SONATE: a Parallel Code for Acoustics

Toufic Abboud [1] Maxime Pallud [1] Claude Teissedre [2]

Abstract

Acoustic simulations are very demanding in computational resources. Time Domain Boundary Element Method (TD-BEM) is a new efficient formalism for large frequency bandwidth problems. It proves to be very well adapted to parallel computing on both SMP and distributed architectures. In this paper, we present the numerical method and the parallelization methodology. We finally show some preliminary results on HP J6000 Computing Farms and HP RP8400 SMP computer. Porting and optimization on Intel Itanium 2 (McKinley) based HP-Cluster is on-going and bigger industrial test cases will be presented at the conference.

1 Introduction

Noise is becoming a more and more sensitive parameter in nowadays automotive, aeronautics and aerospace industries. Frequency Domain Boundary Element Method (FD-BEM) is the reference numerical method for low and mid-range frequency simulations. This method only requires a mesh of the boundary of the fluid domain, with a mesh step h that resolve the wavelength λ. Generally, a value $h \leq \lambda/5$ leads to good results for most applications. If we recall that the wavelength is related to the frequency f by the relation: $\lambda = c/f$, where c is the sound speed, we find that the number of spatial unknowns N_s grows like $O(f^2)$. For each frequency, this method leads to solve a dense symmetric complexe linear system, and thus memory requirements grow as $O(f^4)$ and CPU time as $O(f^6)$ if a direct method is used. In general, this has to be done for $N_f = O(f)$ different frequencies in the intervall $(0, f)$. So, if we want to double the frequency value, memory is mutiplied by 16 and CPU time by 32 for one single frequency and by 64 if all the frequency band is required! In automotive applications, acoustic design engineers rarely use meshes with more than 10,000 unknows. This hardly allows to simulate a full car for frequencies larger than a few hundred Hz, on a modern mono-processor computer. Industrial needs go up to 3000 Hz which involves several hundreds of thousand of unknowns! In aeronautics, simulation of the acoustic radiation of nacelles requires also several hundreds of unknowns! That gives an idea of the big need for new efficient methods and parallel codes.

In this paper, we first present the Time Domain Boundary Element Method (TD-BEM) a new efficient method for acoustic simulation. Its main advantage compared to classical (FD-BEM) is that one single computation followed by a Fast Fourier Transform gives the acoustic response in a large frequency band. The memory requirement is almost the same but computational complexity is much less for broadband responses: $O(N_s^2 \times N_t) = O(f^5)$, where N_t is the number of time steps, $N_t = O(f)$, to compare with $O(N_s^3 \times N_f) = O(f^7)$ for FD-BEM, as $N_f = O(f)$ in this case. Next, we show how we have parallelized the software SONATE that implements TD-BEM. We choose a message passing model based on MPI which allows SONATE to run on SMP architectures as well as distributed ones. We show some preliminary results obtained on a HP J6000 Computing Farm and on a HP RP8400. A good scalability is obtained on both platforms with better performances on the Computing Farm. Porting and optimization on IA-64 architecture is in progress and we plan to perform bigger industrial cases by the Conference.

[1]IMACS, X-TEC, École Polytechnique, 91128 Palaiseau cedex, France
[2]HP France, ZI de Courtaboeuf 2, 1, Avenue du Canada, 91947 Les Ulis Cedex, France

2 Time Domain BEM

A function of the time variable t is said to be causal if it is zero for negative t. We consider a bounded regular domain $\mathcal{I}$ in $\mathbb{R}^3$, it is called the interior domain. $\mathcal{E} = \mathbb{R}^3 \setminus \overline{\mathcal{I}}$ is called the exterior domain. Their common boundary is denoted by $\mathcal{B}$.

The goal is to find causal solutions of the (normalized) acoustic system

$$(1) \qquad \left\{ \begin{array}{rcl} \dfrac{\partial p}{\partial t}(t,x) + \operatorname{div} \vec{v}(t,x) & = & 0 \\ \dfrac{\partial \vec{v}}{\partial t}(t,x) + \vec{\nabla} p(t,x) & = & 0 \end{array} \right. \qquad (t,x) \in \mathbb{R} \times \mathcal{I} \cup \mathcal{E}$$

with some boundary conditions. Boundary Element Methods are based on the existence of an integral representation formula which gives the solution anywhere if we know some quantity on the boundary. In this case, the representation formula is called the *retarded potentials* formula:

