Notes on WarpX PSATD Solver

1 Introduction

In PSATD, Maxwell's equations in Fourier domain from step n to step $n+1$ on staggered grids are as follows:

$$\begin{aligned}
\tilde{\mathbf{E}}^{n+1} &= C\tilde{\mathbf{E}}^{n} + iS\hat{k}\times\tilde{\mathbf{B}}^{n} - \frac{S}{ck}\tilde{\mathbf{J}}^{n+1/2} + i\frac{\hat{k}}{k}\left[\left(\frac{S}{ck\Delta t}-1\right)\tilde{\rho}^{n+1} + \left(C-\frac{S}{ck\Delta t}\right)\tilde{\rho}^{n}\right] \\
\tilde{\mathbf{B}}^{n+1} &= C\tilde{\mathbf{B}}^{n} - iS\hat{k}\times\tilde{\mathbf{E}}^{n} + i\frac{1-C}{ck}\hat{k}\times\tilde{\mathbf{J}}^{n+1/2},
\end{aligned} \tag{1}$$

where $\tilde{a}$ is the Fourier transform of the quantity a, $\mathbf{E}$ is the electric field, $\mathbf{B}$ is the magnetic field, $\mathbf{J}$ is the current density and ρ is the charge density. $\vec{k}$ is the wave vector of length $\mathbf{k} = \sqrt{k_x^2+k_y^2+k_z^2}$, and $\hat{k} = \vec{k}/\mathbf{k}$. c represents the speed of light, Δt is the time step, $S = \sin(kc\Delta t)$ and $C = \cos(kc\Delta t)$. Δt and is the time step and n is the time index. The equations are written as a linear system as

$$\begin{bmatrix} \tilde{\mathbf{E}}^{n+1} \\ \tilde{\mathbf{B}}^{n+1} \end{bmatrix} = M^{s} \times \begin{bmatrix} \tilde{\mathbf{E}}^{n} \\ \tilde{\mathbf{B}}^{n} \\ \tilde{\mathbf{J}}^{n+1/2} \\ \tilde{\rho}^{n} \\ \tilde{\rho}^{n+1} \end{bmatrix}, \tag{2}$$

where M_s is referred to as a transformation matrix.

While the computation is straightforward, it does require extra steps. In time domain, the fields are laid out on staggered grids, which are grouped into cells, as shown in Figure 1. The fields are either centered on the edges of the cells (blue samples) or they are centered on the faces of the cells (red samples). For example, the components of the magnetic field $\mathbf{B}$ are face centered, $\mathbf{B}_x$, $\mathbf{B}_y$ and $\mathbf{B}_z$ are centered on the yz, xz and xy faces of the cells, respectively. However, the components of the electric field $\mathbf{E}_x$, $\mathbf{E}_y$, $\mathbf{E}_z$ and the current density $\mathbf{J}_x$, $\mathbf{J}_y$, $\mathbf{J}_z$ are centered on the corresponding edges of

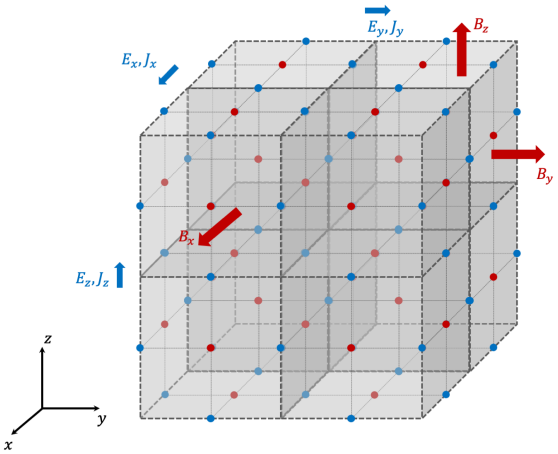


Figure 1: Layout of field components on the staggered "Yee" grid. Current densities and electric fields are defined on the edges of the cells and magnetic fields on the faces.

the cells. Note that (1) the number of samples in each field in each dimension is different and (2) the sampling points are not aligned to the nodal points of the cells.

For a given set of staggered grids of $N \times M \times K$ cells, the dimensions of the the magnetic field, electric field and current density are dependent on whether the components are edge or face centered. For example, the $\mathbf{B}_x$, $\mathbf{B}_y$ and $\mathbf{B}_z$ components are defined as

$$\mathbf{B_x} \in \mathscr{R}^{(N+1)\times M\times K} \tag{3}$$

$$\mathbf{B_y} \in \mathscr{R}^{N\times (M+1)\times K} \tag{4}$$

$$\mathbf{B_z} \in \mathscr{R}^{N\times M\times (K+1)}, \tag{5}$$

respectively. The $\mathbf{E}_x$, $\mathbf{E}_y$, $\mathbf{E}_z$, $\mathbf{J}_x$, $\mathbf{J}_y$, and $\mathbf{J}_z$ components are defined as

$$\mathbf{E_x} \in \mathscr{R}^{N\times (M+1)\times (K+1)} \tag{6}$$

$$\mathbf{E_y} \in \mathscr{R}^{(N+1)\times M\times (K+1)} \tag{7}$$

$$\mathbf{E_z} \in \mathscr{R}^{(N+1)\times (M+1)\times K} \tag{8}$$

$$\mathbf{J_x} \in \mathscr{R}^{N\times (M+1)\times (K+1)} \tag{9}$$

$$\mathbf{J_y} \in \mathscr{R}^{(N+1)\times M\times (K+1)} \tag{10}$$

$$\mathbf{J_z} \in \mathscr{R}^{(N+1)\times (M+1)\times K}, \tag{11}$$

respectively.

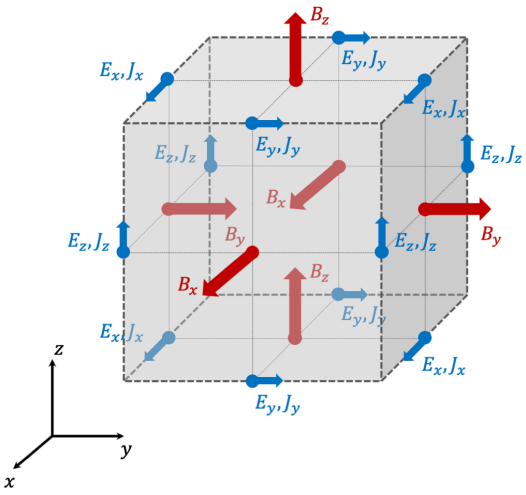


Figure 2: Layout of field components on the staggered "Yee" grid for a single cell.

The PSATD solver requires the fields to have the same number of samples. Data needs be packed before the computation, and subsequently needs to be unpacked after the computation. For example, the samples in $\mathbf{B}_x$ component of size $(N+1) \times M \times K$ are packed into a corresponding $\mathbf{B}'_x$ buffer of size $N \times M \times K$. For the $\mathbf{B}_x$ component, the samples corresponding to the last face in the x-dimension are simply dropped. Similar packing is done on the other fields. Once computation is done, the reverse operation is required, and only the $N \times M \times K$ samples of $\mathbf{B}_x$ are updated.

