Asociación Argentina de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 4753-4767 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

SHAPE FUNCTION OBJECT MODELING FOR MESHFREE METHODS

José M. Morales Sánchez[a], **Paulo B. Gonçalves**[b]

[a]*Departamento de Tecnologia em Arquitetura e Urbanismo, Universidade de Brasília, Campus Universitário Darcy Ribeiro, Brasília, DF, Brasil, sanchez@unb.br, http://unb.br/fau/pos_graduacao/*

[b]*Departamento de Engenharia Civil, Pontifícia Universidade Católica, PUC-Rio, Av. Marquês de São Vicente, 225, Rio de Janeiro, RJ, Brasil, paulo@civ.puc-rio.br, http://www.civ.puc-rio.br/pt/index.php*

Keywords: Meshfree Methods, Shape Functions, Moving Least Square Method, Object Oriented Analysis, JAVA.

Abstract. This article presents a computational object modeling for shape functions used in Meshfree Methods, especially those derived from the of Moving Least Square Method, such as Generalized Finite Differences, Element-Free Galerkin, Hp-Clouds and Finite Point Method, for example. The class model includes abstractions for the basis of functions, the weighting functions and the strategy of building the sparse point cloud. These abstractions allow the definition of classes of shape functions considering specific problems and methods. It should be noted that the Finite Element Method fits this class structure; the points that define the element are the cloud points in Meshfree Method. As ultimate goal, this work aims to build a framework of classes, which allows the construction of agile applications for numerical experiments of Meshfree Methods. In this work the Java language was adopted to implement the proposed design.

1 INTRODUCTION

This article continues the study of Meshfree Methods and the possibilities of its modeling by computational objects. This work aims to facilitate the computational implementation and the numerical experimentation with these methods (Sánchez, 2003).

Meshfree Methods kernels are shape functions that allows constructing approximation functions based in a set of scattered points (cloud). Some variation of this shape functions are used in different variants of these methods. The study presented here details and extends the portion concerning the object-oriented analysis for Meshfree Methods shown in previous article (Sanchez *et al.*, 2004-b).

The abstractions developed are based in the theoretical framework of weighted residuals methods. This approach clarifies and encompasses many specialized methods described in the literature. It is worth mentioning that the Meshfree Methods treated here are those known in the literature as domain approach (Belystchko *et al.*, 1996; Fries and Matthies, 2004). Therefore boundary approaches such as Boundary Elements Method or Fundamental Solutions Method are not considered in this study.

This article develops in section 2, the mathematical background for shape functions used in Meshfree methods, considering the Moving Least Square Method, proposed by Šalkauskas and Lancaster (1981). Then in section 3 presents some criteria for the construction of the cloud of points. The modeling of the classes that allow creating objects of shape functions is in Section 4. Some aspects of modeling implementation in the Java language are discussed in Section 5.

The Finite Element Method can be demonstrated as a particular case of the method of weighted residuals and given its importance in the aspect of comparing results and even the coupling with Meshfree methods, abstractions have been developed to include this important method of discretization.

Obviously, it is not the purpose of this work to encompass all Meshfree methods. The virtue of the object analysis is the possibility of its extension. From the analysis presented here it is intended that other methods of determining the shape functions can be quickly incorporated allowing the evolution of object-oriented system proposed.

2 MESHFREE METHODS SHAPE FUNCTIONS

Approximation function based on a set of nodal points scattered and without interconnection follows in the literature three lines of research. The oldest approach uses the kernel approximation $u^h(x)$ over the domain as:

$$u^h(x) = \int_\Omega w(x-y,h)\, u(y)\; d\Omega_y \tag{1}$$

where *w(x-y, h)* is the kernel or weighting function, *h* is a measure of the extent of support in which *w(x-y)* is nonzero (Lucy, 1977 *apud* Belytschko *et al.*, 1994).

The approach of Meshfree Methods can also be based on partitions of unity (Babuška and Melenk, 1997), which constitutes a paradigm in which sub-domains, each associated with a function that is nonzero in it, covers the domain. In this case the approximation function takes the following form:

$$u^h(x) = \sum_i \phi_i^k(x)\left(u_i + \sum_{j=1}^{m} b_{ji} q_i(x) \right) \tag{2}$$

where ϕ is the function partition of unit, b are coefficients and q are base functions, generally monomials.

The third form of generating functions for approximation of Meshfree Methods is to use the method of Moving Least Square Method (Šalkauskas and Lancaster, 1981). The technique consists in minimizing the functional of the weighted squared error, on the form:

$$E_x(u) = \sum_{i=1}^{N} w_i(x)\left(u(x_i) - u_i\right)^2 \tag{3}$$

where, w_i is a weighting function.

In this work will be used as technique to determine the approximation functions the Moving Least Square Method of Lancaster and Šalkauskas, limiting the study of Meshfree Methods to those using this technique in the generation of shape functions.

2.1 Moving Least Square Method

Most Meshfree Methods have shape functions generated from some particular way by the Moving Least Square Method. Let $u(x)$ be a function defined on a domain. Adopting an approximate solution in the form

$$u(x) \approx u^h(x) = \sum_{i=1}^{m} p_i a_i = \boldsymbol{p}^T \boldsymbol{a} \tag{4}$$

where a_i are coefficients and p_i are the terms of the basis, usually monomials due to the facilities involved in the numerical analysis with polynomials. For a 2D problem, for example, linear and quadratic basis are given by

$$\begin{aligned} \boldsymbol{p}^T &= \begin{bmatrix} 1 & x & y \end{bmatrix}, \quad m = 3 \\ \boldsymbol{p}^T &= \begin{bmatrix} 1 & x & y & x^2 & xy & y^2 \end{bmatrix}, \quad m = 6 \end{aligned} \tag{5}$$

where, m defines the dimension of each basis.

Adopting a set of n nodal points and applying expression (4) comes to organizing in matrix form

$$\boldsymbol{u} = \boldsymbol{P}\ \boldsymbol{a} \tag{6}$$

If the number of terms of base (dimension) equals the number of nodal points of the sub domain, *i.e.*, $m = n$, the system (6) can be solved, imposing certain conditions on the geometric organization of the nodal points, which in replacing (4) allows to relate the function approximation in terms of nodal values, i.e.,

$$\begin{aligned} u^h(x) &= \boldsymbol{N}^T(x)\ \boldsymbol{u} \\ \boldsymbol{N}^T(x) &= \boldsymbol{p}^T(x)\boldsymbol{P}^{-1} \end{aligned} \tag{7}$$

where $\boldsymbol{N}^T$ are called ***shape functions***, that with a convex geometry coincides with the definition found in the Finite Element Method (Zienkiewicz and Taylor, 1989).

