Fast hierarchical algorithms for generating Gaussian random fields

Pierre Blanchard, Olivier Coulaud, Eric Darve

Stanford University

Fast hierarchical algorithms for generating Gaussian random fields

Research Report - N°8811

November 13, 2015

Pierre Blanchard[1], Olivier Coulaud[1], Eric Darve[2]

[1]**HiePACS team, Inria Bordeaux Sud-Ouest**
[2] **Stanford Mechanical Engineering, Stanford University**

Contact:

Tel: +33637777417
Mail: pierre.blanchard@inria.fr
Address: 200, avenue de la Vieille Tour
33400 Talence

Abstract

Low-rank techniques for the approximation of matrices have become crucial tools in scientific computing in order to reduce the cost of storing matrices and compute usual matrix operations. Since standard techniques like the SVD or Cholesky do not scale well with the problem size N there has been a growing interest for alternative methods like randomized low-rank techniques. The main benefit of these methods lies in their simplicity, in fact only very basic operations like Matrix Vector Products (MVPs) or orthogonalizations are involved. As a result the cubic cost required to perform a matrix factorization with a standard technique is reduced to the quadratic cost required to apply a few MVPs, namely $\mathcal{O}(r \times N^2)$ where r is the numerical rank of the matrix. We present a new efficient algorithm for performing MVPs in $\mathcal{O}(N)$ operations called the *Uniform FMM* (*ufmm*) based on polynomial interpolations combined with a hierarchical (data sparse) representation of a kernel matrix. The algorithm is close to the *Black Box FMM* by Fong and Darve [1], however it uses an interpolation scheme based on an equispaced grid, which allows the use of *FFT* and consequently reduce both running time and memory footprint but has implications on accuracy and stability. In this work the *ufmm* is used to speed-up the MVPs involved in the *randomized SVD* and consequently reduce its cost to $\mathcal{O}(r^2 \times N)$. Numerical experiments were performed on this accelerated *randomized SVD* in order to evaluate the computational cost of building a low-rank square root of a covariance matrix. Such method is of particular interest in a wide range of scientific fields where spatially correlated multivariate Gaussian random variables have to be generated with very large sets of points. As expected and confirmed by our experiments the hierarchical nature of the algorithm leads to very competitive performances when the distribution of point is heterogeneous.

Keywords: $\mathcal{H}^2$-methods, *FMM*, *FFT*, *randomized SVD*, covariance kernel matrices, multivariate Gaussian random variables.

Contents

1 Introduction

1.1 Motivations and contributions

Recently, randomized algorithms for numerical linear algebra have gained in popularity due to their promising performances and accuracy but mostly due to their simplicity. Let us consider two of the most popular randomized algorithms for the low-rank approximation (LRA) of symmetric positive semi-definite (spsd) matrices:

- The randomized SVD (*RandSVD*) is a LRA algorithm based on *Gaussian random projection* that only involves Matrix-to-Vector Products (MVPs), matrix orthogonalizations and factorization of a small matrix. In the present paper we show how the randomized SVD can be accelerated using fast hierarchical MVPs.
- The Nyström method, a *random columns/rows sampling* algorithm, only involves rows and columns selection, some MVPs and the (pseudo-)inversion of a small matrix. Since only very basic matrix computations are involved an important place is left for optimization.

The main contribution of this paper is an *FFT*-powered $\mathcal{H}^2$-method for computing MVPs with kernel matrices. The algorithm is implemented within the ScalFMM package, an open-source generic parallel *FMM* library (available online at http://scalfmm-public.gforge.inria.fr/doc/), along with several other schemes including the original *FMM* and the *bbfmm* schemes. A complete documentation is provided online along with links to associated publications, see [2, 3, 4] for further information on the package.

This fast MVP method is then used to accelerate a *Gaussian random projection*-based LRA technique, namely the randomized SVD (*RandSVD*) described in Halko et al. [5]. The resulting algorithm, a hierarchical randomized SVD, allows for computing LRA for kernel matrices in a very efficient manner when the grid is highly heterogeneous. Due to its genericity such method applies to a very large number of scientific problems. The algorithm is implemented within the FMR package, an open-source library for computing randomized LRA (available online at https://gforge.inria.fr/projects/fmr), along with other square root algorithms and test cases inspired from various scientific applications. In the present paper focus is on the generation of low-rank square roots for covariance kernel matrices, that are in turn used to efficiently generate correlated random fields.

1.2 Overview

First of all, in section 2 we discuss works that relate to the present paper. Section 3 introduces the *uniform FMM* (or *ufmm*), a new interpolation based hierarchical method for computing MVPs powered by *FFT*. Section 4 recalls the fundamentals of randomized techniques for the LRA of matrices and provides details on their hierarchical variants as well as numerical benchmarks. Finally, realizations of Gaussian random fields are addressed in section 5.

2 Related works

***FMM* and $\mathcal{H}^2$-methods.** An *FFT* conversion of the original Fast Multipole Algorithm (FMA) was already proposed by Greengard et al. [6] and later by Elliott et al. [7]. Both methods consist in reformulating the expansion formulae in convolution form to allow their fast computation in Fourier domain using *FFT*.

In the present paper we use a polynomial interpolation based $\mathcal{H}^2$-method derived from the *black-box FMM* (*bbfmm*) by Fong et al. [1]. The method implies working with an equispaced interpolation grid which not only allows for significantly accelerating the original *bbfmm* scheme but also dramatically decreases the amount of data to be stored. The accuracy of the associated interpolation scheme, namely Lagrange interpolation, is slightly worse than the *bbfmm*, since the later is based on the near minimax Chebyshev interpolation. However in most cases of interest (including the one studied here) this loss of accuracy is offset by *FFT* performances.

Fast Randomized LRA. The acceleration of randomized LRA algorithms based on *random projection* usually relies on the ability to apply fast Matrix-to-Matrix Products (MMPs), hence most research works in this area focus on the structure and the sparsity of the matrices.

Woolfe et al. [8] and Liberty et al. [9] proposed accelerated variants of the randomized Interpolative Decomposition (ID) and SVD introduced by Martinsson et al. [10] based on the application of structured random matrices to the input matrix and thus reducing the cost of the Randomized ID or SVD from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. This method relies on the discrete Fourier transform but extends to other discrete operators such as the Walsh-Hadamard transform as mentionned by the authors.

To our knowledge, only very few papers actually address fast MVP methods for *Gaussian random projection*-based algorithm applied to dense structured matrices. For instance, in the context of low-rank approximations of matrices, Martinsson [11] describes a way to build HSS matrices using a *FMM*-powered *Gaussian random projection* algorithm (namely the *RandSVD*).

Generating correlated random fields. The generation of multivariate Gaussian random variables (or Gaussian random fields) is an important tool in many research fields such as cosmology [12, 13], geostatistics [14], hydrogeology [15, 16], brownian dynamics [17],... Approaches usually differ by the way the square root $\mathbf{A}$ of the prescribed covariance matrix is computed. The most popular ones are based on standard matrix decompositions, although they are known to be computationally prohibitive for large N. Alternative methods are often considered such as spectral methods [18], the turning bands method [19],... Please refer to [20] for further references. They usually provide approximate square roots but most importantly they do not always extend (well or at all) to 3D grids.

A popular approach for efficiently generating random fields on regular grids was introduced in 1993 by Dietrich and Newsam [20] and simulta-

neously developed by Wood and Chan [21]. It is often referred to as the *FFT* approach (or Circulant Embedding) and it provides an exact square root but presents some significant numerical limitations especially in 3D as explained in [20], not to mention that it only applies to regular grids.

Using the recent benefit of randomized techniques for fast matrix computations, Dehdari et al. [22] made use of the randomized SVD to efficiently approximate $\mathbf{A}$. This approach combines the robustness of the standard matrix factorizations with the efficiency of the randomized numerical linear algebra.

In the present paper we propose to use a randomized SVD accelerated by $\mathcal{H}^2$-MMPs to approximate $\mathbf{A}$ even more efficiently for covariance kernel matrices and highly heterogeneous grids.

3 An *FFT* accelerated $\mathcal{H}^2$-method: the *ufmm*

In this section we present a new variant of the *black-box FMM* [1] (or *bbfmm*) where the interpolation is done on a uniform (*i.e.*, equispaced) grid and the scheme is accelerated by *FFT*, we call this new method the *uniform FMM* or *ufmm*. We also propose adjustement to the original algorithm in order to increase its efficiency for globally smooth kernels. Finally the numerical performances and accuracy of the *ufmm* are compared to the *bbfmm* for various kernels.

3.1 A new interpolation based $\mathcal{H}^2$-method

Let us consider a system of N particles interacting with each other. We want to efficiently compute the contributions of all N *source* particles on each *target* particle $i = 1, \ldots, N$, *i.e.*, in less than $\mathcal{O}(N^2)$ operations. If we denote $\mathbf{x}_i$ the position of particle i, then its potential $f(\mathbf{x}_i) = f_i$ due to the densities $w(\mathbf{x}_j) = w_j$ can be expressed as follows

$$f(\mathbf{x}_i) = \sum_{j=1}^{N} k(\mathbf{x}_i, \mathbf{x}_j) w(\mathbf{x}_j) \tag{1}$$

where $k(\cdot, \cdot)$ represents the kernel of interactions.

Hierarchical clustering. The *bbfmm* is a $\mathcal{H}^2$-method for computing sums like (1) based on a hierarchical partitionning of the domain using a cluster tree structure, namely an octree. The root cluster $\mathcal{C}^{(0)}$ of the tree is the smallest cube enclosing all particles. At the level L of the octree we subdivide all *parent* cells $\mathcal{C}^{(L)}$ in 8 cubes of equal size to get all *child* cells $\mathcal{C}^{(L+1)}$ at level $L+1$. This recursive partition is stopped at the *leaf* level $\bar{L}$ once a suitable criterion is reached (*e.g.*, minimum or average number of particles per leaf cell, minimum leaf cell size).

Interaction list. The method makes use of an octree in order to identify admissible cluster pairs, *i.e.*, clusters of particles whose interactions can be approximated using a low-rank approach. More precisely, two cells of width ω distant from D form an admissible cell pair if they satisfy the admissibility criterion $D \geq \gamma\omega$, where γ can be prescribed ($\gamma = 2$ in the original *bbfmm*). Let $\mathcal{C}^{(L)}$

denote an arbitrary cell at level L. Cells that form admissible pairs with $\mathcal{C}^{(L)}$ are also said to be in farfield interactions with $\mathcal{C}^{(L)}$, while the other are in nearfield interactions with $\mathcal{C}^{(L)}$. The neighbor list $\mathcal{N}(\mathcal{C}^{(L)})$ is defined as the set of cells in nearfield interactions with $\mathcal{C}^{(L)}$. The interaction list $\mathcal{I}(\mathcal{C}_{\mathbf{x}}^{(L)})$ is defined as the set of non-empty cells $\mathcal{C}_{\mathbf{y}}^{(L)}$ in farfield interactions with $\mathcal{C}_{\mathbf{x}}^{(L)}$ such that the parent of $\mathcal{C}_{\mathbf{x}}^{(L)}$ and $\mathcal{C}_{\mathbf{y}}^{(L)}$ are in nearfield interactions. The latter condition ensures that all contributions are only computed once during the hierarchical summation scheme. Figure 1 illustrates how interaction and neighbor lists are built on the last two levels of a 1D tree structure.

