

Parallel Fast Direct Solver: applications to Uncertainty Management

An overview of the \mathcal{H} -matrix framework

Kieran Delamotte

IMACS

18th May, 2018



Context

\mathcal{H} -matrices

Applications

Conclusions & perspectives

Context

\mathcal{H} -matrices

Applications

Conclusions & perspectives

Introduction

Several applications in statistics lead to large and dense matrices :

- Kriging ;
- applications using large covariance matrices (ex : random Gaussian sampling).

Kriging

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

Kriging

in a nutshell

Let $Z : \mathbb{R}^d \mapsto \mathbb{R}$ be some random process, stationnary of order 2.
We have N observation points $\{x_i \in \mathbb{R}^d / i = 1, \dots, N\}$ with values $Z(x_i)$. Let 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 the linear system

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

Kriging

writing the systems

- K is SPD (covariance matrix);
- usually K is not known exactly :
 - modelled as a convolution matrix of a kernel $k : \mathbb{R}^d \mapsto \mathbb{R}$:

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

- what is k ?
 - Exponential :

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

- Gaussian :

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

- Quadratic :

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

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

Kriging

solving the system

- K is ill-conditioned, many RHS : direct solver ;
- K is SPD : Cholesky.

Complexity (LAPACK)

- Cholesky factorization (DPOTRF) : $1/3N^3 + 1/2N^2 + 1/6N$
- Solving (DPOTRS) : $N_{\text{RHS}} \times 2N^2$
- Storage : $8N^2/2$

Need of a fast direct solver : \mathcal{H} -matrix framework.

Toy model : exponential kernel in 1D

Let X_N be a uniform discretization of $[0, 1]$ with the discretization step $h = \frac{1}{N-1}$:

$$X_N = \{0 = x_1, \dots, x_N = 1\}$$

The correlation length λ is set to $\lambda := 5h$ and the exponential kernel in 1D reads as

