A FINITE POINT METHOD FOR COMPRESSIBLE FLOW

Rainald Löhner, Carlos Sacco, Eugenio Oñate and Sergio Idelssohn

School of Computational Science and Informatics
M.S. 4C7, George Mason University
Fairfax, VA 22030-4444, USA

International Center for Numerical Methods in Engineering
Universidad Politécnica de Catalunya, Edificio C1
Campus Norte, Gran Capitan s/n
08034 Barcelona, Spain

ABSTRACT

A weighted least squares finite point method for compressible flow is formulated. Starting from a global cloud of points, local clouds are constructed using a Delaunay technique with a series of tests for the quality of the resulting approximations. The approximation factors for gradient and the Laplacian of the resulting local clouds are used to derive an edge-based solver that works with approximate Riemann solvers. The results obtained show accuracy comparable to equivalent mesh-based finite volume or finite element techniques, making the present finite point method competitive.

Keywords: Finite Point Methods, Mesh Free Techniques, Compressible Flows, CFD

1. INTRODUCTION

Over the course of the last decade, a number of 'gridless' or 'mesh free' schemes have appeared in the literature (see, e.g. [Nay72, Bat93, Bel94, Dua95, Oña96a, Liu96, Oña98, Oña00] and the references cited therein). The interest in these schemes stems from two main lines of reasoning:

- Even though mesh generation has progressed rapidly over the last decade, there exists a perceived difficulty in generating volume filling grids for problems characterized by complex geometries and/or complex physics. The generation of simply points instead of grids is seen as an easier (and faster) task, and this was shown in [Löh98];
- The construction of higher-order schemes on unstructured grids has encountered severe obstacles in the areas of stability, operation count and storage. To date, most production codes are still based on linear elements, or, equivalently, linear reconstruction procedures. The use of gridless schemes should facilitate the construction of higher order discretizations.

The numerical analysis process for 'gridless', 'mesh free', or 'finite point' schemes consists in generating first a set of points, termed global cloud of points, within the analysis domain. Then, for each of these points, a local cloud of neighbouring points is selected. A local approximation is chosen for the sought unknowns in terms of the point values using typically least squares procedures. Finally, the derivation of a discrete set of algebraic equations is obtained by substituting the point approximations into the governing partial differential equations of the problem, expressed either in local differential or in an averaged weighted residual form. The first option is a truly mesh-free approach as no integration within internal subdomains is needed [Bat93, Dua95, Oña96a, Oña96b, Oña98, Oña00a]. Conversely, in Galerkin-type procedures, a background grid for numerical integration purposes is required [Nay72, Bel94, Liu96, Alt99, De00]. The name 'Finite points method' proposed in [Oña96a, Oña98, Oña00a] conbine a weighted least square approximation of the unknowns over each local cloud whit a stabilized point collocations procedure elliminating any numerical inestability. Examples of succesful aplications of the finite point method have been shown in advection difution [Oña96a, Oña96b, Oña00a], incompressible flows [Oña96a, Oña96b] and solid mechanics problems [Oña00b].

The present paper is organized as follows: Section 2 treats the basic weighted least squares procedure for finite point methods. Section 3 describes the construction of local clouds, and Section 4 the flow solver. Numerical examples are shown in Section 5, and some conclusions are drawn in Section 6.

2. WEIGHTED LEAST SQUARES APPROXIMATIONS

In order to define the subsequent notation, we recall the basic weighted least squares procedure. Throughout the paper, the Einstein summation convention will be employed, i.e. a sum is always performed over repeated indices. Let Ω_i be the (local) interpolation domain of a function $u(\mathbf{x})$. Assume, furthermore, that Ω_i contains n points with coordinates $\mathbf{x}_j \in \Omega_i$. The unknown function u may be approximated within Ω_k by

$$u(\mathbf{x}) \approx u^h(\mathbf{x}) = p_k(\mathbf{x})\alpha_k = \mathbf{p}(\mathbf{x})^T \cdot \boldsymbol{\alpha} \quad , \quad k = 1, ..., m \quad , \tag{2.1}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_m]^T]$ and the vector $\mathbf{p}(\mathbf{x})$ contains so-called 'base interpolating functions' which are typically monomials. For 3-D problems we have used:

$$\mathbf{p}_2 = [1, x, y, z, x^2, xy, xz, y^2, yz, z^2]^T \quad , \tag{2.2a}$$

$$\mathbf{p}_{2.5} = [1, x, y, z, x^2, xy, xz, y^2, yz, z^2, x^2y, x^2z, xy^2, xyz, xz^2, y^2z, yz^2]^T \quad , \tag{2.2b}$$

$$\mathbf{p}_3 = [1, x, y, z, x^2, xy, xz, y^2, yz, z^2, x^3, x^2y, x^2z, xy^2, xyz, xz^2, y^3, y^2z, yz^2, z^3]^T \quad , \tag{2.2c}$$

Using the notation $u_j = u(\mathbf{x}_j)$, $u_j^h = u^h(\mathbf{x}_j)$, $\mathbf{p}_j = \mathbf{p}(\mathbf{x}_j)$ and $\varphi_{ij} = \varphi(\mathbf{x}_j - \mathbf{x}_i)$, the weighted least squares approximation (WLSQ) is obtained by minimizing

$$J_i = \varphi_{ij}(u_j - u_j^h)^2 = \varphi_{ij}(u_j - \mathbf{p}_j^T \cdot \boldsymbol{\alpha})^2 \quad . \quad j = 1, ..., n \quad . \tag{2.3}$$

The weighting function $\varphi(\mathbf{x}_j - \mathbf{x}_i)$ takes a unit value in the vicinity of point i, i.e. the point where the function (or its derivatives) are to be evaluated, and decreases as one moves away from $\mathbf{x}_i$. A typical choice for $\varphi(\mathbf{x}_j - \mathbf{x}_i)$ is the normalized Gaussian function shown schematically for a 1-D case in Figure 1. We remark that $n \geq m$ is always required in the sampling region. Moreover, for $n = m$ the effect of weighting vanishes, and the procedure reverts to standard finite element interpolation.

