Solving the Linear Systems of Equations in the Generalized Direct Boundary Element Method

Stephen Kirkup[0000-0002-9680-7778]
School of Engineering, University of Central Lancashire, UK.
smkirkup@uclan.ac.uk

Abstract. When the direct boundary element method is used to solve problems with a general Robin (or mixed) boundary condition the resulting linear system of equations is not in the standard form. In this paper, a method involving exchanging columns of the matrices within the system is defined. The method is based on the principle of minimising the propagation of error and returns a system in standard form. Implementations of the method are developed in Matlab, Excel-VBA and Fortran. The method is demonstrated through its application to a test linear system and then it is applied within boundary element method test problems.

Keywords: Boundary Element Method, Linear System, Robin

1 Introduction

In this paper a numerical method for solving a general linear system of equations is developed, implemented in three programming languages and is demonstrated on a practical application in which it is useful - the solution of the system of equations that arise in the direct boundary element method with a general Robin (or mixed) boundary condition. The problem is that of solving the system

$$A\underline{x} = B\underline{y} + \underline{c}, \tag{1a}$$

where A and B are known $n \times n$ matrices and $\underline{c}$ is a known n-vector, with

$$\alpha_i\, x_i + \beta_i\, y_i = f_i \text{ for } i = 1 \ldots n \tag{1b}$$

where the α_i, β_i and f_i are known constants with α_i and β_i never both zero for all i. The evaluation of vectors $\underline{x}$ and $\underline{y}$ is the solution of the process. The solution of the standard linear system of equations is typically carried out by a standard method such as LU factorisation [1-2] and forward and back substitution, or by a technique that takes advantage of a particular structure or property of the matrices.

The simplest method for solving the problem (1) can be developed through writing (1b) in the form

$$D_\alpha\, \underline{x} + D_\beta\, \underline{y} = \underline{f}\,, \tag{2}$$

where D_α and D_β are diagonal matrices with $[D_\alpha]_{ii} = \alpha_i$ and $[D_\beta]_{ii} = \beta_i$ and then forming the following system:

$$\begin{pmatrix} A & -B \\ D_\alpha & D_\beta \end{pmatrix} \begin{pmatrix} \underline{x} \\ \underline{y} \end{pmatrix} = \begin{pmatrix} \underline{c} \\ \underline{f} \end{pmatrix}, \tag{3}$$

which can clearly be solved by standard methods. However, in forming the linear system (3) from the system (1), the dimension of the matrix has doubled, with an increased expectation of computational cost for the solution. This method is not subject of this paper, but this does provide a useful technique for validating the resulting method.

Returning to the problem (1), clearly if all the α_i $a$re non-zero, then a straightforward substitution of a rearrangement of (1b) ($x_i = \frac{f_i}{\alpha_i} - \frac{\beta_i}{\alpha_i} y_i$), followed by its substitution into the system (1a), will readily yield a solution through a standard method like the aforementioned LU factorization. A similar solution method can be applied if we can guarantee that all the β_i are non-zero. However, the method that we seek to develop in this paper must be prepared for α_i or β_i taking zero values. There is also the general concern in numerical work in dividing by 'small' numbers, for example, if any of the α_i or β_i are small. The approach adopted in this paper is to automate the rearrangement of the general system above, based on the relative size of α_i and β_i, and of the matrices, into a standard form of a linear system, so that standard techniques can then be applied. For simplicity, for the development and communication of the method, the techniques are initially applied to systems where all values are real, but the method is also applicable to complex-valued systems. Following an algorithmic rearrangement of the equations, that is the subject of this paper, LU factorisation followed by forward and back substitution is used to return the solution.

The *General Linear System* (GLS) method is implemented in Excel-VBA, and is included in the Excel spreadsheets LIBEM2 and LIBEMA that solve Laplace's equation by the boundary element method (BEM) for two-dimensional and axisymmetric three-dimensional problems. The method is also developed in Matlab and Fortran. Test problems are developed, including tests with the intended boundary element method application, in order to demonstrate the effectiveness of the method.