$$K_\lambda(x_i, x_j) = e^{-|x_i - x_j|/\lambda}$$

Toy model : exponential kernel in 1D

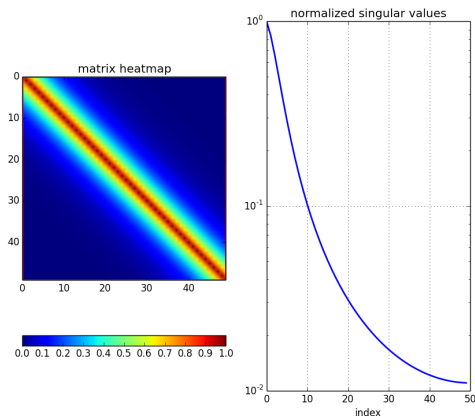


FIGURE : the covariance matrix $K_\lambda([0, 1], [0, 1])$

Toy model : exponential kernel in 1D

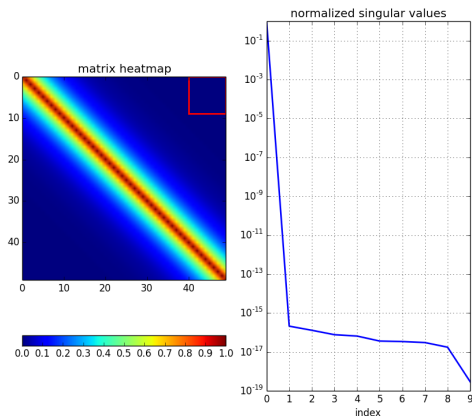


FIGURE : small extra-diagonal block : $K_\lambda([0, 0.2], [0.8, 1])$

Toy model : exponential kernel in 1D

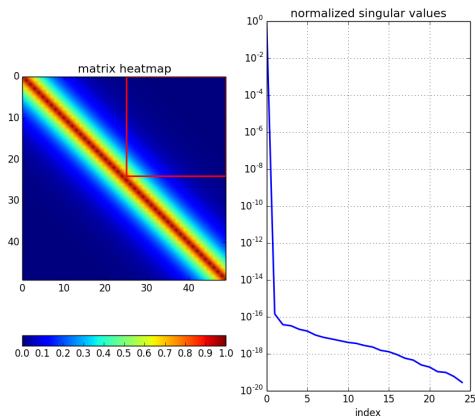


FIGURE : large extra-diagonal block : $K_\lambda([0, 0.5], [0.5, 1])$

Toy model : exponential kernel in 1D

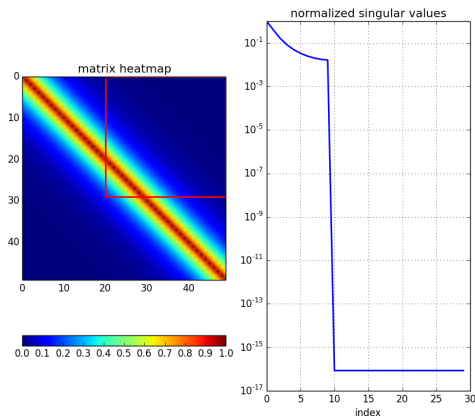


FIGURE : taking a part of the diagonal : $K_\lambda([0, 0.6], [0.4, 1])$

Toy model : exponential kernel in 1D

Whenever x_i and x_j are in disjoint sets the kernel reads as a separated one. For instance ; if $x_i > x_j$ then

$$K_\lambda(x_i, x_j) = e^{(x_j - x_i)/\lambda} = e^{x_j/\sqrt{\lambda}} e^{-x_i/\sqrt{\lambda}},$$

which is of rank 1.

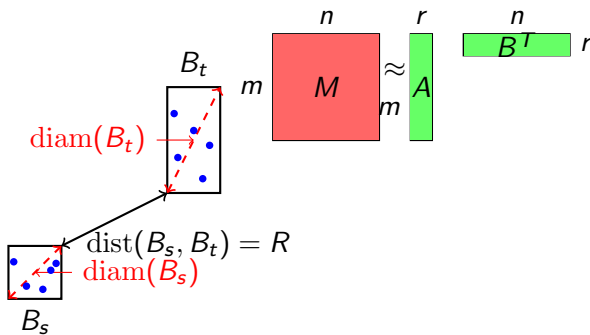
Context

\mathcal{H} -matrices

Applications

Conclusions & perspectives

Admissibility gives low-rank



Usual condition :

$$\min(\text{diam}(B_t), \text{diam}(B_s)) \leq \eta \text{dist}(B_t, B_s) \quad (\text{admissibility condition})$$

The separation condition $R = 0$ gives the **HODLR**(**H**ierarchically **O**ff-**D**agonal **L**ow-**R**ank) structure described by the 1D toy model.

Low-rank approximation : compression techniques

- SVD : $M \approx U\Sigma V^H$
 - Rank and precision controlled ;
 - **Costly** $\mathcal{O}(4m^2n + 8mn^2 + 9n^3)$ (hyp : $m > n$)
- Existence of cross approximations : row/col. extraction (Goreinov & Tyrtysnikov '97) ;
- Gaussian/LU rank-revealing scheme known as **Full Cross Approximations** :

```

1: while  $\|M\| \geq \varepsilon \|M_0\|$  : do
2:    $\text{rank}(M) \leftarrow \text{rank}(M) + 1$ 
3:   Find the coefficient  $M_{i^*j^*}$  so that  $M_{i^*j^*} = \max_{i,j} |M_{ij}|$ ,  $\alpha =$ 
      $M_{i^*j^*}$ 
4:    $M \leftarrow M - \frac{1}{\alpha} M(:, j^*) M(i^*, :)$ 
5: end while

```

Variants

The fast determination of the pivot is the main idea of all fast algorithms. Key points to speed up the full cross approximation :

- **Partially pivoted Cross Approximation.**
 - We seek the largest pivot over a column and/or a row in $\mathcal{O}(m)$ instead of $\mathcal{O}(m^2)$ operations.
 - Only the modified coefficients of the remainder are computed at each step.
- **Adaptive CA algorithm** : a fast (linear) estimation of the remainder.
- **ACA+** and other variants use other heuristics.
- Trade-off between robustness (SVD) and efficiency (ACA/ACA+) : computations from $\mathcal{O}(m^2n + mn^2)$ (SVD) to $\mathcal{O}(mnr)$ (fullCA) to $\mathcal{O}((m+n)r^2)$ (ACA).

Space partitioning : clustering

- Use of **bounding boxes** : easier to handle than point clouds ;
- Recursive splitting strategy (**Divide and Conquer strategy**) thanks to **nested bisection** :
 - geometric : the box is split in two halves along the largest axis ;
 - median : each half contains roughly the same number of unknowns ;
 - others (PCA,...).
- Split the boxes until each box contains a fixed small number of unknowns ;
- Result : **binary tree**.

