]$$

for all $\mathscr{A},\mathscr{B}\in\mathbb{T}$. In other words, the binary operation $*_2$ satisfies the group axioms. Moreover, the mapping f is an isomorphism.

Corollary 1.17

Define $\widetilde{\mathbb{T}}=\{\mathscr{T}\in\mathbb{T}_{I_1,\ldots,I_N,J_1,\ldots,J_N}(\mathbb{R}),\det(f(\mathscr{T}))\neq 0\}$ with $I_m=J_m$ for for m = 1, . . . ,N. Then the ordered pair $\widetilde{\mathbb{T}}$ is a group where the operation $*_N$ is the generalized Einstein product in ((1.2)).

As we will use Einstein-product based SVD to find the pseudoinverse while solving the the system of equations $\mathscr{A}*_N\mathbb{X}=\mathscr{B}$, induced from the Poisson equation, let us define SVD here.

Theorem 1.18 (Singular value decomposition (SVD)).

Let $\mathscr{A}\in\mathbb{R}^{I\times J\times I\times J}$ with $R=\operatorname{rank}(f(\mathscr{A}))$. The singular value decomposition for tensor $\mathscr{A}$ has the form

$$\mathscr{A}=\mathscr{U}*_2\mathscr{D}*_2\mathscr{V}^T$$

where $\mathscr{U}\in\mathbb{R}^{I\times J\times I\times J}$ and $\mathscr{V}\in\mathbb{R}^{I\times J\times I\times J}$ are orthogonal tensors and $\mathscr{D}\in\mathbb{R}^{I\times J\times I\times J}$ is a diagonal tensor with entries σ_{ijij} called singular values. Moreover, the decomposition (2.8) can be written as a sum of fourth-order tensors

$$\mathscr{A}=\sum_{kl}\sum_{ij,\hat{i}\hat{j}}\sigma_{klkl}\left(\mathbf{U}^{(3,4)}_{kl}\right)_{ij}\circ\left(\mathbf{V}^{(3,4)}_{kl}\right)_{\hat{i}\hat{j}},$$

where $\circ$ denotes the outer product. The matrices $\mathbf{U}^{(3,4)}_{ij}$ and $\mathbf{V}^{(3,4)}_{ij}$ are called left and right singular matrices.

Proof. Let $\mathbf{A}=f(\mathscr{A})$. From the isomorphic property and Theorem (2.21), we have

$$\mathbf{A}=\mathbf{U}\cdot\mathbf{D}\cdot\mathbf{V}^T\xrightarrow[f^{-1}]{}A=U*_2D*_2V^T,$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\mathbf{D}$ is a diagonal matrix. In addition, $\mathbf{U}\cdot\mathbf{U}^T=\mathbf{I}$ and $\mathbf{V}\cdot\mathbf{V}^T=\mathbf{I}\xrightarrow{f^{-1}}\mathscr{U}^T*_2\mathscr{U}=\mathscr{I}$ and $\mathscr{V}^T*\mathscr{V}=\mathscr{I}$.

In higher dimensional tensors we will use matlab inbuilt command "*reshape*" to have equivalent homeomorphism as f (defined above).

2.1 Problem Formulation

Poisson problems in a tensor structured domain[21]:

$$-\nabla^2 u = f \text{ in } \Omega$$
$$u = 0 \text{ on } \partial\Omega,$$

where the source function f is given on a d-dimensional domain denoted by $\Omega \subset \mathbb{R}^d$ and solution u satisfy boundary conditions on the boundary $\partial\Omega$ of Ω. The discretized d-dimensional Poisson equation can be written as follows:

$$Ax = b,$$

where the matrix $A \in \mathbb{R}^{n^d \times n^d}$, the vectors $x \in \mathbb{R}^{n^d \times 1}$, and $b \in \mathbb{R}^{n^d \times 1}$. Because more grid points must be included in the computational domain in order to resolve every structure that appears in the solution, the standard matrix- and vector-based technique is impracticable for solving higher dimensional problems with localized structures or abrupt transitions. For example, consider dimension $d = 15$ and number of interior grid points $n = 1000$ with Dirichlet boundary condition, then we obtain matrix $A \in \mathbb{R}^{10^{45} \times 10^{45}}$, vectors $x \in \mathbb{R}^{10^{45} \times 1}$ and $b \in \mathbb{R}^{10^{45} \times 1}$. Tensors, which are only multidimensional generalizations of vectors and matrices, make it simple to get around this problem.

This section focuses on high-dimensional Poisson problems within the paradigm of multilinear systems. We now offer a productive method for encoding the Laplacian matrix into a domain with tensor structure. By doing this, an effective method for calculating solutions on the computational grid will be developed, making it easier to integrate boundary conditions into the multilinear system. Before discussing the general construction of d-dimensional Poisson problems in a tensor structured domain, it will appropriate, to begin with, the two-dimensional Poisson problem:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x,y) \text{ in } \Omega. - - - - - - - - - - - - - (1.1)$$
$$u = 0 \text{ on } \partial\Omega.$$

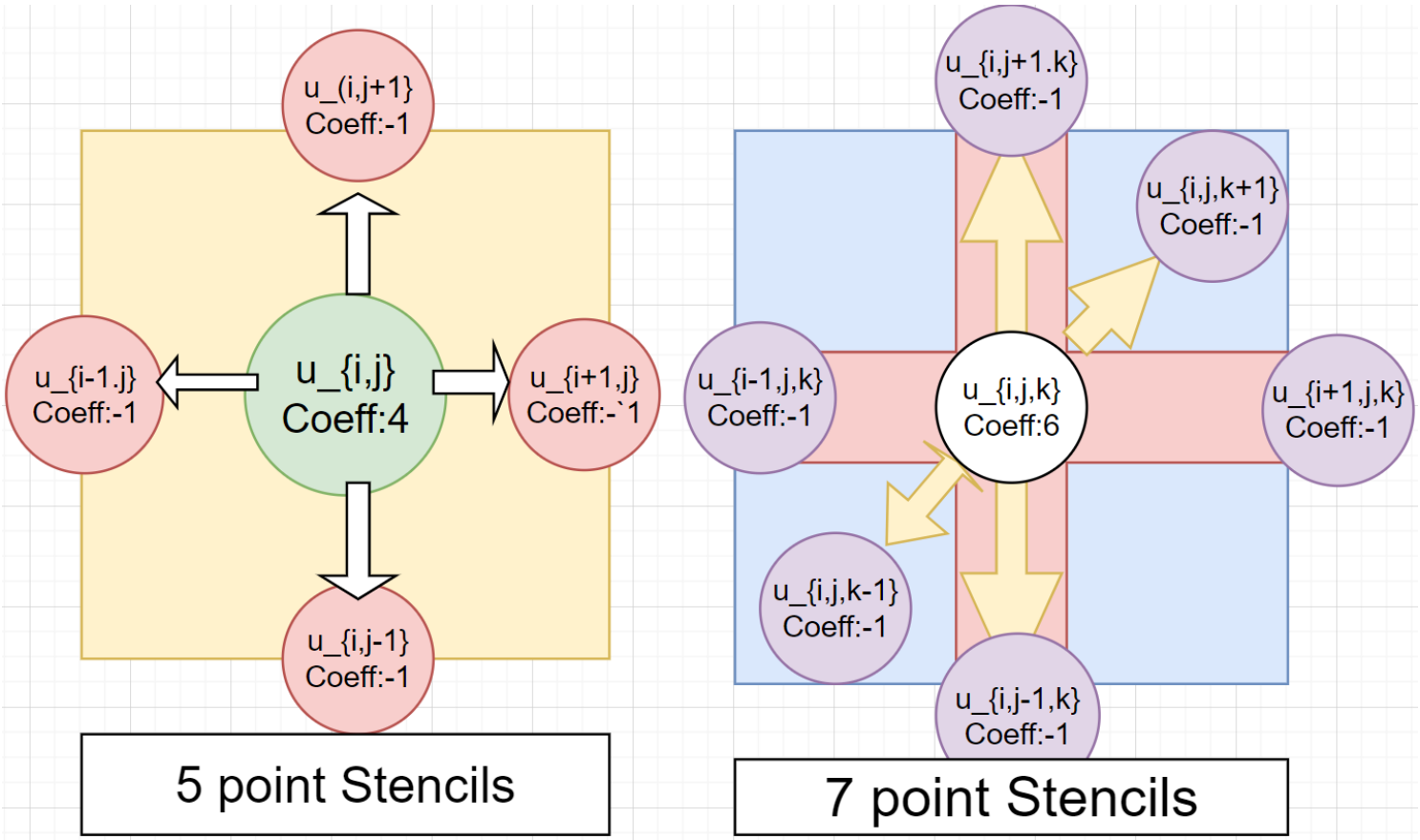


FIG 1.1: mStencils for higher order tensors. The main node of the five-point and seven-point stencils sits on the diagonal entries of fourth order and sixth order Laplacian tensors, respectively.

Let us consider f(x,y)=$2 * \pi^2 * sin(\pi * x) sin(\pi * y)$, and also let say domain is $\Omega \cup \partial\Omega = [0,1] \times [0,1]$.

Several problems in physics and mechanics are modeled by above defined 2D Poisson Equation, where the solution u represents,for example, temperature, electromagnetic potential, or displacement of an elastic membrane fixed at the boundary.The mesh points are obtained by discretizing the unit square domain with step sizes, Δx in the x-direction and Δy in the y-direction. From the standard central difference approximations, the difference formula,

$$\frac{u_{l-1,m} - 2u_{l,m} + u_{l+1,m}}{\Delta x^2} + \frac{u_{l,m-1} - 2u_{l,m} + u_{l,m+1}}{\Delta y^2} = f\left(x_l, y_m\right),$$

is obtained. If we assume $\Delta x = \Delta y$, then the above difference equation is equivalent to

$$\mathbf{A_N U} + \mathbf{U A_N} = (\Delta x)^2 \mathbf{F},$$

[N is the number of inner grid points (i.e. excluding boundary points 0,1) along individual axes] where

$$\mathbf{A_N} = \begin{bmatrix} 2 & -1 & & 0 \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{bmatrix}_{N\times N} \quad ----------(M1),$$

$$\mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1N} \\ u_{21} & u_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{N-1N} \\ u_{N1} & \cdots & u_{NN-1} & u_{NN} \end{bmatrix}, \quad \text{and}$$

$$\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1N} \\ f_{21} & f_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & f_{N-1N} \\ f_{N1} & \cdots & f_{NN-1} & f_{NN} \end{bmatrix}.$$

The entries of $\mathbf{U}$ and $\mathbf{F}$ are the values on the mesh on the unit square where, $(x_i, y_j) = (i\Delta x, j\Delta x) \in [0,1] \times [0,1]$. Here the Dirichlet boundary conditions are imposed so the values of $\mathbf{U}$ are zero at the boundary of the unit square, i.e., $u_{i0} = u_{iN+1} = u_{0,j} = u_{N+1j} = 0$ for $0 < i, j < N+1$[considering the indexing starting from 0, though indexing starts at 1 for matlab.We have taken care of this fact at the time of implementation.]
Typically, $\mathbf{U}$ and $\mathbf{F}$ are vectorized which gives the following linear system:

$$(M2) ----- \mathbf{A_{N^2\times N^2}} \cdot \mathbf{u} = \begin{bmatrix} \mathbf{A_N} + 2\mathbf{I_N} & -\mathbf{I_N} & & 0 \\ -\mathbf{I_N} & \mathbf{A_N} + 2\mathbf{I_N} & \ddots & \\ & \ddots & \ddots & -\mathbf{I_N} \\ 0 & & -\mathbf{I_N} & \mathbf{A_N} + 2\mathbf{I_N} \end{bmatrix} \begin{bmatrix} u_{11} \\ u_{12} \\ \vdots \\ u_{NN} \end{bmatrix}$$
$$= (\Delta x)^2 \begin{bmatrix} f_{11} \\ f_{12} \\ \vdots \\ f_{NN} \end{bmatrix}.$$

