ARMA 13-416

Explicit Three Dimensional Discontinuous Deformation Analysis for Blocky System

Roozbeh Geraili Mikola,
Jacobs Associates and Dept. of Civil and Environmental Engineering, UC Berkeley, San Francisco, CA, USA
Nicholas Sitar,
Edward G. Cahill and John R. Cahill Professor, Dept. of Civil and Environmental Engineering, UC Berkeley, Berkeley, CA, USA

This paper was prepared for presentation at the 47th US Rock Mechanics / Geomechanics Symposium held in San Francisco, CA, USA, 23-26 June 2013.

ABSTRACT: We present a 3D-DDA formulation that uses an explicit time integration procedure and an efficient contact detection algorithm optimized to minimize the computational effort. The advantages of the explicit formulation are that the global stiffness matrix does not need to be assembled and the linear equations do not need to be solved by matrix inversion. Consequently, the computational effort and memory requirement can be reduced considerably, which is important for efficient solution of large 3D problems. In addition, the computational efficiency is increased by eliminating unnecessary contact computations using a grid based nearest neighbor search. The grid divides space into a number of cells of equal size and each object is then associated with the cells it overlaps. As only objects overlapping a common cell can possibly be in contact, in-depth tests are only performed on objects found sharing cells with the block tested for collision. The contacts between the blocks are detected by using Fast Common-Plane (FCP) approach. The halfedge (HE) data structure approach is used to handle the navigation into the topological information associated with polyherdral objects (vertices, edges, faces). The halfedge data structure allows for quick traversal between faces, edges, and vertices due to the explicitly linked structure of the network. Examples are provided which demonstrate the capabilities of new algorithm and the size of problem that can be analyzed.

1. INTRODUCTION

The discontinuous deformation analysis (DDA) is a numerical method for blocky systems [1]. Since its introduction by Shi [1] 2-D DDA has been extensively developed in theory and computer codes. However, the highly directional nature of jointed rock mass limits the application of 2-D DDA to many practical problems, due to importance of 3-D effects. As a result there has been a significant interest in extending the formulation to 3-D. Shi [2,3] presented the 3-D block matrices such as mass matrix, stiffness matrix, point load matrix, body load matrix, initial stress matrix and fixed point matrix. Jang and Lee [4] developed a 3-D DDA technique. One of the challenges is managing contacts in 3-D. Yeung et al. [5] and Jiang and Yeung [6] developed a point-to-face model for 3-D DDA, but their model did not consider the vertex-to-vertex, vertex-to-edge and edge-to-edge contact modes. Liu and Kong [7] and Liu et al. [8] presented a 3-D DDA framework for dealing with 3-D mechanical interactions of blocks, using the 'common-plane' technique. Wu et al. [9] developed a new contact search algorithm for frictionless vertex-to-face contact problems, and presented the 3-D DDA formulation for normal contact force. Wu et al. [10] then applied the 3-D DDA to a slope toppling analysis at Amatoribashi-nishi in Japan. Yeung et al. [11] and Wu [12] further extended the algorithms to edge-to-edge contacts. More recently, Beyabanaki et al. [13] presented a new algorithm to search and calculate geometrical contacts in the 3-D DDA.

Computationally, the original 3D-DDA [1,2,3] formulation uses the piece-pair contact detection method (brute force method), and the performance decreases as the number of blocks increases as O(n2). Implicit time-marching scheme is used to integrate the equation of motion [1]. This requires that the global equations are solved iteratively by repeatedly adding and removing contact springs (i.e. open-close iteration) until each of the contacts converges to a constant state at each time step. If contact convergence is not achieved, typically within six iterations, the time step is reduced and the analysis is repeated with the reduced time step. Also the incremental displacement is restricted by user-specified displacement limit to enforce infinitesimal displacements. If the incremental displacement is greater than the threshold, Δt is decreased and the analysis is

repeated. Large values of Δt may cause large penetrations at contact points; which result in more iterations to satisfy the penetration threshold. Also, large penetrations result in large contact matrices which can reduce the diagonal dominance of the global stiffness matrix leading to poorly conditioned system of equations. Khan [15] concluded that, although a larger time step is specified in DDA, the average time-step size is much smaller than that of Discrete Element Method (DEM) with explicit time integration. Frequent time step cuts not only decrease the Δt but also increase cumulative open-close iterations.

Thus, in order to reduce the amount of collision checks by the piece-pair method, we introduce an efficient method, the uniform spatial discretization method, to accelerate the contact detection based on the Fast Common Plane (FCP) method. Neighboring-cell method divides the workspace into a grid of cells. Each cell maintains a list of the blocks contained within that cell. For a given block, it only needs to check for contact with other blocks in its own cell and neighboring cells.

The FCP method was introduced by Nezami et al. [16]. In this approach, a common-plane is selected as the reference plane for vertex-to-vertex, vertex-to-edge and edge-to-edge contacts in the 3D DDA. We utilize the halfedge (HE), a well-known compact adjacency-based topological data structure, to navigate into the topological information associated with polyhedral object (vertices, edges, faces) more efficiently.

