

Introduction to Practical FFT and NFFT

Michael Pippig and Daniel Potts

Faculty of Mathematics
Chemnitz University of Technology

07.09.2010

supported by BMBF grant 01IH08001B

Table of Contents

- 1 Serial FFT Algorithms
- 2 Parallel FFT Algorithms
- 3 Parallel Fast Summation

Outline

- 1 **Serial FFT Algorithms**
- 2 Parallel FFT Algorithms
- 3 Parallel Fast Summation

Discrete Fast Fourier Transform

Task of 3d-DFT

Consider a three-dimensional dataset of $n_0 \times n_1 \times n_2$ complex numbers $\hat{g}_{k_0 k_1 k_2} \in \mathbb{C}$. Compute

$$g_{l_0 l_1 l_2} = \sum_{k_0=0}^{n_0-1} \sum_{k_1=0}^{n_1-1} \sum_{k_2=0}^{n_2-1} \hat{g}_{k_0 k_1 k_2} e^{-2\pi i (k_2 \frac{l_2}{n_2} + k_1 \frac{l_1}{n_1} + k_0 \frac{l_0}{n_0})}$$

for all $l_t = 0, \dots, n_t - 1$ ($t = 0, 1, 2$).

Realized by 3d-FFT ($n_0 = n_1 = n_2 = n$)

$\Rightarrow \mathcal{O}(n^3 \log n)$ instead of $\mathcal{O}(n^6)$

FFT Software Libraries

Examples of FFT implementations

- IBM ESSL
- Intel MKL
- FFTW

Features of FFTW [Frigo, Johnson]

- public available
- open source
- high performance
- many transforms
- arbitrary size
- d -dim. FFT
- in place FFT
- collect wisdom
- adjust planning
- easy interface

Available at
<http://www.fftw.org>

Using FFTW

FFTW

Plan - only once

- hardware adaptive
- time consuming



Execute - several times

- fast transform



Finalize - only once

- free memory

FFTW_ESTIMATE

- heuristic choice of algorithm

FFTW_MEASURE

- compare different algorithms

FFTW_PATIENT

- compare more algorithms

FFTW_EXHAUSTIVE

- compare all available algorithms

FFTW Interface

Basic Interface

- simple
- do a single transform

Advanced Interface

- do a transform on multiple datasets by one call
- supports strided input and output
- use a plan on different datasets

Guru Interface

- most powerful and most complicated
- combine transform and data permutation
- 64 bit compatible

Nonequispaced Discrete Fourier Transform

Task of 3d-DFT and 3d-NDFT

For $\hat{f}_{k_0 k_1 k_2} \in \mathbb{C}$ compute

$$f_{l_0 l_1 l_2} := \sum_{k_0=0}^{N-1} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \hat{f}_{k_0 k_1 k_2} e^{-2\pi i(k_2 \frac{l_2}{N} + k_1 \frac{l_1}{N} + k_0 \frac{l_0}{N})} \quad (\text{DFT})$$

for all $0 \leq l_t < N$ ($\Rightarrow 0 \leq \frac{l_t}{N} < 1$), $t = 0, 1, 2$, and compute

$$f_j := \sum_{k_0=0}^{N-1} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \hat{f}_{k_0 k_1 k_2} e^{-2\pi i(k_2 x_j^{(2)} + k_1 x_j^{(1)} + k_0 x_j^{(0)})} \quad (\text{NDFT})$$

for $x_j^{(t)} \in [0, 1)$ ($t = 0, 1, 2$), $j = 1, \dots, M$.

Realized by 3d-NFFT [NFFT software library]

$\Rightarrow \mathcal{O}(N^3 \log N + \log^3(\frac{1}{\epsilon})M)$ instead of $\mathcal{O}(N^3 M)$

Matrix-Vector-Notation

Compute $\mathbf{f} = (f_j)_j \in \mathbb{C}^M$ via

$$\mathbf{f} = \mathbf{A}\hat{\mathbf{f}},$$

where $\mathbf{A} = (e^{-2\pi i \mathbf{k} \mathbf{x}_j})_{\mathbf{k}, j} \in \mathbb{C}^{M \times N^3}$ and $\hat{\mathbf{f}} = (\hat{f}_{k_0 k_1 k_2})_{\mathbf{k}} \in \mathbb{C}^{N^3}$.

