Using the Fast Multipole Method to Accelerate Matrix-Vector Products

Antoine Gicquel, Olivier Coulaud

Using the Fast Multipole Method to Accelerate Matrix-Vector Products

- Application to the Magnetostatic Moment Method -

Antoine GICQUEL & Olivier COULAUD

INRIA Bordeaux (Concace team)

SIAM LA24, May 2024

Introduction

Context

Electrical engineering

- **Importance of numerical modeling methods**
  - Small electrical devices (electric motors, printed circuit boards . . .)
  - Rapid prototyping
  - Shape optimization
  - Parametric study

TensorVim ANR (*French National Research Agency*) project

- **Objective:** Developing efficient iterative methods for Magnetostatic problems
  - Compression techniques (FMM, H-matrix. . .)
  - GMRES preconditioners (tensor decomposition. . .)

- **Joint work**

Introduction

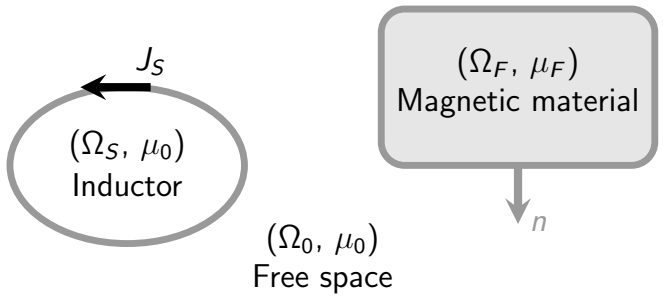
Problem statement

Magnetostatic problem

- Goal: *Find* $\boldsymbol{H}$ *in* Ω
- Static Maxwell's equations (magnetism only):

$$\begin{cases} \nabla.\boldsymbol{B}(x) & = 0 \\ \nabla \times \boldsymbol{H}(x) & = \boldsymbol{J_S}(x) \end{cases} \qquad \forall x \in \Omega$$

- Constitutive law: $\boldsymbol{M} = (\mu_r - 1)\boldsymbol{H}$



Model magnetostatic problem
$\Omega = \Omega_0 \cup \Omega_F \cup \Omega_S$

Solving approaches

- Finite Element Method
  ↪ artificial boundary, Ω is entirely meshed, large sparse matrix
- Volume Integral Methods (*e.g., the Magnetostatic Moment Method (MMM)*)
  ↪ only the magnetic material Ω_F is meshed, less unknowns, small dense matrix

Magnetic vector formulation [Chadebec et al. 2006]

$$\boldsymbol{M}(x) + \int_{\Omega_F} \frac{1-\mu_r(y)}{4\pi} \left(\nabla_y \frac{(y-x).\boldsymbol{M}(y)}{\|y-x\|^3} \right) dy = (\mu_r(x) - 1)\boldsymbol{H_S}(x) \qquad \forall x \in \Omega_F.$$

Introduction

Magnetostatic Moment Method (MMM)

Discretization

The magnetic material Ω_F is meshed with N elements $\Omega_{Fh} = \cup_{i=1}^{N} V_i$

- P_0 approximation of $\boldsymbol{M} = (\boldsymbol{M}_x, \boldsymbol{M}_y, \boldsymbol{M}_z)^T$
- Collocation formulation ($\boldsymbol{M}^i = \boldsymbol{M}(x_i)$) with x_i the barycenter of V_i (i = 1,. . . ,N)

$$(\boldsymbol{AM})(x_i) = (\boldsymbol{AM})_i = \boldsymbol{M}^i + \sum_{j=1}^{N} (1-\mu_{rj}) \int_{V_j} \frac{1}{4\pi} \nabla_y \left(\frac{(y-x_i).\boldsymbol{M}^j}{\|x_i - y\|^3} \right) dy = \underbrace{(\mu_{ri} - 1)\boldsymbol{H_S}(x_i)}_{\boldsymbol{b}_i :=}$$

- Dense nonsymmetric $3N \times 3N$ linear system (3 degrees of freedom / element)

$$\begin{pmatrix} [\boldsymbol{A}_{xx}] & [\boldsymbol{A}_{xy}] & [\boldsymbol{A}_{xz}] \\ [\boldsymbol{A}_{yx}] & [\boldsymbol{A}_{yy}] & [\boldsymbol{A}_{yz}] \\ [\boldsymbol{A}_{zx}] & [\boldsymbol{A}_{zy}] & [\boldsymbol{A}_{zz}] \end{pmatrix} \begin{pmatrix} [\boldsymbol{M}_x] \\ [\boldsymbol{M}_y] \\ [\boldsymbol{M}_z] \end{pmatrix} = \begin{pmatrix} [b_x] \\ [b_y] \\ [b_z] \end{pmatrix}$$

Goal and Contribution

- Acceleration of the matrix-vector product (from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$)
- Coupling Fast Multipole Method (FMM) + Magnetostatic Moment Method (MMM)
- FMM algorithm: generic and flexible (versatile)