Our formulation also uses an explicit solution procedure to reduce the computational effort and memory requirements. In contrast to the implicit time integration scheme, the explicit solution scheme eliminates the need for assembly of global mass or stiffness matrices and inversion of the global matrix.

The resulting code was programmed in C++ and the animation frames were rendered using POVray, a free code ray tracing free rendering program. The performance of the program is demonstrated through a series of examples in 3-D.

2. CONTACT DETECTION USING UNIFORM GRID

Contact detection and resolution is the most time-consuming part of DEM/DDA analyses and generally takes about 80% of the overall computational time for particles [17]. Contact resolution and detection is commonly performed in two consecutive phases [18], namely, neighbor search and contact detection (Figure 1). Neighbor search phase develops a neighbor list of all potential interacting particles within a neighborhood of the target particle. To speed up the contact detection process a problem region is divided into a number of cells and blocks are "mapped" into cells according to the locations of their vertices. The contact detection is carried out only between potential block vertices contained in each cell (Figure 1). Re-mapping of cells is triggered whenever one block moves outside its original cell space. For example in Figure 1, block A's box list includes boxes 7,8,12, and 13; and box B's block list includes particles 14, 15, 19, 20, 24 and 25 and block C's box list includes boxes 4, 5, 9, 10, 14 and 15. These lists are obtained by defining a cuboid bounding volume around each particle and comparing it against the boxes. In this paper the faces of the cuboid bounding volume are confined to be parallel to coordinate planes of the global coordinate system.

The performance of neighbor search algorithm is dependent on the particle shape and the ratio of the box size S to the average bounding box size D_{50} [16]. Nezami et al. [16] suggest that contact search optimal performance correspond to the approximately S/D_{50} = 1.5 [16].

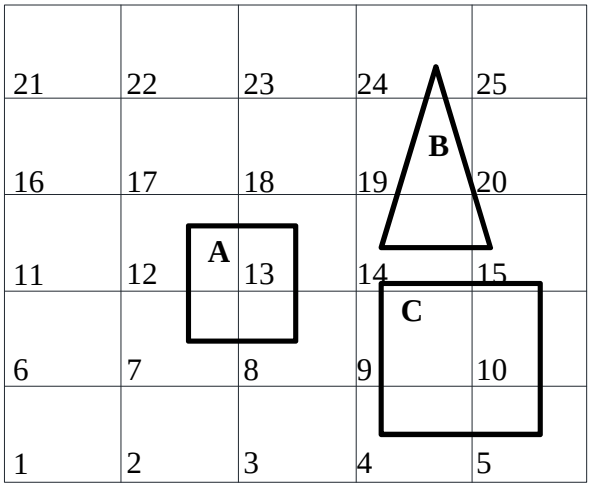


Figure 1. Cell mapping of block (1,2,3,…-Cell Number and A,B,C-Blocks).

3. CONTACT RESOLUTION USING FCP ALGORITHM

After a search for the colliding objects is made for all the discrete blocks, the next step is to find the points of intersection on the sides of the home element with the sides of the near element. Common-plane (CP) algorithm was introduced by Cundall [19] in order to reduce the expensive object-to-object contact detection problem to a less expensive plane-to-object contact problem in DEM. The CP is defined as a plane bisecting the space between the objects. After CP has been located each object is tested separately for contacts with the common-plane. Nezami et al. [16] proposed the fast common plane (FCP) method in which they improved the original CP algorithm by adding a fast method to identify the right candidates for the plane. In FCP, the number of iterations is significantly reduced by limiting

the search space of the CP to a few candidates. In this paper, the idea of the FCP method is applied to the contact detection and the common-plane is selected as the reference plane for vertex-to-vertex, vertex-to-edge and edge-to-edge contacts in the 3D DDA.

The FCP algorithm to find the CP consists of five steps as described below [16]:

(i) If there is a CP from the previous time step then use it as the initial guess for the CP in this time step. Otherwise, set the CP as the perpendicular bisector (PB) plane of the line connecting the centroids of the two blocks.

(ii) Based on the guess CP, find the closest vertices A and B in two contacting blocks. If more than one pair of closest vertices have the shortest length (i.e., those vertices are equidistant), then any of them can be chosen to proceed with the algorithm.

(iii) For the two closest vertices A and B found in Step 2, check all candidate planes of and find the one with the largest gap or smallest overlap. The candidate planes could be on the plane listed below:

(a) Type a: The PB plane of segment AB.

(b) Type b: The plane passing through the midpoint of segment AB parallel to one of the faces of particles A or B. For particle A, only faces which include the vertex A are considered. For particle B, only faces which include the vertex B are considered.

(c) Type c: The plane passing through the midpoint of segment AB parallel to one edge from particle A and one edge from particle B. For particle A, only those edges which share the vertex A are considered. For particle B, only those edges which share the vertex B are considered.

(d) Type d: The plane passing through the midpoint of segment AB parallel to one edge from one of the particles.

