DOI: 10.14529/jsfi180307

GPU-based Implementation of Discrete Element Method for Simulation of the Geological Fault Geometry and Position

Vadim V. Lisitsa[1], *Vladimir A. Tcheverda*[1], *Victoria V. Volianskaia*[2]

We present an algorithm for numerical simulation of the geological fault formation. The approach is based on the discrete elements method, which allows modeling of the deformations and structural discontinuity of the Upper part of the Earth crust. In the discrete elements method, the medium is represented as an combination of discrete particles which interact as elastic or visco-elastic bodies. Additionally, external potential forces, for example gravitational forces, may be introduced. At each time step the full set of forces acting at each particle is computed, after that the position of the particle is evaluated on the base of Newtonian mechanics. We implement the algorithm using CUDA technology to simulate single statistical realization of the model, whereas MPI is used to parallelize with respect to different statistical realizations. Obtained numerical results show that for low dip angles of the tectonic displacements relatively narrow faults form, whereas high dip angles of the tectonic displacements lead to a wide V-shaped deformation zones.

Keywords: Discrete Element Method, geological faults, CUDA, statistical simulation.

Introduction

A classical definition of the geological faults is that they are discontinuities of sedimentary, metamorphic or magmatic rock bodies. Thus, no physical properties are assigned to a fault; however, real geological faults have a complex structure which includes main fault body (fault core) and fractured or damage zones around [5]. Usually, a fault is the result of tectonic movements. The properties of the fault and near-fault damage zone may significantly differ from the intact rocks and can be a fluid nature active flow channel even in tight formations. Very often, to do field observations or laboratory studies of the real fault is difficult or impossible due to some natural reasons. Thus, numerical simulation is a reliable and efficient way to investigate the peculiarities of the structures forming and tectonic movement process. There are numerous techniques to simulate finite deformations in geological formations including finite elements, finite differences, finite volumes, discrete elements; a review is presented in [3].

In this paper, we present an algorithm based on the Discrete Elements Method (DEM). This approach is based on the media representation by a set of discrete particles. These particles interact as stiff elastic bodies according to the mechanical rules; i.e., elastic and frictional forces affect each particle, that leads to the particle movement according to the Newton mechanics [2]. Computation of the forces affecting a particle includes a high number of floating point and logical operations; thus, it is computationally intense and hard to implement on CPU, using vectorization, etc. As a result, the efficiency of the CPU based realizations of DEM is low, and computation time to solve even a 2D problem may be as long as several thousand node-hours. On the contrary, GPU architecture is more appropriate for DEM implementation, because it can efficiently handle a big number of flops with a small amount of memory involved in computations.

[1]Institute of Petroleum Geology and Geophysics SB RAS, Novosibirsk, Russian Federation
[2] G&G Expert, Moscow, Russian Federation

1. Discrete Element Method

Discrete element method is a meshless approach where the media is represented as agglomeration of independent particles (typically spheres of circles). For each particle total force field is computed, which includes potential forces (gravity field), normal elastic forces, frictional forces, and dissipative forces. Consider two particles with the numbers i and j, with the coordinates $\vec{x}^i$ and $\vec{x}^j$ and radii R^i and R^j respectively. Particle j affects particle i with the normal forces:

$$\vec{F}_n^{ji} = \begin{cases} K_r^-(R^i + R^j - \|\vec{X}^{ji}\|)\vec{n}^{ji}, & R^i + R^j - \|\vec{X}^{ji}\| > 0, \quad repulsion, \\ K_r^+(R^i + R^j - \|\vec{X}^{ji}\|)\vec{n}^{ji}, & 0 \le R^i + R^j - \|\vec{X}^{ji}\| \le r_0, \quad active\ \ bond, \\ 0, & R^i + R^j - \|\vec{X}^{ji}\| > r_0, \quad no\ \ bond, \end{cases} \tag{1}$$

and tangential forces (friction):

$$\vec{F}_t^{ji} = \begin{cases} -K_s\delta_t\vec{t}^{ji}, & K_s\delta_t \le \mu^s\|\vec{F}_n^{ji}\|, \quad static\ \ friction, \\ -\mu^d\|\vec{F}_n^{ji}\|\vec{t}^{ji}, & K_s\delta_t > \mu^s\|\vec{F}_n^{ji}\|, \quad dynamic\ \ friction. \end{cases} \tag{2}$$

In these notations, $K_r^{\pm}$ are the repulsion and attraction Bulk moduli, μ^d and μ^s are the dynamic and static friction coefficients respectively, $\vec{X}^{ji} = \vec{x}^i - \vec{x}^j$, $\vec{n}^{ji}$ is a unit vector directed from the center of the j-th particle to the center of the i-th particle, δ_s is tangential displacement of j-th particle over i-th particle, $\vec{t}^{ji}$ is the self-normalized projection of the relative velocity vector to the plane normal to vector $\vec{n}^{ji}$; i.e., it is tangential to both particles.

