



Parallel Fast Direct Solver: \mathcal{H} -Matrix. Applications to Uncertainty Management

HPC and Uncertainty Treatment

Denis BARBIER, IMACS
EDF, Airbus Group, Phimeca, CEA, IMACS
Prace Advanced Training Center

Outline

- 1 Introduction
 - Large linear systems in Statistics
 - Kriging
- 2 \mathcal{H} -Matrices
- 3 Applications
- 4 Conclusion and Perspectives

Introduction

Several applications in Statistics lead to large, dense matrices :

- Kriging
- PCA, *etc.*
- Applications using large covariance matrices

Kriging

- Symmetric Positive Definite (SPD) matrix
- Many solves
- Usually,
 - Cholesky decomposition $M = LL^T$
 - Forward - Backward substitutions

Kriging

In a nutshell

Let $Z : \mathbb{R}^d \rightarrow \mathbb{R}$ be some random process, stationary of order 2.

We have N observation points $\{x_i \in \mathbb{R}^d | i = 1, \dots, N\}$ with values $Z(x_i)$.

Let us assume that the covariance of these points is known and given by a matrix $K \in \mathbb{R}^{N \times N}$, with

$$K_{ij} = \text{Cov}(Z(x_i), Z(x_j))$$

An estimation of the mean trajectory is a linear interpolation $\tilde{Z}(x_0)$ of Z at $x_0 \in \mathbb{R}^d$:

$$\tilde{Z}(x_0) = \sum_{i=1}^N \alpha_i(x_0) Z(x_i)$$

The weights α_i are solution of

$$K\alpha = K_0$$

with

$$(K_0)_i := \text{Cov}(Z(x_i), Z(x_0))$$

Kriging

Writing the systems

- K is SPD (covariance matrix)
- Quite often, K isn't known
 - Modelled as a convolution matrix of a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow R_+$:

$$K_{ij} := k(x_i, x_j)$$

- What is k ?
 - Gaussian :
 - Exponential :
 - Quadratic :

$$k(x, y) = e^{-|x-y|^2/(2\lambda^2)}$$

$$k(x, y) = e^{-|x-y|/\lambda}$$

$$k(x, y) = \left(1 + \frac{|x - y|}{2\lambda}\right)^{-2}$$

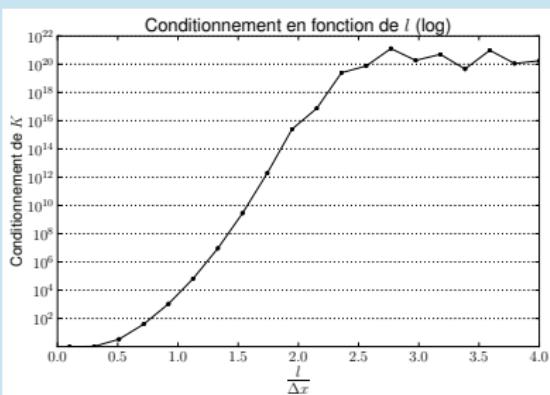
- N can be large. For instance, every node in an FEM discretization
- The interpolation is sought at many points x_0

Kriging

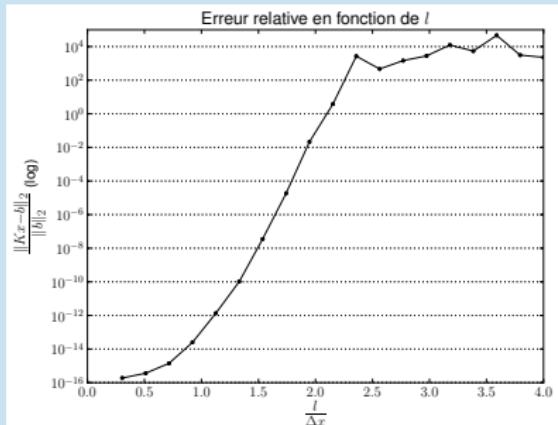
Solving the systems

- K is ill-conditionned [ABW94], many right-hand sides \Rightarrow Direct Solver
- K is SPD \Rightarrow Cholesky

Conditioning



: Condition number



: Relative Error

FIGURE: Condition number and error growth for a gaussian kernel.

Kriging

Solving the systems

Complexity

- Cholesky Decomposition
DPOTRF $1/3N^3 + 1/2N^2 + 1/6N$ [BD99]
- Solving
DPOTRS $N_{rhs} \times 2N^2$
- Storage : $8N^2/2$

Need of a **fast direct solver** : \mathcal{H} -Matrix

Outline

1 Introduction

2 \mathcal{H} -Matrices

- Low-rank Approximation
- Space partitioning
- Operations on \mathcal{H} -Matrices

3 Applications

4 Conclusion and Perspectives

\mathcal{H} -Matrices

Broadly speaking,

- Adapative data-sparse representation of dense matrices
- Hierarchical, adaptative and approximate representation
- Broad range of operations on these compressed matrices (GEMV, AXPY, GEMM, etc.). More versatile than the FMM.

2 parts :

- Data-sparse representation of dense matrices
- Approximate operations (eg matrix-matrix multiplication) that preserve this representation

\mathcal{H} -Matrix

Literature

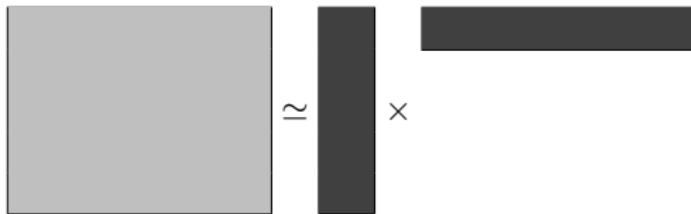
- Genesis in '99, extensions in the 2000s
[Hac99, HK00, GH03, BGH03, BGH04]
- Several application domains : FEM (elliptical operators), BEM (laplacian, waves)
- Several implementations available
- Recently, \mathcal{H}^2 -Matrix, close to the Fast Multipole Method

