Simulation of the fracture of materials and structures under dynamic loads using parallel computing

Pavel Radchenko, Stanislav Batuev, Andrey Radchenko
Tomsk State University of Architecture and Building, Tomsk, Russia
E-mail: radchenko@live.ru

Abstract. The present work suggests algorithm enabling to describe adequately fracture of materials and structures under impulse loadings. The methodology of reconfiguration finite element mesh with re-computation of concentrated masses in nodes is considered. The algorithm uses all the capabilities of modern parallel computing to the full extent and can be easily scaled for all possible objectives. Numerical methodology demonstrates good qualitative and quantitative comparison with the experimental data.

1. Introduction
Description of materials fracture in the area of contact interaction is a challenging issue in numerical simulation of high-velocity interaction of solid bodies. High-velocity loading induces occurrence of the areas with intensive fracture of interacting bodies. Local distortion of computational mesh can lead to both fall of step size over time as well as to the structural failure of computational mesh. Generally it is suggested to remove the fractured finite elements out of numerical calculation.

Current research related to this issue offer various computational mesh adaptive algorithms [1] [2] [3]. Contact problems are solved using different numerical approaches [4] [5] [6]: Lagrange multiplier method, penalty and regularization method, quadratic programming, modifications of Newton method, Schwarz decomposition method, pseudo-medium, and others. Multiple algorithms modify the mesh only in two-dimensional case [7]. Parallel computations have recently become of great concern while developing and modification of finite-element meshes [8].

When solving the objective of solid bodies penetration an extensive area of fractured material is being formed inside the barrier crater; this area influences significantly the step size over time for the evident calculation models. In such case removing of elements and nodes with mass influences negatively the validity of solution; previously authors suggested the recalculation algorithm for local mass in the nodes of computational mesh providing high-quality description of objectives of solid bodies dynamic loading [9].

2. Algorithm description
The erosion algorithm is being implemented when the criterion of fracture in finite element is followed [9] the finite element being on the contact surface. Let us consider the element $E_1$, which is formed by the nodes $N_1, N_2, N_3, N_4$ (Fig. 1a). When following the criterion of fracture in finite element on the surface of a solid body, this element is removed from the computational
grid (Fig. 1a). The element $E_1$ had mutual planes with the elements $E_5$, $E_9$, $E_{13}$, after removal of the element $E_1$ from the grid, the new surface triangles are being built: $N_1N_2N_4$, $N_1N_4N_3$, $N_2N_3N_4$ (Fig. 1b).

One of the crucial issues occurring while numerical implementation of the erosion fracture algorithm is application of the law of conservation of mass. There are various approaches to implementation of surface erosion of interacting bodies under high-velocity loading. Authors used the following approach: in case when one or several nodes of the fractured element no longer belong to the other elements, than their mass is equally distributed between the remaining nodes of this element.

In spite of all the advantages of erosion fracture algorithm data preparation for its implementation can take as much time as the calculation itself. The data required for its implementation is:

(i) Four data sets indicating which surface each of the four finite elements planes belongs to.
(ii) Data set containing information on all of the adjacent planes, namely element number, number of its plane, the number of adjacent element and of its plane.

In order to accelerate data preparation for erosion fracture algorithm searching algorithm of adjacent planes of elements was optimized.

Optimization was performed using the method of sorting of all planes of all elements and their correlation with each other. The exact optimization of the algorithm includes the following steps: initially all finite element mesh is divided into $N$ areas and searching for the adjacent plane for the element $i$ is performed first in the area it belongs to and then it proceeds to adjacent areas, the maximum number of areas where the search of adjacent element can be performed is 27. The areas for search were defined according to the maximum capability of an adjacent plane to get into it, and the process of sorting was taking place according to the order mentioned below:

- the area where the element belongs to (the main, variant a);
- the group of areas which have mutual plane with the main area (6 areas, variant b);
- the group of areas which have mutual edge with the main area (12 areas, variant c);
- the group of areas which have mutual node with the main area (8 areas, variant d).
3. Test calculations

Here are the results of investigations of the body in the shape of cube with the edge 1 m, which was divided into 5.3 million finite elements. The diagram demonstrating the number of elements getting into each of groups of areas is given in Fig. 2.

As can be seen from Fig. 2, when the size of area is decreasing it results in increasing probability that the elements will be able to find the adjacent plane in the other area. As the searching algorithm is occurring consequently in all areas, finding the adjacent plane in other area leads to increasing the time of calculation due to additional sorting of areas. However, this variant of algorithm ensures finding adjacent planes for each element.

Performance improvement relies on substantial decrease in the number of sorting variants. Before optimization the number of sorting was $8 \times N_e^2$, where $N_e$ is the number of finite elements (4 planes of each element were compared with 4 planes of all other elements). After algorithm optimization this number cannot exceed $27 \times 4 \times N_a \times N_e^a$, where $N_a$ is the number of areas, $N_e^a$ is the number of elements in the current area, i.e. only planes of the elements within the area are compared between each other and then in case the adjacent plane is not found the searching proceeds to the adjacent areas.

