Interactive Boundary Element Analysis for Engineering Design

T.M. Foster[a,*], M.S. Mohamed[a], J. Trevelyan[a], G. Coates[a]

[a]*School of Engineering and Computing Sciences, Durham University, South Road, Durham, DH1 3LE, UK.*

Abstract

Structural design of mechanical components is an iterative process that involves multiple stress analysis runs; this can be time consuming and expensive. Significant improvements in the efficiency of this process can be made by increasing the level of interactivity. One approach is through real-time re-analysis of models with continuously updating geometry. Three primary areas need to be considered to accelerate the re-solution of boundary element problems. These are re-meshing the model, updating the boundary element system of equations and re-solution of the system.

Once the initial model has been constructed and solved, the user may apply geometric perturbations to parts of the model. The re-meshing algorithm must accommodate these changes in geometry whilst retaining as much of the existing mesh as possible. This allows the majority of the previous boundary element system of equations to be re-used for the new analysis. For this problem, a GMRES solver has been shown to provide the fastest convergence rate. Further time savings can be made by preconditioning the updated system with the LU decomposition of the original system. Using these techniques, near real-time analysis can be achieved for 3D simulations; for 2D models such real-time performance has already been demonstrated.

Keywords: Boundary Element Method (BEM), Real-time, Re-solution

1. Introduction

Real-time analysis of two-dimensional models is now a reality [1]. However, real-time stress analysis of three-dimensional models presents numerous additional challenges. Over the last decade, various schemes based on the Finite Element Method (FEM), that aim to provide interactivity have been presented [2, 3, 4, 5]. Meier *et al.* [6] review a range of deformable models that utilise both finite element (FE) and boundary element (BE) techniques. In this paper we address only BE implementations. The boundary element method (BEM) is a natural method to use where re-meshing is involved as changes need to be applied only to elements on the surface of the model. If the volume were meshed, as in the case of the FEM, then changes would propagate further through the model and many more degrees of freedom would be affected. Mackie [2] presents a substructuring approach that attempts to ameliorate these difficulties. Wang *et al.* [7] present a BEM based scheme for interactive analysis for surgical simulation. Real-time updating is also of interest to computer game developers to provide realistic deformable objects. For example, James and Pai [8] discuss the use of the BEM to provide physically accurate simulation of three-dimensional objects. However, as the technique is used purely for visual approximation of deformations, a much lower degree of accuracy is required and a coarse mesh can be used resulting in a very fast analysis. Pre-computed solutions are also utilised.

Concept Analyst [1], a 2D BE stress analysis package that features real-time functionality, has already been developed. The current work aims to extend this interactive stress analysis capability into the 3D domain. This involves the development of innovative techniques to generate, update and analyse models. A key part of this work is the subject of this paper, the need to rapidly reconstruct the system of linear equations produced by the analysis. For the small problems assessed in this work, the system construction time takes a significantly higher proportion of the total re-analysis time than re-solving the system.

2. The Boundary Element Method

The BEM is a standard method of analysis in the solution of partial differential equations, and is the subject of numerous texts including Becker [9]. This section contains a brief overview of the principal steps involved. We consider the problem of finding displacements and stresses in a linear elastic material comprising a domain $\Omega \subset \mathbb{R}^3$, having boundary $\partial\Omega = \Gamma$. We seek to solve the equations of linear elasticity subject to boundary conditions $u(q) = \bar{u}$, $q \in \Gamma_u$ and $t(q) = \bar{t}$, $q \in \Gamma_t$, where u, t are displacement and traction components, $\bar{u}, \bar{t}$ are prescribed displacement and traction