(iv) If the CP obtained in Step 3 is the same as the one from Step (ii), then it is the correct common plane. Otherwise go to Step (ii). This is an iterative algorithm, with consisting of steps (ii)–(iv) in each iteration. The number of iterations required to find the CP is usually very small and the position of the selected CP is accurate. This is mainly because the iteration is done to locate the two closest vertices, rather than the CP itself.

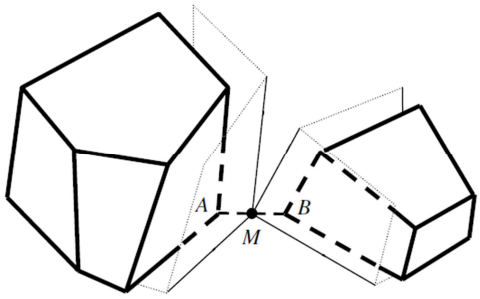


Figure 2. Possible CP for colliding blocks.

4. COMPACT DATA STRUCTURE (HALFEDGE)

FCP algorithm needs access to topological information such as faces and edges sharing the same vertices A and B, as explained in section above. The simplest way of storing the information is a data structure that explicitly stores all topological entities, and all the adjacency relationships among them, tends to perform queries very efficiently, but this demands high, sometimes prohibitive, storage space. Moreover, editing tasks may demand high computational efforts because several adjacency relationships have to be updated. In this paper, authors use the compact adjacency-based topological data structure for representing the polyhedral block so called HalfEdge (HE). The HE data structure allows adjacency information to be found in near real time, making it especially useful for FCP algorithm where frequent access is needed to different types of adjacency information and also reduces the redundancy.

The main idea of the half-edge is to ”split” each edge along its length and store pointers to its previous and next edges, see Figure 3. The full implementation in C++ can be seen in Algorithm 1 [20].

A face is not defined explicitly instead it can be found by using the next or previous pointers. With this data structure, finding neighboring faces of a vertex is O(1) and for a complete mesh O(n) and the topological queries of the polygon mesh can be performed easily and quickly. The time complexity of these queries is linear in the amount of information gathered and independent of global complexity. For example, iterating over the faces adjacent to a vertex is requires the following sequence of steps [20]:

(i) Find an edge connected to the given vertex.

(ii) Step out on the edge loop and insert the face connected to this edge.

(iii) Use edge- > next- > pair- > next to traverse to next face and insert this face into the temporary vector.

(iv) Repeat step (iii) until encountering the face inserted in step (i).

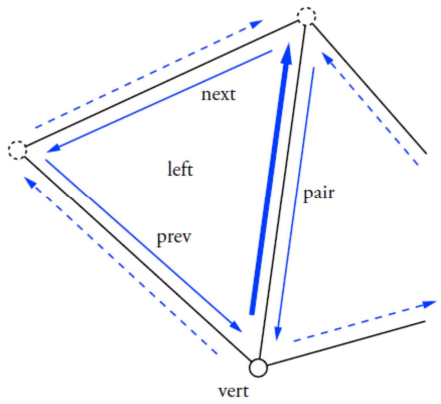


Figure 3. The half-edge data structure as seen from the bold half-edge.

Algorithm 1: Half-edge data structure.

```
class Halfedge  {  // topology
public:
        Vertex* vert ;
        Halfedge* next ;
        Halfedge* prev ;
        Halfedge* pair ;
        Face* left ;
} ;
class Vertex {  //geometry
public:
        float x , y , z ;
        Halfedge* edge ;
} ;
class Face {
pulic:
        Halfedge* edge ;
};
class Polyhedron {
public:
        Vertex verts [V] ;
        Face faces [F] ;
        Halfedge edges [3F] ;
} ;
```

5. EXPLICIT TIME INTEGRATION SCHEME

Let D_n and D_{n+1} denote the approximation to the values $D(t)$ and $D(t+1)$ for a time step Δt, respectively. Recall the system of equations Eq. 1 of motion for a DDA system [1]:

$$M\ddot{D}_{n+1} + C\dot{D}_{n+1} + KD_{n+1} = F_{n+1} \quad (1)$$

with $D(0) = 0, \dot{D}(0) = \dot{D}_0$ as initial boundary conditions. In the above M, C, K are the global mass, damping and stiffness matrices, F is the time dependent applied force vector, and $\ddot{D}$, $\dot{D}$, D and denote acceleration, velocity and displacement vectors, respectively.