The fields are face centered or edge centered. As shown in Figure 2, where we depict a single cell, none of the fields are aligned to the nodal points of the cells. The magnetic field components are shifted by a half a sample in two dimensions, while the electric field and current density components are shifted by half a sample in a single dimension. The PSATD solver requires the fields to be aligned to the nodal cell points as shown in Figure 3. Since the samples are offset by only half a sample, time domain shift by half a sample cannot be done. Therefore, the data needs to be interpolated and re-sampled to the corresponding nodal points. The operation can be done using the DFTs and the algorithm proposed by Yaroslavsky, "Efficient algorithm for discrete sinc interpolation". The proposed method is basically a complex multiplication with complex exponentials in the frequency domain. For example, for a given one dimensional sequence $x_s[n] = x[n+1/2]$ of size N, computing the sequence $x[n]$ from $x_s[n]$ can in three steps such as

$$x[n] = \text{iDFT}\{e^{-jk\frac{2\pi}{N}\frac{1}{2}} \cdot \text{DFT}\{x_s[n]\}\}, \forall 0 \leq n < N, \tag{12}$$

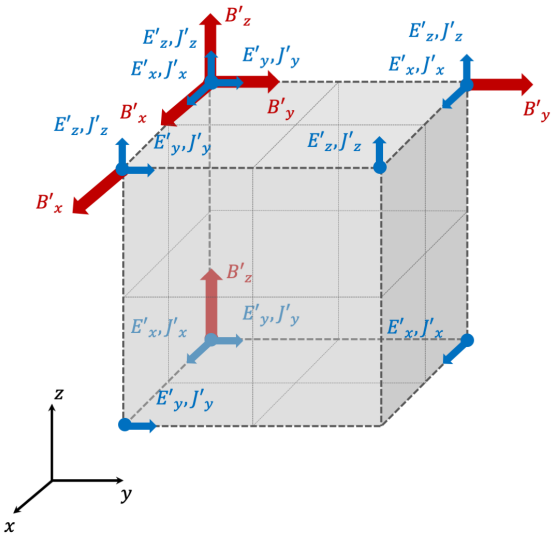


Figure 3: Layout of the filed components after re-sampling on the nodal points for a single cell.

where the complex exponential represents a phase shift by half a sample. In the Appendix, we sketch the proof as to why the multiplication with the complex exponential ends up being a re-sampling of the discrete time sequence. In the following section, we describe the order of operations and the notation to capture the packing and interpolation.

2 Mathematical Representation

2.1 Notation

In this part, we present the notation we use to express the PSATD algorithm. We start with the Kronecker product. For two dense square matrices $A \in \mathrm{R}^{m\times m}$ and $B \in \mathrm{R}^{n\times n}$, the Kronecker product is defined as

$$A \otimes B = \begin{bmatrix} a_{0,0}B & a_{0,1}B & \dots & a_{0,m-1}B \\ a_{1,0}B & a_{1,1}B & \dots & a_{1,m-1}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1,0}B & a_{m-1,1}B & \dots & a_{m-1,m-1}B \end{bmatrix}, \tag{13}$$

where $a_{i,j}$ are the elements within the A matrix and the product $a_{i,j}B$ represents the multiplication of each element of B with the scalar element $a_{i,j}$. The size of the resultant matrix is $mn \times mn$. The

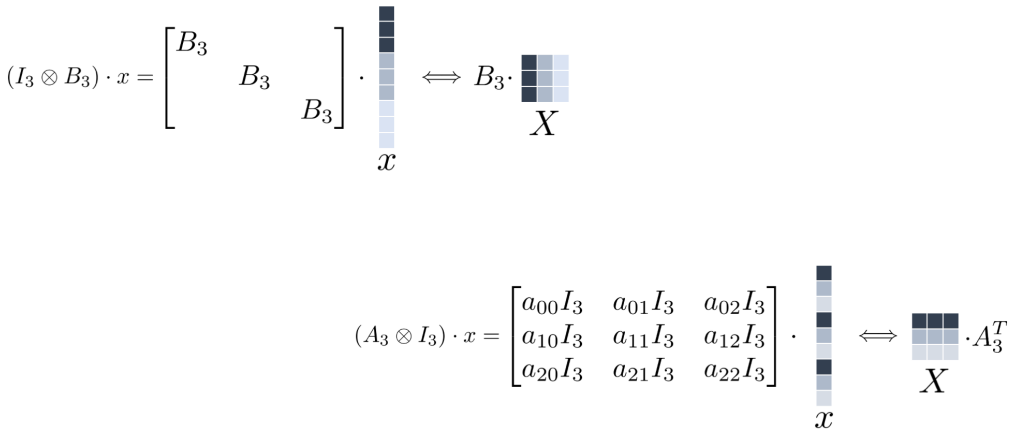


Figure 4: The pictorial representation of the constructs $I_m \otimes B$ and $A \otimes I_n$.

Kronecker product is used to express one and multi-dimensional Fourier transforms. More specifically, the Fourier transform uses the Kronecker product when one of the matrices is the identity matrix.

If the A matrix is replaced with the an identity matrix I_m, the Kronecker product changes such that

$$I_m \otimes B = \begin{bmatrix} B & O & \dots & O \\ O & B & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & B \end{bmatrix}, \tag{14}$$

where O is the zero matrix that has the same size as B. The $I_m \otimes B$ is applied on an input one dimensional array x of size mn. The operation can be viewed as applying the same operation B on contiguous chunks of x of size n, m times. If the matrix B is replaced by the identity matrix I_n then the Kronecker product changes again

$$A \otimes I_n = \begin{bmatrix} a_{0,0}I_n & \dots & a_{0,m-1}I_n \\ \vdots & \ddots & \vdots \\ a_{m-1,0}I_n & \dots & a_{m-1,m-1}I_n \end{bmatrix}. \tag{15}$$

Similar to the previous operation, the $A \otimes I_n$ is applied on an one dimensional array x of size mn. The operation can be viewed as applying the same operation A on m data points that are at a stride of n elements. The A operation is applied n times. Pictorially the two constructs can be seen in Figure 4.

The Kronecker product is separable. Given two matrix A and B, the tensor product between $A \in \mathrm{R}^{m\times m}$ and $B \in \mathrm{R}^{n\times n}$ can be separated as

$$A \otimes B = (A \otimes I_m) \cdot (I_n \otimes B), \tag{16}$$

where the identity matrices have the same size as A and B, respectively. This property suggests that the overall operation can be split into two stages, where matrix B is applied on contiguous data and matrix A is applied on strided data. The Kronecker product allows compute stages to be merged. Given two matrices $A \in \mathrm{R}^{m\times m}$ and $C \in \mathrm{R}^{m\times m}$, fusing computation is done as

$$(A \otimes I_m) \cdot (C \otimes I_m) = ((A \cdot C) \otimes I_m) \tag{17}$$

$$(I_m \otimes A) \cdot (I_m \otimes C) = (I_m \otimes (A \cdot C)) \tag{18}$$

Besides the identity matrix, we can define diagonal matrices. The diagonal matrix D_n defined as

$$D_n = \begin{bmatrix} d_0 & 0 & \dots & 0 \\ 0 & d_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_{n-1} \end{bmatrix}, \tag{19}$$

is the matrix where only the diagonal elements are non-zero. The diagonal matrix can be used to represent points-scaling operations.

Data needs to be shuffle. Therefore permutation matrices can be defined. For example, the L matrix defined as

$$L^{mn}_m : in + j \mapsto jm + i, \quad 0 \le i < m,\ 0 \le j < n \tag{20}$$

represents the stride permutation. The L operator is applied on a one dimensional array. However, viewing the input data as a two dimensional array of size $n \times m$, the L operator transposes the original matrix into a matrix of size $m \times n$. The L operator commutes the $A \in \mathrm{R}^{m\times m}$ and $B \in \mathrm{R}^{n\times n}$ matrix within the Kronecker product as follows

$$(A \otimes B) = L^{mn}_m (B \otimes A) L^{mn}_n \tag{21}$$

$$(A \otimes B) L^{mn}_n = L^{mn}_m (B \otimes A) \tag{22}$$

The Kronecker product can be used if the matrices A or B do not change. A more general construct that allows indexing is the direct sum $\bigoplus$ operator. Given m matrices $B^{(i)} \in \mathrm{R}^{n\times n}$, the direct sum is defined as

$$\bigoplus_{i=0}^{m-1} B^{(i)} = \begin{bmatrix} B^{(0)} & O & \dots & O \\ O & B^{(1)} & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & B^{(m-1)} \end{bmatrix}. \tag{23}$$

Each matrix $B^{(i)}$ is applied on disjoint contiguous data points of size n. If all the matrices $B^{(i)}$ are equal to the same matrix B, then the direct sum is identical to the Kronecker product $I_m \otimes B$.

