Finite Elements in Analysis and Design 46 (2010) 140–151

Contents lists available at ScienceDirect

Finite Elements in Analysis and Design

journal homepage: www.elsevier.com/locate/finel

Advancing front techniques for filling space with arbitrary separated objects

Rainald Löhner[a,*], Eugenio Oñate[b]

[a]CFD Center, Department of Computational and Data Science, MS 6A2, College of Science, George Mason University, Fairfax, VA 22030-4444, USA
[b]CIMNE, Universidad Politécnica de Catalunya, Barcelona, Spain

ARTICLE INFO

Available online 31 July 2009

ABSTRACT

A review is given of advancing front techniques for filling space with arbitrary separated objects. Over the last decade, these techniques have reached a considerable degree of maturity and are being used to generate clouds of points for SPH and FPM simulations, as well as spheres, ellipsoids, objects defined by a collection of spheres or polyhedral objects for DEM simulations. Algorithmic as well as implementational aspects are discussed. Techniques to obtain maximum packing, such as closest object placement (during generation) and move/enlarge (after generation) are also considered. Several examples are included that demonstrate the capabilities developed.

1. Introduction

Many simulation techniques in computational mechanics require a space-filling cloud of arbitrary, separated objects. For continuum problems described by partial differential equations, so-called 'gridless', 'mesh free', smooth particle hydrodynamics (SPH) [13] or finite point method (FPM) solvers (see [1,2,7,13,16,22,26–28]) have been developed. These solvers need a space-filling cloud of points. For discontinua so-called discrete element methods (DEMs) (see, e.g. [3–6,15,30]) are used extensively to characterize and compute granular media, soil, concrete, and other materials not amenable to a characterization via classic continuum formulations. Due to their simplicity, the objects most often considered for DEMs are spheres, ellipsoids, or superquadrics of the form [4]

$$\left(\frac{x}{a}\right)^m + \left(\frac{y}{a}\right)^m + \left(\frac{z}{c}\right)^m = 1. \quad (1)$$

In some cases, the objects are described via an agglomeration of spheres of possibly different sizes [23]. However, in principle, they could be polyhedra or, for that matter, any other arbitrary shape.

The task is therefore to fill a prescribed volume in an automatic way with points or arbitrary separated objects. Given the maturity of unstructured mesh generators [10–12,17–19,29,31,33,34], an obvious way to generate clouds of points or objects would be via a mesh: generate a mesh of tetrahedra, then remove the elements and keep the points or objects desired. Such a scheme has been used in almost all FPM or meshless results published to date, and works well for the generation of clouds of points. However, for the generation of spheres no-penetration may be difficult to enforce. Moreover, for arbitrary objects of random, different sizes this technique will not work. Therefore, alternative ways of filling space with arbitrary, separated objects must be developed.

Several techniques have been used to place objects in space. The so-called 'fill and expand' or 'popcorn' technique [30] starts by generating a coarse mesh for the volume to be filled. This subdivision of the volume into large, simple polyhedra (elements), is, in most cases, performed with hexahedra. The objects required (points, spheres, ellipsoids, polyhedra, etc.) are then placed randomly in each of these elements. These are then expanded in size until contact occurs or the desired fill ratio has been achieved. An obvious drawback of this technique is the requirement of a mesh generator to initiate the process. A second class of techniques are the 'advancing front' or 'depositional' methods [8,9,14,20,23]. Starting from the surface, objects are added where empty space still exists. In contrast to the 'fill and expand' procedures, the objects are packed as closely as required during introduction. Depending on how the objects are introduced, one can mimic gravitational or magnetic deposition, layer growing, or size-based growth. Furthermore, so-called radius growing can be achieved by first generating a coarse cloud of objects, and then growing more objects around each of these [23]. In this way, one can simulate granules or stone. In the sequel, it is assumed that any object is defined either as a point, a sphere, a collection of spheres, or by a coarse mesh of tetrahedra (allowing for a clear identification of external faces). The consideration of arbitrary bodies described via tetrahedra as compared to a collection of spheres opens the way to completely general shapes and allows for strict penetration checks. Starting from the boundary, i.e. the initial 'front' of faces, new

* Corresponding author.
E-mail address: rlohner@gmu.edu (R. Löhner).

0168-874X/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.finel.2009.06.032

objects (and possibly their external faces) are added, until no further objects can be introduced. In the same way as the advancing front technique for the generation of volume grids removes one face at a time to generate elements, the advancing front technique for filling space with arbitrary separated objects removes one object (points, spheres) or object face (general objects) at a time, attempting to introduce as many objects as possible in its immediate neighbourhood.

As this class of scheme belongs to the family of advancing front techniques, it is not surprising that major parts of the algorithms used to fill space with arbitrary, separated objects are very similar to those used to generate tetrahedral space-filling grids [17–19,29].

2. The algorithm

Assume as given:

- A specification of the desired object type(s) via points, spheres, collection of spheres or tetrahedral grids.
- A specification of the desired object size in space (e.g. via a combination of background grids and sources [19]).
- A specification of the desired mean distance between objects in space.
- An initial triangulation of the surface, with the face normals pointing towards the interior of the domain to be filled with objects.

With reference to Fig. 1, which shows the filling of a simple 2-D domain with trapezoidal elements, the complete advancing front space-filling algorithm may be summarized as follows:

- Determine the required object size and distance between objects near the triangulation;
- `while`: there are active objects/faces in the front:
  - Remove from the front the object `ioout` or face `ifout` with the smallest specified object size;
  - With the specified object size and mean object distance: determine the coordinates of `nposs` possible new neighbouring objects; this is done using a stencil, some of which are shown in Fig. 2;
  - Find all existing objects/faces in the neighbourhood of `ifout`;
  - `do`: For each one of the possible new neighbour objects `ionew`:
    - If there exists an object closer than a minimum distance `dminp` from any point of `ionew`, or if `ionew` is penetrating existing objects: ⇒ skip `ionew`;
    - If the new object is outside the computational domain: ⇒ skip `ionew`;
    - Determine the required object size and mean object distance for (the faces) of `ionew`;
    - Increment the number of objects by one;
    - Introduce the point(s) of `ionew` to the list of coordinates;
    - For objects defined via tetrahedra:
      - Introduce the faces of `ionew` to the list of active front faces;
      - Introduce the elements of `ionew` to the list of elements;
    - For objects defined via points/spheres:
      - Introduce `ionew` to the list of active objects;
    - `enddo`
  - `endwhile`

The main search operations required are:

- Finding the active object/face with the smallest mean distance to neighbours;
- Finding the existing points/faces in the neighbourhood of `ioout`, `ifout`;

