Latin American Journal of Solids and Structures
www.lajss.org

Revisiting Some Developments of Boundary Elements for Thick Plates in Brazil [1]

Abstract

This work reviews the developments of Boundary Element Method formulations to solve several types of plate bending problems, including non-linear bending. The formulation is developed and solved using the standard BEM procedure, and different integration approaches were discussed and tested. Object oriented implementation issues are commented. Results were obtained for linear and non-linear elastic bending as well as buckling of selected cases of thick plates, including cases of step variation in thickness under large displacements regime.

Keywords

Boundary element method, thick plate, nonlinear plate bending, plate buckling, object oriented programming.

Rogério José Marczak

Mechanical Engineering Dept. - UFRGS
Rua Sarmento Leite 425, Porto Alegre, Brazil – 90050-170
rato@mecanica.ufrgs.br

http://dx.doi.org/10.1590/1679-78251441

Received 03.07.2014
In revised form 17.11.2014
Accepted 15.12.2014
Available online 03.02.2015

1 INTRODUCTION

Traced back to early 1980's, the development of general boundary element approaches for thick plate bending analysis took almost fifteen years to be fully established. This was partly due to the numerical and mathematical difficulties that have been preventing a more general use of the boundary element method (BEM) to complex problems like dynamical and non-linear problems, but also intrinsically related to the complex nature of plate/shell equations, arguably some of the most challenging among the usual structural theories. The first work on the application of the BEM to moderately thick plates was published in 1982 (van der Weeën, 1982). In spite of several other published works on the matter during the last decades, there are still few works dealing with non-linear bending (Vilmann, 1990) (Xiao-Yan et al., 1990)(Marczak, 2004)(Supriyono, 2006), contact problems, anisotropic material, and variable thickness problems (di Pisa, 2005), to name a few

[1] Much of the work presented here was developed or started its development during the 1990s during the collaboration of the author with Prof. Clóvis S. de Barcellos, and has been under continuous improvement since then. The present review not only summarizes an important research branch of boundary element methods, but also recognizes the pioneering vision of Prof. Barcellos in encouraging a whole generation of then young researchers to pursuit challenging problems, and solve them under well posed formulations.

shear deformable plate research topics solved by the BEM. Characteristics of the BEM in plate bending analysis like the absence of locking and the high accuracy moments and shear forces are among the advantages that justify further works on the subject.

This work collects the relevant boundary element equations for linear and non-linear bending as well as buckling analysis of moderately thick plates. The plate models account for the shear influence by using the Mindlin and the Reissner first order plate theories. A unified integral formulation for the plate models is extended to the differential operators found in von Kármán equations in order to consider geometrically non-linear effects. The integral formulation for membrane-bending coupling is also developed, leading to an integral equation system that describes large displacement bending problems. The linearization of these equations leads to an eigenproblem which can be used for the linear elastic stability analysis. Yet another special case of these equations leads to a linear system corresponding to the boundary value problem of static bending of plates. The direct boundary element method was used to obtain an approximate solution of the integral equation system. The proposed formulation was tested using constant, linear and quadratic elements for non-linear benchmark cases, and up to quartic elements for linear tests.

Most of the present work started back in 1990 in a research group at Federal University of Santa Catarina, Brazil. At the time, improvements on van der Wëeen's work (1992) on Reissner's plate were done and extended to linear bending of Mindlin's plate model (de Barcellos & Monken e Silva1, 1989)(de Barcellos & Westphal Jr., 1992) (Westphal Jr. & de Barcellos, 1990), buckling (Marczak, 1995b, 1995c), higher order elements (Marczak, 1995a). Later non-linear bending was also explored at Federal University of Rio Grande do Sul ((Marczak, 1996) (Marczak & Creus, 2002). Meanwhile, plastic bending of Reissner's plate were solved at Federal University of Rio de Janeiro (Karam & Telles, 1988), while other groups kept developing the method for thin plates (Costa Jr. & Brebbia, 1985) (Chaves et al., 1999)

The integral formulation presented herein is condensed to account for a general boundary integral equation framework of elastic analysis of moderately thick plates. The numerical examples presents novel figures for a selection of benchmarks, and compare the h convergence rates for boundary elements ranging from constant up to quartic degree with several progenitors of now renowned finite elements. Results comparing three singular integration schemes also provide a useful set of reference results to assess other numerical formulations. BEM results for thick plate buckling and non-linear bending of plates with varying thickness seem to be original in the BEM context as well.

2 NAVIER EQUATIONS FOR FOSD PLATE THEORIES

The displacement field usually employed in the first order shear deformation (FOSD) theories, like the Mindlin (1951) plate theory, in conjunction with the 2D elasticity (membrane) equations, is based in the following expansions for the displacements [2]:

[2] Index notation is used throughout this work. Greek indexes range from 1 to 2 while the Latin ones range from 1 to 3. Upper indexes l and n refer to linear and non-linear parts, respectively, while m and f indicate in-plane and bending terms, respectively.

$$\begin{aligned} U_\alpha\left(x_1,x_2,x_3\right) &= v_\alpha + x_3 u_\alpha \\ U_3\left(x_1,x_2,x_3\right) &= u_3 \end{aligned} \tag{1}$$

Where v and u are the membrane and plate displacements, respectively (i.e. v_α and u_3 are the in plane and out of plane translations, respectively, while u_ε are the plate rotations). All variables are referred to the plates's middle surface. The Reissner's plate model (Reissner, 1944, 1945) also leads to eqs.(1) when the mean value of the displacements is taken across the thickness. Substituting eqs.(1) in the finite deformations strain tensor and neglecting the thickness extensibility, the strain-displacement relations are obtained. When these are used in the generalized Hooke's law and integrated through the thickness leads to the constitutive equations of the problem:

$$\begin{aligned} N_{\alpha\beta} &= C\frac{1-\nu}{2}\left(v_{\alpha,\beta} + v_{\beta,\alpha} + u_{3,\alpha}u_{3,\beta}\right) + C\nu\left(v_{\gamma,\gamma} + \frac{u_{3,\gamma}u_{3,\gamma}}{2}\right)\delta_{\alpha\beta} \\ M_{\alpha\beta} &= D\frac{1-\nu}{2}\left[u_{\alpha,\beta} + u_{\beta,\alpha} + \frac{2\nu}{1-\nu}u_{\gamma,\gamma}\delta_{\alpha\beta}\right] \\ Q_\alpha &= D\lambda^2\frac{1-\nu}{2}\left[u_\alpha + u_{3,\alpha}\right] \ , \end{aligned} \tag{2}$$

where $C = Eh/(1-\nu^2)$, $D = Eh^3/12(1-\nu^2)$, $\lambda^2 = 12\kappa^2/h^2$ and h is the plate thickness, while κ^2 is the shear correction factor used to weight the values of the transverse shear stress $\sigma_{\alpha 3}$. The equilibrium equations are obtained from the Principle of Virtual Work and written in terms of resultant stress:

$$\begin{aligned} N_{\alpha\beta,\beta} + f_\alpha &= 0 \\ \left(N_{\alpha\beta}u_{3,\alpha}\right)_{,\beta} + Q_{\alpha,\alpha} + q_3 &= 0 \\ M_{\alpha\beta,\beta} - Q_\alpha + m_\alpha &= 0 \end{aligned} \tag{3}$$

where $N_{\alpha\beta}$ are the in-plane (membrane) forces, Q_α are the shear forces and $M_{\alpha\beta}$ are the bending moments. The symbols f_α and q_3 stand for the in-plane and transverse loadings, respectively, while m_α are the distributed moments. The boundary conditions associated to eqs.(3) are of three types, namely, in-plane (l_α), out of plane (t_3) and binaries (t_α), given by:

$$\begin{aligned} l_\alpha &= N_{\alpha\beta}n_\beta \ \text{ at } \Gamma_u \ , \ \text{ or } \ \bar{l}_\alpha = \bar{N}_{\alpha\beta}n_\beta \ \text{ at } \Gamma_t \\ t_3 &= N_{\alpha\beta}\bar{u}_{3,\alpha}n_\beta + Q_\alpha n_\alpha \ \text{ at } \Gamma_u \ , \ \text{ or} \\ \bar{t}_3 &= \bar{N}_{\alpha\beta}u_{3,\alpha}n_\beta + \bar{Q}_\alpha n_\alpha \ \text{ at } \Gamma_t \ , \\ t_\alpha &= M_{\alpha\beta}n_\beta \ \text{ at } \Gamma_u \ , \ \text{ or } \ \bar{t}_\alpha = \bar{M}_{\alpha\beta}n_\beta \ \text{ at } \Gamma_t \ , \end{aligned} \tag{4}$$

where Γ_u is the portion of the boundary where the Dirichlet boundary conditions are imposed, whereas Γ_t is the portion where the Neumann boundary conditions are specified, and n is the outward vector normal to the boundary. Aiming an unified numerical implementation for both plate models the expression for the moments is rewritten with an additional term (Westphal Jr. et al., 1998):

$$M_{\alpha\beta} = D\frac{1-\nu}{2}\left[u_{\alpha,\beta} + u_{\beta,\alpha} + \frac{2\nu}{1-\nu}\psi_{\gamma,\gamma}\delta_{\alpha\beta}\right] + m_f \; q_3\,\delta_{\alpha\beta} \quad . \tag{5}$$

where $m_f = \nu/(1-\nu)\lambda^2$ for the Reissner's plate model and $m_f = 0$ for the Mindlin's model. It is convenient to distinguish the linear and the non-linear parts:

$$\begin{aligned} N_{\alpha\beta} &= N^l_{\alpha\beta} + N^n_{\alpha\beta} \\ Q_\alpha &= Q^l_\alpha + Q^n_\alpha \end{aligned} \tag{6}$$

so that

$$\begin{aligned} N^l_{\alpha\beta} &= C\frac{1-\nu}{2}\left[v_{\alpha,\beta} + v_{\beta,\alpha} + \frac{2\nu}{1-\nu}v_{\gamma,\gamma}\delta_{\alpha\beta}\right] \\ N^n_{\alpha\beta} &= C\frac{1-\nu}{2}\left[u_{3,\alpha}u_{3,\beta} + \frac{\nu}{1-\nu}u_{3,\gamma}u_{3,\gamma}\delta_{\alpha\beta}\right] \\ Q^l_\alpha &= D\lambda^2\frac{1-\nu}{2}\left(u_\alpha + u_{3,\alpha}\right) \\ Q^n_\alpha &= N_{\alpha\beta}\,u_{3,\beta} \quad , \end{aligned} \tag{7}$$

The Navier equations are found by substituting the above equations for N, M, and Q in eqs.(3), and transferring all the non-linear terms to the loading terms, leading to the following general form:

$$\begin{bmatrix} {}^m\mathbf{L} & \mathbf{0} \\ \mathbf{0} & {}^f\mathbf{L} \end{bmatrix} \begin{Bmatrix} {}^m\mathbf{u} \\ {}^f\mathbf{u} \end{Bmatrix} = \begin{Bmatrix} {}^m\hat{\mathbf{q}} \\ {}^f\hat{\mathbf{q}} \end{Bmatrix} \tag{8}$$

Where ${}^m\mathbf{L}$ is the differential operator of the linear membrane equilibrium problem, ${}^f\mathbf{L}$ is the linear bending operator, ${}^m\mathbf{u} = \{v_1\ v_2\}^T$ are the in-plane displacements and ${}^f\mathbf{u} = \{u_1\ u_2\ u_3\}^T$ are the plate displacements. The membrane-bending coupling is implicit in the corresponding pseudo-loadings ${}^m\hat{\mathbf{q}}$ and ${}^f\hat{\mathbf{q}}$, because ${}^m\mathbf{q}^n$ needs the derivatives of the transverse displacement in its evaluation, and ${}^f\mathbf{q}^n$ uses the membrane stresses (see definitions for the non-linear parts):

$$\begin{aligned} {}^m\hat{q}_\alpha &= -\ {}^mF_{\alpha\beta}(\partial_Q)\,{}^mq^l_\beta(Q) + {}^mq^n_\alpha(Q) \\ {}^f\hat{q}_i &= -\ {}^fF_{ij}(\partial_Q)\,{}^fq^l_j(Q) + {}^fq^n_i(Q) \end{aligned} \tag{9}$$