The direct sum can also be used to describe the Kronecker-equivalent construct for $A \otimes I_n$, when the matrix $A \in \mathrm{R}^{m\times m}$ depends on an index value i. The permutation matrix L commutes the Kronecker product such that $A \otimes I_n = L_m^{mn}\,(I_n \otimes A)\,L_n^{mn}$. The Kronecker product can be replaced with the direct sum for the $A^{(i)} \in \mathrm{R}^{m\times m}$, such that

$$L_m^{mn} \cdot \left(\bigoplus_{i=0}^{n-1} A^{(i)} \right) \cdot L_n^{mn}. \tag{24}$$

Some operations require zero padding and un-padding. For example, the construct

$$I_{m\times n} = \begin{bmatrix} I_n \\ O_{m-n\times n} \end{bmatrix}, \quad m \geq n, \tag{25}$$

where $O_{m-n\times n}$ is a rectangular matrix with all elements equal to 0, defines the zero-padding operation. The matrix $I_{m\times n}$ reads an array of size n and appends $m-n$ 0s to it. The transposed version $I_{n\times m}$ where $m \geq n$ does the opposite, un-padds the array.

2.2 The PSATD Components

In the following paragraphs, we use the notation presented in the previous subsection and present the main components required to do the PSATD algorithm.

Packing. We use the $I_{m\times n}$ construct to represent the packing. In addition, we use the Kronecker product to specify the dimensions in which the packing/unpacking is done. For example, for the $\mathbf{B}_x$ component, which requires the data be packed from a three dimensional cube of size $(N+1)\times M\times K$

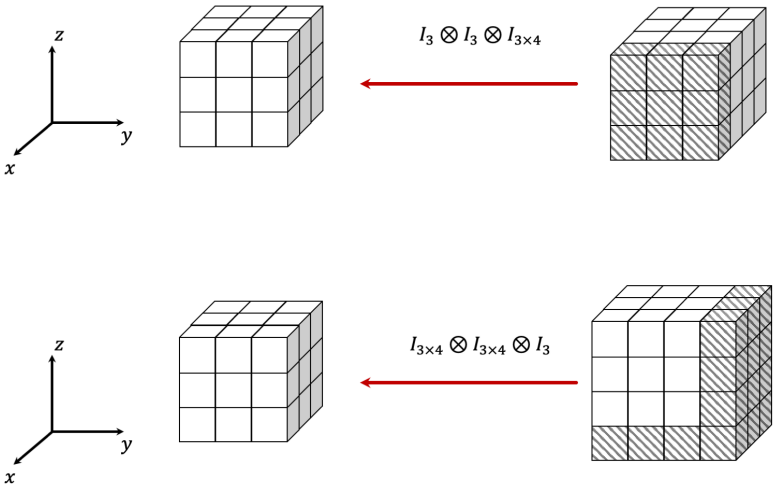


Figure 5: Two packing routines for a face centered and edge centered field of size $4 \times 3 \times 3$ and $3 \times 4 \times 4$, respectively. Both routines pack the data into a cube of size $3 \times 3 \times 3$

samples to a three dimensional cube of size $N \times M \times K$ samples, the packing is expressed as

$$I_K \otimes I_M \otimes I_{N\times(N+1)}. \tag{26}$$

Note that the packing routine is applied in the dimension that correspond to size N, which also represents the fastest dimension in memory. The dimension corresponding to the size K is laid out in the slowest dimension in memory.

For the packing routine of an edge centered field, the packing requires two constructs since the samples need to be reduced on two dimensions. For example, for the $\mathbf{E}_x$ component, which needs the data to be packed from a three dimensional cube of size $N \times (M+1) \times (K+1)$ samples to three dimensional cube of size $N \times M \times K$, the packing routine is expressed as

$$I_{K\times(K+1)} \otimes I_{M\times(M+1)} \otimes I_N. \tag{27}$$

Transposing the constructs gives the unpacking routines, operations that are required when updating the final result.

Fourier Computation. The DFT of size n is expressed as a dense complex matrix of size $n \times n$ such that

$$\mathrm{DFT}_n = [\omega_n^{k\ell}]_{0\le k,\ell<n} \quad \text{with} \quad \omega_n = \exp(-2\pi j/n). \tag{28}$$

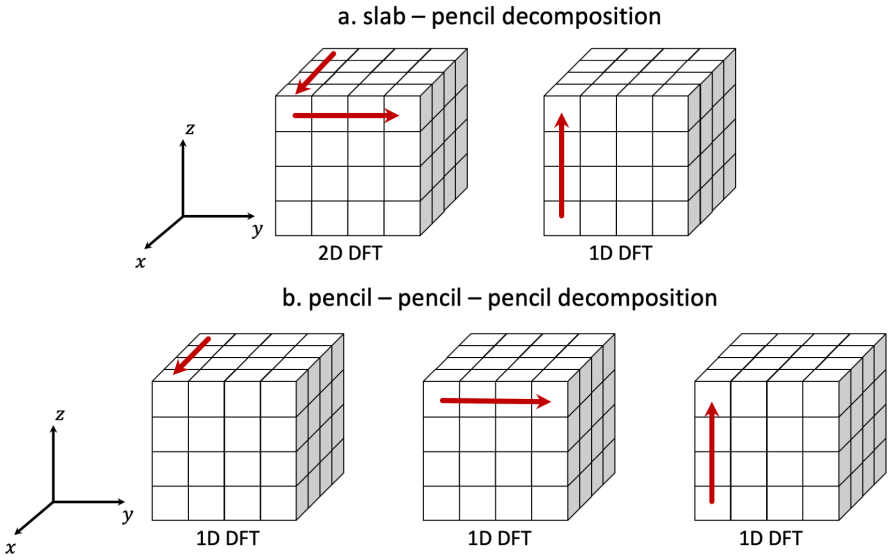


Figure 6: Slab-pencil and pencil-pencil algorithms for computing a 3D DFT of size $n_0 \times n_1 \times n_2$.

Computing the DFT as a matrix-vector multiplication incurs a $O(n^2)$ arithmetic complexity. Fast Fourier transform (FFT) algorithms, such as the Cooley-Tukey algorithm, reduce complexity to $O(n\log(n))$. The Cooley-Tukey algorithm works when the problem size n is a composite number. However, for prime numbers algorithms like Rader or Bluestein need to be used. Basically the Rader and Bluestein algorithm are DFT based convolutions on data that is padded with zeros.

The DFT matrix for a three dimensional DFT is represented as

$$\text{DFT}_{n_0 \times n_1 \times n_2} = \text{DFT}_{n_0} \otimes \text{DFT}_{n_1} \otimes \text{DFT}_{n_2}, \tag{29}$$

where n_0, n_1, n_2 represent the sizes of the three dimensions. One can use the properties of the Kronecker product to separate and group the stages together. Therefore algorithms such as the pencil-pencil or slab-pencil decomposition as seen in Figure 6.