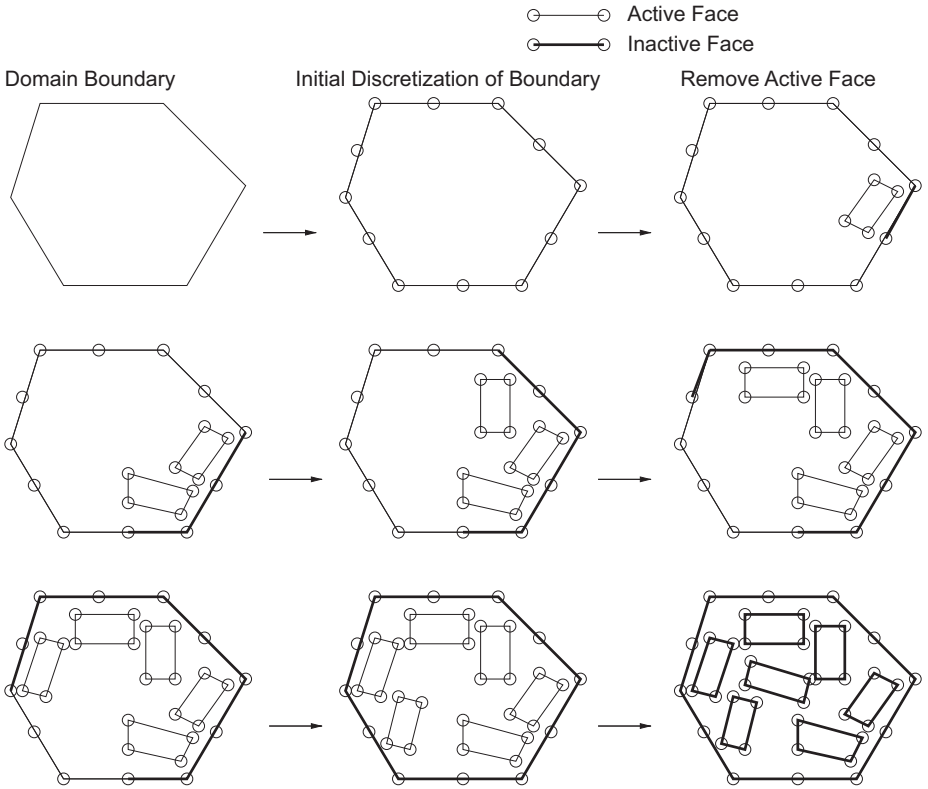

Fig. 1. Advancing front space-filling with trapezoids.

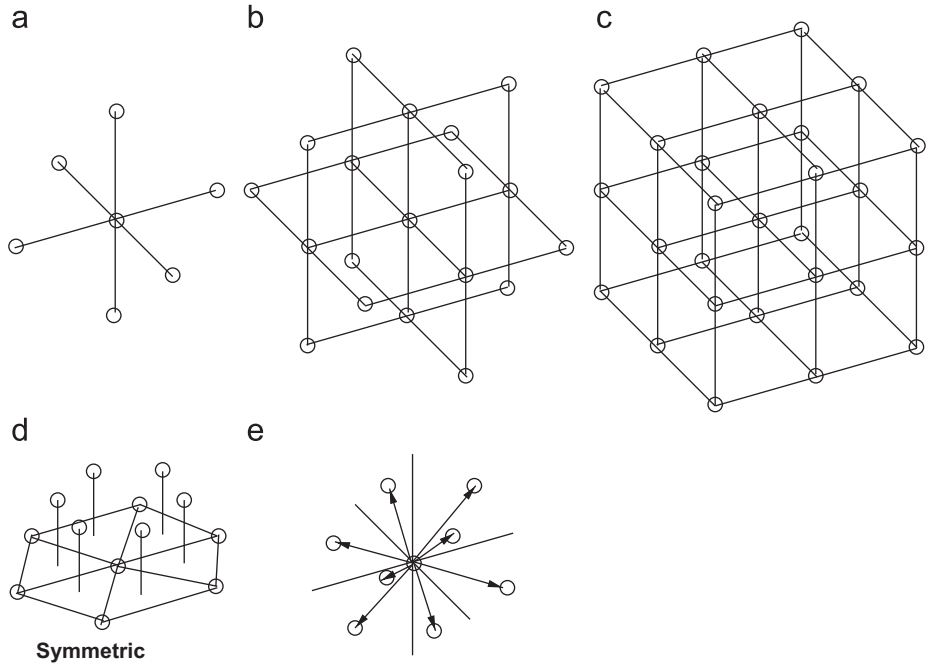

Fig. 2. Point stencils (a) Cartesian [6]; (b) Cartesian [18]; (c) Cartesian [26]; (d) Tetrahedral [17]; and (e) Random.

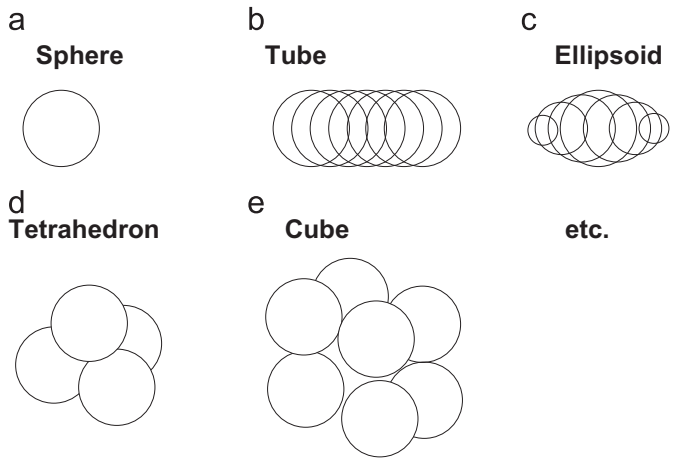

Fig. 3. Arbitrary objects as a collection of spheres.

These search operations can be performed efficiently using heap-lists, octrees and linked lists respectively (see [10–12,17–19,29, 31–34] for more details).

3. Point stencils

A number of different stencils may be contemplated (see Fig. 2). Each one of these corresponds to a particular space-filling object configuration. For the generation of points and spheres, it was found that the 8-point stencil leads to the smallest amount of rejections and unnecessary testing [20,23]. For arbitrary objects it was found that the stencil that takes n randomly selected directions yields the most densely packed 'grids', i.e. the highest volume fill ratios. In most instances, the orientation of the objects in space is assumed to be random. In order to achieve this the 'unit object' is scaled to the required distance and then rotated randomly around its centroid.

4. Front crossing checks

A crucial requirement for a general space-filling object generator is the ability to generate objects in such a way that they do not cross or interpenetrate each other. This requirement is the same as that for advancing front grid generators. Therefore, the same techniques can be employed in this context. For objects defined via points or spheres, the penetration checks are based on the distance between points and the associated radii. Specialized penetration/closeness checks are available for ellipsoids [8], but for general polyhedra the faces have to be triangulated and detailed face/face checks are unavoidable. A possible recourse is to approximate arbitrary objects by a collection of spheres (see Fig. 3).

When adding a new object in space, the penetration/closeness checks are carried out for all spheres comprising the object. The new object is only added if all spheres pass the required tests. Experience indicates that such a definition of geometrically complex objects via

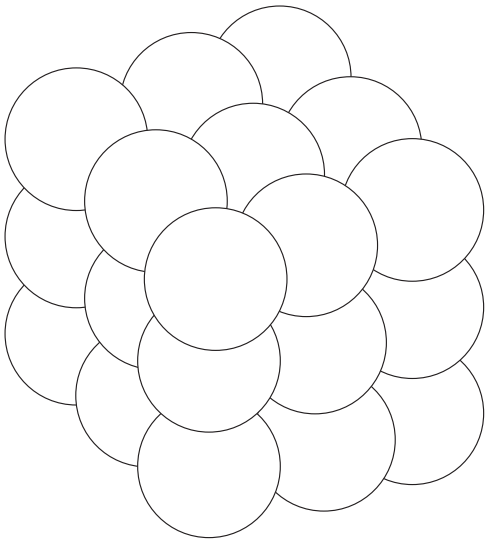

Fig. 4. Hexahedron composed of 27 spheres.

