COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING
Commun. Numer. Meth. Engng, **14**, 1097–1108 (1998)

AN ADVANCING FRONT POINT GENERATION TECHNIQUE

RAINALD LÖHNER[1] AND EUGENIO OÑATE[2]*
[1]*GMU/CSI, George Mason University, Fairfax, Virginia, U.S.A.*
[2]*CIMNE, Universidad Politécnica de Catalunya, Barcelona, Spain*

SUMMARY

An algorithm to construct boundary-conforming, isotropic clouds of points with variable density in space is described. The input required consists of a specified mean point distance and an initial triangulation of the surface. Borrowing a key concept from advancing front grid generators, one point at a time is removed and, if possible, surrounded by admissible new points. This operation is repeated until no active points are left. Timings show that the scheme is about an order of magnitude faster than volume grid generators based on the advancing front technique, making it possible to generate large ($>10^6$) yet optimal clouds of points in a matter of minutes on a workstation. Several examples are included that demonstrate the capabilities of the technique. © 1998 John Wiley & Sons, Ltd.

KEY WORDS grid generation; finite point method; mesh free techniques

1. INTRODUCTION

Over the course of the past decade, a number of 'gridless' or 'mesh-free' schemes have appeared in the literature (see, for example, References 1–7 and the references cited therein). The interest in these schemes stems from the perceived difficulty of generating volume filling grids for problems characterized by complex geometries and/or complex physics. The typical numerical analysis process for 'gridless' or 'mesh-free' 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 and averaged weighted residual form. The first option is a truly mesh-free approach as no integration within internal subdomains is needed.[2,4–6] Conversely, in Galerkin-type procedures, a background grid for numerical integration purposes is required.[1,3,7]

The generation of an appropriate global cloud of points seems, at first, much simpler than the generations of points and elements. However, to date, all clouds of points used for the examples shown were generated using traditional grid generators, removing the elements as a post-processing step. The avoidance of points outside the desired computational domain, which is crucial for the local neighbourhood information and the application of boundary conditions, was

* Correspondence to: E. Oñate, International Center for Numerical Methods in Engineering, Modulo C1, Campus Norte UPC, Gran Capitan, s/n 08034 Barcelona, Spain.

CCC 1069–8299/98/121097–12$17·50

Received 25 July 1997
Accepted 15 January 1998

done basically by hand, with special coding for each case calculated. Here, we present a scheme that allows for the direct generation of clouds of points with the same degree of flexibility as advanced unstructured grid generators.[8–21] The mean distance between points (or, equivalently, the point density) is specified by means of background grids, sources and density attached to CAD-entities. In order not to generate points outside the computational domain, we assume an initial triangulation of the surface that is compatible with the desired mean distance between points specified by the user. Starting from this initial 'front' of points, new points are added, until no further points can be introduced. Whereas the advancing front technique for the generation of volume grids removes one face at a time to generate elements, the present scheme removes one point at a time, attempting to introduce as many points as possible in its immediate neighbourhood.

2. THE ALGORITHM

Assume as given:

(i) a specification of the desired mean distance between points in space. This is done here through a combination of background grids, sources and mean distance to neighbours attached to CAD data (see References 20 and 21 for more details)
(ii) an initial triangulation of the surface, with the face normals pointing towards the interior of the domain to be filled with points.

With reference to Figure 1, the complete advancing front point generation algorithm may be summarized as follows:

```
Determine the required mean point distance for the points of the triangulation;
while: there are active points in the front:
  Remove the point ipout with the smallest specified mean distance to neighbours from the
  front;
  With the specified mean point distance: determine the co-ordinates of nposs possible new
  neighbours. This is done using a stencil, some of which are shown in Figure 2;
  Find all existing points in the neighbourhood of ipout;
  Find all boundary faces in the neighbourhood of ipout;
  do: For each one of the possible new neighbour points ipnew:
    if: no point is closer than a minimum distance dminp from ipnew:
      if: ipnew is inside the computational domain
        Determine the required mean point distance for ipnew;
        Increment the number of points by one;
        Introduce ipnew to the list of co-ordinates;
        Introduce ipnew to the list of active front points;
      endif
    endif
  enddo
endwhile
```

The main search operations required are:

