INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2009; **78**:1618–1630
Published online 22 January 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.2548

An advancing front technique for filling space with arbitrary separated objects

Rainald Löhner[1,*,†] and Eugenio Oñate[2]

[1]*CFD Center, Department of Computational and Data Science, MS 6A2, College of Science, George Mason University, Fairfax, VA 22030-4444, U.S.A.*
[2]*CIMNE, Universidad Politécnica de Catalunya, Barcelona, Spain*

SUMMARY

An advancing front technique for filling space with arbitrary, separated objects has been developed. The input required consists of the specification of the desired object type, the mean object size, the distance between objects in space, as well as an initial triangulation of the surface. The objects are assumed to be described by a coarse mesh of tetrahedra. One face at a time is removed from the active front, and, if possible, surrounded by admissible new objects. This operation is repeated until no active faces are left. Two techniques to obtain maximum packing are discussed: closest object placement (during generation) and move/enlarge (after generation). Several examples are included that demonstrate the capabilities of the technique. Copyright © 2009 John Wiley & Sons, Ltd.

Received 21 August 2008; Revised 25 November 2008; Accepted 1 December 2008

KEY WORDS: grid generation; discrete element method; particle methods; granular media

1. INTRODUCTION

Many simulation techniques in computational mechanics require a space-filling cloud of arbitrary, separated objects. Most notable among these are the discrete element methods (DEMs) (see, e.g. [1–6]), which are extensively used to characterize and compute granular media, soil, concrete, and other discontinua. Owing to their simplicity, the objects most often considered are spheres, ellipsoids, or superquadrics of the form [6]

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

or an agglomeration of these [7]. However, in principle, they could be polyhedra or, for that matter, any other arbitrary shape. During the simulation, these objects will interact with each other via contact and friction forces. In many situations it is desirable to generate separated objects before

*Correspondence to: Rainald Löhner, CFD Center, Department of Computational and Data Science, MS 6A2, College of Science, George Mason University, Fairfax, VA 22030-4444, U.S.A.
†E-mail: rlohner@gmu.edu

the run is started. The task is therefore to fill a prescribed volume in an automatic way with arbitrary objects so that they are close but do not overlap.

Several techniques have been used to place objects in space. The so-called 'fill and expand' or 'popcorn' technique [4] starts by first 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 [7–11]. 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 close 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 [7]. In this way, one can simulate granules or stone.

In [7] an advancing front technique for filling space with arbitrary objects described by spheres was proposed. This scheme allowed for the direct generation of clouds of fairly arbitrary objects with the same degree of flexibility as advanced unstructured mesh generators [12–22]. The description of objects via a set of spheres, as used in [7], has the advantage of very fast crossing checks. However, an accurate and reliable penetration check may require a considerable number of spheres of different sizes for a single object. As an example, it was found that for hexahedra at least 27 spheres were required to achieve reliable object generation (see Figure 1).

This prompted the development of the technique described here. In the sequel, it is assumed that any object is defined by a coarse mesh of tetrahedra, allowing for a clear identification of external faces. The consideration of arbitrary bodies described via tetrahedra as compared with a collection of spheres not only opens the way to completely general shapes, but allows for strict penetration checks. Starting from the boundary, i.e. the initial 'front' of faces, new objects (and 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

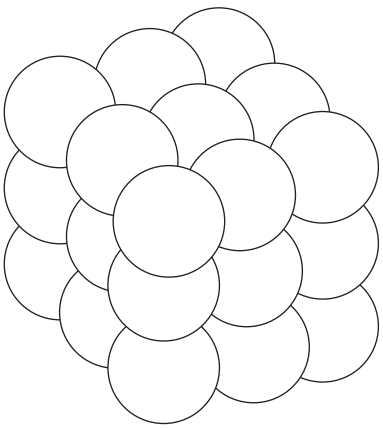


Figure 1. Hexahedron composed of 27 spheres.

DOI: 10.1002/nme

present scheme removes one face at a time, attempting to introduce as many objects as possible in its immediate neighbourhood.

