ppOpen-HPC:
Open Source Infrastructure for Development and Execution of Large-Scale Scientific Applications on Post-Peta-Scale Supercomputers with Automatic Tuning (AT).

ppOpen-APPL/BEM

ppohBEM

ver. 0.5.0

User's guide

Contents

1. Overview

ppOpen-APPL/BEM is software application supporting a boundary element analysis conducted on a parallel computer. It consists of the BEM-BB framework, the BEB-BB template and the distributed parallel H-matrix library, HACApK. Fig. 1 shows the overview of the components of ppOpen-APPL/BEM software package.

The BEM-BB framework supports general three dimensional boundary element analyses based on dense matrix operations on parallel computers. It provides users with distribution of input model data to processes and parallelized linear iterative solvers. An important caveat for the BEM-BB framework is that the framework is not a complete simulation program. Users must give user functions describing each element of the coefficient matrix and the right-hand side vector, or use BEB-BB templates providing the function in a specific problem domain. In some cases, users may modify the framework program by themselves.

The parallel H-matrix library HACApK can be used with or without the BEM-BB framework. Some applications might use only HACApK. The manual for the HACApK and the sample code of the BEM-BB framework (H-matrix) is available in the directory of hacapk.

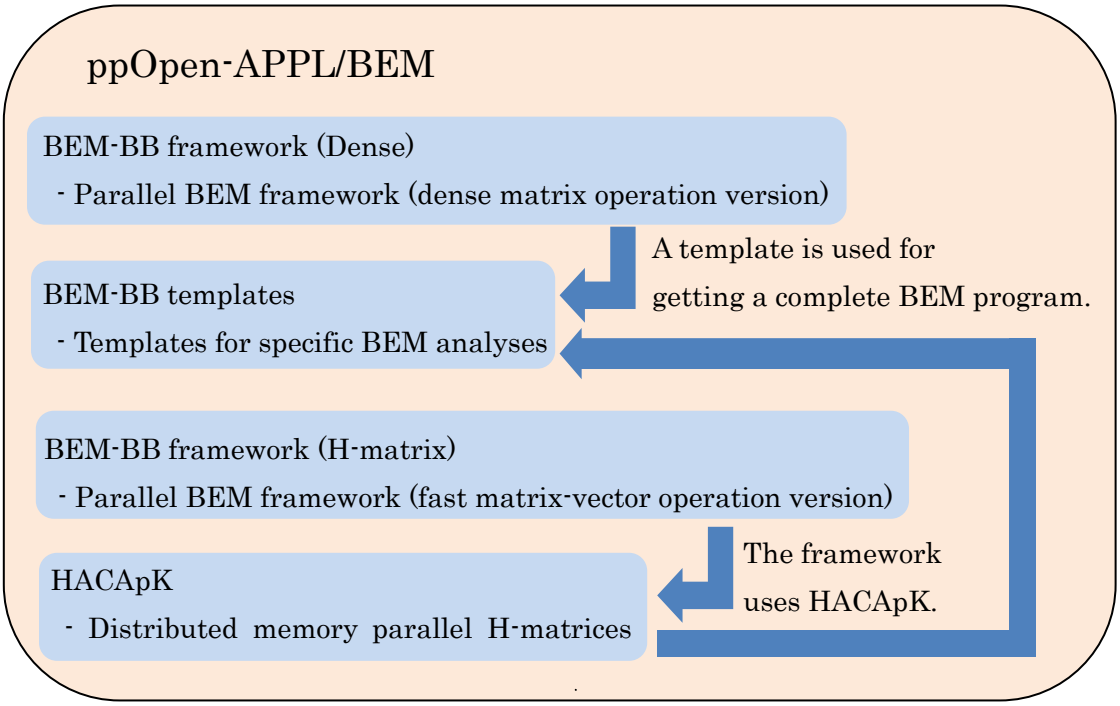

Fig. 1 Diagram of software components of ppOpen-APPL/BEM

2. BEM-BB framework (dense)

2-1 Features

A typical boundary element analysis consists of 5 steps: 1. Reading model data from files, 2. Construction of the coefficient matrix based on integral operations between boundary elements, 3. Generation of the right-hand side vector and setting of boundary conditions, 4. Solving the linear system of equations, and 