Mecánica Computacional Vol. XXI, pp. 1414–1432
S. R. Idelsohn, V. E. Sonzogni and A. Cardona (Eds.)
Santa Fe–Paraná, Argentina, October 2002

ACCURATE COMPUTATIONS ON UNSTRUCTURE3D MESHES

Enrique Pardo

Facultad de Ingeniería, Universidad de Mar del Plata, Argentina
e-mail: epardo@fi.mdp.edu.ar

Key words: meshless methods, blurred derivatives, unstructured meshes

Abstract: *The need to perform computations on irregularly distributed nets of nodes arises in many applications of solid and fluid computational mechanics. This is specially problematic in three dimensions. Typically, the finite element method with tetrahedral elements is used for such purpose. However, this poses a number of problems. On one hand some elements are considerably distorted – with eventually some null-volume elements – leading to poor solutions. Also, in this method only h-refinement is feasible so that solution improvement demands to refine the mesh. In this work we describe a meshless method which we designate as Functional Integral Method (FIM) based on the use of blurred derivatives, that allows to overcome the above mentioned difficulties. The method only requires the connectivity of each node given by first neighbors (Voronoi cells) for discretization yielding the same structure of non-zeros as FEM with tetrahedral elements. The matrix is nevertheless non-symmetric so that storage and solution of the linear system increases by a factor close to two. However, results of several numerical simulations indicate that the error is systematically much smaller than with FEM and it is rather insensitive to node irregularity so that relation cost-benefit is finally enhanced substantially. Also, it allows to perform p-refinement in a trivial manner by just adding more neighbors to the local cloud of each node thus increasing the order of interpolation. In this way the error can be further reduced without re-meshing*

.

1 INTRODUCTION

The development of meshless method during the last decade were mainly fueled by the need to overcome the difficulties posed by mesh generation (i.e. partitions of the domain) in three dimensional problems. Some of the methods developed so far are the Diffuse Element[1] Method and the Element Free Galerkin Method[2] (which use MLS interpolation), hp-Clouds[3], Smooth Particle Hydrodynamics[4], Reproducing Kernel Particle Methods[5], Natural Element Method[6], Generalized Finite Differences[7], etc. However, it is not yet clear that they are generally advantageous over the robust and versatile finite element method. Even though these methods do not require domain discretization into elements, this potentiality not always leads to better performance. Some of these methods – specifically those based on weak formulations – still require a partition of the domain for integration. Also, they are built upon rather complex approximations which demand intensive computation and frequently lead to poor conditioned matrixes. An alternative is to start from a strong formulation which does note require integration over the domain so that a truly meshless method is obtained. However, the precision of these methods is low although research continues to improve them.

In any case, even if a meshless point method does not require a mesh it still requires to specify, for each point (node), which are those linked to it that should be used to build the local approximation. This set, whose construction is far from trivial will be termed the *local cloud*. Very simple criterions – such as selecting the nodes of the cloud according to their distances – lead to poor approximation. On the other extreme, there are well established techniques such as Delauney triangulation which provide an adequate set of neighbors for each node while at the same time discretize the domain into triangles. Hence, a crucial test for a finite point method is to compare its performance with linear finite elements.

Recently, it was shown that some linear problems – elliptic, parabolic and hyperbolic - can be formulated in terms of functional integrals[8-10] instead of the classical strong and weak formulations, leading to novel methods which combine simplicity and good accuracy. The most appealing feature of this method – apart from its good performance – is the fact that unlike traditional formulations it does not require calculation of derivatives of any order of the shape functions. It is shown that a very simple polynomial interpolation provides excelent numerical results both in two and three space dimension. This is demonstrated by solving Poisson equation on a number of irregular nets of nodes, where its performance is compared with linear finite elements.

2 BLURRED DERIVATIVES

2.1 Definition and properties

In this section we redefine the derivative operation in order to devise an adequate tool for approximation of differential equations. It starts with a simple observation: the value of a continuous function at a point can be evaluated as:

$$f(x) = \lim_{\varepsilon\to 0} \int_{-\infty}^{\infty} P^0(\bar{x} - x, \varepsilon)\, f(\bar{x})\, d\bar{x} \tag{1}$$

where the operator P^0 is a Gaussian:

$$P^0(\bar{x} - x, \varepsilon) = \frac{e^{-\frac{(\bar{x}-x)^2}{\varepsilon^2}}}{\sqrt{\pi}\,\varepsilon} \tag{2}$$

Equation (1) is a direct consequence of the definition of the Dirac delta function, which can be expressed as the limit of the Gaussian (2). From what it follows that the derivatives of *f(x)* can be computed as:

$$\frac{d^n f(x)}{dx^n} = \lim_{\varepsilon\to 0} \int_{-\infty}^{\infty} \frac{d^n P^0(\bar{x} - x, \varepsilon)}{dx^n} f(\bar{x})\, d\bar{x} = \lim_{\varepsilon\to 0} \int_{-\infty}^{\infty} P^0(\bar{x} - x, \varepsilon)\, f(\bar{x})\, d\bar{x} \tag{3}$$

The functions P^n in the right side of (3), which are the derivatives of P^0, can be obtained from Rodrigues formula for Hermite polynomials[12]: $\frac{d^n e^{-x^2}}{dx_n} = (-1)^n e^{-x^2} H^n(x)$.

Hence, letting $\lambda = \frac{\bar{x} - x}{\varepsilon}$ we have:

$$P^n(\bar{x} - x, \varepsilon) = \frac{d^n P^0(\bar{x} - x, \varepsilon)}{dx^n} = \left[\frac{d^n P^0(\lambda)}{d\lambda^n}\right]\left(\frac{d\lambda}{dx}\right)^n = \left[(-1)^n P^0(\lambda) H^n(\lambda)\right]\left(\frac{-1}{\varepsilon}\right)^n = \left(\frac{1}{\varepsilon}\right)^n P^0(\bar{x} - x, \varepsilon)\, H^n(\tfrac{\bar{x}-x}{\varepsilon}) \tag{4}$$

Where $H^n(x)$ are Hermite polynomials. These are orthogonal polynomials with definite parity:

$$\int_{-\infty}^{\infty} e^{-x^2} H^n(x) H^m(x)\, dx = 0 \qquad n \neq m \quad ; \qquad H^n(x) = (-1)^n H^n(x) \tag{5}$$