Low-rank approximation

- Hypothesis : it is possible to approximate a matrix by a low-rank one

$$M \simeq AB^T$$

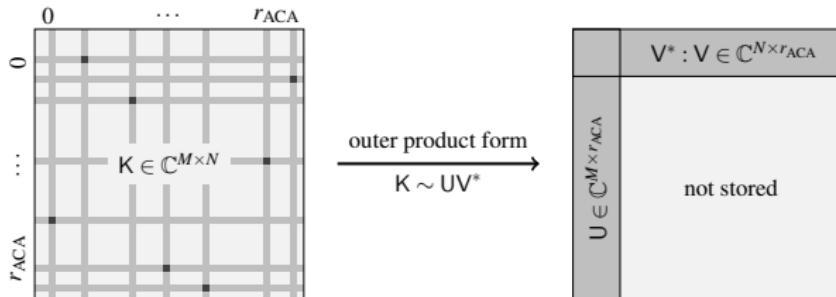
- $M \in \mathbb{C}^{m \times n}$
- $A \in \mathbb{C}^{m \times k}$
- $B^T \in \mathbb{C}^{k \times n}, k < \min(m, n)$



Low-rank approximation

Several compression algorithms

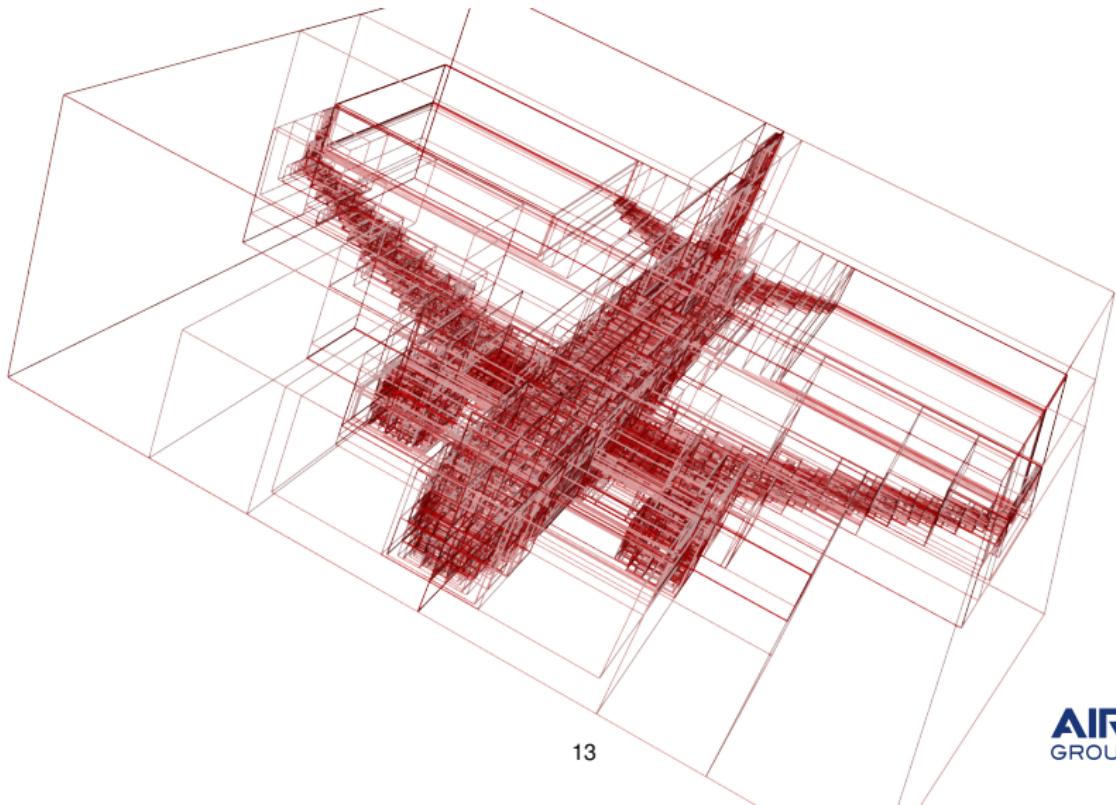
- SVD : $M = U\Sigma V$
 - $U \in \mathbb{C}^{m \times \min(m,n)}$
 - Σ diagonal matrix
 - $V \in \mathbb{C}^{\min(m,n) \times n}$
- Adaptative Cross Approximation (ACA) : Sequence or rank-1 approximations. Heuristics. [Beb11, BR03, Beb00]
 - « Full Pivoting »
 - « Partial Pivoting » (Figure : [Mes12])