5. Output of the result to files. Fig. 1 shows the simulation flow of a boundary element analysis. The BEM-BB framework (dense) deals with general three-dimensional boundary element analyses, in which dense matrix operations are used. In other words, the coefficient matrix is given by a dense matrix and the integral operations are conducted between all pairs of the boundary elements. In our software, the framework only supports steps 1 and 4. This is because steps 3 and 4 cannot be supported if the problem domain, the type of elements, and the integral method are not specified. Accordingly, the functions required for steps 3 and 4 are supported in the form of BEM-BB templates. For further details on the templates, please see section 3.

As mentioned above, the framework only provides users with functions involving model data inputs and linear solvers. However, these functions have been parallelized by using the hybrid parallel programming model, in which multiple processes and threads are used. Accordingly, the framework software is expected to reduce the burden on users for annoying parallel programming required for a boundary element analysis.

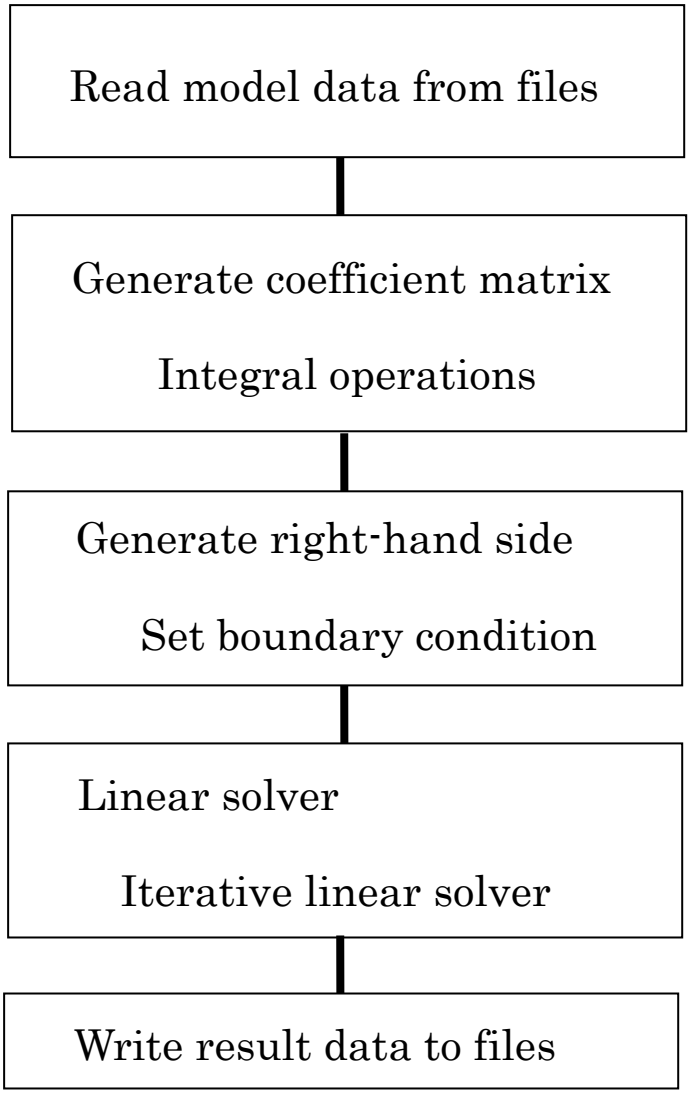

Fig. 1. A typical flow of the simulation for boundary element analysis

2-2 How to use the BEM-BB framework (dense)

First, users should download and put all files for the BEM-BB (dense) application in an arbitrary directory. Next, bem-bb-config.txt must be appropriately edited. However, users can use the default settings of the configuration file. Next, users should provide the user functions describing how the coefficient matrix and the right hand side vector are generated. Finally, the framework program must be modified to address the user's goals for the simulation. For example, the function to be used for outputting the result is expected to be provided.

