VŠB-TECHNICAL UNIVERSITY OF OSTRAVA
FACULTY OF ELECTRICAL ENGINEERING
AND COMPUTER SCIENCE

Parallel Boundary Element Methods in Space and Time

PHD THESIS SUMMARY

2016 Michal Merta

Abstract

Efficient parallel implementation of the boundary element method is crucial for its applicability to the solution of real world engineering problems. Several approaches to the parallelization of BEM are presented in this work, including in-core vectorization using SIMD instruction set extensions, a novel distributed fast boundary element method based on the cyclic graph decompositions, and parallel space-time Galerkin boundary element method for the time dependent wave equation utilizing smooth temporal basis functions. We describe our in-house boundary element environment, in which most of the methods are implemented, and provide a commentary to author's published papers in the field.

Key Words: boundary element method, parallel fast BEM, time-domain BEM, HPC, parallel computing.

Abstrakt

Aby bylo možné metodu hraničních prkvů využít pro řešení reálných inženýrských úloh, je potřebná její efektivní paralelní implementace. V předložené práci je představeno několik přístupů k paralelizaci této metody — vektorizace pomocí SIMD instrukčních sad v rámci jednoho jádra, nová distribuovaná rychlá metoda hraničních prvků založená na cyklických dekompozicích grafů a paralelní prostoro-časová Galerkinova metoda hraničních prvků pro řešení časově závislé vlnové rovnice s využitím hladkých časových bázových funkcí. V této práci je posán námi vyvíjený hraničně prvkový software, ve kterém je většina z výše jmenovaných metod implementována, a poskytnut komentář k autorovým publikacím v tomto oboru.

Klíčová slova: metoda hraničních prvků, paralelní rychlé metody hračních prvků, BEM pro časově závislé problémy, HPC, paralelní počítání.

Contents

Introduction

The purpose of this text is to summarize the main contribution of the Ph.D. thesis “Parallel boundary element methods” submitted by the author. The thesis is presented in the form of a collection of papers published in journals with impact factor. The papers focus on an efficient implementation of the boundary element method for both stationary and time dependent problems.

In the first paper an approach to the acceleration of the boundary element method (BEM) using the SIMD (single instruction multiple data) capabilities of modern processors is presented. These vector instruction set extensions enable a concurrent computation on multiple operands using a single computational core, thus provide an energy efficient way to speed-up the computation. While the original SSE instruction set extension introduced by Intel in 1999 contained 128-bit wide vector registers supporting a parallel computation on two double and four single precision operands, the current instruction set architecture available in the Intel Xeon Phi coprocessors supports a concurrent computation on up to 8 double and 16 single precision operands. The AVX-512 instruction set extension provides similar capabilities to the Intel Xeon server processor based on the Skylake architecture. Therefore to exploit the full potential of current processors one has to modify and optimize his code to enable the utilization of the SIMD instruction sets. In the presented work we propose an approach to vectorize BEM computation based on the existing high level C++ library [17]. This relatively easily enables us to vectorize the original code while keeping its object oriented structure.

The second paper deals with the parallelization of the fast boundary element method. BEM for the stationary (elliptic) problems is now well established. There are techniques for ovecoming two main setbacks - the integration of singular kernels and the dense matter of the system matrices. The former one can be treated using, e.g., semi-analytic integration [24] or a fully numerical integration based on Duffy's substitution [26, 9, 6]. The latter problem is usually treated using a sparsification technique based on the low rank approximations of the system matrices, such as the adaptive cross approximation (ACA) [2, 4, 3, 5], the panel clustering method [16], or the fast multipole method (FMM) [25, 11]. Yet, to enable the solution of large scale problems, a distribution of the system matrices and a parallelization of the solution process is necessary. Parallel implementation of the fast boundary element method in distributed memory is still an open topic, so far it has been addressed, e.g., in [4]. In this work we present a new approach for distribution of the system matrices among nodes of a computational cluster (MPI processes) based on cyclic graph decompositions.

The last paper aims at efficient parallel implementation of the BEM-based solver for time-dependent wave scattering modelled by the hyperbolic partial differential equation. Although the initial ideas about the solution of time dependent problems using the boundary element method date back to

1960s [15] the research in this field is far less established than in the case of elliptic problems. The application of BEM on the governing hyperbolic time dependent wave equation is not as straightforward as in the case of elliptic equations. Besides the collocation methods, which are hard to analyse mathematically, there are two major approaches in the field of the time dependent boundary integral equations [8, 12]: the convolution quadrature method introduced by Lubich [19] and the Galerkin method presented by Bamberger and Ha Duong [1]. In this work we deal with the latter one. The main drawback of the Galerkin formulation is a need for a special quadrature when computing the elements of system matrices, since integration domains are intersections of boundary elements with a discrete light cone. Therefore we implement the new approach introduced by Sauter and Veit [27] which uses compactly supported, infinitely smooth basis functions to overcome this problem. Its efficient parallel implementation, as far as we know, has not yet been presented and tested on large scale problems.

Since the topic of the boundary integral equations and boundary element method is complex, in the first part of the thesis an introduction to BEM and its efficient implementation is provided. This includes a brief description of the in-house boundary element environment BEM4I developed at Department of Applied Mathematics and IT4Innovations National Supercomputing Center which was co-founded by the author. In the second part the full texts of the papers are provided together with author's commentary.