The complete expressions of the terms used in Eqs.(8) and (9) are given by:

$$ {}^{m}\mathbf{L}(\partial_Q) = C\frac{1-\nu}{2}\begin{bmatrix} \Delta + \overline{\nu}\dfrac{\partial^2}{\partial x_1^2} & \overline{\nu}\dfrac{\partial^2}{\partial x_1 \partial x_2} \\ \overline{\nu}\dfrac{\partial^2}{\partial x_1 \partial x_2} & \Delta + \overline{\nu}\dfrac{\partial^2}{\partial x_2^2} \end{bmatrix} $$

$$ {}^{f}\mathbf{L}(\partial_Q) = D\frac{1-\nu}{2}\begin{bmatrix} \Delta - \lambda^2 + \overline{\nu}\dfrac{\partial^2}{\partial x_1^2} & \overline{\nu}\dfrac{\partial^2}{\partial x_1 \partial x_2} & -\lambda^2\dfrac{\partial}{\partial x_1} \\ \overline{\nu}\dfrac{\partial^2}{\partial x_1 \partial x_2} & \Delta - \lambda^2 + \overline{\nu}\dfrac{\partial^2}{\partial x_2^2} & -\lambda^2\dfrac{\partial}{\partial x_2} \\ \lambda^2\dfrac{\partial}{\partial x_1} & \lambda^2\dfrac{\partial}{\partial x_2} & \lambda^2\Delta \end{bmatrix} $$

$$ {}^{m}\mathbf{F}(\partial_Q) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} $$

$$ {}^{f}\mathbf{F}(\partial_Q) = \begin{bmatrix} 1 & 0 & m_f\dfrac{\partial}{\partial x_1} \\ 0 & 1 & m_f\dfrac{\partial}{\partial x_2} \\ 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{m}\mathbf{q}^{l}(Q) = \begin{Bmatrix} f_1 & f_2 \end{Bmatrix}^T $$

$$ {}^{m}\mathbf{q}^{n}(Q) = C\frac{1-\nu}{2}\begin{Bmatrix} (u_{3,1}u_{3,\alpha})_{,\alpha} + \dfrac{\nu}{1-\nu}(u_{3,\gamma}u_{3,\gamma})_{,1} \\ (u_{3,2}u_{3,\beta})_{,\beta} + \dfrac{\nu}{1-\nu}(u_{3,\gamma}u_{3,\gamma})_{,2} \end{Bmatrix} $$

$$ {}^{f}\mathbf{q}^{l}(Q) = \begin{Bmatrix} q_1 & q_2 & q_3 \end{Bmatrix}^T $$

$$ {}^{f}\mathbf{q}^{n}(Q) = D\frac{1-\nu}{2}\begin{Bmatrix} 0 & 0 & (N_{\alpha\beta}u_{3,\beta})_{,\alpha} \end{Bmatrix}^T $$

where $\bar{\nu} = (1+\nu)/(1-\nu)$ and $\Delta = \partial^2/(\partial x_\alpha \partial x_\alpha)$. Equations (8) describe an in-plane elastic problem coupled to a moderately thick plate elastic bending problem due to the consideration of large displacement terms.

3 NUMERICAL IMPLEMENTATION

By means of the weighted residual method, the following Somigliana identities are obtained for eqs.(8), using the steps detailed in Marczak (1998, 2004):

$$\begin{aligned} &{}^{m}C_{\alpha\beta}(p)v_\beta(p) + \int_\Gamma {}^{m}T_{\alpha\beta}(q,p)v_\beta(q)\,d\Gamma_q = \int_\Gamma {}^{m}U_{\alpha\beta}(q,p)l_\beta(q)\,d\Gamma_q \; + \\ &+ \int_\Omega {}^{m}V_{\alpha\beta}(Q,p)f_\beta(Q)\,d\Omega_Q - \\ &+ \int_\Omega {}^{m}U_{\alpha\beta,\gamma}(Q,p)N^n_{\beta\gamma}(Q)\,d\Omega_Q + {}^{m}v_\alpha(p) \;, \end{aligned} \tag{10}$$

and

$$\begin{aligned} &{}^{f}C_{ij}(p)u_j(p) + \int_\Gamma {}^{f}T_{ij}(q,p)u_j(q)\,d\Gamma_q = \int_\Gamma {}^{f}U_{ij}(q,p)t_j(q)\,d\Gamma_q \; + \\ &+ \int_\Omega {}^{f}V_{ij}(Q,p)q_j(Q)\,d\Omega_Q + \\ &- \int_\Omega {}^{f}U_{i3,\beta}(Q,p)N_{\alpha\beta}(Q)u_{3,\alpha}(Q)\,d\Omega_Q + {}^{f}v_i(p) \;, \end{aligned} \tag{11}$$

for the membrane and the bending problem, respectively. The non-integral terms ${}^{m}v_\in$ and ${}^{f}v_i$ were included to account for concentrated loads inside the domain (Kamiya & Sawaki, 1985). The symbols p and q denote source (collocation) and field points, respectively (lower case indicates boundary points and upper case indicates domain points). The corresponding displacement (${}^{m}U_{ij}$ and ${}^{f}U_{ij}$) and traction (${}^{m}T_{ij}$ and ${}^{f}T_{ij}$) fundamental solutions can be found elsewhere (de Barcellos & Westphal Jr., 1992)(Westphal et al., 2001). Eqs.(10) and (11) can be particularized for internal points by making ${}^{m}C_{\alpha\beta} = \delta_{\alpha\beta}$ and ${}^{f}C_{ij} = \delta_{ij}$. They require the evaluation of the derivatives of the transverse displacement which are present in the nonlinear membrane forces of the last integrals of both. These terms are partially resposible for the membrane-bending coupling. In that non-linear form, eqs.(10-11) were first derived by the author (Marczak, 1998), while its linear form unifying both the Mindlin and the Reissner plate models were first published earlier (de Barcellos & Westphal Jr., 1992). This form uses a factor that specifies which thick plate model is being used, and has been adopted by many researches since then.

In domain methods like finite elements, it is typical to employ the derivatives of the shape functions, i.e. $u_{i,\alpha} = \varphi_{i,\alpha} u_i$, where φ_i are the shape functions. This approach is sometimes criticized in BEM articles because it holds a strong dependence of the domain cells shape functions. One should note that, in spite of being very simple, this approach may generate poor results when the global shape function is not able to represent accurately the displacement fields. Besides, the use of constant domain cells is not possible as all derivatives are null.

The use of the derivative boundary integral equations associated to eqs.(10-11) is far more accurate than to assume an *a priori* interpolatory form for the displacements derivatives. Hence a more rigorous solution can be obtained by differentiation of these integral equations with respect to the coordinates $x_\alpha(P)$. The procedure leads to six additional integral equations for $v_{\beta,\alpha}$ and $u_{3,\alpha}$. Assuming that the derivatives fields are required only at internal points, the differentiation of the boundary integrals in eqs.(10-11) under the integration sign is straightforward as their kernels become all regular. However, the differentiation of the domain integrals must be treated by means of the Leibnitz formula (Bui, 1978). The formal derivation of such derivative integral equations produce the so called convective terms which must be added to the final expressions for $v_{\beta,\alpha}(P)$ and $u_{3,\alpha}(P)$, resulting:

$$\begin{aligned} v_{\beta,\alpha}(P) - \int_\Gamma {}^m T_{\beta\gamma,\alpha}(q,P) v_\gamma(q)\, d\Gamma_q &= -\int_\Gamma {}^m U_{\beta\gamma,\alpha}(q,P) l_\gamma(q)\, d\Gamma_q \; + \\ -\int_\Omega {}^m V_{\beta\gamma,\alpha}(Q,P) f_\gamma(Q)\, d\Omega_Q &+ \int_\Omega {}^m U_{\beta\gamma,\delta\alpha}(Q,P) N^n_{\gamma\delta}(Q)\, d\Omega_Q \; + \\ +\; {}^m c^N_{\alpha\beta}(P) &- {}^m v_{\beta,\alpha}(P) \end{aligned} \tag{12}$$

$$\begin{aligned} u_{3,\alpha}(P) - \int_\Gamma {}^f T_{3i,\alpha}(q,P) u_i(q)\, d\Gamma_q &= -\int_\Gamma {}^f U_{3i,\alpha}(q,P) t_i(q)\, d\Gamma_q \; + \\ -\int_\Omega V_{3i,\alpha}(Q,P) q_i(Q)\, d\Omega_Q &+ \int_\Omega {}^f U_{33,\alpha\gamma}(Q,P) N_{\gamma\beta}(Q) u_{3,\beta}(Q)\, d\Omega_Q \; + \\ +\; c^N_{\alpha\beta}(P) u_{3,\beta}(P) &- {}^f v_{3,\alpha}(P) \end{aligned} \tag{13}$$

where a negative sign was added to all the integrals as the derivatives are taken with respect to $x_\alpha(P)$ and (Marczak, 2004)

$$\begin{aligned} {}^m c^N_{\alpha\beta}(P) &= \frac{-1}{8G(1-\nu)} \Bigg[(3-4\nu)\delta_{\alpha\delta}\delta_{\beta\gamma} - \delta_{\alpha\beta}\delta_{\gamma\delta} - \delta_{\alpha\gamma}\delta_{\beta\delta} \; + \\ &\qquad - \frac{1}{4}\delta_{\alpha\beta}\delta_{\gamma\delta}\left(1+2\delta_{\alpha\gamma}\right) \Bigg] N^n_{\gamma\delta}(P) \;, \\ {}^f c^N_{\alpha\beta}(P) &= -\frac{\delta_{\alpha\gamma}}{D(1-\nu)\lambda^2} N_{\gamma\beta}(P) \end{aligned} \tag{14}$$

are the aforementioned convective terms. Eqs. (12-13) are truly hypersingular derivative integral equations and, as such, one can expect the same convergence properties of any equilibrium equations written in its strong form.

Using the traditional collocation process, eqs.(10), (11), (12) and (13) lead to the following set of algebraic equations:

- Membrane (2D elasticity) problem:

$$ {}^{m}\mathbf{H}\,{}^{m}\mathbf{u} = {}^{m}\mathbf{G}\,{}^{m}\mathbf{t} + {}^{m}\mathbf{B} + {}^{m}\mathbf{f} \tag{14} $$

- Bending problem:

$$ {}^{f}\mathbf{H}\,{}^{f}\mathbf{u} = {}^{f}\mathbf{G}\,{}^{f}\mathbf{t} + {}^{f}\mathbf{B}\,\bar{\mathbf{u}}_3 + {}^{f}\mathbf{f} \tag{15} $$

- In-plane displacements derivatives:

$$ \bar{\mathbf{u}}_\beta + {}^{\beta}\mathbf{H}\,{}^{m}\mathbf{u} = {}^{\beta}\mathbf{G}\,{}^{m}\mathbf{t} + {}^{\beta}\mathbf{B} + {}^{\beta}\mathbf{f} \tag{16} $$

- Transverse displacement derivatives:

$$ \bar{\mathbf{u}}_3 + {}^{3}\mathbf{H}\,{}^{f}\mathbf{u} = {}^{3}\mathbf{G}\,{}^{f}\mathbf{t} + {}^{3}\mathbf{B}\,\bar{\mathbf{u}}_3 + {}^{3}\mathbf{f} \ . \tag{17} $$

where $\bar{\mathbf{u}}_\beta = \left\{ v_{\beta,1} \quad v_{\beta,2} \right\}^T$ and $\bar{\mathbf{u}}_3 = \left\{ u_{3,1} \quad u_{3,2} \right\}^T$.