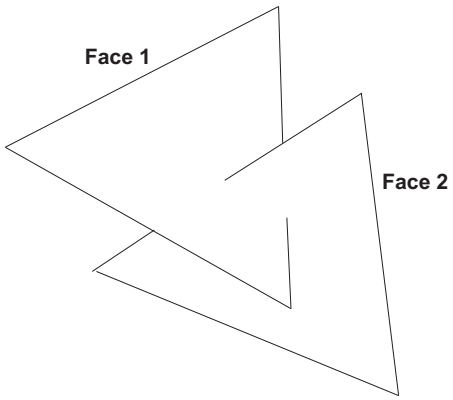

Fig. 5. Face crossing checks.

spheres of different sizes has difficulties in guaranteeing strict non-penetration. As an example, it was found that for hexahedra at least 27 spheres were required to achieve reliable object generation (see Fig. 4).

This leads naturally to a definition of objects via a small mesh of tetrahedra and defined external faces. Once a new object is introduced, all new faces are tested against all current faces in order to see if crossing occurs. Using octrees, linked lists, or other near-optimal data structures as well as local bounding boxes and filtering operations [25], the number of faces to be tested can be reduced drastically. The overall complexity of the algorithm is thus reduced to $O(N \ln N)$, where N is the number of objects.

Given the list of new object faces, and the list of current front faces in the neighbourhood of the new object, each face from one list is tested against all faces from the other list. For any given face pair, crossing will occur if a side from one face pierces through another face (see Fig. 5).

In order to avoid unnecessary tests, the faces of the new object that are closest to the face being removed from the front are tested first.

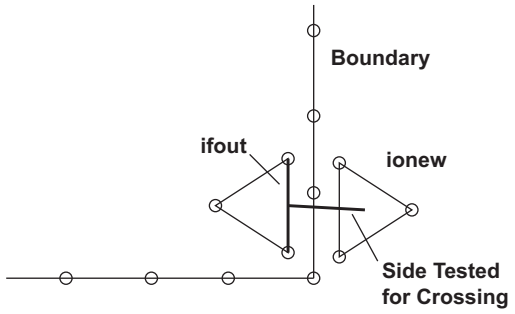

Fig. 6. Boundary consistency checks.

5. Boundary consistency checks

A second crucial requirement for a general space-filling object generator is the ability to only generate objects in the computational domain desired. The penetration checks described before do not guarantee the generation of objects that lie strictly inside the desired domain. This can be seen from Fig. 6, which shows a possible 2-D situation. While the faces of the new object `ionew` do not cross any existing faces, the new object lies outside the domain.

5.1. Objects defined via tetrahedra

Boundary crossings may be detected by constructing a 'side' that connects the centroid of the face being removed (`ifout` in Fig. 6) to the centroid of the new object (`ionew`). If this side crosses any of the close faces, the new object is rejected.

5.2. Objects defined via spheres

For objects defined via spheres, the penetration of spheres into the boundary triangulation needs to be checked. If we assume that the object to be removed from the list of active objects `ionew` lies inside the domain, a new object `ionew` 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 `ionew` and can see `ionew`. This criterion is made more stringent by introducing a tolerated closeness or tolerated distance d_t of new objects to the exterior faces of the domain. Testing for boundary consistency is then carried out using the following algorithm (see Fig. 7):

- Obtain all the faces close to `ioout`;
- Filter, from this list, the faces that are pointing away from `ioout`;
- `do`: For each of the close faces:
  - Obtain the normal distance d_n from `ionew` to this face;
    `if`: $d_n < 0$: `ionew` lies outside the domain
    $\Rightarrow$ reject `ionew` and exit;
    `elseif`: $d_n > d_t$: `ionew` if far enough from the faces
    $\Rightarrow$ proceed to the next close face;
    `elseif`: $0 \leq d_n \leq d_t$: obtain the closest distance d_{min} of `ionew` to this face;
    `if`: $d_{min} < d_t$: `ionew` is too close to the boundary
    $\Rightarrow$ reject `ionew` and exit;
  - `endif`
- `enddo`

Typical values for d_t are $0.707 d_0 \leq d_t \leq 0.9 d_0$, where d_0 denotes the desired mean average distance between points.

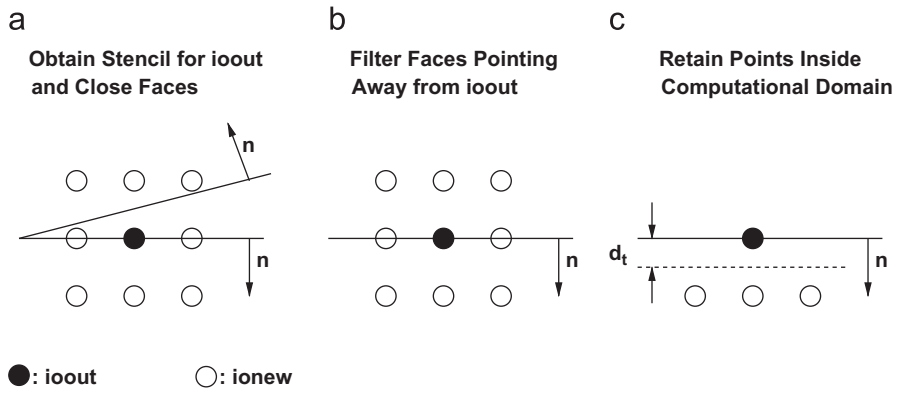

Fig. 7. Boundary consistency checks.

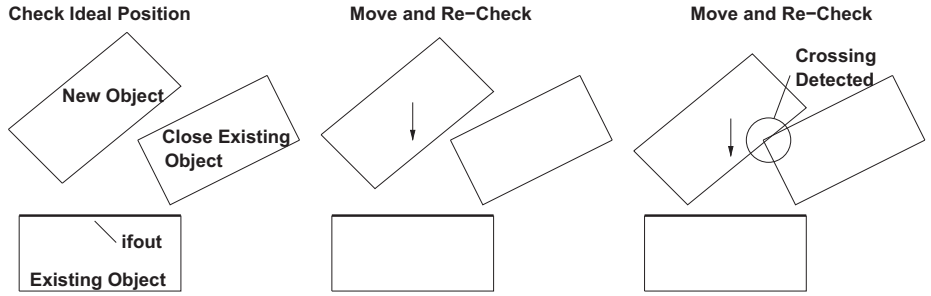

Fig. 8. Closest object placement.

6. Maximum compaction techniques

For SPH and FPM applications, the use of a stencil is sufficient to ensure a proper space filling (discretization) of the computational domain. However, many applications that consider not points but volume-occupying objects such as spheres, ellipsoids and polyhedra, require a preset volume fraction occupied by these objects, and, if possible, a minimum number of contacts. The modelling of discontinua via discrete element methods represents a typical class of such applications. Experience indicates that the use of point stencils does not yield the desired volume fractions and contact neighbours. Two complementary techniques have proven useful to achieve these goals: closest object placement and move/enlarge post-processing.