Space partitioning : clustering

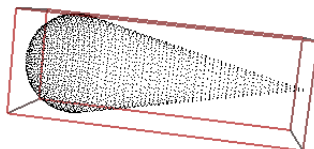


FIGURE : Illustration of the geometric clustering on a cone-sphere.

Space partitioning : clustering

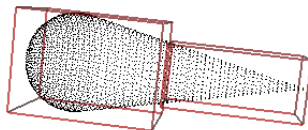


FIGURE : Illustration of the geometric clustering on a cone-sphere.

Space partitioning : clustering

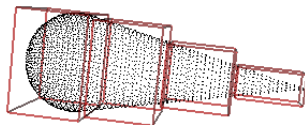


FIGURE : Illustration of the geometric clustering on a cone-sphere.

Space partitioning : clustering

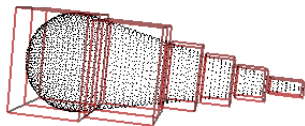


FIGURE : Illustration of the geometric clustering on a cone-sphere.

Blockclustering : Clustering & Admissibility

It is a quad-tree whose nodes are matrix blocks and the leaves are admissible (or small) blocks; a block is split in a 2×2 block structure.

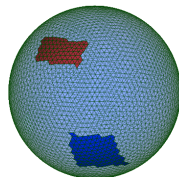
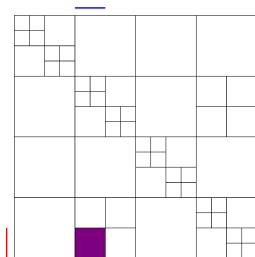


TABLE : Blockclustering and the geometry

The \mathcal{H} -matrix structure

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

- Internal nodes : subdivided \mathcal{H} -matrix ;
- Leaves :
 - admissible block : large & low-rank ;
 - inadmissible block : dense & full rank, but small.

Remarks

- Only the leaves carry data ;
- Big admissible blocks ($10^4 \times 10^4$ and more)
- Small (and few) inadmissible blocks (100×100).

Each admissible block is compressed with a fast method thus determining a numerical rank with a prescribed relative error ε .

Operations : three kinds

- \mathcal{H} -**BLAS1&2** : Assembly, AXPY, GEMV
Simple. Operating only on leaves.
- \mathcal{H} -**BLAS3** : GEMM, TRSV
More involved. Operations at many different levels of the same sub-tree.
- \mathcal{H} -**LAPACK** : Inverse, LU , LL^T .
Uses BLAS2 and BLAS3 operations, harder to implement in parallel.

Base operations

All operations use BLAS/LAPACK. Typical subroutines are : SVD, QR, LU, TRSV, GEMM and GEMV.

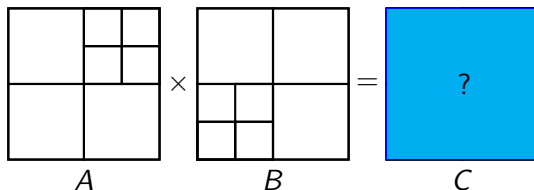
Addition

- *Usually* structures must be the same ;
- 3 different base cases :
 - sum of two dense matrices (usual dense operation) ;
 - sum of a dense and a low-rank matrix ;
 - **sum of two low-rank matrices.**

Useful pointers

- Adding two low-rank matrices : unknown final rank
- Costly operation !

Multiplication

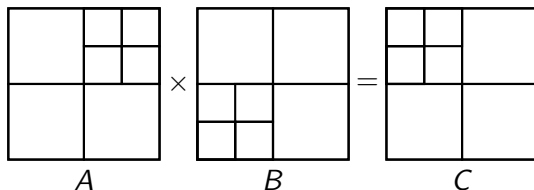


Main issues

Let $C = A \times B$ the product of 2 \mathcal{H} -matrices.

- What is the 'best' structure of C ?
- Uniqueness?
- What if it is imposed?