Poisson's equation in two dimensions is expressed as a sum of Kronecker products [2], i.e.,

$$\mathbf{A_{N^2\times N^2}} = \mathbf{I_N} \otimes \mathbf{A_N} + \mathbf{A_N} \otimes \mathbf{I_N}.$$

Moreover, the discretized problem in three dimensions is

$$(1.2)\left(\mathbf{A_N} \otimes \mathbf{I_N} \otimes \mathbf{I_N} + \mathbf{I_N} \otimes \mathbf{A_N} \otimes \mathbf{I_N} + \mathbf{I_N} \otimes \mathbf{I_N} \otimes \mathbf{A_N}\right) \cdot \mathrm{vec}(\mathbf{V}) = (\Delta x)^3 vec(\mathbf{F})$$

The higher order tensor representation of the 2D discretized Poisson problem (1.1) is

$$\mathscr{A}_N *_2 \mathbf{U} = \mathbf{F} ----------(1.3),$$

where $\mathscr{A}_N \in \mathbb{R}^{N\times N\times N\times N}$ and matrices $\mathbf{V} \in \mathbb{R}^{N\times N}$ and $\mathbf{F} \in \mathbb{R}^{N\times N}$ are the discretized functions u and f on a unit square mesh defined in above formulation. The nonzeros entries of the matrix block $\mathbf{A}_{\mathbf{N}}{}^{2,4}_{\alpha,\beta} \in \mathbb{R}^{N\times N}$ are the followings,

$$\begin{cases} \left(\mathbf{A}_{\mathbf{N}}^{(2,4)}\right)_{\alpha,\beta} = \frac{4}{(\Delta x)^2} \\ \left(\mathbf{A}_{\mathbf{N},\beta}^{(2,4)}\right)_{\alpha-1\beta} = \frac{-1}{(\Delta x)^2} \\ \left(\mathbf{A}_{\mathbf{N}_{\alpha,\beta}}^{(2,4)}\right)_{\alpha+1,\beta} = \frac{-1}{(\Delta x)^2} \\ \left(\mathbf{A}_{\mathbf{N}}^{(2,4)}\right)_{\alpha,\beta-1} = \frac{-1}{(\Delta x)^2} \\ \left(\mathbf{A}_{\mathbf{N},\beta}^{(2,4)}\right)_{\alpha,\beta+1} = \frac{-1}{(\Delta x)^2} \end{cases}$$

for $\alpha,\beta = 2,\ldots,N-1$. These entries form a five-point stencil; as shown Figure 1.1. The discretized three-dimensional Poisson equation is

$$\overline{\mathscr{A}}_N *_3 \mathscr{U} = \mathscr{F} \text{ - - - - - - - - - - - - - - - - } (1.4),$$

where $\overline{\mathscr{A}}_N \in \mathbb{R}^{N\times N\times N\times N\times N\times N}$ and $\mathscr{U},\mathscr{F} \in \mathbb{R}^{N\times N\times N}$. Both $\mathscr{U}$ and $\mathscr{F}$ are discretized on the unit cube. The entries on the tensor block $\left(\overline{\mathscr{A}}_N\right)^{(2,4,6)}_{l,m,n} \in \mathbb{R}^{N\times N\times N}$ of $\overline{\mathscr{A}}_N$ would follow a seven-point stencil, i.e.,

$$\begin{cases} \left(\left(\overline{\mathscr{A}}_N\right)^{(2,4,6)}_{\alpha,\beta,\gamma}\right)_{\alpha,\beta,\gamma} = \frac{6}{(\Delta x)^2}, \\ \left(\left(\overline{\mathscr{A}}_N\right)^{(2,4,6)}_{\alpha,\beta,\gamma}\right)_{\alpha-1,\beta,\gamma} = \frac{-1}{(\Delta x)^2}, \\ \left(\left(\overline{\mathscr{A}}_N\right)^{(2,4,6)}_{\alpha,\beta,\gamma}\right)_{\alpha+1,\beta,\gamma} = \frac{-1}{(\Delta x)^2}, \\ \left(\left(\overline{\mathscr{A}}_N\right)^{(2,4,6)}_{\alpha,\beta,\gamma}\right)_{\alpha,\beta-1,\gamma} = \frac{-1}{(\Delta x)^2}, \\ \left(\left(\overline{\mathscr{A}}_N\right)^{(2,4,6)}_{\alpha,\beta,\gamma}\right)_{\alpha,\beta+1,\gamma} = \frac{-1}{(\Delta x)^2}, \\ \left(\left(\overline{\mathscr{A}}_N\right)^{(2,4,6)}_{\alpha,\beta,\gamma}\right)_{\alpha,\beta,\gamma-1} = \frac{-1}{(\Delta x)^2}, \\ \left(\left(\overline{\mathscr{A}}_N\right)^{(2,4,6)}_{\alpha,\beta,\gamma}\right)_{\alpha,\beta,\gamma+1} = \frac{-1}{(\Delta x)^2} \end{cases}$$

for $\alpha,\beta,\gamma = 2,..,N-1$ since u_{ijk} satisfies :

$$6u_{ijk} - u_{i-1,j,k} - u_{i+1,j,k} - u_{i,j-1,k} - u_{i,j+1,k} - u_{i,j,k-1} - u_{i,j,k+1} = (\Delta x)^2 f_{ijk}$$

The tensor representation of Poisson problems in 2D,3D consists of the multilinear systems (1.3) and (1.4) respectively. We found Pseudoinverse of $\mathscr{A}$,$\mathscr{A}_{pinv} = \mathbf{V_{inv}} *_n \mathbf{S_{inv}} *_n \mathbf{U_{inv}}$[n=2 for 2D PDE, and n=3 for 3D PDE, as shown in following algorithm 1.1,1.2] to find the solution $\mathbf{U} = \mathscr{A}_{pinv} *_n \mathbf{F}$.————————$(M3)$

Listing 1: Algorithm 1.1: Pseudo inverse of 4th order tensor

```
function [U_inv, S_inv, V_inv]=PINV4(A)
    % We will find the Pseudo inverse of 4th order tensor A
    szA=size(A); % Finding the size of A
    % Reducing A to the equivalent matrix B through the homeomorphism 'reshape'
    B=reshape(A,[szA(1)*szA(2),szA(3)*szA(4)]);
    [Um,Sm,Vm]=svd(B); % Getting the SVD of B

    % Getting the pseudoinverse Sm\_pinv of diagonal matrix Sm,
    for i=1:min(size(Sm,1),size(Sm,2))
        if Sm(i,i) ~=0
            Sm(i,i)=1/Sm(i,i);
        else
            Sm(i,i)=Sm(i,i);
        end
    end

    % If SVD decomposition of B=Um*Sm*Vm', then its pseudoinverse is
    % B_pinv=Vm*Sm_pinv*Um'
```

```
    % Computing Um' to find pseudoinverse of B
    Um=Um';

    % Using Homeomorphism operation "reshape" getting U_inv,V_inv,S_inv such
    % that Pseudoinverse of A=V_inv *_2 S_inv *_2 U_inv
    U_inv=reshape(Um,[szA(3),szA(4),szA(1),szA(2)]);
    V_inv=reshape(Vm,[szA(3),szA(4),szA(1),szA(2)]);
    S_inv=reshape(Sm,[szA(1),szA(2),szA(3),szA(4)]);
end
```

Listing 2: Algorithm 1.2: Pseudo inverse of 6th order tensor

```
function [U_inv,S_inv,V_inv]=PINV6(A)
    % We will find the Pseudo inverse of 6th order tensor A
    szA=size(A); % Finding the size of A
    % Reducing A to the equivalent matrix B through the homeomorphism 'reshape'
    B=reshape(A,[szA(1)*szA(2)*szA(3),szA(4)*szA(5)*szA(6)]);
    [Um,Sm,Vm]=svd(B); % Getting the SVD of B

    % Getting the pseudoinverse Sm\_pinv of diagonal matrix Sm,
    for i=1:min(size(Sm,1),size(Sm,2))
        if Sm(i,i) ~=0
            Sm(i,i)=1/Sm(i,i);
        else
            Sm(i,i)=Sm(i,i);
        end
    end

    % If SVD decomposition of B=Um*Sm*Vm', then its pseudoinverse is
    % B_pinv=Vm*Sm_pinv*Um'

    % Computing Um' to find pseudoinverse of B
    Um=Um';

    % Using Homeomorphism operation "reshape" getting U\_inv,V\_inv,S\_inv such
    % that Pseudoinverse of A=V_inv *_3 S_inv *_3 U_inv
    U_inv=reshape(Um,[szA(4),szA(5),szA(6),szA(1),szA(2),szA(3)]);
    V_inv=reshape(Vm,[szA(4),szA(5),szA(6),szA(1),szA(2),szA(3)]);
    S_inv=reshape(Sm,[szA(1),szA(2),szA(3),szA(4),szA(5),szA(6)]);
end
```

Higher-Order Jacobi Method
Given $A \in \mathbb{R}^{N\times N\times N\times N}, B \in \mathbb{R}^{N\times N}, MAX$
Initial guess $X^0 \in \mathbb{R}^{N\times N}$
for $k = 1$ to MAX
for $i = 1$ to N
for $j = 1$ to N
$X_{ij}^{(k+1)} = (B_{ij} - \sum_{i'j'\neq ij}(A_{iji'j'}X_{i'j'}^{(k)}))/A_{ijij}$
end
end
end

2.2 Advantages in Computing in Tensor Domain

There are some advantages in computing in a tensor structured domain.[20] The convergence of Jacobi is slow since the spectral radius with respect to the Poisson's equation is near one .Also, the approximation in Figures 3,7(b) are first order accurate.
First
without vectorization, the solution and the computational grid have a one-to-one correspondence so that sophisticated boundary conditions are easily integrated in the multilinear system.

Second,
the Laplacian tensor preserves the low bandwidth since the main nodal points sit on the tensor diagonal entries and the rest of the stencil points lie on the off-diagonal positions. Although the Laplacian matrices in (M1) and (M2) are banded, the Laplacian matrices in higher dimensions have larger bandwidths. The Laplacian tensor has a lower bandwidth than the Laplacian matrix. In fact, reducing the bandwidth of these sparse matrices directly and substantially improves the number of operations and storage locations; reordering methods of Cuthill and McKee [22] and George and Liu [23] can be found for reference.