If the set of points in the sub domain is much greater than the dimension of the basis functions, the definition of the approximation function must be made to some criterion of adjustment, for example, minimizing the functional of the square of the error between the approximation function and the function value. Adopting a weighting function that allows considering the relative importance of the points nearest arrives at the functional

$$\begin{aligned} E(u,x) &= \left(\boldsymbol{P}\,\boldsymbol{a}-\boldsymbol{u}\right)^{T}\boldsymbol{W}(x)\left(\boldsymbol{P}\,\boldsymbol{a}-\boldsymbol{u}\right) \\ \boldsymbol{W}(x) &= diag\left[\begin{array}{cccc} w_{1}(x-x_{1}) & w_{2}(x-x_{2}) & \cdots & w_{n}(x-x_{n}) \end{array}\right] \end{aligned} \tag{8}$$

The weighting function w_i has major role in the method formulation. If the value is nonzero only in a sub-domain, the number of points will be a function of position x, then $n = n(x)$. This set of points in the neighborhood of x for which $w_i \neq 0$ forms its domain of influence, or ***support***, also called ***cloud***.

Normally it is used weighting functions that depend only on the distance between two points, i.e.

$$w_{i}(x-x_{i}) = w_{i}(d) \tag{9}$$

where $d = \|x - x_i\|$ is the distance between x and x_i.

The weighting function most used is the following Gaussian spherical truncated function

$$w_{i}(x_{i},x) = w_{i}(r) = \frac{\exp(-\beta^{2}r^{2}) - \exp(-\beta^{2})}{1-\exp(-\beta^{2})} \tag{10}$$

In the above expression, $r = d_i / dm_i = \|x - x_i\| / dm_i$, where dm_i is the support dimension of w_i. The parameter beta affects the bell shape of the weighting function, and is usually adopted as $\beta = 4$ (Belytschko *et al.*,1994), in order to nullify the derivative of the function on its contour ($r = 1$). The choice of weighting function is quite arbitrary; other functions are reported in the literature, such as spline functions and cubic B-spline, as seen in the subsection below.

The process of minimizing the functional (8) should be done for every point x where the value of the approximation $u^h(x)$ is required, hence the reference to **Moving Weighted Least Square Method**.

Performing the minimization and substituting in (4), yields

$$\begin{aligned} u^{h}(x) &= \boldsymbol{p}^{T}(x)\boldsymbol{A}^{-1}(x)\boldsymbol{B}(x)\,\boldsymbol{u} = \phi^{T}(x)\,\boldsymbol{u} \\ \phi^{T}(x) &= \boldsymbol{p}^{T}(x)\boldsymbol{A}^{-1}(x)\boldsymbol{B}(x); \quad \boldsymbol{A}(x) = \boldsymbol{P}^{T}\boldsymbol{W}(x)\boldsymbol{P}; \quad \boldsymbol{B}(x) = \boldsymbol{P}^{T}\boldsymbol{W}(x) \end{aligned} \tag{11}$$

where ϕ are the shape functions for a set of scattered points (cloud) in a sub domain Ω_i.

The Moving Least Square approximation above depends crucially on how the weighting functions are applied (Oñate *et al.*, 1996). This weighting can be accomplished in two ways.

Initially it is assumed that there is a set of nodal points, x_i, in a domain, Ω. For any point x_k, not fixed on the domain, determine the support or sub-domain Ω_k, according to some criterion, that must guarantee the existence of solution of normal equations; usually a minimum number of nodal points within Ω_k. The obtainment of the function value focused on x_k is made by using the values of the weighting function at each point x_i, $w_k(x_i)$. Figure 1-a illustrates it.

The other possibility of weighting is constructed from the definition for each nodal point x_i of a support Ω_i chosen so that guarantees the existence of solution of normal equations and covering the whole domain Ω. The weighting of the squared differences between the approximate function $u_h(x_i)$ and exact function, u_i, is performed using the respective value of the weighting function determined by any moving point x_k, as shown in Figure 1-b.

The derivatives of shape functions can be obtained from operating the equation (11). The derivative of the inverse matrix can be obtained by deriving both members of the definition of inverse matrix, $\boldsymbol{AA^{-1}=I}$. So for the direction *s*:

$$\begin{aligned}\phi_s^T(x) &= \boldsymbol{p}_s^T(x)\boldsymbol{A}^{-1}\boldsymbol{B}+\boldsymbol{p}^T\left(\boldsymbol{A}_s^{-1}\boldsymbol{B}+\boldsymbol{A}^{-1}\boldsymbol{B}_s\right)\\ \boldsymbol{A}_s &= \boldsymbol{P}^T\boldsymbol{W}_s(x)\boldsymbol{P};\quad \boldsymbol{B}_s=\boldsymbol{P}^T\boldsymbol{W}_s(x);\quad \boldsymbol{A}_s^{-1}=-\boldsymbol{A}^{-1}\boldsymbol{A}_s\boldsymbol{A}^{-1}\end{aligned}\tag{12}$$

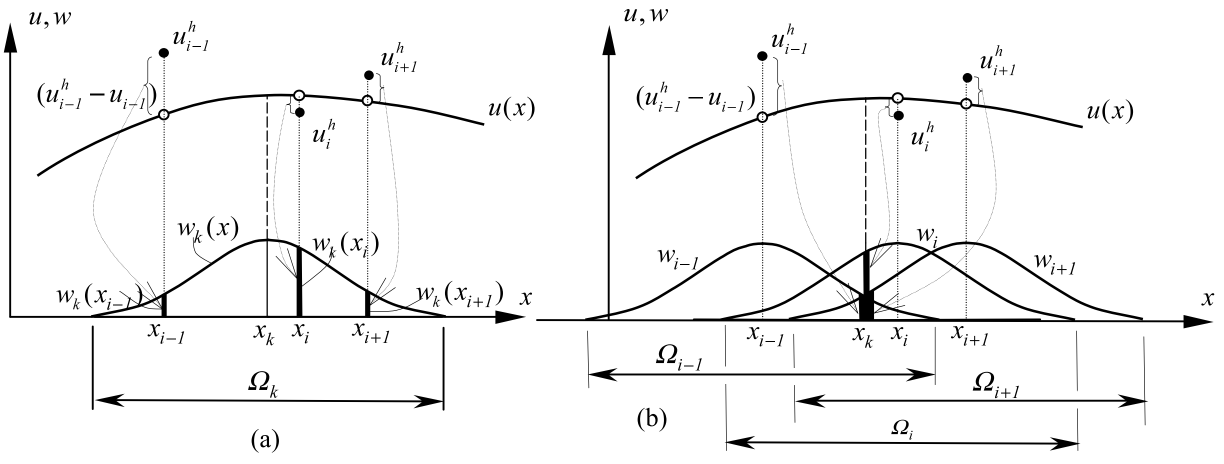


Figure 1: Shape functions from simple support (a), and multiple support (b).