Multiplication

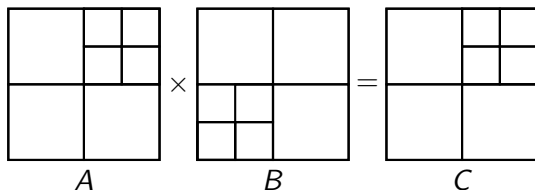


Main issues

Let $C = A \times B$ the product of 2 \mathcal{H} -matrices.

- What is the 'best' structure of C ?
- Uniqueness? (here the literature definition)
- What if it is imposed?

Multiplication

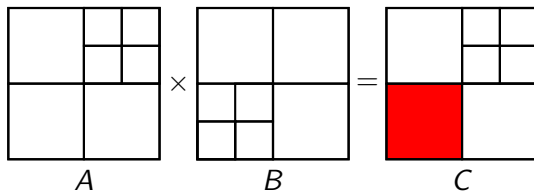


Main issues

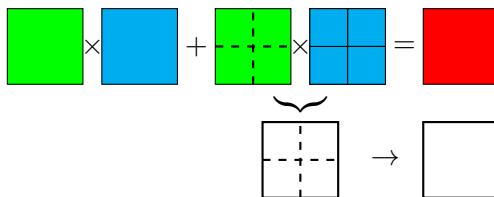
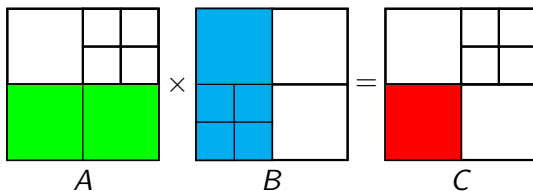
Let $C = A \times B$ the product of 2 \mathcal{H} -matrices.

- What is the 'best' structure of C ?
- Uniqueness?
- What if it is imposed? (e.g. in a LL^T factorization)

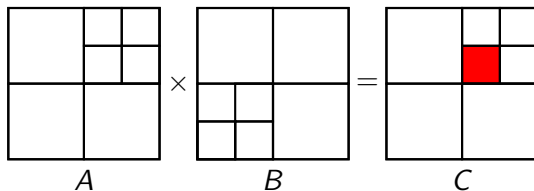
Multiplication : exemple 1



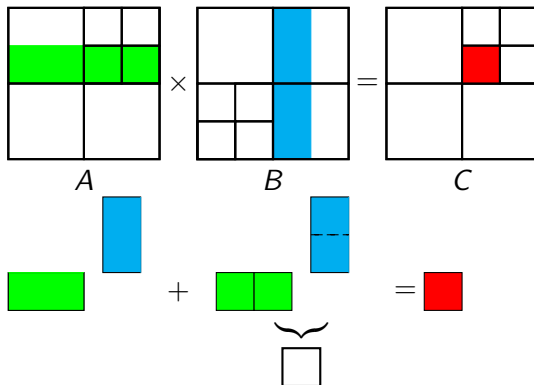
Multiplication : exemple 1



Multiplication : exemple 2



Multiplication : exemple 2



Cholesky factorization

Recall : recursive block splitting based on the 2×2 block structure.

$$\begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \times \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{bmatrix}$$

$$A_{11} = L_{11} L_{11}^T$$

$$A_{21} = L_{21} L_{11}^T$$

$$A_{22} = L_{21} L_{21}^T + L_{22} L_{22}^T$$

Cholesky factorization

Recall : recursive block splitting based on the 2×2 block structure.

$$\begin{array}{|c|c|} \hline A_{11} & A_{21}^T \\ \hline A_{21} & A_{22} \\ \hline \end{array} = \begin{array}{|c|c|} \hline L_{11} & 0 \\ \hline L_{21} & L_{22} \\ \hline \end{array} \times \begin{array}{|c|c|} \hline L_{11}^T & L_{21}^T \\ \hline 0 & L_{22}^T \\ \hline \end{array}$$

$$A_{11} = L_{11} L_{11}^T \quad (\text{recursive call})$$

$$A_{21} = L_{21} L_{11}^T$$

$$A_{22} = L_{21} L_{21}^T + L_{22} L_{22}^T$$

Cholesky factorization

Recall : recursive block splitting based on the 2×2 block structure.

