Modeling complex packing structures and their thermal properties

X. JIA, N. GOPINATHAN and R. A. WILLIAMS

Centre for Particle and Colloid Engineering, School of Process, Environmental and Materials Engineering, University of Leeds, Leeds LS2 9JT, UK

Received 11 April 2001; accepted 29 June 2001

Abstract—A packing algorithm for particles of complex shapes and a method to use the packing structure directly to predict its thermal properties are described. A novel aspect of the packing algorithm is that the particles, their movements and the packing space are all digitized. Using this approach, some major difficulties encountered by traditional packing algorithms can be avoided, for example, handling non-spherical and complex shapes, and detection of collision or overlap between the particles. The digital algorithm is capable of generating packing of particles of arbitrary shapes and sizes in a packing space or container of an arbitrary geometry. To predict effective thermal conductivity of the packing structure, an iterative finite difference method is used. The formulation is performed directly on the individual building blocks using fundamental laws of heat transfer. By performing the calculations at such a microscopic level, only the basic information about the material is needed and the influence of the microstructure on thermal properties can be assessed directly.

Keywords: Particle packing; digitization; arbitrary shape; thermal conductivity; ceramic processing.

NOMENCLATURE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>area ($m^2$)</td>
</tr>
<tr>
<td>$c$</td>
<td>specific heat ($J/kg/K$)</td>
</tr>
<tr>
<td>$c_0$</td>
<td>specific heat of element 0 ($J/kg/K$)</td>
</tr>
<tr>
<td>$i$</td>
<td>subscript used for element</td>
</tr>
<tr>
<td>$k$</td>
<td>thermal conductivity ($W/m/K$)</td>
</tr>
<tr>
<td>$k_{i0}$</td>
<td>thermal conductivity at interface between elements 0 and i ($W/m/K$)</td>
</tr>
<tr>
<td>$L$</td>
<td>width of elements or distance between centers of neighboring elements ($m$)</td>
</tr>
<tr>
<td>$N$</td>
<td>number of elements</td>
</tr>
<tr>
<td>$q$</td>
<td>heat flow ($W$)</td>
</tr>
<tr>
<td>$q_0$</td>
<td>energy volume density ($W/m^3$)</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>time step ($s$)</td>
</tr>
<tr>
<td>$\Delta T_0$</td>
<td>temperature difference ($K$)</td>
</tr>
</tbody>
</table>
\begin{align*}
T & \quad \text{temperature (K)} \\
T_i & \quad \text{temperature of element } i \ (i = 0, 1, 2, 3, 4) \ (\text{K}) \\
x & \quad \text{distance (m)}
\end{align*}

Greek
\begin{align*}
\rho & \quad \text{mass density (kg/m}^3) \\
\rho_0 & \quad \text{mass density of element 0 (kg/m}^3)
\end{align*}

1. INTRODUCTION

Computer generated random packing of particles is obtained either by using a so-called packing algorithm or by adopting a dynamic simulation model. Packing algorithms [1–9] usually neglect the physical process of packing in order to provide a computationally efficient method to generate the structure. Particle dynamics models, on the other hand, are designed to simulate the process of a particulate system [10, 11] and, if used to simulate packing process [12, 13], produce the packing structure as a result. Traditionally, there are three ways by which a shape may be represented. A simple shape, such as a circle or a sphere, is usually represented analytically, by its central coordinates and radius. Detection of collision or overlap between two of these shapes is easy — one only needs to compare the distance between their centers with the sum of their radii to decide whether they overlap or not. A more complex shape may be represented as a composite of some simple building blocks, notably circles or spheres. It can also be represented by a collection of disjoint polygons or, more often, a polygonal mesh. Collision/overlap detection for these is generally more tedious and time consuming, since it may be necessary to consider every pair of constituent primitives (e.g. spheres in sphere composites, line segments and polygons in mesh representations) from the two shapes. With the latter two representations, the coding effort and computational cost of shape representation and collision/overlap detection increase dramatically with the complexity of the shapes. This is one of the main reasons why simulations of both packing structures and dynamic processes rarely use non-spherical particles.