Original DDA time integration scheme adopts the Newmark [21] approach, which for a single degree of freedom can be written in the following manner:

$$u_{i+1} = u_i + \Delta t\dot{u}_i + \left(\frac{1}{2} - \beta\right)\Delta t^2\ddot{u}_i + \beta\Delta t^2\ddot{u}_{i+1} \quad (2)$$

$$\dot{u}_{i+1} = \dot{u}_i + (1-\gamma)\Delta t\ddot{u}_i + \gamma\Delta t\ddot{u}_{i+1} \quad (3)$$

Where $\ddot{u}, \dot{u}$, and u are acceleration, velocity, and displacement respectively, Δt is the time step, β and γ are the collocation parameters defining the variation of acceleration over the time step. Unconditional stability of the scheme is assured for $2\beta \geq \gamma \geq 0.5$. DDA integration scheme uses $\beta = 0.5$ and $\gamma = 1$, thus setting the acceleration at the end of the time step to be constant over the time step. This approach is implicit and unconditionally stable. Substituting Eqs. 2 and 3 into Eq. 1 results in the system of equations for solving the dynamic problem:

$$\left(\frac{2}{\Delta t^2}M + \frac{2}{\Delta t}C + K\right)D_{n+1} = F_{n+1} + \left(\frac{2}{\Delta t}M + C\right)\dot{D}_n \quad (4)$$

The solution of Eq. 4 requires assembling the global mass and stiffness matrices and solving the coupled system of equations using a direct matrix inverse operation or an iterative solver. The global stiffness matrix, K, includes the sub-matrix representing deformability of blocks and contacts, with contact matrices as off-diagonal terms.

In the original DDA program, within each time step, the global equations are solved iteratively by repeatedly adding and removing contact springs (penalty values) until each of the contacts converges to a constant state. This procedure of adding and removing contact springs (penalty values) is known as open-close iterations in the DDA literature [22]. If contact convergence is not achieved typically within six iterations, the time step is reduced and the analysis is repeated with the reduced time step. The incremental displacement is restricted also by user-specified displacement limit to enforce infinitesimal displacements. If the incremental displacement is greater than the threshold, Δt is divided by three and the analysis is repeated. Large values of Δt may cause large penetrations at contact points; which results in more iterations to satisfy the penetration threshold. Also, large penetrations result in large contact matrices which can reduce the diagonal dominance of the global stiffness matrix leading to poorly conditioned system of equations.

In the explicit solution procedure presented herein the discrete blocks are integrated explicitly by the central difference method, which gives

$$u_{i+1} = u_i + \Delta t_{i+1}\dot{u}_{i+1} \quad (5)$$

$$\dot{u}_{i+1/2} = \dot{u}_{i-1/2} + \frac{1}{2}(\Delta t_{i+1} + \Delta t_i)\ddot{u}_i \quad (6)$$

Where $i, i+1/2$ and $i-1/2$ refer to the increment number and mid-increment numbers

$$\ddot{u}_i = M^{-1}(F_i - I_i) \quad (7)$$

where M is mass matrix, F the applied load vector and I is the internal force vector. The equations relating these values to each other are solved locally for each time-step. Moreover, since there is no need to solve a complete system of equations, the incremental calculations for each degree of freedom are done independently at the local level. This uncoupling of the equations of motion is one of the major advantages of explicit integration schemes. In contrast to the implicit time integration scheme, the explicit solution scheme eliminates the need for assembly of global mass or stiffness matrices and inversion of the global matrix. However, computations are conditionally stable, i.e., the time-step size must be smaller than a certain critical value (critical time step, Δt_c) for numerical errors not to grow unbounded. The time increments must satisfy the well-known criterion

$$\Delta t \leq \frac{2}{\omega_{max}} \quad (8)$$

where ω_{max} is the element maximum eigenvalue.

6. SIMULATIONS

Three examples are presented to demonstrate the newly developed 3-D DDA algorithm. The scenes in the following examples have been rendered with POV-ray, a free code ray tracing rendering program [23].