Parallel boundary element environment BEM4I

Most of the techniques presented in the work have been implemented in the library of parallel boundary element solvers BEM4I [22]. The library is written in C++ in an object oriented manner and the computation is parallelized using OpenMP and MPI in shared and distributed memory, respectively. Moreover, some parts of the library feature acceleration using the Intel Xeon Phi coprocessors. The library currently contains modules enabling solution of the Laplace, Lamé, Helmholtz, and time-dependent wave equation.

Structure of the library

The back-end of the library consists of three main sets of classes and several auxiliary classes (see Figure 1). The `BESpace` class and its descendants are used for approximation of the boundary element spaces. Among other things, they keep the degrees of the test and ansatz functions or properties associated with the approximation of the system matrices by means of a fast boundary element method.

The second level of classes is composed of bilinear form approximations stemming from the `BEBilinearForm` class. The main task of these classes is the assembly of system matrices. The parallelization in the shared memory by OpenMP and distributed memory by MPI is performed at this level.

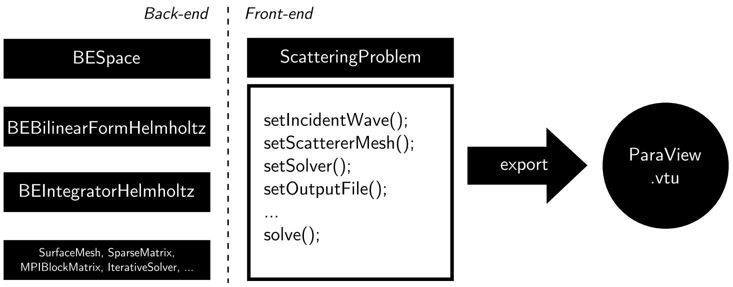

FIGURE 1: Structure of the solver for the time-harmonic wave scattering in the BEM4I library

Moreover, some of the subclasses support acceleration by offloading the computation to the Intel Xeon Phi coprocessors.

At last, the numerical quadrature over pairs of elements which is necessary to assemble local system matrices is performed by the class `BEIntegrator` and its descendants. Both semi-analytic and fully numerical approach [24, 26] is supported for the Laplace and Helmholtz equations. The fully numerical approach can be used to treat the Lamé and time-dependent wave equation. To leverage the SIMD capabilities of the modern processors the vectorization using the Vc library [17] and pragmas of the Intel compiler is employed at this level during the numerical quadrature.

The library also includes the classes enabling the manipulation with surface meshes, the classes representing the full and sparse matrices, iterative solvers, wrappers to the BLAS and LAPACK routines, methods for exporting the solution, etc.

Parallelization of the code

Acceleration of the computation using SIMD instruction set extensions

Utilization of the SIMD (single instruction multiple data) instruction set extensions is necessary to achieve the full potential of modern processors. Using the vector instructions one can perform simultaneous operations on multiple operands during one clock cycle. The number of vector operands varies depending on the length of vector registers. While the 128-bit wide registers of the SSE – SSE4.2 (Streaming SIMD Extensions) instruction sets enabled concurrent computation on four 32-bit single precision and two 64-bit double precision operands, the AVX (Advanced Vector Extensions) instruction set available in Intel's processors since 2011 doubles this capacity. Moreover, currently available Knights Corner Intel Xeon Phi coprocessors enable utilization of 512-bit wide vector registers via their IMCI (Initial Many Core Instruction) instruction set. With the upcoming AVX-512 similar capabilities will be available for all Intel's Xeon server processors. Thus it is clear that the employment of the vector instructions will be even more important in the future.

The BEM4I library features two ways of vectorization. In [21] we have presented an approach based on the high level C++ library Vc [17]. This proved to be very efficient in combination with the GNU compiler. To enable the vectorization in combination with the Intel compiler we use the

```

for ( int i = 0; i < nQuadsOuter) {
   for ( int j = 0; j < nQuadsInner; j++ ) {
      entry += singleLayerKernel(outerX[i],
     innerX[j]) * weightOut[i] * weightIn[j];
   }
}
```

LISTING 1: Original loop over pair of elements during a local system matrix computation

```
int n = nQuadPointsInner * nQuadPointsOuter;
#pragma omp simd linear ( i : 1 ) reduction(
    + : total )
for ( i = 0; i < n; i++ ) {
   entry += singleLayerKernel(outerX1[i],
    outerX2[i], outerX3[i],
   innerX1[i], innerX2[i], innerX3[i] ) *
    weights[i];
}
```

LISTING 2: Loop vectorized using `#pragma simd`

pragma based approach. For example, the original numerical quadrature over two well-separated elements during the assembly of a local system matrix depicted in Listing 1 is replaced with the manually colapsed loop provided in Listing 2. The original variables have to be transformed from the so-called array of structures form to the structure of arrays form (see Listing 3) and the arrays have to be properly aligned during their allocation using, e.g., `_mm_malloc` function. The proper vectorization of the computationally demanding loops is even more important when accelerating

```
int counter = 0;
for ( int i = 0; i < nQuadsOuter) {
   for ( int j = 0; j < nQuadsInner; j++ ) {
      outerX1[counter] = outerX[i][0];
      outerX2[counter] = outerX[i][1];
      ...
      weights[counter] = weightOut[i] *
     weightIn[j];
      counter++;
   }
}
```

Listing 3: Transformation of variables

the code using the Intel Xeon Phi coprocessors.