(i) finding the active point with the smallest mean distance to neighbours

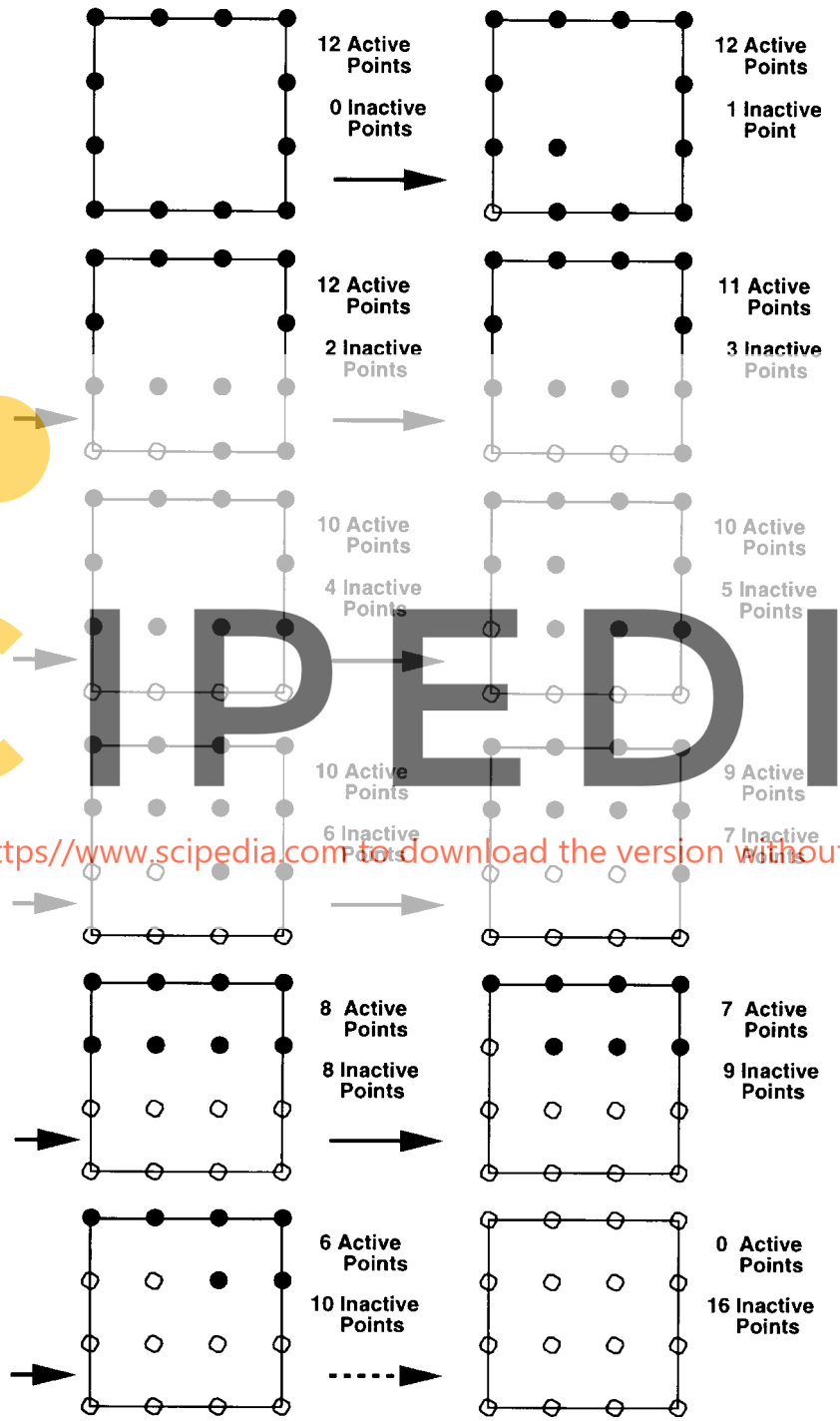

Figure 1. Advancing front point generation

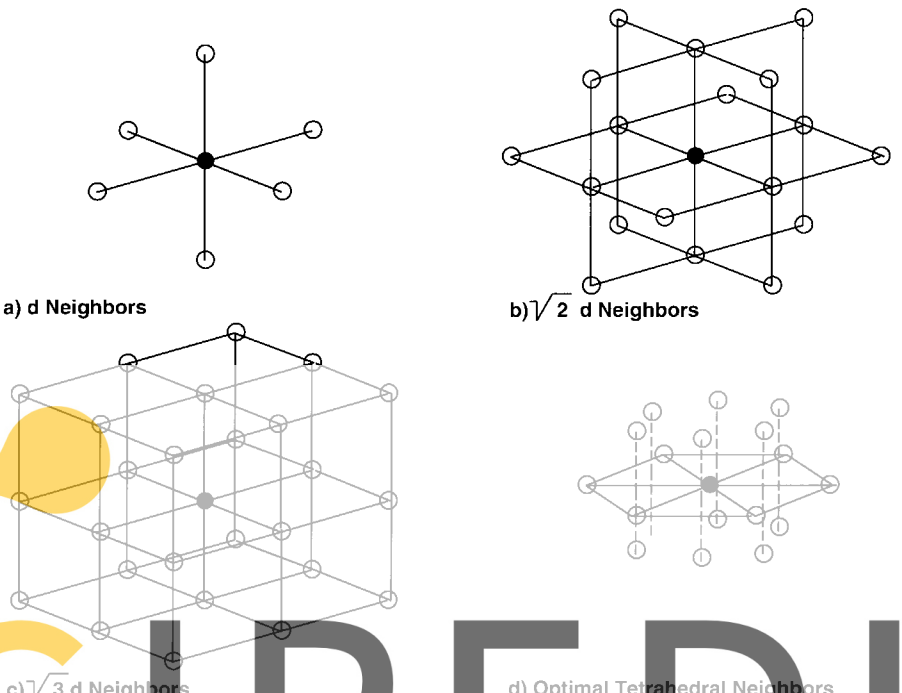

Figure 2. Stencils for point introduction

(ii) finding the existing points in the neighbourhood of ipout
(iii) finding the boundary faces in the neighbourhood of ipout

These three search operations are performed efficiently using heap-lists, octrees and linked lists, respectively. The use of such data structures in the context of grid generation has been described in detail before,[8–21] and does not need to be repeated here.

3. POINT STENCILS

A number of different stencils may be contemplated. Each one of these corresponds to a particular space-filling point configuration. The simplest possible stencil is the one that only considers the six nearest neighbours on a Cartesian gird (see Figure 2(a)). It is easy to see that this stencil, when applied recursively with the present advancing front algorithm, will fill a 3D volume completely. Other Cartesian stencils, which include nearest neighbours with distances $\sqrt{2}$ and $\sqrt{3}$ from ipout are shown in Figures 2(b,c). The 'tetrahedral' stencil shown in Figure 2(d) represents another possibility. The 6-point stencil leads to the smallest amount of rejections and unnecessary testing, and was therefore selected for all the examples shown below.