Half Sample Interpolation. Re-sampling a discrete time sequence by half a sample is computed as specified in Equation 12. For the one dimensional sequence $x[n]$ of size N, the re-sampling is expressed using the matrix notation

$$\text{Interp}_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}}) = \text{iDFT}_N \cdot D_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}}) \cdot \text{DFT}_N, \tag{30}$$

where the DFT_N and iDFT_N are the forward and inverse Fourier transforms of size N, and the $D_n(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}})$ is a diagonal matrix. The diagonal matrix is defined as

$$D_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}}) = \begin{bmatrix} e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}\cdot 0} & 0 & \dots & 0 \\ 0 & e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}\cdot 1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}\cdot(N-1)} \end{bmatrix}. \tag{31}$$

The fields require re-sampling in different dimensions. Similar to how packing is expressed, the re-sampling is combined with the Kronecker product to specify the dimensions in which data needs to be re-sampled. For example, for the $\mathbf{B}_x$ component, the re-sampling is expressed as

$$\text{Interp}_K(e^{-j\frac{2\pi}{K}\cdot\frac{1}{2}}) \otimes \text{Interp}_M(e^{-j\frac{2\pi}{M}\cdot\frac{1}{2}}) \otimes I_N, \tag{32}$$

where the interpolation is applied on two dimensions that correspond to the K and M sizes. Similarly, for the $\mathbf{E}_x$ component the re-sampling is expressed as

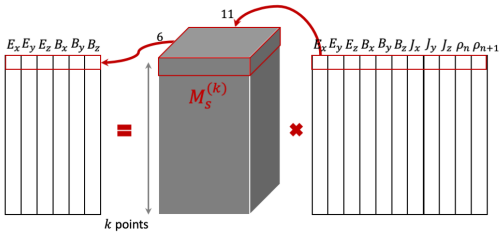
$$I_K \otimes I_M \otimes \text{Interp}_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}}), \tag{33}$$

where the interpolation is applied in one single dimension. Complex conjugating the constructs gives the inverse operations that shift the samples by half a sample in the opposite direction.

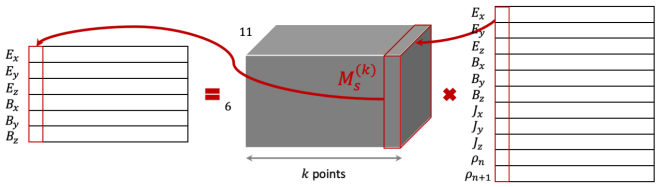
Transformation Matrix M_s. The transformation matrix M_s is constructed from the coefficients in Maxwell's equations from Equation 1. The coefficients are dependent on the frequency points k. The storage of the transformation matrix is dependent on the data layout of the fields, as shown in Figure 7. If the fields are stored as the columns in a two dimensional matrix, the transformation matrix M_s is a $K \times M \times N \times 6 \times 11$ tensor and it can be represented as matrix operations as

$$M_s = \bigoplus_{k=0}^{KMN-1} M_s^{(k)} = \begin{bmatrix} M_s^{(0)} & O & \dots & O \\ O & M_s^{(1)} & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & M_s^{(KMN-1)}, \end{bmatrix}. \tag{34}$$

where $M_s^{(k)}$ is a 6×11 matrix that is dependent on the frequency points, represented by the index k. Pictorially, each matrix $M_s^{(k)}$ is applied on the rows of the field matrix. If the fields are stored as rows in a two dimensional matrix, then the transformation matrix M_s is a five dimensional tensor of size



a. Storing the fields as **columns** of a 2D matrix



b. Storing the fields as **rows** of a 2D matrix

Figure 7: The pictorial representation of the tensor contraction operation given the two data layouts for the fields.

$6 \times 11 \times K \times M \times N$, which is represented as

$$M_s = L_6^{6KMN} \cdot \left(\bigoplus_{k=0}^{KMN-1} M_s^{(k)} \right) \cdot L_{KMN}^{11KMN} \tag{35}$$

$$= L_6^{6KMN} \cdot \begin{bmatrix} M_s^{(0)} & O & \dots & O \\ O & M_s^{(1)} & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & M_s^{(KMN-1)}, \end{bmatrix} \cdot L_{KMN}^{11KMN}.$$

If the matrix that stores the fields is not explicitly permuted by the two permutation matrices L_6^{6KMN} and L_{KMN}^{11KMN}, then the $M_s^{(k)}$ of size 6×11 is basically applied on the columns of the matrix.

2.3 Putting Everything Together

In the following paragraphs, we present the computation for two components, one that has the samples face centered and one that has the samples edge centered. We assume that the fields are stored as rows in a two dimensional matrix, so that the $M_s^{(i)}$ is applied in the column dimension. The reason for choosing this data layout is because both 6 and 11 are odd sizes, whereas cache line lengths, SIMD vector length and warp size are power of two numbers.

Face Centered Field. A field that is face centered requires packing in one dimension and interpolation in the other two dimensions. For example, the computation for the $\mathbf{B}_x$ component is expressed

using the above notation as

$$\underbrace{(\mathrm{DFT}_{K\times M\times N})}_{\text{Stage 3}}\cdot\underbrace{\left(\mathrm{Interp}_K(e^{-j\frac{2\pi}{K}\cdot\frac{1}{2}})\otimes\mathrm{Interp}_M(e^{-j\frac{2\pi}{M}\cdot\frac{1}{2}})\otimes I_N\right)}_{\text{Stage 2}}\cdot\underbrace{\left(I_K\otimes I_M\otimes I_{N\times(N+1)}\right)}_{\text{Stage 1}}. \tag{36}$$

The data is packed from a data cube of size $K\times M\times(N+1)$ into a cube of size $K\times M\times N$ in Stage 1. The second and third dimensions are re-sampled using the DFT-based interpolation in Stage 2. Finally, a three dimensional DFT is applied on the data in Stage 3.

Using the definitions, the multi-dimensional DFT and the other operations are expanded as

$$\begin{aligned}&(\mathrm{DFT}_K\otimes I_M\otimes I_N)\cdot(I_K\otimes\mathrm{DFT}_M\otimes I_N)\cdot(I_K\otimes I_M\otimes\mathrm{DFT}_N)\cdot\\&\left(\mathrm{Interp}_K(e^{-j\frac{2\pi}{K}\cdot\frac{1}{2}})\otimes I_M\otimes I_N\right)\cdot\left(I_K\otimes\mathrm{Interp}_M(e^{-j\frac{2\pi}{M}\cdot\frac{1}{2}})\otimes I_N\right)\cdot(I_K\otimes I_M\otimes I_N)\cdot\\&\left(I_K\otimes I_M\otimes I_{N\times(N+1)}\right).\end{aligned} \tag{37}$$

Using the properties of the Kronecker product the computation is grouped as

$$\begin{aligned}&(\mathrm{DFT}_K\otimes I_M\otimes I_N)\cdot\left(\mathrm{Interp}_K(e^{-j\frac{2\pi}{K}\cdot\frac{1}{2}})\otimes I_M\otimes I_N\right)\cdot\\&(I_K\otimes\mathrm{DFT}_M\otimes I_N)\cdot\left(I_K\otimes\mathrm{Interp}_M(e^{-j\frac{2\pi}{M}\cdot\frac{1}{2}})\otimes I_N\right)\cdot\\&(I_K\otimes I_M\otimes\mathrm{DFT}_N)\cdot(I_K\otimes I_M\otimes I_N)\cdot\left(I_K\otimes I_M\otimes I_{N\times(N+1)}\right).\end{aligned} \tag{38}$$