Shared and distributed memory parallelization

While the vectorization is performed inside the `BEIntegrator` class during the local system matrix computation, the shared memory parallelization using OpenMP is carried out by the `BEBilinearForm` class which combines these local contributions into a global system matrix. The loops over pairs of elements are parallelized using the `#pragma omp parallel` statement. The scalability tests of the OpenMP parallelization in combination with the Vc vectorization are provided in [21]. The BEM4I library provides shared memory parallelization of both dense and sparsified matrix assembly.

Moreover, to enable treatment of larger problems we provide the class `MPIACAMatrix` which distributes the matrix approximated by the adaptive cross approximation method among computational nodes (MPI processes). The ACA method splits a mesh into clusters. A pair of clusters represents a block in the system matrix. Well separated clusters are approximated using low rank approximation while close clusters are assembled as full matrices (for details see [2, 4]). The individual submatrices of the global ACA matrix are assigned in a round-robin fashion to the MPI processes. Since this is not an optimally load balanced method we provide an alternative approach based on the cyclic graph decomposition in [20].

A distribution of the system matrix arising in the time-domain solution of the wave equation is performed at this level as well. It is described in details in [34].

Acceleration by the Intel Xeon Phi coprocessors

In addition to the vectorization and OpenMP and MPI parallelization the BEM4I library accelerates the computation using the Intel Xeon Phi coprocessors. The first commercially available chip using Intel's Many Integrated Core (MIC) architecture released under the brand name Knights Corner (KNC) has been introduced in 2011. It features up to 61 cores with four hardware threads per core running at up to 1.2 GHz. The available memory bandwidth is 350 GB/s. The IMCI (Initial Many Core Instructions) instruction set provides support for a concurrent SIMD operations on eight double-precision or 16 single-precision operands. In overall, the coprocessor card provides over one TFLOPs of compute power. The biggest bottleneck of the technology is currently

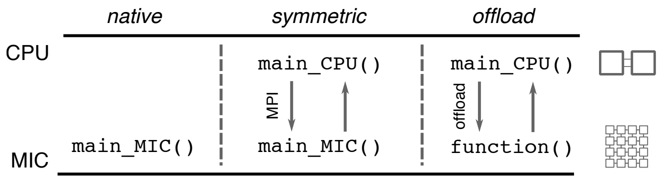

FIGURE 2: Intel Xeon Phi computation modes

the data transfer via PCI Express bus. This will be eradicated in the next generations (Knights Landing and Knights Hill) which will be available in the form of stand-alone processors.

According to the November 2015 edition of the Top 500 list 29 out of 500 most powerful supercomputers are accelerated using the Intel Xeon Phi coprocessors, including the number one, Tianhe-2 (MilkyWay-2). This number is expected to grow with the introduction of the KNL and KNH generations. Announced systems utilizing these architectures include the NERCS's next supercomputing system Cori with the theoretical peak performance of 30 PFLOPs or Argonne's future Aurora cluster with estimated peak performance of up to 450 PFLOPS. With its high level of parallelization the Xeon Phi technology therefore represents one of the possible ways towards exascale computing.

There are three main compute modes when utilizing the Xeon Phi coprocessors (see Figure 2). In the native mode the whole application runs directly on the coprocessor. In the symmetric mode the MPI processes runs on both CPU and the coprocessor. A communication is performed using

MPI messages. Finally, in the offload mode the application runs on CPU and only some parts of the code are offloaded to the coprocessor (see Figure 3).

The acceleration in the BEM4I library is based on the offload mode. The approach is in details described in [23]. Let us briefly explain how the original parallel loop assembling the system matrix on CPU (Listing 4) translates into the MIC-accelerated code. The original code loops through pairs of elements and based on a distance of elements decides whether to use the classical Gaussian quadrature (for far-enough elements) or a special method for close or identical elements. After the evaluation of the local contribution it is added to the appropriate positions of the global system matrix (line 9 of the listing). The computation is done in parallel using OpenMP therefore the addition must be carried out using atomic operation.

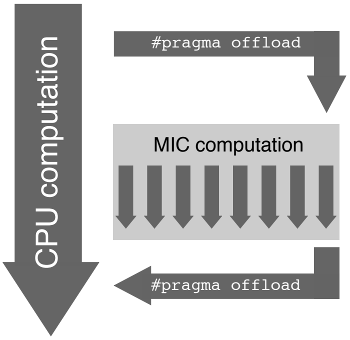

FIGURE 3: The offload compute mode

```
#pragma omp parallel for
for (int i = 0; i < nElements; i++) {
   for (int j = 0; j < nElements; j++) {
      if (areElementsDistant(i, j)) {
         getLocalMatrixFarfield(i, j,
    localMatrix);
      } else {
         getLocalMatrixNearfield(i, j,
    localMatrix);
      }
      globalMatrix.add(i, j, localMatrix);
   }
}
```

LISTING 4: Simplified CPU computation of the system matrix

During the accelerated computation the workload is distributed among the available coprocessors and CPU. The matrix is split into $N_{\mathrm{MIC}} + 1$ horizontal blocks while the dimension of blocks should follow the ratio between the theoretical peak performance of Xeon Phi and host CPU. The evaluation of the integrals over far-enough elements is split among coprocessors and CPU, however the integration over adjacent or identical elements is carried out by CPU in order to keep the computation on coprocessors as simple as possible. This does not result in a significant load balancing issue since the number of close elements only grows linearly, while the number of distant elements grows quadratically.

