Differential geometry management of higher order 2D boundary elements

L. E. T. Ferreira*
Escola Politécnica -Universidade de São Paulo.
Rua João de S. Dias, 520, a. 123, 04618-003, S. Paulo, Brazil.

Abstract

This paper addresses the use of higher order elements in boundary element analysis concerning accuracy and implementation techniques. It is shown that the generation of values of higher order shape functions for elements of any order, as well as for their derivatives, can be accomplished in a simple way through the use of Lagrange polynomials, avoiding the difficulties regarding the explicit deduction and coding of the equations. To this end, an effective computational scheme is given and the generalization of the intrinsic space is discussed within two-dimensional stress analysis. Finally, numerical experiments with slender components subjected to bending are carried out for studies on convergence. The results obtained showed that the appropriate order of the element to be used may depends directly on the nature of the problem being analyzed.

Keywords: shape function, higher order elements, shape functions derivatives, interpolation, inside integration.

1. Introduction

The use of higher order elements in computational mechanics is not new. As it is well known, due to the curvature associated with them, these elements are more appropriate to model arbitrary geometries as well as the responses of the actual components being analyzed, in two and three dimensions [1, 2].

Perhaps, the most popular high order boundary element is the three-noded (continuous or discontinuous) quadratic element that, for most applications, is easy to handle and, possibly for this reason, have been widely implemented. However, when the parabolic nature of the quadratic element is not precise enough to describe in a natural way the geometrical quantities concerned, or when they do not adequately model the nature of the response itself, other procedures are required.

* Currently with Planger Engenharia, Projetos e Ger. Int. Ldta – Tel: +55-11-50424319 Fax: +55-11-43322630, e-mail: leferrei@uol.com.br

Methodologies that decrease the integration space by increasing the number of elements in a region of interest have been developed. Alternatively, an existing mesh can be preserved and the order of the polynomials used to define the elements, increased. In both cases, the responses will be greatly improved.

Nevertheless, within special situation it becomes more convenient to increase the order of the element used, just elevating the number of its functional nodes, for example, in a situation where an accurate geometrical description is necessary. Note that it can be accomplished without any increase in the number of unknowns in the problem, i.e., just by regrouping nodes of an existing mesh. The penalty of this simple method is that the generation of the interpolation functions for the higher order elements will be necessary and the effective use of an existing computer code will be limited to the shape function that have been implemented.

This inconvenient of explicitly generating the interpolation functions becomes more evident when the differential geometry of the boundary element is taken into account, what usually occurs while transforming the variables from the boundary curve, $\boldsymbol{\Gamma}$, to the intrinsic space, ξ. In this case, the Jacobian of the transformation, $\mathbf{J}$, and hence, the spatial derivatives of the shape function are needed for intermediate computations such those related to the directions of local coordinate systems (unit normal and tangential vectors) required for the calculations of field variables. In this regard, another set of equations concerning the derivatives of the shape functions will have to be deduced and coded.

In what follows, it will be shown that the equations for the shape functions and their derivatives don't have to be explicit derived and coded, so that any simple program can be adapted to handle efficiently elements of any order. Moreover, if efficiently implemented, the routines that generate these values can be manipulated in an efficient way, particularly to deal with the integration process of the kernel tensor functions.

Finally, some studies of slender components in bending are carried out. Within these numerical experiments, difficulties are gradually added so that the performance of elements of different orders is analyzed, relatively to convergence. In this paper, only isoparametric boundary elements are considered, and for conciseness, only two-dimensional elasticity problems are addressed.

2. Interpolation functions for higher order elements

The mathematical derivation of the shape functions associated with higher order elements, or merely **HO** elements, can be accomplished in different ways. A straightforward one is to use the Lagrange polynomials that, for one-dimensional elements can be generated for an element with **k** nodes, in a given position ξ, through the use of Eq. 1 [2, 3]:

$$N_i(\xi) = \prod_{j=1, j\neq i}^{k} \frac{\xi - \xi_j}{\xi_i - \xi_j} \tag{1}$$

In the above equation, $\mathbf{N_i(\xi)}$ is the value of the shape function associated with node **i** of the element, ξ_i is the value of the intrinsic or normalized coordinate ξ at this node and ξ_j are the values of the intrinsic coordinates at other nodes. The symbol $\prod$ stands for the product operator.

It seems to be important to observe that only single values of each shape function, calculated at a desired locations ξ, are computationally needed at a time, so that Eq. (1) can be directly implemented for this purpose. The computational implementation of Eq. (1) requires two extra arrays. The first one to store the nodal positions, ξ_i, along the normalized element, while the second will be needed to recover the values of the shape functions computed at a certain position of interest, ξ**.**

As the order of the elements used in the same mesh are not necessarily uniform (in terms of number of nodes), these arrays may be dynamically or temporarily allocated. Appendix A presents the code of a subroutine named SHAPE_FUNCTION, written in FORTRAN 90 language, to deal with Eq. (1). With this code, two variables **a** and **b**, are used to define the lower and upper limits of the element in the intrinsic space, usually **-1** and **1**, respectively. Variable **k** stands for the number of nodes on the element.

A simple check of this implementation can be readily made. For a given position ξ of interest, over an intrinsic element with **k** nodes, it will follow that [1]:

$$\sum_{j=1}^{k} N_j(\xi) = 1 \tag{2}$$

3. Derivatives of the shape functions for HO elements

As mentioned, the difficulties to make use of higher order elements become more evident when the differential geometry of the element is taken into account. In this case, another set of equations related to the derivatives of the shape functions for the **HO** are necessary.

To establish also a higher level of versatility on the use of **HO** elements, a general expression for the derivatives of the shape functions will be needed in advance. This expression can be obtained by differentiating Eq. 1. The final form of the resulting equation may depend on mathematical and programming style and is unimportant from the technical point-of-view. Deduced by the author, a not-so-simple but very versatile equation resulting from the differentiation of Eq. 1 is presented for the determination of the shape functions derivatives (Eq.3). For an element with **k** nodes, these derivatives will be given by:

$$\frac{\partial N_i(\xi)}{\partial \xi} = \frac{1}{\prod\limits_{\substack{j=1 \\ (j\neq i)}}^{k} (\xi_i - \xi_j)} \left(\sum_{n=i}^{k-1} \left[\prod_{\substack{j=1 \\ (j\neq i,\, j\neq n+1)}}^{k} (\xi - \xi_j) \right] + \sum_{n=1}^{i-1} \left[\prod_{\substack{j=i \\ (j\neq i;\, j\neq n)}}^{k} (\xi - \xi_j) \right] \right) \tag{3}$$

In Eq. (3), $\partial \mathbf{N_i}(\xi)/\partial\xi$ is the value of the shape function derivative associated with node **i** of the element, ξ_i is the value of the intrinsic or normalized coordinate ξ at this node and ξ_j are the values of the intrinsic coordinates at other nodes. The symbols $\prod$ and Σ stand for the product and summation operators, respectively.