Outline

Combining the interpolation-based FMM with the MMM

Adaptation of the Fast Multipole Method

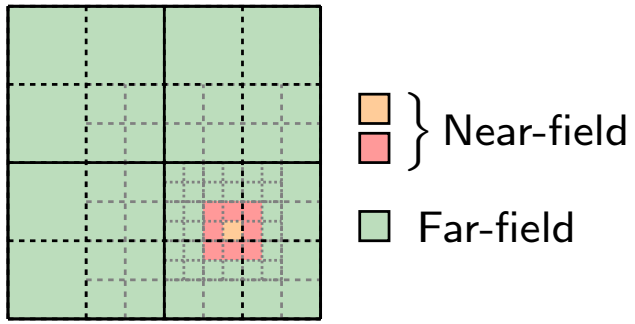
- **Matrix-vector product**

$$\forall i \in \{1,\dots,N\}, \qquad (\boldsymbol{AM})_i = \boldsymbol{M}^i + \sum_{j=1}^{N} (1-\mu_{r_j}) \int_{V_j} \underbrace{\frac{1}{4\pi} \left(\nabla_y \frac{(y-x_i)}{\|y-x_i\|^3} \right)}_{\text{magnetostatic kernel } k_0} dy\, \boldsymbol{M}^j$$

with $k_0(x,y) = \dfrac{1}{4\pi} \left(3 \dfrac{(x-y)\otimes(x-y)}{\|x-y\|^5} - \dfrac{1}{\|x-y\|^3} \boldsymbol{I_{3\times 3}} \right)$

$$\begin{cases} \text{singularity when } x \approx y \\ \text{smooth if } \|x-y\| > a \\ \text{decay } \sim \|x-y\|^{-3} \end{cases}$$

- **Fast Multipole Method (FMM)** [Greengard and Rokhlin 1987]
  - Spatial recursive subdivision into cells $\mathcal{C}$ (octree)
  - Separation of the near field from the far field
  - Hierarchical approximation of the far field



$$x_i \in \mathcal{C}_t, \qquad (A\boldsymbol{M})_i = \boldsymbol{M}^i + \sum_{\mathcal{C}_s} \sum_{V_j \in \mathcal{C}_s} (1-\mu_{r_i}) \int_{V_j} k_0(x_i,y) dy\, \boldsymbol{M}^j = (A_{near}\boldsymbol{M})_i + (A_{far}\boldsymbol{M})_i$$

Combining the interpolation-based FMM with the MMM

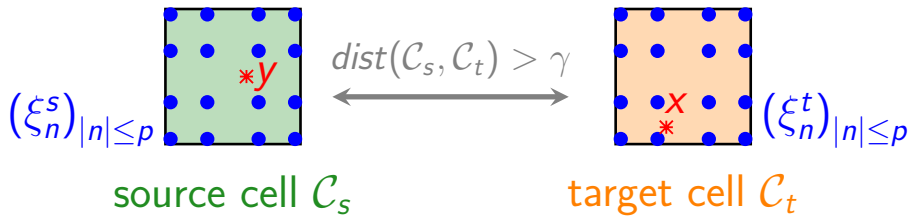
Far-field computation

$\mathcal{N}(\mathcal{C}_t)$: neighboring cells of $\mathcal{C}_t$

$$\forall i \in \{1,\dots,N\} \quad x_i \in \mathcal{C}_t, \qquad (A_{far}\boldsymbol{M})_i = \sum_{\mathcal{C}_s \notin \mathcal{N}(\mathcal{C}_t)} \sum_{V_j \in \mathcal{C}_s} (1-\mu_{r_j}) \int_{V_j} k_0(x_i,y)dy\, \boldsymbol{M}^j$$

- **Interpolation-based approximation** [Fong and Darve 2009]
  - 3D grid of $(p+1)^3$ points $(\xi_n^s)_{|n|\leq p}$ inside cell $\mathcal{C}_s$ (Uniform or Chebyshev points)
  - Lagrange polynomials S_n (s.t. $S_n(\xi_m^s) = \delta_{n,m}$)

$$k_0(x,y) \approx \sum_{|m|\leq p} \sum_{|n|\leq p} S_m(x)\, k_0\left(\xi_m^t, \xi_n^s\right) S_n(y)$$



- **Resulting far-field approximation**

$$x_i \in \mathcal{C}_t, \quad (A_{far}\boldsymbol{M})_i \approx \underbrace{\sum_{|m|\leq p} S_m(x_i) \underbrace{\sum_{\mathcal{C}_s \notin \mathcal{N}(\mathcal{C}_t)} \sum_{|n|\leq p} k_0(\xi_m^t, \xi_n^s) \underbrace{\sum_{V_j \in \mathcal{C}_s} (1-\mu_{r_j}) \int_{V_j} S_n(y)\, dy}_{P2M/M2M}\, \boldsymbol{M}^j}_{M2L}}_{L2P/L2L}$$