Since the amount of memory on the coprocessors is limited the submatrices to be assembled on Xeon Phis are further decomposed into smaller blocks. This enables a processing of the submatrix by parts. Moreover, the method of

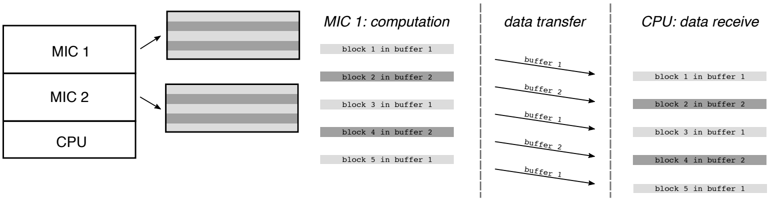

FIGURE 4: Matrix decomposition and double buffering

double buffering can be employed to hide the communication between the coprocessor and host by computation. In this case a pair of data buffers is created on a coprocessor and a host. When one of them is used for computation, data from the other one, computed during the previous iteration, is being sent to the host (see Figure 4).

Author's publications on parallel boundary element method

Acceleration of boundary element method by explicit vectorization

The first presented paper describes the acceleration of BEM using the SIMD (Single Instruction Multiple Data) features of modern processors. While the parallelization of computation in shared and distributed memory has become standard in scientific community, the vectorization capabilities of modern processors are often neglected. Since the state of the art SIMD instruction sets (AVX-512, IMCI) enable concurrent operations on up to eight double precision or sixteen single precision operands, one can easily see why ignoring these capabilities may lead to inefficient code not capable of reaching theoretical performance of modern CPUs. The paper describes vectorization of both main approaches to evaluate singular integrals occuring during assembly of the BEM system matrices (semi-analytical and fully numerical). Details on the implementation in the BEM4I library are given. The numerical experiments demonstrates the speedup of assembly of the system matrices for the Helmholtz equation with respect to the used instruction set and floating point arithmetic. Moreover, the scalability of the vectorized code in shared memory is presented. The paper was published in **Advances in Engineering Software, Volume 86** in **2015**.

Author's main contribution to the paper include:

- **review of the possible vectorization techniques**

  There are several approaches to the vectorization of the code. One can utilize the low level assembly or intrinsic functions, however this leads to the portability issues and a need for rewriting the code for each newly introduced instruction set. Another approach is to exploit the auto-vectorization capabilities of compilers, possibly combined with compilers' pragmas used to assist with the vectorization (e.g., `#pragma simd`,

`#pragma omp simd`, `#pragma ivdep`). While these capabilities of compilers have been improving lately, at the time of writing of the paper, there were still needs for more detailed control of the vectorization to deal with the BEM computation. For this reason the external C++ library Vc was employed. It provides a high level wrapper on various SIMD instruction set extensions and enables branching of the vectorized code using masked instructions.

- **transformation of the quadrature kernels to be used with the vectorization library**

  The original methods used for numerical quadrature have to be accordingly rewritten in order to be used with the Vc library. Among the most important modifications we include the change of data layout from array of structures to structure of arrays. This significantly improves the memory performance since the coordinates of quadrature points are loaded from memory to vector registers with contiguous access. Furthermore, the original loops (especially for the semi-analytic quadrature) contain numerous conditional evaluations. These kind of loops are hard for compiler to auto-vectorize. With the Vc library they are treated using the masked instructions, as described in the paper.

- **implementation in the BEM4I environment**

- **numerical experiments on the Anselm cluster**

The numerical experiments in the paper have shown the relevance of the vectorization of BEM on modern CPU architectures. The tests were performed mainly using the SSE instruction set. In some cases the speed up of the vectorized code achieved the theoretical value of the used instruction set. In other cases lower speed up can be explained by frequent application of the masked vector instructions. Concerning the AVX and newer instruction sets, one can also see that their longer vector registers enable more accurate evaluation of numerical integrals without increasing the computational time. The obtained know-how becomes even more important when porting the code to modern many-core architectures (e.g., the Intel Xeon Phi coprocessors).

A parallel fast boundary element method using cyclic graph decompositions

The parallel implementation of the fast boundary element method for the elliptic problems is still an open topic. Most of the papers deal with the distributed versions of the various tree-based algorithms (e.g., fast multipole method, Barnes-Hut method) and omit the topic of efficient parallel assembly and action of the system matrix. The second paper provided here presents a novel approach to distribution of the BEM system matrices sparsified using the adaptive cross approximation method (ACA) or the fast multipole method (FMM) among N computational nodes (MPI processes). The underlying mesh is decomposed into N submeshes, subsequently the system matrix is decomposed into $N \times N$ submatrices. Each of the processes is assigned N blocks, in such a way that minimizes the number of mesh part owned by the process. Moreover, since the diagonal blocks are the most time and memory consuming within the fast BEM, exactly one of them is assigned to each process. This way, the parallel scalability of $O((n/\sqrt{N}) \log(n/N))$ is obtained for memory consumption per process, time for setup, matrix action and assembly.

