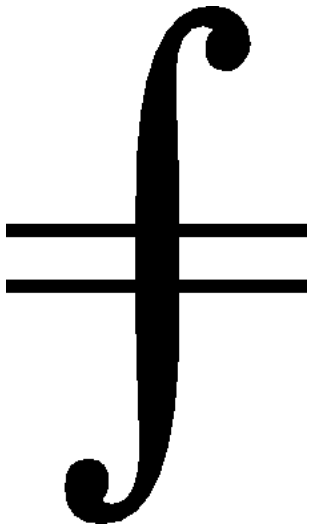
IntETec

CBEM_LAP



Sylvain Nintcheu Fata

June 2012

This guide is for CBEM_LAP (version 1.3, June 2012)

Contents

1 Introduction

This guide describes version 1.3 of CBEM_LAP, a Collocation Boundary Element Method (CBEM) for solving the Laplace equation in 3D. CBEM_LAP is an Integral Equation Technology (IntETec) package for solving the Laplace equation in 3D domains via a collocation Boundary Element Method (BEM). Currently, CBEM_LAP can handle 3D polyhedral domains with surface meshes composed only of flat triangles. In addition, the computational treatment employs piecewise ***constant*** approximations for field variables defined over a surface triangle. The implementation of CBEM_LAP is entirely based on the analytic formulae for surface potentials explicitly provided in [6]. For a general overview on the numerical solution of the Laplace equation via the Boundary Integral Equation (BIE) method, interested readers should consult references [1–3].

The computer routines for this package are available separately in C and Fortran 90. Therefore, a C compiler or a Fortran 90 compiler is required to obtain an executable. Moreover, BLAS (`http://www.netlib.org/blas`) and LAPACK (`http://www.netlib.org/lapack`) routines, and the iterative solver BiCGSTAB(l)[9] with a general-purpose sparse preconditioner for BEM [7] are needed for the solution of the discretized linear system. CBEM_LAP contains a driver routine for the complete solution of a boundary-value problem associated with the 3D Laplace equation.

2 Solution of the Laplace equation via CBEM_LAP

To solve the Laplace equation

$$\nabla^2 u = 0 \tag{1}$$

in a bounded domain $\Omega \subset \mathbb{R}^3$ with boundary Γ, the BIE method uses the Green's representation formula [2, 4, 5] expressed as

$$\int_\Gamma G(\boldsymbol{x},\boldsymbol{y})\, t(\boldsymbol{y})\, \mathrm{d}\Gamma_{\boldsymbol{y}} \; - \int_\Gamma \boldsymbol{H}(\boldsymbol{x},\boldsymbol{y})\cdot\boldsymbol{n}(\boldsymbol{y})\, u(\boldsymbol{y})\, \mathrm{d}\Gamma_{\boldsymbol{y}} = \begin{cases} u(\boldsymbol{x}), & \boldsymbol{x}\in\Omega \\ 0, & \boldsymbol{x}\in\mathbb{R}^3\setminus\overline{\Omega} \end{cases}, \tag{2}$$

where $\overline{\Omega}=\Omega\cup\Gamma$, and the kernels G and $\boldsymbol{H}$ are given respectively by

$$G(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{4\pi}\frac{1}{\|\boldsymbol{x}-\boldsymbol{y}\|}, \quad \boldsymbol{H}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{4\pi}\frac{\boldsymbol{x}-\boldsymbol{y}}{\|\boldsymbol{x}-\boldsymbol{y}\|^3}, \quad \boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^3, \quad \boldsymbol{x}\neq\boldsymbol{y}. \tag{3}$$