The first five polynomials are:

$$H^0(x) = 1 \quad ; \; H^1(x) = 2x \quad ; \quad H^2(x) = 4x^2 - 2 \tag{6}$$
$$H^3(x) = 8x^3 - 12x \quad ; \quad H^4(x) = 16x^4 - 48x^2 + 12$$

The functions P^n can be regarded as operators: after integration they transform a given function, *f(x)*, into a new one, f_ε^1, which in the limit $\varepsilon \to 0$ yields the derivative of the former one:

$$\frac{df(x)}{dx} \equiv f^1(x) = \lim_{\varepsilon\to 0} f_\varepsilon^1(x) \tag{7}$$

At this point it becomes convenient to introduce some notation and designations for further reference. Hence, we will call P^n the n-th *derivative kernel*, and the transformed function

f_ε^n the *blurred derivative* of order *n*. From the preceding equations it follows – and this can be checked by direct computation – that derivative kernels of higher order can be obtained from lower order ones through convolution:

$$P^{\alpha+\beta}(\bar{x}-x,\varepsilon) = \int_{-\infty}^{\infty} P^{\alpha}(\bar{x}-x_1,\varepsilon)\, P^{\beta}(x_1-x,\varepsilon)\, dx_1 \tag{8}$$

The support of the basic operator P^0 defined in formula (2) is infinite. For this reason it seems inadequate to tackle numerical solution of differential equations on finite bodies. But it will be shown in section 5 that the size of parameter ε which gives the best numerical results is small, so that the Gaussian becomes negligible beyond the first neighbor to a given node. However, it is clear from the definitions (1) - (3) that many other functions can be used to generate families of blurred derivative kernels. In particular, functions of local support seem more appropriate. The simplest choice is the use of a truncated Gaussian as the basic operator P^0 but it has a drawback: its derivatives include a term accounting for the sudden change at the boundary of the support. To avoid it the domain of integration of formulas (1) and (3) should reduce to the support of these functions. An alternative are functions of the form:

$$P^0(x) = \begin{cases} \left(x^2-1\right)^n \; if \; x \le 1 \\ 0 \quad elsewhere \end{cases} \tag{9}$$

The exponent "*n*" can be chosen so that derivatives up to the desired order are zero on the boundary of the support. In particular, the derivative of order "n" satisfies Rodrigues formula for Legendre polynomials, $P_n(x)$, namely:

$$P_n(x) = \frac{1}{2^n n!}\,\frac{d^n\left(x^2-1\right)^n}{dx^n} \tag{10}$$

But in this work we will use the Gaussian kernels defined above. They are easy to handle and have a number of properties that allow some clean demonstrations that will be very useful for our purposes. Also, they can be accepted on interpretational grounds as discussed below.

2.2 Connection with SPH

Equations (1) and (3) are the starting point of the Smoothed Particle Hydrodynamics (SPH) method[4]. Hence, it appears that SPH and the method to be described in this paper (which is designated as FIM for reasons that will become clear in the sequel) are closely related. However, it is important to emphasize that the relationship between both methods lies entirely in the point of departure: the way in which these equations are used and the final discrete equations are completely different. Briefly, in SPH equation (1) is used around each node to approximate locally the unknown function whose computational estimate is sought. This is fed into the governing differential equation yielding a discrete equation for each node. In the present paper, on the contrary, the equations of the preceding sub-section will be used to re-express the differential equation as an equivalent functional integral. To approximate the unknown field locally around each node we will use simple polynomials.

In the applications of the blurred derivative to numerical methods, to be described in the next section, we will need a few results and properties of the blurred derivative which are listed below.

2.3 Blurred derivative of a step function

Let $U_{(x)}$ be a step function: U(x) = a if x<0, U(x) = b if x>0. The blurred derivative is

$$U_{\varepsilon}^{1}(x) = a\int_{-\infty}^{0} P^{1}(\bar{x}-x,\varepsilon)\,d\bar{x} + b\int_{0}^{\infty} P^{1}(\bar{x}-x,\varepsilon)\,d\bar{x} \tag{11}$$

But

$$P^{1}(\bar{x}-x,\varepsilon) = \frac{dP^{0}(\bar{x}-x,\varepsilon)}{dx} = -\frac{dP^{0}(\bar{x}-x,\varepsilon)}{d\bar{x}} \tag{12}$$

Hence:

$$U_{\varepsilon}^{1}(x) = -aP^{0}(x,\varepsilon) + bP^{0}(x,\varepsilon) = (b-a)\frac{e^{-\frac{x^{2}}{\varepsilon^{2}}}}{\sqrt{\pi}\varepsilon} \tag{13}$$

So that in the limit:

$$U^{1}(x) = \lim_{\varepsilon\to 0} U_{\varepsilon}^{1}(x) = (b-a)\delta(x) \tag{14}$$

This first simple example shows why we adopted the designation “blurred derivative”: it smears out discontinuities of derivatives for finite ε. This is a desirable property for numerical solution of differential equations, which allows the use of shape functions of low degree of continuity.

2.4 Blurred derivative of a piece-wise linear function

Let L(x) = ax if x<0, L(x) = bx if x>0. Its blurred derivate is:

$$\begin{gathered} L_{\varepsilon}^{1}(x) = a\int_{-\infty}^{0} P^{1}(\bar{x}-x,\varepsilon)\,\bar{x}\,d\bar{x} + b\int_{0}^{\infty} P^{1}(\bar{x}-x,\varepsilon)\,\bar{x}\,d\bar{x} = \\ a\varepsilon\int_{-\infty}^{0} P^{1}(\bar{x}-x,\varepsilon)\frac{(\bar{x}-x)}{\varepsilon}d\bar{x} + b\varepsilon\int_{0}^{\infty} P_{(}^{1}(\bar{x}-x,\varepsilon)\frac{(\bar{x}-x)}{\varepsilon}d\bar{x} + ax\int_{-\infty}^{0} P^{1}(\bar{x}-x,\varepsilon)\,d\bar{x} + bx\int_{0}^{\infty} P^{1}(\bar{x}-x,\varepsilon)\,d\bar{x} \end{gathered} \tag{15}$$

The last two integrals in (15) were already evaluated above. In order to compute the other two integrals we rewrite them in terms of Hermite polynomials. Taking into account formulas (6) we have:

$$2\lambda^{2} = H^{0}(\lambda) + \frac{1}{2}H^{2}(\lambda) \tag{16}$$

So that:

$$P_{(\lambda)}^{1}\lambda = \frac{1}{\varepsilon}P^{0}(\lambda)2\lambda^{2} = \frac{1}{\varepsilon}P^{0}(\lambda) + \frac{1}{2}\varepsilon P^{2}(\lambda) \tag{17}$$