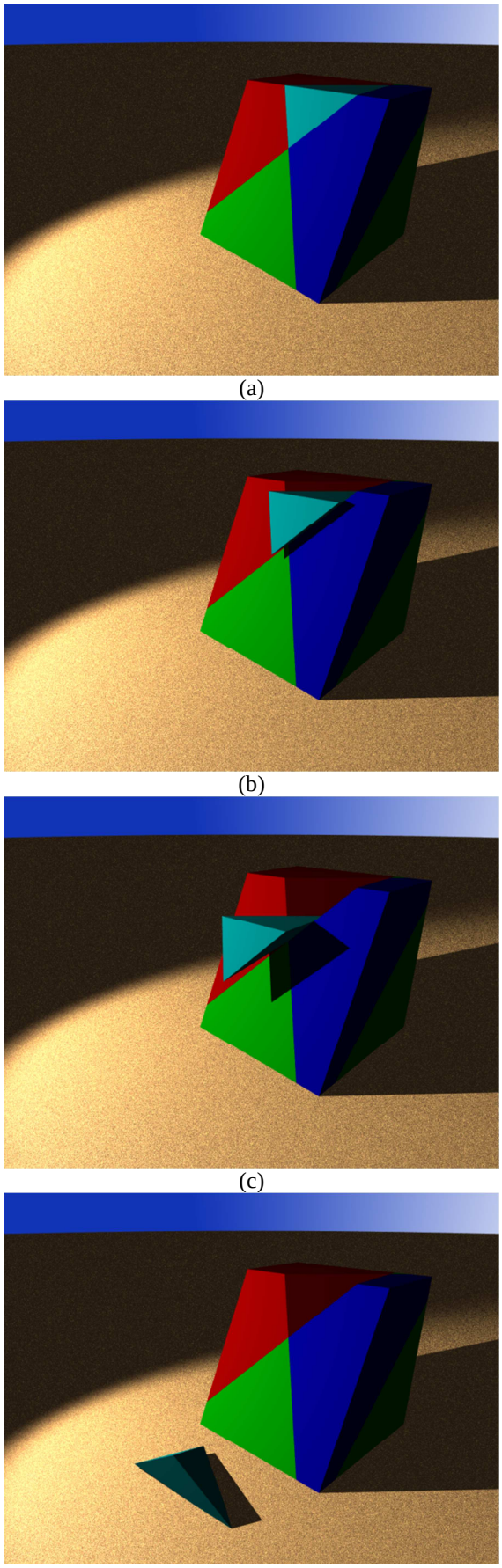
6.1. Example 1

In this simulation, the modified DDA code is used to simulate a sliding wedge problem typically encountered in rock engineering (see Figure 4). The model consists of four rigid blocks. Three blocks are fixed with only one wedge being allowed to move under gravity. The following discontinuities (rock joints) are present:

A: dip=45° dip direction = 5°

B: dip=55° dip direction = 135°

The slope face angle is 65° and the slope height is 10 meter. Unit weight of the block is 2.6 t/m^3. In this simulation cohesion and friction angle of the both joints are 0 and 25° respectively. The critical friction angle at which the free wedge starts sliding was sought. To simulate the problem numerically, a high coefficient of friction was first assigned to the discontinuities. Once the wedge reached static equilibrium, the friction is reduced by a small fraction. This process is repeated until static equilibrium is lost and the wedge starts sliding. The critical friction angle at which sliding occurs in the simulation is 19.6 °. The factor of safety of the wedge block can be calculated using Eq. 9.



(a) (b) (c) (d)

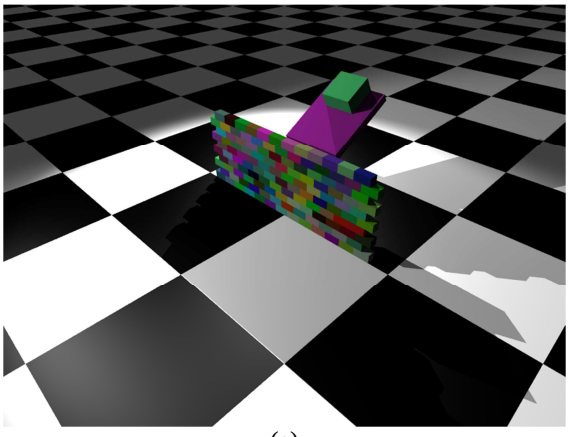
Figure 4. Sliding wedge example.

$$F.S. = \frac{\tan(\varphi_I)}{\tan(\varphi_F)} = \frac{\tan(25)}{\tan(19.6)} = 1.31 \quad (9)$$

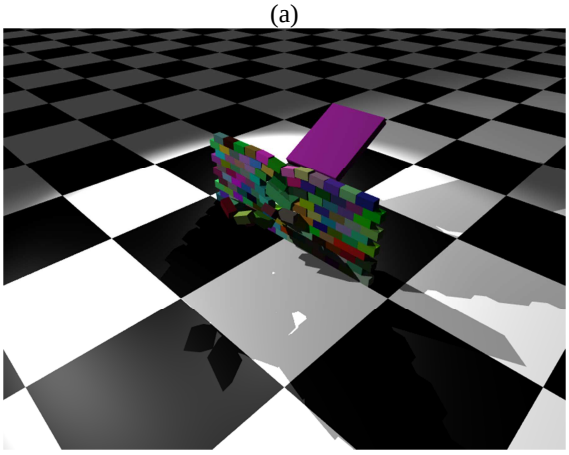
In which φ_I and φ_F are initial and reduced friction angles respectively. The analytical solution [24] for this problem estimates the factor of safety to be 1.302 which is in good agreement with the DDA result.

6.2. *Example 2*

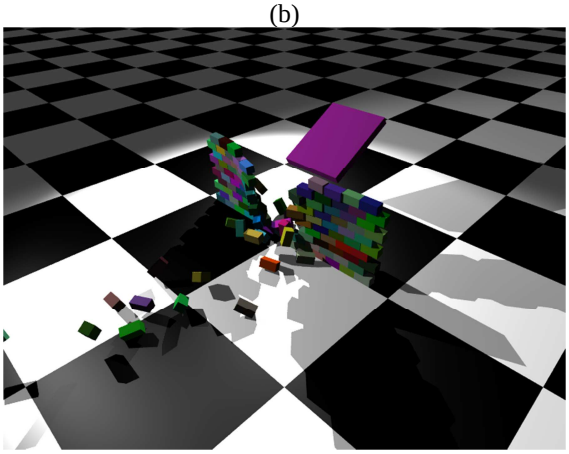
This example models a projectile that collides, penetrates and collapses a simulated brick wall. The brick wall stands on the floor (the ground). A block-like projectile slides on top of inclined block and impacts the wall. The wall consists of 10 layers and each layer has 13 bricks as shown in Figure 5.