3.1 Non-linear Bending

All types of B matrices refer to the membrane-bending coupling, given by the last integral in eqs.(10-13). Equations (14-17) must be solved by an iterative procedure. Figure 1 exemplifies a generic k-th iteration. The convergence was verified against the norm of the curvature error, i.e. $\left\| \left(u_{3,\alpha}\right)_k - \left(u_{3,\alpha}\right)_{k-1} \right\|$, since it showed to be dominant in numerical experiments.

The steps 4 and 8 in Fig.1 assume the application of eqs.(16) e (17). Nevertheless, they are very difficult to extend to other applications. For instance, the inclusion of material non-linearities or thermal effects would imply in new integral terms and possibly new convective terms, resulting in a very cumbersome, application specific numerical implementation.

Anticipating the use of discontinuous domain cells - whose physical nodes never rest on the boundary - eqs.(16-17) will be quasi-singular at most, and a finite difference scheme can be used instead of the true derivatives of the displacements. Finally, it is worth to mention that a full update of the displacement derivatives enables convergence only for mild nonlinearities, so that a relaxation factor must be employed at the end of each iteration to stabilize the process:

$$ \left\{ u_{i,\alpha} \right\}_{k+1} = \rho \left\{ u_{i,\alpha} \right\}_k + (1-\rho) \left\{ u_{i,\alpha} \right\}_{k-1} $$

1. Estimate $\left(\bar{\mathbf{u}}_3\right)_{k-1}$.
2. Evaluate $N^n_{\alpha\beta}$ using eq.(7) and assemble ${}^m\mathrm{B}$.
3. Solve the membrane BVP – eq.(14).
4. Evaluate $\bar{\mathbf{u}}_\beta$ at internal points using eq.(16).
5. Evaluate the total force $N_{\alpha\beta}$ through eq.(6) using the results of steps 2 and 4.
6. Assemble ${}^f\mathbf{B}$.
7. Solve the bending BVP – eq.(15).
8. Solve eq.(17) to obtain a new estimate $\left(\bar{\mathbf{u}}_3\right)_k$ for the curvatures.
9. Return to step 2 if $\left\|\left(\bar{\mathbf{u}}_3\right)_k - \left(\bar{\mathbf{u}}_3\right)_{k-1}\right\| > \varepsilon$.

Figure 1: General algorithm to solve non-linear problems described by eqs. (14-17).

The resulting systems of equations for each class of problem (membrane, bending and displacement derivatives) are summarized below.

3.2 Linear Bending

The equations here are obviously decoupled, and eq.(15) becomes:

$$ {}^f\mathbf{H}\,{}^f\mathbf{u} = {}^f\mathbf{G}\,{}^f\mathbf{t} + {}^f\mathbf{f} \ . $$

Collecting plate displacements and tractions in a vector x results:

$$ {}^f\mathbf{A}\,{}^f\mathbf{x} = {}^f\mathbf{f} $$

3.3 Linearized Stability

In this case, although there is a coupling between the bending and the plane problems, the latter is parameterized by an in-plane load factor $\mathbf{N}^* = \lambda\mathbf{N}$, so that the only remaining unknown of the membrane problem is λ. Therefore the system of equations for buckling problems are derived as particularization of the equations for non-linear bending. Since there is no other loadings, eqs.(15) and (17) are rewritten and regrouped as

$$ {}^f\mathbf{A}\,{}^f\mathbf{x} = \lambda\,{}^f\mathbf{B}\,\bar{\mathbf{u}}_3 $$

$$ \bar{\mathbf{u}} + {}^3\mathbf{A}\,{}^f\mathbf{x} = \lambda\,{}^3\mathbf{B}\,\bar{\mathbf{u}}_3 $$

When combined, the above equations result the eigenproblem:

$$ \left[\,{}^3\mathbf{A}\,{}^f\mathbf{A}^{-1}\,{}^f\mathbf{B} - {}^3\mathbf{B}\right]\bar{\mathbf{u}}_3 = \frac{1}{\lambda}\bar{\mathbf{u}}_3 $$

The smallest eigenvalue $1/\lambda$ gives the critical load, while the eigenvectors $\bar{\mathbf{u}}_3$ can be inserted in (15) to retrieve their extensions ${}^f\mathbf{x}$ which represent the displacement and traction pattern for each

buckling load. Of note is the fact that in BEM, stability problems does not result in a generalized eigenproblem like in FEM, but a classical one encompassing one single matrix. The corresponding displacements for internal points are obtained through the bending Somigliana identity particularized for null transverse loading.

3.4 Numerical Integration

Concerning numerical integration of the singular and quasi-singular kernels found in the present formulation, during the early stages of this work in the 1990s, most of the results obtained by the present formulation used to rely on the integration of singular kernels by rigid body movement (RBM) imposition or Kutt's quadrature (Kutt, 1975), while weakly singular kernels were integrated with Telles' cubic transformation (Telles, 1987). Even considering the limitations of both, the results were very good. However, the RBM method always depend on the quality of the integration over the adjunct part of the boundary (although it always guarantees the fulfillment of equilibrium) while Kutt's quadrature is very difficult to be used with curved elements. During early 2000, the authors greatly improved the quality and the extentensibility of the formulation by deriving the asymptotic expansions of the relevant kernels for 2D elasticity and thick plates and implemented the direct method (Guiggiani et al. 1992) (Guiggiani, 1998) to evaluate all strongly singular integrals. The efficiency of this approach in regularizing the singular integrals is demonstrated in Marczak & Creus (2002).

4 IMPLEMENTATION ISSUES

The continued development of BEM codes has naturally lead to a demand for extensibility and reusability of the codes (or part of them) without demanding the costs associated to the development of new software or due to unwanted changes in source codes successfully tested and used. The present work was developed under an object-oriented architecture used as a general numerical framework for the development of computer programs based on boundary integral equation methods. A number of classes were developed to automatize ordinary tasks like collocation procedures, numerical integration, degrees of freedom mapping, among several others. The framework is also capable of handling an arbitrary number of subregions. The design was able to successfully unlink the so-called domain classes (those containing elements, nodes, loads, etc.) from the analysis classes (linear, non-linear, static, transient etc.).

A description of the OO philosophy will not be covered herein. It is supposed a basic knowledge in OO programming as well as its fundamental aspects (classes and objects, inheritance and polymorphism). Classes hierarchy and relationship will be illustrated following Rumbaugh notation.

The OO design used herein (mcBEM) is focused primarily on the solution of boundary value problems by means of boundary integral equation methods. The mcBEM library started as a research project centered on applying OO programming to develop flexible, modular, and reusable software components for solving differential equations by the BEM. The version presented here represent an extended version of the previous one (Marczak 2004b) which has been successfully used to solve a variety of BEM problems, and fits better to standard implementations of the meth-

od, including non-linear applications. The underlying idea in its design is that different applications share a common mathematical and numerical structure, and more importantly, storage (model) classes do not perform any solution step. The library provides a complete set of auxiliary and model classes, as well as a basic set of analysis classes. To create an application, the analyst assembles the code by collecting the objects necessary to perform the solution. The programming work is limited to implement the new analysis classes deriving them from any of the existing ones, in case they are not provided. In the presently proposed OO architecture, a tentative organization was advised for the analysis classes (and their descendents) in such a way that the most important analysis tasks are logically grouped regarding the differential equation point of view. In this way, the analyst is able to assemble a code according to the problem to be solved: linear or nonlinear, static or dynamic, type of resulting system of equations, solution strategy and so forth. A brief description of some important auxiliary, model and analysis classes used to generate the results of the present work is given below.

Class mcGeometricPartition: The construction of any domain partition (boundary elements or domain cells) is based in a composition of geometric and physical partitions (see fig.2). The mcGeometricPartition class implements the topology of the partition. It is composed basically by a list of points and the associated shape functions. This approach enables the implementation of less conventional geometric mappings, like cubic splines and Bézier curves. Methods for evaluation of Jacobians, dimension of the normalized space, mapping to or from the normalized space, etc. are provided. From this superclass other classes used to define lines, areas and volumes descend. For example, mc1DGeometricPartition and mc2DGeometricPartition are used to define lines and areas, respectively.

Class mcPhysicalPartition: Similar to mcGeometricPartition, but in this case the object has the connectivity composed by a list of physical nodes. Examples of its descendents are the mc1DPhysicalPartition and mc2DPhysicalPartition classes, used to implement physical interpolations on lines and areas respectively.

Other classes like mcDOF and mcDOFSet encapsulate data for a single degree of freedom and for a list of mcDOF objects. The user can assemble a particular set of variables according to the problem to be solved. For use with the BEM, this class generally encapsulates properties for primal and dual variables, and their derivatives as well.

Class mcFundSolution: This class was designed to encapsulate data and methods necessary to describe the type of differential equation, which governs a BEM subregion, as well as a corresponding set of fundamental solution tensors. Once known what is the differential equation is to be solved, the associated mcFundSolution object keeps track of how many and which are the primal and dual DOFs, the order of the system of differential equations, and so forth. Examples of these methods are:

• Matrix Tensor(char* id, mcCoord& q, Vector& n): Evaluates a fundamental tensor (T,U, etc.) using the current load point.

• Matrix Limit(int ns, char* id, Vector& N, Vector& T, Matrix& Shape, Matrix& dShape): Evaluates the asymptotic expansions of a tensor to be used in singular integrations.

• Matrix Jump(Vector& ang, char* id): Evaluates the free terms of a tensor. The most common case are the geometric factors. Other types of jump terms can be considered as well.

• mcArray getSingType(char* id): Returns an array of codes corresponding to the type and severity of the singularity of each component of a tensor.

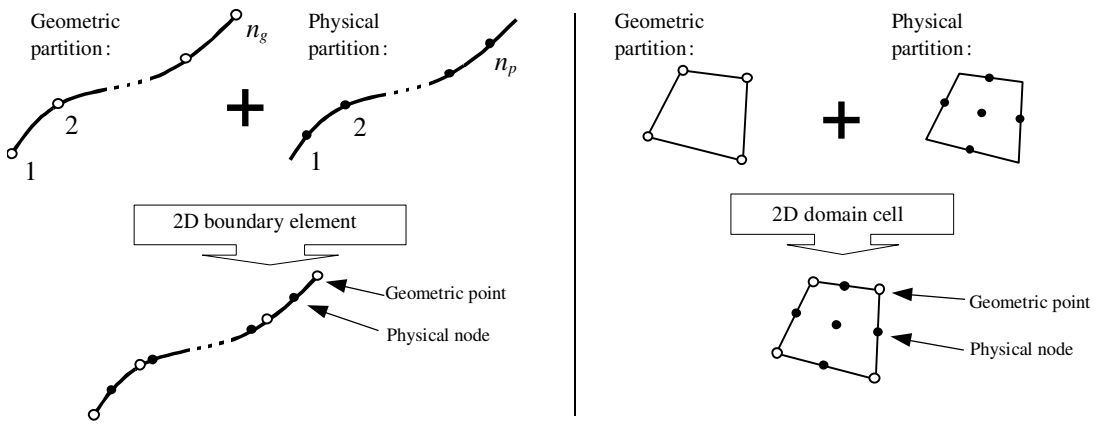


Figure 2: Illustration of the composition of two-dimensional domain partitions using geometric and physical partitions.

The analysis of the problem itself is performed by a set of five superclasses aggregated by the analyst, depending on the type of the problem and solution desired. Many of the ideas adopted here were adapted from a previous work on finite elements (McKeena, 1997). This approach is quite flexible as it enables the user to group each of these five major classes according to the specific needs of an application. If necessary, one can implement a new class by deriving it from any of the superclasses and limiting the coding task to those analysis steps not provided by the library.