Replacing (17) in (15) and considering that an equation similar to (10) applies to P^2 we have:

$$P^2(\bar{x}-x,\varepsilon)=\frac{dP^1(\bar{x}-x,\varepsilon)}{dx}=-\frac{dP^1(\bar{x}-x,\varepsilon)}{d\bar{x}} \tag{18}$$

The final result is:

$$L_\varepsilon^1(x)=\frac{(a+b)}{2}+\frac{(b-a)}{2}erf\left(\frac{x}{\varepsilon}\right) \tag{19}$$

The extreme values of the error function are *erf(0)=0* and $erf(\pm\infty)=\pm 1$, so that the limit of equation (19) is:

$$L^1(x)=\lim_{\varepsilon\to 0}L_\varepsilon^1(x)=\begin{cases} a & if \quad x<0 \\ \dfrac{(a+b)}{2} & if \quad x=0 \\ b & if \quad x>0 \end{cases} \tag{20}$$

2.5 Blurred derivative of polynomials

Let $q(x)=x^2$. The first blurred derivative is:

$$q_\varepsilon^1(x)=\int_{-\infty}^{\infty} P^1(\bar{x}-x,\varepsilon)\,\bar{x}^2\,d\bar{x} \tag{21}$$

Rearranging $\bar{x}^2=(\bar{x}-x)^2+2x(\bar{x}-x)+x^2$, and changing the integration variable $\bar{x}\to(\bar{x}-x)$ in (21) we have:

$$q_\varepsilon^1(x)=q^1(x)=2x \tag{22}$$

Similarly, the second blurred derivative of *q(x)* turns out to be

$$q_\varepsilon^2(x)=q^2(x)=2 \tag{23}$$

This simple relationship is not valid for higher degree monomials. For instance for degree three, $f(x)=x^3$, we have:

$$f_\varepsilon^1(x)=\int_{-\infty}^{\infty} P^1(\bar{x}-x,\varepsilon)\,\bar{x}^3\,d\bar{x}=3x^2+\frac{3}{2}\varepsilon^2\neq f^1(x) \tag{24}$$

2.6 Family of derivative kernels

The set of derivative kernels given by formulas (2) – (4) is not unique. Moreover, the set just given can be considered as the first generation of a whole family. The next generation starts with a new derivative kernel of order zero:

$$ {}^{2}P^{0}(\bar{x}-x,\varepsilon)=\frac{e^{-\frac{(\bar{x}-x)^{2}}{\varepsilon^{2}}}}{\sqrt{\pi}\varepsilon}\left[\frac{3}{2}-\frac{(\bar{x}-x)^{2}}{\varepsilon^{2}}\right]=P^{0}(\bar{x}-x,\varepsilon)\ {}^{2}H^{0}\left(\frac{(\bar{x}-x)}{\varepsilon}\right) \tag{25} $$

The super-index "2" to the left indicates that it belongs to the second generation. In referring to the first generation (eqs. (2)-(8)) however, we will omit the super-index "1" to the left to avoid overload notation in oncoming formulas. The complete set of new derivative kernels is generated by differentiation of ${}^{2}P^{0}(\bar{x}-x,\varepsilon)$ as in formula (4). The polynomials in this new set , ${}^{2}H^{n}(\lambda)$, are still orthogonal among themselves but are no longer Hermite's. However, all of them can be expressed as linear combinations of Hermite polynomials. The following generations of this family are produced in a similar manner, starting with zero order kernels which contain polynomial factors of increasing degree.

An interesting property of this family is that kernels of different generations can eventually be combined to yield kernels of these generations but of different degree. We illustrate this property with two simple examples that will be very useful in the sequel. The first generation zero order kernel can be written as linear combination of two kernels of the second generation

$$ P^{0}(\bar{x}-x,\varepsilon)={}^{2}P^{0}(\bar{x}-x,\varepsilon)+\frac{\varepsilon^{2}}{4}P^{2}(\bar{x}-x,\varepsilon) \tag{26} $$

Similarly, the first order kernel can be expressed as:

$$ P^{1}(\bar{x}-x,\varepsilon)={}^{2}P^{1}(\bar{x}-x,\varepsilon)+\frac{\varepsilon^{2}}{4}P^{3}(\bar{x}-x,\varepsilon) \tag{27} $$

The set of kernels ${}^{i}P^{n}$ are said to form a *family* because all of them originate from the Gaussian (2). The Gaussian family is of class C^{∞}- moreover, kernels of any order are finite – and is easy to manipulate.

A thorough discussion of the application of this methodology to one dimensional differential equations including non-linearities, in which several approximations are analyzed and compared to standard numerical methods is given elsewhere[11]. In this paper we will concentrate on its application to several space dimensions and specifically to Poisson equation.

3 SEVERAL SPACE DIMENSIONS

Generalization of all the preceding equations to the multidimensional case is straightforward. Consider for instance a function of three variables, *f(x,y,z)*. The fundamental formula (1) takes now the form:

$$ f(x,y,z)=\lim_{\varepsilon\to 0}\int_{-\infty}^{\infty} P^{0}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)\,f_{(\bar{\mathbf{r}})}\,d\bar{V} \tag{28} $$

where the operator P^{0} is now:

$$P^0(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)=\frac{e^{-\frac{(\bar{\mathbf{r}}-\mathbf{r})^2}{\varepsilon^2}}}{\left(\sqrt{\pi\varepsilon}\right)^3}\ ;\ \bar{\mathbf{r}}=(\bar{x},\bar{y},\bar{z})^t\ ;\ \mathbf{r}=(x,y,z)^t\ ;\ d\bar{V}=d\bar{x}\,d\bar{y}\,d\bar{z} \tag{29}$$

