A NEW ROCK SLICING METHOD BASED ON LINEAR PROGRAMMING

C.W. Boon [a], G.T.Houlsby [a], S. Utili [b]

[a] University of Oxford, Department of Engineering Science; Parks Road, Oxford OX1 3PJ, United Kingdom

[b] University of Warwick, School of Engineering; Coventry CV4 7AL, United Kingdom; formerly at University of Oxford.

Email: s.utili@warwick.ac.uk

Abstract

One of the important pre-processing stages in the analysis of jointed rock masses is the identification of rock blocks from discontinuities in the field. In 3D, the identification of polyhedral blocks usually involve tedious housekeeping algorithms, because one needs to establish their vertices, edges and faces, together with a hierarchical data structure: edges by pairs of vertices, faces by bounding edges, polyhedron by bounding faces.

In this paper, we present a novel rock slicing method, based on the subdivision approach and linear programming optimisation, which requires only a single level of data structure rather than the current 2 or 3 levels presented in the literature. This method exploits the novel mathematical framework for contact detection introduced in Boon et al. (2012). In the proposed method, it is not necessary to calculate the intersections between a discontinuity and the block faces, because information on the block vertices and edges is not needed. The use of a simpler data structure presents obvious advantages in terms of code development, robustness and ease of maintenance. Non-persistent joints are also introduced in a novel way within the framework of linear programming. Advantages and disadvantages of the proposed modelling of non-persistent joints are discussed in this paper. Concave blocks are generated

using established methods in the sequential subdivision approach, i.e. through fictitious joints.

Highlights

- we present a novel rock slicing method based on linear programming optimisation
- it requires only a single level of data structure rather than the current 2 or 3
- calculation of intersections between a discontinuity and block faces is not needed
- the method has obvious advantages for code development, robustness, maintenance
- Non-persistent joints are introduced in a novel way via linear programming

Keywords: Rock slicing; block generation; distinct element method; sequential subdivision; rock mechanics; linear programming

---

1. Introduction

Jointed rock masses are made up from numerous polyhedral rock blocks, whose faces are cut out by discontinuities in the rock field. The spatial distribution, size and orientation of these discontinuities are rarely regular and usually follow probabilistic distributions. As a result, the size and shape of each block in the jointed rock mass are different. For the purpose of distinct element modelling (DEM) or discontinuous deformation analysis (DDA), one has to invest significant effort to identify polyhedral blocks from the discontinuities (see Fig. 1), whose orientations are typically defined using their dip directions and dip angles (see Fig. 2).

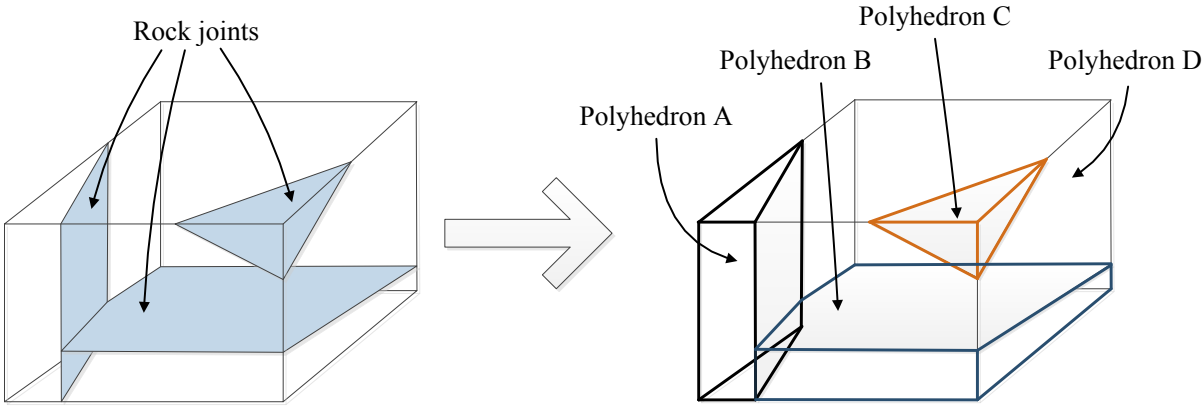


Fig. 1 Illustration of a simple set of rock slices, resulting in polyhedral rock blocks

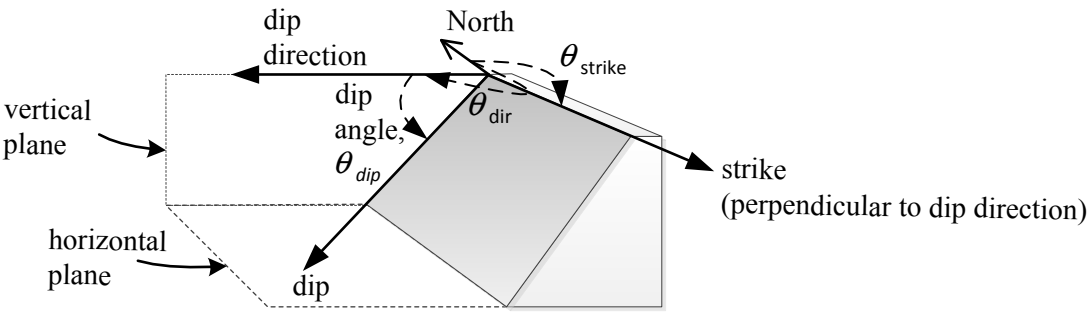


Fig. 2 Definition of strike, dip and dip direction according to Hoek et al. (1995)

Broadly, there exist two approaches in block generation algorithms. The first approach is based on subdivision, in which discontinuities are introduced sequentially (Warburton, 1985;

Heliot, 1988, Yu et al. 2009; Zhang & Lei, 2012). Each discontinuity is introduced one-at-a-time (see Fig. 3 (a)). If a discontinuity intersects a block, the parent block is subdivided into a pair of so-called child blocks. This process is repeated until all the discontinuities are introduced. The number of blocks increases as more "slices" are introduced, and a data structure of every block is maintained throughout the slicing process. The blocks generated through sequential subdivision are convex because a discontinuity has to terminate at the face of a neighbouring block. Concave blocks can, nonetheless, be generated through the use of clustering, which can be automated (Yu et al., 2009) or guided by specifying fictitious construction joints (Warburton, 1985; **Fig. 4**). Blocks subdivided by a construction joint are clustered together by imposing a kinematic constraint which prevents any relative movement between the two sides of the joint. Likewise, non-persistent joints, i.e. joints of finite sizes (Einstein et al., 1983; Zhang & Einstein, 2010), can be modelled through clustering, specifying fictitious construction joints, or subdomains (Heliot, 1988; see **Fig. 5**). This is discussed again in further detail in a later paragraph. On the other hand, in the second approach ('face-tracing' based on simplicial homology theory), discontinuities are introduced all-at-once (see Fig. 3 (b)). All the vertices and edges in the domain are first calculated from the intersections between the discontinuities. From these vertices and edges, there are ways by which the faces and polyhedra in the rock mass can be identified (Lin, Fairhurst & Starfield, 1987; Ikegawa & Hudson, 1992; Jing, 2000; Lu, 2002). The necessary algorithms are, however, rather complex. The advantage of this approach is that convex and concave blocks are identified in the same manner. Non-persistent joints and dangling joints (see **Fig. 6**), i.e. joints which terminate inside intact rock without contributing to block formation (Jing, 2000), are also treated in the same manner as persistent joints, i.e. joints of infinite size. Depending on the type of mechanical analysis which is to be performed on the generated rock mass, these dangling joints may have to be removed; for instance, they have to be removed if

either the distinct element method (Cundall, 1988; Itasca, 2006; Itasca, 2007) or discontinuous deformation analysis (Shi & Goodman, 1985) is used later on for analysis; but they do not need to be removed if fracturing has to be modelled, for instance employing the discrete-finite element method (Pine et al., 2006). A summary of the two approaches is shown in Table 1.

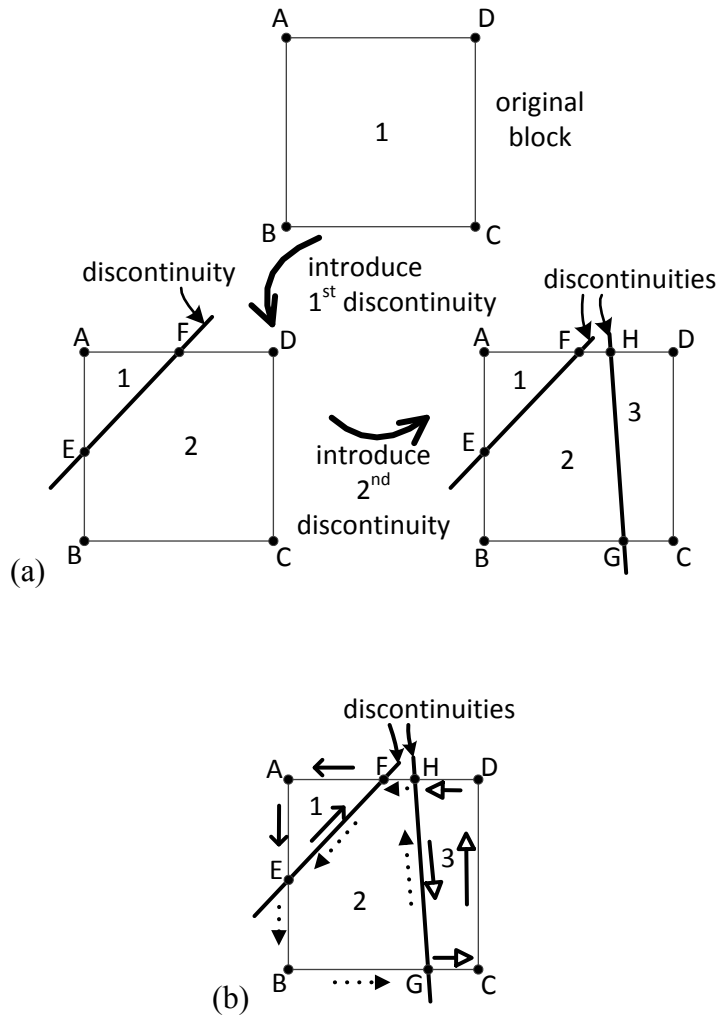


Fig. 3 Block generation algorithm based on (a) sequential subdivision, and (b) all-at once (face-tracing)

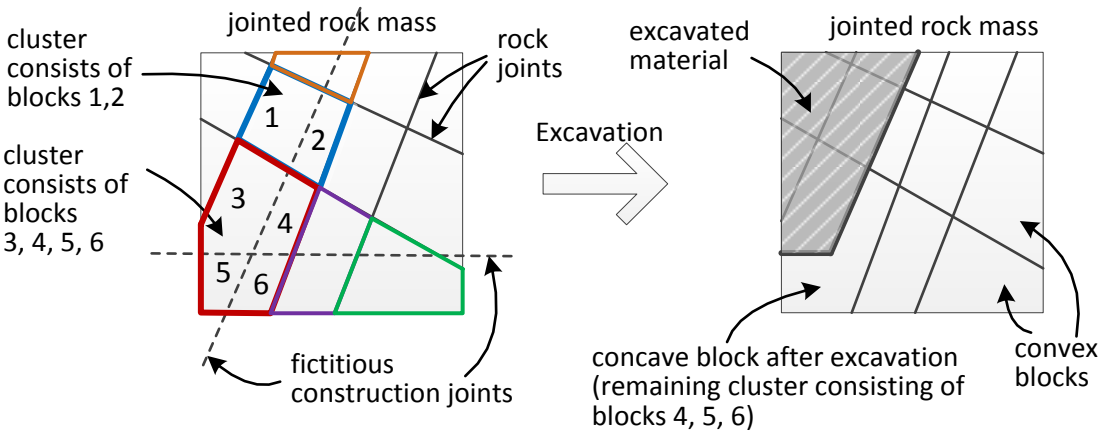


Fig. 4. Illustration of fictitious joints, through which concave blocks are created

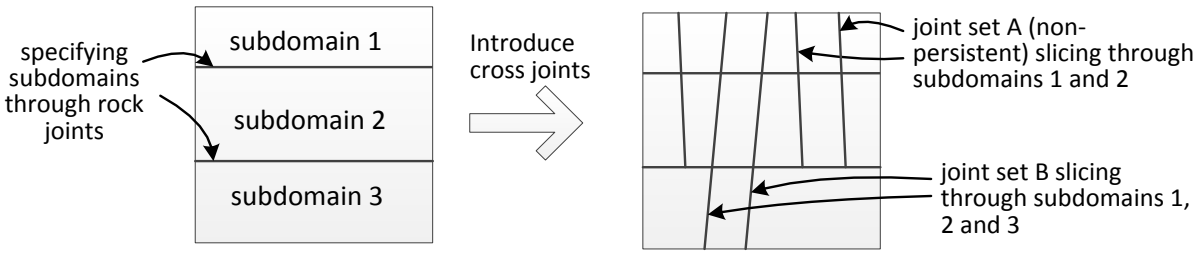


Fig. 5. Use of subdomains to create non-persistent joints

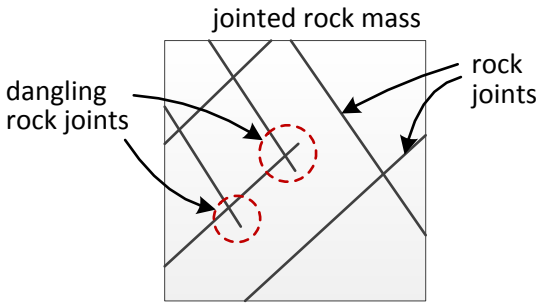


Fig. 6 Illustration of dangling joints in 2-D, terminating inside intact rock

Table 1: Differences between all-at once ('face-tracing') and sequential subdivision algorithms for block generation

Features in generated rock blocks	**All-at-once/ 'face-tracing'**	**Sequential subdivision**
Concave blocks	Treated in the core tracing algorithm	Requires ad-hoc algorithms for clustering, which can be automated.
Non-persistent joints	Treated in the core tracing algorithm	Requires ad-hoc algorithms which can be automated.
Dangling joints which terminate inside intact rock	Treated in the core tracing algorithm	Requires very prescriptive ad-hoc algorithms
Bookkeeping and management of data structures	Complex and requires very careful bookkeeping procedures	Complexity decreases with the level of data structures
Risk of error propagation due to incompatible data structures	High. Rounding errors are prone to occur during tracing of vertices, and special attention has to be invested to avoid incompatible data structures	The simpler the data structure, the simpler the algorithms are.
Suitable applications	Can be used in discontinuum analysis, e.g. DEM or DDA. Dangling joints are removed, before generating blocks. Can be used in coupled numerical codes to model problems involving fracture propagation and fluid-flow in the fracture network, where dangling joints has to be modelled explicitly and correctly	Can be used in discontinuum analysis, e.g. DEM or DDA. Widely used in 3-DEC (Itasca, 2007). Less suitable to model applications where information of dangling joints are important.