In order to use Eq.(3) consistently, an array to store the nodal positions, ξ_i, along the normalized element is required in addition to another one, to store the shape functions derivative computed in a desired position ξ. A complete subroutine to deal with Eq. (3), named SHAPE_FUNCTIONS_DERIV, also written in FORTRAN 90 language, is presented in Appendix B of this paper. To check the values computed using Eq. (3) with subroutine SHAPE_FUNCTIONS_DERIV, for a given position ξ of interest on the intrinsic element, Eq. (4) can be used:

$$\sum_{j=1}^{k} \frac{\partial N_j(\xi)}{\partial \xi} = 0 \tag{4}$$

Also, Eq. (5) can be applied to check the global computations performed at intrinsic nodal positions, ξ_J , as follows:

$$\sum_{i=1}^{k} \sum_{j=1}^{k} \frac{dN_i(\xi_j)}{d\xi} = 0 \tag{5}$$

If the number of nodes **k** is even, what implies the inexistence of a central node on the element, for equally spaced nodes (so that symmetry is ensured) Eq. (6) can be used as a third check:

$$\sum_{j=1}^{k} \frac{dN_i(\xi_j)}{d\xi} = -\sum_{j=1}^{k} \frac{dN_{(k-i+j)}(\xi_j)}{d\xi} \tag{6}$$

On the other hand, if **k** is odd Eq.(6) still applies by both sides of the central node. In this case, Eq. (7) can be used to check the computations at this node:

$$\sum_{j=1}^{k} \frac{\partial N_{\frac{k+1}{2}}(\xi_j)}{\partial \xi} = 0 \tag{7}$$

Note that Eq. (5) applies to elements with nodes equally spaced or not. In the latter case, however, the subroutine presented in this paper will have to be adapted. The computation of other entities of interests like Jacobians and unit vectors can be performed as usual, using the results calculated with Eq. (1) and (3).

4. Generalization of the intrinsic space

As commented, Eq. (1) and (3) can be employed to evaluate shape functions and their derivatives for elements with **k** nodes, within any intrinsic interval (say, from **a** to **b**), thus substituting in a natural and efficient way the traditional sets of equations usually deduced (and coded) to deal with the differential geometry of the elements. This conjunction can be very useful in many situations, especially in those where the boundary element mesh is formed by elements of different order.

It will be particularly true if "macro" elements are present in the mesh. In this case, special treatment to accurately integrate the kernel functions will be required, what can be accomplished if the element subdivision technique is considered. On the other hand, numerical experiments show that this technique becomes imperative if internal point's responses are to be computed using macro elements. To concisely express how this can be achieved, in what follows only two-dimensional elasticity problems, within the framework of the displacement-BIE formulation, will be addressed.

4.1 Outside integration

In two-dimensional elasticity, the solutions for the displacements and tractions in **x** and **y** directions due to unit loads applied in these directions are the kernels to be integrated along the boundary element analysis. The displacements kernels are given by [4] :

$$U_{kk} = \frac{1}{8\pi G(1-\nu)}\left[(3-4\nu)\ln\left(\frac{1}{r}\right)+\left(\frac{r_k}{r}\right)^2\right] \tag{5}$$

$$U_{kL} = \frac{1}{8\pi G(1-\nu)}\left(\frac{r_k}{r}\right)\left(\frac{r_L}{r}\right) \tag{6}$$

In the case where the source point doesn't coincide with the field point, the integration of the kernel tensor functions may be evaluated using Gauss-Legendre quadrature, for each element, on the standard intrinsic interval running from **-1** to **1**, in order to obtain individual contributions of the element, $\mathbf{U}^e$. This contribution may be written in a general form, for a given load point **P**, as [5]:

$$U^e_{KL\,mj} \approx \sum_{n=1}^{ng} N_m(x_n)\,U_{KL}(P_j, x_n)\,J(x_n)\omega_n \tag{7}$$

In the above equation, $\mathbf{x_n}$ and $\boldsymbol{\omega_n}$ are the Gauss Points coordinates and its corresponding weighting, $\mathbf{N_m\,(x_n)}$ is the shape function associated with node **m,** $\mathbf{J(x_n)}$, the Jacobian of transformation, $\mathbf{P_j}$ is the unit load applied at **j** direction and **ng**, the number of Gauss points employed in the integration scheme. Note that within the progression of the computations, the directions of the unit load and the displacements, as well as node number and collocation point order, will have to be observed for each element.

As a very central part of the boundary element method, the integration process of the kernel tensor functions, also when using **HO** elements will have to ensure a certain consistency with regard to the length of each **HO** element and the number of Gauss points employed in the quadrature processes. For this purpose, equivalence between the integration of these functions along a single quadratic element and along all the quadratic elements that could be formed using the nodes of the **HO** element may be evoked as a first approach.

In this paper, the element subdivision is proposed to be performed in such a way that each interval of integration will be the one confined between two consecutive

nodes. Therefore, for a **HO** element with **k** nodes subdivided in to (**k-1)** segments, equally spaced or not, for a single load point **P**, applied at **j** direction, Eq. (7) can be reformulated as:

$$U_{kl}^{e} \approx \sum_{i=1}^{k-1}\left(\sum_{n=1}^{ng} \left(\sum_{m=1}^{k} \left(\sum_{j=1}^{d} N_m\left(x_{T(n)}\right) U_{kl}\left(P_j, x_{T(n)}\right) J\left(x_{T(n)}\right) \omega_n \; J_T \right) \right) \right) \quad (8.1)$$

with:

$$J_T = \frac{\xi_{i+1} - \xi_i}{2} \qquad \text{and} \qquad x_{T(n)} = \frac{\left(\xi_{i+1} + \xi_i\right) + \left(\xi_{i+1} - \xi_i\right) x_n}{2} \qquad (8.2, 3)$$

When element subdivision is applied to integrate over elements of higher order, Equations (8.3) will provide the necessary coordinate transformation so that these points can be sampled within the sub interval that are being integrated. The above transformation usually decreases the numerical value of the coordinates, with more or less significance, what will depend on the number of sub interval employed in the process, or, within the present approach, on the number of nodes existing in the element, since the standard interval that defines the element, is usually kept in the range -1 to 1.

As a consequence and considering also the high number of standard operation involved in the quadrature process, the numerical results may suffer some loss in accuracy, what will be particularly true if single precision arithmetic is employed. To perform the quadrature given by Eq. (8.1), subroutines dealing with Eq. (1) and (3) can be directly employed to perform the transformations given by Eq. (8.3) in a more accurate way.