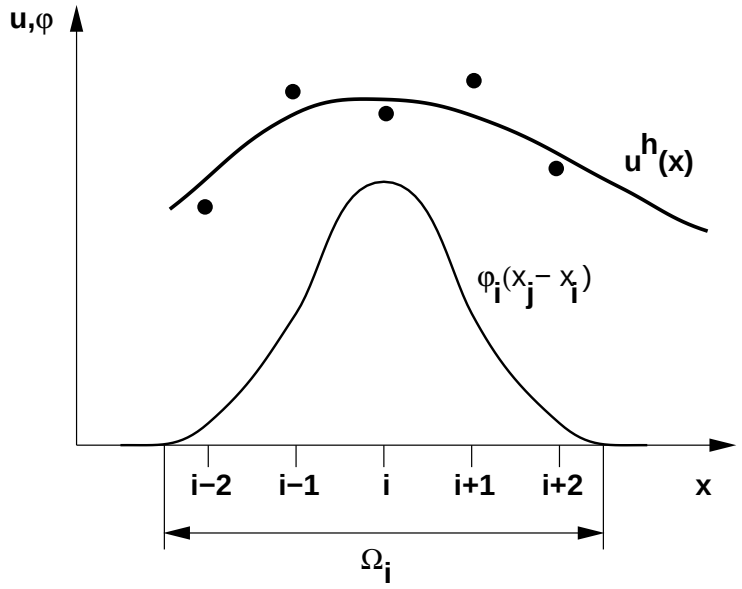


Figure 1 WLSQ Procedure

The minimization of J_i with respect to α_j yields

$$\mathbf{A} \cdot \boldsymbol{\alpha} = \mathbf{B} \cdot \mathbf{u} \quad , \tag{2.4}$$

i.e.

$$\boldsymbol{\alpha} = \mathbf{C} \cdot \mathbf{u} \quad , \quad \mathbf{C} = \mathbf{A}^{-1} \cdot \mathbf{B} \quad , \tag{2.5}$$

where

$$\mathbf{A} = \sum_{j=1}^{n} \varphi_{ij}(\mathbf{p}_j \otimes \mathbf{p}_j^T) \quad , \tag{2.6}$$

$$\mathbf{B} = [\varphi_{1i}\mathbf{p}_1, \varphi_{2i}\mathbf{p}_2, ..., \varphi_{ni}\mathbf{p}_n] \quad , \tag{2.7}$$

If one evaluates all approximations based on 'the local frame' by shifting the coordinate origin to the point i, any value or derivative can be obtained quickly from α. In particular,

$$u_i = \alpha_1 \quad , \tag{2.8}$$

$$\nabla u_i = (\alpha_2, \alpha_3, \alpha_4) \quad , \tag{2.9}$$

and

$$\nabla^2 u_i = 2 * (\alpha_5 + \alpha_8 + \alpha_{10}) \quad . \tag{2.10}$$

These expressions can also be written as:

$$\left.\frac{\partial u}{\partial x_l}\right|_i = D^{lj} u_j \quad , \tag{2.11}$$

where $D^{lj} = C^{qj}$ and $q = l + 1$, and

$$\nabla^2 u_i = L^{lj} u_j \quad , \quad L^{lj} = 2 * (C^{5j} + C^{8j} + C^{10j}) \quad . \tag{2.12}$$

3. GENERATION OF LOCAL CLOUDS

Any finite point method based on point collocation requires the construction of a local cloud of points for every point in the domain. In the sequel, we describe one of the many possible ways to construct these local clouds that has proven reliable. We first give an outline of the technique, and then detail the algorithms employed to make this process as fast as possible. The input required, i.e.

- A list of points with their respective coordinates;
- A list of triangles that define the outer boundaries of the domain;

is obtained using the automatic advancing front point generator described in [Löh98]. Given this information:

`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` intersects a face;
  `endwhile`
  Produce a Delauney grid with the local points;

Initialize the local cloud list with the first layer of 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;
`enddo`
As no proper local cloud was found: increase the search region;
`enddo`
With the surface triangulation: Correlate boundary points to apply boundary conditions.

In the sequel, 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 [Knu73, Sam84, Löh88]. 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 can 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 sub-subdivisions 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 [Knu73].

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 typical case is shown in Figure 2, where face A clearly does 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.

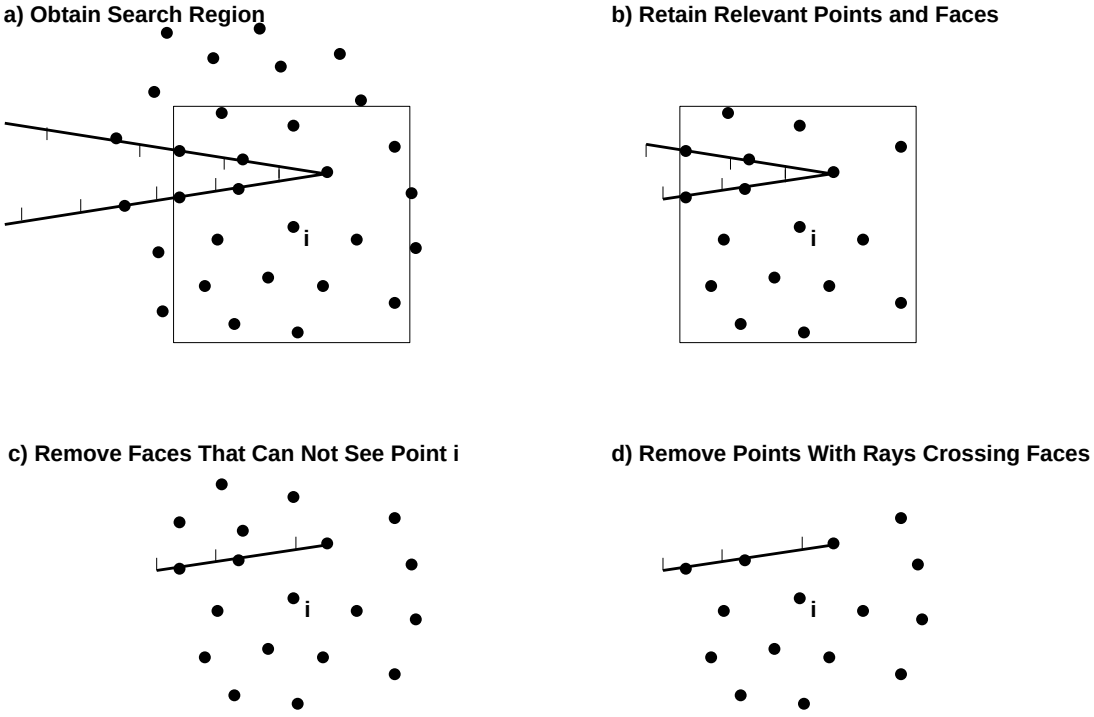


Figure 2 Search for Close Points

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 Figure 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 Delauney Meshing

Given a list of close points, there are many possible ways of obtaining local clouds. Among them, the Delauney technique will produce a graph of nearest neighbours with optimal properties for finite Elements and elliptic PDEs. It was therefore tempting to use this same technique in the present context. We outline the main steps, and refer the reader to George's recent monogram [Geo98] for details.