This paper is about the sequential subdivision approach. In the case of a complex 3-D jointed rock mass, the generation of polyhedral blocks requires tedious and algorithmically

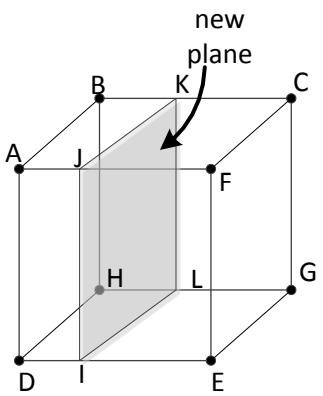
complex updates of the data structure which is used to encapsulate the significant geometrical features of the mass. The number of faces, edges and vertices of the polyhedra in the jointed rock mass is unknown to the modeller, and they become known only at the end of the rock slicing procedure. Therefore, during block generation, the management of this triple-level data structure (faces, edges and vertices) requires careful implementation in a numerical code. Since computing resources, e.g. computing time and memory, is rarely a major concern in rock slicing algorithms by comparison to the simulation runtime of the physical problem considered (e.g. underground excavations, stability analysis of rock slopes, etc.), the choice of code implementation is dictated by factors such as the time needed for code development, ease of code maintenance, and robustness. Algorithms based on the subdivision approach are mainly concerned about the updating of the data structure every time a block is subdivided. A triple-level and a double-level hierarchical data structure have been proposed by Warburton (1985) and Heliot (1988) respectively for their rock slicing algorithms (see Fig. 7). In Warburton (1985), the flow of the algorithm proceeds as follows: (i) intersections (new vertices) are identified and old edges are subdivided, (ii) new edges are identified from the old faces which cross the joint plane and also from their edges which cross the joint plane (not every pair of new vertices can form a new edge), (iii) faces and other data structure for the child blocks are updated (see Fig. 7 (a)). Most of the algorithms proposed recently (e.g. Yu et al. 2009) make use of the data structure proposed by Heliot (1988). In Heliot (1988), every face of a polyhedron is indexed, and a vertex is assumed to result from the intersection of three planes (see Fig. 7 (b)). Each vertex therefore consists of three indices. An intersection check is performed for every pair of vertices which have two indices in common (e.g. between vertex-146 and vertex-346). New vertices are created from the intersection, and their indices are identified. Old vertices are allocated to the new child blocks depending

on whether they are on the positive or negative halfspace. The lists of faces and vertices are rebuilt for each child blocks.

The level of housekeeping (or bookkeeping) algorithms, which is required in a block generation computer code, depends on the choice of data structures. Heliot (1988) has, for instance, made bookkeeping more manageable by reducing the original three-level data structure (Warburton, 1985) to a two-level data structure consisting of only vertices and faces (see Fig. 7). In the rock slicing method presented in this paper, only a single data structure consisting of the block faces is used. It will be shown that this novel procedure makes block generation algorithmically simpler and numerically more robust. Whilst it is necessary to establish whether there is intersection between a block and a discontinuity, the exact intersections between the discontinuity and the block faces need not be calculated in our method. In other words, information on block vertices and edges are not necessary, so there is no longer the need to maintain a complex hierarchical data structure, and problems arising from rounding errors in the case of high vertex density can be avoided (c.f. Elmouttie et al., 2010). According to the proposed novel mathematical treatment based on convex optimisation, the block faces of a polyhedron are defined by linear inequalities, the equation of a joint plane is defined by a linear equality constraint, and the geometrical boundary of a non-persistent joint by linear inequalities. Given a non-persistent joint and a polyhedron which are potentially intersecting, we establish whether there is actual intersection by checking if the optimisation problem defined by the ~~aforementioned~~ linear equality constraint for the joint plane, the inequality constraints for the geometrical boundary of the non-persistent joint, and the inequality constraints for the polyhedron is feasible (i.e. whether the convex set is not empty). The problem is feasible if there is a point lying inside the interior region defined by the linear inequalities and at the same time satisfying the linear equality constraint (Boyd & Vandenberghe, 2004), and not feasible if otherwise. To ascertain the

existence of such a point, i.e. whether the problem is feasible, a linear program is run (illustrated in section 2.3) to find the point with the largest negative distance from all the inequalities, i.e. maximum infeasibility, whilst satisfying the equality constraint. The one-level data structure, consisting of information on the block faces only, can be used in a DEM code employing the new contact detection algorithm recently proposed by Boon et al. (2012), which also is based on linear programming and the concept of analytic centre. In that paper, it is shown that using well-established convex optimisation procedures (Boyd & Vandenberghe, 2004), intersection between a pair of polyhedra defined in terms of their faces only can be established and the contact point between them can be calculated (Boon et al., 2012). Information on polyhedral vertices and edges are unnecessary, because the contact detection algorithm requires only knowledge of the linear inequalities defining the block faces. It is useful also to highlight that the rock slicing procedure proposed here can be employed in other applications too, which use more general non-spherical convex solids partially defined by linear inequalities but with neither vertices nor edges (Houlsby, 2009; Harkness, 2010; Boon et al., 2013).

Depending on the type of discontinuous analysis conducted after block identification, the vertices and edges for each block may be required, for instance in the case of the contact detection algorithms employed in the earlier formulations of 3D DEM analyses for polyhedral particles (Ghaboussi & Barbosa, 1990; Cundall, 1988). However, note that also in this case, the algorithmic operations required for rock mass slicing become simpler because once the subdivision of blocks is completed and the faces of each block have been identified, the remaining calculations consist solely of finding and assembling the vertices and edges of each individual block.



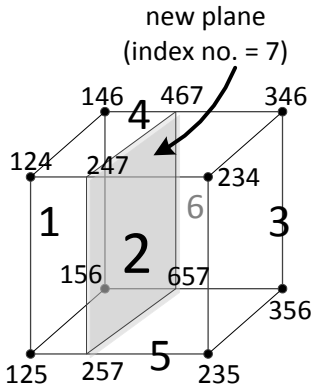
Triple-level data structure under the polyhedron (face-edge-vertex)
Subdivision algorithm is based on the classification of existing data structures:
Possible outcomes for a vertex:
- lies on the positive halfspace of the plane
- lies on the negative halfspace of the plane
- just touches the plane

Possible outcomes for an edge (line in 2-D):
- both ends lie entirely on the positive halfspace of the plane
- both ends lie entirely on the negative halfspace of the plane
- crosses the plane (one end is on the positive halfspace and the other on the negative halfspace)
- one end just touches the plane and the other is on the positive halfspace
- one end just touches the plane and the other is on the negative halfspace
- both ends touch the plane (the edge is fully contained in the plane)

Possible outcomes for a face:
- crosses the plane
- just touches the plane on an edge
- just touches the plane on a vertex

General flow of algorithm:
- Identify intersections and update old edges (e.g. BC subdivided into BK and KC)
- Introduce new edges (e.g. JK but not JL) by identifying (i) old faces which crosses the plane and (ii) their edges which crosses the plane
- Identify faces of child blocks through the above classifications

(a)



Double-level data structure under the polyhedron (vertex-face)
Every face of a polyhedron is indexed. A vertex is formed by intersections of three planes and consists of three indices. If a pair of vertices has two indices in common, they belong to an edge of the polyhedron.
Gist of algorithm:
- Allocate vertices to the positive or negative halfspaces of the new plane
- Calculate intersections between vertices of different half spaces. An intersection is calculated between a pair of vertices if they have two indices in common; e.g. an intersection is calculated between 146 and 346 but not between 146 and 234.
- Allocate old vertices to the child blocks based on which halfspace of the new plane they occupy. The new vertices are duplicated and allocated to each child block.

(b)

Fig. 7. Rock slicing algorithm derived from (a) three-level (Warburton, 1985) and (b) two-level (Heliot, 1989) data structures

Several numerical techniques have been proposed in the literature to model non-persistent joints in the sequential subdivision approach. Heliot (1988) divided the domain into finite subdomains so that non-persistent joints are made to terminate against the boundaries of the introduced subdomains (see Fig. 5). This method is used, for instance, by

the commercial code 3DEC, and it can be inferred that at least one joint set has to be persistent (cf. Kim et al., 2007). In another method (Zhang et al., 2012), the subdivision operations are carried out on a gridded mesh so that all the finite joints are made to terminate against the boundaries of the mesh elements. In Yu et al (2009), the subdivided blocks at the end of block generation are checked against the actual extents of the joints: if a pair of blocks is not completely sliced, they are clustered together. In our algorithm proposed here, non-persistent joints are introduced during the subdivision stage by introducing additional constraints into the linear programming optimisation to prevent the subdivision of blocks which are not intersected by finite joints (see Section 2.3). This technique exploits the fact that a joint always terminates at some joints introduced earlier in the subdivision procedure. Hence, it is not necessary to have one or more of the joint sets to be persistent as required by 3DEC (Itasca, 2007) (cf. Kim et al., 2007). The proposed method is also simpler than the methods used in Yu et al. (2009) and Zhang et al. (2012), since we do not need to employ specific algorithms to combine element blocks. This technique for introducing non-persistent joints gives rise to a rock mass geometry with the number of generated blocks falling between the first extreme case in which all the rock joints are persistent and the other extreme case in which all the dangling joints are removed. This approach of treating non-persistent joints is comparable to the latest release of the commercial DEM code, 3DEC (Itasca, 2013), i.e. only blocks which touch the non-persistent joints are subdivided. Although the mathematical treatment and the algorithmic implementation details of the latest 3DEC release are not in the public domain, however, on the basis of the information available on the current and previous releases, the algorithms employed are based on conventional data structures, *i.e.* vertices and edges.

It is important to distinguish the types of algorithms or subroutines within the sequential subdivision framework. The algorithms could be categorised into core algorithms,

which are associated with the management and updating of data structures, and ad-hoc subroutines, which aim to replicate the jointed rock mass structure. The core algorithm that manages the update of the rock data structure during subdivision form the backbone of a block generation computer program. Core algorithms are mutually exclusive to each other, *i.e.* only one type of core algorithm can be used during subdivision. This implies that only one type of data structure has to be used during block generation, i.e. either 3 level or 2 level or 1 level. However, at the end of the entire subdivision process, it is possible to switch to a different data structure (e.g. 3 level or 2 level) by determining the missing data for each block. This task is easy at this stage, because the blocks have already been identified. On the other hand, ad-hoc subroutines can be implemented in any of the core algorithms, and can be used together with other ad-hoc subroutines in the literature. A summary of the core algorithms and ad-hoc subroutines is given in Table 2. The contribution of this paper is on the core algorithm. The implementation of some of the ad-hoc subroutines is also illustrated in this paper, showing how they could be adopted for the novel data structure proposed here. The main mathematical function required in most of the ad-hoc subroutines is to identify whether a block is touching a plane, of which the method is established in the core algorithm.

In summary, the main advantages of the proposed algorithm is that it uses a simpler data structure based on one level only, making code implementation and maintenance significantly easier. This algorithm is also more robust since the new data structure is less sensitive to rounding errors. Note that after all the subdivisions from all the rock joints are completed, it is possible to convert the data structure to another one if this is required by the numerical code employed to carry out the calculations for the discontinuum analysis (DEM or DDA).

Table 2: Core algorithms and ad-hoc subroutines in the sequential subdivision method

Core algorithms related to bookkeeping and the	**Known ad-hoc subroutines which can be added to any of the core algorithms.**

management of data structures.	
Triple data structure algorithm consisting of vertices, edges and faces (Warburton, 1985)	Concave blocks – clustering/ clumping (Warburton, 1985)
Double data structure algorithm consisting of vertices and faces (Heliot, 1988)	Non-persistent joints – Introducing subdomains, so that non-persistent joints slice through the designated subdomains only (Heliot, 1988)
Single data structure algorithm consisting of faces (currently proposed)	Non-persistent joints (including dangling joints) – Specifying fictitious joints to demarcate the boundaries of non-persistent joints (Wang, 1992; Kulatilake et al., 1992).
Conversion of data structures is possible after all the blocks are identified	Non-persistent joints – Probability of slicing a rock block to model the persistence of a joint set (Itasca, 2007)
	Non-persistent joints – Clustering, or assigning different contact properties, through checking the subdivided blocks against actual fracture extents (Yu et al., 2009; Fu et al., 2010; Zhang & Lei, 2013; Itasca, 2013).
	Non-persistent joints – Slice only blocks which touch the boundaries of non-persistent joints (Zhang et al., 2012; Itasca, 2013). Joint extents can be better controlled by introducing a few fictitious joints or a gridded mesh at the beginning (Zhang et al., 2010; Zhang et al., 2012; Itasca 2013).
	Bounding objects to increase efficiency – To establish intersection between non-persistent rock joints and rock blocks (Yu et al., 2009; Zhang & Lei, 2013).
	Termination criterion for slicing – To achieve a prescribed fracture intensity in terms of the block edges (2-D view)

2. The method

The proposed method generates convex blocks. Concave blocks can nevertheless be generated through clustering two or more convex blocks. This is discussed later in this section. In rock mechanics, the orientation of a joint is described in terms of dip direction, θ_{dir} , and dip angle, θ_{dip} of the joint plane (see Fig. 2). However, for ease of coding, in the implemented algorithm the normal vector of the joint plane is used. The relationship of the normal vector to the aforementioned angles is as follows. Define $\mathbf{N}_{\text{plane}}$ as the plane normal vector and d the distance of the plane from the global origin. Define two auxiliary angles, α:

$$\alpha = \pi/2 - \theta_{\text{dip}} \tag{1}$$

and β:

$$\left.\begin{aligned} \beta &= \theta_{\text{dir}} + \pi \quad \text{for} \quad 0 \le \theta_{\text{dir}} < \pi \\ \beta &= \theta_{\text{dir}} - \pi \quad \text{for} \quad \pi \le \theta_{\text{dir}} < 2\pi \end{aligned}\right\} \tag{2}$$

from which the unit vector $\mathbf{N}_{\text{plane}}$ can be calculated as:

$$\mathbf{N}_{\text{plane}} = (\cos\beta\cos\alpha,\ \sin\beta\cos\alpha,\ \sin\alpha) \tag{3}$$

The distance, *d*, of the plane from the global origin can be calculated if any point lying in the plane, $\mathbf{x}_0$, is known:

$$d = \mathbf{N}_{\text{plane}}^{\text{T}}\mathbf{x}_0 \tag{4}$$

The derivation of this normal vector follows the sign convention proposed by Priest (1983) in which +**x** points North, +**y** points East and +**z** downwards.

2.1 Defining the polyhedron

Rather than defining a polyhedron using the conventional vertex-edge-face data structure (Warburton, 1985), we specify the space occupied by a convex polyhedron solely using a set of linear inequalities (also known as half-spaces) forming the faces. For a block consisting of *N* planes, the space that it occupies can be defined using the following inequalities (see Fig. 8):

$$a_i x + b_i y + c_i z \le d_i, \quad i = 1,\ldots,N, \tag{5}$$

where (a_i, b_i, c_i) is the normal vector of the ith plane, and d_i is the distance of the plane to the (local) origin. For brevity, we use the vector notation:

$$\mathbf{a}_i^{\text{T}}\mathbf{x} \le d_i, \quad i = 1,\ldots,N, \tag{6}$$

where **a** and **x** are 3 × 1 vectors.