This can be accomplished by performing a single operation on the limits **a** and **b**, defining the intrinsic space of the element, so that the intrinsic space is increased. For example, if a 6-noded element is to be subdivided as proposed, 5 segments, confined between adjacent nodes, will result. To integrate the central sub element, the third node is set to -1 and the fourth to 1, so that the initial node will be -3 and the final node 3. This makes the intrinsic space equals to 6, that is, 3 times greater than the standard interval, so that Eq. (8.3) can be applied with better approximation. To generalize this approach, the extremes **a** and **b** of an element with **k** nodes, to be subdivided into **k-1** segments, can be set as follows:

$$a_i = -SE_i \qquad \text{and} \qquad b_i = k + a_i \qquad (9.1, 2)$$

where $\mathbf{SE_i}$ is the number of the sub element being integrated.

4.2 Inside integration

The displacements kernels $\mathbf{U_{kk}}$, given by Eq. (5) are of particular interest when self-influence (inside integration) are considered. Due to the logarithmic terms present in these functions, two different intrinsic spaces will be involved, the first, ξ, related to the boundary element definition (frequently running from -1 to 1), and the second, η,

coupled to the integration quadrature space (usually running from 0 to 1) so that these functions are generally separated into weakly-singular and singular parts to be numerically integrated using standard Gauss-Legendre and Logarithmic quadratures. However, to accomplish this, several spatial transformations are necessary [5, 6, 7].

Conversely, the linear transformations just referred can be avoided by adopting a non-linear approach (quadratic or cubic) as the one proposed by Telles [8] that makes a direct treatment of the different spaces unnecessary. In this case, standard Gauss quadrature can be adopted. However, in order to ensure accuracy while integrating the kernel tensor functions when **HO** elements are involved, some additional care must be taken to ensure accuracy of results. In this paper, the Lin-Log quadrature [9] will be focused on, within element subdivision process, by manipulating the intrinsic space where the element is defined. Also in this case, the subroutines to deal with Eq. (1) and (3) will show to be useful.

Based on the moments **ln**(x),1; x **ln**(x), x; x^2.**ln**(x), x^2, etc, the Lin-Log quadrature can be used to exactly integrate, using a one-point rule, the logarithmic function appearing in Eq. (5). To illustrate how this can be accomplished, consider the integral:

$$I = \int_0^1 \ln\left(\frac{1}{\eta}\right) d\eta = -\int_0^1 \ln(\eta) d\eta = 1 \tag{10}$$

For a one-point rule, the functions f(x) = (1; ln(x)) will be used to obtain the coordinate and the weight related to a one-point quadrature rule:

$$f(x) = 1 \Rightarrow w_1 = \int_0^1 dx = 1 \tag{11}$$

and:

$$f(x) = \ln(x) \Rightarrow w_1.\ln(x_1) = \int_0^1 \ln(x) dx = -1 \Rightarrow x_1 = e^{-1} = 0.3678749441 \tag{12}$$

so that:

$$I = \int_0^1 \ln\left(\frac{1}{\eta}\right) d\eta = -\int_0^1 \ln(\eta) d\eta = -\sum_{i=1}^{1} w_i.\ln(x_i) = -1.\ln\left(e^{-1}\right) = 1 \tag{13}$$

To verify the performance of this technique relatively to Telles's cubic transformation (that uses standard Gauss-Legendre quadrature), another function is integrated, now on the interval running from -1 to 1. Also, the Gauss-Legendre quadrature, without any transformation, is checked. To illustrate this, ponder the following example:

$$I = \int_{-1}^1 \ln\left(\frac{1}{\eta^2}\right) d\eta = 4 \tag{14}$$

Considering the singularity that occurs at η=0 (so that the sample points are clustered toward the middle of the interval, within Telles's transformation process), the results obtained for this integration, using quadratures of different order, are those presented in Tab. 1.

Table 1 – Numerical results: Gauss-Legendre, Telles's cubic transformation and Lin-Log quadratures.

# Sample Points	Gauss-Leg. [1]	Telles's Transform. [2]	Lin-Log [3]	% Error [1]	% Error [2]	% Error [3]
2	2.197224577	6.591673732	4.000000000	-45.069	64.792	0.000
4	3.022328051	4.316009015	4.000000000	-24.442	7.900	0.000
6	3.326751404	4.098221146	4.000000000	-16.831	2.456	0.000
8	3.486194603	4.042885716	4.000000000	-12.845	1.072	0.000
10	3.584464195	4.022493422	4.000000000	-10.388	0.562	0.000
12	3.651138019	4.013247997	-	-8.722	0.331	-
16	3.735855641	4.005723289	-	-6.604	0.143	-
20	3.787451230	4.002975243	-	-5.314	0.074	-
24	3.822178527	4.001740055	-	-4.446	0.044	-
28	3.847149253	4.001104299	-	-3.821	0.028	-
32	3.865969292	4.000744190	-	-3.351	0.019	-
36	3.880662278	4.000525118	-	-2.983	0.013	-

In the above example, the two sample points used with Lin-Log quadrature are relative to a one-point rule (Eq. (11) and (12)). Due to the singular point existing in the middle of the interval, the numerical integration in this case was performed using a single sample point, over each side of the singularity. It seems to be important to observe that not only accuracy is obtained with the Lin-Log quadrature, as it was shown to be possible to accomplish using a minimal number of sample points. Also, the number of standard operation and computational effort involved in the quadrature process is attractively smaller, if compared with Telles's transformation.

To apply the Lin-Log quadrature and element subdivision simultaneously, a simple case where the source point of the fundamental solution is placed on node 3 of a hypothetical four-noded **HO** will be analyzed. Also in this case, the subintervals of integration will be those confined between two consecutive nodes. To place the integration region directly on each subinterval of the subdivided element, the extremes of the element will be conveniently defined so that the intrinsic space and the integration space will be coincident, running from 0 to 1, within a two-step analysis where the element is integrated by both sides of the node, as depicted in Fig. 1.

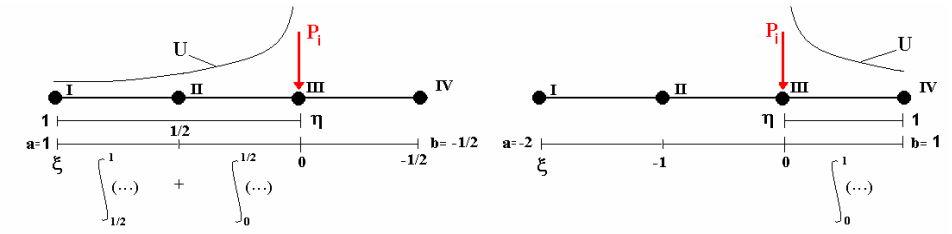

Figure 1 – Element subdivision - Inside logarithmic integration of a four-noded HO element.