2.3 Observations

Poisson 2D Problem
Using the method, discussed in (M3), as well as the Gauss Jacobi Method , we solved the 2D Poisson's equation on $[0,1] \times [0,1]$:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = 2 * \pi^2 * sin(\pi * x) * sin(\pi * y) \text{ in } \Omega.$$
$$u = 0 \text{ on } \partial\Omega.$$

- For number of inner-grid points N=[10,20,30,40,50,60,70,80], we found that:
  Error($E1$=Frobenius norm of ($\mathscr{A} *_2 \mathbf{X}_{predicted} - \mathbf{F}$)) for N=10,20,30,40,50,60,70, and 80 are : 1.0e-12 *[0.0236 , 0.0617 , 0.1182 , 0.1953 , 0.2545 , 0.3507 ,0.7838],respectively.

- Using Gauss Jacobi method, with N=15, and iteration count=[10,20,30,40,50,60,70,100,200,500,1000,2000,4000], the error E1's are: [0.42364, 0.29004,0.19799,0.13484,0.091688, 0.062283,0.042284, 0.013211,0.00027278,2.3998e-09,2.181e-15,2.181e-15,2.181e-15]

- When the error $E2$ is Frobenius norm of $u_{true} - u_{pred}$, it is(N=number of inner grid points) : Error between numerical and true solutions for N=10: 0.037538
  Error between numerical and true solutions for N=20: 0.019604
  Error between numerical and true solutions for N=30: 0.013272
  Error between numerical and true solutions for N=40: 0.010033
  Error between numerical and true solutions for N=50: 0.0080649
  Error between numerical and true solutions for N=60: 0.0067424
  Error between numerical and true solutions for N=70: 0.0057926
  Error between numerical and true solutions for N=80:0.0050773

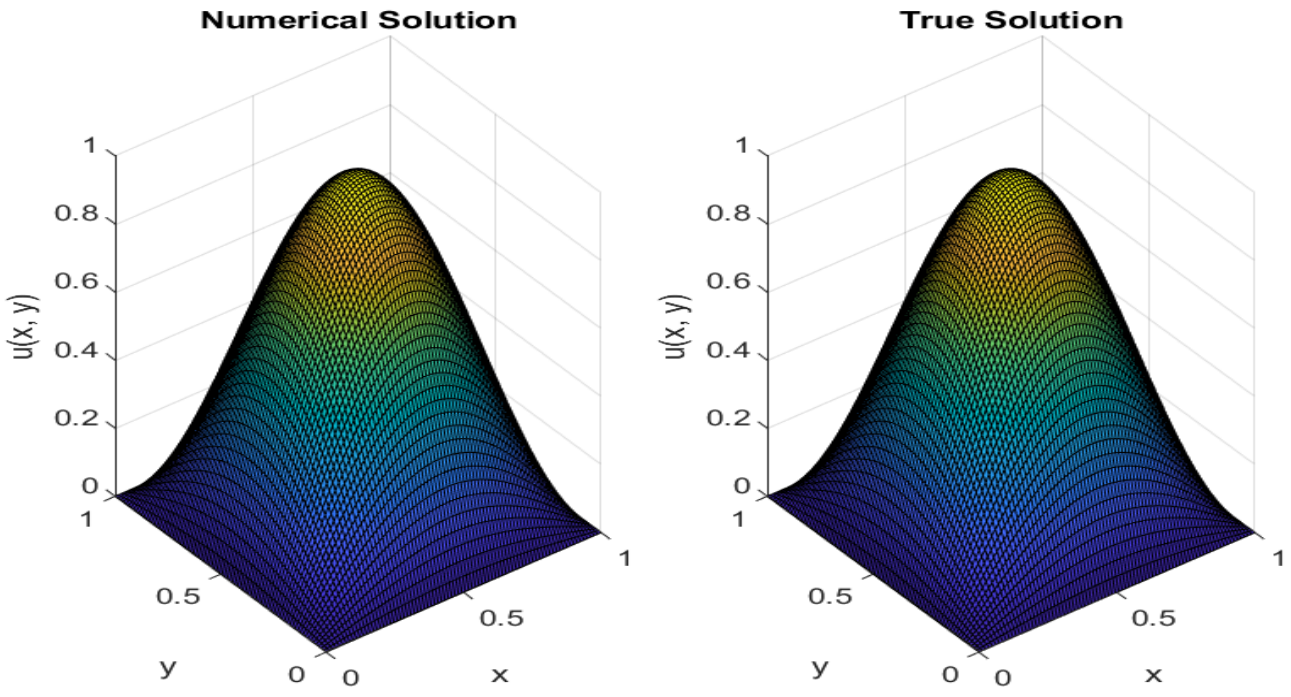


Figure 1: True solution vs Predicted Solution for N(number of inner grid points in each axes)=80, with Pseudoinverse based 2D PDE Solver