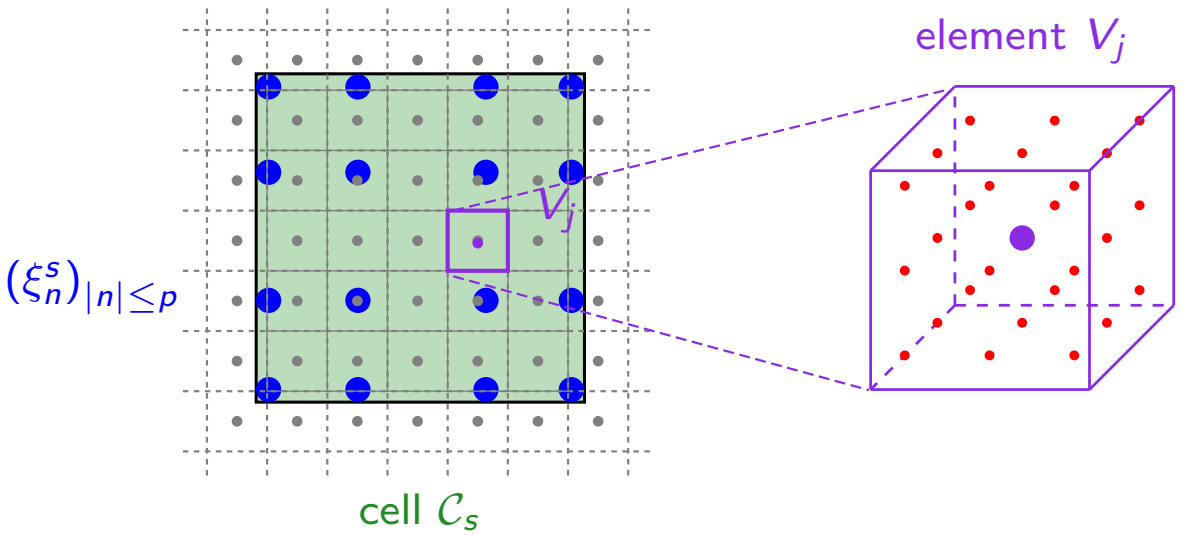
Combining the interpolation-based FMM with the MMM

Far-field computation

- **Focus on the P2M pass**

$$x_i \in \mathcal{C}_t, \quad (A_{far}\boldsymbol{M})_i \approx \sum_{|m|\leq p} S_m(x_i) \sum_{\mathcal{C}_s \notin \mathcal{N}(\mathcal{C}_t)} \sum_{|n|\leq p} k_0(\xi_m^t, \xi_n^s) \underbrace{\sum_{V_j \in \mathcal{C}_s} (1-\mu_{r_j}) \int_{V_j} S_n(y)\, dy\, \boldsymbol{M}^j}$$

$$\mathcal{M}_n(\mathcal{C}_s) = \sum_{V_j \in \mathcal{C}_s} (1-\mu_{r_j}) \int_{V_j} S_n(y)\, dy\, \boldsymbol{M}^j \qquad |n| \leq p \qquad \forall \text{ cell } \mathcal{C}_s$$



$$\int_{V_j} S_n(y)\, dy$$

(Gauss-Legendre quadrature rule)
"Exact computation"

Combining the interpolation-based FMM with the MMM

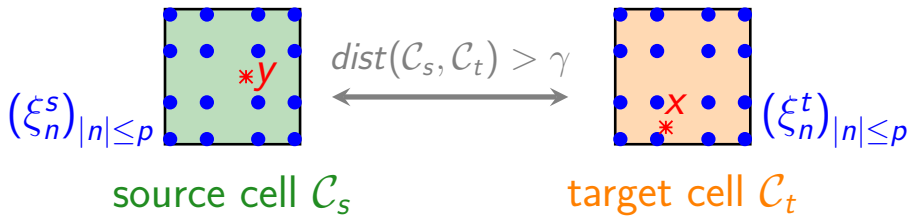
Far-field computation

- **New expression of the magnetostatic kernel**

$$k_0(x,y) = -\nabla_y k_1(x,y) = \nabla_x k_1(x,y) \qquad \text{with} \qquad k_1(x,y) = \frac{1}{4\pi}\frac{x-y}{\|x-y\|^3}$$

- **New kernel interpolation (2nd approach)**

$$k_0(x,y) \approx \sum_{|m|\leq p}\sum_{|n|\leq p} (\nabla_x S_m)(x)\, k_1\left(\xi_m^t, \xi_n^s\right) S_n(y)$$



- **New far-field approximation (2nd approach)**

$$x_i \in \mathcal{C}_t, \quad (A_{far}\boldsymbol{M})_i \approx \underbrace{\sum_{|m|\leq p} (\nabla_x S_m)(x_i) \underbrace{\sum_{\mathcal{C}_s \notin \mathcal{N}(\mathcal{C}_t)} \sum_{|n|\leq p} k_1(\xi_m^t, \xi_n^s) \underbrace{\sum_{V_j \in \mathcal{C}_s} (1-\mu_{r_j}) \int_{V_j} S_n(y)\, dy\, \boldsymbol{M}^j}_{P2M/M2M}}_{M2L}}_{L2P/L2L}$$

Combining the interpolation-based FMM with the MMM

Comparison of the two approaches

1st approximation approach

$$(A_{far}\boldsymbol{M})_i \approx \sum_{|m|\leq p} S_m(x_i) \sum_{\mathcal{C}_s \notin \mathcal{N}(\mathcal{C}_t)} \sum_{|n|\leq p} k_0(\xi_m^t, \xi_n^s) \sum_{V_j \in \mathcal{C}_s} (1-\mu_{r_j}) \int_{V_j} S_n(y)\, dy\, \boldsymbol{M}^j$$