*Corresponding author
Email address: timothy.foster@durham.ac.uk (T.M. Foster)

boundary conditions, and $\Gamma = \Gamma_u \bigcup \Gamma_t$. In practice, the use of different boundary condition types in different coordinate directions at the same location is common, so that this division of Γ into separate Neumann and Dirichlet boundaries in this fashion is purely symbolic. The boundary integral equation (BIE) can be formulated for displacements at a source point, $p \in \Gamma$, due to tractions and displacements on Γ:

$$c(p)u_j(p) + \int_\Gamma T_{ij}(p,q)u_i(q)d\Gamma(q) = \int_\Gamma U_{ij}(p,q)t_i(q)d\Gamma(q) \tag{1}$$

where $c(p)$ is a term introduced as a result of the limits applied to allow the strongly singular integral containing the traction kernel to be evaluated, so that the integral on the left hand side of equation (1) is evaluated in the Cauchy Principal Value sense. T_{ij} and U_{ij} refer respectively to the traction and displacement kernels which are not given here but take into account the material properties and the distance between each source-field point pairing. The subscripts define directional components, so that T_{ij} and U_{ij} refer to a traction or displacement in the Cartesian direction i at field point q, caused by a unit load in direction j at the source point p. To solve the system numerically the boundary of the object must be discretized into a series of elements, forming a surface mesh. By computing the integrations in the discretized form of (1) a system of linear equations can be derived. These are given in matrix form as:

$$[H]\{u\} = [G]\{t\} \tag{2}$$

where $[H]$ and $[G]$ contain the integrated traction and displacement kernels respectively. If an appropriate set of boundary conditions is defined, equation (2) can be rewritten in the form:

$$[A]\{x\} = \{b\} \tag{3}$$

This system can now be solved as a set of linear equations to find the unknown tractions and displacements contained in $\{x\}$. Internal stresses may be found by declaring p at the point of interest, substituting the now fully defined values of displacement and traction at the nodes into equation (1) and summing boundary integrals to yield $u(p)$.

3. Integration strategy

3.1. Initial integration

In standard boundary element integration, a customised integration scheme is computed for each element-node pairing in the model. The choice of scheme is based on the distance between the pair with specialist schemes used to deal with singular and near-singular integrals. The geometric data associated with the element: shape functions, Gauss point locations, weights and normals, are computed specifically for the custom integration scheme. For large problems it is not possible to fit all the data associated with the integration of all the elements in the model into memory; they must therefore be recomputed each time they are required.

The element is integrated according to the specified integration scheme and the associated $[H]$ and $[G]$ sub-matrices constructed. With the application of boundary conditions these are assembled to form the $[A]$ matrix and $\{b\}$ vector given in (3). Finally the unknown singular terms on the diagonal of $[A]$ are calculated by applying rigid body motion. The complete system is then passed to the linear solver.

The number of Gauss points, m, required to integrate over an element is a function of the size of the element, L, and the minimum distance between the element and the source node, R. In 2D, L is the length of the element. In 3D, L is the length or breadth of the element and m must be computed independently for each dimension, producing a grid of Gauss points.

Gao and Davies [10] suggest an approximation to an algorithm developed by Lachat and Watson [11] and generalised by Mustoe [12] for finding m for quadrilateral elements. The original algorithm uses an iterative scheme to find the maximum ratio R/L; Gao and Davies propose the following approximation to improve efficiency:

$$m = \frac{p' \ln\left(\frac{e}{2}\right)}{2\ln\left(\frac{L}{4R}\right)}, \qquad p' = \sqrt{\frac{2}{3}p + \frac{2}{5}} \tag{4}$$

where e is the prescribed tolerance of the relative integration error and p is the order of the singularity. The scheme is plotted in figure 1. It should be noted that if $R/L < 0.25$ the integration order grows to infinity. If this is the case the element must be subdivided to ensure accurate integration.