In (2), $\boldsymbol{n}$ is the unit normal to Γ directed towards the *exterior* of Ω and $t = \partial u/\partial n$ is the flux associated with the potential u. One can see from (2) that solving for u in Ω reduces to finding u and t on Γ. To this end, let $\boldsymbol{x}_\varepsilon \in \mathbb{R}^3\setminus\overline{\Omega}$. The potential u and flux t on Γ can be determined by solving the singular BIE

$$\lim_{\boldsymbol{x}_\varepsilon\to\boldsymbol{x}\in\Gamma}\left(\int_\Gamma G(\boldsymbol{x}_\varepsilon,\boldsymbol{y})\, t(\boldsymbol{y})\, \mathrm{d}\Gamma_{\boldsymbol{y}} \; - \int_\Gamma \boldsymbol{H}(\boldsymbol{x}_\varepsilon,\boldsymbol{y})\cdot\boldsymbol{n}(\boldsymbol{y})\, u(\boldsymbol{y})\, \mathrm{d}\Gamma_{\boldsymbol{y}}\right) = 0. \tag{4}$$

Note that in (4), $\boldsymbol{x}_\varepsilon$ approaches the boundary Γ from outside the domain Ω. The integral statement expressed in (4) corresponds to the so-called *limit to the boundary* approach.

2.1 Numerical approximation

To deal with (4), assume that (i) $\Gamma = \bigcup \overline{\Gamma}_q$ can be triangulated into closed and non-overlapping surface elements such that Γ_q is an open flat triangle (see Fig. 1), and (ii) on each element (triangle) u and t are ***constants***. Let N be the total number of boundary elements on Γ. With these

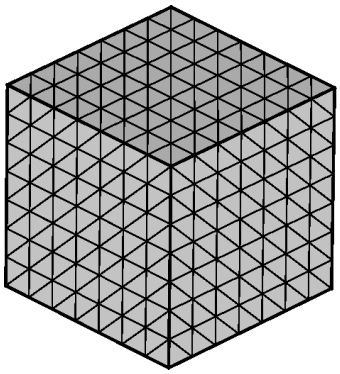


Figure 1: Triangulation of a standard cube using 588 triangles and 384 nodes.

assumptions, a collocation approach for resolving (4) requires that the singular BIE be satisfied exactly at a set of collocation points $\{\boldsymbol{x}^i\}_{i=1}^N$ resting on Γ. This requirement leads to a dense linear system of algebraic equations for boundary potential u and flux t as

$$\mathbf{G}\{t\} = \mathbf{H}\{u\}, \tag{5}$$

where $\{u\}$ and $\{t\}$ are respectively vectors containing potentials u^j and fluxes t^j on each boundary element Γ_j $(j=1,2,\ldots,N)$; components of influence matrices $\mathbf{G}$ and $\mathbf{H}$ can be written as

$$\mathrm{G}_{ij} = \lim_{\boldsymbol{x}_\varepsilon \to \boldsymbol{x}^i} g_j(\boldsymbol{x}_\varepsilon), \quad \mathrm{H}_{ij} = \lim_{\boldsymbol{x}_\varepsilon \to \boldsymbol{x}^i} h_j(\boldsymbol{x}_\varepsilon), \quad \boldsymbol{x}^i \in \Gamma, \tag{6}$$

with the single-layer potential g_j and double-layer potential h_j expressed as

$$g_j(\boldsymbol{x}) = \int_{\Gamma_j} G(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\Gamma_{\boldsymbol{y}}, \quad h_j(\boldsymbol{x}) = \int_{\Gamma_j} \boldsymbol{H}(\boldsymbol{x},\boldsymbol{y})\cdot\boldsymbol{n}(\boldsymbol{y})\,\mathrm{d}\Gamma_{\boldsymbol{y}}, \quad \boldsymbol{x}\in\mathbb{R}^3. \tag{7}$$

For mathematical consistency, u^j and t^j $(j=1,2,\ldots,N)$ are assumed to be potential and flux at the *centroid* of the boundary element Γ_j respectively. Upon prescribing the boundary conditions for the specific boundary-value problem associated with (1), the linear system (5) can be rearranged as

$$\mathbf{A}\{z\} = \{b\}, \tag{8}$$

where $\{z\}\in\mathbb{R}^N$ is a vector containing unknown potentials or fluxes on Γ, and $\{b\}\in\mathbb{R}^N$ is a vector whose entries are obtained from known boundary data. For an arbitrary source point $\boldsymbol{x}\in\mathbb{R}^3$, surface potentials $g_j(\boldsymbol{x})$ and $h_j(\boldsymbol{x})$ are calculated *exactly* via recursive expressions provided in [6].