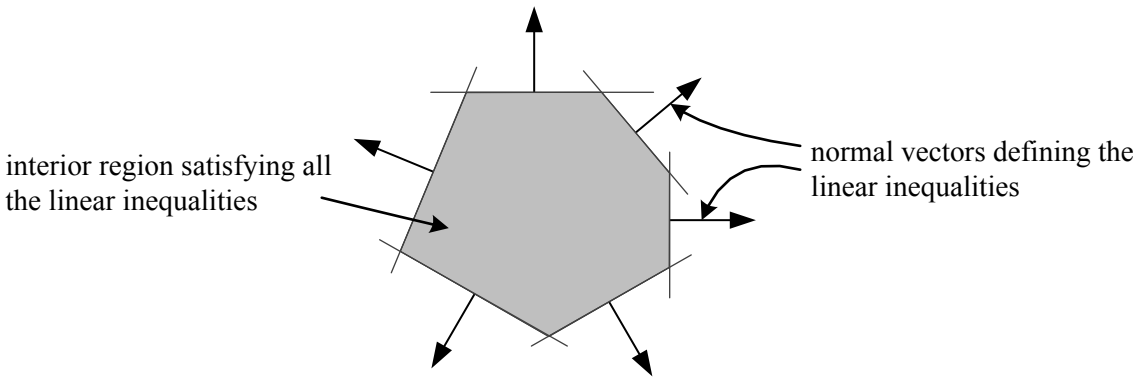


Fig. 8. The 2-D polygon defined using a set of six inequalities as shown in Eq. (6). The arrows represent the directions of the normal vectors. The shaded region is the convex set which satisfies all the inequalities.

2.2 Establishing intersection

In the subdivision approach, one needs to establish whether the existing blocks are intersected by the new discontinuity. To check whether there is intersection, we use a standard procedure for establishing feasibility in the field of convex mathematical optimization. The problem is here recast in terms of establishing whether there is a feasible point which satisfies all the linear inequality and equality constraints. This can be done by solving the following linear program:

$$\left.\begin{aligned} &\text{minimize } s \\ &\mathbf{a}_i^{\mathrm{T}}\mathbf{x} - d_i \le s, \quad i = 1,\ldots,N \\ &\mathbf{a}_{\text{new}}^{\mathrm{T}}\mathbf{x} - d_{\text{new}} = 0 \end{aligned}\right\} \tag{7}$$

where N is the number of planes of the parent block, and the new discontinuity is introduced as an equality constraint in Eq. (7). If s < 0, there is intersection and the parent block is subdivided; the linear inequalities from the parent block are inherited by each child block and a linear inequality from the new discontinuity is appended to each child block (see Fig. 9)

(with opposite sign for each child block). For example, let us consider that a parent block with N planes is subdivided into blocks A and B. Block A will inherit $\mathbf{a}_i^T\mathbf{x} \le d_i$, $i = 1,\ldots,N$ faces from its parent and have a new face, $\mathbf{a}_{\text{new}}^T\mathbf{x} \le d_{\text{new}}$, from the new discontinuity; whilst block B will have the inequalities $\mathbf{a}_i^T\mathbf{x} \le d_i$, $i = 1,\ldots,N$ and $-\mathbf{a}_{\text{new}}^T\mathbf{x} \le d_{\text{new}}$. Physical properties such as, for instance, friction angle and cohesion of the discontinuity are also stored with the new block face, with the possibility that each block face possesses different physical properties.

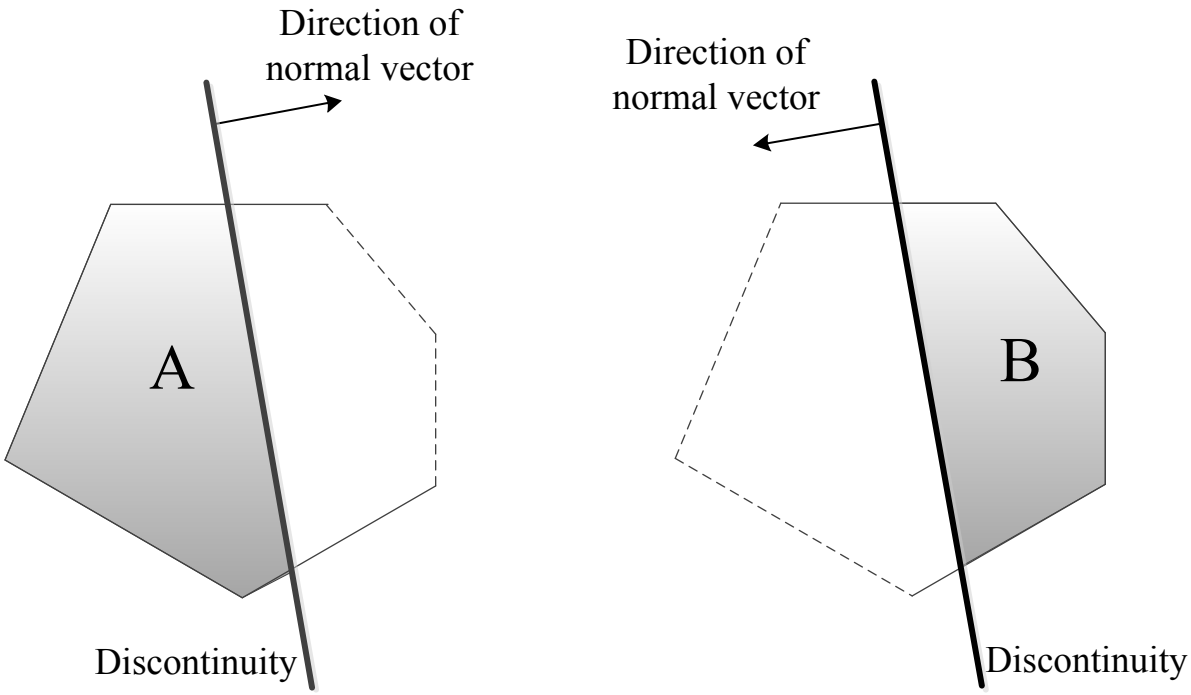


Fig. 9. The parent block in Fig. 8 is subdivided into a pair of children blocks (A and B). Opposite signs of the linear inequality of the new discontinuity is appended. Dashed lines are geometrically redundant for the shaded block.

Some of the linear inequalities inherited from the parent block could be geometrically redundant (dashed lines in Fig. 9). Geometrically redundant inequalities can be removed without changing the interior region of the polyhedron. To check whether a linear inequality

constraint $\mathbf{c}^{\mathrm{T}}\mathbf{x} \le d$ is redundant, we can solve the following linear program (Caron et al., 1989):

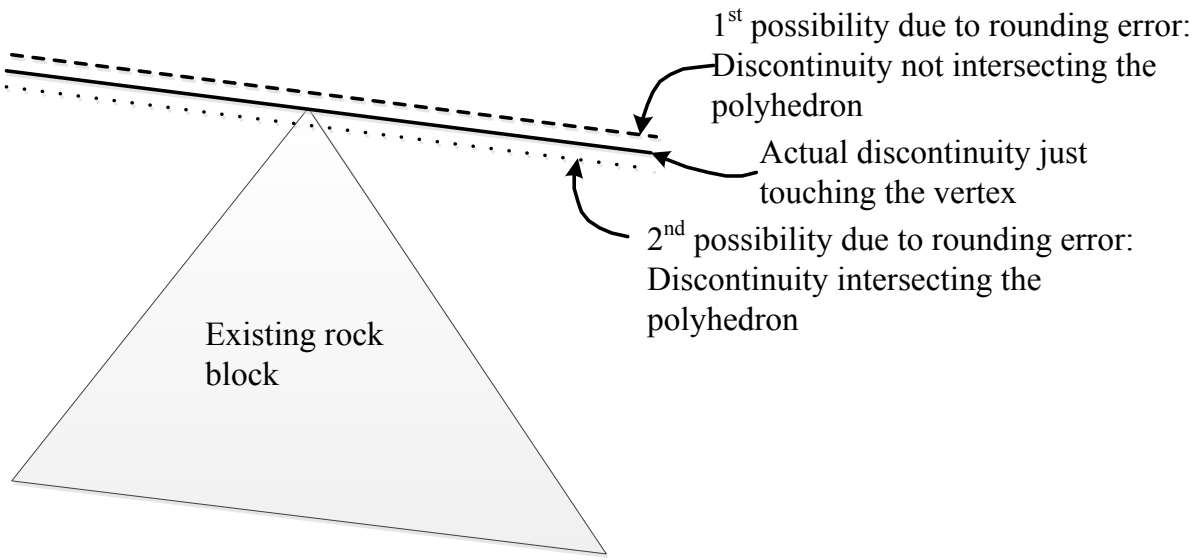
$$\left.\begin{array}{l} \text{maximise } \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \mathbf{a}_i^{\mathrm{T}}\mathbf{x} \le d_i, \quad i = 1,\ldots,N \end{array}\right\} \tag{8}$$

where $\mathbf{c}$ is one of the normal vectors $\mathbf{a}_i$. The linear inequality constraint is not redundant if $\left|\mathbf{c}^{\mathrm{T}}\mathbf{x} - d\right| < \varepsilon$, where ε is a numerical tolerance close to zero. The linear program (Eq. (8)) must be performed in turn for each linear inequality defining the block. It is not necessary to check whether the new discontinuity is redundant because we know beforehand that it forms a new face with the subdivided block. To increase efficiency, instead of checking for geometrically redundant planes after every subdivision procedure, users can do this at the end of the rock slicing process after all the blocks have been subdivided by all the rock joints.

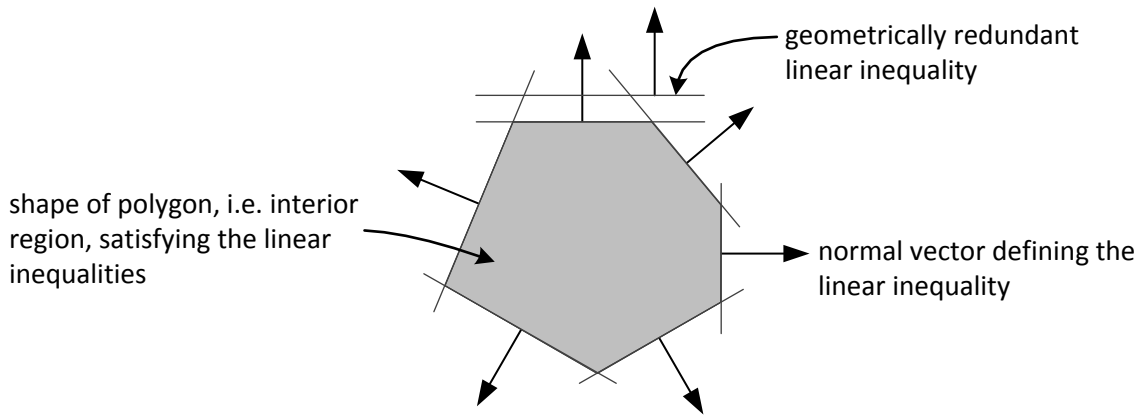
By comparison to existing methods in the literature, the data structure proposed in this method is less sensitive to rounding errors. For instance, Fig. 10 (a) shows that a joint plane just touching the vertex of a polyhedron may result in different outcomes due to rounding errors, i.e. whether or not the joint plane intersects the rock block. In the current numerical methods based on the subdivision approach, the number of new vertices and edges that are generated is sensitive to these rounding errors. In severe cases, these rounding errors originating from the use of a multi-level data structure and poor tolerance management may cause pitfalls such as the edges defining a face not to form a single closed loop, as was highlighted in Elmouttie et al. (2010). Instead, in the proposed method, the level of precision is related to the value assigned to the variable *s* in Eq. (7). If the joint plane is found to intersect the block, it is appended to the list of faces defining the block shape, otherwise it is omitted. Either outcome does not give rise to a significantly different data structure since in

our method it is no longer necessary to calculate edges and vertices, or to maintain a compatible hierarchical tree. In fact, if the numerical tolerance for *s* in Eq. (7) is too tight and the optimisation problem is ill-conditioned such as shown in Fig. 10 (a), the user is alerted by the optimisation software with a non-convergence warning, indicating that the new data structure would be extremely close to the existing data structures. In this case, the new data structure is not generated. That is to say, a self-defence mechanism is in place. Since the subdivision approach is sequential, the influence of rounding errors has a progressively larger impact on the subsequent data structures, and therefore the increase in robustness from using a simpler one level data structure presents an important advantage.

Also the proposed algorithm is more robust than conventional rock slicing algorithms because geometrically redundant faces, generated by rounding errors, do not cause harm to the calculations performed in subsequent subdivisions. This is due to the way a polyhedron is defined in our method (see Eq. (5) and Fig. 10 (b)). Conversely, in the conventional subdivision approach, redundant data structure could make the rock slicing code break down since these rock slicing algorithms require a compatible hierarchical data structure to do their job.



(a)



(b)

Fig. 10. Illustrations of increased robustness of data structures and algorithms: (a) the three possible outcomes due to numerical rounding errors are shown; (b) geometrically redundant planes.

An example of a C++ data structure for the blocks and discontinuities is shown in Fig. 11. The data structure for the discontinuities can be discarded after the rock slicing process is complete since this information is no longer needed in subsequent computations which only require data from the block faces (see Fig. 11 (b)). It is worth to point out that in the case of very densely jointed rock masses (i.e. thousands of joints), it may be more efficient to use pointers to link every block face to its original rock joint. In this case, the physical properties of the joint data could be stored into the discontinuity data structure only rather than in the block data structure.

In summary, the proposed algorithm employs a substantially simpler data structure than the hierarchical data structure used conventionally, e.g., edges by pair of vertices, faces by bounding edges, etc. More importantly, tedious housekeeping algorithms for the updating of the vertex-edge-face data structure in conventional subdivision procedures are no longer needed. The flow of the algorithmic implementation of our method is illustrated in Fig. 12.

```
struct Discontinuity{
        /* Comment: (centre_x,centre_y,centre_z) is the position of the
discontinuity */
        double centre_x;
        double centre_y;
        double centre_z;
        /* double dip angle and dip direction */
        double dip;
        double dipDir;
        /* Comment: (a,b,c) is the vector normal to the plane of the discontinuity
        */
        double a;
        double b;
        double c;
        /* Comment: d is the distance of the discontinuity from the local centre*/
        double d;
        /* Comment: phi is the friction angle along the discontinuity */
        double phi;
        /* Comment: cohesion is the cohesion along the discontinuity */
        double cohesion
        /* Comment: number of lines delimiting the joint extent */
        int N_lines;
        /* Comment:  (shape_a[i], shape_b[i]) is the i^th line representing the
        polygonal shape of the joint with respect to local coordinates orthogonal
        to the joint normal. Define the arrays with entry sizes larger than the
        largest expected number of lines delimiting the joint extent, N_lines. */
        double shape_a[10];
        double shape_b[10];
        /* Comment:  shape_d[i] is the distance of the i^th trace line with respect to
        the local coordinates orthogonal to the joint normal */
        double shape_d[10];
};
```

(a)

```
struct Block{
        /*Comment: (centre_x, centre_y, centre_z) is the position of the block */
        double centre_x;
        double centre_y;
        double centre_z;
        /* Comment: number of planes defining the block */
        int N_planes;
        /* Comment:  (a[i],b[i],c[i]) is the normal vector of the i^th entry of the
        faces.  Define the arrays with entry sizes larger than the largest expected
        number of faces */
        double a[40];
        double b[40];
        double c[40];
        /* Comment:  d[i] is the distance of the (a[i],b[i],c[i]) face from the local
        centre*/
        double d[40];
        /* Comment: phi[i] is the friction angle of the (a[i],b[i],c[i]) face */
        double phi[20];
        /* Comment: cohesion[i] is the cohesion of the (a[i],b[i],c[i]) face */
        double cohesion[20];
};
```

(b)

Fig. 11. Example of the C++ data structure for (a) discontinuities and (b) blocks used in the algorithm. Geometrical and physical properties of the block faces are stored in arrays.