Given that this is an advancing front technique, 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 [13, 14, 20, 21], clouds of points [8], or objects via spheres [7]. Nevertheless, for the sake of completeness we include in the following all aspects required by the proposed technique.

2. THE ALGORITHM

Assume as given:

- A specification of the desired object type(s) via tetrahedral grids.
- A specification of the desired object size in space (e.g. via a combination of background grids and sources [21]).
- 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 Figure 2, 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 faces in the front:
  - Remove from the front, the 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 Figure 3;
  - Find all existing faces in the neighbourhood of `ifout`;
  - `do`: For each one of the possible new neighbour objects `ionew`:
    - If there exists a face closer than a minimum distance `dminp` from any point of `ionew`, or if the faces of `ionew` objects are penetrating the existing front of faces: $\Rightarrow$ skip `ionew`;
    - If the line connecting `ifout` and the centre of `ionew` crosses existing faces: $\Rightarrow$ 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 points of `ionew` to the list of coordinates;
    - Introduce the faces of `ionew` to the list of active front faces;
    - Introduce the elements of `ionew` to the list of elements;

    `enddo`
- `endwhile`

The main search operations required are:

- Finding the active face with the smallest mean distance to neighbours;
- Finding the existing faces in the neighbourhood of `ifout`;

DOI: 10.1002/nme

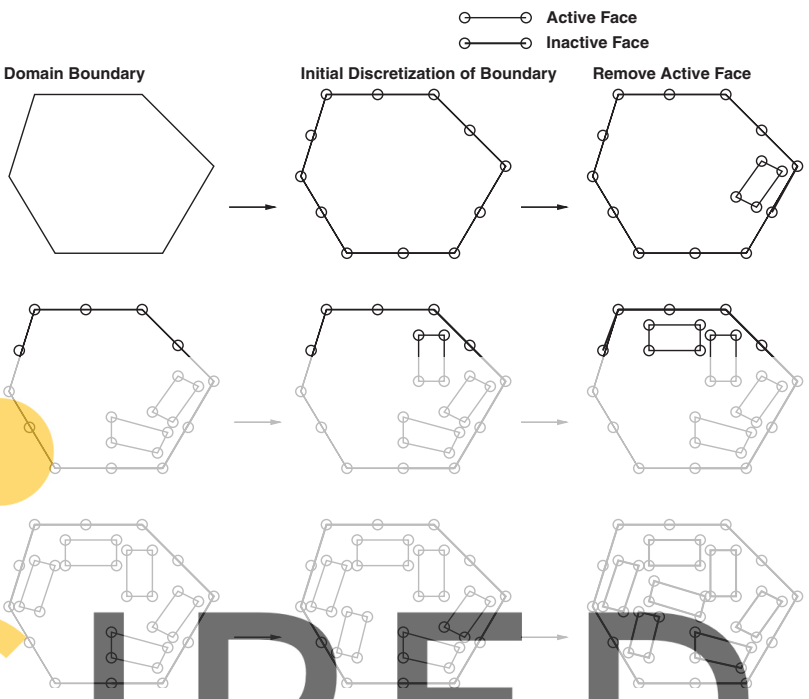


Figure 2. Advancing front space filling with trapezoids.

These search operations are performed efficiently using heap-lists, octrees, and linked lists, respectively (see [12–22] for more details).

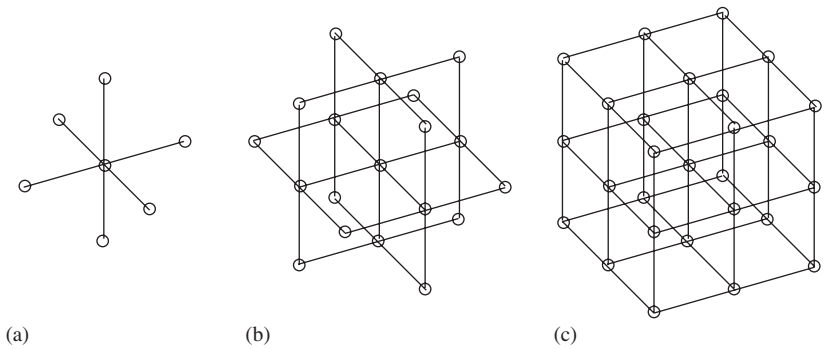
3. POINT STENCILS

A number of different stencils may be contemplated (see Figure 3). 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 [7, 8]. 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. Once a new object is introduced, all faces are tested against all current faces in order to