To employ this concept to elements of any order, a simple rule to define the extremes **a** and **b** of the intrinsic space can be written, to integrate the left and the right parts of the element, as follows:

By the left:

$$a = 1 \qquad \text{and} \qquad b = \frac{-(k-n)}{n-1} \tag{15.1, 2}$$

By the right:

$$a = \frac{-(n-1)}{k-n} \qquad \text{and} \qquad b = 1 \tag{16.1, 2}$$

In the above equations, **k** is the number of nodes of the element and **n** is the node where the source point is considered. Note that, while implementing the element subdivision technique, within each step of the integration process the respective Jacobian will have to be considered, in order to take into account the new limits, say **c** and **d**, of the subinterval. Conversely, the integration process can be performed in terms of element length, **L** (computed with a minimal quadrature order). In this case, the variables **a** and **b** defining the element limits will be:

By the left: (for n>1)

$$a = L.\left(\frac{n-1}{k-1}\right) \qquad \text{and} \qquad b = -L.\left(\frac{k-n}{k-1}\right) \tag{17.1, 2}$$

By the right: (for n<k)

$$a = -L.\left(\frac{n-1}{k-1}\right) \qquad \text{and} \qquad b = L.\left(\frac{k-n}{k-1}\right) \tag{18.1, 2}$$

To logarithmic integrate a subinterval which limits are **c** and **d**, Eq. (19) and (20) can be used:

$$\int_c^d \ln\left(\frac{1}{r}\right)_\eta \left(\frac{\partial \xi}{\partial \eta}\right)_{c,d} J d\eta = \approx -\sum_{i=1}^{np} \left(\ln(\xi_i)_{\eta_i} \omega_i \left(\frac{\partial \xi}{\partial \eta}\right)_{c,d} J \right) \tag{19}$$

with:

$$\xi_i = \eta_i (d-c) + c \qquad \text{and} \qquad \frac{\partial \xi}{\partial \eta} = d - c \tag{20.1, 2}$$

In the above, $\boldsymbol{\eta_i}$ and $\boldsymbol{\omega_i}$ are the coordinates and weights of the Lin-Log quadrature, **np** is the number of points used in integration process and **J** the Jacobian that makes the transformation from the boundary path, **Γ**, to the intrinsic coordinate. The coordinates and weights computed for a five-point rule (sufficient to accurately integrate the kernels appearing in 2D elasticity problems, over HO elements) are presented in Tab. 2.

Table 2 - Coordinates and weights for a five-point rule - Lin-Log quadrature

xi	wi
0.565222820508009	0.210469457918546
0.734303717426522	0.130705540744446
0.284957404462558	0.289702301671314
0.619482264084778	0.350220370120398
0.915758083004698	0.208324841671985

The strategy just outlined has been successfully employed by the author to solve 2D elasticity problems [10]. The referred program was implemented in such a way

that models, discretized with mixed elements of any order, can be elastically analyzed with the proposed techniques. The approach just outlined seems to be attractive, since it can be easily implemented in existing codes, permitting not only a greater level of automation (if subroutines written to deal with Eq. (1) and (3) are used), but also accurate computation of the singular integrands appearing in some of the displacement kernel tensor functions.

5. Secondary responses computation with HO elements

As commented before, accuracy can be accomplished by raising the number of functional nodes defining the elements, passing, for example, from quadratic to cubic, or from cubic to quartic elements, without necessarily increasing the number of unknowns in the problem. In elasticity problems, for example, quadratic elements do not perform efficiently in bending, especially in the case where slender components are being analyzed. In this specific case, numerical experiments have shown that coarse meshes of cubic elements will produce accurate results, since the parabolic nature of the quadratic element is not sufficient to describe the cubic (or higher) variation of the displacements occurring in bending problems.

Nevertheless, some care must be taken when defining the order of the element to be used. In addition to the problems concerning element subdivision and accurate inside integration of higher order elements, an issue that has to be addressed is the accuracy of the derivative responses, as fluxes and stresses. These quantities are generally computed by taking the derivatives of the functional representation of the primary responses, as temperatures and displacements, so that Eq. (3) will be used in the process

This question becomes more evident when Lagrange (or Serendipity) elements are employed, in two- or three-dimensional analysis. Although easy to implement, these elements provide only C^0 continuity at their boundaries, so that continuity of secondary or derivative responses, as stress values for example, are not accomplished. Thus, discontinuity or "jumps" occurring at the element interfaces needs especial treatment. On the other hand, at the central node of the quadratic element, for example, the continuity of the responses is fulfilled completely, since a single value of traction, in each direction, will be obtained at this node. Similar behavior is observed with the strains that are computed by taking the derivatives of the functional representation for nodal displacements, and, consequently, with the stress tensor, usually computed from traction and strain responses [e.g. 4, 5, 6, 7].

Different approaches to circumvent this problem have been proposed, as stress average, smoothing techniques and the use of high order and complex functions representation [e.g. 11, 12, 13]. However, these "jumps" would be effectively avoided only though elements that provides C^1 continuity across element boundaries, as Hermite or Overhauser and Spline elements [3]. As previously discussed, the former depends on the nature of the boundary element analysis approach used, since the tangential derivatives of tractions and displacements fields are needed while using Hermite polynomials. The latter will require specialist treatment to generate the elements, so that

they are not easily implemented in existing codes. Therefore, the use of higher order elements will restrict the later problem of "jumps" and averaging to a very few element boundaries, as many nodes could be positioned inside a single element, because all internal nodes are perfectly C^1 continuous.

6. Numerical experiments and results

In order to compare the performance of **HO** elements, a linear elastic cantilever beam subjected to bending was analyzed under plane stress, within two different problems. The BE models where analyzed using the program ELASCON [10].

First problem. Within the first study, the cantilever beam was subjected to a vertical load that simulates a concentrated force applied at it's free end. The proposed problem was subdivided into two parts. Within the first part, an apparently difficult situation where the ratio between the high **h** and the span **L** of the beam is 0.1 was considered. To evaluate the performance of **HO** elements, 3 internal points were also placed inside the solution domain, in the middle of the span, as depicted in the inset of Figure 2.

In the second part, some difficulty was added to the problem, reducing the ratio **h/L** to 0.01 and employing elements two times longer in the discretization process. Physical and geometrical data for the bending problem, as well as the BE meshes with same number of degrees of freedom are presented in Figure 2.