The derivative of the function, taking a second direction, *t*, is given by

$$\begin{aligned}\phi_{st}^T &= \boldsymbol{p}_{st}^T\boldsymbol{A}^{-1}\boldsymbol{B}+\boldsymbol{p}_s^T\left(\boldsymbol{A}_t^{-1}\boldsymbol{B}+\boldsymbol{A}^{-1}\boldsymbol{B}_t\right)+\\ &\quad \boldsymbol{p}_t^T\left(\boldsymbol{A}_s^{-1}\boldsymbol{B}+\boldsymbol{A}^{-1}\boldsymbol{B}_s\right)+\boldsymbol{p}^T\left(\boldsymbol{A}_{st}^{-1}\boldsymbol{B}+\boldsymbol{A}_s^{-1}\boldsymbol{B}_t+\boldsymbol{A}_t^{-1}\boldsymbol{B}_s+\boldsymbol{A}^{-1}\boldsymbol{B}_{st}\right)\\ \boldsymbol{A}_{st} &= \boldsymbol{P}^T\boldsymbol{W}_{st}(x)\boldsymbol{P};\quad \boldsymbol{B}_{st}=\boldsymbol{P}^T\boldsymbol{W}_{st}(x)\\ \boldsymbol{A}_{st}^{-1} &= -\boldsymbol{A}^{-1}\left[\boldsymbol{A}_{st}\boldsymbol{A}^{-1}+\boldsymbol{A}_s\boldsymbol{A}_t^{-1}+\boldsymbol{A}_t\boldsymbol{A}_s^{-1}\right]\end{aligned}\tag{13}$$

Derivatives of higher order can be obtained by successive differentiation, the order of differentiation being limited by the degree of continuity of the weighting function w_i *(x)*. The expressions deduced for the derivatives depend on the inversion of matrix $\boldsymbol{A}$. The inverse of the derivatives of $\boldsymbol{A}$ were expressed only in terms of $\boldsymbol{A^{-1}}$. An alternative computationally interesting, not presented here, is to determine the shape functions and derivatives obtained by adopting a procedure of LU decomposition of matrix $\boldsymbol{A}$ and making backward substitutions with independent terms that allow to find ϕ, ϕ_s and ϕ_{st}.

The shape functions generated using the Moving Least Square Method present characteristics of reproducibility and continuity. These functions can reproduce any of the functions of its base. Thus, if the constant unit function is the only function included in the base, the shape functions are a partition of unity. These functions, known as Shepard functions has the form

$$\phi_i=\frac{W_i(x)}{\sum_{j=1}^{n}W_j(x)}\tag{14}$$

They also replicate the functions *x* and *y* if included, an important condition to ensure convergence.

The other property mentioned is the high continuity obtained, which depends only on the degree of continuity of the weighting function.

2.2 Basis Functions

From the previous item exposition it appears that the choice of basis functions is crucial in defining an approximate function to an exact function. This basis function should contain functions linearly independent of a complete sequence. The classes of function most commonly found are monomials that generate polynomial approximations, sines and cosines, and exponential functions.

So for one-dimensional problems the following basis is used

$$\boldsymbol{p}^T = [1 \quad x \quad x^2 \ldots \, x^{m-1}] \tag{15}$$

where m is the size of the basis function.

For two-dimensional problems the base can be represented by Pascal's triangle. For a base of quadratic order, for example, the base has a dimension $m = 6$ and is given by

$$\boldsymbol{p}^T = [1 \quad x \quad y \quad x^2 \quad xy \quad y^2] \tag{16}$$

As mentioned above, other functions can form the basis, selected from the analysis of specific problems in order to accelerate the convergence. Fleming *et al.* (1997) use for fracture problems in two-dimensional elasticity, the following base

$$\boldsymbol{p}^T = [1 \quad x \quad y \quad \sqrt{r}\cos\theta/2 \quad \sqrt{r}\sin\theta/2 \quad \sqrt{r}\sin\theta/2\sin\theta \quad \sqrt{r}\cos\theta/2\sin\theta] \tag{17}$$

where the dimension $m = 7$.

Another example of enrichment of the basis functions was performed by Suleau *et al.* (2000), which used sine and cosine for the analysis of problems governed by Helmholtz equation. For the problem of wave propagation in one dimension used the following basis with $m = 3$

$$\boldsymbol{p}^T = [1 \quad \cos kx \quad \sin kx] \tag{18}$$

In the Generalized Finite Differences Method (Liszka and Orkisz, 1980) the basis functions can be interpreted as the terms of a Taylor series, as below

$$\boldsymbol{p} = \left[1, x, y, \frac{x^2}{2}, xy, \frac{y^2}{2}, \cdots\right]^T \tag{19}$$

Liszka *et al.* (1996) include a degree of freedom of rotation to the approximation of the point near to the contour that can be considered as a Taylor basis for the normal derivatives

$$\boldsymbol{p}_n = \left[\; 0 \quad n_x \quad n_y \quad xn_x \quad (yn_x + xn_y) \quad yn_y \;\right]^T \tag{20}$$

Thus the basis function is related to both, type of problem to investigate, as may be linked to a specific numerical method.

2.3 Weighting Functions

The weighting function w_i has a prominent role in the formulation of the weighted Moving Least Square Method. By definition, the weighting function must comply with

i. $w_i(x) > 0 \; \forall \; x$.

ii. w_i should decrease monotonically with the growth of $|x|$.

This together with the linearly independent basis functions ensures that the matrix ***A(x)*** is positive definite and therefore invertible.

Another important effect is the possibility of interpolation obtained if the weighting functions satisfy the condition

iii. $$\begin{cases} w_i \to \infty & \text{if } x \to 0, \quad \text{or} \\ w_i(|x - x_i|) \to \infty & \text{if } x \to x_i \end{cases}$$

This feature is not explored enough in Meshfree Methods in general. An exception is the Generalized Finite Differences Method where the interpolation is essential.

The monotonic decrease suggests that the most distant points are considered less significantly. Thus we should define a truncated weighting function, which meets

iv. $$\begin{cases} w_i(d) > 0 & \text{for} \quad d \le dm_i \\ w_i(d) = 0 & \text{for} \quad d > dm_i \end{cases}$$

where $d = x - x_i$ is the distance between points x and x_i, and dm_i is the size of the support of the weighting function that determines the domain of influence of x_i. Thus the number of points (n) within the domain Ω_i may be variable for each point x_i. However the number of points must be at least greater than the size of the basis functions (m) used, in order to ensure that the normal equations have unique solution. Thus,

v. $n \in \boldsymbol{\Omega}_i \quad | \quad n > m$

This feature is crucial in Meshfree Methods, because once defined the support, a set of points are defined, or vice versa, which allows to eliminate the traditional nodal connectivity. Or, otherwise, we can say that connectivity is established mathematically, or more precisely through computational geometry, from the support of the weighting function.

