BEM SOLVER IMPLEMENTATIONS WITHIN A DISCRETE METHODS APPLICATION FRAMEWORK

Diego Santiago, dsantiago@fi.mdp.edu.ar
Department of Mechanics Engineering (CONICET), UNMDP, JB Justo 4302, (7600) Mar del Plata, BsAs, Argentina

Marco Dondero, mdondero@fi.mdp.edu.ar
Adrián Cisilino, cisilino@fi.mdp.edu.ar
Welding & Fracture Division – INTEMA (CONICET), UNMDP, JB Justo 4302, (7600) Mar del Plata, BsAs, Argentina

Santiago Urquiza, santiago.urquiza@fi.mdp.edu.ar
Department of Mechanics Engineering, UNMDP, JB Justo 4302, (7600) Mar del Plata, BsAs, Argentina

Abstract. A practical strategy to increase the software reliability and to reduce programming time is to implement the code using Application Frameworks (AF). It is presented in this work a strategy for the implementation of a Boundary Element Method (BEM) solver within a discrete methods Application Framework (AF). The rationale behind this methodology lies in the possibility of reusing existing code to implement BEM. Thus, the effort put into the implementation of the BEM code is much less than that required for an ad-hoc development from scratch. A classical benchmark is solved in order to validate the developed code. Obtained results are compared to a direct BEM formulation in terms of execution times and memory requirements. The proposed approach was found to be an effective means for the implementation of Sparse BEM formulations some of which were evaluated in this work.

Keywords: Boundary Element Method, Finite Element Method, Application Frameworks.

1. INTRODUCTION

Solving problems of partial differential equations with numerical methods is a topic of great interest in engineering and science. The Finite Element Method (FEM) is widely used to find approximate solutions to such problems. A practical strategy to increase the software reliability and to reduce programming time is to implement the code using Application Frameworks (AF). An Application Framework is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by a user in order to provide specific functionality. Frameworks are similar to software libraries in that they are reusable abstractions of code wrapped in an Application Programming Interface (API); but unlike libraries, the overall control is not governed by the program, but by the framework.

It is presented in this work, a strategy for the implementation of a Boundary Element Method (BEM) solver within an AF. The cornerstone for this task is the Application Framework “SolverGP” developed in the Department of Mechanical Engineering of the University of Mar del Plata, Argentina. This framework was coded using FORTRAN and Object Oriented Programming tools. The framework has been successfully employed to implement FEM applications to solve a variety of problems in the fields of hemodynamics (Urquiza, 2006), friction stir welding (Santiago, 2004) and mold-filling using the resin transfer molding process (Santiago, 2007). The rationale behind the methodology introduced in this work lies on the possibility of reusing the existing code to implement BEM solvers. Thus, the effort put into the implementation of the BEM code is much less than that required for an ad-hoc development from scratch.

The resulting system of equations is solved iteratively using two proposed strategies in order to reduce the memory requirements for the storage of the fully populated BEM matrix. The developed code and the proposed iterative strategies are validated solving a benchmark example for the Laplace equation (the so-called Motz problem) which has analytical solution. The performance of the algorithm is compared to that of a standard direct BEM solver and assessed in terms of computing time and memory requirements.

2. BEM IMPLEMENTATION IN APPLICATION FRAMEWORK

2.1. Direct BEM formulation

The BEM discrete formulation for stationary potential problems in two dimensions is

\[ 0 = g_j q_j - f_j u_j, \quad g_j = \int_{\Delta S_j} G(x,y) dS(y), \quad f_j = \int_{\Delta S_j} F(x,y) dS(y) + \frac{1}{2} \delta_{ij} \]  \hspace{1cm} (1)
where \( u_i \) and \( q_i \) \((i = 1,2,...,N)\) are nodal values of potential and flux in the element \( \Delta S \) (see Fig. 1) respectively; \( G(x,y) \) and \( F(x,y) \) are the well-known Green functions for 2D problems; the points \( x \) and \( y \) are the collocation point and the field point respectively, and \( N \) is the number of constant elements of the discretization. Afterwards, the problem boundary conditions are replaced in the system of equations in Eq. (1). In this way the nodal values of the fluxes, \( q_i \), on the boundary \( S_1 \) and the nodal values of the potentials, \( u_i \), on the boundary \( S_2 \), will remain as the only unknowns. Finally, the system of \( N \) equations with \( N \) unknowns is reordered and solved. For further details is the reader is referred to the book by Aliabadi and Wrobel (2002).

