INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2006; **00**:1–28 *Prepared using* nmeauth.cls *[Version: 2002/09/18 v2.02]*

A study on the performance of Natural Neighbour-based Galerkin Methods

I. Alfaro[1], J. Yvonnet[2], F. Chinesta[3] and E. Cueto[1,*]

[1] *Group of Structural Mechanics and Material Modelling. Aragón Institute of Engineering Research (I3A). University of Zaragoza. María de Luna, 5. Campus Rio Ebro. E-50018 Zaragoza, Spain.*
[2] *Laboratoire de Mécanique. Université Marne-la-Vallée -5. Bd. Descartes, F-77454 Marne-la-vallée Cedex 2. France.*
[3] *Laboratoire de Mécanique des Systèmes et des Procédés, LMSP UMR 8106 CNRS-ENSAM-ESEM. 105, Bvd. de l'Hôpital. F-75013 Paris, France.*

KEY WORDS: Meshless, Natural Element Method, Sibson interpolation, Laplace interpolation, *pseudo*-NEM, Natural neighbour search, computational cost.

SUMMARY

In this paper we address the problem of improving Natural Element simulations in terms of computational cost. Several problems are discussed, that include an efficient natural neighbour search algorithm and a comparison of different natural neighbour-based interpolation algorithms. In particular, we review the so-called *pseudo*-NEM, a MLS-like approximation scheme that employs natural neighbours, and compare it with traditional Sibson and Laplace interpolation schemes in terms of both accuracy and computational cost. Some examples in linear Elasticity and visco-plasticity are analyzed in order to test the proposed schemes in engineering problems. Copyright © 2006 John Wiley & Sons, Ltd.

Contents

*Correspondence to: Elías Cueto. Mechanical Engineering Department. Edificio Betancourt. University of Zaragoza. María de Luna, 5. E-50018 Zaragoza, Spain. e-mail: ecueto@unizar.es

Contract/grant sponsor: Spanish Ministry of Education and Science; contract/grant number: CICYT-DPI2005-08727-C02-01

Received July 2006
Revised
Accepted

1. INTRODUCTION

Performance of a numerical method in terms of computational cost is a critical issue. Although computers are growing exponentially in terms of speed of computation and capacity of storage, so does user's needs. The case of meshless methods is particularly noteworthy. It is well-known that nowadays meshless methods greatly alleviate the user time employed in meshing the domain of interest, and significantly reduce the time employed in remeshing, that is nearly completely eliminated in practice, since the distortion of the mesh (or cloud of nodes) does not affect the accuracy of the results.

In a meshless method, instead of constructing a mesh on a given domain (thus placing nodes in appropriate positions so as to obtain well-shaped elements), usually the approach is reversed and the objective is, given a cloud of points, to obtain their connectivity. This connectivity can be sought based on a distance criterion (as in the EFGM [3] or the RKPM [19]) or other type of neighbourhood (natural neighbourhood in the case of the NEM, for instance, see [23]).

In a recent paper devoted to these topics [15] it has been said that for a meshless method to be competitive, " ... the evaluation of the nodal connectivity [should be] bounded in time and linear with the total number of nodes in the domain". Despite that the constants accompanying the linear law are also important (i.e., the relative amount of time employed in generating the connectivity, despite that it varies linearly or not), it is clear that the generation of the connectivity is a crucial issue in meshless methods.

In the NEM the generation of the connectivity is done by searching for natural neighbours of the nodes. Natural neighbours of a given node are those nodes that share with it an edge of a Delaunay triangle or tetrahedron. Usually, however, natural neighbourhood is sought between an integration point and the nodes whose support cover the mentioned integration point. Thus efficient natural neighbour search algorithm should be employed in order to make the NEM competitive. This constitutes one of the main objectives of this papers.

In a Galerkin procedure, there is another issue that should be addressed. The computation of the approximation (shape) functions and their derivatives is also important. Since the irruption of the very first meshless methods [21] it was noted that the computation of the shape functions, and especially their derivatives, can be a very costly process. This was one of the early motivations of the so-called "diffuse derivatives" in the work of Villon and colleagues before mentioned in the context of MLS approximations.

In the NEM framework, two different interpolants have been proposed. The original version of the method [4] employed natural neighbour (Sibson) interpolation [23]. Sibson interpolation is known to be a costly procedure, although it is also well-known in the approximation community as a high-quality interpolation scheme with many advantages (see, for instance, [13]). Sukumar [26] later proposed the use of Laplace (also known as non-Sibsonian) interpolation [14]. This interpolant is considerably less costly than the original Sibson interpolant, although somewhat less smooth.

Some of the advantages of Natural Neighbour interpolation have been attributed to the particular geometry of the shape functions' support, that remains quite isotropic even for large distortions of the original cloud of points. Based on this, Reddy [22] and Yvonnet [30] proposed independently a MLS procedure that employs a weight function whose support is the same as the one of the NEM (we hereafter refer to this approximation as *pseudo*-NEM). This method runs considerably faster than traditional NEM methods (for equivalent reproducing conditions) and can constitute an interesting alternative to alleviate this high cost of the NEM.

These and other issues are discussed in this paper. Firstly, in section 2, a review of natural neighbour interpolation is made, with special emphasis on the newly proposed *pseudo*-NEM approximation before mentioned.

In section 3 we propose a efficient natural-neighbour search algorithm that runs in linear time and that has proved to be very fast, especially when using a large amount of nodes.

In section 4 we analyze the computational cost associated to each interpolation procedure in a series of examples in linear elasticity and visco-plasticity. We discuss the relative importance of this cost in linear and non-linear problems, with regular and irregular clouds of different densities. Finally, in section 5, some conclusions are drawn.

2. NATURAL NEIGHBOUR INTERPOLATION

Essentially, the NEM is a Galerkin procedure that relies on natural neighbour interpolation to construct the trial and test functions characteristic of this method. As mentioned before, there exist nowadays different interpolation procedures that are based upon natural neighbourhood. In this section we review three of them, although there exist others.

Prior to this definition we introduce some basic geometrical entities that are needed for further developments.

2.1. Voronoi/Dirichlet diagrams

Consider a model composed by a cloud of points $\boldsymbol{N} = \{n_1, n_2, \ldots, n_m\} \subset \mathbb{R}^d$, for which there is a unique decomposition of the space into regions such that each point within these regions is closer to the node to which the region is associated than to any other in the cloud. This kind of space decomposition is called a Voronoi diagram (also Dirichlet tessellation) of the cloud of points and each Voronoi cell is formally defined as (see figure 1):

$$T_I = \{\boldsymbol{x} \in \mathbb{R}^d : d(\boldsymbol{x}, \boldsymbol{x}_I) < d(\boldsymbol{x}, \boldsymbol{x}_J) \; \forall \; J \neq I\}, \tag{1}$$

where $d(\cdot, \cdot)$ is the Euclidean distance function.

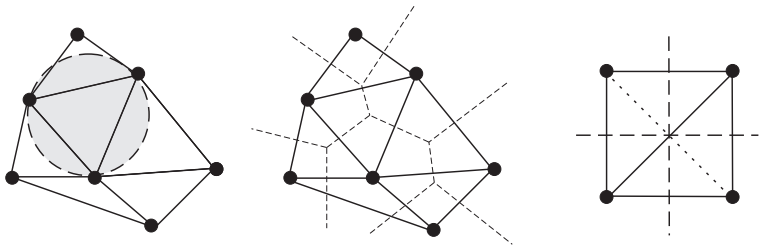


Figure 1. Delaunay triangulation and Voronoi diagram of a cloud of points.

The dual structure of the Voronoi diagram is the Delaunay triangulation[†], obtained by connecting nodes that share a common $(d-1)$-dimensional facet. While the Voronoi structure is unique, the Delaunay triangulation is not, there being some so-called *degenerate* cases in which there are two or more possible Delaunay triangulations (consider, for example, the case of triangulating a square in 2D, as depicted in Fig. 1 (right)). Another way to define the Delaunay triangulation of a set of nodes is by invoking the *empty circumcircle* property, which means that no node of the cloud lies within the circle covering a Delaunay triangle. Two nodes sharing a facet of their Voronoi cell are called *natural neighbours* and hence the name of the technique.

Equivalently, the second-order Voronoi diagram of the cloud is defined as

$$T_{IJ} = \{\boldsymbol{x} \in \mathbb{R}^d : d(\boldsymbol{x}, \boldsymbol{x}_I) < d(\boldsymbol{x}, \boldsymbol{x}_J) < d(\boldsymbol{x}, \boldsymbol{x}_K) \; \forall \; J \neq I \neq K\}. \tag{2}$$

Based on these definitions, different natural neighbour interpolation schemes have been proposed. We review some of the most popular.

2.2. Thiessen interpolation

The simplest of the natural neighbour-based interpolants is the so-called Thiessen's interpolant [27]. Its interpolating functions are defined as

$$\psi_I(\boldsymbol{x}) = \begin{cases} 1 \text{ if } \boldsymbol{x} \in T_I \\ 0 \text{ elsewhere} \end{cases}. \tag{3}$$

The Thiessen interpolant is a piece-wise constant function, defined over each Voronoi cell. It defines a method of interpolation often referred to as *nearest neighbour* interpolation, since a point is given a value defined by its nearest neighbour. Although it is obviously not valid for the solution of second-order partial differential equations, it can be used to interpolate the pressure in formulations arising from Hellinger-Reissner-like mixed variational principles, as proven in [12].

[†]Even in three-dimensional spaces, it is common to refer to the Delaunay tetrahedralisation with the word *triangulation* in the vast majority of the literature

2.3. Sibson's interpolation

The most extended natural neighbour interpolation method, however, is the Sibson interpolant [23] [24]. Consider the introduction of the point $\boldsymbol{x}$ in the cloud of nodes. Due to this introduction, the Voronoi diagram will be altered, affecting the Voronoi cells of the natural neighbours of $\boldsymbol{x}$. Sibson [23] defined the natural neighbour coordinates of a point $\boldsymbol{x}$ with respect to one of its neighbours I as the ratio of the cell T_I that is transferred to T_x when adding $\boldsymbol{x}$ to the initial cloud of points to the total volume of T_x. In other words, if $\kappa(\boldsymbol{x})$ and $\kappa_I(\boldsymbol{x})$ are the Lebesgue measures of T_x and T_{xI} respectively, the natural neighbour coordinates of $\boldsymbol{x}$ with respect to the node I is defined as

$$\phi_I(\boldsymbol{x}) = \frac{\kappa_I(\boldsymbol{x})}{\kappa(\boldsymbol{x})}. \tag{4}$$

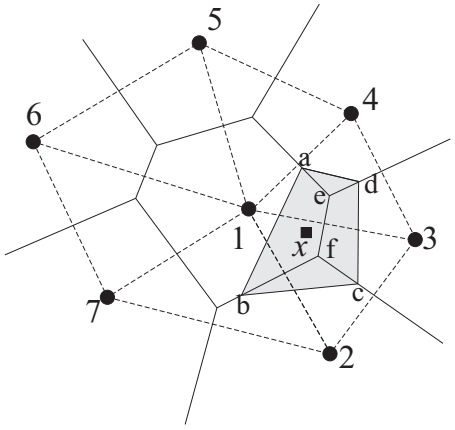


Figure 2. Definition of the Natural Neighbour coordinates of a point $\boldsymbol{x}$.

In Fig. 2 the shape function associated to node 1 at point $\boldsymbol{x}$ may be expressed as

$$\phi_1(\boldsymbol{x}) = \frac{A_{abfe}}{A_{abcd}}. \tag{5}$$

Sibson's interpolation scheme possesses the usual reproducing properties for this class of problems, i.e., verify the *partition of unity* property (constant consistency), linear consistency (and therefore are suitable for the solution of second-order PDE). Other interesting properties such as the Kronecker delta property [25] and linear interpolation on the boundary [7] [31] are also verified by the NEM.