The form of support is generally circular, in two dimensions, can also be rectangular. In this case the weighting functions are generated using the tensor product

vi. $w(\boldsymbol{x} - \boldsymbol{x}_i) = w(x - x_i) . w(y - y_i)$

These weighting functions can be advantageous for regular arrangements of nodal points.

Another important aspect in choosing a weighting function is the order of differentiability of the expected functions to be obtained. Lancaster and Šalkauskas (1981) show that if the basis function is of continuity class C^l and weighting function is of class C^m then the shape functions are of class C^k, where $k = min\ (l, m)$. In most cases the order of basis functions is that controls the differentiability of the shape functions.

In general the choice of weighting function is arbitrary since it meets the requirements listed above, which is mainly a function positive and continuous derivatives up to the desired degree. In literature are proposed different weighting functions applied to Meshfree Methods. Table 1 summarizes the main functions found with its expression and its bibliography.

3 CRITERIA FOR CONSTRUCTION OF THE CLOUD OF POINTS

In general, some mode of connectivity between nodes in a sub domain is adopted by discretization methods to generate the shape functions (Figure 2). In the method of finite differences a connection point called molecule (or star) is used. In finite element method, the connectivity between nodes of the sub domain defines the element. This element that merges with the sub domain is also used as domain of integration. But, in Meshfree Methods the shape functions are obtained from nodes belonging to sub domain without consideration of any explicit connectivity between these nodes, treated as a whole, as a cloud.

Actually what is sought is the association of a given point on the field with a set of nodal points surrounding to enable the construction of a local approach. So we can say that both the organization as molecule or as element is a special case of the organization by cloud.

Several criteria are used in the definition of the point cloud. In general, procedures found in computational geometry are used due to the large amount of points and the need for efficiency. Some methods require specific algorithms in order to realize its mathematical definition. For example, the method Hp-Clouds, it is necessary that the whole area be covered by an appropriate number of supports (Sánchez, 2003).

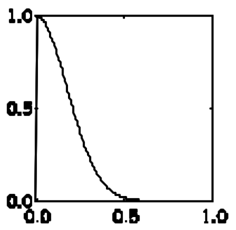
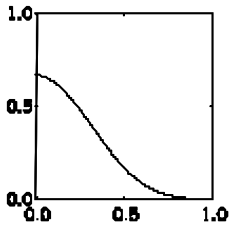
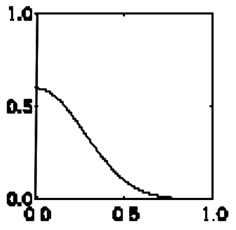
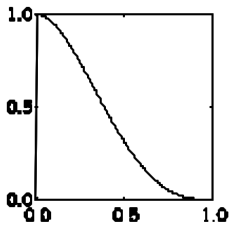
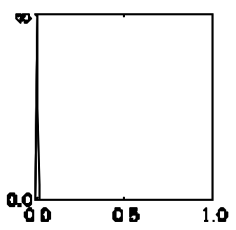
Picture	Expression	Reference
	***Truncated Gaussian Function*** $w=\dfrac{\exp(-\beta^2 r^2)-\exp(-\beta^2)}{1-\exp(-\beta^2)}$ $\quad 0\le r\le 1$ $\beta=4$	Belytschko *et al.* (1994)
	***3rd Order Spline*** $w=\begin{cases}\frac{2}{3}-4r^2+4r^3 & 0\le r\le 0.5\\ \frac{4}{3}-4r+4r^2+\frac{4}{3}r^3 & 0.5<r\le 1\end{cases}$	Dolbow and Belytschko (1998)
	***4th Order B-Spline Function*** $w=\begin{cases}C\left[(r-1)^4-5(r-3/5)^4+10(r-1/5)^4\right] & 0\le\\ C\left[(r-1)^4-5(r-3/5)^4\right] & 1/5\\ C\,(r-1)^4 & 3/5\le\end{cases}$ $C=\dfrac{1}{24}\dfrac{1}{(2/5)^4}=\dfrac{625}{384}$	Duarte and Oden (1996) Garcia *et al.* (2000)
	***4th Order Spline Function*** $w=1-6r^2+8r^3-3r^4 \quad 0\le r\le 1$	Beissel, Belytschko (1996)
	***Inverse Power Function*** $w=\dfrac{1}{c\,r^{\beta}}$ $c=1$ e $\beta=6$	Liszka *et al.* (1980) Liszka *et al.* (1996)

Table 1: Weighting functions.

The intention here is to build a framework with a collection of algorithms that can be tested and compared in order to investigate which of them allows a better result with a good computational response. The details of the algorithms mentioned in the next section will be subject of future publication.

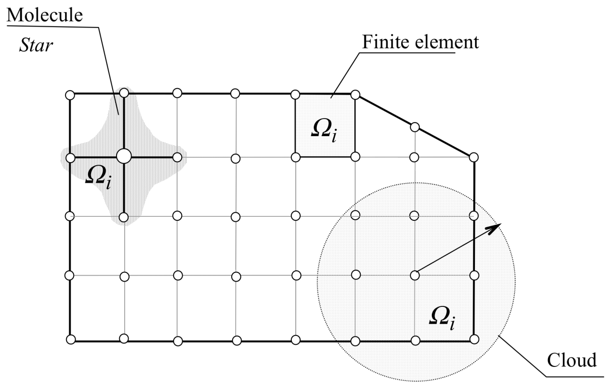


Figure 2: Connectivity of nodal points per molecule, element and cloud.

4 SHAPE FUNCTION OBJECT MODELING

The object modeling of shape functions involves the treatment of the following abstractions: the shape functions, the basis functions, the weighting functions and strategies for building the nodal point clouds.

A principle of reusable object-oriented design requires that one should plan for an interface and not an implementation (Gamma *et al.*, 2000). An interface is a protocol with a list of method declarations, not code, which must be realized by classes that implement this interface. That settles inside the inheritance mechanism an important decision. It seeks to inherit the behavior and not implementation code. The programming through interfaces is the basis for building scalable software. Thus all classes that implement a given interface will have similar external behavior differing by internal form they behave.

4.1 Modeling Interfaces and Classes for Shape Functions

The shape functions are built from the interface **ShapeFunction**. Figure 3 shows the UML diagram (Fowley and Scott, 2000; Gentleware, 2003) with the protocol of this interface. A class that wants to behave like a **ShapeFunction** should implement the code of all methods listed. In this case, must provide methods to determine the shape functions and their derivatives up to second order, which meets most of the problems addressed by the meshfree discretization methods. You should tell yet what were the point cloud and the pole in Cartesian coordinates used for its determination.

