Available online at www.sciencedirect.com
ScienceDirect
Advances in Engineering Software 39 (2008) 156–166
ADVANCES IN ENGINEERING SOFTWARE
www.elsevier.com/locate/advengsoft

Adaptive methodology for meshless finite point method

F. Perazzo [a,*], R. Löhner [b], L. Perez-Pozo [a]

[a] Department of Mechanical Engineering, Universidad Técnica Federico Santa María, Avenida España 1680, Valparaíso, Chile
[b] School of Computational Science and Informatics, George Mason University, M.S. 4C7, Fairfax, VA 22030-4444, USA

Received 9 September 2006; received in revised form 13 February 2007; accepted 15 February 2007
Available online 27 April 2007

Abstract

In this work, a posteriori error estimator and an adaptive refinement process for the meshless finite point method (FPM), which is based on point collocation, are presented. The error indicator is formulated by the least-squares functional evaluation, used in the shape function development. New degrees of freedom or additional points can be incorporated without difficulty, in zones where the error estimator presents a high value, by means of *h–p* refinement processes. The validity of the proposed error estimator can be demonstrated by developments of numerical problems in mechanics of solids, using an adaptive refinement process of the solution.

Keywords: Meshless; Point collocation; Finite point method; Adaptivity; Error estimation

1. Introduction

In the last decades, the development of numerical meshless methods has increased, as much in the theoretical formulation as in the computational implementation. The finite point method (FPM) has been developed like a numerical technique, due to the investigations from several authors [1–6].

In a weighting least-squares approximation, like a FPM, the first step is the definition of sub-domains interpolation. The correct definition of these sub-domains, called also "clouds" is very important for the numerical results of the method, reason why it must develop clouds of point generation technique. In this investigation, local cloud of point generation technique is presented. This technique is based on the proposed by Löhner et al. [5].

Also it is possible to determinate the error of the approximation in the FPM context. This error can be calculated or estimated by means of many techniques. Various investigation groups have been dedicated to develop routines of error estimation in meshless methods. In "meshless particles methods" Rabczuk et al. [7] have developed an error estimator based on the superior derivatives evaluation, with the result of adaptive refinement. On the other hand Kim et al. [8] have developed an error estimation technique in the meshless local Petrov-Galerkin method (MLPG), which incorporates "secondary nodes" in the original discrete model, without any change in the interpolation sub-domains used. This allows an adequate error control and in addition geometrical concavities are possible to be analyzed. Finally, Park et al. [9] use a least-squares approximation, where in order to add points in interesting areas, Voronoi cells are used. From these investigations it is deduced that the estimated error is used to redistribute the discretization or to insert nodes in specific zones, being based on a particular criterion. Considering the FPM consistency, both for regular and irregular discretization, it is possible to add new points in relevant zones, independent of the already existing ones, or to redistribute them, fixing in this way the degrees of freedom.

In this work, a posteriori error estimator based on a least-squares functional will be analyzed in point collocation context. This method uses the differences between nodals contributions and the calculated values after the approximation obtained by point collocation. After the error estimation is evaluated, an adaptive refinement is

* Corresponding author.
E-mail addresses: franco.perazzo@usm.cl (F. Perazzo), rlohner@gmu.edu (R. Löhner), luis.perez@usm.cl (L. Perez-Pozo).

0965-9978/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2007.02.007

formulated, based on the results from the error estimate process, i.e., it considers aspects like: geometry optimal points distribution, places where to insert new points and insertion technique for them. The present development considers a geometrical refinement techniques based on *h*- and *p*-refinement, widely used in the finite element method, but in the context of FPM.

Finally, in order to verify the correct analysis and development of the proposed error estimation techniques, a numerical solid mechanics example is presented.

2. Fixed weighting least-squares (FWLS) approximations

In order to define the parameters that will be used in this paper, general aspects of FPM approximation will be specified.

Let Ω_I be the interpolation sub-domain or cloud from a $u(\mathbf{x})$ function, and s_j with $j = 1,2,\ldots,n$ a collection of n points with coordinates $\mathbf{x}_j \in \Omega_I$. The subscript I in the expressions means a point where the approximation will have to evaluated. This point is called "star node". The unknown function $u(\mathbf{x})$ can be approximated into the Ω_I by

$$u(\mathbf{x}) \cong \hat{u}(\mathbf{x}) = \sum_{l=1}^{m} p_l(\mathbf{x})\alpha_l = \mathbf{p}^{\mathrm{T}}(\mathbf{x})\boldsymbol{\alpha} \quad \forall \mathbf{x}_I \in \Omega,\ \forall \mathbf{x} \in \Omega_I, \tag{1}$$

where $\boldsymbol{\alpha}^{\mathrm{T}} = [\,\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_m\,]$ and the $\mathbf{p}(\mathbf{x})$ vector, called "interpolation base", contains typically monomials. For 2D problems, the following can be used:

$$\boldsymbol{p}_1 = [1,x,y]^{\mathrm{T}} \quad \text{for } m = 3, \tag{2a}$$

$$\boldsymbol{p}_2 = [1,x,y,x^2,xy,y^2]^{\mathrm{T}} \quad \text{for } m = 6. \tag{2b}$$

The unknown function $u(\mathbf{x})$ can be evaluated in the n points from the cloud, obtaining

$$\mathbf{u}^h = \begin{Bmatrix} u_1^h \\ u_2^h \\ \vdots \\ u_n^h \end{Bmatrix} \cong \begin{Bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_n \end{Bmatrix} = \begin{Bmatrix} \mathbf{p}_1^{\mathrm{T}} \\ \mathbf{p}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{p}_n^{\mathrm{T}} \end{Bmatrix} \boldsymbol{\alpha} = \boldsymbol{C}\boldsymbol{\alpha}, \tag{3}$$

where $u_j^h = u(\mathbf{x}_j)$ are the unknowns values.

