SCIENCE CHINA
Technological Sciences

A compact 3D block cutting and contact searching algorithm

XingChao LIN, Xu LI, XiaoGang WANG and YuJie WANG

Citation: SCIENCE CHINA Technological Sciences **62**, 1438 (2019); doi: 10.1007/s11431-018-9318-2

View online: http://engine.scichina.com/doi/10.1007/s11431-018-9318-2

View Table of Contents: http://engine.scichina.com/publisher/scp/journal/SCTS/62/8

Published by the Science China Press

SCIENCE CHINA
Technological Sciences

•Article•

August 2019 Vol.62 No.8: 1438–1454
https://doi.org/10.1007/s11431-018-9318-2

A compact 3D block cutting and contact searching algorithm

LIN XingChao[1,3], LI Xu[2*], WANG XiaoGang[1,3] & WANG YuJie[1,3]

[1] *State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Beijing 100038, China;*
[2] *Key Laboratory of Urban Underground Engineering of Ministry of Education, Beijing Jiaotong University, Beijing 100044, China;*
[3] *Department of Geotechnical Engineering, China Institute of Water Resources and Hydropower Research, Beijing 100048, China*

Received January 3, 2018; accepted June 28, 2018; published online April 10, 2019

The geometry relation and the contact point-pairs detection between two three dimensional (3D) objects with arbitrary shapes are essential problems involved in discontinuous computation and computational geometry. This paper reported a geometry relation judgment and contact searching algorithm based on Contact Theory. A contact cover search algorithm is proposed to find all the possible contact cover between two blocks. Two blocks can come to contact only on these covers. Each contact cover can define a possible contact point-pair between two blocks. Data structure and flow chart are provided, as well as some examples in details. Contact problems involving concave blocks or parallel planes are considered to be very difficult in past and are solved by this algorithm. The proposed algorithm is compacted and applicable to the discontinuous computation, such as robotic control, rock mass stability, dam stability etc. A 3D cutting and block searching algorithm is also proposed in this study and used to search the outer boundary of the 3D entrance block when 3D concave blocks are encountered. The 3D cutting and block searching algorithm can be also used to form the block system for jointed rock.

contact, discontinuous computation, block cutting, computational geometry, jointed rock, cover

Citation: Lin X C, Li X, Wang X G, et al. A compact 3D block cutting and contact searching algorithm. Sci China Tech Sci, 2019, 62: 1438–1454, https://doi.org/10.1007/s11431-018-9318-2

1 Introduction

In technological and natural process, contact between media often play an important role, especially moving interfaces are encountered. Contact is essential in the problems, such as assembling the mechanical parts, analyzing the stability of rock mass, analyzing the contact force between foundation and soil, attacking a target with missiles, or holding an object with a mechanical arm etc. In these problems, one of the key tasks is to calculate when and where two objects contact.

The calculation of when and where two objects contact can be named as contact computation [1]. Because contact computation is essential in many engineering problems, including the mechanical analysis of rock, stability analysis of dam, the stress analysis of mechanical control, and the mechanical control of robot, it has gained many concerns of scientists and engineers.

Neighbor searching is a rough preparation for contact computation. A blocky system contains more than one discontinues object. But only two blocks which are close enough have the possibility of contact and these two blocks are defined as neighboring blocks. Before contact computation, algorithms for finding neighbor blocks can improve the efficiency for detecting contacts. There are mainly two kinds of neighbor searching algorithms, such as space-based grids and object-based cells. Space-based grids [2,3] are generated by dividing the system space into a set of spatial grids with some topological structure. Thus, blocks or elements involve the same grid or adjacent grids are considered as the neighboring blocks. Object-based cell [3–10] is a cover system based on a block. A rough relation between two blocks can be found by computing the geometrical and to-

*Corresponding author (email: ceXuLi2012@163.com)

pological relation of their cells.

The contact problem on two neighbored blocks has been continuously studied in the past decades. Some special contact problems have been solved [2,3,11–22]. However, the contact among two 3D blocks with arbitrary shapes still lacks of a thorough theory and has blocked the numerical analysis of problem involving 3D discontinuity until Shi [1] published the "contact theory".

Contact theory [1] provides a solid mathematical base of contact theory. It is still a deep gap between the theory and its application in contact computation. A numerical algorithm that can be used to calculate when and where two objects contact is still in desire in various numerical methods, such as finite element method (FEM [23]), discontinuous deformation analysis (DDA [24–28]), discrete element method (DEM [29–31]), numerical manifold method (NMM [32–35]), and boundary element method etc.

Two algorithms are proposed in this study. First, a contact cover searching algorithm is proposed to find the entire possible contact interfaces between two blocks with arbitrary shapes. Second, a compact block cutting and algorithm is proposed to form the entrance block which is the key concept in contact theory.

In this paper, contact theory will be first introduced briefly, as quoting the key formula to be used in the algorithm. Then the algebraic presentation of a 3D object will be introduced. Further, the contact cover searching algorithm is introduced followed by the entrance block generation algorithm. At last, several examples are reported.

2 A brief introduction of contact theory

Ref. [1] proposed a general contact theory for 2D and 3D discontinuous computation, in which the concept of an entrance block is introduced and the complex contact problem is simplified as the geometric relation between a reference point and the entrance block.

Figure 1 illustrates the simplification process of the contact problem between two 2D blocks, as from the original geometric relation between block A and block B (Figure 1(a)), to the generation of entrance block (Figure 1(b)), and further to the boundary of entrance block and reference point $\mathbf{a}_0$ (Figure 1(c)). In Figure 1(c), the geometry relation between a polygon $\partial E(A,B)$ and a point $\mathbf{a}_0$ is very simple and computable. However, the geometric relation between block A and block B in Figure 1(a) is complex and incalculable. Contact theory [1] provides a solid basis for discontinuous computation and offers an algebraic computation method for judging the geometric relation among complex 3D block systems. In this section, the basic concept and formula to be used in the programming will be introduced here.

2.1 Contact searching with the help of the entrance block

Ref. [1] has proved that the complicated contact conditions between two blocks A and B, i.e. whether they overlap, separate or contact, can be simplified as the relations that the reference point $\mathbf{a}_0$ is located inner, outer, or lies on the boundary of entrance block $\partial E(A, B)$, respectively.

Obviously, for a 2D block, its boundary is composed by a series of lines (referring to Figure 1) and for a 3D block, its boundary is composed by a series of polygons. If the boundary of the entrance block, $\partial E(A, B)$ is found, the relation between blocks A and B can be judged by

$$\begin{cases} d < 0 & \Rightarrow \quad \text{overlap happen,} \\ d = 0 & \Rightarrow \quad \text{contact happen,} \\ d > 0 & \Rightarrow \quad \text{seperation condition,} \end{cases} \tag{1}$$

where d is the minimum distance between a boundary polygon (or line) of $\partial E(A, B)$ and the reference point $\mathbf{a}_0$ on block A. It can be calculated as

$$d = \min_{k=1,2,\ldots m} \left(\operatorname{sign}\left(\mathbf{n} \cdot (\mathbf{a}_0 - \mathbf{a}_\mathrm{p}) \right) \right) \left| \mathbf{a}_0 - \mathbf{a}_\mathrm{p} \right|, \tag{2}$$

where P_k is the k boundary polygon (or line) of $\partial E(A, B)$; m is the total number of the boundary polygons (or lines); $\mathbf{n}_k$ is the outer normal vector of P_k; $\mathbf{a}_0$ is a reference point fixed on block A; $\mathbf{a}_\mathrm{p}$ is the vertical projection of $\mathbf{a}_0$ on P_k. As illustrated in Figure 1(d), the distance between $\mathbf{a}_0$ and $\mathbf{a}_\mathrm{p}$ is equal to the distance between $\mathbf{a}_1$ and $\mathbf{b}_\mathrm{p}$, which is the closest distance between blocks A and B.

If d is equal to 0, $\mathbf{a}_0$ and $\mathbf{a}_\mathrm{p}$ coincide, so do $\mathbf{a}_1$ and $\mathbf{b}_\mathrm{p}$. Thus,

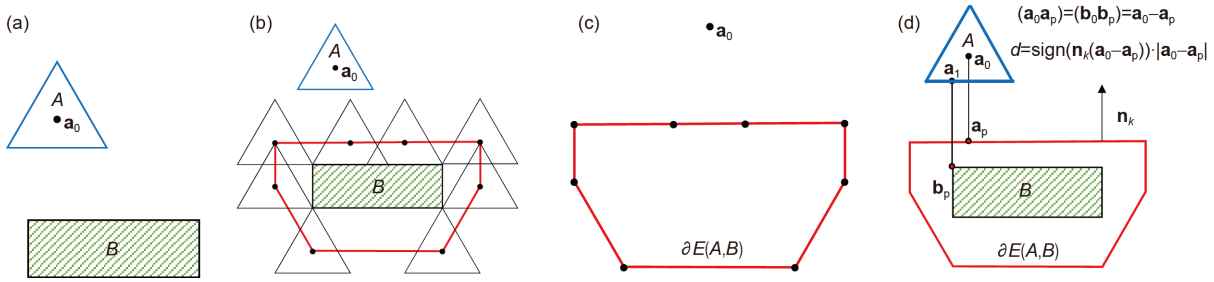


Figure 1 (Color online) Simplification of the geometric relation between a triangle block A and a triangle block B. (a) Relation between block A and block B; (b) relation between $E(A,B)$ and $\mathbf{a}_0$; (c) relation between $\partial E(A, B)$ and $\mathbf{a}_0$; (d) the distance between two blocks and contact point pairs.

block A is in contact with block B at that overlapped points ($\mathbf{a}_1$ and $\mathbf{b}_p$). For these points, the following relation exists (referring to Figure 1(d)):

$$\mathbf{a}_p = \mathbf{b}_p - \mathbf{a}_1 + \mathbf{a}_0. \tag{3}$$

The overlapped points ($\mathbf{a}_1$ and $\mathbf{b}_p$) are named as a contact point pairs. When two blocks contact, at least a contact point pairs, as one on block A and another on block B, can be found. The contact point pairs have the same coordinate. In mechanical analysis, suitable contact springs or penalties can be applied to the contact point pairs to avoid the penetration between block A and block B. In dynamic analysis, the tracks of the contact point pairs can be used to calculate when and where they will crush together.

2.2 The definition of entrance block

Denote A, B as two blocks and choose $\mathbf{a}_0$ as a reference point of block A, which is moving together with A. In geometric meaning, the entrance block $E(A, B)$ is the track of $\mathbf{a}_0$ while translating block A along the boundary and keeping in contact with block B. The mathematical definition of entrance block is as follows:

$$E(A,B) = B - A + \mathbf{a}_0 = \cup_{\mathbf{a}\in A,\mathbf{b}\in B}(\mathbf{b} - \mathbf{a} + \mathbf{a}_0), \tag{4}$$

where B–A is defined as $B + (-A)$, with + being the Minkowski Sum operation [36] that is an operand acting two geometrical objects; $\mathbf{a}$ is an arbitrarily point on block A; $\mathbf{b}$ is an arbitrarily point on block B; – is the inverse operation, that's to say point $-\mathbf{a}$ and point $\mathbf{a}$ is symmetrical about the origin (0, 0, 0).

Ref. [1] has extended the Minkowski Sum and Inverse operation [36] from points to any geometrical objects. All the geometrical objects to be handling are real subsets in the 3D affine space, where points and vectors are two separate categories of elements, and obey different operations. In this study, however, the difference between points and vectors will be ignored, since our discussion is always in a fixed Descartes system, with no other Descartes systems involved. In this study, a point, $\mathbf{a}_0$ say, is also the vector (still denoted by $\mathbf{a}_0$) by connecting the origin and $\mathbf{a}_0$.

As shown in Figure 2, $\mathbf{b}$ is a point indicated by the vector $\mathbf{n}_j$, and A is a general geometry objects, i.e. either a point, a 2D polygon or a 3D block. Referring to Figure 2, $\mathbf{a}$+$\mathbf{b}$ obeys the vector addition operation and A+$\mathbf{b}$ is a parallel translation of block A along vector $\mathbf{b}$ in geometric meaning (Figure 2).

When calculating $E(A, B)$ by eq. (4), every point of $E(A, B)$ is a position of $\mathbf{a}_0$. When block A and block B have common points $\mathbf{b}$=$\mathbf{a}$. So if assuming $\mathbf{x}$ as a point or a vector, $E(A, B)$ is the union of all points $\mathbf{a}_0$+$\mathbf{x}$ with the condition that $(A + \mathbf{x}) \cap B \neq \varnothing$ and can be also written as

$$E(A,B) = \cup_{(A+\mathbf{x})\cap B\neq\varnothing}(\mathbf{a}_0 + \mathbf{x}). \tag{5}$$

Figures 1(b) and 3 illustrate two 2D examples of $E(A, B)$.

The use of entrance block can help to transfer the contact relations between two blocks to the entrance relations between one block and one point. Therefore, the contact computation can be simplified as two tasks: (1) solving the boundary of entrance block, $\partial E(A, B)$; and (2) judging whether $\mathbf{a}_0$ is located inner, outer, or lies on $\partial E(A, B)$.

3 Data structure

3.1 Data structure of geometry objects

In computation program, different types of geometry object are described by different classes, as listed in Table 1. In Table 1, Class Cpoint, Cline, Cpolygon, Cdihedral, Cangle, Cblock represents a point, a line, a polygon, a dihedral angle, a 3D solid angle, and a 3D block, respectively.

A 3D block is illustrated in Figure 4, where $A(0)$, $A(1)$, $A(2)$ are the vertex set, edge set, and boundary polygon set of block A. For example, the concave block shown in Figure 4 has 12 vertexes, 18 edges and 8 boundary polygons. The block can be defined by its boundary, i.e. the 8 boundary polygons. Each vertex is corresponding to a solid angle and each edge is corresponding to a dihedral angle. So there are 12 solid angles and 18 dihedral angles in Figure 4.

Among those data, the boundary polygons $A(2)$ is the necessary and sufficient to define block A. Each boundary polygon of the block, i.e. one element in $A(2)$, is a polygon with normal vector points outer the block (as shown in Figure 4(c)). $A(2)$ is saved in polygonList of Class Cblock. Thus, polygonList is the basic information required for a Cblock object. Other data in a Cblock object can be null and initialized by using the information from polygonList.

For convenience, $A(0)$, $A(1)$ should be generated and will

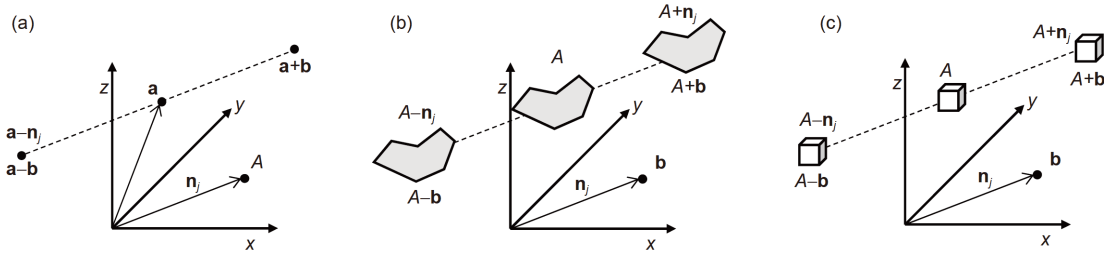


Figure 2 The operation of A+$\mathbf{b}$, where $\mathbf{b}$ is a point. (a) A is a point; (b) A is a polygon; (c) A is a block.