The solution of the linear system (8) together with the specified boundary conditions complete the task of finding u and t on the entire boundary Γ. Finally, the remaining exercise of computing

the potential $u(\boldsymbol{x})$ at an arbitrary interior point $\boldsymbol{x}$ in Ω can be accomplished by use of (2) and (7) as

$$u(\boldsymbol{x}) = \sum_{j=1}^{N} t^j \, g_j(\boldsymbol{x}) \, - \sum_{j=1}^{N} u^j \, h_j(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega. \tag{9}$$

Note that the numerical analysis presented herein is applicable to the Dirichlet, Neumann and mixed boundary conditions.

3 How to use CBEM_LAP

To use CBEM_LAP, simply provide the *triangulated surface mesh* and *boundary conditions* in the file "bndata.dat" and invoke the routine `cbem_lap_srf_eval` in your code to solve the singular BIE (4) for the unknown potential u and flux t on the boundary. In addition call the routine `cbem_lap_fld_eval` to calculate the potential at user-specified interior points, i.e., to evaluate (9). CBEM_LAP package is provided with a sample problem to illustrate the utilization of the code. The driver routine is in the file "lap_ex.f90" for the Fortran 90 package (or "lap_ex.c" for the C package). Also "flux_ex.dat" and "potl_ex.dat" are the corresponding output files for the boundary flux and potential respectively.

3.1 Getting started with CBEM_LAP

On a Linux system with a Fortran 90 compiler (a C compiler, gcc, is always available), BLAS and LAPACK installed, the prospective user of CBEM_LAP should (i) gunzip and tar the package cbem_lap-f-i-x.tgz (cbem_lap-c-i-x.tgz) in a directory of its choice (x is the version number), (ii) edit the file "makefile", and (iii) issue "make". The first step can be accomplished by typing "tar -zxvf cbem_lap-f-i-x.tgz" ("tar -zxvf cbem_lap-c-i-x.tgz") in the shell. In the second step, open "makefile" with a text editor and replace gfortran by the Fortran 90 compiler on your system (replace gcc by an ANSI C compiler of your choice). The third step simply generates an executable called `lap_ex`.

Once the executable `lap_ex` has been generated, the user must now provide in the input file "bndata.dat" (i) the triangulated surface mesh describing the geometry of the domain of interest, and (ii) the boundary condition for every surface element (triangle). In general, the input file has three sections. In the first section, the number of boundary nodes followed by the number of boundary elements are prescribed. The second section contains Cartesian coordinates of all boundary nodes. In the third section, element connectivity and boundary values (Dirichlet or Neumann) are prescribed for each boundary element.

The input file is best illustrated via an example. To this end, consider a mixed boundary-value problem for the Laplace equation in the standard cube $\Omega = \{(x, y, z) \in \mathbb{R}^3 : 0 < x, y, z < 1\}$. The potential $u|_\Gamma = e^x \sin z + e^z \cos y$ is given on the face $\{x = 0,\ 0 \le y, z \le 1\}$, and the flux $t = \boldsymbol{n} \cdot \nabla u$ is prescribed on the remaining faces, where $\nabla u = (e^x \sin z, -e^z \sin y, e^x \cos z + e^z \cos y)$. Here x, y, z represent the Cartesian coordinates of a point in the 3D space.