2.4. Laplace interpolation

Recently, some new interpolation schemes based on the concept of natural neighbors have been proposed [14] [2]. Its application in the context of the NEM dates back to the work of Sukumar [26]. This so-called Laplace or *non-Sibsonian* interpolation, has received considerable attention, since it involves magnitudes of one order less of the space dimension (i.e., the calculation of areas in three-dimensional problems, for instance, instead of volumes). If we define the cell

intersection $t_{IJ} = \{\boldsymbol{x} \in T_I \bigcap T_J, J \neq I\}$ (note that t_{IJ} may be an empty set) we can define the value

$$\alpha_J(\boldsymbol{x}) = \frac{|t_{\boldsymbol{x}J}|}{d(\boldsymbol{x}, \boldsymbol{x}_J)}. \tag{6}$$

Thus, the shape function related to node 4 at point $\boldsymbol{x}$ in Fig. 3 is defined as

$$\phi_4^{ns}(\boldsymbol{x}) = \frac{\alpha_4(\boldsymbol{x})}{\sum_{J=1}^{n} \alpha_J(\boldsymbol{x})} = \frac{s_4(\boldsymbol{x})/h_4(\boldsymbol{x})}{\sum_{J=1}^{n} \left[s_J(\boldsymbol{x})/h_J(\boldsymbol{x})\right]}, \tag{7}$$

where s_J represent the length of the Voronoi segment associated to node J and n represents the number of natural neighbours of the point under consideration, $\boldsymbol{x}$. $n = 4$ in this example.

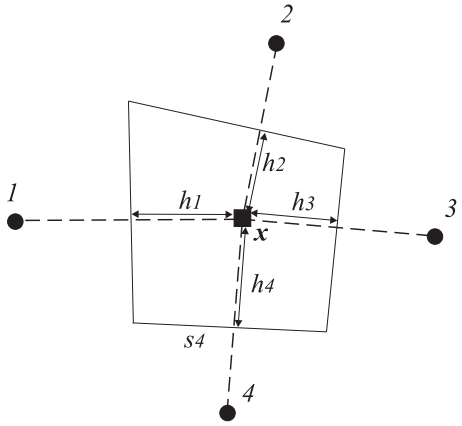


Figure 3. Definition of non-Sibsonian coordinates.

Derivatives of the Laplace shape function are not defined along the edges of the Delaunay triangles that lie inside its support (see [26]).

This same interpolation scheme is used in the Meshless Finite Element Method (MFEM) context [16], but in this case, the space is divided into polyhedral elements and Laplace interpolation is constructed within each polyhedron. Since the same properties mentioned for Sibson interpolation (i.e., linear completeness, exact interpolation on the nodes and linear interpolation along the boundaries) hold, a conforming method is achieved.

2.5. Pseudo-NEM interpolation

As mentioned before, one of the key issues in the high quality of the interpolation achieved by the NEM despite the large distortion of the underlying triangulation is that the support of the shape function remains quite isotropic, since it is composed by the union of circumcircles covering a given evaluation point.

Based on this analysis, Reddy and coworkers [22] and Yvonnet et al. [30] proposed independently a Moving Least Squares (MLS) technique whose weighting function possesses NEM-like support. But in order to alleviate the burden associated to the computation of natural neighbour shape functions, they substitute it by a simplified one, composed by the union of conical volumes.

Let

$$\boldsymbol{u}^h = \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{a}(\boldsymbol{x}) \tag{8}$$

be a polynomial approximation to the essential field of the problem, $\boldsymbol{u}$, with $\boldsymbol{p}^T(\boldsymbol{x}) = [1, x, y, z, \ldots]$ a polynomial basis to a given order, and $\boldsymbol{a}(\boldsymbol{x})$ a vector of unknown coefficients. To determine $\boldsymbol{a}(\boldsymbol{x})$, the functional

$$J = \frac{1}{2}\sum_{i=1}^{n} w_i(\boldsymbol{x})[\boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{a}(\boldsymbol{x}) - u_i]^2 \tag{9}$$

is minimized with respect to $\boldsymbol{a}$. $w_i(\boldsymbol{x})$ represents a weighting function associated to the node i. See [21] [3] for more details. This leads to the linear system

$$\boldsymbol{A}\boldsymbol{a} = \boldsymbol{B}\boldsymbol{u}, \tag{10}$$

where

$$A_{jk} = \sum_{i=1}^{n} w_i(\boldsymbol{x})p_j(\boldsymbol{x}_i)p_k(\boldsymbol{x}_i), \tag{11}$$

$$B_{ij} = w_i(\boldsymbol{x})p_j(\boldsymbol{x}). \tag{12}$$

This leads to

$$\boldsymbol{u}^h(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{u}, \tag{13}$$

and thus, we can speak of a shape function

$$\phi(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{A}^{-1}\boldsymbol{B}. \tag{14}$$

The key point in this approach is the choice of the weight function, that in this case is a conical function. Let f be a cone, of unit height, whose basis matches the corresponding Delaunay circumcircle containing the evaluation point $\boldsymbol{x}$ and whose tip projection over the horizontal plane is the node n_i (see Fig. 4):

$$f = \frac{\|\overrightarrow{\boldsymbol{n}_i\boldsymbol{P}}\| - \|\overrightarrow{\boldsymbol{n}_i\boldsymbol{x}}\|}{\|\overrightarrow{\boldsymbol{n}_i\boldsymbol{P}}\|}, \tag{15}$$

with

$$\overrightarrow{\boldsymbol{n}_i\boldsymbol{P}} = -2\Big(\frac{\overrightarrow{\boldsymbol{c}\boldsymbol{n}_i} \cdot \overrightarrow{\boldsymbol{n}_i\boldsymbol{x}}}{\overrightarrow{\boldsymbol{n}_i\boldsymbol{x}} \cdot \overrightarrow{\boldsymbol{n}_i\boldsymbol{x}}}\Big)\overrightarrow{\boldsymbol{n}_i\boldsymbol{x}}. \tag{16}$$

In order to avoid the overlapping of cone functions, whereby conserving the continuity of the weight function, a cone portion is associated with each of the Delaunay triangles connected to node n_i. The cone function is thus non-zero if a point $\boldsymbol{x}$ belongs to the intersection between the Delaunay circumcircle and the portion of the plane such as any point in the basis formed by the origin node $\boldsymbol{n}_i$ and the vectors $\boldsymbol{n}_i\boldsymbol{n}_j$ and $\boldsymbol{n}_i\boldsymbol{n}_k$ has positive coordinates in this basis. $\boldsymbol{n}_j$ and $\boldsymbol{n}_k$ being the other two vertices of the triangle (see Fig. 5). Due to the particular shape of its support, which is defined for any nodal distribution, this weight function guarantees interpolation conditions ($w_i(\boldsymbol{x}_j) = \delta_{ij}$); as Delaunay circles passes through the nodes. Furthermore, the properties of positiveness and monotonically decrease are verified. Since the cone functions are linear between two nodes, the continuity of the weight function is guaranteed.

Sibson, Laplace and *pseudo*-NEM shape functions for a regular lattice of 5×5 nodes are depicted in Fig. 6.

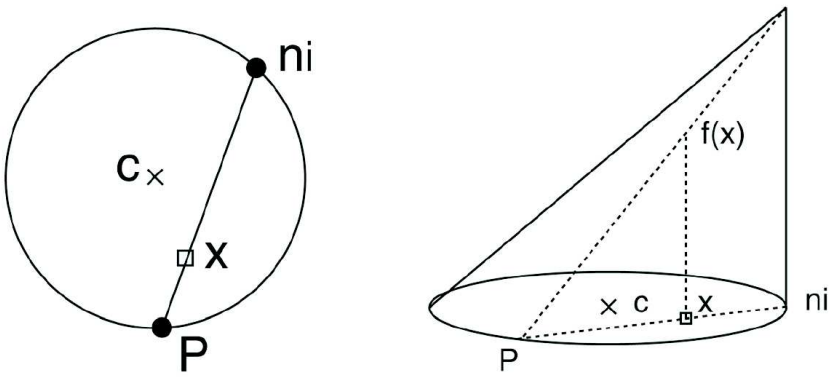


Figure 4. Definition of the conical weighting function f.

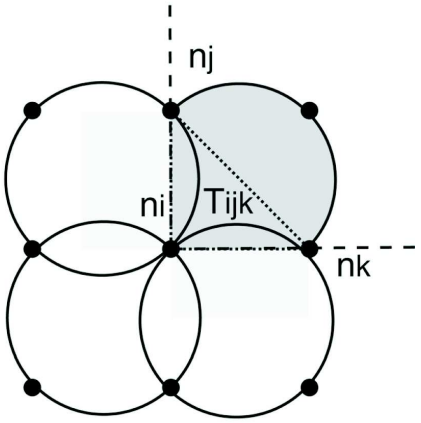


Figure 5. Zone associated with a cone portion.

2.6. Computation of natural neighbour shape functions

The computation of natural neighbour (Sibson or Laplace) shape functions deserves some comments. In three-dimensional settings, it is necessary to compute volumes of Voronoi cells (polyhedra) for the Sibson interpolant and areas of the corresponding Voronoi facets (poligons) for Laplace interpolants. Both schemes share many features, although expressed in different dimensions.

In two dimensions, the algorithm proposed by Watson [29] is especially efficient. Unfortunately, this algorithm seems not to be straightforwardly extended to three-dimensional settings.

In our experience, the algorithm proposed by Lasserre [17] for the computation of the volume

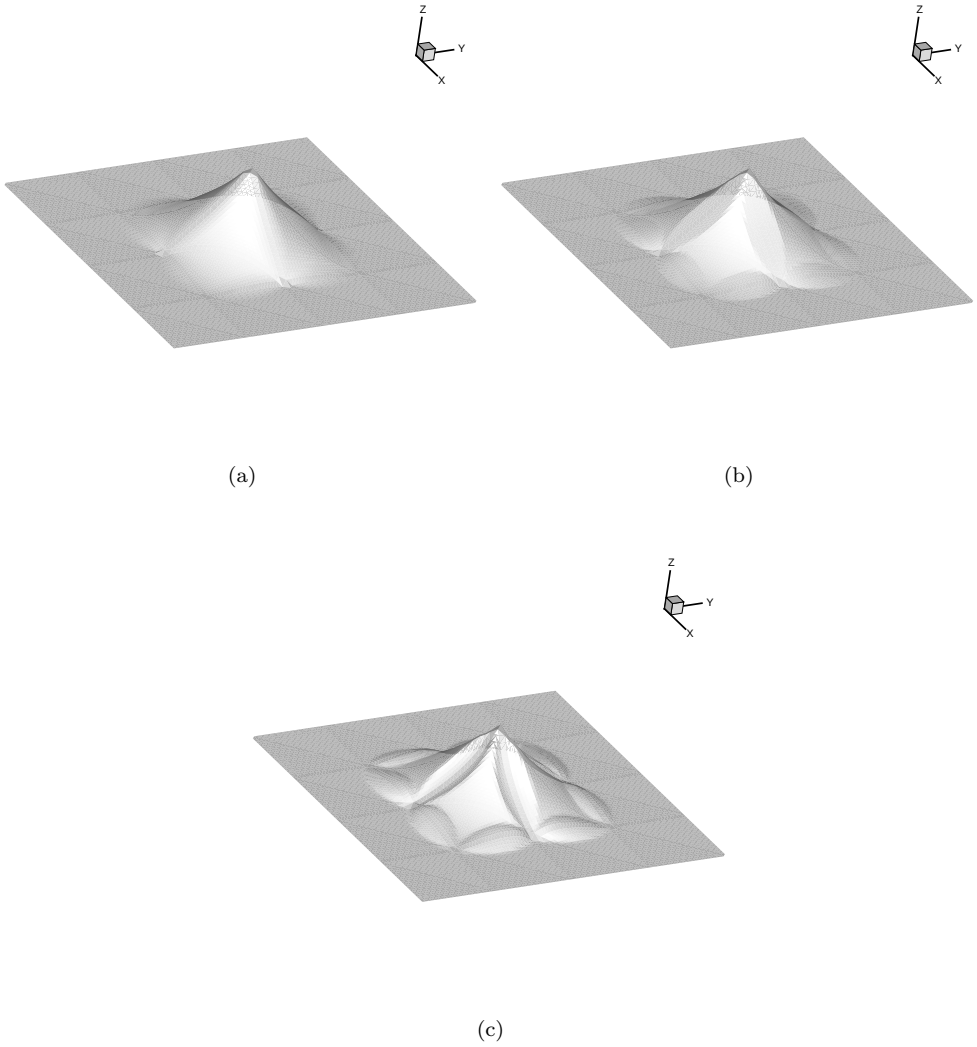


Figure 6. Sibson (a), Laplace (b) and *pseudo*-NEM (c) shape functions.

of polyhedra seems to be the most appropriate and has been employed in the three-dimensional examples included in this work. For an in-depth discussion on the Watson's algorithm, the interested reader should read [25] or [8].

