Introduction

This document aims at presenting the work achieved for project ElastoΦ during the CEMRACS summer school that took place at the Centre de Rencontres Mathématiques (CIRM) at Luminy from July the 24th to August the 25th, 2016. This project originated from mathematical challenges encountered by IFP Energies Nouvelles (IFPEN) for the numerical solution of an elastostatic problem in an infinite homogeneous background medium containing crack networks with highly complex geometrical structure, see the pictures below.

This problem was reformulated as boundary integral equation posed at the surface of cracks, and discretized by means of Galerkin procedure based on piecewise constant functions, which is commonly known as Boundary Element Method (BEM). The fully non-local structure of boundary integral operators leads to densly populated matrices. As a consequence if the matrix of the problem is of size N, then any matrix-vector product (which is the most elementary building block of any iterative linear solver) requires at least $\mathcal{O}(N^2)$ operations.

As in most industrial applications, for the problems considered by IFPEN this situation is not acceptable: computational complexity of N^2 makes it impossible to perform a matrix-vector product for N larger than say 10^6: this would be too costly in terms of time and memory storage. At the same time, the applications considered by IFPEN typically require N to be of this order.

To deal with this difficulty, IFPEN developped a heuristic method consisting in forcing coefficients of the BEM matrix to zero whenever this coefficient corresponds to the interaction between sufficiently distant points of the crack network. With this procedure, the matrix of the problem is approximated by a sparsified matrix that allows matrix-vector products with $\mathcal{O}(N)$ complexity. But this strategy also induce substantial consistency error: measured in Frobenius norm, the perturbation on the matrix is typically 40% large.

On the other hand, current literature on boundary integral equation nowadays offers a panel of refined complexity reduction techniques: fast multipole methods, hierarchical matrix strategies, and the like. These techniques, that have been developped during the last two decades, have been introduced to accelerate computations in a wide variety of problems ranging from molecular dynamics [**?**], to astrophysics [**?**]. Acceleration of boundary integral equations on smooth surfaces has also been historically an key challenge for stimulating the development of such methods [**?**].

The main goal of the CEMRACS project ElastoΦ was to test the performance of one of these reduction techniques on the crack network problem of IFPEN. Although the above mentionned complexity reduction techniques are particularly well suited for boundary integral equations, the problem under consideration from IFPEN was not just a straightforward application of the state of the art, because it involves a strongly irregular geometry. Besides development issues, this was the main challenging aspect of the project.

The ElastoΦ project mainly conisisted in developping a numerical mockup code in C++ and testing it on the matrices sent by IFPEN. We implemented one particular complexity reduction technique: Hierarchical Matrix [**?**] format combined with the Adaptative Cross Approximation (ACA) compression method [**?**]. For the sake of brevity, we shall refer to this approach as HM-ACA. We could not test several complexity reduction techniques, this would have required more time (CEMRACS is only 5 weeks long...). Among all complexity reduction techniques already available (multipoles, panel clustering, etc...), we chose HM-ACA because this was the only approach that treats generation of the matrix of the problem in a black-box manner. All other methods are far more intrusive, and most of the time their developement is also more time consuming. On the other hand, due to confidentiality restrictions, we did not have access to the source code of IFPEN, and in particular could not actually get the routine generating the coefficients of the matrix. These constraints led us to choose HM-ACA.

There already exists freely accessible libraries written in C or C++ that implement HM-ACA, see e.g. HLib (`http://hlib.org/`), H2Lib (`http://www.h2lib.org/`) or Ahmed (`https://github.com/xantares/ahmed`), but they are poorly documented. Besides IFPEN required that the code numerical mockup developped during the project can be reused by IFPEN after the CEMRACS summerschool, and the licence of all existing libraries was incompatible with this condition. For these reasons, we decided to redevelopped simply an "home made" implementation of HM-ACA. The code that we developped was put under Lesser Gnu Public Licence (LGPL) and put on a Github repository freely accessible at

`https://github.com/xclaeys/ElastoPhi`

The outline of this report is as follows. We will first describe the Adptative Cross Approximation method, and show its efficiency for the compression of dense matrices admitting fastly dcreasing singular values (such matrices shall be refered to in the sequel as admissible). Unfortunately the matrices generated by boundary element methods (and in particular the matrices considered by IFPEN) do not have fastly decreasing eigenvalues. We shall then comment on this point, describing the problem and the boundary integral equation considered by IFPEN. Since the ACA compression method is not directly applicable, we are led to decompose the matrix in sub-blocks, each of which is either small or admissible. To be efficient, such a decomposition has to follow certain rules, and in particular admit a hirerachical structure. Such technique, that we shall then describe, is referred to as a "hierarchical matrix format". Then we shall give an detailed overview of the code developped during the project, and conclude the report by a series of test cases where we report the performances of our code.

