



HOKKAIDO UNIVERSITY

Title	Development of a GPGPU-parallelized hybrid finite-discrete element method for modeling rock fracture
Author(s)	Fukuda, Daisuke; Mohammadnejad, Mojtaba; Liu, Hongyuan et al.
Citation	International journal for numerical and analytical methods in geomechanics, 43(10), 1797-1824 https://doi.org/10.1002/nag.2934
Issue Date	2019-07
Doc URL	https://hdl.handle.net/2115/78767
Rights	This is the peer reviewed version of the following article: Fukuda, D, Mohammadnejad, M, Liu, HY, et al. Development of a GPGPU - parallelized hybrid finite - discrete element method for modeling rock fracture. Int J Numer Anal Methods Geomech. 2019; 43: 1797- 1824. , which has been published in final form at https://doi.org/10.1002/nag.2934 . This article may be used for non-commercial purposes in accordance with Wiley Terms and Conditions for Use of Self-Archived Versions.
Type	journal article
File Information	Article-NAG-revision-NormalCopy.pdf



1 **Development of a GPGPU-Parallelized Hybrid Finite-Discrete Element Method for**
2 **Modelling Rock Fracture**

3

4 D. Fukuda ^{a,b}, M. Mohammadnejad ^{b,c}, H.Y. Liu ^{b,*}, S. Dehkoda ^c, A. Chan ^b, S.H. Cho ^d,
5 G.J. Min ^d, H. Han ^b, J. Kodama ^a, Y. Fujii ^a

6 ^a *Faculty of Engineering, Hokkaido University, Hokkaido 060-8628, Japan*

7 ^b *College of Sciences and Engineering, University of Tasmania, TAS 7001, Australia*

8 ^c *CSIRO Minerals Resources Business Unit, Queensland Centre for Advanced*
9 *Technologies, Brisbane, QLD 4069, Australia*

10 ^d *Department of Mineral Resources and Energy Engineering, Chonbuk National*
11 *University, Jeollabuk-do 561-756, South Korea*

12

13 *Corresponding author: H.Y. Liu

14 E-mail: Hong.Liu@utas.edu.au

15 Tel/Fax: + 03-6226-2113

16

17 *Footnote

18 If you are interested in using our GPGPU-parallelized hybrid finite-discrete element code
19 (Y-HFDEM IDE) for your research or verification, please write an email to the
20 corresponding author. However, we do not accept the use of the code for military or
21 commercial purposes.

22

23 **Abstract**

24 The hybrid finite–discrete element method (FDEM) is widely used for engineering
25 applications, which, however, is computationally expensive and needs further
26 development, especially when rock fracture process is modelled. This study aims to
27 further develop a sequential hybrid FDEM code formerly proposed by the authors and
28 parallelize it using compute unified device architecture (CUDA) C/C++ on the basis of a

29 general purpose graphics processing unit (GPGPU) for rock engineering applications.
30 Because the contact detection algorithm in the sequential code is not suitable for GPGPU
31 parallelization, a different contact detection algorithm is implemented in the GPGPU-
32 parallelized hybrid FDEM. Moreover, a number of new features are implemented in the
33 hybrid FDEM code, including the local damping technique for efficient geostatic stress
34 analysis, contact damping, contact friction and the absorbing boundary. Then, a number
35 of simulations with both quasi-static and dynamic loading conditions are conducted using
36 the GPGPU-parallelized hybrid FDEM, and the obtained results are compared both
37 quantitatively and qualitatively with those from either theoretical analysis or the literature
38 to calibrate the implementations. Finally, the speed-up performance of the hybrid FDEM
39 is discussed in terms of its performance on various GPGPU accelerators and a
40 comparison with the sequential code, which reveals that the GPGPU-parallelized hybrid
41 FDEM can run more than 128 times faster than the sequential code if it is run on
42 appropriate GPGPU accelerators, such as the Quadro GP100. It is concluded that the
43 GPGPU-parallelized hybrid FDEM developed in this study is a valuable and powerful
44 numerical tool for rock engineering applications.
45 Keywords: Rocks, Fracture process analysis, Hybrid FDEM, Quasi-static loading, Impact
46 loading, GPGPU, and CUDA C/C++