Class mcSolutionAlgorithm: This superclass is the responsible for orchestrating the major solution steps. Its main task is to coordinate the assembly of the right and left hand sides of the resulting system of equations, and to trigger its solution. This is accomplished by invoking the proper methods for each type of analysis. In most linear problems, these calls are executed just once, but in nonlinear or transient problems the operation must be repeated until convergence is achieved. In the present work, two main subclasses are derived: mcBEMEigenvalueSolAlgo and mcBEMEquilibriumSolAlgo. Two examples of static problems were used to generate the results of the present work, namely the main subclasses mcBEMLinear and mcBEMLargeDisp, for elastic linear and geometrically nonlinear problems, respectively. Three methods compose the basic interface of mcSolutionAlgorithm objects, as described below:

• int solveCurrentStep():Controls the assembly of left and right hand sides of the system of equations for the current time and load steps. In the BEM subclasses, this is generally related to the solution of the associated boundary value problem (BVP). Figure 3 shows a typical implementation of the solveCurrentStep member for linear problems. Its counterpart for nonlinear cases is illustrated in fig.4.

• int postProcessCurrentStep():Performs the post-processing of the current step. For example, in BEM applications the evaluation of the remaining variables at internal points is carried out using this method.

```
int mcBEMLinear::solveCurrentStep() {

  theAssembler->formBVPTangent();
  theAssembler->formBVPUnbalance();
  theConstraint->applyConstraints();
  theSysOfEqs->solve();
  theAssembler->updateBVPDOF(theSysOfEqs->getX());

  return 0;

}
```

Figure 3: Code fragment of the solveCurrentStep member for linear applications.

```
int mcBEMLargeDisp::solveCurrentStep() {

  do {

    theAssembler->formBVPTangent();
    theAssembler->formBVPUnbalance();
    theConstraint->applyConstraints();
    theSysOfEqs->solve();
    theAssembler->updateBVPDOF(theSysOfEqs->getX());
    theAssembler->evaluateInternalPoints();
    theAssembler->evaluateGradients();

    // Evaluate the error ...

  } while (error > tol);

  return 0;
}
```

Figure 4: Code fragment of the solveCurrentStep member for non-linear applications.

Class mcAssembler: An object mcAssembler provides the methods necessary to form the system of equations. It is responsible for accessing the boundary elements and domain cells, and for adding their contributions to the global system of equations. Its major subclass is the mcBEMIncrementalAssembler class, from which others like mcBEMStaticAssembler (for static problems), mcBEMTransientAssembler (for transient problems) and mcBEMEigenvalueAssembler (for eigenproblems) descend. Figure 12 shows the basic hierarchy.

Similar classes can be easily implemented for finite elements. In the current framework, all BEM applications are based on the solution of the boundary value problem and possibly a corresponding domain value problem. It is assumed that even in nonlinear problems, the nonlinear contributions can be included in the right hand side of the system of equations. As a consequence, the relevant methods are those responsible for the assembly of the final solution system of equations. Some of the relevant methods currently implemented are described below.

- int formBVPTangent():Responsible for the assembly of the left hand side of the BVP system of equations. In the BEM, this is generally accomplished by imposing the boundary conditions and grouping the contributions of the matrices H and G during the collocation process. It is worth to note that, unlike most FEM formulations, part of the right hand side is evaluated during this phase.

- int formBVPUnbalance():Here the assembly of the right hand side of the system is finished by adding the domain contributions, such as domain loadings and nonlinear terms. This method is not used in pure boundary value problems.

As an example, the basic code of the method formBVPTangent is listed in figs.5. It is worth to point out that, besides their simplicity, these members remain the same for linear and nonlinear applications, provided the nonlinear terms are handled by a member formBVPUnbalance.

```
1:  int mcBEMStaticAssembler::formBVPTangent() {

2:   theSOE->reset(theConstraint->getNumberOfBVPDOFs());
3:   while (SR) {                // Loop over the subregions:
4:     mcBElementITR BE (theModel->getBElementsSharing(*SR));
5:     mcNodeITR ND (theModel->getBVPCollocationPointsFor(*SR));
6:     while (BE) {               // Loop over the elements:
7:       while (ND) {             // Loop over the collocation nodes:
8:         SR->getFundSolution().setLoadPoint(coor);
9:         sing_node = theModel->detectSingularNode(*SR,*BE,*ND);
10:        BE->formMatrix(H,"T",sing_node);
11:        BE->formMatrix(G,"U",sing_node);
12:        theConstraint->applyBC(*BE, *ND, H, G);
13:        ND ++;
14:      }
15:      BE ++;
16:    }
17:    SR ++;
18:  }

19:  return 0;

20: }
```

Figure 5: Code fragment of the formBVPTangent member for linear static applications.

Class mcAnalysis: This is the analysis superclass of the present OO design. A mcAnalysis object is actually an aggregation of the analysis objects described above. In spite of the fact that it does not perform any explicit calculation, the mcAnalysis class defines which type of problem will be solved

and how. It is also responsible for checking the validity of the objects in the aggregation so that it makes sense. A single virtual method called analyze() triggers the analysis execution. The computational model is constantly checked to verify if another analysis is necessary. Figure 6 shows some examples of analysis subclasses derived in the present work.

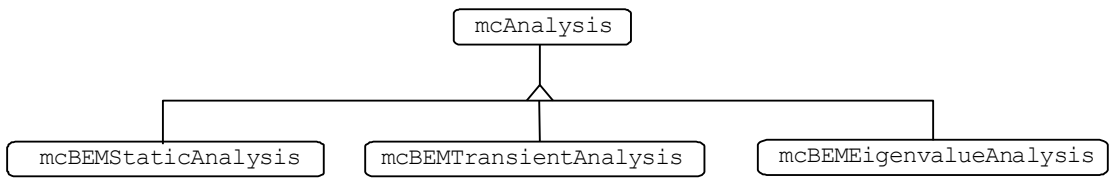


Figure 6: The two major hierarchy levels of the mcAnalysis class.

The classes introduced in this work can be used in an stand-alone fashion as tools incorporated to other computer codes. But the advantages of the suggested approach become more evident when all the major classes are connected to generate a given application. Figure 7 depicts a diagram of a complete application, showing the relationship between the major classes. Although other variations are possible, this basic framework has been proving to be sufficiently general for most cases.

One of the main goals of the OO design proposed in this work is to enable the analyst to write a few programming lines to built a basic BEM code, and yet be able to customize the code for other applications with little additional programming effort. Thus, even more complex analysis would be straightforward reusing the relevant objects. For example, if the analyst were interested in a linear elastic analysis, then the code fragment shown in fig.19 would suffice. Note that the solution of, say, a linear static plate bending or a steady state heat conduction problem would use the same code (the inherent differences are hidden in the subregion objects). But if the interest is to solve the same problem using a dynamic transient analysis, it is necessary to change only lines 4, 5 and 10 in the original code of fig.19 according to fig.20.

A comparison of the codes shown in figs.8 and 9 helps to illustrate the uncoupling of the analysis classes from the model classes. This uncoupling was one of the goals aimed by the present OO design from the outset. Once the analysis subclasses are adequately designed, this uncoupling is always achieved since the type of analysis can be changed without modifying the model classes. In essence, a new analysis will possibly differ from the previous one by needing another solution algorithm, a different matrix assembly sequence, another system of equations type/solver, etc. This is evident in figs.8-9 because the lines 6-9 remained unchanged after switching from a linear elastic analysis to a transient dynamic analysis.

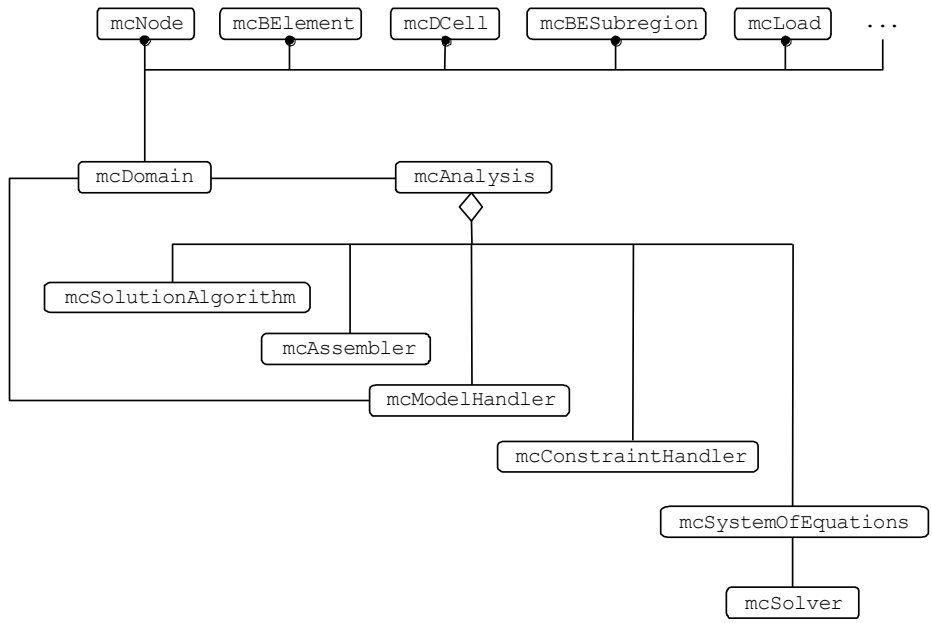


Figure 7: Minimal class diagram of a complete application code.

```
1.  mcDomain  theDomain;

2.  theDomain.readInputFile("2Dproblem.dat");
3.  theDomain.setCurrentLoadCase(theDomain.getDefaultLoadCase());

4.  mcBEMLinear            theSolAlgo;
5.  mcBEMStaticAssembler   theAssembler;
6.  mcBEMModelHandler      theModel;
7.  mcBEMConstraintHandler theConstraint;
8.  mcDenseNonSymLS        theSolver;
9.  mcDenseNonSymLSOE      theSysOfEqs(theSolver);
10. mcBEMStaticAnalysis    theAnalysis(theDomain,theSolAlgo,
                                       theAssembler,theModel,
                                       theConstraint,theSysOfEqs);

11. theAnalysis.analyse();
```

Figure 8: Code for a BEM linear elastic analysis.

```
1.  mcDomain  theDomain;

2.  theDomain.readInputFile("2Dproblem.dat");
3.  theDomain.setCurrentLoadCase(theDomain.getDefaultLoadCase());

4.  mcBEMTransientSolAlgo  theSolAlgo;
5.  mcBEMNewmark           theAssembler;
6.  mcBEMModelHandler      theModel;
7.  mcBEMConstraintHandler theConstraint;
8.  mcDenseNonSymLS        theSolver;
9.  mcDenseNonSymLSOE      theSysOfEqs(theSolver);
10. mcBEMTransientAnalysis theAnalysis(theDomain,theSolAlgo,
                                       theAssembler,theModel,
                                       theConstraint,theSysOfEqs);

11. theAnalysis.analyse();
```

Figure 9: Code for a BEM transient dynamic analysis.

Finally, another interesting feature of the present class library is the ability to integrate any algebraic type of integrand and handle singular integrals. The former can be accomplished by using fully template-based definitions. This leads to the concept of numerical integration of objects, which would allow a member function of an arbitrary object to be integrated (provided it overloads the basic algebraic operations) as a Riemann sum:

$$\texttt{Result} = \int_{-1}^{+1} \texttt{Object}(\xi) d\xi = \sum_{i=1}^{K} \texttt{Object}(\xi_i) w_i$$

where Result is of the same type of Object. Since the aforementioned Telles' transformation was chosen here as a primary method to integrate weakly singular integrals, while the direct method was used for the strongly singular ones, only Gaussian stations and weights need to be used.