We review here Lasserre's algorithm, since we consider that three-dimensional problems are both the most interesting cases and the most time-consuming. Lasserre's algorithm begins by expressing a convex polyhedron's volume (first or second order Voronoi cell) in the form of a

set of inequalities in $\mathbb{R}^d$ that may be written as

$$\{\boldsymbol{x}|\boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{b}\}, \tag{17}$$

where $\boldsymbol{x}$ represents, as usual, a point in $\mathbb{R}^d$, $\boldsymbol{R}$ is a matrix of dimension (m, d) and $\boldsymbol{b}$ is a column vector of dimension m (number of restrictions that define the volume). The volume enclosed by the polytope (d-dimensional counterpart of a polygon) is then given by

$$V(d, \boldsymbol{R}, \boldsymbol{b}), \tag{18}$$

the i-th face of the polytope given by:

$$\{\boldsymbol{x}|(\boldsymbol{r}_i \cdot \boldsymbol{x}) = b_i, \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{b}\}, \tag{19}$$

where $\boldsymbol{r}_i$ represents the i-th row of $\boldsymbol{R}$. This face's volume, in $\mathbb{R}^{d-1}$ space is denoted by:

$$V_i(d-1, \boldsymbol{R}, \boldsymbol{b}). \tag{20}$$

A traditional way of computing the volume of the polytope is given by:

$$V(d, \boldsymbol{R}, \boldsymbol{b}) = \frac{1}{d} \sum_{i=1}^{m'} d(\boldsymbol{p}, \boldsymbol{H}_i) \times V_i(d-1, \boldsymbol{R}, \boldsymbol{b}), \tag{21}$$

where $\boldsymbol{p}$ is a fixed point in the space, m' the minimum number of restrictions that define the polytope (no redundant restrictions are considered) and $d(\boldsymbol{p}, \boldsymbol{H}_i)$ the distance from point $\boldsymbol{p}$ to the hyperplane $\boldsymbol{H}_i$ given by the i-th restriction that defines the volume.

This algorithm can be constructed in a recursive way, so the volume computation is performed in the form of a binary tree, beginning by dimension d and leading to the computation of some lengths in $\mathbb{R}$. This volume can then be computed by:

$$V(d, \boldsymbol{R}, \boldsymbol{b}) = \frac{1}{d} \sum_{i=1}^{m} \frac{b_i}{|r_{it}|} V'_{it}(d-1, \overline{\boldsymbol{R}}_{i,t}, \overline{\boldsymbol{b}}_t). \tag{22}$$

In this expression $\overline{\boldsymbol{R}}_{i,t}$ represents the reduced matrix, obtained from $\boldsymbol{R}$ by elimination of the t-th variable, by means of the equation $\boldsymbol{r}_i\boldsymbol{x} = b_i$; $\overline{\boldsymbol{b}}_t$ is the reduced vector after this elimination and r_{it} is the t-th element of $\boldsymbol{r}_i$. V'_{it} represents the volume in dimension $d-1$ obtained with the reduced matrix $\overline{\boldsymbol{R}}_{i,t}$ and the reduced vector $\overline{\boldsymbol{b}}_t$. The formulae used to calculate the $\overline{r}_{JK}$ and $\overline{b}_J$ terms of $\overline{\boldsymbol{R}}_{i,t}$ and $\overline{\boldsymbol{b}}_t$ are:

$$\overline{r}_{JK} = r_{jk} - \frac{r_{jt}}{r_{it}} r_{ik}. \tag{23}$$

$$\overline{b}_J = b_j - \frac{r_{jt}}{r_{it}} b_i. \tag{24}$$

In the work of Braun and Sambridge [4] the value of t is chosen such that

$$|r_{it}| = \max_j |r_{ij}|. \tag{25}$$

Note that in this case the sum is set up to m instead of m' since redundant restrictions are automatically eliminated by the algorithm.

Node	x	y
1	0	0
2	1	0
3	1.5	0.5
4	2	1
5	1.4	1.6
6	0.6	1.8
7	0	1
8	1	1

Table I. Nodes for the calculation of Sibson shape function.

In order to calculate the derivatives of the shape functions, Eqs. (22), (23) and (24) should be differentiated:

$$\frac{\partial V(d,\boldsymbol{R},\boldsymbol{b})}{\partial \boldsymbol{x}_I} = \partial_{\boldsymbol{x}_I} V(d,\boldsymbol{R},\boldsymbol{b}) = \frac{1}{d}\sum_{i=1}^{m}\left[\frac{b_i\cdot\partial_{\boldsymbol{x}_I}V'_{it}(d-1,\overline{\boldsymbol{R}}_{i,t},\overline{\boldsymbol{b}}_t)}{|r_{it}|}+\right.$$
$$\left.+\frac{V'_{it}(d-1,\overline{\boldsymbol{R}}_{i,t},\overline{\boldsymbol{b}}_t)}{|r_{it}|}\cdot\Big(\partial_{\boldsymbol{x}_I}b_i-\frac{b_i}{|r_{it}|}\partial_{\boldsymbol{x}_I}|r_{it}|\Big)\right], \quad (26)$$

$$\partial_{\boldsymbol{x}_I}\overline{r}_{JK} = \partial_{\boldsymbol{x}_I}r_{jk} - \partial_{\boldsymbol{x}_I}\Big(\frac{r_{jt}}{r_{it}}\Big)r_{ik} - \frac{r_{jt}}{r_{it}}\partial_{\boldsymbol{x}_I}r_{ik}, \quad (27)$$

$$\partial_{\boldsymbol{x}_I}\overline{b}_J = \partial_{\boldsymbol{x}_I}b_j - \partial_{\boldsymbol{x}_I}\Big(\frac{r_{jt}}{r_{it}}\Big)b_i - \frac{r_{jt}}{r_{it}}\partial_{\boldsymbol{x}_I}b_i, \quad (28)$$

where $\partial_{\boldsymbol{x}_I}$ represents the derivative in the $\boldsymbol{x}_I$ direction.

2.6.1. Example: computation of shape function through Lasserre's algorithm. In order to clarify the previous algorithm, consider the set of points in two dimensions given in Table I. We compute Sibson's shape function associated to node 8 at the point $(0.8, 0.8)$.

The evaluation point (labelled P in Fig. 7) has six neighbours. So it will be necessary to compute six areas corresponding to second-order Voronoi cells, formed by the integration point and its neighbours (highlighted in Fig. 7). To this end, we compute the list of neighbours:

- Neighbours of node 8: 8, 7, 1, 2, 3, 4, 5, 6
- Neighbours of the integration point: 8, 7, 1, 2, 3, 6
- Common neighbours: 8, 7, 1, 2, 3, 6

The equation defining the intersection of the hyperplane between two generic points A and B and the plane $\mathbb{R}^2$ (or, equivalently, $z = 0$ in $\mathbb{R}^3$), that contains these two points, is

$$(\boldsymbol{x}_A - \boldsymbol{x}_B)^T\boldsymbol{x} = \frac{\|\boldsymbol{x}_A\|^2 - \|\boldsymbol{x}_B\|^2}{2} \quad (29)$$

so, computing the different restrictions to form the matrix $\boldsymbol{R}$, we arrive to

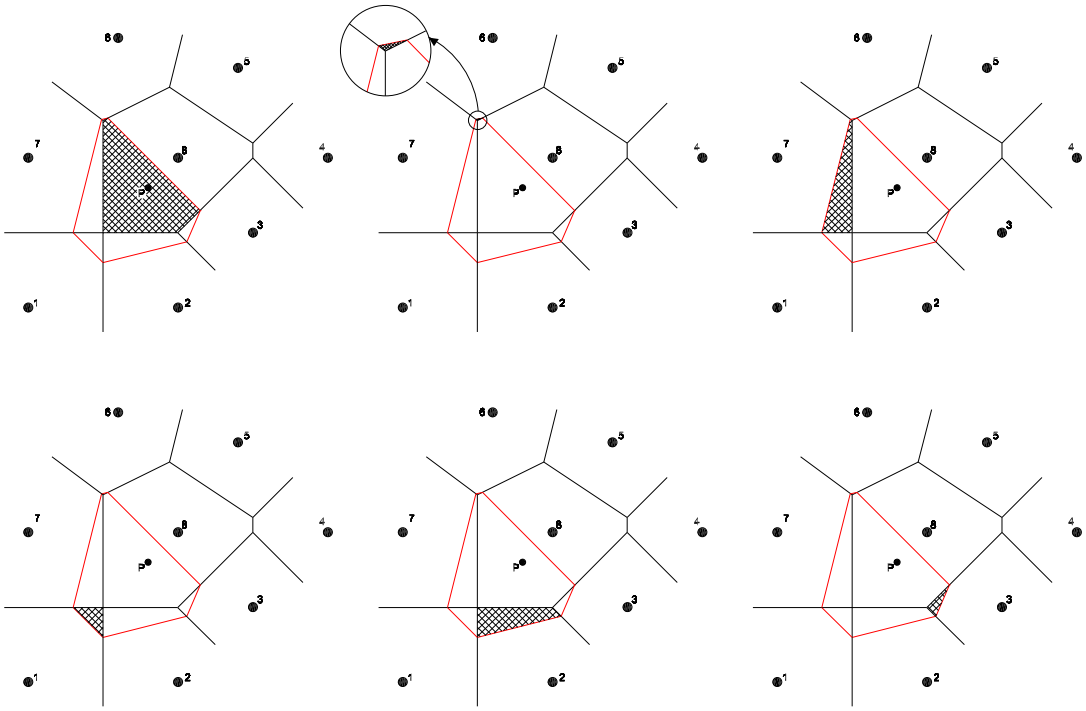


Figure 7. Configuration of the point set given in Table I and position of the integration point. The areas needed to perform the computation are highlighted.