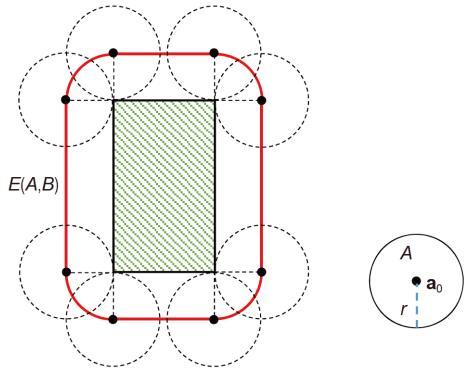


Figure 3 (Color online) An example of $E(A, B)$ where A is a disk and B is a square.

be used in the checking of contact condition. Each edge of the block can be regarded as the common edge of a dihedral angle. These dihedral angles are saved in diheList of Class Cblock, and used to represent $A(1)$. For example, edge $\mathbf{a}_1\mathbf{a}_6$ is the common edge of P_1 and P_8 (Figure 4). $\mathbf{a}_1\mathbf{a}_6$ can be defined by the dihedral angle $D(P_1, P_8)$.

For a 3D block, each vertex of the block can be regarded as the vertex of a solid angle. These solid angles are saved in angleList of Class Cblock and used to represent $A(0)$. Each solid angle can be defined by its vertex and several ordered vectors. For example, vertex $\mathbf{a}_1$ can be defined by the vertex of solid angle composed by $\mathbf{a}_1-\mathbf{a}_7\mathbf{a}_2\mathbf{a}_6$ (Figure 4). A 3D solid angle is illustrated in Figure 5, where the vertex is $\mathbf{a}$ and there are 3 boundary faces. The i boundary face has two edges, i.e. $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$, and its inner normal vector $\mathbf{n}_i$ can be calculated as $\mathbf{e}_{i+1}\times\mathbf{e}_i$.

3.2 The algebraic representation of geometry objects

In computation program, a geometry object must be algebraically defined. In this study, a geometry object is defined by its boundaries. Common geometric objects are illustrated

Table 1 Data structure for the representation of geometry objects (language: C#)

Class name	Geometry	Data type	Variables	Meaning
Cpoint	Point or vector, e.g. $\mathbf{a}$, $\mathbf{e}_i$, $\mathbf{n}_k$	Integer	ID	Vertex No.
		Double	x, y, z	Coordinate values
Cpine	Oriented edge, e.g. **ab**	Integer	ID	Edge No.
		Cpoint	p0, p1	Start and end
		Cpoint	vector	Direction of line
Cpolygon	Oriented polygon, e.g. Figures 4(c) and 6(b)	Integer	ID	Polygon No.
		List<Cpoint>	plist	Vertexes ordered in anti-clockwise direction
		Cpoint	inner	An inner point, which is often used as the reference point of this polygon
		Cpoint	vector	The inner normal vector of the polygon, i.e. $\mathbf{n}$
Cdihedral	Solid dihedral angel, e.g. Figure 8(b)	Integer	ID	The dihedral angle No.
		Bool	isConcave	Indicator of concave dihedral angel
		Cline	edge	The edge of dihedral angle. Its direction is $\mathbf{e}_r$
		Cpoint	n0, n1	The inner normal vectors of the two boundary face, i.e. $\mathbf{n}_{11}$ $\mathbf{n}_{12}$
		Cpoint	v0, v1	The vector that is perpendicular to the edge vector and inner the boundary face, i.e. $\mathbf{e}_1$ $\mathbf{e}_2$
Cangle	3D solid angle, e.g. Figure 7	Integer	ID	The solid angle No.
		Cpoint	vertex	The vertex of the angle, which is often used as the reference point of this angle
		Bool	isConcave	Indicator of concave angle
		List<Cpoint>	vectorlist	The boundary vectors ordered in right hand rule, i.e. $\mathbf{e}_i$
		List< Cpoint >	nodelist	The node related to this angle and ordered in right hand rule
Cblock	3D block, e.g. Figure 6(c) and (e)	Integer	ID	The block No.
		Cpoint	a0	An inner point, which is often used as the reference point of this angle
		Bool	isConcave	Indicator of concave block
		List<Cpolygon>	polyList	The boundary polygons of this block with normal vector pointed to inner block, i.e. $A(2)$ in Table 2 (Figure 4(c))
		List<Cangle>	angleList	The solid angles at the vertexes of this block that corresponds to the vertexes belonging to $A(0)$ in Table 2 (Figure 4(a))
		List<Cdihedral>	diheList	The dihedral angles containing the block edges that correspond to the edges belonging to $A(1)$ in Table 2 (Figure 4(b))

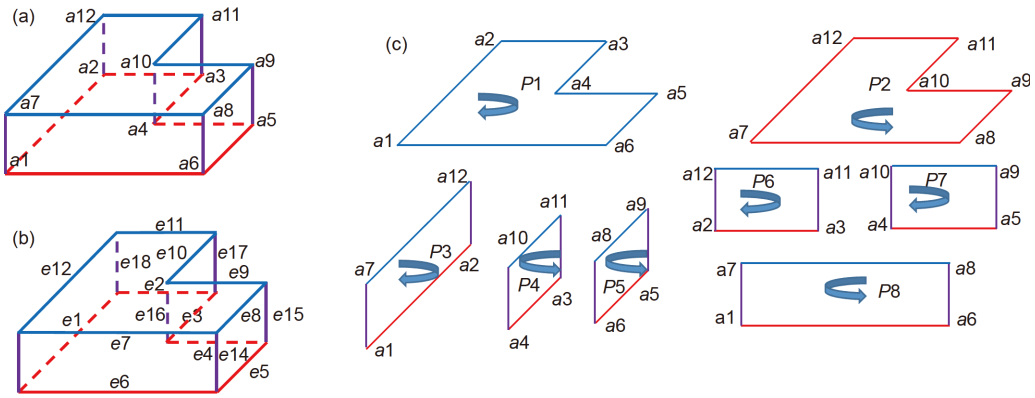


Figure 4 (Color online) The definition of a 3D block A. (a) The vertexes, $A(0)$; (b) the edges, $A(1)$; (c) the boundary polygons, $A(2)$.

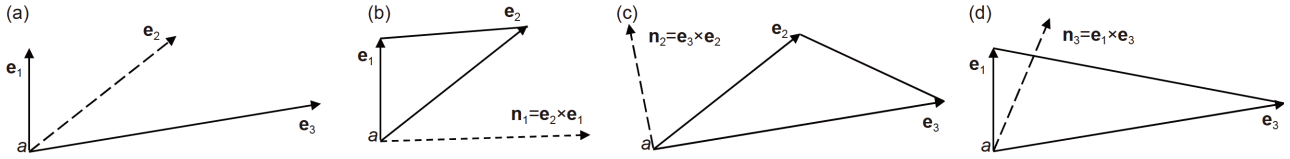


Figure 5 The definition of a 3D solid angle $\measuredangle \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$. (a) Solid angle with vertex of **a**; (b) boundary face P_1; (c) boundary face P_2; (d) boundary face P_3.

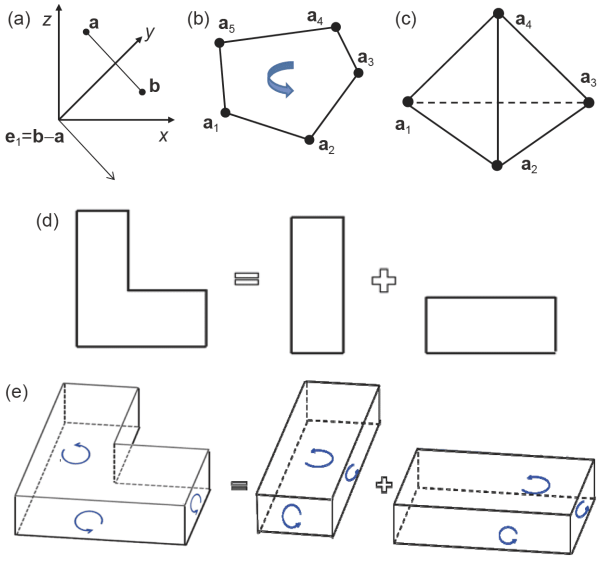


Figure 6 (Color online) Geometric objects and their expressions of point set. (a) Edge; (b) polygon; (c) tetrahedron; (d) concave polygon and its sub convex polygons; (e) concave block and its sub convex blocks.

in Figure 6. Following the symbols (referring to Table 2) used by ref. [1], geometry objects can be expressed in the form of point sets, as listed in Table 3.

Some common geometry objects are illustrated in Figure 6. Referring to Table 3 and Figure 6, a n (n = 2 or 3) dimension block can be defined by its boundaries and each boundary is a n–1 dimension object. An edge is limited by its two ends and has a direction from the start point to the end point. For example, an edge **ab** can be defined by its boundaries, i.e. points **a** and **b** (Figure 6(a)). A polygon is limited by its edges and has a normal vector, which is normal to its edges and obeying the right-hand rule. For example, a polygon on the ground plane with vertexes in anti-clockwise turn has a normal vector in upward direction (Figure 6(b)). A 3D block is limited by its boundary polygons. For example, A triangle $P(\mathbf{a}_1\mathbf{a}_2\mathbf{a}_3)$ can be defined by its boundaries, oriented edges $\mathbf{a}_1\mathbf{a}_2$, $\mathbf{a}_2\mathbf{a}_3$ and $\mathbf{a}_3\mathbf{a}_1$. A tetrahedron A with four vertexes **a**, **b**, **c**, **d** can be defined by its boundaries, i.e. the oriented triangles P(**acb**), P(**cdb**), P(**bda**) and P(**dca**) (Figure 6(c)).

It should be emphasized that, for a 3D block, as illustrated in Figures 4, 6(c)–(e), it has a series of boundary polygons and each boundary polygon $P(\mathbf{a}_1\mathbf{a}_2...\mathbf{a}_k)$ is with vertexes numbered in right hand rule, as $\mathbf{a}_{k-1}\mathbf{a}_k \times \mathbf{a}_k\mathbf{a}_{k+1}$ points outer of the block. That's to say, the normal vector of a boundary polygon is pointing outwards.

The inner normal vectors of these boundary polygons are used to define the 3D block (Table 3). A concave block can be regarded as the union of several sub-convex blocks (Figure 6(d) and (e)), as the expression in Table 3.

4 Algorithm I: The contact cover searching algorithm

4.1 Search the possible contact covers

If two blocks contact, the contact point pairs (Figure 1(d)) must be located on the boundaries of the entrance block, i.e. $\partial E(A, B)$. Following ***Theorem of finite covers of entrance blocks*** [1], $\partial E(A, B)$ has a cover system

$$\partial E(A,B) \subset C(0,2) \cup C(2,0) \cup C(1,1), \tag{6}$$

Table 2 Symbols used in this paper (after ref. [1])

Symbol	Physical meaning
$E(A, B)$	The entrance block with a definition of $E(A, B)=\{\mathbf{a}_0+\mathbf{x}\|(A+\mathbf{x})\cap B\neq\emptyset\}$
A, B	1D, 2D, 3D geometry object, e.g. point, edge, polygon, block, etc.
∂A, ∂B	The boundaries of A,B
int(A)	The inner points of A, $\text{int}(A) = \{x \in A \text{ and } x \in\notin \partial A\}$
$A(0)$	The vertex of A, e.g. $A(0) = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_k\}$
$A(1)$	The boundary edge of A, e.g. $A(1)=\mathbf{a}_1\mathbf{a}_2\cup\mathbf{a}_2\mathbf{a}_3\cup...\cup\mathbf{a}_{p-1}\mathbf{a}_p\cup\mathbf{a}_p\mathbf{a}_1$
$A(2)$	The boundary polygon of A
x, y, z, t	Real number
i, j, k, r, s, u, v, w	Natural number
$\mathbf{x}$	Point $\mathbf{x}=\{x, y, z\}$
$\mathbf{a}$	Point $\mathbf{a}=\{x_i, y_i, z_i\}$
$\mathbf{b}$	Point $\mathbf{b}=\{x_j, y_j, z_j\}$
$L(\mathbf{ab})$, or $\mathbf{ab}$	An edge from $\mathbf{a}$ to $\mathbf{b}$
$P(\mathbf{a}_1\mathbf{a}_2...\mathbf{a}_k\mathbf{a}_{k+1})$	A polygon with ordered vertexes $\mathbf{a}_1\mathbf{a}_2...\mathbf{a}_k\mathbf{a}_{k+1}$ with $\mathbf{a}_{k+1}=\mathbf{a}_1$
$D(P_1, P_2)$	A dihedral angle with the two outer boundary surfaces of $P_1$and P_2
$B(P_1, P_2, ..., P_k)$	A 3D block with outer boundary surfaces of P_1, P_2, ..., P_k
$B(D_1, D_2, ..., D_k)$	A 3D block containing inner dihedral angles of D_1, D_2, ..., D_k
$\measuredangle\mathbf{e}_1\mathbf{e}_2...\mathbf{e}_u\mathbf{e}_{u+1}$	A solid angle with boundary edges $\mathbf{e}_r$, $0<r<u+1$ and $\mathbf{e}_{u+1}=\mathbf{e}_1$
S	A plane
V	A space
$\mathbf{n}_i$	$\mathbf{n}_i=(x_i, y_i, z_i)$ is a vector from (0,0,0) to (x_i, y_i, z_i). It often refers to an inner normal vector, which is normal to a boundary polygon or plane and points to the inside of the 3D block
$\mathbf{m}_j$	$\mathbf{m}_j=(x_j, y_j, z_j)$ is a vector. It often refers to an inner normal vector which is normal to a boundary edge and points to the inside of the 2D polygon
$\mathbf{e}_r$	$\mathbf{e}_r$ is a vector parallel to the r edge of block A
$\mathbf{h}_s$	$\mathbf{H}_s$ is a vector parallel to the s edge of block B
$\|\|$	Parallel, e.g. $\mathbf{n}_i\|\|\mathbf{n}_j$ means $\mathbf{n}_i=t\mathbf{n}_j$, t is a real number
$\uparrow\uparrow$	Parallel and in the same direction, e.g. $\mathbf{n}_i\uparrow\uparrow\mathbf{n}_j$ means $\mathbf{n}_i=t\mathbf{n}_j$, $t>0$
$\perp$	Normal, e.g. $\perp\mathbf{n}_i$ is a plane passing (0,0,0), $\mathbf{n}_i$ is the normal vector of the plane
$\uparrow$	At the same side, $\mathbf{n}_i\uparrow\mathbf{n}_j$ means $\mathbf{n}_i\mathbf{n}_j\geq 0$

where

$$\begin{cases} C(0,2) = E(A(0), B(2)), C(2,0) = E(A(2), B(0)), \\ C(1,1) = E(A(1), B(1)). \end{cases} \tag{7}$$

Obviously, $C(0,2)$ refers to the angle of block A is in contact with the boundary face of block B; $C(2,0)$ refers to the boundary face of block A is in contact with the angle of block B; $C(1,1)$ refers to the edge of block A is in contact with the edge of block B.

Eq. (6) can be used to search all the possible boundaries of $E(A, B)$. However, most of the covers (or elements) in the right part of eq. (6) are inner $E(A, B)$ and not the boundary of $E(A, B)$. Therefore, criterion of checking whether a cover is the boundary of $E(A, B)$ is required. Such criterions are named as contact conditions [1].

The contact conditions are used to screen out whether an element in $C(0,2)$, $C(2,0)$ or $C(1,1)$ can serve as the boundary.