2-2-1 bem-bb-config.txt

The bem-bb-config.txt describes the parameters for the BEM simulation. It includes

four variables:

ppohBEM_number_element_dof (integer*4)

ppohBEM_linear_solver (character*16)

ppohBEM_tor (real*8)

ppohBEM_max_steps (integer*4)

The variable “ppohBEM_number_element_dof” gives the number of unknowns set on each (face) element. The total number of unknowns is given by the number of elements (faces) multiplied by ppohBEM_number_element_dof. The linear solver used is determined by “ppohBEM_linear_solver”. It must be noted that only BICGSTAB has been implemented. The value of “ppohBEM_tor” gives the convergence criterion for the linear solver. When the relative residual norm is less than “ppohBEM_tor”, the iteration is terminated. The variable ppohBEM_max_steps determines the maximum number of iteration counts. Currently, the default setting of the BEM-BB-config.txt is given as follows:

1

BICGSTAB

1d-8

5000

2-2-2 User functions

In order to use the BEM-BB (dense) application, the user must provide two functions which yield the coefficient matrix and the right-hand side vector. The function for the coefficient matrix has the following form:

```
real(8) function ppohBEM_matrix_element_ij( i, j, ppohBEM_nond, &
        ppohBEM_nofc, ppohBEM_nond_on_face,st_ppohBEM_np, &
        ppohBEM_int_para_fc, ppohBEM_nint_para_fc, ppohBEM_dble_para_fc, &
        ppohBEM_ndble_para_fc, ppohBEM_face2node )
        type :: coordinate
              real(8) :: x ,y ,z
        end type coordinate
        integer ,intent(in) :: i, j, ppohBEM_nond, ppohBEM_nofc, &
                                ppohBEM_nond_on_fc, ppohBEM_nint_para_fc, &
                                ppohBEM_ndble_para_fc
        type(coordinate), intent(in) :: st_ppohBEM_np(*)
        integer, intent(in) :: ppohBEM_face2node(ppohBEM_nond_on_fc, *), &
                                ppohBEM_int_para_fc(ppohBEM_nint_para_fc, *)
        real(8), intent(in) :: ppohBEM_dble_para_fc(ppohBEM_dble_para_fc, *)
```

The return value of the function should be the i-th row j-th column element of the coefficient matrix. In the function, the value of the element is calculated by using the above arrays and variables. The integer variables “ppohBEM_nond” and “ppohBEM_nofc” designate the number of nodes and faces (boundary elements), respectively. The structure type variable “st_ppohBEM_np” gives the three dimensional coordinate value of the node. The variables st_ppohBEM_np(i)%x, st_ppohBEM_np(i)%y, st_ppohBEM_np(i)%z preserve the x- y- z- coordinates of the i-th node. The integer array ppohBEM_face2node designates three nodes to conform to each face. The integer value ppohBEM_nond_on_fc gives the number of nodes on a face element. The values of ppohBEM_face2node(k,i), (k=1,2,..., ppohBEM_nond_on_fc) mean the number of nodes conforming to the i-th face. The two dimensional arrays ppohBEM_int_para_fc and ppohBEM_dble_para_fc are the integer and real (double precision) parameters set on a face, and are determined through the model data input depending on the user’s requirements. The number of integer parameters set on a face is given by ppohBEM_nint_para_fc. The number of real parameters on a face is given by ppohBEM_ndble_para_fc. For example, the first integer parameter of the i-th face can be referred to by using ppohBEM_int_para_fc(1,i). In the model data, users explicitly give the values for ppohBEM_nint_para_fc, ppohBEM_ndble_para_fc, ppohBEM_int_para_fc, and ppohBEM_dble_para_fc. If there is no need to set any parameters on faces in an application, ppohBEM_nint_para_fc and ppohBEM_ndble_para_fc must be set to 0. (For further information, please refer to section 2-2-3 describing the model data input.)