Restriction	x_A	y_A	x_B	y_B	r_{ix}	r_{iy}	b_i
7-8	0	1	1	1	-1	0	-0.5
1-8	0	0	1	1	-1	-1	-1
2-8	1	0	1	1	0	-1	-0.5
3-8	1.5	0.5	1	1	0.5	-0.5	0.25
6-8	0.6	1.8	1	1	-0.4	0.8	0.8
8-p	1	1	0.8	0.8	0.2	0.2	0.36

Thus, matrix $\boldsymbol{R}$ and vector $\boldsymbol{b}$ defining the restrictions are in this case:

$$\boldsymbol{R} = \begin{pmatrix} x_7 - x_8 & y_7 - y_8 \\ x_1 - x_8 & y_1 - y_8 \\ x_2 - x_8 & y_2 - y_8 \\ x_3 - x_8 & y_3 - y_8 \\ x_6 - x_8 & y_6 - y_8 \\ x_8 - x_p & y_8 - y_p \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ -1 & -1 \\ 0 & -1 \\ 0.5 & -0.5 \\ -0.4 & 0.8 \\ 0.2 & 0.2 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} \frac{x_7^2+y_7^2-x_8^2-y_8^2}{2} \\ \frac{x_1^2+y_1^2-x_8^2-y_8^2}{2} \\ \frac{x_2^2+y_2^2-x_8^2-y_8^2}{2} \\ \frac{x_3^2+y_3^2-x_8^2-y_8^2}{2} \\ \frac{x_6^2+y_6^2-x_8^2-y_8^2}{2} \\ \frac{x_8^2+y_8^2-x_p^2-y_p^2}{2} \end{pmatrix} = \begin{pmatrix} -0.5 \\ -1 \\ -0.5 \\ 0.25 \\ 0.8 \\ 0.36 \end{pmatrix}. \tag{30}$$

This set of restrictions is shown in a graphic manner in Fig. 8.

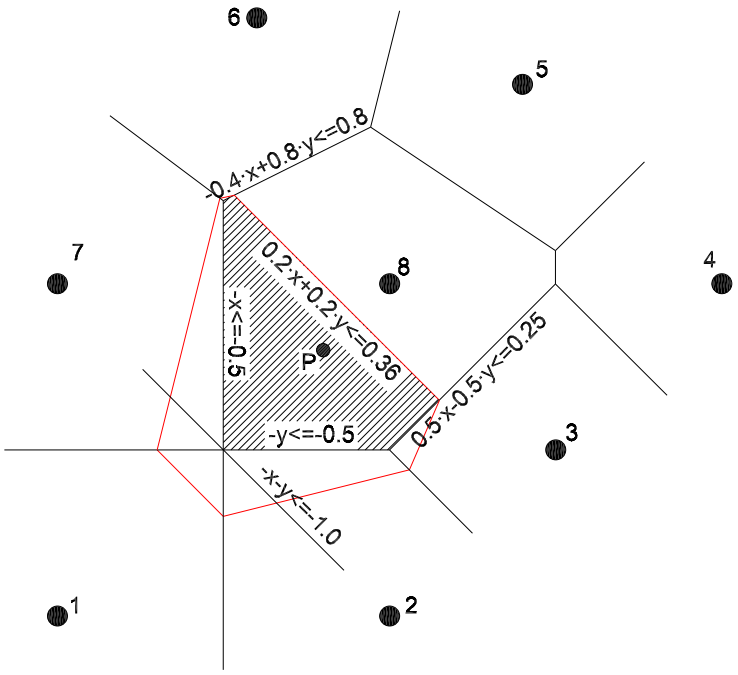


Figure 8. Graphical representation of the restriction set.

Note that in this case the only row of matrices $\boldsymbol{R}$ and $\boldsymbol{b}$ in Eq. (30) depending on the position of the evaluation point P is the last one. Thus, by deriving the Eq. (29), defining each row of the restriction set, with respect to the coordinates of the evaluation point, we arrive to:

$$\frac{\partial}{\partial x_p}\Big[x_i - x_p\Big] = -1, \quad \frac{\partial}{\partial y_p}\Big[x_i - x_p\Big] = 0, \quad \frac{\partial}{\partial x_p}\Big[y_i - y_p\Big] = 0, \quad \frac{\partial}{\partial y_p}\Big[y_i - y_p\Big] = -1. \tag{31}$$

$$\frac{\partial}{\partial x_p}\left[\frac{x_i^2 + y_i^2 - x_p^2 - y_p^2}{2}\right] = -x_p, \quad \frac{\partial}{\partial y_p}\left[\frac{x_i^2 + y_i^2 - x_p^2 - y_p^2}{2}\right] = -y_p. \tag{32}$$

$(i = 1, \ldots, m)$ and thus the derivatives of the restriction matrices are

$$\boldsymbol{dR} = \begin{pmatrix} \frac{\partial \boldsymbol{R}}{\partial x_p} & \frac{\partial \boldsymbol{R}}{\partial y_p} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 \end{pmatrix} \tag{33}$$

$$\boldsymbol{db} = \begin{pmatrix} \frac{\partial \boldsymbol{b}}{\partial x_p} & \frac{\partial \boldsymbol{b}}{\partial y_p} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.8 & -0.8 \end{pmatrix} \tag{34}$$

We start with $i = 1$ with respect to the sum in Eq. (22). The first step is to select the pivot and, therefore, the variable to be eliminated in the reduced matrices. In this case, in reference to Eq. (25), $|r_{it}| = max_j|r_{1j}| = |r_{11}| = 1$ and the new $\mathbb{R}^{d-1}$ space is the y-axis. We compute the x value and introduce it in the rest of equations, giving

$$\overline{\boldsymbol{R}}_{1,1} = \begin{pmatrix} \overline{r}_1 \\ \overline{r}_2 \\ \overline{r}_3 \\ \overline{r}_4 \\ \overline{r}_5 \end{pmatrix} = \begin{pmatrix} r_{22} - \frac{r_{21}}{r_{11}} r_{12} \\ r_{32} - \frac{r_{31}}{r_{11}} r_{12} \\ r_{42} - \frac{r_{41}}{r_{11}} r_{12} \\ r_{52} - \frac{r_{51}}{r_{11}} r_{12} \\ r_{62} - \frac{r_{61}}{r_{11}} r_{12} \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ -0.5 \\ 0.8 \\ 0.2 \end{pmatrix}, \quad \overline{\boldsymbol{b}}_1 = \begin{pmatrix} \overline{b}_1 \\ \overline{b}_2 \\ \overline{b}_3 \\ \overline{b}_4 \\ \overline{b}_5 \end{pmatrix} = \begin{pmatrix} b_2 - \frac{r_{21}}{r_{11}} b_1 \\ b_3 - \frac{r_{31}}{r_{11}} b_1 \\ b_4 - \frac{r_{41}}{r_{11}} b_1 \\ b_5 - \frac{r_{51}}{r_{11}} b_1 \\ b_6 - \frac{r_{61}}{r_{11}} b_1 \end{pmatrix} = \begin{pmatrix} -0.5 \\ -0.5 \\ 0 \\ 1 \\ 0.26 \end{pmatrix} \tag{35}$$

$$\overline{\boldsymbol{R}}_{1,1} \cdot y \leq \overline{\boldsymbol{b}}_1 \tag{36}$$

The reduced derivative matrices can be calculated using Eq. (27) and (28) and the data stored in $\boldsymbol{dR}$ and $\boldsymbol{db}$ matrices, i.e.:

$$\partial_x \overline{b}_5 = \partial_x b_6 - \frac{\partial_x r_{61}}{r_{11}} b_1 = db_{61} - \frac{dr_{61}}{r_{11}} b_1 = -0.8 - \frac{-1}{-1}(-0.5) = -0.3 \tag{37}$$

The entire matrices are:

$$\overline{\boldsymbol{dR}}_{1,1} = \begin{pmatrix} \partial_x \overline{r}_1 & \partial_y \overline{r}_1 \\ \partial_x \overline{r}_2 & \partial_y \overline{r}_2 \\ \partial_x \overline{r}_3 & \partial_y \overline{r}_3 \\ \partial_x \overline{r}_4 & \partial_y \overline{r}_4 \\ \partial_x \overline{r}_5 & \partial_y \overline{r}_5 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -1 \end{pmatrix}, \quad \overline{\boldsymbol{db}}_1 = \begin{pmatrix} \partial_x \overline{b}_1 & \partial_y \overline{b}_1 \\ \partial_x \overline{b}_2 & \partial_y \overline{b}_2 \\ \partial_x \overline{b}_3 & \partial_y \overline{b}_3 \\ \partial_x \overline{b}_4 & \partial_y \overline{b}_4 \\ \partial_x \overline{b}_5 & \partial_y \overline{b}_5 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.3 & -0.8 \end{pmatrix}. \tag{38}$$

So we finally arrive to

$$y \geq 0.5 \tag{39}$$
$$y \geq 0.5 \tag{40}$$
$$y \geq 0 \tag{41}$$
$$y \leq 1.25 \tag{42}$$
$$y \leq 1.3 \tag{43}$$

where $y \geq 0.5$ and $y \leq 1.25$ are the restrictions that limit the length of the segment, 0.75 (see Fig. 9).

Let us call $y_{min} = \frac{\overline{b}_1}{\overline{r}_1} = \frac{-0.5}{-1} = 0.5$ and $y_{max} = \frac{\overline{b}_4}{\overline{r}_4} = \frac{1}{0.5} = 1.25$. Then, the length on the y axis is:

$$L = V'_{it}(d-1, \overline{\boldsymbol{R}}_{i,t}, \overline{\boldsymbol{b}}_t) = y_{max} - y_{min} = 0.75 \tag{44}$$

and its derivative is:

$$\partial_{\boldsymbol{x}_I} L = \partial_{\boldsymbol{x}_I} V'_{it}(d-1, \overline{\boldsymbol{R}}_{i,t}, \overline{\boldsymbol{b}}_t) = \partial_{\boldsymbol{x}_I} y_{max} - \partial_{\boldsymbol{x}_I} y_{min}. \tag{45}$$

The value of $\partial_{\boldsymbol{x}_I} y_{max}$ can be calculated using the chain rule and the $\overline{\boldsymbol{dR}}_{1,1}$ and $\overline{\boldsymbol{db}}_1$ matrices, i.e.:

$$\partial_x y_{max} = \frac{1}{\overline{r}_4}\left(\partial_x \overline{b}_4 - y_{max} \cdot \partial_x \overline{r}_4\right) = 0 \tag{46}$$

In this case $\partial_x L$ and $\partial_y L$ are both zero. In other words, this segment's length (see Fig. 9) does not change with a small change in the position of the evaluation point, P.

Introducing this value in the recursive formula (22) and (26) we obtain:

$$A_{8p} = A_{8p} + \frac{b_1 \cdot L}{d \cdot |r_{11}|} = 0 + \frac{-0.5 \cdot 0.75}{2 \cdot 1} = -0.1875. \tag{47}$$

$$\partial_x A_{8p} = \partial_x A_{8p} + \frac{b_1 \cdot \partial_x L}{d \cdot |r_{11}|} + \frac{L}{d \cdot |r_{11}|}\left(\partial_x b_1 - \frac{b_1}{|r_{11}|}\partial_x |r_{11}|\right) = 0 + \frac{-0.5 \cdot 0}{2 \cdot 1} + \frac{0.75}{2 \cdot 1}\left(0 - \frac{-0.5}{1}\ 0\right) = 0. \tag{48}$$

$$\partial_y A_{8p} = \partial_y A_{8p} + \frac{b_1 \cdot \partial_y L}{d \cdot |r_{11}|} + \frac{L}{d \cdot |r_{11}|}\left(\partial_y b_1 - \frac{b_1}{|r_{11}|}\partial_y |r_{11}|\right) = 0 + \frac{-0.5 \cdot 0}{2 \cdot 1} + \frac{0.75}{2 \cdot 1}\left(0 - \frac{-0.5}{1}\ 0\right) = 0. \tag{49}$$

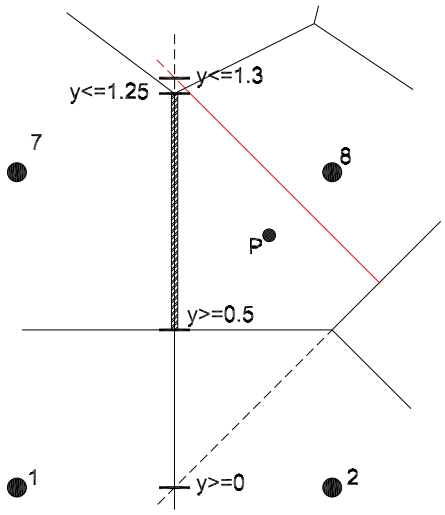


Figure 9. Final length of the first segment in the computation.

