Available online at www.sciencedirect.com

SCIENCE DIRECT®

Powder Technology 155 (2005) 33 – 41

POWDER TECHNOLOGY

www.elsevier.com/locate/powtec

Sphere packing with a geometric based compression algorithm

K. Han*, Y.T. Feng, D.R.J. Owen

Civil and Computational Engineering Centre, School of Engineering, University of Wales Swansea, Singleton Park, Swansea, SA2 8PP, UK

Received 17 February 2004; received in revised form 14 February 2005; accepted 28 April 2005
Available online 23 June 2005

Abstract

An efficient algorithm for the random packing of spheres can significantly save the cost of the preparation of an initial configuration often required in discrete element simulations. It is not trivial to generate such random packing at a large scale, particularly when spheres of various sizes and geometric domains of different shapes are present. Motivated by the idea of compression complemented by an efficient physical process to increase packing density, shaking, a new approach, termed compression algorithm, is proposed in this work to randomly fill any arbitrary polyhedral or cylindrical domains with spheres of various sizes. The algorithm features both simplicity and high efficiency. Tests show that it takes 181 s on a 1.4-GHz PC to complete the filling of a cylindrical domain with a total number of 26,787 spheres, achieving a packing density of 52.89%.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Sphere packing; Compression; Shaking; Discrete elements; Contact search

1. Introduction

In discrete element simulations, the pre-processing often involves the preparation of an initial random discrete object configuration that represents realistic situations. It is however not trivial to generate such a random packing at a large scale, particularly when discrete objects of various sizes and geometric domains of different shapes are present in many practical applications.

Random packing of disks/spheres is a research topic that has attracted considerable attention in different areas with different objectives over the past decades, and a number of packing approaches have been developed [1–5]. A fairly comprehensive review on existing packing algorithms of both 2D and 3D is conducted in [6]. A common feature of these algorithms, including the packing of 2D objects, is often the substantial CPU cost involved especially for large-scale problems. Therefore, the development of an effective packing procedure, in terms of computational costs, for a large number of disks/spheres becomes an important numerical and practical issue in the discrete element simulation of many industrial applications.

Such an effort is reported in our previous work [6], which introduces a novel algorithm based on the idea of advancing front techniques for finite element mesh generations for the random packing of 2D disks. This is a pure geometric packing algorithm without physical forces involved. Further development has considered a primary packing direction, which emulates, for instance, gravitational compression, during the advancing packing process. As only the fronts, each comprising two disks, need to be maintained, the approach has proven to be very efficient with a linear complexity. It takes only a few seconds to pack 1,000,000 disks on a normal desktop PC, and can achieve 80% packing density in general. The same procedure has also been successfully employed in the packing of ellipses and polygons [7].

In principle, this approach can also be extended to sphere packing. However, both the algorithm and implementation are much more complex. At the algorithmic level, the fronts consist of triangular facets, each obtained by joining the centres of three spheres. As the facet cannot be fully covered by the associated spheres, some spheres may penetrate through the facets without necessarily overlapping the existing spheres.

* Corresponding author.
E-mail addresses: k.han@swansea.ac.uk (K. Han), y.feng@swansea.ac.uk (Y.T. Feng), d.r.j.owen@swansea.ac.uk (D.R.J. Owen).

0032-5910/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.powtec.2005.04.055

These difficulties significantly increase the algorithmic complexity. At the implementation level, a far larger number of different circumstances need to be dealt with properly when different shaped geometric domains are considered.

It should be noted that in many applications, the actual packing configurations are produced under various physical forces, especially the action of gravity. Consequently, a compressive packing direction is introduced to represent the realistic situation as closely as possible.

Based upon the above observation, this work attempts to develop a new numerical approach, termed the *compression algorithm*, to generate a random packing of spheres of various sizes within a given geometric domain. As will be detailed below, the new algorithm is again geometric based and employs several existing numerical techniques, particularly the recent development of a very highly efficient contact search algorithm [8,10]. This algorithm is characterized by its simplicity, high efficiency and applicability not only to the random packing of disks and spheres, but to other shaped particles. The novelty of the algorithm lies not in the proposal of those individual techniques employed but in the unique way that they are combined together to achieve a simple and effective sphere packing algorithm. To the authors' best knowledge, no such an algorithm or such a combination of those techniques used has been previously reported for the packing of spheres.

In the remainder of the paper, the algorithmic description of the compression algorithm is presented. The key issues of the algorithm are discussed in detail. Filling spheres of various sizes within different geometric domains are illustrated via examples.

2. Algorithmic description

Consider the problem of randomly filling a domain with spheres of different sizes. The sphere radius r is randomly generated by a prescribed distributional function, and the domain Ω can be any arbitrary polyhedron or simple shape, such as a cube or cylinder. The compression direction V_g is specified.

Starting with an initial packing configuration, the algorithm proposed is a two-step procedure: (1) compress the initial packing; and (2) refill the space remaining, and if successful, compress the spheres by using the techniques in (1). Repeat the procedure until the domain is full.

There are different ways to create an initial packing within a geometric domain. The spheres can be randomly distributed or regularly positioned. All the spheres are checked against each other so that no overlap exists.

It is highlighted that no real force is involved in the packing procedure, therefore, it is a geometric-based approach.

2.1. Compression of a given packing

The basic idea of compressing a given packing can be stated as: if a sphere's immediate neighbours are known, it can be moved along the given compression direction to a new position in touch with the first neighbouring sphere(s) by assuming that the neighbours are temporarily static. In one iteration after all spheres have been repositioned, a more tightened packing should be achieved. Such iterations are repeated until no further compression is possible. Apparently, determining the dynamically changing neighbours of a sphere, which is accomplished by a search algorithm, is the key to the success of the algorithm.

There exist two slightly different options in searching for the neighbourhood information of a sphere. In the first option, the neighbour list of a sphere is built, then followed immediately by repositioning before proceeding to the next sphere. As a result, the neighbour lists of subsequent spheres may be affected by the new positions of the spheres already processed. In the second option, the neighbour lists of all the spheres are obtained altogether at the beginning of the iteration, and the spheres are then compressed sequentially. To ensure that the neighbour information obtained in the beginning remains valid for the whole iteration, a bounding box is assigned to each sphere, and the movement of any sphere must be confined within the bounding box. Note that such a requirement is also necessary for the first option to essentially define the 'close' neighbours of a sphere and to compute the maximum allowable moving distance. Consequently, the determination of the neighbour information of spheres is equivalent to the contact detection among spheres represented by axis aligned bounding boxes.

Since more efficient contact detection algorithms are available for handling the contact among all the objects than the contact of one object with others at a time, the second option is therefore chosen. As contact detection comprises a major proportion of the total CPU time involved in the complete packing procedure, it will be addressed separately in the following section.

2.1.1. Compression

Prior to any iteration, a bounding box with buffer zone is assigned to each sphere. Issues relevant to the bounding boxes will be discussed later.

Once the neighbour list is built, the next step is to evaluate the maximum moving distance of each sphere along the compression direction. For the convenience of illustration, a 2D problem is taken as an example as shown in Fig. 1. Assume that $V_g=(t_x,\ t_y)^T$ is the given compression direction with its normal $n_g=(n_x,\ n_y)^T$; C is the sphere to be compressed, and $T_i(i=1;\ \ldots,\ n_t$, where n_t is the number of neighbours of sphere C) is one of C's neighbour spheres. The maximum moving distance of C just touching T_i can be calculated as:

$$d_i = t - \sqrt{(r_c + r_t)^2 - s^2}$$