Notice that the exponent in the denominator of the zero order kernel, P^0 , (which is 3 in this case) equals the number of independent variables of the function *f*. This stems from the fact that P^0 is built simply as the product of the zero order kernels of the independent variables. From this, partial derivatives are obtained with the same procedure as (3) and (4). For instance, the n-th order partial derivative with respect to *x* is:

$$\frac{\partial^n f(\mathbf{r})}{\partial x^n}=\lim_{\varepsilon\to 0}\int_{-\infty}^{\infty}\frac{\partial^n P^0(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)}{\partial x^n}f(\bar{\mathbf{r}})\,d\bar{V}=\lim_{\varepsilon\to 0}\int_{-\infty}^{\infty}P^{n,x}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)f_{(\bar{\mathbf{r}})}\,d\bar{V} \tag{30}$$

where:

$$P^{n,x}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)=\left(\frac{1}{\varepsilon}\right)^n P^0(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)H^n\left(\frac{\bar{x}-x}{\varepsilon}\right) \tag{31}$$

From the preceding it is easy to compute the usual vector operators. In this way, the gradient kernel is just:

$$\mathbf{P}^1(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)=\left(\frac{1}{\varepsilon}\right)P^0(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)2(\bar{\mathbf{r}}-\mathbf{r})=\left(P^{1,x}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon),P^{1,y}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon),P^{1,z}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)\right)^t \tag{32}$$

So that the gradient of a scalar field is:

$$\vec{\nabla}f(\mathbf{r})=\lim_{\varepsilon\to 0}\vec{\nabla}\varepsilon\, f(\mathbf{r})=\lim_{\varepsilon\to 0}\int\mathbf{P}^1(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)f(\bar{\mathbf{r}})\,d\bar{V} \tag{33}$$

The Laplacian, in turn, can be built either as the sum of the second partial derivatives or as the convolution of the divergence and gradient operator (33) as indicated in formula (8). The result now depends on the number of independent variables:

$$\nabla^2 f(\mathbf{r})=\lim_{\varepsilon\to 0}\nabla^2_{\varepsilon}f(\mathbf{r})=$$
$$\lim_{\varepsilon\to 0}\begin{cases}\dfrac{4}{\varepsilon^2}\displaystyle\int P^0(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)\left[(\bar{\mathbf{r}}-\mathbf{r})^2-1\right]f(\bar{\mathbf{r}})\,d\bar{V} & ,in\ 2-D\\ \dfrac{4}{\varepsilon^2}\displaystyle\int P^0(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)\left[(\bar{\mathbf{r}}-\mathbf{r})^2-\frac{3}{2}\right]f(\bar{\mathbf{r}})\,d\bar{V} & ,in\ 3-D\end{cases} \tag{34}$$

As in one dimension, also in the multidimensional case the zero order kernel P^0 given by (29) gives rise to the first generation of a whole family of operators. For instance in two dimensions the second generation of this family starts with:

$$ {}^{2}P^{0}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)=\frac{e^{-\frac{(\bar{\mathbf{r}}-\mathbf{r})^{2}}{\varepsilon^{2}}}}{\left(\sqrt{\pi\varepsilon}\right)^{2}}\left[2-\frac{(\bar{\mathbf{r}}-\mathbf{r})^{2}}{\varepsilon^{2}}\right] \quad ; \quad \bar{\mathbf{r}}=(\bar{x},\bar{y})^{t} \quad ; \quad \mathbf{r}=(x,y)^{t} \qquad (35) $$

An important consequence of the preceding is that there exists the same relation among generations that was pointed out for the one-dimensional case in section *2.6*. As an example, it can be verified that the following holds:

$$ P^{0}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)={}^{2}P^{0}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)+\frac{\varepsilon^{2}}{4}\nabla^{2}_{\varepsilon}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon) \qquad (36) $$

The relevance of relation (36) – which is a generalization of the one dimensional equation (26) - lies in that it allows to formulate the functional integral for any number of space dimensions. We describe it in detail because it will be used in the numerical examples. Consider again the diffusion equation, but now with two or three space dimensions:

$$ \frac{\partial u(\mathbf{r},t)}{\partial t}=\kappa\nabla^{2}u(\mathbf{r},t)+g(\mathbf{r}) \qquad (37) $$

We now replace both members of (37) by their respective blurred derivatives:

$$ \int_{-\infty}^{\infty} P^{0}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)P^{1}(\bar{t}-t,\delta)\,u(\bar{x},\bar{t})\,d\bar{t}d\bar{V} = $$
$$ \int_{-\infty}^{\infty} \left[\kappa\nabla^{2}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)P^{0}(\bar{t}-t,\delta)u(\bar{x},\bar{t})+P^{0}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)P^{1}(\bar{t}-t,\delta)g(\bar{\mathbf{r}}-\mathbf{r})\right]d\bar{t}d\bar{V} \qquad (38) $$

Notice that in (38) we have used the correct definition of partial blurred derivative. However, after integration over the space variables, $d\bar{V}=d\bar{x}.d\bar{y}$ in the left member of (38), and over time, $d\bar{t}$, in the right member the following expression is obtained:

$$ \int_{-\infty}^{\infty} P^{1}(\bar{t}-t,\delta)\,u(x,\bar{t})\,d\bar{t}=\int_{-\infty}^{\infty}\nabla^{2}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)\,u(\bar{x},t)\,d\bar{V}+\int_{-\infty}^{\infty}P^{0}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)\,g(\bar{x})\,d\bar{V} \qquad (39) $$

Next, the local variation of time is approximated linearly so that:

$$ u(\mathbf{r},t+\delta)=\int_{-\infty}^{\infty}\left(P^{0}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)+\delta\kappa\nabla^{2}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)\right)u(\bar{\mathbf{r}},t)\,d\bar{V}+\delta\int_{-\infty}^{\infty}P^{0}(\bar{\mathbf{r}}-\mathbf{r},\varepsilon)\,g(\bar{\mathbf{r}})\,d\bar{V} \qquad (40) $$

We then replace P^{0} by ${}^{2}P^{0}$, take $\delta\kappa=\varepsilon^{2}/4$ and use formula (36) in the first integral of (40) to obtain:

$$ u(\mathbf{r},t+\delta)\cong\int_{-\infty}^{\infty}\frac{e^{-\frac{(\bar{\mathbf{r}}-\mathbf{r})^{2}}{4\kappa\delta}}}{\left(\sqrt{4\pi\kappa\delta}\right)^{n}}u(\bar{\mathbf{r}},t)\,d\bar{V}+\delta\int_{-\infty}^{\infty}\frac{e^{-\frac{(\bar{\mathbf{r}}-\mathbf{r})^{2}}{4\kappa\delta}}}{\left(\sqrt{4\pi\kappa\delta}\right)^{n}}g(\bar{\mathbf{r}})\,d\bar{V} \qquad (41) $$