1 Low rank approximations

In this section we are interested in fully populated matrices $\mathrm{A} \in \mathbb{C}^{n\times n}$, $\mathrm{A} = (\mathrm{A}_{j,k})_{j,k=1\dots n}$ with, a priori, none of its entries vanishing, its size n being potentially large. With no particular assumption on this matrix, the cost of a matrix-vector product is $\mathcal{O}(n^2)$.

This cost is substantially reduced if we assume that A is of low rank though. We say that a matrix has the low rank property, with rank $k \leq n$, if there exist vectors $\mathbf{u}_j, \mathbf{v}_j \in \mathbb{C}^n, j = 1\dots k$, such that

$$\mathrm{A} = \sum_{j=1}^{k} \mathbf{u}_j \cdot \mathbf{v}_j^T \quad \text{with} \quad k \leq n/2.$$

Indeed if this representation holds, then a matrix-vector product requires $2kn$ flops, which is smaller than n^2 provided that the condition on k given above is satisfied. Matrices A encountered in applications rarely have the loaw rank property. A simple primary observation is that general matrices A can as a sum rank one matrices through its singular value decomposition

$$\mathrm{A} = \sum_{j=1}^{n} \sigma_j \, \mathbf{u}_j \cdot \mathbf{v}_j^T \quad \text{where} \;\; \mathfrak{S}(\mathrm{A}^*\mathrm{A}) = \{\sigma_j^2\}_{j=1\dots n} \tag{1}$$

where $(\mathbf{u}_j)_{j=1}^n, (\mathbf{v}_j)_{j=1}^n$ are orthonormal basis of $\mathbb{C}^n$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$. A closer inspection of this formula leads to the conclusion, provided that the sequence (σ_j) decreases fast, then truncating the singular value decomposition (1) yields a good approximation of the matrix A. This is the essence of the next result.

Proposition 1.1.
Let $\mathrm{A} \in \mathbb{C}^{n\times n}$ admit the singular value decomposition (1), and denote A_k the matrix obtained by truncating this decomposition at rank k namely $\mathrm{A}_k := \sum_{j=1}^{k} \sigma_j \, \mathbf{u}_j \cdot \mathbf{v}_j^T$. Then we have the error estimates

$$\|\mathrm{A} - \mathrm{A}_k\|_2^2 = \sigma_{k+1}^2 \quad \textit{and} \quad \|\mathrm{A} - \mathrm{A}_k\|_{\mathrm{F}}^2 = \sum_{j=k+1}^{n} \sigma_j^2$$

where $\|\ \|_2$ refers to refers to the matrix norm induced by the vector norm $|\mathbf{u}|_2 = (\sum_{j=1}^n |u_j|^2)$ for $\mathbf{u} = (u_j)_{j=1}^n \in \mathbb{C}$ and $\|\ \|_{\mathrm{F}}$ refers to the Fröbenius norm.

Truncating the SVD is thus an efficient way to approximate a matrix, and thus to reduce the cost of the matrix-vector product, provided that the singular values decrease fast. Assume that singular values decrease exponentially, say $\sigma_k \leq q^{-k}$ for a fixed $q \in (0,1)$. Then for a relative error expressed in Fröbenius norm of order $\epsilon > 0$, it suffices to take $k \simeq |\ln \epsilon|$.

Unfortunately, computing the singular value decomposition of a matrix is expensive: it costs $\mathcal{O}(n^3)$ flops. To circumvent this issue, starting from an arbitrary matrix A whose singular values are assumed to decrease exponentially, the Adaptative Cross Approximation algorithm provides a collection of vectors $\mathbf{u}_j, \mathbf{v}_j \in \mathbb{C}^n, j = 1\dots n$ such that the matrices $\widetilde{\mathrm{A}}_k = \sum_{j=1}^{k} \mathbf{u}_j \cdot \mathbf{v}_j^T$ satisfy

$$\|\mathrm{A} - \widetilde{\mathrm{A}}_k\|_{\mathrm{F}} \leq C \|\mathrm{A} - \mathrm{A}_k\|_{\mathrm{F}}$$

for some constant $C > 0$ independent of k. Moreover, which is probably the most interesting feature of this method, the cost of computing $\widetilde{\mathrm{A}}_k$ is $\mathcal{O}(kn)$. This cost is thus quasi-linear

provided that the singular values decrease exponentially. The detailed analysis of the ACA algorithm is out of the scope of this report. We only give the algorithm it self.

