Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 2200–2209

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

# Software Framework for Parallel BEM Analyses with H-matrices Using MPI and OpenMP

Takeshi Iwashita[1, 5], Akihiro Ida[2, 5], Takeshi Mifune[3, 5], and Yasuhito Takahashi[4, 5]

*[1]Hokkaido University, Sapporo, Japan, [2]The University of Tokyo, Tokyo, Japan, [3]Kyoto University, Kyoto, Japan, [4]Doshisha University, Kyotanabe, Japan, [5]JST CREST, iwashita@iic.hokudai.ac.jp, ida@cc.u-tokyo.ac.jp, mifune@fem.kuee.kyoto-u.ac.jp, ytakahashi@mail.doshisha.ac.jp*

A software framework has been developed for use in parallel boundary element method (BEM) analyses. The framework program was parallelized in a hybrid parallel programming model, and both multiple processes and threads were used. Additionally, an H-matrix library for a distributed memory parallel computer was also developed to accelerate the analysis. In this paper, we describe the basic design concept for the framework and details of its implementation. The framework program, which was written with MPI functions and OpenMP directives, is mainly intended to reduce the user's parallel programming costs. We also show the results of a sample analysis performed with approximately 60,000 unknowns. The numerical results verify the effectiveness of both the parallelization and the H-matrix method. In the test analysis, which was performed using a single core, the H-matrix version of the framework is 17-fold faster than the dense matrix version. The parallel framework program with the H-matrix attains an approximately 50-fold acceleration using 128 cores when compared with sequential computation.


Peer-review under responsibility of the scientific committee of the International Conference on Computational Science

*Keywords:* Boundary element analysis, H-matrix, parallel processing, software framework

## 1 Introduction

The boundary element method (BEM) is a partial differential equation (PDE) solver and is used in various numerical simulations [1]. One of the advantages of the BEM is that it is only necessary for the surfaces of the objects to be discretized in the analyzed model. The number of unknowns and the meshing costs are generally less than would be required for other volume discretization methods, such as the finite element method. However, because a dense coefficient matrix arises in the BEM, the total computational cost and the memory footprint of the method tend to be significantly high. To remedy these problems, the Fast Multipole Method (FMM) [2], the hierarchical matrices (H-matrices) [3], and their related techniques [4][5] have been widely used as approximation techniques for the dense coefficient matrix. Additionally, parallel processing is also used to reduce the overall simulation time.

1877-0509 
Peer-review under responsibility of the scientific committee of the International Conference on Computational Science
10.1016/j.procs.2017.05.263

However, it is not easy to develop the parallel simulation code when using these sophisticated numerical techniques.

Based on these considerations, we have developed an open-source software framework for use in parallel BEM analysis that is parallelized using the hybrid parallel programming model. In addition, to reduce both the simulation time and the memory footprint of the method, we have also developed a distributed parallel H-matrix library [6] that can be used together with the framework program. The programming costs for the fast parallel BEM codes are expected to be reduced by using the framework.

In this paper, we describe the design concept for the framework and the detail of its implementation. Additionally, numerical results from an analysis using the proposed framework program confirm the effectiveness of the software.

# 2 Design Concept of BEM Framework

When we designed the software framework for parallel BEM analysis, which we have named the BEM-BB framework, we adopted a software model in which the complete simulation code consists of the framework program and additional user-defined functions. A BEM analysis program generally involves the following steps: 1) input of the model data; 2) definition of the boundary element integral; 3) setting of the boundary conditions; 4) generation of a coefficient matrix and a right-hand vector; 5) application of a linear solver; and 6) output of the analysis results.

Our software framework mainly supports steps 1, 4, and 5. In other words, the BEM-BB framework provides programs for input of the model data, assembly of the coefficient matrix and the right hand vector, and solution of the linear system. The programs are parallelized in a hybrid parallel programming model using MPI and OpenMP. In our software design, users should prepare a function program in which each element of the coefficient matrix is calculated and then returned to the main program. This means that users must develop a program to calculate the integrals of boundary elements. They also handle the right hand side vector, the boundary conditions, and the output of the results themselves, although an application program interface that is similar to that for the coefficient matrix setting step is provided for the right hand vector. In the following, we explain the background of the software design.