Place a large tetrahedron (or box with 5/6 tetrahedra) around the point to be gridded;
`do`: for all close points:
Find the element(s) $\mathbf{x}_i$ falls into;
Obtain all elements whose circumsphere encompasses $\mathbf{x}_i$;
Remove from this list of elements all those that would not form a proper element (volume, angles) with $\mathbf{x}_i$; this results in a properly constrained convex hull;
Reconnect the outer faces of the convex hull with $\mathbf{x}_i$ to form new elements;
`enddo`
Retain only the elements with all nodes belonging to the list of close points.

The basic procedure has been sketched in Figure 3.

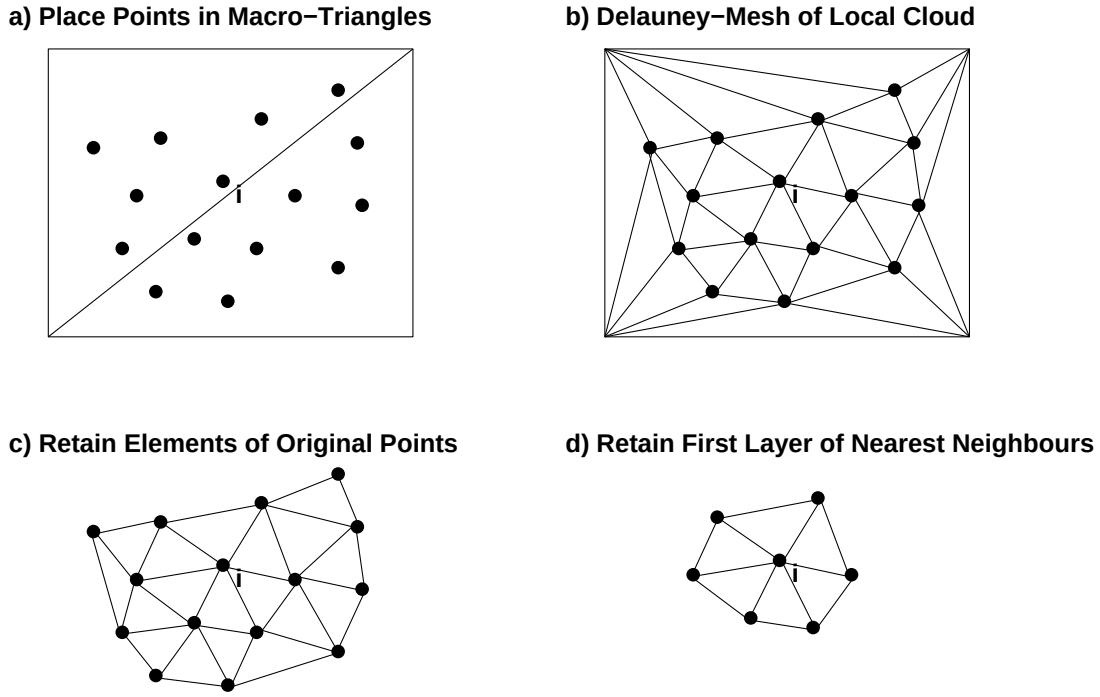


Figure 3 Formation of Local Cloud

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

3.6 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 $\mathbf{A}$ (Eqn.(2.5) above). For this reason, several tests are carried out for every local cloud.
The first test is to see if the matrix $\mathbf{A}$ will be singular. The most obvious indication that $\mathbf{A}$ is singular or not adequate will occur during inversion. However, we have found cases where the local cloud is not suitable for FPMs although the inverse exists and $\mathbf{A} \cdot \mathbf{A}^{-1} \approx 1$. We therefore also test for the largest (absolute) entry in $\mathbf{A}^{-1}$. 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 will deviate from the exact values. For the first derivatives, we take $u = x + y + z$, which should yield a gradient of $\nabla u = (1, 1, 1)$. For the second derivatives, we take $u = x^2 + y^2 + z^2$, which should yield a Laplacian of $\nabla^2 u = 6$. If the values obtained via local cloud approximation and the exact values deviate by more than 10^{-10}, the local cloud is rejected.

3.7 Approximation Order of Local Cloud

From eqns.(2.2a-c) one can infer that the smallest number of points in the local cloud required to achieve a quadratic function, a 'serendipity cubic' and a cubic function is 10, 17 and 20 respectively. A considerable percentage of the local clouds obtained from the first layer of Delauney-neighbours will allow for these higher order approximations. Therefore, a test is carried out as before for the $\mathbf{A}$-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.