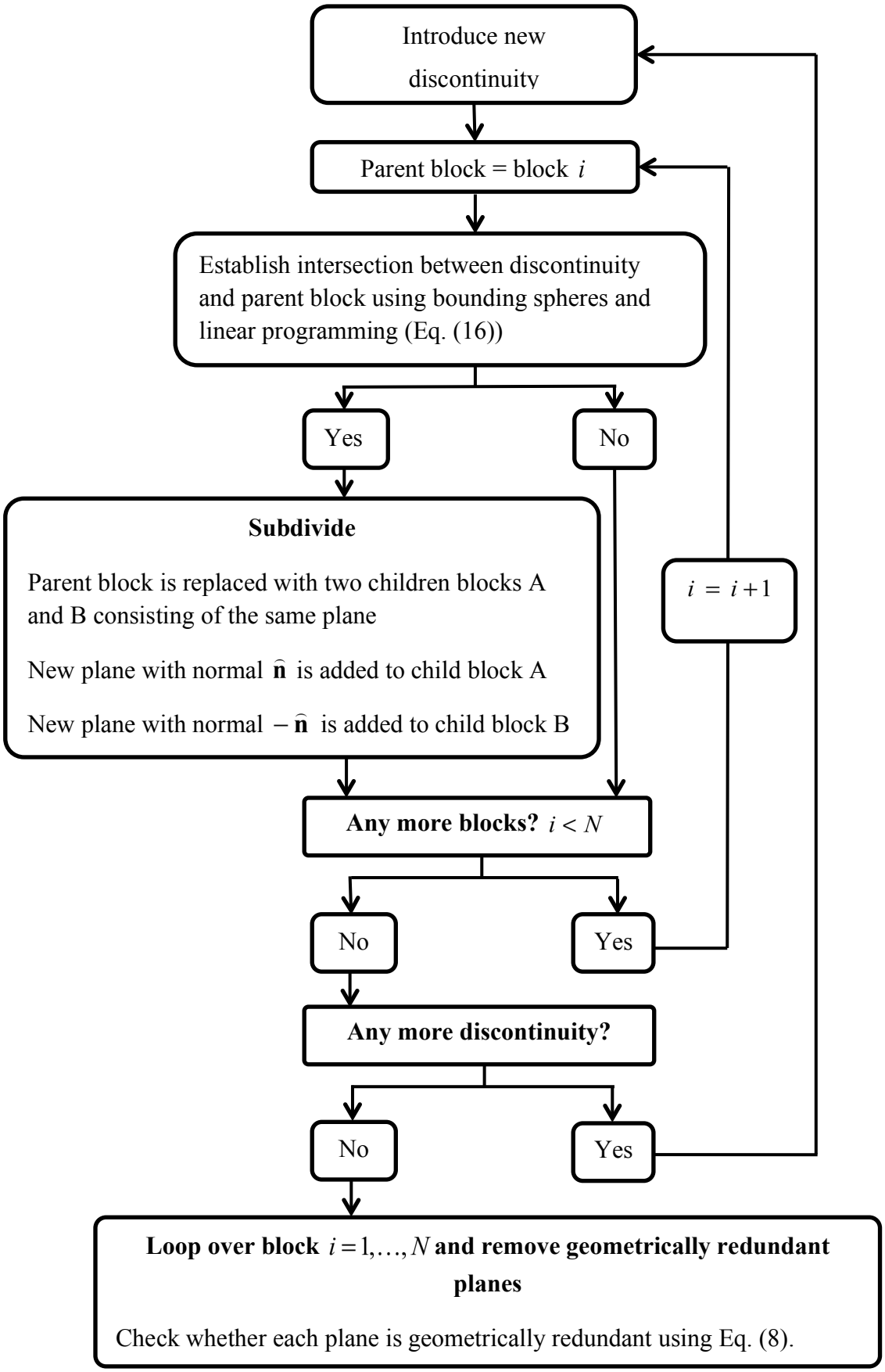


Fig. 12. Flow chart of the algorithmic implementation of the proposed rock slicing method (optimisation of efficiency is discussed later in Section 2.5)

2.3 Taking into account the shape of the discontinuities

The previous analysis assumes that discontinuities are through-going in the domain of interest, i.e. persistent or infinite in extent. However, in reality joints are non-persistent, i.e. finite in extent. Also recent studies in the literature emphasize the three-dimensional nature of rock discontinuities (see Fig. 13), the probabilistic nature of joint extents, and the probabilistic spatial distribution of joint centres (Zhang et al., 2002). Furthermore recent field investigations found that most discontinuities are either polygonal or elliptical in shape (Zhang & Einstein, 2010). Considering now a polygonal discontinuity, its boundaries must lie in a plane. Hence they can be specified using the following linear inequalities (see Fig. 14):

$$a_{i\text{local}} x + b_{i\text{local}} y \le d_{i\text{local}}, \quad i = 1,\ldots,M \tag{9}$$

Employing vector notation, this becomes:

$$\mathbf{a}_{i\text{local}} \mathbf{x} \le d_{i\text{local}}, \quad i = 1,\ldots,M \tag{10}$$

Eq. (10) is defined in terms of local coordinates written with reference to an axis origin located at the discontinuity centre with the normal vector of the discontinuity taken to be the local z-direction. Note that, in the special case of rectangular-shaped joints, the input required from the user can be specified in a simpler form consisting of only the length and width of the joint (see Fig. 14 (b)). In the case of elliptical joints, the approach proposed here can still be employed by replacing the elliptical joint with a linear polygonal approximation (see Fig. 14). The simplest technique would be to use polygons (see Fig. 14 (a) and (b)) circumscribing the ellipse. In this case, the approximation of the joint is on the safe side since a larger joint area is considered. Alternatively, polygons with areas equivalent to the elliptical joints could be used but their computation is more cumbersome.

The inequalities in Eq. (10) have to be transformed into global coordinates. The rotation matrix has to express the geometrical transformation needed to rotate the local y-axis in such a way to make it point along the dip vector (refer to Fig. 2). Therefore, if the rock

joint is elongated, the polygon in Eq. (10) must be defined in such a way that the local y-axis is oriented along the dip vector (refer to Fig. 13). To obtain such as rotation matrix, first let us define the vectors $\mathbf{N}_{\text{strike}}$ and $\mathbf{N}_{\text{dip}}$ as:

$$\mathbf{N}_{\text{strike}} = (\cos\theta_{\text{strike}}, \sin\theta_{\text{strike}}, 0) \tag{11}$$

$$\mathbf{N}_{\text{dip}} = \mathbf{N}_{\text{plane}} \times \mathbf{N}_{\text{strike}} \tag{12}$$

after which $\mathbf{N}_{\text{dip}}$ is normalised. Recalling that $\mathbf{N}_{\text{plane}}$ is the normal to the joint plane, and θ_{strike} is the strike angle, the rotation matrix, $\mathbf{Q}_{\text{plane}}$, is obtained as:

$$\mathbf{Q}_{\text{plane}} = \begin{bmatrix} N_{\text{strike\_x}} & N_{\text{dip\_x}} & N_{\text{plane\_x}} \\ N_{\text{strike\_y}} & N_{\text{dip\_y}} & N_{\text{plane\_y}} \\ N_{\text{strike\_z}} & N_{\text{dip\_z}} & N_{\text{plane\_z}} \end{bmatrix} \tag{13}$$

The coefficients of the linear inequalities with reference to the transformed axes for M polygonal boundaries are therefore:

$$\mathbf{a}_{j\text{bound}} = \mathbf{Q}_{\text{plane}} \mathbf{a}_{j\text{local}}, \qquad j = 1,\ldots,M, \tag{14}$$

$$d_{j\text{bound}} = \mathbf{a}_{j\text{bound}}^{\text{T}} \mathbf{x}_0 + d_{j\text{local}}, \qquad j = 1,\ldots,M, \tag{15}$$

Our method is based on the assumption that non-persistent discontinuities inside any intact block develop fully so that the block is completely sliced, or in other words, no polygonal discontinuity can terminate inside an intact block but it always terminates at another discontinuity. In order to account for the shape of the discontinuities when establishing intersection, the linear program of Eq. (7) becomes:

$$\left.\begin{aligned} &\text{minimize } s \\ &\mathbf{a}_i^{\text{T}} \mathbf{x} - d_i \le s, \quad i = 1,\ldots,N \\ &\mathbf{a}_{\text{new}}^{\text{T}} \mathbf{x} - d_{\text{new}} = 0 \\ &\mathbf{a}_{j\text{bound}}^{\text{T}} \mathbf{x} - d_{j\text{bound}} \le s, \quad j = 1,\ldots,M \end{aligned}\right\} \tag{16}$$

where $\mathbf{a}_{j\text{bound}}^{\mathrm{T}}\mathbf{x} - d_{j\text{bound}}$ defines the boundaries of the new discontinuity $\mathbf{a}_{\text{new}}^{\mathrm{T}}\mathbf{x} - d_{\text{new}}$. If s < 0, there is intersection and the parent block is subdivided. With regard to the extent of the generated discontinuities, note that at the beginning of the slicing procedure, ~~the extent of~~ the first joint must ~~be as large as~~ slice through the first block, which is the entire domain. As more blocks are subdivided, the extent of the new joints progressively reduces. To avoid generating cuts deeper than necessary, discontinuities with larger extents have to be introduced before introducing discontinuities with smaller extents. This is discussed further in the next section.

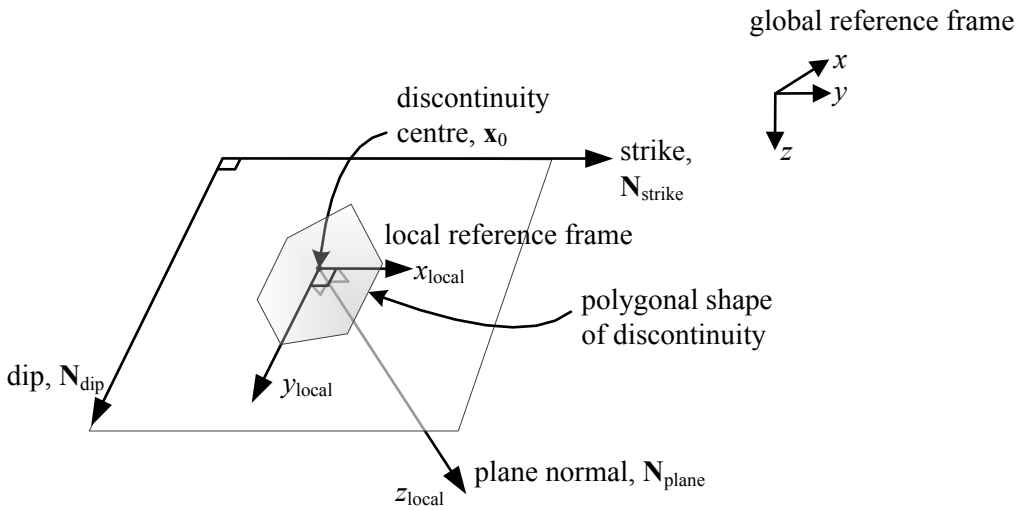


Fig. 13. In 3D, the discontinuity plane is bounded by lines forming a polygon.

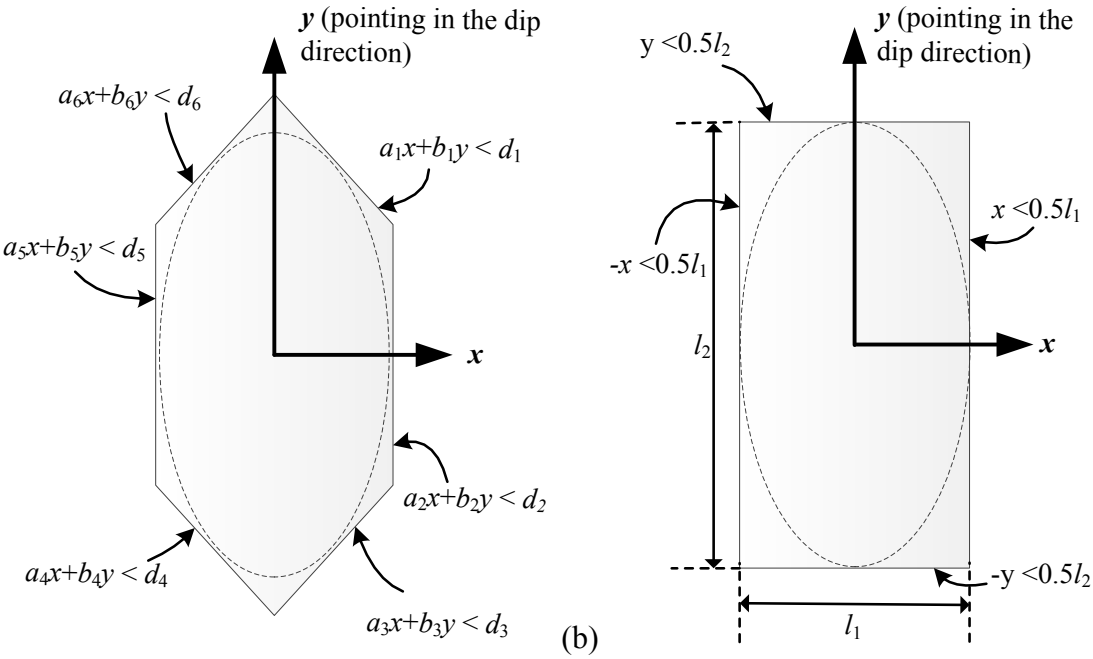


Fig. 14. Examples of how the extent of a rock joint can be delimited using linear inequalities. An ellipse can be approximated as a polygon consisting of *N*-lines, for instance as (a) a hexagon or (b) a rectangle