This same procedure is then repeated for the m rows of the $\boldsymbol{R}$ matrix. The sum of the $m = 6$ terms in Eq. (22) will give the total area of the second-order Voronoi cell corresponding to node 8 and evaluation point P. The shape function $\phi_8(P)$ can then be obtained as

$$\phi_8(P) = \frac{A_{8P}}{A_T}, \tag{50}$$

where $A_T = \sum_{i=1}^{nnn} A_{iP}$, being nnn the number of natural neighbours of the considered point. The resulting area A_{8P} is depicted in Fig. 8.

Finally, the derivative of the shape function is obtained by applying the chain rule to Eq. (50):

$$\frac{\partial \phi_8(P)}{\partial \boldsymbol{x}_I} = \frac{1}{A_T}\left[\frac{\partial A_{8P}}{\boldsymbol{x}_I} - \phi_8(P)\frac{\partial A_T}{\partial \boldsymbol{x}_I}\right], \text{ with } I = 1, 2, \ldots, d. \tag{51}$$

3. AN EFFICIENT NATURAL NEIGHBOUR-SEARCH ALGORITHM

For the computation of the weak form of the problem some kind of numerical integration of the type

$$\int_\Omega f(\boldsymbol{x}) d\Omega \approx \sum_{i=1}^{nip} \omega_i f(\boldsymbol{x}_i) \tag{52}$$

must be employed, since there is no closed form for the natural neighbour shape functions just described, in general. In Eq. (52) ω_i represent the weights associated to the particular quadrature scheme employed, nip the number of integration points and $\boldsymbol{x}_i$ the associated quadrature points.

It is well-known that the use of three-points (four in three-dimensional cases) numerical quadrature on each triangle (or, respectively, tetrahedra) leads to a numerical integration error in the NEM and it was judged as the cause of the apparent lack of conformity of the NEM with standard patch-tests. In [11] a study was performed on the influence of this error in the results and how to alleviate it. It was demonstrated that, despite the integration error due to the non-polynomial form of the shape functions, the NEM is frequently more accurate than standard, linear triangular FEM.

Whatever method employed to perform the integration, a search for the neighbours of the integration point must be accomplished. This is equivalent to search for all the triangles whose circumcircle contains the point $\boldsymbol{x}$. In that case the three nodes of the triangle will be natural neighbours of the integration point. The naive approach to this problem is to perform an $\mathcal{O}(m^2)$ search (being m the number of nodes in the set, as mentioned in the preceding section), by testing if the evaluation point is a natural neighbour of each node in the set. As will be shown later on, this scheme performs extremely badly, and should be employed only when using very coarse clouds of nodes.

In the original work by Braun and Sambridge [4], the search for natural neighbours was done by employing the *walking triangle* algorithm, due to Lawson [18]. This method is used to search for the triangle containing the evaluation point.

In our approach this search is not necessary since we employ a method that closely resembles the structure of traditional FE codes. Thus, a "natural element" will be composed by a triangle,

which defines, in general, the integration domain, and a number of nodes whose associated shape function's support covers any of the integration points. A similar definition can be given if stabilized conforming nodal integration [6] is used, for instance, instead of traditional, Gauss-based quadratures.

Recently, a new algorithm has been proposed in [5] for the natural neighbour search. In it, the possible neighbour candidates are limited to a squared region of length $2r$ around the given integration point. In this way the search is done in an algorithm of order $\mathcal{O}(n \cdot m)$ where $n < m$. Although in principle valid, this algorithm possesses a main drawback derived from the nature of natural neighbourhood itself. Two points can be neighbours even if they are far away from each other, depending on the relative position of the other nodes. This circumstance, that constitutes one of the main differences between NEM and other meshless methods, complicates the practical application of the before mentioned algorithm, so that the number of possible neighbour candidates n to be considered should be high enough. Otherwise, the risk of eliminating natural neighbours of the given point is always present, and these will be omitted, thus altering the approximation result.

We have developed a new searching algorithm that performs the search in an "expansive" way, starting with the triangle that defines the integration points. We refer in this explanation to two-dimensional settings for simplicity, although the algorithm is extended to three-dimensional settings straightforwardly. Obviously, the Delaunay triangulation of the points must be previously computed, prior to the application of the algorithm. This is done in our case by employing the "Detri" software [20], that has proven to be very fast and efficient. It employs a symbolic perturbation technique to avoid degenerate cases [9]. Nevertheless, this particular choice does not affect the conclusions of the work here presented.

Consider, for instance, a set of nodes whose triangulation gives 11 triangles, as shown in Fig. 10. The algorithm starts with the triangle that defines the integration domain, here labelled as 1. Obviously, the three nodes defining triangle 1 are natural neighbours, by definition, of the three integration points.

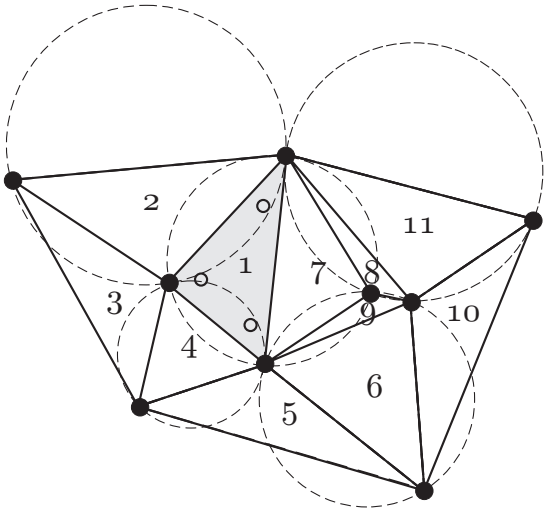


Figure 10. Step 1 in the search algorithm. Integration points are represented by circles.

We then continue (step 2, Fig. 11) by moving through the edges of the starting triangle, say, to triangle number 2. The circumcircle of triangle 2 covers some of the integration points of the element, so we add its nodes to the list of neighbours. Repeat this search iteratively by moving through triangle edges, say to triangle number 3. In this case, the circumcircle does not cover any of the integration points and we can stop the search trough this branch of the tree and come back to triangle number 2.

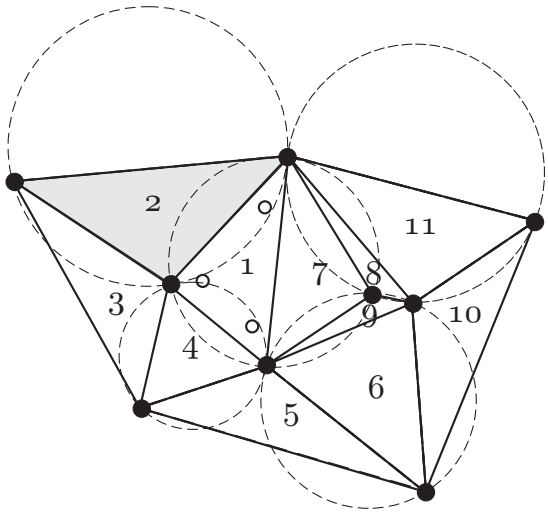


Figure 11. Step 2 in the search algorithm. We move to triangle number 2 and check the neighbourhood.

All the triangles neighbouring triangle number 2 have been now checked, so we move back to triangle number 1 and perform the search again beginning by other, non checked, edge (we move, for instance, to triangle number 4). The process finishes when checking triangle number 7 and its associated triangles.

This iterative-search algorithm provides all element's neighbours, while keeping the number of checked triangles proportional to n, the number of nodes (or, equivalently, triangles) on the cloud. This algorithm can be programmed in the form of a tree, whose pseudo-code is summarized in Algorithm 1.

Algorithm 1. *Natural neighbour search for a given natural element.*

triangle = initial triangle
for *triangles sharing an edge* **do**
  if *any of the integration points is in the circumcircle* **and** *not "checked"*
    then *add the three nodes to the list of neighbours*
    then *label the triangle as "checked"*
    then *triangle = next neighbouring triangle*
  else *come back to previous triangle*
  end if
end for

return

The algorithm runs in approximate constant time for each triangle, since the number of checked neighbouring triangles is nearly constant for each of them. Thus, the total time for the complete neighbour search is $\mathcal{O}(n)$. For this to be true, it is necessary to perform an storage of triangle neighbouring a given one. This can be done in the triangulation process, thus saving computational time. We refer the reader to [20] for further details on the particular data storage algorithm.

Computer savings by employing this algorithm are shown in Fig. 12. Savings reach a relative amount of 4500 times for meshes of around 20000 tetrahedra.

4. NUMERICAL EXAMPLES

4.1. Two-dimensional plate with a hole problem

In order to compare the behaviour of the different approximations previously proposed, we begin by the well-known two-dimensional problem of an infinite plate with a hole under traction. Domain's geometry is shown in Fig. 13.

Analytical solution for this problem exists, and it can be found in many classical books like, for instance, in [28]

$$u_1(r,\theta) = \frac{a}{8\mu}\Big[\frac{r}{a}(\kappa+1)\cos\theta + 2\frac{a}{r}((1+\kappa)\cos\theta + \cos 3\theta) - 2\frac{a^3}{r^3}\cos 3\theta\Big], \tag{53}$$

$$u_2(r,\theta) = \frac{a}{8\mu}\Big[\frac{r}{a}(\kappa-3)\sin\theta + 2\frac{a}{r}((1-\kappa)\sin\theta + \sin 3\theta) - 2\frac{a^3}{r^3}\sin 3\theta\Big], \tag{54}$$

where

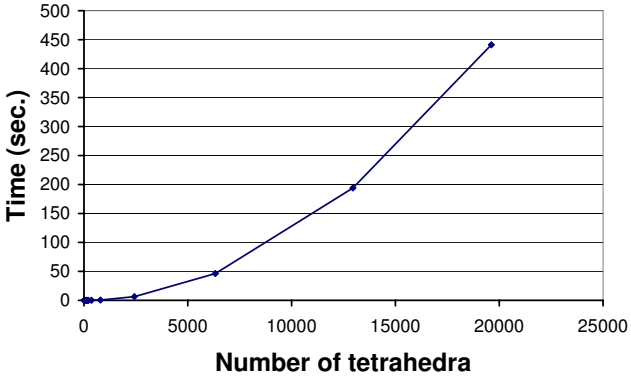
$$\kappa = 3 - 4\nu, \tag{55}$$

$$\kappa = \frac{3-\nu}{1+\nu} \tag{56}$$

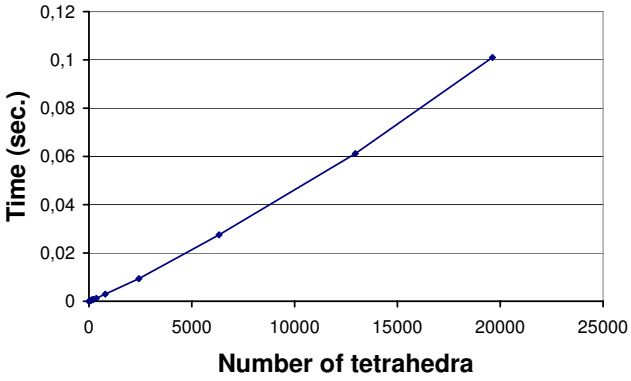
is the Kolosov coefficient for plane strain and stress, respectively. In this case, we have assumed $E = 1.0$ and $\nu = 0.25$. μ represents the shear modulus and a is the radius of the circular hole. The traction applied at infinity, σ_0, was taken unitary.