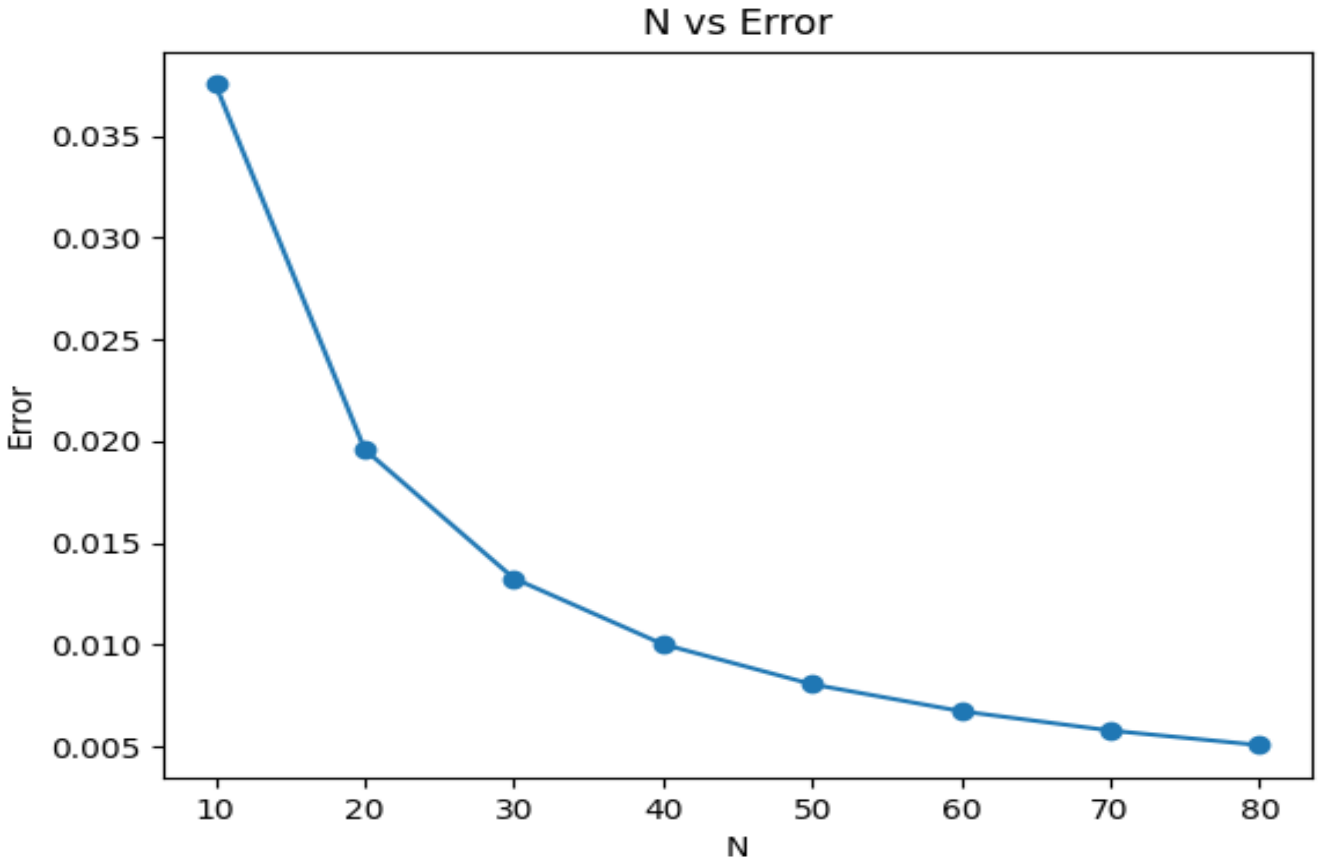


Figure 2: N vs Error E2 for Pseudoinverse based 2D Poisson Equation Solver

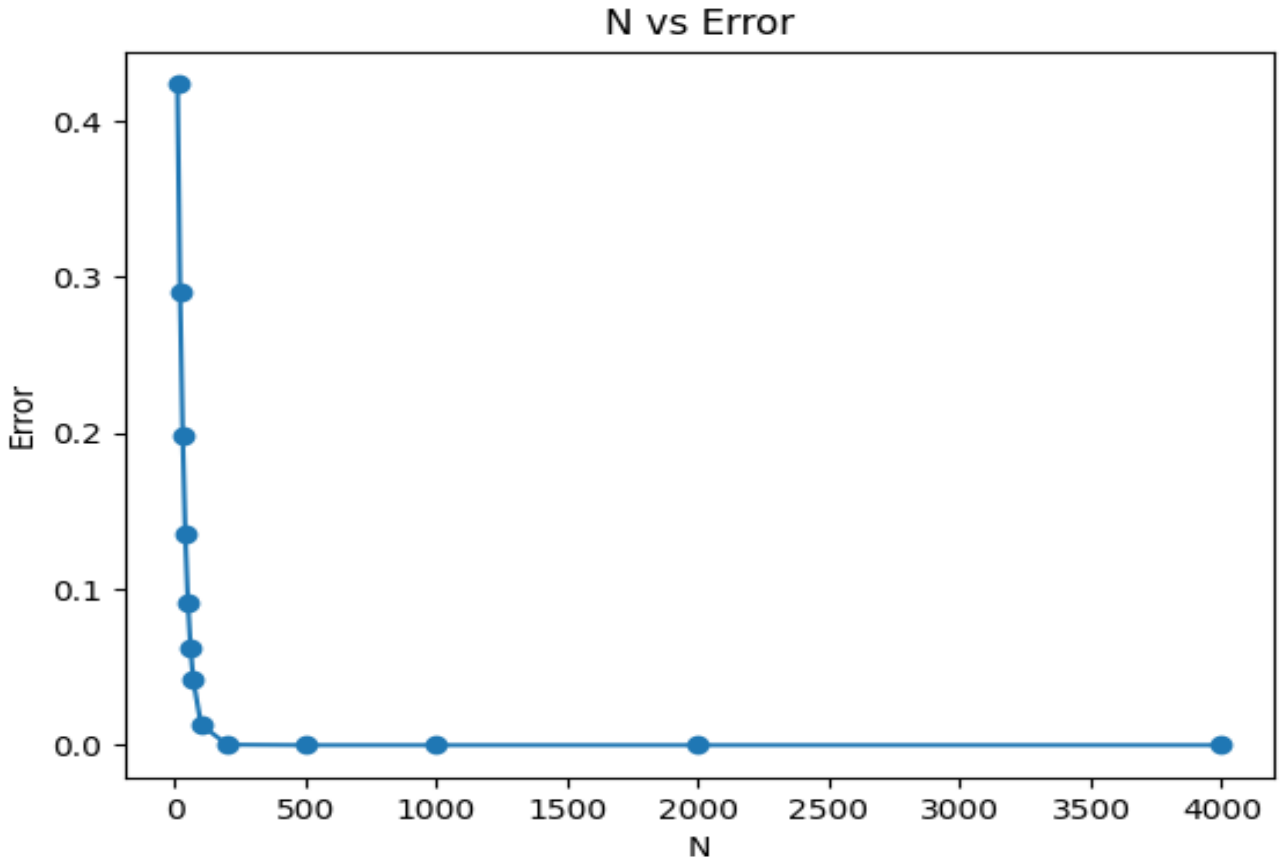


Figure 3: Iteration count vs Error E1 with Jacobi Method for 2D Poisson Problem

Poisson 3D Problem

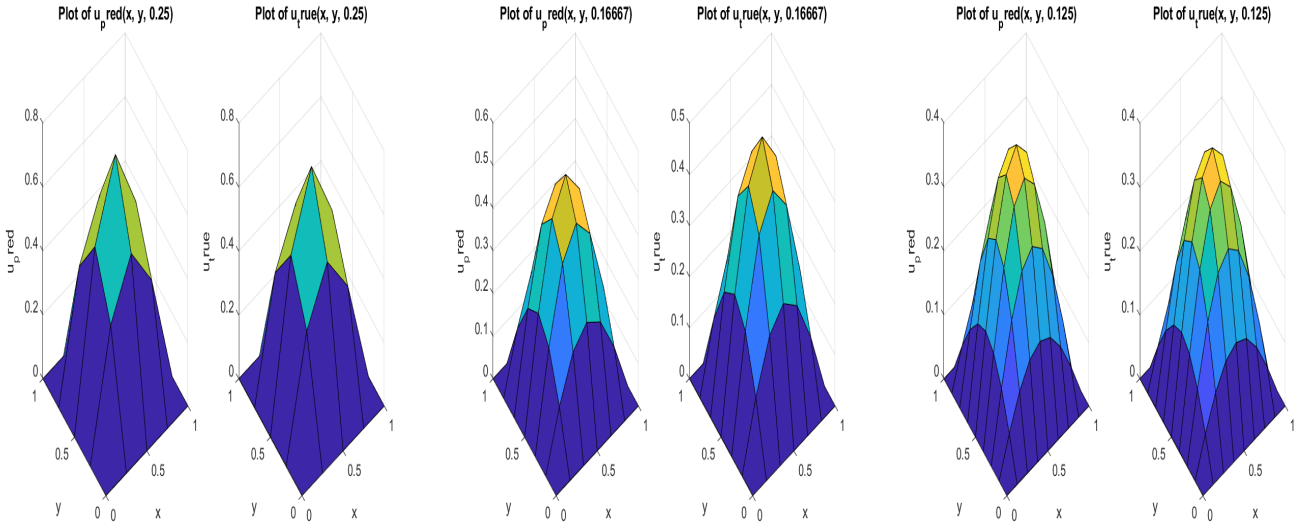
Using the method, discussed in(M3), as well as Gauss Jacobi method , we solved the 3D Poisson's equation on $[0,1]\times[0,1]\times[0,1]$:

$$-\frac{\partial^2 u}{\partial x^2}-\frac{\partial^2 u}{\partial y^2}-\frac{\partial^2 u}{\partial z^2}=3*\pi^2*sin(\pi*x)*sin(\pi*y)*sin(\pi*z) \text{ in } \Omega.$$
$$u=0 \text{ on } \partial\Omega.$$

- For number of inner-grid points N = [3,5,7,9,10,13,15,18], we found that:
  Error E2 between numerical and true solutions[0.14999,0.12036,0.10361,0.09241,0.088034,0.077912,0.072837 ,0.066802] .

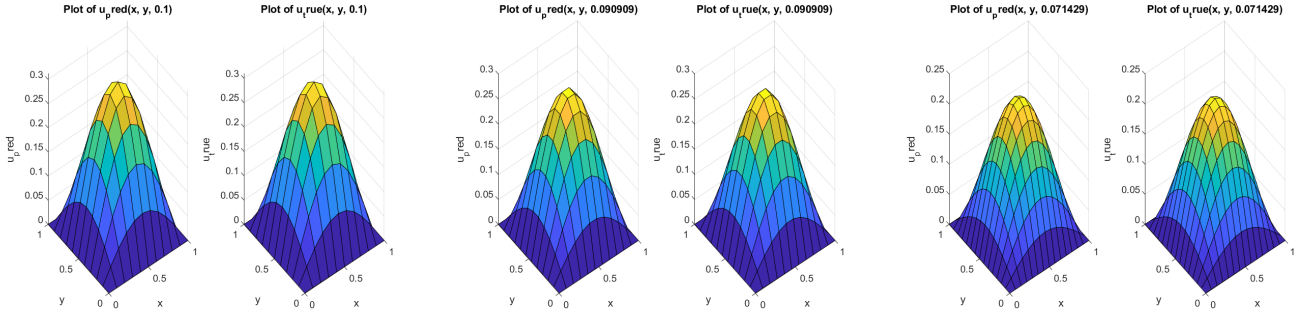
- With iteration count=[10,20,30,40,50,60,70,100,200,500,1000,2000]E1 error with Gauss Jacobi method for N=15, [1.7916,1.2247, 0.83597,0.56977,0.38786,0.26376,0.17924, 0.056104, 0.0011595,1.0201e-08,1.5551e-14,1.5551e-14]

- E1 error using Einstein product based SVD and Pseudoinverse for N=[3,5,7,9,10,13] is :Error between numerical and true solutions for N=3: 1.2909e-14
  Error between numerical and true solutions for N=5: 3.5238e-14
  Error between numerical and true solutions for N=7: 6.1338e-14
  Error between numerical and true solutions for N=9: 1.106e-13
  Error between numerical and true solutions for N=10: 1.4206e-13
  Error between numerical and true solutions for N=13:2.8328e-13



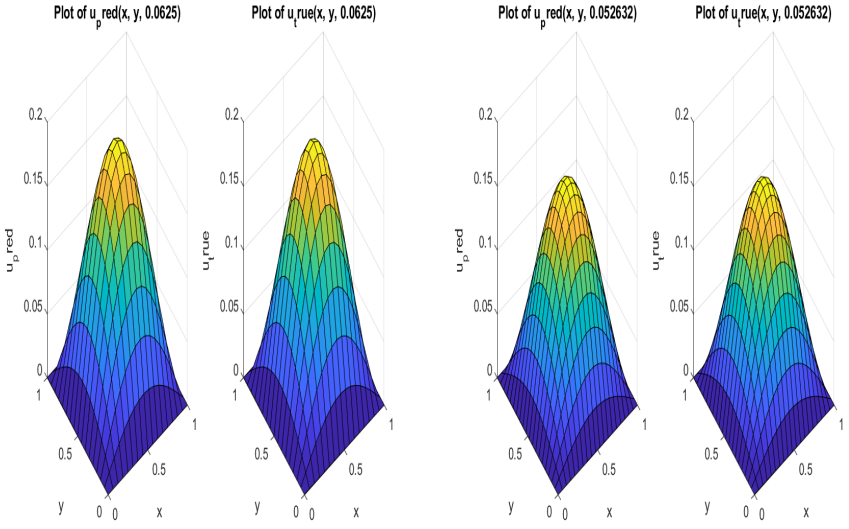
(a) a (b) b (c) c

Figure 4: True vs Predicted solution(Pseudoinverse based) for N=3(a),5(b),7(c), with fixed z values, for 3D Poisson Problem



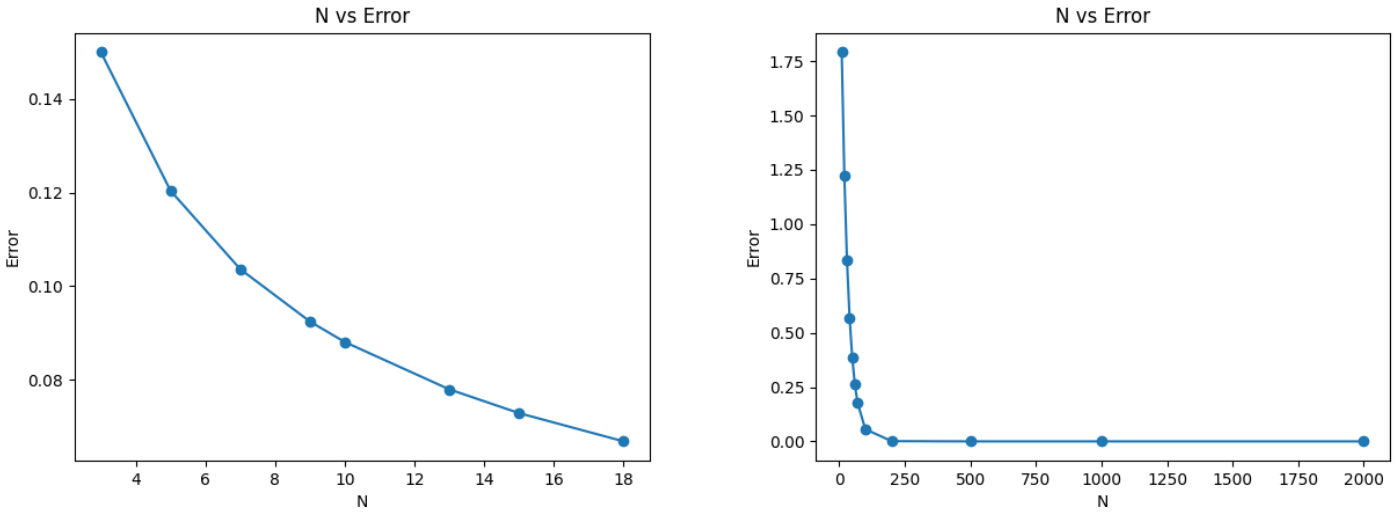
(a) a (b) b (c) c

Figure 5: True vs Predicted solution(Pseudoinverse based) for N=9(a),10(b),13(c), with fixed z values for 3D Poisson Equation



(a) a (b) b

Figure 6: True vs Predicted solution(Pseudoinverse Based) for N=15(a),18(b) with fixed z values for 3D Poisson Equation



(a) N vs E2 Error with Pseudoinverse method for 3D Poisson Equation (b) Iteration vs E1 Error with Jacobi method for 3D Poisson Equation

Figure 7: Error for 2 different methods for solving 3D Poisson Equation

2.4 Physics Informed Neural Network-based approach

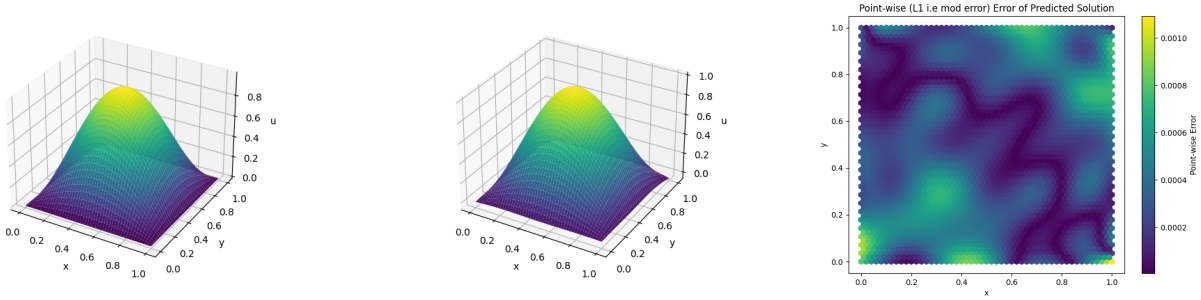
Recently, solving the governing partial differential equations of physical phenomena using deep learning has emerged as a new field of scientific machine learning (SciML), leveraging the universal approximation theorem and high expressivity of neural networks. In general, deep neural networks could approximate any high-dimensional function given that sufficient training data are supplied. However, such networks do not consider the physical characteristics underlying the problem, and the level of approximation accuracy provided by them is still heavily dependent on careful specifications of the problem geometry as well as the initial and boundary conditions. Without this preliminary information, the solution is not unique and may lose physical correctness. On the other hand, physics-informed neural networks (PINNs) leverage governing physical equations in neural network training. Namely, PINNs are designed to be trained to satisfy the given training data as well as the imposed governing equations. In this fashion, a neural network can be guided with training data that do not necessarily need to be large and complete. Potentially, an accurate solution of partial differential equations can be found without knowing the boundary conditions. Therefore, with some knowledge about the physical characteristics of the problem and some form of training data (even sparse and incomplete), PINN may be used for finding an optimal solution with high fidelity.
PINNs can be thought of as a meshfree alternative to traditional approaches (e.g., CFD for fluid dynamics), and new data-driven approaches for model inversion and system identification. In addition, they allow for exploiting automatic differentiation (AD) to compute the required derivatives in the partial differential equations, a new class of differentiation techniques widely used to derive neural networks assessed to be superior to numerical or symbolic differentiation.