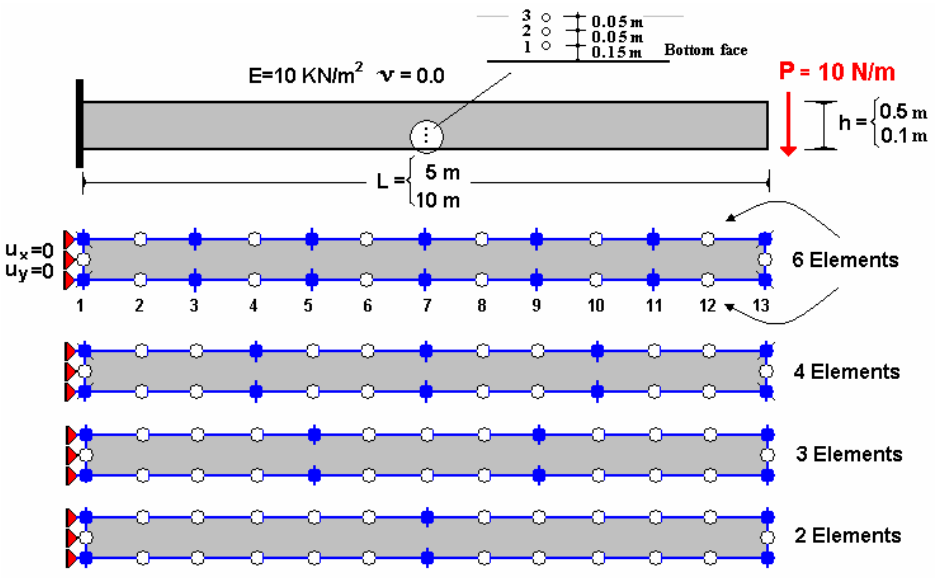

Figure 2 – Slender cantilever beam– geometry, boundary conditions and BE meshes

Results – First analysis

Case 1 - h/L=0.1. The boundary stress results of σ_{xx} are presented in Tab. 3 for nodes number 2, 6, 8 and 12, intermediate nodes in all meshes, so that stress results are not averaged. Tab. 3 also shows the analytical solution obtained using beam theory [14]. Tab. 4 and 5 shows the computed results for the internal points.

Table 3 – Stress σ_{xx} (Pa) at the boundary nodes computed with different HO elements (h/L=0.1).

Node	Position (m)	Theory	3-Noded	4-Noded	5-Noded	7-Noded
2	0.4167	1100.00	-1053.06	-1102.52	-1102.61	-1102.30
6	2.0833	700.00	-670.05	-701.94	-702.13	-702.08
8	2.9167	500.00	-477.63	-501.86	-502.01	-501.88
12	4.5833	100.00	-93.75	-100.99	-101.22	-101.49

Table 4 – Stress σ_{xx} (Pa) at internal points computed with different HO elements.

Point	Position (m)	Theory	3-Noded	4-Noded	5-Noded	7-Noded
1	0.15	-240.00	-229.85	-241.40	-240.88	-238.93
2	0.20	-120.00	-114.94	-120.72	-120.46	-119.62
3	0.25	0.00	0.00	0.00	0.00	0.00

Table 5 – Stress τ_{xy} (Pa) at internal points computed with different HO elements.

Point	Position (m)	Theory	3-Noded	4-Noded	5-Noded	7-Noded
1	0.15	-25.200	-33.40	-29.67	-27.53	-23.74
2	0.20	-28.800	-37.00	-33.31	-31.12	-27.64
3	0.25	-30.000	-38.16	-34.52	-32.31	-28.94

Case 2 – h/L=0.01. In this case, the results of boundary stresses with the expected analytical results are shown in Tab. 6.

Table 6 – Stress σ_{xx} (Pa) at the boundary nodes computed with different HO elements (h/L=0.01).

Node	Position (m)	Theory	3-Noded	4-Noded	5-Noded	7-Noded
2	0.4167	-55000.00	-12216.51	-55134.91	-55149.54	-55156.49
6	2.0833	-35000.00	-7709.35	-35096.15	-35102.61	-35108.40
8	2.9167	-25000.00	-5311.52	-25076.94	-25077.45	-25087.06
12	4.5833	-5000.00	-791.56	-5016.66	-5021.36	-5026.48

Second problem. Within the second study, the cantilever beam was again analyzed, however in a quite different way. The new circumstances were related to the

discretization and loading processes. Within this study, the cantilever beam was subjected to a uniform load applied over its top face. To graphically analyze the result for the displacements conveniently, the Young Modulus, **E**, was also changed to a value ten times greater. Again in this case, the problem was subdivided into two parts.

With the objective to check the individual performance of elements of different order, relatively to the displacements values computed at internal points as well as at boundary nodes, the model was firstly discretized using single 3-, 4- and 5-noded elements, to model the upper and lower faces of the beam. To represent the lateral faces, two quadratic elements where used, one per face, so that in all cases 4 elements have been employed, as depicted in Fig. 3.

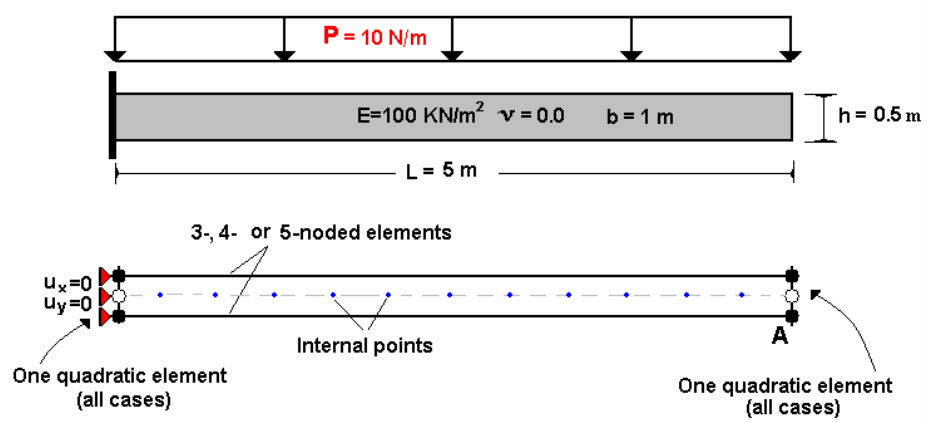

Figure 3 – Slender cantilever beam subjected to distributed load (only boundary nodes shown).

In the second part of this study, the model was discretized using a variable number of the same elements (of different order) employed in first part of the problem. The objective in this case was to check for convergence in displacements results, relatively to theoretical values given by beam theory [14], computed at 11 points placed inside the solution domain, along the structural axis.

Results - Second analysis

Case 1 –Figure 4 shows the results for vertical displacements computed at internal points.

Case 2 – Within this case, a convergence study has been performed for the 3-, 4- and 5-noded elements, refining the meshes in order to reach an error on the vertical displacements of internal points smaller than 1.5%. Individual results are shown in Fig. 5, 6 and 7.