A proposed mcQuadrature class was developed to assist numerical integration in BEM (Marczak, 2006). Being the class fully template based, the user do not need to take care of the algebraic type of the integrand. In addition, the members invoked to perform the numerical integration remain the same regardless the regularity of the kernel. The user specifies the dimension of the integration domain, the order of the quadrature and the rule to be used. The current integrand is defined by pointers to functions, enabling the integration of any kernel provided it is defined in a standalone subroutine or in an object member function. Since the mcQuadrature object is informed of the subroutine by pointers, there is no need to recompile the code when the integrand is changed. In addition, a single object can be reused as many times as necessary. These features fit particularly well to BEM applications, as shown in the generic code of fig.10. This piece of code is used to generate the global system of equations and any domain loads were disregarded for simplicity's sake. Following a classical BEM implementation two loops are used, one run through the boundary collocation points, and another for the boundary elements. Inside these loops, a member called formMatrix (lines 8-9) integrates a general kernel of the form $\mathbf{A}(p,q)\mathbf{N}(q)J$, where A is a fundamental solution tensor. The second argument in the formMatrix call indicates which one to use

("T" for the traction fundamental solution T etc.). The member applyBC invoked in line 10 enforces the boundary conditions to generate the final system of equations.

Unfortunately, not all fundamental solution tensors behave with the same degree of singularity for all its components. For instance, in the traction fundamental solution of the Mindlin plate model all the components are either regular or weakly singular except for T_{12} and T_{21} which are strongly singular. In such cases, a single call to the NIntegrate member function in order to integrate the whole (matrix) kernel at once would not work properly.

```
1:   be = BEMModel->getBElements();
2:   nd = BEMModel->getBVPCollocationPoints();
3:   while (be) {              // Loop over the elements:
4:     while (nd) {            // Loop over the collocation nodes:
5:       fundamental_solution = be.current->getFundSol();
6:       fundamental_solution.setLoadPoint(nd.coor());
7:       sing_node = BEMModel->detectSingularNode(*be,*nd);
8:       be->formMatrix(H,"T",sing_node);
9:       be->formMatrix(G,"U",sing_node);
10:      theConstraint->applyBC(*be, *nd, H, G);
11:      nd ++;
12:    }
13:    be ++;
14:  }
```

Figure 10: Code excerpt showing the integration and assembly of typical BEM matrices.

In such cases, the formMatrix member of fig.10 has to be implemented in a more elaborate way. Figure 7 illustrates an excerpt of an actual code, where each entry of $\mathbf{A}(p,q)$ is verified against its singularity type, and then a suitable type of quadrature is chosen for each component. Object member functions may have limited visibility or be vulnerable to changes. Since it is interesting to require little commonality between the present class design and the code under development, function wrappers can be used to fit the proposed class into independently developed parts. When using a function wrapper, any changes in the interface of the user's class is reflected only in the definition of its corresponding function wrapper. Other parts of the program are not affected by the change. This is the approach suggested here, particularly for BEM codes like the one shown in fig. 11. Lines 2 and 3 of the code are used to wrap an independent (boundary element class) member function which evaluates BEM matrices.

```
 1: int mcBElement::formMatrix(Matrix& A, char* id, int const sing) {

 2: typedef Matrix (mcBElement::*mcBEInt3)(char*, double, double, double);
 3: mcBEInt3 integrand;

 4:  mcBESubregion*  SR = &this->getOwner();
 5:  mcFundSolution* FS = &SR->getFundSolution();

     Initialization:
 6:  mcArray typ   = this->getOwner().getFundSolution().getSingType(id);
 7:  integrand = &(mcBElement::elemMatrix);
 8:  mcQuadrature<Matrix> Integrator(ncoor, nip);

     Set the initial number of integration points:
 9:  ngp(i) = SR->estimateNIP(i);
10:  Integrator.setOrder(ngp(REGULAR),ngp(REGULAR),ngp(REGULAR));

11:  switch (sing) {
     |
     |  The collocation point is outside the current boundary element:
12:  |  case 0:
13:  |     A = Integrator.BEMIntegrate(*this,integrand, id);
14:  |     break;
     |
     |  The collocation point belongs to the current boundary element:
15:  |  default: {
16:  |  |   for (int i=1; i<= nrows; i++)
17:  |  |     for (int k=1; k<= nrows; k++) {
     |  |
18:  |  |       Integrator.setOrder(ngp(REGULAR),ngp(REGULAR),ngp(REGULAR));
     |  |
     |  |       Check the type of singularity for each entry of the integrand:
19:  |  |       switch (typ(i,k)) {
     |  |
     |  |         Regular entries – Use the standart Gauss-Legendre quadrature:
20:  |  |         default : {
21:  |  |           Integrator.setType(REGULAR,REGULAR,REGULAR);
22:  |  |         } break;
     |  |
     |  |         Weakly singular entries – Use the Telles' transformation [22]:
23:  |  |         case WEAK: {
24:  |  |           Integrator.setOrder(ngp(WEAK),ngp(WEAK),ngp(WEAK));
25:  |  |           Integrator.setType(WEAK,WEAK,WEAK);
26:  |  |         } break;
     |  |
     |  |          Strongly singular entries – Use the direct method [11]:
27:  |  |          case STRONG: {
28:  |  |            Integrator.setOrder(ngp(STRONG),ngp(STRONG),ngp(STRONG));
29:  |  |            A(i,j) += F1(i,j) * CorrTerm1 + F2(i,j) * CorrTerm2;
30:  |  |          } break;
     |  |
31:  |  |       }
32:  |  |       A(i,j) += Integrator.BEMIntegrate(*this,integrand,id)(i,j);
33:  |  |     }
34:  |  }
35:  |  break;
36:  }

     Add the Cij terms for the singular elements:
37:  if (sing) this->addJumpTerm(A, id, sing, angle);

38:  return 0;
39: }
```

Figure 11: Code showing a more elaborate integration procedure to handle different singularities in BEM matrices.

5 RESULTS

The numerical examples presented in this section represent a selection of benchmarks typically used to show the quality of the results in several plate formulations. Results are presented for three groups of problems: linear bending, buckling, and non-linear bending. They not only allow a verification of the accuracy of the BEM scheme described in section 3, but also compare the h convergence rates for boundary elements ranging from constant up to quartic degree with several progenitors of now renowned finite elements. To the authors' knowledge, this has never been published for plate problems in the open literature. Another important class of results shown herein compares all three singular integration schemes developed, and they provide a useful set of reference results for comparison with other numerical formulations, therefore extensive use of tabular results was also made. BEM results for thick plate buckling is another kind of result seldom presented in the literature. The results for non-linear bending of plates with varying thickness seem to be original in the BEM context as well.

Unless specified otherwise, the results presented herein employ the Mindlin's model with the following non-dimensional data: Young modulus $E = 3.0 \times 10^6$, Poisson coefficient $\nu = 0.30$, shear correction factor $\kappa^2 = \pi^2/12$, lateral loading (when applicable) $q_3 = 1000$, lateral dimension of square plates or radius of circular plates $a = 1.0$. It is well known that the FOSD plate theories enable the imposition of two types of boundary conditions: hard support (which prescribes the transverse displacement, the normal rotation and the tangential moment) and soft support (which prescribes the transverse displacement, and both the normal and tangential moments). Each set of boundary conditions leads to different results, and this difference becomes more significant as the plate thickness is increased (Arnold & Falk, 1989). The soft and the hard boundary conditions are herein indicated by SS1 and SS2, respectively (if not indicated, SS1 b.c. is assumed).

5.1 Results for Linear Bending Problems

This section show numerical results for thin and thick square plates. Square thin plate cases have the results normalized with the Navier solution (Timoshenko & Woinowski-Krieger, 1970): $\bar{w} = w_{\max} / w_{\text{Navier}}$ where $w_{\max}$ is the centre deflection of the plate. Tests were performed using 2 boundary elements per side of square plates and per quadrant of circular plates in order to assess the p-convergence rate of the constant up to quartic elements, seldom found in the literature. These plots are shown in fig.12, all obtained with the RBM technique to integrate the singular kernels. Clamped plates result in a system of equations where the contributions of the H matrix are cancelled by the null displacement boundary conditions, while supported plates have the G matrix made null by the zero traction boundary conditions. Because the weakly singular integrals in the G matrix are more easily integrated than the strongly singular ones present in the H matrix, it is common to find slightly better results for clamped plates (for the same mesh).

For thick plate applications, the performance of the present formulation was analyzed by varying the ratio h/a of square plates from 0.05 to 0.25, the center deflection being used as the comparison parameter again. Despite of the large number of proposed finite elements for the Mindlin plate model, the number of analytical or semi-analytical counterpart solutions available is surprisingly

small when compared to the Reissner model. Furthermore, some confusion about the inherent differences of these plate models still persist among many researchers, making the selection of reliable results for normalization even more difficult (Wang et al., 2001). In the present case, the values of central displacement are normalized in the following form: $\bar{w} = w_{\max} qa^4 / Eh^3$, where q is the distributed load.

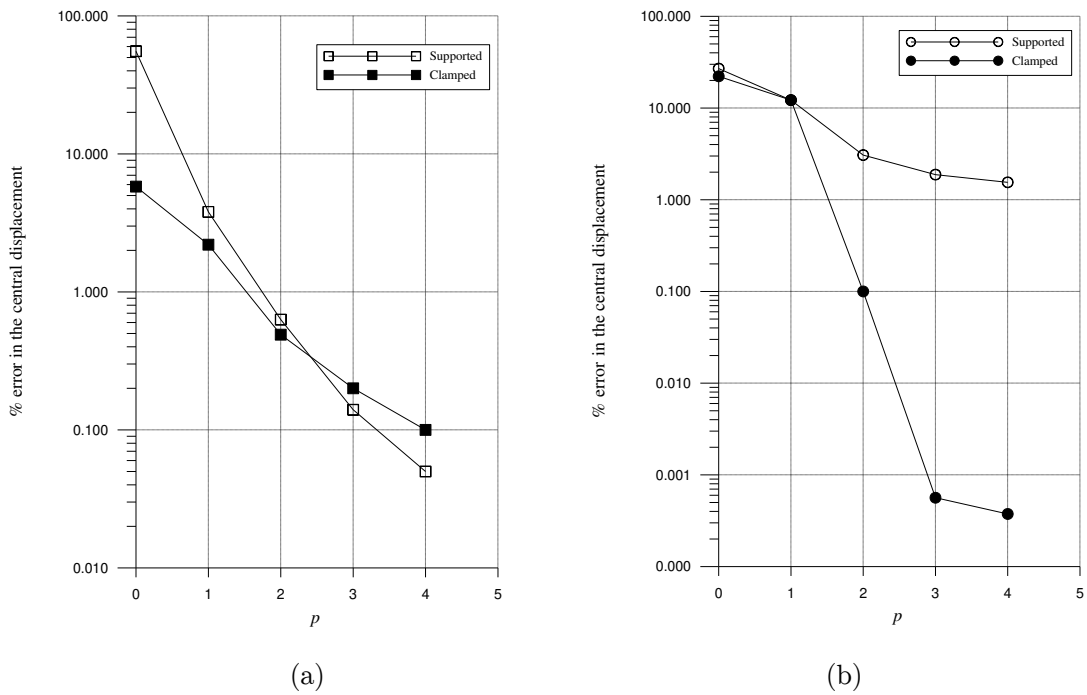


Figure 12: p-convergence rates for plates under uniform loading. (a) square plate. (b) circular plate.

Figure 13 compares linear, quadratic, cubic and quartic elements of the present implementation with several precursor finite elements now found in commercial software for the maximum normalized displacement of a clampled thin square plate obatined with RBM. It is clear why the addtional mathematical complexity of integral equation methods pays off when it comes to accuracy of the results. Similar curves were obtained for boundary forces and moments.