The function for the right-hand side has a form similar to that of the coefficient matrix, as follows:

```
real(8) function ppohBEM_right_hand_side_vector_element_i( i, &
                 ppohBEM_nond, ppohBEM_nofc, ppohBEM_nond_on_face, &
                 st_ppohBEM_np, ppohBEM_int_para_fc, &
                 ppohBEM_nint_para_fc, ppohBEM_dble_para_fc, &
                 ppohBEM_ndble_para_fc, ppohBEM_face2node )
```

Here, we would note that the functions included in the BEM-BB templates can be used as a reference for programming the user functions.

2-2-3 Input data file for the analysis model

After appropriate modification of the BEM-BB (dense) program, an input model data file is prepared. The file name should be specified by the first argument of the execution command when the program is executed. If no file is specified, the file “input_sample.pbf” is used. The input data file should include the following items in

order.

(i) Number of nodes (1 integer*4 number): ppohBEM_nond

(ii) Coordinates of the nodes in three dimensions (3 real*8 numbers for each node)

(iii) Number of faces (1 integer*4 number): ppohBEM_nofc

(iv) Number of nodes on each face (1 integer*4 number): ppohBEM_nond_on_fc

(v) Number of integer parameters set on each face (1 integer*4 number): ppohBEM_nint_para_fc

(vi) Number of real parameters set on each face (1 integer*4 number): ppohBEM_ndble_para_fc

(vii) Node number constructing for each face (ppohBEM_nond_on_fc x ppohBEM_nofc integer*4 numbers)

(viii) Integer parameters set on faces (ppohBEM_ nofc x ppohBEM_nint_para_fc integer*4 numbers)

(ix) Real parameters set on faces (ppohBEM_ nofc x ppohBEM_ndble_para_fc real*8 numbers)

A sample set of model data is given as follows:

```
4
0.000000000000e+00     0.000000000000e+00     0.000000000000e+00
0.000000000000e+00     1.000000000000e+00     0.000000000000e+00
1.000000000000e+00     1.000000000000e+00     0.000000000000e+00
2.000000000000e+00     0.000000000000e+00     2.000000000000e+01
2
3
0
1
1 2 3
2 4 3
0.000000000000e+00
1.000000000000e+00
```

In this example, the sample model consists of 2 triangular face elements and it includes 4 nodes. The model has no integer parameter and one real parameter for each face. In this example, this real parameter is used for generating the right hand side vector. A user can control the number of parameters on faces by using ppohBEM_nint_para_fc and ppohBEM_ndble_para_fc.

In the default setting, the software reads data in the ASCII format file. However, a user can use the binary format for the file if she/he modifies the framework program.

2-3 Parallelization method

In this subsection, the parallelization method used in the BEM-BB framework (dense) application is explained. In this framework, all processes hold a copy of the model data file. This is because the data size of the model data is significantly less than the memory requirement for the dense coefficient matrix. Moreover, the valuables and arrays for the model data are shared among threads. Consequently, a user can use all model data in user functions. The copy of the model data to the processes is performed by an MPI collective communication function.

In the step for the coefficient matrix generation, each thread (or process) calls the ppohBEM_matrix_element_ij function independently, without MPI communications. Accordingly, a high level of parallel efficiency is expected in this step. The data distribution of the coefficient matrix to processes is quite simple. Each process holds one of the row blocks of the matrix. In each process, the data for the row block will be stored in a shared array. However, for the efficient use of the NUMA memory architecture, the array is first touched by multiple threads.

In the iterative solver, the process is fully parallelized in processes and threads. Although vectors are shared by the processes, the inner product, the matrix vector multiplication, and the vector updates are performed by threads (processes) in parallel.

3 BEM-BB templates

3-1 How to use templates with the BEM-BB framework

In the BEM-BB template directory, a modified version of the BEM-BB framework program is provided. Users can run the simulation after they put the model data file in the same directory. If integer or real model parameters are used in the template, the details are described in each subsection for the template.

3-2 BEM-BB template (SCM)

The BEM-BB template (SCM) is used for the surface charge method in a static electric field problem. The details of the problem supported by the template are as follows. In addition, how the problem is solved by the template is also addressed briefly.