Listing 1: Sample input file for a mixed problem on a standard cube

```
 1          24
 2          12
 3     0.00000000      0.00000000      0.00000000
 4     0.00000000      1.00000000      0.00000000
 5     0.00000000      0.00000000      1.00000000
 6     0.00000000      1.00000000      1.00000000
 7     1.00000000      0.00000000      0.00000000
 8     1.00000000      1.00000000      0.00000000
 9     1.00000000      0.00000000      1.00000000
10     1.00000000      1.00000000      1.00000000
11     0.00000000      0.00000000      0.00000000
12     1.00000000      0.00000000      0.00000000
13     0.00000000      0.00000000      1.00000000
14     1.00000000      0.00000000      1.00000000
15     0.00000000      1.00000000      0.00000000
16     1.00000000      1.00000000      0.00000000
17     0.00000000      1.00000000      1.00000000
18     1.00000000      1.00000000      1.00000000
19     0.00000000      0.00000000      0.00000000
20     1.00000000      0.00000000      0.00000000
21     0.00000000      1.00000000      0.00000000
22     1.00000000      1.00000000      0.00000000
23     0.00000000      0.00000000      1.00000000
24     1.00000000      0.00000000      1.00000000
25     0.00000000      1.00000000      1.00000000
26     1.00000000      1.00000000      1.00000000
27      1       4       2        1.42398872   0
28      1       3       4        2.45889461   0
29      8       5       6        0.88940740   1
30      7       5       8        1.68090340   1
31     12       9      10        0.00000000   1
32     11       9      12        0.00000000   1
33     13      16      14       -1.17436736   1
34     13      15      16       -1.63896168   1
35     17      20      18       -2.89269099   1
36     17      19      20       -2.18149969   1
37     24      21      22        3.62102449   1
38     23      21      24        2.89031567   1
```

For the foregoing problem, the contents of the input file "bndata.dat" can be seen in Listing 1. The line numbers in the first column of the listing do not, in reality, exist in the file. It is included here to facilitate the presentation. On the first line, `24` represents the number of boundary nodes. On the second line, `12` specifies the number of boundary elements. Next, the x, y, z coordinates of all boundary nodes are given on line 3 through line 26. Namely, the x, y, z coordinates of the 1-st boundary node is given on line 3, the Cartesian coordinates of the 2-nd boundary node on line 4 and so on. In this manner, a global index is assigned to each boundary node in a sequential order at which they are entered in the input file. Next, the characteristics of each boundary element is specified from line 27 through line 38 via the format `i j k xxx m`. The triplet `i` $\rightarrow$ `j` $\rightarrow$ `k` represents the so-called element connectivity, where `i`, `j` and `k` are respectively the global indices of three boundary nodes that form together a flat triangle in $\mathbb{R}^3$. In other words, boundary node `i` is connected to node `j` which, in turn, is connected to node `k` defining in this order the orientation of the boundary element. This orientation may be specified in either clockwise or anti-clockwise when viewed in the direction of the *outward* normal. The real number `xxx` represents the prescribed boundary condition on element `i` $\rightarrow$ `j` $\rightarrow$ `k` (i.e. at the centroid of the flat triangle `i` $\rightarrow$ `j` $\rightarrow$ `k`). If `m` $= 0$, then the potential is given on the element. If `m` $= 1$ instead, then the flux is specified on element `i` $\rightarrow$ `j` $\rightarrow$ `k`.

It is also important to note that a global index is assigned to each boundary element in the order at which they are listed in the input file. Namely, the characteristics of the 1-st boundary element is given on line 27, the attributes of the 2-nd element on line 28 and so on.

Since in C, array elements are indexed beginning with 0, the listing from line 27 through line 38 for the C package cbem_lap-c-i-x.tgz will look like this:

```
27      0      3      1       1.42398872   0
28      0      2      3       2.45889461   0
29      7      4      5       0.88940740   1
30      6      4      7       1.68090340   1
31     11      8      9       0.00000000   1
32     10      8     11       0.00000000   1
33     12     15     13      -1.17436736   1
34     12     14     15      -1.63896168   1
35     16     19     17      -2.89269099   1
36     16     18     19      -2.18149969   1
37     23     20     21       3.62102449   1
38     22     20     23       2.89031567   1
```

Note that the node numbers `i`, `j` and `k`, indeed, range from 0 through 23.

In CBEM_LAP, the input data are read into the code via the routine `cbem_lap_input` in the file "cbem_lap_input.f90" ("cbem_lap_input.c"). For the sample problem provided in this package, the file "bndata.dat" contains 384 boundary nodes and 588 elements on a standard cube.