2 Boundary Element Method

The boundary element method is an established method for solving partial differential equations and has been widely applied in science and engineering [3]. Linear systems, in the general form stated, arise in the *direct* boundary element method with the general Robin or mixed boundary condition. As an illustration of this, let us consider the solution of the Laplace equation

$$\nabla^2 \varphi(\boldsymbol{p}) = 0 \qquad (\boldsymbol{p} \in D) \tag{4a}$$

in an interior domain *D*, bounded by a closed boundary *S*, with a Robin boundary condition of the form

$$\alpha(\boldsymbol{p})\boldsymbol{\varphi}(\boldsymbol{p}) + \beta(\boldsymbol{p})\frac{\partial\varphi}{\partial n_p}(\boldsymbol{p}) = f(\boldsymbol{p}) \quad (\boldsymbol{p} \in S), \tag{4b}$$

where $\boldsymbol{n}_p$ is the unit outward normal to the boundary at $\boldsymbol{p}$. Note that, in the special case in which $\alpha(\boldsymbol{p}) = 1$ and $\beta(\boldsymbol{p}) = 0$, the boundary condition is in the Dirichlet form and, if $\beta(\boldsymbol{p}) = 1$ and $\alpha(\boldsymbol{p}) = 0$, the boundary condition is in the Neumann form. The solutions of *potential* equations of this type by the sorts of methods outlined in this section are introduced in Jaswon and Symm [4] and Symm [5, 6].

There are two distinct derivation routes in the boundary element method; the *direct* method, based on Green's second theorem, and the *indirect* method, in which the solution is written in terms of a layer potential function on the boundary. In both the direct and indirect boundary element methods, the first stage in deriving a computational solution involves finding a function or functions on the boundary by solving a boundary integral equation (an integral equation that is defined only on the boundary). Once a boundary solution is obtained, an integral relation can be used to determine the domain solution from the boundary functions that are known. It is the first stage of the BEM that is of interest in this paper and we will see that this problem of obtaining a general 'Robin' solutions is particular to direct boundary element method results in problems like the system (1).

2.1 Direct Boundary Element Method

The direct BEM is most straightforwardly derived from the following integral equation reformulation of the interior Laplace equation:

$$\{M\varphi\}_S(\boldsymbol{p}) + ½\,\varphi(\boldsymbol{p}) = \{L\,v\}_S(\boldsymbol{p}) \quad (\boldsymbol{p} \in S), \tag{5a}$$

$$\{M\varphi\}_S(\boldsymbol{p}) + \varphi(\boldsymbol{p}) = \{L\,v\}_S(\boldsymbol{p}) \quad (\boldsymbol{p} \in D), \tag{5b}$$

where in equation (5a) it is presumed that S is smooth at $\boldsymbol{p}$ and in which the further shorthand is used to replace the boundary function $\frac{\partial\varphi}{\partial n_q}$ by v ($v = \frac{\partial\varphi}{\partial n_q}$).

. The equations (5a,b) are often written in a more concise form using operator notation. The operators L and M are defined as follows:

$$\{L\zeta\}_\Gamma(\boldsymbol{p}) = \int_\Gamma G(\boldsymbol{p},\boldsymbol{q})\,\zeta(\boldsymbol{q})\,dS_q \tag{6}$$

$$\text{and } \{M\zeta\}_\Gamma(\boldsymbol{p}) = \int_\Gamma \frac{\partial G(\boldsymbol{p},\boldsymbol{q})}{\partial n_q}\,\zeta(\boldsymbol{q})\,dS_q\,, \tag{7}$$

where (in this case) Γ is the whole or part of the surface S and $G(\boldsymbol{p},\boldsymbol{q})$ is the free-space Green's function for the Laplace equation.

An integral equation method is then applied to the boundary integral equation (5a) in order to derive the BEM. One of the most straightforward methods of derivation from the integral equation is by the method of collocation and this technique will be applied in this paper. The application of collocation to an integral equation like (5a) generally requires that the boundary is approximated and simplified by a set of *panels*, so that S is replaced by $\tilde{S}$. The functions defined on the surface are also represented by simplified functions. In the simplest application of collocation to an integral equation the boundary is approximated by a set of n straight line panels $\Delta\tilde{S}_1, \Delta\tilde{S}_2, \dots, \Delta\tilde{S}_n$ (in 2D) (so that $\tilde{S} = \sum_{i=1}^{n} \Delta\tilde{S}_i$) and the boundary functions are approximated by a constant on each panel. With the collocation points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, … $\boldsymbol{p}_n$, located on the panels each with the same index, the integral equation (5a) becomes the following approximation, written in matrix-vector form

$$(\mathrm{M_{SS}} + \tfrac{1}{2}\mathrm{I})\ \underline{\varphi_S} \approx \mathrm{L_{SS}}\underline{v_S}\ , \tag{8}$$

where $\mathrm{L_{SS}}$ and $\mathrm{M_{SS}}$ are matrices, I is the identity matrix and the vectors $\underline{\varphi_S}$ and $\underline{v_S}$, are the representative values of the function φ and v at the collocation points. Hence the collocation method requires the solution of the following system

$$(\mathrm{M_{SS}} + \tfrac{1}{2}\mathrm{I})\ \underline{\hat{\varphi}_S} = \mathrm{L_{SS}}\ \underline{\hat{v}_S}\ , \tag{9a}$$

$$\alpha_i\ \hat{\varphi}_{S_i} + \hat{\beta}_i\ \hat{v}_{S_i} = f_{S_i}\ \text{ for } i = 1 \dots n\ , \tag{9b}$$

where the $\hat{\varphi}_{S_i}$ and $\hat{v}_{S_i}$ are the approximate or representative values of the functions φ and v at the collocation point $\boldsymbol{p_i}$.