4.2 Contact condition for the angle to face contact

$C(0,1)$ and $C(2,0)$ handle the angle to face contacts. Referring to Figure 7, the contact condition for an angle to face contact is

$$\mathbf{e}_i \cdot \mathbf{n} < 0, \quad i = 1, 2, ..., u, \tag{8}$$

where $\mathbf{e}_i$ is a boundary vector of the solid angle containing the vertex and starts from the vertex; u is the count of the boundary vectors, as 4 in Figure 7; $\mathbf{n}$ is the inner normal vector of the face. Referring to Figure 7, the solid angle of block A is located in the upper half spaces, i.e. $A \subset \uparrow(-\mathbf{n}) + \mathbf{a}_0$. Thus, the solid angle of block may contact with block B on the boundary polygon P and eq. (8) serves as the contact condition, and the contact cover is $E(\mathbf{a}_0, P)$. The calculation of $E(\mathbf{a}_0, P)$ can be referring to Table 4 and illustrated in Figure 2(b). Clearly, $E(\mathbf{a}_0, P)$ is a polygon and has a

Table 3 The representation of 3D geometric objects in form of point sets

Geometric object	Symbol	Point set expression
Point	$\mathbf{a}$	$\mathbf{a}$
Vector (Figure 6(a))	$\mathbf{e}_1$	$\mathbf{e}_1 = \mathbf{b} - \mathbf{a}$
Edge (Figure 6(a))	$L(\mathbf{ab})$	$\mathbf{ab} = \mathbf{a} + \cup_{0\le t\le 1} t(\mathbf{b}-\mathbf{a})$
Plane passing point $\boldsymbol{O}$	$\perp \mathbf{n}_r$	$\perp \mathbf{n}_r = \{\mathbf{x}\cdot\mathbf{n}_r = 0\}$
Plane passing $\mathbf{a}_0$	S	$S = \mathbf{a}_0 + (\perp \mathbf{n}_r)$
Half space passing point $\boldsymbol{O}$	$\uparrow \mathbf{n}_r$	$\perp \mathbf{n}_r = \{\mathbf{x}\cdot\mathbf{n}_r = 0\}$
Convex polygon (Figure 6(b))	P	$P = S \cap A,\ S = \perp \mathbf{n}_r$ $A = \cap_{k=1,2\ldots f}(\mathbf{m}_k + \mathbf{a}_k)$
Half space with boundary passing $\mathbf{a}_0$	V	$V = \mathbf{a}_0 + \uparrow \mathbf{n}_r$
Convex block (Figure 6(c))	A or B	$A = \cap_{k=1,2\ldots f}(\mathbf{n}_k + \mathbf{a}_k)$, where $k \in \{1,\cdots,u\}$; $\mathbf{n}_k$ is the inner normal vector of boundary polygon P_k, which is pointing to the inner of block A; $\mathbf{a}_k$ is a point on the boundary polygon P_k
Concave polygon (Figure 6(d))	P	$P = \cup_1 C_1$, where C_i is the i sub convex polygon, $i \in \{1,\cdots,u\}$, and u is the number of the sub convex polygon
Concave block (Figure 6(e))	A or B	$A = \cup_1 C_1$, where C_i is the i sub convex block, $i \in \{1,\cdots,u\}$, and u is the number of the sub convex block

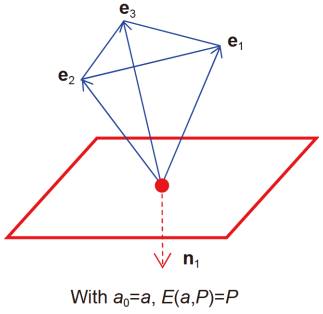


Figure 7 (Color online) Angle to face contact ($P\subset B(2)$, $\mathbf{a}\subset A(0)$, $\mathbf{a}_0=\mathbf{a}$).

shape as same as that of P.

4.3 Contact condition for the edge to edge contact

$C(1,1)$ is corresponding to the edge to edge contacts. Referring to Figure 8(a), the contact condition for an edge to edge contact is

$$\mathbf{e}_1\cdot\mathbf{n}_{rs}\ge 0,\ \ \mathbf{e}_2\cdot\mathbf{n}_{rs}\ge 0,\ \ \mathbf{h}_1\cdot\mathbf{n}_{rs}\le 0,\ \ \mathbf{h}_2\cdot\mathbf{n}_{rs}\le 0, \tag{9}$$

where edge .$\mathbf{e}_r$. of dihedral angel A is in contact with edge $\mathbf{h}_s$ of dihedral angel B; $\mathbf{n}_{rs}$ is the normal vector of the contact face, as

$$\mathbf{n}_{rs} = \mathbf{e}_r \times \mathbf{h}_s \tag{10}$$

and $\mathbf{e}_1$, $\mathbf{e}_2$ are two boundary vectors of the dihedral angel A with a common edge of $\mathbf{e}_r$; $\mathbf{h}_1$, $\mathbf{h}_2$ are two boundary vector of the dihedral angel B with a common edge of $\mathbf{h}_s$. As shown in Figure 8(b),

$$\begin{aligned}&\mathbf{e}_1 = \mathbf{n}_{11}\times\mathbf{e}_r,\ \mathbf{e}_2 = \mathbf{e}_r\times\mathbf{n}_{12},\\ &\mathbf{h}_1 = \mathbf{n}_{21}\times\mathbf{h}_s,\ \mathbf{h}_2 = \mathbf{h}_s\times\mathbf{n}_{22},\end{aligned} \tag{11}$$

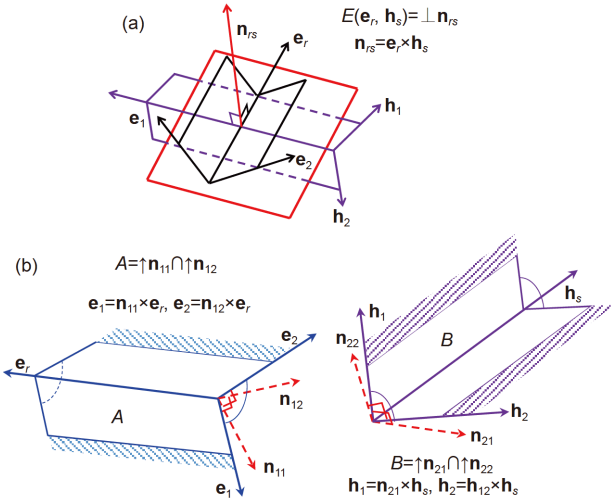


Figure 8 (Color online) Edge to edge contact. (a) Edge $\mathbf{e}_r$ of dihedral angel A contacts with edge $\mathbf{h}_s$ of dihedral angel B; (b) dihedral angles A and B.

where $\mathbf{n}_{11}$, $\mathbf{n}_{12}$ are the inner normal vectors of the two boundary face of dihedral angel A; $\mathbf{n}_{21}$ and $\mathbf{n}_{22}$ are the inner normal vectors of the two boundary face of dihedral angel B. Obviously, when $\mathbf{e}_r$ touches and moves along $\mathbf{h}_s$, the contact face is formed, i.e. $E(\mathbf{e}_r, \mathbf{h}_s)$. Referring to Figure 8(a), dihedral angel A and dihedral angel B are in two half spaces separated by the contact face $E(\mathbf{e}_r, \mathbf{h}_s)$. They may contact on the boundary polygon $E(\mathbf{e}_r, \mathbf{h}_s)$ and eq. (9) serves as the contact condition.

The calculation of $E(\mathbf{e}_r, \mathbf{h}_s)$ can be referring to Table 4 and illustrated in Figure 9. Referring to Figure 9, if $\mathbf{ab}$ is not parallel to $\mathbf{cd}$, $\mathbf{ab}\pm\mathbf{cd}$ forms a parallelogram with four or-

dered vertexes, **a**±**d**, **b**±**d**, **b**±**c**, **a**±**c**. If **ab** is parallel to **cd**, the parallelogram yields into two overlapped edge and **ab**±**cd** will be an edge from **a**±**c** to **b**±**d**, or an edge from **a**±**d** to **b**+**c**. It should be mentioned that **ab** and **ba** share the same expression of point set, but with inverse directions. Subsequently, **ab**+**cd** and **ba**+**cd** will form polygons with same region, but with inverse directions, i.e. the inverse turns of vertexes.

Obviously, the contact cover $E(\mathbf{e}_r, \mathbf{h}_s)$ is a parallelogram.

4.4 Implementation of the contact searching process

Eq. (6) simplifies the complex operation for solving $E(A, B)$ to a series of simple cases. There are only two kinds of contacts existed in the calculation of $\partial E(A, B)$.

(1) For $C(0,2)$ or $C(2,0)$, there are angle to face contacts. For the angle to face contacts, i.e. $C(0,2)$ and $C(2,0)$, eq. (8) will be used to judge whether the contact cover is a possible boundary of $E(A, B)$. For an angle to face contact, the contact cover is the entrance block between a point and a polygon. The flow chart of searching the possible boundary polygons of entrance block is shown in Figure 10. The flow chart for solving $C(0,2)$ is shown in Figure 11. After switching A and B in Figure 11, the flow chart can be used to solve $C(2,0)$. The formula used in Figure 11 will be explained in details by an example in next section.

(2) For $C(1,1)$, there are edge to edge contacts. For the edge to edge contacts, i.e. $C(1,1)$, eq. (9) will be used. For an edge to edge contact, the contact cover is the entrance block between two edges. The flow chart for solving $C(1,1)$ is illustrated in Figure 12. The formula used in Figure 12 will be explained in details by an example in next section.

The solutions of both the upper two kinds of operation are polygons. Therefore, $\partial E(A, B)$ has a cover structure and each cover is a polygon. A cover (or element) met the contact condition is named as a contact cover or contact polygon.

If both blocks A and B are convex blocks, every contact cover is a boundary polygon of $E(A, B)$, i.e. on $\partial E(A, B)$ and all these contact covers will form a closed $E(A, B)$.

However, if one of A and B is concave, some covers (or elements) met the contact condition may be inner of $E(A, B)$. There are at least two choices for handling the contact problem involving concave block. One is cutting the concave into sub-convex blocks and solving the entrance block by using the sub-convex blocks. Another is forming the outer boundary by a block cutting and boundary searching algorithm based on the cover set containing all these covers. In this study, the second choice is adopted.

In fact, in the contact searching process, the outer

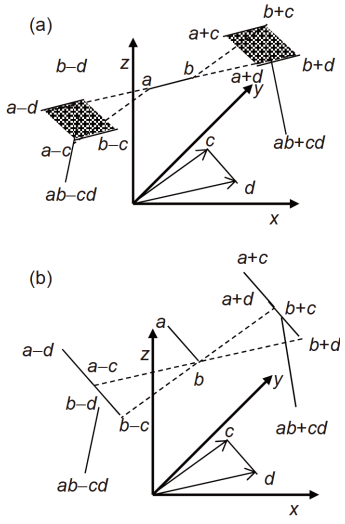


Figure 9 An edge **ab** sum another edge **cd**. (a) **ab** is not parallel to **cd** (b) **ab** ∥ **cd**.

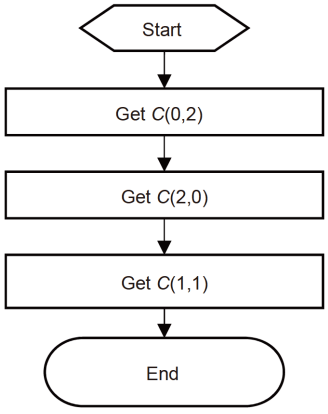


Figure 10 Flow chart of searching the contact covers (subroutine Covers_solve).

Table 4 The sum or inverse operation between two simple point sets

Operation	Figure	Rule	Result
+		$A+B=\cup_{\mathbf{a}\in A,\ \mathbf{b}\in B}(\mathbf{a}+\mathbf{b})$	A block
−		$\cup_{\mathbf{b}\in B}(-\mathbf{b})$	A point
a±**b**		$\mathbf{a}\pm\mathbf{b}=(x_i\pm x_j,\ y_i\pm y_j,\ z_i\pm z_j)$	A point
A±**b**	Figure 2(a)	$A\pm\mathbf{b}=\cup_{\mathbf{a}\in A}(\mathbf{a}\pm\mathbf{b})$	A block
P±$\mathbf{a}_0$	Figure 2(b)	$P(\mathbf{a}_1,\ \mathbf{a}_2,\ \ldots,\ \mathbf{a}_k)\pm\mathbf{a}_0=P(\mathbf{a}_1\pm\mathbf{a}_0,\ \mathbf{a}_2\pm\mathbf{a}_0,\ \ldots,\ \mathbf{a}_k\pm\mathbf{a}_0)$	A polygon
ab±**cd**	Figure 9	$\mathbf{ab}\pm\mathbf{cd}=P(\mathbf{a}\pm\mathbf{c},\ \mathbf{b}\pm\mathbf{c},\ \mathbf{b}\pm\mathbf{d},\ \mathbf{a}\pm\mathbf{d})$	A parallelogram

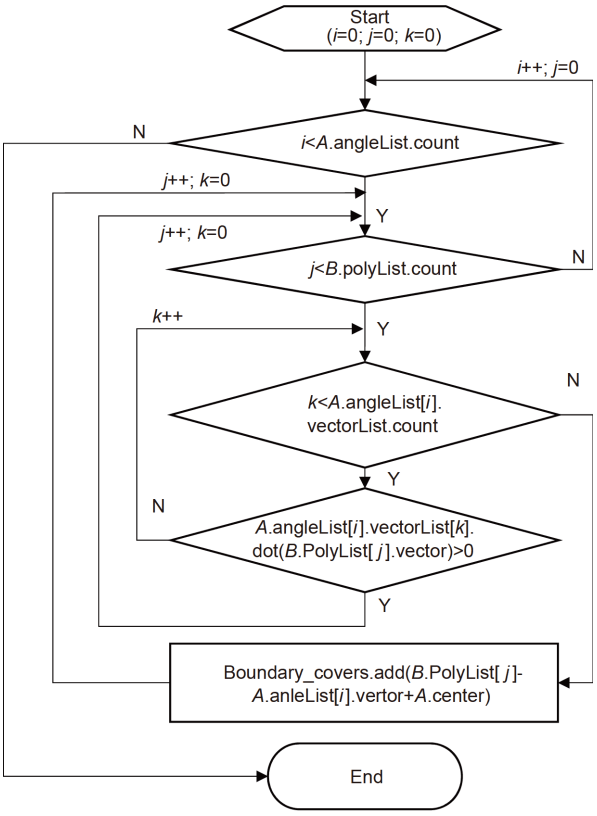


Figure 11 Flow chart for solving the contact convers of the angle to face contacts, i.e. for $C(0,2)$.

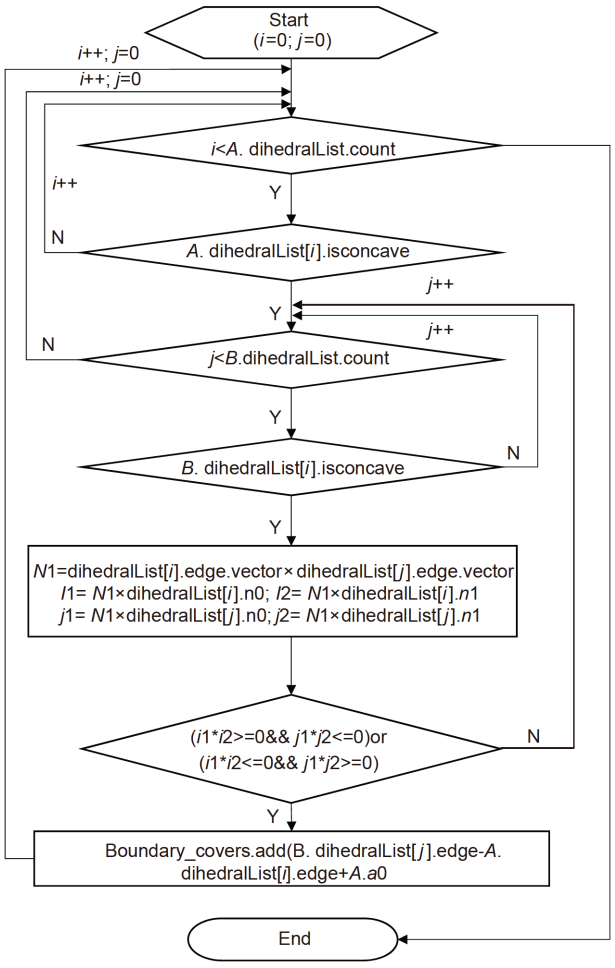


Figure 12 Flow chart for solving the contact covers of the edge-edge contacts, i.e. for $C(1,1)$.

boundary of the entrance block $\partial E(A, B)$ is not required. The contact covers inner $E(A, B)$ will has large distances to the reference point and never be found when searching the possible contact position. That's to say, they will never serve as a contact position in reality and the set containing all the contact covers can be used to search the possible contact position directly.