Figure 3 also shows two specializations from **ShapeFunction** interface, which are **FEM_SF** and **MLS_SF**. The first will be used to define the shape functions of isoparametric finite elements. For that it is included a method for determining the determinant of the Jacobian. All functions in a finite element must implement both protocols **ShapeFunction** and **FEM_SF**. These functions have a dual purpose, allowing building the finite element method and enabling the Gaussian integration in Meshfree methods with Galerkin variant.

Figure 4 shows the isoparametric finite elements developed in this work following the interface FEM_SF: quadrilateral elements, triangular and linear, which allow the integration of one-dimensional and two-dimensional domains, and surface element for integration into the two-dimensional contour. The construction aggregates data and operations common to different elements in the abstract class (which does not create objects) called **FEM_Operations**. The operations and data are transmitted by inheritance to the classes that wish to reuse the code or part of it. The Quad class, for example, implements the interface

FEM_SF and extends the abstract class **FEM_Operations**. It can be said that **Quad** inherits its genetic, or rather their behavior from **FEM_SF**, and that it inherits or not the assets of existing code in **FEM_Operations**. For simplicity were not represented the methods of the interfaces in concrete classes nor the constructors of the classes (methods that create objects of classes at runtime). The second derivatives of shape functions return null, once the elements implemented are class continuity C^1.

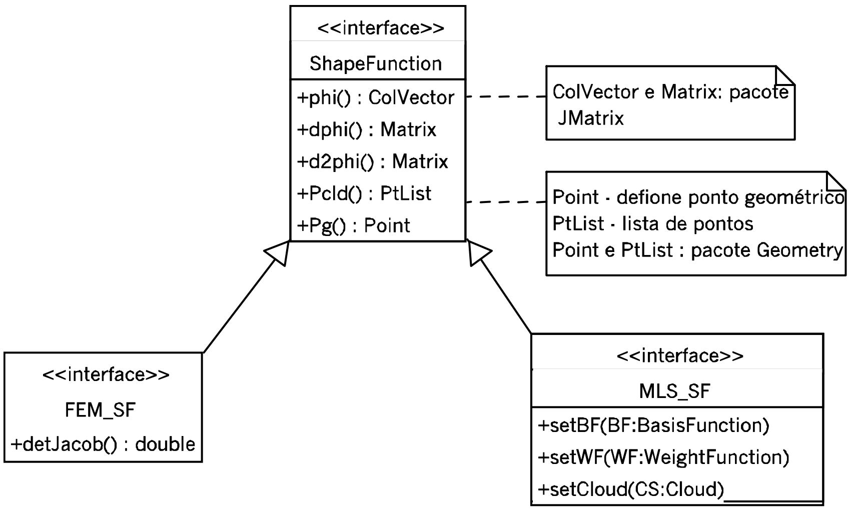


Figure 3: Shape functions interfaces.

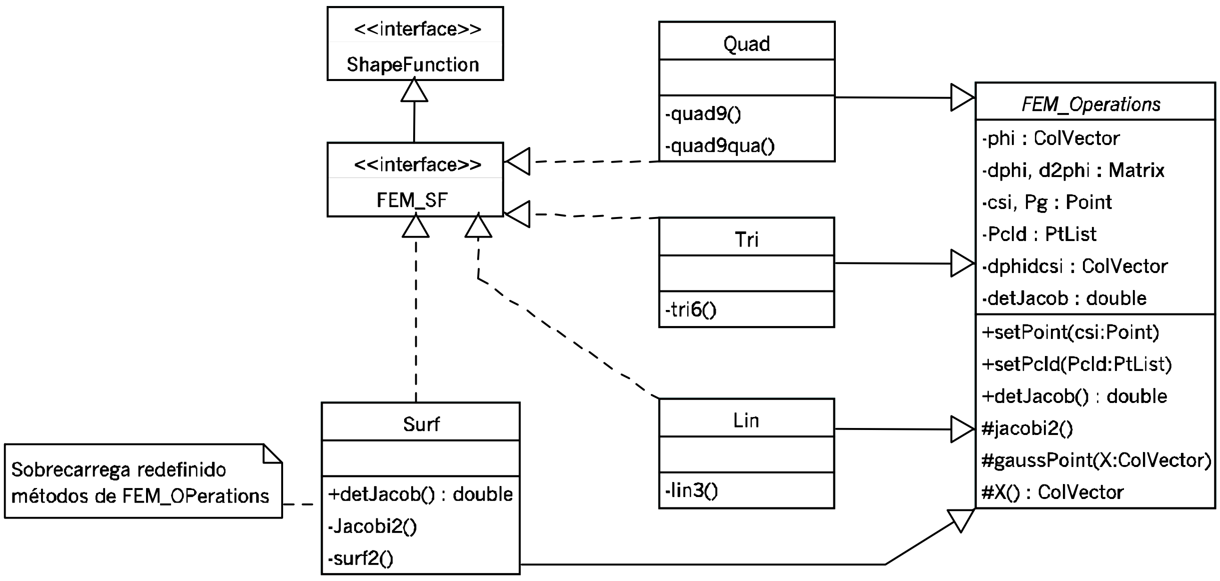


Figure 4: Model classes for isoparametric finite elements.

The interface **MLS_SF**, represented in Figure 5, allows creating the classes necessary to construct shape functions for the method of Moving Least Square Method. Its interface adds to the inherited interface **ShapeFunction** methods for defining the basis functions, weighting function and strategy of construction of the point cloud, classes which are detailed below. The class **MLS_UNI** determines the shape functions considering that a weighting support is set at any point. Already **MLS_MULT** class follows the strategy of weighting with support set for each nodal point. Here was also created an abstract class that collects the data and common methods that can be inherited or not by other classes. The **MLS_Taylor** class is a subclass of **MLS_UNI**, which determines the shape functions using a basis function consisting of Taylor series required for the Generalized Finite Differences Method. The class **HPC_SF** is responsible for generating the shape functions for the hierarchical method Hp-Clouds, which uses a partition function of the unit based on Shepard functions together with a monomial basis.