For the computation mesh which is divided into 1000 areas ($N_a = 1000$), the number of elements in one area will be approximately 1000 times less than the number of elements in the whole body ($N_e/N_e^a = N_a$). The number of iterations before optimization is $8 \times (5 \times 10^6)^2 = 2 \times 10^{14}$. The maximum number of iterations after optimization is $27 \times 4 \times 1000 \times (5 \times 10^3)^2 = 2.7 \times 10^{12}$. After optimization the algorithm will be completed at least 74 times faster.

It is necessary to consider that the optimized algorithm requires extra processing time for areas division. And this time grows along with the increase in the number of areas (Fig. 3). The number of areas along each of the dimensions is defined as $N_x$, $N_y$, $N_z$. Yet in this example the number of areas along each dimension is the same and is denoted as follows: $N_{xyz}$.

Dividing the mesh of 5.3 mln. elements into 125000 areas ($50 \times 50 \times 50$) requires preparation time of about 90 seconds.

Figure 4 shows the time required for the searching algorithm of adjacent planes after optimization on a workstation with four 16-core CPU AMD Opteron 6376.

The largest performance improvement is reached when the area size is 3 mesh spaces. When the size of area increases, time for algorithm data preparation decreases. However, it is less
Figure 3. Processing time for data preparation in each area

Figure 4. Time for calculation of searching algorithm of adjacent planes after optimization

Figure 5. Time for calculation of searching algorithm of adjacent planes for different number of finite elements

than increase in searching time of adjacent plane, therefore, total time for algorithm completion increases.

Figure 5 indicates time comparison required for searching algorithm of adjacent planes for different meshes with different number of finite elements, while the number of areas along each dimension equals to 33.

It can be seen from the graph that increase in the calculation time and increase in the number of elements in the mesh occur almost linearly. Meanwhile, it should be highlighted
that the optimal calculation time by increasing number of elements in computation mesh can be reached when number of divisions along each dimension is different from 33. This issue demands further investigation.

After the algorithm of finite element mesh preparation integration cycle is launching (Fig. 6).

Figure 7 shows percentage-based labour input for calculation of corresponding variables in nodes and mesh elements, as well as for implementation of algorithms of contact interaction and erosion fracture. As can be seen from the diagram the algorithm of erosion fracture is less labour-consuming and takes less than 0.5% of processing time.

The efficiency of parallelization of software package was also analyzed (Fig. 8). For the test calculations workstation with four 16-core CPU AMD Opteron 6376 was used.

Unfortunately, not full program code can be realized simultaneously, there are some areas
Figure 8. The graph of parallelization efficiency

which need to be calculated consequently; therefore performance improvement with increasing the number of cores is non-linear. However, along with increase in the size of computation mesh the efficiency of using larger number of cores grows significantly.

4. Conclusions
(i) Application of erosion fracture algorithm is required for appropriate description of the processes of fracture propagation under high-velocity interaction of solid bodies.
(ii) Dividing of computation mesh into areas provides significant performance improvement while data preparation for erosion fracture algorithm.
(iii) Decrease in the size of area results in reduced relative number of elements getting into the main area, which influences negatively the processing time.
(iv) Time which the processor spends on data preparation should be taken into account, still it contributes to the total time to the less extent as compared to the iterations on searching the adjacent planes of finite elements.

Acknowledgments
This research was partly supported by RFBR (grant No. 18-48-700035, No. 18-41-703003).

References
[1] Renev S A and Shelofast V V 2017 Marine Intellectual Technologies vol. 3 4 (38). pp. 72-78.
[2] Benin A V, Semenov A S, Semenov S G and Melnikov B E 2012 Magazine of Civil Engineering 7. pp. 32-42.
[3] Yadrov V I 2014 Omsk Scientific Bulletin 3 (133). pp. 152-156
[4] Bourago N G and Kukudzhanov V N 2005 Mechanics of Solids vol. 40 1, pp. 3571.
[5] Laursen T A 2003 New York: Springer xv+454 p. DOI: 10.1007/978-3-662-04864-1
[6] Karavaev A S and Kopytov S P 2017 The Bulletin of Udmurt Univ. Mechanics. Vol. 27. pp. 396-413
[7] Lukashevich A A 2007 The Bulletin of Pacific National University 1 (4). pp. 69-82.
[8] Andre K, Gluschenko O N, Ivanov E G and Kudyavtsev A N 2008 Comput. Math. and Math. Phys. Vol. 48 pp. 1448-1457.
[9] Radchenko P A, Batuev S P and Radchenko A V 2018 AIP Conference Proceedings doi: 10.1063/1.5065273