The flow chart for contact searching is shown in Figure 13, where the contact covers is used directly without forming $\partial E(A, B)$. When searching the contact point pairs, the distance between the reference point and a contact cover will be calculated (referring to Figure 1(d)). Once the distance is negative, the contact will be considered to be close and contact force will be existed between that contact point pairs. All the contact covers will be checked. A detailed example will be reported in the followed Sect. 6.

5 An example of the contact cover searching algorithm

An example is reported here to show the computation details of the contact cover searching algorithm. As illustrated in Figure 14, block A is a tetrahedral. Block A has four vertexes (Figure 14(a)), six edges (Figure 14(b)), and four outer boundary polygons (Figure 14(c)). Each vertex is corresponding to a 3D solid angle and each edge is corresponding to a dihedral angle. Thus there are four solid angles, six dihedral angles and four boundary polygons to be used in the contact detection, as saved in angleList, polyList, and diheList, respectively. A reference point $\mathbf{a}_0$ is linking to block A, and assigned to the centroid point of block A.

Assuming block B is identical to block A. The entrance block $E(A, B)$ has three groups of possible boundaries, i.e. $C(0,2)$, $C(2,0)$, and $C(1,1)$.

5.1 Searching the contact cover for vertex-to-face contacts

Following the flow chart in Figure 11, all the elements in $C(2,0)$ or $C(0,2)$ will be checked one by one. When considering an angle to face contact, all the edges of the angle will be checked (eq. (8)). Considering the case that the i 3D angles of block A is contacting with the j boundary polygon of block B, if the i 3D angles of block A (remembered as bkA.

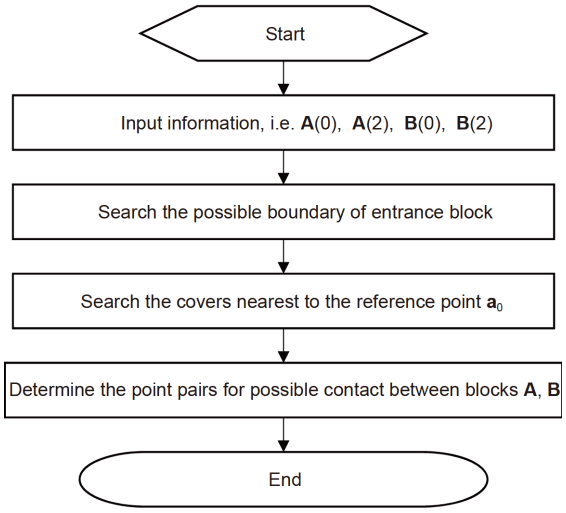


Figure 13 The flow chart for searching the contact covers.

angleList[i]) has k edges, there will be k flags to be calculated as

$$Fg(k) = bkA.\text{angleList}[i].\text{vecotorList}[k].\text{dot}(bkB.\text{polyList}[j].\text{vector}), \quad (12)$$

where bkA and bkB represent the block A and B, respectively; k is the ID of an edge of the i 3D angles of Block A; i is the ID of a 3D solid angle in block A; j is the ID of a boundary polygon of block B; Fg is the flags calculated by eq. (12); bkB.polyList[j].vector refers to the inner normal vector of the face, i.e. $\mathbf{n}$ in eq. (8); bkA.angleList[i].vectorList[k] refers to the k boundary vector of the solid angle, i.e. $\mathbf{e}_i$ in eq. (8); dot is a function for dealing with the dot product operation between two vectors. Further, the contact condition of eq. (8) can be used as

$$\forall k, Fg(k) \le 0 \quad \Rightarrow \quad \text{isBp} = \text{true}, \quad (13)$$

where isBp is an indicator for whether the polygon between the i 3D angles of block A and the j boundary polygon of block B may sever as a boundary of $E(A, B)$, i.e. a contact cover.

As listed in Table 5 and illustrated in Figure 15, the case that the 3D solid angle with vertex $\mathbf{a}_1$ of block A contacts with the four boundary polygons of block B is checked. The solid angle with vertex $\mathbf{a}_1$ has three vectors, i.e. $\mathbf{e}_k, k = 1, 2, 3$.

Table 5 The searching process for $C(0,2)$ in case of $i = 1$

j	j	Fg (eq. (14))	isBp (eq.(15))
1	1	=0	False
	2	=0	
	3	>0	
2	1	>0	False
	2	<0	
	3	<0	
3	1	<0	True
	2	<0	
	3	<0	
4	1	<0	False
	2	>0	
	3	<0	

Each boundary polygon of Block B has its own inner normal vector $\mathbf{n}_j, j = 1, 2, 3, 4$. Referring to Figure 15, only when j=4, the solid angle bkA.angleList[1] is possible to contact with bkB.ployList [4] (Figure 15(d)) and the formula for calculating the boundary contact polygon is

$$\text{Covers}[h] = bkB.\text{polyList}[j] - bkA.\text{angleList}[i].\text{vertex} + bkA.a0, \quad (14)$$

where Covers[h] is the h boundary polygon of $E(A, B)$. Obviously, eq. (14) demonstrates a parallel move of bkB.ployList[j] with a vector of (bkA.angleList[i].vertex), i.e. the result is $P_j + (\mathbf{a}_0 - \mathbf{a}_i)$. Such operation is quite similar to that illustrated in Figure 2(b).

Varying i and j in eq. (14), all the contact cover between A(0) and B(2) can be found, as illustrated in Figure 16(a). Together 4 contact covers are found. Similarly, all the contact cover between A(2) and B(0) can be found, as illustrated in Figure 16(b). For C(2,0), the formula for calculating the boundary contact polygon is

$$\text{Covers}[h] = bkB.\text{angleList}[j].\text{vertex} - bkA.\text{polyList}[i] + bkA.\text{a0}. \quad (15)$$

It can be seen that eq. (15) is an inverse operation of bkA.ployList[i] and a parallel move of $-bkA$.ployList[i] with a vector of (bkB.angleList[j].vertex + bkA.a0), i.e. the result

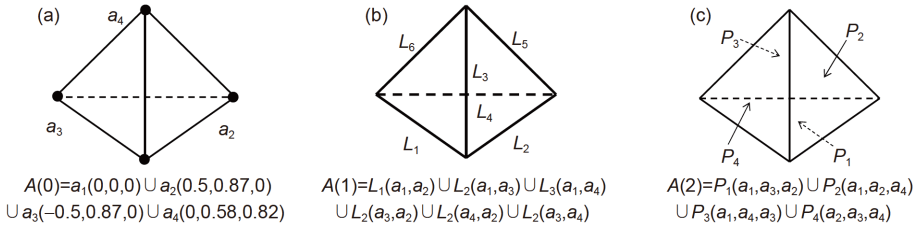


Figure 14 Numbering system of two identical blocks A and B. (a) Angle numbering system; (b) dihedral numbering system; (c) polygon numbering system.

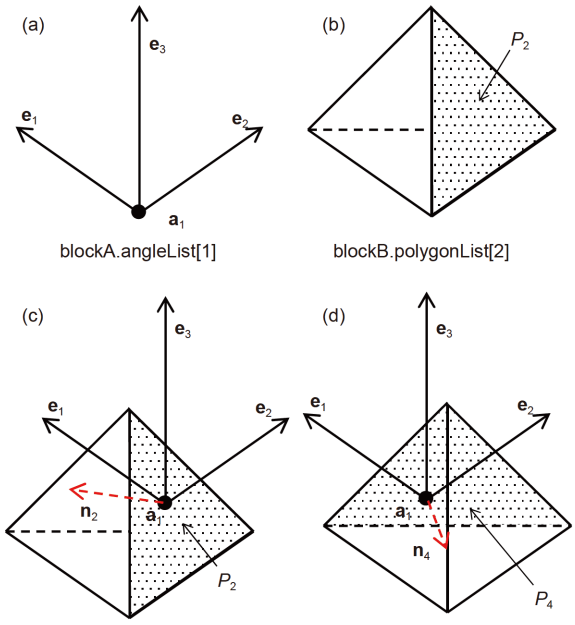


Figure 15 (Color online) An example for searching the boundary polygon of the entrance block, i.e. the contact cover (for case of i = 1). (a) The solid 3D angle of block A; (b) one of the boundary polygon of block B, P_j with j = 2; (c) a case against eq. (8) (P_j, j=3); (d) a case satisfying eq. (8) (P_j, j=4).

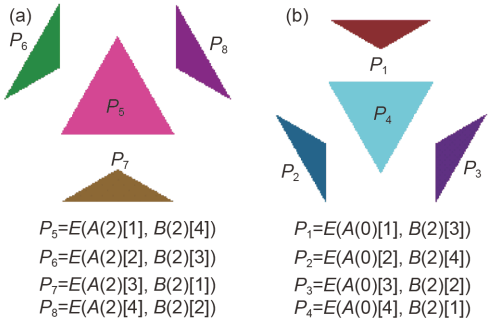


Figure 16 (Color online) Top view of the solved boundary polygons (or contact covers) on $\partial E(A, B)$. (a) Four covers from $C(0,2)$; (b) four covers from $C(2,0)$.

is $\left(-P_j+(\mathbf{a}_i+\mathbf{a}_0)\right)$. Referring to Figure 16, four polygons from $C(2,0)$ and four polygons from $C(0,2)$ are picked out and belong to $\partial E(A, B)$.

5.2 Searching the contact cover for edge-to-edge contacts

Following the flow chart in Figure 12, all the elements in $C(1,1)$ will be also checked one by one. When considering an edge to edge contact, eq. (9) will be checked.

Denote i is the ID of a dihedral angle in block A; j is the ID of a dihedral angle in Block B. There are 6 edges in $A(1)$, i.e. L_i with i=1, 2, 3, 4, 5, 6 and 6 edges in $B(1)$, i.e. L_j with j=1,2,3,4,5,6. Total 36 elements will be checked, as $E(L_i, L_j)$. Only the polygon satisfying the contact condition (eq. (9)) may serve as a boundary polygon of $E(A, B)$ The contact condition (eq. (9)) can be written as

$$\begin{cases} Fg_{i1} = bk\,\text{A.diheList}[i].\text{v1.dot(nvector)} \le 0, \\ Fg_{i2} = bk\,\text{A.diheList}[i].\text{v2.dot(nvector)} \le 0, \\ Fg_{j1} = bk\,\text{B.diheList}[j].\text{v1.dot(nvector)} \ge 0, \\ Fg_{j2} = bk\,\text{B.diheList}[j].\text{v2.dot(nvector)} \ge 0, \end{cases}$$
$$\text{or} \quad \begin{cases} Fg_{i1} \ge 0, \\ Fg_{i2} \ge 0, \\ Fg_{j1} \le 0, \\ Fg_{j2} \le 0, \end{cases} \qquad (16)$$
$$\Rightarrow \quad \text{isBP} = \text{true},$$

where v1 and v2 are the two vectors of a dihedral angle, i.e. $\mathbf{e}_1$, $\mathbf{e}_2$ or $\mathbf{h}_1$, $\mathbf{h}_2$ in eq. (9); dot is the dot product operation between two vectors; Fg_{i1}, Fg_{i2}, Fg_{j1}, Fg_{j2} are four Fgs; nvector is the normal vector of $E(L_i, L_j)$, i.e. $\mathbf{n}_{rs}$ in eq. (9), and can be calculated as

$$\text{nvector} = bkA.\text{diheList}[i].\text{edge.vector} \times bkB.\text{diheList}[j].\text{edge.vector}. \qquad (17)$$

It should be mentioned that the vectors in eqs. (9)–(11) and Figure 8 are sorted by right-hand rule whereas nvector used in eqs. (16) and (17) may be not. So there are two possible contact conditions existed in eq. (16).

In Table 6, the case i=1 is listed and corresponds to edge L_1 of block A. There are six edges in $B(1)$ to be checked, as L_j with j=1, 2, 3, 4, 5, 6. When j=1, the two edges are parallel. In such case, no boundary polygon can be formed for these two edges and $E(L_1, L_1)$ will be ignored. Referring to Table 6, only when j=6, $A(1)$ [1] is possible to contact with $B(1)$ [6] and the formula for calculating the boundary contact polygon is

$$\text{Covers}[h] = \text{sgn}^*(bkB.\text{diheList}[j].\text{edge} - bkA.\text{diheList}[i].\text{edge} + bkA.\text{a0}), \qquad (18)$$

where Covers[h] is the h boundary polygon of $E(A, B)$; sgn is the sign of Fg_{j1} and indicates whether Covers[h] is pointing outer the entrance block. It can be seen that eq. (16) is an operation of moving edge L_j along $-L_i$ and a parallel shifting of the polygon (L_j–L_i) by a vector of $\mathbf{a}_0$ (referring to Table 4 and Figure 9). The result is ((L_j–L_i)+$\mathbf{a}_0$).

Varying i and j, all the possible contacts among $A(1)$, $B(1)$ can be found, as listed in Table 7. Together 6 contact covers are found in $C(1,1)$.

Finally, these contact covers (or contact polygons) selected from $C(0,2)$, $C(2,0)$ and $C(1,1)$ form a closed $E(A, B)$, as illustrated in Figure 17. They will be used to identify the contact position in discontinuous computation.

Table 6 The searching process for $C(1,1)$ when i=1 (Figure 14(b))

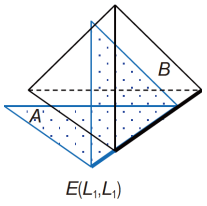
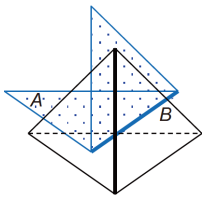
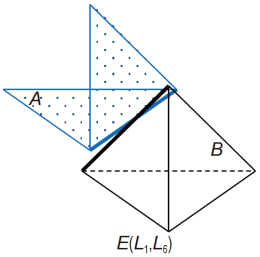
j	Fg (eq. (18))		isBP (eq. (18))	Sketch
1	$i1$	–	False (Parallel condition)	 $E(L_1,L_1)$
	$i2$	–		
	$j1$	–		
	$j2$	–		
3	$i1$	>0	False	 $E(L_1,L_3)$
	$i2$	0		
	$j1$	0		
	$j2$	>0		
6	$i1$	>0	True	 $E(L_1,L_6)$
	$i2$	>0		
	$j1$	<0		
	$j2$	<0		

Table 7 The result for solving the boundary polygons of edge-edge contacts, i.e. $C(1,1)$

i	j	Sketch
1	5	 (Top view)
2	6	
3	4	
4	3	
5	1	
6	2	

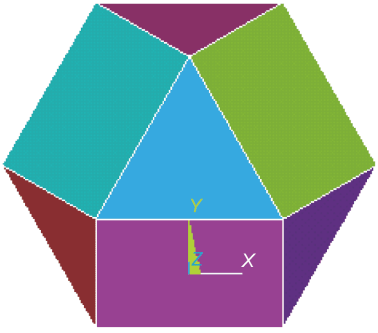


Figure 17 (Color online) Top view of a closed $E(A, B)$, which contains 4 covers from $C(0,2)$, 4 covers from $C(2,0)$ and 6 covers from $C(1,1)$.

6 Discussion: How to determine the contact mode with the solved contact covers

6.1 Application of the contact cover searching algorithm

When dealing with contact problems in numerical methods, such as FEM, DDA, DEM, and NMM, the proposed contact cover searching algorithm can offer all the possible contact boundaries between two blocks. When two blocks are approaching, the cover which has the closest distance with the reference point will serve as the contact boundary. The actual contact position can be solved by the relative position between that closest cover and the reference point (as shown in Figure 1(d)). If two blocks are moving, the contact time and position can be solved by the Newton's law of motion (Chapter 3, in ref. [32]).