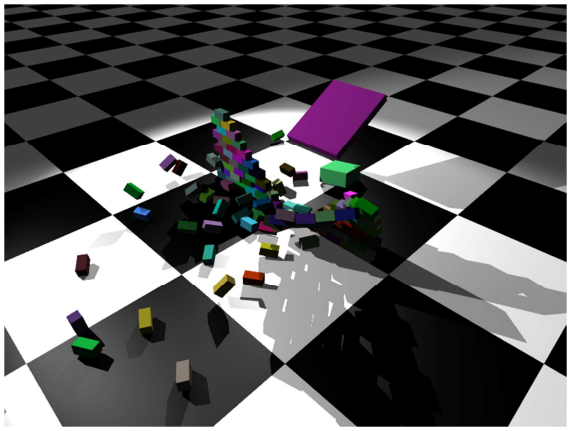
(a)



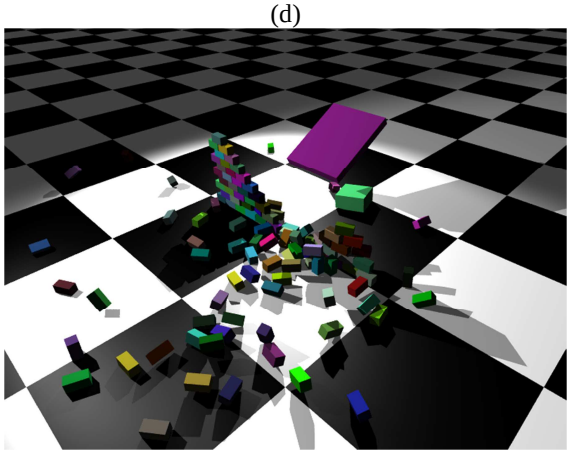
(b)



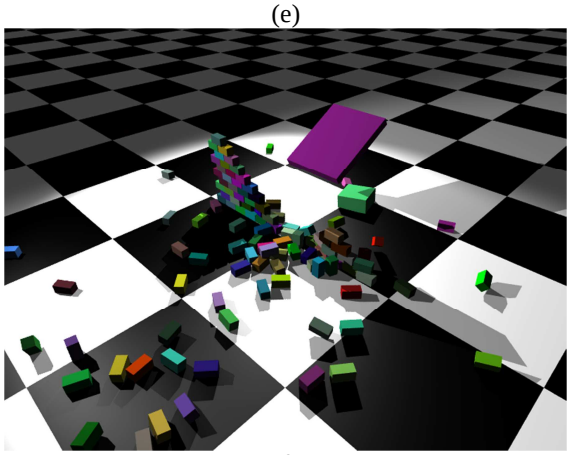
(c)



(d)



(e)



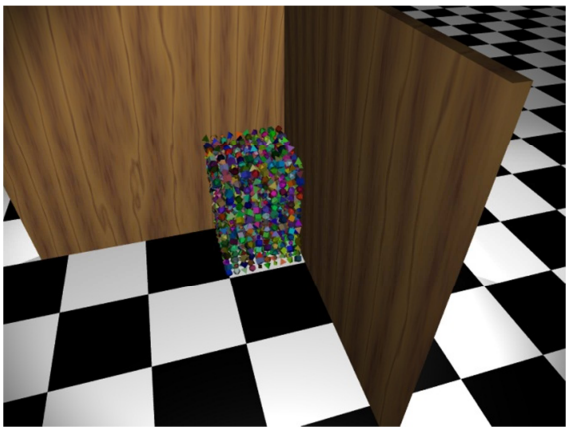
(f)

Figure 5. impact of single block with brick wall. (a) after 0 time step (b) after 1000 time steps (c) after 2000 time steps (d) after 3000 time step (e) after 4000 time steps (f) after 5000 time steps.

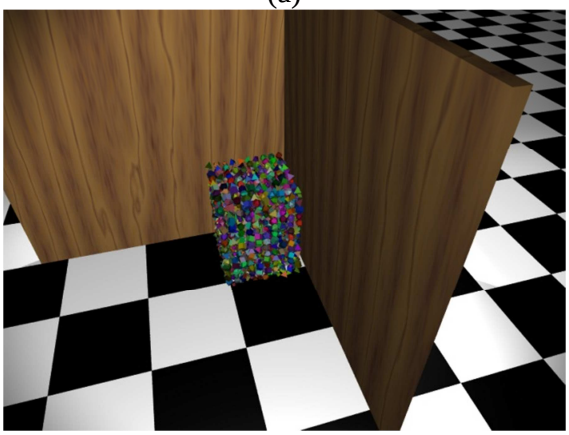
6.3. *Example 3*

The robustness of the algorithms was tested by simulating a variety of contact conditions for 3-D blocks. A loose packing of 1500 polyhedra was generated as shown in Figure 6. Each block was given an initial random orientation and shape (i.e. tetrahedron, pyramid, cube and octahedron). Then the blocks were allowed to fall under gravity to impact on a surface. This exercise is suitable to test the robustness of the algorithms of contact detection, since it involves particles colliding under free gravitational fall (dynamic