4 Structure of CBEM_LAP

CBEM_LAP is provided with a driver routine to solve a complete Laplace problem, computational routines to perform basic BEM tasks, and auxiliary routines to accomplish specific subtasks. Common to all BEM program, the computational subroutines of CBEM_LAP are listed below along

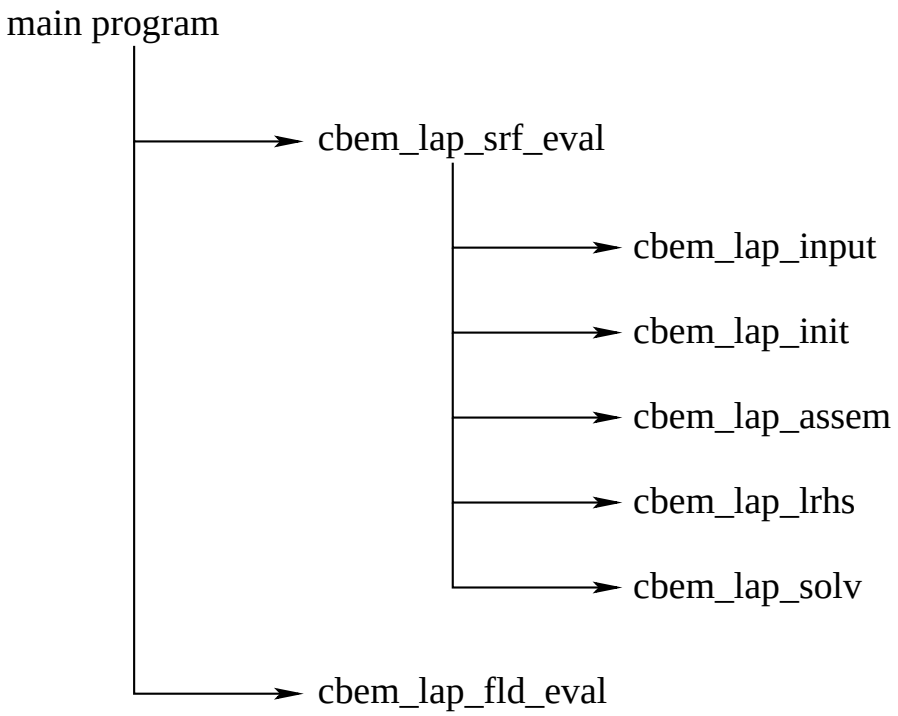


Figure 2: Structure of a BEM program.

with a brief description of the purpose of each routine:

Routine	**Description**
`cbem_lap_srf_eval`	Solve the Singular BIE for the Laplace equation
`cbem_lap_input`	Read input data and parameters
`cbem_lap_init`	Initialize arrays and parameters
`cbem_lap_assem`	Assemble **H** and **G** influence matrices
`cbem_lap_lrhs`	Form the left- and right-hand sides of the discretized BIE
`cbem_lap_solv`	Solve the Discretized BIE
`cbem_lap_fld_eval`	Evaluate potential at interior points

The acronym CBEM stands for Collocation Boundary Element Method. Fig. 2 illustrates the structure of the driver routine (see "lap_ex.f90" or "lap_ex.c") provided in this package. As can be seen from the figure, the main program (i.e. the driver routine) simply invokes `cbem_lap_srf_eval` to compute the potential and flux at the centroid of each boundary element, and `cbem_lap_fld_eval` to calculate the potential at a set of prescribed interior points. The source code for an individ-

ual routine listed in Fig. 2 can be found in a file with corresponding name. For instance, the routine `cbem_lap_srf_eval` can be found in "cbem_lap_srf_eval.f90" for the Fortran 90 package (or "cbem_lap_srf_eval.c" for the C package). In addition, the file "cbem_util.f90" ("cbem_util.c") contains useful routines needed for a successful implementation of a collocation boundary element code. For a detailed description of a specific CBEM_LAP routine, interested users (especially software developers) should consult the source code.

Since `cbem_lap_solv` employs the iterative solver BiCGSTAB(l)[9] and a general-purpose sparse preconditioner for BEM [7] to solve the discretized linear system (8), it is important to further describe this routine.