Here we will solve the same 2D Poisson equation on $[0,1] \times [0,1]$:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -2 * \pi^2 * sin(\pi * x) * sin(\pi * y) \text{ in } \Omega.$$
$$u = 0 \text{ on } \partial\Omega.$$

Our PINN model has 2 input nodes(x,y) , 3 hidden layer with 30 neurons each and 1 output node.We used tanh activation function.Let say, u_{pred} is the predicted solution, and f_{pred} be predicted RHS.Let $(x_i, y_i)_{i=1}^N$ be the set of training data.Then loss in interior points L1= $1/N * \Sigma_{i=1}^N (f(x_i, y_i) - f_{pred}(x_i, y_i, W))^2$, and let say there are b boundary points, then loss due to boundary points will be L2=$1/b\Sigma_{j=1}^b (u_{pred}(x_j, y_j, W))^2$.Our aim is to minimize L=w1*L1+w2*L2[weighted sum has been taken for better generalizability.w1,w2 are hyperparameters], more precisely finding $\hat{W} = \arg\min_W L$ [W being lernable parameter in our PINN]

As we knew the actual solution we used it find the error($||u_{pred} - u_{true}||_1$ after training the model.
We continued for 30000 epochs , and ended up with training loss 0.005843926686793566.Here is the plot of predicted u ,actual u, test loss.Test loss over 10000000*10000000 is 0.0002867316.



(a) a (b) b (c) c

Figure 8: a):actual solution, (b):Predicted Solution , (c)Loss plot over 10000000 test points

3 Question2:Video-Recovery: An application of t-product and M-product based CUR Decomposition

3.1 Application of Tensor Completion

- Image Analysis[1,2]
- Recommender System[3,4]
- wireless spectrum map construction[5]
- seismology signal processing[6]
- computer vision[7]

3.2 Matrix Based CUR

Let us first tell briefly about the matrix based CUR decomposition[13,14], before jumping into Tensor T-product based CUR.

- **RESULT1**

  Let the matrix A $\in$ F$^{n1\times n2}$ have rank(A)= r .Let I $\subset$ {1:n1} and J $\subset$ {1:n2} satisfying —I—$\geq r, |J| \geq r$. Let matrices C=A(:,J),R=(I,:),U=A(I,J),if rank(U)=rank(A),then A=CUR.[U, being pseudoinverse of U]

- **THEOREM**[15] **(Mahone and Drineas):**

  CUR in O(n1$\times$n2) time achieves $||A - CUR||_F \leq ||A - A_k||_F + \epsilon||A||_F$ with probability at least (1-δ), by picking O(klog(1/δ)/ϵ^2) columns , and O($\frac{k^2 log^3(1/\delta)}{\epsilon^6}$)

- **RESULT**[16]:

  Select c=O(klogk/ϵ^2) number of columns of Ausing column select algorithm[16], and select r=O(klogk/ϵ^2) rows of A using row select algorithm[16].Set U=W, W being intersection of r rows and c columns.Then , we have $||A - CUR||_F \leq ||A - A_k||_F * (2 + \epsilon)$, with probability 98%.

We used a special sampling technique(based on finding row and columns with maximum relative Frobenius norm) for sampling row and columns.Here is the the pseudocode for row sampling(same method applied for column sampling)

Algorithm 1 ROW_SAMPLING_ALGO(Input: matrix A, number of sampled row c.Output:selected row indices and submatrix of A with selected rows.)

procedure ROW_SAMPLING_ALGO(A, c)
  $m \leftarrow$ size$(A, 1)$
  $n \leftarrow$ size$(A, 2)$
  if $c > m$ **then**
    Print "Number of rows you want to consider exceeding total number of rows"
  end if
  $p \leftarrow$ zeros$(m, 1)$
  $indices \leftarrow$ zeros$(c, 1)$
  $t \leftarrow$ norm$(A, fro)^2$
  for x from 1 to m **do**
    $p(x) \leftarrow$ norm$(A(x, :), fro)^2/t$
  end for
  $A\_row\_sub \leftarrow$ zeros(c, n)
  $i \leftarrow 1$
  while $i \leq c$ **do**
    [max_value, max_index] $\leftarrow$ max(p)
    $indices(i) \leftarrow$ max_index
    $A\_row\_sub(i, :) \leftarrow A$(max_index, :)
    p(max_index) $\leftarrow$ min$(p) - i$
    $i \leftarrow i + 1$
  end while
  $indices \leftarrow$ sort$(indices)$
  $A\_row\_sub \leftarrow A(indices, :)$
  return $A\_row\_sub, indices$
end procedure

3.3 Useful Definitions

T-Product based

- *Definition* 2.1 (T-Product[12]): The T-product $\mathscr{A} * \mathscr{B}$ of tensor $\mathscr{A} \in R^{I_1 \times I_2 \times I_3}$ and $\mathscr{B} \in R^{I_2 \times I_4 \times I_3}$ is an $I_1 \times I_4 \times I_3$ tensor whose $(i, j)^{\text{th}}$ tube $\mathscr{C}(i, j, :)$ is given by

$$\mathscr{C}(i, j, :) = \sum_{k=1}^{I_2} \mathscr{A}(i, k, :) * \mathscr{B}(k, j, :),$$

  where $*$ denotes the circular convolution between two tubes of the same size. We anchor the MatVec command to the frontal faces of the tensor. $\text{MatVec}(\mathscr{A})$ takes an $I_1 \times I_2 \times I_3$ tensor and returns a block $I_1 I_2 \times I_3$ matrix, whereas the fold command undoes this operation

$$\text{MatVec}(\mathscr{A}) = \begin{bmatrix} \mathscr{A}^{(1)} \\ \mathscr{A}^{(2)} \\ \vdots \\ A^{(I_3)} \end{bmatrix}, \quad \text{fold}\left(\text{MatVec}(\mathscr{A})\right) = \mathscr{A}.$$

  Then the T-product $\mathscr{A} * \mathscr{B}$ is the $I_1 \times I_4 \times I_3$ tensor

$$\mathscr{A} * \mathscr{B} = \text{fold}(\text{circ}(\mathscr{A}) \cdot \text{Mat Vec}(\mathscr{B}))$$

  Zhang and Aeron [12] showed that for any tensor $\mathscr{A} \in R^{I_1 \times I_2 \times 3}$ and $\mathscr{B} \in R^{I_2 \times l \times 3}$, then $\mathscr{A} * \mathscr{B} = \mathscr{C} \Leftrightarrow \overline{\mathscr{A}\mathscr{B}} = \overline{\mathscr{C}}$. We are able to transform the t-product into its equivalent form in Fourier domain. On the other hand, we can also transform an operator in Fourier domain back to the original domain as needed.

- *Definition* 2.2 (Identity Tensor [12]): The identity tensor $\mathscr{I} \in R^{n_1 \times n_1 \times n_3}$ is defined to be a tensor whose first frontal slice $\mathscr{I}^{(1)}$ is the $n_1 \times n_1$ identity matrix and all other frontal slices $\mathscr{I}^{(i)}; i = 2, \ldots, n_3$ are zero.

- *Definition* 2.3 (Orthogonal Tensor [12]): A tensor $\mathscr{A} \in R^{n_1 \times n_1 \times n_3}$ is orthogonal if it satisfies

$$\mathscr{A}^T * \mathscr{A} = \mathscr{A} * \mathscr{A}^T = \mathscr{I}$$

- *Definition* 2.4 (Inverse of Tensor [12]): The inverse of a tensor $\mathscr{A} \in R^{I_1 \times I_2 \times I_3}$ is written as $\mathscr{A}^{-1}$ satisfying

$$\mathscr{A}^{-1} * \mathscr{A} = \mathscr{A} * \mathscr{A}^{-1} = \mathscr{I}$$

M-Product based

- *Definition* 2.5 k-mode product
Let $\mathscr{A} = (a_{i_1 i_2 \ldots i_N}) \in \mathbb{R}^{I_1 \times \cdots I_{k-1} \times I_k \times I_{k+1} \times \cdots I_N}$ be a tensor and $B = (b_{i_1 i_2}) \in \mathbb{R}^{p \times I_k}$ be a matrix, then their k-mode product, denoted as $\mathscr{A} \times_k B$, is in $\mathbb{R}^{I_1 \times \cdots I_{k-1} \times p \times I_{k+1} \times \cdots I_N}$ and defined as

$$(\mathscr{A} \times_k B)_{i_1 \ldots i_{k-1} j i_{k+1} \ldots i_N} = \sum_{i_k=1}^{I_k} a_{i_1 \ldots i_{k-1} i_k i_{k+1} \ldots i_N} b_{j i_k}.$$

- *Definition* 2.6 (M-product).
Let $\mathscr{A}$ be $n_1 \times n_2 \times n_3$ and $\mathscr{B}$ be $n_2 \times l \times n_3$. Then the M-product $\mathscr{A} *_M \mathscr{B}$ is the $n_1 \times l \times n_3$ tensor

$$\mathscr{A} *_M \mathscr{B} = ((\mathscr{A} \times_3 M) \Delta (\mathscr{B} \times_3 M)) \times_3 M^{-1},$$

where $M \in \mathbb{F}^{n_3 \times n_3}$ is an arbitrary invertible matrix.

- *Definition* 2.7 C-product
- Let $C_n = \mathrm{dct}(\mathrm{eye}(n))$ be the $n \times n$ orthogonal Discrete Cosine Transform (DCT) matrix - Let $Z = \mathrm{diag}(\mathrm{ones}(n-1,1),1)$ be a $n \times n$ singular circular upshift matrix - $W = \mathrm{diag}\,(C_n(:,1))$ be the diagonal matrix made of the first column of the DCT matrix - $M = W^{-1} C_n\,(I_n + Z)$, where I_n is the $n \times n$ identity matrix.
Let $\mathscr{A}$ be $n_1 \times n_2 \times n_3$ and $\mathscr{B}$ be $n_2 \times l \times n_3$. Then the C-product $\mathscr{A} *_C \mathscr{B}$ is the $n_1 \times l \times n_3$ tensor

$$\mathscr{A} *_C \mathscr{B} = ((\mathscr{A} \times_3 M) \Delta (\mathscr{B} \times_3 M)) \times_3 M^{-1},$$

where $M = W^{-1} C_{n_3}\,(I_{n_3} + Z)$.