Recently, a new packing algorithm for particles of arbitrary shapes has been developed [14, 15]. In contrast to traditional methods, the new packing algorithm represents particle shapes digitally. Since a particle is now simply a collection of basic building blocks [e.g. pixels in two-dimensions (2D) and voxels in 3D] regardless of the complexity of its shape, the algorithm is able to deal with complex shapes as easily and efficiently as for simple shapes. It will be shown that digitization of particle shapes, their movements and the packing space helps to avoid many difficulties encountered by traditional methods in dealing with arbitrary shapes.

With the digital packing algorithm, we are no longer restricted to using sphere packing to represent structure — the structure of a soil, sediment, filter cake and
granular filter media may all be modeled using the real shapes of the constituent particulate solids. Having a realistic and detailed structure model paves the way for some structure-related properties to be studied and predicted at a microscopic level, based on first principles. There are some numerical techniques particularly suited for the complex digital structures involving large numbers of elements. We demonstrate one of these techniques in predicting thermal conductivity of the packing structure. Other techniques will be discussed briefly.

2. PARTICLE PACKING

In a digital computer image, everything is pixel based. An object, no matter how complex in shape, is represented by pixels. The same applies to a volume rendering in 3D. This ‘pixelation’ (2D) or ‘voxelation’ (3D) of objects, and of the space, is the basis of the new packing algorithm. As shown in Fig. 1, an arbitrary shape is now simply a coherent collection of pixels, and the space is mapped onto a grid. In a 2D implementation — a software program called DigiPac — a shape may be digitized in two ways. Simple and analytical shapes, for which library support is provided by software development tools (e.g. Visual C++), are created and digitized directly in the computer memory. For example, a polygon of specified number of sides and dimensions is drawn on a background canvas in the computer memory, and pixels belonging to the drawing are collected and subsequently used to represent the shape. Shapes which can be generated in this manner include arcs/chords, ellipses/circles, line segments, sine waves, polygons, polylines/curves and text/letters with user-definable fonts. More complex shapes are imported as bitmap images (Fig. 1) with each image containing a single shape or object. When importing, any pixel that has a color different from the defined background color (usually white) is deemed to belong to the shape and collected. The bitmap itself may be obtained by scanning the object directly or its photo image, or created using some drawing packages (e.g. Paint or AutoCAD).

Any size distribution can be used, although only the most frequently used, such as linear, uniform, bimodal, normal and log-normal distributions, are built in the

Figure 1. The real particle shape extracted from a photo image is imported as a bitmap and digitized. A small portion of the particle is magnified 8 times to show the pixelation, which is how the particle is represented internally by the digital packing algorithm.
simulation program. Particles of the same shape and size are generated by copying, which is faster than the background drawing and pixel collecting procedure. Although particles of the same shape but different sizes can obtained by duplicating and scaling the original, for which there is again good library support in Visual C++, it is not implemented as such because simple scaling tends to exaggerate aliasing (or ruggedness) in the digitized shapes. Antialiasing (or smoothing technique for grey-scale or color images) is not adopted because internally the simulation uses binary representation of shapes to avoid ambiguity and for the sake of speed, even though it can retain and use the original color or texture of the imported shapes for the purpose of display (Fig. 2).

The resolution depends on how accurate the shape needs to be represented which, in turn, is often dictated by the requirements of a particular application. The drawing functions provided in Visual C++ generally work on a canvas of $2^{15} \times 2^{15}$ — more than enough for most practical purposes. If an even larger area or higher resolution is required, it is possible to write bespoke drawing functions, many of which are well documented [16].

Since the packing space or container is digitized and represented in the same way as the particles, using a container with a more complex geometry (Fig. 3) presents no additional difficulties. Moreover, just as a particle may be added at any time, the container may be introduced or changed at any time, during packing. This helps to create the scenario shown in Fig. 4, where particles were poured into a container and later released to create an avalanche.