Using the previous nomenclature, the fixed weighting least-squares approximation (FWLS) is obtained minimizing the following functional:

$$\begin{aligned} \boldsymbol{J}_I &= \sum_{j=1}^{n} w(\mathbf{x}_I - \mathbf{x}_j)(u_j^h - \hat{u}(\mathbf{x}_j))^2 \\ &= \sum_{j=1}^{n} w(x_I - x_j)(u_j^h - p_j^{\mathrm{T}} \cdot \alpha)^2. \end{aligned} \tag{4}$$

The weighting function $w(x_I - x_j)$ evaluated in the star node takes the unit value, and will be decreasing in far distances. Out of the sub-domain, the function w is null. In this work, the Gauss function is used to weigh the approximation error. In Fig. 1, the FWLS approximation is showed with a fix weighting function w. More antecedents respect to this and other weighted functions used in meshless methods are presented in [10].

When the J_I functional is minimized respect to α, the following is obtained:

$$\alpha = A_I^{-1} B_I u^h, \tag{5}$$

where

$$A_I = P^{\mathrm{T}}(x_I) W_I P(x_I) \wedge B_I = P^{\mathrm{T}}(x_I) W_I, \tag{6}$$

where P and W are expressed in vectorial form.

The final FPM approximation is obtained replacing (5) into (1)

$$u(x) \cong \hat{u}(x) = p^{\mathrm{T}}(x) A_I^{-1} B_I u^h. \tag{7}$$

If all approximations based on "the local frame" are evaluated by shifting the coordinate origin to the point I, any

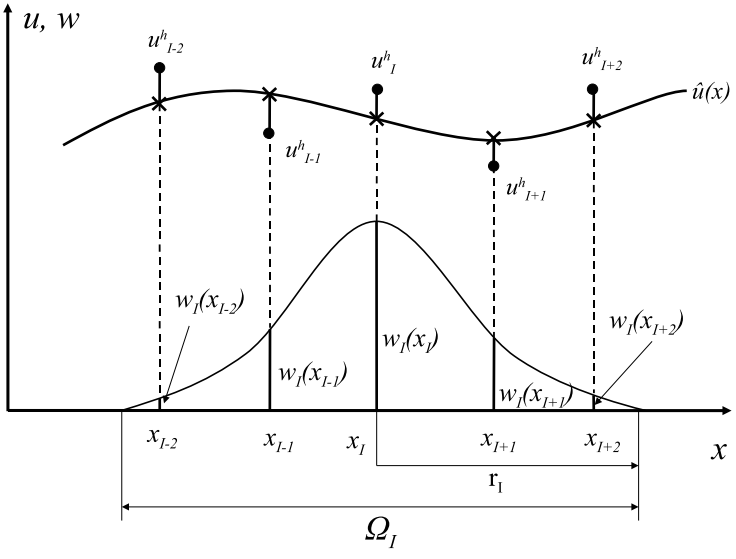

Fig. 1. FWLS procedure.

value or derivative can be obtained quickly from α. In particular,

$$\hat{u}_I = \alpha_1, \tag{8}$$

$$\nabla\hat{u}_I = (\alpha_2, \alpha_3, \alpha_4), \tag{9}$$

and

$$\nabla^2\hat{u}_I = 2 * (\alpha_5 + \alpha_8 + \alpha_{10}). \tag{10}$$

These expressions can also be written as

$$\hat{u}|_I = C^{1j}u_j^h, \tag{11}$$

$$\left.\frac{\partial\hat{u}}{\partial x_l}\right|_I = D_l^{ij}u_j^h, \tag{12}$$

where $D_l^{ij} = C^{qj}$ and $q = l + 1$, and

$$\nabla^2\hat{u}_I = L^{ij}u_j^h, \quad L^{ij} = 2 * (C^{5j} + C^{8j} + C^{10j}). \tag{13}$$

3. Local clouds generation

In all numerical meshless method, three processes are defined: interpolation or approximation process, weighting process and function discretization. From [10] can be seen that the defined processes determines the relevance of the sub-domains or "clouds" generation. By this, geometrical task of cloud generation determines the approximation quality, into this numerical analysis in particular.

After the geometry discretization, both for the boundary as well as for the domain, the clouds of points will be generated.

It is possible to summarize the concept by the following manner:

- Discretization, geometry list of points.
- For each node I (star node) the respective cloud is generated, that is conformed by itself and the closest points or the points that more information give to this cloud. This cloud is stored in a data file.
- Then, the previous step is repeated for the all points, both for the boundary and the domain.

The investigators dedicated to the meshless methods, have proposed several geometrical techniques for the clouds generation [18–20]. This investigation, however, uses the methodology presented by Löhner et al. [12].

Firstly an outline of the technique is presented. The input required the list of points with their respective coordinates and the list of triangles that define the outer boundaries domain.

Do: For each point ipoin
Initialize the search region around ipoin
While: not enough close points ($30 < n_c < 120$):
Enlarge the search region;
Obtain the points in the search region;
Obtain the boundary faces in the region;
Remove, from the list of close faces, those that can not see ipoin;
Remove, from the list of close points, those whose ray ipoin:jpoin intersect a face;
End while
Produce a Delaunay grid with the local points;
Initialize the local cloud list with the first layer of the nearest neighbours;
If the local cloud of points is acceptable: exit;
Do: For all points, according to layers
Add a further point to the local cloud;
If the local cloud of points is acceptable: exit;
End do
As no proper local cloud was found: increase the search region;
End do
With the surface triangulation: Correlate boundary points/faces with the points of the global cloud in order to apply boundary conditions.

In what follows, we describe in more detail the techniques and parameters used in each one of these steps.

3.1. Search for close points

The search for close points is performed using an octree [11,12,14]. Before generating any local clouds, all points are placed in an octree. Whenever a search for close points in the vicinity of *ipoin* is required, a small search region is placed around *ipoin*. The octree is then queried for all points in this search region. This takes approximately $O(\log_8(N_p))$ operations. If the number of close points found is too small, the search region is enlarged by 30%. Conversely, if too many points were found, the search region is reduced by 15%. This procedure is repeated until an acceptable number of close points has been found.