Approximation [Dutt, Rohklin 93, Beylkin 95, Steidl 96, ...]

$$\mathbf{A} \approx \mathbf{B}\mathbf{F}\mathbf{D}, \quad \mathbf{A}^H \approx \mathbf{D}\mathbf{F}^H\mathbf{B}^T$$

where

- $\mathbf{D} \in \mathbb{R}^{N^3 \times N^3}$ diagonal matrix
- $\mathbf{F} \in \mathbb{C}^{n^3 \times N^3}$ truncated Fourier matrix ($n \geq N$)
- $\mathbf{B} \in \mathbb{R}^{M \times n^3}$ sparse matrix

NFFT Software Library [Keiner, Kunis, Potts]

Available at

<http://www.tu-chemnitz.de/~potts/nfft>

Using NFFT

FFTW

Plan



Execute



Finalize

NFFT

Plan



Precompute



Execute



Finalize

NFFT Precompute

PRE_FULL_PSI

- fully precomputed window function
- Storage: $(2m + 2)^d M$, Computation: None

PRE_PSI

- tensor product based precomputation
- Storage: $d(2m + 2)M$, Computation: $(2m + 2)^d M$

PRE_LIN_PSI

- linear interpolation from lookup table
- Storage: dK , Computation: $2(2m + 2)^d M$

PRE_FG_PSI

- fast Gaussian gridding
- Storage: $2dM$, Computation: $(2m + 2)^d M$

Outline

- 1 Serial FFT Algorithms
- 2 Parallel FFT Algorithms**
- 3 Parallel Fast Summation

Discrete Fast Fourier Transform

Task of 3d-DFT

Consider a three-dimensional dataset of $n_0 \times n_1 \times n_2$ complex numbers $\hat{g}_{k_0 k_1 k_2} \in \mathbb{C}$. Compute

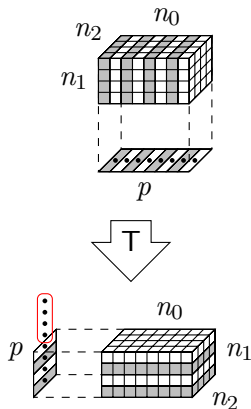
$$\begin{aligned} g_{l_0 l_1 l_2} &:= \sum_{k_0=0}^{n_0-1} \sum_{k_1=0}^{n_1-1} \sum_{k_2=0}^{n_2-1} \hat{g}_{k_0 k_1 k_2} e^{-2\pi i(k_2 \frac{l_2}{n_2} + k_1 \frac{l_1}{n_1} + k_0 \frac{l_0}{n_0})} \\ &= \sum_{k_0=0}^{n_0-1} \left(\sum_{k_1=0}^{n_1-1} \sum_{k_2=0}^{n_2-1} \hat{g}_{k_0 k_1 k_2} e^{-2\pi i(k_2 \frac{l_2}{n_2} + k_1 \frac{l_1}{n_1})} \right) e^{-2\pi i k_0 \frac{l_0}{n_0}} \end{aligned}$$

for all $l_t = 0, \dots, n_t - 1$ ($t = 0, 1, 2$).

Realized by 3d-FFT ($n_0 = n_1 = n_2 = n$)

$\Rightarrow \mathcal{O}(n^3 \log n)$ instead of $\mathcal{O}(n^6)$

One-Dimensional Data Distribution



p - number of processors

Features of FFTW [Frigo, Johnson]

- open source
- easy interface
- communicator
- arbitrary size
- d -dim. FFT
- in place FFT
- high performance
- many transforms
- adjust blocksize
- adjust planning
- collect wisdom

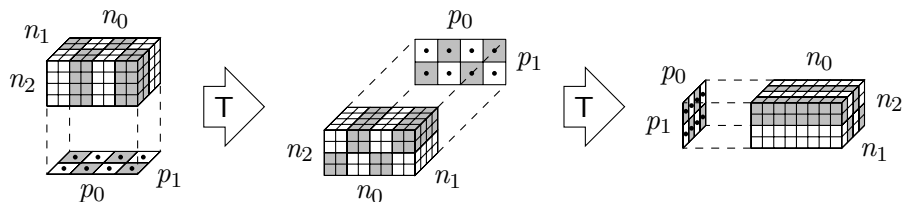
Maximum Number of Processors p_{\max}^{1D} ($n_0 = n_1 = n_2 = n$)

$$p_{\max}^{1D} = n$$

FFTW combines portable performance and good usability, but is not scalable enough.