![Figure 1. Boundary discretization using constant elements.](image)

### 2.2. Proposed BEM strategy

It is proposed in this work to assemble the BEM system of equations in Eq. (1) based on the strategy traditionally used in FEM. To this end, an “i-j archetypal” assemble-element is defined (Urquiza, 2002). The i-j archetypal element accounts for the contribution of each boundary element pair “i-j” to the system of equations. The elemental matrix for each pair “i-j” is

\[
\begin{bmatrix}
0 & 0 & f_{ij} & g_{ij} \\
0 & 0 & f_{ij} & g_{ij} \\
f_{ji} & g_{ji} & 0 & 0 \\
f_{ji} & g_{ji} & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
u_i \\
u_j \\
q_i \\
q_j \\
\end{bmatrix} =
\begin{cases}
0 & \text{if } i = j \\
0 & \text{if } i \neq j
\end{cases}
\begin{bmatrix}
u_i \\
u_j \\
q_i \\
q_j \\
\end{bmatrix} =
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\]

The archetypal element embeds the information of the \( i \) and \( j \) boundary elements so that the \( f_{ij} \) and \( g_{ij} \) entries can be evaluated. The global system of equations is constructed using a loop over all the archetypal elements. The elemental matrices contribute to the global system as in FEM implementations, i.e., by assembling their contributions into the global matrix. Thus, a system of equations with exactly the double of unknowns than equations is formed. Then, the AF applies a reduction technique which eliminates the equation related to the known value of \( j \) \( u \) or \( j \) \( q \) at each node and leaves the equation for the \( q_j \) or \( u_j \) unknown accordingly. The process results in a determined system of equations \( Ax=b \).

### 2.3. Sparse BEM formulations

This work proposes two alternative BEM assembly strategies which aim to reduce the computational requirements of the direct approach. The first one consists in an iterative procedure in which some terms of the BEM system of equations are moved to the to the right hand side (RHS) of the system \( Ax=b \) multiplied by the corresponding value of the unknown (or boundary conditions) evaluated in the previous iteration \((it)\). These terms are selected based on the distance \( R_{ij} = |x_i-x_j| \) between the collocation and field points \( i \) and \( j \), respectively.

When the distance \( R_{ij} \) for the i-j archetypal element is minor than a given threshold value \( R \) \( (R_{ij}<R) \), the elemental matrix (see Eq. 2) are assembled in the global matrix. In contrast, when the distance \( R_{ij} \) is greater than critical \( R \) \( (R_{ij}>R) \), the local matrix is moved to the RHS as follows:
The second formulation proposes an alternative strategy by incorporating the additional matrix $A_G$. This new matrix accounts for the contribution of several clusters of BEM neighbors elements which influences are condensed in a sparse manner. The BEM elements clusters are $G = \{E_{l1}, \ldots, E_{lr}, \ldots, E_{ln}\}$, where $E_{lr}$ is the element who condenses the group influence, and ‘$n$’ is the number of elements in the group. The global system of equations is:

$$
\begin{align*}
\left[ A_C + A_G \right] \{x\}^{(n+1)} &= \left[ -A_C + A_G \right] \{x\}^{(n)} + b \\
\text{where } A_{ij} &= 0, \quad A_{ij} = A_{ij} \text{ if } R_{ij} < R \\
A_{ij} &= A_{ij}, \quad A_{ij} = 0 \quad \text{if } R_{ij} > R \\
A_{ij} &= \sum_{j=E_l}^{E_r} a_{ij} \text{ if } E_{lj} = E_{lr} \quad \text{and } A_{ij} = 0 \text{ if } E_{lj} \neq E_{lr}
\end{align*}
$$

Using both approaches a sparse global matrix is constructed, which reduces the memory requirements when compared to the dense matrix of the direct BEM. It is worth to mention that these strategies in contrast to other acceleration techniques (fast multipole, panel clustering, wavelets, etc.), keep the precision of the direct BEM but they solve the linear system of equations iteratively.

### 3. Numerical Example and Validation

The Motz problem (Motz, 1946) is used to validate the proposed implementations. The problem geometry and boundary conditions are depicted in Fig. 2. The problem presents the challenge of possessing a singularity at $x=y=0$, where the boundary conditions suddenly changes from $u=0$ to $\partial u/\partial y=0$. An analytical solution to the problem is given in terms of the series expansion

$$
\begin{align*}
u(r, \theta) &= \sum_{j=1}^{m} a_j r^{(2j-1)/2} \cos \left[ \theta (2j-1) / 2 \right]
\end{align*}
$$

where the origin of the polar system is in the location of the singularity (see Fig. 2). The coefficients $a_j$ are tabulated in literature (Georgiou, 1996).

![Figure 2. The Motz problem: geometry, governing equations and boundary conditions.](image)

The problem was solved firstly using a 240-element discretization (case 1). Figure 3 depicts the computed solution for the direct approach. The global system of equations was solved in this case using Gauss factorization. The computed solution has a difference less that 0.5% relative to the analytical solution.
Additionally, the same problem was solved using both strategies described in Section 2.3. For the first strategy the problem was solved with values of $R$ ranging from 0.4 to 1. It can be seen in Table 1 that for $R>0.5$ convergence is achieved, obtaining the same solution as direct BEM. In contrast, for values of $R<0.5$, the solution diverges (see Fig. 4). For the second strategy, the convergence is achieved for all the values of $n$ studied. Table 1 reports the memory requirements and the solution status for the different strategies considered. Figure 4 shows a log-log plot of the relative error (compared to the analytical solution) with iteration number for both strategies where each characteristic parameter is varied through its valid range.