47

48 **1. Introduction**

49 Understanding the fracture process mechanism of rocks is significantly important in the
50 field of civil and mining engineering as well as other fields, such as geothermal,
51 hydraulic, oil and gas engineering, in which rock fracture plays an important role.
52 Recently, numerical methods have been increasingly applied to analyze the fracture
53 process of rocks (e.g., ¹). Recent advances in computational mechanics have realized a
54 better understanding of complex fracture processes. Generally, approaches based on
55 computational mechanics can be classified into continuum and discontinuum
56 formulations. In the framework of the fracture process analysis of rocks, continuum-based

57 methods include the finite element method (FEM), the finite difference method (FDM),
58 the boundary element method (BEM), the scaled-boundary finite element method
59 (SBFEM), the extended finite element method (XFEM), meshless methods, methods
60 based on peridynamics and phase-field methods, while discontinuum-based methods
61 include the distinct element method (DEM), the lattice model (LM) method, and
62 molecular dynamics (MD). More detailed information on recent advances in the
63 computational fracture mechanics of rocks can be found in recently published review
64 articles ¹⁻³. To realistically simulate the fracture process of rock, numerical techniques
65 must be capable of capturing crack onset, arbitrary crack growth, the correct crack length
66 within a given time interval and the propagating directions. In recent years, increasing
67 attention has been paid to these techniques, which can unify the advantages of the
68 aforementioned continuum-based and discontinuum-based methods. Attempts in this
69 direction lead to the development of coupled methods, hybrid/hybrid methods and
70 multiscale coupled methods ¹.

71 The hybrid finite–discrete element method (FDEM) proposed by Munjiza ⁴ has been
72 employed successfully to model problems that address the transition process from
73 continuum to discontinuum such as rock fracturing and fragmentation. The hybrid FDEM
74 incorporates the advantages of both continuum and discontinuum methods and can
75 realistically simulate the transition from continuum to discontinuum caused by rock
76 fracture. Two main implementations of the hybrid FDEM include Y code ⁵⁻⁸ and the
77 commercial code ELFEN ⁹⁻¹². Several attempts have been made to actively extend the Y
78 code, such as Y-GEO ¹³⁻¹⁹, IRAZU ²⁰⁻²², Solidity ²³⁻²⁹, HOSS with MUNROU ^{30,31} and Y-
79 Flow ³²⁻³⁶. In addition, the authors also developed the Y-HFDEM IDE (integrated
80 development environment) ³⁷⁻⁴⁰. The principles of all of the hybrid FDEM codes are
81 based on continuum mechanics, the cohesive zone model (CZM) and contact mechanics,
82 which make the codes very computationally expensive. Therefore, developing a capable
83 parallel computation scheme is important for handling large-scale problems with a
84 massive number of nodes, elements and contact interactions.