The main contributions of the author are:

- **adapting the method for usage with FMM**

  The fast multipole method has lower memory requirements in comparison to the ACA method, thus enables a solution of larger problems. In the paper the method was used to approximate the individual blocks of the global system matrix.

- **numerical experiments on the Anselm cluster**

The paper was published in **Numerical Algorithms, Volume 70 (4)** in **2015**. The numerical experiments performed up to 2744832 surface elements and 273 MPI processes show that the computational time scales with $O(1/N)$ while the memory demands scale with $O(1/\sqrt{N})$. Although our method was applied to the problems modelled by the Laplace equation, it is applicable to other types of problems (such as linear elasticity or acoustic scattering) as well.

Efficient solution of time-domain boundary integral equations arising in sound-hard scattering

The last paper aims at efficient parallel solution of the time-dependent wave equation. Efficient implementation and parallelization of the boundary element solver utilizing the smooth temporal basis functions is necessary to enable us to solve large engineering problems. Although the method simplifies the numerical quadrature and thus the assembly of system matrices, the evaluation of the smooth temporal basis functions is costly. Moreover, due to the structure of the system matrix the application of the methods like marching-on-time (MOT) is not possible. The system matrix is global in space and time. In order to preserve its special sparsity properties an iterative solver has to be applied to find a solution. Therefore, a development of a suitable preconditioner is necessary to reduce a high number of iterations of the conjugate gradient method. All of these tasks were aimed in the following paper which was published in **International Journal for Numerical Methods in Engineering 107** in **2016**.

Let us summarize the author's contribution to the paper:

- **development of the scheme for parallel distribution of block of system matrix**

  The system matrix for the space-time Galerkin method with smooth temporal basis function possesses a special structure. Since most of the blocks are duplicated, one has to take this into consideration when distributing the matrix blocks among computational nodes. At the same time the workload during the matrix vector multiplication has to be addressed. A possible approach to treat this by distribution of the block diagonals of matrix among computation nodes has been presented in the paper.

- **parallel assembly of the blocks of the system matrix**

  Despite several approaches to simplification of the temporal basis funcions evaluation and reduction of the number of their evaluation described in the paper, there is still the need for parallel assembly the individual matrix blocks. In the paper a hybrid parallelization by OpenMP and MPI has been presented. For each matrix block a non-zero pattern is precomputed and the pairs of elements contributing to the block are distributed among MPI processes. Assembly of the process' portion of data is parallelized in shared memory using OpenMP. This kind of work distribution enables us to parallelize the computation in both space and time.

- **development of a suitable iterative solver and preconditioner**

  Because of the implicitness of the scheme and the special block structure of the system matrix an iterative solver has to be employed to solve the arising linear system. In the paper several variants of the restarted

GMRES algorithm have been used and compared – GMRES without preconditioning, DGMRES preconditioned by deflations, and FGMRES preconditioned by an experimental algebraic preconditioner. The original recursive algebraic preconditioner proposed by A. Veit was modified in order to be used with the FGMRES which enables nonexact solution of inner systems, which significantly reduces its application time.

- **parallel version of the recursive algebraic preconditioner**
- **numerical experiments on the Anselm cluster**

Without these contributions the method would only be able to solve problems of hundreds of spatial unknowns in reasonable time. The numerical experiments presented in the paper show that the code for the parallel assembly of the system matrix is able to scale up to more than thousand of cores, reducing the computational time from hours to minutes. Moreover, the solution time is significantly reduced using the DGMRES iterative solver or the FGMRES iterative solver in combination with the proposed recursive algebraic preconditioner, as is shown on several examples.

Conclusion and outlook

We have presented several approaches to parallelization of the boundary element method in both space and time. We have shown the approach to accelerate the computation using the explicit vectorization leveraging the SIMD instruction set extensions of modern processors. We further extend this work and exploit the gained knowledge when accelerating the computation using the Intel Xeon Phi coprocessors installed in the Salomon supercomputer at IT4Innovations National Supercomputing Center.

Furthermore, the new method for a distribution of the sparsified system matrices proved to be suitable for usage combined with the adaptive cross approximation as well as the fast multipole method. The presented approach has been tested with piecewise constant basis functions and its extension to piecewise polynomial basis functions is a topic of current research.

In addition to the stationary problems modelled by the elliptic partial differential equations we have been focusing on the parallelization of the Galerkin boundary element method for solution of the time-dependent wave equation in 3D. To deal with the problems associated with the numerical quadrature during the assembly of the system matrices we employed the novel smooth temporal basis functions introduced by Sauter and Veit [27]. Parallelization is crutial in order to enable solution of large scale problems since this approach significantly increases the computational and memory requirements. To the best of our knowledge the parallelization of this technique has not yet been presented. Moreover, since the special structure of the system matrices requires a usage of an iterative solver, we presented solvers based on the GMRES method preconditioned by deflation and by algebraic recursive preconditioner. In future this approach can be extended to support solution of other time dependent problems, e.g., the heat equation.

