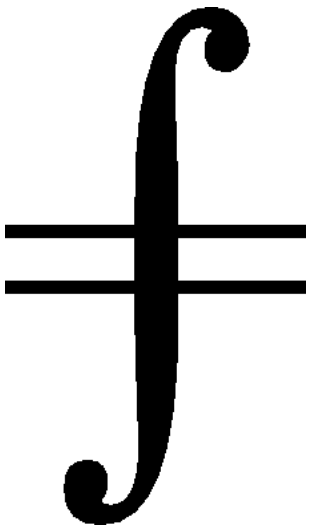
IntETec

CBEM_POI



Sylvain Nintcheu Fata

August 2012

This guide is for CBEM_POI (version 1.1, August 2012)

Contents

1 Introduction

This guide describes version 1.1 of CBEM_POI, a Collocation Boundary Element Method (CBEM) for solving the Poisson equation in 3D. CBEM_POI is an Integral Equation Technology (IntETec) package for solving the Poisson equation in 3D domains via a collocation Boundary Element Method (BEM). Currently, CBEM_POI 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_POI is based on the analytic formulae for surface potentials explicitly provided in [6], and on the treatment of the Newton potential elucidated in [7]. For a general overview on the discretization of the Poisson equation using 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, LAPACK routines (`http://www.netlib.org/lapack`) and BLAS routines (`http://www.netlib.org/blas`) are needed for the solution of the discretized linear system. CBEM_POI contains a driver routine for the complete solution of a boundary-value problem associated with the 3D Poisson equation.

2 Solution of the Poisson equation via CBEM_POI

To solve the Poisson equation

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

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

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

where $\overline{\Omega} = \Omega \cup \Gamma$, and the non-trivial term b represents a continuous source function prescribed on $\overline{\Omega}$. The kernels G and $\boldsymbol{H}$ are given respectively by

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

Since the Poisson equation (1) naturally occurs, for example, in the problem of finding the electrostatic *potential* of an electric field in a region of continuously distributed charges, the solution u of (1) will simply be called potential. 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 field variable 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

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

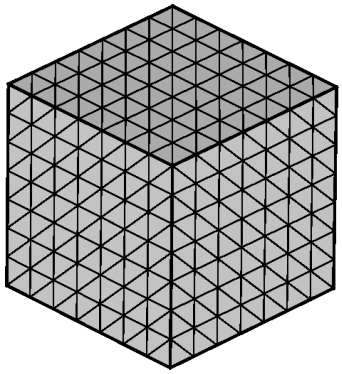


Figure 1: Triangulation of a standard cube using 588 triangles and 384 nodes.

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.

For a successful treatment of the singular BIE (4) with boundary-only discretization, it is useful to introduce the Newton potential as

$$\mathrm{V}(\boldsymbol{x}) = \int_\Omega G(\boldsymbol{x},\boldsymbol{y})\, b(\boldsymbol{y})\, \mathrm{d}\Omega_{\boldsymbol{y}}, \quad \boldsymbol{x}\in\mathbb{R}^3. \tag{5}$$

Now, one can invoke the following result established in [7] stating that the Newton potential (5) admits the representation

$$\mathrm{V}(\boldsymbol{x}) = \frac{1}{4\pi}\int_\Gamma \frac{(\boldsymbol{y}-\boldsymbol{x})\cdot\boldsymbol{n}(\boldsymbol{y})}{\|\boldsymbol{y}-\boldsymbol{x}\|}\, \mathcal{F}(\boldsymbol{x},\boldsymbol{y})\, \mathrm{d}\Gamma_{\boldsymbol{y}}, \quad \boldsymbol{x}\in\mathbb{R}^3, \tag{6}$$

where

$$\mathcal{F}(\boldsymbol{x},\boldsymbol{y}) = \int_0^1 z\, h(\boldsymbol{x}+z\,(\boldsymbol{y}-\boldsymbol{x}))\, dz, \quad \boldsymbol{y}\in\overline{\Omega} \tag{7}$$

with h denoting an extension of the source function b into any ball centered at $\boldsymbol{x}$ and containing $\overline{\Omega}$. In particular, a continuation h of the source function b can be specified as

$$h(\boldsymbol{x}) = \begin{cases} b(\boldsymbol{x}), & \boldsymbol{x}\in\overline{\Omega}, \\ 0, & \boldsymbol{x}\in\mathbb{R}^3\setminus\overline{\Omega}. \end{cases}$$