85 To date, some successful parallel implementations of the hybrid FDEM codes using
86 MPI (message-passing interface) (e.g., ^{8,9,11,30,31,41,42}) and shared-memory programming
87 such as OpenMP (e.g., ²³) have been reported. Among these, Lukas et al. ⁸ proposed a
88 novel approach for the parallelization of 2D hybrid FDEM using MPI and parallelization
89 solvers based on dynamic domain decomposition and successfully applied the
90 parallelized Y code to a large-scale 2D problem on a PC cluster. Meanwhile, Lei et al. ³⁰
91 successfully developed the concept of a virtual parallel machine for the hybrid FDEM
92 using MPI, which can be adapted to various computer architectures ranging from few to
93 thousands of CPU (central processing unit) cores. Furthermore, Rougier et al. ³¹
94 introduced the HOSS with MUNROU code, which notably used 208 processors
95 controlled by MPI and developed novel contact detection and contact force calculation
96 algorithms. The developed code was applied successfully to 3D simulation of a dynamic
97 Brazilian test of rock with a split Hopkinson pressure bar apparatus. ELFEN uses MPI in
98 its parallelization scheme and has been employed successfully in 2D and 3D simulations
99 of the rock fracture process. For example, 3D fracture process analysis of a conventional
100 laboratory test using up to 3 million elements has been reported ¹¹. Additionally, Xiang et
101 al. ²³ optimized the contact detection algorithm in their Solidity code and parallelized the
102 code using OpenMP; they modeled a packing system with 288 rock-like boulders and
103 showed that a speedup of 9 times on 12 CPU threads can be achieved (although the
104 details of the applied algorithm and its implementation were not provided). Relevant to
105 the MPI-based parallelization, Guo and Zhao ^{43,44} developed a hierarchical
106 FEM/DEM coupling, which had been extended for modeling of granular rocks
107 more recently ^{45,46}. In general, MPI requires a large and expensive PC cluster to achieve
108 the best performance. Meanwhile, the application of shared-memory programming such
109 as OpenMP is limited by the total number of multi-processors that can reside in a single
110 computer; thus, MPI is still required for large-scale problems in which each computer
111 uses both OpenMP and MPI to transfer data among multiple computers. This means that
112 the hybrid MPI/OpenMP is necessary.

113 In addition to CPU-based parallelization schemes, the GPGPU (general purpose
114 graphics processing unit) accelerator controlled by either OpenCL (open computing
115 language) ⁴⁷ or CUDA (compute unified device architecture) ⁴⁸ can be considered another
116 promising method for the parallelization of hybrid FDEM codes. Thousands of GPU-core
117 processors can reside and concurrently work in a small GPGPU accelerator within an
118 ordinary laptop/desktop PC or workstation with lower energy consumption than the CPU-
119 based PC cluster. Moreover, a GPGPU cluster with a massive number of GPGPU
120 accelerators is also possible.

121 Zhang et al. ⁴⁹ developed a CUDA-based GPGPU parallel version of Y code (2D)
122 without considering the fracture process and the contact friction. Batinić et al. ⁵⁰
123 implemented GPGPU-based parallel hybrid FDEM based on the Y code for analysis of
124 cable structures using CUDA. However, neither of the above implementations have been
125 employed in the simulation of rock fracture. In this regard, a GPGPU-based hybrid
126 FDEM commercial code, namely, IRAZU ²⁰⁻²², has just been developed with OpenCL and
127 was used successfully in rock fracture simulations. Additionally, some novel GPGPU-
128 based DEM modeling methods coupled with FEM, i.e., coupled FEM/DEM, have been
129 proposed. For example, Nishiura et al. ⁴⁰ developed the quadruple DEM (QDEM) code
130 using GPGPU and successfully applied the code to the investigation of ballasted railway
131 track dynamics. In addition, Ma et al. ⁵¹ and Wang et al. ⁵² developed a GPGPU-based
132 implementation of continuum-based DEM. These studies ^{40,51,52} showed that the
133 performance of coupled FEM/DEM can be significantly improved using GPGPU
134 parallelization. However, the coupled FEM/DEM is a physical coupling of two methods
135 (DEM and FEM), unlike the hybrid FDEM, in which the transition from continuum to
136 discontinuum is modeled by CZM under the explicit FEM formulation. In other words,
137 there are three essential differences between the coupled FEM/DEM and the hybrid
138 FDEM: 1) the continuous behavior of the coupled FEM/DEM is modelled through
139 springs with normal and tangential stiffnesses while that of the hybrid FDEM is through
140 intact cohesive elements with high penalty parameters; 2) the transition from continuum

141 to discontinuum of the coupled FEM/DEM is implemented through removing the springs
142 while that of the hybrid FDEM is through the softening of the cohesive elements; and 3)
143 the contact interaction between the discrete elements of the coupled FEM/DEM is
144 calculated through contact springs while that of the hybrid FDEM is through potential
145 contact force calculation method in the framework of explicit FEM. In this sense, the
146 hybrid FDEM parallelized in this study is called the hybrid FDEM to distinguish it from
147 the coupled FEM/DEM. Therefore, the hybrid FDEM is computationally more
148 demanding than the coupled FEM/DEM.