3.2. Search for close faces

The search for close faces is performed using a modified octree that stores faces. The bounding box for each face is first determined. The faces are then placed in the octants as if they were points, marking all octants covered. Given that the bounding boxes of faces can overlap, it may happen that the bounding boxes of more than eight faces can share the same point. In this case, the classic octree would divide ad infinitum. Therefore, only two subsubdivisions are allowed when introducing a new face to the octree, and a provision is made to allow the storage of more than eight faces per octant. Given the search region used for the points, all faces whose bounding boxes fall into this region are retrieved from the modified octree. This takes approximately $O(\log_8(N_f))$ operations. Repeated faces are then removed using hashing techniques [13].

3.3. Filtering close faces

The search for close faces may yield some that are not related to the point whose local cloud is to be found. A

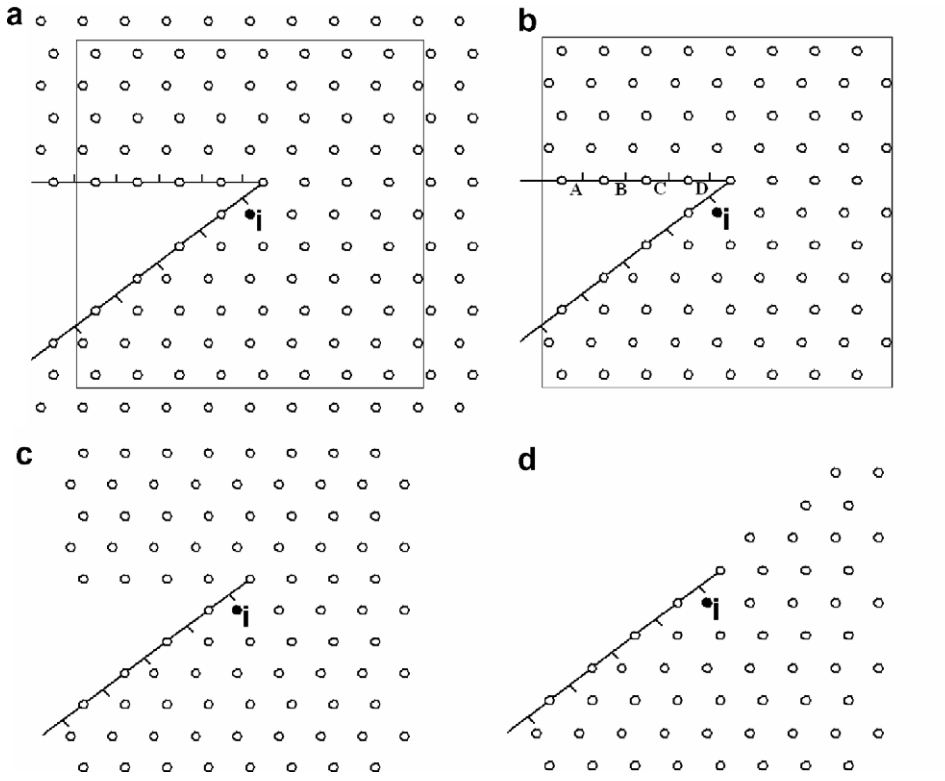

Fig. 2. Search for close points: (a) obtain search region, (b) retain relevant points and faces, (c) remove faces that can not see *point i*, and (d) remove points with rays crossing faces.

typical case is shown in Fig. 2a, where faces A, B, C, D clearly do not belong to the set of faces associated with ipoin. These faces can not "see" *ipoin*, and this observation can be used to remove them. One simply computes the normal distance of *ipoin* to this face, and, if negative, removes the faces from the list.

3.4. Filtering close points

The search for close points can yield some that are on the "wrong" side of a boundary, as shown in Fig. 2, for the tail of a wing. This situation will happen frequently for sharp corners, multimaterial applications, and in general for complex geometries with coarse clouds of points. The close faces obtained previously can be used to filter the points further. The points on the "wrong" side of a boundary will have to pierce through one of the boundary faces. Therefore, any point *jpoin* whose ray *jpoin:ipoin* intersects one of the close faces is removed from the list.

3.5. Delaunay meshing

Given a list of close points, there are many possible ways of obtaining local clouds. Among them, the Delaunay technique will produce a graph of nearest neighbours with optimal properties for finite elements and elliptic partial differential equations (PDE). This therefore is a plausible technique for the present context. We outline the main steps, and refer the reader to George's monogram [15] for details.

Place a large tetrahedron (or box with 5/6 tetrahedra) around the point to be grided;

Do: for all close points:
Find the element(s) x_i fall into;
Obtain all elements whose circumsphere encompasses x_i;
Remove from the list of elements all those that would not form a proper element (volume, angles) with x_i; this result in a properly constrained convex hull;
Reconnect the outer faces of the convex hull with x_i to form new elements;
End do

Retain only the elements with all nodes belonging to the list of close points.

The basic procedure has been sketched in Fig. 3.

Given this tetrahedral mesh of close points, the graph of nearest neighbours can be constructed.

3.6. Change of polynomial order for clouds for boundary points

The application of Neumann conditions typically requires derivatives that are of lower order than those required by the PDE to be solved in the domain. It is

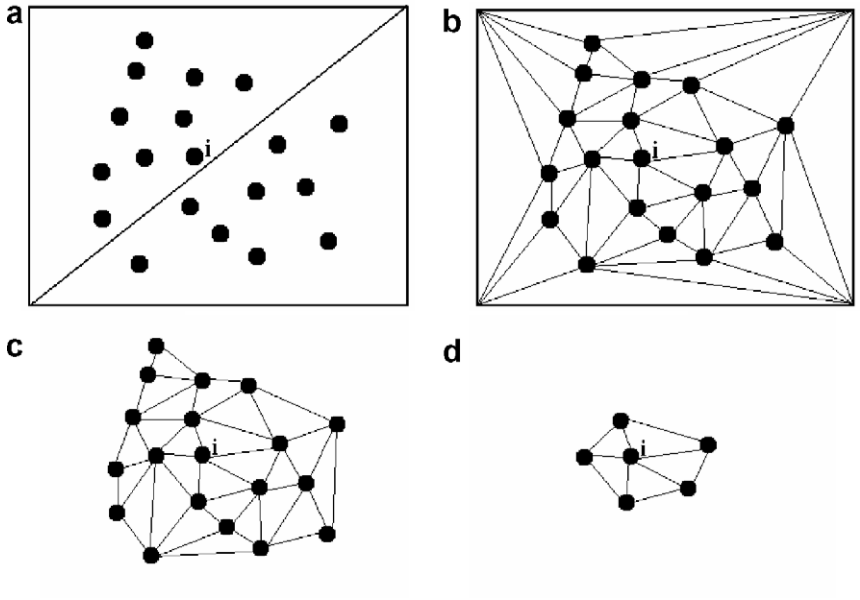

Fig. 3. Formation of local cloud: (a) place points in macro-triangles, (b) Delaunay-mesh of local clouds, (c) retain elements of original points, and (d) retain first layer of nearest neighbours.

therefore advantageous (requiring less storage and CPU) to reduce the polynomial order of the local clouds for these boundary points. As we need to identify the closeness of boundary faces in any case for the proper generation of local clouds (see above), we have the information as to whether a point is on the boundary or not. If so desired by the user, the polynomial order of these points is reduced accordingly.