The similarity in form of equations (9) and the system addressed in this paper is clear. In this case there is no equivalent to the ‘$\underline{c}$’ term in (1a), however its inclusion is useful in problems of this form when there is an existing free-field solution or source within the domain and the boundary and boundary condition modify that solution.

Once the surface solution have been found, the solutions within the domain can be found through the application of equation (5b):

$$\varphi(\boldsymbol{p})\ =\ \{\,L\,v\,\}_S\,(\boldsymbol{p})\ \ -\{\,M\varphi\}_S\,(\boldsymbol{p})\ \ (\boldsymbol{p}\ \in\ D)\,. \tag{10}$$

Usually the solution is sought at a set of points in the domain and the discrete equivalent of this equation is expressed as follows

$$\underline{\hat{\varphi}_D} = \mathrm{L_{DS}}\ \underline{\hat{v}_S} - \mathrm{M_{DS}}\ \underline{\hat{\varphi}_S}\ , \tag{11}$$

where $\underline{\hat{\varphi}_D}$ represents the approximate solution at the domain points.

2.2 Indirect Boundary Element Method

The non-standard form of the linear system does not arise in the implementation of the 'indirect' boundary element method, but it is useful to also consider it here in order to contrast it with the direct boundary element method. In the indirect method, it is presumed that the solution of Laplace's equation in the domain is related to a layer potential on the boundary. For example with a 'single layer' potential σ;

$$\varphi(\boldsymbol{p}) = \{L\sigma\}_S(\boldsymbol{p}) \quad (\boldsymbol{p} \in D \cup S), \tag{12a}$$

$$v(\boldsymbol{p}) = \{(M^t + \frac{1}{2}I)\sigma\}_S(\boldsymbol{p}) \quad (\boldsymbol{p} \in S)\,, \tag{12b}$$

where the operator M^t is defined as follows

$$\{M^t\zeta\}_\Gamma(\boldsymbol{p}) = \int_\Gamma \frac{\partial G(\boldsymbol{p},\boldsymbol{q})}{\partial n_p}\,\zeta(\boldsymbol{q})\,dS_q\,.$$

The application of an integral equation method to the indirect equations, following a similar technique to the method outlined earlier, gives linear systems of the following form:

$$\underline{\hat{\varphi}_S} = \mathrm{L_{SS}}\,\underline{\hat{\sigma}_S} \tag{13a}$$

$$\underline{\hat{v}_S} = \left(\mathrm{M}_{SS}^t + \frac{1}{2}\mathrm{I}\right)\underline{\hat{\sigma}_S}\,. \tag{13b}$$

as the discrete analogues of equations (12a,b). By writing the boundary condition (9b) in the form

$$D_\alpha\,\underline{\hat{\varphi}_S} + D_\beta\,\underline{\hat{v}_S} = \underline{f_S}\,, \tag{20}$$

where D_α and D_β are diagonal matrices (2) and substituting the expressions from equations (13a) and (13b) gives

$$\left[D_\alpha\,\mathrm{L_{SS}} + D_\beta\left(\mathrm{M}_{SS}^t + \frac{1}{2}\mathrm{I}\right)\right]\underline{\hat{\sigma}_S} = \underline{f_S}\,, \tag{19}$$

a linear system of equations in the standard form. Having found the approximations to the surface solution $\underline{\hat{\sigma}_S}$ the solutions at a set of points in the domain can be found using the discrete equivalent of equation (12a),

$$\underline{\hat{\varphi}_D} = \mathrm{L_{DS}}\,\underline{\hat{\sigma}_S}\,. \tag{20}$$

3 Algorithm for Solving the General Linear System

In this section we return to the fundamental problem addressed in this paper; the solution of a linear system defined at the beginning of the paper by equations (1a,b). The author's book [7] and associated software and subsequent work in developing boundary element software libraries [8], the implementations are based on the Robin boundary condition in order to maintain generality. In this section, a method for exchanging columns is developed on the principle of minimising error propagation.