149 Overall, it can be concluded that IRAZU, which is parallelized using OpenCL, is the
150 only available GPGPU-based hybrid FDEM code to date that is capable of modeling the
151 rock fracture process ²⁰⁻²². Thus, further studies are required to develop the GPGPU-based
152 hybrid FDEM. In this regard, we have developed free research code*, Y-HFDEM IDE ³⁷⁻
153 ⁴⁰, and recently parallelized it using GPGPU with CUDA C/C++. Moreover, it is
154 desirable to fully describe any newly implemented GPGPU-based code because the
155 implementation of GPGPU-based codes differs from that of CPU-based sequential codes.
156 In addition, while there is no freely available GPGPU-based hybrid FDEM code, our code
157 is free to use and may significantly contribute to many studies in the field of rock
158 engineering.

159 Based on the above background, this paper aims to explain the recently developed
160 GPGPU-based hybrid FDEM algorithm implemented in 2D Y-HFDEM IDE, along with
161 newly implemented functions, to increase the applicability of the algorithm in the field of
162 rock engineering. The capability of the code in realistic modeling is demonstrated by
163 providing examples. Thus, this paper will provide a useful basis for further improvement
164 and development of the hybrid FDEM codes based on GPGPU.

165 The paper is arranged as follows. First, the theory used in 2D Y-HFDEM IDE is
166 introduced, and then its implementation in the GPGPU parallel computation framework is
167 explained in detail. Section 3 investigates the accuracy and capability of the developed
168 code through several examples such as 2D fracture process analyses of rock fracture in

169 conventional laboratory testing and blasting. In section 4, the performance of the
170 developed GPGPU-based code is compared against that of the original CPU-based
171 sequential code. Finally, section 5 concludes and highlights future work.

172 **2. Hybrid finite–discrete element method implemented in Y-HFDEM IDE**

173 The Y-HFDEM IDE was originally developed using object-oriented programming in
174 visual C++³⁷ on the basis of the CPU-based sequential open-source Y-code^{5,7}. The Y-
175 HFDEM IDE can not only significantly simplify the process of building and manipulating
176 the input models and greatly reduce the possibility of erroneous model setup but can also
177 display calculated results graphically in real-time with OpenGL. The preprocessor of the
178 Y-HFDEM IDE can even generate simple FDEM mesh and specify initial conditions,
179 physical properties, contact properties, boundary conditions, fracture criteria, and
180 explosive charges if necessary. More complex FDEM meshes are usually generated using
181 third-party software such as ABAQUS®, ANSYS LS-DYNA® and various open-source
182 mesh generators such as Netgen and Gmsh. Then, the generated mesh data can be easily
183 imported into the Y-HFDEM IDE for hybrid FDEM analyses. The postprocessor can
184 visually display the calculated stress, displacement, velocity, force, damage, fracture and
185 fragmentation in real-time pictures or query calculated results in specified locations and
186 graphically display them. In addition, a number of operations such as pan, rotation, zoom,
187 various viewports in perspective and/or orthographic modes, and slideshow are developed
188 to manipulate the numerical models and calculated results (see³⁷ for further detail). The
189 code has been successfully employed in the simulation of the fracture process in various
190 geotechnical engineering problems³⁷⁻⁴⁰. Because of the nature of sequential
191 programming, its main application was limited to small-scale 2D problems using a
192 relatively rough mesh. To overcome this limitation, the parallel programming scheme
193 using the GPGPU accelerator controlled by CUDA C/C++ is implemented into the Y-
194 HFDEM 2D IDE through this study. Because the various hybrid FDEM-based codes have
195 been independently developed in different research institute/organization and have

different features, the fundamental features of Y-HFDEM 2D IDE along with its GPGPU-based parallelization scheme are explained in detail through the following subsections.