The exponent “n” in the denominators is the number of space dimensions. Ee now concentrate in the application of (41) for the numerical solution of Poisson equation. For this, we assume again that the diffusion process has already taken place and a stationary state has been reached. Assuming for simplicity that the coefficient κ equals 1 equation (41) leads to:

$$u(\mathbf{r}) \cong \int_{-\infty}^{\infty} \frac{e^{-\frac{(\bar{\mathbf{r}}-\mathbf{r})^2}{4\delta}}}{\left(\sqrt{4\pi\delta}\right)^n} u(\bar{\mathbf{r}})\, d\bar{V} + \delta \int_{-\infty}^{\infty} \frac{e^{-\frac{(\bar{\mathbf{r}}-\mathbf{r})^2}{4\delta}}}{\left(\sqrt{4\pi\delta}\right)^n} g(\bar{\mathbf{r}})\, d\bar{V} \tag{42}$$

Where the source term has been divided by κ.

4 NUMERICAL IMPLEMENTATION OF POISSON EQUATION

Equation (42) can be discretized in many different manners. One of the most convenient ones is to fit a *local* polynomial for each internal node, whose coefficients relate the unknown at the current node with values on some number, M, of neighboring nodes. Consider for brevity a quadratic polynomial:

$$\tilde{u}_{(x,y)} = \mathbf{a}.\mathbf{R}_{(x,y)} \tag{43}$$

where

$$\mathbf{R}_{(x,y)} = (1, x, y, x^2, xy, y^2)^t \quad ; \quad \mathbf{a} = [a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6]^t \tag{44}$$

The vector of coefficients **a** is calculated in terms of the M nodal values of $\mathbf{u}$ solving[13]:

$$\mathbf{a} = (\mathbf{V}^{\mathbf{t}}.\mathbf{V})^{-1}.\mathbf{V}^{\mathbf{t}}.\mathbf{u} \tag{45}$$

where

$$\mathbf{u} = (u_0 \quad u_1 \quad \cdot \quad \cdot \quad u_M)^t \quad ; \quad \mathbf{V} = \begin{bmatrix} 1 & x_0 & y_0 & x_0^2 & x_0 y_0 & y_0^2 \\ 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ \cdot & & & & & \cdot \\ \cdot & & & & & \cdot \\ 1 & x_M & y_M & x_M^2 & x_M y_M & y_M^2 \end{bmatrix} \tag{46}$$

The matrix $(\mathbf{V^tV})$ of equation (45) is very poorly conditioned, a fact that may deleteriously influence numerical result. Hence, a very convenient alternative is to directly compute the pseudo-inverse of matrix **V** using singular value decomposition. Numerical results to be discussed in the next section confirm that this is indeed the case.

In equation (46) a local numbering has been used, where the index of the current node is “0” and its neighbors range from 1 to M. Replacing approximation (43) for node “i” in formula (42) yields:

$$u_i = a_1^u + 2\kappa\varepsilon(a_4^u + a_5^u) + \varepsilon\left(a_1^g + 2\kappa\varepsilon(a_4^g + a_5^g)\right) \tag{47}$$

The super-index "*u*" in coefficients a_j of equation (47), indicates that those coefficients are linear combinations of nodal values of the field "*u*" in the local cloud. Similarly, the super-index "*g*" indicates that the combination is with nodal values of source *g(x,y)*. As regards the coefficients themselves, they depend on the number and distribution of the M neighbors in the cloud. As an example, if there are six neighbors forming a regular hexagon the coefficients are:

$$\left.\begin{aligned} a_1 &= u_0 \\ a_4 &= \frac{1}{h^2}\left[-u_0 + \frac{1}{2}(u_1 + u_4)\right] \\ a_6 &= \frac{1}{h^2}\left[-u_0 + \frac{1}{3}(u_2 + u_3 + u_5 + u_6) - \frac{1}{6}(u_1 + u_4)\right] \end{aligned}\right\} \qquad (48)$$

Where the local number of the center node is "0" and its neighbors are numbered from 1 to 6 anti-clockwise as shown in figure 1. Similarly, if there are eight neighbors as shown in figure 2 the coefficients are

$$\left.\begin{aligned} a_1 &= \frac{1}{9}[5u_0 + 2(u_1 + u_3 + u_5 + u_7) - (u_2 + u_4 + u_6 + u_8)] \\ a_4 &= \frac{1}{3h^2}\left[-u_0 + \frac{1}{2}(u_1 + u_2 + u_4 + u_5 + u_6 + u_8) - (u_3 + u_7)\right] \\ a_6 &= \frac{1}{3h^2}\left[-u_0 + \frac{1}{2}(u_2 + u_3 + u_4 + u_6 + u_7 + u_8) - (u_1 + u_5)\right] \end{aligned}\right\} \qquad (49)$$

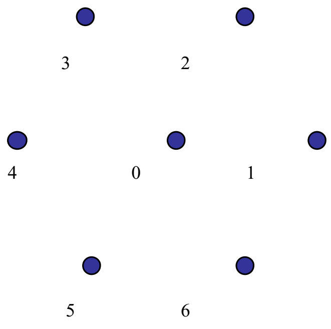

Figure 1: Local cloud with six neighbors

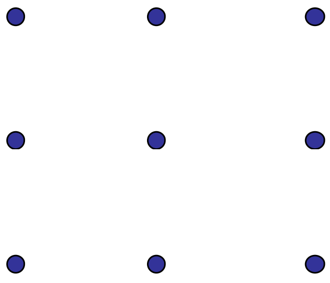

Figure 2: Local cloud with eight neighbors

It is worth mentioning some features of these coefficients that will be relevant in the computational implementation to be described in the next section. In the first place, notice that if the parameter ε is sufficiently large the first two terms of equation (47), u_i y a_1, can be neglected so that the factor ε is cancelled yielding:

$$0 = 2\kappa(a_4^u + a_5^u) + a_1^g \tag{50}$$

In equation (50) it has also been assumed that the source term can be approximated as a constant, a_1^g. This is the same equation that would be obtained with a point collocation scheme using local polynomial approximation or with Generalized Finite Differences. This was already pointed out in the one dimensional case but the result is general. Also, for certain nodal arrays equation (47) coincides with the collocation of GFDM whatever the value of parameter ε. This happens when the first term of the polynomial approximation, a_1, is just the value of the field at the center node, u_0 , as in the case of the hexagonal array given by equations (48). In this case the terms u_i and a_i of (47) are cancelled and (50) is recovered. But this cancellation would not occur if the coefficients (49) – which link the center node to eight neighbors – are used.