DOI: 10.1002/nme



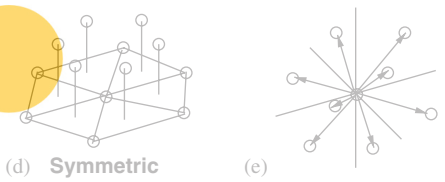


Figure 3. Point stencils: (a) cartesian [6]; (b) cartesian [18]; (c) cartesian [?]; (d) tetrahedral [17]; and (e) random.

Figure 4. Face crossing checks.

see if crossing occurs. Using octrees, linked-lists or other near-optimal data structures as well as local bounding boxes and filtering operations [23], 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 Figure 4).

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.

DOI: 10.1002/nme

5. BOUNDARY CONSISTENCY CHECKS

The face-crossing tests described before do not guarantee the generation of objects inside the desired domain. This can be seen from Figure 5, which shows a possible 2-D situation. While the faces of the new object `ifout` do not cross any existing faces, the new object `ionew` lies outside the domain. Such boundary crossings may be detected by constructing a 'side' that connects the centroid of the face being removed (`ifout`) to the centroid of the object being introduced (`ionew`). If this side crosses any of the close faces, the new object is rejected.

6. MAXIMUM COMPACTION TECHNIQUES

Many applications require a preset volume fraction occupied by the objects generated, and, if possible, a minimum number of contacts. The modelling of discontinua via DEMs represents a typical class of such applications. Experience indicates that the use of 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 Figure 6). 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 with 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 proximity/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,

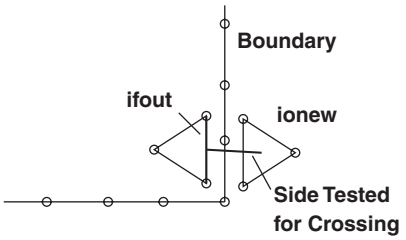


Figure 5. Boundary consistency checks.

DOI: 10.1002/nme

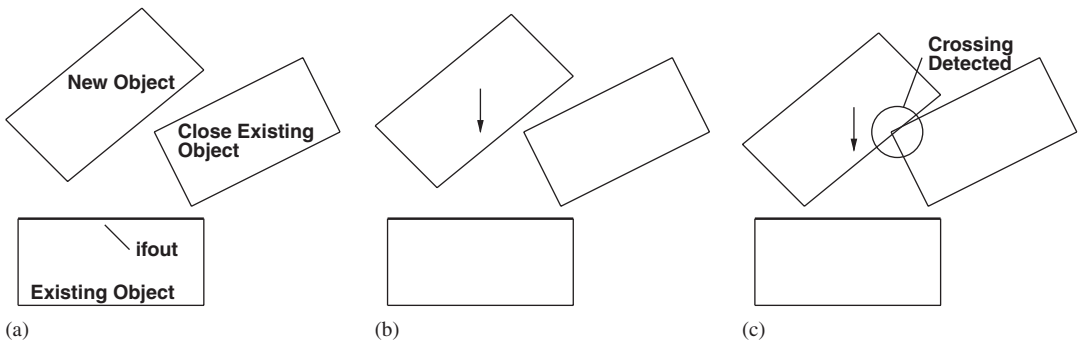


Figure 6. Closest object placement: (a) check ideal position; (b) move and re-check; and (c) move and re-check.

shown schematically in Figure 7, may be summarized as follows:

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

`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.