2nd approximation approach

$$(A_{far}\boldsymbol{M})_i \approx \sum_{|m|\leq p} (\nabla_x S_m)(x_i) \sum_{\mathcal{C}_s \notin \mathcal{N}(\mathcal{C}_t)} \sum_{|n|\leq p} k_1(\xi_m^t, \xi_n^s) \sum_{V_j \in \mathcal{C}_s} (1-\mu_{r_j}) \int_{V_j} S_n(y)\, dy\, \boldsymbol{M}^j$$

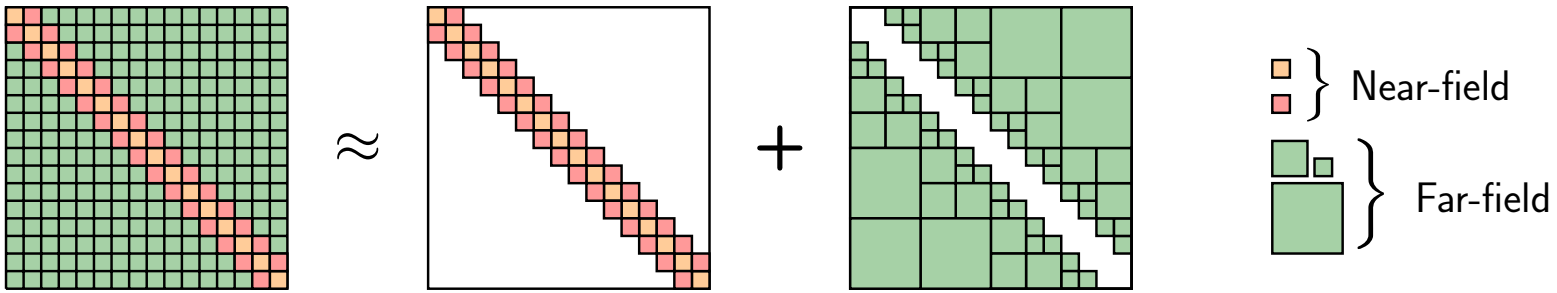
1st approach	**2nd approach**
$k_0(x,y) = \frac{1}{4\pi}\left(3\frac{(x-y)\otimes(x-y)}{\|x-y\|^5} - \frac{1}{\|x-y\|^3}\boldsymbol{I_{3\times 3}}\right)$	$k_1(x,y) = \frac{1}{4\pi}\frac{x-y}{\|x-y\|^3}$
$K = \left[k_0(\xi_m^t, \xi_n^s)\right]_{n,m} \in \mathbb{R}^{3p^3 \times 3p^3}$	$K = \left[k_1(\xi_m^t, \xi_n^s)\right]_{n,m} \in \mathbb{R}^{p^3 \times 3p^3}$
3 inputs, 3 outputs	3 inputs, 1 output
More accurate	Faster + less memory required (M2L pass)

Combining the interpolation-based FMM with the MMM

Summary

$$A = \begin{pmatrix} [\boldsymbol{A}_{xx}] & [\boldsymbol{A}_{xy}] & [\boldsymbol{A}_{xz}] \\ [\boldsymbol{A}_{yx}] & [\boldsymbol{A}_{yy}] & [\boldsymbol{A}_{yz}] \\ [\boldsymbol{A}_{zx}] & [\boldsymbol{A}_{zy}] & [\boldsymbol{A}_{zz}] \end{pmatrix} \in \mathbb{R}^{3N \times 3N} \qquad [\boldsymbol{A}_{xx}] = \begin{pmatrix} A_{xx}^{11} & \cdots & A_{xx}^{1N} \\ \vdots & & \vdots \\ A_{xx}^{N1} & \cdots & A_{xx}^{NN} \end{pmatrix} \in \mathbb{R}^{N \times N}$$

- **"Simplified" representation of the matrix partitioning**



Direct computation

$$\int_{V_j} k_0(x_i, y)\, dy$$

MMM [Oubaid 2014]

Approximation

$$\int_{V_j} k_0(x_i, y)\, dy \approx \sum_{|m| \leq p} S_m(x_i) \sum_{|n| \leq p} k_0\left(\xi_m^t, \xi_n^s\right) \int_{V_j} S_n(y)\, dy$$

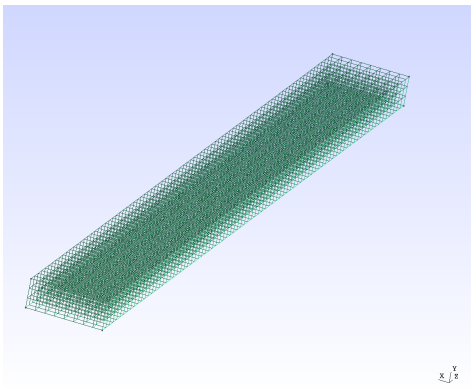
$$\int_{V_j} k_0(x_i, y)\, dy \approx \sum_{|m| \leq p} (\nabla_x S_m)(x_i) \sum_{|n| \leq p} k_1\left(\xi_m^t, \xi_n^s\right) \int_{V_j} S_n(y)\, dy$$

