Advances in Engineering Software 121 (2018) 39–58

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

π-BEM: A flexible parallel implementation for adaptive, geometry aware, and high order boundary element methods

Nicola Giuliani*,[a], Andrea Mola[a], Luca Heltai[a]

SISSA—International School for Advanced Studies, Via Bonomea 265, Trieste 34136, Italy

ARTICLE INFO

Keywords:
BEM
Fast multiple method
High order elements
Local refinement
MPI
Multi-threaded
OpenSOURCE
CAD

ABSTRACT

Many physical phenomena can be modelled using boundary integral equations, and discretised using the boundary element method (BEM). Such models only require the discretisation of the boundary of the domain, making the setup of the simulation straightforward and lowering the number of degrees of freedom. However, while many parallel efficient libraries are available for the Finite Element Method (FEM), the implementation of scalable BEM solvers still poses many challenges. We present the open source framework π-BEM (where π stands for *parallel*): a novel boundary element method solver, combining distributed and shared memory paradigms to achieve high scalability. π-BEM exploits high performance libraries and graph partitioning tools to deliver a parallel solver employing automatic domain decomposition, high order elements, local refinement capabilities, and exact geometry-adaptivity (using CAD files). A preliminary fast multipole accelerator is included in the implementation. Every aspect of the library is modular and easily extendible by the community. We discuss the internal structure of the code, and present some examples to demonstrate the reliability and scalability of our implementation.

1. Introduction

Many fields of engineering benefit from accurate and reliable solvers for Boundary Integral Equations. Existing open source BEM solvers (like, for example, BEM++ [55]) focus on ease of use, rather than extensibility, and although they often offer convenient interfaces for common scenarios, they do not focus on extensibility and flexibility of the library infrastructure. In this work we propose a BEM library that leverages one of the most actively developed open source libraries, the `deal.II` library [4,7], to derive a novel implementation of the Boundary Element Method (BEM) that couples several characteristics, that cannot be found together in any other open source software for BEM. In particular our library provides support for complex geometries, high order elements, and local adaptivity that automatically respects the CAD description of the underlying geometry. The library supports hybrid parallelization combining shared and distributed memory paradigms, and we provide a preliminary Fast Multiple Method (FMM) solver. It is our opinion that a high performance BEM library that exploits high order elements with local refinement on complex surfaces fills a gap in the literature and is a key aspect for the community to solve interesting problems. We are aware that some aspects of our work are not completely innovative, for instance we could exploit existing high performance external FMM libraries instead of our own, that would grant a significance performance gain but our efforts to manage such interactions have failed. We remark that the library, since it is a documented open source software, can be easily extended and improved by the users community.

The most noteworthy equations which admit a Boundary Integral Formulation are the Laplace equation, the Helmholtz equation and the Stokes system. Such equations govern several physical phenomena of both scientific and engineering interest. Boundary Integral Formulations have been applied, among others, to problems involving hydrodynamic flows [15,25,42,49,54], flow around aerodynamic lifting bodies [20,39,45], structural mechanics [12], electrostatics [55], quantum mechanics [59], and acoustics [2,19,32].

The most convenient aspect of the Boundary Integral Representation —if the problem admits one— is that the solution at each point of the domain can be expressed in terms of convolutions on the domain boundaries between a fundamental solution and the boundary trace of the solution and of its normal gradient. In the associated numerical method (BEM), only the boundary of the domain must be discretised with a computational grid, leading to a significant reduction in the number of degrees of freedom of the discretized problem, if compared to the more common Finite Elements (FEM) or Finite Volumes (FV) approaches.

This reduction, however, comes with a drawback: the associated

* Corresponding author.
E-mail address: ngiuliani@sissa.it (N. Giuliani).

https://doi.org/10.1016/j.advengsoft.2018.03.008
Received 15 January 2018; Accepted 16 March 2018

discrete systems are dense, their assembly has a computational cost of order $O(n^2)$, and their numerical solution is far from trivial. Moreover if one considers mixed Neumann Dirichlet boundary value problems the final linear system is generally ill conditioned. If one choses to assemble the full matrix on a regular computer/laptop/workstation, the number of degrees of freedom is roughly limited to $O(10^4)$, mainly due to the computational effort required to assemble and store the $O(10^8)$ elements of the system matrix. Increasing the number of unknowns makes the assembly of such matrices almost unbearable also from a memory point of view. Storing a full matrix for 40 thousands double precision unknowns already requires more than 10 GB in RAM memory, which is close to the limit of user-level desktops.

A number of steps are possible to improve the applicability of BEM. We start by mentioning high order elements, which reduce the number of degrees freedom required to obtain a specific tolerance for problems with smooth solutions. High order BEMs are more attractive when compared to their finite element counterparts, since for finite elements, increasing the order results in a less sparse, more ill conditioned, system matrix. For BEMs the matrices are already full, and this is no longer a disadvantage. When non smooth problems are considered, high order elements can be combined with local refinement techniques to reduce the final size of the problem. In general these two techniques together work well when the domain itself is smooth. They are are no longer sufficient when the domain is only Lipschitz: if the domain presents a sharp edge the normal vector has a jump across such interface, inducing a jump also in the normal component of the solution gradient. While this is not an issue for discontinuous elements of arbitrary orders, it may be an issue if we consider continuous elements. A possible solution is the so-called double nodes technique [29], where continuity is preserved only on the solution, while its normal gradient is allowed to have a jump across physical edges. The usage of this technique allows for an accurate solution of mixed Neumann Dirichlet boundary value problems on domains with sharp edges. In recent years the developing of Computer Aided Design technologies has posed new challenges to both FEM [16] and BEM communities [31]. The integration of complex CAD geometries in BEM widens the application possibilities [43]. We exploit refining and integration strategies directly on the exact geometry specified by user provided CAD files.