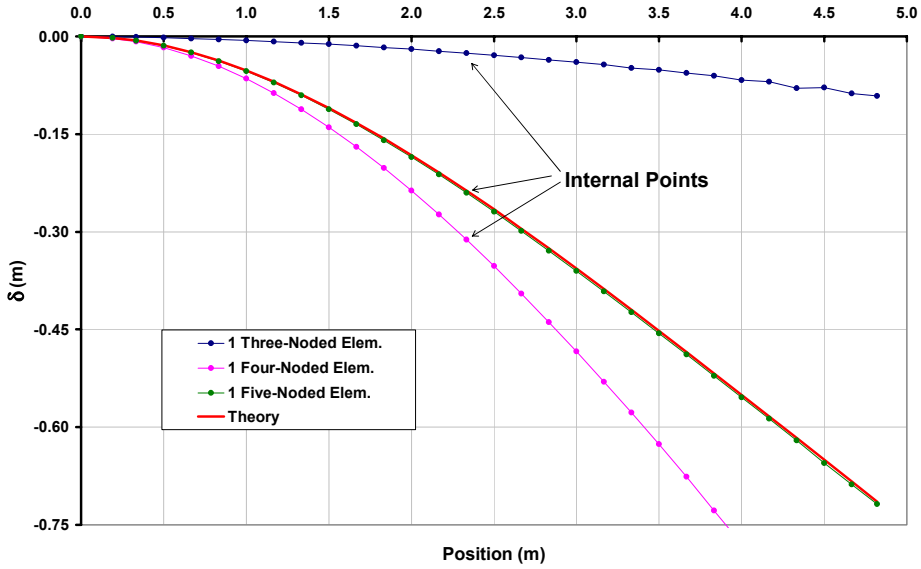

Figure 4 – Plots of vertical displacements – Internal Points.

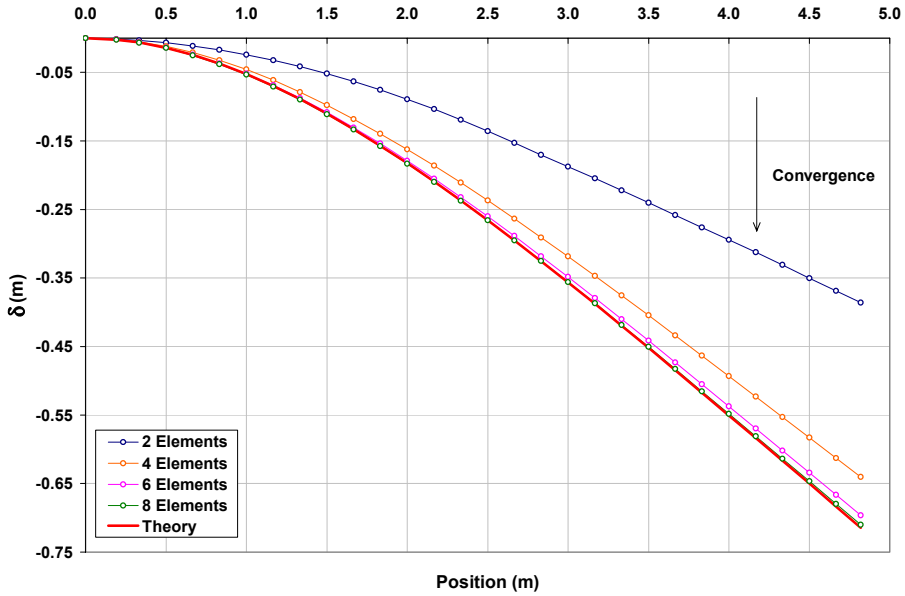

Figure 5 – Vertical displacements of Internal Points – Quadratic elements.

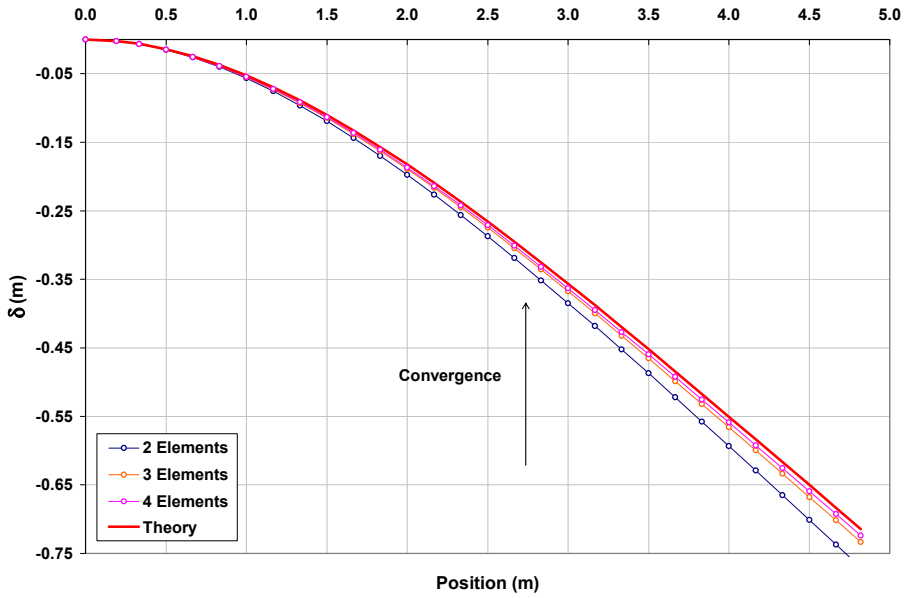

Figure 6 – Vertical displacements of Internal Points – Cubic elements.

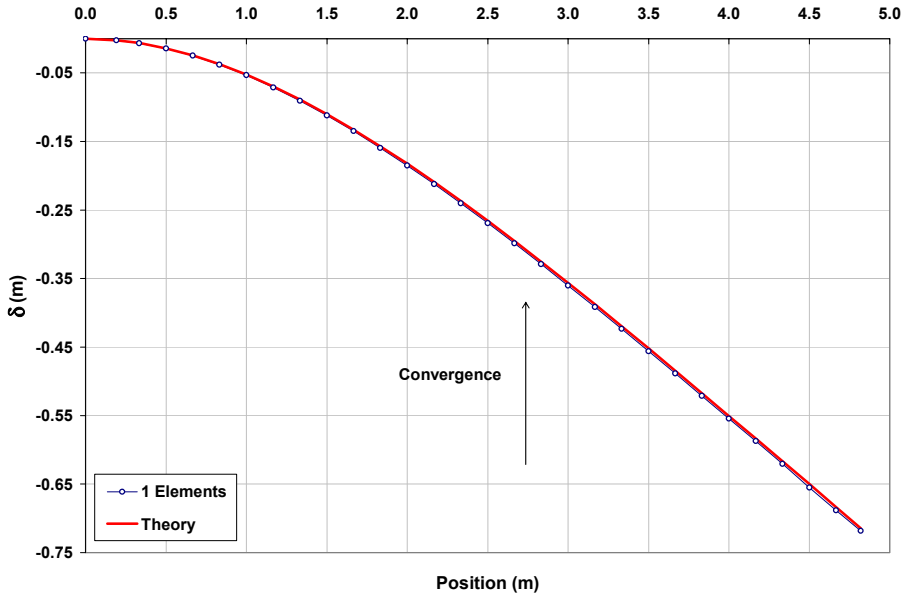

Figure 7 – Vertical displacements of Internal Points – Quartic elements.