As contact theory [1] is with solid mathematical derivation, the proposed algorithm is precise and no exception. At the meantime, the algorithm has the advantage of minimal calculation cost. The proposed algorithm is applicable to various numerical simulation methods and can be used in the contact computation involved in various engineering problems, such as robotic control, rock mass stability, dam stability, and computation geometrics etc.

Among these solved contact covers, each contact cover (a polygon or a line) selected from $C(0,2)$, $C(2,0)$ represents a

possible angle to face contact. Simultaneously, each contact cover (a polygon or a line) selected from $C(1,1)$ represents a possible edge to edge contact. These contact covers yield to a finite cover system. Such finite contact cover system is very convenient in the identification of contact mode. If the reference point is close to a contact cover, i.e. may contact with that polygon (or line) in the next move, that contact mode will be recorded in the discontinuous analysis and used in the open-close iteration (or any other contact dealing method, e.g. complementarity theory proposed by Zheng and Li [37]).

6.2 Special treatment for the overlapped contact covers

However, if sometimes there are overlapped contact covers, the contact mode becomes complex and special treatment is required. If a boundary polygon (or edge) of block A is parallel to and possible contact with another boundary polygon (or edge) of block B, overlapped contact covers will be found in $C(0,2)$, $C(2,0)$ or $C(1,1)$.

An example with parallel contact problem is shown in Figures 18 and 19. The coordinate and numbering system of cubic block A is shown in Figure 18. In Figure 19, the entrance block between two same cubic blocks, i.e. A=B, is solved. The front face illustrated in Figure 19 is composed by 16 contact covers, as listed in Table 8. These contact covers overlap each other and form 4 fragments. The right top fragment on the front face is overlapped by four contact covers (Figure 19), i.e. ($A.\mathbf{a}_4$, $B.P_3$) from $C(0,2)$, $E(A.P_5, B.\mathbf{a}_6)$ from $C(2,0)$, $E(A.L_6, B.L_5)$ from $C(1,1)$, and $E(A.L_8, B.L_9)$ from $C(1,1)$. Because one cover indicates a possible contact mode, the right top fragment on the front face is corresponding to four possible contact modes, including two edge-edge contacts, one angle-face contact, and one polygon-angle contact.

In fact, for the entrance block illustrated in Figure 19, each fragment on the front face is corresponding to 4 contact covers. The front face contains 16 contact covers. For the whole surface of the entrance block, $\partial E(A, B)$ is composed by 96 covers, as 24 covers from $C(0,2)$, 24 covers from $C(2,0)$. , and 48 covers from $C(1,1)$.

As another example, the entrance block between two regular octahedrons is shown in Figure 20. It has 8 faces and each face is composed by 12 covers. These covers are partially overlapped and include 6 parallelograms formed by edge-edge contacts and 6 triangles formed by angle-face contacts or face-angle contacts.

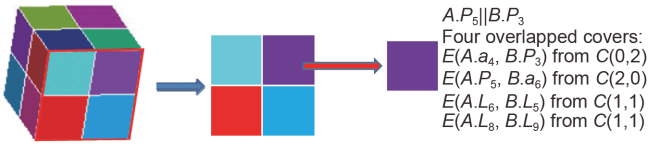


Figure 18 (Color online) Numbering system of cubic A, B.

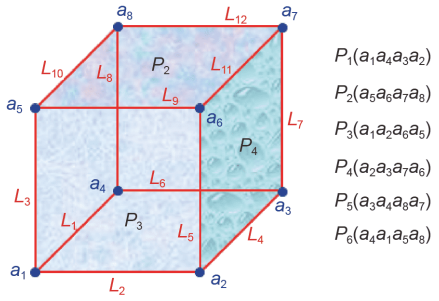


Figure 19 (Color online) Finite contact covers on the front face of the entrance block of two cubic (A=B).

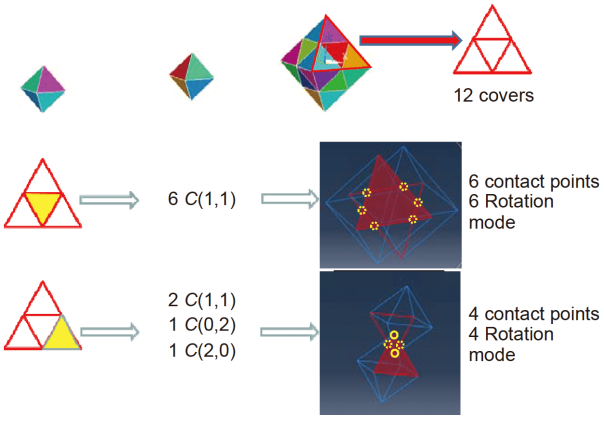


Figure 20 (Color online) Overlapped contact covers on $\partial E(A, B)$ of two regular octahedrons.

If the reference point is close to k overlapped contact covers, all k contact covers will be recorded and k contact modes will be used in the open-close iteration. For the four overlapped contact covers in Figure 19, these are four contacts and will control the relative movement between block A

Table 8 The contact covers on the front face of $E(A, B)$ (referring to Figures 20 and 21)[a)]

$C(0,2)$		$C(2,0)$		$C(1,1)$			
i	j	j	i	$k1$	$k2$	$k1$	$k2$
3	3	5	1	7	2	6	5
4	3	5	2	8	2	12	5
7	3	5	5	6	3	7	9
8	3	5	6	12	3	8	9

a) i is the angle ID; j is the polygon ID, $k1$ is the first dihedral ID; $k2$ is the second dihedral ID.

and block B as following two cases:

(a) If the face $A.P_5$ is sliding on the face $B.P_3$, all four contacts are close and have contact force.

(b) If $A.P_5$ has an angle with $B.P_3$, e.g. larger than 3° (a tolerance specified in the programming), these four contact covers will not be overlapped anymore. Therefore, only one contact cover is the closest and corresponding to a unique contact mode.

To determine which contact cover is the controlling one in discontinuous analysis, all the overlapped contact covers will be used in the open-close iteration. If one contact is close (as d<0 in eq. (1)), it will work and has contact force. If it is open, it will not work and has nothing to do the relative movement between block A. and block B. That's tsay, the controlling contact covers will be solved by open-close iteration automatically and without any exception. By checking the distance between the reference point and the contact cover, all the close contacts (or contact pair points) can be found.

In short, $\partial E(A, B)$ is a finite cover system and each cover is corresponding to a contact mode. All the contact covers must be recorded. These contact covers is necessary and sufficient for the further determination of contact mode and contact position.

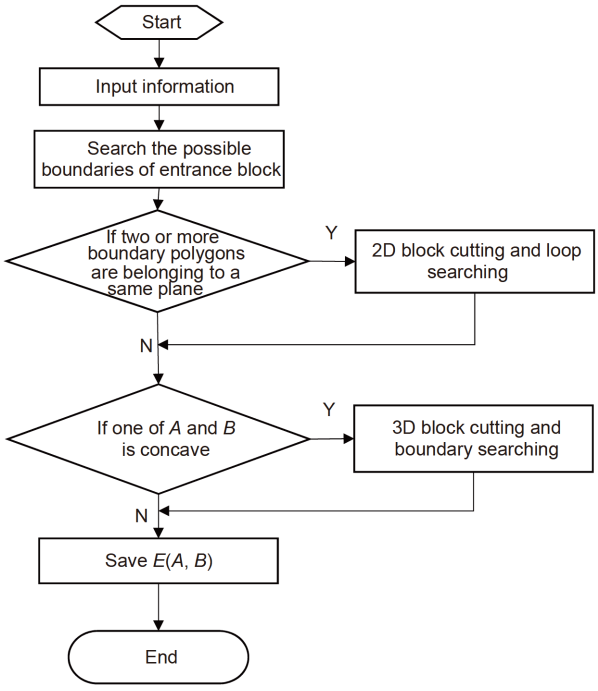


Figure 21 The flow chart for solving 3D entrance block $E(A, B)$.

7 Algorithm II: The entrance block forming algorithm

The entrance block is the key concept of contact theory. Even it is not required in the contact computation, it can help to understand contact theory. The contact covers solved by the upper contact cover searching algorithm can be used to generate the entrance block with using a proper block cutting algorithm.

7.1 The flow chart of program for solving 3D entrance block

The flow chart of the main program for searching the 3D entrance block is illustrated in Figure 21. The program has the following steps:

(1) Input information of A(2) and B(2)(2), which will be used to generate two Cblock objects, as block_A and block_B.

(2) Search the contact covers. This function is programmed based on ***Theorem of finite covers of entrance blocks*** (eq. (6)) and the two contact conditions (eqs. (8) and (9)). A subroutine named as "Covers_solve(Cblock A, Cblock B)" is used to search the contact covers and returns a list of boundary polygons, i.e. a List<Cpolygon> object covers. Each boundary polygon is regarded as a cover and corresponds to a contact mode. All these covers will be used to form $\partial E(A, B)$.

(3) Deal with the overlapped boundary polygons. Among all the boundary polygons in covers, if two or more boundary polygons are overlapped, they must be divided into small pieces, i.e. sub-polygons, by a 2D block cutting and loop searching subroutine named as "Block_Cut_2D(List<CPolygon>polygons)". All sub-polygons are individual and will not overlap each other.

(4) All the boundary covers (or polygons) found in step 3 will be used to form $E(A, B)$, block_EAB. If both block A and block B are convex, these boundary covers can absolutely be a closed $\partial E(A, B)$ and will not have any unwanted face. Therefore, these boundary covers are saved to define $E(A, B)$, i.e. in polygonList of block_EAB. If one of A and B is concave, a 3D block cutting and boundary searching subroutine named as "Block_Cut_3D(List<Cpolygon>polygons) " will be carried out to form a closed $\partial E(A, B)$.

It can be seen from the flow chart in Figure 21 that the key subroutine is to search the possible boundaries of the entrance block, i.e. Covers_solve(Cblock A, Cblock B). If parallel faces are encountered, a 2D block cutting and loop searching subroutine will be used, i.e. Block_Cut_2D (List<CPolygon>polygons). If concave block is involved, a 3D block cutting and boundary searching subroutine, i.e. Block_Cut_3D(List<Cpolygon> polygons), is required to form $E(A, B)$. The subroutine of Covers_solve has been introduced in Sect. 4. Block_Cut_2D and Block_Cut_3D subroutines will be introduced in Supporting Information, respectively.

7.2 Examples of solvedE(A,B)

Some examples of the solved $E(A, B)$ are shown in Figures 22 and 23. Figure 22 illustrates the entrance block between a concave block A, i.e. the moving shape and a convex block, i.e. the solid and fixed one, which is indicating by the block with black frame. Figure 23 illustrates several examples of 3D entrance block. The left, middle and right column in Figure 23 is block A, block B, and the entrance block $E(A, B)$, respectively.

8 The application of contact computation in a rock fall simulation

A rock fall study is carried out with using the contact searching algorithm reported in this paper. The rock is sliding down and jumping on the fixed slope base (Figure 24). The material is assuming to be rigid and no energy lost is considered in this simulation. The simulation is not accurate but it demonstrates that the proposed contact searching algorithm is effective and applicable to the numerical simulation of 3D discontinuous system.

In this example, the slope is made up of 5000 small triangles. Both convex and concave triangles are used in this simulation. Only the contact covers are used in the contact searching process and without forming the real entrance block. As discussed in Sect. 6, the entrance block is not required in the contact calculation.

In the calculation, only the local contact covers are searched without searching all the global contact covers. The local contact covers are determined by the coordinates of vertexes. For example, in step 2000 (Figure 24(c)), only 18 triangles is required to be checked when searching the local contact covers. Compared with checking all 5000 triangles existed in the slope for obtaining the global contact covers, the computational efficiency of the local contact covers is 278 times higher.

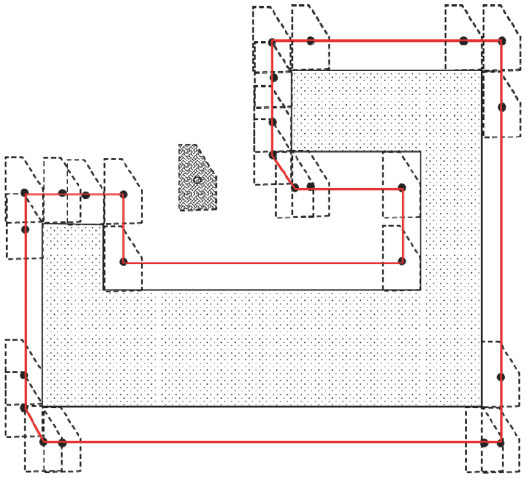


Figure 22 (Color online) $E(A, B)$ between a 2D concave block and a 2D convex block.

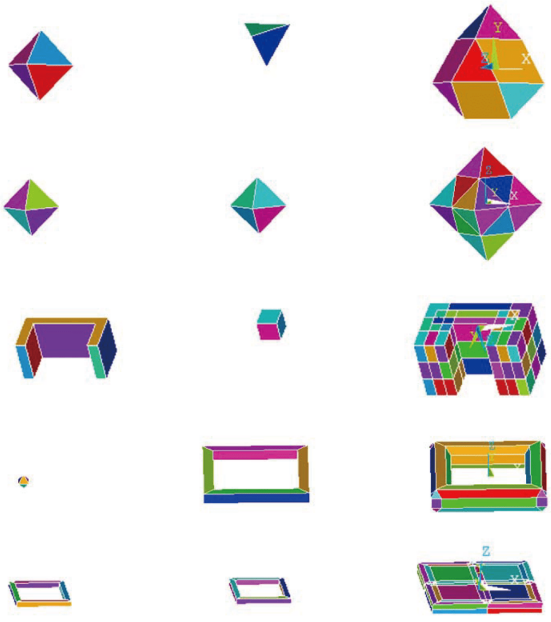


Figure 23 (Color online) $E(A, B)$ between two 3D blocks.

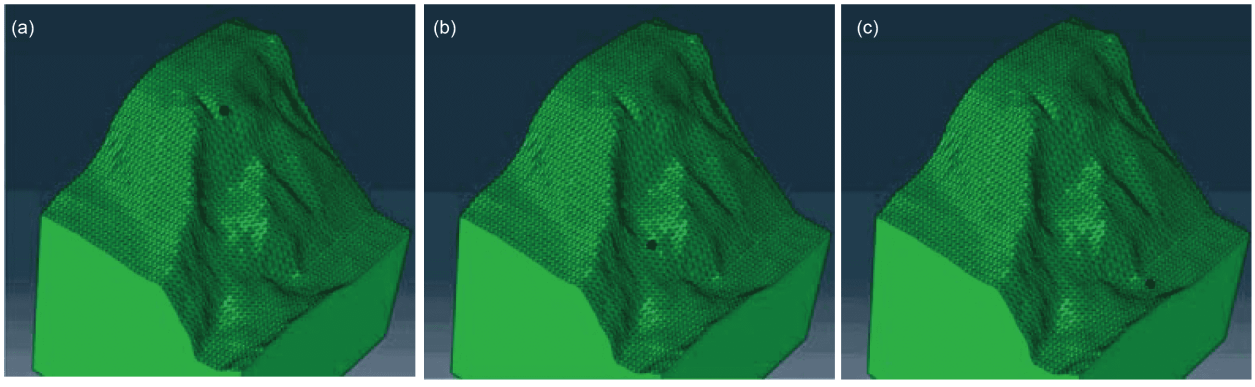


Figure 24 (Color online) Application of contact searching algorithm in a rock fall analysis. (a) Initial state; (b) 1000 step; (c) 2000 step.