As the size of problems increases, none of these techniques alone is enough to reduce memory problems. Splitting of the problem into subdomains and distributing its execution over multiple CPUs is one possible option, which is also beneficial for wall-time computational costs. Domain decomposition can be achieved at the partial differential equation level, or at the algebraic level. Each approach has its own advantages and disadvantages, but both methods increase the limit on the degrees of freedoms by exploiting distributed memory techniques, and reducing the amount of degrees of freedom handled by each single processor.

In recent years many library have been developed to address the BEM requirements. In [48,55] the authors present an effective implementation of high order boundary element methods. In [55] the boundary element method is parallelized using a shared memory parallelization and the authors focus on the implementation of different kernels using high order methods, while in [48] the authors present a hybrid parallelization scheme and the combination of high order methods with adaptive quadrature formulas. To the best of our knowledge, no opensource library is available in the literature that combines high order elements with local refinement strategies for boundary element methods on arbitrary geometries.

In this work, we present a new opensource BEM library that gathers together several algorithms and ideas in a flexible, modular, and extendible way, exploiting both shared and distributed memory parallelisms. We split the computational effort at the algebraic level for distributed parallelization, by combining a domain decomposition method based on the graph partitioning tool METIS [21], with the high performance computing library Trilinos [33] used to tackle distributed linear algebra. We use Intel Threading Building Block (TBB) [50] to exploit multicore architectures. A similar combination has been successfully applied to achieve high computational efficiency in fluid dynamics, as demonstrated in ASPECT [36]. In our BEM library the distributed memory parallelism leads to very high performance benefits, due to the structure of the matrix assembling procedures.

We present an application of our BEM methodologies to a model Laplace problem, providing both convergence and scalability tests, to assess the accuracy, and to verify the performance of our implementation. However, we remark that the same coding principles can be straightforwardly applied to any BIE solver (see, for example, [23] for the Stokes boundary integral formulation). The finite element library `deal.II` [4,6,7] is the base of our BEM implementation. The use of `deal.II` allows for a combination of high order continuous or discontinous finite elements together with local refinement strategies, including the possibility to treat sharp edges using either the double nodes techniques or discontinuous finite elements.

In recent years many methods have been developed to approximate the action of a BEM matrix-vector product in order $O(n)$ operations instead of assembling the full matrix. Examples of such methods are the Fast Multiple Method [28] or Hierarchical matrices [26]. Succesful BEM libraries and solvers should feature some or all of these expedients aimed at the reduction of the main BEM bottle-necks. Many high performance libraries take advantage of hierarchical matrices, or H matrix concept. For example BETL [34], is based on AHMED [9,10], which is a highly optimised H-matrix library. Alternative acceleration methods based on non uniform fast Fourier transforms have been recently developed in [3], with very promising results. Parallelisation of such techniques is at the moment still lacking. For most BEM libraries the fast multiple method is the acceleration method of choice, as it guarantees very high reductions from the computational point of view [28]. Greengard proposed a pure multithreaded version in 1990 [27], and only recently Yokota *et al.* developed a parallel version of the algorithm using hybrid coding techniques [60]. Several works are dedicated to assessing the applicability of FMMs to exascale problems (see, e.g., [8] and the references therein). In many cases the panels coming from the boundary discretisation are directly used in order to set up the FMM hierarchical space subdivision. This approach leads to some modification to the algorithm in order to guarantee that all mathematical assumptions are properly satisfied [48].

We have performed a preliminary acceleration test of `π-BEM` using a FMM. Given the extreme flexibility and generality of `π-BEM` the coupling with an existing FMM library, as the one depicted in [60], is extremely difficult. Moreover, the relative small size of the problem (at most $O(n^6)$) sets us in a different setting with respect to exascaling libraries as the ones developed in [8], and we don't approximate near BEM interaction by means of dielectric models, or BIIBE, as in [61]. We developed our own implementation of FMM capable of dealing with the characteristics of `π-BEM`. High order methods, local refinement and double nodes handling bring a considerable increase in the algorithm complexity when compared to classical BEMs, making it mandatory to use hybrid parallelisation techniques. A hybrid MPI-TBB parallelisation strategy allows for an optimisation of the various algorithms, while maintenance and ease of use is achieved by exploiting the `deal2lkit` library [52]. For the sake of clarity we report a comparison between the presented library and notable softwares available in the literature in Table 1.

The library we present is the only one that supports CAD based local refinement. Our contribution is structured as follows: in Section 2 we introduce the mathematical problem, while in Sections 3 and 4 we