In may instances it is advisable to generate 'body conforming' clouds of points in the vicinity of surfaces. This can be accomplished by evaluating the average point-normals for the initial triangulation. When creating new points, the stencil used is rotated in the direction of the normal. The newly created points inherit the normal from the point ipout they originated from.

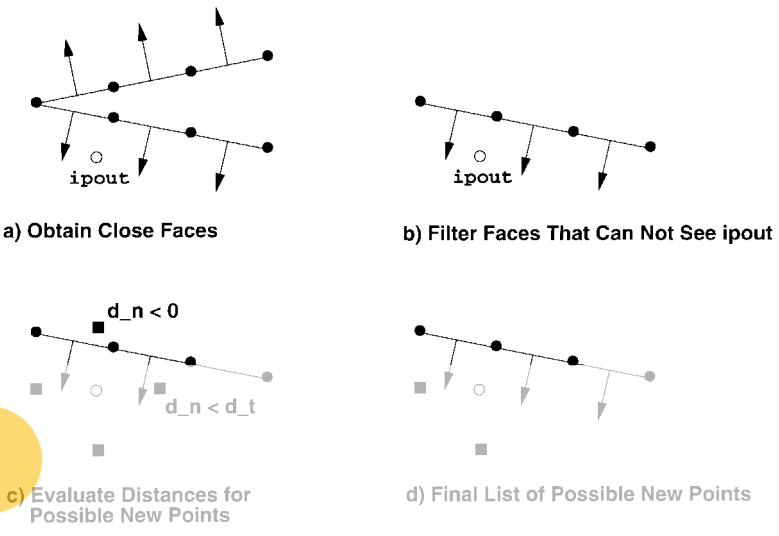

Figure 3. Evaluation of boundary conformity for new points

4. BOUNDARY CONSISTENCY CHECKS

A crucial requirement for a general point generator is the ability to only generate points within the computational domain desired. If we assume that the point to be removed from the list of active points ipout is inside the domain, a new point ipnew will cross the boundary triangulation if it lies on the other side of the plane formed by any of the faces that are in the proximity of ipout and can see ipout. This criterion is made more stringent by introducing a tolerated closeness or tolerated distance d_t of new points to the exterior faces of the domain. Testing for boundary consistency then proceeds as follows (see Figure 3):

Obtain all the faces close to ipout;
Filter, from this list, the faces that are pointing away from ipout;
do: for each of the close faces left:

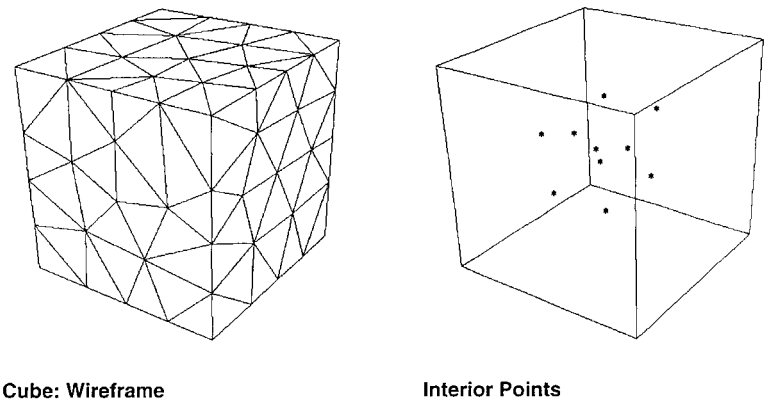

Figure 4. Cube

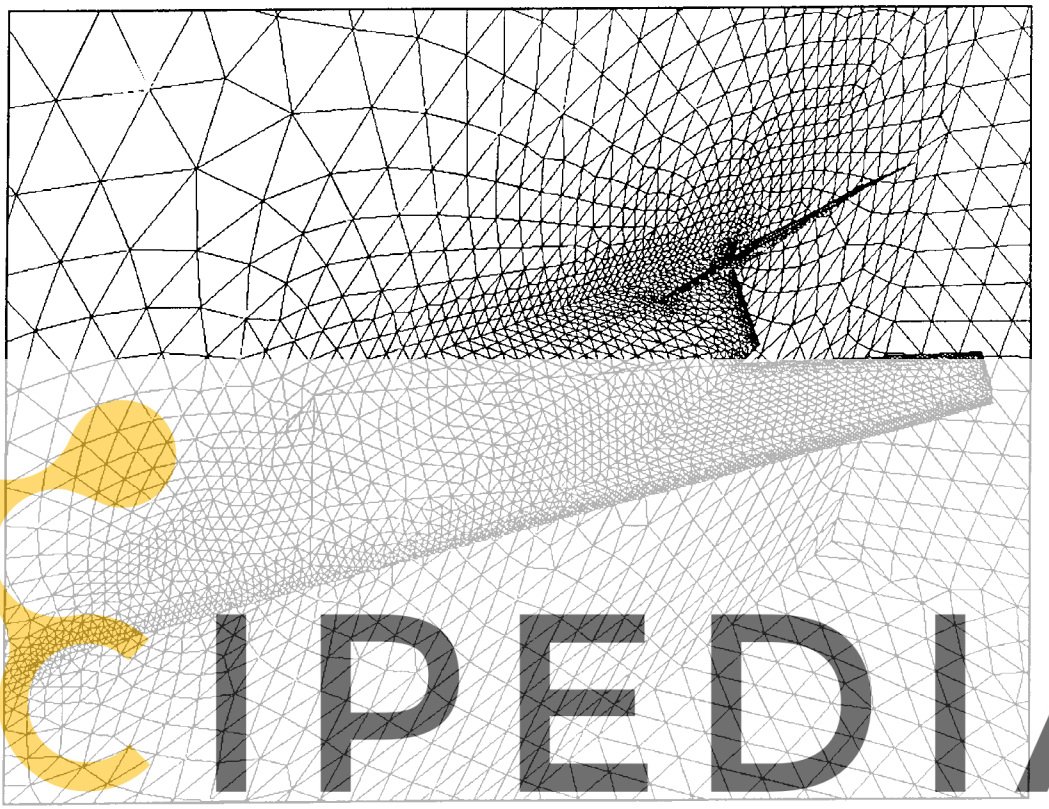

Figure 5(a). F117: surface mesh

Obtain the normal distance d_n from `ipnew` to this face;
  `if`: $d_n < 0$: `ipnew` lies outside the domain
    $\Rightarrow$ reject `ipnew` and exit;
  `elseif`: $0 \leqslant d_n \leqslant d_t$: obtain the closest distance d_{min} of `ipnew` to this face;
    if: $d_{min} < d_t$: `ipnew` is too close to the boundary
      $\Rightarrow$ reject `ipnew` and exit;
    `endif`
  endif
`enddo`

Typical values for d_t are $0{\cdot}707 d_0 \leqslant d_t \leqslant 0{\cdot}9 d_0$, where d_0 denotes the desired average distance between points.