In the simulation, the particles are allowed to move randomly, one grid at a time, on a square lattice. In 2D, there are eight possible directions — four orthogonal and four diagonal — to choose from, all with equal probability. In 3D, the number is 26, six orthogonal and 20 diagonal. It is convenient to treat diagonal moves as composed of two orthogonal moves. For example, a move in the upper-left direction can be thought of as an upward move followed by a left move. In order to encourage particles to settle down, the upward component of a move is only accepted with a so-called rebounding probability. The result is a directional and diffusive motion.

![Figure 2.](image)

(a) (b)

Figure 2. Although the original texture or colors/shades can be retained for display, as in (a), internally the shapes are represented and manipulated as if they are all binary images, as in (b).
Figure 3. (a) Packing space with a complex boundary geometry. (b) An early stage of packing. (c) A later stage of packing. The simulation was performed on a $600 \times 400$ grid, with shapes like circles, ellipses, lines, polygons, arcs and chords; no rotation; 50% rebounding probability, adding 10 particles every 200 time steps.
Figure 4. Initially, particles are packed into a container. (a) A gap is then created by importing a second container of the same geometry except the opening at the bottom part of the right wall. (b) Particles begin to slide down the slope and (c) create a heap at the bottom whilst the container is being emptied.
for the particles, rather like a random walks based sedimentation model described elsewhere [17]. This diffusive movement helps the particles to effectively penetrate and explore every available packing space.

Since particles reside and move in a grid, collision and overlap detection is now a simple matter of detecting whether two objects occupy the same site(s) at a given time, rather than having to compute and test intersections between objects, which is usually the most computationally expensive part of particle simulations. Since a particle moves only one grid at a time, the above overlap detection procedure ensures that the particle will not jump over, or enter the hollow part, of another particle during packing. It also allows solid particles to be represented by their outlines, which would substantially speed up the packing process since much fewer pixels per particle need to be processed at each trial move. Of course, it is possible, perhaps even necessary if the effects of the actual particle interactions are considered, to move a particle more than one grid at a time. In this case, care must be taken to prevent particles jumping over or into other particles.

During the packing, particles are allowed to rotate. In the present implementation, rotation is either completely random or by a small random amount from its previous orientation. Rotation can be turned on or off at any time. Rotation usually results in denser packing structure since it increases the chance of a better fit (Fig. 5). However, for some regular shapes and/or in some special cases, rotation may actually leads to a less dense packing structure, because it can destroy an otherwise ordered structure or better fit (Fig. 6). In the simulation, rotation is carried out by a succession of three shears [18], because the normal rotation transformation, whilst much faster, would produce holes in the rotated digital structure (Fig. 7).

Although the packing algorithm does not explicitly involve physical forces, some effects of physical interactions can still be simulated. Since particles are allowed to move side ways as well as up-down all the time — even after they form part of the packing — the effects of high frequency, small amplitude vibrations are simulated. The result is usually a denser packing structure (Fig. 5). For particles of different sizes/shapes trickling down a point source, this also leads to size segregation (Fig. 8). The vertical vibration is controllable, by the rebounding probability. A value of 0 means no vertical vibration, and a value of 1 means equal opportunity for the particles to move up and down, thus no chance to settle down and form a packing. Typically, a value between 0.2 and 0.5 is used.

Particles may be introduced in any way. Two have been implemented. One is to allow particles to ‘trickle’ down, one at a time, from a specified point above the packing, at a defined rate. The other is to spread the particles across a specified width above the packing and let them ‘rain’ down all at once, at a defined interval. The former results in a heap and promotes size segregation (Fig. 8). The latter fills the container more evenly and leaves little room for stratified structure to developed.