4. FLOW SOLVER

In order to set the notation, we recall the compressible Navier-Stokes equations:

$$\mathbf{u}_{,t} + \nabla \cdot \mathbf{F} = \mathbf{u}_{,t} + \nabla \cdot (\mathbf{F}^a - \mathbf{F}^v) = 0 \quad , \tag{4.1}$$

where

$$\mathbf{u} = \begin{Bmatrix} \rho \\ \rho v_i \\ \rho e \end{Bmatrix} \ , \ \mathbf{F}^a_j = \begin{Bmatrix} \rho v_j \\ \rho v_i v_j + p\delta_{ij} \\ v_j(\rho e + p) \end{Bmatrix} \ , \ \mathbf{F}^v_j = \begin{Bmatrix} 0 \\ \sigma_{ij} \\ v_l \sigma_{lj} + kT_{,j} \end{Bmatrix} \quad . \tag{4.2}$$

Here ρ, p, e, T, k, v_i denote the density, pressure, specific total energy, temperature, conductivity and fluid velocity in direction x_i respectively. This set of equations is closed by providing an equation of state, e.g. for a polytropic gas:

$$p = (\gamma - 1)\rho[e - \frac{1}{2} v_j v_j] \quad , \quad T = c_v[e - \frac{1}{2} v_j v_j] \quad , \tag{4.3a,b}$$

where γ, c_v are the ratio of specific heats and the specific heat at constant volume respectively. Furthermore, the relationship between the stress-tensor σ_{ij} and the deformation rate must be supplied. For water and almost all gases, Newton's hypothesis:

$$\sigma_{ij} = \mu(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}) + \lambda \frac{\partial v_k}{\partial x_k}\delta_{ij} \quad , \tag{4.4}$$

complemented with Stokes Hypothesis:

$$\lambda = -\frac{2\mu}{3} \quad . \tag{4.5}$$

is an excellent approximation. The compressible Euler equations are obtained by neglecting the viscous fluxes, i.e. setting $\mathbf{F}^v = 0$.

From the expressions (2.9, 2.11) of the WLSQ approximation procedure, we obtain the following expression for the divergence of the flux function $\mathbf{F}$:

$$\nabla \cdot \mathbf{F}|_i \approx \mathbf{r}^i = D^{lj}\mathbf{F}^l_j \quad , \tag{4.6}$$

with D^{lj} defined by eqn(2.11). This expression can be 'symmetrized' by setting:

$$\mathbf{r}^i = D^{lj}\big|_{j\neq i}(\mathbf{F}^l_j + \mathbf{F}^l_i) + (D^{li} - D^{lj}\big|_{j\neq i})\mathbf{F}^l_i \quad , \tag{4.7}$$

or:

$$\mathbf{r}^i = D^{lj}\big|_{j\neq i}(\mathbf{F}^l_j + \mathbf{F}^l_i) + (\tilde{D}^{ii}\mathbf{F}^l_i \quad , \tag{4.8}$$

For perfectly centered clouds and symmetric weighting functions φ, the $\tilde{D}^{ii}$-term has to vanish. Let us consider the first term in more detail. The inner product over the dimensions l may be written in compact form as

$$\mathbf{r}^i = d^{ij}\mathcal{F}_{ij} = d^{ij}(\mathbf{f}_i + \mathbf{f}_j) \quad , \tag{4.9}$$

where the $\mathbf{f}_i$ are the 'fluxes along edges' or 'fluxes along directions', obtained from the scalar product

$$\mathbf{f}_i = S^{ij}_l \mathbf{F}^l_i \quad , \quad S^{ij}_l = \frac{D^{ij}_l}{d^{ij}} \quad , \quad d^{ij} = \sqrt{D^{ij}_l D^{ij}_l} \quad . \tag{4.10}$$

This expression is reminiscent of edge-based or face-based finite Volume and finite Element solvers, and is equivalent to the Galerkin weighted residual method, which has central difference character. This is known to be an unstable discretization, and must be augmented by stabilizing terms. This can be achieved either by adding directly second-, fourth- or higher-order damping [Mav91], or by modifying the flux function according to the local physics [Lee74, Roe81, Whi89, Luo93 ,Luo94]. The second route seems attractive in the present context, as it does not require any intrinsic measure of length. We have implemented both flux-vector and flux-difference splitting schemes for the flux functions. In particular, the vanLeer, Roe and AUSM+ schemes were tested.

All of them gave acceptable results for a large class of problems. A brief desription of these schemes is given here for completeness.

4.1 vanLeer Solver

The approximate Riemann solver of vanLeer represents one of the first modern high resolution schemes. Even though deficient in its standard form for Navier-Stokes calculations, it yields excellent results for the Euler equations, particularly for high Mach-number flows. The key idea is to separate the 'fluxes along an edge' according to their upwind character, i.e.

$$\mathcal{F}_{ij} = \mathbf{f}^+(\mathbf{u}_i) + \mathbf{f}^-(\mathbf{u}_j) \tag{4.11}$$

where, in 1-D,

$$\mathbf{f}^{\pm} = \begin{Bmatrix} f^{\pm} \\ f^{\pm}\left[(\gamma-1)v \pm 2c\right]/\gamma \\ f^{\pm}\left[(\gamma-1)v \pm 2c\right]^2/2(\gamma^2-1) \end{Bmatrix} \quad , \quad f^{\pm} = \pm\rho c\left[\frac{1}{2}(M \pm 1)\right]^2 \quad ,$$

$$c = \sqrt{\frac{\gamma p}{\rho}} \quad , \quad M = \frac{v}{c} \quad . \tag{4.12}$$

4.2 Roe Solver

By far the most popular of the approximate Riemann solvers based on flux-difference splitting is the one derived by Roe (1981). The first order flux for this solver is of the form:

$$\mathcal{F}_{ij} = \mathbf{f}_i + \mathbf{f}_j - |\mathbf{A}^{ij}|(\mathbf{u}_i - \mathbf{u}_j) \tag{4.13}$$

where $|\mathbf{A}^{ij}|$ denotes the Roe matrix evaluated in the direction D^{ij}.

4.3 Higher Order Schemes

In order to achieve a scheme of order higher than one, the amount of dissipation must be reduced. This implies reducing the magnitude of the difference $\mathbf{u}_i - \mathbf{u}_j$ by 'guessing' a smaller difference of the unknowns at the location where the approximate Riemann flux is evaluated (i.e. the middle of the edge). The assumption is made that the function behaves smoothly in the vicinity of the edge. This allows the construction or 'reconstruction' of alternate values for the unknowns at the middle of the edge, denoted by $\mathbf{u}_j^-, \mathbf{u}_i^+$, leading, e.g. for the Roe-solver, to a flux function of the form

$$\mathcal{F}_{ij} = \mathbf{f}^+ + \mathbf{f}^- - |\, A(\mathbf{u}_i^+, \mathbf{u}_j^-)\,|\,(\mathbf{u}_j^- - \mathbf{u}_i^+) \quad , \tag{4.14}$$

where

$$\mathbf{f}^+ = \mathbf{f}(\mathbf{u}_i^+), \qquad \mathbf{f}^- = \mathbf{f}(\mathbf{u}_j^-) \quad . \tag{4.15}$$