Two-Dimensional Data Distribution

[Ding, Eleftheriou et al. 03, Plimpton, Pekurovsky - P3DFFT]



$p_0 \times p_1$ - size of processor grid

Maximum Number of Processors p_{\max}^{2D}
($n_0 = n_1 = n_2 = n$)

$$p_{\max}^{2D} = n^2$$

n	$p_{\max}^{1D} = n$	$p_{\max}^{2D} = n^2$
64	64	4096
128	128	16384
256	256	65536
512	512	262144
1024	1024	1048576

Algorithms Supported by FFTW3.3alpha1

Aim

Implement a new parallel FFT software library (PFFT) based on FFTW and the highly scalable two-dimensional data distribution.

1d-FFT Combined with Local Transposition

$$\begin{array}{lcl} \hat{n}_0 \times \hat{n}_1 \times \hat{n}_2 & \begin{array}{c} \text{FFT2} \\ \rightarrow \\ 012 \end{array} & \hat{n}_0 \times \hat{n}_1 \times n_2 \\ \hat{n}_0 \times \hat{n}_1 \times \hat{n}_2 & \begin{array}{c} \text{FFT2} \\ \rightarrow \\ 102 \end{array} & \hat{n}_1 \times \hat{n}_0 \times n_2 \\ \hat{n}_0 \times \hat{n}_1 \times \hat{n}_2 & \begin{array}{c} \text{FFT2} \\ \rightarrow \\ 021 \end{array} & \hat{n}_0 \times n_2 \times \hat{n}_1 \\ (\hat{n}_0 \times \hat{n}_1) \times \hat{n}_2 & \begin{array}{c} \text{FFT2} \\ \rightarrow \\ 201 \end{array} & n_2 \times (\hat{n}_0 \times \hat{n}_1) \end{array}$$

Algorithms Supported by FFTW3.3alpha1

Transposition of One-Dimensional Distributed Data

$$N_0 \times \frac{N_1}{P} \xrightarrow{\mathbf{T}} \frac{N_0}{P} \times N_1$$

Group two of the three dimensions to use FFTWs matrix transposition on two-dimensional decomposed data, e.g.

$$N_0 = n_2, N_1 = n_0 \times \frac{n_1}{p_1}, P = p_0.$$

Transposition of Two-Dimensional Distributed Data

$$n_2 \times \left(\frac{n_0}{p_0} \times \frac{n_1}{p_1} \right) \xrightarrow{\mathbf{T}} \frac{n_2}{p_0} \times \left(n_0 \times \frac{n_1}{p_1} \right)$$