Figure 14 compares the influence of the three integration methods used on the central displacement of a thin supported square plate under lateral loading. A single quadrilateral domain cell was used to integrate the loading. Thin plate ($h/a = 0.015$) was considered. The results show that Kutt's quadrature delivers results with the same level of accuracy than the RBM. They also indicate that there is no significant accuracy improvement for any of the approaches when the constant element is used. On the other hand, the convergence of the direct method seems faster than Kutt's quadrature for linear elements, as shown in Fig. 14b. Figure 14c shows the convergence for quadratic elements, both the rigid body technique and the direct method providing similar accuracy level. However, the accuracy of the linear and quadratic element can be further improved by optimizing the offset of the end nodes, since these are discontinuous elements.

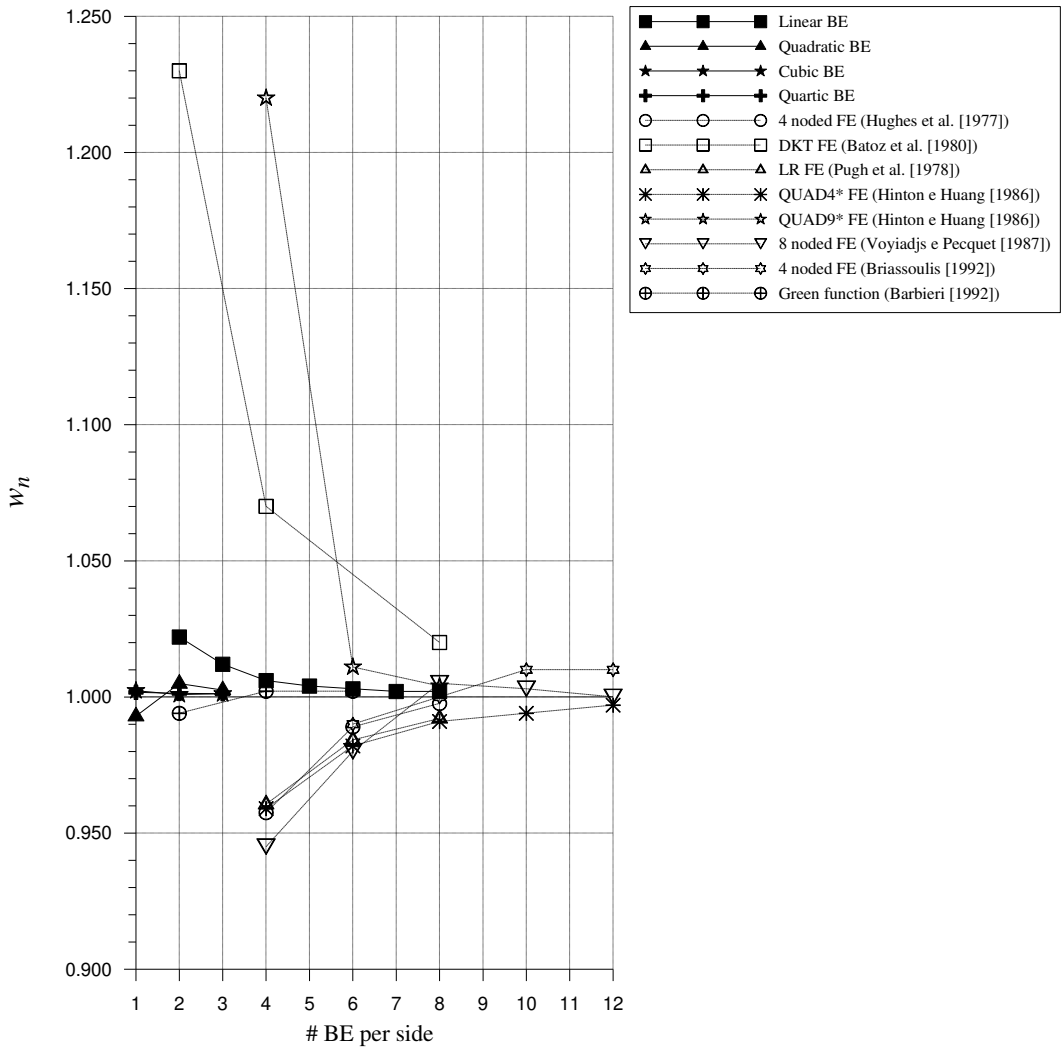


Figure 13: Comparison of convergence curves for several numerical formulations. Clamped square plate under uniform loading. Solid symbols refer to the present results.

Tables 1 and 2 present results of normalized central displacement, obtained earlier by the authors using RBM and Kutt's quadrature. They remain excellent reference results. Results in Table 1 refers to clamped square plate under uniform loading while Table 2 refers to SS1-supported square plate under uniform loading. Present results are compared to the works of (Yuan & Miller, 1988) and (Deshmukh & Archer, 1973), reportedly still some of the best numerical benchmarks for thick plates. Similar results were obtained for concentrated loading, but the distributed loading represents a more harsh test to the method, as it has to be integrated using domain cells or converted to the boundary, whenever possible.

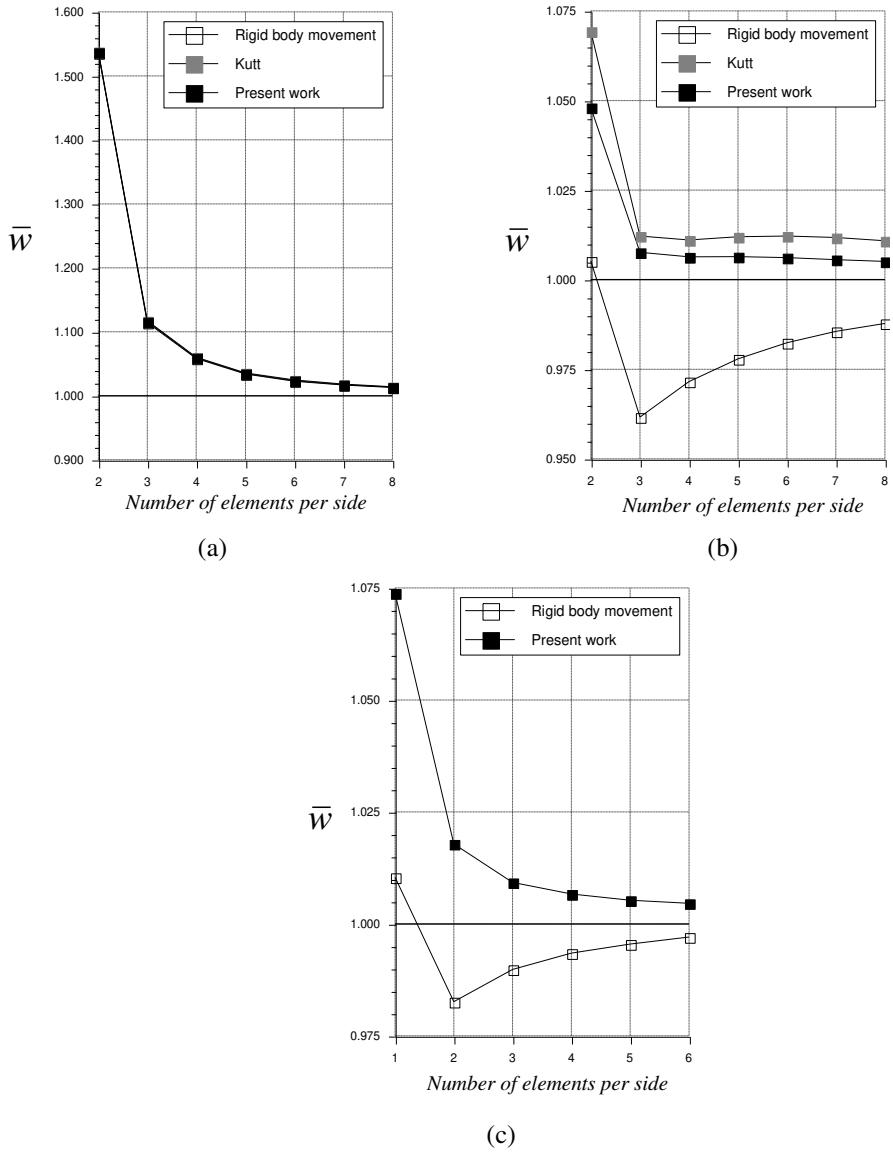


Figure 14: Convergence curves for thin square plates under uniform loading using RBM technique, Kutt's quadrature or the direct method to integrate singular kernels. (a) Constant BE. (b) Linear BE. (c) Quadratic BE.

The present results for SS2-supported square plates under uniform distributed load were normalized against the results of Lee et al. (2002). Figure 15 presents the convergence curves for $h/a = 0.1$ and 0.2. Both cases show very good agreement, producing errors smaller than 1% regardless the mesh used. Further results are compared with other solutions in Table 3, where the values of central displacement are normalized as above. The other results considered in Tab. 3 include series solutions for the Mindlin (Salerno & Goldberg, 1960) (Lee et al., 2002) and for the Reissner (Wang et al., 2001) plate models, finite differences solution for the Reissner model (Craig, 1987) and a finite element solution using a higher order plate model (Kant & Hinton, 1980). Two BEM solutions are also included, both for the Reissner's model (Katsikadelis & Yotis, 1993) (Long et al., 1988).

	h/a				
	0.05	0.10	0.15	0.20	0.25
Yuan & Miller[yuan88]	0.01417	0.01618	0.01939	0.02371	0.02913
Deshmukh & Archer[des73]	0.01451	0.01643	-	0.02366	-
Present – Linear (6×6 mesh)	0.01453	0.01648	0.01957	0.02378	0.02009
Present – Quadratic (3×3 mesh)	0.01451	0.01645	0.01953	0.02373	0.02904

Table 1: Numerical results for clamped square plates under uniform loading.

	h/a				
	0.05	0.10	0.15	0.20	0.25
Yuan & Miller[yuan88]	0.04650	0.05019	0.05480	0.06018	0.06683
Deshmukh & Archer[des73]	0.04677	0.05009	-	0.05900	-
Present – Linear (6×6 mesh)	0.04535	0.04882	0.05367	0.05954	0.06631
Present – Quadratic (3×3 mesh)	0.04578	0.04942	0.05432	0.06014	0.06684

Table 2: Numerical results for SS-1 supported square plates under uniform loading.

	h/a			
	0.05	0.10	0.15	0.20
Salerno & Goldberg [1960] (Mindlin)	0.04486	0.04632	0.04676	0.05360
Wang et al. [wang01] (Reissner)	0.04488	0.04630	0.04870	0.05220
Lee et al. [lee01] (Mindlin)	0.04488	0.04663	0.04958	0.05355
Kant & Hinton [kant80] (Higher order model)	--	0.04663	--	0.05351
Craig [craig87] (Reissner)	0.04492	0.04683	0.05183	0.05353
Katsikadelis & Yotis [kats93] (Reissner)	0.04487	0.04634	--	0.05221
Long et al.[long88] (Reissner)	0.04458	0.04612	--	0.05220
Present – Linear (6×6 mesh)	0.04493	0.04668	0.04959	0.05366
Present – Quadratic (3×3 mesh)	0.04497	0.04672	0.04961	0.05368

Table 3: Numerical results for SS2-supported square plates under uniform loading.

The results presented in Tables 1-3 were obtained using RBM technique to integrate the singular kernels. Similar results were obtained for very thick plates using the direct method, presented in Fig.15, and normalized against the analytical results of Lee et al. (2002) for the Reissner's plate model. Therefore we used a shear correction factor $\kappa^2 = 5/6$, due to the lack of a modern analytical solution for thick Mindlin plates, and a 6×6 mesh.

Figure 16 shows the shear force along the side of clamped square plate under uniform load, using quadratic elements. It is interesting to note how well the effect is captured in spite of the coarse mesh used.