$$(2) \qquad \begin{array}{rl} p(t,x) = & \displaystyle\int_{\mathcal{B}} \frac{1}{4\pi r} \dot{v}_n(t-r,y) d\mathcal{B}_y + \\ & \displaystyle\int_{\mathcal{B}} \frac{(\vec{x}-\vec{y})\cdot\vec{n}_y}{4\pi r^3} \mu(t-r,y) d\mathcal{B}_y + \int_{\mathcal{B}} \frac{(\vec{x}-\vec{y})\cdot\vec{n}_y}{4\pi r^2} \dot{\mu}(t-r,y) d\mathcal{B}_y \end{array}$$

for $(t,x) \in \mathbb{R} \times \mathcal{I} \cup \mathcal{E}$, with $r = |x-y|$, $v_n = [\vec{v}\cdot\vec{n}] = (\vec{v}\cdot\vec{n})^- - (\vec{v}\cdot\vec{n})^+$, $\mu = [p] = p^- - p^+$, the dot denotes the time derivative. We have an equivalent formula for $\vec{v}$. Imposing the boundary conditions, we obtain the so called integral equation. The problem is then written in a variational form that can be discretized by a Time-Space Finite Element Method. In order to obtain a stable numerical scheme, we have satisfied some energy conservation in the discrete formulation. Whatever the boundary conditions are, the discretization always implies a convolution system:

$$(3) \qquad \sum_{k=0}^{K} M^k q^{n-k} = b^n \qquad n = 0, 1, \cdots, N_t$$

where $M^k ij$ are linear combination of terms of the form

$$\int_{T_i} \int_{T_j} \frac{1}{4\pi R} \, dT_i \, dT_j$$
$$t_k \leq R \leq t_{k+1}$$

M^k is sparse since only elements which are of order $k\Delta t$ distant are interacting (*cf.* fig1). Total number of non-zero elements in all matrices is $O(N_s^2)$. q^{n-k} is the vector of degrees of freedom at time $(n-k)\Delta t$ and b^n the right-hand side, *i.e.* the excitation, at time $n\Delta t$.

The resolution procedure is as follows:

- Equation (3) also writes

$$(4) \qquad M^0 q^n + \sum_{k=1}^{K} M^k q^{n-k} = b^n$$

  $q^{n-k} = 0$ for $n-k < 0$ because of the causality condition!

- From (4), execute the recursive algorithm
  step 1: convolution

$$(5) \qquad \tilde{b}^n = b^n - \sum_{k=1}^{K} M^k q^{n-k}$$

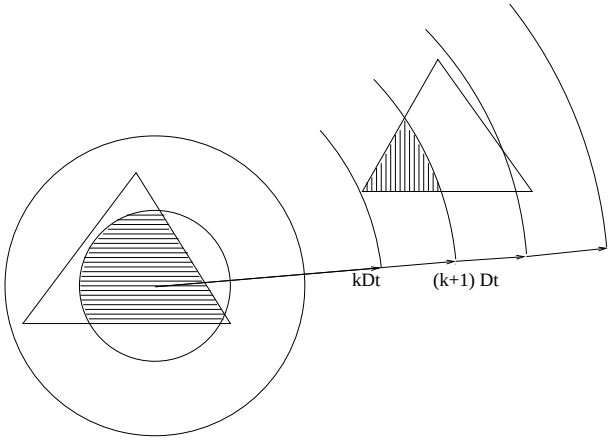


FIG. 1. *Elementary Interactions.*

where at each step q^n is updated from
step 2: linear system

$$M^0 q^n = \tilde{b}^n \tag{6}$$

with a matrix M^0 which is symmetric and positive definite.
Steps 1 and 2 are then repeated until $n = N_t$.

- **Post-processing:** Extract the field using the integral representation formula and perform the FFT to obtain the frequency domain fields.

3 Parallelism model

As frequency grows, SONATE becomes more and more demanding in memory space and CPU cycles. The two important phases to optimize are the solving of the convolution system (3) and the post-processing phase which consist in using the integral representation formula (2) in order to obtain the pressure and the velocity and deduce other quantities as the radiated power. The generic term appearing in the post-processing phase is of the form:

$$\int_{\mathcal{B}} \frac{q(t-|x-y|,y)}{4\pi|x-y|} d\mathcal{B}(y) = \sum_{L=1}^{NTRI} \int_{\mathcal{B}_L} \frac{q(t-|x-y|,y)}{4\pi|x-y|} d\mathcal{B}_L(y) \tag{7}$$

where $\mathcal{B}_L$ is triangle number L, and $NTRI$ the number of triangles in the mesh. Post-processing can be easily parallelized by distributing the triangles equally on all nodes.