By imposing appropriate symmetry conditions we model only one quarter of the plate. Analytical tractions are imposed on the boundary of the domain so as to simulate the infinite length of the plate.

This problem has been analyzed by employing Sibson, Laplace, *pseudo*-NEM interpolations, and FEM. Comparative results for the convergence rates in L_2 and H^1 norms are shown in Fig. 14 (a) and (b).



(a)



(b)

Figure 12. Time employed in the neighbour search for a traditional, $\mathcal{O}(n^2)$ algorithm (a) and by the proposed $\mathcal{O}(n)$ algorithm (b).

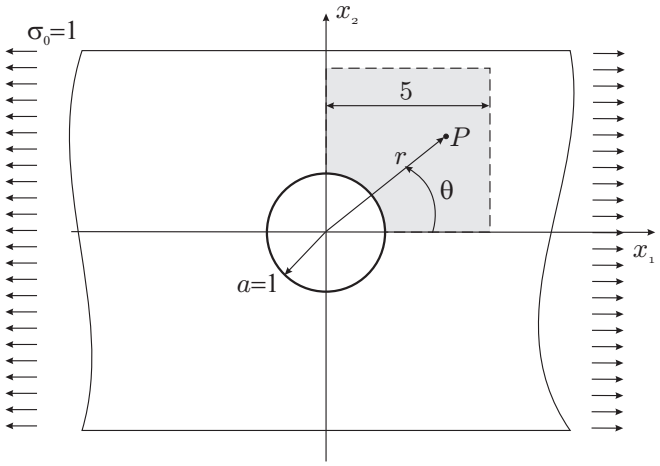


Figure 13. Geometry of the plate with a hole problem.

As expected, all methods posses similar convergence rates, indicated by R. If we analyze the computational cost of each simulation, we conclude that FEM simulations are the fastest ones, without significant differences between all the methods, see Fig. 15.

Two-dimensional results are not very representative, in our opinion, since there exist competitive algorithms for the computation of natural neighbour shape functions (i.e., the before mentioned Watson's algorithm [29]). *Pseudo*-NEM does not appear to be very competitive, but in three-dimensional problems the situation changes drastically, as will be seen in the following section.

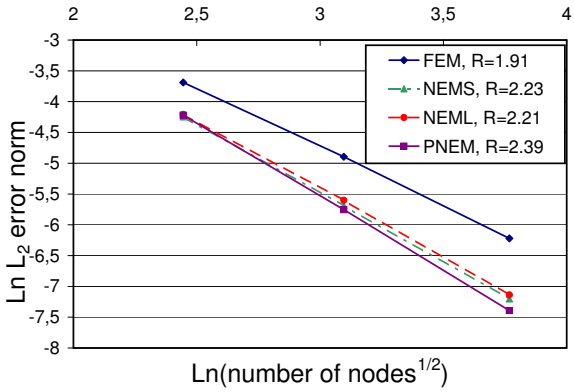
4.2. Three-dimensional linear elastic compression test

To test the performance of the methods in three-dimensional settings, we have chosen a very simple problem of a cubic block under compression, composed by a linear, elastic material. The schematic representation of this problem is shown in Fig. 16

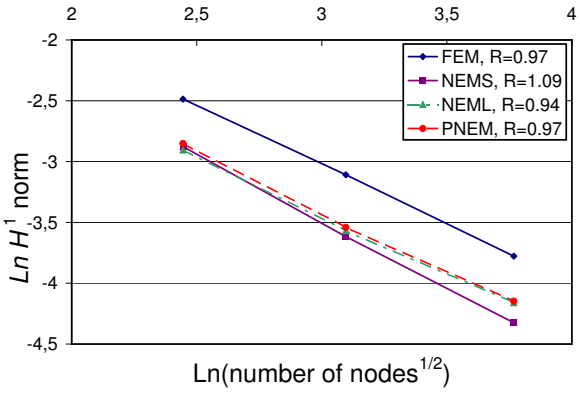
The exact solution of this problem is spanned by each of the four methods, so the analysis of the accuracy will not be performed. Instead, we have focused in the computation time employed in solving the problem, by checking different clouds, both regularly and irregularly distributed over the unit cube. Examples of such clouds are shown in Fig. 17. We checked both regular and irregular meshes since the behaviour of the proposed neighbour-search algorithm could differ in its speed of computation. For regular meshes, the search process is somewhat faster.

The computational costs generated with this problem are analysed in Figs. 18 and 19 for regular and irregular meshes, respectively. We analyze both the total time employed in the calculation, as well as the part of this time employed in the computation of the shape function and its derivatives.

As expected, the use of Sibson interpolation becomes prohibitive in three-dimensional cases. But Laplace and *pseudo*-NEM approaches, despite being more costly than FEM, appear to be competitive. Note that, although employing more computation time, the absence of need of

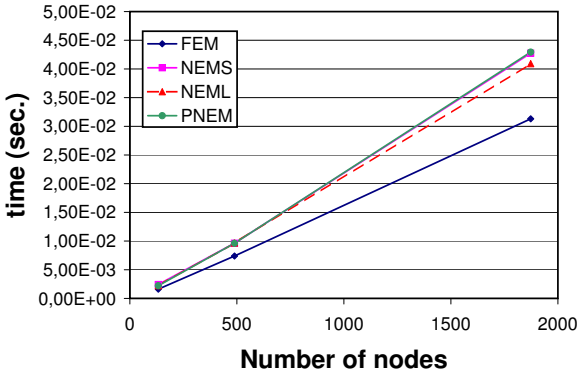


(a)

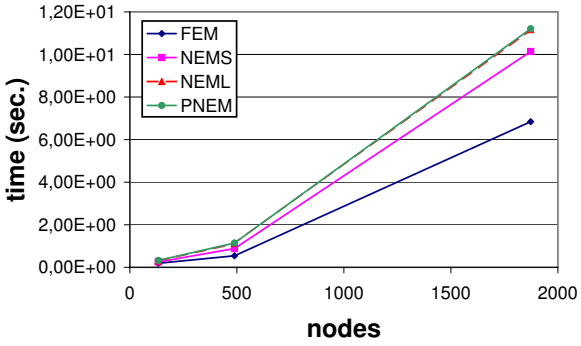


(b)

Figure 14. Convergence of the results for the different methods in the plate with a hole problem. (a) L_2-norm error and (b) H^1-norm error. R denotes the convergence rate found for the different methods.



(a)



(b)

Figure 15. Time employed in the computation of the shape functions and their derivatives (a) and total computation time for the resolution of the plate with a hole problem (b).

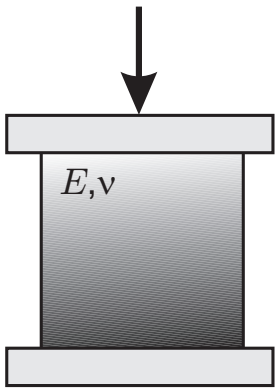
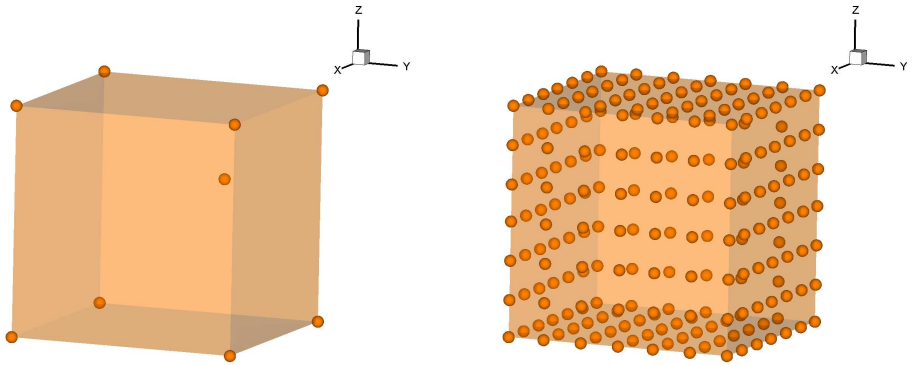


Figure 16. Geometry of the compression test.



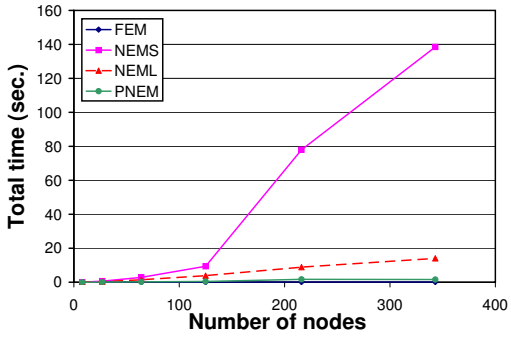
(a) (b)

Figure 17. Different clouds of nodes employed in the simulation of the block under compression.

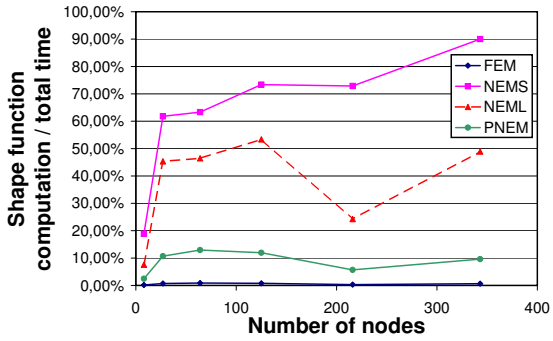
remeshing is not considered here. Especially noteworthy is the fact that nearly ninety per cent of the computation time is employed in the computation of the shape function when using Sibson interpolation.

Results for irregular meshes are similar. Again, Sibson interpolation becomes prohibitive, while Laplace and *pseudo*-NEM approaches result to be competitive. While FEM employs a negligible part of the calculation in computing the shape function, Sibson interpolation constitutes up to ninety per cent of the computing time.

FEM is a competitive method for this kind of problems as has been proved in the last decades. It has revealed to be among the most used numerical techniques both in industry and academia, for obvious reasons. In our opinion, meshless methods should not be used for problems in which FEM has proven to be a very robust technique. The difficulties associated with FEM begin in

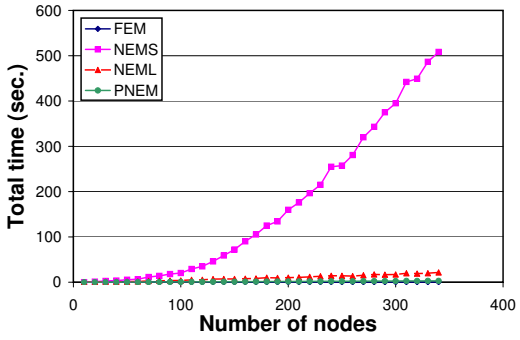


(a)

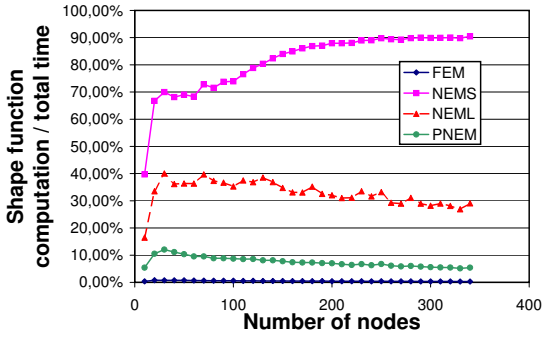


(b)

Figure 18. Total time employed in the compression test (a) and part of this time employed in the computation of the shape function and its derivatives. Regular clouds.



(a)



(b)

Figure 19. Total time employed in the compression test (a) and part of this time employed in the computation of the shape function and its derivatives. Irregular clouds.

problems with large transformations, where remeshing is necessary, especially when transfer of internal variables is necessary from the old meshes to the new ones. This kind of problems is analysed in the following section.