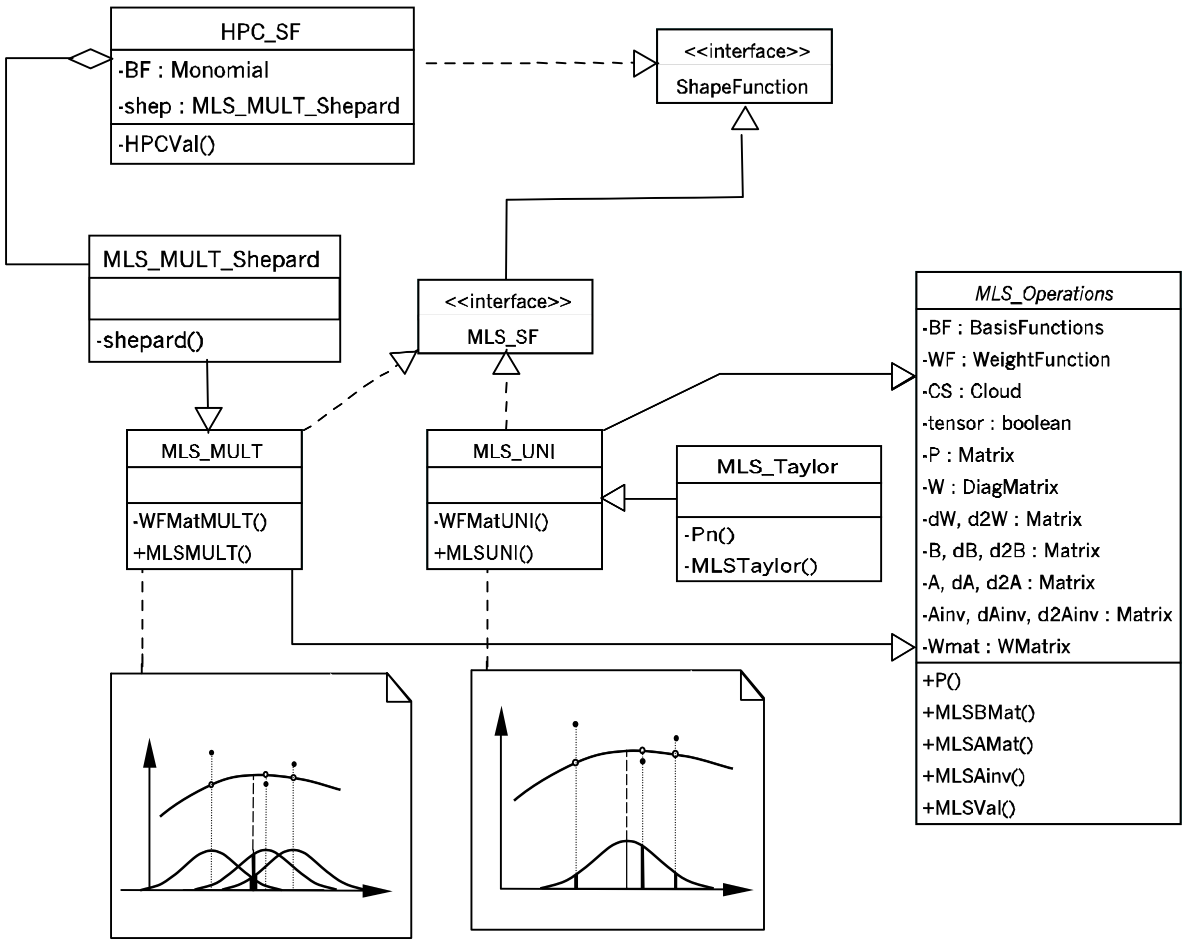


Figure 5: Model classes for Moving Least Square Method.

4.2 Modeling Basis Functions

Computer modeling of the basis functions follows similarly to that presented to the shape functions, by defining an interface protocol that specifies a behavior (**BasisFunctions**) and the creation of an abstract class (**BF_Operations**) container of data and methods that can be reused in classes that implement the interface. Figure 6 shows this structure and the concrete classes of named **Monomial** and **Crack**. The first implements the basis functions of monomials. Already **Crack** class represents the base proposed by Fleming *et al.* (1997) for fracture problems in Element-Free Galerkin Method. Classes **Taylor** and **TaylorN** implement basis functions obtained by Taylor series under the Generalized Finite Differences Method.

4.3 Modeling Weighting Functions

The computational modeling of objects that represent the weighting functions in the Moving Least Square Method should cover three abstractions: the weighting functions properly, the point cloud and the strategy for its construction. The first, most immediate, is determining the values of the weighting function and its derivatives. All functions were parametrized with the dimensionless radius *r* ranging from zero to one, corresponding to the size of the cloud of nodal points support, dm_i. Figure 7 shows the interface **WeightFunction** specifying the protocol for obtaining the function and its derivatives up to second order with respect to the radius *r*. Given the relative complexity of determining the first and second derivatives of the weighted matrix with respect to Cartesian coordinates is presented also in Figure 7 the class **WFDeriv**, which is used by classes that implement **MLS_SF** interface to determine the weighting diagonal matrix and its derivatives.

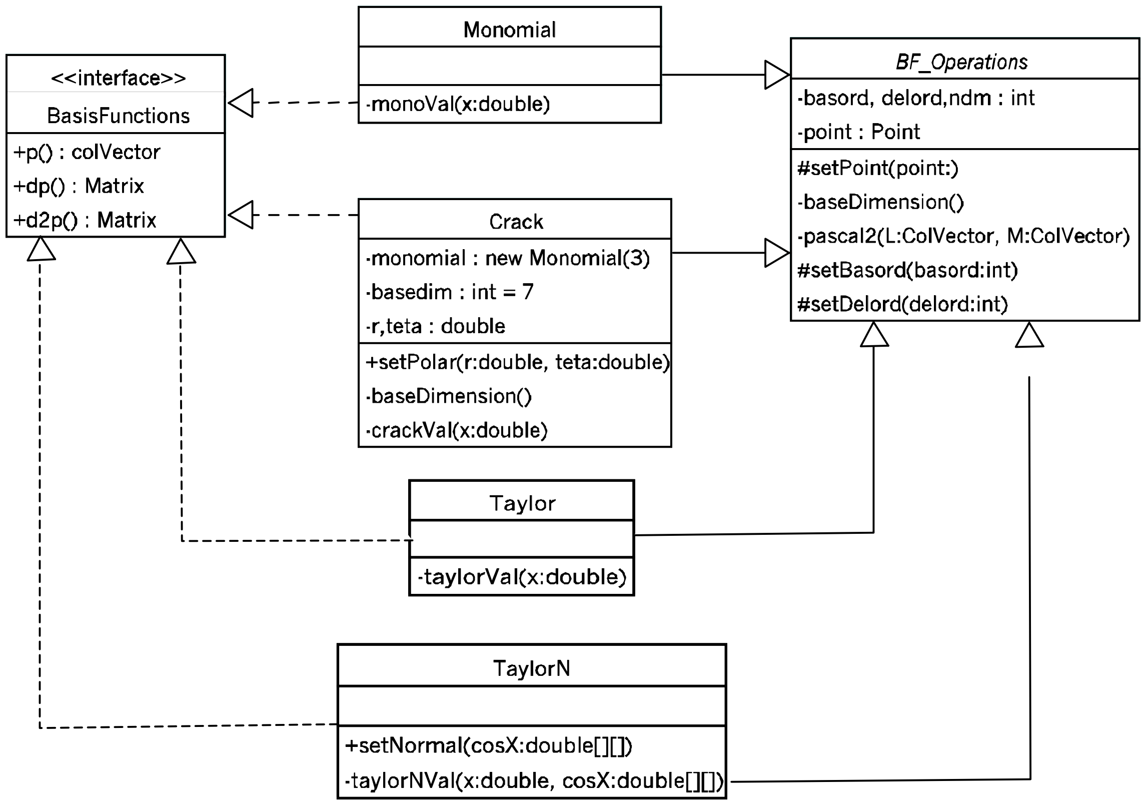


Figure 6: Model classes for basis functions.