6.1. Closest object placement

Closest object placement attempts to position new objects as close as possible to existing ones (see Fig. 8). The initial position for new objects is taken from a stencil as before. If this position passes all the crossing checks, the new object is moved closer to the face being removed from the active front, and the test is repeated. Should the crossing tests fail for this location, the last acceptable position is taken as the final position for the new object. One should note that compared to the simple object placement, this 'closest object placement' does not represent a substantial increase in effort, as the existing objects/faces in the vicinity of the face being removed can be reused for the subsequent closeness/penetration tests, and in most cases the original position already leads to a rejection as the number of active faces is typically much larger than the available space for new objects.

6.2. Move/enlarge post-processing

Whereas closest object placement is performed while space is being filled with objects, post-processing attempts to enlarge and/or move the objects in such a way that a higher volume ratio of objects is obtained, and more contacts with nearest neighbours are established. The procedure, shown schematically in Fig. 9, may be summarized as follows:

- `while`: objects can be moved/enlarged:
- `do`: loop over the objects `iomov`:
  - Find the closest existing faces of `iomov`;
  - Move the object away from the closest existing faces/objects so that:
    - The minimum distance to the closest existing objects increases;
    - The faces of `iomov` do not penetrate faces from other objects;
  - Enlarge object `iomov` by a small percentage
    - If the faces/spheres of the enlarged object `iomov` penetrate other faces/spheres: revert to original size;
    - If the faces/spheres of the moved (and possibly enlarged) object `iomov` penetrate other faces/spheres: ⇒ skip `iomov`;
- `enddo`

The increase factors are progressively decreased for each new loop over the objects. Typical initial increase ratios are 5%. As the movement of objects is moderate (e.g. less than the size of the objects), the spatial search data structures (bins, octrees) required during space filling can be reused without modification.

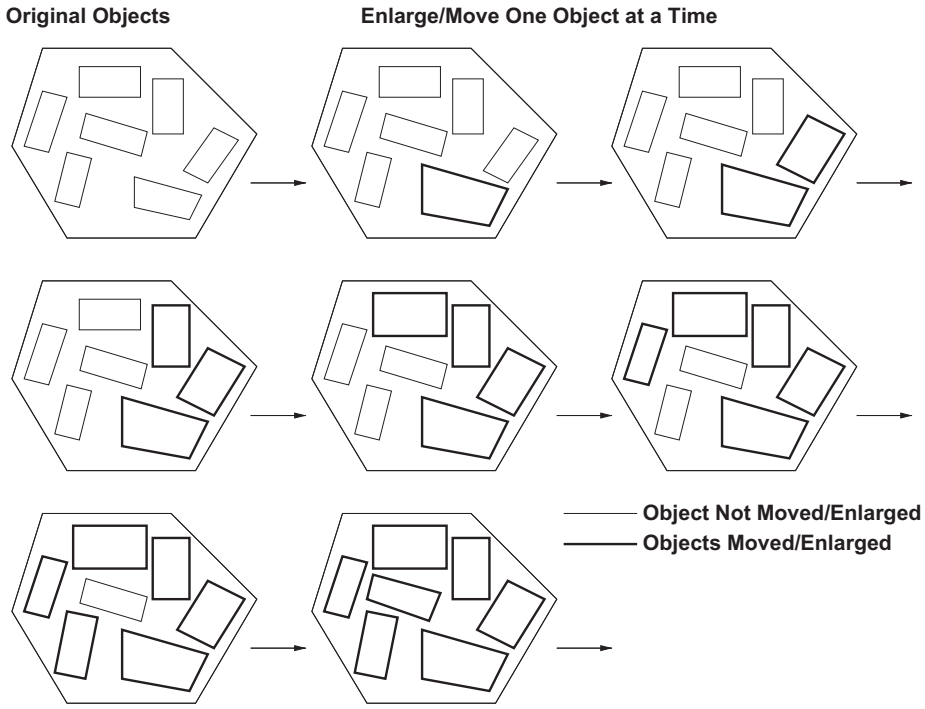

Fig. 9. Movement and enlargement of objects.

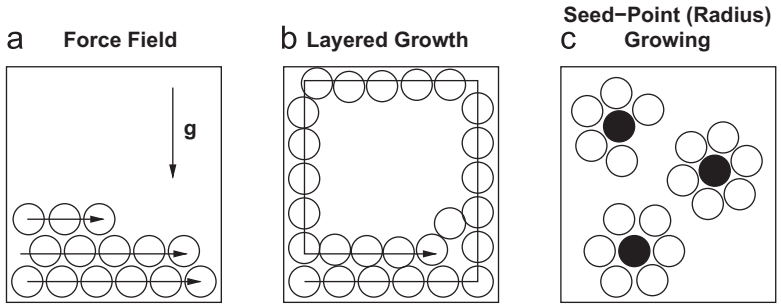

Fig. 10. Deposition patterns.

7. Deposition patterns

Depending on how the objects are removed from the active front, different deposition patterns can be achieved. The advancing front technique always removes the object with the smallest average distance (size) to new neighbours from the active front. Different size distributions will therefore lead to different deposition patterns (see Fig. 10). Gravitational deposition can be achieved by specifying a size distribution that decreases slightly in the direction of the gravity vector. The objects that are at the 'bottom' will then be removed first from the active front and surrounded by new objects. The same technique can be applied if magnetic fields are present. Layered growth can be obtained be assigning a slight increase in size based on the object number:

$$\delta = (1 + \varepsilon n_{obj})\delta_0, \tag{2}$$

where $\delta, \delta_0, n_{obj}, \varepsilon$ denote the size used, original size, object number and a very small number (e.g. $\varepsilon = 10^{-10}$). So-called radius growing, used to simulate granules or stone [30], can be achieved by first generating a coarse cloud of objects, and then using layered growth around each of these.

8. Examples

The proposed advancing front object generation algorithm has been used to fill space with points, spheres, ellipsoids and arbitrary polyhedral objects for many applications: FPM simulations for continua, DEM/DPM simulations for discontinua, granular media, etc. Of these, we show a representative cross-section.

8.1. Point cloud for F117

The first case considers the generation of a cloud of points for the aerodynamic simulation of inviscid, transonic flow past (half) an F117 fighter via FPM [22]. The point density (i.e. the average closeness of points) was specified through the combination of a background grid and background sources [25]. The surface triangulation consisted of approximately 50 Kpts and 100 Ktria. The advancing front point generator added another 200 Kpts using simple stencil placement. Fig. 11 shows the CAD definition of the computational domain, the global cloud of points, a close-up of the surface mesh, the cloud of points close to the plane, as well as some slices through the volume. The spatial variation of point density is clearly visible.