An alternative scheme, based on numerical tests for surface integrals, is proposed by Bu and Davies [13]. This is generalised by Gao and Davies [10] and can be summarised as:

$$m = -0.1p' \ln\left(\frac{e}{2}\right)\left(\left(\frac{8L}{3R}\right)^{\frac{3}{4}} + 1\right) \tag{5}$$

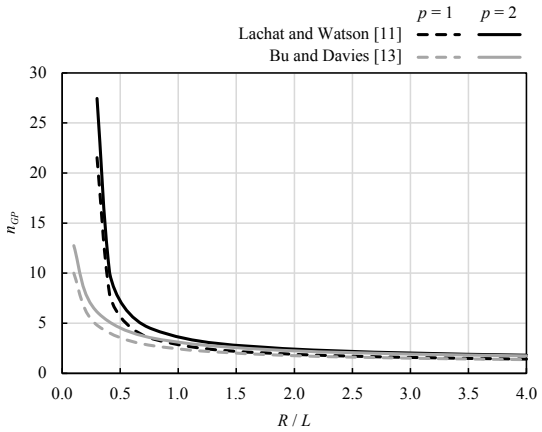


Figure 1: Integration order.

In this work a reusable intrinsic sample point (RISP) integration scheme is applied [14]. This differs from the standard scheme in that RISP uses a discrete number of integration schemes. This means that the shape functions and derivatives for each integration scheme need only be calculated once for each element type. The appropriate integration scheme, and hence the number of Gauss points, m, is chosen by taking R as the distance from the centroid of the field element to the source node and L as the average side length of the element. This scheme can now be applied to both quadrilateral and triangular elements.

Due to the coarser implementation scheme required by Gao and Davies [10] a 10% speedup can be gained over the Lachat and Watson [11] scheme, with an increase in error of less than 0.1%. Equation (5) can be rearranged to find the maximum ratio R/L for a given number of Gauss points. This is used in conjunction with the RISP integration scheme to directly determine the appropriate number of Gauss points to use across the entire element. If $e = 0.001$ table 1 is constructed where $\hat{m}$ is the total number of Gauss points on the element.

Near-singular integrals are dealt with by splitting the element into sub-elements about the singular node. Each sub-element is integrated as a quadrilateral element with two nodes lying at the singularity. However, to avoid the calculations involved in constructing the sub-elements, each element is built so that the weights and gauss point locations associated with each sub-element are applied across the complete element so that it may be treated in exactly the same way as the non-singular elements. The total number of Gauss points used for each near-singular integral are given in table 1.

Condition	Triangular $\hat{m}$	Quadrilateral $\hat{m}$
Corner singularity	32	64
Mid-side singularity	36	96
$0.0 \leq R/L < 0.7$	13	16
$0.9 \leq R/L < 1.1$	7	9
$1.4 \leq R/L < 2.7$	4	4
$3.1 \leq R/L$	1	1

Table 1: Integration schemes for $e = 0.001$.

Using a discrete number of integration schemes leads to a reduction in memory requirements and, as the models encountered in this work generally contain fewer than 2000 elements, it is possible to store all the geometric data associated with the integration in memory. The Gauss point locations, associated normals and Jacobians for all integration schemes are therefore only computed once for each element in the model. This data is stored for use in the re-analysis. The singular and near-singular integration schemes are formulated such that they can be applied using the same algorithm as the standard integrals. Aside from this, the remainder of the construction of the linear system is carried out in the same manner as for the standard integration scheme.

It is not normally necessary to explicitly calculate the full $[H]$ and $[G]$ matrices, the linear system may be constructed directly from the BIE. However, in the implementation proposed in this work, the full $[H]$ and $[G]$ sub-matrices associated with each element-node pairing are stored so that, where possible, they may be reused in the reanalysis.

3.2. Re-integration

The re-meshing algorithm used in this work aims to limit propagation of geometric changes through the mesh to those areas close to the updated geometry. This means that large parts of the mathematical system are unchanged. Re-integration

of the system can therefore be accelerated by re-using any unchanged $[H]$ and $[G]$ sub-matrices. This is done by following the flowchart given in figure 2. The same integration scheme is used for each node-element pairing as was used in the initial integration, this saves re-computing the distance between the two and alters the error by less than 0.1%. However, depending on whether the model is being stretched or compressed the error may reduce or increase respectively.