The template solves a three-dimensional static field problem on free space:

$$\nabla^2\phi = 0,$$

$$\phi = \phi_\Gamma \text{ on } \Gamma.$$

The electric scalar potential is given by

$$\phi(\boldsymbol{x}) = \int_\Gamma G(\boldsymbol{x}, \boldsymbol{x}')\frac{\sigma(\boldsymbol{x}')}{\epsilon_0}dS',$$

where ϵ_0 is the permittivity, σ is the surface charge density which appears on Γ, and

G is the green function given by

$$G(\boldsymbol{x}, \boldsymbol{x}') = 1/4\pi|\boldsymbol{x} - \boldsymbol{x}'|.$$

When the step-wise triangular face element is used based on the surface division $\Gamma = \sum S_i$, we obtain the following linear system of equations:

$$A\boldsymbol{u} = \boldsymbol{b},$$

$$A_{ij} = \frac{1}{\epsilon_0}\int_{S_j} G(\boldsymbol{x}_i, \boldsymbol{x}')dS',$$

$$b_i = \phi_\Gamma(\boldsymbol{x}_i),$$

where $\boldsymbol{x}_i$ is the center of balance of the triangle S_i. In this template, the integration

$$I_{ij} = \int_{S_j} \frac{1}{|\boldsymbol{x}_i - \boldsymbol{x}'|} dS',$$

which is required to determine the element of the coefficient matrix, is calculated by using the analytical method reported in [1].

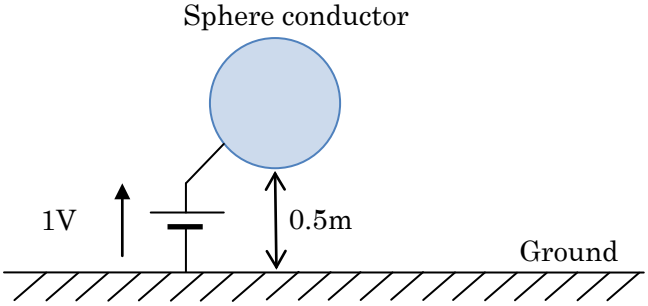

Fig. 2. Test problem 1

4 Performance evaluations

4-1 Example 1

Fig. 2 shows the model analyzed in test problem #1. A sphere conductor is set in a space with a distance of 0.5 m from its ground. The radius of the conductor is 0.25 m. The conductor is charged satisfying the electric potential of 1 V. We calculate the surface charge induced on the surface of the conductor by means of the boundary element method, and then the distribution of the electric potential in the space is determined.

In the model analyzed, we add another sphere conductor to keep the potential of the ground surface 0 V. The surfaces of the two conductors are discretized with 21600 triangular face elements. In other words, the dof of the problem is given by 21600. The basic equation is given by Laplace's equation. The problem is solved using the BEM-BB framework (dense) application with the BEM-BB (SCM) template.

The parallel computer used for the analysis is the T2K Open Supercomputer at Kyoto

University [2]. It is a cluster of multi-processor nodes, each of which consists of four AMD Opteron 8356 (2.3GHz) CPUs and 32 GB of memory (DDR2-667). The internal network between the nodes is based on InfiniBand DDR x 4. The Fujitsu Fortran compiler 3.2 and the Fujitsu MPI library 3.0 were used.

Fig. 3 depicts the surface charge calculated. Fig. 4 shows the parallel speedup ratio on the T2K Open Supercomputer.