4.3. Extrusion of an aluminium profile

Direct aluminium extrusion is one of the most extended forming processes worldwide. Roughly speaking, it is a process used to produce long profiles by pressing a billet of hot aluminium through a hole with a certain shape. A schematic representation of the process is given in Fig. 20. Obviously, very large strains are present throughout the process. Numerical simulation of extrusion has been studied largely by the FEM, by employing ALE or Eulerian approaches. The interested reader is submitted to the early references by Zienkiewicz and co-workers for examples on this topic [34] [32].

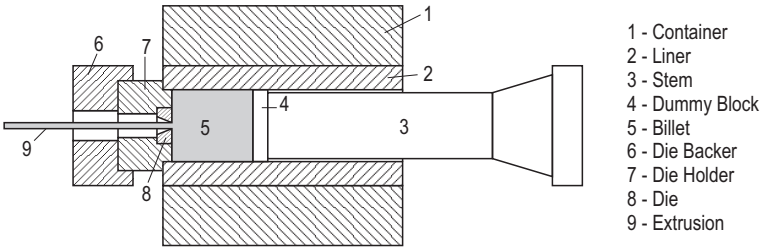


Figure 20. Schematic representation of the extrusion process.

If we neglect elastic deformations when compared to plastic ones, the flow of hot aluminium can be assimilated to that of a non-newtonian fluid. This approach has received the name of "flow formulation" [33]. This is the approach followed here. For an in-deep description of aluminium behaviour and its simulation with NEM, the reader is referred to [1].

We consider the balance of momentum equations, without inertia and mass terms

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} = \mathbf{0}, \tag{57}$$

and the assumed incompressibility of a von Mises-like flow:

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0, \tag{58}$$

where $\boldsymbol{v}$ represents the velocity field. The stress-strain rate relationship is given by Eq. (59):

$$\boldsymbol{\sigma} = 2\mu \boldsymbol{d} - p\boldsymbol{I}, \text{ with } \mu = \frac{\sigma_y}{3\overline{d}}, \tag{59}$$

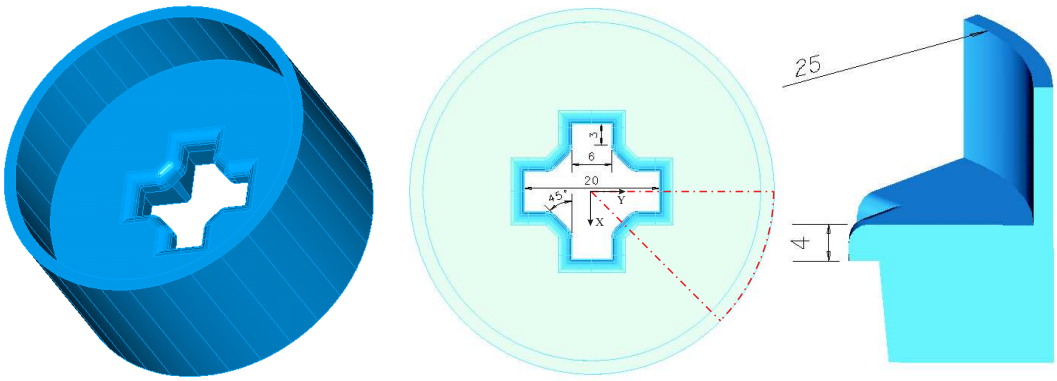
where $\boldsymbol{d}$ represents the strain rate tensor (deviatoric part of the velocity gradient), σ_y represents aluminium's yield stress and $\overline{d}$ represents the so-called "effective strain rate" (second invariant of the strain rate tensor). The dependence of the yield stress on the strain rate is given by the well-known Sellars-Tegart equation:

$$\sigma_y(\overline{d}) = S_m \text{arcsinh}\Big[\big[(\frac{\overline{d}_1}{A})e^{\frac{Q}{RT}}\big]^{\frac{1}{m}}\Big], \tag{60}$$

with $\overline{d}_1 = max\{\overline{d}, \overline{d}_0\}$ and $\overline{d}_0$ an initial threshold in the strain rate.

Parameter	Units	Value
$\overline{d}_0$	(s^{-1})	0.005
S_m	(MPa)	25
m		2.0
A	(s^{-1})	6×10^9
Q	$(\frac{J}{m\rho l})$	1.4×10^5
R	$(\frac{J}{molK})$	8.314

Table II. Material parameters for AA6063 Aluminium alloy.



(a) (b)

Figure 21. Geometry of the die for the extrusion of the cross-shaped profile. (a) perspective view and (b) geometry.

Parameters governing the aluminium yield stress are summarised in Table II.

With this model we performed the simulation of the extrusion of a cross-shaped profile, whose geometry is shown in Fig. 21. Only one half of the geometry was analysed, taking advantage of the symmetry of the domain. The model was composed by 3021 nodes, 15843 tetrahedra and 63372 integration points (four per tetrahedra).

Obviously, large deformations appear during the simulation. For comparison purposes, we ran the simulation using Laplace interpolation. At a given time step, we stopped the simulation and performed the following time step using the four different approximation techniques, namely, finite element, Sibson, Laplace and *pseudo*-NEM approximations. The geometry of the domain at this time step is shown in Fig. 22.

One of the main results of the comparison between the different approximation techniques is the differences found in the results. In Fig. 23 a comparison is made of the small area marked in Fig. 22.

Note the highly distorted mesh appearing in this time step. This produces spurious high levels of strain rate in the FEM solution —in the zone highlighted by a white circle—, far from the extrusion die, due undoubtedly to the distortion of the tetrahedra. At this step, FEM

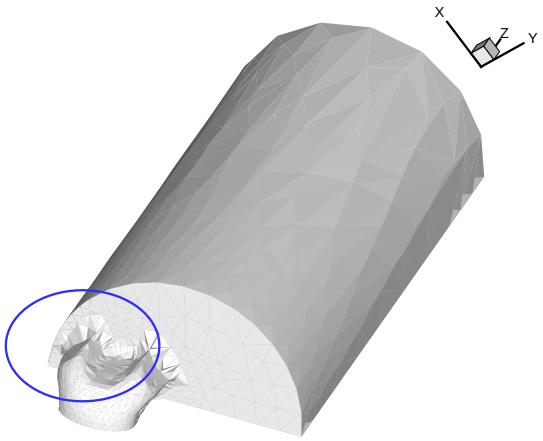


Figure 22. Geometry of the domain for the extrusion simulation test. The ellipse highlights the zone detailed in Fig. 23.

mesh would have necessitated of a remeshing process, with the well-known inherent diffusion of internal variables during the field transfer of internal variables.

While Sibson interpolation predicts slightly higher levels of strain, Sibson, Laplace and P-NEM interpolations have demonstrated to provide good results when compared to experiments (see [10]). The results seem to be insensible to mesh distortion. Note, however, that near the symmetry planes, the number of natural neighbours for a given tetrahedron is low. In these circumstances, Sibson, Laplace, P-NEM and FEM approximations are very similar (note that Sibson tetrahedron with only four neighbours is a CST tetrahedron, see [25]) This could be the reason of the high of level of equivalent strain rate appearing in the four results near the symmetry plane.

The total amount of time employed in the simulations of the before-mentioned time step are summarised in Table III.

	Number iterations	Calc. shape functs.	Total time	Percentage
FEM	13	0.21	1040.01	0.02019
NEM-Sibson	15	15339.81	21979.48	67.79
NEM-Laplace	12	70.3	5580.03	1.260
pseudo-NEM	10	10.99	4674.96	0.2351

Table III. Total amount of time (seconds) employed for the simulation of one time step in the extrusion problem.

These results are even more notorious if we perform the same simulation employing a purely newtonian behaviour. These results are outlined in Fig. 24. In this case the spurious high levels

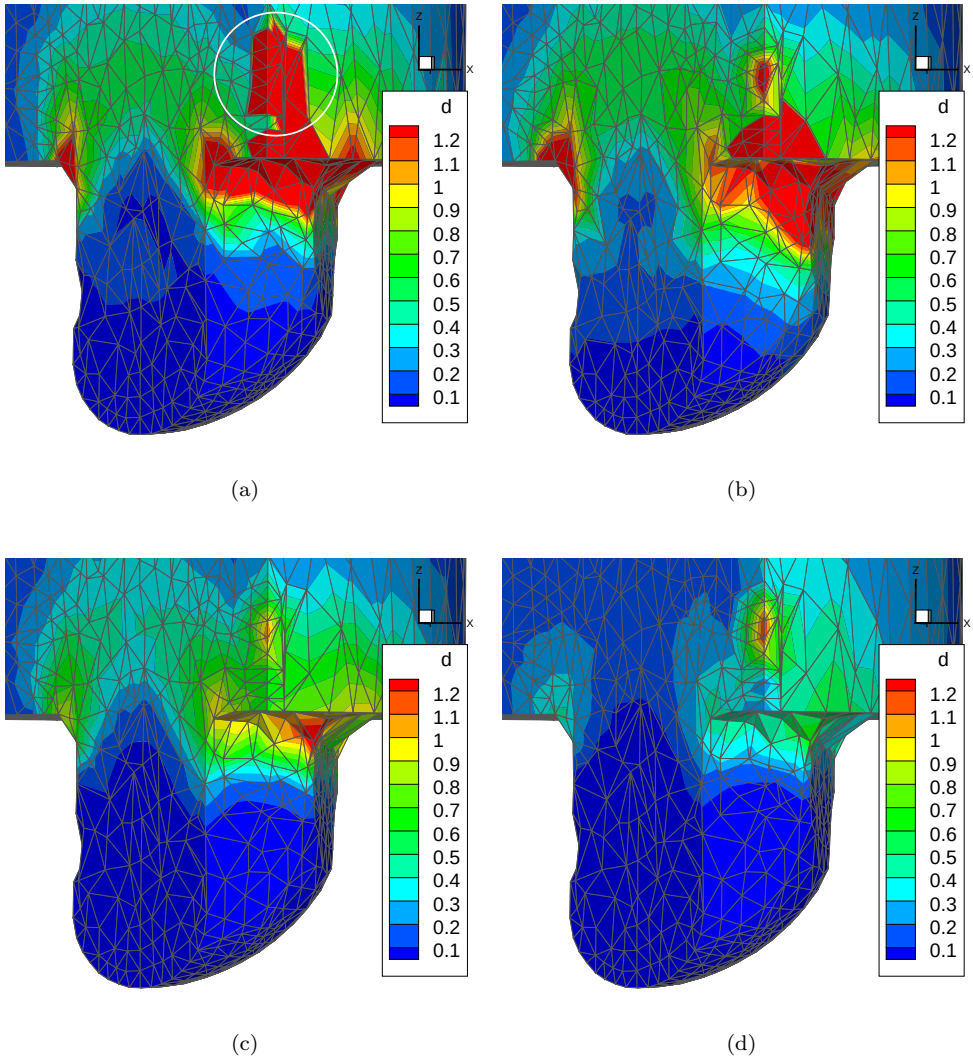


Figure 23. Contour plot of the second invariant of the strain rate tensor at the location indicated in Fig. 22. (a) FEM, (b) Sibson, (c) Laplace and (d) *pseudo*-NEM results.