5. EXAMPLES AND TIMINGS

The advancing front point generation algorithm was used to generate several point clouds. While the first case is rather academic, the others represent more realistic configurations.

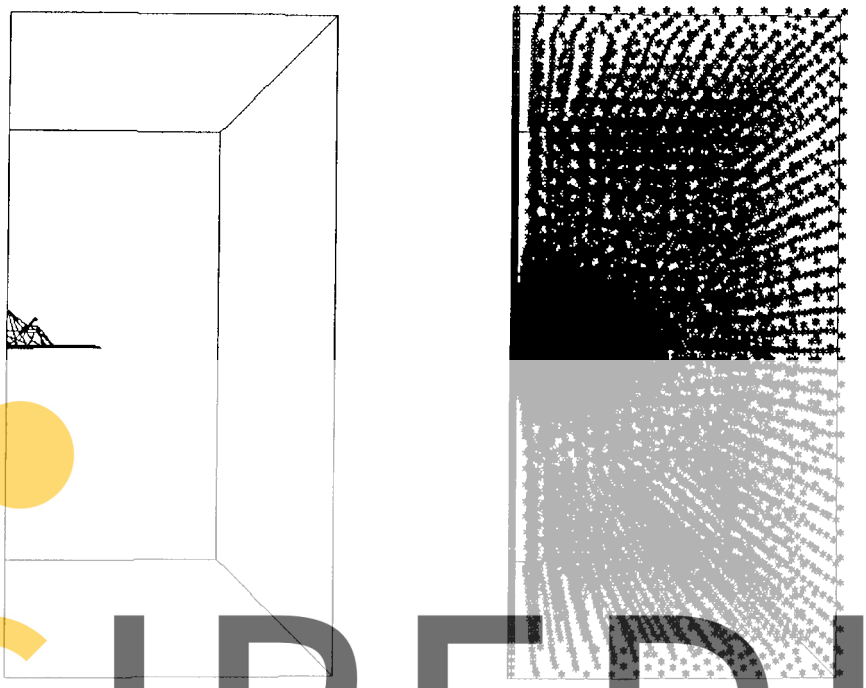

Figure 5(b). F117: global cloud of points

5.1. Cube

A $1 \times 1 \times 1$ cube was filled with a uniform cloud of points. The main aim of this example was to show the timings achieved. Figure 4 shows the surface triangulation and the cloud of interior (i.e. non-boundary) points for a mean nearest neighbour distance of $d_0 = 0{\cdot}3$. The total number of points generated, as well as timings, are shown in Table I. One can see that the speed increases with the number of points generated. This is due to the smaller number of boundary checks required for the larger point clouds.

5.2. F117

A cloud of points for the aerodynamic simulation of inviscid, transonic flow past an F117 fighter is considered next. The point density was specified through the combination of a background grid and background sources. The surface triangulation consisted of approximately 9·7 kpts and 18·6 ktria. The advancing front point generator added another 20 kpts. Figure 5(a) shows the surface mesh, 5(b) the global cloud of points, as well as (5(c)) some slices through the volume. The spatial variation of point density is clearly visible.

5.3. Sphere

Here, the generation of a cloud of points suitable for a viscous, low-Reynolds-number incompressible flow simulation past a sphere was required. For symmetry reasons, only a quarter