In order to consider the development of the method, let us look at the system (1a) in more detail:

$$\begin{pmatrix} a_{11} & \cdot\cdot & a_{1n} \\ : & \cdot\cdot & : \\ a_{n1} & \cdot\cdot & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ : \\ x_n \end{pmatrix} = \begin{pmatrix} b_{11} & \cdot\cdot & a_{1n} \\ : & \cdot\cdot & : \\ b_{n1} & \cdot\cdot & a_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ : \\ y_n \end{pmatrix} + \begin{pmatrix} c_1 \\ : \\ c_n \end{pmatrix}, \qquad (20)$$

for any index of $\underline{x}$ and $\underline{y}$ we can make one or the other of the following substitutions, through rearranging the equations (1b):

$$y_i = \frac{f_i}{\beta_i} - \frac{\alpha_i}{\beta_i}\, x_i, \quad x_i = \frac{f_i}{\alpha_i} - \frac{\beta_i}{\alpha_i}\, y_i\,. \qquad (19)$$

Clearly, if either α_i or β_i is zero then the choice of substitution is clear-cut. In numerical work there is also the need to avoid division by 'small' numbers, and hence the above guidance also extends to the situation in which either α_i or β_i is 'small'. The approach adopted by Symm [5] was to base the choice of substitution in (19) on the relative modulus of α_i and β_i, so that the division was always by the larger of the two values. However, in this work, the matrices are also taken into account within the method.

In order to arrive at a criterion for the decision to exchange sides for a particular x_j and y_j in the system above. Let us rewrite the system above, highlighting the index 'j':

$$\begin{pmatrix} a_{11} & \cdot\cdot & a_{1j} & \cdot\cdot & a_{1n} \\ : & \cdot\cdot & : & \cdot\cdot & : \\ : & \cdot\cdot & : & \cdot\cdot & : \\ : & \cdot\cdot & : & \cdot\cdot & : \\ a_{n1} & \cdot\cdot & a_{nj} & \cdot\cdot & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ : \\ x_j \\ : \\ x_n \end{pmatrix} = \begin{pmatrix} b_{11} & \cdot\cdot & b_{1j} & \cdot\cdot & b_{1n} \\ : & \cdot\cdot & : & \cdot\cdot & : \\ : & \cdot\cdot & : & \cdot\cdot & : \\ : & \cdot\cdot & : & \cdot\cdot & : \\ b_{n1} & \cdot\cdot & b_{nj} & \cdot\cdot & b_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ : \\ y_j \\ : \\ y_n \end{pmatrix} + \begin{pmatrix} c_1 \\ : \\ c_j \\ : \\ c_n \end{pmatrix}.$$

For clarity, let us consider one $'x_j'$ in the system as a candidate to be replaced by y_j. The replacement is show in the following equation, the x_j and y_j and the jth columns of the matrices are exchanged:

$$\begin{pmatrix} a_{11} & \cdot\cdot & -b_{1j} & \cdot\cdot & a_{1n} \\ : & \cdot\cdot & : & \cdot\cdot & : \\ : & \cdot\cdot & : & \cdot\cdot & : \\ : & \cdot\cdot & : & \cdot\cdot & : \\ a_{n1} & \cdot\cdot & -b_{nj} & \cdot\cdot & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ : \\ y_j \\ : \\ x_n \end{pmatrix} = \begin{pmatrix} b_{11} & \cdot\cdot & -a_{1j} & \cdot\cdot & b_{1n} \\ : & \cdot\cdot & : & \cdot\cdot & : \\ : & \cdot\cdot & : & \cdot\cdot & : \\ : & \cdot\cdot & : & \cdot\cdot & : \\ b_{n1} & \cdot\cdot & -a_{nj} & \cdot\cdot & b_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ : \\ x_j \\ : \\ y_n \end{pmatrix} + \begin{pmatrix} c_1 \\ : \\ c_j \\ : \\ c_n \end{pmatrix}.$$

With the substitutions (19), the left hand side can be altered as follows

$$\begin{pmatrix} a_{11} & \cdot\cdot & -b_{1j} & \cdot\cdot & a_{1n} \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ a_{n1} & \cdot\cdot & -b_{nj} & \cdot\cdot & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ y_j \\ \vdots \\ x_n \end{pmatrix}$$

$$= \begin{pmatrix} b_{11} & \cdot\cdot & -a_{1j} & \cdot\cdot & b_{1n} \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ b_{n1} & \cdot\cdot & -a_{nj} & \cdot\cdot & b_{nn} \end{pmatrix} \begin{pmatrix} \frac{f_1}{\beta_1} - \frac{\alpha_1}{\beta_1} x_1 \\ \vdots \\ \frac{f_j}{\alpha_j} - \frac{\beta_j}{\alpha_j} y_j \\ \vdots \\ \frac{f_n}{\beta_n} - \frac{\alpha_n}{\beta_n} x_n \end{pmatrix} + \begin{pmatrix} c_1 \\ \vdots \\ c_j \\ \vdots \\ c_n \end{pmatrix}.$$