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 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 unknowns u and t as

$$\mathbf{G}\{t\} - \mathbf{H}\{u\} = -\{B\}, \tag{8}$$

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

$$\mathrm{G}_{ij} = \lim_{\boldsymbol{x}_\varepsilon \to \boldsymbol{x}^i} g_j(\boldsymbol{x}_\varepsilon), \quad \mathrm{H}_{ij} = \lim_{\boldsymbol{x}_\varepsilon \to \boldsymbol{x}^i} h_j(\boldsymbol{x}_\varepsilon), \quad \boldsymbol{x}^i \in \Gamma, \tag{9}$$

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

$$g_j(\boldsymbol{x}) = \int_{\Gamma_j} G(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\Gamma_{\boldsymbol{y}}, \quad h_j(\boldsymbol{x}) = \int_{\Gamma_j} \boldsymbol{H}(\boldsymbol{x},\boldsymbol{y})\cdot\boldsymbol{n}(\boldsymbol{y})\,\mathrm{d}\Gamma_{\boldsymbol{y}}, \quad \boldsymbol{x}\in\mathbb{R}^3. \tag{10}$$

With the aid of the Newton potential (5), the vector $\{B\}$ characterizes the contributions of the source function b with component given by

$$B_i = \lim_{\boldsymbol{x}_\varepsilon \to \boldsymbol{x}^i} \mathrm{V}(\boldsymbol{x}_\varepsilon), \quad \boldsymbol{x}^i \in \Gamma. \tag{11}$$

For mathematical consistency, u^j and t^j $(j=1,2,\ldots,N)$ are assumed to be quantities evaluated 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 (8) can be rearranged as

$$\mathbf{A}\{z\} = \{f\}, \tag{12}$$

where $\{z\}\in\mathbb{R}^N$ is a vector containing unknown potentials or fluxes on Γ, and $\{f\}\in\mathbb{R}^N$ is a vector whose entries are obtained from the contributions of the source term $\{B\}$ and the prescribed 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]. In addition, it was demonstrated in [7] that V can be approximated as

$$\mathrm{V}(\boldsymbol{x}) = \frac{1}{4\pi}\sum_{j=1}^{N} \mathrm{A}_j(\boldsymbol{x})\,\mathcal{F}(\boldsymbol{x},\boldsymbol{y}^j), \quad \boldsymbol{x}\in\mathbb{R}^3, \tag{13}$$

where $\boldsymbol{y}^j$ denotes the centroid of the flat triangle Γ_j and

$$\mathrm{A}_j(\boldsymbol{x}) = \int_{\Gamma_j} \frac{(\boldsymbol{y}-\boldsymbol{x})\cdot\boldsymbol{n}(\boldsymbol{y})}{\|\boldsymbol{y}-\boldsymbol{x}\|}\,\mathrm{d}\Gamma_{\boldsymbol{y}}, \quad \boldsymbol{x}\in\mathbb{R}^3.$$

It was also shown in [7] that the coefficient $\mathrm{A}_j(\boldsymbol{x})$ can be computed *exactly* using the recursive expressions given in [6]. Lastly, the straight-line integral (7) can be effectively carried out using well-known quadrature formulae.