Some numerical results

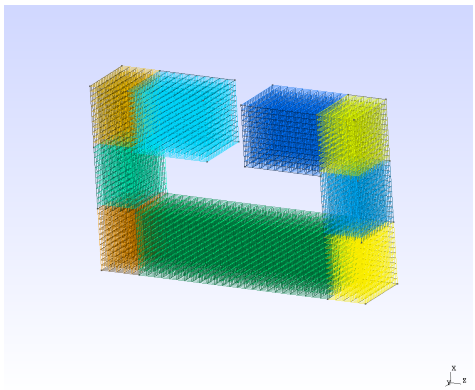
Configuration of the experiments

Experimental setup

- **Near-field computation**
  - *Fortran 90* implementation of the *MMM* (Laplace, *Toulouse*) [Oubaid 2014]
- **Far-field computation**
  - *ScalFMM* C++ library: multi-dim, kernel-independent, interpolation support
    - Acceleration: FFT (Uniform) [Blanchard 2017]
    - Compression: low-rank $K \sim UV^T$ (Chebyshev) [Messner et al. 2012]
- *Plafrim*: experimental platform



(a) Antenna geometry
$(1,000 < N < 15,000)$

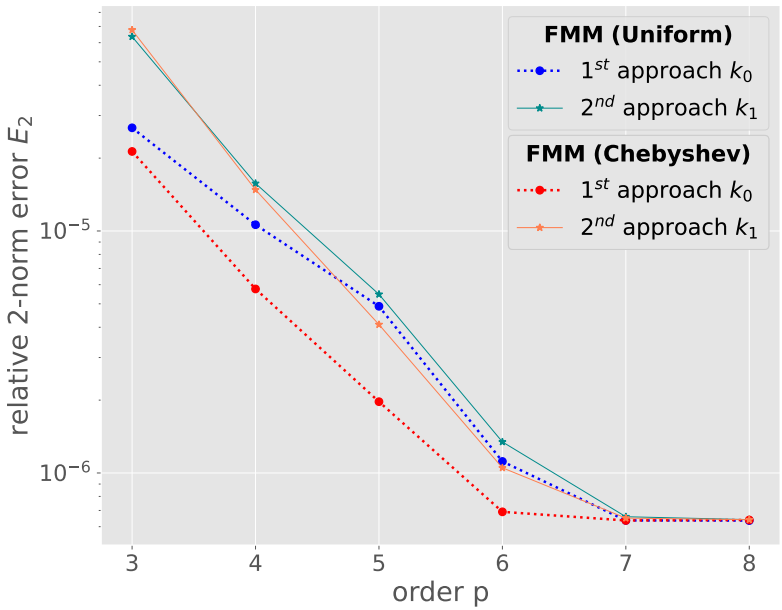


(b) U-shaped geometry
$(15,000 < N)$

Some numerical results

Accuracy: error vs. order p

(Antenna geometry)



Relative 2-norm error E_2 vs. order p

$\mu_r = 3000$, tree height $= 4$

Machine: plafrim/bora 2x 18-core Cascade Lake Intel Xeon Skylake Gold 6240 @ 2.6 GHz RAM 192 GB

Accuracy

- Relative 2-norm error E_2:

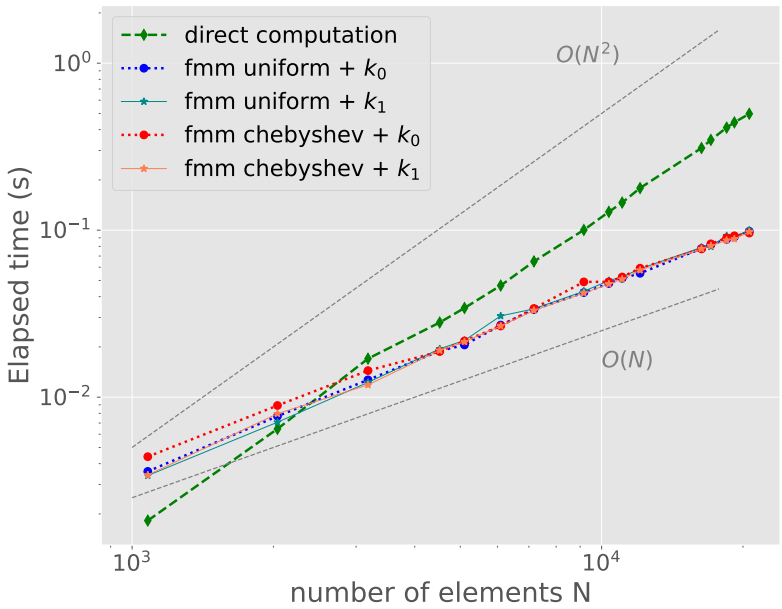
$$E_2 := \frac{\|\boldsymbol{A}^{FMM}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{x}\|_2}{\|\boldsymbol{A}\boldsymbol{x}\|_2}, \quad \boldsymbol{x} \in \mathbb{R}^{3N}$$