Collecting the terms in the x_i and y_i on the right hand side, gives the following equation:

$$\begin{pmatrix} a_{11} + \frac{\alpha_1}{\beta_1} & \cdot\cdot & -b_{1j} + \frac{\beta_j}{\alpha_j} & \cdot\cdot & a_{1n} + \frac{\alpha_n}{\beta_n} \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ a_{n1} + \frac{\alpha_1}{\beta_1} & \cdot\cdot & -b_{nj} + \frac{\beta_j}{\alpha_j} & \cdot\cdot & a_{nn} + \frac{\alpha_n}{\beta_n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ y_j \\ \vdots \\ x_n \end{pmatrix}$$

$$= \begin{pmatrix} \frac{b_{11}}{\beta_1} & \cdot\cdot & \frac{-a_{1j}}{\alpha_j} & \cdot\cdot & \frac{b_{1n}}{\beta_n} \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ \vdots & \cdot\cdot & \vdots & \cdot\cdot & \vdots \\ \frac{b_{11}}{\beta_1} & \cdot\cdot & \frac{-a_{1j}}{\alpha_j} & \cdot\cdot & \frac{b_{nn}}{\beta_n} \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ f_j \\ \vdots \\ f_n \end{pmatrix} + \begin{pmatrix} c_1 \\ \vdots \\ c_j \\ \vdots \\ c_n \end{pmatrix}. \tag{20}$$

Let us now consider the decision to exchange the x_j term by comparing the relative error made in the components of the right hand side of the matrix for the two alternatives of exchanging the components. For the jth column, the ijth element is $-(b_{ij} + \frac{\beta_j}{\alpha_j})$ if the matrix components have been exchanged and $a_{ij} + \frac{\alpha_j}{\beta_j}$ if they had not.

Let δ be the relative error in α_j and let ε be the relative error in β_j. Hence the error propagated by process is $\frac{\alpha_j(1+\delta)}{\beta_j(1+\varepsilon)} - \frac{\alpha_j}{\beta_j} \approx \frac{\alpha_j}{\beta_j}(\delta - \varepsilon)$ if x_j and y_j remain in place and $\frac{\beta_j(1+\varepsilon)}{\alpha_j(1+\delta)} - \frac{\beta_j}{\alpha_j} \approx \frac{\beta_j}{\alpha_j}(\varepsilon - \delta)$ if the x_j and y_j are exchanged. Hence the relative error approximation in the *ij*th element of the left hand side matrix is $\frac{1}{a_{ij}}\frac{\alpha_j}{\beta_j}(\delta-\varepsilon)$ if x_j and y_j remain in place and $\frac{1}{b_{ij}}\frac{\beta_j}{\alpha_j}(\varepsilon-\delta)$ if the x_j and y_j are exchanged.

Let the decision on whether to exchange x_j and y_j be based on the relative error that is propagated. From the expressions above, this would infer that if $\left|\frac{1}{b_{ij}}\frac{\beta_j}{\alpha_j}(\varepsilon-\delta)\right| > \left|\frac{1}{a_{ij}}\frac{\alpha_j}{\beta_j}(\delta-\varepsilon)\right|$ or, simplifying, $\left|\frac{1}{b_{ij}}\frac{\beta_j}{\alpha_j}\right| > \left|\frac{1}{a_{ij}}\frac{\alpha_j}{\beta_j}\right|$then the x_j and y_j are exchanged. However, the decisions to exchange have to be made on a column-by-column (of the matrices) basis rather than element-by-element basis. Hence the a_{ij} and b_{ij} terms in the inequality above are replaced by $\|a_j\|$ and $\|b_j\|$; the norms of the *j*th columns of the matrices *A* and *B*. Hence, if $\left|\frac{1}{\|b_j\|}\frac{\beta_j}{\alpha_j}\right| > \left|\frac{1}{\|a_j\|}\frac{\alpha_j}{\beta_j}\right|$ then x_j and y_j are exchanged, or, simplifying, if $\|a_j\||\beta_j|^2 > \|b_j\||\alpha_j|^2$.

Once all the exchanges and rearrangements have occurred, the resulting system is the form

$$A^*\underline{x}^* = B^*\underline{f} + \underline{c}, \tag{21}$$

where $\underline{x}^*$ is the only unknown and $x_i^* = x_i$ if the i^{th} column has not been exchanged and $x_i^* = y_i$ if it has. The resulting matrix-vector problem is amenable to solution by standard methods. For example with LU factorisation of the matrix A^*, used in the coded examples, this results in

$$PLU\underline{x}^* = B^*\underline{f} + \underline{c}. \tag{22}$$

where P is a permutation matrix, L is a lower-tringular matrix and U is an upper-triangular matrix.