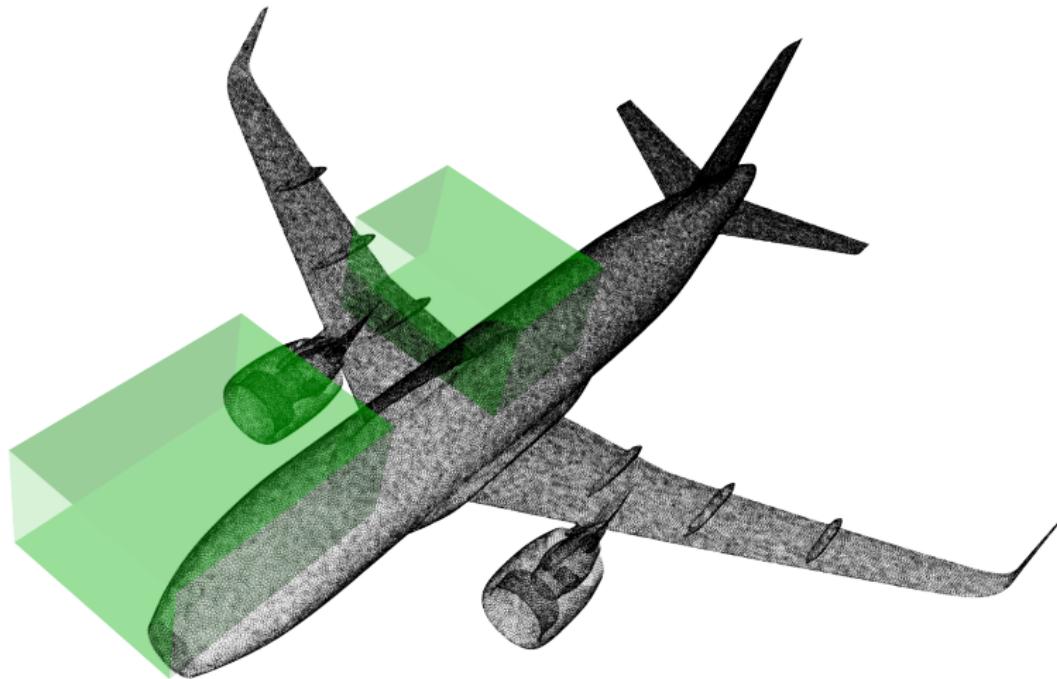
Space Partitioning

- Spatial gathering of the degrees of freedom (bounding boxes)
- Binary Space Partitioning :
 - Halves the largest dimension of the bounding box
 - Halves the degrees of freedom in each box along the largest dimension
- Stopping criteria : DoFs in a leaf