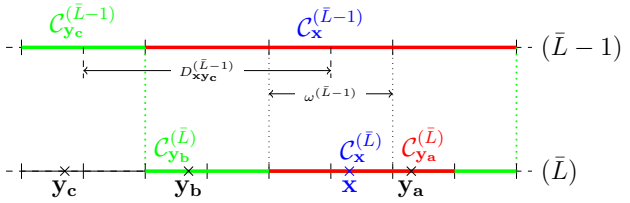

Figure 1: Portion of a 1D tree at level $\bar{L}$ and $\bar{L}-1$. The interaction lists $\mathcal{I}(\mathcal{C}_{\mathbf{x}}^{(L)})$ (green) and neighbor lists $\mathcal{N}(\mathcal{C}_{\mathbf{x}}^{(L)})$ (red) of a target particle $\mathbf{x}$ is represented for $\gamma = 2$. The contribution of the source particle $\mathbf{y_a}$ is computed analytically as part of the nearfied while contributions of $\mathbf{y_b}$ and $\mathbf{y_c}$ are approximated as part of the farfield at level $\bar{L}$ and $\bar{L}-1$ respectively.

Analytic expansion. The low-rank approximation technique involved in the *bbfmm* is based on analytic expansions of the kernel $k(\cdot,\cdot)$, namely Chebyshev polynomial interpolations. For any source $\mathbf{y}$ and target $\mathbf{x}$ lying in admissible clusters, the kernel $k(\mathbf{x},\mathbf{y})$ can be approximated using the following polynomial interpolation formula

$$k(\mathbf{x},\mathbf{y}) \approx \sum_{|\boldsymbol{\alpha}|\leq p} S_{\boldsymbol{\alpha}}^{p}(\mathbf{x}) \sum_{|\boldsymbol{\beta}|\leq p} K_{\boldsymbol{\alpha}\boldsymbol{\beta}} S_{\boldsymbol{\beta}}^{p}(\mathbf{y}) \qquad (2)$$

where $S_{\boldsymbol{\alpha}}^{p}$ is referred to as a 3D polynomial interpolator (of order p) and $K_{\boldsymbol{\alpha}\boldsymbol{\beta}}$ denotes the evaluation of $k(\cdot,\cdot)$ at interpolation points $\bar{\mathbf{x}}_{\boldsymbol{\alpha}}$ for targets and $\bar{\mathbf{y}}_{\boldsymbol{\beta}}$ for sources. The multi-index notations is defined as follows

$$\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$$

with

$$\alpha_i = 0, \ldots, p \text{ and } |\boldsymbol{\alpha}| = \max_{i\leq 3}(\alpha_i)$$

Error bounds for (2) are provided in [1]. The sum (1) can thus be approximated by

$$f(\mathbf{x}_i) \approx \sum_{|\boldsymbol{\alpha}|\leq p} S_{\boldsymbol{\alpha}}^{p}(\mathbf{x}_i) \sum_{|\boldsymbol{\beta}|\leq p} K_{\boldsymbol{\alpha}\boldsymbol{\beta}} \sum_{j=1}^{N} S_{\boldsymbol{\beta}}^{p}(\mathbf{x}_j) w(\mathbf{x}_j)$$

with $S_{\boldsymbol{\alpha}}^{p} = S_{\alpha_1}^{p} \times S_{\alpha_2}^{p} \times S_{\alpha_3}^{p}$ where S_n^p is a 1D polynomial interpolator.

Computational cost. Algorithm 1 presents the original *bbfmm* summation scheme as introduced by Fong and Darve [1]. The cost of both storing and applying the $(p+1)^3 \times (p+1)^3$ matrices $\bar{\mathbf{K}} = \{K_{\boldsymbol{\alpha}\boldsymbol{\beta}}\}_{|\alpha|,|\beta|\leq p}$ (*a.k.a.*, `M2L` operators) to the expansions is $\mathcal{O}(p^6)$. Moreover these matrices must be applied a potentially large number of times (max. 189 interactions per cell per level), which usually makes the `M2L` step the most computationally expensive step of any Fast Multipole scheme. In order to accelerate the *bbfmm* Messner et al. [23] described several optimizations based on the compression of the `M2L` operators and ex-

Algorithm 1: *black-box FMM (bbfmm)*

Input: Kernel $k(\cdot,\cdot)$, Densities $\mathbf{w}$, Positions $\mathbf{x}$, interpolation order p, leaf level $\bar{L}$

Output: Potentials $\mathbf{f}$

// Precomputation

for *level* $L = 2, \ldots, \bar{L}$ **do**

  // Assemble M2L operators given a fictitious target cell $\mathcal{C}_{\mathbf{x}}^{(L)}$

  for *source cell* $\mathcal{C}_{\mathbf{y}}^{(L)} \in \mathcal{I}(\mathcal{C}_{\mathbf{x}}^{(L)})$ **do**

    $K_{\boldsymbol{\alpha}\boldsymbol{\beta}} = k(\bar{\mathbf{x}}_{\boldsymbol{\alpha}}, \bar{\mathbf{y}}_{\boldsymbol{\beta}}),\ \forall \boldsymbol{\alpha}, \boldsymbol{\beta}/|\boldsymbol{\alpha}|, |\boldsymbol{\beta}| \leq p$

// Upward pass

for *level* $L = \bar{L}, \ldots, 2$ **do**

  for *source cell* $\mathcal{C}_{\mathbf{y}}^{(L)} \in$ *tree* **do**

    // P2M/M2M

    if $L = \bar{L}$ **then**

      // P2M: interpolation

      $$\mathcal{M}_{\boldsymbol{\beta}}^{(\bar{L})} = \sum_{j=1}^{N} S_{\boldsymbol{\beta}}^{p}(\mathbf{y}_j) w(\mathbf{y}_j),\ \forall \boldsymbol{\beta}/|\boldsymbol{\beta}| \leq p$$

    else

      // M2M:

      $$\mathcal{M}_{\boldsymbol{\beta}}^{(L)} = \sum_{|\boldsymbol{\beta}'| \leq p} S_{\boldsymbol{\beta}}^{p}(\bar{\mathbf{y}}_{\boldsymbol{\beta}'}) \mathcal{M}_{\boldsymbol{\beta}'}^{(L+1)},\ \forall \boldsymbol{\beta}/|\boldsymbol{\beta}| \leq p$$

// Downward pass

for *level* $L = 2, \ldots, \bar{L}$ **do**

  for *target cell* $\mathcal{C}_{\mathbf{x}}^{(L)} \in$ *tree* **do**

    // M2L: transfer between interacting cells

    for *source cell* $\mathcal{C}_{\mathbf{y}}^{(L)} \in \mathcal{I}(\mathcal{C}_{\mathbf{x}}^{(L)})$ **do**

      $$\mathcal{L}_{\boldsymbol{\alpha}}^{(L)} += \sum_{|\boldsymbol{\beta}| \leq p} K_{\boldsymbol{\alpha}\boldsymbol{\beta}} \mathcal{M}_{\boldsymbol{\beta}}^{(L)},\ \forall \boldsymbol{\alpha}/|\boldsymbol{\alpha}| \leq p$$

    // L2L/L2P/P2P

    if $L \leq \bar{L}$ **then**

      // L2L:

      $$\mathcal{L}_{\boldsymbol{\alpha}}^{(L)} = \sum_{|\boldsymbol{\alpha}'| \leq p} S_{\boldsymbol{\alpha}}^{p}(\bar{\mathbf{x}}_{\boldsymbol{\alpha}'}) \mathcal{L}_{\boldsymbol{\alpha}'}^{(L-1)},\ \forall \boldsymbol{\alpha}/|\boldsymbol{\alpha}| \leq p$$

    else

      // L2P: interpolation

      $$f(\mathbf{x}_i) = \sum_{|\boldsymbol{\alpha}| \leq p} S_{\boldsymbol{\alpha}}^{p}(\mathbf{x}_i) \mathcal{L}_{\boldsymbol{\alpha}}^{(\bar{L})}$$

      // P2P: direct computation

      for *source cell* $\mathcal{C}_{\mathbf{y}}^{(L)} \in \mathcal{N}(\mathcal{C}_{\mathbf{x}}^{(L)})$ **do**

        $$f(\mathbf{x}_i) = \sum_{j=1}^{N} k(\mathbf{x}_i, \mathbf{y}_j) w(\mathbf{y}_j)$$

Algorithm 2: *uniform FMM (ufmm)*

Input: Kernel $k(\cdot,\cdot)$, Densities $\mathbf{w}$, Positions $\mathbf{x}$, interpolation order p, leaf level $\bar{L}$

Output: Potentials $\mathbf{f}$

// Precomputation

for *level* $L = 2, \ldots, \bar{L}$ **do**

  // Assemble M2L operators (in Fourier domain)

  for *source cell* $\mathcal{C}_{\mathbf{y}}^{(L)} \in \mathcal{I}(\mathcal{C}_{\mathbf{x}}^{(L)})$ **do**

    // Compute first row of 3D block Toeplitz M2L operators

    $R_{\boldsymbol{\beta}} = k(\bar{\mathbf{x}}_{\mathbf{0}}, \bar{\mathbf{y}}_{\boldsymbol{\beta}}),\ \forall \boldsymbol{\beta}/|\boldsymbol{\beta}| \leq p$

    // Embed R in the first row $\tilde{R}$ of a block circulant matrix

    $\tilde{R}_{(\beta_0,\beta_1,:)} = (R_{(\beta_0,\beta_1,0:p)} | R_{(\beta_0,\beta_1,p-1:1)})$

    $\tilde{R}_{(\beta_0,:,\beta_2)} = (R_{(\beta_0,0:p,\beta_2)} | R_{(\beta_0,p-1:1,\beta_2)})$

    $\tilde{R}_{(:,\beta_1,\beta_2)} = (R_{(0:p,\beta_1,\beta_2)} | R_{(p-1:1,\beta_1,\beta_2)})$

    // Apply 3D DFT

    $\hat{\tilde{\mathbf{R}}} = \mathbf{F}\tilde{\mathbf{R}}$ with $F_{\boldsymbol{\alpha}\boldsymbol{\beta}} = e^{-\frac{2i\pi}{\tilde{p}}(\boldsymbol{\alpha}\cdot\boldsymbol{\beta})}$, $\forall(\boldsymbol{\alpha},\boldsymbol{\beta})/|\boldsymbol{\alpha}|, |\boldsymbol{\beta}| < \tilde{p}$ and $\tilde{p} = 2p$

// Upward pass

for *level* $L = \bar{L}, \ldots, 2$ **do**

  for *source cell* $\mathcal{C}_{\mathbf{x}}^{(L)} \in$ *tree* **do**

    // P2M/M2M

    // ... see Algo. 1

    // Pad expansion with zeros and transfer to Fourier domain.