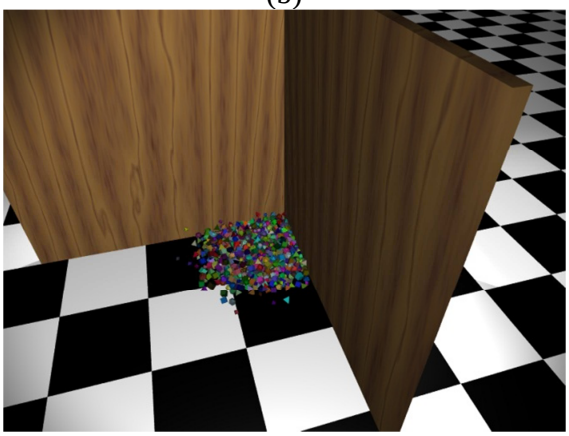
conditions). The particles then bounce against each other until they reach a final configuration at rest (quasi-static conditions). During the simulations, all types of contacts (e.g. face–face, edge–face, etc.) were encountered. Figure 5 shows the simulation after 0, 7000, 14000, 21000, 28000, and 35000 time steps. The computational time for this simulation was almost 20 minutes for 1 second of real time on a PC with 2.4 GHz CPU.



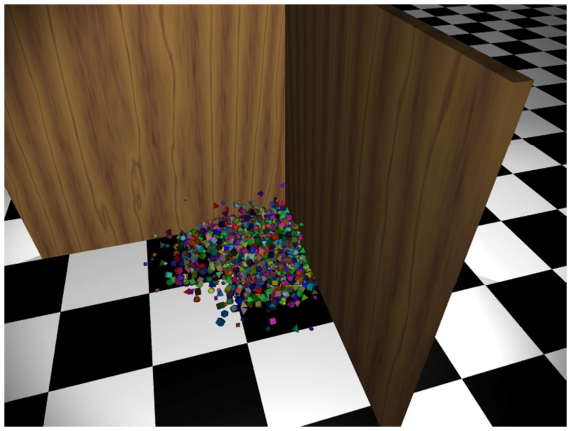
(a)



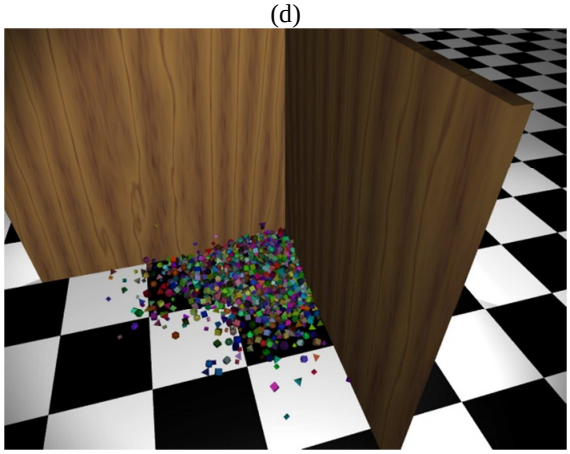
(b)



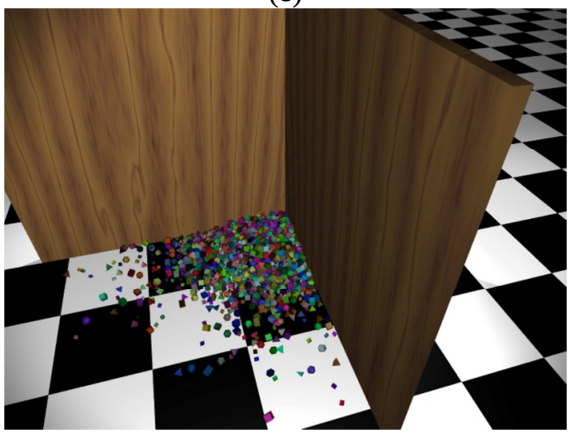
(c)



(d)



(e)



(f)

Figure 6. 1500 polyhedral blocks falling under gravity. (a) after 0 time steps (b) after 7000 time steps (c) after 14000 time steps (d) after 21000 time steps (e) after 28000 time steps (f) after 35000 time steps.

7. CONCLUSION

We present new explicit time integration procedure for the solution of 3D-DDA algorithm in order to reduce the computational effort and memory requirement. A uniform spatial discretization method is utilized to eliminate the unnecessary contact computations. The contact resolution is been handled by FCP approach and HalfEdge data structure is used to handle the frequent navigation into the topological information associated

with polyhedral blocks. The algorithms are programmed in C++. The presented simulations demonstrate the performance of the proposed algorithms under a variety of conditions and show that dynamic 3-D problems with significant number of blocks can be solved using typical desktop CPU's

ACKNOWLEDGMENT

This work was carried out with partial funding from Edward G. Cahill and John R. Cahill chair at the University of California, Berkeley.