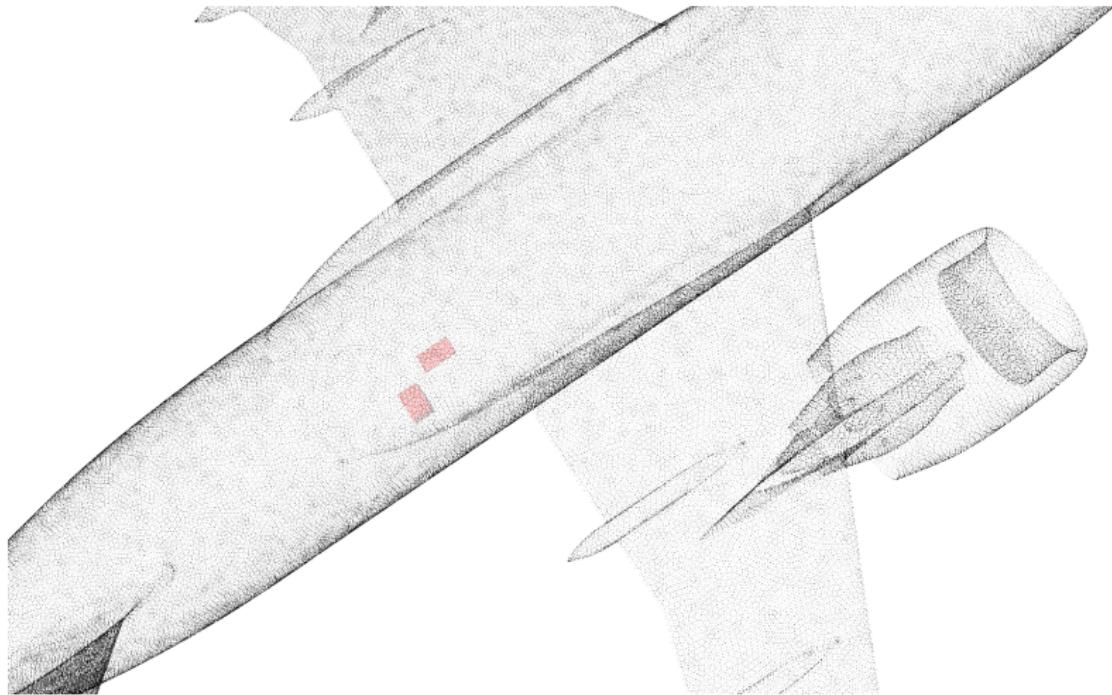
A Picture is worth a thousand words



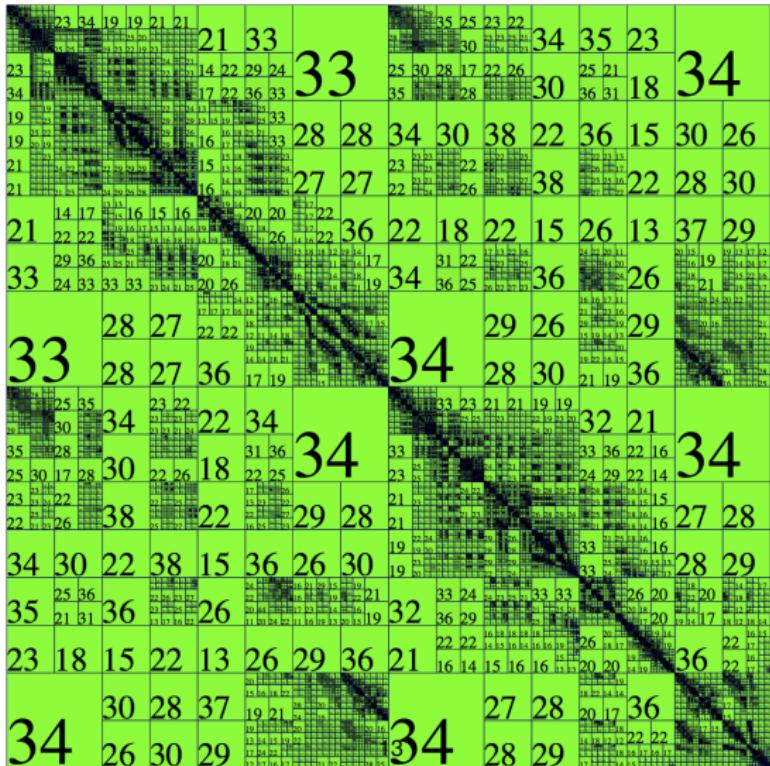
A Picture is worth a thousand words



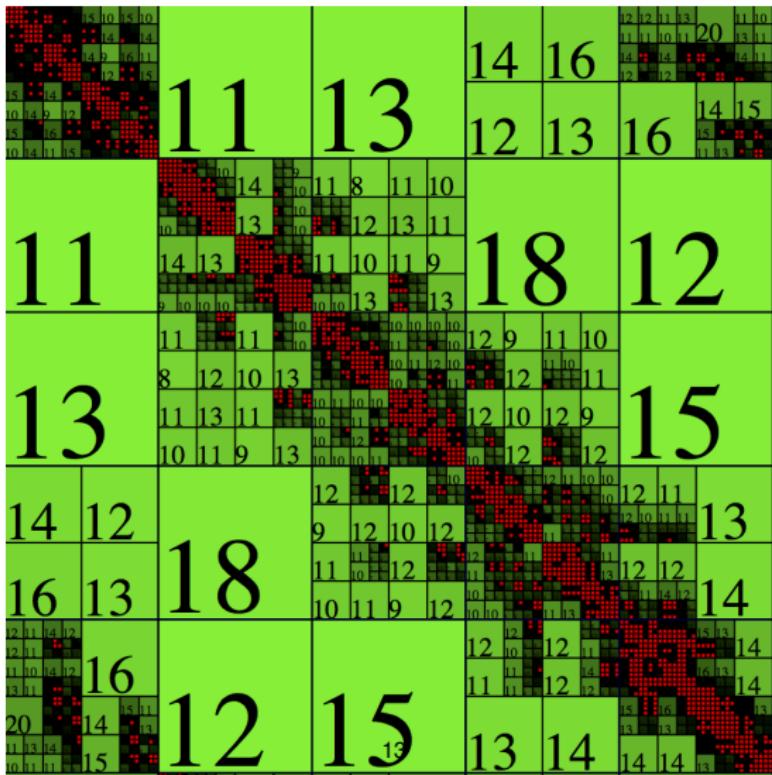
A Picture is worth a thousand words



A Picture is worth a thousand words



A Picture is worth a thousand words



\mathcal{H} -Matrix

A \mathcal{H} -Matrix is a **quadtree** (with Binary Space Partitioning) :

- Internal nodes : subdivided \mathcal{H} -Matrix
- Leaves :
 - Admissible block : $\mathcal{R}k$ -Matrix (green)
 - Non-admissible block : full block (red)

Remarks

- Only the leaves carry data
- Big admissible blocks ($10^4 \times 10^4 +$)
- Small (and few) non-admissible blocks (100×100)

Operations

3 classes of operations :

\mathcal{H} -BLAS1 - \mathcal{H} -BLAS2 Assembly, AXPY, GEMV.

Simple. Operating only on the leaves (adaptive recompression for AXPY).

\mathcal{H} -BLAS3 GEMM, TRSV.

More involved. Operations at several levels of the same sub-tree. 27 cases for GEMM, 1 recursion case and 26 base cases, hierarchical conversion.

\mathcal{H} -LAPACK Inverse, LU , LDL^T .

Uses « BLAS » 2 et 3 operations, harder to implement in parallel.

Base operations

All the operations use BLAS/LAPACK.

Subroutines : SVD, QR, LU, TRSV, GEMM, GEMV

Matrix-Vector Product

GEMV

```
function VECTORPRODUCT( $M, x, y, \sigma \times \tau$ )
  if IsLeaf( $\sigma \times \tau$ ) then
     $y|_{\sigma \times \tau} \leftarrow M|_{\sigma \times \tau} x|_{\sigma \times \tau}$ 
  else
    for all  $(\sigma', \tau') \in S(\sigma \times \tau)$  do
      VectorProduct( $M, x, y, \sigma' \times \tau'$ )
    end for
  end if
end function
```

▷ Full or Compressed

Addition

```
function ADD( $M_1, M_2, \sigma \times \tau$ )
    if IsLeaf( $\sigma \times \tau$ ) then
         $M_1|_{\sigma \times \tau} \leftarrow M_1|_{\sigma \times \tau} + M_2|_{\sigma \times \tau}$ 
    else
        for all  $(\sigma', \tau') \in S(\sigma \times \tau)$  do
            Add( $M_1, M_2, \sigma' \times \tau'$ )
        end for
    end if
end function
```

▷ Full or Compressed

Remarks

- Adding two $\mathcal{R}k$ -Matrices changes the rank (recompression) \Rightarrow unknown final rank
- Costly operation

Multiplication

$$C \leftarrow AB + C$$

- Each operand (A, B, C) can be an internal node, compressed or full leaf
- 27 cases !
- In fact, guided by C structure.

Ideas :

- A, B, C internal nodes : recursive call
- C full leaf : convert A et B in full matrices
- A ou B full or compressed leaf : full or compressed product and recursive formatted add into C
- C compressed leaf : multiplication and hierarchical conversion

Multiplication

```
function GEMM( $C, \alpha, A, B, \beta, \sigma \in \mathcal{T}_I, \tau \in \mathcal{T}_J, \rho \in \mathcal{T}_K$ )
    if IsRoot( $C$ ) then
         $C \leftarrow \beta C$ 
    end if
    if  $\neg \text{ISLEAF}(\sigma \times \tau)$  et  $\neg \text{ISLEAF}(\sigma \times \rho)$  et  $\neg \text{ISLEAF}(\rho \times \tau)$  then
        for all  $\sigma' \in S(\sigma)$  do
            for all  $\tau' \in S(\tau)$  do
                for all  $\rho' \in S(\rho)$  do
                    GEMM( $C, \alpha, A, B, 1, \sigma', \tau', \rho'$ )
                end for
            end for
        end for
    else
         $C|_{\sigma \times \tau} \leftarrow \alpha A|_{\sigma \times \rho} \cdot B|_{\rho \times \tau} + C|_{\sigma \times \tau}$  ▷ Base Case
    end if
end function
```