The rate of particle addition affects the packing density, since it determines how long a particle has to explore the packing before being locked-in by new additions.
Figure 5. Packing of initially randomly orientated short fibers of 3 : 10 aspect ratio. (a) No rotation but with 50% rebounding probability. Packing density is 0.71. (b) With rotation and 50% rebounding probability. Fibers dropped from a point source. Packing density is 0.81. (c) No rotation and no rebounding. Packing density is 0.64.
Figure 6. Packing of initially identically orientated rectangles of 3:2 aspect ratio. (a) No rotation. Packing density is 0.92. (b) With rotation. Packing density is 0.87.

Figure 7. Rotation of a polygon shown in (a) by (b) direct transformation and (c) three shears. The former produces patterned arrays of holes due to rounding errors, whereas the latter does not but is significantly slower.
Figure 8. Examples of size segregation using (a) the rain model, size ratio 100 : 5 and (b) point source, size ratio 40 : 20 : 10, both in a 400 × 400 container.

Generally, slow admission leads to denser packing structure, and high rate to a less dense structure.

Laterally, the boundaries can be either solid walls, or periodical, or virtual. With solid walls, the packing is usually less dense near the wall than in the central part. This is the so-called wall effect. With periodical boundary conditions, if a particle moves out of one side, its image comes in from the opposite side at the same time. This avoids the wall effect and allows bulk properties to be predicted by a small sample. The width of the simulation box should be much larger than that of the largest particle (e.g. by a factor of 10). With virtual walls, particles can penetrate the walls and once go out will disappear forever. This is rather wasteful and thus not implemented in the simulation, but it can be useful in certain circumstances.

A range of parameters and options are used to control the packing process. In addition to the ones already mentioned — the rate of particle addition, rebounding probability, boundary conditions and rotation — there are options to allow immobilization and removal of selected particles. This helps to generate the packing in a controlled manner to achieve, for example, higher packing density by removing certain particles that are ‘in the way’. It may also be used as a way to achieve faster execution — by immobilizing thus taken out of consideration certain particles that have not been successfully moved for a given number of time steps and therefore are not expected to make successful trial moves in the future.
Although demonstrated in 2D, the algorithm is equally applicable in 3D. Automated procedures are currently being developed to use 3D laser scanner and X-ray tomography systems to obtain digitized objects in 3D. X-ray tomography can also be used to validate the packing structure of small particles generated by the digital packing algorithm [19].

3. THERMAL CONDUCTIVITY

In terms of applications, the usefulness of the packing algorithm is obvious for cases where the packing structure and packing density of real particles are needed. The model structure can also be directly input into some numerical methods for studying and predicting properties of materials/systems the model structure represents. For example, effective thermal or electrical conductivity of the packing structure may be computed using an iterative finite difference scheme, gas permeability by a random-walk based technique [20] and fluid flow distribution by lattice Boltzmann method [21]. Computer models of compaction, sintering and melting may also be built directly on the digital packing structure. Indirectly, the digital structure may serve as a starting point to obtain a more suitable mesh grid for some other established numerical methods (e.g. CFD). It is recognized that the uniform grid used in the packing algorithm may not be the most appropriate for CFD and is orders of magnitude smaller than the usual length scale used in a Finite Element Method mesh. However, it is possible to merge, split and/or convert certain elements to create a more suitable mesh grid for use. It is also possible to convert the digital shapes back to the traditional vector forms (e.g. polygonal mesh) after the packing, so that some existing models based on the traditional shape description can make use of the packed shapes. While many of the above methods are being developed and implemented by the authors, computation of temperature field in a packing structure is described below, as an illustration.