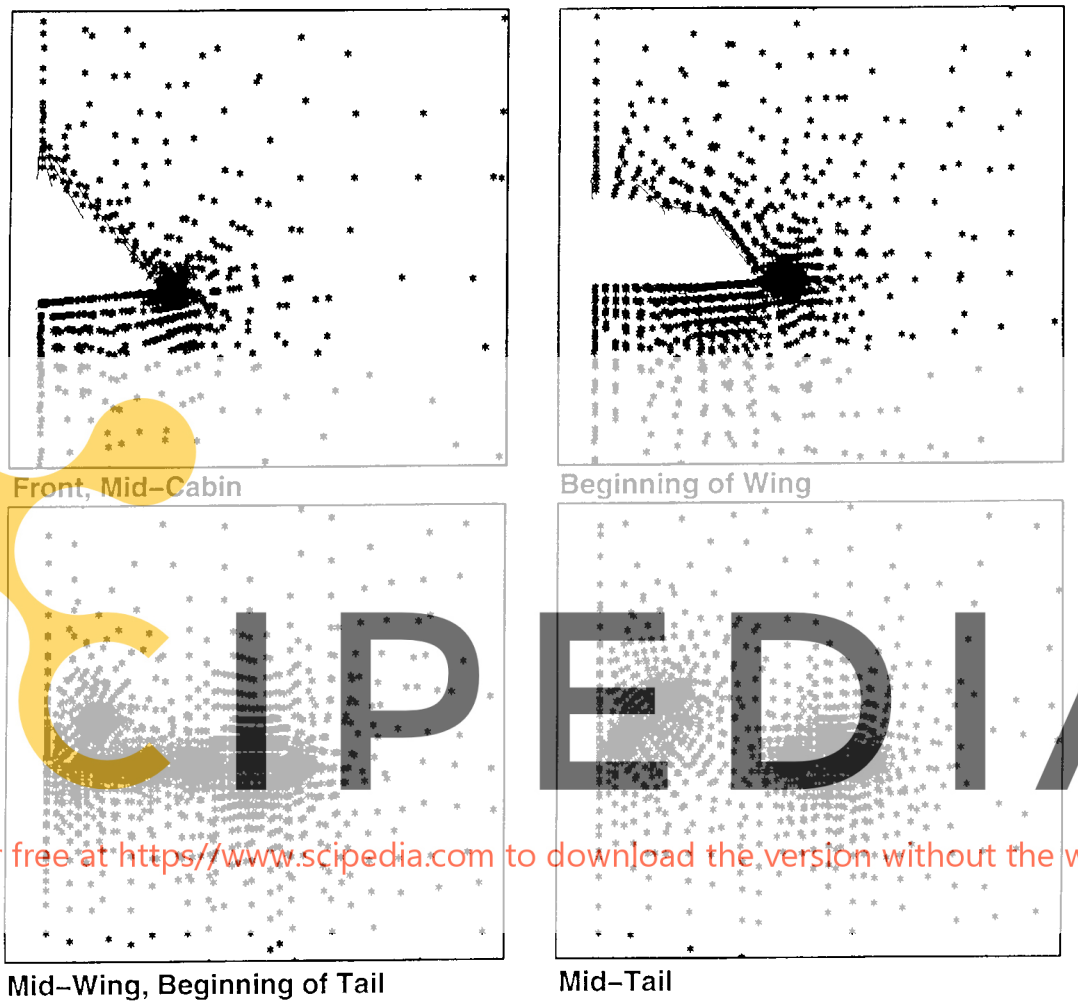

Figure 5(c). F117: four slices through the volume

of the sphere is modelled. The point density was again specified through the combination of background grids and sources. The surface triangulation had approximately 20 kpts and 40 ktria. The final cloud of points had 116 kpts. Figure 6 shows the surface mesh, as well as a slice through the volume. The spatial variation of point density is clearly visible.

5.4. Generic ship hull

The final example shows the cloud of points for a generic ship shape, useful for free surface hydrodynamic simulations. The surface triangulation had approximately 40 kpts and 80 ktria. The final cloud of points had 129 kpts. Figure 7(a) shows the surface mesh, and 7(b) some slices through the volume.