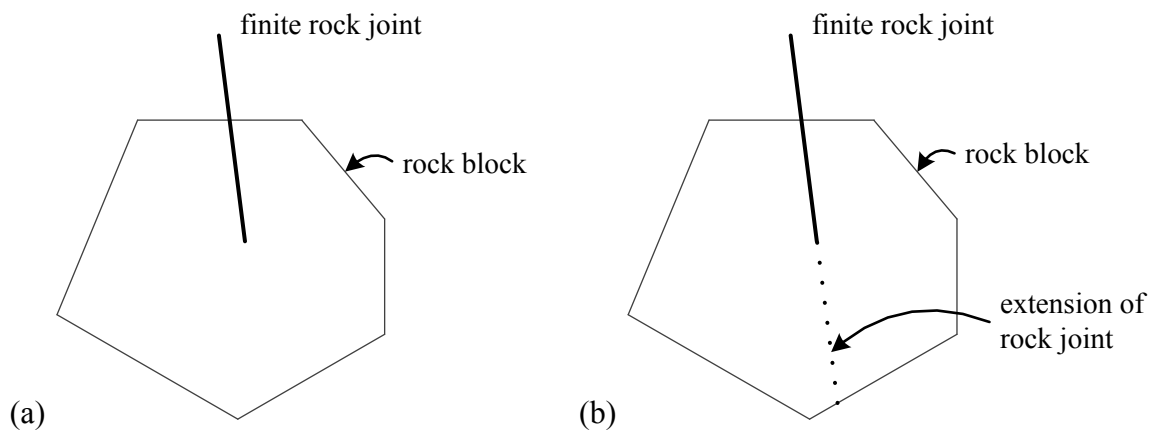


Fig. 15. 2-D illustration of (a) actual rock joint extent (b) rock joint as modelled in the proposed method (the fracture is fully extended)

2.4 Implementation of the method into a numerical code for discontinuum analysis

At the end of the rock slicing procedure, each block is defined solely by its faces. Thereafter, one may need to work out the vertices and edges defining the polyhedral blocks, depending on the type of numerical simulation to be performed, e.g. DEM, DDA or Finite Element Method (FEM) with interface elements. Traditionally, the vertices have to be calculated in order to work out the volume of the blocks and their bounding boxes which are required by the algorithms sorting out the neighbouring pairs of blocks in DEM and DDA codes. Additionally, the contact detection algorithm of a discontinuum analysis may require information concerning the block vertices and edges, as well as their adjacency relations, *i.e.* edges by pairs of vertices, faces by bounding edges, polyhedra by bounding faces. If this is the case, after the faces of the blocks have been identified using the method proposed here, vertices and edges can be determined using existing methods in the literature, *e.g.* the method by Ikegawa & Hudson, (1992).

However, a contact detection algorithm between polyhedral blocks, which does not require information about their vertices and edges, has recently been proposed by Boon et al. (2012) for DEM analyses. Between a pair of blocks potentially in contact, i.e. two blocks whose bounding boxes overlap, it has been shown that there are well-established convex optimisation procedures (Boyd & Vandenberghe, 2004) which one can use to check whether they intersect and to calculate the contact point between them. The calculation only requires information on the linear equalities defining the block faces. If this contact detection algorithm is employed in a discontinuum analysis, the same data structure, i.e. in terms of block faces, calculated using the rock slicing method proposed here can be used as input.

A comprehensive literature review on contact detection algorithms for DEM analyses for polygonal and polyhedral objects can be found in Boon (2013). The contact detection algorithm proposed by Boon et al. (2012) is based on a centering algorithm that determines the contact point

between two polyhedral blocks in contact (i.e. with a small overlapping region) as the analytic centre of the linear inequalities defining the contacting blocks. The analytic centre is found employing standard convex optimisation techniques. The advantage of this algorithm in comparison with traditional contact detection algorithms for polyhedral blocks in the literature (*e.g.* Cundall, 1988; Nezami et al., 2006) are three-fold: *i*) the algorithm does not need to identify the different types of interaction between contacting polyhedra, *i.e.* —face–face, face–edge, face–vertex, edge–edge, edge– vertex, or vertex–vertex, so that a simpler data structure only containing information on block faces can be used; *ii*) a smooth transition between different contact types, for instance from edge–edge to vertex–edge, is ensured (on the contrary for traditional contact detection algorithms based on the distinction of contact types, physically unjustified ''jumps'' of the contact point may occur when the contact type changes); *iii*) ambiguity in the calculation of the contact point for complex contact types such as edge – edge in 3-D (Cundall, 1988) is eliminated. Several validation examples have been reported in Boon et al. (2012), where the novel contact detection algorithm is used for problems involving various contact scenarios between polyhedral blocks. Also the algorithm has been used to analyse the stability of tunneling excavations performed in jointed rock masses with 3 independent sets of non-persistent joints (Boon et al., 2014a) and to model the 1963 Vajont rock slide (Boon et al., 2014b). Finally, with regard to the position and direction of the resulting contact forces, in the case of a jointed Voussoir beam (Boon, 2013), this contact detection algorithm was also found to provide results in good agreement with finite difference method analyses of Tsesarsky & Talesnick (2007).

2.5 Joint generation sequence

For algorithms based on sequential subdivision, unless the joint extents are assumed to be infinite during the procedure of subdivision, the sequence employed to introduce non-

persistent joints in general affects the generated pattern of joints and blocks. The extents of most rock joint sets are characterised by either a log-normal or an exponential statistical distribution (Baecher, 1983; Zadhesh et al., 2012), therefore joint extents vary from small to very large. To avoid creating 'slices' which are too large compared to the assigned joint extents, joints with larger extents should be introduced first. Because the first few slices inevitably have to span through the entire domain, they could be assigned as fictitious joints possessing the mechanical strength of the intact rock. Similar approaches in controlling the joint extents have also been adopted in 3DEC (2013), i.e. a distribution of fictitious joints is generated before generating the actual joints with joints of larger extents being created first. In the subsequent paragraphs, we discuss further the influence of the sequence employed to generate the joints. The practice of introducing joints with larger extents first is to mimic the mechanical genesis of rock joints as close as possible from a geometrical standpoint, when limited knowledge on the past geological stress history of the rock mass is available. Reasons for which new joints tend to terminate at existing joints are explained in Mandl (2005).

In order to generate random joint patterns, such as the ones found in homogeneous rock masses where the formation of discontinuities is not dominated by lithological or structural variability (Priest & Hudson, 1976), fractures should be introduced in a sequence mimicking random generation. To this end, joints belonging to different sets could be introduced in alternating succession so that the formation of long parallel blocks, i.e. pancake shaped blocks, is avoided. In this case, each slicing sequence can be viewed as reproducing a particular geometry extracted from the prescribed probability distribution characterising the joint pattern of the analysed rock mass. It is worth to note that this approach of accounting for non-persistent joints will result in a block assembly with the number of generated blocks falling between two extreme cases: the first case where all the rock joints are persistent and

the other case in which all the dangling joints are removed. If the user desires to check the generated blocks against the actual patchwork of discontinuities and perform clustering as illustrated by Yu et al. (2009), it will be necessary to work out the vertices at the end of the proposed procedure. The vertices are normally calculated to estimate the volumes of the blocks, after which clustering can be carried out.

If the joint pattern is fairly regular, such as the pattern found in bedded sedimentary rocks (Pollard & Aydin, 1988; Priest & Hudson, 1976), it is important that fractures are introduced according to a sequence which is consistent with their mechanical genesis. For instance, large bedding planes are usually formed in the rock mass before cross-joints develop. The fracture sequence in a limestone rock mass identified by Hudson (2012), based on the way the fractures terminate, is shown in Fig. 16. In the figure, the numbers refer to the order of chronological formation of the joint sets. In this regard, note that the philosophy of our method for block generation is consistent with the more recent development of hierarchical fracture system models which distinguish between primary and secondary fractures (e.g. Lee, 1990; Ivanova, 1995). In section 3.1, it will be illustrated how the proposed rock slicing (or block generation) method is capable of reproducing joint patterns in a manner consistent with their mechanical genesis.

Fig. 16. Sequence of fracturing identified by Hudson in a limestone rock mass featured by 4 joint sets (image after Hudson (2012)).

2.6 Bounding spheres and conditioning of sizes

In the previous sections, the discussion has been limited to the essential features of the novel mathematical treatment of rock slicing via linear programming. Some non essential features of the method will now be discussed. In the previous sections, every rock joint has to be checked against every other polyhedron for intersection, i.e., to establish whether a polyhedron should be subdivided. When the number of rock joints is large and the joint extents are small relative to the size of the domain, this can be inefficient. Due to the inherent~~ly~~ sequential nature of the methods based on subdivision, as the number of subdivided polyhedra increases, there are progressively more polyhedra against which the rock joint has to be checked for intersection. Adopting a procedure typical of contact detection algorithms in the DEM, it is convenient to associate each polyhedron or rock joint to a simpler shape (such as a sphere) completely enclosing the block or joint, so that a faster check can be executed to decide whether it is necessary to run more complex intersection tests. Yu et al. (2009) introduced the use of prismatic bounding boxes in rock slicing. Here, we employ bounding spheres instead.

To work out the radius and centre of the sphere bounding a polyhedron, it is necessary to know the extents of the polyhedron. The extents of a polyhedron can be calculated by running a linear program (Eq. (17)) along each principal axis $\mathbf{e}_i$ in the positive and negative directions, i.e. x, -x, y, -y, z, -z in 3-D or x, -x, y, -y in 2-D.

$$\left.\begin{aligned} &\text{maximise } \mathbf{e}_i^{\mathrm{T}}\mathbf{x} \\ &\mathbf{a}_j^{\mathrm{T}}\mathbf{x} \le d_j, \quad j = 1,\ldots,N \end{aligned}\right\} \tag{17}$$

where $\mathbf{e}_i$ is the unit vector directed along the principal axis of interest. Having done this, we will get a pair of coordinates $\mathbf{x}_{\mathrm{p}} = (x_{\mathrm{p}}, y_{\mathrm{p}}, z_{\mathrm{p}})$ and $\mathbf{x}_{\mathrm{n}} = (x_{\mathrm{n}}, y_{\mathrm{n}}, z_{\mathrm{n}})$ which are the most

positive and negative x, y, z coordinates respectively on the particle boundaries (see Fig. 17 (a) for a 2-D illustration). The radius of the bounding sphere can be calculated as $R = 0.5\|\mathbf{x}_{\mathrm{p}} - \mathbf{x}_{\mathrm{n}}\|$. The centre of the bounding sphere can be taken as the average of the extents, i.e. $0.5\times(x_{\mathrm{p}} + x_{\mathrm{n}}, y_{\mathrm{p}} + y_{\mathrm{n}}, z_{\mathrm{p}} + z_{\mathrm{n}})$. On the other hand, the bounding sphere for a rock joint can be approximated easily from its extents. Before running the actual intersection test between a rock joint and a polyhedron, it is more efficient to first check whether their bounding spheres overlap (see Fig. 18 (a) for a 2-D illustration). In fact, if their bounding spheres do not overlap, it is not necessary to run the more expensive linear program of Eq. (16). For a joint of infinite extent, one can check whether the distance of the centroid of the bounding sphere to the joint plane is less than the radius of the bounding sphere, before running the actual intersection test (see Fig. 18 (b)).

Especially when rock joints are generated according to probabilistic distributions, it is desirable to control the size of the polyhedra. For instance, the maximum extent of the time step in a DEM simulation is restricted by the size of the smallest polyhedron in the simulation. Removing small polyhedra from the domain after they have been generated, will create voids in the model; so if this approach is adopted, the tolerance has to be very small to avoid creating excessively large voids. An alternative approach is to ensure, throughout the slicing procedure, that the size of the subdivided child blocks is above an assigned tolerance. In this approach, when the size of one of the child block is found to be below the prescribed tolerance, the data structure of the parent block is restored so that the block is not subdivided.

In principle, it is possible to estimate the size of the blocks from the radius of their bounding spheres (Fig. 17(a)). However, this would not be a robust method since slices which subdivide a parent block into either needle or pancake shaped child blocks can satisfy the tolerance more easily, which in turn would lead to a model consisting of numerous highly elongated blocks. A better way to approximate the size of a block is by employing its largest

inscribable sphere. There are several ways to work out the radius of a sphere inscribable in a convex polyhedron, one of which is to solve:

$$\left.\begin{array}{l} \text{minimize } t \\ \mathbf{a}_i^{\mathrm{T}}\mathbf{x} - d_i \le t, \quad i = 1,\ldots,N \end{array}\right\} \tag{18}$$

with $\mathbf{a}_i$ being unit vectors, t the largest inscribed radius of a sphere in a polyhedron and $\mathbf{x}$ being the solution of the optimisation problem expressed by Eq. (18), i.e. being the Chebyshev centre of the polyhedron (Boyd & Vandenberghe, 2004). Geometrically, the Chebyshev centre represents the centre of the largest sphere inscribable in the polyhedron.

In some cases, it is also of interest to control the maximum aspect ratio of the polyhedra in the simulations, e.g. in order to reduce the occurrence of "outliers" with elongated shapes near an excavation. Therefore, to achieve a better "conditioned" blocky rock mass, it is worthwhile to examine the aspect ratio of a subdivided block during the slicing process. The aspect ratio of a block can be approximated as the ratio of the radii of the bounding sphere to the inscribed one (see Fig. 17 (b)). A large ratio suggests that the block is either pancake or needle shaped. The sizes and aspect ratios of the child blocks after potential subdivision can be checked against their respective tolerances, before operating the subdivision. If one of the tolerances is not satisfied, although the data structures for the child blocks have been calculated, the original data structure of the parent block is restored.