4.4 Strategies to Construct the Cloud of Points

The second concept or abstraction important in determining the weighting function is the definition of the cloud of points that defines the radius of support for calculation of the weighting function. In literature, the construction of the point cloud is performed with various criteria with its algorithms (or strategies). Different strategies constitute the third concept or abstraction to consider. So the two concepts have great coupling, it is important the possibility of using the same cloud with different algorithms and would be interesting expand the number of algorithms. In computing object-oriented the construction, which decouples objects (classes), allowing multiple algorithms in one of these, is known as the Strategy design pattern (Gamma *et al.*, 2000). Thus Figure 8 shows the **Cloud** class which allows the construction of nodal point clouds and getting its data such as radius of support, the set of nodal points (cloud) for a given point in the domain considering different strategies.

The connection between the class **Cloud**, which builds the clouds, and algorithms for construction is done through the interface **CloudStrategy**. This interface defines the protocol that allows interaction between the class **Cloud** and the various concrete algorithms implemented. The operation **computeStrategy()**, which must be present in all concrete strategies, is who runs the algorithm actually. As noted in Figure 8 the operation **buildCloud()** in class **Cloud** runs the algorithm corresponding to the current strategy.

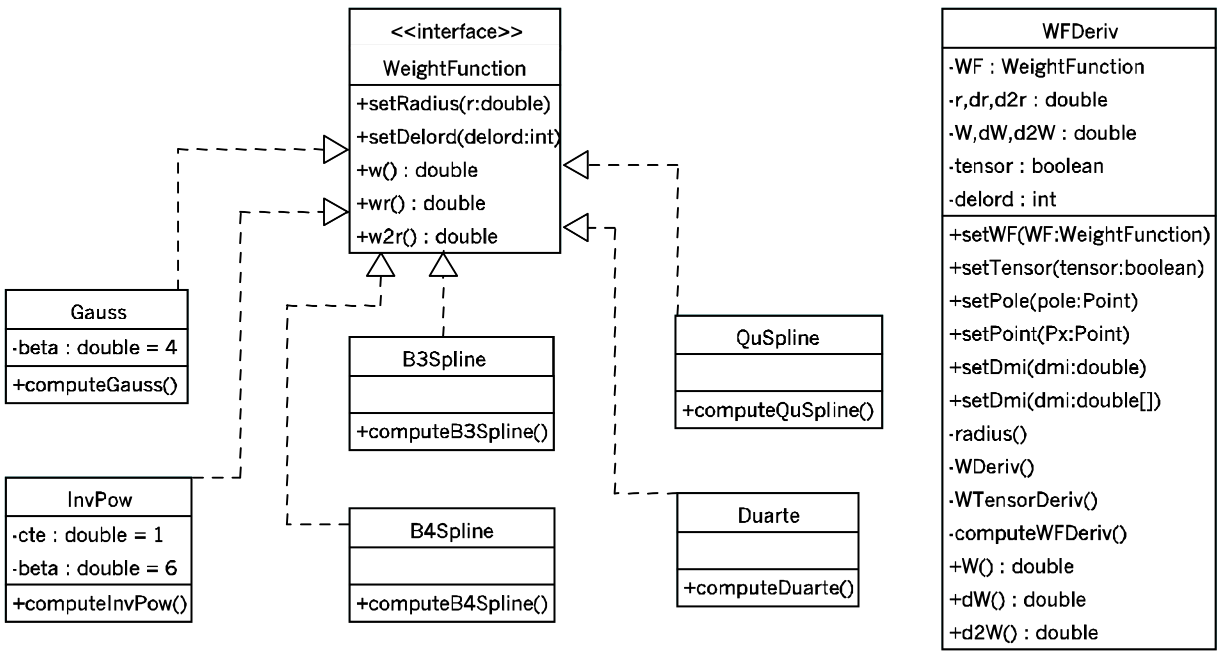


Figure 7: Model classes for weighting functions.

Figure 8 presents five strategies for setting the point cloud. **Distance** determines the cloud selecting *m* nodal points closer to the pole, where *m* is the size of the basis functions used. Due to problems of numerical instability the support of the cloud is increased by an empirical parameter α. The criterion **FixedSupport** determines the cloud of point starting from a predetermined value for the size of the support of the cloud. Both criteria allow determining the cloud with a rectangular support for the use of weighting functions obtained by tensor product. The third criterion presented, **Cell**, is actually a special feature to obtain the set of points of integration present in a cell of integration, or if is the case in a finite element.

Other criteria can be easily implemented without changing the existing structure, for example, a criterion that determines the cloud from a fixed number of nodal points, which is a procedure similar to finite differences. The criterion **FourQuadrants** is widely used in the generation of clouds with symmetry by both Generalized Finite Difference Method and the Finite Point Method. The strategy **MultSupport** is employed in the construction of point clouds in Hp-Clouds Method.

5 CONCLUSIONS

The use of object-oriented analysis, more than a computational tool is an important methodology for addressing complex problems. The deduction of some Meshfree Methods based on the Method of Weighted Residuals (Sánchez et al., 2004) and hence the construction of a common framework is result from the principle of seeking the highest level of abstraction in the modeling of a phenomenon. In this way the computational classes were proposed for modeling the shape functions for Meshfree Methods.

It is believed that a system of computational classes, studied and tested will help consolidate the practical application of Meshfree Methods agilizing the development of robust software. The ultimate goal of this work is to foster discussion of analysis and design of computational objects for Meshfree Methods.