LU

```

function LUDECOMPOSITION( $M$ )
  if ISLEAF( $M$ ) then
     $M \leftarrow LU$                                       $\triangleright M$  is full, use LAPACK
  else
     $M := \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}, L := \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}, U := \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}$ 
    LUDECOMPOSITION( $M_{11}$ )                          $\triangleright M_{11} \leftarrow L_{11}U_{11}$ 
    Solve  $L_{11}U_{12} = M_{12}$                        $\triangleright$  Upper Triangular System
    Solve  $L_{21}U_{11} = M_{21}$                        $\triangleright$  Lower Triangular System
     $M_{22} \leftarrow M_{22} - L_{21}U_{12}$ 
    LUDECOMPOSITION( $M_{22}$ )                          $\triangleright M_{22} \leftarrow L_{22}U_{22}$ 
  end if
end function

```

Complexity Estimates

- Most complexity estimates assume a constant rank k for the $\mathcal{R}k$ -Matrices
- For ACA with Full Pivoting, $\mathcal{O}(N^2)$, for SVD $\mathcal{O}(N^3)$ (time)
- Assuming a **sparse** Cluster Tree and **almost idempotent** Block Tree [BGH04]
 - Storage
$$N_{st}(\mathcal{T}_{I \times I}, k) \leq 2C_{sp}(p+1) \max\{k, N_{leaf}\} N$$
with $\mathcal{T}_{I \times I}$ the Block Tree of depth $p \propto \log(N)$.
 - Multiplication
$$N_{\times}(\mathcal{T}_{I \times I}, k) \leq K_5 C_{sp}^3 C_{id}^3 k^3 (p+1)^3 \max\{|I|, |\mathcal{L}(\mathcal{T}_{I \times I})|\}$$

Complexity Estimates

- Most complexity estimates assume a constant rank k for the $\mathcal{R}k$ -Matrices
- For ACA with Full Pivoting, $\mathcal{O}(N^2)$, for SVD $\mathcal{O}(N^3)$ (time)
- Assuming a **sparse** Cluster Tree and **almost idempotent** Block Tree [BGH04]
 - Storage $\mathcal{O}(N \log_2(N))$
 - Multiplication $\mathcal{O}(N \log_2^2(N))$

Computations

Issues

\mathcal{H} -Matrix algorithms are not nice for the hardware :

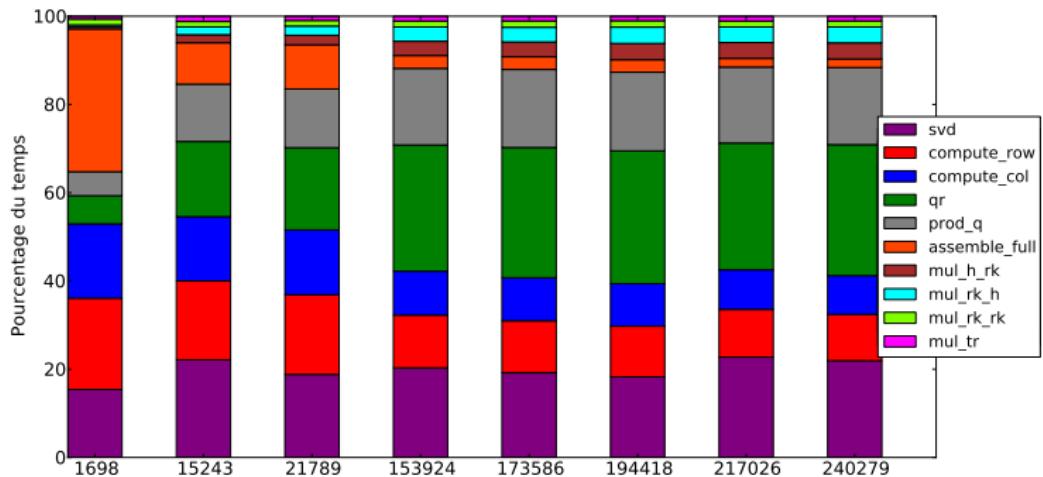
- Very small operations
- Oddly-shaped matrices : « Tall and Skinny »
- High memory bandwidth

Observations

- Most (70-80%) of the time spent in :
 - QR decompositions of T&S matrices (eg. $10^4 \times 30$)
 - SVD decompositions of small matrices (eg. 30×30)
- BLAS implementations cannot reach the peak performance
- Very high memory bandwidth. Ex : on 12 cores, allocation rate : $\sim 1\text{GB/s}$

Relative cost of operations

LU decomposition, same object (Cone-Sphere), different sizes



Computations

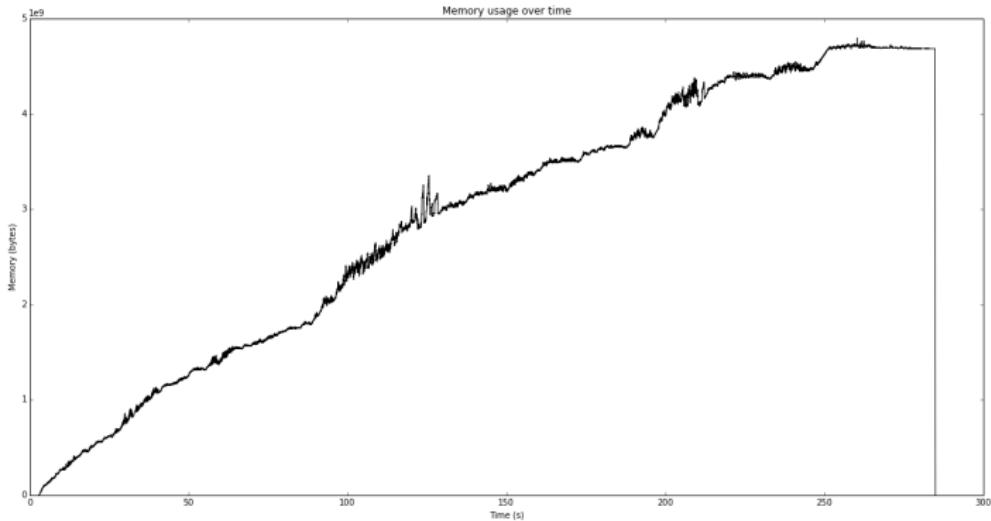
- Exemple : Cone-Sphere, 135 429 unknowns, simple complex, LU
- 2 × 6 Intel Xeon « Sandy Bridge », 2.3GHz, MKL 11, `icc` 13, Linux
- 205.6s
- 90.8% of the time spent in BLAS/LAPACK
- 148 GFlops (5.36 Flop / cycle)
- ...but requires 217.48× less operations than a dense solver