- k_0 more accurate than k_1
- Chebyshev interpolation more accurate than Uniform interpolation
- All errors stabilize around $10^{-6} \sim$ error of the near-field computation

Some numerical results

Performance: elapsed time vs. number of elements N (sequential)

(Antenna geometry)

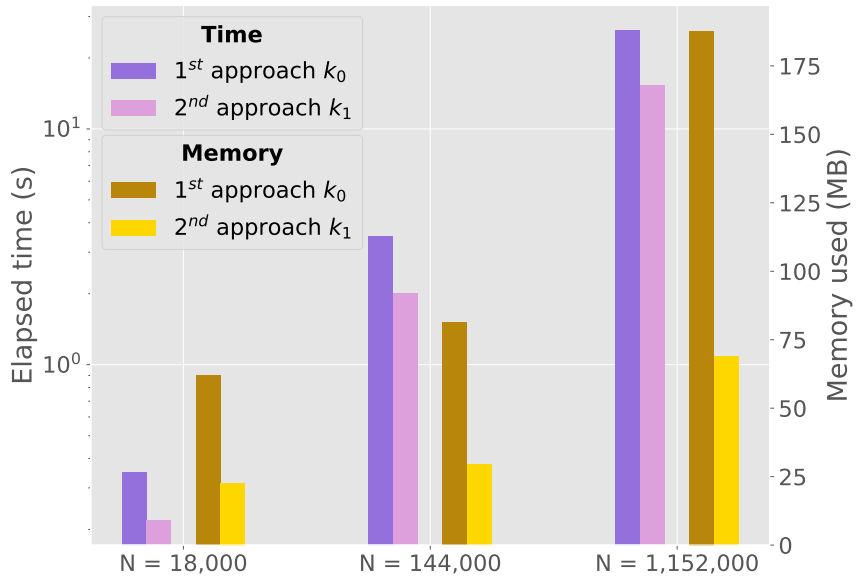


Elapsed time (s) vs. number of elements N

$\mu_r = 3000$, target accuracy $= 10^{-3}$

Machine: plafrim/bora 2x 18-core Cascade Lake Intel Xeon Skylake Gold 6240 @ 2.6 GHz RAM 192 GB

(U-shaped geometry)



Elapsed time (s) and memory consumption (MB) for the 2 approaches on larger meshes

Uniform interpolation (FFT), $\mu_r = 3000$, order $p = 5$,

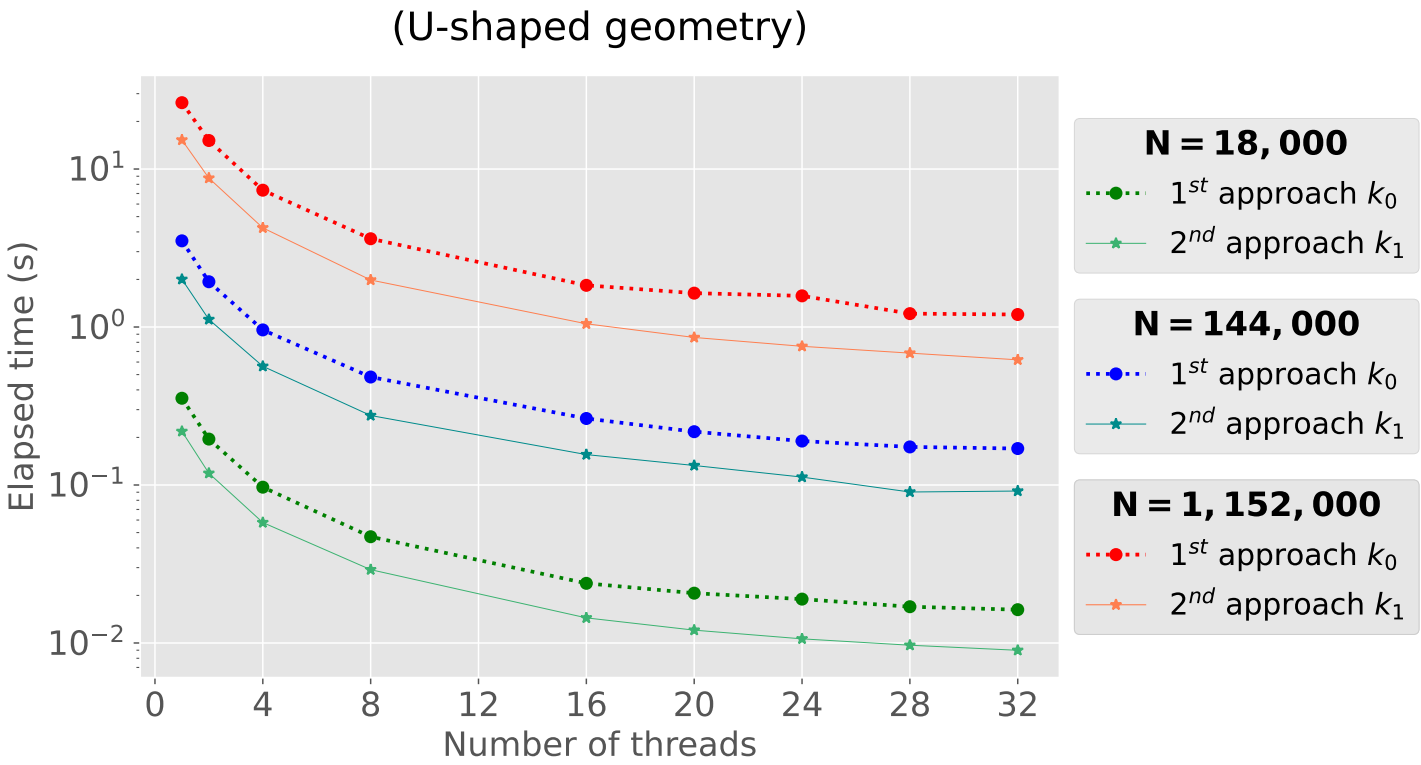
Memory consumption = multipoles and local expansions