3.7. Trimming of clouds for boundary points

The application of Neumann conditions typically requires gradients in the direction normal to the wall. It is found that for these cases it is best to assign very low weights, or to remove completely from the local cloud, those points that are co-planar with the points whose local cloud is being formed. This can be done in a straightforward way by identifying these points and assigning to them a high "local neighbour" value (see Fig. 4). In this way, the co-planar points are used for the local Delaunay mesh, yielding the desired result. Removing them before performing the local Delaunay mesh does not yield a proper list of near neighbours, as evidenced in Fig. 4b.

If several faces are present (see Fig. 5), a point is tested against all faces. If the maximum normal distance falls below a tolerance (typically 5% of the local side length), the point is marked. Otherwise, the point is treated as if it was a domain point.

3.8. Acceptable cloud criteria

The local clouds produced by the procedure outlined above will not always be useful for finite point methods. The main reason is that a local cloud can yield a singular approximation matrix A_I (5). For this reason, several tests are carried out for every local cloud. Before carrying out the tests, the coordinates of the local cloud are "non-dimensionalized" as follows:

$$x' = \frac{(x - x_0)}{\delta}, \tag{14}$$

where x_0 are the coordinates of the point whole local cloud is sought, and δ a representative distance between points. Using the minimum distance between points for δ, an average, or the maximum distance does not change the resulting clouds.

The first test is to see if the matrix A_I will be singular. The most obvious indication that A_I is singular, or not adequate, will occur during inversion. However, we have found cases where the local cloud is not suitable for

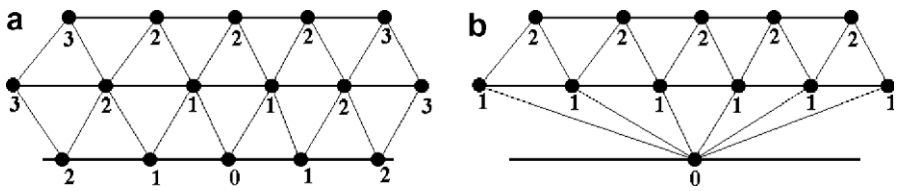

Fig. 4. Trimming of clouds for boundary points: (a) marking of layers and (b) pre-removal.

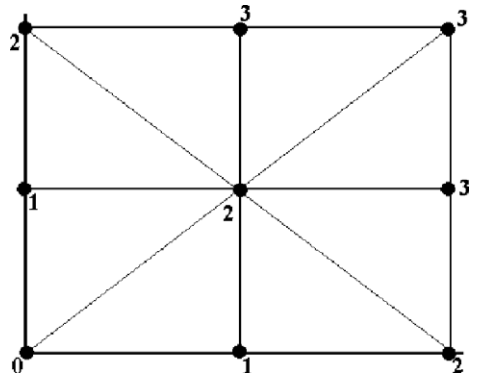

Fig. 5. Points at corners.

FPM although the inverse exists and $A \cdot A^{-1} \approx 1$. We therefore also test for the largest (absolute) entry in A^{-1}. If this value exceeds a tolerance (e.g. 10^6), the local cloud is rejected.

The second test is to take a known function, and see how much the derivatives deviate from the exact values. For the first derivatives, we take $u = x + y$, which should yield a gradient of $\Delta u = (1, 1)$. For the second derivatives (in case we have a polynomial of order $p > 1$), we take $u = x^2 + y^2$, which should yield a Laplacian of $\Delta^2 u = 4$. If the values obtained via local cloud approximation and the exact values deviate by more than 10^{-10}, the local cloud is rejected.

3.9. Test for Neumann boundary conditions

The application of Neumann boundary conditions requires that the coefficients corresponding to the first derivatives should be significant. From (12) this implies that

$$|C^{q1}| > c_t \max(|C^{qI}|), \quad I = 2, \ldots, n, \tag{15}$$

where $q = 2; 3; 4$ (i.e. each dimension), n is the number of points in the local cloud and the tolerance ct is typically of order $ct = \mathrm{O}(0.01)$. It was found that not applying this acceptance criterion could lead to clouds that pass all the criteria described above in Section 3.8, but are numerically unstable.

3.10. Approximations order of local cloud

If one consider the edges of a typical tetrahedral mesh and use these to obtain local clouds, would have a determinated number of points in each local cloud. For a Cartesian distribution of points, the nearest neighbours would yield local clouds with 27 points in 3D case.

From Fig. 2a–d one can infer that the smallest number of points in the local cloud required to achieve a linear function, a quadratic function and a cubic function is 3, 6 and 10 respectively. A considerable percentage of the local clouds obtained from the first layer of Delaunay-neighbours will allow for these higher-order approximations. Therefore, a test is carried out as before for the A-matrices and the derivatives of a known function. If these tests yield a better result than the quadratic approximation, the higher-order approximation is retained.

3.11. Symmetrization of local cloud

From an intuitive argument of symmetry, it was thought that the symmetry of local clouds (i.e. if point j belongs to the local cloud of point i, then point i should belong to the local cloud of point j) could provide a better numerical approximation. Therefore, options were implemented to symmetrize all clouds, or only the clouds of points that are not on the boundary. This latter option was motivated by the fact that some of the boundary clouds can exhibit up to two levels of points inside the domain.