Operation	Gflop/s	#Calls	Total Runtime (s)
$M = U\Sigma V^*$	14.09	883 796	814.22
$M = QR$	12.7	1 635 818	708.25
QM	11.7	1 635 818	506.28
MU	13.50	817 909	33.12
$\mathcal{H} \times (AB^T)$	10.72	36437	66.4

Computations

Memory

- 9 974 346 allocations (48 893/s)
- 317 GB (1.55 GB/s)



Outline

- 1 Introduction
- 2 \mathcal{H} -Matrices
- 3 Applications
- 4 Conclusion and Perspectives

Environment

Solver

- Airbus Group Innovations in-house solver
- C++
- Parallel implementation : Enumerated Task Flow (ETF) on top of a runtime system (eg. StarPU) [Liz14]
- Shared and distributed memory

Configuration

- 2 × 6 Intel Xeon « Sandy Bridge », 2.2GHz, 48GB
- Linux, ICC 13, MKL

2D Grid

- x_i are on a regular 2D grid with Δx pitch.
- $\lambda = \Delta x$
- Sequential LDL^T decomposition
- $\varepsilon = 10^{-4}$, $\eta = 3$
- Gaussian, exponential and quadratic kernels

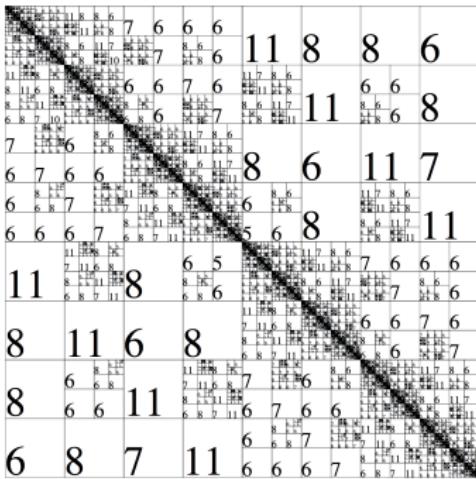
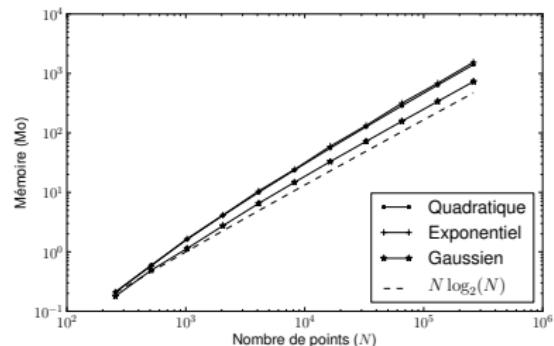
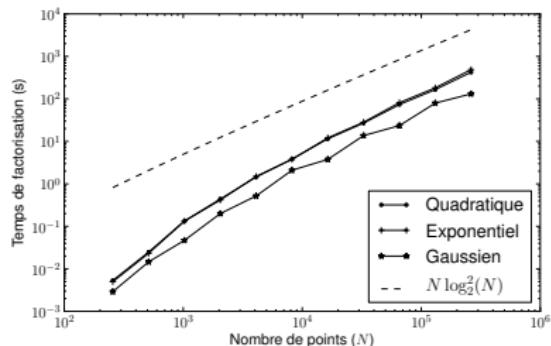


FIGURE: \mathcal{H} -Matrix for $N = 2^{18}$ with the quadratic kernel.

2D Grid



: Memory



: Decomposition time

FIGURE: Memory and computation time for several kernels.