4.1 Description of `cbem_lap_solv`

To maintain an $O(N^2)$ algorithm, the Bi-Conjugate Gradient Stabilized (BiCGSTAB(l)) method [9] will be utilized to solve the discretized BIE (8) iteratively. In addition, since linear systems arising from BEM approximations are often ill-conditioned especially when dealing with mixed boundary-value problems defined on Lipschitz domains, a preconditioner is necessary to effectively accomplish the foregoing task. As elucidated in [7], a left preconditioner for the fully-populated linear system (8) is a non-singular matrix $\mathbf{P}$ such that $\mathbf{P}^{-1}\mathbf{A}$ is better conditioned than the original matrix $\mathbf{A}$. In practice, $\mathbf{P}$ is often constructed as a sparse matrix so that the cost of generating and storing $\mathbf{P}^{-1}$, and solving the system $\mathbf{P}\{x\} = \{y\}$ scales linearly with the number of boundary unknowns N.

To extract such a sparse preconditioner $\mathbf{P}$ from $\mathbf{A}$, an auxiliary axis-parallel box containing the discretized boundary Γ of the domain Ω is subdivided into parallelepipeds called cells. Next, boundary elements on Γ are distributed *without repetition* into cells. Further, every non-empty cell (i.e. cell containing boundary elements) is assigned a set of non-empty *near-neighboring* cells. With these preliminaries, the sparsity pattern of $\mathbf{P}$ is defined by a user-specified parameter $L_c = 1 + n_c$, where n_c is the number of near-neighboring cells (if any) used in the actual construction of $\mathbf{P}$. Namely, if $L_c = 1$, the entries of $\mathbf{P}$ are gathered from those a_{ij} of $\mathbf{A}$ obtained only from the cell self-interactions. All other interactions between boundary elements are simply ignored. This situation will generate a sparse approximation $\mathbf{P}$ to the matrix $\mathbf{A}$ that contains the dominant part of $\mathbf{A}$ and has the smallest number of non-zero entries. For any other value L_c, the preconditioner $\mathbf{P}$ is constructed from the cell self-interactions plus the interaction of each non-empty cell and its n_c nearest neighbors. Once $\mathbf{P}$ has been generated, an incomplete LU factorization with no fill-ins (ILU(0)) [8] is further employed to indirectly compute an approximate inverse $\mathbf{P}^{-1}$.

In `cbem_lap_solv`, if the sparsity flag $L_c \leq 0$, the iterative solver BiCGSTAB(l) given by the routine `bicgstabl` (see "bicgstabl.f90" or "bicgstabl.c") runs without preconditioner. If however $L_c \geq 1$, the sparsity pattern of $\mathbf{P}$ is determined by calling (i) the routine `cbem_cell_init` to build and initialize the axis-parallel cells, and (ii) the routine `cbem_cell_neighbor` to generate the list of near-neighbors for all non-empty cells. Next, the preconditioner $\mathbf{P}$ is formed and an ILU(0) is accomplished by invoking the routine `csr_ilu0`. After these steps, the iterative solver `bicgstabl` is called to solve the discretized BIE (8) using the constructed preconditioner.

References

[1] P. K. Banerjee. *The Boundary Element Methods in Engineering*. McGraw-Hill, London, 1994.

[2] M. Bonnet. *Boundary Integral Equation Methods for Solids and Fluids*. Wiley & Sons, New York, 1995.

[3] C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel. *Boundary Element Techniques*. Springer Verlag, Berlin, 1984.

[4] R. Kress. *Linear Integral Equations*. Springer-Verlag, New York, 1999.

[5] S. Nintcheu Fata. Fast Galerkin BEM for 3D-potential theory. *Comput. Mech.*, 42(3):417–429, 2008.

[6] S. Nintcheu Fata. Explicit expressions for 3D boundary integrals in potential theory. *Int. J. Numer. Meth. Eng.*, 78(1):32–47, 2009.

[7] S. Nintcheu Fata and L. J. Gray. On the implementation of 3D Galerkin boundary integral equations. *Eng. Anal. Boundary Elem.*, 34(1):60–65, 2010.

[8] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2003.

[9] G. L. G. Sleijpen and D. R. Fokkema. BiCGSTAB(l) for linear equations involving unsymmetric matrices with complex spectrum. *ETNA*, 1:11–32, 1993.