The upwind-biased interpolations for $\mathbf{u}_i^+$ and $\mathbf{u}_j^-$ are defined by

$$\mathbf{u}_i^+ = \mathbf{u}_i + \frac{1}{4}\left[(1-k)\Delta_i^- + (1+k)(\mathbf{u}_j - \mathbf{u}_i)\right] \quad , \tag{4.16a}$$

$$\mathbf{u}_j^- = \mathbf{u}_j - \frac{1}{4}\left[(1-k)\Delta_j^+ + (1+k)(\mathbf{u}_j - \mathbf{u}_i)\right] \quad , \tag{4.16b}$$

where the forward and backward difference operators are given by

$$\Delta_i^- = \mathbf{u}_i - \mathbf{u}_{i-1} = 2\mathbf{l}_{ji} \cdot \nabla \mathbf{u}_i - (\mathbf{u}_j - \mathbf{u}_i) \quad , \tag{4.17a}$$

$$\Delta_j^+ = \mathbf{u}_{j+1} - \mathbf{u}_j = 2\mathbf{l}_{ji} \cdot \nabla \mathbf{u}_j - (\mathbf{u}_j - \mathbf{u}_i) \quad , \tag{4.17b}$$

and $\mathbf{l}_{ji}$ denotes the edge difference vector $\mathbf{l}_{ji} = \mathbf{x}_j - \mathbf{x}_i$. The parameter k can be chosen to control the degree of approximation. Setting $k \neq 1/3$ results in a second order scheme (Hirsch (1991)) while $k = 1/3$ leads to a third order scheme. The additional information required for $\mathbf{u}_{i+1}, \mathbf{u}_{j+1}$ can be obtained by evaluation of gradients (Whitaker (1989), Luo (1993, 1994)), as shown in Figure 4.

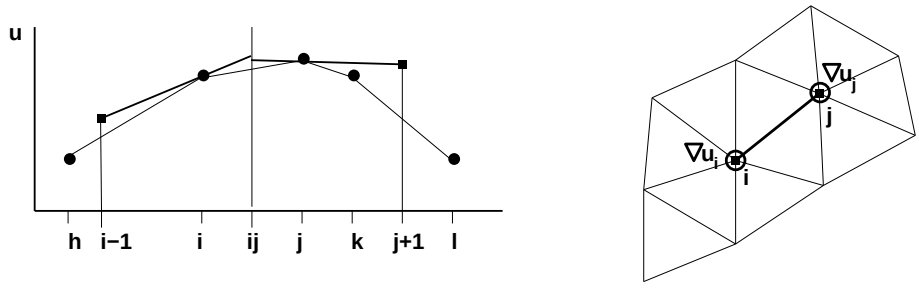


Figure 4 Higher Order Approximations

4.4 Limiting

The inescapable fact stated in Godunov's theorem that no linear scheme of order higher than one is free of oscillations implies that with these higher order extensions, some form of limiting will be required. The flux limiter modifies the upwind-biased interpolations $\mathbf{u}_i, \mathbf{u}_j$, replacing them by

$$\mathbf{u}_i^+ = \mathbf{u}_i + \frac{s_i}{4}\left[(1-ks_i)\Delta_i^- + (1+ks_i)(\mathbf{u}_j - \mathbf{u}_i)\right] \quad , \tag{4.18a}$$

$$\mathbf{u}_j^- = \mathbf{u}_j - \frac{s_j}{4}\left[(1-ks_j)\Delta_j^+ + (1+ks_j)(\mathbf{u}_j - \mathbf{u}_i)\right] \quad , \tag{4.18b}$$

where s is the flux limiter. For $s = 0, 1$, the first and high order schemes are recovered respectively. A number of limiters have been proposed in the literature, and this area is still a matter of active research (see Sweby (1984) for a review). We include the van Albada limiter here, one of the most popular ones. This limiter acts in a continuously differentiable manner and is defined by

$$s_i = max\left\{0, \frac{2\Delta_i^-(\mathbf{u}_j - \mathbf{u}_i) + \epsilon}{(\Delta_i^-)^2 + (\mathbf{u}_j - \mathbf{u}_i)^2 + \epsilon}\right\} \quad , \tag{4.19a}$$

$$s_j = max\left\{0, \frac{2\Delta_j^+(\mathbf{u}_j - \mathbf{u}_i) + \epsilon}{(\Delta_j^+)^2 + (\mathbf{u}_j - \mathbf{u}_i)^2 + \epsilon}\right\} \quad , \tag{4.19b}$$

where ϵ is a very small number to prevent division by zero in smooth regions of the flow. For systems of PDEs one can consider limiting on conservative variables, primitive variables, and characteristic variables. Using limiters on characteristic variables seems to give the best results, but due to the lengthy algebra this option is very costly. For this reason, primitive variables are more often used for practical calculations as they provide a better accuracy vs. CPU ratio.

5. NUMERICAL EXAMPLES

The proposed methodology was coded and run on a number of test cases. The visualization of results from finite point methods present a number of interesting problem in itself. Given that the unknowns at any given point and its interpolating value are not the same, any mesh representation would yield discontinuities. Here, we present the results obtained by coluring the points according to their values. Plane cuts and iso-surfacs are obtained by searching the local clouds cut, and then interpolating to the cut position.

5.1 Supersonic Flow Past a Wedge: A recurring question often asked about finite point methods is whether they maintain conservation at the discrete level, a property considered vital for proper shock capturing. For this reason, supersonic flow at $Ma = 3.0$ past a 15^o wedge was considered. The geometry, boundary conditions and discretization information of the problem, together with the mach-number in a plane-cut are shown in Figure 5.1.