Most of the abovementioned techniques have been implemented in the boundary element library BEM4I developed at IT4Innovations National Supercomputing Center. In addition to the solvers for the Laplace and wave equation the library contains the modules treating the Helmholtz and Lamé equations. It is parallelized by OpenMP and MPI in shared and distributed memory, respecively, utilizes SIMD vectorization, and is accelerated using the Intel Xeon Phi coprocessors.

Due to its high computational demands the boundary element method is a suitable candidate for efficient utilization of current peta-scale and upcoming exa-scale computer architectures. Mainly the space-time Galerkin boundary element methods for time-dependent problems produce very large system matrices which are global in space and time. While this may be from one point of view considered a disadvantage, from the other point of view it enables to introduce another level of parallelism in temporal dimension which would be impossible with most of other methods for solving time-dependent problems. For this reason we believe that there are many promising topics of research that may in future stem from this work.

Author's publications

Papers in journals with impact factor

- A. Veit, M. Merta, J. Zapletal, and D. Lukáš. "Efficient solution of time-domain boundary integral equations arising in sound-hard scattering". In: *International Journal for Numerical Methods in Engineering* 107 (2016). IF 2.055 (Q1), pp. 430–449. `DOI: 10.1002/nme.5187`.

- D. Lukáš, P. Kovář, T. Kovářová, and M. Merta. "A parallel fast boundary element method using cyclic graph decompositions'. In: *Numerical Algorithms* 70.4 (2015). IF 1.417 (Q1), pp. 807–824. `DOI: 10.1007/s11075-015-9974-9`.

- M. Merta and J. Zapletal. "Acceleration of boundary element method by explicit vectorization". In: *Advances in Engineering Software* 86 (2015). IF 1.402 (Q2), pp. 70–79. `DOI: 10.1016/j.advengsoft.2015.04.008`.

- M. Čermák, V. Hapla, D. Horák, M. Merta, and A. Markopoulos. "Total-FETI domain decomposition method for solution of elasto-plastic problems". In: *Advances in Engineering Software* 84 (2015). IF 1.402 (Q2), pp 48–54. `DOI: 10.1016/j.advengsoft.2014.12.011`.

- M. Merta and J. Zapletal "A parallel library for boundary element discretization of engineering problems". Accepted to Mathematics and Computers in Simulation (2014). IF 0.949.

Papers in conference proceedings

- M. Merta and J. Zapletal. “BEM4I applied to shape optimization problems”. In: *AIP Conference Proceedings* 1738 (2016). DOI: 10.1063/1.4952145.

- M. Merta, J. Zapletal, J. Jaros. “Many core acceleration of the boundary element method”. In: *Lecture Notes in Computer Science* 9611 LNCS (2016), pp. 116–125. DOI: 10.1007/978-3-319-40361-8_8

- M. Čermák, M. Merta, and J. Zapletal. “A novel boundary element library with applications”. In: *AIP Conference Proceedings* 1648 (2015). DOI: 10.1063/1.4913036.

- Merta, M., Zapletal, J. “Library of parallel boundary element method based solvers for solution of the time-dependent wave equation”. In: *Civil-Comp Proceedings 107* (2015).

- M. Merta, A. Vašatová, V. Hapla, and D. Horák. “Parallel implementation of total-FETI DDM with application to medical image registration”. In: *Lecture Notes in Computational Science and Engineering* 98 (2014), pp. 917–925. DOI: 10.1007/978-3-319-0578
9-7_89

- V. Hapla, D. Horák, and M. Merta. “Use of direct solvers in TFETI massively parallel implementation”. In: *Lecture Notes in Computer Science* 7782 LNCS (2013), pp. 192–205. DOI: 10.1007/987-3-642-36803-5_14.

- D. Horák, P. Kabelíková, M. Merta, and V. Vondrák. “The OOSol scalable library based on a domain decomposition method”. In: *Civil-Comp Proceedings 95* (2011).

Conferences[1]

- M. Merta, A. Veit, J. Zapletal, and D. Lukáš. "Parallel time-domain boundary element method for 3-dimensional wave equation". MAFELAP 2016, London, United Kingdom, 2016.

- M. Merta and J. Zapletal. "Acceleration of the boundary element library BEM4I using the Intel Xeon Phi coprocessors". CMSE 2016, Rožnov p. Radhoštěm, Czech Republic, 2016 & PRACEDays16, Prague, Czech Republic, 2016 (poster).

- M. Merta. "Introduction to the Intel Xeon Phi programming". PRACE Winter School 2016, Bratislava, Slovakia, 2016.

- M. Merta, J. Zapletal, L. Říha, T. Brzobohatý, A. Markopoulos, and O. Meca. "Acceleration of the numerical libraries using the Intel Xeon Phi coprocessors". 4th IT4Innovations Annual Conference, Přerov, Czech Republic, 2015.

- M. Merta and J. Zapletal. "Acceleration of the BEM4I library using the Intel Xeon Phi coprocessors". 13. Workshop on fast boundary element methods in industrial applications, Hirschegg, Austria, 2015.