Both the particles and the packing structure are composed of regular-shaped building blocks — squares in 2D and cubes in 3D — which make them ideal elements for some numerical techniques to be formulated. Take the 2D digital structure in Fig. 9 as an example and consider only heat conduction for the moment. Each element is assigned its own basic properties, which may or may not be the same as those of its neighbors. Each element is surrounded by four contact neighbors and only interacts with the four neighbors. Applying Fourier’s law of heat conduction:

\[ q = -kA \frac{\partial T}{\partial x}, \]  

(1)
on each interface between element 0 and its neighbors (1–4) and the energy balance, one has:

\[ \dot{q}_0AL - \sum_{i=1}^{4} q_{i0} = \rho_0 c_0 AL \frac{\Delta T_0}{\Delta t}, \]  

(2)
\[ q_{i0} = k_{i0}(T_i - T_0) \frac{A}{L}, \]

where \( q_0 \) is the energy generated per unit volume (in W/m\(^3\)), \( A \) is the area of the interface (in m\(^2\)), \( L \) is the width of the element and also the distance between the centers of two adjacent elements (in m), \( T_i \) (\( i = 0, 1, 2, 3, 4 \)) is the temperature of element \( i \) (in K), \( \rho_0 \) is the mass density of element 0 (in kg/m\(^3\)), \( c_0 \) is the specific heat of element 0 (in J/kg/K), \( \Delta T_0 \) is the temperature difference (in K) of element 0 during time step \( \Delta t \) (in s), and \( k_{i0} \) (\( i = 1, 2, 3, 4 \)) is the equivalent thermal conductivity for the interface between element \( i \) and element 0 (in W/m/K). If the elements are identical and belong to the same particle, \( k_{i0} \) becomes the thermal conductivity of the material in its normal sense. If not, \( k_{i0} \) defines the rate of heat conduction between the elements and is likely to be a function of the respective thermal conductivity of the elements, whether the interface is between particles of the same type or material and the contact characteristics such as surface roughness.

Given the materials of the particles and contact characteristics, \( k_{i0} \) are in principle measurable or computable independently, and thus part of the input to the numerical technique aimed at studying the relationship between the packing structure and the effective thermal conductivity of the packing. In our implementation, they are user-definable parameters.

For internal elements of the same particle of the same material, the four coefficients (\( k_{i0}, i = 1, 2, 3, 4 \)) are identical and equal to the thermal conductivity of the material. If the particle is not a heat source/sink (i.e. \( q_0 = 0 \)) and steady-state is assumed (i.e. \( \Delta T_0/\Delta t = 0 \)), (2) reduces to the so-called five-point formula in finite difference methods which, when expressed in the form for Gauss–Seidel iteration, is written as:

\[ T_0 = (T_1 + T_2 + T_3 + T_4)/4. \]

Even in the transient or non-steady state, (3) may still be used under certain conditions including that \( \rho_0, c_0 \) and \( k_{i0} \) are identical and constant for all elements, and the time step satisfies the following condition [22]:

\[ \Delta t = \frac{\rho c L^2}{4k}. \]

In the packing structure, \( L \) is fixed. Equation (4) in effect provides the link between simulation cycles and the real time.

If element 0 has neighbor(s) representing vacuum (isolating) or fluid (convection), \( q_{i0} \) is set to zero (for vacuum) or replaced with Newton’s law of cooling (for fluid). For exposed element 0, heat transfer by radiation may also be included using, for example, the Stefan–Boltzmann law modified to take account of gray bodies. It is necessary to identify, for element 0, all the elements facing it and therefore contributing radiation heat flux. The identification procedure, however, only needs to be performed once per simulation, so the computational cost, whilst high, is bearable.
Figure 9. (a) An example of layered packing of various shapes (ellipses, line segments, polygons, arcs, letters, symbols and circles). (b) Packing density profile along the height of the packing. Different shades indicate contributions from different components or shapes. (c) Formulation of heat conduction based directly on the elements forming the digital microstructure.
Given that the digital packing structure usually involves a very large number of building blocks (e.g. $10^5$ in 2D and $10^7$ in 3D), it is not practical to use direct matrix inversion — a process whose computational cost is, roughly, proportional to $N^3$ where $N$ is the number of elements — to solve the problem. In this case, Gauss–Seidel method may be used. It is an iterative method, whose computational cost (per time step) is approximately linear to $N$ and is highly parallelizable. The methods works as follows: An initial set of values for the $T_i$ ($i = 0, 1, 2, \ldots, N$) is assumed. New values of the $T_i$ are calculated according to (2) or (3), always using the most recent values of the $T_i$. The process is repeated until the specified time steps have elapsed (for transient cases) or until successive calculations differ by a specified small amount (for steady states). Using this method, steady-state is treated as a special case of transient solution. Other, more efficient, iterative methods [23] may also be used, at the cost of increased complexity in formulation and implementation.