    $\tilde{\boldsymbol{\mathcal{M}}}^{(L)} = \left(\boldsymbol{\mathcal{M}}^{(L)} | \mathbf{0}_{p-1}\right)$

    $\hat{\tilde{\boldsymbol{\mathcal{M}}}}^{(L)} = \mathbf{F}\tilde{\boldsymbol{\mathcal{M}}}^{(L)}$

// Downward pass

for *level* $L = 2, \ldots, \bar{L}$ **do**

  for *target cell* $C_x \in$ *tree* **do**

    // M2L: transfer between interacting cells

    for *source cell* $\mathcal{C}_{\mathbf{y}}^{(L)} \in \mathcal{I}(\mathcal{C}_{\mathbf{x}}^{(L)})$ **do**

      $\hat{\tilde{\mathcal{L}}}_{\boldsymbol{\alpha}}^{(L)} += \hat{\tilde{R}}_{\boldsymbol{\alpha}} \hat{\tilde{\mathcal{M}}}_{\boldsymbol{\alpha}}^{(L)},\ \forall \boldsymbol{\alpha}/|\boldsymbol{\alpha}| < \tilde{p}$

    // L2L/L2P/P2P

    // ... see Algo. 1

    // Transfer back to physical domain and unpad.

    $\tilde{\boldsymbol{\mathcal{L}}}^{(L)} = \mathbf{F}^{-1}\hat{\tilde{\boldsymbol{\mathcal{L}}}}^{(L)}$

    $\boldsymbol{\mathcal{L}}_{\boldsymbol{\alpha}}^{(L)} = \tilde{\boldsymbol{\mathcal{L}}}_{\boldsymbol{\alpha}}^{(L)},\ \forall \boldsymbol{\alpha}/|\boldsymbol{\alpha}| \leq p$

ploiting their symmetries.

3.2 Acceleration by *Fast Fourier Transform* (*FFT*)

In this paper, we present a new optimized hierarchical summation scheme based on a Lagrange polynomial interpolation in place of Chebyshev. This scheme is described in Algo. 2 and will now be referred to as the *uniform FMM* or *ufmm*. As described in the present section, Lagrange interpolation combined with the *Fast Fourier Transform* (*FFT*) allows for dramatically decreasing the memory footprint and the computational cost of the `M2L` operators, namely from $\mathcal{O}(p^6)$ to $\mathcal{O}(p^3 \log p)$. In 1D, the Lagrange interpolators S_n^p take the following form

$$S_n^p(x) = L_n^p(x) = \prod_{\substack{m=0 \\ m \neq n}}^{p} \frac{x - \bar{x}_m}{\bar{x}_n - \bar{x}_m}, \forall x \in [-1, 1]$$

for $n = 0, \ldots, p$ where $\bar{x}_m = -1 + 2m/p$ for $m = 0, \ldots, p$. In our implementations we actually used a slightly different form that allows for reducing the round-off errors.

Addressing Runge phenomenon. One should bear in mind that the Lagrange interpolation scheme becomes unstable for high orders of interpolation p (Runge phenomenon), however as described in a review by Boyd et al. [24] regularization methods have been introduced that are for instance based on optimization (Tikhonov or overdetermined-least squares), series expansions (Gegenbauer), subsampling (Mock-Chebyshev) or even multi-domain approaches. These methods defeat Runge phenomenon while preserving subgeometric convergence. In our approach divergence is not very likely to occur since we use interpolation on multiple subdomains and the value of p remains relatively low ($p < 20$) in almost any cases of interest. In particular, machine accuracy has been reached with our algorithm for a wide range of non-oscillatory kernels, *e.g.*, $1/r$, $1/r^2$, correlations (Matérn functions, spherical model) and various isotropic elastostatic Green's functions.

Structure of the `M2L` operators. The main interest of using Lagrange polynomials is that the interpolation is based on an equispaced (*uniform*) grid allowing for an easy conversion of the algorithm to Fourier domain and thus leading to a faster scheme. Indeed when $k(\cdot, \cdot)$ is evaluated on a uniform 1D grid then $\bar{\mathbf{K}}$ is a Toeplitz matrix, *i.e.*, each diagonal contains constant values. This is a consequence of the fact that the entries of $\bar{\mathbf{K}}$ only depend on the distance between the corresponding particles, *i.e.*, $k(x_i, x_j) = k(x_i - x_j)$. In the case of 2D grids the resulting matrix is block Toeplitz, *i.e.*, the matrix is composed of constant blocks over its diagonals while each block is itself Toeplitz. In 3D we include an extra level of blocking meaning that the constant diagonal blocks are now block Toeplitz.

Circulant embedding. Dietrich [20] and Wood [21] almost simultaneously proposed a scheme in which a Toeplitz matrices is embedded in a larger circulant matrix and then applied as a convolution in Fourier space. We briefly recall this method in the case of a 1-dimensional Toeplitz `M2L` operator $\bar{\mathbf{K}}$ of order p. For the sake of clarity let us consider the case where $\bar{\mathbf{K}}$ is sym-

metric (the non symmetric case is very similar), then $\bar{\mathbf{K}}$ is characterized by its first row

$$\bar{\mathbf{K}}_{0:} = (\rho_0, \dots, \rho_p)^t \in \mathbb{R}^{p+1}$$

with

$$\rho_i = k(\bar{x}_0, \bar{x}_i),\ i = 0, \dots, p$$

This row can be embedded into the first row of a (symmetric) circulant matrix $\bar{\mathbf{E}} \in \mathbb{R}^{\tilde{p}\times\tilde{p}}$ with $\tilde{p} = p + 1 + p - 1 = 2p$ such that

$$\begin{aligned}\bar{\mathbf{E}}_{0:} &= (\rho_0, \dots, \rho_p, \rho_{p-1}, \dots, \rho_1)^t \\ &= \left(\bar{\mathbf{K}}_{:0}, \rho_{p-1}, \dots, \rho_1\right)^t \in \mathbb{R}^{2p}\end{aligned}$$

Consequently $\bar{\mathbf{K}}$ is embedded in the upper left corner of $\bar{\mathbf{E}}$, *i.e.*, the application of both matrices is equivalent if the last $p-1$ columns and rows of $\bar{\mathbf{E}}$ are masked. In the non-symmetric case the embedding is slightly larger, namely $\tilde{p} = p + 1 + p = 2p + 1$.

Conversion to Fourier domain. The discrete convolution theorem implies that the set of eigenvectors of any circulant matrix $\bar{\mathbf{E}}$ forms the Discrete Fourier Transform (DFT) operator, *i.e.*, $\mathbf{F} = \left\{(2p)^{-1/2} e^{-2i\pi mn/2p}\right\}_{m,n=0,\dots,2p}$, while the eigenvalues $\mathbf{\Lambda}$ are known to be the DFT of its first column, *i.e.*, $\mathbf{\Lambda} = \mathbf{F}\bar{\mathbf{E}}_{:0}$. Let us consider a multipole expansion $\mathbf{M} \in \mathbb{R}^{p+1}$ and the resulting local expansion $\mathbf{L} = \bar{\mathbf{K}}\mathbf{M} \in \mathbb{R}^{p+1}$. If we pad $\mathbf{M}$ with $p-1$ zeros and then apply $\bar{\mathbf{E}}$ to it we get $\mathbf{L}$ after truncation of the last $p-1$ values. If $\bar{\mathbf{E}}$ is replaced by its singular value decomposition $\mathbf{F^*diag(\Lambda)F}$ this implies

$$\begin{aligned}(\mathbf{L}|_{\mathbf{p-1}}) &= \bar{\mathbf{E}}\,(\mathbf{M}|\mathbf{0_{p-1}}) \\ &= \mathbf{F^*diag(\Lambda)F}\,(\mathbf{M}|\mathbf{0_{p-1}}) \\ &= \mathbf{F^*}\left[\left(\mathbf{F}\bar{\mathbf{E}}_{:0}\right) \odot \left(\mathbf{F}\,(\mathbf{M}|\mathbf{0_{p-1}})\right)\right]\end{aligned}$$

where $(\mathbf{U}|\mathbf{V})$ and $\mathbf{U} \odot \mathbf{V}$ respectively denote the concatenation and the entrywise product of the arbitrary vectors $\mathbf{U}$ and $\mathbf{V}$. Hence, the matrix vector product $\mathbf{L} = \bar{\mathbf{K}}\mathbf{M}$ can be performed in the form of an entrywise product in Fourier space (up to a circulant embedding). Transfers back and forth between Fourier and physical domains are done by *FFT*, which results in an asymptotic overall cost of $\mathcal{O}((2p)^d \log(2p))$ where d denotes the ambiant dimension.

Numerical complexity. The theoretical complexities of the *bbfmm* and *ufmm* algorithms are given in Table 1. They are not only given in term of computational cost but also in term of memory footprint for the `M2L` (always precomputed) and the `P2P` (precomputed if matrix to matrix operations are considered). In the *ufmm* the precomputation of the `M2L` operators requires $\mathcal{O}(p^3 \log p)$ operations (*i.e.*, the cost of a *FFT*) while their application requires only $\mathcal{O}(p^3)$ operations (*i.e.*, the cost of an entrywise product). All algorithms scale linearly in N but they optimize the interpolation steps differently in particular during the critically expensive step of the method, namely the `M2L` step. In fact, the cost of storing and applying `M2L` operators scales like $\mathcal{O}(p^3)$ in the *ufmm* and $\mathcal{O}(p^6)$ in the *bbfmm*, therefore we expect significant differences in memory requirements and computational times.

	bbfmm	*ufmm*	*smooth-ufmm*
`P2P` (CPU/memory)	$n_0 \times N$	$n_0 \times N$	0
`P2M/L2P` (CPU)	$p^3 \times N$	$p^3 \times N + p^3 \log(p) \times N/n_0$	$p^3 \times N + p^3 \log(p) \times N/n_0$
`M2M/L2L` (CPU)	p^4	$p^4 + p^3 \log(p)$	$p^4 + p^3 \log(p)$
build `M2L` (CPU)	p^6	$p^3 \log(p)$	$p^3 \log(p)$
apply `M2L` (CPU)	p^6	p^3	p^3
`M2L` (memory)	p^6	p^3	p^3

Table 1: Asymptotic complexities of the algorithms. The complexities of the `M2L` and `M2M/L2L` are given per level and per non empty cell, they should be multiplied by the number of non-empty cells ($\mathcal{O}(2^{3L})$) and then summed over levels. For the `P2P` and `P2M/L2P` the complexities are given globally. The constant $n_0 = N/2^{3\bar{L}}$ denotes the average number of particles per leaf.

3.3 A block low-rank algorithm for smooth kernels: *smooth-ufmm*

The *Fast Multipole Method* (*FMM*) introduced by Greengard et al. [25] was originally developed for kernels with a strong singularity at the origin, *e.g.*, $k(x, y) = 1/|x - y|$. This property leads to the design of a multilevel scheme that allowed for approximating a larger part of the field without ever reaching the singularity.

Correlation kernels. In the present paper we investigate low-rank representations of covariance matrices characterized by correlation kernels k such as

$$\text{Exponential:} \quad k_{1/2}(\mathbf{r}) = e^{-|\mathbf{r}|_\ell} = e^{-r/\ell}$$
$$\text{Gaussian:} \quad k_{\infty}(\mathbf{r}) = e^{-|\mathbf{r}|_\ell^2/2} = e^{-r^2/2\ell^2}$$

namely extreme cases of Matérn functions k_ν, where the length scale ℓ characterizes the decreasing speed of the correlation. Most correlation kernels do not present any strong singularity (*e.g.*, Matérn family) and the Gaussian k_∞ is even perfectly smooth (the larger ν the smoother k_ν).