Algorithm 1 Partially Pivoted ACA

Initialise j_*
$r = 0$
while(stopping criterion not satisfied){

$\mathbf{w} = \mathrm{A}(j_*, :)^T - \sum_{\ell=1}^{r} \mathbf{u}_\ell(j_*)\, \mathbf{v}_\ell$
$k_* = \mathrm{argmax}_{k=1\ldots n} |\mathbf{w}(k)|$
$w_* = \mathbf{w}(k_*)$

if$(w_* \neq 0)$**{**

$\mathbf{v}_{r+1} = \mathbf{w}$
$\mathbf{w} = \mathrm{A}(:, k_*) - \sum_{\ell=1}^{r} \mathbf{v}_\ell(k_*)\, \mathbf{u}_\ell$
$\mathbf{u}_{r+1} = w_*^{-1}\mathbf{w}$
$j_* = \mathrm{argmax}_{j=1\ldots n} |\mathbf{w}(j)|$
$r = r + 1$

}

else{terminate algorithm}
}

In our pseudo-code notation, for any vector $\mathbf{w} \in \mathbb{C}^n$, the number $\mathbf{w}(j)$ refers to the jth entry of $\mathbf{w}$. Likewise for $\mathrm{A} \in \mathbb{C}^{n\times n}$, the number $\mathrm{A}(:, k)$ refers to the kth column, and $\mathrm{A}(j, :)$ refers to the jth row.

At the beginning of Algorithm 1, there is an initialisation step for the choice of the index of the first j_*. For this initialisation, one could take $j_* = 1$. Other choices may speed up the convergence of the algorithm. Such heuristics are problem dependent. Algorithm 1 also involves a stopping criterion based on an error estimator. During the ElastoΦ project, we took

Stopping criterion:
$|\mathbf{u}_r|_2 |\mathbf{v}_r|_2 \leq \epsilon \| \sum_{\ell=1}^{r} \mathbf{u}_\ell \cdot \mathbf{v}_\ell^T \|_{\mathrm{F}}$

where $\| \sum_{\ell=1}^{r} \mathbf{u}_\ell \cdot \mathbf{v}_\ell^T \|_{\mathrm{F}}^2 = \sum_{j=1}^{r} \sum_{k=1}^{r} (\overline{\mathbf{u}}_j^T \mathbf{u}_k)(\overline{\mathbf{v}}_k^T \mathbf{v}_j)$

2 The matrix under consideration

We now shall give some more detail about the matrices under consideration during the ElastoΦ project. As previously mentionned, these matrices stemmed from a Galerkin discretization of a boundary integral equation. Assume that Γ is a surface embedded in $\mathbb{R}^3$ that is the union of a collection $\mathcal{T} = \{\tau_j\}_{j=1}^N$ of flat triangles and quadrangles

$$\Gamma = \bigcup_{j=1}^{N} \tau_j \,, \qquad \tau_j = \text{triangle or quadrangle.}$$

The elements τ_j might very intersect each other, which makes Γ a potentially very rough surface. Let us briefly describe the bilinear form associated to the variational formulation under consideration for the Galerkinn discretisation. We need to introduce the Green kernel of the problem that is given by the explicit formula

$$\mathscr{G}(\boldsymbol{x}) = \frac{1}{8\pi E}\frac{1+\rho}{1-\rho}\Big(\frac{3-4\rho}{|\boldsymbol{x}|}\mathrm{Id} + \frac{\boldsymbol{x}\cdot\boldsymbol{x}^T}{|\boldsymbol{x}|^3}\Big)$$

It is fundamental to observe, and to keep in mind, that this function is singular at $\boldsymbol{x}=0$. In this expression, ρ refers to the Poisson ratio, and E is the elasticity module. Next, for each element τ, let $\boldsymbol{n}_\tau$ refer to a vector field normal at τ , and for any vector field $\boldsymbol{u}:\Gamma\to\mathbb{R}^3$, define the trace operator

$$\mathscr{R}_\tau(\boldsymbol{u}) = \lambda \boldsymbol{n}_\tau \mathrm{div}(\boldsymbol{u}) + 2\mu\,(\boldsymbol{n}_\tau\cdot\nabla)\boldsymbol{u} + \mu \boldsymbol{n}_\tau\times\mathbf{curl}(\boldsymbol{u})$$

$$\text{with}\quad \lambda := \frac{E\rho}{(1+\mu)(1-2\rho)},\quad \mu := \frac{E}{2(1+\rho)}$$