Finally, we notice that whatever the nodal array the quadratic coefficients, a_4 y a_6, are inversely proportional to the square of the nodal spacing, h, as can be seen in equations (48-49) (although this is rather obvious from dimensional considerations). The importance of this lies in the following. The discrete equation corresponding to a given node (i.e. its contribution to the global matrix), can be written in the form:

$$u_0(\alpha_0 - 1 + \zeta\beta_0) + u_1(\alpha_1 + \zeta\beta_1) + \cdots + u_M(\alpha_M + \zeta\beta_M) = f_0 \tag{51}$$

Where we have used local numbering and abbreviated the independent term as f_0. In equation (51) we have called:

$$\zeta = \frac{\kappa\varepsilon}{h^2} \tag{52}$$

which is a non-dimensional quantity. The numbers α_i are the coefficients of u_i in the term a_1 of the polynomial fitting. Similarly, the numbers β_i of (51) are the sum of the coefficients of u_i in the terms a_4 and a_5, *divided by the factor* h^2. It can be demonstrated[9] that for each nodal array there exists an optimum value of the non dimensional parameter ζ which minimizes the local discretization error. Hence, once this value is known the optimum "time step" ε is chosen according to the nodal distance with (52). This is crucial for the numerical implementation of the method on irregular meshes as described in the next section.

5 NUMERICAL RESULTS

In this section we describe two and three dimensional tests designed to study the performance of the numerical scheme described above. The potential advantages of a meshless method should become evident in the case of irregular node distributions. Hence, in all simulations we used nodal arrays generated as follows. Starting from a regular array of nodes, each interior point is moved at random a distance proportional to the initial nodal spacing in such a way that boundary nodes remain equally spaced. For the two dimensional simulations we used four sets of 10 meshes each with 5X5 nodes, 9X9, 17X17 and 33X33 nodes. All meshes discretize a unit square. For each model the corresponding Delauney triangulation was calculated, for reasons stated below.

The set of nodes used to fit the polynomial for a given internal node are called its local cloud. The number of nodes in the cloud is the *degree of connectivity* (or *connectivity* for short) of the given node. Figures 1 and 3 represent degrees of connectivity 7 and 9 respectively.

In the first place, the use of Delauney triangulation allows to compare FIM with linear finite elements in equal conditions: in both schemes all interior nodes are linked to the same neighbors – using the same local clouds - and hence produce the same structure of non-zeroes in the stiffness matrix. Second, the triangulation can be used to identify the second neighbors. These can be added to the local cloud in order to fit higher order polynomials. An example is shown in figure 3, where the neighbors of an interior node of a 9X9 mesh are shown together with the triangulation. It can be seen that the connectivity is six if only first neighbors are used, so that a second degree polynomial can be fitted. Adding the second neighbors (identified with dashed lines in figure 3) a third degree polynomial can be fitted, and consequently the convergence rate is enhanced.

In all the meshes used in the present examples connectivities given by first neighbors of Delauney triangulation range from 5 to 9. When the degree is 5 a complete second degree polynomial cannot be fitted. An alternative would be to use an incomplete polynomial (for instance ignoring the monomial $x.y$). But we just added one second neighbor in that case so that the degree of connectivity is never lower than six. A better alternative is to use a complete quadratic polynomial and calculate the pseudo-inverse of the Vandermonde matrix using singular value decomposition. This scheme is called *FIM2* in the numerical examples. Notice that this scheme is not fully equivalent to linear finite elements, because the local cloud is enriched to six in nodes with primary connectivity 5.

In the numerical examples we also used FIM with *cubic approximation*. For this purpose the second neighbors were added to the local cloud as described above. This scheme is called *FIM3* in the numerical examples. In all meshes the degree of connectivity with FIM3 range from 10 to 17. The task of neighbor identification, enrichment and addition of second neighbors is done by a very simple pre-processing program.

As regards selection of the optimum parameter ε for each node we proceeded as follows. We first determined empirically a set of nearly optimum non-dimensional parameters $\zeta_{opt} = \dfrac{\varepsilon_{opt}}{h_{ave}^2}$ for several degrees of connectivity. We did it by simply running a few tests and minimizing the error. The results are shown in Table 1 cloud whatever the degree of connectivity. As an error measure we used the following formula[14]:

$$E = \frac{1}{\left|u^e\right|_{máx}} \sqrt{\frac{1}{NP} \sum_{i=1}^{NP} \left[u_i^e - u_i^{Num}\right]^2} \tag{53}$$

Where NP is the number of mesh points, u^e is the solution and u^{Num} the numerical value.

Finally, for the calculation of partial derivatives at nodes FIM simply evaluates derivatives of the local polynomial approximation. This procedure is coincident with the *superconvergent stress recovery* procedure used in finite element calculations. While in the context of finite elements this is an *ad-hoc* methodology, it is completely natural in FIM. To evaluate the error in derivatives we used formula (107) replacing u by its derivatives. We also used the same procedure to evaluate derivatives with FEM.

5. 1 Two dimensional examples

The first example corresponds to the field:

$$u(x,y) = -x^3 + 3x^2y - y^3 + 3y^2x \qquad (54)$$

This field satisfies Laplace's equation so that the source term vanishes in this case.

All meshes have Dirichlet boundary conditions so the error is calculated at interior nodes. The error for a given number of nodes (and hence for a given mean nodal spacing h) was calculated by averaging the error for the ten meshes of that type.

In this first example we compare FIM with other three methods: GFDM, FEM and the Element Free Galerkin Method (EFG). The latter uses MLS interpolants for trial and test functions with a variational principle. For the latter we used standard MLS shape functions without the orthogonal basis functions. Linear basis functions were used. For integration, the square domain was divided in $\frac{\sqrt{NP}-1}{2}$ cells - where NP is the number of nodes – an a 6X6 Gauss quadrature rule was used. For integration on the boundary we used three node quadratic line ebments with a 6 point Gauss rule. Finally, Gaussian weight functions were used. The support or domain of influence was chosen as $5.5h$ for all nodes, where h is the mean nodal spacing of the mesh, while the constant C that controls the relative weight [18] was set to $2.6h$.