- *Definition* 2.8
Let $\mathscr{A} \in \mathbb{F}^{m \times n \times k}$ be a tensor and $B \in \mathbb{F}^{p \times k}$ be a matrix. The 3 -mode product of $\mathscr{A}$ with B is denoted by $\mathscr{A} \times_3 B \in \mathbb{F}^{m \times n \times p}$ and element-wise defined as

$$(\mathscr{A} \times_3 B)_{ijl} = \sum_{s=1}^{k} a_{ijs} b_{ks} \quad i = 1, 2, \ldots, m, j = 1, 2, \ldots, n, l = 1, 2 \ldots, p.$$

In particular, if $\mathscr{A} \in \mathbb{F}^{m \times n \times p}$ and $M \in \mathbb{F}^{p \times p}$ be an invertible matrix, we will use "hat" notation to denote a tensor in the transform domain specified by M, that is,

$$\hat{\mathscr{A}} := \mathscr{A} \times_3 M \in \mathbb{R}^{m \times n \times p}.$$

The "hat" notation should be understood in context relative to the M applied. From here onwards, we will assume M is an invertible matrix. Next, we define mat operation for transforming a tensor into a matrix with respect to M.

- *Definition* 2.9
Let $\mathscr{A} \in \mathbb{F}^{m \times n \times p}, M \in \mathbb{F}^{p \times p}$ and $(\hat{\mathscr{A}})^{(i)} \in \mathbb{F}^{m \times n}$ be the i^{th} frontal slice of $\hat{\mathscr{A}}$, for $i = 1, 2, \ldots, p$. Then mat: $\mathbb{F}^{m \times n \times p} \mapsto \mathbb{F}_M^{mp \times np}$ is defined as

$$\mathrm{mat}(\mathscr{A}) = \begin{pmatrix} (\hat{\mathscr{A}})^{(1)} & 0 & \cdots & 0 \\ 0 & (\hat{\mathscr{A}})^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (\hat{\mathscr{A}})^{(p)} \end{pmatrix}.$$

Let us consider $\mathbb{F}_M^{mp \times np} = \{A \in \mathbb{F}^{mp \times np} : A$ is of the form $\begin{pmatrix} A_1 & & O \\ & A_2 & \\ & & \ddots \\ O & & A_p \end{pmatrix}\}$. The inverse operation of mat, i.e., mat $^{-1} : \mathbb{F}_M^{mp \times np} \mapsto \mathbb{F}^{m \times n \times p}$ is defined by

$$\mathrm{mat}^{-1}(A) = \mathscr{A},$$

where $\mathscr{A} = \hat{\mathscr{A}} \times 3M^{-1}$ and $\hat{\mathscr{A}} \in \mathbb{F}^{m\times n\times p}$ with $(\hat{\mathscr{A}})^{(i)} = A_i, i = 1, 2, \ldots, p$. Thus, we can represent a tensor $\mathscr{A} \in \mathbb{F}^{m\times n\times p}$ as follows:

$$\mathscr{A} = \text{mat}^{-1}(\text{mat}(\mathscr{A})).$$

Now, the M-product of two tensors is defined with the help of 'mat' and 'mat $^{-1}$ '.

- *Definition* 2.10
  Let $\mathscr{A} \in \mathbb{F}^{m\times n\times p}, \mathscr{B} \in \mathbb{F}^{n\times k\times p}$ and $M \in \mathbb{F}^{p\times p}$. Then the M-product of tensors $\mathscr{A}$ and $\mathscr{B}$ is denoted by $\mathscr{A} *_M \mathscr{B} \in \mathbb{F}^{m\times k\times p}$ and defined as

  $$\mathscr{A} *_M \mathscr{B} = \text{mat}^{-1}(\text{mat}(\mathscr{A})\,\text{mat}(\mathscr{B})).$$

  Note that if we choose M as the unnormalized DFT matrix (MATLAB command $dftmtx(n)$ can be used to generate the DFT matrix of order n), then the M-product coincides with the t-product [3]. The cosine transform product (c-product [1]) similarly obtained by taking the matrix $M = W^{-1}C(I+Z)$, where C is the DCT matrix (MATLAB command $\text{dctmtx}(n)$) of order $n, Z = \text{diag}(\text{ ones }(n-1, 1), 1)$ and $W = \text{diag}(C(:, 1))$. We denote the matrix $W^{-1}C(I + Z)$ by M_1. Thus c-product is obtained while choosing $M = M_1$.

- *Definition* 2.11
  Let $\mathscr{A} \in \mathbb{F}^{m\times m\times p}$ and $M \in \mathbb{F}^{p\times p}$. Then $\mathscr{A}$ is said to be an identity tensor if $(\mathscr{A} \times {}_3M)^{(i)} = I_m$ for $i = 1, 2, \ldots, p$ and I_m is the identity matrix of order m.

  Equivalently, we can say that $\mathscr{A}$ is an identity tensor if the block diagonals of $\text{mat}(\mathscr{A})$ are identity matrices of order m. We denote the identity tensor $\mathscr{I} \in \mathbb{F}^{m\times m\times p}$ by $\mathscr{I}_{mmp}$. Further, if the block diagonals of $\text{mat}(\mathscr{A})$ are permutation matrices of order m then we call $\mathscr{A}$ is a permutation tensor.

- *Definition* 2.12
  Let $\mathscr{A} \in \mathbb{F}^{m\times n\times p}$ and $M \in \mathbb{F}^{p\times p}$. Then the transpose conjugate of $\mathscr{A}$ is denoted by $\mathscr{A}^*$, which can be obtained from the relation $(\mathscr{A}^* \times {}_3M)^{(i)} = \left((\mathscr{A} \times {}_3M)^{(i)}\right)^*$ for $i = 1, 2, \ldots, p$. Equivalently, $\mathscr{A}^* = \text{mat}^{-1}\,(\text{mat}(A)^*)$.

- *Definition* 2.13
  Let $\mathscr{A} \in \mathbb{F}^{m\times m\times p}$ and $M \in \mathbb{F}^{p\times p}$. If there exist a tensor $\mathscr{X}$ such that $\mathscr{A} *_M \mathscr{X} = \mathscr{I}_{mmp} = \mathscr{X} *_M \mathscr{A}$ then $\mathscr{A}$ is invertible and $\mathscr{A}^{-1} = \mathscr{X}$. The following results can be easily verified from the mat and mat $^{-1}$ operations.

- *Proposition* 2.14 Let $\mathscr{A} \in \mathbb{C}^{m\times n\times p}, \mathscr{B} \in \mathbb{C}^{n\times k\times p}$ and $M \in \mathbb{C}^{p\times p}$.
  Then
  (a) $\text{mat}(\alpha\mathscr{A} + \beta\mathscr{B}) = \alpha\,\text{mat}(\mathscr{A}) + \beta\,\text{mat}(\mathscr{B})$, where $\alpha, \beta \in \mathbb{F}$.
  (b) $\text{mat}\,(\mathscr{A} *_M \mathscr{B}) = \text{mat}(\mathscr{A})\,\text{mat}(\mathscr{B})$.
  (c) $\text{mat}(\mathscr{A})^* = \text{mat}\,(\mathscr{A}^*)$.

- *Proposition* 2.15
  Let $A \in \mathbb{C}^{mp\times np}, B \in \mathbb{C}^{np\times kp}$ and $M \in \mathbb{C}^{p\times p}$. Then
  (a) $\text{mat}^{-1}(\alpha A + \beta B) = \alpha mat^{-1}(A) + \beta mat^{-1}(B), \alpha, \beta \in \mathbb{F}$.
  (b) $mat^{-1}(AB) = mat^{-1}(A) *_M mat^{-1}(B)$.

- *Corollary* 2.16
  Let $A \in \mathbb{C}^{m\times m\times p}$ be an invertible tensor and $M \in \mathbb{C}^{p\times p}$. Then
  (a) $\text{mat}(\mathscr{A})^{-1} = \text{mat}\left(\mathscr{A}^{-1}\right)$.
  (b) $\mathscr{A}^{-1} = \text{mat}^{-1}\left(\text{mat}(\mathscr{A})^{-1}\right)$.
  (c) $\text{mat}(\mathscr{A})\,\text{mat}\left(\mathscr{A}^{-1}\right) = I_{mp} = \text{mat}(\mathscr{A})\,\text{mat}(\mathscr{A})^{-1}$.

3.4 Used Tensor Decompositions

Tensor t-CUR algorithm

Suppose we sample a 3-way tensor $\mathscr{A} \in \mathbb{R}^{I_1\times I_2\times I_3}$ at the set of indices in a set Ω. Let P_Ω denotes the sampling operator

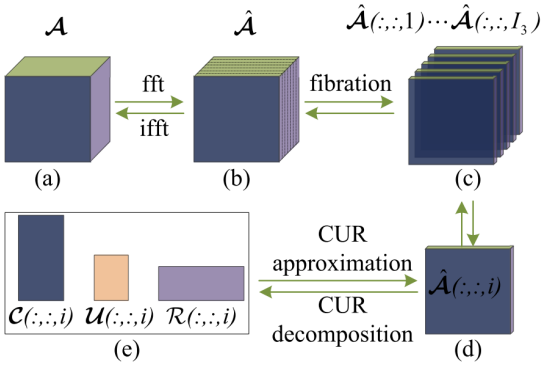
$$P_\Omega : \mathbb{R}^{I_1\times I_2\times I_3} \to \mathbb{R}^{I_1\times I_2\times I_3},$$

which is defined by

$$P_\Omega(\mathscr{A})_{ijk} = \begin{cases} \mathscr{A}_{ijk} & (i,j,k) \in \Omega \\ 0 & otherwise \end{cases}.$$

- **Algorithm** 2.1:Tensor-TCUR Approximation From Partially Observed Entries (T-CUR) Require: Observation data $P_\Omega(\mathscr{A}) \in R^{I_1 \times I_2 \times I_3}$, sampled columns/rows number li/di.Random Sampling has been used.