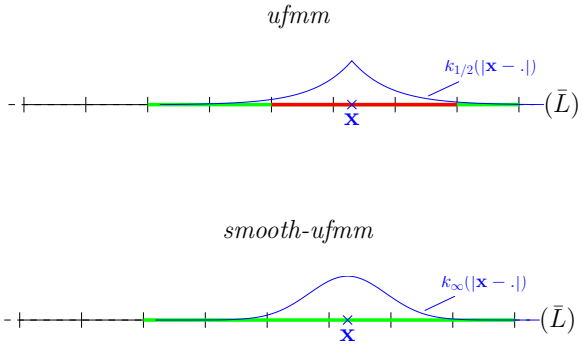

Figure 2: Schematic view of both *ufmm* variants on a 1D tree. The target particle $\mathbf{x}$ (blue) lies in the leaf cell $\mathcal{C}_{\mathbf{x}}^{\bar{L}}$. The interaction list $\mathcal{I}(\mathcal{C}_{\mathbf{x}}^{L})$ at a given level L is represented in green, while the nearfield $\mathcal{N}(\mathcal{C}_{\mathbf{x}}^{L})$ is represented in red.

smooth-ufmm. Provided the kernel is *sufficiently* smooth near the origin we want to consider a summation scheme where the nearfield interactions are approximated as well and thus no direct interaction is ever computed. An algorithm of this nature can be derived from the *ufmm* algorithm by changing the admissibility criterion $\gamma^{(\bar{L})}$ at the leaf level from 2 to 0. While in the *ufmm* well-separation of leaves ($\gamma^{(\bar{L})} = 2$) is imposed, in the new variant coinciding leaves form admissible

pairs as well ($\gamma^{(\bar{L})} = 0$). In fact, since the kernel is *sufficiently* smooth at the origin then the diagonal subblocks of the kernel matrix associated with coincident and adjacent cell pairs are *relatively* low-rank. The interaction list of a given leaf now includes the direct neighbors (and the leaf itself), which means that all interactions will be transferred by means of `M2L` operations and thus no interaction needs to be computed at the `P2P` step. This variant is denoted *smooth-ufmm* and can be seen as a block low-rank approximation technique. Figure 2 illustrates the difference between both variants in terms of interaction lists. In the *smooth* variant the interaction list at the leaf level $\mathcal{I}(\mathcal{C}_x^{\bar{L}})$ includes all leaf cells whose parent is a direct neighbor of the target parent cell $\mathcal{C}_x^{\bar{L}-1}$ and consequently $\mathcal{N}(\mathcal{C}_x^{\bar{L}}) = \varnothing$, whereas in the standard variant the direct neighbors of the target leaf cell form the nearfield.

Optimal setup. Since the `P2P` and the `M2L` do not compete anymore in the *smooth-ufmm*, these steps do not have to be balanced. Consequently, the concept of level is not relevant anymore and the actual tuning parameter becomes the width of the leaf cells. More precisely, having bypassed the `P2P` step, we only need to minimize the cost of the `M2L` step by using the fewest number of clusters, *i.e.*, the largest leaf cells. However, as our algorithm remains a multi-level scheme, the depth for the octree is still used to control the width of the leaves. The optimal setup for the algorithm is the lowest tree depth leading to a similar accuracy as the original *ufmm* scheme.

Computational cost. Due to the extension of the interaction list, the maximum number of `M2L` operators to store and apply at the leaf level increases from 189 to $189 + 27 = 216$ in 3D. However, as shown in S Section 3.4, for a given accuracy the number of level is usually lower than the level required for the standard *ufmm*. Moreover, since no direct computation is involved, the computational time of the *smooth-ufmm* is expected to be lower than for the *ufmm*. Since these algorithms are used here for computing MMPs then it is crucial to precompute and store the `P2P` operators, hence the *smooth-ufmm* obviously requires a significantly lower amount of memory than the *ufmm* (see memory requirements in Table 1).

3.4 Numerical benchmarks

We first compare the relative accuracy and numerical performances of the *bbfmm* and the *ufmm* on MVPs for $k(x, y) = 1/|x - y|$. Then we compare *ufmm* with the *smooth-ufmm* on MMPs for a Gaussian correlation kernel $k = k_\infty$ with $\ell_G = 0.5$. Various geometries are considered (a cube and the unit sphere) with a common root bounding box of width $w^0 = 2$. All experiments are conducted using double precision arithmetic (ScalFMM also implements single precision).

***bbfmm* vs *ufmm*.** The comparative cost of the *bbfmm*, with or without compression of the `M2L` operators (see [23]), and the *ufmm* is represented Fig. 3. The *ufmm* outperforms the unoptimized *bbfmm* in terms of both computational time and memory requirements. Let us recall that the *symmetric bbfmm* is taylored for symmetric kernels allowing for a massive reduction of the interaction list and that it involves individual compression of the `M2L` operators, while the *compressed*

bbfmm only involves a global compression of the `M2L` operators. As expected the *ufmm* is slightly less accurate than the *bbfmm* (due to the near minimax property of the Chebyshev interpolation) but still performs better for a given precision. As shown on the graphs the performances of the *ufmm* nearly approaches those of the *symmetric bbfmm* though it is applicable to any kernel. More precisely the *ufmm* is faster in terms of computational times but requires a little more memory than the *symmetric bbfmm*.

***ufmm* vs *smooth-ufmm*.** In the graphs Fig. 4 we compare the relative performances of the *ufmm* and the *smooth-ufmm* for a given accuracy of about 10^{-3}. The particles are either distributed on the surface of the unit sphere (*i.e.*, many cells of the octree are empty) or inside a cube (*i.e.*, octree is densely populated). The *smooth-ufmm* unsurprisingly has better performances than the *ufmm* since the cost of approximating the nearfield for a fixed tree depth *sooner or later* becomes cheaper than computing it in a direct fashion for an increasing tree depth.

***global fft*.** The *smooth-ufmm* with $h = 1$ boils down to a *FFT*-accelerated Lagrange interpolation on the bounding box. It is refered to as the *global fft* as it generalizes the idea of *FFT* on an arbitrary grid. Therefore, for a given bounding box, the performances of the *global fft* are independent of the shape of the distribution. As shown on Fig. 4, even in this range of accuracy, the global approach is almost always slower than the hierarchical variants that furthermore benefit from the heterogeneity of the distribution. Figures shown in Appendix A confirm these observations for a fixed N and a varying accuracy.

4 Fast randomized low-rank approximation (LRA) of matrices

This paper proposes various techniques for obtaining low-rank representation of matrices at a reasonable cost. Section 3 introduces a new variant of the *bbfmm*, which is a well known method for computing MVPs based on *analytic* LRA of kernel matrices. On the other hand, the numerical benchmarks considered in section 5 require efficient methods for building and applying $\mathbf{C}^{1/2}$ that are commonly based on *algebraic* LRA of the covariance matrix $\mathbf{C}$, namely standard matrix decomposition techniques. However such methods are usually not tractable to large N, namely up to several millions of particles. In this section we recall the basics of the randomized LRA and we introduce hierarchical variants of the algorithms provided by Halko et al. [5]. Then, we discuss their optimal tuning as well as their numerical performances compared to standard methods.

4.1 Fundamentals of the randomized LRA and their hierarchical variant

Matrix decomposition techniques such as Cholesky or SVD are often considered as the standard *algebraic* techniques, however they are usually untractable to large systems since they scale like $\mathcal{O}(N^3)$. On the other hand *random projection*-based LRA provide simple, easy to

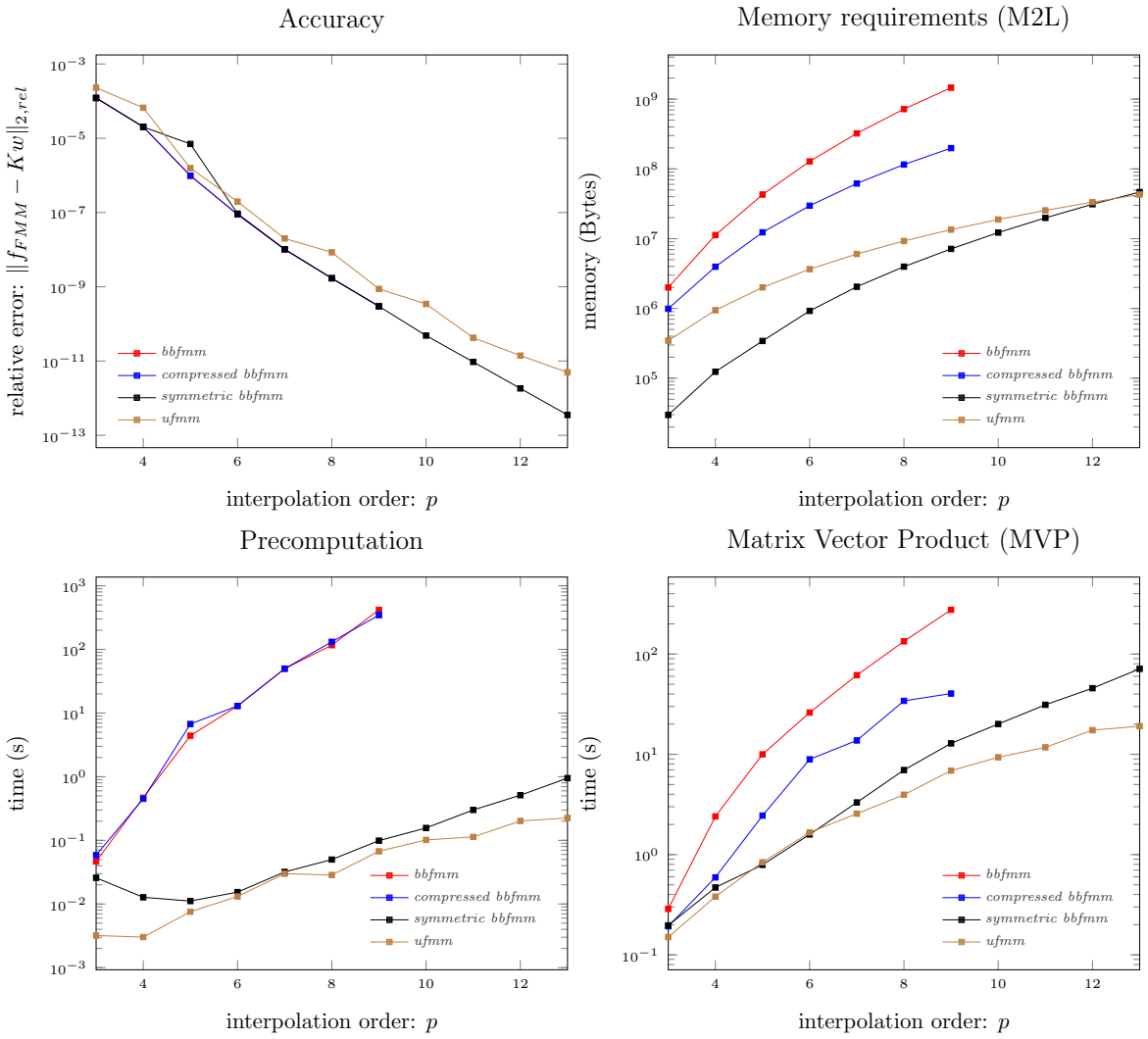

Figure 3: Accuracy and performances of the *ufmm* and variants of the *bbfmm* with respect to the interpolation order p. We used $k(x, y) = 1/|x - y|$ and 20.000 particles randomly distributed in the $2 \times 2 \times 2$ cube. Observations: For a given accuracy, the *ufmm* performs better than generic variants of *bbfmm* in terms of both computational time and memory footprint. Its performances are close to the *bbfmm* optimized for symmetric kernels.
Machine: desktop computer - Intel Core i7-3520M CPU @ 2.90GHz x 4 with 8GB ram.