Figure 8 depicts the performance of the elements investigated, with regard accuracy and the number of nodes employed in the bottom-face of the model. It's also shown the results obtained with the 7- and 11-noded elements.

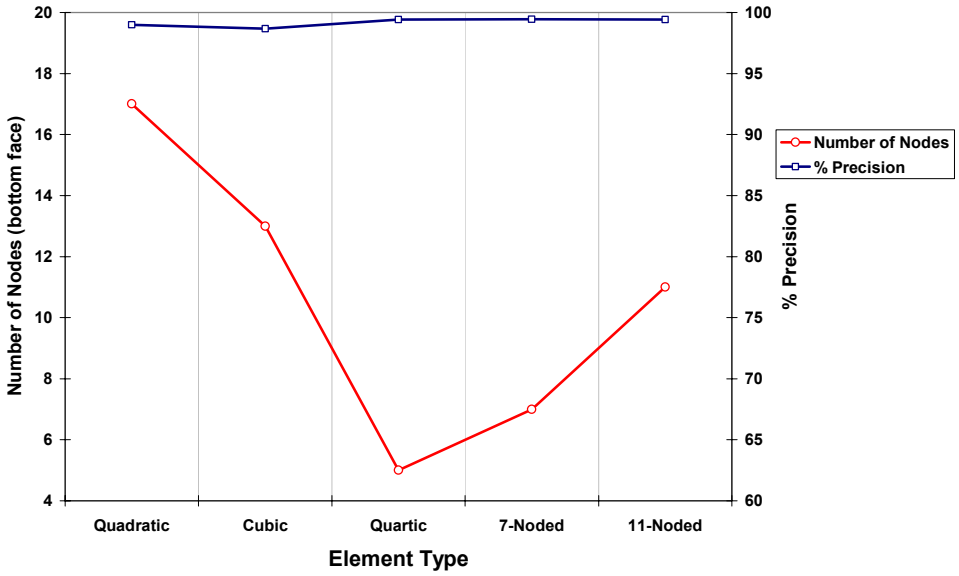

Figure 8 – Performance of different HO elements – Internal points results

7. Analysis of results

Problem 1 – "Point" load

Boundary results

In this case, all elements with order higher than quadratic showed the ability to deal satisfactorily with extreme situation where quadratic elements presented bad results. This demonstrates that the parabolic nature of the quadratic element is not sufficient to promptly describe the cubic variation of the displacements in the bending problem.

For the quadratic elements, some quick calculations show that, for case 1, the error in normal stresses is about 4.5%, while for the HO elements the best result is achieved with 13 noded elements, with 0.2% error. In case 2, the worst performance was obtained again with quadratic elements, with 352.1%, as expected, in comparison with 0.26% presented by cubic elements.

Internal results

In general, the internal results are very accurate in BEA, due to the fact that the governing differential equations are satisfied in the solution domain. For the quadratic

elements, some calculations reveal that the error of internal point results for the normal stresses is about 4.4% comparatively to the analytical solution, what can be considered a satisfactory performance, due to the small number of elements used (6). On the other hand, the error in shear stresses is in the order of 22.7%. For the higher order elements, the errors in normal stresses, for all elements, are under 0.6%, with better performance of the 5-and 7-noded elements, with 0.38%. With regard to the shear stresses, these errors vary between 13.9% for the cubic element and 4.65%, for the 7 noded elements.

Problem 2 – Uniform load

The result obtained shows that, to correctly capture the theoretical variation of the displacements (that is of order 4 in this case), higher order elements are well-suited. However, it remains difficult to define exactly what should be an adequate order to use, to solve a particular problem. Figure 5 to 7 shows that convergence can be accomplished with elements of different order.

Nevertheless, if a fine first approximation is desired or a faster convergence is intended, Figures 7 and 8 shows that the element which order matches the order of the variation of the displacements may be understood, primarily, as an adequate alternative. In the same sense, Figure 8 revels that, in this simple case, the global performance of quartic elements was superior.

8. Conclusion

The main goal of this paper was to present adequate computational tools to deal with the differential geometry of boundary elements, so that higher order elements can be used with versatility in BE analysis. Also, it became apparent that the adoption of a generalist approach to deal with the intrinsic space constitutes a central aspect to accomplish flexibility in BEA, particularly while implementing the integration process of the kernel tensor functions and element subdivision for these macro elements.

Tested in a somewhat simple manner, these elements confirmed their effectiveness to analyze slender components in bending. On the other hand, the results made possible to confirm that the order of the element required to solve a particular case, may be strongly dependent on the nature of the problem, even when higher order elements are employed. So, the expected behavior of the response may be decisive while choosing appropriate elements to model a specific problem. To accomplish this, an adaptive process can be conveniently implemented to check for convergence using elements of different order, just regrouping nodes, thus preserving the number of unknowns in the problem.

Acknowledgements

This research has not been sponsored. However, the continued support provided by FAPESP and CNPQ (Brazilian Founding Agencies) to researchers working in different areas, is gratefully recognized and appreciated.

References

1. Brebbia, C. A, Telles, J. C. F, Wrobel, L. C. - "Boundary element techniques – Theory and applications in engineering". Springer-Verlag Berlin, Heidelberg, Germany, 1984.

2. Zienkiewicz, O. C., Taylor, R. L. -"The finite element method - Basic formulation and linear problems". Vol. 1, 4th ed., McGraw-Hill Book Company, England, 1994.

3. Miranda-Valenzuela, J. C., Muci-Küchler, K. H. – "Adaptive meshing with boundary elements", Topics in Engineering, Vol. 41, WIT Press Southampton, UK, 2002.

4. Kane, J. H.-"Boundary element analysis in engineering continuum mechanics". Prentice-Hall, Inc., New Jersey, 1994.

5. Beer, G. "Programming the boundary element method – An introduction for engineers". John Wiley & Sons, Ltd, England, 2001.

6. Becker, A. A. "The boundary element method in engineering – A complete course". McGraw-Hill Book Company, UK, 1992.

7. Brebbia, C. A., Domingues, J. "Boundary elements – An introductory course". Computational Mechanics Publications, 1992.

8. Telles J. C. F. "A self-adaptive co-ordinate transformation for efficient numerical evaluation of general boundary element integrals", International Journal for Numerical Methods in Engineering; 24:959-973, 1987.

9. Ma, J., Rokhlin, V., Wandzura, S. "Generalized Gaussian quadrature rules for systems of arbitrary functions". SIAM Journal on Numerical Analysis; 33(3):971-996, 1996.