$$\begin{array}{|c|c|} \hline A_{11} & A_{21}^T \\ \hline \textcolor{red}{A_{21}} & A_{22} \\ \hline \end{array} = \begin{array}{|c|c|} \hline L_{11} & 0 \\ \hline \textcolor{green}{L_{21}} & L_{22} \\ \hline \end{array} \times \begin{array}{|c|c|} \hline \textcolor{blue}{L_{11}^T} & L_{21}^T \\ \hline 0 & L_{22}^T \\ \hline \end{array}$$

$$\begin{aligned}
 A_{11} &= L_{11} L_{11}^T \\
 \textcolor{red}{A_{21}} &= \textcolor{green}{L_{21}} \textcolor{blue}{L_{11}^T} \quad (\text{upper triangular solve}) \\
 A_{22} &= L_{21} L_{21}^T + L_{22} L_{22}^T
 \end{aligned}$$

Cholesky factorization

Recall : recursive block splitting based on the 2×2 block structure.

$$\begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \times \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{bmatrix}$$

$$A_{11} = L_{11} L_{11}^T$$

$$A_{21} = L_{21} L_{11}^T$$

$$A_{22} = L_{21} L_{21}^T + L_{22} L_{22}^T \quad (\mathcal{H}\text{-matrix GEMM then recursive call})$$

Computation details

Typical issues

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

- Very small operations ;
- Oddly-shaped matrices : "Tall & skinny" ;
- High memory band.

Observations

- Most (70-80%) of the time spent in :
 - QR decompositions of T&S matrices ;
 - SVD decompositions of small matrices.
- BLAS implementations cannot reach the peak performance ;
- Very high memory bandwidth.

Complexity estimates

Useful pointers

- Most complexity estimates assume a fixed upper bound k for the low-rank matrices involved ;
- The structure of the matrix (as represented by a tree) is important as well : a large depth with small blocks is typically a bad omen.

Common operations

For a matrix size of $N \times N$ with the previous assumptions :

- assembly : time and storage is in $\mathcal{O}(kN \log N)$;
- addition : $\mathcal{O}(k^2 N \log N)$ operations ;
- multiplication and Cholesky factorisation : $\mathcal{O}(k^3 N \log^3 N)$ operations ;
- in practice a $\mathcal{O}(N \log^2 N)$ complexity is observed.

Context

\mathcal{H} -matrices

Applications

Conclusions & perspectives

Recall the 1D toy model ?

- exponential kernel : $K(s, t) = e^{-|s-t|/\lambda}$
- prescribed relative error $\varepsilon = 10^{-4}$;
- **HODLR** admissibility ;
- 1D exponential kernel : provably rank-one when **HODLR**.
- 'exact model' is the second Hermite function
 $\psi_2(x) = (2x^2 - 1)e^{-\frac{1}{2}x}$;
- input data as a gaussian distribution ;
- here : one iteration of optimization loop, correlation length $\lambda = 0.01$.

Recall the 1D toy model ?

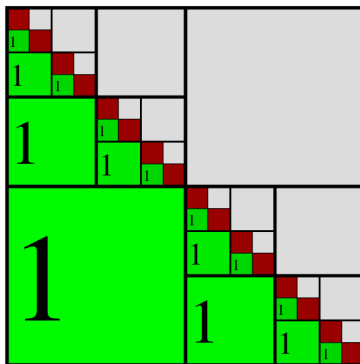


FIGURE : Lower part of the covariance matrix : rank map.

- matrix size 1000×1000 ;
- compression ratio : $\approx 12\%$ (small case !)

Recall the 1D toy model ?

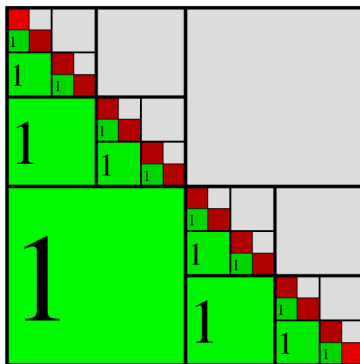


FIGURE : Cholesky factor : rank map.

- matrix size 1000×1000 ;
- compression ratio : $\approx 12\%$ (small case !)

Recall the 1D toy model ?

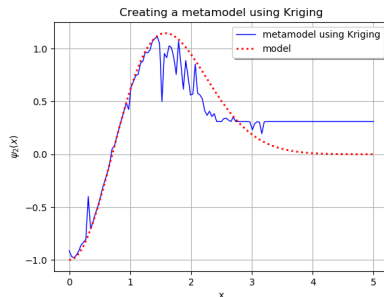


FIGURE : Surface response using kriging

- compression ratio : $\approx 12\%$ (small case !);
- same response with LAPACK and HMAT solver;
- maximum absolute error between two approximates : 1.55×10^{-15} .