An element may be updated in two different ways: It may be translated or distorted. Updates to nodes may be thought of in the same manner: Translated nodes are those that lie on translated elements; distorted nodes lie on distorted elements. Where a node lies on both types of element it is classed as translated. Where a paired element and node have the same translation applied, the associated $[H]$ and $[G]$ sub-matrices will not change. Any element or node classified as distorted will require recalculation of all the associated sub-matrices.

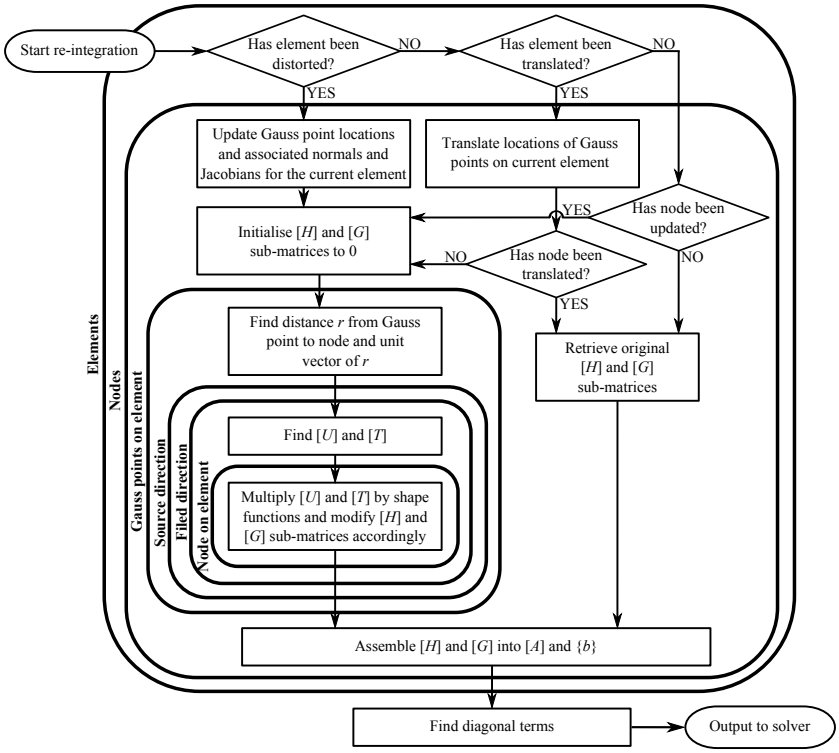


Figure 2: Flowchart of re-integration algorithm.

4. Results

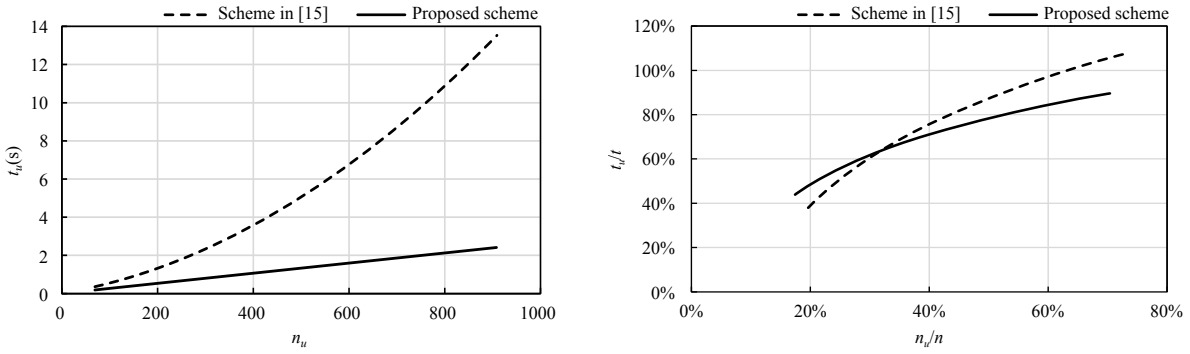
Several different approaches were implemented to assess their ability to accelerate the construction of the linear system of equations. If each element were to be divided into four sub-elements, using a quarter of the number of Gauss points on each subelement should in theory yield the same accuracy as using larger elements. However, subdivision proved to be slower and did not produce the expected accuracy. For the most efficient analysis few high quality elements with a detailed integration scheme should be used.