10. Ferreira, L. E. T. "ELASCON – A code to solve 2D linear elastic problems using BEM and HO elements" – software available from http://leteixei.vilabol.uol.com.br

11. Zhao, Z. "On the calculation of boundary stresses in boundary elements". Engineering Analysis with Boundary Elements, 16: 317-322, 1995.

12. Zhao, Z. "Interelement stress evaluation by boundary elements", International Journal for Numerical Methods in Engineering, 39: 2399-2415, 1996.

13. Chen, Y. Z."An accurate technique for evaluating stress at boundary points in boundary element method", Engineering Analysis with Boundary Elements, 24: 357-360, 2000.

14. Timoshenko, S. "Resistencia de materiales - teoría y problemas más complejos, Vol. 2, Espasa-Calpe S.A., Madrid, 1975.

Appendix A – Subroutine for the computation of shape functions

```
      SUBROUTINE SHAPE_FUNCTION (a, b, K, QSI, N)
! ------------------------------------------------------------------------------------------------------
! To compute the values of the shape functions for a given one-dimensional element
! Written by: Luiz Eduardo T. Ferreira (leferrei@uol.com.br) -Last modified: 05/23/04
!                NOTE: ALL REAL VARIABLES ARE DOUBLE PRECISION
!-------------------------------------------------------------------------------------------------------
      IMPLICIT NONE
!     Entries:
      REAL(8),    INTENT(IN) :: a, b  !Lower and upper limits of intrinsic elem.
      INTEGER, INTENT(IN) :: K     !Number of nodes of element
      REAL(8),   INTENT(IN):: QSI   !Position in which the S.F are to be evaluated
      REAL(8),   INTENT(OUT),DIMENSION(K)  :: N    !Array with results
!     Local variables:
      REAL(8)              :: DELTA_QSI, NUMERATOR, DENOMINATOR
      REAL(8),DIMENSION(K)                    :: QSI_N
      INTEGER                                 :: I, J
!     Process begins here, initialize variables:
      I=0; J=0
      N                    = 0.0D0      ! Array to store results
      QSI_N                = 0.0D0      ! Nodal positions
      NUMERATOR            = 1.0D0      ! Numerator   of Eq.1
      DENOMINATOR          = 1.0D0      ! Denominator of Eq.1
      DELTA_QSI      = (b-a) / DBLE (K-1) ! Increment for the norm. coordinates
!     Fill the array of the nodal positions:
      NODES_OF_ELEM: &
            DO I=1, K
                  QSI_N(I)=a + DBLE(I-1)*DELTA_QSI
            END DO  &
      NODES_OF_ELEM
!     Compute shape functions values, Ni, for all the 'K' nodes of element:
      SHAPE_FUN_Ni: &
      DO I=1, K
!     For all 'J' positions over the intrinsic element, compute Eq.1:
            PRODUCTORY: &
                  DO J=1, K
                  IF (J==I) CYCLE PRODUCTORY
                  NUMERATOR= NUMERATOR*(QSI-QSI_N(J))
                  DENOMINATOR= DENOMINATOR*(QSI_N(I)-QSI_N(J))
                  END DO  &
            PRODUCTORY
      N(I)=NUMERATOR/DENOMINATOR
      NUMERATOR     = 1.0D0
      DENOMINATOR  = 1.0D0
      END DO &
      SHAPE_FUN_Ni
      END SUBROUTINE SHAPE_FUNCTION
```

Appendix B – Subroutine for the computation of shape functions derivatives

```
SUBROUTINE SHAPE_FUNCTION_DERIV (a, b, K, QSI, dN)
! ------------------------------------------------------------------------------------------------------
! To compute the values of the deriv. of shape functions for a given element
! Written by: Luiz Eduardo T. Ferreira (leferrei@uol.com.br) - Last modified: 05/23/04
!                    NOTE: ALL REAL VARIABLES ARE DOUBLE PRECISION
!-------------------------------------------------------------------------------------------------------
        IMPLICIT NONE
!       Entries:
        REAL(8), INTENT(IN)  :: a, b ! Lower and upper limits of intrinsic elem.
        INTEGER, INTENT(IN):: K    ! Number of nodes of element
        REAL(8), INTENT(IN)  :: QSI ! Position in which the deriv. is to be evaluated
        REAL(8), INTENT(OUT), DIMENSION(K) :: dN ! Array with results
!       Local variables:
        REAL(8) :: DELTA_QSI, DENOMINATOR, SUM1, SUM2, PROD1, PROD2
        REAL(8),DIMENSION(K) :: QSI_N
        INTEGER                         :: I, J, N
!       Process begins here, initialize variables:
        I=0; J=0; N=0
        dN              = 0.0D0
        QSI_N           = 0.0D0                 ! Nodal positions
        DELTA_QSI= (b-a)/DBLE (K-1)  ! Increment for the norm. coordinate
!       Fill the array of the nodal positions:
        NODES_OF_ELEM: &
                DO I=1, K
                        QSI_N(I)=a + DBLE(I-1)*DELTA_QSI
                END DO  &
        NODES_OF_ELEM
!       Compute derivatives of shape functions, dNi, for all the 'k' nodes of element:
        D_SHAPE_FUNCTION: &
        DO I=1, K
        DENOMINATOR = 1.0D0
!       For all 'J' positions on the normalized element, compute first productory:
                PRODUCTORY1: &
                        DO J=1, K
                        IF (J==I) CYCLE PRODUCTORY1
                        DENOMINATOR=DENOMINATOR*(QSI_N(I)-QSI_N(J))
                        END DO  &
                PRODUCTORY1
        SUM1 =0.0D0; SUM2 =0.0D0
!       Compute first summation:
                SUMMATION1: &
                        DO N=I, K-1
                        PROD1=1.0D0
!       Now, the inner productory:
```

```
                    PRODUCTORY2: &
                    DO J=1, K
                        IF (J==I.OR.J==N+1)CYCLE PRODUCTORY2
                        PROD1 = PROD1*(QSI-QSI_N(J))
                    END DO  &
                    PRODUCTORY2
!
                    SUM1=SUM1+PROD1
                    END DO &
                SUMMATION1
!       Compute second summation:
                SUMMATION2: &
                    DO N=1, I-1
                    PROD2=1.0D0
!       Now, the inner productory :
                    PRODUCTORY3: &
                    DO J=1, K
                        IF (J == I.OR.J == N) CYCLE PRODUCTORY3
                        PROD2 = PROD2*(QSI-QSI_N(J))
                    END DO  &
                    PRODUCTORY3
!
                    SUM2=SUM2+PROD2
                    END DO &
                SUMMATION2
!       Store the value computed at this node:
        dN(I)=(SUM1+SUM2)/DENOMINATOR
!
        END DO &
        D_SHAPE_FUNCTION
!
        END SUBROUTINE SHAPE_FUNCTION_DERIV
```