Two-Dimensional Distributed FFT Based on FFTW

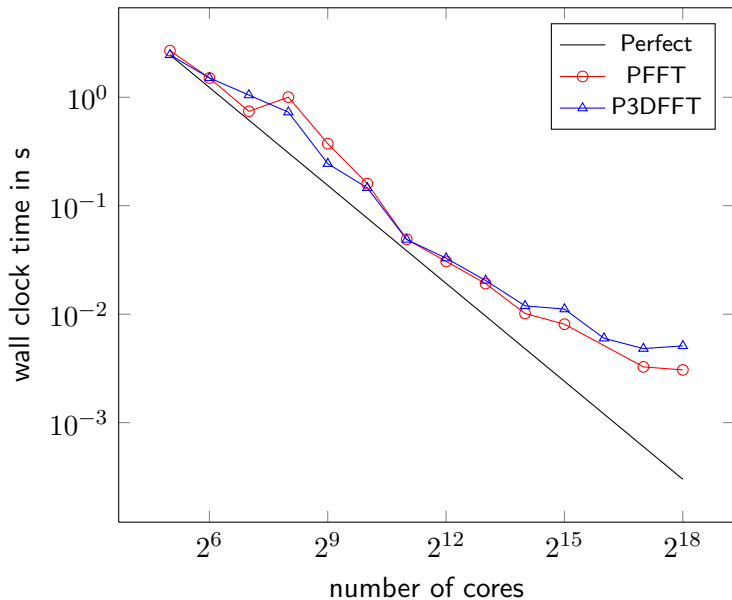
PFFT Forward Transform

$$\begin{array}{ccc} \frac{\hat{n}_0}{p_0} \times \frac{\hat{n}_1}{p_1} \times \hat{n}_2 & \xrightarrow{\text{FFT2}} & n_2 \times \frac{\hat{n}_0}{p_0} \times \frac{\hat{n}_1}{p_1} \xrightarrow{\text{T}} \\ \frac{n_2}{p_1} \times \frac{\hat{n}_0}{p_0} \times \hat{n}_1 & \xrightarrow{\text{FFT2}} & n_1 \times \frac{n_2}{p_1} \times \frac{\hat{n}_0}{p_0} \xrightarrow{\text{T}} \\ \frac{n_1}{p_0} \times \frac{n_2}{p_1} \times \hat{n}_0 & \xrightarrow{\text{FFT2}} & \frac{n_2}{p_1} \times \frac{n_1}{p_0} \times n_0 \end{array}$$

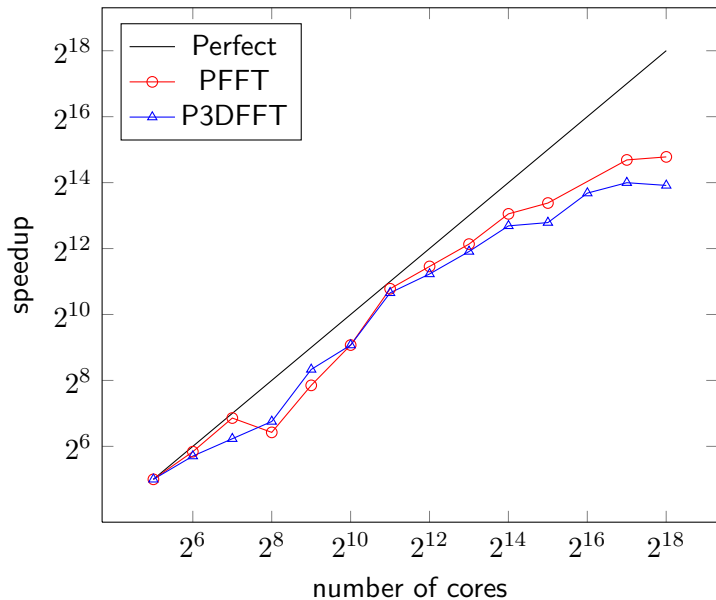
PFFT Backward Transform

$$\begin{array}{ccc} \frac{n_2}{p_1} \times \frac{n_1}{p_0} \times n_0 & \xrightarrow{\text{FFT2}} & \hat{n}_0 \times \frac{n_2}{p_1} \times \frac{n_1}{p_0} \xrightarrow{\text{T}} \\ \frac{\hat{n}_0}{p_0} \times \frac{n_2}{p_1} \times n_1 & \xrightarrow{\text{FFT2}} & \hat{n}_1 \times \frac{\hat{n}_0}{p_0} \times \frac{n_2}{p_1} \xrightarrow{\text{T}} \\ \frac{\hat{n}_1}{p_1} \times \frac{\hat{n}_0}{p_0} \times n_2 & \xrightarrow{\text{FFT2}} & \frac{\hat{n}_0}{p_0} \times \frac{\hat{n}_1}{p_1} \times \hat{n}_2 \end{array}$$