Once the O(n^3) LU factorisation has been completed, the O(n^2) forward and backward solution process to find particular solutions can be applied any number of times. This is precisely the approach that will be followed in this paper; for the equations the goal is to 'save' the O(n^3) LU factorisation. It can be observed in equation (22) that changing $\underline{c}$ and/or $\underline{f}$ alters the left hand side and has no effect on the matrix; changes in $\underline{c}$ and/or $\underline{f}$ only requires a further O(n^2) computational cost to find the new solution vectors $\underline{x}$ and $\underline{y}$ using the *REGLS* method. Hence, in the coding, the matrices L, U and B^* are stored along with the permutation information P and the record of exchanges.

The algorithms for solving systems of the form (1) are implemented in Fortran (file GLS2.FOR for real-valued systems and CGLS2.FOR for complex-valued systems), Matlab (gls.m for both real- and complex-valued systems) and in Visual Basic (gls.bas for real-valued systems). The appended algorithm for finding secondary solutions, as explained in the last paragraph are also implemented in Fortran (REGLS.FOR for real-valued systems and CREGLS.FOR for complex-valued systems), Matlab (regls.m for both real- and complex-valued systems) and in Visual Basic (regls.bas for real-valued systems). Test cases for the codes are implemented in the files GLS_T.FOR, gls_real_t.m and regls.bas for real-valued systems and CGLS_T.FOR, gls_complex_t.m for complex-valued systems. These files can be downloaded from the author's website [8].

4 Test Problems and Results

The GLS algorithm is demonstrated and tested principally in the context of the initial application area; within the implementation of the direct boundary element method with a general mixed (Robin) boundary condition. A test problem is defined and the GLS method is tested and the results are to be compared with alternative methods of solution. For this purpose the Matlab codes for solving the two-dimensional interior Laplace problem, introduced in the author's previous paper [9], are adapted.

4.1 Demonstration of the GLS method

In order to demonstrate the GLS method, it is applied to the following 5×5 system

$$\begin{pmatrix} 19 & -32 & -92 & 62 & -82 \\ 93 & 91 & -17 & 42 & 26 \\ 76 & 98 & -5 & -92 & -66 \\ -31 & -91 & -94 & 13 & -80 \\ -91 & 11 & -89 & -4 & -37 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

$$= \begin{pmatrix} -0.89 & 0.16 & 0.92 & -0.79 & -0.96 \\ 0.32 & 0.83 & -0.41 & -0.47 & 0.55 \\ 0.67 & 0.87 & 0.94 & -0.37 & 0.58 \\ -0.53 & 0.42 & -0.73 & -0.46 & -0.12 \\ -0.57 & 0.96 & -0.14 & 0.82 & 0.9 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} + \begin{pmatrix} 0.56 \\ -15.7 \\ 1.83 \\ -5.38 \\ 2.7 \end{pmatrix},$$

with

$$\begin{gathered} 8x_1 + 2.5y_1 = 14.52 \\ 66x_2 - 0.3y_2 = -3.54 \\ -0.96y_3 = 8.64 \\ -53x_4 - 0.74y_4 = 0.11 \\ 45x_5 = -0.9 \end{gathered}$$

having the solution $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} -0.06 \\ -0.04 \\ 0.08 \\ -0.03 \\ -0.02 \end{pmatrix}$ and $\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 6 \\ 3 \\ -9 \\ 2 \\ -7 \end{pmatrix}$.

In the GLS method, the column norms of the matrices are evaluated. In the implementations of the method then 1-norm is applied. Table 1 lists the pertinent values for the system. The decision to exchange is taken if $\|a_j\|\left|\beta_j\right|^2 > \|b_j\|\left|\alpha_j\right|^2$ and this is listed in the final column of the Table. Once the relevant columns of the matrices are exchanged an LU factorisation of the matrix is computed.