These two phases can be naturally parallelized using message passing model as MPI. The OPEN MP parallel programming model may also be used but is limited to SMP architectures, moreover, we believe that it would be harder to reach the same parallel efficiency. In the sequel, we will only discuss the parallelization of the convolution.

4 Solving the convolution system

The convolution system (3) is solved step by step: it is a marching-on-in-time algorithm. At each time step, solving the linear system (6) can be done either by using a conjugate gradient algorithm or an efficient direct sparse solver (HP `mlib`) which is better in the case of ill-conditioned matrices occuring with very irregular meshes. With both methods, this is an insignificant step compared to the convolution. This may not be true for small meshes and large number of processors, but parallel computing with SONATE is meant for big cases, *i.e.* more than 10,000 unknowns. Its complexity grows almost linearly whereas (5) is quadratic. So, at the moment, this step is left sequential.

The principle of the parallelization of (5) is simple and purely algebraic: it consists in achieving the same computational tasks on subsets of data distributed on all the nodes. More precisely:

- each matrix M^k is splitted round-robin by row into `Nproc` sparse matrices $N_s \times N_s$ having approximately the same number of non-zero elements:

$$M^k = M_0^k + \cdots + M_{\texttt{Nproc}-1}^k$$

- matrices $(M_i^k)_{k=0,\cdots,K}$ are respectively stored on node i's local disk. During computation, processor i accesses matrices $(M_i^k)_{k=0,\cdots,K}$
- at time step n, processor i computes :

$$\tilde{b}_i^n = b_i^n - \sum_{k=1}^{K} M_i^k q^{n-k} \tag{8}$$

As matrices $(M_i^k)_{i=0,\cdots,\texttt{Nproc}}$ have almost the same number of non-zero elements, load balancing is fair and the local complexity is $O(N_s^2/\texttt{Nproc})$.

Once $\tilde{b}_i^n$ is computed, it is sent to the master process that will perform the reduction operation:

$$\tilde{b}^n = \sum_{i=0}^{\texttt{Nproc}} \tilde{b}_i^n \tag{9}$$

- the master process solves the linear system (6) then broadcasts the resulting vector q^n to all the slaves.

This paradigm of parallelization of TD-BEM does not need a reformulation of the equations and always gives the same numerical result for any number of processors. This is to be compared with Domain Decomposition used to parallelize Finite Element codes.

5 Sizing

The characteristic numbers governing memory, computational and communications needs are:

- memory/node: $O(N_s^2/\texttt{Nproc})$
- CPU/iter: $O(N_s^2/\texttt{Nproc})$ for convolution (8)
- Communication/iter: $O(2 \times N_s \times \texttt{Nproc})$

In order to balance CPU and communications, `Nproc` has to grow as $O(N_s^{1/2})$. If so, memory, CPU and communications grow as $O(N_s^{3/2})$. If CPU/iter was privileged, `Nproc` may grow up to $O(N_s)$, but it is not reasonable in practice for big cases.

The growth of memory per node $O(N_s^{3/2})$ can be satisfied by out-of-core implementation. SONATE implements the out-of-core in the same way for sequential and parallel treatment and uses asynchronous I/O. Further, the algorithm has been optimized in order to minimize the number of read operations by precomputing as much as possible matrix-vector products in the convolution process.

Taking into account the above rules, and the preliminary tests, we expect that a case with $N_s = 100{,}000$ will be computed in about 6h on a 48 processors cluster, each node having 4GB of memory.

6 Performances on HP J6000 Computing Farm

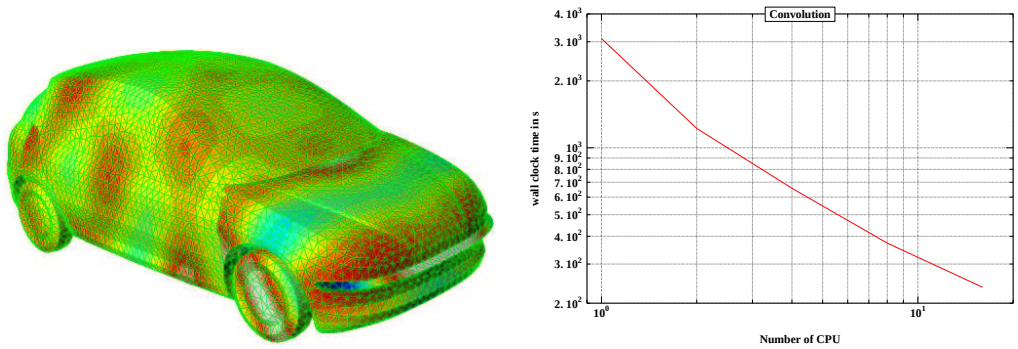
We have tested a 16 nodes / 2 PA-RISC 8600 - 550MHz per node / 4GB per node farm interconnected by a HP Hyperfabric network based on Myrinet 1 protocol. All computations have been achieved using 1 CPU per node.