- M. Merta. "Acceleration of the boundary element method". WOFEX 2015, Ostrava, Czech Republic, 2015.

- M. Merta and J. Zapletal. "Library of parallel boundary element method based solvers for solution of the time dependent wave equation". PARENG 2015, Dubrovnik, Croatia, 2015.

[1] Speaker or poster presenter

- M. Merta and J. Zapletal. "BEM4I applied to 3D wave scattering problems". EASC 2015, Edinburgh, United Kingdom, 2015 (poster).

- M. Merta and J. Zapletal. "A library of parallel solvers based on the boundary element method". 3rd IT4Innovations Annual Conference, Ostrava, Czech Republic, 2014.

- M. Merta and J. Zapletal. "BEM4I – Parallel boundary element library". Modelling 2014, Rožnov p. Radhoštěm, Czech Republic, 2014 & ESCO 2014, Pilsen, Czech Republic, 2014 & EASC 2014, Stockholm, Sweden, 2014 (poster).

- M. Merta. "Parallel solution of the time dependent wave equation". WOFEX 2014, Ostrava, Czech Republic, 2014.

- M. Merta and J. Zapletal. "BEM4I – Parallel BEM library with applications to the wave equation". Modelling 2014, Rožnov p. Radhoštěm, Czech Republic, 2014.

- M. Merta and J. Zapletal. "BEM4I – Preview of the new BEM library". SPOMECH Workshop 2013, Ostrava, Czech Republic, 2013.

- M. Merta. "Parallel implementation of fast boundary element method". WOFEX 2013, Ostrava, Czech Republic, 2013.

- M. Merta and D. Lukáš. "Parallel implementation of fast boundary element method". SNA 2013, Rožnov p. Radhoštěm, Czech Republic, 2013.

- M. Merta and M. Čermák. "Parallel implementation of TFETI DDM for the solution of elasto-plastic problems". PARENG 2013, Pécs, Hungary, 2013.

- M. Merta. "Introduction to Scientific Computing Using Trilinos". PRACE Spring School 2012, Kraków, Poland, 2012.
- M. Merta, A. Vašatová, V. Hapla, and D. Horák. "Massively parallel implementation of total-FETI method with application to medical image registration". Workshop Fast BEM and BETI, Ostrava, Czech Republic, 2012.
- M. Merta, A. Vašatová, V. Hapla, and D. Horák. "Massively parallel implementation of total-FETI method with application to medical image registration". DD 21, Rennes, France, 2012.

Software

- M. Merta and J. Zapletal. "BEM4I" [Software]. Available from `http://bem4i.it4i.cz`.

Bibliography

[1] A. Bamberger and T. Ha Duong. “Formulation variationnelle espace-temps pour le calcul par potentiel retardé de la diffraction d’une onde acoustique (I)”. In: *Mathematical methods in the applied sciences* 8.1 (1986), pp. 405–435.

[2] M. Bebendorf. “Approximation of boundary element matrices”. In: *Numer. Math.* 86 (2000).

[3] M. Bebendorf. *Hierarchical Matrices.* Springer, 2008.

[4] M. Bebendorf and R. Kriemann. “Fast parallel solution of boundary integral equations and related problems”. In: *Comp. Vis. Sci.* 8 (2005).

[5] M. Bebendorf and S. Rjasanow. “Adaptive Low-Rank Approximation of Collocation Matrices”. English. In: *Computing* 70.1 (2003), pp. 1–24. ISSN: 0010-485X. DOI: `10.1007/s00607-002-1469-6`.

[6] J. Bouchala and J. Zapletal. “Effective Semi-Analytic Integration for Hypersingular Galerkin Boundary Integral Equations for the Helmholtz Equation in 3D”. In: *Appl. Math.* (to appear).

[7] E. Bécache. *Equations intégrales pour l’équation des ondes.* Available at `http://www-rocq.inria.fr/~becache/cours_eqinteg.ps.gz.`. [Online; accessed 24-November-2015]. 1994.

[8] M. Costabel. “Time-Dependent Problems with the Boundary Integral Equation Method”. In: *Encyclopedia of Computational Mechanics* (2004).

[9] M.G. Duffy. “Quadrature over a pyramid or cube of integrands with a singularity at a vertex”. In: *SIAM J. Numer. Anal.* 19 (1982).

[10] J. El Gharib. “Problèmes de potentiels retardés pour l’acoustique”. PhD thesis. École Polytechnique, 1999.

[11] L. Greengard and V. Rokhlin. “A Fast Algorithm for Particle Simulations”. In: *Journal of Computational Physics* 135.2 (1997), pp. 280 –292. ISSN: 0021-9991. DOI: `http://dx.doi.org/10.1006/jcph.1997.5706`.

[12] T. Ha-Duong. “On retarded potential boundary integral equations and their discretisation”. In: *Topics in computational wave propagation.* Springer, 2003, pp. 301–336.

[13] T. Ha Duong, A. Bamberger, and J. C. Nedelec. “Formulation variationnelle pour le calcul de la diffraction d’une onde acoustique par une surface rigide”. In: *Mathematical Methods in the Applied Sciences* 8.1 (1986), pp. 598–608. ISSN: 1099-1476. DOI: `10.1002/mma.1670080139`.