The operations are simplified and the number of stages are reduced by merging the computation as

$$\begin{aligned}&\left(\left(\mathrm{DFT}_K\cdot\mathrm{Interp}_K(e^{-j\frac{2\pi}{K}\cdot\frac{1}{2}})\right)\otimes I_M\otimes I_N\right)\cdot\\&\left(I_K\otimes\left(\mathrm{DFT}_M\cdot\mathrm{Interp}_M(e^{-j\frac{2\pi}{M}\cdot\frac{1}{2}})\right)\otimes I_N\right)\cdot\\&\left(I_K\otimes I_M\otimes\left(\mathrm{DFT}_N\cdot I_{N\times(N+1)}\right)\right).\end{aligned} \tag{39}$$

The interpolation is expanded as a forward DFT, point-wise computation and inverse DFT such that

$$\begin{aligned}&\left(\left(\mathrm{DFT}_K\cdot\mathrm{iDFT}_K\cdot D_K(e^{-j\frac{2\pi}{K}\cdot\frac{1}{2}})\cdot\mathrm{DFT}_K\right)\otimes I_M\otimes I_N\right)\cdot\\&\left(I_K\otimes\left(\mathrm{DFT}_M\cdot\mathrm{iDFT}_M\cdot D_M(e^{-j\frac{2\pi}{M}\cdot\frac{1}{2}})\cdot\mathrm{DFT}_M\right)\otimes I_N\right)\cdot\\&\left(I_K\otimes I_M\otimes\left(\mathrm{DFT}_N\cdot I_{N\times(N+1)}\right)\right).\end{aligned} \tag{40}$$

The property $DFT_M \cdot iDFT_M = I_M$ gives the final form of the computation as

$$\begin{aligned}&\left(\left(D_K(e^{-j\frac{2\pi}{K}\cdot\frac{1}{2}})\cdot \mathrm{DFT}_K\right)\otimes I_M\otimes I_N\right)\cdot \\ &\quad\left(I_K\otimes\left(D_M(e^{-j\frac{2\pi}{M}\cdot\frac{1}{2}})\cdot \mathrm{DFT}_M\right)\otimes I_N\right)\cdot \\ &\quad\left(I_K\otimes I_M\otimes\left(\mathrm{DFT}_N\cdot I_{N\times(N+1)}\right)\right).\end{aligned} \tag{41}$$

The computation has the packing routines and the point-wise point-wise computations merged with the one dimensional Fourier transforms. If the operations were separate, then the steps are the exact steps in the WarpX code. Applying a complex conjugate on the entire construct gives the operation required for the inverse step. That step is applied on the corresponding $\mathbf{B}_x$ once the tensor contraction is applied.

Edge Centered Field. A field that is edge centered requires packing in two dimensions and interpolation in the other dimension. For example, the computation for the $\mathbf{E}_x$ component is expressed using the above notation as

$$\underbrace{(\mathrm{DFT}_{K\times M\times N})}_{\text{Stage 3}}\cdot\underbrace{\left(I_K\otimes I_M\otimes \mathrm{Interp}_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}})\right)}_{\text{Stage 2}}\cdot\underbrace{\left(I_{K\times(K+1)}\otimes I_{M\times(M+1)}\otimes I_N\right)}_{\text{Stage 1}}. \tag{42}$$

The data is packed from a data cube of size $K\times M\times(N+1)$ into a cube of size $K\times M\times N$ in Stage 1. The second and third dimensions are re-sampled using the DFT-based interpolation in Stage 2. Finally, a three dimensional DFT is applied on the data in Stage 3.

Using the definitions, the multi-dimensional DFT and the other operations are expressed as

$$\begin{aligned}&(\mathrm{DFT}_K\otimes I_M\otimes I_N)\cdot(I_K\otimes \mathrm{DFT}_M\otimes I_N)\cdot(I_K\otimes I_M\otimes \mathrm{DFT}_N)\cdot \\ &\quad(I_K\otimes I_M\otimes I_N)\cdot(I_K\otimes I_M\otimes I_N)\cdot\left(I_K\otimes I_M\otimes \mathrm{Interp}_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}})\right)\cdot \\ &\quad\left(I_{K\times(K+1)}\otimes I_{M\times(M+1)}\otimes I_N\right).\end{aligned} \tag{43}$$

Using the properties of the Kronecker product the computation is grouped as

$$\begin{aligned}&(\mathrm{DFT}_K\otimes I_M\otimes I_N)\cdot(I_K\otimes I_M\otimes I_N)\cdot \\ &\quad(I_K\otimes \mathrm{DFT}_M\otimes I_N)\cdot(I_K\otimes I_M\otimes I_N)\cdot \\ &\quad(I_K\otimes I_M\otimes \mathrm{DFT}_N)\cdot\left(I_K\otimes I_M\otimes \mathrm{Interp}_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}})\right)\cdot\left(I_{K\times(K+1)}\otimes I_{M\times(M+1)}\otimes I_N\right).\end{aligned} \tag{44}$$

The operations are simplified and the number of stages is reduced by merging the computation as

$$
\begin{aligned}
&(\mathrm{DFT}_K \otimes I_M \otimes I_N) \cdot \\
&\quad (I_K \otimes \mathrm{DFT}_M \otimes I_N) \cdot \\
&\qquad \left(I_{K\times(K+1)} \otimes I_{M\times(M+1)} \otimes \left(\mathrm{DFT}_N \cdot \mathrm{Interp}_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}})\right)\right).
\end{aligned}
\tag{45}
$$

The interpolation is expanded as a forward DFT, point-wise computation and inverse DFT such that

$$
\begin{aligned}
&(\mathrm{DFT}_K \otimes I_M \otimes I_N) \cdot \\
&\quad (I_K \otimes \mathrm{DFT}_M \otimes I_N) \cdot \\
&\qquad \left(I_{K\times(K+1)} \otimes I_{M\times(M+1)} \otimes \left(\mathrm{DFT}_N \cdot \mathrm{iDFT}_N \cdot D_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}}) \cdot \mathrm{DFT}_N\right)\right).
\end{aligned}
\tag{46}
$$

The property that $DFT_N \cdot iDFT_N = I_M$. gives the final form of the computation as

$$
\begin{aligned}
&(\mathrm{DFT}_K \otimes I_M \otimes I_N) \cdot \\
&\quad (I_K \otimes \mathrm{DFT}_M \otimes I_N) \cdot \\
&\qquad \left(I_{K\times(K+1)} \otimes I_{M\times(M+1)} \otimes \left(D_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}}) \cdot \mathrm{DFT}_N\right)\right).
\end{aligned}
\tag{47}
$$

The computation has the packing routines and the point-wise point-wise computations merged with the one dimensional Fourier transforms. If the operations were separate, then the steps are the exact steps in the WarpX code. Applying a complex conjugate on the entire construct gives the operation required for the inverse step.