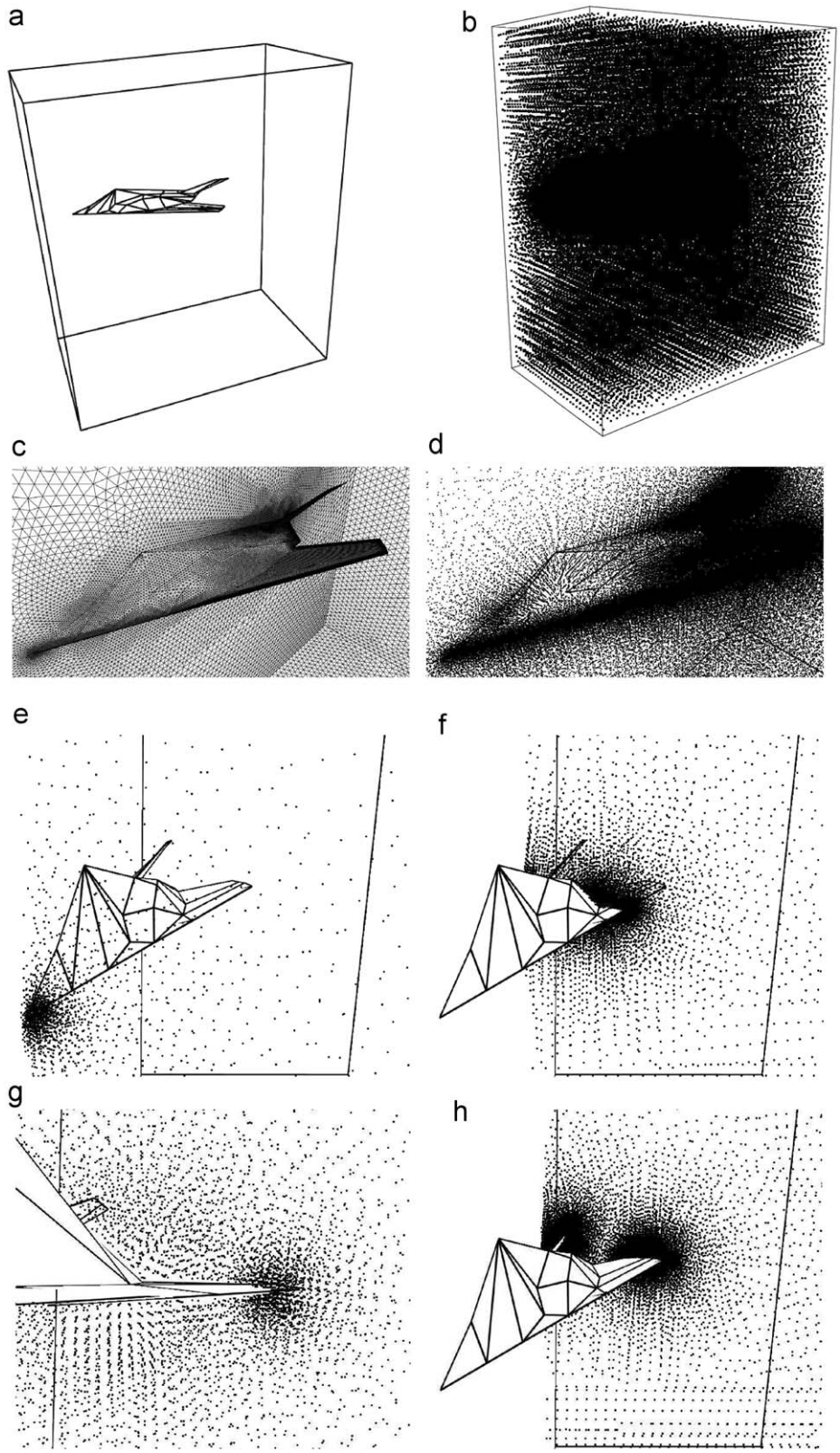

Fig. 11. (a) and (b) F117: CAD definition and global cloud of points. (c) and (d) F117: close-up of surface mesh and cloud of points. (e) and (f) F117: cuts at $x = 0, 120$. (g) and (h) F117: cut at $x = 120$ (Detail) and $x = 190$.

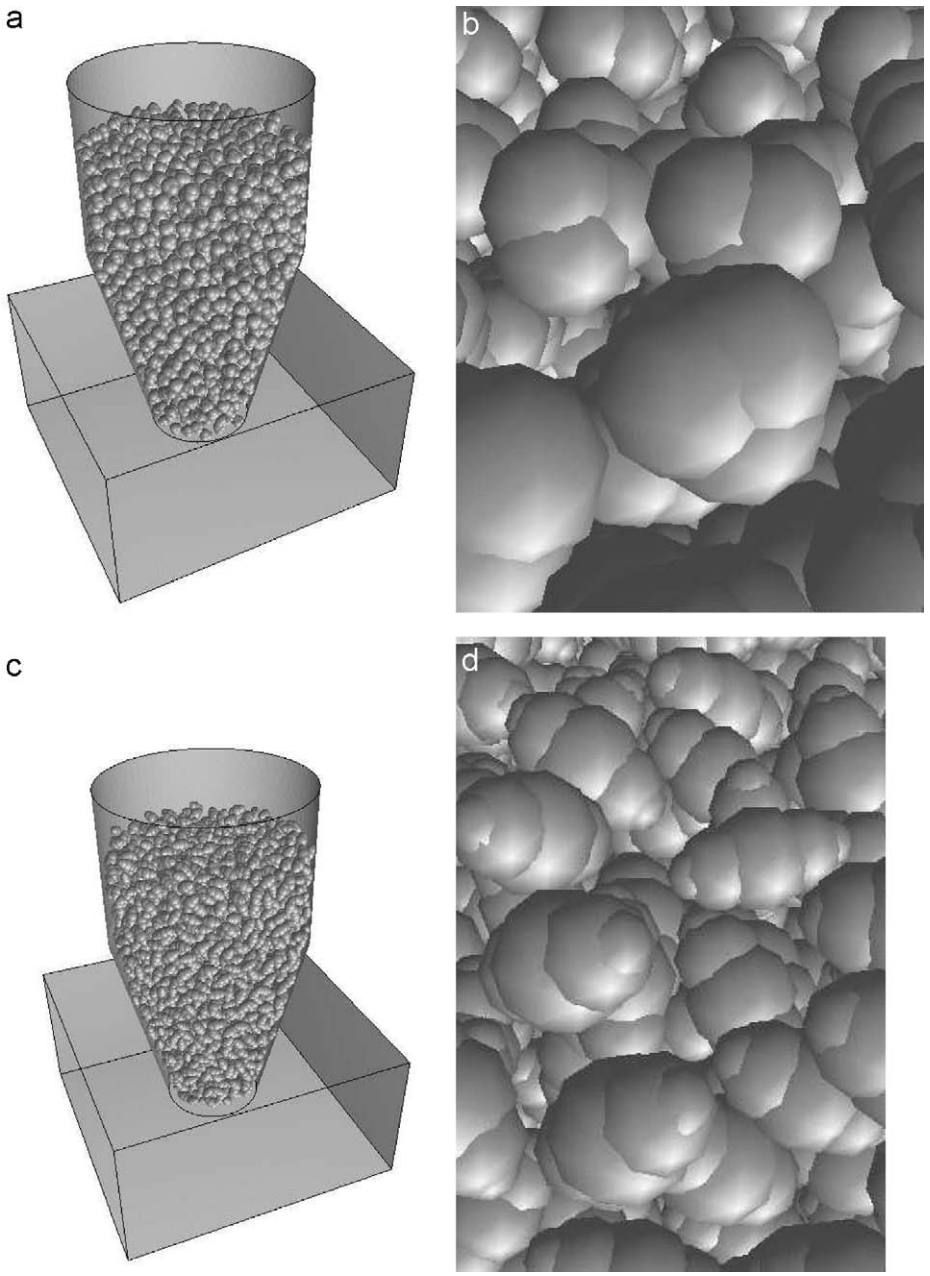

Fig. 12. (a) and (b) Hopper filled with beans. (c) and (d) Hopper filled with ellipsoids.