Tests were carried out on models using all triangular and a combination of triangular and quadrilateral elements. It was found that meshes containing a mix of element types provided similar accuracy to, and were around 1.2 times faster on average, than meshes formed entirely from triangular elements. The reduction in the number of elements and therefore the number of degrees of freedom also led to a reduction in the re-solution time.

Figure 3(a) shows the update time, t_u, for a range of geometries with different numbers of updated nodes, n_u for the re-integration algorithm proposed in this work and the traditional integration scheme used by Foster *et al.* in [15]. Both methods only update parts of the system that have changed. The proposed algorithm performs significantly faster than the traditional algorithm providing a speed up, S, of the form:

$$S \simeq 0.004 n_u + 1.6 \tag{6}$$

The ratio t_u/t, where t is the initial integration time, is plotted against the percentage of updated nodes in figure 3(b). This shows that, for problems where $n_u/n > 0.3$ the proposed scheme improves on the speedup achieved by Foster *et al*. It should be noted that the majority of models lie in this region. The effect of overheads that resulted in $t_u > t$ when $n_u/n > 0.63$ in the Foster *et al.* scheme have also been reduced.



(a) Absolute update time.

(b) Normalised update time.

Figure 3: Comparison of update times using the re-integration scheme.

5. Parallelization

The first algorithm for parallelization of the BEM appeared in 1984 in a work by Symm [16]. Study progressed slowly for the first few years, Davies [17] comprehensively reviews the early work on parallelizing the method. Many authors, including Kane [18] and Baltz and Ingber [19], use block partitioning to send different parts of the $[A]$ matrix to be computed on different processors. However the overlap between these blocks, where elements share a node, adds an extra level of complexity. Kamiya *et al*. [20] and Erhart [21] use domain decomposition to solve the BE problem. This is easily parallelized as each domain can be sent to a different processor. However, additional overheads are incurred in assessing the performance of each processor for optimal distribution of the domains.

All of the literature discussed above deals with parallelizing the BEM across multiple distributed memory devices. Here we aim to parallelize the method on a single shared memory machine. This is a much simpler process, simplified yet further by the way the integration algorithm described in this work has been constructed. First the 'Assemble $[H]$ and $[G]$ into $[A]$ and $\{b\}$' command must be moved outside the main pair of loops. This will remove the overlap between the output from each processor so that there will be no clashes where two processors are trying to write to the same part of the computer memory. As everything is now independent, the 'Elements' loop can be parallelized at the top level, sending each element to a different processor. The assembly of $[A]$ and $\{b\}$ may be parallelized by generating each row of the system on a different processor.

The average speed up, S, achieved by carrying out the integration on multiple processors on a single shared memory machine is shown in figure 4. For the re-integration S is 3% smaller than for the initial integration. This is because the overheads associated with the parallelization have a greater impact on the faster re-integration algorithm. However, both schemes show good scalability which is independent of the percentage of the system that is updated.

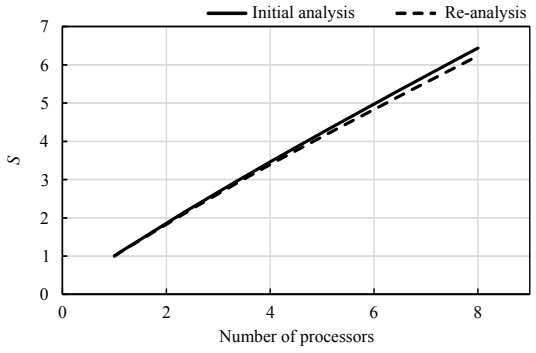


Figure 4: Integration speed gain for multiple processors.