Figure 10 show the evolution of the temperature field at selected time steps in a packing structure shown in Fig. 9, where, for simplicity, all elements were assumed identical, $k_{i0} = 0.2$. The text and letters are set as heat sources, maintaining a constant temperature of 100°C, whereas the initial temperature for the rest is 0°C. The packing structure is represented by 141 908 elements, 41 000 time steps took 4000 CPU s (or about 0.1 CPU s per time step) on a PC with a Dell XPS T600 with a 600 MHz Pentium III CPU. The memory requirement for the heat transfer simulation was 6.4 MB of RAM. With a faster computer or more patience, larger-scale 2D or even full 3D simulations are feasible.

4. CONCLUSIONS

A digital packing algorithm for particles of arbitrary shapes and sizes in a container of arbitrary geometry has been described and demonstrated in 2D. For packing of arbitrary shapes, the digital approach has several advantages over the traditional methods. For example, it is not limited to mathematically or computationally manageable shapes. The computing resources required to represent a particle depends on the area (in 2D) or volume (in 3D) of the particle shape, rather than on its complexity. Collision and overlap detection is a simple matter of checking whether two objects occupy the same site(s) at a given time step. Digitization also means that the computations involved are mostly integer operations, thus requiring no more than an ordinary PC. The algorithm should be equally applicable in 3D, although digitizing 3D objects is more involved than in 2D. Automated procedures are currently being developed to use 3D laser scanners and X-ray tomography systems to obtain digitized objects in 3D.

As with any other algorithm, the digital approach has its own drawbacks. For example, it is just as, if not more, memory intensive as some traditional shape representations, although it is possible to reduce memory requirement substantially at the cost of execution speed [1]. For circles/spheres, the digital algorithm
Figure 10. Simulated evolution of temperature field and temperature profiles along the indicated section at (a) 1500, (b) 15 000 and (c) 41 000 time steps.
may be slower than some ballistic algorithms. Quantitative prediction of packing characteristics such as packing density is sensitive to the resolution used, and the sensitivity is usually non-linear and shape dependent. Despite these, the simplicity, ease of implementation and the speed of the digital algorithm make it an attractive alternative to the traditional packing algorithms when it comes to packing particles of arbitrary shapes. This has a wide range of applications to bulk solids handling (conveying, haulage), road materials (cement, tar), and at small length scales in the composition of thin films and composites (polymers, coatings) and colloidal aggregates.

At present, the digital packing algorithm can only deal with rigid solids. The feasibility of extending the algorithm to handle deformable or soft solids is currently being investigated. This enables the packing program to simulate structural changes due to melting, sintering or high pressure in, for example, rapid prototyping and powder metallurgy. The feasibility of building physical interactions into the packing algorithm is also under investigation. Direct applications of the packing algorithm and use of the packing structure for property prediction are also being actively pursued.

Acknowledgements

The authors acknowledge financial support of the EPSRC under grants GR/N13357 and GR/M91563. Useful discussions with colleagues such as Drs M. Wang and R. M. West are acknowledged. N. G. is also grateful for financial support of the Keyworth Institute, University of Leeds.

REFERENCES

15. X. Jia and R. A. Williams, Object interaction simulation, UK Patent Application P88839GB.
16. For example, the *Graphics Gems* Series, published by Academic Press, from 1994 onwards.