Machine: plafrim/bora 2x 18-core Cascade Lake Intel Xeon Skylake Gold 6240 @ 2.6 GHz RAM 192 GB

Some numerical results

Performance: elapsed time vs. number of elements N (parallel)

- **Parallel implementation of the FMM**
  - Task-based approach [Agullo et al. 2014] [Agullo et al. 2016] (via OpenMP)



Elapsed time (s) vs. number of threads (OpenMP) for the parallel task-based FMM algorithm
Uniform interpolation (FFT), $\mu_r = 3000$, group size $= 5$ (granularity), order $p = 5$
Machine: plafrim/bora 2x 18-core Cascade Lake Intel Xeon Skylake Gold 6240 @ 2.6 GHz RAM 192 GB

Conclusion

- **Summary**
  - Development of an FMM-based matrix-vector product coupled with the MMM
  - Focus on the far-field computation using an interpolation approach
  - Emphasis on genericity and flexibility, allowing the algorithm to be adapted to two kernel functions (k_0 and k_1)
  - Implementation using the *ScalFMM* library (kernel-independent framework)
- **Perspective and possible future work**
  - Scalar potential formulation (less unknowns)
  - Comparison between the FMM and approaches based on $\mathcal{H}$-matrices
  - Coupling GMRES algorithm with the FMM-based matrix-vector product (requires a suitable preconditioning method)

References I


Agullo, Emmanuel, Berenger Bramas, Olivier Coulaud, Eric Darve, Matthias Messner, and Toru Takahashi (2016). "Task-based FMM for heterogeneous architectures". In: *Concurrency and Computation: Practice and Experience* 28.9, pp. 2608–2629.


Agullo, Emmanuel, Bérenger Bramas, Olivier Coulaud, Eric Darve, Matthias Messner, and Toru Takahashi (2014). "Task-based FMM for multicore architectures". In: *SIAM Journal on Scientific Computing* 36.1, pp. C66–C93.


Blanchard, Pierre (2017). "Fast hierarchical algorithms for the low-rank approximation of matrices, with applications to materials physics, geostatistics and data analysis". PhD thesis. Université de Bordeaux.


Chadebec, Olivier, J-L Coulomb, and Fleur Janet (2006). "A review of magnetostatic moment method". In: *IEEE Transactions on magnetics* 42.4, pp. 515–520.


Fong, William and Eric Darve (2009). "The black-box fast multipole method". In: *Journal of Computational Physics* 228.23, pp. 8712–8725.


Greengard, Leslie and Vladimir Rokhlin (1987). "A fast algorithm for particle simulations". In: *Journal of computational physics* 73.2, pp. 325–348.


Messner, Matthias, Bérenger Bramas, Olivier Coulaud, and Eric Darve (2012). "Optimized M2L kernels for the Chebyshev interpolation based fast multipole method". In: *arXiv preprint arXiv:1210.7292*.


Oubaid, Rania (2014). "Contribution à la modélisation du champ électromagnétique dans les dispositifs basses fréquences par la méthode des moments". PhD thesis.

Appendices

Near-field computation

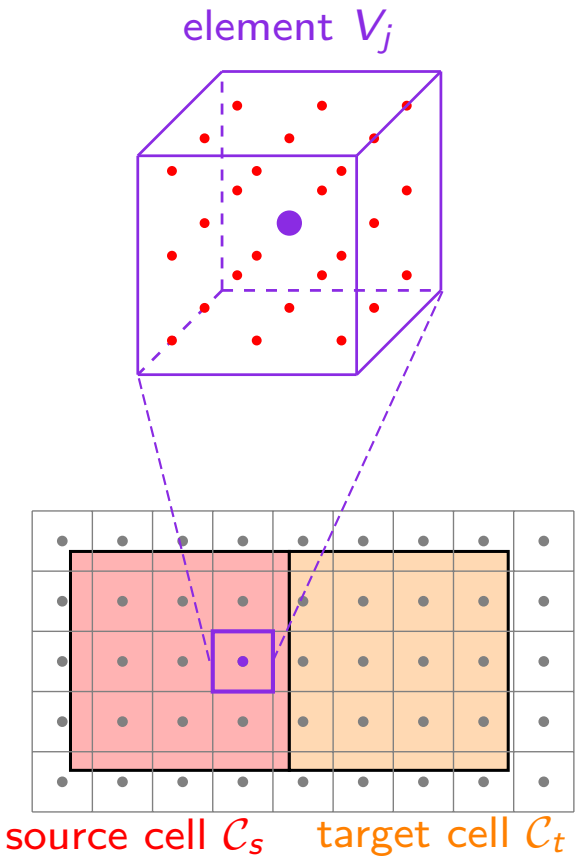
$$\forall i \in \{1, \dots, N\} \quad x_i \in \mathcal{C}_t, \qquad (A_{near}\boldsymbol{M})_i = \boldsymbol{M}^i + \sum_{\mathcal{C}_s \in \mathcal{N}(\mathcal{C}_t)} \sum_{V_j \in \mathcal{C}_s} (1 - \mu_{r\,j}) \int_{V_j} k_0(x_i, y) dy \boldsymbol{M}^j$$