Having computed all external forces acting at j-th particle one may recompute its position using classical mechanics principles:

$$M^j\frac{d^2\vec{x}^j}{dt^2} = \sum_{i\in J(i)}\left(\vec{F}_{ji}^n(\vec{x}^j,\vec{x}^i) + \vec{F}_{ji}^t(\vec{x}^j,\vec{x}^i)\right) - \nu\frac{d\vec{x}^j}{dt}, \tag{3}$$

where M^j is the mass of the particle, ν is the artificial viscosity. To integrate equation (3) numerically, we use the Verlet scheme [4].

2. Implementation of the Algorithm

According to the general formulation of the particle-based methods, one has to compute the forces affecting each particle due to the interaction with all other particles. However, in geomechanical modeling by the discrete element method, for each particle only a small number of neighboring particles directly contact the considered one. The adjacency matrix is sparse, but it can evolve. Thus, two related problems should be solved. First, organizing the process of adjacency matrix construction (approximation). Second, computing forces and applying the time stepping.

To construct the adjacency matrix, we suggest using the lattice method. As it follows from the equations (1) and (2), only directly contacting particles affect each other; thus, for each particle, the domain of dependence does not exceed $2R_{max} + r_0$, where R_{max} is the maximal radius of the particles. Also, due to the stability criterion of the Verlet scheme, a single particle cannot move more than $0.1R_{min}$ per a single time step, where R_{min} is the minimal radius over all particles. Thus, we can introduce a grid with the lattice size equal to $2R_{max}+r_0$, so that each particle and all its neighbors belong to the same lattice of directly adjoint lattice. Now we can state the rule of adjacency matrix approximation - for each particle, all the particles belonging

to the same or directly adjoint lattices are neighbors. In this case, we overestimate the number of connected particles but strictly simplify the process of the matrix construction.

The initial assignment of the particles to the lattices is performed by a sequential code by CPU. It is implemented particle-by-particle so that we determine the lattice number for considered particle and add the particle number to the list of particles for this lattice. This procedure is inapplicable under OMP of CUDA parallelization. Thus, the GPU implementation of the reassignment of the particles to the lattices is done lattice-by-lattice. The lattices are large enough, so that after one time step a particle may either stay in the same lattice or move to a directly adjoint lattice. Thus, to update the list of particles for each lattice, we need to check the particles which previously belonged to this lattice or the directly adjusted one. Similar ideas are used in the molecular dynamics and lattice Boltzmann methods but with different principles of lattices construction [1].

Computation of the forces and the numerical solution of the equation of motion is implemented on GPU. The parallelization is applied particle-by-particle, so that a GPU core computes forces for one particle at a time.

3. Numerical Experiments

We use the algorithm to simulate geological fault position for different directions of tectonic displacements in 2D. The computational domain was 4000 m by 500 m. The boundary conditions introduced the displacements. We consider five scenarios with the displacement direction equal to 0° (vertical displacement), 15°, 30°, 45°, and 60° from vertical direction pointing downward. So, in all cases, these are normal dip-slip faults. The vertical displacement was 100 m. Radii of the elements were homogeneously distributed from 0.75 to 1.5 m. The dynamic friction coefficient $\mu_d = 0.3$. We provide the results of simulation – distribution of the horizontal strains ε_{xx} in Fig. 1. Note, that the sub-vertical displacements lead to the formation of a wide damage zone, which has a form of a wedge. The closer the displacements to horizontal direction, the narrower is the wedge with higher deformations in the fault body. Moreover, the main direction of the fault; i.e., the dip angle tends to about 30° which is governed by the properties of the media, rather than displacement direction.

Conclusions

We presented an algorithm for numerical simulation of the geological fault geometry and position. The approach is based on the Discrete Elements Method which allows modeling of finite deformations and structural discontinuities in the Earth's crust. We implement the algorithm using CUDA technology to simulate single statistical realization of the model, whereas MPI is used to parallelize for different statistical realizations. The presented numerical experiments illustrate the formation of the V-shaped fault damage zone. The obtained strains distributions can be used further to estimate the mechanical and transport properties of the fault damage zone.