While either a fundamental solution or a Green's function is an important element of the BEM, it is dependent on the governing equation of the targeted physical phenomenon. Furthermore, even if the same fundamental solution is used, the element shape, the degree of the elements, and the integration method used (i.e., analytical or numerical) may vary depending on the model used and the required accuracy of the analysis. To deal with these diverse aspects of BEM analyses, we have separated the program for integration of the elements, which is written by the user, from the framework. In other words, the software framework developed here is mainly focused on support for programming on a distributed parallel computer, which is often troublesome, and the framework can be used for various BEM analyses.

While we have developed a basic software model, we understand that some users will want a black-box-type BEM analysis code. To support these users, we have also prepared template programs for typical BEM analyses. Each template program includes a function program for boundary element integration in a specific problem domain. Use of the template in combination with the BEM-BB framework allows the user to obtain a complete BEM analysis program. Fig. 1 illustrates the basic concept of the developed framework.

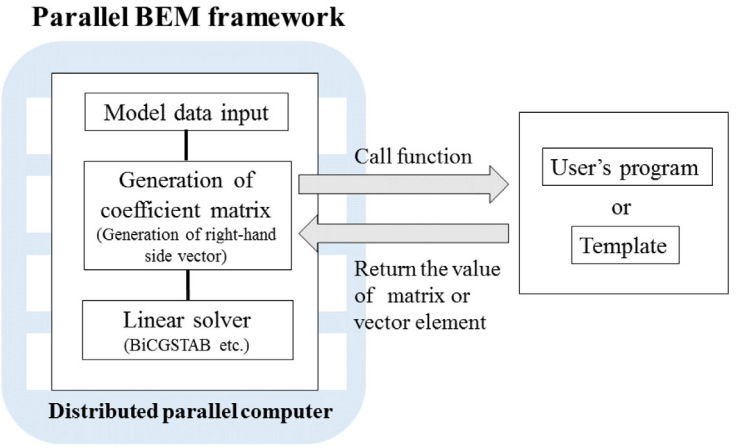


**Figure 1:** Design concept for parallel BEM analysis framework

# 3 Implementation of Framework for Parallel Boundary Element Analyses

We have developed two versions of the framework for parallel BEM analysis from the basic design presented above. The first version is based on dense matrix computations, and supports many varieties of BEM analyses. The second version uses the H-matrix technique, and is targeted for high performance (i.e., high-speed and low memory) BEM analysis. The framework is used with a distributed parallel H-matrix library, called HACApK [6], which has been developed by the authors. In this section, we explain these two implementations in detail. The main language used in both the BEM-BB framework and the HACApK library is Fortran90.

## 3.1 Implementation of the Framework using Dense Matrix Computations

The main parts of the framework are used for the data inputs, coefficient matrix generation, and the linear iterative solvers.

### 3.1.1. Model data input part

Within the developed framework, all processes share the input data for the analyzed model, i.e., each process has a copy of the data. The master process loads the data from a file system, and these data are distributed to the other processes using the MPI_Bcast function. In this case, the model data can be used in the user-defined function for element integration without process communications, which makes the programming of this function much simpler. The memory space required for the model data is substantially smaller than that required for the coefficient matrix in a large-scale analysis. Although the dense coefficient matrix requires $O(n^2)$ memory footprints, the size of the model data is proportional to $n$, where $n$ is the number of boundary elements.

Because the developed framework is intended for a three-dimensional analysis, the model data include:

(1) Scalar values

(1-1) Number of nodes (integer*4): nond
(1-2) Number of faces (integer*4): nofc
(1-3) Number of nodes on each face (integer*4): nond_on_fc
(1-4) Number of integer parameters set on each face (integer*4): nint_para_fc
(1-5) Number of real parameters set on each face (integer*4): ndble_para_fc

(2) Arrays
(2-1) Coordinates of nodes in three dimensions (real*8): np(3,nond)
(2-2) Node number constructing for each face (integer*4): face2node(nond_on_face, nofc)
(2-3) Integer parameters set on faces (integer*4): int_para_fc(nint_para_fc, nofc)
(2-4) Real parameters set on faces (real*8): dble_para_fc(ndble_para_fc, nofc)

The model data include essential information such as the coordinates of the nodes and the user-defined integers or real parameters that can be used in the function program for element integration.