All Steps. The packing and the re-sampling is dependent on the field type and the components of the fields. For the magnetic field **B** one can spec out three different computations

1. for the $\mathbf{B}_x$ component the computation is expressed as

$$
\begin{aligned}
&\left(\left(D_K(e^{-j\frac{2\pi}{K}\cdot\frac{1}{2}}) \cdot \mathrm{DFT}_K\right) \otimes I_M \otimes I_N\right) \cdot \\
&\quad \left(I_K \otimes \left(D_M(e^{-j\frac{2\pi}{M}\cdot\frac{1}{2}}) \cdot \mathrm{DFT}_M\right) \otimes I_N\right) \cdot \\
&\qquad \left(I_K \otimes I_M \otimes \left(\mathrm{DFT}_N \cdot I_{N\times(N+1)}\right)\right)
\end{aligned}
\tag{48}
$$

2. for the $\mathbf{B}_y$ component the computation is expressed as

$$\begin{aligned}&\left(\left(D_K(e^{-j\frac{2\pi}{K}\cdot\frac{1}{2}})\cdot\mathrm{DFT}_K\right)\otimes I_M\otimes I_N\right)\cdot\\&\quad(I_K\otimes\mathrm{DFT}_M\otimes I_N)\cdot\\&\qquad\left(I_K\otimes I_{M\times(M+1)}\otimes\left(D_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}})\cdot\mathrm{DFT}_N\right)\right)\end{aligned}\tag{49}$$

3. for the $\mathbf{B}_z$ component the computation is expressed as

$$\begin{aligned}&(\mathrm{DFT}_K\otimes I_M\otimes I_N)\cdot\\&\quad\left(I_K\otimes\left(D_M(e^{-j\frac{2\pi}{M}\cdot\frac{1}{2}})\cdot\mathrm{DFT}_M\right)\otimes I_N\right)\cdot\\&\qquad\left(I_{K\times(K+1)}\otimes I_M\otimes\left(D_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}})\cdot\mathrm{DFT}_N\right)\right)\end{aligned}\tag{50}$$

Similar, for the electric field and current density, the packing, DFT computation and point-wise multiplication can be specified as

1. for the $\mathbf{E}_x$ and $\mathbf{J}_x$ components the computation is expressed as

$$\begin{aligned}&(\mathrm{DFT}_K\otimes I_M\otimes I_N)\cdot\\&\quad(I_K\otimes\mathrm{DFT}_M\otimes I_N)\cdot\\&\qquad\left(I_{K\times(K+1)}\otimes I_{M\times(M+1)}\otimes\left(D_N(e^{-j\frac{2\pi}{N}\cdot\frac{1}{2}})\cdot\mathrm{DFT}_N\right)\right).\end{aligned}\tag{51}$$

2. for the $\mathbf{E}_y$ and $\mathbf{J}_y$ components the computation is expressed as

$$\begin{aligned}&(\mathrm{DFT}_K\otimes I_M\otimes I_N)\cdot\\&\quad\left(I_K\otimes\left(D_M(e^{-j\frac{2\pi}{M}\cdot\frac{1}{2}})\cdot\mathrm{DFT}_M\right)\otimes I_N\right)\cdot\\&\qquad\left(I_{K\times(K+1)}\otimes I_M\otimes\left(\mathrm{DFT}_N\cdot I_{N\times(N+1)}\right)\right).\end{aligned}\tag{52}$$

3. for the $\mathbf{E}_z$ and $\mathbf{J}_z$ components the computation is expressed as

$$\begin{aligned}&\left(\left(D_K(e^{-j\frac{2\pi}{K}\cdot\frac{1}{2}})\cdot\mathrm{DFT}_K\right)\otimes I_M\otimes I_N\right)\cdot\\&\quad(I_K\otimes\mathrm{DFT}_M\otimes I_N)\cdot\\&\qquad\left(I_K\otimes I_{M\times(M+1)}\otimes\left(\mathrm{DFT}_N\cdot I_{N\times(N+1)}\right)\right).\end{aligned}\tag{53}$$

The plain 3D DFT computation

$$\begin{aligned}&(\mathrm{DFT}_K \otimes I_M \otimes I_N)\cdot \\ &(I_K \otimes \mathrm{DFT}_M \otimes I_N)\cdot \\ &(I_K \otimes I_M \otimes \mathrm{DFT}_N)\,.\end{aligned} \tag{54}$$

is applied on the ρ_n and ρ_{n+1}.

In general, the computation can be parameterized such that

$$\begin{aligned}&((D_K(vk_i)\cdot \mathrm{DFT}_K)\otimes I_M \otimes I_N)\cdot \\ &(I_K \otimes (D_M(vm_i)\cdot \mathrm{DFT}_M)\otimes I_N)\cdot \\ &\left(I_{K\otimes offk_i} \otimes I_{M\times offm_i} \otimes \left(D_N(vn_i)\cdot \mathrm{DFT}_N \cdot I_{N\times offn_i}\right)\right),\end{aligned} \tag{55}$$

where the values vk_i, vm_i, vn_i, $offk_i$, $offm_i$ and $offn_i$ are dependent on i, which represents the field number and field component. For the $\mathbf{B}_x$ i is equal to 0 and

$$\begin{aligned}vk_i = e^{-j\frac{2\pi}{K}\cdot\frac{1}{2}} \quad vm_i = e^{-j\frac{2\pi}{M}\cdot\frac{1}{2}} \quad vn_i = 1 \\ offk_i = K \quad offm_i = M \quad offn_i = N+1\end{aligned} \tag{56}$$

2.4 Optimization

The above computation is just one variant. For all the components, the computation is applied first on the dimension corresponding to size N, then on the dimensions that correspond to the M and K sizes, respectively. However, the order of operations can be permuted. The K dimension is done first, followed by the M dimension and finally the N dimension as

$$\begin{aligned}&(I_K \otimes I_M \otimes (D_N(vn_i)\cdot \mathrm{DFT}_N))\cdot \\ &(I_K \otimes (D_M(vm_i)\cdot \mathrm{DFT}_M)\otimes I_N)\cdot \\ &\left(\left(D_K(vk_i)\cdot \mathrm{DFT}_K \cdot I_{K\otimes offk_i}\right)\otimes I_{M\times offm_i} \otimes I_{N\times offn_i}\right),\end{aligned} \tag{57}$$

The change in the order of applying the computation because (1) optimizations between the DFT computation and the tensor contraction can be further done and (2) the Fourier transform is real.

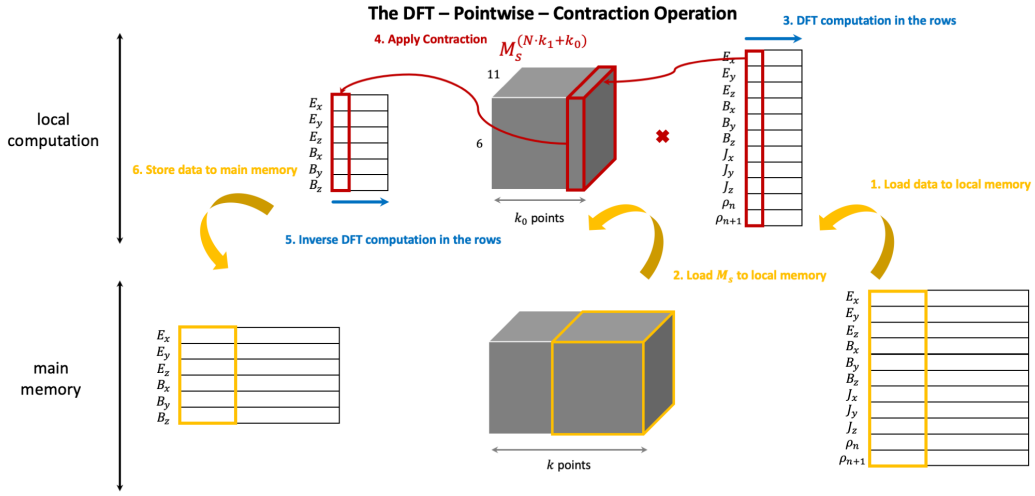


Figure 8: Merging the operations. The forward DFT computation is applied on 11 rows, the tensor contraction is applied on the columns, the inverse DFT computation is applied on 6 rows.