4. Adaptive refinement

Automatic adaptive refinement procedures have shown great benefits in all areas of computational mechanics. They achieve the highest possible accuracy for a given number of degrees of freedom, and alleviate the burden of constructing a near-optimal mesh of point cloud by the user. Any adaptive refinement procedure is composed of three main ingredients:

- an optimal-mesh criterion,
- an error indicator, and
- an algorithm or strategy to refine and coarsen the mesh.

They give answers to the questions

- How should the optimal mesh be defined?
- Where is refinement/coarsening required? and
- How should the refinement/coarsening be accomplished?

The topic of adaptation being now three decades old, it is not surprising that a variety of answers have been proposed by several authors for each of these questions. In the following, we consider the equivalent of h- and p-refinement in the FPM context.

4.1. Error estimation/indication

A large number of error estimators/indicators have been proposed in the literature [16]. The FPM offers some unique possibilities, due to its approximation basis. The most obvious one is to consider the difference between the point value of u_I^h and the approximation value obtained after least-squares approximation. From (11) we have

$$\hat{e}_I = |\hat{u}|_I - u_I^h| = |C^{1j} u_j^h - u_I^h|. \tag{16}$$

It is intuitively clear that regions where this error indicator is high require an increase in the degrees of freedom, i.e. adaptation. On the other hand, one can also use interpolation error indicators on the edges (local clouds) by comparing the difference of gradients:

$$\hat{e}_{Ij} = |x_{Ij} \cdot (\nabla u_I^h + \nabla u_j^h)|, \tag{17}$$

where $x_{Ij} = x_I - x_j$.

From the above, one can deduce that in the FPM formulation the weighting least-squares functional J (4) can be considered like an error indicator. This functional J calculates the quadratic deviation between the approximation and the unknown value, called nodal contribution. Then the error estimator in each node, can be defined like

$$\hat{e}_{Ij} = \sum_{j=1}^{n} w(x_I - x_j)[u_j^h - \hat{u}(x_j)]^2. \tag{18}$$

4.2. Introduction of new points

Once the error estimator/indicator for the points or edges has been evaluated, a new degree of freedom (i.e. points) can be introduced. In principle, FPM offers the liberty to introduce new points in a completely general and arbitrary way. In order to arrive at a more orderly introduction, the new points are introduced at the mid-point location of edges. This has the advantage of ensuring that in the limiting case of a perfectly regular, Cartesian point distribution, the regularity of the initial point distribution is maintained. The algorithmic steps required for new point cloud may then be summarized as follows:

- evaluate the desired error estimator/indicator;
- obtain the edges to be refined;
- introduce the new points;
- update the boundary conditions.

Next, we describe in more detail the techniques and parameters used in each one of these steps.

4.2.1. Edges to be refined

The initial list of edges marked for refinement needs to be trimmed further in order to avoid too many new points. The following steps are taken:

(a) *Retention of closest near-neighbour layer*: Local clouds, and hence edges, may go beyond nearest neighbours. The introduction of a new point for such an edge could lead to new points being very close to existing nodes. Therefore, taking advantage that we know the near-neighbour layer number of any local cloud of points from the Delaunay triangulation, we only allow the introduction of points for edges that belong to the first near-neighbour layer.

(b) *Retention of ordered edges*: Nearly all edges to be refined appear twice (once in the local cloud of node i and than also in the local cloud of node j). Therefore, we only retain for refinement those edges for which $i < j$.

(c) *Rejection of small edges*: In cases with singularities or physical discontinuities, the introduction of new points will continue unhindered unless a stopping criterion is imposed. The first of these is the proximity that points can have to each other. If an existing edge is smaller than this preset length tolerance, the edge is not retained for refinement.

(d) *Rejection of edges with new points close to existing points*: Some of the edges marked for refinement may lead to the introduction of points that are very close to existing points. In order to avoid such problems, the existing points are introduced in an octree. Every edge marked for refinement. If the new point position is too close to an existing point, the edge is rejected. Note that the bounding box of the edge provides a natural search region for the octree.

(e) *Rejection of edges with new points close to new points*: Some of the edges marked for refinement may lead to the introduction of points that are very close to new points. A typical case is shown in Fig. 6. Edges i, k and j, l will introduce new points that are at almost identical positions. In order to avoid such problems, the points introduced by the edges marked for refinement are stored in an octree. Every edge marked for refinement is then checked. If a new point position from an edge with lower number is too close to the new point, the edge is rejected.

4.2.2. Introduction of new points

The new points are introduced at the mid-points of all edges marked for refinement. The unknowns are averaged. Alternatively, one could use the values recovered from a higher-order polynomial approximation.

4.2.3. Update of boundary conditions

The edges marked for refinement also provide a mechanism for the update of boundary conditions. Assuming that we know, for each boundary point, the CAD entity it belongs to (end-point of line, line, surface), one can

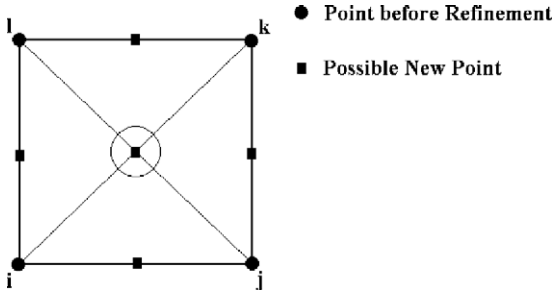

Fig. 6. Removal of close new points.

establish if new points originating from edges are on the boundary or not, as well as their proper CAD entity. The new boundary points thus found are added to the respective lists.

5. Computational implementation

In this section, the computational implementation of elastic solids mechanics applying FPM is described.

Let the differential equations system that describes the behavior of a linear-elastic solid, where inertia forces are not considered, be

$$\nabla \cdot \sigma(x) + \rho \cdot b(x) = 0 \quad \forall x \in \Omega \tag{19}$$

with the Neumann condition

$$\sigma(x) \cdot \hat{n} = \bar{t}(x) \quad \forall x \in \Gamma_t \tag{20}$$

and the Dirichlet condition

$$u(x) = \bar{u}(x) \quad \forall x \in \Gamma_u. \tag{21}$$

Being σ, ρ, b, $\hat{n}$, u, $\bar{t}$, $\bar{u}$; the stress tensor, the solid density, vector of internal forces, normal vector, displacement vector, traction prescribed vector and displacement prescribed vector over the boundaries Γ_t and Γ_u respectively.