2D Grid

$$\varepsilon_{rel} := \frac{\|Kx - K_0\|_2}{\|K_0\|_2}$$

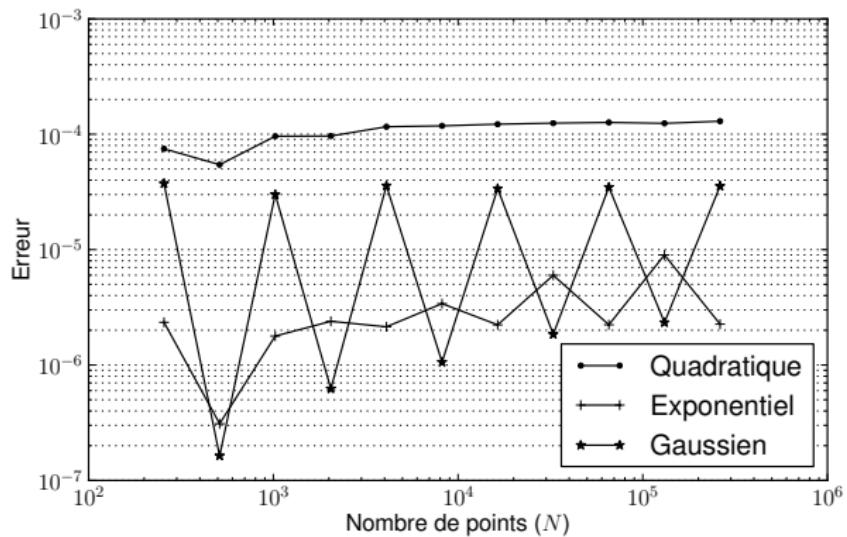


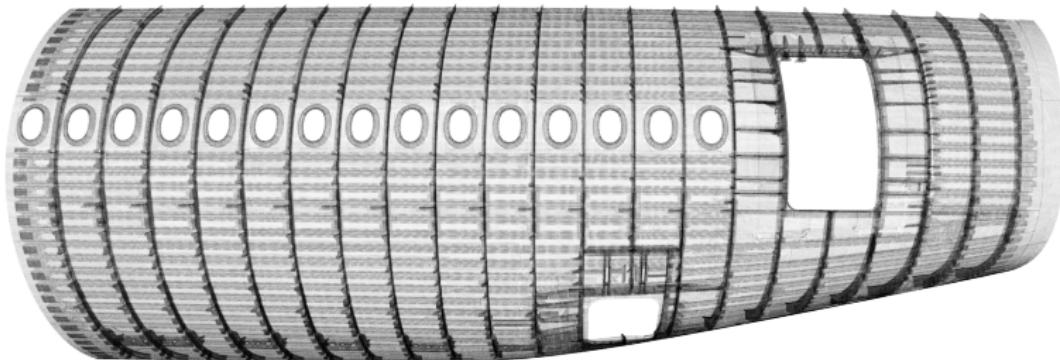
FIGURE: ε_{rel} for several kernels.

3D FEM Mesh

- 3D FEM Mesh
- Using a random subset of the mesh
- Exponential kernel

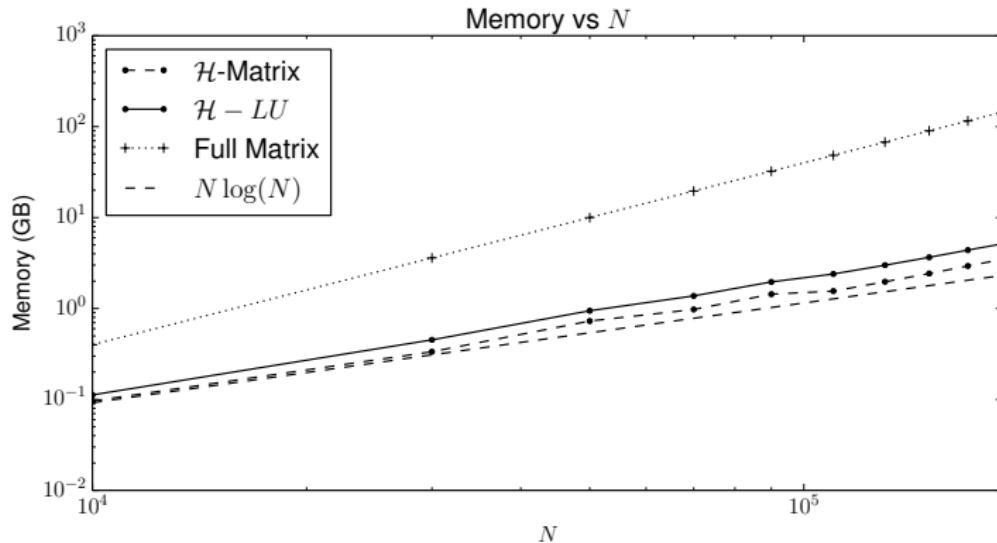
$$\lambda = \frac{1}{10} \max_{k=1,2,3} \left(\max_i x_i^k - \min_i x_i^k \right)$$