2.1. Fundamental theory of 2D Y-HFDEM IDE

The principles of the hybrid FDEM are based on continuum mechanics, cohesive zone modeling and contact mechanics, all of which are formulated in the framework of explicit FEM⁵.

The continuum behavior of materials including rocks is modeled by an assembly of continuum 3-node triangular finite elements (TRI3s) (Fig. 1(a)). Two types of isotropic elastic constitutive models have been implemented. In the first type, which is implemented in the original Y-code and has been widely used, the isotropic elastic solid obeys Eq. (1) of the Neo-Hookean elastic model:

$$\sigma_{ij} = \frac{\lambda}{2} \left(J - \frac{1}{J} \right) \delta_{ij} + \frac{\mu}{J} (B_{ij} - \delta_{ij}) + \eta D_{ij} \quad (i, j = 1, 2, 3) \quad (1)$$

where σ_{ij} denotes the Cauchy stress tensor, B_{ij} is the left Cauchy–Green strain, λ and μ are the Lamé’s constants, J is the determinant of the deformation gradient, η is the damping coefficient, δ_{ij} is the Kronecker’s delta and D_{ij} is the rate of the deformation tensor. However, Eq. (1) cannot model anisotropic elasticity and the plane strain problem, which are also very important in the field of rock engineering. Thus, in the second type, a hyper elastic solid obeying Eqs. (2) and (3) is also implemented:

$$S_{KL} = C_{KLMN} E_{MN} \quad (K, L, M, N = 1, 2) \quad (2)$$

$$\sigma_{ij} = \frac{1}{J} F_{iK} S_{KL} F_{jL} + \eta D_{ij} \quad (i, j, K, L = 1, 2) \quad (3)$$

where S_{KL} denotes the 2nd Piola–Kirchhoff stress tensor, C_{KLMN} is the effective elastic stiffness tensor, E_{MN} is the Green–Lagrange strain tensor and F_{iK} is the deformation gradient. Note that the Einstein’s summation convention applies in Eqs. (2) and (3). By setting C_{KLMN} in Eq. (2) properly, isotropic and anisotropic elastic behaviors can be simulated. In Eqs. (1) and (2), the infinitesimal strain tensor is not used and thus a large displacement and a large rotation can be simulated. For the simulation of the fracture process of materials under quasi-static loading, $\eta = \eta_{crit} = 2h\sqrt{(\rho E)}$ is used to achieve critical damping⁵, where h , ρ and E are the element length, density and Young’s modulus

224 of the target material, respectively. Accordingly, the hybrid FDEM can address both
 225 dynamic and quasi-static problems. The σ_{ij} within each TRI3 is converted to the
 226 equivalent nodal force $\mathbf{f}_{\text{int}}$ (e.g., ⁵²).

227 The fracture of rock under mode I and mode II loading conditions, i.e., the opening
 228 and sliding of cracks, is modeled via CZM using the smeared crack model ⁵³. To model
 229 the behavior of the fracture process zone in front of the crack tips, tensile and shear
 230 softening is applied using an assembly of 4-node cohesive elements with an initial
 231 thickness of zero (CE4s) (Fig. 1(a)) as a function of crack opening and sliding
 232 displacements, (o , s) (Fig. 1(b)). Two methods can be used for CE4 insertion. One
 233 method involves inserting the CE4s into all of the boundaries of the TRI3s at the
 234 beginning of the analysis; this method is known as the intrinsic cohesive zone model
 235 (ICZM) ⁵⁴. The second method involves adaptively inserting the CE4s into the particular
 236 boundaries of the TRI3s with the help of adaptive remeshing techniques where a given
 237 failure criterion is met; this method is referred as the extrinsic cohesive zone model
 238 (ECZM) ⁵⁴. Many existing hybrid FDEM codes, e.g., Y code including Y-HFDEM IDE,
 239 have employed the ICZM, while some codes such as ELFEN have used the ECZM. One
 240 of the advantages of the ICZM is that the implementation and application of the parallel
 241 computing algorithm is straightforward since adaptive remeshing is unnecessary in this
 242 case. However, an “artificial” elastic behavior of CE4s before the onset of fracturing
 243 must be specified, which requires the introduction and correct estimation of penalty terms
 244 and the careful selection of the time step increment Δt to avoid numerical instability.