Merging the computation. Assuming the fields are stored as row in a matrix, the entire PSATD algorithm can be written as

$$
\begin{aligned}
&\bigoplus_{i=0}^{5} \Bigg[\Big(\big(I_{offk_i \times K} \cdot \mathrm{DFT}_K^* \cdot D_K(vk_i^*) \big) \otimes I_{offm_i \times M} \otimes I_{offn_i \times N} \Big) \cdot \\
&\quad \big(I_K \otimes (\mathrm{DFT}_M^* \cdot D_M(vm_i^*)) \otimes I_N \big) \cdot \\
&\quad \big(I_K \otimes I_M \otimes (\mathrm{DFT}_N^* \cdot D_N(vn_i^*)) \big) \Bigg] \cdot \\
&\quad L_6^{6KMN} \cdot \left(\bigoplus_{k=0}^{KMN-1} M_s^{(k)} \right) \cdot L_{KMN}^{11KMN} \cdot \\
&\quad \bigoplus_{i=0}^{10} \Bigg[\big(I_K \otimes I_M \otimes (D_N(vn_i) \cdot \mathrm{DFT}_N) \big) \cdot \\
&\quad\quad \big(I_K \otimes (D_M(vm_i) \cdot \mathrm{DFT}_M) \otimes I_N \big) \cdot \\
&\quad\quad \Big(\big(D_K(vk_i) \cdot \mathrm{DFT}_K \cdot I_{K \otimes offk_i} \big) \otimes I_{M \times offm_i} \otimes I_{N \times offn_i} \Big) \Bigg],
\end{aligned}
\tag{58}
$$

where the $(\cdot)^*$ represents the complex conjugate operation. Note that the for the forward and inverse Fourier computation the operations are in mirror image. If the forward computation applies the operations in the K, M and N dimensions, the inverse DFT computation applies the operation in N, M and K. This allows for the computation to be better aligned.

The last stage of the forward computation and the first stage of the inverse computation can be split from the main computation such that

$$\begin{aligned}
&\bigoplus_{i=0}^{5}\Bigg[\Big(\big(I_{offk_i\times K}\cdot \mathrm{DFT}_K^{*}\cdot D_K(vk_i^{*})\big)\otimes I_{offm_i\times M}\otimes I_{offn_i\times N}\Big)\cdot \\
&\qquad (I_K\otimes(\mathrm{DFT}_M^{*}\cdot D_M(vm_i^{*}))\otimes I_N)\Bigg]\cdot \\
&\qquad \bigoplus_{i=0}^{5}\Big[(I_K\otimes I_M\otimes(\mathrm{DFT}_N^{*}\cdot D_N(vn_i^{*})))\Big]\cdot \\
&\qquad L_6^{6KMN}\cdot\left(\bigoplus_{k=0}^{KMN-1}M_s^{(k)}\right)\cdot L_{KMN}^{11KMN}\cdot \\
&\qquad \bigoplus_{i=0}^{10}\Big[(I_K\otimes I_M\otimes(D_N(vn_i)\cdot\mathrm{DFT}_N))\Big]\cdot \\
&\qquad\quad \bigoplus_{i=0}^{10}\Bigg[(I_K\otimes(D_M(vm_i)\cdot\mathrm{DFT}_M)\otimes I_N)\cdot \\
&\qquad\qquad \Big(\big(D_K(vk_i)\cdot\mathrm{DFT}_K\cdot I_{K\otimes offk_i}\big)\otimes I_{M\times offm_i}\otimes I_{N\times offn_i}\Big)\Bigg],
\end{aligned}\tag{59}$$

The middle computation represented by

$$\begin{aligned}
&\bigoplus_{i=0}^{5}\Big[(I_K\otimes I_M\otimes(\mathrm{DFT}_N^{*}\cdot D_N(vn_i^{*})))\Big]\cdot \\
&L_6^{6KMN}\cdot\left(\bigoplus_{k=0}^{KMN-1}M_s^{(k)}\right)\cdot L_{KMN}^{11KMN}\cdot \\
&\bigoplus_{i=0}^{10}\Big[(I_K\otimes I_M\otimes(D_N(vn_i)\cdot\mathrm{DFT}_N))\Big]
\end{aligned}\tag{60}$$

can be merged using properties of the Kronecker product and the direct sum, such that

$$\begin{aligned}
&\Big(L_6^{6KM}\otimes I_N\Big)\cdot \\
&\left(\bigoplus_{k_1=0}^{KM-1}\left(\left(\bigoplus_{i=0}^{5}(\mathrm{DFT}_N^{*}\cdot D_N(vn_i^{*}))\right)\cdot L_6^{6N}\cdot\left(\bigoplus_{k_0=0}^{N-1}M_s^{(N\cdot k_1+k_0)}\right)\cdot L_N^{11N}\cdot\left(\bigoplus_{i=0}^{10}(D_N(vn_i)\cdot\mathrm{DFT}_N)\right)\right)\right)\cdot \\
&\Big(L_{KM}^{11KM}\otimes I_N\Big).
\end{aligned}\tag{61}$$

Pictorially the optimization can be viewed in Figure 8. The computation requires a buffer of $11*N$ complex data points for temporary storage. That buffer must reside in local memory to improve overall computation.

Computing the real DFT. TBD

3 Pseudo-Code

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
2
// temp field 0
tField0[0].size = 100;
tField0[0].off = 0;
tField0[0].dist = 1;

tField0[1].size = 100;
tField0[1].off = 0;
tField0[1].dist = 1;

tField0[2].size = 100;
tField0[2].off = 0;
tField0[2].dist = 1;

tField0[3].size = 1;
tField0[3].off = 0;
tField0[3].dist = 1;

// temp field 1
tField1[0].size = 51;
tField1[0].off = 0;
tField1[0].dist = 1;

tField1[1].size = 100;
tField1[1].off = 0;
tField1[1].dist = 1;

tField1[2].size = 100;
tField1[2].off = 0;
tField1[2].dist = 1;

tField1[3].size = 1;
tField1[3].off = 0;
tField1[3].dist = 1;
```

```
BoxShape in_Ex(4), in_Ey(4), in_Ez(4), in_Bx(4), in_By(4), in_Bz(4), in_Jx
    (4), in_Jy(4), in_Jz(4), in_rho0(4), in_rho1(4);

BoxShape temp_Ex(4), temp_Ey(4), temp_Ez(4), temp_Bx(4), temp_By(4),
    temp_Bz(4), temp_Jx(4), temp_Jy(4), temp_Jz(4), temp_rho0(4), temp_rho1
    (4);

BoxShape *in_fields, *temp_fields;
BoxShape

in_Ex[0].size = 100;
in_Ex[0].off = dx;
in_Ex[0].start = 0.5;



ObjShape gridShape(1);
gridShape[0] = 4;
const Grid g(MPI_COMM_WORLD, gridShape);

// create the shape and size of the data tensors
ObjShape shapeX(2);
shapeX[0] = 16;
shapeX[1] = 16;

// input and output tensors
DistTensor<complex<double>> A(shapeX, "[(0),()]", g);
DistTensor<complex<double>> B(shapeX, "[(),(0)]", g);

fftw_complex *ABuff = (fftw_complex*) A.Buffer();
fftw_complex *BBuff = (fftw_complex*) B.Buffer();

// batch DFT_16 in the rows
fftw_execute_dft(batch_4_dft_16, ABuff, ABuff);

// point-wise multiplication with twiddle factors
for(int i = 0; i != 64; ++i) {
    ABuff[i] = ABuff[i] * twidd_local[i];
}

// redistribution
```

```
B.AllToAllRedistFrom(A, 0, 0);