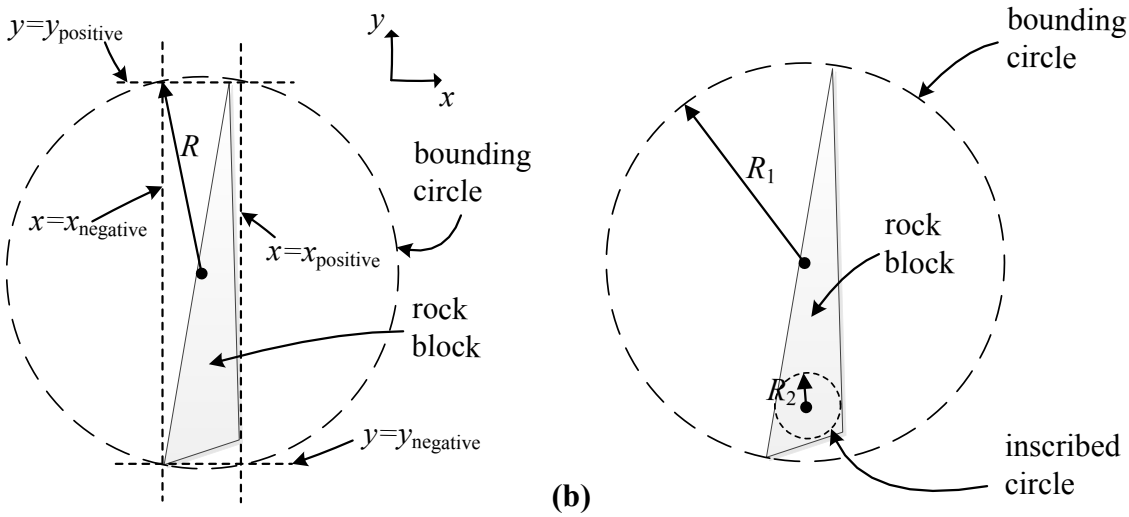
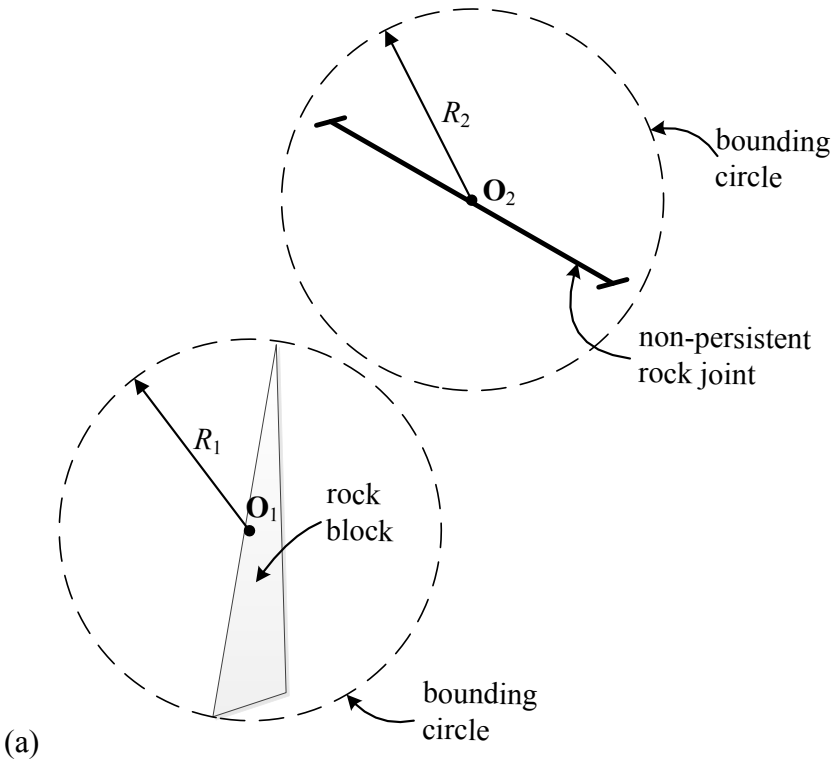


Fig. 17. Illustration of bounding and inscribed circles in 2-D (a) Approximating the bounding circle (b) Checking for pancake or needle-shaped blocks based on the ratio of bounding to inscribed circle.



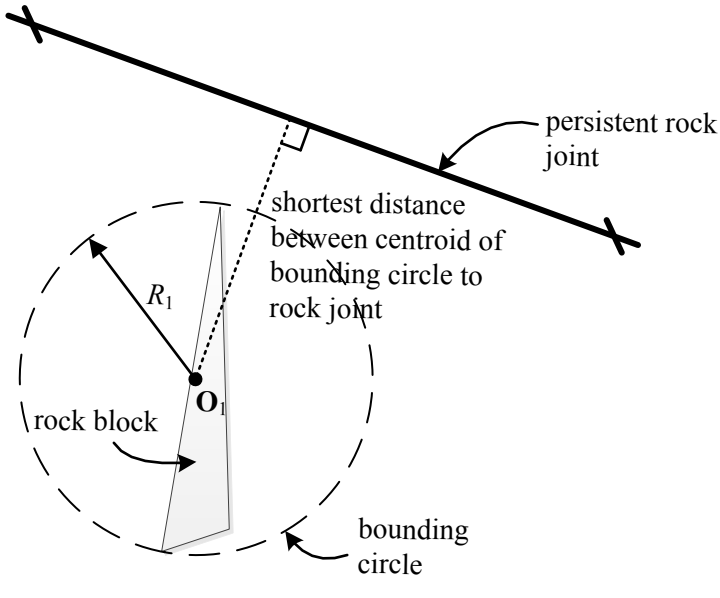


Fig. 18. Use of bounding spheres to check for potential intersection for (a) non-persistent rock joints ($Overlap = (R_1 + R_2) - \|\mathbf{O}_1 - \mathbf{O}_2\|$) and (b) persistent rock joints.

3. Validation

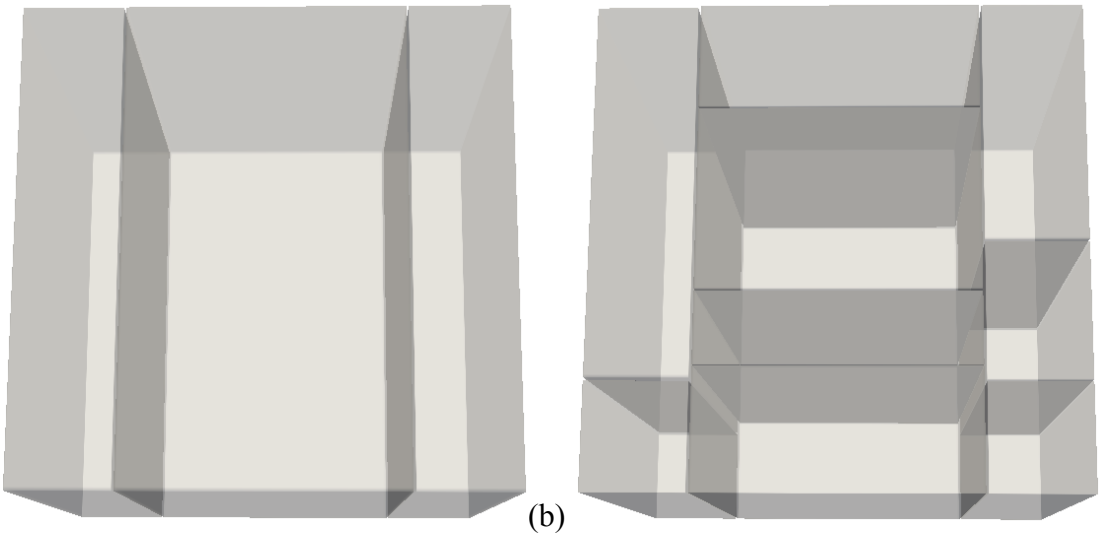
Some examples are provided here for validation purposes. The proposed rock slicing method was coded into a series of routines in C++, provided in the supplementary material. For visualisation purposes, the open-source DEM code YADE (Kozicki & Donzé, 2008) was employed to plot the obtained rock joint patterns. The linear programs were solved using the simplex algorithm of the linear programming software IBM ILOG CPLEX (CPLEX, 2003) accessed via a C++ interface. CPLEX is freely available to academics through the IBM academic initiative program. We chose the simplex algorithm over the log-barrier algorithm since it proved to be more robust.

3.1 Slicing sequence based on the mechanical genesis of fractures

The development of fracture system models based on fracture mechanical genesis is foreseen to be an important research topic in rock mechanics (Hudson (2012)). Although the generation of the discrete fracture network (DFN) is beyond the scope of this paper, it is

worthwhile to show that, if the mechanical genesis of the fractures is known, fractures could be introduced in the proposed rock slicing (or block generation) algorithm based on the actual sequence of fracturing (section 2.5). In a large number of cases, the fracture sequence can be inferred by visual inspection based on the way in which the fractures terminate, as illustrated in Hudson (2012) (see Fig. 16).

The rock mass intersected by four joint sets of Fig. 16 is used here to demonstrate that the proposed method is capable of generating fractures according to a sequence consistent with their mechanical genesis. In the algorithm, we introduced first all the fractures from joint set '1' (Fig. 16). Then, we introduced all the fractures from joint set '2' (Fig. 16), so that they terminate against the fractures from joint set '1'. In the same manner, we introduced all the fractures from joint set '3' (Fig. 16), followed by the fractures from joint set '4' (Fig. 16). The blocks generated according to this slicing sequence are shown in Fig. 19 (a) – (d). The plan view of the generated block assembly is shown in Fig. 19(e). Comparing Fig. 14(e) with Fig. 16, it emerges that the final patchwork of the generated joints agrees very well with the observed rock patchwork. Further, note that in this example non-persistent fractures were generated via the sequential subdivision of blocks in an automated manner.



(a) (b)

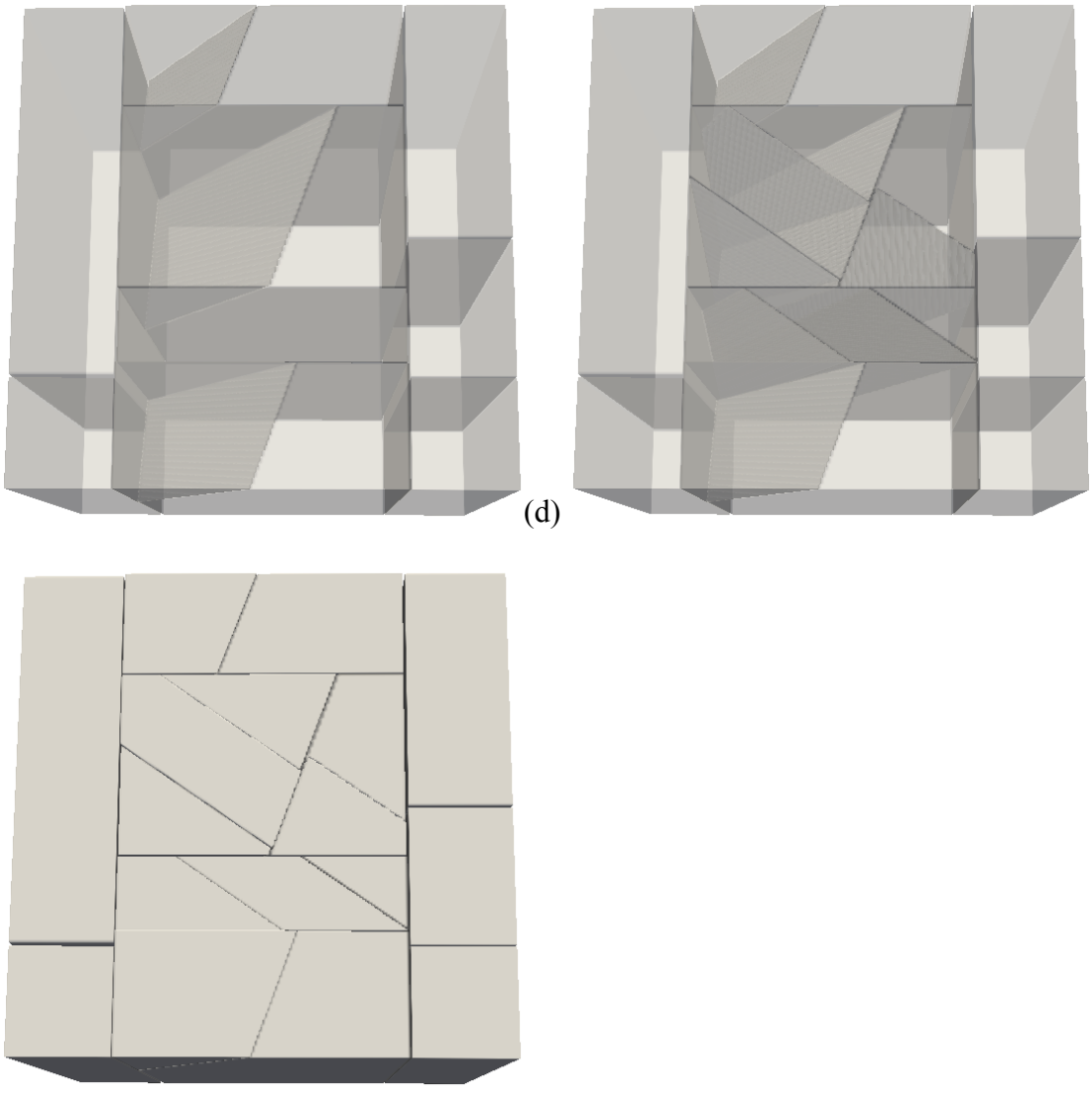


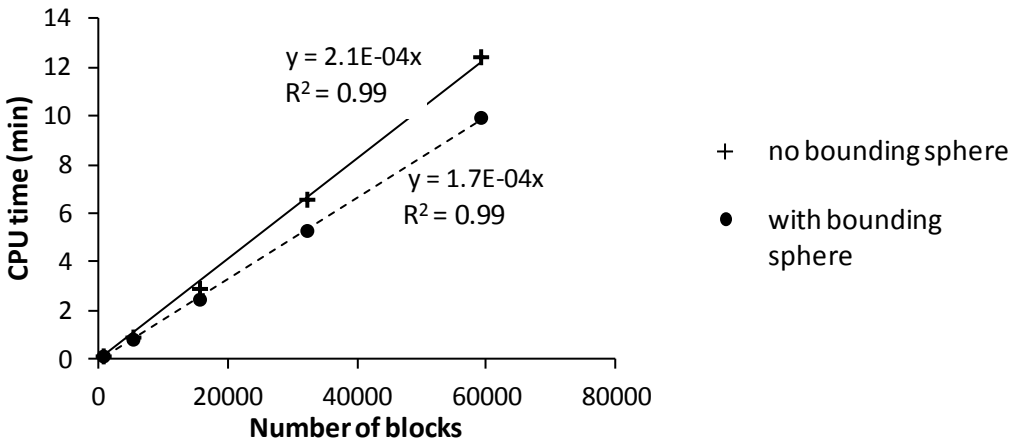
Fig. 19. Blocks generated based on the sequence of fracturing shown in Fig. 16: (a) joint set '1' is first introduced, (b) then joint set '2' is introduced, followed by (c) joint set '3' and (d) joint set '4' (the opacity of the illustrations is reduced for clarity), (e) generated blocks (fully opaque illustration compare with Fig. 16). The number of fractures in this example was kept small for the sake of clarity of the illustrations.

3.2. Algorithm scaling

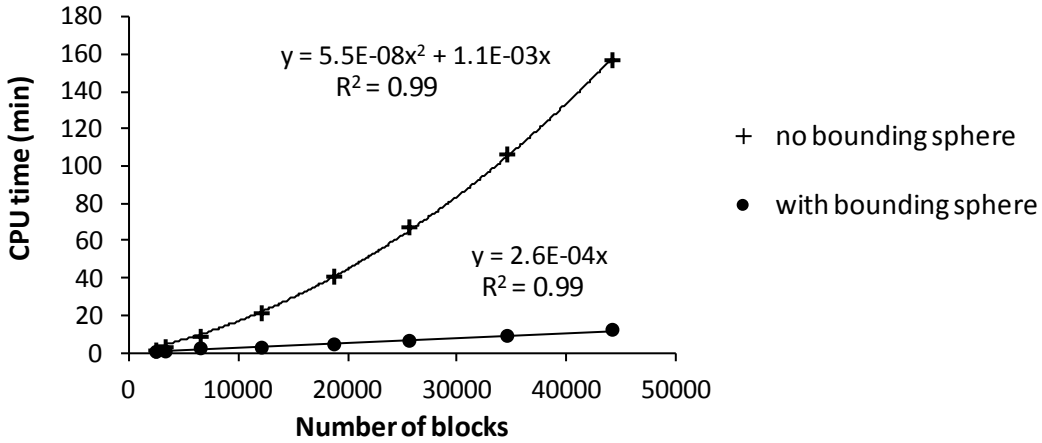
The scaling of the implemented algorithm with the number of generated blocks was investigated for a 3D configuration. The input is shown in Table 3. The number of generated blocks increases with the volume density of the rock joint centres. Non-persistent joints were generated based on a log-normal distribution. The additional efficiency derived from using bounding spheres was also investigated for both persistent and non-persistent joints. Fig. 20

(a) shows the times required for block generation for the case that all the three joint sets are persistent. The calculation was carried out on a standard desktop PC using one core of a Core-2-Duo CPU (3.1 GHz). It turned out that the computation time for block generation scales linearly with the number of generated blocks. The efficiency derived from using bounding spheres is not very significant; however, the computational saving becomes more significant as the number of blocks increases.