245 In the GPGPU-based 2D Y-HFDEM code, normal and shear cohesive tractions (σ^{coh}
 246 and τ^{coh} , respectively) acting on each face of CE4 are computed using Eqs. (2) and (3)
 247 assuming tensile and shear softening behaviors, respectively:

$$248 \quad \sigma^{\text{coh}} = \begin{cases} \frac{2o}{o_{\text{overlap}}} T_s & \text{if } o < 0 \\ \left[\frac{2o}{o_p} - \left(\frac{o}{o_p} \right)^2 \right] f(D) T_s & \text{if } 0 \leq o \leq o_p \\ f(D) T_s & \text{if } o_p < o \end{cases} \quad (4)$$

$$\tau^{coh} = \begin{cases} \left[\frac{2|s|}{s_p} - \left(\frac{|s|}{s_p} \right)^2 \right] (-\sigma^{coh} \tan(\phi) + f(D)c) & \text{if } 0 \leq |s| \leq s_p \\ -\sigma^{coh} \tan(\phi) + f(D)c & \text{if } s_p < |s| \end{cases} \quad (5)$$

where o_p and s_p are the elastic limits of o and s , respectively, $o_{overlap}$ is the representative overlapping when o is negative, T_s is the tensile strength of CE4, c is the cohesion of CE4 and ϕ is the internal friction angle of CE4. Positive o and σ^{coh} are crack opening and tensile cohesive traction, respectively. Eq. (5) corresponds to the Mohr–Coulomb (MC) shear strength model with the tension cut-off. When ICZM is used, the “artificial” elastic behavior of each CE4 characterized by o_p and s_p along with $o_{overlap}$ is necessary to connect the TRI3s to express the intact deformation process, which is given as follows⁵³:

$$o_p = 2hT_s / P_f \quad (6)$$

$$s_p = 2hc / P_{tan} \quad (7)$$

$$o_{overlap} = 2hT_s / P_{overlap} \quad (8)$$

where P_f , P_{tan} and $P_{overlap}$ are the penalty terms of CE4s for opening in the normal direction, sliding in the tangential direction and overlapping in the normal direction, respectively, and h is the element length; the values of the penalty terms can be considered the stiffness of the CE4 for its opening, sliding and overlapping, respectively. Ideally, these penalty values should be infinity to satisfy elastic behavior of rocks according to Eq. (1), but this condition would require an infinitesimal Δt . Therefore, a reasonably large value of the penalty terms, compared to the Young’s modulus or Lamé’s constants, is required, as it is impossible to use infinity in actual numerical simulations. Otherwise, the intact behavior of the bulk rock shows significantly different behavior from that specified by Eqs. (1) and (2), and the elastic constants used in Eqs. (1) and (2) lose their meaning. This consideration is very important, especially for problems in which the speed of the stress wave is important (see section 3.4). The function $f(D)$ in Eqs. (4) and (5) is the characteristic function for the tensile and shear softening curves (Fig. 1(b)) and depends on the damage value D of the CE4. The following definitions of D and $f(D)$

are used to consider not only mode I and II fracturing but also mixed mode I–II fracturing^{13,53}.

$$D = \text{Minimum} \left(1, \sqrt{\left(\frac{o - o_p}{o_t} \right)^2 + \left(\frac{|s| - s_p}{s_t} \right)^2} \right) \text{ if } o \geq o_p \text{ or } |s| > s_p, \text{ otherwise } 0 \quad (9)$$

$$f(D) = \left[1 - \frac{A+B-1}{A+B} \exp \left(D \frac{A+CB}{(A+B)(1-A-B)} \right) \right] [A(1-D) + B(1-D)^C] \quad (0 \leq D \leq 1) \quad (10)$$