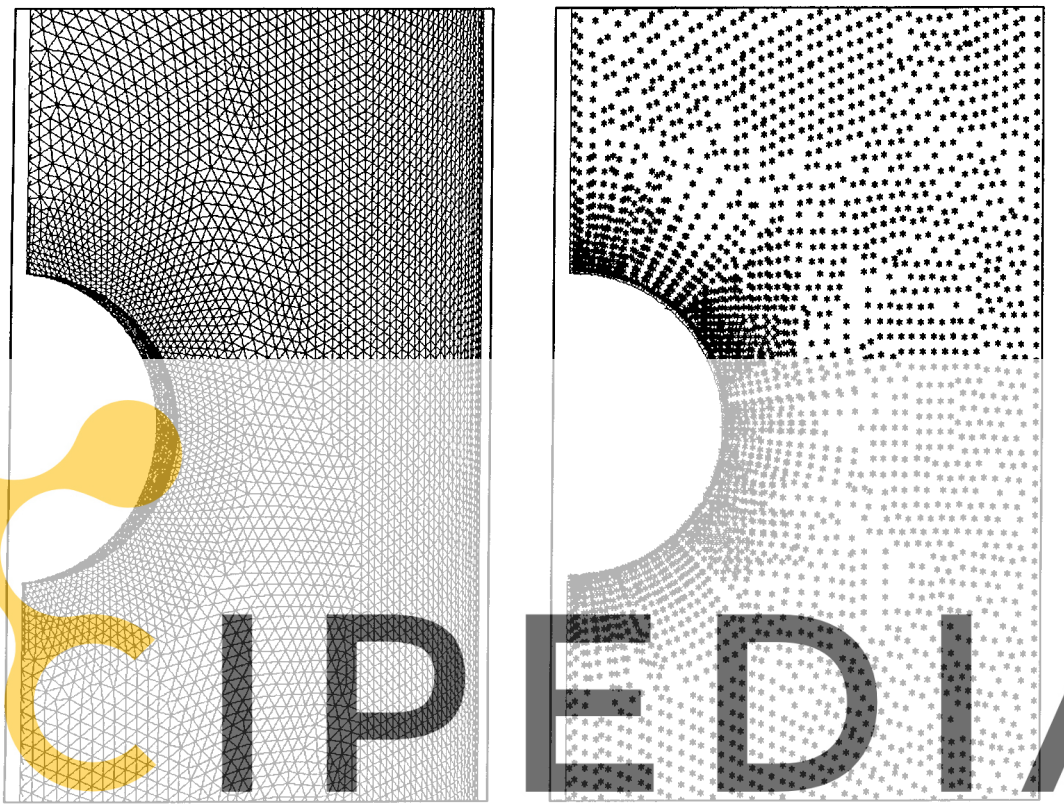

Figure 6. Sphere in tube

6. CONCLUSIONS AND OUTLOOK

An advancing front point generator has been developed. The input required consists of the specification of the desired mean point distance in space and an initial triangulation of the surface. One point at a time is removed and, if possible, surrounded by admissible new points. This operation is repeated until no active points are left. The scheme is about an order of magnitude faster than volume grid generators based on the advancing front technique. The main reason for this large disparity in speeds is the extra volume coherency (elements crossing front) check of grid generators, which is costly. These checks are not required for clouds of points. This observation confirms, to a certain degree, the premise that point generation is simpler than element and point generation. Although very competitive, the present procedure may be parallelized along the lines of unstructured grid generators with the advancing front,[15,19] i.e. via domain decomposition of background grids.

Future work will consider:

(i) the fast selection of local clouds of points allowing for different least-squares based polynomical interpolations of the point unknowns

Figure 7(a). Generic ship hull: surface triangulation

(ii) the development of metrics to assess the quality of clouds of points for the solution of partial differential equations (e.g. local variance of point distance in different spatial directions)
(iii) the generation of clouds of points that exhibit a high degree of spatial anisotropy, such as those required for Reynolds-averaged Navier–Stokes simulations.

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 1997. The support for this visit is gratefully acknowledged.