7. EXAMPLES AND TIMINGS

The proposed advancing front object generation algorithm has been used to generate object clouds for many applications: DEM/DPM simulations for discontinua, granular media, etc. Of these, we show a representative cross-section. All the object clouds were generated on a PC with Intel Centrino chip running at 2.0 GHz, 2 GByte Ram, Linux OS and Intel Compiler.

7.1. Cube

This first configuration considered is a unit cube, and is used to highlight the variety of elements that may be generated, the effectiveness of the optimal packing procedures described, and to obtain

DOI: 10.1002/nme

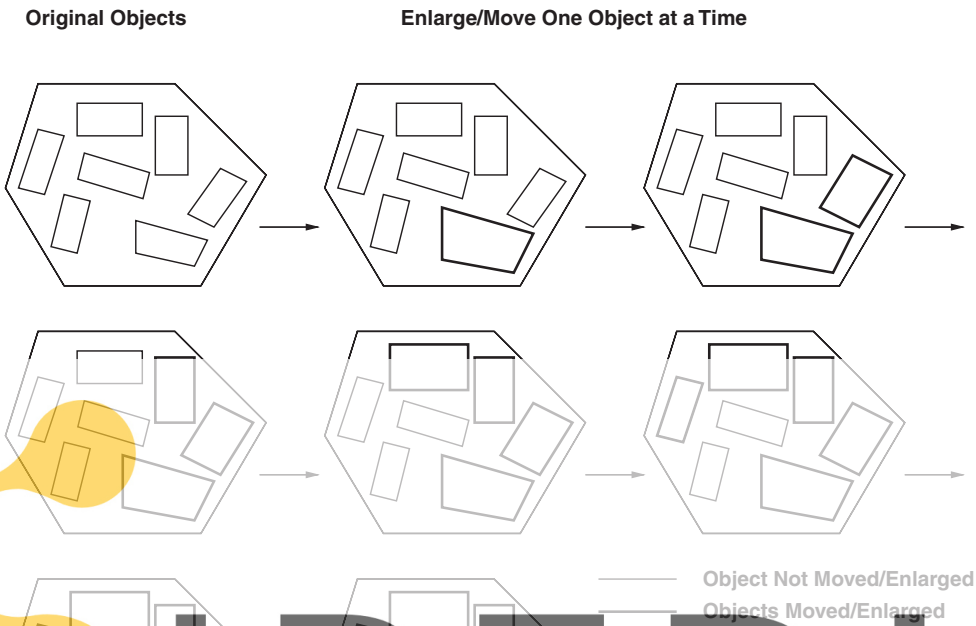


Figure 7. Movement and enlargement of objects.

some timing trends. The prescribed object size was $\delta = 0.112$, and a uniformly random variation of the orientation was allowed. Figures 8(a–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 I.

The generation times vary considerably depending on the element type and the amount of effort spent for optimum packing. For simple objects, such as 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 5 times more CPU time to post-process a cloud of objects than to generate it.

7.2. *Breakwater*

The second 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 Figure 9(a). 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