- **Volume integral computation**
  ↪ Gauss-Legendre quadrature rule (n_g^3 points)

- **Singularity** [Chadebec et al. 2006]
  ↪ when V_i interacts with itself
  ↪ need a complex analytical expression

- **Adaptive strategy** [Oubaid 2014]
  ↪ choice of the order n_g
  ↪ targeted accuracy: $\sim 10^{-6}$

$$n_g \searrow \quad \text{as} \quad dist(V_i, V_j) \nearrow$$



Appendices

Cell width extension

$$\mathcal{M}_n(\mathcal{C}_s) = \sum_{V_j \in \mathcal{C}_s} (1 - \mu_{r\,j}) \int_{V_j} S_n(y)\, dy\, \boldsymbol{M}^j \qquad |n| \leq p \qquad \forall \text{ cell } \mathcal{C}_s$$

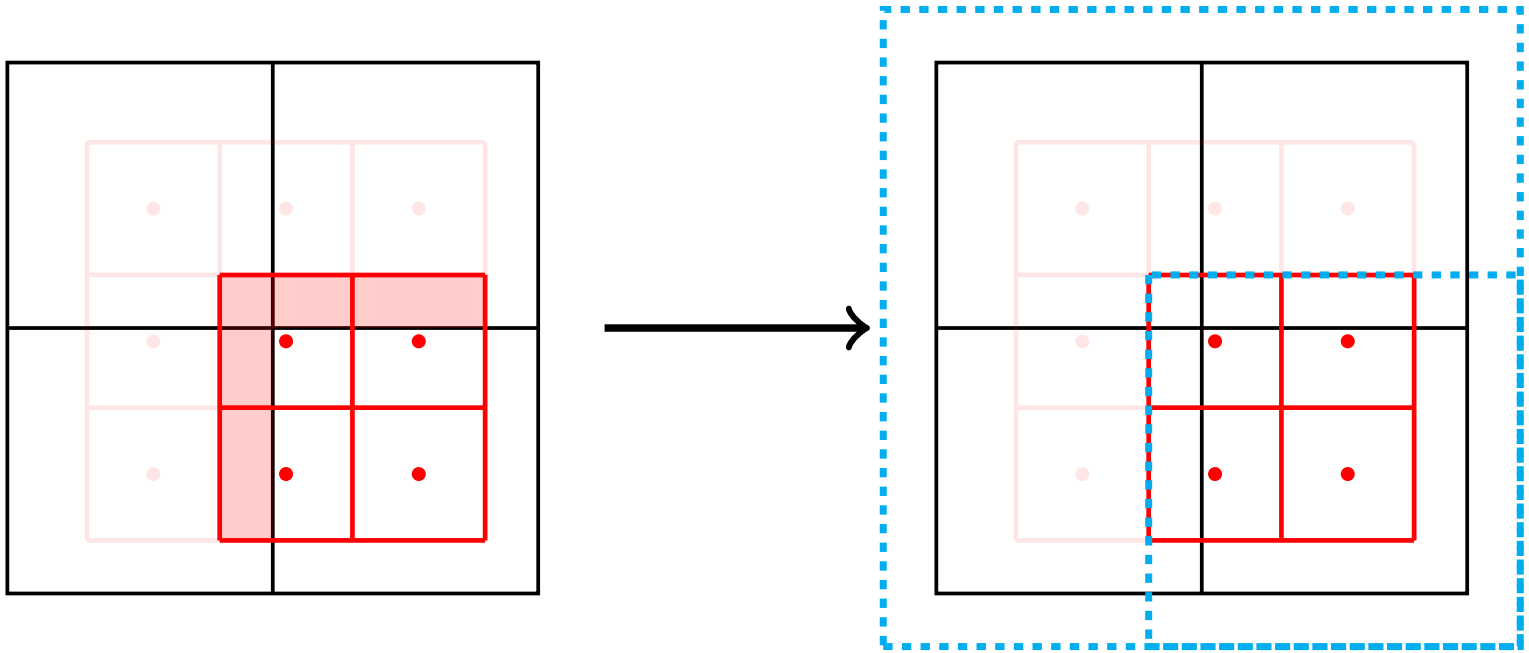


Illustration of the cell width extension principle [Blanchard 2017]