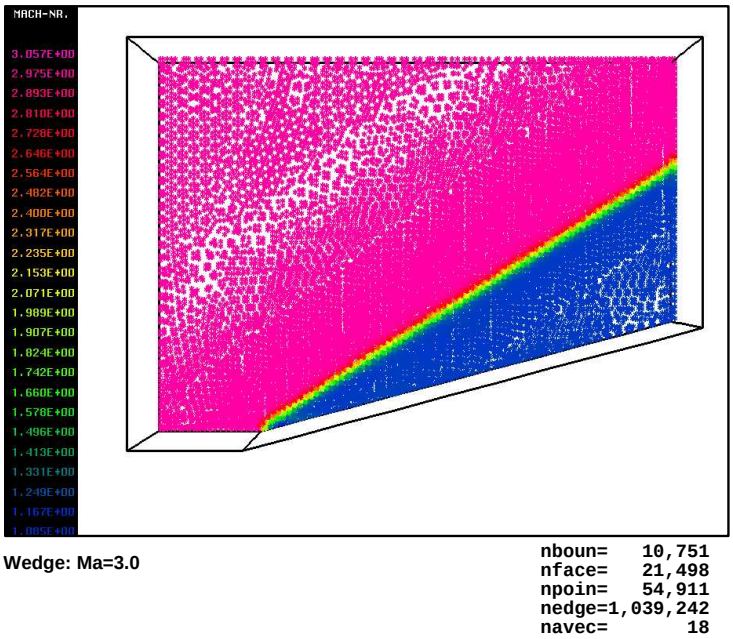


Figure 5.1 Wedge

The pressure on the surface is shown in Figure 5.2. One can clearly see the shock, which is obtained at an angle of $\alpha_s = 37.15^o$, in perfect correlation with analytical results. For this case, the approximate Riemann solver of vanLeer was employed, together with the vonAlbada limiter on conserved quantities. Note that the shock is captured across 2-3 points, which is similar to mesh-based finite volume or finite element methods.

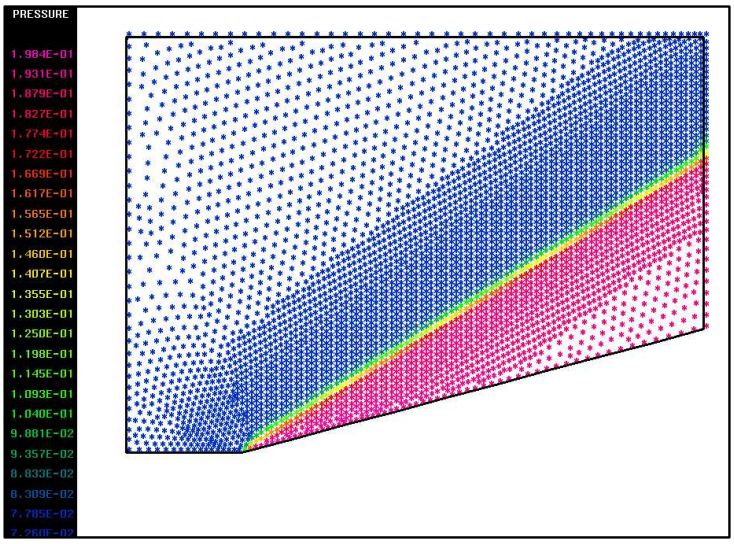


Figure 5.2 Wedge

5.2 Ni-Bump: This classic test case was run to assess the ability of the finite point method to simulate transonic flow. The geometry, boundary conditions and discretization information of the problem, together with the sonic iso-surface are shown in Figure 6.1.

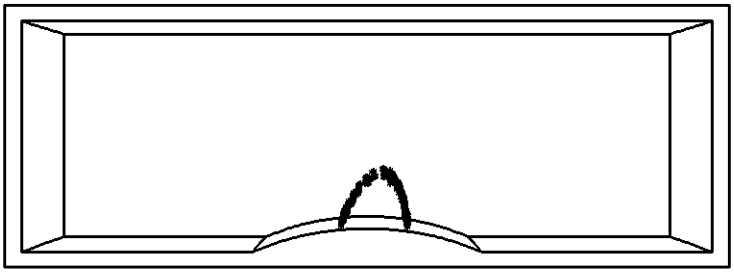
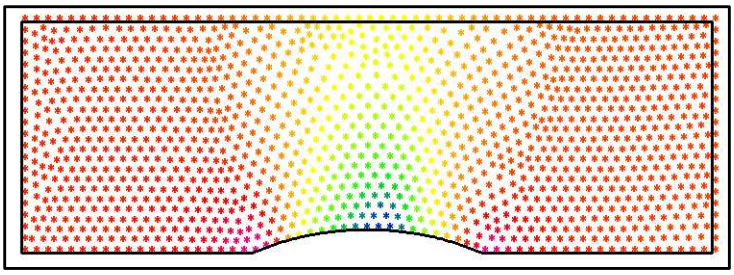
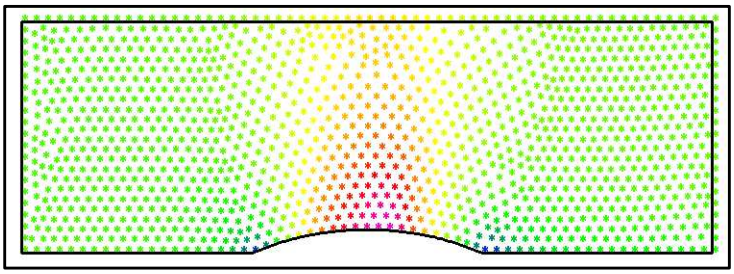


Figure 6.1 Ni-Bump

For this case, the approximate Riemann solver of Roe was employed, together with the vonAlbada limiter on conserved quantities. The Mach-number and pressures in a cut plane can be seen in Figure 6.2. A close-up of the Mach-number in the bump region is shown in Figure 6.3.