3D exemple : a cantilever beam

- The vertical deflection y of a cantilever beam's free end of fixed length L reads as

$$y = \frac{FL^3}{3EI},$$

where :

- E is the Young modulus ;
- F is the load ;
- I is the moment of inertia.
- Input* variables $x = (F, E, I)$ are assumed random ;
- Variable of interest (*output*) is the deflection y estimated thanks to the model \mathcal{M}

$$\mathcal{M} : x \mapsto y$$

- Building a metamodel $\tilde{\mathcal{M}}$ through Kriging and optimization loop for parameters : one \mathcal{H} -matrix at each iteration to treat the covariance matrix associated with a specified covariance model.

The chosen covariance model $K_\lambda(s, t)$ is the following tensor product :

$$e^{-(s_1-t_1)/\lambda_1} e^{-(s_2-t_2)/\lambda_2} e^{-(s_3-t_3)/\lambda_3}.$$

- Let $\lambda_1 = 3.96528$, $\lambda_2 = 5.8237$ and $\lambda_3 = 9.0679$ be the starting coefficients for the optimization.
- Degrees of freedom to be clustered within the \mathcal{H} -matrix framework :

"Young modulus"; "Load"; "Inertia"

3.6258375026+07; 4.797336026+04; 3.5171332517+02

3.1382130227+07; 3.350317520+04; 3.5324327901+02

... ..

Results

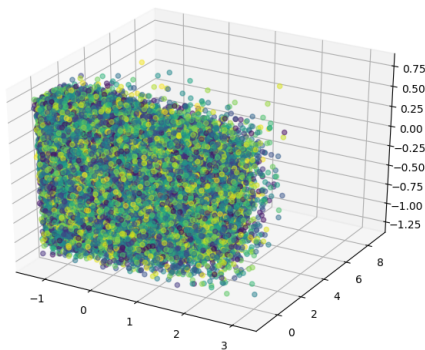


FIGURE : Input data : 5×10^4 entries.

- covariance matrix of size $5.10^4 \times 5.10^4$, symmetric and double precision : full size in memory is 10Go.

Results

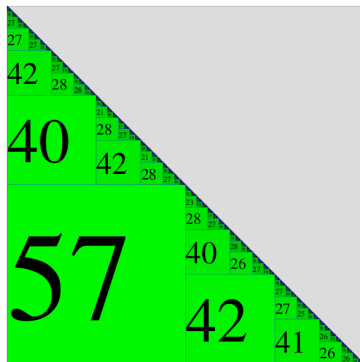


FIGURE : Lower part of the covariance matrix : rank map.

- memory : 167Mo (compression ratio : 1.67%);
- assembly time : 11.83s

Results

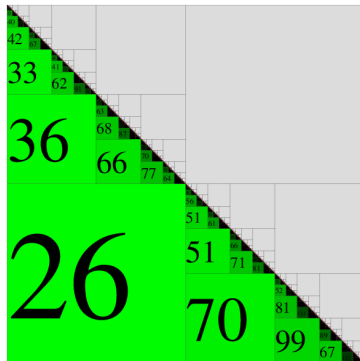


FIGURE : Cholesky factor : rank map.

- memory : 263Mo (compression ratio : 2.63%);
- Cholesky time : 17.84s

Context

\mathcal{H} -matrices

Applications

Conclusions & perspectives

Summing up the \mathcal{H} -matrices method

- Three key components for assembling a \mathcal{H} -matrix :
 - The clustering of degrees of freedom (e.g. geometric) ;
 - An admissibility condition = which block to compress ;
 - A fast on-the-fly algorithm to assembly low-rank admissible blocks.
- An algebra on \mathcal{H} -matrices :
 - Multiplications and additions of \mathcal{H} -matrices ;
 - Fast factorization of an \mathcal{H} -matrix with the same structure :
 - Fast direct solver ;
 - Good preconditioner for iterative solver.

Conclusion

- The \mathcal{H} -matrix framework is an enabler for many statistics problems ;
- Sequential solver is freely available through OpenTurns ;
- Efficient parallel solver through licensing (contact Airbus & Imacs).