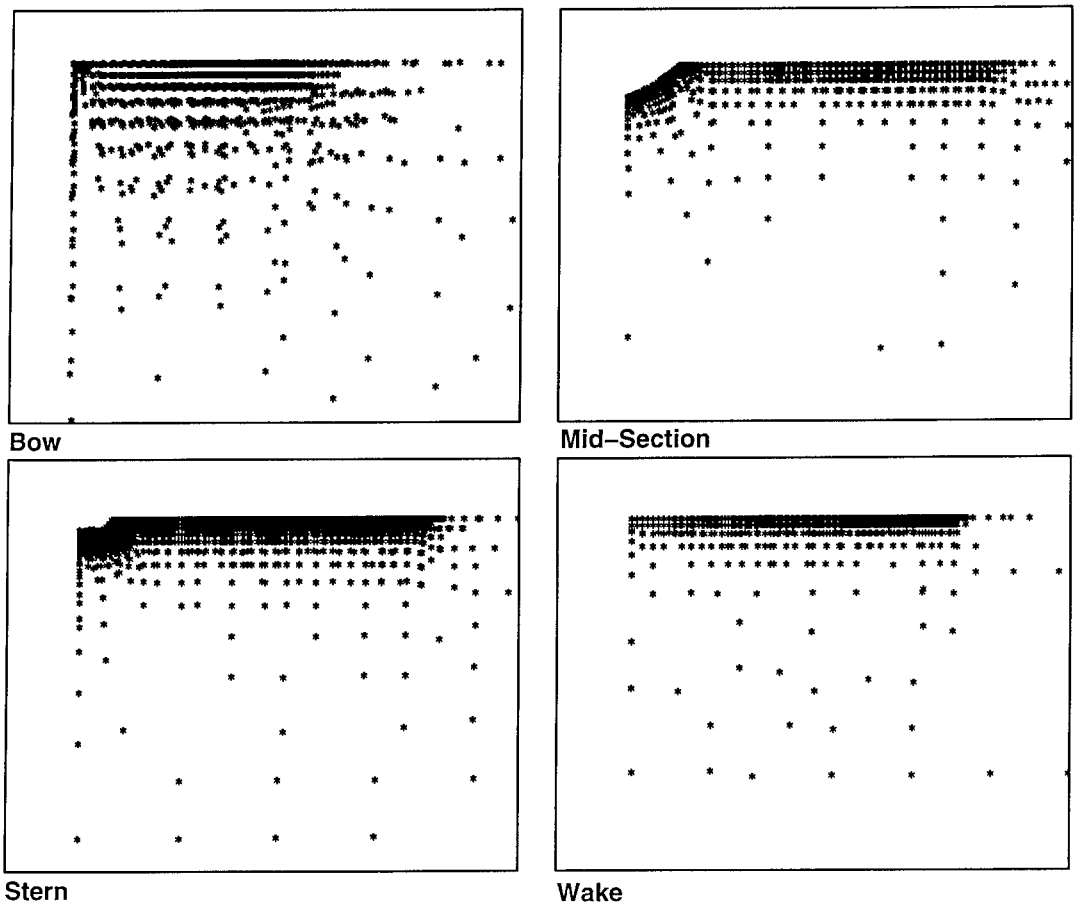

Figure 7(b). Generic ship hull: four slices through the volume

REFERENCES

1. R. A. Nay and S. Utku, 'An alternative for the finite element method', *Variat. Methods Eng.*, **1** (1972).
2. J. Batina, 'A gridless Euler/Navier–Stokes solution algorithm for complex aircraft configurations', *AIAA*-93-0333 (1993).
3. T. Belytschko, Y. Lu and L. Gu, 'Element free Galerkin methods', *Int. j. numer. methods eng.*, **37**, 229–256 (1994).
4. C. A. Duarte and J. T. Oden, 'H_p clouds–A meshless method to solve boundary-value problems', *TICAM-Rep.* 95-05 (1995).
5. 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. numer. methods eng.*, **39**, 3839–3866 (1996).
6. E. Oñate, S. Idelsohn, O.C. Zienkiewicz, R. L. Taylor and C. Sacco, 'A stabilized finite point method for analysis of fluid mechanics problems', *Comput. Methods Appl. Mech. Eng.*, **139**, 315–346 (1996).

7. 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 Comput. Methods Eng.*, **3**(1), 3–80 (1996).
8. M. A. Yerry and M. S. Shepard, 'Automatic three-dimensional mesh generation by the modified-octree technique', *Int. j. numer. methods eng.*, **20**, 1965–1990 (1984).
9. R. Löhner and P. Parikh, 'Three-dimensional grid generation by the advancing front method', *Int. j. numer. methods fluids*, **8**, 1135–1149 (1988).
10. J. Peraire, K. Morgan and J. Peiro, 'Unstructured finite element mesh generation and adaptive procedures for CFD', *AGARD-CP-464*, **18** (1990).
11. P. L. George, F. Hecht and E. Saltel, 'Fully automatic mesh generation for 3D domains of any shape', *Impact Comput. Sci. Eng.*, **2**(3), 187–218 (1990).
12. M. S. Shepard and M. K. Georges, 'Automatic three-dimensional mesh generation by the finite octree technique', *Int. j. numer. methods eng.*, **32**, 709–749 (1991).
13. P. L. George, F. Hecht and E. Saltel, 'Automatic mesh generator with specified boundary', *Comput. Methods Appl. Mech. Eng.*, **92**, 269–288 (1991).
14. N. P. Weatherill, 'Delaunay triangulation in computational fluid dynamics', *Comput. Math. Appl.*, **24**(5/6), 129–150 (1992).
15. R. Löhner, J. Camberos and M.. Merriam, 'Parallel unstructured grid generation', *Comput. Methods Appl. Mech. Eng.*, **95**, 343–357 (1992).
16. R. Löhner, 'Matching semi-structured and unstructured grids for Navier–Stokes calculation', *AIAA*-93-3348-CP (1993).
17. N. Weatherill, O. Hassan, M. Marchant and D. Marcum, 'Adaptive inviscid flow solutions for aerospace geometries on efficiently generated unstructured tetrahedral meshes', *AIAA*-93-3390-CP (1993).
18. S. Pirzadeh, 'Viscous unstructured three dimensional grids by the advancing-layers method', *AIAA*-94-0417 (1994).
19. A. Shostko and R. Löhner, 'Three-dimensional parallel unstructured grid generation', *Int. j. numer. methods eng.*, **38**, 905–925 (1995).
20. R. Löhner, 'Extensions and improvements of the advancing front grid generation technique', *Commun. Numer. Methods Eng.*, **12**, 683–702 (1996).
21. R. Löhner, 'Automatic unstructured grid generators', *Finite Elements in Analysis and Design*, **25**, 111–134 (1997).