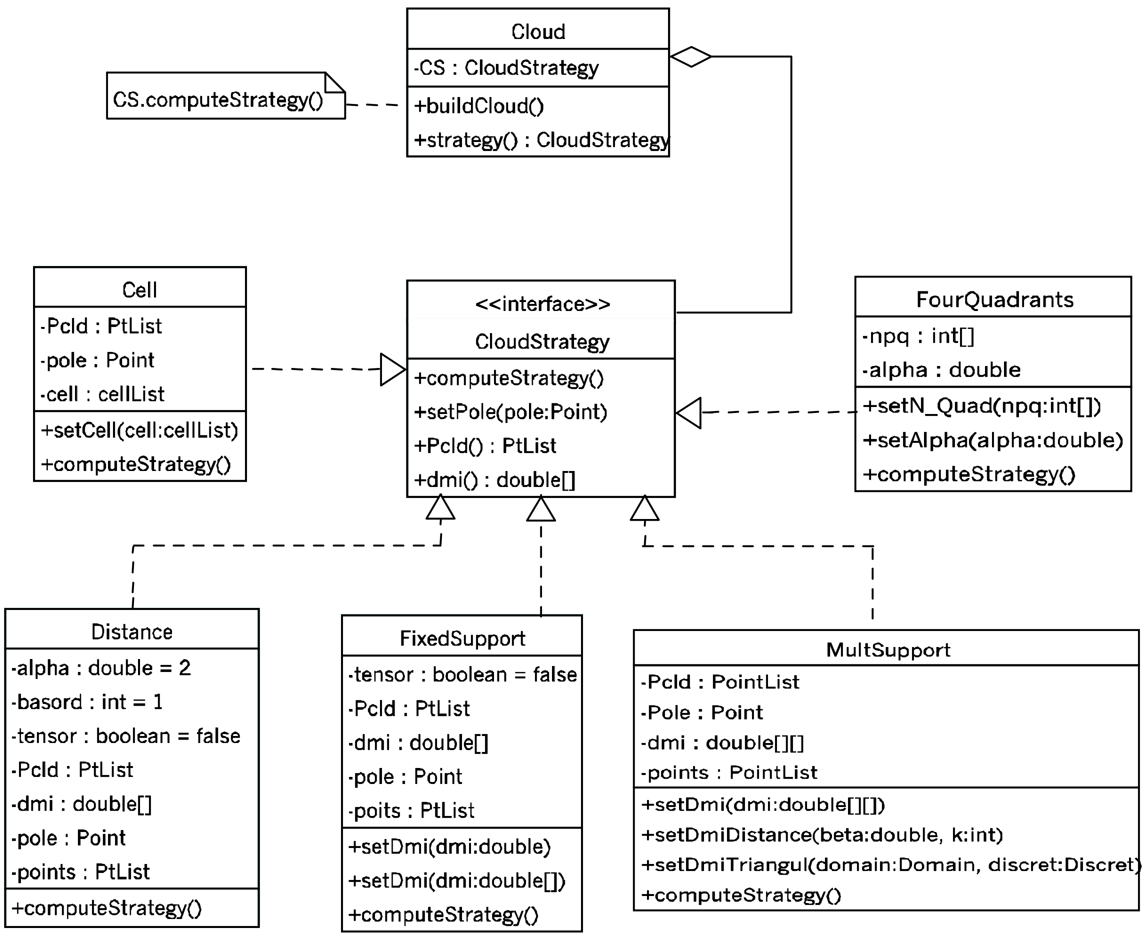


Figure 8: Classes for modeling the cloud of points and the building strategies.

ACKNOWLEDGMENTS

The authors wish to express their gratitude to the Dean for Research and Graduate Studies at the University of Brasilia for support in developing this work. Also to the Brazilian Federal Agency for the Support and Evaluation of Graduate Education (CAPES).

REFERENCES

Babuška, I.; Melenk, J. M. The Partition of Unity Method. *Int. J. Numer. Meth. Engng.,* v. 40, p. 727-758. 1997.

Beissel , S.; Belytschko, T. Nodal Integration of the Element-Free Galerkin Method. *Comput. Methods Appl. Mech. Engrg.*, v. 139, p. 49-74. 1996.

Belytschko, T.; Lu, Y. Y.; Gu, L. Element-Free Galerkin Method. *Int. J. Numer. Meth. Engng.*, v. 37, p. 229-256. 1994.

Belytschko, T., Krongauz, Y., Organ, D., Fleming, M., & Krysl, P. Meshless Methods : An Overview and Recent Developments. *Comput. Methods Appl. Mech. Engrg*., v. 139, p. 3-47, 1996.

Dolbow, J., & Belytschko, T. An Introduction to Programming the Meshless Element Free Galerkin Method. *Archives in Computational Mechanics and Engineering*, 1998.

Duarte, C. A., & Oden, J. T. An hp Adaptive Method Using Clouds. *Comput. Methods Appl. Mech. Engrg*., v. 139, p. 237-262, 1996.

Fleming, M., Chu, Y. A., Moran, B., & Belytschko, T. Enriched Element-Free Galerkin Methods for Crack Tip Fields. *Int. J. Numer. Meth. Engng*., v. 40, p. 1483-1504, 1997.

Fowley, M., & Scott, K. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Longman, 2000.

Fries, T.-P., & Matthies, H. G. *Classication and Overview of Meshfree Methods*. Institute of Scientic Computing, Technical University, Braunschweig, Brunswick, Germany, 2004.

Garcia, O.; Fancello, E. A.; Barcellos, C. S.; Duarte, C. A. hp-Clouds in Midlin`s Thick Plate Model. *Int. J. Numer. Meth. Engng*., v. 47, p. 1381-1400, 2000.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented*. Addison-Wesley, 1995.

Gentleware, 2009. Poseidon for UML. Disponível em http://www.gentleware.com/. Acesso em 21 set. 2009.

Lancaster, P., & Šalkauskas, K. Surface Generated by Moving Least Squares Methods. *Math. Comp.*, v. 37, p. 141-158, 1981.

Liszka, T., & Orkisz, J. The Finite Difference Method at Arbitrary Irregular Grids and its Application in Applied Mechanics. *Comp. Structures*, v. 11, p. 83-95, 1980.

Liszka, T., Duarte, C. A., & Tworzydlo, W. W. hp-Meshless cloud method. *Comput. Methods Appl. Mech. Engrg*., v. 139, p. 263-288, 1996.

Oñate, E.; Idelsohn, S.; Zienkiewicz, O. C.; Taylor, R. L. A Finite Point Method in Computational Mechanics. Applications to Convective Tranport and Fluid Flow. *Int. J. Numer. Meth. Engng.,* v. 39, p. 3839-3866, 1996.

Sánchez, J. *Análise Orientada a Objetos de Métodos Numéricos de Discretização sem Malha*. Tese de Doutorado, Universidade de Brasília – UnB, Brasil, 2003.

Sánchez, J., Pulino Filho, A., & Gonçalves, P. Formulação dos métodos sem malha através do método de resíduos ponderados. *In CILAMCE - XXV Iberian Latin-American Congress on Computational Methods in Engineering*., Recife, PE, Brasil, 2004.

Sánchez, J., Pulino Filho, A., & Gonçalves, P. Análise Orientada a Objetos de Métodos Numéricos de Discretização sem Malha. *In CILAMCE - XXV Iberian Latin-American Congress on Computational Methods in Engineering*., Recife, PE, Brasil, 2004

Suleau, S.; Deraemaeker, A.; Bouillard, P. Dispersion and Pollution of Meshless Solution for the Helmholtz Equation. *Comput. Methods Appl. Mech. Engrg*., v.190, p. 639-657, 2000.

Zienkiewicz, O. C., & Taylor, R. L., 1989. *The Finite Element Method.* MacGraw-Hill, vol. 1.