9 Conclusions

This paper bridged the gap between the contact theory and the contact computation. The following conclusions can be drawn:

(1) A contact cover searching algorithm is proposed to identify the possible contact covers between two blocks. The algorithm is complete and compact. Following the proposed flow chart and formula, all the contact covers can be selected by checking whether the contact condition is fulfilled or not. In discontinuous analysis, if the distance between the reference point and the contact cover is negative, that contact will be close and has contact force.

(2) The contact time and position can be determined by the closest cover and the reference point. For overlapping contact covers, open-close iteration is required to identify the actual contact mode.

(3) A compact 3D block cutting algorithm is included in this paper and is used to form 3D entrance block when concave blocks are encountered. The entrance block can help to understand the contact theory, but is not in need for contact computation.

(4) The proposed method is applied to analyze the rockfall problem. The results verify that the proposed algorithm is precise and only local contact covers are required in the actual contact computation.

This work was supported by the National Natural Science Foundation of China (Grant Nos. 51479001, 41471052), and the China Institute of Water Resources and Hydropower Research Research & Development Support Program (Grant Nos. GE0145B462017, GE0145B692017). Many thanks are given to Dr. Shi GenHua for his guidance and encourage in this research.

Supporting Information

The supporting information is available online at tech.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

1 Shi G H. Contact theory. Sci China Technol Sci, 2015, 58: 1450–1496
2 Zhong Z H, Nilsson L. A contact searching algorithm for general contact problems. Comput Struct, 1989, 33: 197–209
3 Benson D J, Hallquist J O. A single surface contact algorithm for the post-buckling analysis of shell structures. Comput Methods Appl Mech Eng, 1990, 78: 141–163
4 Belytschko T, Neal M O. Contact-impact by the pinball algorithm with penalty and Lagrangian methods. Int J Numer Meth Engng, 1991, 31: 547–572
5 Bonnet J, Peraire J. An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problem. Int J Numer Methods Eng, 1991, 31: 1–17
6 Williams J R, O'Connor R. A linear complexity intersection algorithm for discrete element simulation of arbitrary geometries. Eng Computations, 1995, 12: 185–201
7 Perkins E, Williams J R. A fast contact detection algorithm insensitive to object sizes. Eng Computations, 2001, 18: 48–62
8 Munjiza A, Andrews K R F. NBS contact detection algorithm for bodies of similar size. Int J Numer Meth Engng, 1998, 43: 131–149
9 Diekmann R, Hungershofer J, Lux M. Efficient contact search for finite element analysis. In: Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering. Barcelona, 2000
10 Wu W, Zhu H, Zhuang X, et al. A Multi-shell cover algorithm for contact detection in the three dimensional discontinuous deformation analysis. Theor Appl Fract Mech, 2014, 72: 136–149
11 Cundall P A. 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 Mecha Mining Sci Geomech, 1988, 25: 107–116
12 Jelenic G, Crisfield M A. Non-linear master-slave relationships for joints in 3D beams with large rotations. Comp Meth Appl Mech Eng, 1996, 135: 211–228
13 Li S, Zhao M, Wang Y, et al. A new numerical method for dem-block and particle model. Int J Rock Mech Min Sci, 2004, 41: 436
14 Chen W S, Zheng H, Cheng Y M, et al. Detection of 3D rock block contacts by penetration edges. Chin J Rock Mech Eng, 2004, 23: 565–571
15 Nezami E G, Hashash Y M A, Zhao D, et al. Shortest link method for contact detection in discrete element method. Int J Numer Anal Meth Geomech, 2006, 30: 783–801
16 Wu J H. New edge-to-edge contact calculating algorithm in three-dimensional discrete numerical analysis. Adv Eng Software, 2008, 39: 15–24
17 Keneti A R, Jafari A, Wu J H. A new algorithm to identify contact patterns between convex blocks for three-dimensional discontinuous deformation analysis. Comput Geotechnics, 2008, 35: 746–759
18 He L. Three Dimensional Numerical Manifold Method and Rock Engineering Applications. Dissertation for Dcotoral Degree. Singapore: Nanyang Technological University, 2010
19 Ahn T Y, Song J J. New contact-definition algorithm using inscribed spheres for 3D discontinuous deformation analysis. Int J Comput Methods, 2011, 08: 171–191
20 An X, Ma G, Cai Y, et al. A new way to treat material discontinuities in the numerical manifold method. Comput Methods Appl Mech Eng, 2011, 200: 3296–3308
21 Konyukhov A, Schweizerhof K. Geometrically exact theory for contact interactions of 1D manifolds. Algorithmic implementation with various finite element models. Comput Methods Appl Mech Eng, 2012, 205-208: 130–138
22 Mousakhani M, Jafari A. A new model of edge-to-edge contact for three dimensional discontinuous deformation analysis. GeoMech GeoEng, 2016, 11: 135–148
23 Nejati M, Paluszny A, Zimmerman R W. A finite element framework for modeling internal frictional contact in three-dimensional fractured media using unstructured tetrahedral meshes. Comput Methods Appl Mech Eng, 2016, 306: 123–150
24 Jiang Q, Chen Y, Zhou C, et al. Kinetic energy dissipation and convergence criterion of discontinuous deformations analysis (DDA) for geotechnical engineering. Rock Mech Rock Eng, 2013, 46: 1443–1460
25 Jiao Y Y, Zhang H Q, Tang H M, et al. Simulating the process of reservoir-impoundment-induced landslide using the extended DDA method. EngGeol, 2014, 182: 37–48
26 Fan H, He S. An angle-based method dealing with vertex-vertex contact in the two-dimensional discontinuous deformation analysis (DDA). Rock Mech Rock Eng, 2015, 48: 2031–2043
27 Zheng H, Zhang P, Du X. Dual form of discontinuous deformation analysis. Comput Methods Appl Mech Eng, 2016, 305: 196–216
28 Sun Y, Feng X, Xiao J, et al. Discontinuous deformation analysis coupling with discontinuous galerkin finite element methods for contact simulations. Math Problems Eng, 2016, 2016: 1–25
29 Wang T, Zhou W, Chen J, et al. Simulation of hydraulic fracturing using particle flow method and application in a coal mine. Int J

CoalGeol, 2014, 121: 1–13
30 Guo Y, Curtis J S. Discrete element method simulations for complex granular flows. Ann Rev Fluid Mech, 2015, 47: 21–46
31 Feng Y T, Han K, Owen D R J. A generic contact detection framework for cylindrical particles in discrete element modelling. Comput Methods Appl Mech Eng, 2017, 315: 632–651
32 Shi G H. Manifold method. In: Proceedings of the First International Forum on Discontinuous Deformation Analysis (DDA) and Simulations of Discontinuous Media. Bekerley, 1996. 52–204
33 Ma G, An X, He L. The numerical manifold method: A review. Int J Comput Methods, 2010, 07: 1–32
34 Zheng H, Liu F, Du X. Complementarity problem arising from static growth of multiple cracks and MLS-based numerical manifold method. Comput Methods Appl Mech Eng, 2016, 295: 150–171
35 Fan H, Zhao J, Zheng H. A high-order three-dimensional numerical manifold method enriched with derivative degrees of freedom. Eng Anal Bound Elem, 2017, 83: 229–241
36 Minkowski H. Volumen and oberflache. Mathematische Annalen, 1903, 57: 447–495
37 Zheng H, Li X. Mixed linear complementarity formulation of discontinuous deformation analysis. Int J Rock Mech Min Sci, 2015, 75: 23–32

Supporting Information

A compact 3D block cutting and contact searching algorithm

*Corresponding author (email: ceXuLi2012@163.com)

LIN XingChao[1,3], LI Xu[2*], WANG XiaoGang[1,3] & WANG YuJie[1,3]

[1]State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, Beijing 100038, China;

[2]Key Laboratory of Urban Underground Engineering of Ministry of Education, Beijing Jiaotong University, Beijing 100044, China;

[3] Department of Geotechnical Engineering, China Institute of Water Resources and Hydropower Research, Beijing 100048, China

Received January 3, 2018; accepted June 28, 2018

1 The 2D block cutting and loop searching subroutine

Once overlapped boundary polygons are found in solving $\partial E(A,B)$, 2D block cutting and loop searching subroutine must be used to search the sub boundary polygons, i.e. the finite covers of entrance boundary. The idea of 2D block cutting and loop searching algorithm is first proposed in Shi (1988) [1]. In this study, the data structure is different with that in Shi (1988) and the new contact theory is used to check the geometry relation among edges. The flow chart of 2D block cutting and loop searching subroutine is illustrated in Figure S1.

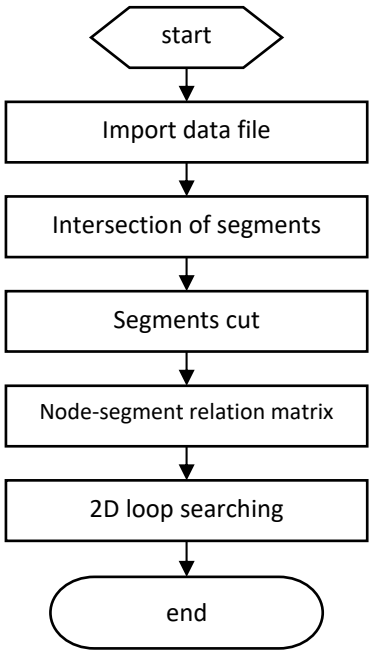


Figure S1 The flow chart of 2D block cutting and loop searching subroutine.

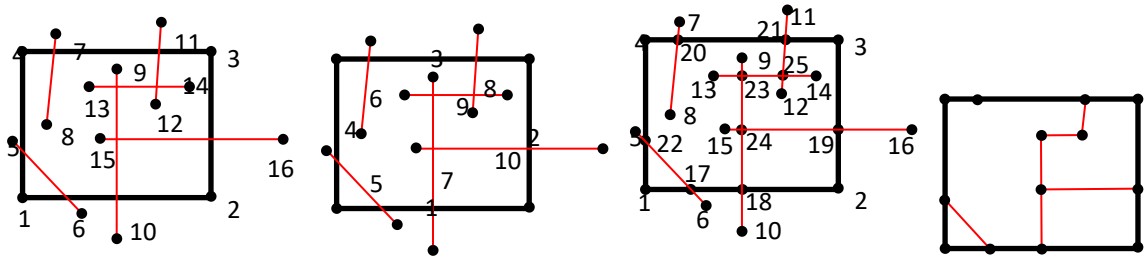
An example is reported here to show the implementation of 2D block cutting and loop searching process, which includes the following steps:

(1) As shown in Figures S2(a) and S2(b), 10 edges with 16 nodes are inputted.

(2) After the intersection of edges (Figure S2(c)), another 9 intersections, i.e. node 17 to 25, are found.

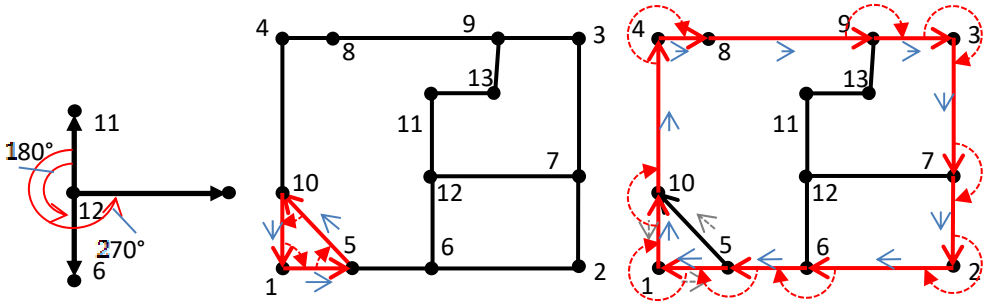
(3) The branches which do not intersect with other edges are collected as cracks and removed from the network for loop searching. The final network for loop searching is shown in Figure S2(d) and only includes 13 nodes. In this step, a node-edge relation matrix is formed as Table S1. In the node-edge relation matrix, the vectors is sorted by anti-clockwise

direction, e.g. node 12 in Figure S2(d) has three sorted vectors, i.e. v1 from 12 to 11, v2 from 12 to 6 and v3 from 12 to 7, as shown in Figure S3(a) and saved in the matrix as Column Node 12 in Table S1.

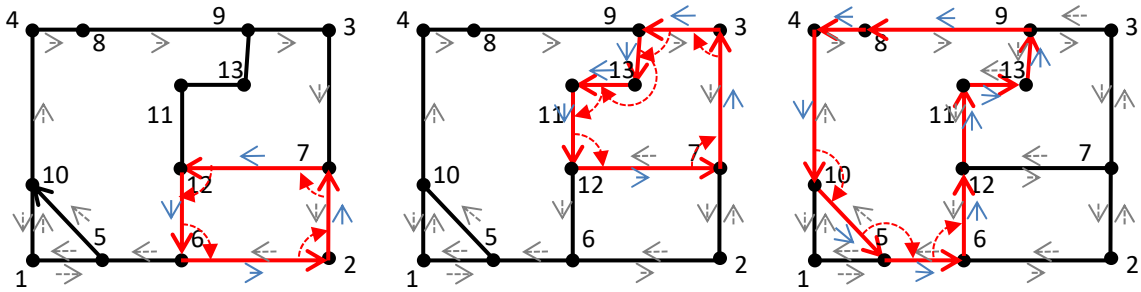


(a) Input nodes (b) Input edges (c) Intersections (d) Final network

Figure S2 Preparing a network for loop searching without any branches. (a) Input nodes; (b) input edges; (c) intersections; (d) final network.



(a) Vectors from node 12 (b) Loop 1 (c) Loop 2



(d) Loop 3 (e) Loop 4 (f) Loop 5

Figure S3 Loop searching process where the gray arrows indicate the traces used and the blue arrows indicate the trace of current loop. (a) Vectors from node 12; (b) loop 1; (c) loop 2; (d) loop 3; (e) loop 4; (f) loop 5.

(4) Based on the node-edge relation matrix, 5 close loops can be found, i.e. as shown in Figures S3(b) to S3(f). For each loop, the path follows the clockwise direction. A simple rule can be used as:

if M(N0, *k*) = N0, the id of next node N1will be M(N0, *k*-1) where M is the node-edge relation matrix; N0 is the No. of last node in loop; *k* is the element id in column N0 of matrix M; and if M(N0, *k*) is the first element in column N0, N1 will be the last element in column N0.

Taking loop 1 as an example, it starts from Node 1. Referring to Table S1, loop 1 goes to Node 5. At Node 5, its coming trace is a vector from Node 5 to Node 1. If the trace vector is rotated following the clockwise direction, it will firstly touch vector from Node 5 to Node 10, which will be used as the next path. Referring to *Column Node 5* in Table S1, Node 10 is just at the left side of Node 1. Thus, loop 1 goes to Node 10. Similarly referring to *Column Node 10* in Table S1, at Node 10, Node 1 is just at the left side of Node 5 which is the last node for Node 10 in loop 1. So loop 1 goes to Node 1 and forms a close loop. After loop 1 is found, the node-edge relation matrix yields to the *line After loop 1* in Table S1. Continuing the upper searching process, five loops can be easily found just based on the node-edge relation matrix. After all the loops have been found, all the node marks in the matrix will be addressed as referring to the final matrix indicated by the last line in Table S1.

Using the upper 2D block cutting and loop searching subroutine, the close boundary loop can be found both for the concave polygon and convex polygon.

(5) After all the loops have been found, solid block will be identified further as follows:

(i) A loop with nodes ordered in anti-clockwise direction means a solid region, remembered as a solid loop, e.g. the loops 1, 3, 4, 5 in Figure S3.

(ii) A loop with nodes ordered in clockwise direction means a hole, remembered as a hole loop.

(iii) If a hole loop is outer of all solid loop, it will be the outer boundary of the solid regions, e.g. the loop 2 in Figure S3.

(iv) If a hole loop is inner of a solid loop, they will form a holed 2D block together. There is no holed block in Figure S3. Two examples which contain holed blocks are illustrated in Figure S4.