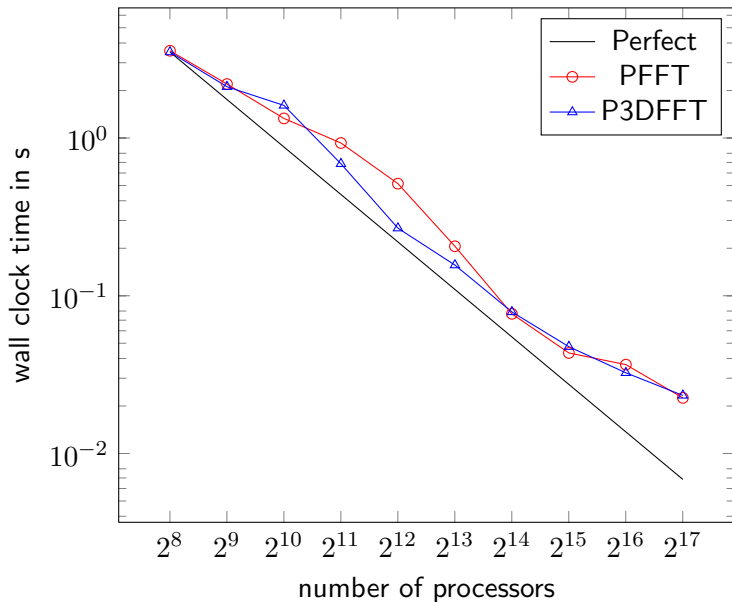
Scaling FFT of Size 512^3 on BlueGene/P



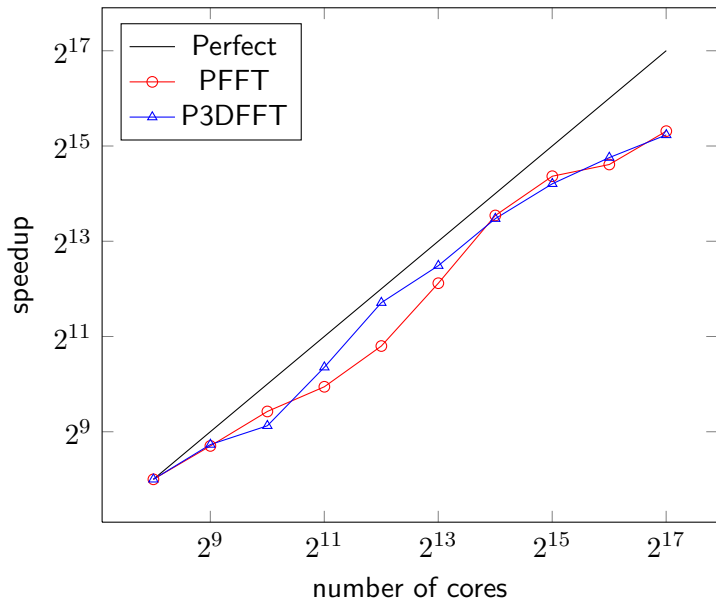
Scaling FFT of Size 512^3 on BlueGene/P



Scaling FFT of Size 1024^3 on BlueGene/P



Scaling FFT of Size 1024^3 on BlueGene/P



Comparison of PFFT and P3DFFT

P3DFFT Unique Features [Pekurovsky]

- r2c FFT
- Fortran interface

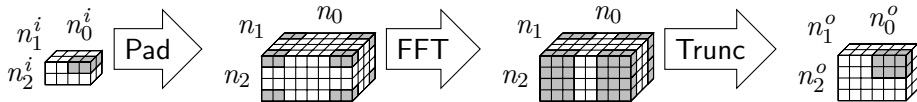
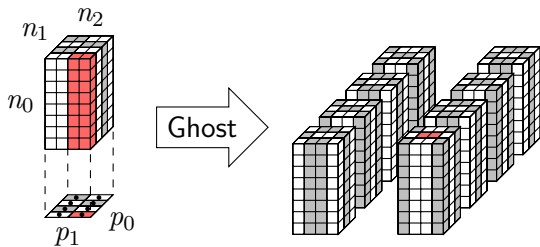
Common Features

- open source
- high scalability
- portability
- multiple precisions
- C interface
- ghost cell support

PFFT Unique Features

- c2c FFT
- completely in place FFT
- FFTW like interface
- basic, advanced and guru interface
- adjustable blocksize
- separate communicator
- accumulated wisdom
- change of planning effort without recompilation
- d -dimensional parallel FFT
- truncated FFT support

Ghost Cell Support, Truncated FFT



PFFT Software Library [Pippig]

Available at

<http://www.tu-chemnitz.de/~mpip/software>

Parallel NFFT

Matrix-Vector-Notation

Compute $\mathbf{f} = (f_j)_j \in \mathbb{C}^M$ via

$$\mathbf{f} = \mathbf{A}\hat{\mathbf{f}},$$

where $\mathbf{A} = (e^{-2\pi i \mathbf{k} \mathbf{x}_j})_{\mathbf{k}, j} \in \mathbb{C}^{M \times N^3}$ and $\hat{\mathbf{f}} = (\hat{f}_{k_0 k_1 k_2})_{\mathbf{k}} \in \mathbb{C}^{N^3}$.