The variationnal formulation associated to the problem under consideration typically consists in finding $\boldsymbol{u}\in \mathrm{H}_{00}^{1/2}(\Gamma)^3$ such that $a(\boldsymbol{u},\boldsymbol{v}) = f(\boldsymbol{v})$ for all $\boldsymbol{v}\in\mathrm{H}_{00}^{1/2}(\Gamma)^3$, where $\mathrm{H}_{00}^{1/2}(\Gamma) := \Pi_{\tau\in\mathcal{T}}\mathrm{H}_{00}^{1/2}(\tau)^3$ and the bilinear form is given by

$$a(\boldsymbol{u},\boldsymbol{v}) = \sum_{\tau\in\mathcal{T}}\sum_{\tau'}\int_{\tau\times\tau'}\mathscr{R}_\tau^{\boldsymbol{y}}(\mathscr{R}_{\tau'}^{\boldsymbol{x}}\mathscr{G}(\boldsymbol{x}-\boldsymbol{y}))\boldsymbol{u}(\boldsymbol{x})\boldsymbol{v}(\boldsymbol{y})d\sigma(\boldsymbol{x})d\sigma(\boldsymbol{y}). \tag{2}$$

In this formula the operator $\mathscr{R}_\tau^{\boldsymbol{x}}$ is the operator $\mathscr{R}_\tau$ applied with respect the $\boldsymbol{x}$ variable. The operator $\mathscr{R}_{\tau'}^{\boldsymbol{y}}$ is defined accordingly. Formula (2) is rather abstract. Another, more explicit expression of this operator can be obtained, see e.g. [**?**], but this is pointless for the present report. The only important thing here is that the integral kernel coming into play in this integral operator is singular at $\boldsymbol{x}=\boldsymbol{y}$ (which is possible only if $\tau\cap\tau'\neq\emptyset$), and it is regular otherwise. In particular, if τ and τ' are distant from each other, then the operator associated to $a(\ ,\ )$ is regularising, and it will induce matrices with exponentially decreasing singular values. Now let us describe how the bilinear form (2) is discretised. Each space $\mathrm{H}_{00}^{1/2}(\tau)^3$ is approximated by a three dimensional space

$$\mathrm{H}_h(\tau)\simeq\mathrm{H}_{00}^{1/2}(\tau)^3\quad \dim(\mathrm{H}_h(\tau)) = 3\quad \text{for each}\ \ \tau\in\mathcal{T}.$$

Next the global variational space is obtained by a simple cartesian product $\mathrm{H}_h(\Gamma) = \Pi_{\tau\in\mathcal{T}}\mathrm{H}_h(\tau)$. Finally the discrete variational formulation writes: find $\boldsymbol{u}_h\in\mathrm{H}_h(\Gamma)$ such that

$$a(\boldsymbol{u}_h,\boldsymbol{v}_h) = f(\boldsymbol{v}_h)\quad \boldsymbol{v}_h\in\mathrm{H}_h(\Gamma) \tag{3}$$

Following the classical Galerkin discretisation procedure, a matrix is derived from (3) simply by specifying a basis for $\mathrm{H}_h(\Gamma)$. In the present case, such a basis is specified as follows. Denote $\boldsymbol{\psi}_{3j-2}(\boldsymbol{x}),\boldsymbol{\psi}_{3j-1}(\boldsymbol{x}),\boldsymbol{\psi}_{3j}(\boldsymbol{x})$ three vector fields generating $\mathrm{H}_h(\tau_j)$ (recall that $\dim\mathrm{H}_h(\tau_j)=3$ by hypothesis). Then we have

$$\mathrm{H}_h(\Gamma) = \operatorname*{span}_{j=1\ldots3N}\{\boldsymbol{\psi}_j\}.$$

Of course, here, each $\boldsymbol{\psi}_{3j-q}$ with $q=0,1,2$ is regarded as a function defined on Γ that is supported only on τ_j. Then the matrices that we had to deal with during Project ElatsoΦ are defined as $\mathrm{A} = (\mathrm{A}_{j,k})_{j,k=1\ldots3N}$ where

$$\mathrm{A}_{j,k} := a(\boldsymbol{\psi}_j,\boldsymbol{\psi}_k),\quad j,k = 1\ldots3N.$$