1. $\hat{\mathscr{A}} = FFT\left(P_\Omega(\mathscr{A}), [], 3\right);$
2. for $i \leftarrow 1, \ldots, I_3$ do
3. $\left[\hat{\mathscr{C}}^{(i)}, \hat{\mathscr{U}}^{(i)}, \hat{\mathscr{R}}^{(i)}\right] = \text{MATRIX-CUR}\left(P_\Omega\left(\hat{\mathscr{A}}^{(i)}\right), d_i, l_i\right)$
4.end for
5.$\mathscr{C} = \text{IFFT}(\hat{\mathscr{C}}, [], 3); \mathscr{U} = \text{IFFT}(\hat{\mathscr{U}}, [], 3); \mathscr{R} = \text{IFFT}\ (\hat{\mathscr{R}}, [], 3)$
6.Return $\mathscr{C}, \mathscr{U}, \mathscr{R}$ such that a tensor cur approximation of $\mathscr{A} : \mathscr{Y} = \mathscr{C} * \mathscr{U} * \mathscr{R} \in R^{I_1 \times I_2 \times I_3}$



(a) Image 1

```
function [C, U, R] = TENSOR_TCUR(A, d1, d2)
    % Input: A - a tensor of size n1 x n2 x n3
    %        d1 - sample size for rows
    %        d2 - sample size for columns

    A_tilde = fft(A,[],3);
    s1=size(A,1);
    s2=size(A,2);
    s3=size(A,3);
    % Sample row index I and column index J
    I = sort(randsample(size(A,1), d1));
    J = sort(randsample(size(A,2), d2));

    C = zeros([size(A,1), d2, size(A,3)]);
    R = zeros([d1, size(A,2), size(A,3)]);
    U = zeros([d2,d1,size(A_tilde ,3)]);

    for k= 1:size(A_tilde ,3)
        C(:,:,k) = A_tilde(:,J,k);
        R(:,:,k) = A_tilde(I,:,k);
        U(:,:,k) = MAT_PSEUDOINV(A_tilde(I,J,k))
    end

    C = ifft(C,[],3);
    R = ifft(R,[],3);
    U = ifft(U,[],3);
end
```

(b) Image 2

Figure 9: T-CUR

Tensor M-CUR Clearly 'mat' operation as discussed in Definition 2.9 is our required homeomorphism.

- **Algorithm**2.2 : Tensor-MCUR Approximation From Partially Observed Entries. Require: Observation data $P_\Omega(\mathscr{A}) \in R^{I_1 \times I_2 \times I_3}$, vector of sample columns/rows number li/di

1. $\hat{\mathscr{A}} = mat\left(P_\Omega(\mathscr{A}), [], 3\right);$
2. for $i \leftarrow 1, \ldots, I_3$ do
3. $\left[\hat{\mathscr{C}}^{(i)}, \hat{\mathscr{U}}^{(i)}, \hat{\mathscr{R}}^{(i)}\right] = \text{MATRIX-CUR}\left(P_\Omega\left(\hat{\mathscr{A}}^{(i)}\right), d_i, l_i\right)$
4.end for
5.$\mathscr{C} = \text{mat}^{-1}(\hat{\mathscr{C}}); \mathscr{U} = \text{mat}^{-1}(\hat{\mathscr{U}}); \mathscr{R} = \text{mat}^{-1}\ (\hat{\mathscr{R}})$
6.Return $\mathscr{C}, \mathscr{U}, \mathscr{R}$ tensor cur approximation of $\mathscr{A} : \mathscr{Y} = \mathscr{C} *_M \mathscr{U} *_M \mathscr{R} \in R^{I_1 \times I_2 \times I_3}$

```
function [C, U, R] = TENSOR_MCUR(A, M,d1,d2)
    % Input: A - a tensor of size n1 x n2 x n3
    %          d1 - sample size for rows
    %          d2 - sample size for columns

    A_tilde = kmode_product(A,M);
    M_inv=inv(M);
    s1=size(A,1);
    s2=size(A,2);
    s3=size(A,3);
    % Sample row index I and column index J
    I = sort(randsample(size(A,1), d1));
    J = sort(randsample(size(A,2), d2));

    C = zeros([size(A,1), d2, size(A,3)]);
    R = zeros([d1, size(A,2), size(A,3)]);
    U = zeros([d2,d1,size(A_tilde ,3)]);

    for k= 1:size(A_tilde ,3)
        C(:,:,k) = A_tilde(:,J,k);
        R(:,:,k) = A_tilde(I,:,k);
        U(:,:,k) = MAT_PSEUDOINV(A_tilde(I,J,k));
    end

    C =kmode_product(C,M_inv);
    R = kmode_product(R,M_inv);
    U = kmode_product(U,M_inv);
end
```

Figure 10: TENSOR m-CUR

3.5 CUR vs Other Methods

This section compares t-CUR with past approaches.

CP vs CUR Suppose we sample a 3-way tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ at the set of indices in a set Ω. Let P_Ω denotes the sampling operator

$$P_\Omega : \mathbb{R}^{I_1 \times I_2 \times I_3} \to \mathbb{R}^{I_1 \times I_2 \times I_3},$$

which is defined by

$$P_\Omega(\mathscr{A})_{ijk} = \begin{cases} \mathscr{A}_{ijk} & (i,j,k) \in \Omega \\ 0 & otherwise \end{cases}.$$

Jain and Oh [17] tried to complete the tensor by solving the following optimization problem

$$minimize_{\hat{\mathscr{A}}, rank(\hat{\mathscr{A}})=r} ||P_\Omega(\mathscr{A} - \Sigma_{l=1}^{r} \sigma_l (a_l^{(1)} a_l^{(2)} a_l^{(3)}))||_F^2.$$

They showed that under certain standard assumptions, the proposed method can recover a three-mode $n \times n \times n$ dimensional rank-r tensor exactly from $O(n^{3/2} \cdot r^5 log^4 n)$ randomly sampled entries.But knowing r is NP complete.

Tucker Decomposition vs CUR

Liu et al. [7] proposed a tensor completion algorithm based on minimizing tensor n-rank in Tucker decomposition format.

It uses the matrix nuclear norm instead of matrix rank, and try to solve the convex problem as follows

$$\min_{\mathcal{X}} \sum_{i=1}^{n} \alpha_i \left\| X_{(i)} \right\|_*$$
$$\text{subject to } P_\Omega(\mathcal{X}) = P_\Omega(\mathcal{A}),$$

where $X_{(i)}$ is the mode- n matricization of $\mathcal{X}$ and $\alpha_{(i)}$ are prespecified constants satisfying $\alpha_{(i)} \geq 0, \sum_{i=1}^{n} \alpha_{(i)} = 1$.

Unlike the optimal dimensionality reduction in matrix PCA which can be obtained by truncating the SVD, there is no trivial multi-linear counterpart to dimensionality reduction for Tucker type. Alternatively, suitable choices of the truncation values of n-rank are not likely to be known a priori [18]. As a contrast, our proposed algorithm use the actual rows and columns of the tensor to efficiently compute the low rank approximation of a given tensor, without having to know the rank a prior.

SVD vs CUR

The tensor tubal rank is used in tensor-SVD, also referred to as tensor rank. Zhang et al. tried to complete the tensor by solving the following convex optimization problem

$$\min_{\mathcal{X}} \|\mathcal{X}\|_{TNN}$$
$$\text{subject to } P_\Omega(\mathcal{X}) = P_\Omega(\mathcal{A}),$$

where the tensor nuclear norm is taken as the convex relaxation of tensor tubal rank, $\| * \|_{TNN}$ is the tensor nuclear norm. Zhang et al. showed that one can perfectly recover a tensor of size $I_1 \times I_2 \times I_3$ with rank r under tensor-SVD as long as $O\left(rI_1I_3 \log\left(\left(I_1 + I_2\right) I_3\right)\right)$ samples are observed.

Our suggested method, which computes the low rank approximation of a given tensor using the tensor's actual rows and columns, extends matrix-CUR decomposition to tensor. Because it simply has to resolve a typical regression problem, it is computationally efficient. Furthermore, our suggested method may effectively compute the low rank approximation of a given tensor utilizing the tensor's real rows and columns without requiring knowledge of the rank beforehand.

3.6 Mathematical Guarantees

1)CUR decomposition of a real valued tensor always exists.(Because CUR decomposition of a matrix is guaranteed, and with the isomorphism operation fft, we are having a isomorphic relation between matrix space and tensor space)

2)If $\mathscr{B}$ is the approximation of tensor $\mathscr{A}$, then the used error formulation for the approximation was: $\frac{||\mathscr{A}-\mathscr{B}||_F}{||\mathscr{A}||_F}$

3)Let the tensor $\mathscr{A} \in \mathrm{F}^{n1 \times n2 \times n3}$ have multi-rank($\mathscr{A}$)= R[Let $\hat{\mathscr{A}}$=fft($\mathscr{A}$), and U^i be its i th frontal slice.multi rank of tensor $\mathscr{A}$ is an array of length n3, where ith entry is rank of U^i.And tubal rank of A is maximum entry in the array of multi rank] , and tubalrank=r.Let I $\subset$ {1:n1} and J $\subset$ {1:n2} satisfying $|I| \geq r, |J| \geq r$. Let tensors $\mathscr{C} = \mathscr{A}$(:,J,:),$\mathscr{R} = \mathscr{A}$(I,:,:),$\mathscr{U} = \mathscr{A}$(I,J,:),if multirank($\mathscr{U}$)=multirank($\mathscr{A}$),then $\mathscr{A} = \mathscr{C} * \mathscr{U}_{pinv} * \mathscr{R}$[$\mathscr{U}_{pinv}$, being pseudoinverse of $\mathscr{U}$]

3.7 Observation

We took a video of duration 18 seconds consists of total 326 frames.We used M=[-1,2,0;0,1,-2;0,0,1].As we can see the from the below image-frames error using M-product based CUR(M-CUR) is more than error using t-CUR for number of sampled row=column=100.

For error(frobenius norm between actual frame and cur based reconstructed frame) as well as time calculation , we took average of time ,error in computation over all frames for every value of sampled rows/columns.

For the number of sampled row as well as column, N=[50,100,300,500,700,900], the error (in Frobenius norm) for t-CUR is :[3.5935, 5.9604, 1.0253, 0.7189, 3.1763, 0.3464], and the error for M-CUR is :[21.8257, 23,882, 7.7357, 14.2377, 2.4815, 2.7236],

With the same N, time taken in M-CUR based recovery (in second):[0.7104 , 0.7672 ,1.2196 , 1.9174 , 2.8082, 4.9806].Time taken for t-CUR based recovery is:[0.2019, 0.4120, 0.5303, 1.5411 , 2.6459, 4.0452].