### 3.1.2. Coefficient matrix generation part

After the model data are loaded, each process prepares the memory space (array) required for the segment of the coefficient matrix that is assigned to that process. Because the developed framework supports hybrid parallel processing, the entire program is multithreaded. In multi-thread parallel processing, it is preferable that the data that are accessed by a thread are stored in a memory module that is located as closely as possible to the computing core that runs the thread. To achieve this, each thread performs the first touch operation of a part of the array for the coefficient matrix [7]. Next, each thread calls the function to compute the coefficient matrix elements in parallel; this is either developed by a user or included in a template. Because the matrix element computations are mutually independent, the parallel efficiency of the matrix generation part is expected to be high.

### 3.1.3. Linear solver part

Within the developed framework, iterative solvers are used to solve the linear system of equations that is derived from the BEM. At present, BiCGSTAB [8] and GPBiCG [9] methods are available. These iterative solvers are categorized within the Krylov subspace methods, for which the computational kernels are the inner product and matrix vector multiplication. The multi-thread and multi-process parallelization of these kernels is not difficult. We have developed two versions for each solver, one of which uses the functions of the Basic Linear Algebra Subprograms (BLAS) library. The other version includes a full set of programs for the solver and does not require an external library for its execution.

## 3.2 Framework Implementation using H-matrix Computations

The framework that is based on H-matrix computation uses the parallel H-matrix library HACApK. This library supports both H-matrix generation and H-matrix vector multiplication. The library is also parallelized in the hybrid parallel programming model. The usage of the H-matrix-based framework is almost the same as that of a dense matrix-based framework. However, the two frameworks have some differences in terms of the implementation of their matrix generation and linear solver parts.

In the coefficient matrix generation part, the framework calls the H-matrix generation routine in HACApK. In the generation routine, the library program calls the element integration function (user-defined or in a template) and generates an H-matrix, which is an approximation of the coefficient matrix. In the proposed framework, no temporal dense coefficient matrix is generated to keep the benefits of the use of the H-matrix in the memory footprint and the reduced computational effort. Because the element integration function is separated from the framework program in our software design, it can be called directly from the H-matrix generation program.

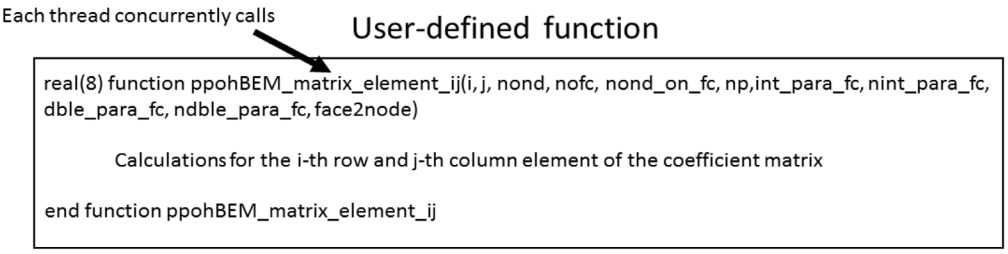


**Figure 2:** Interface of user-defined function for the element of the coefficient matrix

Additionally, the linear solver requires a different treatment to that used in the dense matrix case. We have developed BiCGSTAB and GCR [10] solvers. The matrix-vector multiplication that is involved in these solvers is replaced here with H-matrix vector multiplication. The multiplication of the generated H-matrix and the vectors is performed using the routine from the HACApK library.

## 3.3 How to Use the Framework

This subsection summarizes the usage of the framework from the user's viewpoint. First, the configuration file is edited. In this file, the user fixes the linear solver to be used and certain parameters, such as the convergence criterion for the iterative solver. Next, the model data are prepared. When a template is used, the user simply compiles the framework program with the template and subsequently runs the executable binary file. When the template is not used, the user prepares a function that returns the element of the coefficient matrix. Fig. 2 shows the interface of the user-defined function. In the function, the matrix element is calculated by using the input model data. Although the function is written in the serial programming model, it is concurrently called by multiple threads. After the boundary condition and the right-hand side vector are appropriately set, the program is then compiled and executed. Note that the MPI library and its runtime are required for the execution, because the program uses the MPI functions. The simulation code runs in the multiple processes and threads mode from start to finish, though some limited portions of tasks are executed by only the master process or thread.