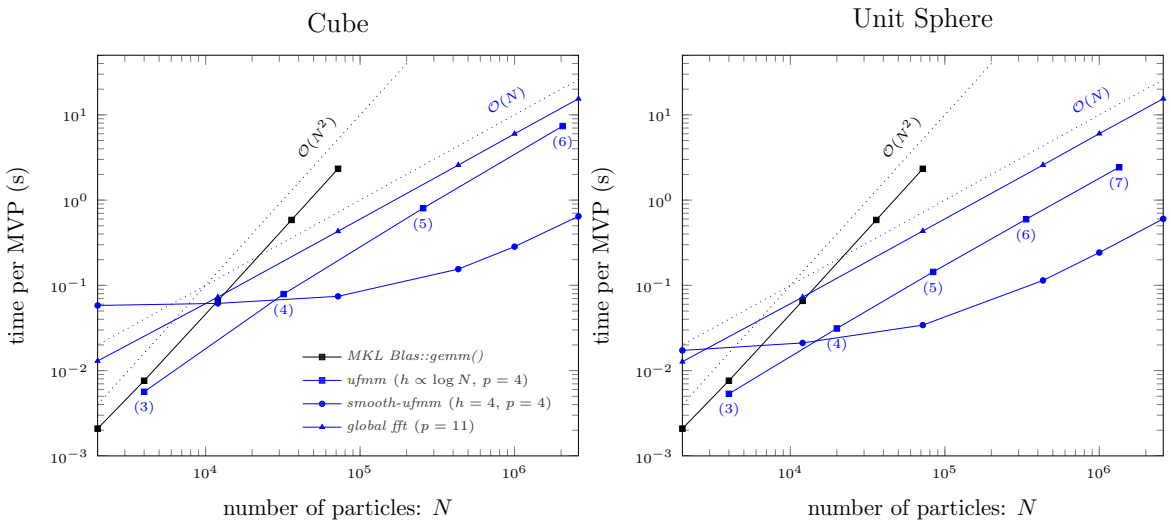

Figure 4: Time per MVP for a total of 10 MVPs using either *ufmm* (with pre-assembled `P2P` operators) or *smooth-ufmm*. We choose p such that the relative L2 error is below 10^{-3}. Particles are either randomly distributed in the $2 \times 2 \times 2$ cube (left) or on the unit sphere (right). The tree depth h of the *ufmm* (written below blue circles) ensures a relatively constant average number of particles per leaf: $n_0 = 74$ (left) and $n_0 = 62$ (right). The correlation is Gaussian with $\ell_G = 0.5$.
Machine: plafrim/riri - Deca-core Intel Xeon E7-4870 @ 2.40GHz with 1TB ram and 30MB L3 Cache.

implement and efficient alternatives to these methods. Along with *random column selection*-based techniques like the Nyström method [26] (see Zhang et al. [27] for Nyström-based LRA) they have recently drawn much attention in scientific fields such as geostatistics [22], machine learning [28], data assimilation [29],... These methods were introduced and further enhanced in a series of papers [10, 9, 8] and a complete review can be found in [5].

Basics. A *random projection*-based LRA is a 2-stage process (see Algorithm 3 for the randomized SVD): during the first stage an approximation $\mathbf{Q} \in \mathbb{R}^{N \times r}$ of the range of the input matrix $\mathbf{C} \in \mathbb{R}^{N \times N}$ is performed by applying $\mathbf{C}$ to a set of r random vectors. In the second stage a LRA of $\mathbf{C}$ is achieved in the form $\mathbf{C_r} = \mathbf{Q}\mathbf{Q}^\mathbf{T}\mathbf{C}$ (or the symmetric variant $\mathbf{C_r} = \mathbf{Q}\mathbf{Q}^\mathbf{T}\mathbf{C}\mathbf{Q}\mathbf{Q}^\mathbf{T}$) with well-controlled error bounds, for instance in Frobenius norm

$$\mathbb{E}(\|\mathbf{C} - \mathbf{C_r}\|_F) \leq f_F(r, s, q) \times \|\mathbf{C} - \mathbf{C_r}\|_F^{opt.} \quad (3)$$

where

$$\|\mathbf{C} - \mathbf{C_r}\|_F^{opt.} = \left(\sum_{i=r+1}^{N} \sigma_i^2(\mathbf{C}) \right)^{1/2} \quad (4)$$

denotes the deterministic lower bound (*a.k.a.*, baseline). The oversampling parameter s and the number of power iterations q are tune-up parameters, a topic that will be discussed later on. Depending on the nature of the correlation kernel k we will consider 2 subroutines for approximating the range of $\mathbf{C}$. The most convenient one is the *Adaptive Randomized Range Finder* (*ARRF*, Algorithm 4.1 in [5]) since it returns a *nearly* optimal range approximation for a prescribed accuracy. Depending on the context one might also like to use a fixed rank variant such as the *Ran-*

domized Subspace Iterations (*RSI*, Algorithm 4.4 in [5]).

Hierarchical variant. Originally, both algorithms have similar computational costs and scale like $\mathcal{O}(r \times N^2)$ since the dominant cost arise from the MMPs performed at stage I and II. One main contribution of the present paper is to propose a hierarchical variant of the algorithm in [5] in order to reduce the costs of these products to $\mathcal{O}(r \times N)$ and thus the overall complexity to $\mathcal{O}(r^2 \times N)$. Another important benefit of using hierarchical matrix vector products ($\mathcal{H}^2$-MV) is that $\mathbf{C}$ is never explicitely built. However the major drawback is the introduction of an extra error in the matrix-matrix products, for this reason the order of interpolation needs to be carefully tuned. Although an $\mathcal{H}^2$-MV powered randomized LRA was already proposed in Li et al. [29], here we propose a new highly optimized $\mathcal{H}^2$-method that is in essence very close to a FMM as it is based on analytical expansions. On the other hand March et al. [30] used randomized techniques for the optimization of the *Kernel Independent FMM*.

4.2 Numerical analysis and tune up of the *Hierarchical Randomized SVD*

In the present paper we are interested in providing low-rank square root of usual covariance matrices (see Section 5) with a compression rate r/N below 10% and within an accuracy of 10^{-3}–10^{-4}. All experiments are conducted using double precision arithmetic (FMR also implements single precision).

Standard *RandSVD*. Figure 6 and 7 present the accuracy of the standard *RandSVD* for varying compression rates (between 1 and 10%). The optimal error in Frobenius norm (4) is represented in blue and corresponds to the lower error bound, *i.e.*, the baseline described in [5]. Figure 6 shows that oversampling alone is enough to reach optimal accuracy for a Gaussian covariance. Besides the target compression rate and accuracy are both satisfied down to a length scale of $\ell_G = 0.5$ (included). Figure 7 shows that the effect of power iterations is strong but on the other hand in the case of an exponential covariance the target accuracy and compression rate are not satisified simultaneously even if we consider the optimal case with the largest length scale $\ell_E = 1$.

Tune-up. A small oversampling ($s \approx 5$) will always be considered because it provides the baseline for the error estimators, *i.e.*, $\sigma_{r+1}(\mathbf{C})$ in spectral norm and $\left(\sum_{i=r+1}^{N} \sigma_i^2(\mathbf{C})\right)^{1/2}$ in Frobenius norm. If necessary s can be set to a larger value ($s \leq r$). For covariances with slow decreasing singular values like the exponential a few power iterations ($q \leq 3$) are highly recommended.

Hierarchical variant. The optimal order of the interpolation is determined so that it does not significantly affect the output rank of the *ARRF* or the accuracy of the *RSI*. For a Gaussian correlation with $\ell = 0.5$ and particles distributed on the unitSphere the covariance matrix can be represented by a matrix of rank $r \approx 70$ with a precision of about $\varepsilon = 10^{-2}$ in Frobenius norm. In this case experiments showed that the smallest interpolation order that does not affect the randomized algorithm is $p = 4$. If the desired ac-

Algorithm 3: Hierarchical Randomized SVD

Input: Correlation kernel $k(.,.)$, grid size N, Positions $\mathbf{x}$, rank r or accuracy ε, number of power iterations q, oversampling parameter s, interpolation order p, octree depth h

Output: $[\mathbf{U}, \mathbf{\Sigma}]$ approximate SVD of $\mathbf{C} = \{k(x_i, x_j)\}_{i,j<N}$

// Stage I: Approximate the range of $\mathbf{C}$

if *accuracy ε is prescribed* **then**

  $[\mathbf{Q}, r] = ARRF(k, x, N, p, h, q, s, \varepsilon)$ using $\mathcal{H}^2$-MVPs. $\mathcal{O}(r^2 \times N)$

if *rank r is prescribed* **then**

  $[\mathbf{Q}, \varepsilon] = RRF(k, x, N, p, h, q, s, r)$ using $\mathcal{H}^2$-MVPs. (idem)

// Stage II: Decompose $\mathbf{C_r} = \mathbf{Q}(\mathbf{Q^T C Q})\mathbf{Q^T} \approx \mathbf{C}$ as $\mathbf{U\Sigma U^T}$

Build $\mathbf{B} = \mathbf{Q^T C Q} \in \mathbb{R}^{r \times r}$ using $\mathcal{H}^2$-MVPs. $\mathcal{O}(r^2 \times N)$

Perform SVD of $\mathbf{B} = \mathbf{U_B \Sigma U_B^T}$ $\mathcal{O}(r^3)$

Form $\mathbf{U} = \mathbf{Q U_B}$ $\mathcal{O}(N \times r^2)$

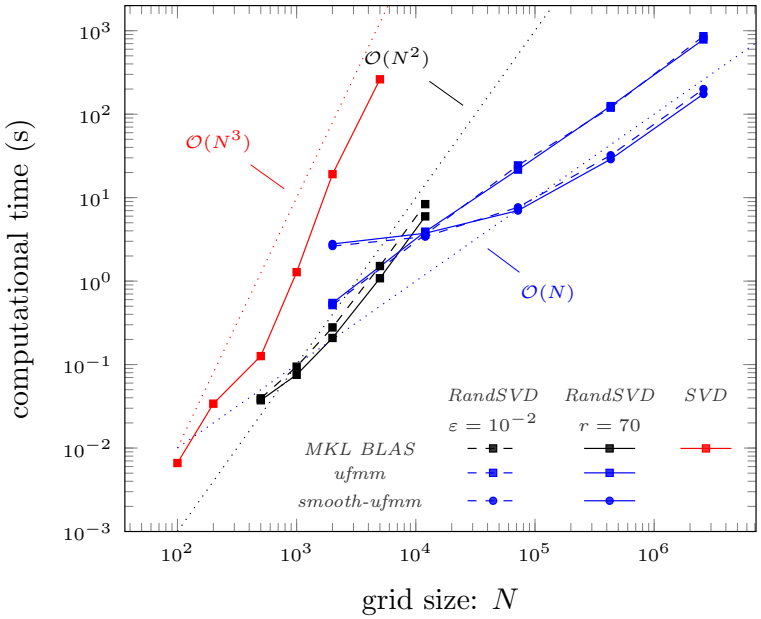

Figure 5: Time for computing a randomized SVD using either the fixed rank (RRF) or fixed accuracy ($ARRF$) algorithm with $q = 0$ and $s = 10$. MMPs are computed either in a dense way or by means of the *ufmm* or *smooth-ufmm* with $p = 4$, particles are randomly distributed on the unit sphere and the correlation is Gaussian with $\ell = 0.5$. Observations: As expected, both fixed rank and fixed accuracy variants exhibit similar performances. On the other hand, the graphs confirm the theoretical asymptotic costs (linear in blue, quadratic in black and cubic in red).
Machine: plafrim/riri - Deca-core Intel Xeon E7-4870 @ 2.40GHz with 1TB ram and 30MB L3 Cache.