Utilizing the FWLS approximation of the $u(x)$ function (1) in the sub-domain Ω_I for the displacement field and the point collocation technique, finally the discrete system obtained is

$$(\lambda+\mu)\nabla(\nabla \cdot \hat{u}(x_I)) + \mu\nabla^2\hat{u}(x_I) + \rho \cdot b(x_I) = 0 \quad \forall x_I \in (\Omega - \Gamma), \tag{22a}$$

$$\lambda(\nabla \cdot \hat{u}(x_I)) \cdot n + \mu(\hat{u}(x_I) \otimes \nabla + \nabla \otimes \hat{u}(x_I)) \cdot n = \bar{t}(x_I) \quad \forall x_I \in \Gamma_t, \tag{22b}$$

$$\hat{u}(x_I) = \bar{u}(x_I) \quad \forall x_I \in \Gamma_u. \tag{22c}$$

Note that in the above system, only interior nodes support the equilibrium equation (22a).

On the other hand, the Neumann boundary condition is imposed at Γ_t. In the cited notation, λ, μ are the *Lamé* constants that characterize the elastic behavior of the material, and the operators $\otimes$ and ∇ are the gradient and divergence respectively.

The numerical implementation, produce the following equations system

$$K_I \cdot u^h = f_I \quad I = 1, \ldots, N. \tag{23}$$

In same cases K matrix presents instabilities problems with the imposition of Neumann boundary condition by means of Eq. (22b) [4]. For this reason, a stabilization process has been implemented, based in the finite calculus procedure (FIC), developed by Oñate [17,22].

Then applying the FIC techniques, the stabilized form of the equations is

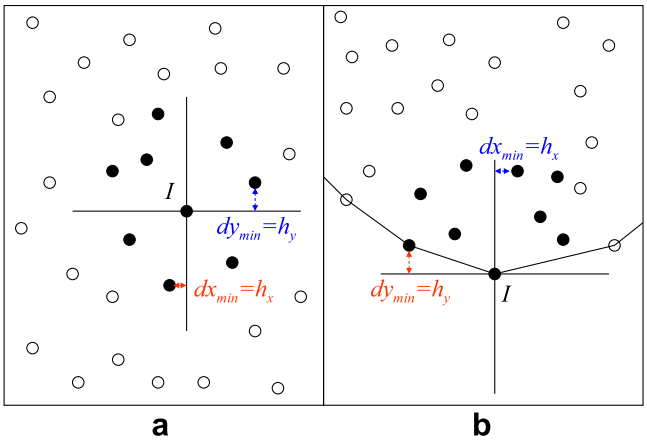

Fig. 7. Characteristic length h for (a) interior nodes and (b) boundary nodes.

$$\left\{(\nabla \cdot \sigma(x_I) + \rho \cdot b(x_I)) - \frac{1}{2}h^{\mathrm{T}} \cdot \nabla(\nabla \cdot \sigma(x_I) + \rho \cdot b(x_I))\right\} = 0 \quad \forall x_I \in (\Omega - \Gamma), \tag{24a}$$

$$\left\{\sigma(x_I) \cdot \hat{n} - \bar{t}(x_I) - \frac{1}{2}h_n \cdot (\nabla \cdot \sigma(x_I) + \rho \cdot b(x_I))\right\} = 0 \quad \forall x_I \in \Gamma_t, \tag{24b}$$

$$\hat{u}(x_I) = \bar{u}(x_I) \quad \forall x_I \in \Gamma_u, \tag{24c}$$

where h is a characteristic length of the finite sub-domain (Fig. 7) and $h_n = |h^{\mathrm{T}} \cdot \hat{n}|$.

In this work the h vector was chosen as

$$h^{\mathrm{T}} = [d_{x\min} d_{y\min}], \tag{25}$$

where $d_{x\min}$ is the minimal distance in X-direction from the I node to the nodes of the cloud and $d_{y\min}$ is the minimal distance in Y-direction from the I node to the nodes of the cloud.

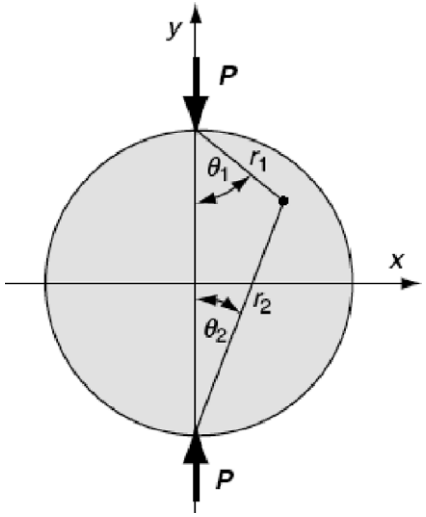

Fig. 8. Disc under diametral compression.

6. Example

In order to validate the proposed error indicator based in J functional (4) a 2D example is presented.

The following problem consists of a disc under diametral compression load, where $P = 1.0$, diameter $D = 0.5$ as shown in Fig. 8. A plane stress condition is considered, a material with $E = 1.0$ and $v = 0.25$. This problem is used in standardized tests (ASTM D-4123, 1987) for bituminous and fragile materials.