[14] T. Ha-Duong, B. Ludwig, and I. Terrasse. “A Galerkin BEM for transient acoustic scattering by an absorbing obstacle”. In: *International Journal for Numerical Methods in Engineering* 57.13 (2003), pp. 1845–1882. ISSN: 1097-0207. DOI: `10.1002/nme.745`.

[15] W. Hackbusch, Wendy Kress, and S. Sauter. “Boundary Element Analysis: Mathematical Aspects and Applications”. In: ed. by Martin Schanz and Olaf Steinbach. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. Chap. Sparse Convolution Quadrature for Time Domain Boundary Integral Formulations of the Wave Equation by Cutoff and Panel-Clustering, pp. 113–134. ISBN: 978-3-540-47533-0. DOI: `10.1007/978-3-540-47533-0_5`.

[16] W. Hackbusch and Z.P. Nowak. “On the fast matrix multiplication in the boundary element methods by panel clustering”. In: *Numer. Math.* 54 (1989).

[17] M. Kretz and V. Lindenstruth. “Vc: A C++ library for explicit vectorization”. In: *Software: Practice and Experience* 42.11 (2012), pp. 1409–1430. ISSN: 1097-024X. DOI: `10.1002/spe.1149`.

[18] J.L. Lions and E. Magenes. *Non-Homogeneous Boundary Value Problems and Applications.* Springer-Verlag, 1972.

[19] Ch. Lubich. “On the multistep time discretization of linear initial-boundary value problems and their boundary integral equations”. In: *Numerische Mathematik* 67.3 (1994), pp. 365–389.

[20] D. Lukáš, P. Kovář, T. Kovářová, and M. Merta. “A parallel fast boundary element method using cyclic graph decompositions”. In: *Numerical Algorithms* 70.4 (2015). IF 1.417 (Q1), pp. 807–824. DOI: `10.1007/s11075-015-9974-9`.

[21] M. Merta and J. Zapletal. “Acceleration of boundary element method by explicit vectorization”. In: *Advances in Engineering Software* 86 (2015). IF 1.402 (Q2), pp. 70–79. DOI: `10.1016/j.advengsoft.2015.04.008`.

[22] M. Merta and J. Zapletal. *BEM4I.* Available at `http://industry.it4i.cz/en/products/bem4i/`. [Online; accessed 13-October-2014]. 2014.

[23] M. Merta, J. Zapletal, and J. Jaros. “Many core acceleration of the boundary element method”. Accepted to Lecture Notes in Computer Science, 2015.

[24] S. Rjasanow and O. Steinbach. *The Fast Solution of Boundary Integral Equations.* Springer, 2007. ISBN: 978-0-387-34042-5.

[25] V. Rokhlin. “Rapid solution of integral equations of classical potential theory”. In: *Journal of Computational Physics* 60.2 (1985), pp. 187 –207. ISSN: 0021-9991. DOI: `http://dx.doi.org/10.1016/0021-9991(85)90002-6`.

[26] S. Sauter and C. Schwab. *Boundary Element Methods*. Springer Series in Computational Mathematics. Springer, 2010. ISBN: 9783540680932.

[27] S. Sauter and A. Veit. “A Galerkin method for retarded boundary integral equations with smooth and compactly supported temporal basis functions”. In: *Numerische Mathematik* 123.1 (2013), pp. 145–176.

[28] S. Sauter and A. Veit. “A Galerkin Method for Retarded Boundary Integral Equations with Smooth and Compactly Supported Temporal Basis Functions. Part II: Implementation and Reference Solutions.” In: *Preprint* (2011).

[29] S. Sauter and A. Veit. “Adaptive Time Discretization for Retarded Potentials”. In: *arXiv preprint arXiv:1404.2322* (2014).

[30] S. Sauter and A. Veit. “Adaptive time discretization for retarded potentials”. In: *Numerische Mathematik* (2015). Article in Press. DOI: `10.1007/s00211-015-0726-5`.

[31] S. Sauter and A. Veit. “Retarded boundary integral equations on the sphere: exact and numerical solution”. In: *IMA Journal of Numerical Analysis* (2013). DOI: `10.1093/imanum/drs059`. eprint: `http://imajna.oxfordjournals.org/content/early/2013/07/14/imanum.drs059.full.pdf+html`.

[32] O. Steinbach. *Numerical Approximation Methods for Elliptic Boundary Value Problems: Finite and Boundary Elements*. Texts in applied mathematics. Springer, 2008. ISBN: 9780387313122.

[33] E. P. Stephan, M. Maischak, and E. Ostermann. "Transient Boundary Element Method and Numerical Evaluation of Retarded Potentials". English. In: *Computational Science – ICCS 2008*. Ed. by Marian Bubak, GeertDick van Albada, Jack Dongarra, and PeterM.A. Sloot. Vol. 5102. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 321–330. ISBN: 978-3-540-69386-4. DOI: 10.1007/978-3-540-69387-1_35.

[34] A. Veit, M. Merta, J. Zapletal, and D. Lukáš. "Efficient solution of time-domain boundary integral equations arising in sound-hard scattering". In: *International Journal for Numerical Methods in Engineering* 107 (2016). IF 2.055 (Q1). DOI: 10.1002/nme.5187.