# 4 Numerical Results

## 4.1 Test Model and Computers

In this section, we present a sample BEM analysis result based on use of the proposed framework. In the test model, three perfectly conductive spheres are set at distances of 0.25 m, 0.75 m, and 0.25 m from the ground (0 V). The radius of each sphere is 0.25 m. These spheres are excited with voltages of 1 V, -1 V, and 1 V, respectively. Fig. 3 shows the test model that was analyzed. The electric charge induced on the surface of the spherical conductor is calculated using BEM.

The electric image method is used in the analysis. In this method, the images of three spheres are set on the opposite side to the ground. The resulting six spheres are discretized using a total of 64,800 triangular face elements.

The basic equation for the analysis is given by Laplace's equation. One unknown is set on each face element. The analytical method that was reported in [11] is used for integration of the elements. The integration function is open for public use as a template for the surface charge method.

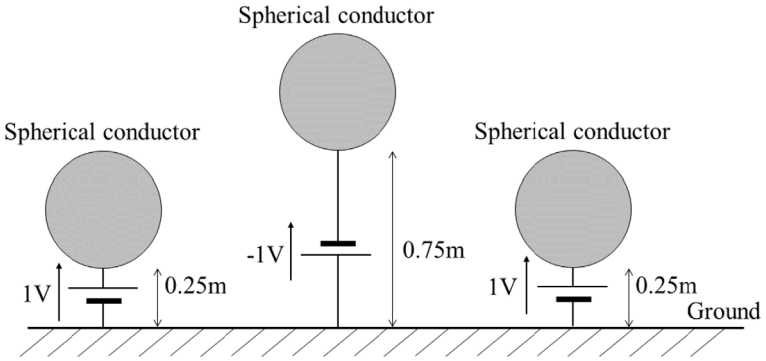


**Figure 3:** Analyzed test model

For the numerical test, we used the Appro GreenBlade 8000 system at Kyoto Univ. Each node is equipped with two Intel Xeon E5-2670 processors (with eight cores). The memory bandwidth of each node is 102 GB/s. The internal network between the nodes is based on the InfiniBand (IB) FDR. The Intel Fortran compiler ver. 13.1.3 is used with options of -O3 -xHost -ipo -openmp –mcmodel=medium -shared-intel.

## 4.2 Numerical Results

Figure 4 shows the surface charge density that was calculated in the sample analysis. In the analysis, we used the BiCGSTAB method with the convergence criterion of a relative residual norm of less than $10^{-8}$. The surface charge is induced more strongly on surface areas that are close to other spheres.

Figure 5 shows the parallel speedup of the framework with dense matrix computation. The numerical test was performed using the flat-MPI configuration (single thread execution). The number of nodes was varied from 1 to 16. The figure shows strong scalability when compared with the results of serial computation. While a slight degradation of the parallel performance on 256 cores is observed, an almost ideal speed-up is obtained. These results are explained as follows.

In the BEM analysis, the dominant elements in terms of computation time are the matrix generation and linear solver parts. In the matrix generation part, because each process executes its computation independently, high parallel efficiency can be expected. In the linear solver part, communication between the processes is necessary for the inner products and the matrix vector multiplication. Therefore, the parallel efficiency of the iterative solver generally deteriorates as the number of processes increases.

The influence of the two computational kernels on the total computation time varies based on the balance between the number of arithmetic operations required for the integration and the number of iterations required to produce convergence. In the test model, the matrix generation part affects the simulation time more significantly in serial computation (2307 s for matrix generation and 436 s for the iterative solution). Relatively high parallel efficiency is therefore achieved. However, when 256 cores are used, the computation time for matrix generation is reduced, and is then comparable to that for the linear solver (9.1 s for matrix generation and 3.4 s for the iterative solution). Consequently, the influence of communications in the iterative solver on the parallel efficiency becomes apparent and the speedup results in a slightly less than ideal value.

Table 1 compares the computation times of the two framework implementations using dense or H-matrix computations. The flat-MPI configuration was used. For serial computation, the H-matrix implementation is approximately 17 times as fast as the dense matrix implementation.