With an appropriate coordinate change [21] and considering $R = D/2$ (disc radium) the theoretical stress field is

$$r_1 = \sqrt{x^2 + (R-y)^2} \quad r_2 = \sqrt{x^2 + (R+y)^2}, \tag{26}$$

$$\sigma_{xx} = -\frac{2P}{\pi}\left[\frac{(R-y)x^2}{r_1^4} + \frac{(R+y)x^2}{r_2^4} - \frac{1}{D}\right], \tag{27a}$$

$$\sigma_{yy} = -\frac{2P}{\pi}\left[\frac{(R-y)^3x^2}{r_1^4} + \frac{(R+y)^3}{r_2^4} - \frac{1}{D}\right], \tag{27b}$$

$$\tau_{xy} = \frac{2P}{\pi}\left[\frac{(R-y)^2x}{r_1^4} - \frac{(R+y)x^2}{r_2^4}\right]. \tag{27c}$$

The maximum shear stress is calculated using the following expression:

$$\tau_{\max} = \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2}. \tag{28}$$

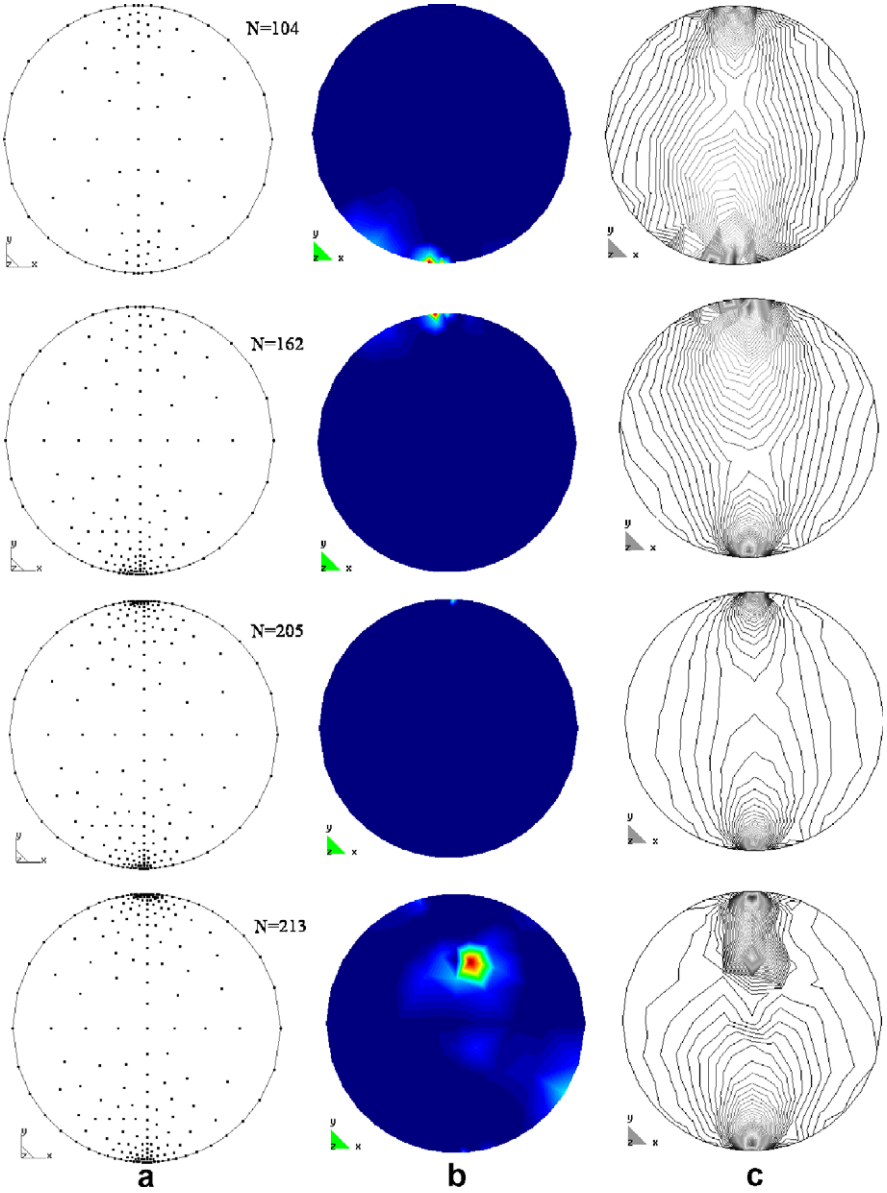

Fig. 9. Disc under diametral compression: (a) discretization, (b) error indicator and (c) maximum shear stress iso-lines for $N = 104$, 162, 205, and 213 points.

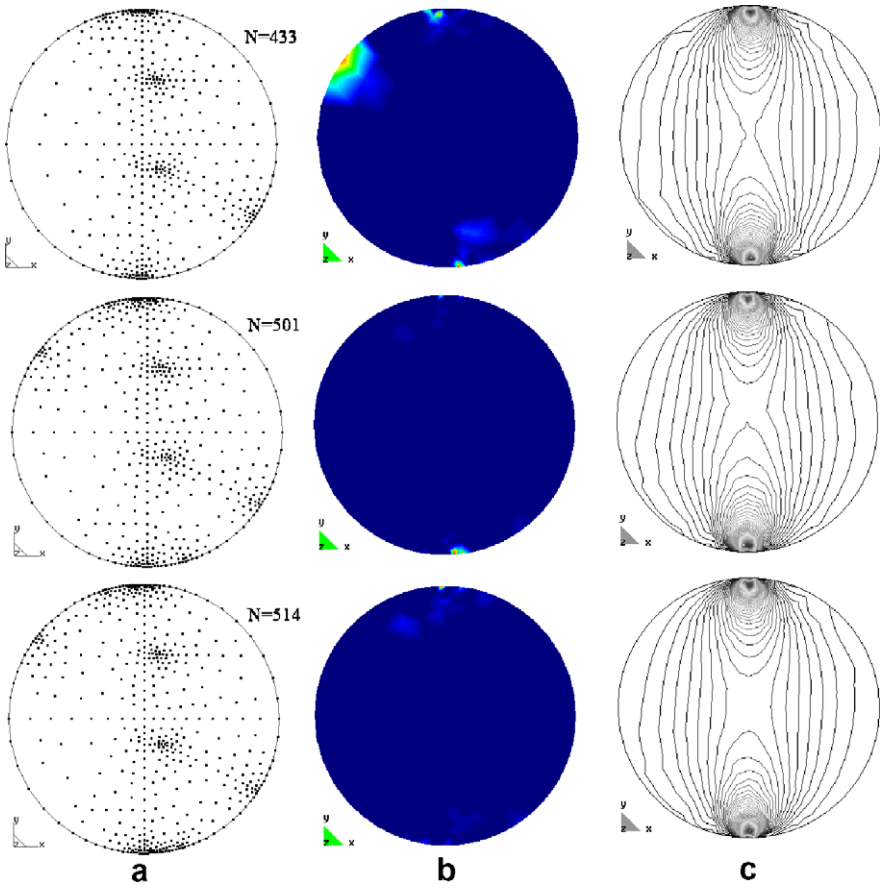

Fig. 10. Disc under diametral compression: (a) discretization, (b) error indicator and (c) maximum shear stress iso-lines for $N = 433$, 501, and 514 points.