DOI: 10.1002/nme

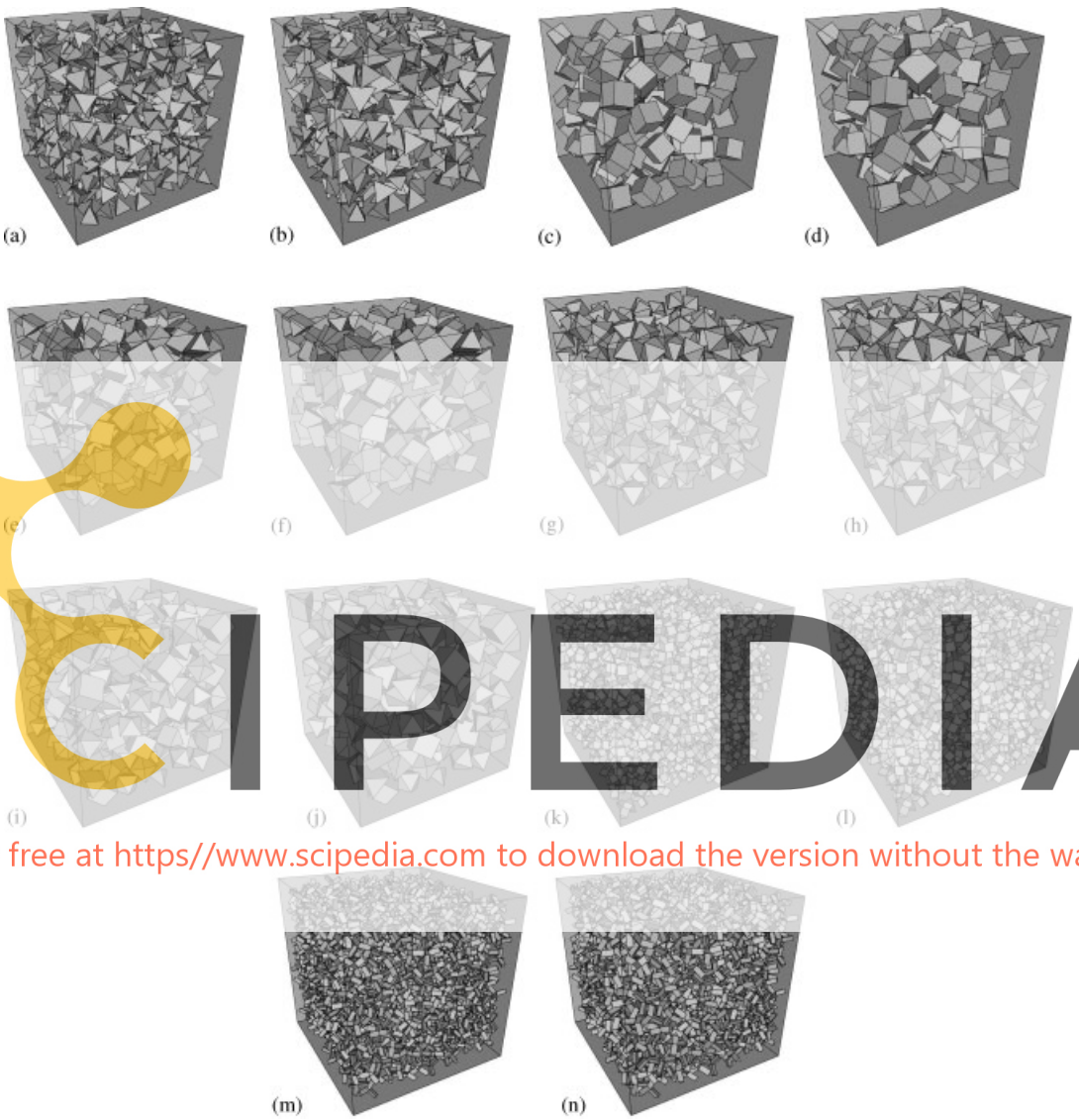


Figure 8. (a, b) Cube filled with tetrahedra; (c, d) cube filled with hexahedra; (e, f) cube filled with prisms; (g, h) cube filled with octahedra; (i, j) cube filled with a mix of objects; (k, l) cube filled with tetrapods (Type I); and (m, n) cube filled with tetrapods (Type II).

shown in Figure 9(b). The results of a typical simulation of the complete configuration using a volume of fluid method [24] are shown in Figures 9(c, d).

7.3. Filter

The third configuration shown is a filter. The objects created inside represent materials with different properties. Figures 10(a–d) show the object distributions obtained for hexahedra as well as a mix

DOI: 10.1002/nme

Table I. 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

Figure 9. (a, b) Breakwater: domain definition and resulting mesh and (c, d) breakwater: results for a typical simulation.

of tetrahedra, hexahedra, prisms, and octahedra. As before, the figures on the left show the meshes before the move and enlarge option is invoked, the figures on the right after.