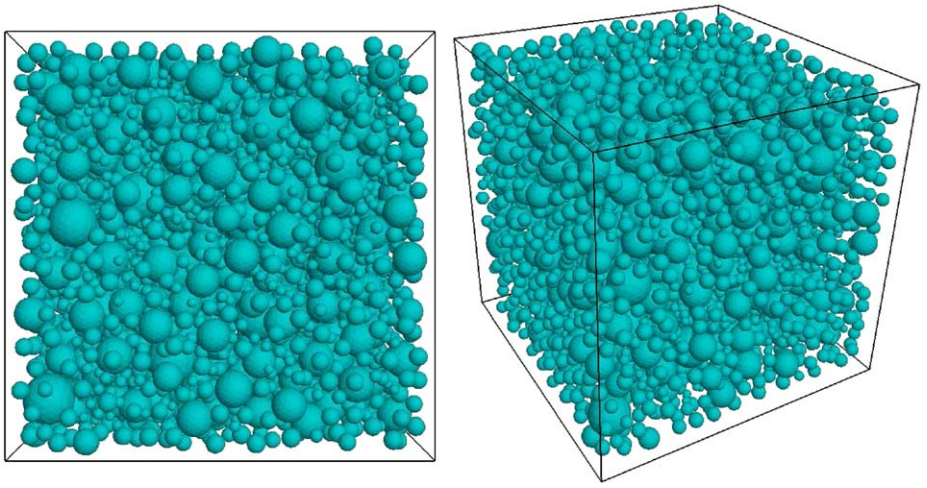

Fig. 13. Cube filled with spheres of different sizes.

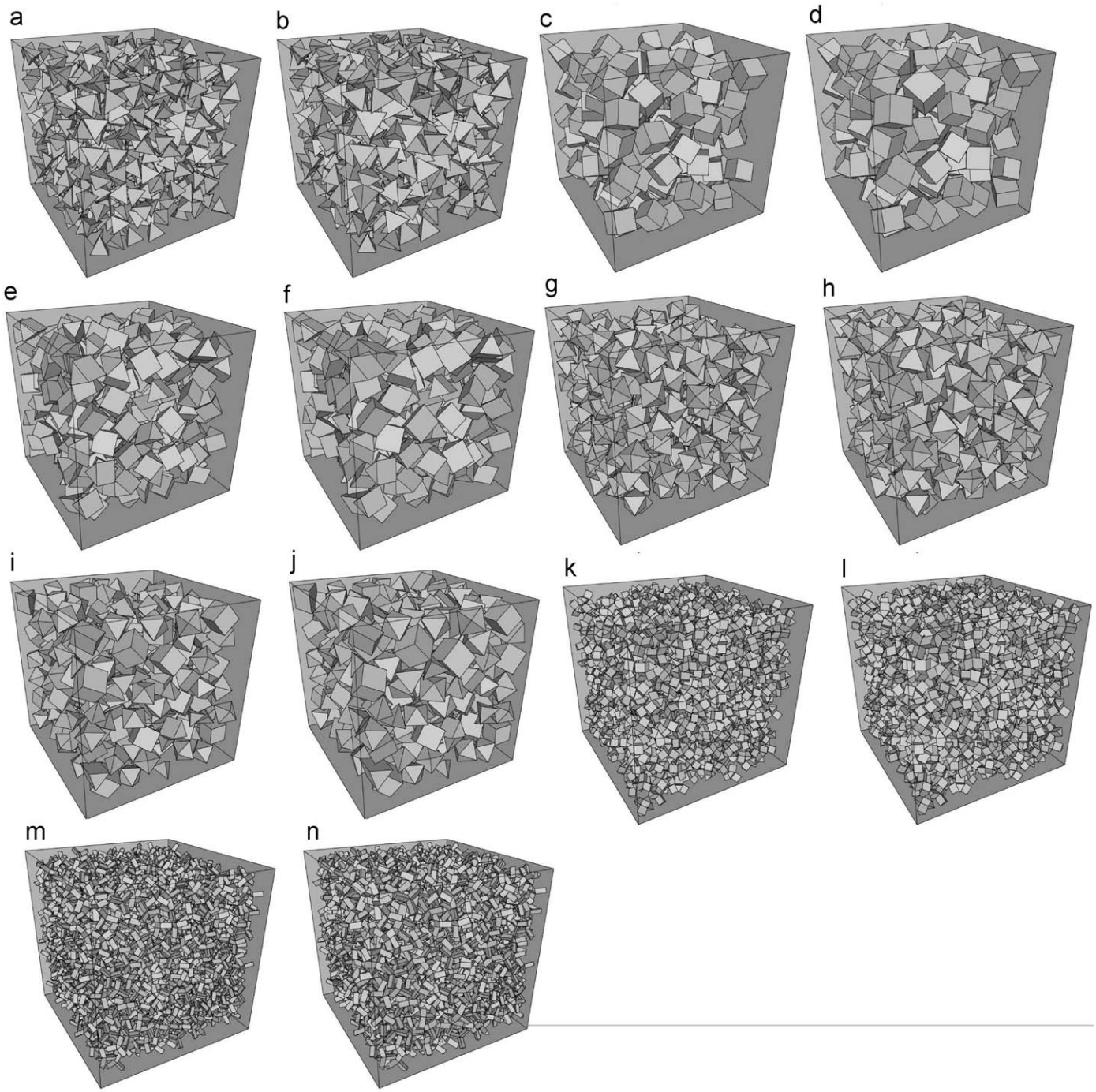

Fig. 14. (a) and (b) Cube filled with tetrahedra. (c) and (d) Cube filled with hexahedra. (e) and (f) Cube filled with prisms. (g) and (h) Cube filled with octahedra. (i) and (j) Cube filled with a mix of objects. (k) and (l) Cube filled with tetrapods (type I). (m,n): Cube Filled With Tetrapods (type II).

The complete generation process (boundary triangulation, volume fill, output of files, etc.) took 44 s on a PC with Intel P4 chip running at 3.2 Ghz, 1 Gbyte Ram, Linux OS and Intel Compiler.

8.2. Hopper filled with beans/ellipsoids

Granular materials that require simulation include grains, ground stone, woodchips, and many other materials [3,5]. Bridging in silos and hoppers can cause severe structural damage, and has always been a concern. Fig. 12a and b show a hopper configuration with

Table 1
Grid statistics for cube.