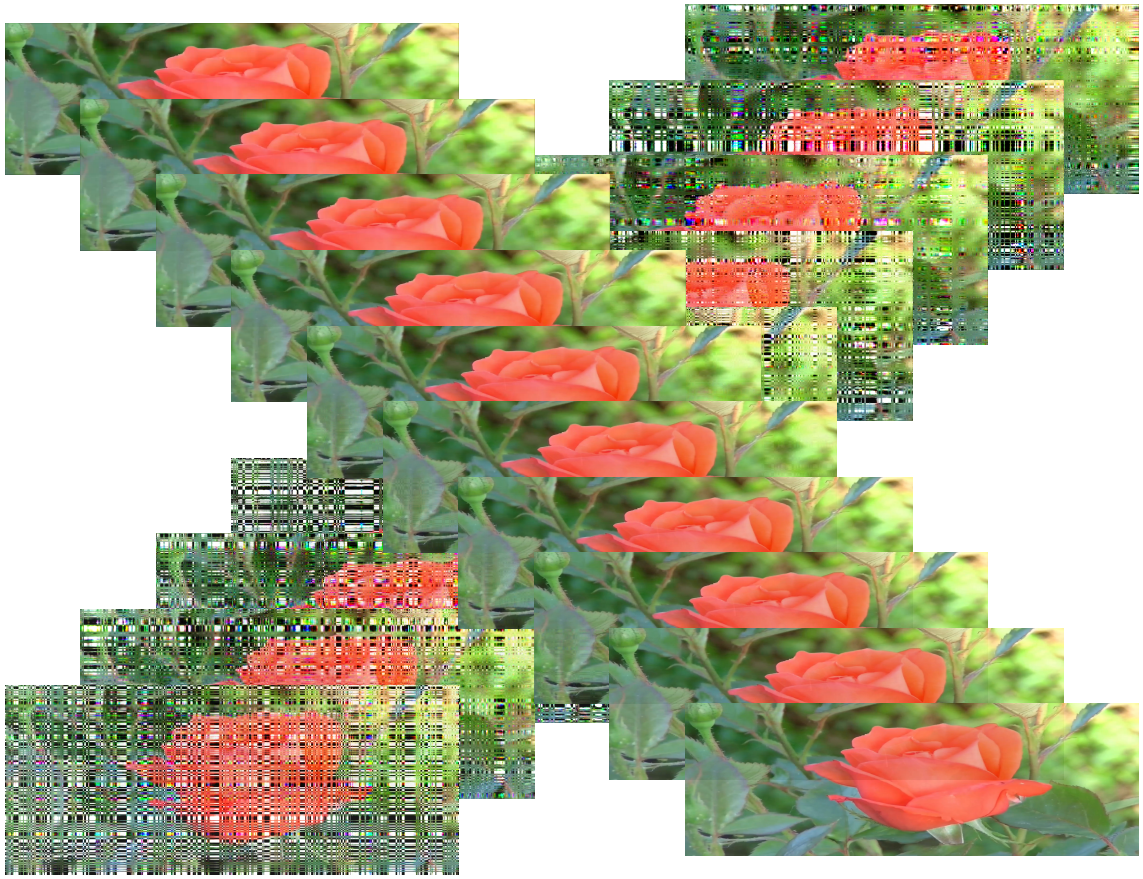


Figure 11: Comparison of actual frames vs Frames recovered bt t-CUR with number of rows=columns=100

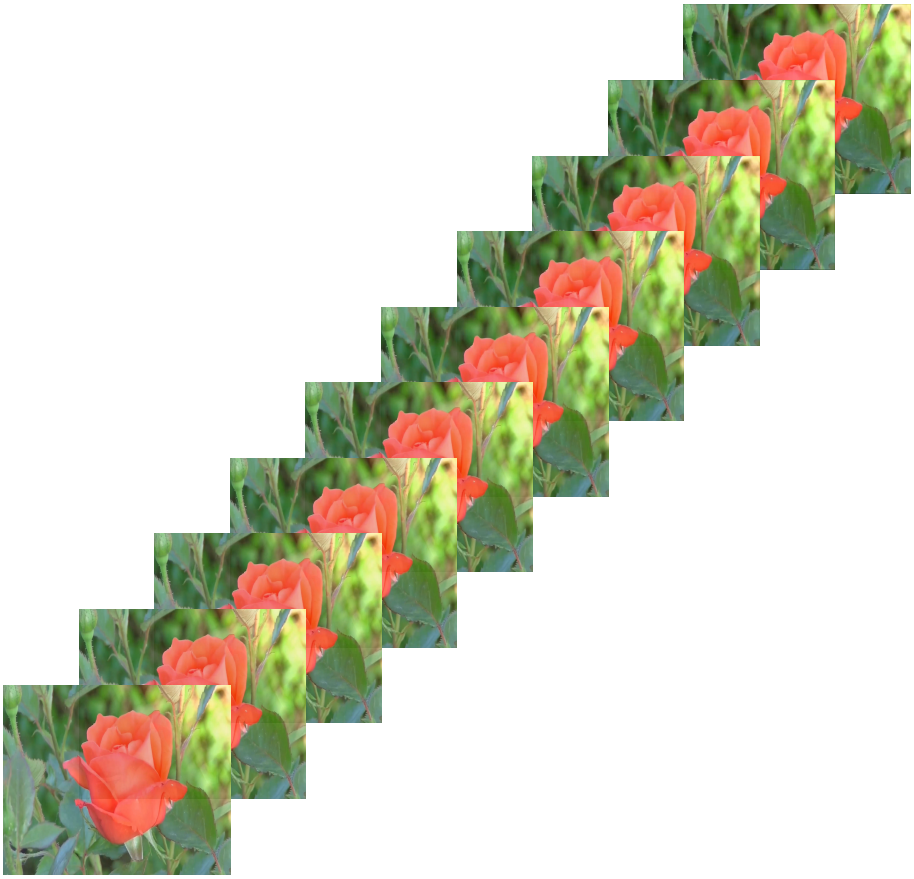
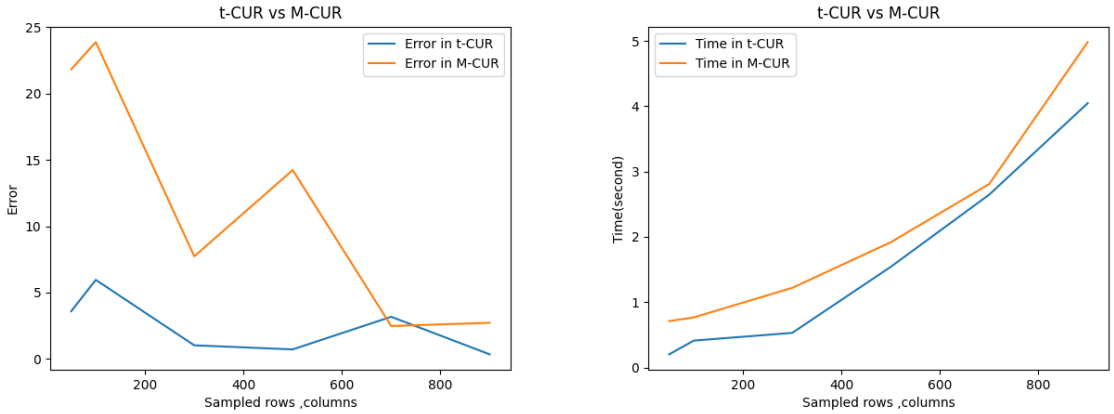


Figure 12: Comparison of actual frames vs Frames recovered bt M-CUR with number of rows=columns=100



(a) a

(b) b

Figure 13: a):Error in t-cur vs m-cur (b):time taken in t-cur vs m-cur

Our experiment, indeed imply that with more number of samples , error will be lower , and time will increase.But from the experiment it is clear that with respect to both time , as well as error t-CUR performs relatively better that m-CUR.

4 Conclusion

Tensor based approach to ease the computation cost of solving higher order PDEs were studied in this work with a special emphasis on Poisson equation.Both iterative methods (Gauss Jacobi) , and direct method (Einstein Product related pseudoinverse computation to solve high-way tensor based representation of system of equations) were used to solve the problem.Experimental result concluded that, with lower number of grid points direct methods are the best to get lower error,

while for higher number of grids iterative solver outperform.

The missing data recovery problem of a 3-way tensor was defined as a tensor completion problem in our previous study[19].Our suggestion was a new tensor completion approach that could efficiently retrieve the absent data from small samples.It's a novel approach of turning the matrix-CUR into a three-way tensor. This means that one might express a 3-way tensor as a product of other 3-way tensors.

As a valid extension we studied color video recovery using t-CUR,M-CUR. Our experiments shows that with respect to time as well as error in computation, t-CUR outperforms over M-CUR.

5 References

1[13])J. A. Bengua, H. N. Phiem, H. D. Tuan, and M. N. Do, “Efficient tensor completion for color image and video recovery: Low-rank tensor train,” IEEE Trans. Image Process., vol. 26, no. 5, pp. 2466–2479, May 2017.
2[14])] N. Li and B. Li, “Tensor completion for on-board compression of hyperspectral images,” in Proc. IEEE Int. Conf. Image Process., Sep. 2010, pp. 517–520.
3[15]) A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, “Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering,” in Proc. ACM Conf. Recommender Syst. (RecSys), Barcelona, Spain, Sep. 2010, pp. 79–86.
4[16])W. W. Sun, J. Lu, H. Liu, and G. Cheng, “Provable sparse tensor decomposition,” J. Roy. Stat. Soc., vol. 79, no. 3, pp. 899–916, 2016.
5[17])M. Tang, G. Ding, Q. Wu, Z. Xue, and T. A. Tsiftsis, “A joint tensor completion and prediction scheme for multi-dimensional spectrum map construction,” IEEE Access, vol. 4, no. 99, pp. 8044–8052, 2016.
6[18])Y. Zhang, C. D. Silva, R. Kumar, and F. J. Herrmann, “Massive 3D seismic data compression and inversion with hierarchical tucker,” in Proc. SEG, 2017, pp. 1347–1352.
7[19])J. Liu, P. Musialski, P. Wonka, and J. Ye, “Tensor completion for estimating missing values in visual data,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 1, pp. 208–220, Jan. 2013.
8[20])B. Ran, H. Tan, Y. Wu, and P. J. Jin, “Tensor based missing traffic data completion with spatial–temporal correlation,” Phys. A, Stat. Mech. Appl., vol. 446, pp. 54–63, Mar. 2016.
9[21])E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, “Scalable tensor factorizations for incomplete data,” Chemometrics Intell. Lab. Syst., vol. 106, no. 1, pp. 41–56, 2011.
10[22]) H. Tan, G. Feng, J. Feng, W. Wang, Y.-J. Zhang, and F. Li, “A tensorbased method for missing traffic data completion,” Transp. Res. C, Emerg. Technol., vol. 28, pp. 15–27, Mar. 2013.
11[23])J. Håstad, “Tensor rank is NP-complete,” J. Algorithms, vol. 11, no. 4, pp. 644–654, 1990.
12[25])Z. Zhang and S. Aeron, “Exact tensor completion using t-SVD,” IEEE Trans. Signal Process., vol. 65, no. 6, pp. 1511–1526, Mar. 2017.
13) Cai, H., Hamm, K., Huang, L., Li, J., Wang, T. (2021). Rapid robust principal component analysis: CUR accelerated inexact low rank estimation. IEEE Signal Process. Lett. 28:116–120. DOI: 10.1109/LSP.2020.3044130.
14) Hamm, K., Huang, L. (2020). Perspectives on CUR decompositions. Appl. Comput. Harmon. Anal. 48(3):1088–1099. DOI: 10.1016/j.acha.2019.08.006.
15)CUR Decomposition
16)Advanced CUR Decomposition
17[30])P. Jain and S. Oh, “Provable tensor factorization with missing data,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 1431–1439.
18[24])N. Hao, M. E. Kilmer, K. Braman, and R. C. Hoover, “Facial recognition using tensor-tensor decompositions,” SIAM J. Imag. Sci., vol. 6, no. 3, pp. 437–463, 2013.
19)t-CUR based tensor completion
20)M. Brazell, N. Li, C. Navasca, et al. Solving multilinear systems via tensor inversion. SIAM J. Matrix Anal,Appl. 2013;34(2):542-570.
21)Further results on the Drazin inverse of even-order tensors;Ratikanta Behera, Ashish Kumar Nandi,Jajati Keshari Sahoo,WILEY,DOI: 10.1002/nla.2317
22)E. Cuthill and J. McKee, Reducing the bandwidth of sparse symmetric matrices, in Proceedings of the 24th ACM National Conference, New York, 1969, pp. 157–172.
23) J.A. George and J.W.-H. Liu, Computer Solution of Large Sparse Positive Definite Systems,Prentice-Hall, Englewood Cliffs, NJ, 1981.