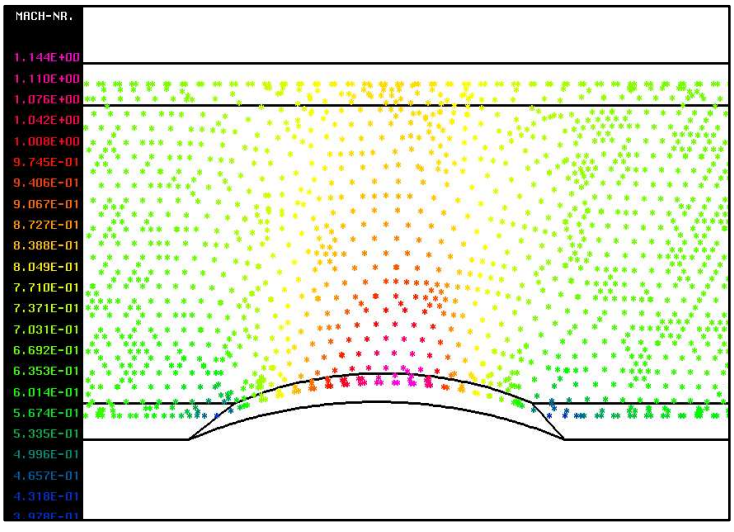


Pressure



Mach−Number

Figure 6.2 Ni-Bump



Mach−Number

Figure 6.3 Ni-Bump

5.3 Sphere: The subsonic flow past a sphere provides as good test case at study the intrinsic dissipation of schemes. The potential solution is completely symmetric, and

any spurious entropy creation will be reflected in the the numerical solution. The geometry, boundary conditions and discretization information of the problem are shown in Figure 7.1. A close-up of the surface discretization used is given in Figure 7.2. The results, which were obtained with the approximate Riemann solver of Roe with vonAlbada limiter on conserved quantities, are shown in Figures 7.3,7.4. A small unsymmetry in the pressure can be seen. However, one should remark that the results compare very well with similar mesh-based finite volume or finite element methods.