7.4. Truckload of bricks

The fourth configuration shown is a truckload of bricks. The bricks have the standard U.S. size of $0.203 \times 0.102 \times 0.057$ (m). Figure 11 shows the distribution obtained (7893 bricks).

DOI: 10.1002/nme

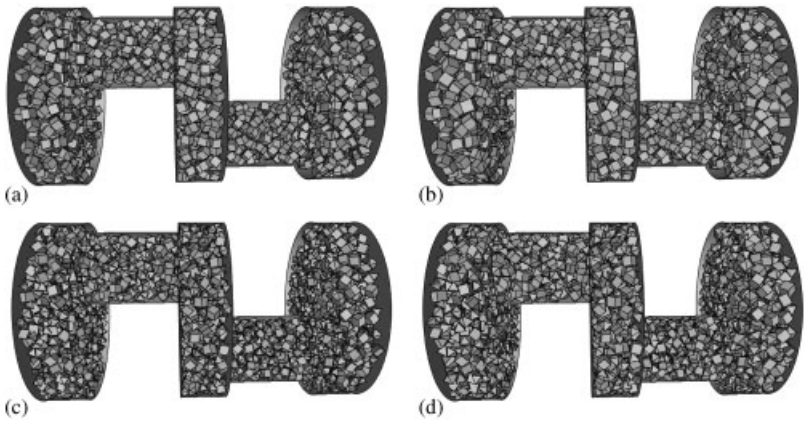


Figure 10. (a, b) Filter filled with hexahedral elements and (c, d) filter filled with a mix of elements.

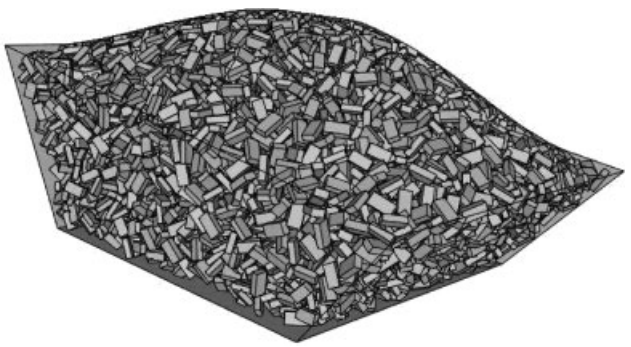


Figure 11. Truckload of bricks.

8. CONCLUSIONS AND OUTLOOK

A general advancing front technique to fill volumes with arbitrary objects has been developed. The input required consists of the specification of the desired object size and mean distance between objects in space and an initial triangulation of the surface. Each of the objects to be generated is given by a small tetrahedral mesh with an associated set of external faces. As with ordinary advancing front techniques for the generation of grids, one 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, i.e. no active faces are left.

Two techniques to obtain maximum packing are discussed: closest object placement (during generation) and move/enlarge (after generation). The last one nearly triples the required CPU time, but is still very competitive in comparison with direct contact calculations and deposition via gravity.

DOI: 10.1002/nme

Future work will consider:

- Alignment of objects: while all the examples shown have considered random object orientation, many applications require a spatial alignment. This may be due to crystal growth, deposition patterns, or external forces. It is relatively easy to modify the proposed procedure to deal with such scenarios.
- Library of objects: at present, an arbitrary object is read in from a small file that contains the information for the tetrahedral mesh of this object. For users, it would be highly beneficial to have access to a library of complex object shapes.
- Parallel object generation: although very competitive in terms of objects generated per second, for some large-scale applications even faster generation times may be required, prompting the development of a parallel object generator. The present procedure may be parallelized along the lines of unstructured grid generators with the advancing front technique [25], i.e. via domain decomposition.

ACKNOWLEDGEMENTS

A considerable part of this work was carried out while the first author was visiting the Centro Internacional de Métodos Numéricos en Ingeniería (CIMNE, www.cimne.com), Barcelona, Spain, in the Summer of 2008. The support for this visit is gratefully acknowledged.