Fig. 20 (b) shows the times required for block generation for the case that all the three joint sets are non-persistent. Without bounding spheres (crosses in Fig. 20 (b)), the computation time increases with the number of generated blocks in a non-linear manner. With bounding spheres (dots in Fig. 20 (b)), the computation time was found to increase approximately linearly with the number of blocks. The efficiency derived from using bounding spheres is very significant; for more than 20000 blocks, the computation time for block generation is reduced more than 10 times. One of the joint patterns generated (number of blocks = 2495) is shown in Fig. 21. In this example, fractures from each joint set are introduced in an alternating manner, i.e. fracture from joint set A – fracture from joint set B – fracture from joint set C – fracture from joint set A and so on in a repeating sequence. This procedure is more appropriate for generating rock masses with random fracture patterns (Fig. 21), for instance fractures in granite rock masses.



(a)



(b)

Fig. 20. Computation time versus number of generated blocks for (a) persistent joints, (b) non-persistent joints

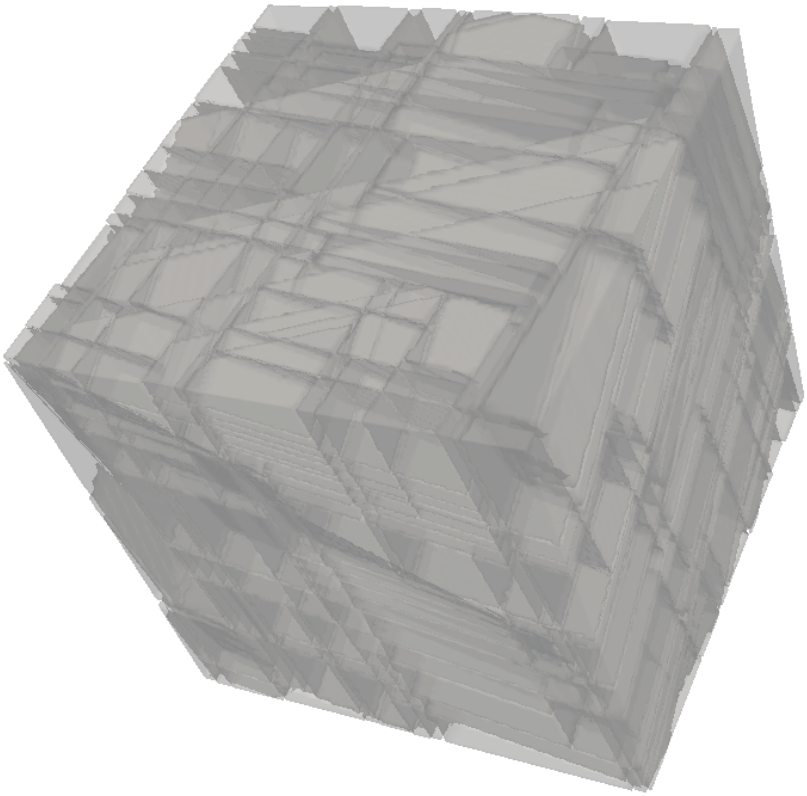


Fig. 21. Generated block assembly (2495 blocks) with three near-orthogonal joint sets (non-persistent joints).

Table 3: Input for algorithm scaling

Parameter	Input
Dimension of box sample	100 m × 100 m × 100 m
Orientation of joint set A	Dip direction = 122°, dip angle = 8°
Orientation of joint set B	Dip direction = 112°, dip angle = 80°
Orientation of joint set C	Dip direction = 9°, dip angle = 85°
Distribution of joint centres	Poisson's process
Joint extents (persistent case)	1000 m × 1000 m for all three joint sets
Joint extents (non-persistent case)	Square shape, lognormal distribution (mean = 5 m, standard deviation = 1 m) for all three joint sets
Minimum size (diameter of inscribed sphere)	0.4 m
Maximum aspect ratio (axes aligned box)	8000

3.3. Generated block assembly and discussion of suitable applications

In this verification example, the 2D joint pattern assigned as input into UDEC by Kim et al. (2007) (see Fig. 22 (a)) was generated using our proposed rock slicing method for comparison purposes. In Fig. 22(b), the blocks generated by the DEM software UDEC are plotted: it can be observed that dangling joints were removed since in UDEC rock joints have to either form a block face (so contributing to block formation) or be removed completely (Cundall, 1988). The same block assembly was generated anew via the proposed rock slicing method. The position and orientation of every joint in Fig. 22 (a) was determined and given as a deterministic input for the proposed rock slicing algorithm. Fig. 23 (a) shows the generated blocks having assumed joints of infinite extent; Fig. 23 (b) shows the generated blocks with non-persistent joints; Fig. 23 (c) shows the generated blocks having enforced a minimum block size of 0.4 m, estimated from visual observation of Fig. 22 (a), throughout the slicing procedure. Note that suitably conditioning the size of the smallest blocks in the assembly may significantly reduce the simulation runtime of a DEM analysis since the critical timestep in the simulation is a function of the smallest block size. This method is neater than scaling the mass of smaller blocks. In fact non-uniform mass scaling may result in generating heavy small masses in the system, that in turn may lead to unrealistic behaviour if they are subject to little confinement such as in the case of excavation openings.

Regarding the modelling of excavations in jointed rock masses, to-date in the literature, block assemblies are generated based on discrete fracture networks (Dershowitz & Einstein, 1988) determined before the excavation is carried out. The engineer should be cautious against the degree of fracture propagation which is expected to take place during the process of excavation. The assumption of infinite joints as shown in Fig. 23 (a) is well-known to be overly conservative for stability considerations since the number of generated blocks is significantly larger than the actual case (Fig. 22(a)). So, if a block generation code

only able to generate persistent joints is employed, a different measure of joint intensity should be used (Dershowitz & Herda, 1992). Conversely, the removal of dangling joints (Fig. 22 (b)) increases the rock mass strength since there are fewer fractures than the actual case (Fig. 22 (a)). This may lead to an unsafe estimate of the stability of the excavation walls. However, if the intact rock is hard and dangling joints are unlikely to propagate, this could be a realistic estimate (cf. Kim et al., 2007). The number of blocks generated using the proposed rock slicing algorithm (Fig. 23 (c)) falls between the two extreme cases (Fig. 22 (b) and Fig. 23 (a)). Among the different options available for generating the rock mass, the engineer has to decide whether a generated block assembly is representative of the jointed rock mass for his stability analysis by in-situ monitoring as the excavation is carried out, and from his experience.

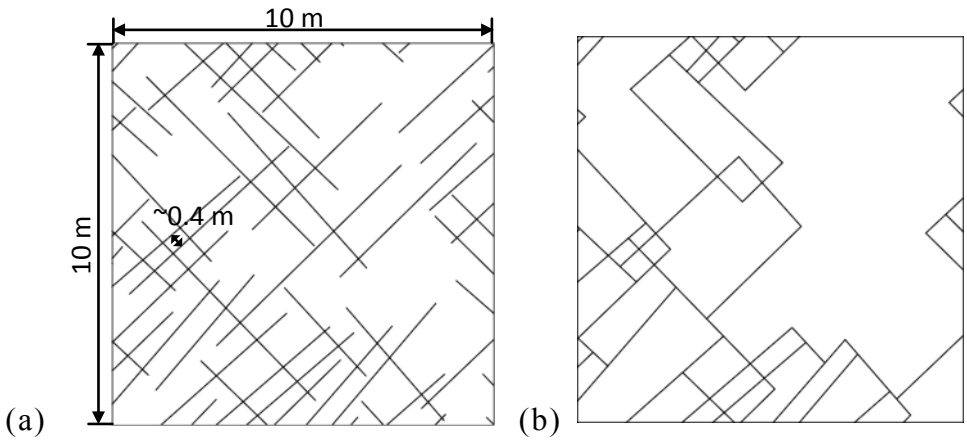


Fig. 22. Two dimensional joint pattern (a) input (b) model generated by UDEC (after Figure 8 in Kim et al. (2007))

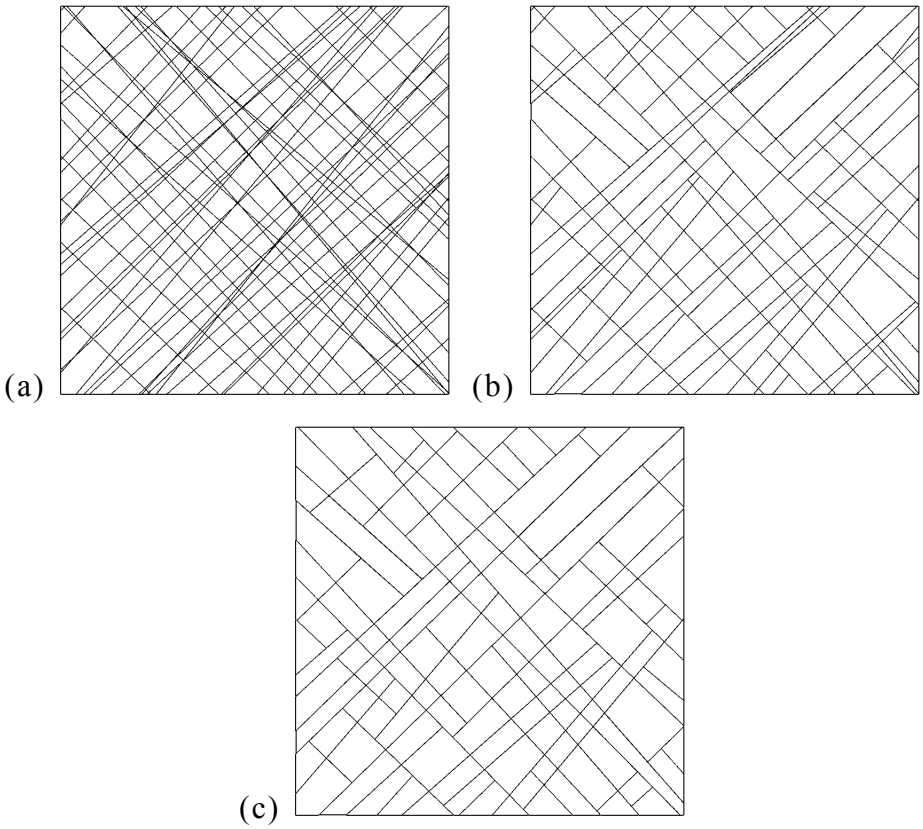


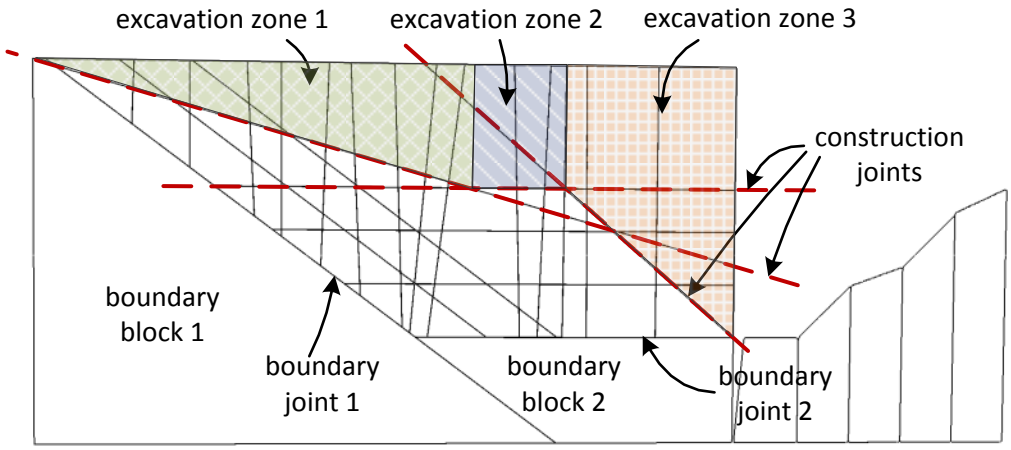
Fig. 23. Joint patterns generated using the proposed rock slicing algorithm (a) fully-persistent extents, (b) non-persistent extents (c) non-persistent extents enforcing a minimum block size of 0.4 m (largest diameter of inscribed circle)

3.4. Illustration of construction joints, concave blocks and non-persistent joints

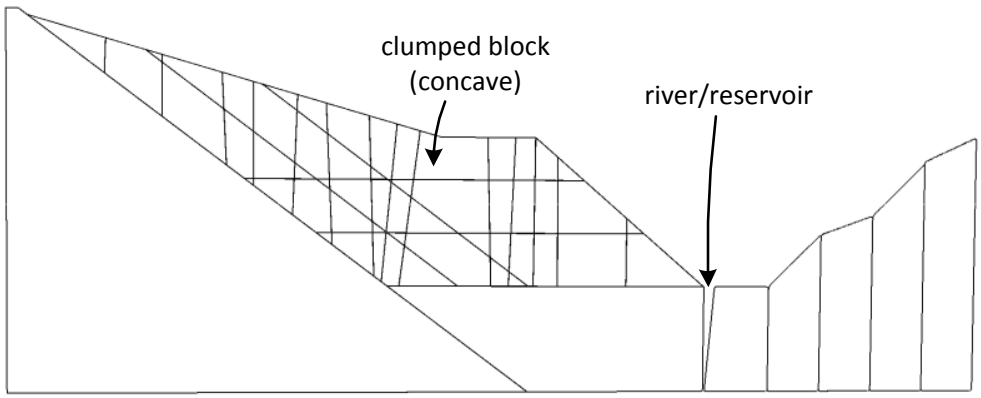
The Vajont rock slope whose instability led to a famous catastrophic slide in 1963 (Alonso & Pinyol, 2010; Müller-Salzburg, 2010) was selected as example. Fig. 24 shows the blocks generated using our proposed rock slicing method in a 2D section of the slope which underwent failure. Some of the key elements when generating a jointed rock mass are highlighted in Fig. 24 (a)-(c). First, two 'boundary' joints defining the slide surface were introduced (see Fig. 24 (a)). During the slicing calculation, the resulting child block which was located in the lower halfspace of the 'boundary' joint was automatically identified as a boundary block, so that it would not be subdivided by subsequent slices. Then, rock joints defining the rock mass were introduced. After the rock joints had been introduced,

construction joints (dashed red lines in Fig. 24(a)) were introduced to outline the free surface of the slope (see Fig. 24(b)), so that the blocks lying outside the slope profile may be removed. Blocks subdivided by construction joints were clustered (automatically) together by the imposition of a kinematic constraint preventing any relative movement between the two sides of the joints (see Fig. 24(c)) to avoid creating artificial planes of weakness which may unduly affect the mechanical response of the jointed rock mass. The excavation area outside the slope profile was divided into three separate excavation zones (see Fig. 24 (a)). All blocks falling within the specified excavation zones were then removed, as shown in Fig. 24 (b). Blocks located close to convex-shaped excavation openings or slope profiles are likely to be concave, and can be modelled based on this approach (Fig. 24 (b)). In some circumstances, it is desirable to control the extents of non-persistent joints to capture certain geometrical characteristics of the jointed rock mass (compare between Fig. 24 (b) and Fig. 24 (c)). In this example, it is desired to model bedding planes which are chair shaped and change abruptly at the 'seat' of the chair