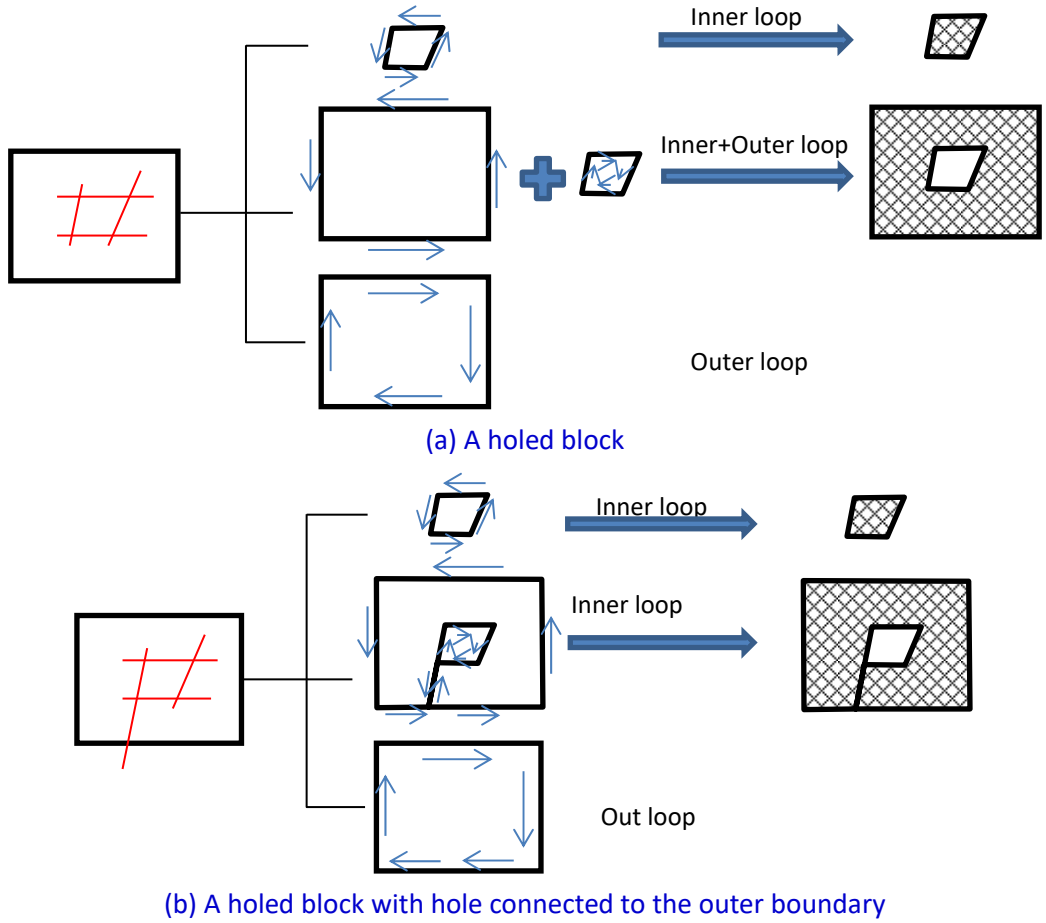


Figure S4 Loop searching for blocks containing holes. (a) A holed block; (b) a holed block with hole connected to the outer boundary.

Table S1 the node-edge relation matrix for the network in Figure S2(d) [a)]

Node NO.	1	2	3	4	5	6	7	8	9	10	11	12	13
Initial state (Figure S2(d))	5,10	6,7	7,9	8,10	1,6,10	5,2,12	2,3,12	9,4	3, 8,13	4,1,5	12,13	11,6,7	11,9
After loop 1 (Figure S3(b))	~~5~~,10	6,7	7,9	8,10	1,6,~~10~~	5,2,12	2,3,12	9,4	3, 8,13	4,~~1~~,5	12,13	11,6,7	11,9
After loop 2 (Figure S3(c))	-5,~~10~~	~~6~~,7	~~7~~,9	~~8~~,10	~~1~~,6,-10	~~5~~,2,12	~~2~~,3,12	~~9~~,4	~~3~~, 8,13	~~4~~,-1 ,5	12,13	11,6,7	11,9
After loop 3 (Figure S3(d))	-5,-10	-6,~~7~~	-7,9	-8,10	-1,6,-10	-5,~~2~~,12	-2,3,~~12~~	~~9~~,4	-3, 8,13	-4,-1,5	12,13	11,~~6~~,7	11,9
After loop 4 (Figure S3(e))	-5,-10	-6,-7	-7,~~9~~	-8,10	-1,6,-10	-5,-2,12	-2,~~3~~,-12	~~9~~,4	-3, 8,~~13~~	-4,-1,5	~~12~~,13	11,-6,~~7~~	~~11~~,9
After loop 5 (Figure S3(f))	-5,-10	-6,-7	-7,-9	-8,~~10~~	-1,~~6~~,-10	-5,-2,~~12~~	-2,-3,-12	-9,~~4~~	-3, ~~8~~,-13	-4,-1,~~5~~	-12,~~13~~	~~11~~,-6,-7	-11,~~9~~

a) - indicates the trace used and – indicates the trace of current loop.

2 The 3D block cutting and boundary searching subroutine

Even some developments have been achieved in the past decades in the block cutting [2,3] or three dimensional discontinuity layout optimization [4], the block cutting algorithm proposed in this study is believed to be of high efficiency and can be used to form 3D block system with complex discontinuities and connectivity.

The 3D block cutting and boundary searching algorithm used in this study has same base with that used by Zhang (2015) [2], i.e. the oriented block definition as that illustrated in Figure 6(f)–(h). The block data structure is listed in Table 1.

If one of A and B. is concave (e.g. one of its dihedral angles is concave), a 3D block cutting and boundary searching subroutine named as "Block_Cut_3D(List<CPolygon> polygons) " will be carried out. Otherwise, only need to store all the possible boundary polygons. They can absolutely be a closed $\partial E(A,B)$ and will not have any unwanted face. The 3D block cutting and boundary searching subroutine not only can be used to solving $\partial E(A,B)$ for concave blocks but also can be used to generate block system based on the discontinuities geometry info, e.g. the joints and faults or rock.

The flow chart of 3D block cutting and boundary searching subroutine is illustrated in Figure S5.

An example is reported here to show the implementation of3D block cutting and boundary searching process, which includes the following steps:

(1) As shown in Table S2, a block with 6 faces and a cutting face is inputted. Totally 7 polygons are inputted and saved as a geometry info Q_0.

(2) After the intersection of polygons (Table S3), total 4 intersected edges are found, i.e. edge from node 13 to node 14, edge from node 14 to node 15, edge from node 15 to node 16 and edge from node 16 to node 13.

(3) Every polygon containing intersected edges will be treated by the 2D block cutting and loop searching subroutine and divided into several sub-polygons. Finally, the geometry info yields to Q_2 where total 13 polygons are found.

(4) Delete the branches.

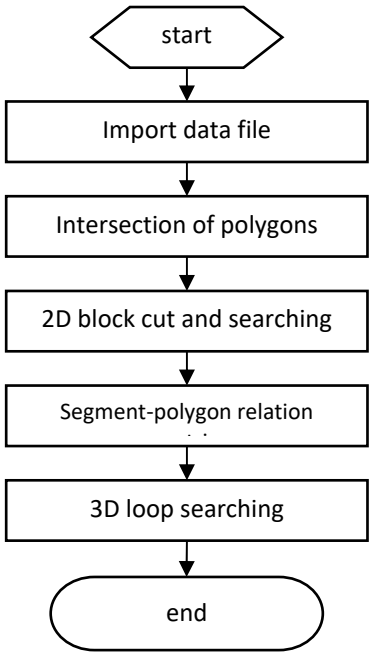


Figure S5 The flow chart of 3D block cutting and boundary searching subroutine.

Table S2 input info, a polygon list Q_0

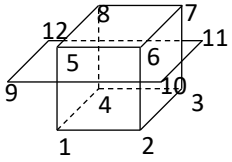
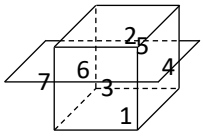
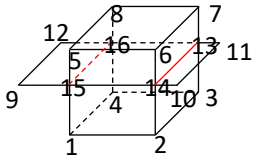
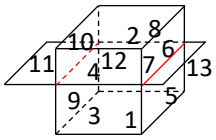
Polygon ID	Vertex Nos.	Geometry info
1	4,3,2,1	 (a)Node number system
2	5,6,7,8	
3	1,2,6,5	
4	2,3,7,6	
5	3,4,8,7	 (b)polygeon number system
6	4,1,5,8	
7	9,10,11,12	

Table S3 Geometry info Q_1 after the intersection of polygons

Polygon ID	Vertex Nos.	Intersected edges	Geometry info
1	4,3,2,1		 (a) Node number system
2	5,6,7,8		
3	1,2,6,5	14,15	
4	2,3,7,6	13,14	
5	3,4,8,7	16,13	 (b) Polygon number system
6	4,1,5,8	15,16	
7	9,10,11,12	(13,14), (14, 15), (15, 16), (16,13)	

For forming a closed block, a boundary polygon of a block must intersect with at least two boundary polygons. If a polygon only intersects with one polygon, it is considered to be a crack polygon and will not serve as the close boundary of a block. Therefore, if an edge only owns to one polygon (Table S4), its owner will intersects no more than one polygon and removed from the block forming process. For example, the rows with bold and red text in Table S4 indicate edges with single owner polygon. These edges and their owner polygons, i.e. $P_{11}(9,15,16,12)$ and $P_{13}(14,10,11,13)$, should be removed from the geometry info matrix Q_2 (Table S5). After deleting these edges and renumbering, the final Geometry info matrix Q will be as that in Tables S6 and S7.

Based on Q_2 (Table S4), a final edge – polygon relation matrix are formed as M_0, as shown in Table S7. If one edge $\bar{e}_r$ owns to k polygons P_k, its owners will be sorted by their inner vectors $\bar{v}_k$ in right-hand screw rule, as illustrated in Figure S6. For each polygon , its inner vector can be calculated as:

$$\bar{v}_k = \bar{n}_k \times \bar{e}_r \qquad \text{(S1)}$$

where $\bar{n}_k$ is the inner normal vector of the polygon P_k (referring to Figure 6(f) and (g)); $\bar{e}_r$ is the vector of the common edge for these polygons. Noticing that, if all P_k have an edge vector of $\bar{e}_r$, a series of dihedral angles can be easily

identified based on M_0, such as dihedral angles (- P_k, P_{k+1}) where $k \in \{1,\cdots,u\}$, $P_{u+1} = P_1$, u is the maximum number of the Ordered Polygon IDs (Table S7) and + means polygon has normal vector pointing inner the dihedral angle and - means polygon has normal vector pointing outer the dihedral angle. Each dihedral angle is defined by two outer boundary polygons and these two boundary polygons are ordered in the right-hand screw rule and pointed outer of the dihedral angle. Clearly, a dihedral angle has two outer boundary polygons with a common edge, but with inverse direction.

For example, for edge 1 in Table S6, i.e. V_1V_2, there are two possible dihedral angles, i.e. $D(P_1,P_3)$ and $D(-P_3,-P_1)$. For edge 20, i.e. $V_{11}V_{12}$, there are three possible dihedral angles, i.e., $D(P_9,P_{11})$, $D(-P_{11},P_{10})$, and $D(-P_{10},-P_9)$. Clearly, all the polygon are addressed twice when forming the dihedral angles, as one time P_k is used and another time $-P_k$ is used.

(5) The block forming process is as follows:

(a) Each polygon have two face and will be treated as two boundary polygons, one is with vertex ordered in anti-clockwise direction and another is with vertex ordered in clockwise direction. For the example in (Tables S8 and S9), because there are 11 polygons in Table S8, together they are 22 boundary polygons to be used in the 3D block searching.

Table S4 Geometry info Q_2 before deleting the branches

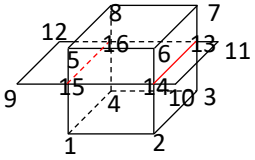
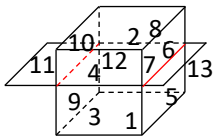
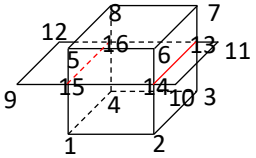
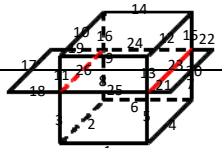
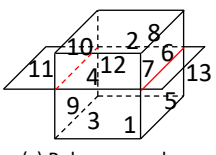
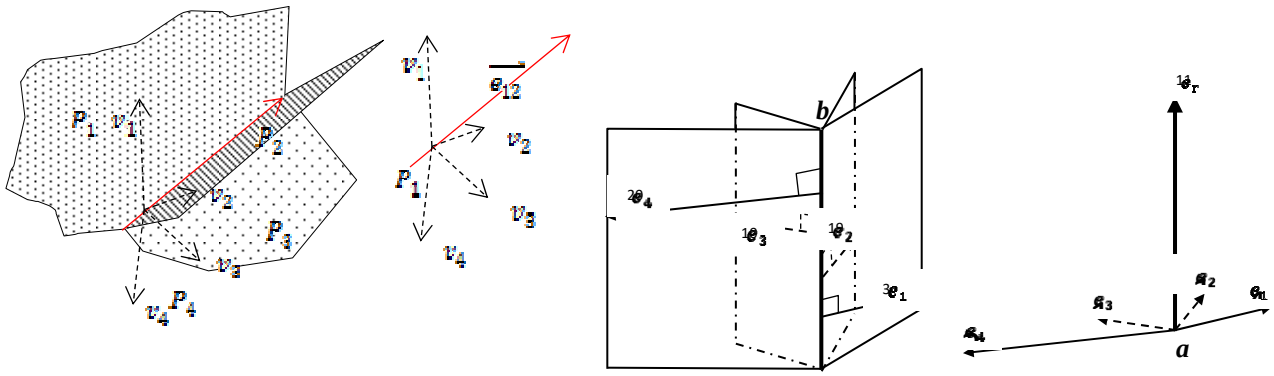
Edge ID	Vertex IDs	Owner polygon IDs	Geometry info
1	1,2	1,3	 (a) Node number system  (b) Polygon number system
2	1,4	1,9	
3	1,15	3,9	
4	2,3	1,5	
5	2,14	3,5	
6	3,4	1,7	
7	3,11	5,7	
8	4,16	7,9	
9	5,6	2,4	
10	5,8	2,10	
11	5,15	4,10	
12	6,7	2,6	
13	6,14	4,6	
14	7,8	2,8	
15	7,13	6,8	
16	8,16	8,10	
17	**9,12**	**11**	
18	**9,15**	**11**	
19	**10,11**	**13**	
20	**10,14**	**13**	
21	**11,13**	**13**	
22	**12,16**	**11**	
23	13,14	5,6,12,13	
24	13, 16	7,8,12	
25	14,15	3,4,12	
26	15,16	9,10,11,12	

Table S5 Geometry info Q_2 after searching sub-polygons

Initial ID	Vertex Nos., V_i	Geometry info
1	4,3,2,1	 (a) Node number system  (b) Line number system 
2	5,6,7,8	
3	1,2,14,15	
4	6,5,15,14	
5	2,3,13,14	
6	7,6,14,13	

7	3,4,16,13
8	8,7,13,16
9	4,1,15,16
10	5,8,16,15
~~11~~	~~9,15,16,12~~
12	15,14,13,16
~~13~~	~~14,10,11,13~~



(a) Polygons with a common edge (b) Inner vectors (c) Right-hand rule

Figure S6 Polygons passing same edge are sorted by the right-hand rule. (a) Polygons with a common edge; (b) Inner vectors; (c) Right-hand rule.