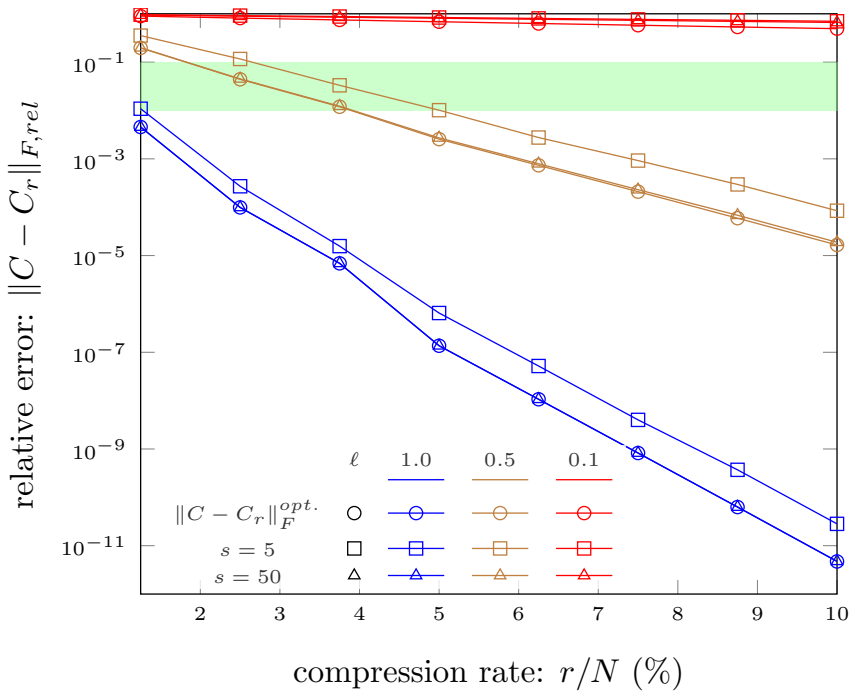

Figure 6: Accuracy of the *RandSVD w.r.t.* the compression rate for a Gaussian correlation. We analyze the effects of the oversampling s alone ($q = 0$). Observations: The Gaussian kernel is smooth, therefore its spectrum decreases fast and $\mathbf{C}$ is relatively low-rank. Consequently, the *RandSVD* performs well even without power iterations and with low oversampling ($s = 5$). An oversampling of $s = 50$ leads to a near optimal error. However, $\mathbf{C}$ becomes high-rank for small ℓ, *e.g.*, $\ell = 0.1$.

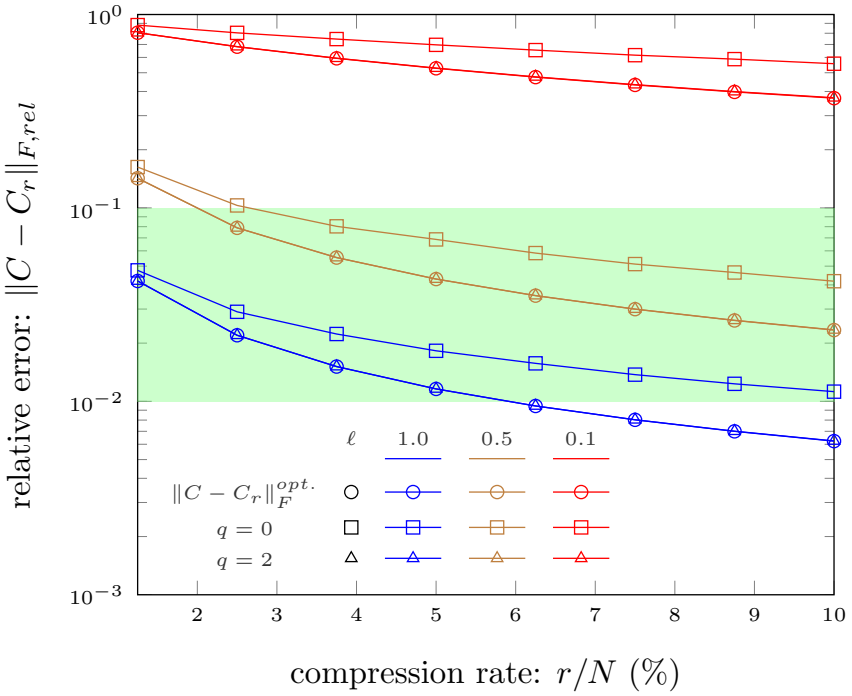

Figure 7: Accuracy of the *RandSVD w.r.t.* the compression rate for an exponential correlation. We analyze the effects of q power iterations using a fixed oversampling of $s = 50$. Observations: The exponential correlation is not smooth, therefore its spectrum is relatively flat. A few power iterations lead to a near optimal error but the rank remains relatively high. $\mathbf{C}$ becomes high-rank for small ℓ, *e.g.*, $\ell = 0.1$.

curacy equals $\varepsilon = 10^{-3}$ then $r \approx 100$ and $p = 5$ is optimal.

5 Simulation of Gaussian random fields

As described in Section 1, it is crucial in numerous scientific applications, especially for geostatistics, to efficiently generate large ensembles of vectors $\mathbf{Y} \in \mathbb{R}^N$ of correlated Gaussian random variables on a spatial grid $\{x_i\}_{i=1\dots N}$ with a given correlation function $k(r_{ij})$ where $r_{ij} = |x_i - x_j|$. In Section 4 we described a matrix-free algorithm to efficiently perform an approximate matrix decomposition (namely a *SVD*) in $\mathcal{O}(N)$ operations. Here we make use of this method to efficiently generate correlated random fields.

5.1 Benefiting from randomization

A standard approach. Let $\mathbf{C}$ denote the prescribed covariance matrix, *i.e.*, $\mathbf{C} = \{k(r_{ij})\}_{i,j=1\dots N}$. Realizations of correlated Gaussian random fields $\mathbf{Y} \sim \mu(\mathbf{0}, \mathbf{C})$ are usually obtained by application of the square root $\mathbf{A}$ of $\mathbf{C}$ to a white noise $\mathbf{X} \sim \mu(\mathbf{0}, \mathbf{I_N})$, *i.e.*, a Gaussian random field $\mathbf{X}$ verifying $\mathbb{E}(\mathbf{X}) = \mathbf{0}$ and $\mathbb{E}(\mathbf{X}\mathbf{X}^t) = \mathbf{I_N}$. It is easy to verify that the resulting field $\mathbf{Y} = \mathbf{A}\mathbf{X}$ has covariance $\mathbf{C}$, indeed

$$\mathbb{E}(\mathbf{Y}\mathbf{Y}^t) = \mathbb{E}(\mathbf{A}\mathbf{X}\mathbf{X}^t\mathbf{A}^t) = \mathbf{A}\mathbb{E}(\mathbf{X}\mathbf{X}^t)\mathbf{A}^t = \mathbf{C}$$

As mentioned several times earlier, computing the square root $\mathbf{A}$ using standard methods can be extremely expensive. Existing alternatives often lack robustness and some even present important numerical limitations.

A fast randomized approach. Random projection based LRAs usually provide powerful and robust alternatives to standard matrix factorizations. Not only do they perform independently of the grid shape but they outperform standard methods for matrices of relatively low-rank. A randomized SVD powered by dense MVPs was already considered in [22] for the efficient generation of Gaussian random fields. Here, we want to benefit from hierarchical algorithms to further improve the performances of the randomized approach and especially when the grid is highly heterogenous. Results are shown in the next subsection where the RandSVD powered by hierarchical MVPs is applied to a Gaussian covariance on arbitrary grids.

5.2 Numerical benchmarks

Random fields were simulated using approximate square roots of various covariance matrices for particles distributed on the unit sphere. The approximation is done by mean of a fixed precision ($\varepsilon = 10^{-2}$) RandSVD with *ufmm*-accelerated MVPs. Realizations are displayed on Figure 8 for various length scales.

Accuracy. The sample covariance matrix $\tilde{\mathbf{C}}$, computed from the realizations as

$$\tilde{\mathbf{C}} = \frac{1}{n_{real}} \sum_{i=1}^{n_{real}} (\mathbf{Y}_i - \mathbb{E}(\mathbf{Y}_i))(\mathbf{Y}_i - \mathbb{E}(\mathbf{Y}_i))^t \quad (5)$$

, provides a good approximation of the experimental covariance. Therefore we analyze the accuracy of the method using the error between $\tilde{\mathbf{C}}$

and the actual covariance matrix $\mathbf{C}$. In order to limit the computational cost required by verifications, the error is computed on a subset of the matrices. The accuracy of the method *w.r.t.* the number of realizations n_{real} is presented in Tables 2 and 3. They show that *ufmm*- and *smooth-ufmm*-accelerated RandSVD provide square roots that generate correlated random fields with similar accuracies.

Performances. As shown by figure 4, for a Gaussian correlation with $\ell = 0.5$, building an approximate square root with $N = 72k$, $r = 70$ takes about 20 seconds with the *ufmm*, 8 seconds with the *smooth-ufmm* while it would take about 5 minutes with dense MMPs. For $\ell = 0.25$, the rank r and the computational times are about 3 times larger.

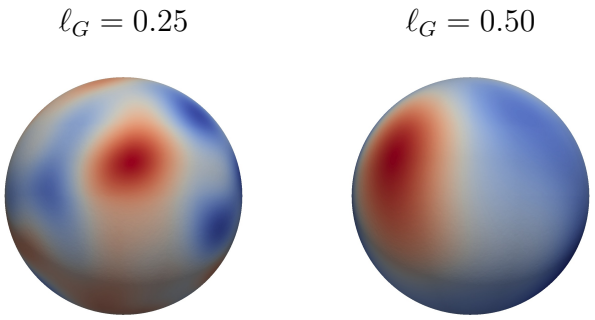

Figure 8: Realization of a Gaussian random field on $72k$ points distributed on the unit sphere using a Gaussian correlation. The approximate square root was obtained by mean of a *smooth-ufmm*-accelerated RandSVD ($\varepsilon = 10^{-2}$).