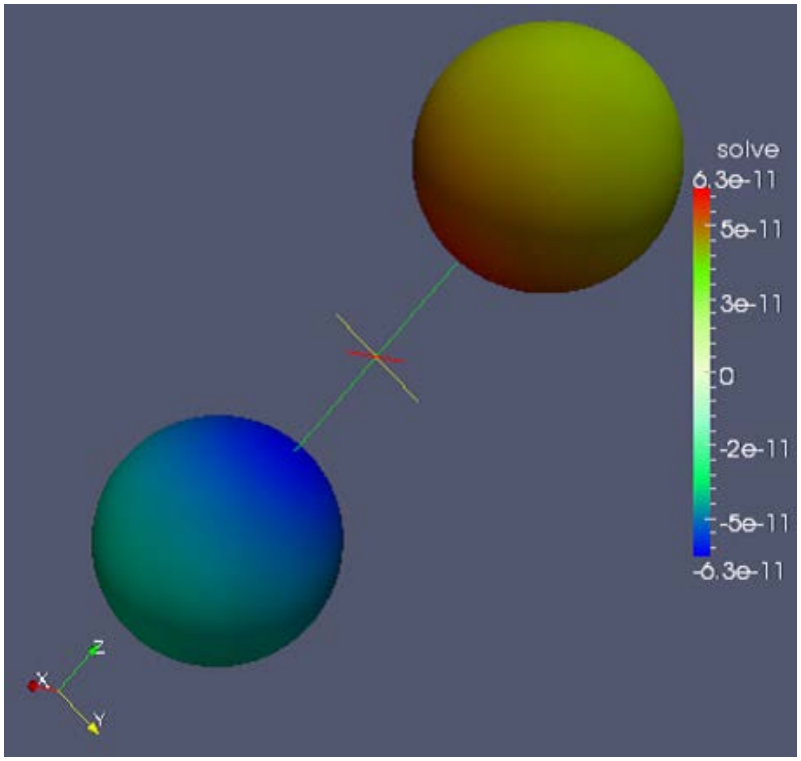

Fig. 3 Distribution of surface charge

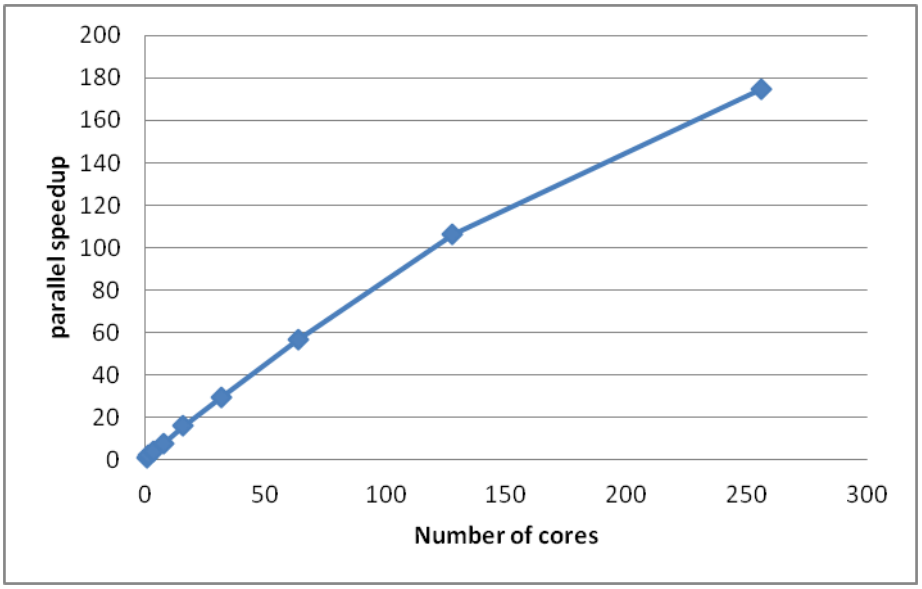

Fig. 4 Parallel speedup on the T2K Open Supercomputer

5 References

[1] T. Kuwabara and T. Takeda, “Boundary Element Method Using Analytical Integration for Three-Dimensional Laplace Problem”, Electrical Engineering in Japan, Vol. 106, No. 6, pp. 25-31, (1986).

[2] H. Nakashima, “T2K open supercomputer: inter university and interdisciplinary collaboration on the new generation supercomputer”, Proc. Int. Conf. Informatics Education and Research for Knowledge-Circulating Society, pp. 137-142, (2008).

6 Contact address

Inquiries about ppOpen-APPL/BEM and BEM-BB software components can be sent to Dr. Takeshi Iwashita (Information Initiative Center, Hokkaido University, Japan) via e-mail. His e-mail address is iwashita@iic.hokudai.ac.jp.