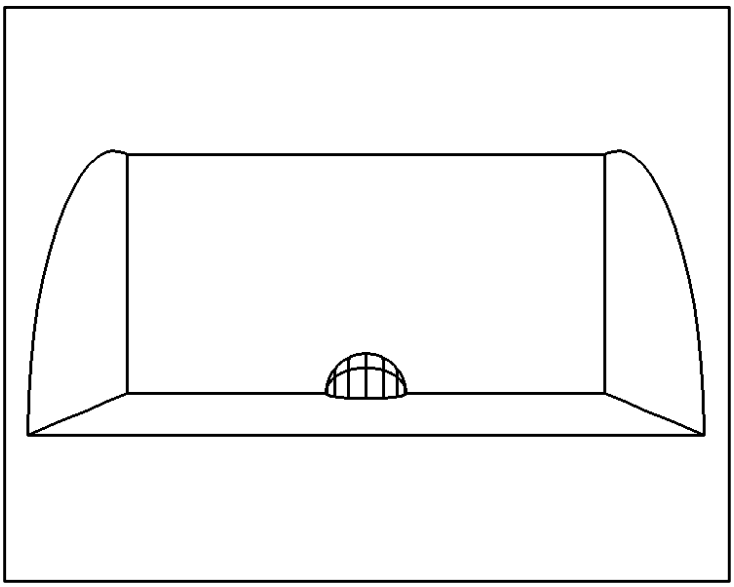


Figure 7.1 Sphere

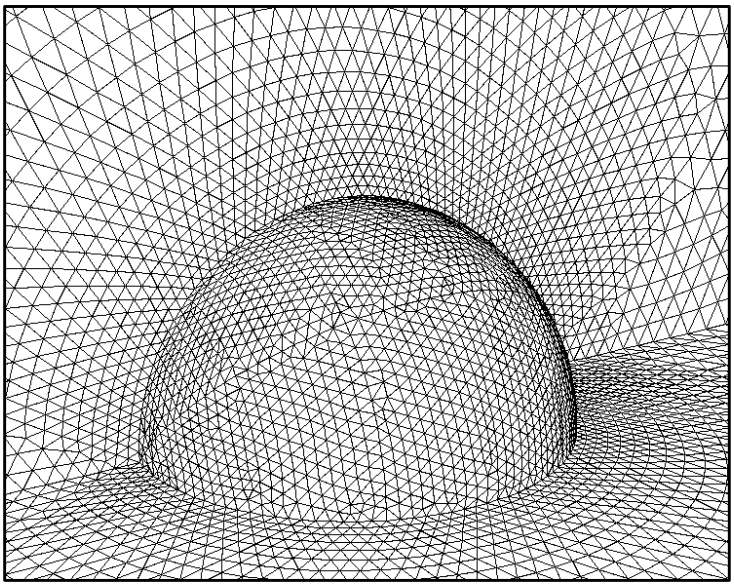


Figure 7.2 Sphere

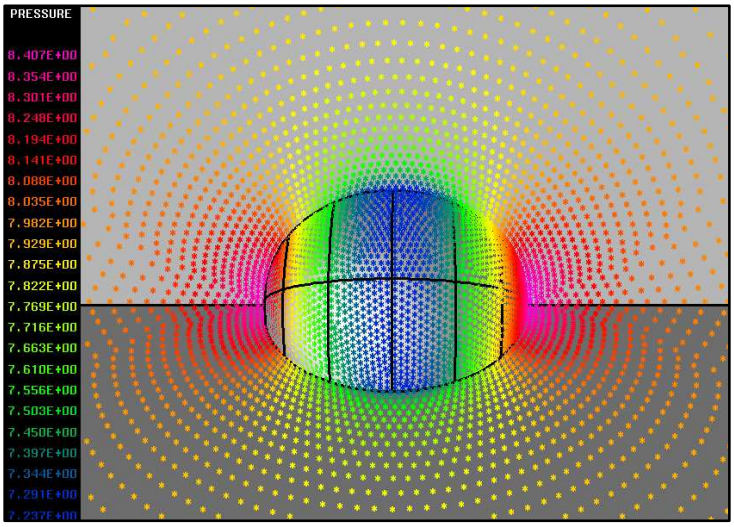


Figure 7.3 Sphere

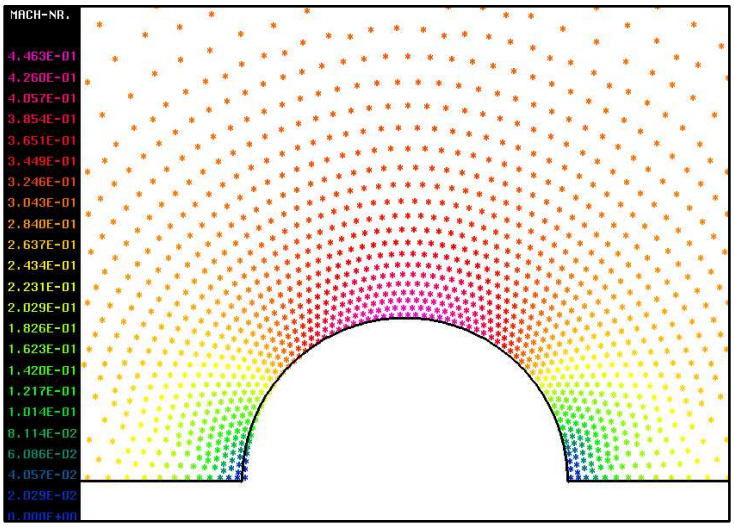


Figure 7.4 Sphere

6. CONCLUSIONS AND OUTLOOK

A weighted least squares finite point method for compressible flow has been developed. Starting from a global cloud of points, local clouds are constructed using a Delaunay technique with a series of tests for the quality of the resulting approximations. The approximation factors for gradient and the Laplacian of the resulting local clouds are used to derive an edge-based solver that works with approximate Riemann solvers. The results obtained show accuracy comparable to equivalent mesh-based finite volume or finite element techniques, making the present finite point method competitive.
Future efforts will center on:

- Improving efficiencies (cache-miss reduction, avoidance of duplicate information, etc.);
- Higher-order schemes (full cubic recovery on the edges).

7. ACKNOWLEDGEMENTS

A considerable part of this work was carried out while the first author was visiting the Centro Internacional de Métodos Numéricos en Ingeniería, Barcelona, Spain, in the Summer of 2000. The support for this visit is gratefully acknowledged.

8. REFERENCES

[Atl99] S. N. Atluri, H. G. Kim and J. Y. Cho - A Critical assessment of the truly Meshless Local Petrov-Galerkin (MLPG), and Local Boundary Integral Equation (LBIE) methods; *Computational Mechanics* 24, 348-372 (1999).

[Bat93] J. Batina - A Gridless Euler/Navier-Stokes Solution Algorithm for Complex Aircraft Configurations; *AIAA*-93-0333 (1993).

[Bel94] T. Belytschko, Y. Lu and L. Gu - Element Free Galerkin Methods; *Int. J. Num. Meth. Eng.* 37, 229-256 (1994).

[Dua95] C.A. Duarte and J.T. Oden - H_p Clouds - A Meshless Method to Solve Boundary-Value Problems; *TICAM-Rep.* 95-05 (1995).

[De00] S. De and K. J. Bathe - The method of finite spheres; *Computational Mechanics* 25, 329-345 (2000).

[Geo98] P.L. George and H. Borouchaki - *Delaunay Triangulation and Meshing*; Editions Hermes, Paris (1998).

[Knu73] D.N. Knuth - *The Art of Computer Programming* , Vols. 1-3; Addison-Wesley (1973).

[Lee74] B. van Leer - Towards the Ultimate Conservative Scheme. II. Monotonicity and Conservation Combined in a Second Order Scheme; *J. Comp. Phys.* 14, 361-370 (1974).

[Liu96] W.K. Liu, Y. Chen, S. Jun, J.S. Chen, T. Belytschko, C. Pan, R.A. Uras and C.T. Chang - Overview and Applications of the Reproducing Kernel Particle Methods; *Archives Comp. Meth. Eng.* 3(1), 3-80 (1996).

[Löh88] R. Löhner - Some Useful Data Structures for the Generation of Unstructured Grids; *Comm. Appl. Num. Meth.* 4, 123-135 (1988).

[Löh97] R. Löhner - Automatic Unstructured Grid Generators; *Finite Elements in Analysis and Design* 25, 111-134 (1997).

[Löh98] R. Löhner - An Advancing Point Grid Generation Technique; *Comm. Num. Meth. Eng.* 14, 1097-1108 (1998).

[Luo93] H. Luo, H., J.D. Baum, R. Löhner and J. Cabello - Adaptive Edge-Based Finite Element Schemes for the Euler and Navier-Stokes Equations; *AIAA*-93-0336 (1993).

[Luo94] H. Luo, H., J.D. Baum and R. Löhner - Edge-Based Finite Element Scheme for the Euler Equations; *AIAA J.* 32, 6, 1183-1190 (1994).

[Mav91] D. Mavriplis, D. - Three-Dimensional Unstructured Multigrid for the Euler Equations; *AIAA*-91-1549-CP (1991).

[Nay72] R.A. Nay and S. Utku - An Alternative for the Finite Element Method; *Variational Methods Eng.* 1 (1972).

[Oña96a] E. Oñate, S. Idelsohn, O.C. Zienkiewicz and R.L. Taylor - A Finite Point Method in Computational Mechanics. Applications to Convective Transport and Fluid Flow; *Int. J. Num. Meth. Eng.* 39,3839-3866 (1996).

[Oña96b] E. Oñate, S. Idelsohn, O.C. Zienkiewicz, R.L. Taylor and C. Sacco - A Stabilized Finite Point Method for Analysis of Fluid Mechanics Problems; *Comp. Meth. Appl. Mech. Eng.* 139, 315-346 (1996).

[Oña98] E. Oñate and S. Idelsohn - A Mesh-Free Finite Point Method for Advective-Diffusive Transport and Fluid Flow Problems; *Computational Mechanics* 21, 283-292 (1998).

[Oña00a] E. Oñate C. Sacco and S. Idelsohn - A Finite Point Method for Incompressible Flow Problems; *Comput. Visual. Sci.* 3, 67-75 (2000).

[Oña00b] E. Oñate F. Perazzo and S. Idelsohn - Análisis de problemas de mecánica computacional mediante el método de puntos finitos estabilizados; *VI Cong. nacional de Mecanica aplicada y computacional* Abril 17-19, Aveiro, Portugal (2000).

[Roe81] P.L. Roe - Approximate Riemann Solvers, Parameter Vectors and Difference Schemes; *J. Comp. Phys.* 43, 357-372 (1981).

[Sam84] H. Samet - The Quadtree and Related Hierarchical Data Structures; *Computing Surveys* 16, 2, 187-285 (1984).

[Swe84] P.K. Sweby, P.K. - High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws; *SIAM J. Num. Anal.* 21, 995-1011 (1984).

[Whi89] D.L. Whitaker, D.L., B. Grossman and R. Löhner - Two-Dimensional Euler Computations on a Triangular Mesh Using an Upwind, Finite-Volume Scheme; *AIAA*-89-0365 (1989).