6 Conclusion

We presented a new optimized $\mathcal{H}^2$-method for computing fast MMPs, namely a *FFT* accelerated variant of the *bbfmm*. The method is very efficient compared to other optimized *bbfmm* in both computational time and memory footprint. The algorithm has been optimized for smooth matrix kernels allowing for further improvement of the performances. This hierarchical MMP method has been implemented within a randomized SVD resulting in a significant acceleration of the algorithm for low-rank covariance kernel matrices. Finally, we used the hierarchical randomized SVD to compute approximate low-rank square root of covariance matrices and generate correlated Gaussian random fields at a reasonable cost compared to standard methods. The resulting approach is more robust than most existing alternatives but obviously requires the matrices to be relatively low-rank. The benefits of using a hierarchical randomized SVD to approximate square root of covariance matrices is illustrated on a very basic but widely used and generic application. More evolved applications are currently addressed such as Sigma-Point Kalman Filters (SPKF) for data assimilation in climatology or Multi-Dimensional Scaling (MDS) for the classification of amazonian tree species.

Aknowledgments

Experiments presented in this paper were carried out using the PLAFRIM experimental testbed, being developed under the Inria PlaFRIM development action with support from LABRI and IMB and other entities: Conseil Régional d'Aquitaine, FeDER, Université de Bordeaux and CNRS (see https://plafrim.bordeaux.inria.fr/).

ufmm

n_{real}	$\|\tilde{C} - C\|_{2,rel}$	$\|\tilde{C} - C\|_{\infty,rel}$
$1 \cdot 10^3$	$2.48 \cdot 10^{-1}$	$1.47 \cdot 10^{-1}$
$1 \cdot 10^4$	$7.88 \cdot 10^{-2}$	$4.53 \cdot 10^{-2}$
$1 \cdot 10^5$	$2.63 \cdot 10^{-2}$	$1.57 \cdot 10^{-2}$
$1 \cdot 10^6$	$1.02 \cdot 10^{-2}$	$8.53 \cdot 10^{-3}$

smooth-ufmm

n_{real}	$\|\tilde{C} - C\|_{2,rel}$	$\|\tilde{C} - C\|_{\infty,rel}$
$1 \cdot 10^3$	$2.48 \cdot 10^{-1}$	$1.57 \cdot 10^{-1}$
$1 \cdot 10^4$	$7.88 \cdot 10^{-2}$	$5.68 \cdot 10^{-2}$
$1 \cdot 10^5$	$2.67 \cdot 10^{-2}$	$1.84 \cdot 10^{-2}$
$1 \cdot 10^6$	$1.09 \cdot 10^{-2}$	$1.09 \cdot 10^{-2}$

Table 2: Error on the covariance matrix $\mathbf{C}$ *w.r.t.* the number of realizations n_{real} for a Gaussian correlation and $\ell = 0.25$. The sample covariance $\tilde{\mathbf{C}}$, *i.e.*, the experimental covariance, is computed from the n_{real} realizations as Eq. 5.

ufmm

n_{real}	$\|\tilde{C} - C\|_{2,rel}$	$\|\tilde{C} - C\|_{\infty,rel}$
$1 \cdot 10^3$	$1.31 \cdot 10^{-1}$	$1.12 \cdot 10^{-1}$
$1 \cdot 10^4$	$3.75 \cdot 10^{-2}$	$4.56 \cdot 10^{-2}$
$1 \cdot 10^5$	$1.41 \cdot 10^{-2}$	$1.76 \cdot 10^{-2}$
$1 \cdot 10^6$	$4.72 \cdot 10^{-3}$	$5.95 \cdot 10^{-3}$

smooth-ufmm

n_{real}	$\|\tilde{C} - C\|_{2,rel}$	$\|\tilde{C} - C\|_{\infty,rel}$
$1 \cdot 10^3$	$1.32 \cdot 10^{-1}$	$1.21 \cdot 10^{-1}$
$1 \cdot 10^4$	$3.72 \cdot 10^{-2}$	$3.95 \cdot 10^{-2}$
$1 \cdot 10^5$	$1.39 \cdot 10^{-2}$	$1.35 \cdot 10^{-2}$
$1 \cdot 10^6$	$4.45 \cdot 10^{-3}$	$5.69 \cdot 10^{-3}$

Table 3: Error on the covariance matrix $\mathbf{C}$ *w.r.t.* the number of realizations n_{real} for a Gaussian correlation and $\ell = 0.5$. The sample covariance $\tilde{\mathbf{C}}$, *i.e.*, the experimental covariance, is computed from the n_{real} realizations as Eq. 5

Appendices

A Convergence of the hierachical methods *w.r.t.* the point distribution

Here we present comparative results on the convergence of the *ufmm* and *smooth-ufmm* with respect to the interpolation order p for various geometries (unit sphere, cube, prolate sphere and hyperbolic paraboloid). In particular we analyze the influence of the length scale on the MVP error for the Gaussian correlation kernel. For all geometries the width of the bounding box is equal to 2. All computations where performed on a cluster computer, namely plafrim/mirabelle: Hexacore Westmere Intel Xeon X5670 @ 2.93GHz with 96GB ram and 12MB L3 Cache.

Observations. The *global fft* will always have the same cost at a given interpolation order, since this method is oblivious of the shape of the distribution. On the other hand, the cost of the hierarchical methods may vary significantly from one distribution to another. Let us for instance consider a Gaussian correlation with $\ell = 0.5$. If the particles are distributed in the cube (*i.e.*, an homogenous distribution) the cost of the *global fft* lies somewhere between the *ufmm* and the *smooth-ufmm* with optimal h, see Figure 9. If the particles are distributed on a sphere (*i.e.*, an heterogenous distribution) then the hierachical methods become faster than the *global fft*, see Figure 10.

Other distributions Fig. 11 and Fig. 12 respectively confirm the previous observations on the prolate ellipsoid and the hyperbolic paraboloid, *i.e.*, highly heterogeneous distributions. The associated octrees have larger depths ($h = 7$) than for the unit sphere ($h = 5$), however in the lowest level a large number of cells are empty. Consequently, the computational times are only slightly larger than for the unit sphere.

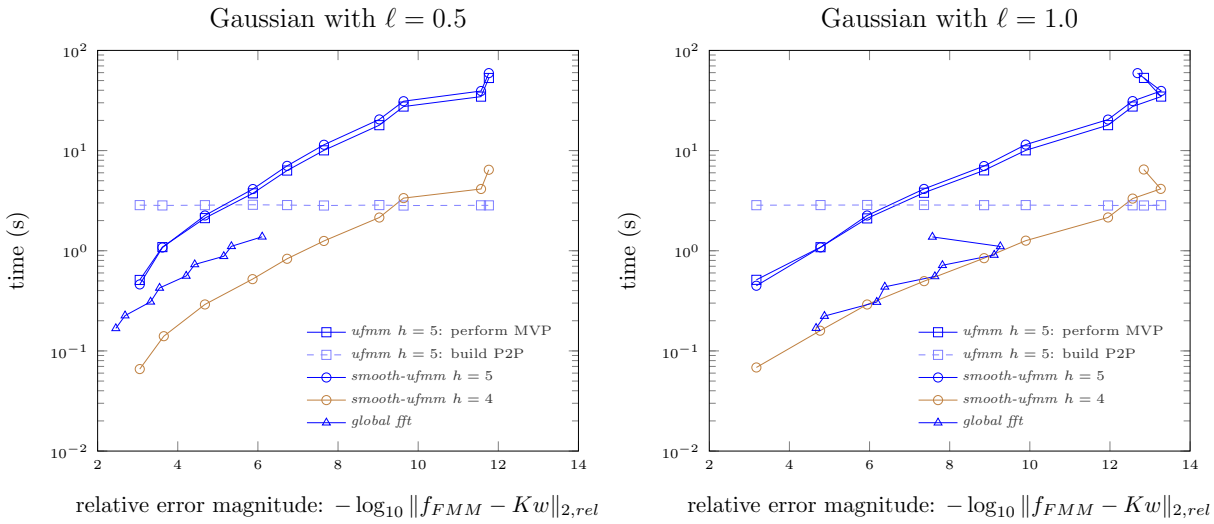

Figure 9: Computational time of a MVP *w.r.t.* the relative error magnitude using various algorithms: *ufmm* and *smooth-ufmm* with $p = 2 \dots 12$, and *global fft* with $p = 7 \dots 15$. We used $72k$ particles randomly distributed in a cube. Observations: *ufmm* and *smooth-ufmm* have approximately the same cost when they share the same tree depth. The *smooth-ufmm* is significantly faster if we choose an optimal tree depth, *e.g.*, $h = 4$ represented in brown. If the Gaussian decreases sufficiently slow (*i.e.*, the rank is sufficiently low), then the cost of the *global fft* is similar to the optimal *smooth-ufmm*. However the *global fft* exhibits an instability for the highest interpolation order.

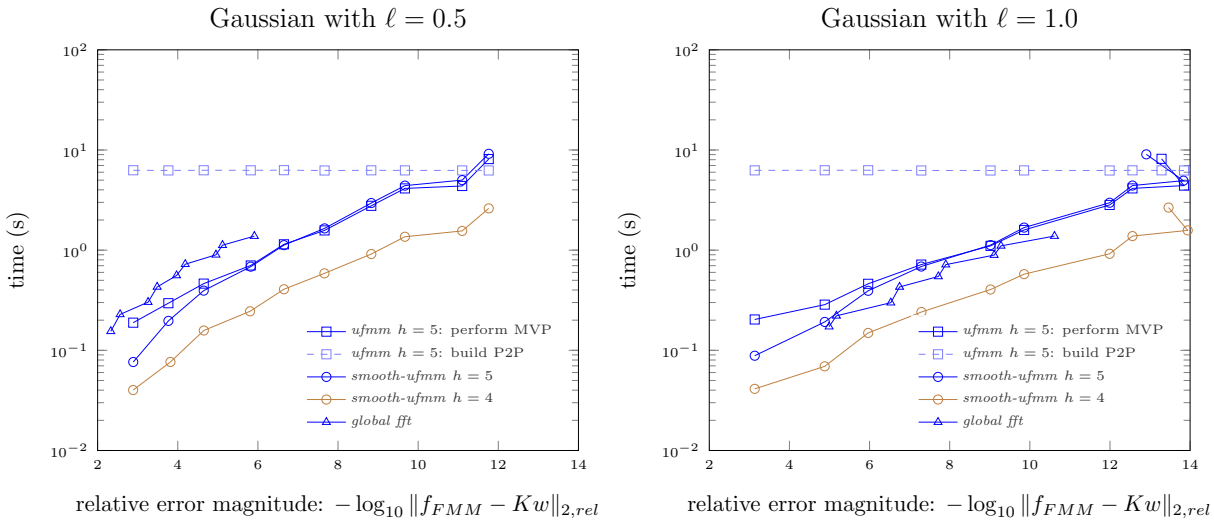

Figure 10: Computational time of a MVP *w.r.t.* the relative error magnitude using various algorithms: *ufmm* and *smooth-ufmm* with $p = 2 \dots 12$, and *global fft* with $p = 7 \dots 15$. We used $72k$ particles randomly distributed on the unit sphere. Observations: *ufmm* and *smooth-ufmm* have approximately the same cost when they share the same tree depth. The *smooth-ufmm* is significantly faster if we choose an optimal tree depth, *e.g.*, $h = 4$ represented in brown. If the Gaussian decreases sufficiently slow (*i.e.*, the rank is sufficiently low), then the cost of the *global fft* is slightly lower than the *ufmm* but the optimal *smooth-ufmm* still performs better.