where A , B and C are the intrinsic rock properties that determine the shape of softening curves, and o_t and s_t are the critical values at which a CE4 breaks and turns into a macro/explicit fracture of o and s , respectively. If tensile loading condition is presented only (i.e. $o > o_p$ and $|s| \leq s_p$), the damage variable D in Eq. 9 becomes the pure mode I damage D_I (= minimum (1, $(o - o_p)/o_t$)). If shear loading condition is presented only (i.e. $o \leq o_p$ and $|s| > s_p$), it becomes the pure mode II damage D_{II} (= minimum (1, $(|s| - s_p)/s_t$)). If both the tensile and shear loading conditions are presented (i.e. $o \leq o_p$ and $|s| > s_p$), it becomes the mixed mode I-II damage D_{I-II} defined in Eq. 9. The true damage is defined as the sum of the pure mode I, pure mode II and mixed mode I-II damages, which is $D = D_I + D_{II} + D_{I-II}$. To avoid unrealistic damage recovery, if the trial f computed from Eq. (10) at the current time step becomes larger than that at the previous time step f_{pre} , a condition of $f = f_{pre}$ is assigned. The o_t and s_t in Eq. (9) satisfy the mode I and II fracture energies G_{fI} and G_{fII} specified in Eqs. (11) and (12), respectively:

$$G_{fI} = \int_{o_p}^{o_t} \sigma^{coh}(o) do \quad (11)$$

$$G_{fII} + W_{res} = \int_{s_p}^{s_t} \{ \tau^{coh}(|s|) \} d|s| \quad (12)$$

where W_{res} is the amount of work per area of CE4 done by the residual stress term in the MC shear strength model. This paper uses the same $f(D)$ with A , B and C equal to 0.63, 1.8 and 6.0⁵³, respectively, for both mode I and II fracture processes because of the lack of experimental evidence. In addition, unloading, i.e., the decrease of o or $|s|$, can also

occur during the softening regime, i.e., $o > o_p$ or $|s| > s_p$ (see Fig. 1(b)), which is modeled based on Eqs. (13) and (14) ⁵⁵:

$$\sigma^{coh} = f(D_{max})T_s \frac{o}{o_{max}} \text{ if } 0 < o < o_{max} \text{ and } o_{max} > o_p \quad (13)$$

$$\tau^{coh} = \left\{ -\sigma^{coh} \tan(\phi) + f(D_{max})c \right\} \frac{|s|}{s_{max}} \text{ if } |s| < s_{max} \text{ and } s_{max} > s_p \quad (14)$$

In each CE4, the computed σ^{coh} and τ^{coh} are converted to the equivalent nodal force $\mathbf{f}_{coh}$ using a 3-point Gaussian integration scheme ⁵³. When either o_t or s_t is achieved in a CE4, the element becomes deactivated and its surfaces can be considered new macro-fracture surfaces.

The contact processes between the material surfaces, including the newly created macro fractures by the separation of each CE4, are modeled by the penalty method ⁴; a complete and excellent explanation of the method is given in the literature ⁴. As a brief explanation, when any two TRI3 elements subjected to contact detection (see the next subsection for the implementation of contact detection in the framework of GPGPU) are found to overlapped, the contact potential due to the overlapping of the two TRI3s, i.e., the contacting couple, is exactly computed. The normal contact force f_{con_n} , which is normally acting on the contact surface and is proportional to the contact potential, is then computed for each contacting couple. The proportional factor is called the normal contact penalty P_{n_con} . After the normal contact force f_{con_n} and its acting point are obtained, the nominal normal overlap o_n and the relative displacement Δu_{slide} at the acting point of f_{con_n} are readily computed. The contact damping model proposed by An and Tannant ⁵⁶ (Fig. 2) can also be applied if the role of contact damping is very important. When this scheme is applied, the normal contact force f_{con_n} mentioned above is regarded as a trial contact force $(f_{con_n})^{try}$, and a trial contact stress $(\sigma_{con_n})^{try}$ is then computed by dividing $(f_{con_n})^{try}$ by the contact area A_{con} . Then, Eq. (15) is used to determine the contact stress σ_{con_n} :