This example is solved using a quadratical interpolation bases (2b), is to say, $m = 6$.

At the beginning the disc is discretized by $N = 104$ points, later the location of the points is adapted because of the values of the proposed indicator (18).

As seen in Figs. 9 and 10, the maximum shear stress tends to the optimal distribution (Fig. 11).

As is shown in Figs. 9 and 10, in this example the position of new points is not always symmetric, this is because the initial clouds are not symmetric respect to the "star node". Nevertheless the distribution of the numerical result (shear stress) is symmetric like the theoretical solution.

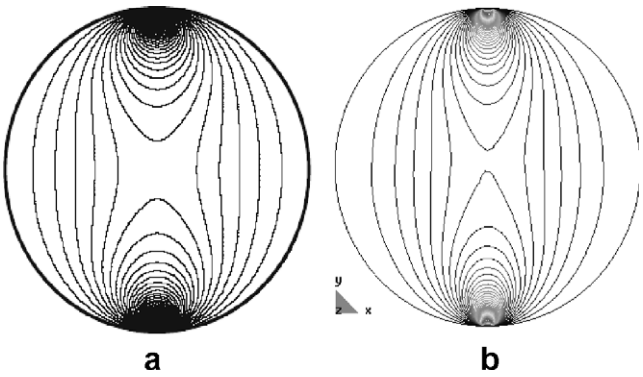

Fig. 11. Lines of maximum shear stress for disc under diametral compression: (a) theoretical [21], (b) numerical $N = 5.851$ points (applying a non-adaptive process).

7. Conclusions

In the present work, a strategy of adaptive refinement applicable to meshless method based on point collocation has been presented, which has been evaluated in a numerical 2D example in order to verify its quality. The methodology of local clouds generation presented, permits to avoid all of the discretization problems generated by complex geometries (sharp faces).

The strategy of change of polynomial order in clouds for boundary points is a good alternative in order to minimize the induced errors caused with the imposition of Neumann condition. This allows decreasing the necessary refinement in a particular boundary. The use of the error indicator based on J (4) is an important advantage of the proposed techniques, because FPM formulation is precisely based on the minimisation of this functional.

From the present results, one can observe that the proposed methodology is easily extensible to the 3D case; in fact, the authors are working on extending the analysis to more real solid pieces. On the other hand, all of this

proposed methodology can be applied in future research to other kind of computational mechanics problems, for example CFT, fracture mechanics, etc., always in the FPM context.

Acknowledgements

The authors acknowledge the financial support from FONDECYT Proyect 1040371 and UTFSM DGIP Proyect 250521.

References

[1] Oñate E, Idelsohn S, Zienkiewicz OC, Taylor RL. A finite point method in computational mechanics. Applications to convective transport and fluid flow. Int J Numer Methods Eng 1996;39:3839–66.
[2] Oñate E, Idelsohn S. A mesh-free finite point method for advective-diffusive transport and fluid flow problems. Comput Mech 1998;21:283–92.
[3] Oñate E, Sacco C, Idelsohn S. A finite point method for incompressible flow problems. Comput Visual Sci 2000;3:67–75.
[4] Oñate E, Perazzo F, Miquel J. A finite point method for elasticity problems. Comput Struct 2001;79:2151–63.
[5] Löhner R, Sacco C, Oñate E, Idelsohn S. A finite point method for compressible flow. Int J Numer Methods Eng 2002;53:1765–79.
[6] Boroomand B, Tabatabaei AA, Oñate E. Simple modifications for stabilization of the finite point method. Int J Numer Methods Eng 2005;63:351–79.
[7] Rabczuk T, Belytschko T. Adaptivity for structured meshfree particle methods in 2D and 3D. Int J Numer Methods Eng 2004;63:1559–82.
[8] Kim HG, Atluri SN. Arbitrary placement of secondary nodes, and error control, in the meshless local Petrov-Galerkin (MLPG) method. CMES 2000;3:11–32.
[9] Park SH, Kwon KC, Youn SK. A posteriori error estimates and an adaptive scheme of least-square meshfree method. Int J Numer Methods Eng 2003;58:1213–50.
[10] Perazzo F. Una metodología numérica sin malla para la resolución de las ecuaciones de elasticidad, mediante el método de Puntos Finitos. Doctoral thesis UPC 2002.
[11] Löhner R. Some useful data structure for the generation of unstructure grids. Commun Appl Meth 1988;4:123–35.
[12] Löhner R, Oñate E. An advancing point grid generation technique. Commun Numer Methods Eng 1998;14:1097–108.
[13] Knuth DN. The art of computer programming, vols. 1–3. Addison-Wesley; 1973.
[14] Samet H. The quadtree and related hierarchical data structures. Comput Surv 1984;2:187–285.
[15] George PL, Borouchaki H. Delaunay triangulation and meshing. Editions Hermes; 1998.
[16] Löhner R. Applied CFD techniques. J. Wiley & Sons; 2001.
[17] Oñate E. On the stabilization of numerical solution for advective-diffuse transport and fluid flow problems. CIMNE 1996:81.
[18] Perrone N, Kao R. A general finite difference method for arbitrary meshes. Comput Struct 1975;5:45–58.
[19] Liszka T, Orkisz J. The finite difference method at arbitrary irregular grids and its application in applied mechanics. Comput Struct 1980;11:83–95.
[20] Liszka T, Duarte CA, Tworzydlo W. hp-Meshless cloud method. Comput Methods Appl Mech Eng 1996;139:263–88.
[21] Saad MH. Elasticity. Elsevier Butterworth-Heinemann; 2005, p. 480.
[22] Oñate E. Possibilities of finite calculus in computational mechanics. Int J Numer Methods Eng 2004;60:255–81.