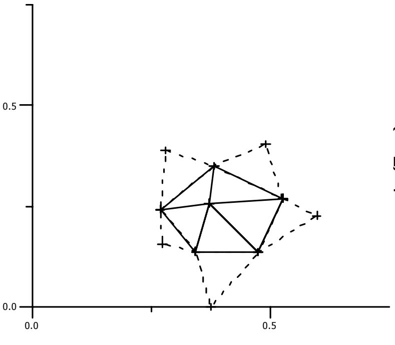

Figure 3: Interior node of figure 5 showing first and some second neighbors.

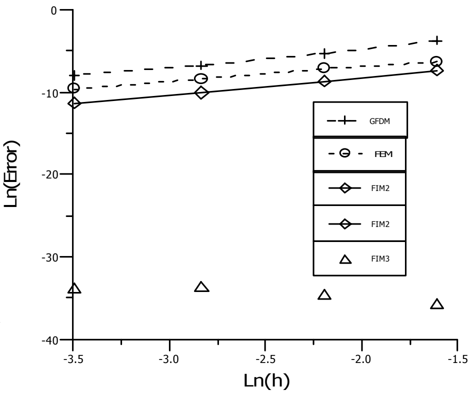

Figure 7: Cubic polynomial. Error with FEM, GFDM, FIM2 and FIM3

Figure 4 shows a plot of error in u versus mean nodal spacing, h, in logarithmic scale. It can be seen that the error is larger with Generalized Finite Differences, followed by linear finite elements (FEM). The smallest error occurs with EFG and FIM2 that are similar. The four regression lines have slope 2 as expected. It is worth pointing out that while comparison of FIM2 with FEM and GFDM is conclusive, comparison with EFG should be taken with caution. This is because FEM and GFDM do not contain free parameters that must be adjusted for better performance, so that results do not depend on implementation details. The performance of EFG, on the other hand, depends on the type and parameters of the weight function, the integration method, and implementation of the MLS shape functions. In particular, use of orthogonal basis functions might enhance performance. For this reasons the present comparison with EFG is indicative of a trend and not a definite result. However, there is an important point concerning the computational cost of both methods. As mentioned above, FIM2 uses only first neighbors for interpolation, so that the connectivity of interior nodes ranges from 6 to 9 in all meshes. This yields a very sparse matrix which is also very well conditioned. In our implementation of EFG, on the other hand, the connectivity of interior nodes is close to 100, the computer cost of matrix inversion for all Gauss point is rather high, and the matrix is somewhat poorly conditioned. By the same token, we do not compare FIM2 with SPH and related methods. This is because the convergence rate of SPH is low and it is less accurate than EFG. Moreover, several correction schemes for SPH (i.e. RKPM) have been compared with EFG and shown to be less accurate[15].

The second two dimensional example corresponds to the field

$$u(x,y) = \sin(\pi x)\sin(\pi y) \tag{55}$$

which is zero on the boundaries. The source term is:

$$g(x,y) = 2\pi^2 \sin(\pi x)\sin(\pi y) \tag{56}$$

Unlike the FEM, the present method allows more versatility in the selection of approximations which can be different for the source and the field. In particular, we noticed that the use of ordinary least squares polynomial approximation of the source term leads to rather poor results. On the other extreme, if the source is assumed to be constant for each node the results are also unsatisfactory. Hence, we resorted to the simple procedure of using a weighted average of both approximations mentioned above. We found empirically that very good results – both with FIM2 and with FIM3 – can be obtained by assigning a weight of 40% to the constant value of the source on the node, g_0 , and 60% to the polynomial approximation. Hence, the nodal equation has the form:

$$u_i = a_1^u + 2\kappa\varepsilon(a_4^u + a_5^u) + \varepsilon\left\{0.4g_0 + 0.6\left[a_1^g + 2\kappa\varepsilon(a_4^g + a_5^g)\right]\right\} \tag{57}$$

Figure 5 shows a plot of error in u versus mean nodal spacing, h, in logarithmic scale. As in the previous example, FIM2 has better precision than FEM, while FIM3 has even better precision and also a higher rate of convergence. Also, the slope of the regression line with FIM2 is slightly higher than two. This might be a consequence of the approximation used for the source, which better captures the local variation as the mesh is refined.

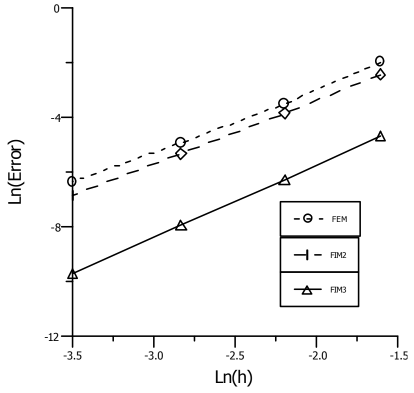

Figure 5: Sine solution. Error with FEM, FIM2 and FIM3.

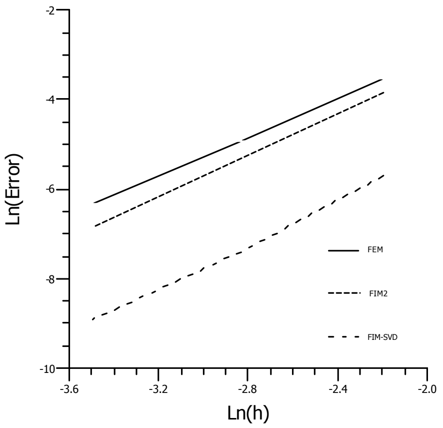

Figure 6: Sine solution. Use of singular value decomposition.

As mentioned in section 4, inversion of matrix (V^tV) may have a deleterious effect on numerical results due to poor conditioning. In fact, if singular value decomposition is used for the case of quadratic interpolation there is a substantial improvement of results as shown in figure 6. Also, in such case it is not necessary to add second neighbors to nodes with primary connectivity less than six. On the contrary, a complete polynomial is used since SVD automatically sets to zero negligible eigenvalues.