// transposition and batch DFT_16 in the rows
fftw_execute_dft(transpose_batch_4_dft_16, BBuff + , BBuff);
```

A Resample by Half a Sample

Let $x[n]$ be a one dimensional sequence of N samples. The $x_s[n] = x[n+1/2]$ can be computed as:

$$x_s[n] = \text{iDFT}\{e^{jk\frac{2\pi}{N}\frac{1}{2}} \cdot \text{DFT}\{x[n]\}\}, \forall 0 \leq n < N, \tag{62}$$

where the DFT and the iDFT represent the forward and inverse discrete Fourier transform. Multiplication in frequency domain with a complex exponential is equivalent to a shift in time domain. The complex exponential $e^{jk\frac{2\pi}{N}\frac{1}{2}}$ intuitively shifts the $x[n]$ in time domain by half a sample.

Proof:

Step 1. Construct two periodic sequences:

$$x_p[n] = x[n \bmod N]$$
$$x_{ps}[n] = x_s[n \bmod N].$$

Compute the frequency representation of the two periodic sequences using the discrete time Fourier series (DTFS) for one single period N as

$$X_p[k] = \sum_{n=0}^{N-1} x_p[n] e^{-j\frac{2\pi}{N}kn}, \forall 0 \leq k < N$$
$$X_{ps}[k] = \sum_{n=0}^{N-1} x_{ps}[n] e^{-j\frac{2\pi}{N}kn}, \forall 0 \leq k < N.$$

Note that the $X_p[k]$ and $X_{ps}[k]$ coefficients are the coefficients obtained using the discrete Fourier transform (DFT) on the two finite sequences $x[n]$ and $x_s[n]$.

Step 2. Relate the discrete periodic sequences with the corresponding continuous functions, such that

1. $x_p[n] = x_{pc}(nT)$, where $x_{pc}(t)$ is a continuous function and T represents the sampling period
2. $x_{ps}[n] = x_{psc}(nT)$, where $x_{psc}(t)$ is a continuous function and T represents the sampling period
3. $x_{psc}(t) = x_{pc}(t + T/2)$ where T represents the sampling period

Step 3. Apply the Fourier transform (FT) on $x_{psc}(t) = x_{pc}(t + T/2)$, such that:

$$\begin{aligned}
X_{psc}(j\omega) &= FT\{x_{psc}(t)\} \\
&= FT\{x_{pc}(t + T/2)\} \\
&= \int_{-\infty}^{\infty} x_{pc}(t + T/2)e^{-j\omega t}dt, \text{ let } s = t + T/2 \\
&= \int_{-\infty}^{\infty} x_{pc}(s)e^{-j\omega(s-T/2)}ds \\
&= e^{j\omega T/2}\int_{-\infty}^{\infty} x_{pc}(s)e^{-j\omega s}ds \\
&= e^{j\omega T/2}X_{pc}(j\omega)
\end{aligned}$$

Step 4. Use ideal reconstruction with a low pass sinc filter to express the relationship between the continuous function and the discrete samples. Use the frequency representation of the reconstructions express as

$$\begin{aligned}
X_{psc}(j\omega) &= H_f(j\omega)\text{FT}\{x_{ps}[n]\} \\
X_{pc}(j\omega) &= H_f(j\omega)\text{FT}\{x_p[n]\},
\end{aligned}$$

where H_f is the frequency representation of the ideal low pass sync filter.

Step 5. Replace the $X_{psc}(j\omega)$ and $X_{pc}(j\omega)$ terms from Step 4. in the equality from Step 3. as follows

$$\begin{aligned}
X_{psc}(j\omega) &= e^{j\omega T/2}X_{pc}(j\omega) \\
H_f(j\omega)\text{FT}\{x_{ps}[n]\} &= e^{j\omega T/2}H_f(j\omega)\text{FT}\{x_p[n]\} \\
H_f(j\omega)\left(\text{FT}\{x_{ps}[n]\} - e^{j\omega T/2}\text{FT}\{x_p[n]\}\right) &= 0, \forall\omega \\
\text{FT}\{x_{ps}[n]\} &= e^{j\omega T/2}\text{FT}\{x_p[n]\}
\end{aligned}$$

Step 6. Given that $x_p[n] = x_{pc}(nT)$ and $x_{ps}[n] = x_{psc}(nT)$, compute the Fourier transform (FT) of the discrete time sequences using Dirac δ functions such that

$$\begin{aligned}
\text{FT}\{x_{ps}[n]\} &= \frac{2\pi}{T}\sum_{k=-\infty}^{\infty} X_{ps}[k]\delta\left(\omega - k\frac{2\pi}{NT}\right) \\
\text{FT}\{x_p[n]\} &= \frac{2\pi}{T}\sum_{k=-\infty}^{\infty} X_p[k]\delta\left(\omega - k\frac{2\pi}{NT}\right),
\end{aligned}$$

where $X_{ps}[k]$ and $X_p[k]$ are the Fourier coefficients of the periodic discrete time sequences $x_{ps}[n]$ and $x_p[n]$, respectively.

Step 7. Replace the expressions of $\mathrm{FT}\{x_{ps}[n]\}$ and $\mathrm{FT}\{x_p[n]\}$ from Step 6. in the last expression from Step 5.

$$\begin{aligned}
\mathrm{FT}\{x_{ps}[n]\} &= e^{j\omega T/2}\mathrm{FT}\{x_p[n]\} \\
\frac{2\pi}{T}\sum_{k=-\infty}^{\infty} X_{ps}[k]\delta\left(\omega - k\frac{2\pi}{NT}\right) &= e^{j\omega T/2}\frac{2\pi}{T}\sum_{k=-\infty}^{\infty} X_p[k]\delta\left(\omega - k\frac{2\pi}{NT}\right) \\
\sum_{k=-\infty}^{\infty} X_{ps}[k]\delta\left(\omega - k\frac{2\pi}{NT}\right) &= e^{j\omega T/2}\sum_{k=-\infty}^{\infty} X_p[k]\delta\left(\omega - k\frac{2\pi}{NT}\right) \\
\sum_{k=-\infty}^{\infty} X_{ps}[k]\delta\left(\omega - k\frac{2\pi}{NT}\right) &= \sum_{k=-\infty}^{\infty} X_p[k]e^{j\omega T/2}\delta\left(\omega - k\frac{2\pi}{NT}\right) \\
\sum_{k=-\infty}^{\infty} X_{ps}[k]\delta\left(\omega - k\frac{2\pi}{NT}\right) &= \sum_{k=-\infty}^{\infty} X_p[k]e^{jk\frac{2\pi}{NT}T/2}\delta\left(\omega - k\frac{2\pi}{NT}\right) \\
\sum_{k=-\infty}^{\infty} X_{ps}[k]\delta\left(\omega - k\frac{2\pi}{NT}\right) &= \sum_{k=-\infty}^{\infty} X_p[k]e^{jk\frac{2\pi}{N}\frac{1}{2}}\delta\left(\omega - k\frac{2\pi}{NT}\right) \\
\sum_{k=-\infty}^{\infty}\left(X_{ps}[k] - X_p[k]e^{jk\frac{2\pi}{N}\frac{1}{2}}\right)\delta\left(\omega - k\frac{2\pi}{NT}\right) &= 0
\end{aligned}$$

This implies that

$$X_{ps}[k] = X_p[k]e^{jk\frac{2\pi}{N}\frac{1}{2}}, \forall 0 \leq k < N$$

The coefficients are the DFT coefficients of the two discrete sequences $x_s[n]$ and $x[n]$.

$$\begin{aligned}
\mathrm{DFT}\{x_s[n]\} &= e^{jk\frac{2\pi}{N}\frac{1}{2}}\mathrm{DFT}\{x[n]\} \\
x_s[n] &= \mathrm{iDFT}\{e^{jk\frac{2\pi}{N}\frac{1}{2}}\mathrm{DFT}\{x[n]\}\}, \forall 0 \leq n < N.
\end{aligned}$$