6. Conclusions

A new, efficient algorithm for accelerating the re-integration of the BIE after a geometric change has been applied to a model has been introduced and shown to be effective both in serial and in parallel. By combining the work carried out in this paper with the meshing and solution schemes suggested by Foster *et al.* [15] it is possible to perform real-time re-analysis for small problems and significantly accelerate re-analysis of larger systems.

Acknowledgements

This research is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant EP/H000046/1. The authors would like to thank Dr. Stuart Spence of BAE Systems and Mr. Simon Walker of Jesmond Engineering for their input and support.

References

[1] Concept Analyst Ltd., URL: www.conceptanalyst.com (2006).

[2] R. Mackie, An object-orientated approach to fully interactive finite element software, Advances in Engineering Software 29 (1998) 139–149.

[3] M. Ryken, J. Vance, Applying virtual reality techniques to the interactive stress analysis of a tractor lift arm, Finite Elements in Analysis and Design 35 (2000) 141–155.

[4] L. Margetts, C. Smethurst, R. Ford, Interactive finite element analysis, in: NAFEMS World Congress, Malta, 2005.

[5] S. Terdalkar, J. Rencis, Graphically driven interactive finite element stress analysis for machine elements in the early design stage, Finite Elements in Analysis and Design 42 (2006) 884–899.

[6] U. Meier, O. López, C. Monserrat, M. Juan, M. Alcañiz, Real-time deformable models for surgery simulation: A survey, Computer Methods and Programs in Biomedicine 77 (2005) 183–197.

[7] P. Wang, A. Becker, I. Jones, A. Glover, S. Benford, C. Greenhalgh, M. Vloeberghs, Virtual reality simulation of surgery with haptic feedback based on the boundary element method, Computers and Structures 85 (2007) 331–339.

[8] D. James, D. Pai, ArtDefo - Accurate real time deformable objects, in: SIGGRAPH 99, 1999, pp. 65–72.

[9] A. Becker, The Boundary Element Method in Engineering, McGraw-Hill International, 1992.

[10] X. Gao, T. Davies, Adaptive integration in elasto-plastic boundary element analysis, Journal of the Chinese Institute of Engineers 23(3) (2000) 349–356.

[11] J. Lachat, J. Watson, Effective numerical treatment of boundary integral equations: A formulation for three-dimensional elastostatics, International Journal for Numerical Methods in Engineering 10 (1976) 991–1005.

[12] G. Mustoe, Developments in Boundary Element Methods, Elsevier, London, 1984, Ch. Advanced integration schemes over boundary elements and volume cells for two- and three-dimensional non-linear analysis.

[13] S. Bu, T. Davies, Effective evaluation of non-singular integrals in 3D BEM, Advances in Engineering Software 23 (1995) 121–128.

[14] J. Kane, Boundary Element Analysis in Engineering Continuum Mechanics, Prentice-Hall, 1994.

[15] T. Foster, M. Mohamed, J. Trevelyan, G. Coates, Rapid re-meshing and re-solution of three-dimensional boundary element problems for interactive stress analysis, Engineering Analysis with Boundary Elements 36 (2012) 1331–1343.

[16] G. Symm, Boundary elements on a distributed array processor, Engineering Analysis 1 (1984) 162.165.

[17] A. Davies, Parallel implementations of the boundary element method, Computers and Mathematics with Applications 31(6) (1996) 33–40.

[18] J. Kane, Boundary element analysis on vector and parallel computers, Computing Systems in Engineering 5(3) (1994) 239–252.

[19] B. Baltz, M. Ingber, A parallel implementation of the boundary element method for heat conduction analysis in heterogeneous media, Engineering Analysis with Boundary Elements 19 (1997) 3–11.

[20] N. Kamiya, H. Iwase, E. Kita, Parallel implementation of boundary element method with domain decomposition, Engineering Analysis with Boundary Elements 18 (1996) 209–216.

[21] K. Erhart, E. Divo, A. Kassab, A parallel domain decomposition boundary element method approach for the solution of large-scale transient heat conduction problems, Engineering Analysis with Boundary Elements 30 (2006) 553–563.