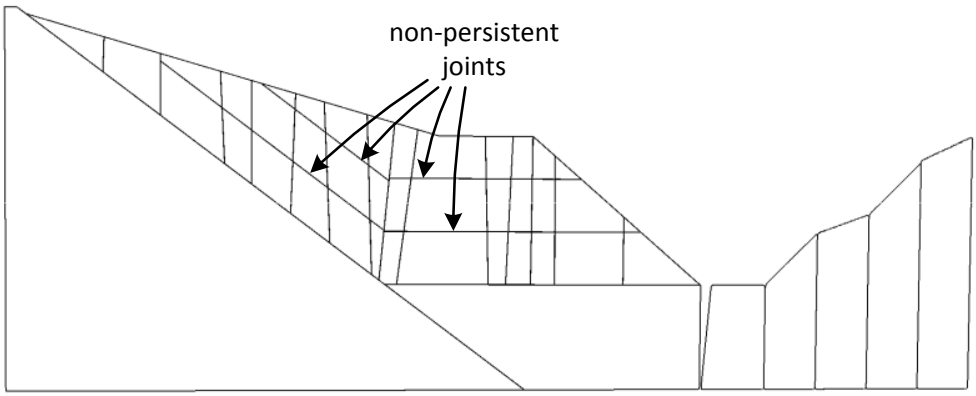
It is worthy to note that in the proposed method the increase in complexity when 3-D problems are considered rather than 2-D ones is minimal. In fact, the bookkeeping of data structures consists solely of the faces belonging to a polyhedron in 3-D, or the lines belonging to a polygon in 2-D. This is in contrast with the existing methods in the literature where the upgrade from 2-D to 3-D requires additional thorough code development (Warburton, 1985; Heliot, 1988; Yu et al., 2009).



(a)



(b)



(c)

Fig. 24. Rock slope (2-D section) generated from the new rock slicing method: (a) Construction joints were introduced to "outline" the free-surface of the rock slope. (b) Blocks whose centres are outside this "outline" were removed. Discontinuities in the model are persistent, i.e, through-going. Blocks subdivided by construction joints are clumped together. (c) Example of use of non-persistent joints.

4. Conclusions

In this paper, a novel rock slicing method which makes use of a single level data structure, consisting of only block faces, is introduced. As a consequence, the managing and updating of the block data structure for sequential subdivision becomes significantly more tractable. The main steps of the proposed method can be summarised as follows: (i) check whether there is intersection between a non-persistent joint plane and a block; (ii) if there is intersection, append the joint plane to each of the subdivided child block; (iii) at the end of the rock slicing process, identify and remove the geometrically redundant planes which do not form a block face. Unlike current methods in the literature, the updating of vertices and edges as a block is subdivided is no longer necessary. The use of a simpler data structure presents obvious advantages in terms of code development, ease of maintenance, and robustness (the updating of data structure being far less sensitive to rounding errors, which are not amplified with the sequential progression of the slicing). Another distinctive advantage of the proposed method is the fact that the increased complexity of a 3-D analysis by comparison to a 2-D one is minimal.

The rock slicing methodology here presented based on a single level data structure makes use of the mathematical theory of linear programming. The identification of blocks was cast as a set of linear programming optimization problems which can be solved efficiently using standard software for linear programming, such as CPLEX (2003). Non-persistency of joints was accounted via adding constraints into the linear program.

Because the computation time for block identification is minimal compared to the simulation runtime of the physical problems using DEM or DDA (e.g. underground excavations or stability analysis of rock slopes), robustness in terms of code implementation is more important. In the algorithm proposed in this paper, problems related to incompatible hierarchical data structures, i.e. vertices, edges or faces not joining correctly, are eliminated.

Nevertheless, we have shown that the rock slicing algorithm scales linearly with the number of generated blocks, when used together with bounding spheres. This feature is highly desirable for the algorithm to be computationally efficient also in the case of problems involving a large number of blocks.

The new rock slicing method has been coded into a series of C++ routines (see the supplementary material) and was applied to generate block assemblies in both 2-D and 3-D domains. Also DEM analyses of complex jointed rock masses can be carried out without relying on vertices and edges of the polyhedral blocks in the rock masses for a variety of problems (Boon et al., 2014a; 2014b) when the contact detection algorithm proposed by Boon *et al.* (2012) is employed.

References

1. Alonso, E. E., & Pinyol, N. M. Criteria for rapid sliding I. A review of Vaiont case. Engineering Geology 2010; **114**(3-4): 198-210.
2. Baecher, G. B. Statistical analysis of rock mass fracturing. Journal of the International Association for Mathematical Geology 1983; **15**(2) : 329-348.
3. Boon, C.W., Houlsby, G.T., Utili, S. A new algorithm for contact detection between convex polygonal and polyhedral particles in the discrete element method. *Computers and Geotechnics*, 2012; **44**: 73-82.
4. Boon, C.W., Houlsby, G.T, Utili, S. A new contact detection algorithm for three dimensional non-spherical particles. *Powder Technology*, SI on DEM, 2013; **248**: 94-102.
5. Boon, C. W., Houlsby ,G.T., Utili, S. Designing tunnel support in jointed rock masses via the DEM. *Rock Mechanics and Rock Engineering* 2014 a, Available online.

6. Boon, C.W., Houlsby, G.T., Utili, S. New insights in the 1963 Vajont slide using 2D and 3D distinct element method analyses. *Geotechnique* 2014 b, _Doi: 10.1680/geot.14.P.041.
7. Boon, C.W. Distinct element modelling of jointed rock masses: algorithms and their verification. D.Phil. University of Oxford 2013.
8. Boyd, S.P., & Vandenberghe, L. Convex Optimization. 2004; Cambridge University Press, pp. 716.
9. Caron, R. J., McDonald, J. F., & Ponic, C. M. A degenerate extreme point strategy for the classification of linear constraints as redundant or necessary. Journal of Optimization Theory and Applications 1989; **62**(2), 225-237.
10. 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. *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* 1988; **25**(3): 107-116.
11. CPLEX (2003). User's manual: IBM ILOG CPLEX 9.0.
12. Dershowitz, W. S., & Einstein, H. H. Characterizing rock joint geometry with joint system models. *Rock Mechanics and Rock Engineering*, 1988; **21**(1): 21-51.
13. Dershowitz, W., & Herda, H. H. (1992). Interpretation of fracture spacing and intensity. Paper presented at the U.S. Symposium on Rock Mechanics.
14. Einstein, H.H., Veneziano, D., Baecher, G.B., O'Reilly, K.J. The effect of discontinuity persistence on rock slope stability. *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* 1983; **20**(5): 227-236.
15. Elmouttie, M., Poropat, G., & Krähenbühl, G. Polyhedral modeling of rock mass structure. *International Journal of Rock Mechanics and Mining Sciences* 2010; **47**(4): 544-552.

16. Ghaboussi, J. & Barbosa, R. Three-dimensional discrete element method for granular materials. *International Journal for Numerical and Analytical Methods in Geomechanics* 1990, **14**(7), 451-472.
17. Harkness, J. Potential particles for the modelling of interlocking media in three dimensions. *International Journal for Numerical Methods in Engineering* 2009; **80**(12): 1573-1594.
18. Heliot, D. Generating a blocky rock mass. *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* 1988; **25**(3): 127-138.
19. Hoek, E., Kaiser, P.K., Bawden, W.F. Support of underground excavations in hard rock, 1995; Taylor & Francis, pp. 1-215.
20. Houlsby, G. T. Potential particles: a method for modelling non-circular particles in DEM. Computers and Geotechnics 2009; **36**(6): 953-959.
21. Hudson, J. A. Design methodology for the safety of underground rock engineering. Journal of rock mechanics and geotechnical engineering; 2012, **4**(3), 205-214.
22. Ikegawa, Y., & Hudson, J. A. Novel automatic identification system for three-dimensional multi-block systems. *Engineering computations* 1992; **9**(2): 169-179.
23. Itasca. FLAC Fast Lagrangian Analysis of Continua, ver. 4. 2000; Itasca Consulting Group Inc. Minneapolis, MN.
24. Itasca. UDEC Universal distinct element code. User manual. 2006; Itasca Consulting Group Inc. Minneapolis, MN.
25. Itasca. 3DEC 3-dimensional distinct element code, ver 4.1. 2007; Itasca Consulting Group Inc. Minneapolis, MN.
26. Itasca. 3DEC 3-dimensional distinct element code, ver 5.0. 2013; Itasca Consulting Group Inc. Minneapolis, MN.

27. Ivanova, V. M. Stochastic and topological fracture geometry model. M.S. Thesis. Department of Civil and Environmental Engineering, MIT 1995.

28. Jing, L. Block system construction for three-dimensional discrete element models of fractured rocks. *International Journal of Rock Mechanics and Mining Sciences* 2000; **37**(4): 645-659.

29. Kim, B. H., Cai, M., Kaiser, P. K., & Yang, H. S. Estimation of Block Sizes for Rock Masses with Non-persistent Joints. *Rock Mechanics and Rock Engineering* 2007; **40**(2), 169-192.

30. Kozicki, J., & Donzé, F. V. A new open-source software developed for numerical simulations using discrete modeling methods. [doi: DOI: 10.1016/j.cma.2008.05.023]. *Computer Methods in Applied Mechanics and Engineering* 2008; **197**(49-50): 4429-4443.

31. Kulatilake, P. H. S. W., Ucpirti, H., Wang, S., Radberg, G., & Stephansson, O. Use of the distinct element method to perform stress analysis in rock with non-persistent joints and to study the effect of joint geometry parameters on the strength and deformability of rock masses. *Rock Mechanics and Rock Engineering* 1992; **25**(4): 253-274.

32. Lee, J.-S. Stochastic and topological fracture geometry model. M.S. Thesis. Department of Civil and Environmental Engineering, MIT 1990.

33. Li, J., Xue, J., Xiao, J., & Wang, Y. Block theory on the complex combinations of free planes. *Computers and Geotechnics* 2012; **40**: 127-134.

34. Lin, D., Fairhurst, C., & Starfield, A. M. Geometrical identification of three-dimensional rock block systems using topological techniques. [doi: DOI: 10.1016/0148-9062(87)92254-6]. *International Journal of Rock Mechanics and Mining Science & Geomechanics Abstracts* 1987; **24**(6): 331-338.

35. Lu, J. Systematic identification of polyhedral rock blocks with arbitrary joints and faults. *Computers and Geotechnics* 2002; **29**(1): 49-72.

36. Mandl, G. *Rock Joints: The Mechanical Genesis*. Springer-Verlag, Berlin Heidelberg 2005; pp. 221.

37. Müller-Salzburg, L. The Vajont slide. *Engineering Geology* 1987; **24**(1-4): 513-523.

38. Pine, R. J., Coggan, J. S., Flynn, Z. N., & Elmo, D. The Development of a new Numerical Modelling Approach for Naturally Fractured Rock Masses. *Rock Mechanics and Rock Engineering 2006*, **39**(5): 395-419.

39. Pollard, D. D. & Aydin, A. Progress in understanding jointing over the past century. *Geological Society of America Bulletin* 1988, **100**: 1181-1204.

40. Priest, S.D. Discontinuity analysis for rock engineering. 1993; Chapman & Hall, pp. 1-473.

41. Priest, S.D. & Hudson, J. A. Discontinuity spacings in rock. *International Journal of Rock Mechanics and Mining Science & Geomechanics Abstracts* 1976; **13**: 135-148.

42. Shi, G. H., & Goodman, R. E. Two dimensional discontinuous deformation analysis. *International Journal for Numerical and Analytical Methods in Geomechanics* 1985; **9**(6): 541-556.

43. Tsesarsky, M., & Talesnick, M.L. Mechanical response of a jointed rock beam. Numerical study of centrifuge models. *International Journal for Numerical and Analytical Methods in Geomechanics* 2007; **31**(8), 977-1006.

44. Utili S., Crosta G.B. (2011a). Modelling the evolution of natural slopes subject to weathering: Part I. Limit analysis approach. *Journal of Geophysical Research – Earth Surface,* **116**, F01016, doi:10.1029/2009JF001557.

45. Utili S., Crosta G.B. (2011b) Modelling the evolution of natural slopes subject to weathering: Part II. Discrete element approach *Journal of Geophysical Research – Earth Surface,* **116**, F01017, doi:10.1029/2009JF001559.

46. Wang, S. *Fundamental studies of the deformability and strength of jointed rock masses at three dimensional level*. Ph.D. thesis, The University of Arizona 1992.

47. Warburton, P. M. A computer program for reconstructing blocky rock geometry and analyzing single block stability. *Computers & Geosciences* 1985; **11**(6): 707-712.

48. Yu, Q., Ohnishi, Y., Xue, G., & Chen, D. A generalized procedure to identify three-dimensional rock blocks around complex excavations. *International Journal for Numerical and Analytical Methods in Geomechanics* 2009; **33**: 355-375.

49. Zadhesh, J., Jalali, S. M. E., & Ramezanzadeh, A. Estimation of joint trace length probability distribution function in igneous, sedimentary, and metamorphic rocks. *Arabian Journal of Geosciences* 2013; 1-9.

50. Zhang, L., Einstein, H. H., & Dershowitz, W. S. Stereological relationship between trace length and size distribution of elliptical discontinuities. *Geotechnique* 2002; **52**(6): 419-433.

51. Zhang, L., & Einstein, H. The Planar Shape of Rock Joints. *Rock Mechanics and Rock Engineering* 2010; **43**(1): 55-68.

52. Zhang, Y., Xiao, M., Ding, X., & Wu, A. Improvement of methodology for block identification using mesh gridding technique. *Tunnelling and Underground Space Technology* 2012; **30**: 217-229.

53. Zhang, Z. X., & Lei, Q. H. Object-oriented modeling for three-dimensional multi-block systems. *Computers and Geotechnics* 2013; **48**: 208-227.