5. 2 Three dimensional example

Three dimensional simulations were performed on unit cubes of 5^3, 7^3, 9^3 and 12^3 nodes.
The field

$$u(x,y,z) = \sin(\pi x)\sin(\pi y)\sin(\pi z) \tag{58}$$

was modeled. The mesh of points was discretized into tetrahedra to perform finite element calculations, and the same neighbors were used for FIM using singular value decomposition for calculation of polynomial fitting. Connectivity of interior points range from 10 to 18, except at a few nodes where it is higher. Notice that in three dimensions a complete quadratic polynomial has 10 coefficients while a complete cubic polynomial has 19 coefficients. Hence, a complete cubic polynomial cannot be fitted for all interior nodes. The corresponding FIM scheme is designated as FIM A in figure 7. However, if the Voronoi cells are used to identify first neighbors in the case of a regular array they define a cube around each interior node with connectivity 27. Hence, a complete cubic polynomial can be used with a dramatic increase of precision as shown in the curve designated FIM B of figure 7. Notice in particular that for the finer mesh used in this simulation (12^3 nodes) the error with FIM is less than 1/50 the FEM error. Clearly, the use of the mesheless method offers a substantial advantage in three dimensions.

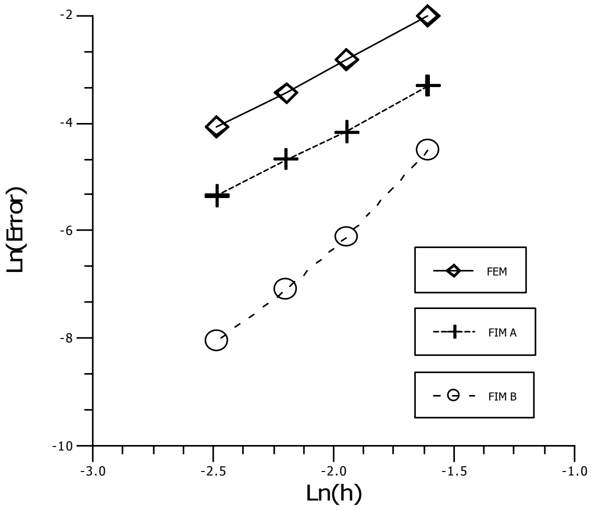

Figure 7: Three dimensional simulation. Comparison of FEM, FIM with the same first neighbors and FIM with higher connectiviy.

6 CONCLUSIONS

It has been shown in this work that the concept of blurred derivatives provides a valuable tool in computational mechanics. On one hand it leads to fully meshless methods, since there is no need of partitioning the domain of interest. Also, it is more versatile than standard strong and weak formulations since it allows a higher degree of flexibility for selection of trial functions: even piecewise constant approximations lead to meaningful discrete equations. Another important byproduct of blurred derivative is that it allows to derive in a simple manner the Functional Integral Method for elliptic problems – of which generalized finite differences is a special case – and to generalize it to non-linear problems. The numerical examples with unstructured meshes indicate that when FIM uses the closest neighbors for quadratic interpolation its precision is higher than FEM. In particular, when polynomial coefficients are calculated using the pseudo-inverse (singular value decomposition) it is possible to attain very high precision. This is specially remarkable in three dimensions.

Finally, there is an interesting consequence of the definition of blurred derivative that might be the source of new developments. As stated in the first pages of this article, blurred derivative kernels are proportional to Hermite polynomials. But each polynomial, $H^n(x)$ can be regarded as the solution of the associated differential equation:

$$\frac{d^2 f(x)}{dx^2} - 2x\frac{df(x)}{dx} + 2n\, f(x) = 0 \tag{59}$$

Hence, if the index "n" of equation (59) is allowed to take real values – not only integers – the so called fractional derivatives are naturally obtained. This definition differs from the most usual ones, which nevertheless are not equivalent among themselves. The solutions of (59) can be expressed in terms of confluent hypergeometric functions[12]. In particular, Riewe[16,17] has shown recently that fractional derivatives allow to formulate Lagrangians and Hamiltonians for non- conservative systems. The concept of blurred derivative might then provide an adequate tool for new numerical implementations of such systems.

7 REFERENCES

[1] B. Nayroles, G. Touzot and P. Villon, "Generalizing the finite element method: diffuse approximation and diffuse elements", *Computational Mechanics*, **10**, 307-318 1992.

[2] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming and P. Krysl, "Meshless methods: an overview and recent developments", *Computer Methods in Applied Mechanics and Engineering,* **139**, 3-47 1996.

[3] C.A. Duarte and J.T. Oden, "H-p adaptive method using clouds", *Computer Methods in Applied Mechanics and Engineering,* **139**, 237-262 1996.

[4] J.J. Monaghan, "An introduction to SPH", *Computer Physics Communications*, **48**, 89-96 1998.

[5] W.K. Liu, S. Jun and Y.F. Zhang, "Reproducing Kernel Particle Methods", *Int. J. Num. Meth. Engng*; **20**: 1081-1106, 1995.

[6] N. Sukumar, B. Moran and T. Belytschko, "The Natural Element Method in Solid Mechanics", *Int. J. Num. Meth. Engng* , **43**: 839-887, 1998.

[7] T. Liszka, "An Interpolation Method for an Irregular Net of Nodes", Int. J. Num. Meth. Engng, **20**, 1599-1612 1984.

[8] E. Pardo, "Meshless method for linear elastostatics based on a path integral formulation", *Int. J. Num. Meth. Engng,* **47**: 1463-1480 2000.

[9] E. Pardo, "Convergence and accuracy of the path integral approach for elastostatics". *Computer Methods in Applied Mechanics and Engineering*, **191 (20**): 2219-2247 2002.

[10] E. Pardo, "Functional integral formulation of classical wave equations", *Journal of Sound and Vibration* (in press).

[11] E. Pardo, "Blurred derivatives and meshless methods", *Int. J. Num. Meth. Engng*, (in press).

[12] Abramowitz M, Stegun IA. Handbook of Mathematical Functions. Dover: New York, 1973.

[13] Lancaster P, Salkauskas K. Curve and Surface Fitting. Academic Press: London, 1986.

[14] N. R. Aluru, "A point collocation method based on reproducing kernel approximations". *Int. J. Num. Meth. Engng*, **47**, 1083-1121 2000.
[15] T. Belytschko, Y. Krongauz, J. Dolbow and C. Gerlach, "On the completeness of meshfree particle Methods", Int. J. Num. Meth. Engng, **43**, 785-819 1998.
[16] F. Riewe, "Nonconservative Lagrangian and Hamiltonian mechanics", *Physical Review E* , **53**, 1890-1899 1996.
[17] F. Riewe, "Mechanics with fractional derivatives", *Physical Review E,* **55**, 3581-3592 1997.