- $\varepsilon = 10^{-4}$, $\eta = 2$, ACA+, LU decomposition



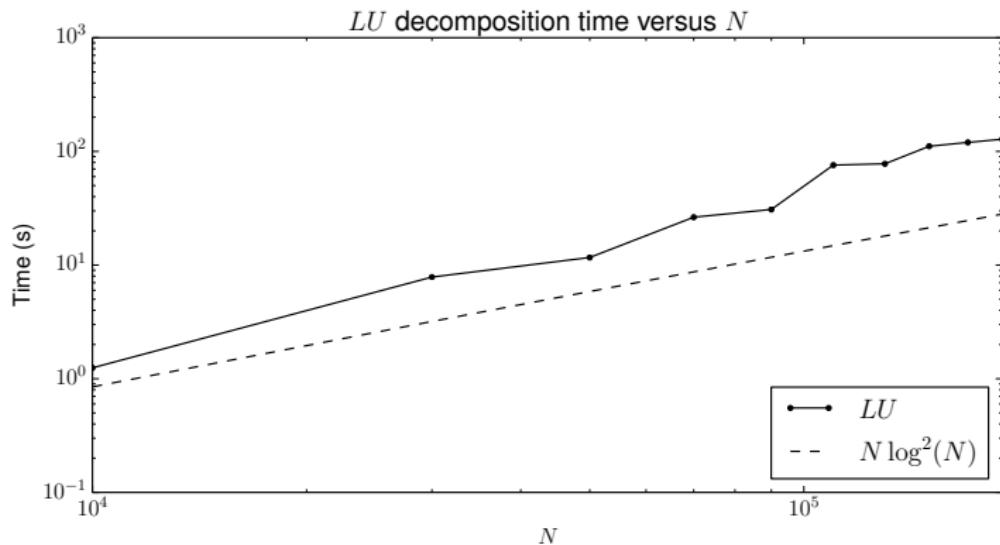
3D FEM Mesh

Memory



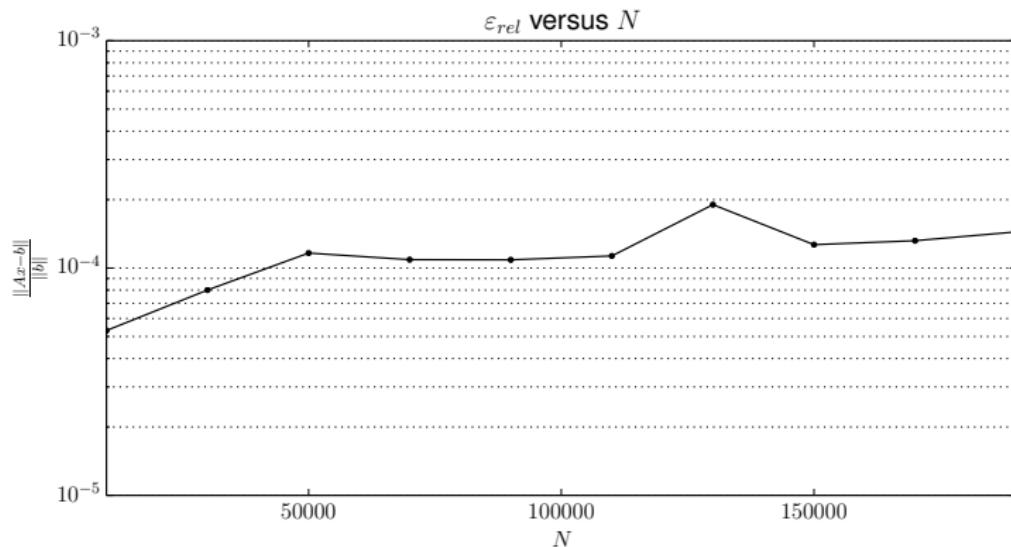
3D FEM Mesh

Time



3D FEM Mesh

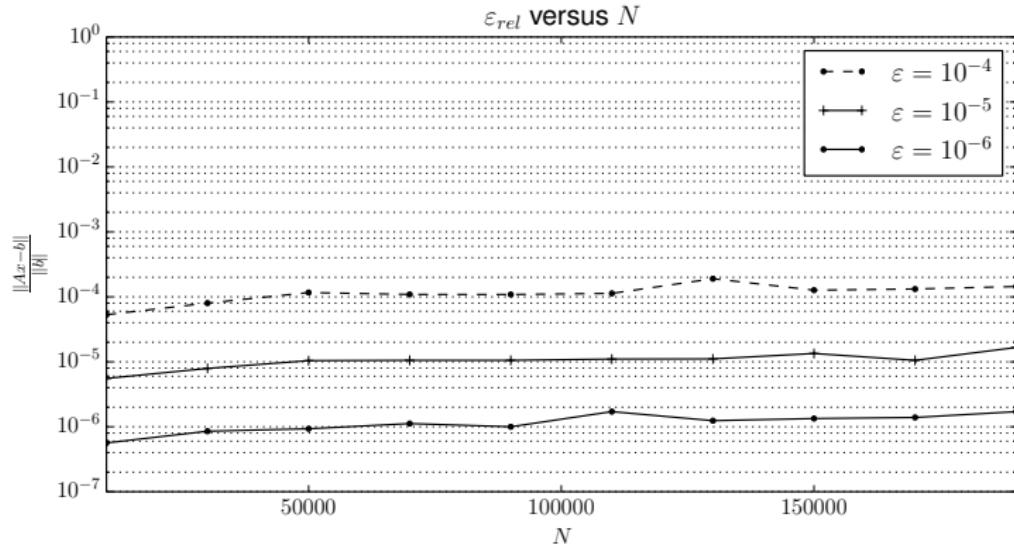
Error



3D FEM Mesh

Error Control

Comparing the error with $\varepsilon = 10^{-4}$, 10^{-5} and 10^{-6}



Outline

- 1 Introduction
- 2 \mathcal{H} -Matrices
- 3 Applications
- 4 Conclusion and Perspectives

Conclusion

- \mathcal{H} -Matrices are an enabler for many statistics problems
- Efficient parallelization is still needed, unfortunately the \mathcal{H} -Matrix arithmetic is highly irregular
- At least one order of magnitude for the addressable problem size on a laptop or workstation



Parallel Fast Direct Solver: \mathcal{H} -Matrix. Applications to Uncertainty Management

HPC and Uncertainty Treatment

Denis BARBIER, IMACS
EDF, Airbus Group, Phimeca, CEA, IMACS
Prace Advanced Training Center

Bibliographie I

- [ABW94] Rachid Ababou, Amvrossios C Bagtzoglou, and Eric F Wood.
On the condition number of covariance matrices in kriging,
estimation, and simulation of random fields.
Mathematical Geology, 26(1) :99–133, 1994.
- [BD99] Susan Blackford and Jack J. Dongarra.
Installation guide for LAPACK.
Technical Report 41, LAPACK Working Note, June 1999.
originally released March 1992.
- [Beb00] M. Bebendorf.
Approximation of boundary element matrices.
Numer. Math., 86(4) :565–589, 2000.
- [Beb11] M. Bebendorf.
Adaptive cross approximation of multivariate functions.
Constructive Approximation, 34 :149–179, 2011.

Bibliographie II

- [BGH03] Steffen Börm, Lars Grasedyck, and Wolfgang Hackbusch.
Introduction to hierarchical matrices with applications.
Engineering Analysis with Boundary Elements, 27(5) :405 – 422, 2003.
Large scale problems using BEM.
- [BGH04] Steffen Börm, Lars Grasedyck, and Wolfgang Hackbusch.
Hierarchical matrices.
Technical report, Max Planck Institut für Mathematik, 2004.
- [BR03] M. Bebendorf and S. Rjasanow.
Adaptive low-rank approximation of collocation matrices.
Computing, 70 :1–24, 2003.

Bibliographie III

- [GH03] Lars Grasedyck and Wolfgang Hackbusch.
Construction and arithmetics of h-matrices.
Computing, 70 :295–334, 2003.
10.1007/s00607-003-0019-1.
- [Hac99] W. Hackbusch.
A sparse matrix arithmetic based on h-matrices. part i :
Introduction to h-matrices.
Computing, 62 :89–108, 1999.
10.1007/s006070050015.
- [HK00] W. Hackbusch and B. N. Khoromskij.
A sparse h-matrix arithmetic.
Computing, 64 :21–47, 2000.
10.1007/s006070050002.

Bibliographie IV

- [Liz14] Benoît Lizé.
Résolution Directe Rapide pour les Éléments Finis de Frontière en Électromagnétisme et Acoustique : \mathcal{H} -Matrices. Parallelisme et Applications Industrielles.
PhD thesis, Université Paris 13, 2014.
- [Mes12] Matthias Messner.
Fast Boundary Element Methods in Acoustics.
PhD thesis, Université Bordeaux I, 2012.