REFERENCES

1. Shi GH. 1993. *Block system modeling by discontinuous deformation analysis*. Southampton UK and Boston USA: Computational Mechanics Publications.
2. Shi GH. 2001. Three-dimensional discontinuous deformation analysis. In: Bicenic N, editor. *Proceedings of the forth international conference on analysis of discontinuous deformation (ICADD-4)*, Glasgow, Scotland, June 6–8, p. 1–21.
3. Shi GH. 2001. Three-dimensional discontinuous deformation analysis. In: Ellsworth et al., editors. *Proceedings of the 38th US rock mechanics symposium*, Washington DC, July 7–10, p. 1421–8.
4. Jang HI, Lee CI. 2002. Development of a three-dimensional discontinuous deformation analysis technique and its application to toppling failure. In: Hatzor YH, editor. *Proceedings of the 5th international conference on analysis of discontinuous deformation (ICADD-5)*, Abingdon, p. 225–9.
5. Yeung MR, Jiang QH, Sun N. 2003. Validation of block theory and three dimensional discontinuous deformation analysis as wedge stability analysis methods. *Int. J. Rock Mech. Min. Sci*. 40(2):265–75.
6. Jiang QH, Yeung MR. 2004. A model of point-to-face contact for three dimensional discontinuous deformation analysis. *Rock Mech. Rock Eng.*37:95–116.
7. Liu J, Kong X. 2003. Development of three-dimensional discontinuous deformation analyses. In: Lu M, editor. *Proceedings of the 6th international conference on analysis of discontinuous deformation (ICADD-6)*, Norway,p. 45–54.
8. Liu J, Kong X, Lin G. 2004. Formulation of the three-dimensional discontinuous deformation analysis method. *Acta. Mech. Sinica.* 20(3):270–82. June.
9. Wu JH, Ohnishi Y, Shi GH, Nishiyama S. 2005. Theory of three dimensional discontinuous deformation analysis and its application to a slope toppling at Amatoribashi, Japan. *Int J Geomech*179–95.
10. Wu JH, Juang CH, Lin HM. 2005.Vertex-to-face contact searching algorithm for three-dimensional frictionless contact problems. *Int. J Numer. Methods. Eng.*63(6):876–97.
11. Yeung MR, Jiang QH, Sun N. 2007. A model of edge-to-edge contact for three-dimensional discontinuous deformation analysis. *Comput. Geotech.*34(3):175–86.
12. Wu JH, Ohnishi Y, Shi GH, Nishiyama S. 2005. Theory of three-dimensional discontinuous deformation analysis and its application to a slope toppling at Amatoribashi, Japan. *International Journal of Geomechanics* 5(3):179–195.
13. Beyabanaki SAR. ,Mikola RG, Hatami K. 2008. Three-dimensional discontinuous deformation analysis (3-DDDA) using a new contact resolution algorithm. *Comp. Geotech.*35:346–56.
14. Grayeli R, Hatami K, 2008. Implementation of the finite element method in the three-dimensional discontinuous deformation analysis (3D-DDA), *International Journal for Numerical and Analytical Methods in Geomechanics*, 32:1883–1902.
15. Khan, MS., 2010. *Investigation of discontinuous deformation analysis for application in jointed rock masses*, Ph.D. Dissertation, University of Toronto,.
16. Nezami E, Hashash YMA, Zhao D, Ghaboussi J, 2004. A fast contact detection algorithm for 3-D discrete element method. *Comput. Geotech.*31(7):575–87.
17. Horner, DA., Carrillo A, et al. 2000. Very large scale coupled discrete element-finite element modeling for simulating excavation mechanics. *Fourteenth Engineering Mechanics Conference*, American Society of Civil Engineerings, Austin, Texas.
18. Perkins E. and JR. Williams 2001. A fast contact detection algorithm insensitive to object sizes. *Engineering Computations* 18(1-2): 48-62.
19. Cundall PA. 1988. 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 Mech. Min. Sci. & Geomech. Abstr*.25(3):107–16.
20. Gustav G, 2010. The Half-edge data structure, *http://www.gustavgahm.com/wp-content/upload/moa/Gustav_Gahm_The_Half_edge_mesh_data_structure.pdf*
21. Newmark, NM. 1959. A Method of Computation for Structural Dynamics", *ASCE Journal of the Engineering Mechanics Division*, Vol. 85 No. EM3.
22. Doolin DM. and Sitar N. 2004. Time integration in discontinuous deformation analysis DDA. *ASCE Journal of Engineering Mechanics*, 130:249-258.
23. Persistence of Vision Pty. Ltd. 2004. Persistence of Vision Raytracer (Version 3.6), http://www.povray.org/download/
24. Hoek E, Bray J. 1974. *Rock slope engineering*. London: The Institution of Mining and Metallurgy; p. 309.