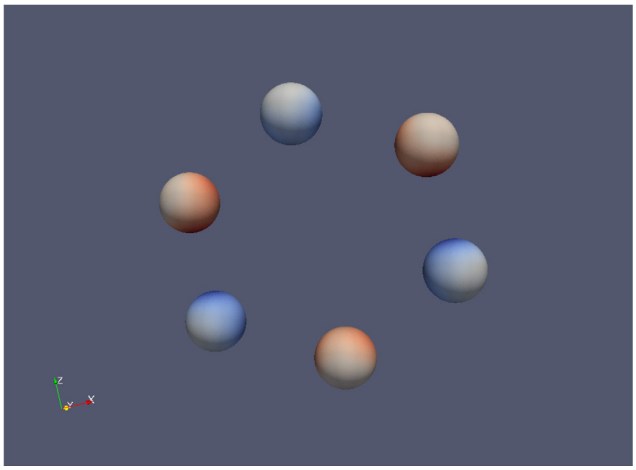

**Figure 4:** Numerical result for surface charge density

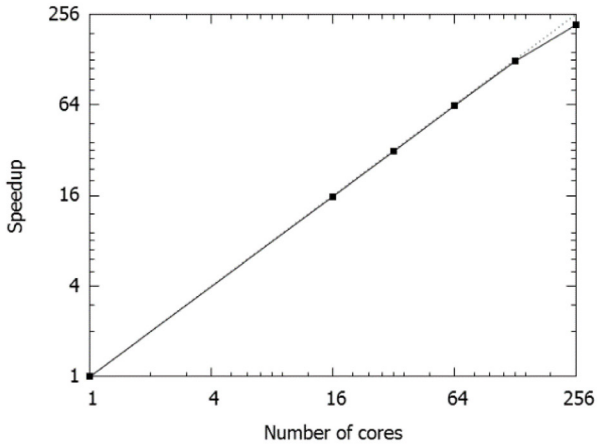


**Figure 5:** Parallel speedup of analysis when using framework with dense matrix computations

| # Cores | 1 | 16 | 64 | 128 | 256 |
|---|---|---|---|---|---|
| Dense matrix | 2753 | 174 | 43.4 | 21.9 | 12.6 |
| H-matrix | 156 | 14.2 | 7.8 | 6.0 | 7.6 |

**Table 1:** Computation time (s) for analysis based on the developed framework (flat-MPI configuration)

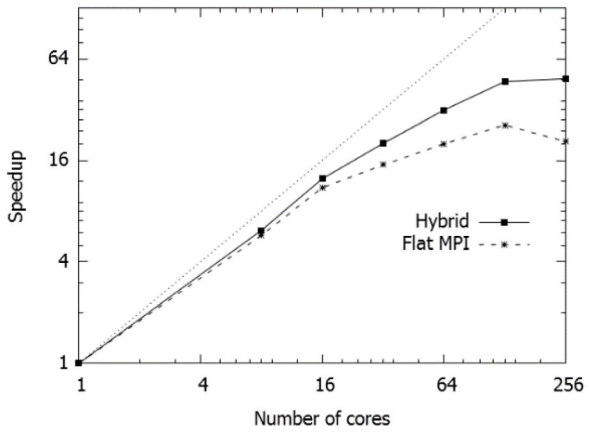


**Figure 6:** Parallel speedup of analysis when using framework with H-matrix computations (showing comparison of parallel programming models)

Figure 6 shows a comparison of the flat-MPI and hybrid (multi-process and multi-thread) parallel executions. The number of unknowns (over 60,000) is not small for a conventional BEM analysis using dense matrix computations. However, when the H-matrix technique is introduced, the total amount of data required for the coefficient matrix is significantly reduced and the computational efforts required for element integration are reduced in proportion to the quantity of data. Consequently, the effect of the iterative solver on the total computation time is enhanced. In the linear solver, all-gather type collective communications are used. The communications are often implemented using the multiple tasks in proportion to the number of processes. Therefore, a reduction in the number of processes can be effective in reducing the communications overheads. Fig. 6 shows the advantages of the hybrid parallel programming model in the framework with the H-matrix. In the analysis, when using the H-matrix technique and the parallel processing on 256 cores, we attain an 856-fold speedup when compared with sequential BEM analysis using dense matrix computations.

# 5 Related works

In the area of software frameworks for (parallel) PDE solvers, many studies have used the finite element method (FEM), such as the well-known GeoFEM [12], HPC-MW [13], and Free FEM++ [14] frameworks. Although there are fewer frameworks for parallel BEM analysis than for FEM analysis, several software applications and libraries support BEM analysis and related acceleration methods such as FMM and H-matrices. For example, ExaFMM [15] and FMMTL [16] support the implementation of FMM. In [17], an intensive review of recent FMM software development is provided. HLIBpro [18], a popular H-matrix software library, supports many aspects of simulations using H-matrices, and employs various boundary element integration functions from which users can develop a parallel BEM code. BEM++ [19] is a recent software framework for BEM analyses, and is equipped with an interface to an external H-matrix library. The difference between our open-source framework and other software products is that we focus on the hybrid MPI and OpenMP parallelization of a whole BEM analysis code.