Table S6 final polygon info

Polygon ID	Vertex Nos.
1	4,3,2,1
2	5,6,7,8
3	1,2,10,11
4	6,5,11,10
5	2,3,9,10
6	7,6,10,9
7	3,4,12,9
8	8,7,9,12
9	4,1,11,12
10	5,8,12,11
11	11,10,9,12

An Boolean array MP is used to save the usage of polygons. MP has 22 members. Initially all these polygons are not used and $MP[k] = true$ for $k = 1,2,...,22$.

(b) The necessary and sufficient condition of a successful block searching process is that all polygons must be used twice. Thus, the searching for a new block will be started from polygon P_i with $MP[k] = true$. If $MP[k] = false$, this polygon will be skipped.

An Boolean array ML is used to mark whether the edge has been used in the searching process. At the start of searching a new block, all the edges are not used and $ML[j] = true$ for $j = 1,...,20$.

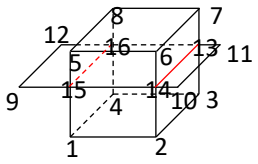
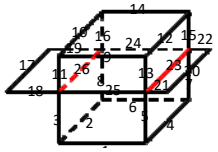
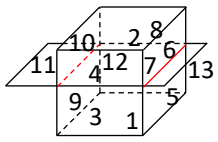
(c) Polygon P_k is used as the outer boundary of the new block to be searched. Each edges of polygon P_k will serve as an edge of the new block and also the edge of a dihedral angle of the new block. These edges and dihedral angles can be easily extracted from M_0 .

For example, the searching for a new block B1 is starts from the first true mark in MP, i.e. $MP[1]$. The mark $MP[1]$ refers to polygon P_1. There are four edges in M_0 (Table S8) are with an owner of polygon P_1, i.e. L_6, L_4, L_1, L_2. If $ML[j] = \text{true}$, L_j will be used to search a dihedral angle. Otherwise, L_j will be skipped in the searching process. For the first searching round, all $ML[j]$ is true and all L_6, L_4, L_1, L_2 should be checked.

For L_6, the dihedral angle with one outer boundary polygon of P_1 is $D(P_1, P_7)$. . There are two rules to be obeyed in the searching process: one is that on the common edge L_6, the edges of P_1 and P_7 are of inverse directions; another is that the turn of the two boundary polygon should follow the right-hand rule (Table S9).

Similarly, dihedral angles $D(P_1, P_5)$, $D(P_1, P_3)$, $D(P_1, P_9)$ can be found for L_4, L_1, L_2 respectively. Because polygons P_1, P_7, P_5, P_3, P_9 are used in this searching round, the usage state of these polygons will be set to be false, i.e., $MP[i] = false$ with $i = 1,3,5,7,9$. At the same time, if the line edge is used in this searching round, its mark, i.e. $ML[j]$ will be set to be false. That's to say, $ML[j] = false$ with $j = 1,2,4,6$.

Table S7 the edge – polygon relation matrix M_0 for block forming

Edge ID, S_i	Vertex IDs., V_i	Ordered Polygon IDs, P_k	Geometry info
1	1,2	1,3	 (a) Node number system  (b) Line number system  (c) Polygon number system
2	1,4	1,9	
3	1,11	3,9	
4	2,3	1,5	
5	2,10	3,5	
6	3,4	1,7	
7	2,9	5,7	
8	4,11	7,9	
9	5,6	2,4	
10	5,8	2,10	
11	5,11	4,10	
12	6,7	2,6	
13	6,10	4,6	
14	7,8	2,8	
15	7,9	6,8	
16	8,12	8,10	
17	9,10	5,6,11	
18	9,12	7,8,11	
19	10,11	3,4,11	
20	11,12	9,10,11	

(d) In step c, there are several polygons addressed in these dihedral angles found (e.g. polygons P_3, P_5, P_7, P_9 in the cell with blue frame in Table S10). Substitute these polygon IDs into step c and check them one by one. The searching process is listed in Table S10. Referring to Table S10, the 2nd, 3rd, 4th and 5th searching round are used to check P_3, P_5, P_7, and P_9, respectively. If an edge in M_0 has been used in former searching process, i.e. it is with a mark $ML[j]$ of false, it will be skipped in the searching process.

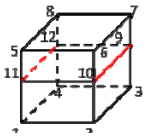
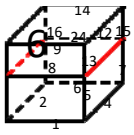
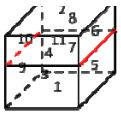
(e) Repeat step d, if all the polygon IDs addressed in the dihedral angles found in former searching process have been checked and no more dihedral angles can be found, the searching process will be stopped and a close block should be formed. Either the polygon used or the dihedral angles found can define the 3D block with a data structure listed in Table 4.

Table S8 final polygon info

Polygon ID, k	Vertex Nos., i	Edge IDs, j
1	4,3,2,1	6,4,1,2
2	5,6,7,8	9,10,11,13
3	1,2,10,11	1,3,5,19

4	6,5,11,10	9,11,13,19
5	2,3,9,10	4,5,7,17
6	7,6,10,9	12,13,15,17
7	3,4,12,9	6,7,8,18
8	8,7,9,12	14,15,16,18
9	4,1,11,12	2,3,8,20
10	5,8,12,11	10,11,16,20
11	11,10,9,12	17,18,19,20

Table S9 the edge – polygon relation matrix M_0 for block forming

Edge ID, j	Vertex IDs., i	Ordered Polygon IDs, k	Geometry info
1	1,2	1,3	
2	1,4	1,9	
3	1,11	3,9	
4	2,3	1,5	
5	2,10	3,5	
6	3,4	1,7	
7	2,9	5,7	
8	4,11	7,9	
9	5,6	2,4	
10	5,8	2,10	
11	5,11	4,10	
12	6,7	2,6	
13	6,10	4,6	
14	7,8	2,8	
15	7,9	6,8	
16	8,12	8,10	
17	9,10	5,6,11	
18	9,12	7,8,11	
19	10,11	3,4,11	
20	11,12	9,10,11	

(f) If there is any $MP[i]$ is true, repeat step b to e for another round of block searching. If all the boundary polygons have been used, i.e. all $MP[i] = \text{false}$. , the block searching work will be completed and all the blocks have been found. In this example, the detailed searching process is listed in Table S10. Total 3 blocks are found. Two of them, i.e. block 1 and block 3, are solid and block 2 is void, i.e. it is the outer boundary of a hole.

(g) After all the solid and void blocks have been found, the relation between these blocks will be checked. This process is similar to that in 2D case. If there is a hole, two overlapped blocks will be found, one is solid with inner normal vectors point to inner of hole and another is void with inner normal vectors point to the outer of hole, as shown in Figure S8.

Table S10 The searching process for blocks in Figure 15[a)]

Block ID	Check round	outer Polygon ID	dihedral angles
1 (Figure S7(a))	1	1+	(1+,3+), (1+,9+), (1+,5+), (1+,7+)
	2	3+	(3+, 9+), (3+, 5+), (3+,11+)
	3	5+	(5+,7+), (5+,11+)
	4	7+	(7+,9+), (7+,11+)
	5	9+	(9+,11+)
	6	11+	Non
2 (Figure S7(b))	1	1-	(3-,1-), (9-,1-), (5-,1-), (7-,1-)
	2	3-	(9-,3-), (5-,3-), (4-,3-)
	3	5-	(7-,5-), (6-,5-)
	4	7-	(9-,7-), (8-,7-)
	5	9-	(10-, 9-)
	6	4-	(2-, 4-)
	7	6-	(2-, 6-)
	8	8-	(2-, 8-)
	9	10-	(2-, 10-)

	10	2-	Non
3 (Figure S7(c))	1	2+	(4+, 2+), (6+, 2+), (8+, 2+), (10+, 2+)
	2	4+	(6+, 4+), (4+, 10+), (11-, 4+)
	3	6+	(11-, 6+), (8+, 6+)
	4	8+	(10+, 8+), (8+, 11-)
	5	10+	(10+, 11-)
	6	11-	Non

a) + means polygon with vertex ordered in the turn listed in Table S9, and - means polygon with vertex ordered in the inverse turn listed in Table S9.

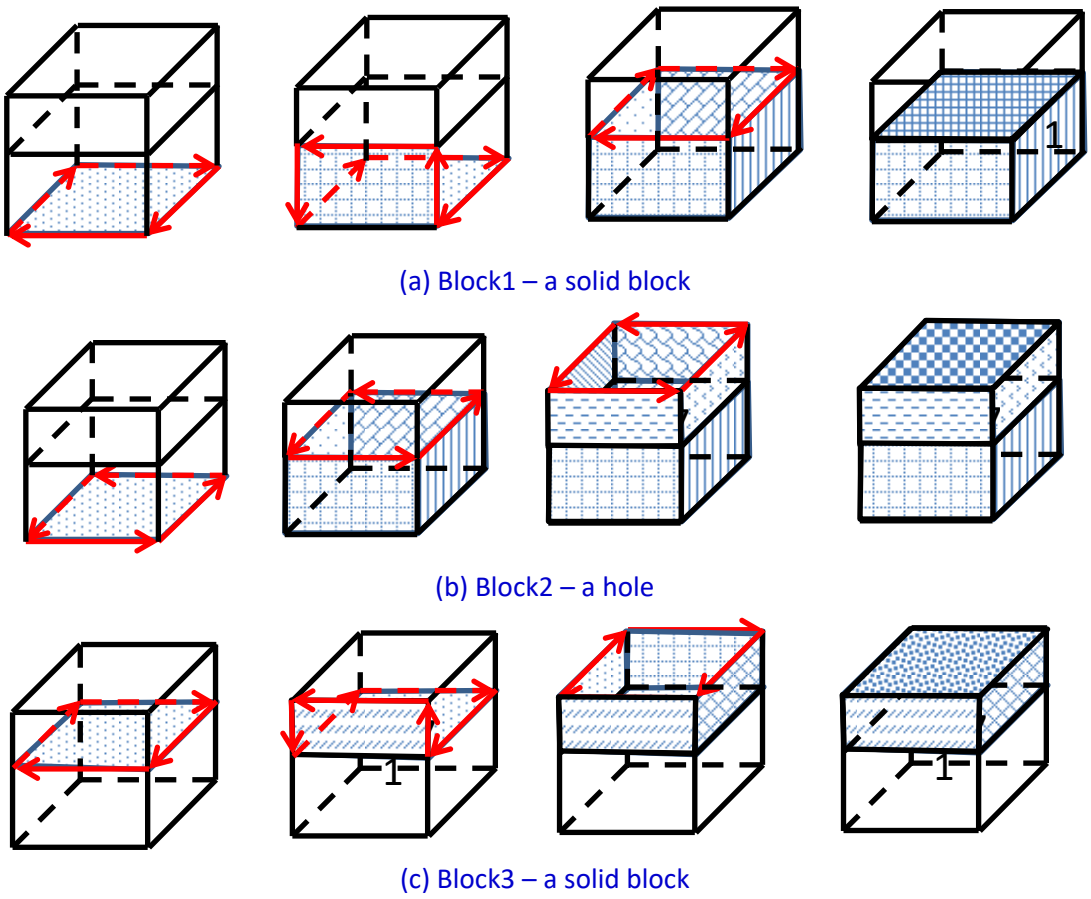
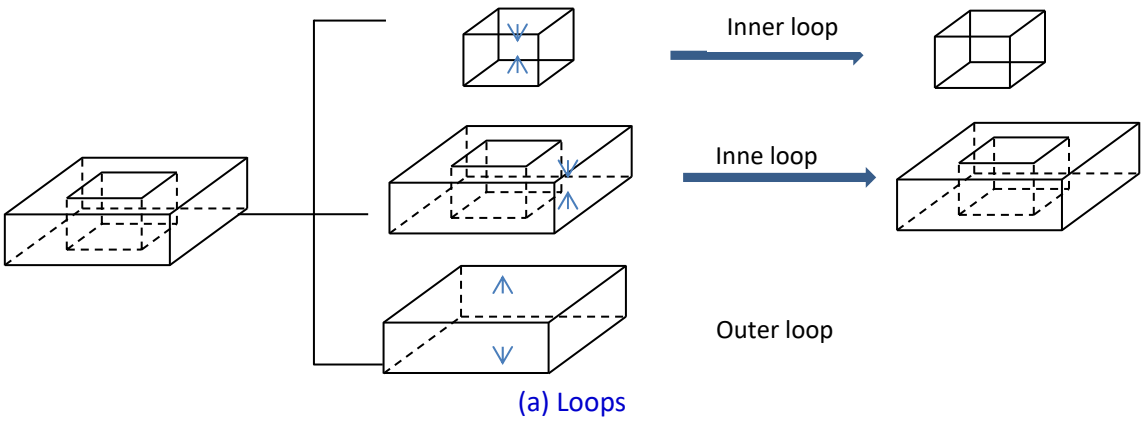


Figure S7 An example of block searching process. (a) Block1 – a solid block; (b) block2 – a hole; (c) block3 – a solid block.



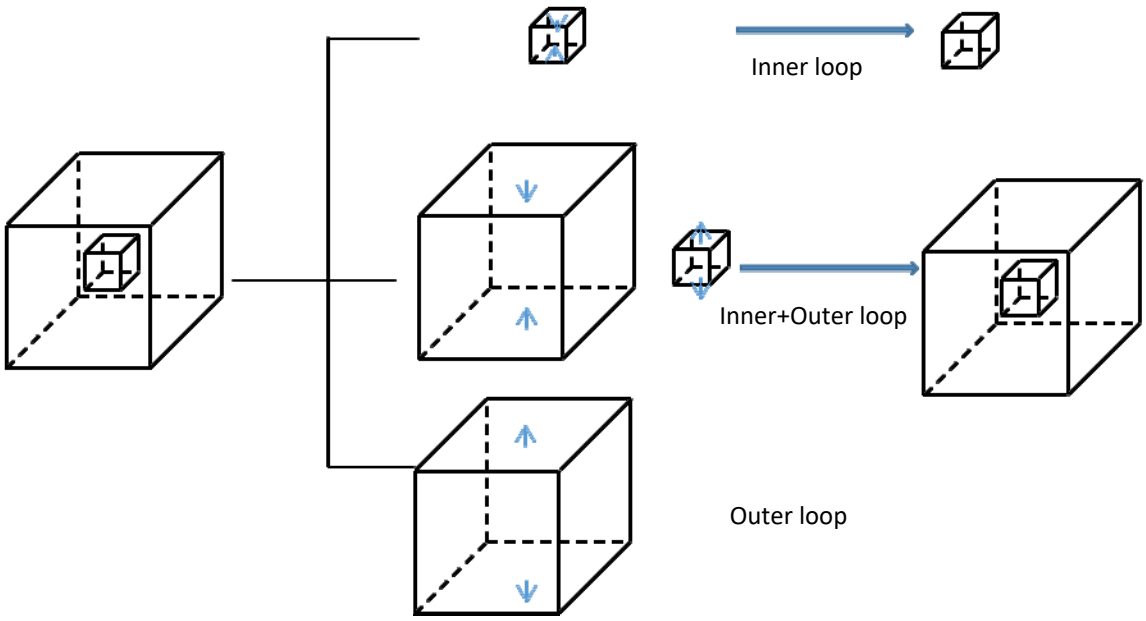


Figure S8 Some special cases of 3D block forming. (a) Loops; (b) a holed block.

Reference

[1] Shi G H. Discontinuous Deformation Analysis—A New Numerical Model for the Statics and Dynamics of Block Systems. Dissertation for Dcotoral Degree. Berkeley: University of California, 1988

[2] Zhang Q H. Advances in three-dimensional block cutting analysis and its applications. Computers and Geotechnics, 2015 : 26-32

[3] Yang SK, Ren XH, Zhang JX. Study on cutting of 3D complex blocks with finite discontinuities. Rock and Soil Mechanics, 2016, 37: 2206-2212

[4] Zhang Y. Multi-slicing strategy for the three-dimensional discontinuity layout optimization (3D DLO). International journal for numerical and analytical methods in geomechanics, 2017, 41: 488-507