Type	nelem	vol-bef	vol-aft
tet	744	0.2055	0.3131
hex	157	0.3679	0.4610
pri	302	0.3064	0.4132
oct	340	0.3756	0.5089
mix	292	0.3515	0.4499
tetrp-1	1021	0.2425	0.3014
tetrp-2	1478	0.1598	0.2250

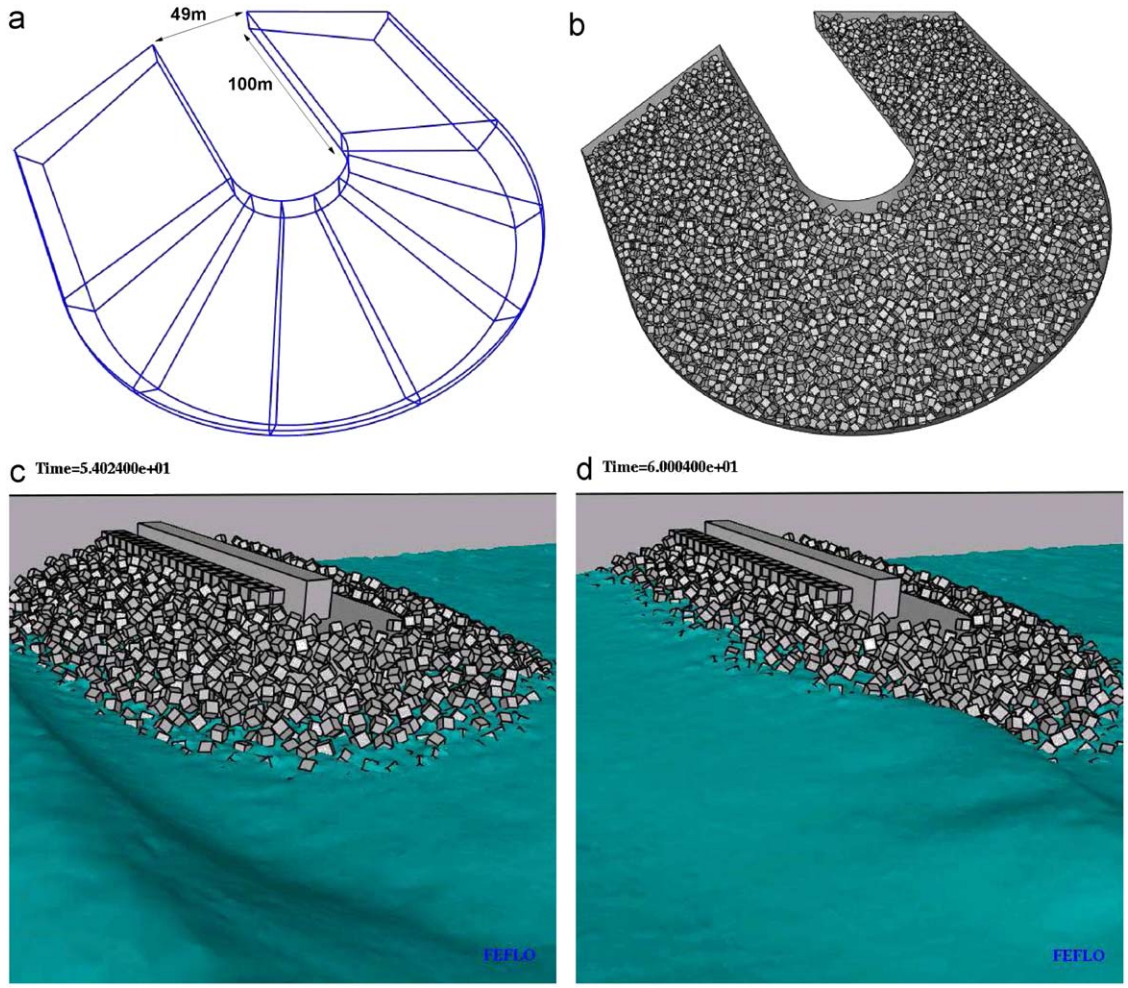

Fig. 15. (a) and (b) Breakwater: domain definition and resulting mesh. (c) and (d) Breakwater: results for a typical simulation.

bean-like objects composed of four spheres each. The total number of beans is 2124, i.e. 8496 spheres, and took 7 s to generate. Fig. 12c and d show the same configuration filled with ellipsoidal objects composed of five spheres each. The total number of ellipsoids is 2794, i.e. 13,970 spheres, and took 10 s to generate.

8.3. Cube filled with spheres of different sizes

Concrete and other aggregate materials often exhibit spherical or ellipsoidal objects of different sizes. The number of objects can be recorded in a statistical distribution of size vs. number of particles. Fig. 13 shows a case for concrete: three different average sphere sizes, each on average half the size of the previous one and each with a standard (Gaussian) variation of 10%, are used to fill the cube shown. The total number of spheres is 3958.

8.4. Cube filled with different objects

This configuration is used to illustrate the variety of elements that may be generated, as well as the effectiveness of the optimal packing procedures described. The prescribed object size was $\delta = 0.112$, and a uniformly random variation of the orientation was allowed. Fig. 14a–n show the object distributions obtained for tetrahedra, hexahedra, prisms, octahedra, a uniform mix of the former, as well two tetrapod-like objects with different thinness-ratios. The left figures show the meshes before the move and enlarge option is invoked, the right figures after. Some mesh statistics have been compiled in Table 1.

The generation times vary considerably depending on the element type and the amount of effort spent for optimum packing. For simple objects, like tetrahedra, hexahedra, prisms and octahedra, several tens of thousands may be generated per minute. For a more complex object like the tetrapod, CPU times can be 2–3 times as high. The post-processing (post-packing), on the other hand, is considerably more expensive: in some cases, it takes more than five times more CPU time to post-process a cloud of objects than to generate it.

8.5. Breakwater

The next configuration considered is a breakwater. The stones or concrete blocks that are laid down on the surface in order to attenuate wave impact are often strewn in a random way.

The domain considered is shown in Fig. 15a. The concrete blocks were assumed to be cubes of size $h = O(2.5\,\mathrm{m})$. The surface of the resulting mesh, which consists of 23,030 elements, is shown in Fig. 15b. The generated blocks were then used to generate a body-fitted, unstructured mesh for a flow simulation. The results of a typical simulation of the complete configuration using a volume of fluid method [24] are shown in Fig. 15c and d.

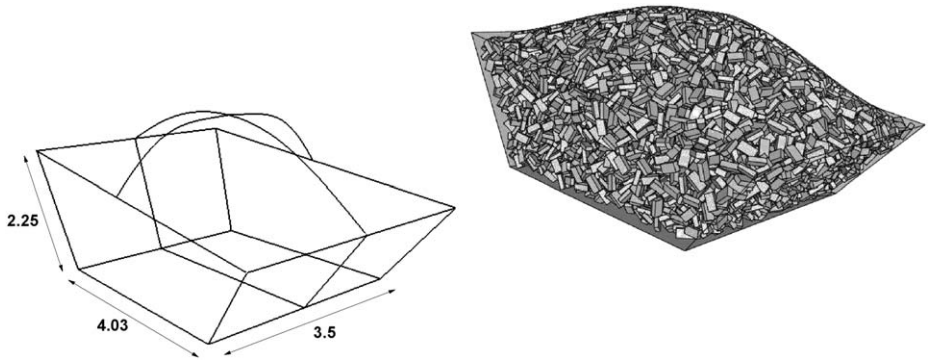

Fig. 16. Truckload of bricks.

8.6. Truckload of bricks

A question often raised is how much volume is wasted by loading materials or objects in a random as opposed to an ordered way. A typical case is a truckload of bricks. Packing them orderly would increase labour costs, while loading them in a random way may lead to higher transportation costs. Fig. 16 shows the domain considered as well as its characteristic dimensions. The volume comprised is of 43.81 m^3, allowing theoretically for 37,119 bricks of standard US size ($0.203 \times 0.102 \times 0.057$ (m)). Fig. 16 shows the distribution obtained (7893 bricks), which leads to a volume-fill ratio of (only) 21%.