Table 1. The criterion for the decision to exchange columns in the test problem.

j	α_j	β_j	$\|a_j\|_1$	$\|b_j\|_1$	$\|a_j\|\left\|\beta_j\right\|^2$	$\|b_j\|\left\|\alpha_j\right\|^2$	Exchange
1	8	2.5	310	2.98	1937.5	190.72	YES
2	66	-0.3	323	3.24	29.07	14113.44	NO
3	0	-0.96	297	3.14	273.715	0	YES
4	-53	-0.74	213	2.91	116.638	8174.19	NO
5	45	0	291	3.11	0	6297.75	NO

Changing c and f to $c = \begin{pmatrix} -30.11 \\ -9.8 \\ -148.44 \\ -44.46 \\ -134.38 \end{pmatrix}, f = \begin{pmatrix} 62.4 \\ -75.9 \\ 0 \\ -15.68 \\ 90 \end{pmatrix}$, using REGLS returns

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0.3 \\ -1.3 \\ 0.4 \\ -0.5 \\ 2 \end{pmatrix} \text{ and } \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 24 \\ -33 \\ 0 \\ 57 \\ -18 \end{pmatrix}.$$

4.2 Tests with the Direct Boundary Element Method

In this subsection the GLS method is used in the boundary element method with mixed boundary conditions. The BEM is applied to the test problem is that of the unit square in which the solution of Laplace's equation is $\varphi = 2(x^2 - y^2)$. Hence on the left side $\varphi = -2y^2$, on the upper side $\varphi = 2(x^2 - 1)$, on the right side $\varphi = 2(1 - y^2)$ and on the lower side $\varphi = 2x^2$. If this is differentiated then we have $\frac{\partial\varphi}{\partial x} = 4x$ and $\frac{\partial\varphi}{\partial y} = 4y$. The normal on the upper side is in the y-direction, $\frac{\partial\varphi}{\partial n} = 4$, the normal on the right side is in the x-direction, $\frac{\partial\varphi}{\partial n} = 4$, the normal on the lower side is in the negative y-direction, $\frac{\partial\varphi}{\partial n} = 0$ and the normal on the left side is in the negative x-direction, $\frac{\partial\varphi}{\partial n} = 0$. The test problem is illustrated in Figure 1. The interior points (0.25,0.75) and (0.75,0.25) are the points at which the results will be investigated and are shown in Figure 1, at which points the exact solutions are φ =-1.0 and φ =1.0.

Test 1: Half Dirichlet and Half Neumann Boundary Condition. In the first test, the direct method involving the GLS algorithm is compared with alternative methods.

The Dirichlet boundary condition is applied on the left and upper sides and the Neumann condition is applied on the other sides. Three methods are used for solving the resulting linear system. The first is that of exchanging the first half of the columns of the matrices in equation (9a). In the second test the $2n \times 2n$ system (3) is formed from the equations (9). In the third test the GLS method is implemented. All methods gave the same solutions and these are listed in Table 2. Results for the corresponding indirect method are given in Table 3.

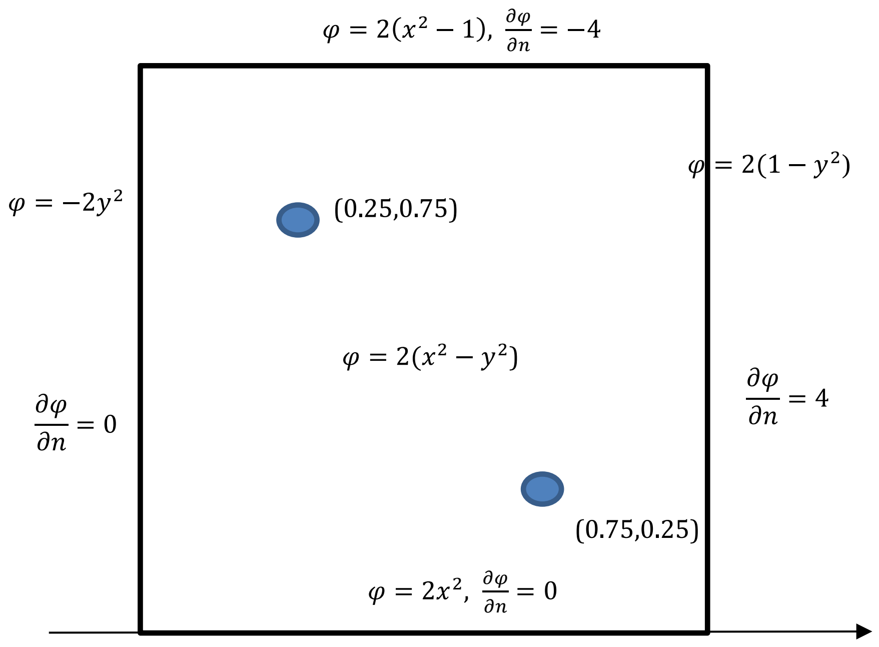
$\varphi = 2(x^2 - 1),\ \frac{\partial \varphi}{\partial n} = -4$
$\varphi = 2(1 - y^2)$
$\varphi = -2y^2$
(0.25,0.75)
$\varphi = 2(x^2 - y^2)$
$\frac{\partial \varphi}{\partial n} = 4$
$\frac{\partial \varphi}{\partial n} = 0$
(0.75,0.25)
$\varphi = 2x^2,\ \frac{\partial \varphi}{\partial n} = 0$

Fig. 1. The test problem.