# 6 Conclusions

We have developed a software framework to reduce programming costs for parallel BEM analyses. The framework is parallelized in a hybrid multi-process and multi-thread parallel programming model. We have also developed a distributed parallel H-matrix library that is used in the framework. The software package is called ppOpen-APPL/BEM and is available from the website [20]. We verified that the software could run on the Appro GB8000, Cray XC30, and Fujitsu FX10 systems. A sample analysis using 64,800 unknowns confirmed the effectiveness of the H-matrix and hybrid parallel processing method. In future, we will examine the framework in a more practical and larger model and enhance its functionality (e.g. support for nonlinear analysis).

# Acknowledgements

A part of this work was supported by JSPS KAKENHI Grant Number JP26289075.

# References

1. C. A. Brebbia, J. C. F. Telles and L. C. Wrobel, Boundary Element Techniques, Theory and Application in Engineering, Springer-Verlag, 1984.
2. H. Cheng, L. Greengard and V. Rokhlin, “A Fast Adaptive Multipole Algorithm in Three Dimensions”, J. Comput. Phys., Vol. 155, 1999, pp. 468-498.
3. L. Grasedyck and W. Hackbusch, “Construction and arithmetics of H-matrices”, Computing, Vol. 70, 2003, pp. 295-334.
4. S. Chandrasekaran, M Gu and W Lyons, “A fast adaptive solver for hierarchically semiseparable representations”, CALCOLO, Vol. 42, No. 3. 2005, pp 171-185.
5. W. Hackbusch and Steffen Börm, “$H^2$-matrix approximation of integral operators by interpolation”, Appl. Numer. Math., Vol. 43, No. 1–2, 2002, pp. 129-143.
6. A. Ida, T. Iwashita, T. Mifune and Y. Takahashi, “Parallel Hierarchical Matrices with Adaptive Cross Approximation on Symmetric Multiprocessing Clusters”, JIP, Vol. 22, 2014, pp. 642-650.
7. W. J. Bolosky, M. L. Scott, R. P. Fitzgerald, R. J. Fowler and A. L. Cox, “NUMA policies and their relation to memory architecture”, ASPLOS-IV, 1991, pp. 212-221.
8. Henk A. van der Vorst, “Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems”, SIAM J. Sci. Stat. Comp., Vol. 13, No. 2, 1992, pp.631–644.
9. S. L. Zhang, “GPBi-CG: Generalized Product-type Methods Based on Bi-CG for Solving Nonsymmetric Linear Systems”, SIAM J. Sci. Comput., Vol. 18, No. 2, 1997, pp. 537-551.
10. Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM 2003.
11. T. Kuwabara and T. Takeda, “Boundary Element Method Using Analytical Integration for Three-Dimensional Laplace Problem”, Electrical Engineering in Japan, Vol. 106, No. 6, 1986, pp. 25-31.
12. H. Okuda, K. Nakajima, M. Iizuka, L. Chen, H. Nakamura, “Parallel Finite Element Analysis Platform for the Earth Simulator: GeoFEM”, ICCS 2003, LNCS, Vol. 2659, 2003, pp. 773-780.
13. http://hpcmw.tokyo.rist.or.jp/index_en.html
14. http://www.freefem.org/
15. http://www.bu.edu/exafmm/
16. C. Cecka and S. Layton, “FMMTL: FMM Template Library A Generalized Framework for Kernel Matrices”, ENUMATH 2013, LNCSE, Vol. 103, 2015, pp. 611-620.
17. R. Yokota, “An FMM Based on Dual Tree Traversal for Many-Core Architectures”, J. Algorithms Comput. Technol., Vol. 7, 2013, pp. 301-324.

18. http://www.hlibpro.com/
19. W. Śmigaj, S. Arridge, T. Betcke, J. Phillips and M. Schweiger, “Solving Boundary Integral Problems with BEM++”, ACM Trans. Math. Software, Vol. 41, 2015, pp. 6:1–6:40.
20. http://ppopenhpc.cc.u-tokyo.ac.jp/ppopenhpc/