Acknowledgments

V. Lisitsa and V. Tcheverda proposed and justified this approach to geological modeling of tectonic motions and faults formation. V. Lisitsa developed the algorithm and implemented the

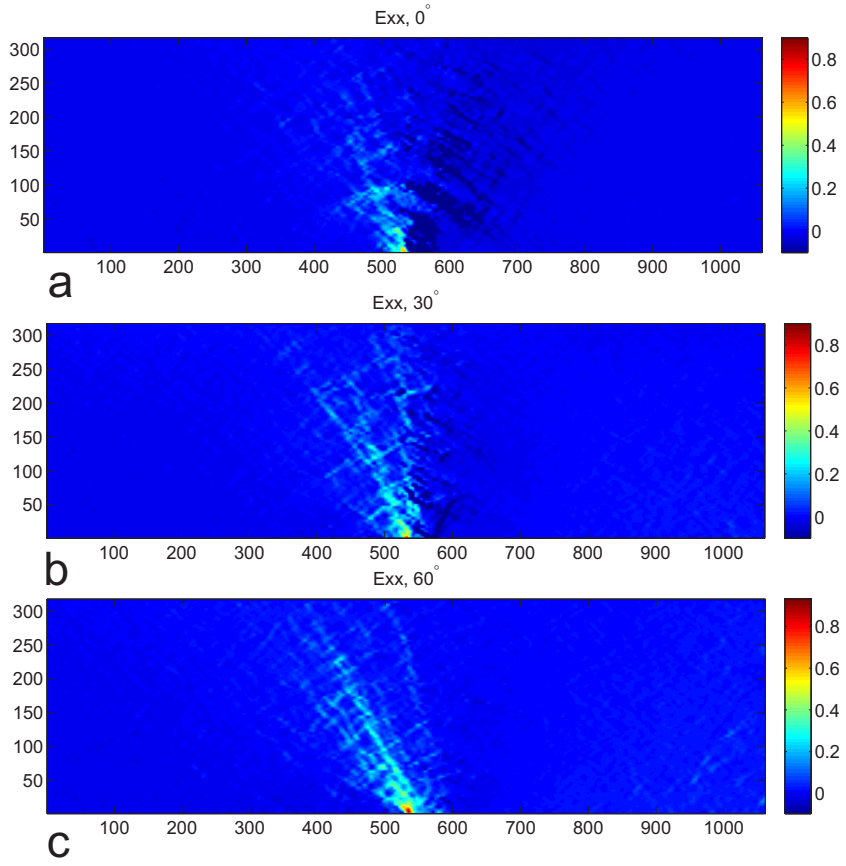

Figure 1. Distribution of horizontal strains ε_{xx} for the displacement directions 0° (a), 30° (b), and 60° (c)

series of numerical experiments. V. Volianskaia did the geological explanation of the simulation results. V. Lisitsa and V. Tcheverda were sponsored by the Russian Science Foundation grant 17-17-01128. The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by the project RFMEFI62117X0011 and cluster NKS-30T+GPU of the Siberian supercomputer center.

References

1. Alpak, F.O., Gray, F., Saxena, N., Dietderich, J., Hofmann, R., Berg, S.: A distributed parallel multiple-relaxation-time lattice boltzmann method on general-purpose graphics processing units for the rapid and scalable computation of absolute permeability from high-resolution 3D micro-ct images. Computational Geosciences 22(3), 815–832 (2018), DOI: 10.1007/s10596-018-9727-7,

2. Hardy, S., Finch, E.: Discrete-element modelling of detachment folding. Basin Research 17(4), 507–520 (2005), DOI: 10.1111/j.1365-2117.2005.00280.x

3. Lisjak, A., Grasselli, G.: A review of discrete modeling techniques for fracturing processes in discontinuous rock masses. Journal of Rock Mechanics and Geotechnical Engineering 6(4),

301–314 (2014), DOI: 10.1016/j.jrmge.2013.12.007

4. Mora, P., Place, D.: Simulation of the frictional stick-slip instability. Pure and Applied Geophysics 143(1), 61–87 (1994), DOI: 10.1007/BF00874324

5. Peacock, D.C.P., Dimmen, V., Rotevatn, A., Sanderson, D.J.: A broader classification of damage zones. Journal of Structural Geology 102, 179–192 (2017), DOI: 10.1016/j.jsg.2017.08.004