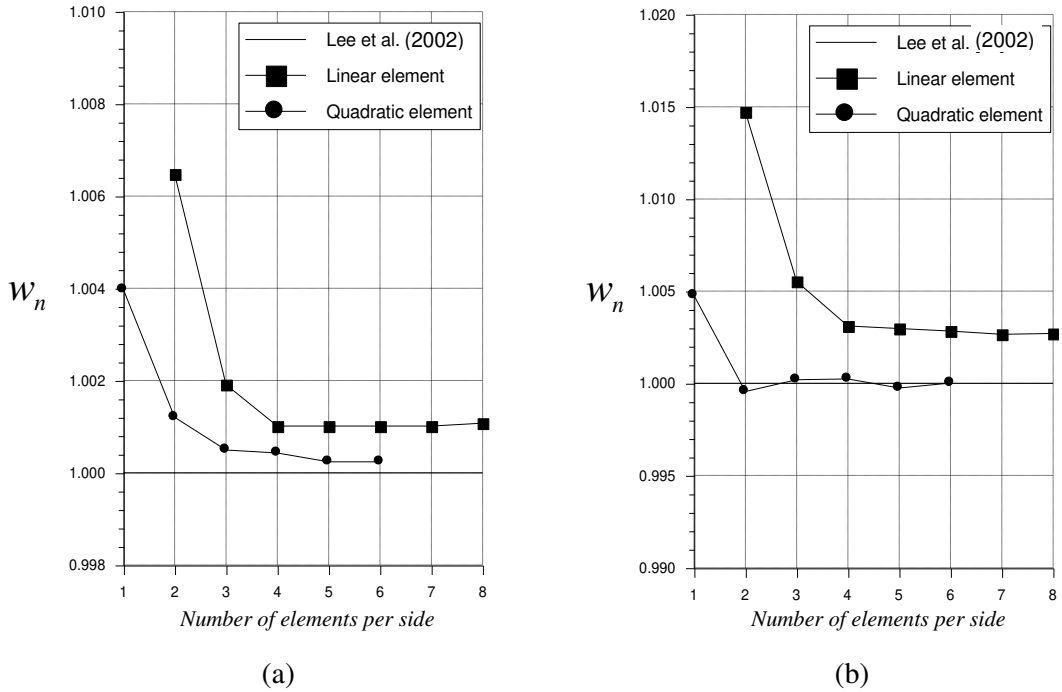


Figure 15. Convergence curves for thick square plates under uniform loading using the direct method to integrate singular kernels. (a) h/a = 0.15. (b) h/a = 0.20.

5.2 Results for Elastic Stability Problems

The performance of the proposed formulation for buckling of supported square plates are presented in Tables 4 (axial compression) and 5 (biaxial compression) and compared with some other references. These results are presented in the form of buckling coefficient $\kappa = N_{cr} a / \pi^2 D$, where N_{cr} is the critical compression load. The mesh chosen is 6×6 constant boundary elements and domain cells. In spite of using only constant elements the results are good.

The typical quality of the eigenvectors obtained is illustrated in Figure 17, which shows a renderized view of the first buckling mode of a square plate under pure shear.

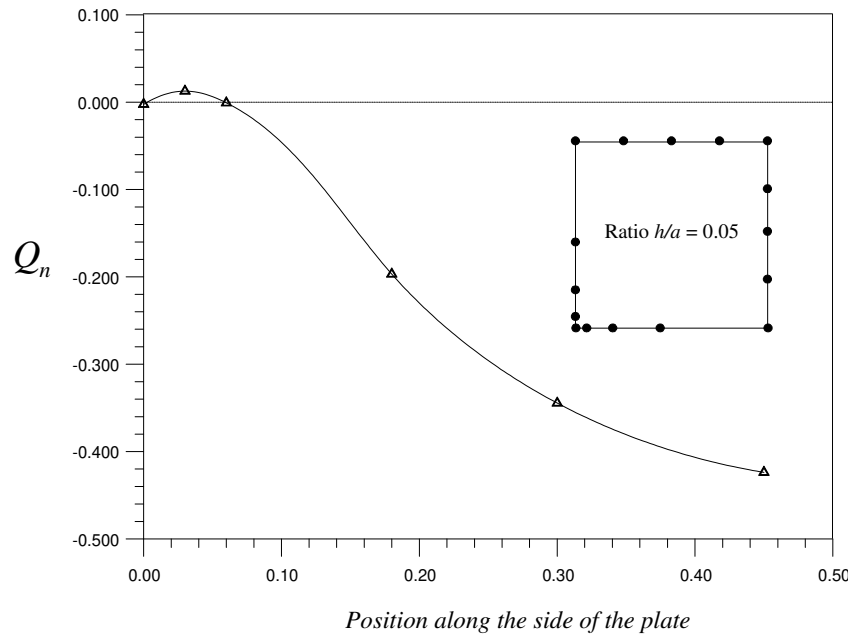


Figure 16. Shear force along the side of a clamped, uniformly loaded square plate.
The continuous line refers do the analytical solution and the symbols to the present results.

	h/a			
	0.001	0.01	0.05	0.1
3D elasticity (Dawe & Roufaeil, 1982)	4.000	-	3.911	3.741
Rayleigh-Ritz (Dawe & Roufaeil, 1982)	4.000	-	3.929	3.731
FE - 9LE (Cheung et al., 1986) - 8×8 mesh	-	4.100	-	3.758
FE - 36LE (Cheung et al., 1986) - 2×2 mesh	-	3.998	-	3.732
BE – Present work (6×6 mesh)	3.9671	3.9646	3.8977	3.7694

Table 4: Numerical results for critical load factor of supported square plates under axial compression. Constant boundary elements and domain cells.

	h/a	
	0.01	0.1
FE - 8LE (Cheung et al., 1986) - 8×8 mesh	2.030	1.880
FE - 17LE (Cheung et al., 1986) - 3×3 mesh	2.000	1.870
BE – Present work (6×6 mesh)	1.9841	1.8818

Table 5: Numerical results for critical load factor of supported square plates under biaxial compression. Constant boundary elements and domain cells.

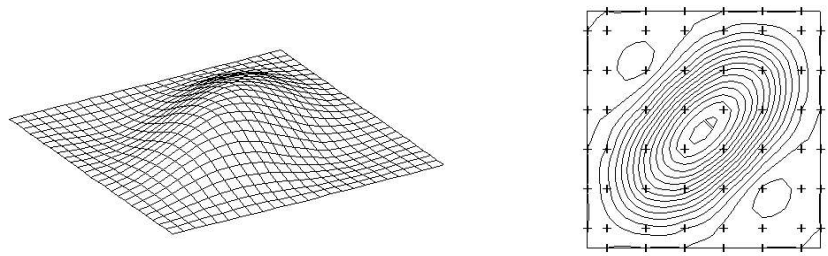


Figure 17. First buckling mode of a clamped square plate under pure shear. Result generated using 6×6 constant domain cells inside the domain, and 26 constant boundary elements.

5.3 Results for Non-linear Bending Problems

In the results presented in this section, the load and the maximum displacement are normalized with the plate thickness: $R = qa^4 / Eh^4$ and $r = u_{\max} / h$, where a is the lateral dimension of the plate. Most results refer to linear boundary elements (BE1), whereas both constant (DC0) and linear (DC1) domain cells were used to discretize the domain.

Table 6 compares the present results with some other numerical solutions. Worth to note is the results of Xiao-Yan et al. (1990), a rare BEM large displacement solution for thick plates. The results of the proposed formulation are presented bor both types of supports for completeness sake. Table 7 analyzes the same case with two opposite sides supported and the two others clamped. A load × displacement curve obtained for supported square plates is shown in fig.18.

	R			
	0.9158	4.579	6.868	9.158
Rayleigh-Ritz (Azizian & Dawe, 1985)	0.04053	0.1929	0.2750	0.3467
FSM (Azizian & Dawe, 1985)	0.04205	0.1950	0.2776	0.3494
BEM (Xiao-Yan et al., 1990)	0.04090	0.1942	0.2767	0.3489
Present work (SS2) BE1/DC0	0.04200	0.1969	0.2775	0.3464
Present work (SS2) BE1/DC1	0.04199	0.1958	0.2753	0.3425
Present work (SS1) BE1/DC0	0.04428	0.2056	0.2878	0.3573
Present work (SS1) BE1/DC1	0.04426	0.2041	0.2849	0.3534

Table 6: Numerical results for non-linear bending of SS1 square plates under uniform load.

	R			
	0.9158	4.579	6.868	9.158
Rayleigh-Ritz (Azizian & Dawe, 1985)	0.01915	0.09513	0.1416	0.1867
FSM (Xiao-Yan et al., 1990)	0.01991	0.09883	0.1469	0.1936
BEM [xiao90]	0.01991	0.09840	0.1455	0.1904
Present work BE1/DC0	0.02020	0.1002	0.1488	0.1957
Present work BE1/DC1	0.02020	0.1001	0.1485	0.1952

Table 7: Numerical results for non-linear bending of SS2 square plates under uniform load.

Finally, in order to show results for a subregion case, the plate with step variation in thickness depicted in Fig.18 was analyzed with 16 quadratic elements and one single domain cell for each subregion. The results are compared against a quadratic FE solution (mesh 20×20 elements) obtained with a commercial software for the rotations in Fig.19. The agreement is good even considering the use of constant cells.

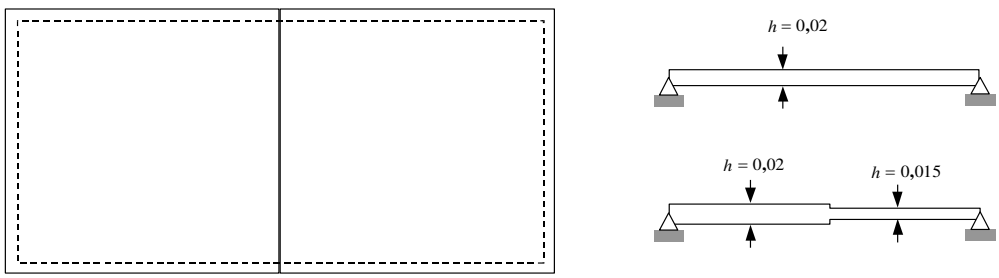


Figure 18. Two subregion non-linear plate bending example under uniform load.

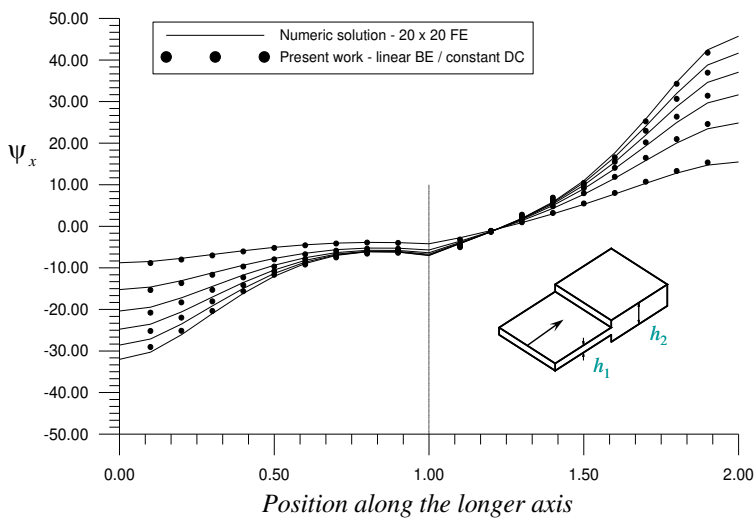


Figure 19. Non-linear results for non-null rotation along the central line for the case illustrated in Fig.18.