The solution of the linear system (12) 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 $u(\boldsymbol{x})$ at an arbitrary interior point $\boldsymbol{x}$ in Ω can be accomplished by use of (2), (5) and (10) as

$$u(\boldsymbol{x}) = \sum_{j=1}^{N} t^j\, g_j(\boldsymbol{x}) \,-\sum_{j=1}^{N} u^j\, h_j(\boldsymbol{x}) \,+\mathrm{V}(\boldsymbol{x}), \quad \boldsymbol{x}\in\Omega. \tag{14}$$

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

3 How to use CBEM_POI

To use CBEM_POI, simply provide the *triangulated surface mesh* and *boundary conditions* in the file "bndata.dat", and specify or provide a means to compute the source function h in "src_func.f90" for the Fortran 90 package (or "src_func.c" for the C package). Then, invoke the routine `cbem_poi_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_poi_fld_eval` to calculate u at user-specified interior points, i.e., to evaluate (14). CBEM_POI package is provided with a sample problem to illustrate the utilization of the code. The driver routine is in the file "poi_ex.f90" for the Fortran 90 package (or "poi_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_POI

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_POI should (i) gunzip and tar the package cbem_poi-f-d-x.tgz (cbem_poi-c-d-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_poi-f-d-x.tgz" ("tar -zxvf cbem_poi-c-d-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 `poi_ex`.

Once the executable `poi_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 Poisson equation in the standard cube $\Omega = \{(x, y, z) \in \mathbb{R}^3 : 0 < x, y, z < 1\}$. The source function $b(x, y, z) = -(2\,y^3+6\,y)\,e^{x+z}$. The exact solution $u(x, y, z) = y^3\,e^{x+z}$ is used to specify the Dirichlet boundary conditions on the bottom face $\{z = 0,\ 0 \le x, y \le 1\}$ and top face $\{z = 1,\ 0 \le x, y \le 1\}$. Neumann boundary conditions are prescribed on the remaining faces as $t = \boldsymbol{n} \cdot \nabla u$, where $\nabla u = (y^3\,e^{x+z}, 3\,y^2\,e^{x+z}, y^3\,e^{x+z})$. Here x, y, z represent the Cartesian coordinates of a point in the 3D space.

Listing 1: Sample input file for a mixed problem in 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       -0.41351479   1
28      1       3       4       -0.07213830   1
29      8       5       6        1.12404975   1
30      7       5       8        0.19609222   1
31     12       9      10        0.00000000   1
32     11       9      12        0.00000000   1
33     13      16      14        8.15484549   1
34     13      15      16        8.15484549   1
35     17      20      18        0.07213830   0
36     17      19      20        0.41351479   0
37     24      21      22        0.19609222   0
38     23      21      24        1.12404975   0
```

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 u is given on the element. If `m` $= 1$ instead, then the flux t 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_poi-c-d-x.tgz will look like this:

```
27      0      3      1      -0.41351479   1
28      0      2      3      -0.07213830   1
29      7      4      5       1.12404975   1
30      6      4      7       0.19609222   1
31     11      8      9       0.00000000   1
32     10      8     11       0.00000000   1
33     12     15     13       8.15484549   1
34     12     14     15       8.15484549   1
35     16     19     17       0.07213830   0
36     16     18     19       0.41351479   0
37     23     20     21       0.19609222   0
38     22     20     23       1.12404975   0
```

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

In CBEM_POI, the input data are read into the code via the routine `cbem_poi_input` described in the file "cbem_poi_input.f90" ("cbem_poi_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_POI

CBEM_POI is provided with a driver routine to solve a complete Poisson problem, computational routines to perform basic BEM tasks, and auxiliary routines to accomplish specific subtasks. Common to all BEM

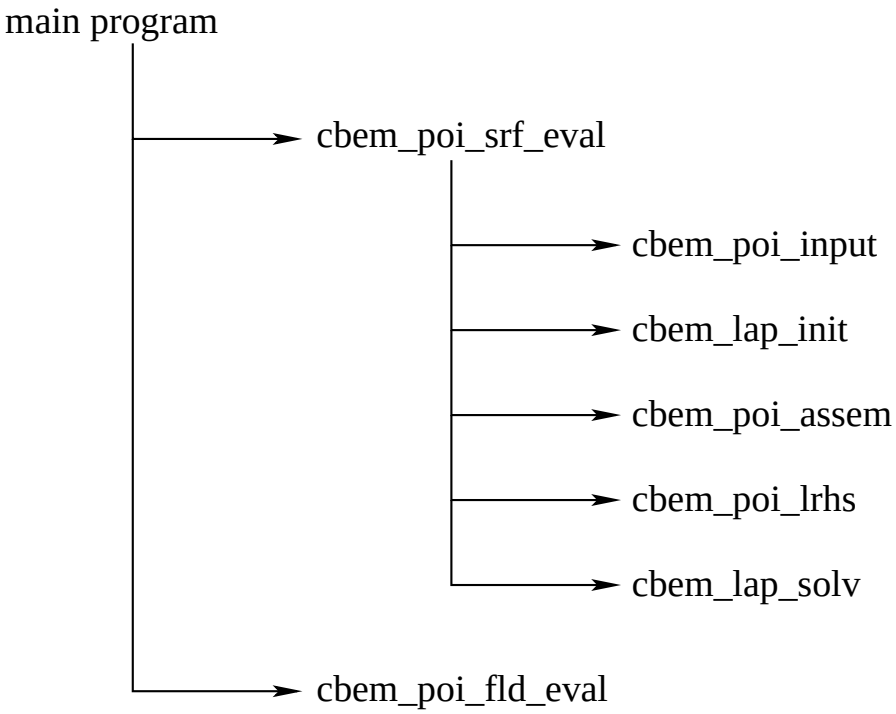


Figure 2: Structure of a BEM program.

program, the computational subroutines of CBEM_POI are listed below along with a brief description of the purpose of each routine:

Routine	**Description**
`cbem_poi_srf_eval`	Solve the Singular BIE for the Poisson equation
`cbem_poi_input`	Read input data and parameters
`cbem_lap_init`	Initialize arrays and parameters
`cbem_poi_assem`	Assemble **H** and **G** influence matrices, and the contributions of the source term
`cbem_poi_lrhs`	Form the left- and right-hand sides of the discretized BIE
`cbem_lap_solv`	Solve the Discretized BIE
`cbem_poi_fld_eval`	Evaluate solution at interior points

The acronym CBEM stands for Collocation Boundary Element Method. Fig. 2 illustrates the structure of the driver routine (see "poi_ex.f90" or "poi_ex.c") provided in this package. As can be seen from the figure, the main program (i.e. the driver routine) simply invokes `cbem_poi_srf_eval` to compute the potential u and flux t at the centroid of each boundary element, and `cbem_poi_fld_eval` to calculate the potential u at a set of prescribed interior points. Moreover, `cbem_lap_solv` merely calls LAPACK routines `dgetrf` and

`dgetrs` to directly solve the discretized BIE (12). The source function is specified in "src_func.f90" for the Fortran 90 package (or "src_func.c" for the C package). Note that `cbem_lap_init` and `cbem_lap_solv` are exactly the same routines used in CBEM_LAP package to solve the Laplace equation in 3D. The source code for an individual routine listed in Fig. 2 can be found in a file with corresponding name. For instance, the routine `cbem_poi_srf_eval` can be found in "cbem_poi_srf_eval.f90" for the Fortran 90 package (or "cbem_poi_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 boundary element code. For a detailed description of a specific CBEM_POI routine, interested users (especially software developers) should consult the source code.

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] W. Hackbusch. *Elliptic Differential Equations: Theory and Numerical Treatment*. Springer, New York, 1992.

[5] George C. Hsiao and Wolfgang L. Wendland. *Boundary Integral Equations*. Applied Mathematical Sciences, Volume 164. Springer, New York, 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. Treatment of domain integrals in boundary element methods. *Appl. Numer. Math.*, 62(6):720–735, 2012.