Table 2. The results from the direct solution on square test with Dirichet and Neumann boundary conditions

elements	point (0.25,0.75)	point (0.75, 0.25)
32	-1.00116340579288	0.99140414916650
64	-1.00032267851548	0.99749398360249
128	-1.00009049851497	0.99926478494698
256	-1.00002558523010	0.99978267595968
512	-1.00000729360382	0.99993525789670
1024	-1.00000209812315	0.99998056513669

Table 3. The results from the indirect solution on square test with Dirichet and Neumann boundary conditions

elements	point (0.25,0.75)	point (0.75, 0.25)
32	-1.00723281386399	0.93455909390238
64	-1.00406088516754	0.95961702269782
128	-1.00246190337801	0.97476365754309
256	-1.00153247427505	0.98415050828163
512	-1.00096190256784	0.99002659494003
1024	-1.00060530545130	0.99371954438783

Test 2: Robin boundary condition with strong variation in prameter values. The second test is based on the same solution as Test 1, but the boundary condition is such that $\alpha(\boldsymbol{p})$ and $\beta(\boldsymbol{p})$ have values that strongly vary, in order to activate the GLS method fully. In this test the values $\alpha(\boldsymbol{p_j}) = 10^{6j/n}$ and $\beta(\boldsymbol{p_j}) = 10^{6(n-j)/n}$ are assigned for $j = 1,2, \dots, n$. The method based on the compound matrix (3) is used to verify the results from the GLS method. The results from these tests are given Table 4.

Table 4. The results from the direct solution on the problem with a Robin boundary condition of strongly varying parameter.

elements	point (0.25,0.75)	point (0.75, 0.25)
32	-0.99313784783564	1.00082851722167
64	-0.99803407600987	1.00022336648713
128	-0.99941881322429	1.00006375224490
256	-0.99982435277907	1.00001885053645
512	-0.99994612588802	1.00000570093421
1024	-0.99998331140774	1.00000174981983

5 Conclusion

In developing the boundary element method, one of the choices is whether to base it on the *direct* or *indirect* integral equation formulation. It has been shown, with a generalised Robin boundary condition, that the indirect pathway has the advantage of delivering a method that is already in standard form, whereas the direct route does not. Hence, in such circumstances, the indirect method could be favoured. However, with the GLS algorithm of this paper, the systems arising from the direct and indirect BEM can be solved in a similar computation time.

The GLS method, based on minimising the potential generated error and returning a standard system, has been designed. The algorithm has been implemented in three programming languages – Matlab, VBA, and Fortran and test harnesses have been included to demonstrate the working method.

The GLS method has been included in typical implementations of the direct boundary element method in these languages with LU factorisation method used to solve the resulting standard system. It has been shown that the method automatically gives the same result as alternative methods that either requires user-intervention or doubling the dimensions of the matrices. Results from the direct and indirect boundary element methods have been compared.

It may be thought that, for the BEM solution of problems with general Robin boundary conditions, the indirect approach should be chosen for convenience. However, the direct BEM is considerably more popular in the literature and hence it is important to develop a method that can resolve this issue. In this paper the GLS algorithm is shown to be easy to apply and is an $O(n^2)$ addition to the systems solution method and hence it levels the playing field between the direct and indirect BEM in the quest to establish generality in the resulting software.

References

1. Golub, G. H., Van Loan, C. F.: Matrix Computations, The John Hopkins University Press, Baltimore and London, 2nd Edition (1989)
2. N.J.Higham, N. J.:The Accuracy and Stability of Numerical Algorithms, SIAM (2002)
3. Wrobel. L.,: The Boundary Element Method: Applications in Thermo-Fluids & Acoustics Vol. 1. John Wiley and Sons (2002)
4. Jaswon, M. A. and Symm, G. T. :Integral Equation Methods in Potential Theory and Elastostatics, Academic Press (1977)
5. Symm, G. T. The Robin problem for Laplace's equation, NPL Report DNACS 32/80, National Physical Laboratory (1980)
6. Symm, G. T. Comtributons to a Numerical Library in Ada, NPL Report DITC 98/87, National Physical Laboratory (1987)
7. Kirkup, S. M.:The Boundary Element Method in Acoustics, Integrated Sound Software (1998/2007)
8. Boundary Element Method website. www.boundary-element-method.com
9. S M Kirkup, J. Yazdani A Gentle Introduction to the Boundary Element Method in Matlab/Freemat. (MAMECTIS '08). Corfu (2008)