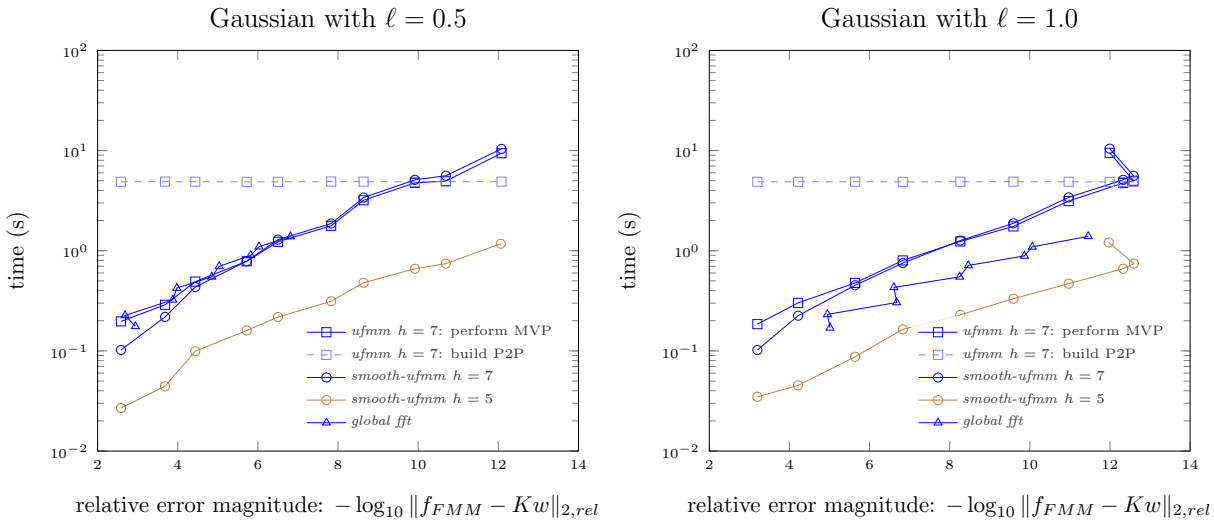

Figure 11: Computational time of a MVP *w.r.t.* the relative error magnitude using various algorithms: *ufmm* and *smooth-ufmm* with $p = 2 \dots 12$, and *global fft* with $p = 7 \dots 15$. We used $72k$ particles randomly distributed on a prolate ellipsoid (ratio 1:1:10). Observations: *ufmm* and *smooth-ufmm* have approximately the same cost when they share the same tree depth. The *smooth-ufmm* is significantly faster if we choose an optimal tree depth, *e.g.*, $h = 5$ represented in brown. If the Gaussian decreases sufficiently slow (*i.e.*, the rank is sufficiently low), then the *global fft* performs relatively well compared to the hierarchical variants.

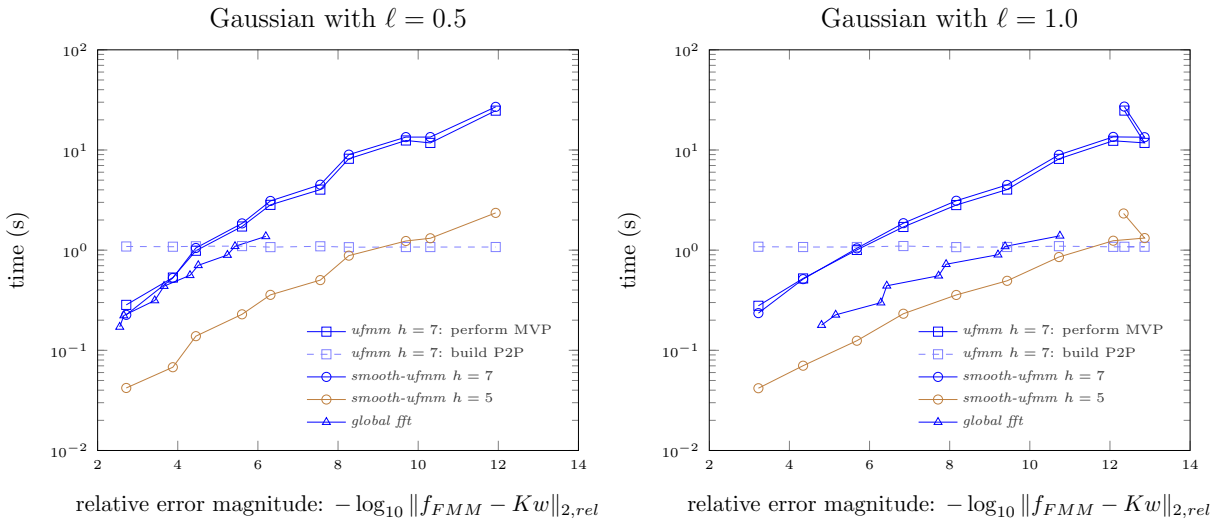

Figure 12: Computational time of a MVP *w.r.t.* the relative error magnitude using various algorithms: *ufmm* and *smooth-ufmm* with $p = 2 \dots 12$, and *global fft* with $p = 7 \dots 15$. We used $72k$ particles randomly distributed on a hyperbolic paraboloid (ratio 10:10:1). Observations: *ufmm* and *smooth-ufmm* have approximately the same cost when they share the same tree depth. The *smooth-ufmm* is significantly faster if we choose an optimal tree depth, *e.g.*, $h = 5$ represented in brown. If the Gaussian decreases sufficiently slow (*i.e.*, the rank is sufficiently low), then the *global fft* performs relatively well compared to the hierarchical variants.

References

[1] William Fong and Eric Darve. The black-box fast multipole method. *Journal of Computational Physics*, 228(23):8712–8725, 2009.

[2] Pierre Blanchard, Bérenger Bramas, Olivier Coulaud, Eric Darve, Laurent Dupuy, Arnaud Etcheverry, and Guillaume Sylvand. Scalfmm: A generic parallel fast multipole library. In *Computational Science and Engineering (CSE)*. SIAM, March 2015.

[3] Emmanuel Agullo, Bérenger Bramas, Olivier Coulaud, Eric Darve, Matthias Messner, and Toru Takahashi. Task-based fmm for multicore architectures. *SIAM Journal on Scientific Computing*, 36(1):C66–C93, 2014.

[4] Emmanuel Agullo, Bérenger Bramas, Olivier Coulaud, Eric Darve, Matthias Messner, and Takahashi Toru. Pipelining the fast multipole method over a runtime system. *arXiv preprint arXiv:1206.0115*, 2012.

[5] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[6] Leslie Greengard and Vladimir Rokhlin. *On the efficient implementation of the fast multipole algorithm*. Yale University, Department of Computer Science, 1988.

[7] William D Elliott and John A Board, Jr. Fast fourier transform accelerated fast multipole algorithm. *SIAM Journal on Scientific Computing*, 17(2):398–415, 1996.

[8] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335 – 366, 2008.

[9] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167–20172, 2007.

[10] Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. A randomized algorithm for the approximation of matrices. Technical report, DTIC Document, 2006.

[11] Per-Gunnar Martinsson. A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1251–1274, 2011.

[12] Edmund Bertschinger. Multiscale gaussian random fields and their application to cosmological simulations. *The astrophysical journal supplement series*, 137(1):1, 2001.

[13] Julien Carron, Melody Wolk, and Istvan Szapudi. On fast generation of cosmological random fields. *Monthly Notices of the Royal Astronomical Society*, 444(1):994–1000, 2014.

[14] Mickaële Le Ravalec, Benoît Noetinger, and Lin Y Hu. The fft moving average (fft-ma) generator: An efficient numerical method for generating and conditioning gaussian simulations. *Mathematical Geology*, 32(6):701–723, 2000.

[15] MJL Robin, AL Gutjahr, EA Sudicky, and JL Wilson. Cross-correlated random field generation with the direct fourier transform method. *Water Resources Research*, 29(7):2385–2397, 1993.

[16] Alberto Bellin and Yoram Rubin. Hydro_gen: A spatially distributed random field generator for correlated properties. *Stochastic Hydrology and Hydraulics*, 10(4):253–278, 1996.

[17] Adolfo J Banchio and John F Brady. Accelerated stokesian dynamics: Brownian motion. *The Journal of chemical physics*, 118(22):10323–10332, 2003.

[18] Masanobu Shinozuka and C-M Jan. Digital simulation of random processes and its applications. *Journal of sound and vibration*, 25(1):111–128, 1972.

[19] Aristotelis Mantoglou and John L Wilson. The turning bands method for simulation of random fields using line generation by a spectral method. *Water Resources Research*, 18(5):1379–1394, 1982.

[20] CR Dietrich and GN Newsam. A fast and exact method for multidimensional gaussian stochastic simulations. *Water Resources Research*, 29(8):2861–2869, 1993.

[21] Andrew TA Wood and Grace Chan. Simulation of stationary gaussian processes in [0, 1] d. *Journal of computational and graphical statistics*, 3(4):409–432, 1994.

[22] Vahid Dehdari and ClaytonV. Deutsch. Applications of randomized methods for decomposing and simulating from large covariance matrices. In Petter Abrahamsen, Ragnar Hauge, and Odd Kolbjørnsen, editors, *Geostatistics Oslo 2012*, volume 17 of *Quantitative Geology and Geostatistics*, pages 15–26. Springer Netherlands, 2012.

[23] Matthias Messner, Bérenger Bramas, Olivier Coulaud, and Eric Darve. Optimized m2l kernels for the chebyshev interpolation based fast multipole method. *CoRR*, abs/1210.7292, 2012.

[24] John P Boyd and Jun Rong Ong. Exponentially-convergent strategies for defeating the runge phenomenon for the approximation of non-periodic functions, part i: Single-interval schemes. commun. *Comput. Phys*, 5(2-4):484–497, 2009.

[25] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[26] Petros Drineas and Michael W Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.

[27] Kai Zhang, Ivor W Tsang, and James T Kwok. Improved nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1232–1239. ACM, 2008.

[28] Michael W Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[29] Judith Yue Li, Sivaram Ambikasaran, Eric F Darve, and Peter K Kitanidis. A kalman filter powered by h2-matrices for quasi-continuous data assimilation problems. *Water Resources Research*, 2014.

[30] William B March and George Biros. Far-field compression for fast kernel summation methods in high dimensions. *arXiv preprint arXiv:1409.2802*, 2014.