We have used a Peugeot 206 car mesh shown on fig. 2 (a), with 18264 triangles and 9355 vertices, provided with the courtesy of PSA. This mesh allows accurate simulations up to 500Hz. In this case, with CFL=0.5, aggregate matrices size is 1.8GB.

Figure 2 (b) shows in-core wall clock time for 2, 4, 8 and 16 nodes and out-of-core for one node (3100s). The in-core one node time (3988s) has been penalized by the big size of this case for a single node. On 16 CPU, we solve acoustic problem from 0 up to 500Hz (several hundreds of frequencies) in less than 4mn. An optimized parallel FD-BEM code would solve one single frequency in 2mn.

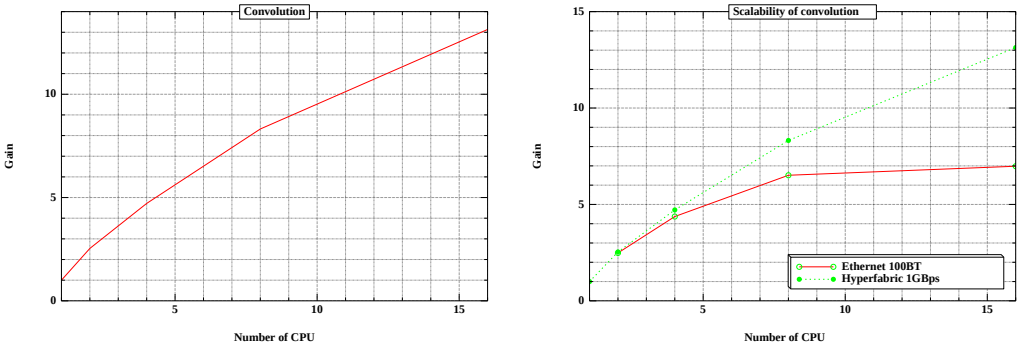
Figure 3 (a) shows superlinear speed-up up to 8 CPU. The speed up was computed by taking the out-of-core wall clock time as reference.

Figure 3 (b) compares Hyperfabric and Ethernet 100BT performances. For this (relatively small) case, a high performance network is not too critical up to 8 CPUs.



(a) Mesh (18264 triangles). (b) Wall clock time.

FIG. 2. *Scalability of* SONATE *on Peugeot 206 case (PSA).*



(a) Speed up (b) Hyperfabric vs Ethernet 100BT.

FIG. 3. *Scalability of* SONATE *on the Peugeot 206 case.*

7 Performances on HP RP8400

In this section, we show results on two cases: First, we consider a refined Peugeot 206 mesh shown in fig. 4: 73056 triangles and 37056 vertices. We have run on an HP RP8400 / 16 CPU PA-8700 700 MHz / 16GB of memory. Resolution wall clock time of the acoustic response in the interval (0, 1000Hz) is 1h30. A FD-BEM code would have computed one frequency in about 10h. We also show the case of an engine block (fig. 5 (a)) 16158 triangles and 8086 vertices. Figure 5 (b) shows the speed-up from 1 to 8 CPU.

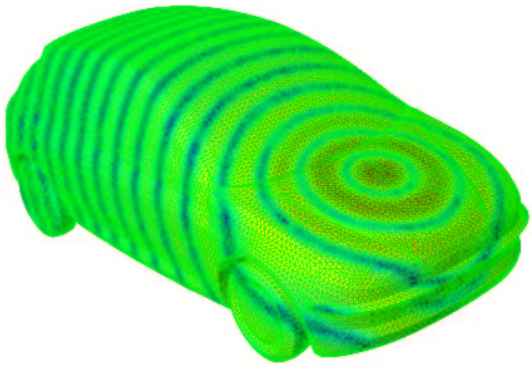
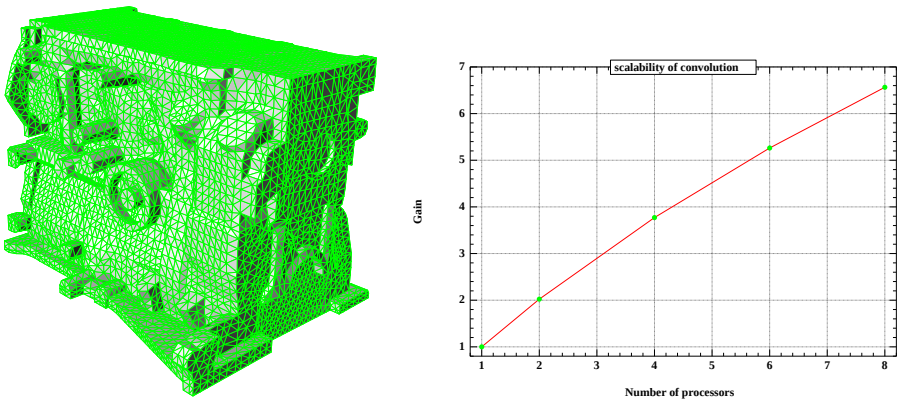