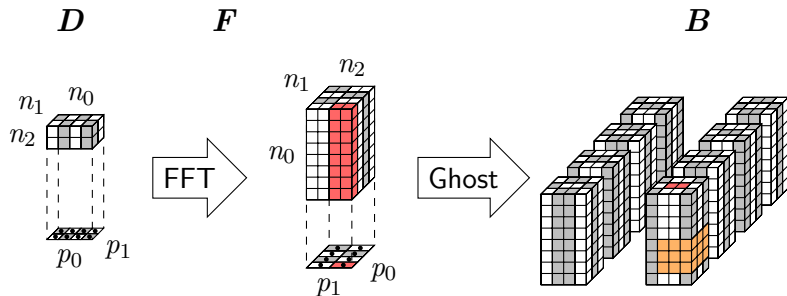
Approximation [Dutt, Roklin 93, Beylkin 95, Steidl 96, ...]

$$\mathbf{A} \approx \mathbf{B}\mathbf{F}\mathbf{D}, \quad \mathbf{A}^H \approx \mathbf{D}\mathbf{F}^H\mathbf{B}^T$$

where

- $\mathbf{D} \in \mathbb{R}^{N^3 \times N^3}$ diagonal matrix
- $\mathbf{F} \in \mathbb{C}^{n^3 \times N^3}$ truncated Fourier matrix ($n \geq N$)
- $\mathbf{B} \in \mathbb{R}^{M \times n^3}$ sparse matrix

PNFFT in Pictures



PNFFT Software Library [Pippig]

Available at

<http://www.tu-chemnitz.de/~mpip/software>

Outline

- 1 Serial FFT Algorithms
- 2 Parallel FFT Algorithms
- 3 Parallel Fast Summation**

Fast Summation

Task

Fast computation of

$$h_j = \sum_{l=1}^L a_k K(\|\mathbf{y}_j - \mathbf{x}_l\|_2), \quad j = 1, \dots, M, \quad \mathbf{y}_j, \mathbf{x}_l \in \mathbb{R}^3$$

Example of Radial Kernel Function

$$K(\|\mathbf{x}\|_2) = \frac{1}{\|\mathbf{x}\|_2}, \dots$$

Realized with 3d-NFFT [NFFT Software Library]

$\Rightarrow \mathcal{O}(\log^3(\frac{1}{\epsilon})(M + L))$ instead of $\mathcal{O}(ML)$

Parallel Fast Summation

Matrix-Vector-Notation

Compute $\mathbf{h} = (h_j)_{j=1}^M$ via

$$\mathbf{h} = \mathbf{K} \mathbf{a},$$

where $\mathbf{K} = (K(\|\mathbf{y}_j - \mathbf{x}_l\|))_{j,l=1}^{M,M}$ and $\mathbf{a} = (a_l)_{l=1}^M \in \mathbb{C}^M$.

Standard Algorithm for Equispaced Nodes

$$\mathbf{K} = \mathbf{F} \mathbf{D} \mathbf{F}^H$$

where

- $\mathbf{F} \in \mathbb{C}^{M \times M}$ equispaced Fourier matrix
- $\mathbf{D} \in \mathbb{C}^{M \times M}$ diagonal matrix

Parallel Fast Summation

Matrix-Vector-Notation

Compute $\mathbf{h} = (h_j)_{j=1}^M$ via

$$\mathbf{h} = \mathbf{K} \mathbf{a},$$

where $\mathbf{K} = (K(\|\mathbf{y}_j - \mathbf{x}_l\|))_{j,l=1}^{M,L}$ and $\mathbf{a} = (a_l)_{l=1}^L \in \mathbb{C}^L$.

Approximation [Potts, Steidl, Nieslony 2004]

$$\mathbf{K} \approx \mathbf{A}_2 \mathbf{D} \mathbf{A}_1^H + \mathbf{K}_{NF}$$

where

- $\mathbf{A}_2 = (e^{-2\pi i \mathbf{k} \mathbf{y}_j})_{j,\mathbf{k}} \in \mathbb{C}^{M \times N^3}$ nonequispaced Fourier matrix
- $\mathbf{D} \in \mathbb{C}^{N^3 \times N^3}$ diagonal matrix
- $\mathbf{A}_1 = (e^{-2\pi i \mathbf{k} \mathbf{x}_l})_{l,\mathbf{k}} \in \mathbb{C}^{L \times N^3}$ nonequispaced Fourier matrix
- $\mathbf{K}_{NF} \in \mathbb{C}^{M \times L}$ sparse near field correction

Summary