9. Conclusions and outlook

Advancing front techniques for filling space with arbitrary separated objects have reached a considerable degree of maturity. These techniques are being used to generate clouds of points for SPH and FPM simulations, as well as spheres, ellipsoids, objects defined by a collection of spheres or polyhedral objects for DEM simulations. The input required consists of the specification of the desired object type, size and mean distance between objects in space together with an initial triangulation of the surface. As with ordinary advancing front techniques for the generation of grids, one point, sphere, object or face at a time is considered and, if possible, surrounded by admissible new objects. This operation is repeated until no further objects can be introduced. Two techniques to obtain maximum packing have been used extensively: closest object placement (during generation) and move/enlarge (after generation).

Several deposition or layering patterns can be achieved by selecting the order in which objects are eliminated from the active front.

Like any other class of techniques, these advancing front techniques for filling space with arbitrary separated objects may be improved further. Options currently under consideration include:

- Generation of clouds of points that exhibit a high degree of spatial anisotropy, such as those required for Reynolds-Averaged Navier–Stokes simulations.
- Alignment of objects, in order to model crystal growth, deposition patterns, or external forces.
- Extension of the library of objects in order to include more complex, yet 'standard' shapes.
- Parallel object generation along the lines of unstructured grid generators with the advancing front technique [21], i.e. via domain decomposition.

References

[1] J. Batina, A gridless Euler/Navier–Stokes solution algorithm for complex aircraft configurations, in: AIAA-93-0333, 1993.
[2] T. Belytschko, Y. Lu, L. Gu, Element free Galerkin methods, Int. J. Numer. Methods Eng. 37 (1994) 229–256.
[3] P.W. Cleary, Discrete element modeling of industrial granular flow applications, TASK Q. S. Bull. 2 (1998) 385–416.
[4] P.W. Cleary, M.D. Sinnott, R.D. Morrison, DEM prediction of particle flows in grinding processes, Int. J. Numer. Methods Fluids 58 (3) (2008) 319–353.
[5] P.A. Cundall, O.D.L. Stack, A discrete numerical model for granular assemblies, Geotechnique 29 (1) (1979) 47–65.
[6] P.A. Cundall, Formulation of a three-dimensional distinct element model—part I: a scheme to detect and represent contacts in a system composed of many polyhedral blocks, Int. J. Rock Mech. Min. Sci. 25 (1988) 107–116.
[7] C.A. Duarte, J.T. Oden, H_p clouds, a meshless method to solve boundary-value problems, TICAM-Report 95–05, 1995.
[8] Y.T. Feng, K. Han, D.R.J. Owen, An advancing front packing of polygons, ellipses and spheres, in: B.K. Cook, R.P. Jensen (Eds.), Discrete Element Methods, ASCE, New York, 2002, pp. 93–98.
[9] Y.T. Feng, K. Han, D.R.J. Owen, Filling domains with disks: an advancing front approach, Int. J. Numer. Methods Eng. 56 (2003) 699–713.
[10] P.L. George, F. Hecht, E. Saltel, Fully automatic mesh generation for 3D domains of any shape, Impact Comput. Sci. Eng. 2 (3) (1990) 187–218.
[11] P.L. George, F. Hecht, E. Saltel, Automatic mesh generator with specified boundary, Comput. Methods Appl. Mech. Eng. 92 (1991) 269–288.
[12] P.L. George, H. Borouchaki, Delaunay Triangulation and Meshing, Editions Hermes, Paris, 1998.
[13] R.A. Gingold, J.J. Monahghan, Shock simulation by the particle method SPH, J. Comput. Phys. 52 (1983) 374–389.
[14] K. Han, Y.T. Feng, D.R.J. Owen, Sphere packing with a geometric based compression algorithm, Powder Technol. 155 (1) (2005) 3–41.
[15] K. Han, Y.T. Feng, D.R.J. Owen, Coupled lattice Boltzmann and discrete element modelling of fluid–particle interaction problems, Comput. Struct. 85 (11–14) (2007) 1080–1088.
[16] W.K. Liu, Y. Chen, S. Jun, J.S. Chen, T. Belytschko, C. Pan, R.A. Uras, C.T. Chang, Overview and applications of the reproducing kernel particle methods, Arch. Comput. Methods Eng. 3 (1) (1996) 3–80.
[17] R. Löhner, P. Parikh, Three-dimensional grid generation by the advancing front method, Int. J. Numer. Methods Fluids 8 (1988) 1135–1149.
[18] R. Löhner, Extensions and improvements of the advancing front grid generation technique, Comm. Numer. Methods Eng. 12 (1996) 683–702.
[19] R. Löhner, Automatic unstructured grid generators, Finite Elem. Anal. Des. 25 (1997) 111–134.
[20] R. Löhner, E. Oñate, An advancing point grid generation technique, Comm. Numer. Methods Eng. 14 (1998) 1097–1108.
[21] R. Löhner, A parallel advancing front grid generation scheme, Int. J. Numer. Methods Eng. 51 (2001) 663–678.
[22] R. Löhner, C. Sacco, E. Oñate, S. Idelsohn, A finite point method for compressible flow, Int. J. Numer. Methods Eng. 53 (2002) 1765–1779.
[23] R. Löhner, E. Oñate, A general advancing front technique for filling space with arbitrary objects, Int. J. Numer. Methods Eng. 61 (2004) 1977–1991.
[24] R. Löhner, C. Yang, E. Oñate, On the simulation of flows with violent free surface motion, Comput. Methods Appl. Mech. Eng. 195 (2006) 5597–5620.
[25] R. Löhner, Applied CFD Techniques, Wiley, New York, 2008.
[26] R.A. Nay, S. Utku, An alternative for the finite element method, in: Variational Methods in Engineering 1, University of Southampton, 1972.
[27] E. Oñate, S. Idelsohn, O.C. Zienkiewicz, R.L. Taylor, A finite point method in computational mechanics applications to convective transport and fluid flow, Int. J. Numer. Methods Eng. 39 (1996) 3839–3866.
[28] E. Oñate, S. Idelsohn, O.C. Zienkiewicz, R.L. Taylor, C. Sacco, A stabilized finite point method for analysis of fluid mechanics problems, Comput. Methods Appl. Mech. Eng. 139 (1996) 315–346.

[29] J. Peraire, K. Morgan, J. Peiro, Unstructured finite element mesh generation and adaptive procedures for CFD, in: AGARD-CP-464, vol. 18, 1990.
[30] H. Sakaguchi, A. Murakami, Initial packing in discrete element modeling, in: B.K. Cook, R.P. Jensen (Eds.), Discrete Element Methods, ASCE, New York, 2002, pp. 104–106.
[31] M.S. Shepard, M.K. Georges, Automatic three-dimensional mesh generation by the finite octree technique, Int. J. Numer. Methods Eng. 32 (1991) 709–749.
[32] N.P. Weatherill, Delaunay triangulation in computational fluid dynamics, Comput. Math. Appl. 24 (5/6) (1992) 129–150.
[33] N. Weatherill, O. Hassan, M. Marchant, D. Marcum, Adaptive inviscid flow solutions for aerospace geometries on efficiently generated unstructured tetrahedral meshes, in: AIAA-93-3390-CP, 1993.
[34] M.A. Yerry, M.S. Shepard, Automatic three-dimensional mesh generation by the modified-octree technique, Int. J. Numer. Methods Eng. 20 (1984) 1965–1990.