FIG. 4. *The refined Peugeot 206 mesh (PSA).*



(a) Mesh (16158 triangles). (b) Speed up.

FIG. 5. SONATE *performances on HP RP8400 - engine case : gain.*

References

[1] T.Abboud, J. El Gharib, B. Zhou, *Retarded potentials for acoustic impedance problems*, Proceedings of the 5th International Conference on Mathematical and Numerical Aspects of Waves Propagation (10-14 july 2000 - Saint Jacques de Compostelle, Spain).

[2] D. Barbier, *Méthode des potentiels retardés pour la simulation de la diffraction d'onde élastodynamique par une fissure 3D*, Thèse de l'École Polytechnique 1998.

[3] A. Bamberger et T. Ha Duong, "*Formulation variationnelle espace-temps pour le calcul par potentiel retardé d'une onde acoustique*", Math. Meth. in Appl. Sci., 8:405-435, 1986.

[4] A. Bamberger et T. Ha Duong, "*Formulation variationnelle pour le calcul de la diffraction d'une onde acoustique par une surface rigide*", Math. Meth. in Appl. Sci., 8:598-608, 1986.

[5] É. Bécache, "*Résolution par une méthode d'équations intégrales d'un problème de diffraction d'ondes élastiques transitoires par une fissure*", Thèse de doctorat, Université Paris 6, 1991.

[6] Y. Ding, "*Méthodes numériques sur l'équation intégrale aux bords pour le problème des ondes acoustiques diffractées par une surface rigide en 3D*", Thèse de doctorat, Université Paris Sud, 1989.

[7] M. Felipe, "*Étude mathématique et numérique d'un problème d'interaction fluide-structure dépendant du temps par la méthode de couplage éléments finis – équations intégrales*", Thèse de doctorat, École Polytechnique, 1994.

[8] T. Ha Duong, "*Équations intégrales pour la résolution numérique de problèmes de diffraction d'ondes acoustiques dans* $\mathbb{R}^3$*"*, Thèse de doctorat, Université Paris 6, 1987.

[9] Hewlett-Packard Company. *HP MPI User's Guide*, 1999. Part Number B6011-96010.

[10] Hewlett-Packard Company. *Parallel Programming Guide for HP-UX Systems*, 2000. Part Number B3909-90003.

[11] V. Lange, "*Équations intégrales espace-temps pour les équations de Maxwell. Calcul du champ diffracté par un obstacle dissipatif*, Thèse de doctorat, Université de Bordeaux I, 1995.

[12] Message-Passing Interface Forum. *MPI: A Message-Passing Interface STandard*, University of Tennessee, 1994.

[13] Message-Passing Interface Forum. *MPI-2: Extension to the Message-Passing Interface*, University of Tennessee, 1997. `http://www.mpi-forum.org/`

[14] A. Pujols, "*Équations intégrales espace-temps pour le système de Maxwell. Application au calcul de la surface équivalent radar*", Thèse de doctorat, Université de Bordeaux 1, 1991.

[15] T. Sayah, "*Méthode des potentiels retardés pour les milieux hétérogènes et l'approximation des couches minces par conditions d'impédance généralisées n électromagnétisme*", Thèse de doctorat, Université Paris 6, 1998.

[16] I. Terrasse, "*Résolution mathématique et numérique des équations de Maxwell instationnaires par une méthode de potentiels retardés*", Thèse de doctorat, École Polytechnique, 1993.

[17] K.R. Wadleigh, I.L. Crawford, "*Software Optimization for High Performance Computing*, Prentice Hall PTR, Hewlett-Packard Professional Books, 2000.