Figure 3. Left: Contour plot of potential field for case 1. Right: plots of the analytical and BEM solutions for the potential and flux fields versus node number along the boundary (see arrows and node numbers in the left figure).

Table 1. BEM memory requirements and convergence for the case 1.

<table>
<thead>
<tr>
<th>First Strategy</th>
<th>Second Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>Memory (KB)</td>
</tr>
<tr>
<td>0.40</td>
<td>424</td>
</tr>
<tr>
<td>0.45</td>
<td>471</td>
</tr>
<tr>
<td>0.50</td>
<td>519</td>
</tr>
<tr>
<td>0.70</td>
<td>711</td>
</tr>
<tr>
<td>1.00</td>
<td>1008</td>
</tr>
<tr>
<td>Direct BEM</td>
<td>2194</td>
</tr>
</tbody>
</table>

Figure 4. Log-log plot of the relative error vs. iteration number for both strategies (case 1).
To further investigate the performance of both strategies, the problem was discretized using 2400 elements (case 2). This case was used to study the execution time and error. The results for the memory requirements, the execution times and relative error with respect to the analytical solution up to the $10^{th}$ iteration for both strategies are reported in Table 2. In the first strategy it can be seen that the larger the value of $R$ the smaller the memory requirements, but they conduct to longer execution times. Besides, the number of iterations is also increased in order to achieve similar level of accuracy. The second strategy, in contrast, gives better results even for $n = 400$ and gets the direct BEM solution for $n = 5$ in only 10 iterations. In contrast the memory reduction is not as good as the first strategy. It can be inferred from the analysis of the results in Table 2 that the optimal value for the parameter $n$ in the second strategy is in the range $50 < n < 100$. Within this range the memory requirements are minimal and at the same time the relative error after 10 iterations is lower than any of the values achieved using the first strategy.

An additional analysis was carried out with the aim to investigate the memory requirement of the second strategy. For a fixed value of total number of BEM elements $N$, there exists an optimal value of $n$ that minimizes the memory requirements. A comparison of this minimum (calculated for each $N$) to the memory required for a direct BEM with the number of elements $N$ (relative storage in %), is shown in Fig. 5. It can be seen that for problems with more than $10^5$ unknowns a reduction of two or more orders of magnitude in the memory requirement is achieved. This improvement increases with $N$ letting this strategy be a candidate to solve large-scale problems with BEM.

<table>
<thead>
<tr>
<th>First Strategy</th>
<th>Second Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>Memory (MB)</td>
</tr>
<tr>
<td>0.50</td>
<td>46</td>
</tr>
<tr>
<td>0.70</td>
<td>64</td>
</tr>
<tr>
<td>1.00</td>
<td>93</td>
</tr>
<tr>
<td>50</td>
<td>62</td>
</tr>
<tr>
<td>100</td>
<td>62</td>
</tr>
<tr>
<td>200</td>
<td>65</td>
</tr>
<tr>
<td>Direct BEM</td>
<td>209</td>
</tr>
</tbody>
</table>

(* time accumulated to the 10th iteration.)

4. CONCLUSIONS

This work introduced an effective mean for the implementation of BEM solvers using a FEM Application Framework. The Application Framework takes advantage of existing OpenMP parallel structures. In this way the effort needed for the implementation of the BEM solver is much less than that required for an ad-hoc development from scratch.
It is also presented in this work an iterative implementation of the direct BEM which aims to reduce its memory requirements. Obtained results showed that although effective to reduce the memory requirements, the proposed method requires of longer execution times when compared to the standard direct BEM in order to achieve the same level of accuracy. Actual work is devoted to investigate different strategies to improve the convergence of the iterative solver and to further reduce the memory requirements. In what respects to the performance of the solver it is proposed to use the solutions of multiple previous iterations to estimate the solution for the next iteration. Besides memory requirements can be improved by means of multigrid methods.

The second strategy exhibits the potential to solve large-scale problems allowing important reductions in the memory requirements. Besides, the method showed robust and convergent for the solved example.

Although, actual implementation is based on the classical direct BEM formulation, the Application Framework possesses the potential for the implementation of accelerated BEM formulations such as Fast Multipole, Adaptive Cross Approximation and Hierarchical Matrices.

5. ACKNOWLEDGEMENTS

The authors acknowledge the support of the UNMdP, grant PICT 1154 (2007) of the ANPyCT of the República Argentina and the CONICET.

6. REFERENCES


7. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.