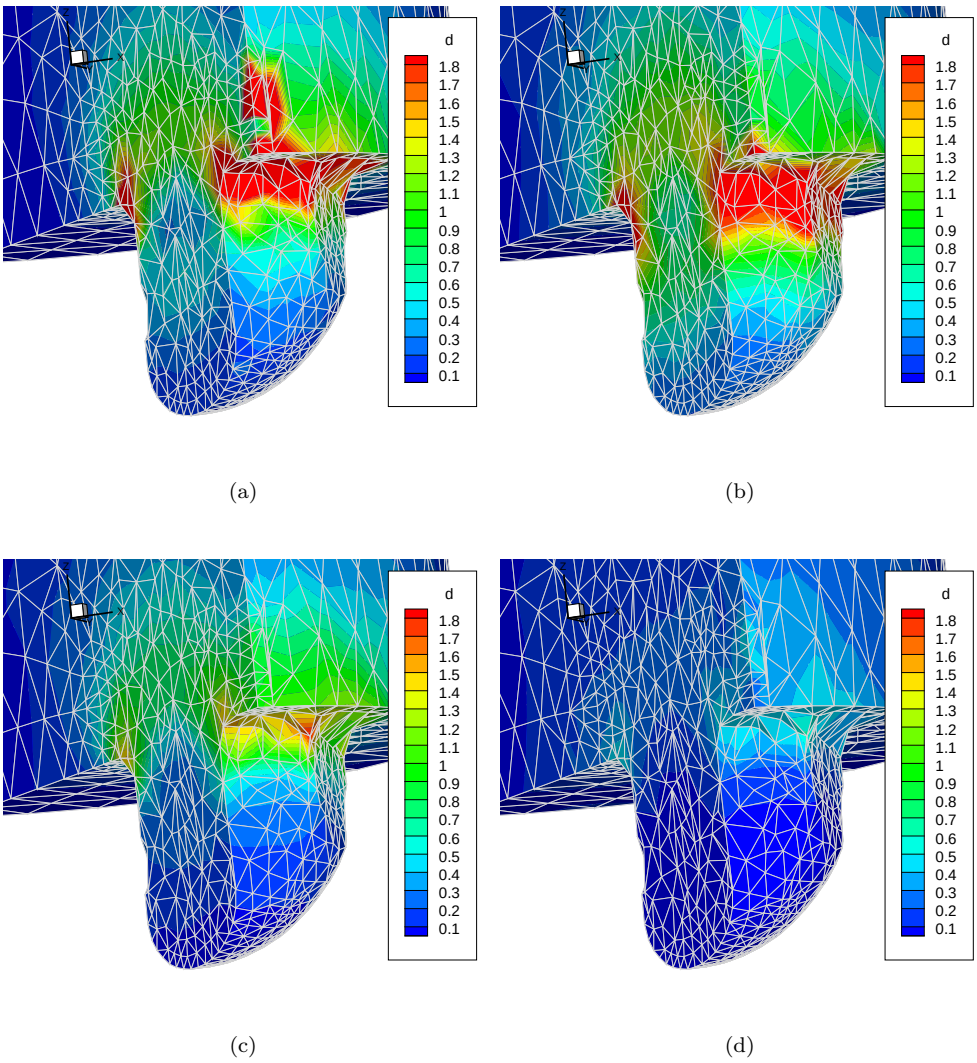


Figure 24. Contour plot of the second invariant of the strain rate tensor at the location indicated in Fig. 22. Newtonian behaviour: (a) FEM, (b) Sibson, (c) Laplace and (d) *pseudo*-NEM results.

of strain rate present in the FEM simulation are even higher.

It is worth noting that the *pseudo*-NEM approach seems to present locking in this case, since virtually no strains are predicted in this step. Maybe by employing richer consistency approximations in the MLS adjustment of the approximation would provide locking-free results, but there is a limited number of nodes in the support of the shape functions and this is not possible everywhere with the formulation presented before.

The stiffness matrices in the NEM are banded and sparse, as in the FEM, but the number of

neighbours of a given "element" is usually much higher than in the FEM. Thus, calculation and inversion of the tangent matrix takes around 450 seconds with the three NEM methods, while in FEM it takes around 80 seconds. Again, Laplace approximation seems to provide a good compromise between accuracy and speed of computation, with no lack of accuracy detected due to mesh distortion. Is in this type of problems where the authors believe that meshless in general, and NEM in particular, approximations can be competitive with FEM simulations. When large transformations in a Lagrangian framework are present in a problem, possibly with internal variables related to history, NEM seems to provide an attractive alternative to more traditional approaches. Note also that in the before presented results, remeshing time is not considered for FEM results.

5. CONCLUSIONS

In this paper we have presented some techniques to improve the computational performance of natural neighbour Galerkin methods. In particular, we have developed a new neighbour-search algorithm that proves to run in $\mathcal{O}(n)$ time with big efficiency.

An in-deep study of the performance of the method has been accomplished, both from the accuracy point of view and from the computational cost one. The P-NEM approach has revealed some very interesting features, especially its efficient computation times, but should be employed, while further analysis is performed, to problems where the LBB condition is not relevant. While Sibson approximation seems to be prohibitive in three-dimensional settings, Laplace interpolation within a Galerkin NEM approach has proven to give an excellent compromise between computational costs (obviously higher than those of the FEM) and efficiency and accuracy, since no need of remeshing is needed.

Thus, the diffusion of internal variables (like those appearing in plasticity, for instance) during to field transfers between meshes is avoided. Despite the highly distorted meshes employed in our experiments, no need of remeshing was encountered.

REFERENCES

1. I. Alfaro, D. Bel, E. Cueto, M. Doblaré, and F. Chinesta. Three-dimensional simulation of aluminium extrusion by the alpha-shape based natural element method. *Computer Methods in Applied Mechanics and Engineering*, 195(33-36):4269–4286, 2006.
2. V. V. Belikov, V. D. Ivanov, V. K. Kontorovich, S. A. Korytnik, and A. Yu. Semenov. The non-Sibsonian interpolation: A new method of interpolation of the values of a function on an arbitrary set of points. *Computational Mathematics and Mathematical Physics*, 37(1):9–15, 1997.
3. T. Belytschko, Y. Y. Lu, and L. Gu. Element-Free Galerkin Methods. *International Journal for Numerical Methods in Engineering*, 37:229–256, 1994.
4. J. Braun and M. Sambridge. A numerical method for solving partial differential equations on highly irregular evolving grids. *Nature*, 376:655–660, 1995.
5. Yongchang Cai and Hehua Zhu. A local search algorithm for natural neighbours in the natural element method. *International Journal of Solids and Structures*, 42:6059–6070, 2005.
6. Jiun-Shyan Chen, Cheng-Tang Wu, Sangpil Yoon, and Yang You. A stabilized conforming nodal integration for galerkin mesh-free methods. *International Journal for Numerical Methods in Engineering*, 50:435–466, 2001.
7. E. Cueto, M. Doblaré, and L. Gracia. Imposing essential boundary conditions in the Natural Element Method by means of density-scaled α-shapes. *International Journal for Numerical Methods in Engineering*, 49-4:519–546, 2000.

8. E. Cueto, N. Sukumar, B. Calvo, M. A. Martínez, J. Cegoñino, and M. Doblaré. Overview and recent advances in Natural Neighbour Galerkin methods. *Archives of Computational Methods in Engineering*, 10(4):307–384, 2003.
9. H. Edelsbrunner and E. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.
10. L. Filice, I. Alfaro, F. Gagliardi, E. Cueto, F. Micari, and F. Chinesta. Cross extrusion with asymmetric die: a first preliminary comparison between finite element and meshless formulation predictions. In N. Juster and A. Rosochowsky, editors, *ESAFORM 2006 Conference proceedings*, pages 79–82. European Scientific Association for Material Forming, 2006.
11. D. Gonzalez, E. Cueto, M. A. Martinez, and M. Doblare. Numerical integration in Natural Neighbour Galerkin methods. *International Journal for Numerical Methods in Engineering*, 60(12):2077–2104, 2004.
12. D. González, E. Cueto, and M. Doblaré. Volumetric locking in Natural Neighbour Galerkin methods. *International Journal for Numerical Methods in Engineering*, 61(4):611–632, 2004.
13. H. Hiyoshi. A numerical comparison of the natural neighbor interpolation and the finite element method. In K. Sugihara, editor, *International Symposium on Voronoi diagrams in Science and Engineering*, pages 1–10. University of Tokio, 2004.
14. H. Hiyoshi and K. Sugihara. Two generalizations of an interpolant based on Voronoi diagrams. *International Journal of Shape Modeling*, 5(2):219–231, 1999.
15. S. R. Idelsohn and E. Oñate. To mesh or not to mesh: that is the question. *Computer Methods in Applied Mechanics and Engineering*, in press, available at www.sciencedirect.com, 2006.
16. S. R. Idelsohn, E. Oñate, N. Calvo, and F. del Pin. The meshless finite element method. *International Journal for Numerical Methods in Engineering*, 58:893–912, 2003.
17. J. B. Lasserre. An analytical expression and an algorithm for the volume of a convex polyhedron in $\mathbb{R}^n$. *Journal of Optimization Theory and Applications*, 39(3):363–377, 1983.
18. C. L. Lawson. Software for C^1 surface interpolation. In *J. R. Rice (Ed.) Mathematical Software III. Vol. 3*. Academic Press, N.Y., 1977.
19. W. K. Liu, S. Jun, S. Li, J. Adee, and T. Belytschko. Reproducing kernel particle methods. *International Journal for Numerical Methods in Engineering*, 38:1655–1679, 1995.
20. E. P. Muecke. *Shapes and Implementations in Three-Dimensional Geometry*. PhD thesis, University of Illinois at Urbana-Champaign, 1993.
21. B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics*, 10:307–318, 1992.
22. P. Schembri, D.L. Crane, and J. N. Reddy. A three-dimensional computational procedure for reproducing meshless methods and the finite element method . *International Journal for Numerical Methods in Engineering*, 61(6):896–927, 2004.
23. R. Sibson. A Vector Identity for the Dirichlet Tesselation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 87:151–155, 1980.
24. R. Sibson. A brief description of natural neighbour interpolation. In *Interpreting Multivariate Data. V. Barnett (Editor)*, pages 21–36. John Wiley, 1981.
25. N. Sukumar, B. Moran, and T. Belytschko. The Natural Element Method in Solid Mechanics. *International Journal for Numerical Methods in Engineering*, 43(5):839–887, 1998.
26. N. Sukumar, B. Moran, A. Yu Semenov, and V. V. Belikov. Natural Neighbor Galerkin Methods. *International Journal for Numerical Methods in Engineering*, 50(1):1–27, 2001.
27. A. H. Thiessen. Precipitation averages for large areas. *Monthly Weather Report*, 39:1082–1084, 1911.
28. S. Timoshenko and J. N. Goodier. *Teoría de la Elasticidad*. Editorial Urmo, Spain, 1972.
29. D. Watson. *Nngridr. An Implementation of Natural Neighbor Interpolation*. Published by the author, 1994.
30. J. Yvonnet and F. Chinesta. A hybrid element-free Galerkin and natural element meshfree method for direct imposition of boundary conditions and faster three-dimensional computations. In K. J. Bathe, editor, *Proceedings of the Third MIT Conference on Computational Fluid and Solid Mechanics*, pages 861–890, 2005.
31. J. Yvonnet, D. Ryckelynck, P. Lorong, and F. Chinesta. A new extension of the Natural Element method for non-convex and discontnuous problems: the Constrained Natural Element method. *International Journal for Numerical Methods in Enginering*, 60(8):1452–1474, 2004.
32. O. C. Zienkiewicz and P. N. Godbolet. Flow of plastic and visco-plastic solids with special reference to extrusion and forming processes. *International Journal for Numerical Methods in Engineering*, 8:3–16, 1974.
33. O. C. Zienkiewicz, E. Oñate, and J. C. Heinrich. Plastic flow in metal forming. (I) Coupled thermal (II) Thin sheet forming. In *Applications of Numerical Methods to Forming Processes. AMD-vol 28*, pages 107–120, 1978.
34. O. C. Zienkiewicz, P. C. Pain, and E. Oñate. Flow of solids during forming and extrusion: some aspects

of numerical solutions. *International Journal of Solids and Structures*, 14:15–38, 1978.