REFERENCES

1. Cundall PA, Stack ODL. A discrete numerical model for granular assemblies. *Geotechnique* 1979; **29**(1):47–65.
2. Cundall PA. 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. *International Journal of Rock Mechanics and Mining Sciences* 1988; **25**:107–116.
3. Cleary PW. Discrete element modeling of industrial granular flow applications. *TASK. Quarterly-Scientific Bulletin* 1998; **2**:385–416.
4. Sakaguchi H, Murakami A. Initial packing in discrete element modeling. In *Discrete Element Methods*, Cook BK, Jensen RP (eds). ASCE: New York, 2002; 104–106.
5. Han K, Feng YT, Owen DRJ. Coupled lattice Boltzmann and discrete element modelling of fluid–particle interaction problems. *Computers and Structures* 2007; **85**(11–14):1080–1088.
6. Cleary PW, Sinnott MD, Morrison RD. DEM prediction of particle flows in grinding processes. *International Journal for Numerical Methods in Fluids* 2008; **58**(3):319–353.
7. Löhner R, Oñate E. A general advancing front technique for filling space with arbitrary objects. *International Journal for Numerical Methods in Engineering* 2004; **61**:1977–1991.
8. Löhner R, Oñate E. An advancing point grid generation technique. *Communications in Numerical Methods in Engineering* 1998; **14**:1097–1108.
9. Feng YT, Han K, Owen DRJ. An advancing front packing of polygons, ellipses and spheres. In *Discrete Element Methods*, Cook BK, Jensen RP (eds). ASCE: New York, 2002; 93–98.
10. Feng YT, Han K, Owen DRJ. Filling domains with disks: an advancing front approach. *International Journal for Numerical Methods in Engineering* 2003; **56**:699–713.
11. Han K, Feng YT, Owen DRJ. Sphere packing with a geometric based compression algorithm. *Powder Technology* 2005; **155**(1):3–41.
12. Yerry MA, Shepard MS. Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering* 1984; **20**:1965–1990.
13. Löhner R, Parikh P. Three-dimensional grid generation by the advancing front method. *International Journal for Numerical Methods in Fluids* 1988; **8**:1135–1149.
14. Peraire J, Morgan K, Peiro J. Unstructured finite element mesh generation and adaptive procedures for CFD. *AGARD-CP-464*, vol. 18, 1990.

DOI: 10.1002/nme

15. George PL, Hecht F, Saltel E. Fully automatic mesh generation for 3D domains of any shape. *Impact of Computing in Science and Engineering* 1990; **2**(3):187–218.
16. Shepard MS, Georges MK. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering* 1991; **32**:709–749.
17. George PL, Hecht F, Saltel E. Automatic mesh generator with specified boundary. *Computer Methods in Applied Mechanics and Engineering* 1991; **92**:269–288.
18. Weatherill NP. Delaunay triangulation in computational fluid dynamics. *Computer and Mathematics with Applications* 1992; **24**(5/6):129–150.
19. Weatherill N, Hassan O, Marchant M, Marcum D. Adaptive inviscid flow solutions for aerospace geometries on efficiently generated unstructured tetrahedral meshes. *AIAA-93-3390-CP*, 1993.
20. Löhner R. Extensions and improvements of the advancing front grid generation technique. *Communications in Numerical Methods in Engineering* 1996; **12**:683–702.
21. Löhner R. Automatic unstructured grid generators. *Finite Elements in Analysis and Design* 1997; **25**:111–134.
22. George PL, Borouchaki H. *Delaunay Triangulation and Meshing*. Editions Hermes: Paris, 1998.
23. Löhner R. *Applied CFD Techniques*. Wiley: New York, 2008.
24. Löhner R, Yang C, Oñate E. On the simulation of flows with violent free surface motion. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:5597–5620.
25. Löhner R. A parallel advancing front grid generation scheme. *International Journal for Numerical Methods in Engineering* 2001; **51**:663–678.

DOI: 10.1002/nme