6 CONCLUSIONS

The present work presented a compilation of the relevant integral equations for linear and geometrically non-linear bending, as well as elastic stability of moderately thick plates. The hypersingular derivative integral equations for the displacement field were presented, including the corresponding convective terms. The equations were solved using the standard BEM procedure, and different integration approaches were discussed and tested. Object oriented implementation issues are commented aiming a simple, extensible, and reusable research platform architecture. Results were obtained for linear and non-linear elastic bending of selected cases of thick plates under transverse loading. From the results presented it is clear that the application of the BEM for linear bending problems leads to exceptionally good results, even for very coarse meshes. The results shown herein cast an interesting set of benchmarks for comparison with similar methods.

This works summarizes more than 15 years of development and application of the boundary element method for the analysis of thick plates. A work which started shy in early 1990s and resulted in a solid research line for several years. This work later derived to other branches of computational mechanics like structural optimization and Green Functions, but most of this achievements would not be reached without the vision and bold scientific talent of Prof. Clóvis S. de Barcellos.

References

Arnold, D.N. & Falk, R.S., (1989). Edge Effects in the Reissner-Mindlin Plate Theory, in: Noor AK, Belytschko T, Simo JC, editors, Analytical and Computational Models of Shells - The Winter Annual Meeting of the ASME, San Francisco, California, December 10--15, pp.71--89.

Azizian, Z. G. & Dawe, D. J., (1985). Geometrically nonlinear analysis of rectangular mindlin plates using the finite strip method, Computers & Structures 21, pp.423--436.

Barbieri, R. & Barcellos, C.S., (1992). A Modified Local Green's Function Technique for the Mindlin's Plate Problem, In: Brebbia, C.A. & Gipson, G.S. (edss), Boundary Elements XIII - Proc. 13th Int. Conf., Tulsa, Comp. Mech. Publ./Elsevier Appl. Science.

de Barcellos, C.S. & Westphal Jr., T., (1992). Reissner/Mindlin's Plate Models and the Boundary Element Method, in: Brebbia CA, Ingber MS, editors, Proc. 7th Conf. on Boundary Element Technology, pp.589--604, Computational Mechanics Publ.

de Barcellos, C.S., Monken e Silva, L.H., (1989). A Boundary Element Formulation for the Mindlin's Plate Model, in: Brebbia CA, Venturini WC, editors, Proc. of the III Int. Conf. On Boundary Element Technology, pp.123--130, Computational Mechanics Publ.

Batoz, J.L., Bathe, K.J. and Ho, L.W., (1980). A Study of Three-Node Triangular Plate Bending Elements, Int. J. Numer. Methods Eng. 15, pp.1771--1812.

Bui, H.D., (1978). Some Remarks about the Formulation of Three-Dimensional Thermoelastoplastic Problems by Integral Equations, Int. J. Solids & Structures 14, pp.935--939.

Chaves, E.W.V., Fernandes, G.R. & Venturini, W.S. (1999), Plate bending boundary element formulation considering variable thickness', Engineering Analysis with Boundary Elements 25, 405—418.

Cheung, Y.K., Chan, A.H.C. & Tham, L.G., (1986). A Study on the Linear Elastic Stability of Mindlin Plates, Int. J. Numer. Methods Eng. 22, pp.117--132.

Costa Jr., J.A. & Brebbia, C.A. (1985), The boundary element method applied to plates on elastic foundations, Engineering Analysis 2, 174—183.

Craig, R.J., (1987). Finite Difference Solutions of Reissner's Plate Equations, ASCE Journal of Engineering Mechanics, pp.113:31--48.

Dawe, D.J. & Roufaeil, O.L., (1982). Buckling of Rectangular Mindlin Plates, Computers & Structures 15, pp.461--471.

Deshmukh, R.S. & Archer, R.R., (1973). Numerical Solution of Moderately Thick Plates, ASCE Journal of Engineering Mechanics Division 100, pp.903--917.

Guiggiani, M., (1998). Formulation and Numerical Treatment of Boundary Integral Equations with Hypersingular Kernels, in: Sladek V, Sladek J. Singular Integrals in Boundary Element Methods, Chapter 3, pp.85--124, Comp. Mechanics Publ.

Guiggiani, M., Krishnasamy, G., Rudolphi, T.J., Rizzo, F.J. (1992). A General Algorithm for the Numerical Solution of Hypersingular Boundary Integral Equations, J. Applied Mechanics, pp.59:604--614.

Hinton, E. & Huang, H.C., (1986). A Family of Quadrilateral Mindlin Plate Elements with Substitute Shear Strain Fields, Computers & Structures 23, pp.409--431.

Kamiya, N. & Sawaki, Y., (1985). An Efficient BEM for Some Inhomogeneous and Nonlinear Problems, in: Brebbia, C.A. e Maier, G. (editors), Proc. 7th. Int. Sem. BEM Eng., Villa Olmo, Italy, pp.13-59--13-68.

Kant, T. & Hinton, E., (1980). Numerical Analysis of Rectangular Mindlin Plates by the Segmentation Method, tech. report C/R/365/80, Civil Engineering Department, University of Wales.

Karam, V.J. & Telles, J.C.F., (1988). On Boundary Elements for Reissner's Plate Theory, Engineering Analysis 5, pp.21--27.

Katsikadelis, J.T. & Yotis, A.J., (1993). A New Boundary Element Solution of Thick Plates Modelled by Reissner's Theory, Engineering Analysis with Boundary Elements, pp.12:65--74.

Kutt, H.R., (1975). The Numerical Evaluation of Principal Value Integrals by Finite-Part Integration, Numer. Math. 24, pp.205--210.

Lee, K.H., Lim, G.T., Wang, C.M. (2002). Thick Lévy Plates Re-Visited, International Journal of Solids and Structures, pp.39:127--144.

Long, S.Y., Brebbia, C.A., Telles, J.C.F., (1988). Boundary Element Bending Analysis of Moderately Thick Plates, Engineering Analysis, pp.5:64--74.

Marczak, R. J., (1998). A boundary element formulation for linear and nonlinear bending of plates, in S. Idhelson, ed., Computational Mechanics - New Trends and Application, Int. Assoc. Comp. Mech.

Marczak, R.J., (1995a). h & p Convergence Behavior of Boundary Element Analysis of Mindlin-Reissner Plates; Proc. of the XIII Brazilian Cong. Mech. Engineering.

Marczak, R.J., (1995b). Buckling Analysis of Variable Thickness Plates Using a BEM Formulation With Subregions; Proc. 13th. Conf. Structural Mech. Reactor Technology, Porto Alegre - Brazil, Vol. II, pp.759--764.

Marczak, R.J., (1995c). Linear Elastic Stability Analysis of Mindlin-Reissner Plates by the Boundary Element Method (in Portuguese); Revista Internacional de Metodos Numericos Para Calculo Y Diseño En Ingeniaria; Universidade Politecnica de Catalunya, Vol. 11, No.4, pp.625--635.

Marczak, R.J., (1996). Analytical Integration of Membrane-Bending Coupling Terms in Non-Linear Boundary Element Analysis of Mindlin-Reissner Plates; RBCM - J. of the Braz. Soc. Mechanical Sciences, Vol. XVIII, No.3, pp.218--226.

Marczak, R.J., (2004). On The Free Terms of Boundary Integral Equations for Thick Plates Undergoing Large Displacements. Latin American Journal of Solids and Structures, v. 1, pp.343--361.

Marczak, R.J. (2004b). An Object-Oriented Programming Framework for Boundary Integral Equation Methods. Computers & Structures, v. 82, n.15-16, pp.1237-1257.

Marczak, R.J. (2006). Object-oriented numerical integration - A template scheme for FEM and BEM applications. Advances in Engineering Software, Inglaterra, v. 37, n.3, pp.172-183.

Marczak, R.J., Creus, G.J., (2002). Direct evaluation of singular integrals in boundary element analysis of thick plates, Engineering Analysis with Boundary Elements, v. 26, pp.653-665.

McKenna, F.T. (1997), Object-Oriented Finite Element Programming: Frameworks for Analysis, Algorithms and Parallel Computing, PhD thesis, University of California at Berkeley.

Mindlin, R.D., (1951). Influence of Rotatory Inertia and Shear on Flexural Motions of Isotropic, Elastic Plates, Journal of Applied Mechanics 18, pp.31--38.

Pica, A. & Wood, R.D. (1980), Postbuckling behaviour of plates and shells using Mindlin shallow shell formulation, Computers & Structures 12, 759—768.

di Pisa, C. (2005). Boundary Element Analysis of Multi-layered Panels and Structures, Ph.D. Thesis, Queen Mary, University of London.

Pugh, E.D.L., Hinton, E. & Zienkiewicz, O.C., (1978). A Study of Quadrilateral Plate Bending Elements With Reduced Integration, Int. J. Numer. Methods Eng. 12, pp.1059--1079.

Reissner, E., (1944). On the Theory of Bending of Elastic Plates, Journal of Mathematical Physics 23, pp.184--191.

Reissner, E., (1945). The Effect of Transverse Shear Deformation on the Bending of Elastic Plates, Journal of Applied Mechanics 12, pp.A69--A77.

Rushton, K.R. (1970), Large deflexion of plates with initial curvature, Int. J. Mech. Sci. 12, 1037—1051.

Salerno, V.L. & Goldberg, M.A., (1960). Effect of Shear Deformation on the Bending of Rectangular Plates, Journal of Applied Mechanics, pp.27:54--58.

Supriyono (2006). Boundary Element Method for Shear Deformable Plates with Combined Geometric and Material Nonlinearities, Ph.D. Thesis, Imperial College.

Tanaka, M., Sladek, V., Sladek, J., (1994). Regularization Techniques Applied to Boundary Element Methods, Appl. Mech. Rev., pp.47:457--499.

Telles, J.C.F., (1987). A Sef-Adaptive Co-ordinate Transformation for Efficient Numerical Evaluation of General Boundary Element Integrals, Int. J. Numer. Meth. Engng., pp.24:959--973.

Timoshenko, S.P. & Woinowski-Krieger S., (1970). Theory of Plates and Shells, 2nd ed., McGraw-Hill.

van der Weeën, F., (1982). Application of the Boundary Integral Equation Method to Reissner's Plate Model, Int. J. Numer. Methods Eng. 18, pp.1--10.

Vilmann, O., (1990). The Boundary Element Method Applied in Mindlin Plate Bending Analysis, Ph.D. thesis, Technical University of Denmark, Lyngby.

Wang, C.M., Lim, G.T., Reddy, J.N., Lee, K.H., (2001). Relationships Between Bending Solutions of Reissner and Mindlin Plate Theories, Engineering Structures, pp.23:838--849.

Westphal Jr. T, Andrä H, Schnack E., (2001). Some Fundamental Solutions for the Kirchhoff, Reissner and Mindlin Plates and an Unified BEM Formulation, Engineering Analysis with Boundary Elements, pp.25:129--139.

Westphal Jr., T. & Barcellos, C.S., (1990), Applications of the Boundary Element Method to Reissner's and Mindlin's Plate Models, in: Tanaka, M., Brebbia, C.A. e Honma, T. (editors), Proc. 12th Int. Conf. BEM - BEM 12, Vol. 1, Sapporo, Japan, pp.467--477.

Westphal Jr., T., Schnack, E., de Barcellos, C.S., (1998). The General Fundamental solution of the Sixth-Order Reissner and Mindlin Plate Bending Models Revisited, Comput. Methods Appl. Mech. Eng., pp.166:363--378.

Xiao-Yan, L., Mao-Kwang, H. & Xiuxi, W., (1990), Geometrically Nonlinear Analysis of a Reissner Type Plate by the Boundary Element Method, Computers & Structures 37, pp.911--916.

Yuan, F.G. & Miller, R.E., (1988). A Rectangular Finite Element for Moderately Thick Flat Plates, Computers & Structures 30, pp.1375--1387.