$$\sigma_{\text{con}_n} = \begin{cases} \text{Minimum}\left((\sigma_{\text{con}_n})^{\text{try}}, T\right) & \text{during the increase of } o_n \text{ (Loading)} \\ T(o_n / o_{n_max})^b & \text{during the decrease of } o_n \text{ (Unloading)} \end{cases} \quad (15)$$

where T is the transition force, b is the exponent, and o_{n_max} is the maximum value of o_n experienced during the loading process at the contact. T limits σ_{con_n} and defines the transition between a linear elastic stress–displacement relationship and a ‘recoverable’ displacement at a constant contact stress. The values of T may be related to the physical properties of the rocks being simulated, such as the uniaxial compressive strength. The exponent b adjusts the power of the damping function that is applied to the rebound or extension phase of the contact. The value of the exponent controls the energy loss during an impact event. A very similar contact damping model is implemented in the 2D Y-Geo code in the framework of the hybrid FDEM, in which only b is modeled¹³. After σ_{con_n} is computed using Eq. (15), it is converted to f_{con_n} ($= A_{\text{con}} \times \sigma_{\text{con}_n}$). The verification of the implemented contact damping is discussed in subsection 3.2. After f_{con_n} is determined, the tangential contact force f_{con_tan} is computed according to the classical quasi-static Coulomb friction law. First, the trial tangential contact force is computed by $(f_{\text{con}_tan})^{\text{trial}} = (P_{\text{tan_con}} \times f_{\text{con}_tan} \times \Delta u_{\text{slide}})$, where $P_{\text{tan_con}}$ is a tangential contact penalty. The actual f_{con_tan} is computed based on Eq. (16):

$$f_{\text{con}_tan} = \begin{cases} (f_{\text{con}_tan})^{\text{trial}} & \text{if } (f_{\text{con}_tan})^{\text{trial}} \leq \mu_{\text{fric}} f_{\text{con}_n} \\ \mu_{\text{fric}} f_{\text{con}_n} & \text{if } (f_{\text{con}_tan})^{\text{trial}} > \mu_{\text{fric}} f_{\text{con}_n} \end{cases} \quad (16)$$

where μ_{fric} is the friction coefficient between the contact surfaces. The tangential contact force f_{con_tan} is applied parallel to the contact surface against the direction of the relative sliding of the contact faces. The verification of the implementation of the contact friction is discussed in subsection 3.2. In each contacting couple, the contact force is converted to the equivalent nodal force $\mathbf{f}_{\text{con}}$ ⁴.

By computing the nodal forces mentioned above, the following equation of motion, Eq. (17), is obtained and solved in the framework of explicit FEM⁴:

$$\mathbf{M} \partial^2 \mathbf{u} / \partial t^2 = \mathbf{f}_{\text{ext}} + \mathbf{f}_{\text{int}} + \mathbf{f}_{\text{coh}} + \mathbf{f}_{\text{con}} \quad (17)$$

where $\mathbf{M}$ is a lumped nodal mass, $\mathbf{u}$ is the nodal displacement and $\mathbf{f}_{\text{ext}}$ is the nodal force

348 corresponding to the external load. The application of fluid-driven pressure due to
349 detonation phenomena such as that described by An et al.³⁹ can also be considered. The
350 central difference scheme is employed for the explicit time integration to solve Eq. (17).
351 Careful selection of the time step Δt is necessary to avoid numerical instability. An
352 excellent explanation of the reasonable selection of Δt in the hybrid FDEM can be found
353 in section 2.3.5.2 of the literature²⁹.

354

355 **2.2. GPGPU-based parallelization of 2D Y-HFDEM IDE by CUDA C/C++**

356 To speed-up the simulation process of the 2D Y-HFDEM IDE, a parallel
357 computation scheme based on the NVIDIA® GPGPU accelerator is incorporated. In our
358 case, the computation on the GPGPU device is controlled through the NVIDIA's CUDA
359 C/C++⁴⁸, which is essentially an ordinary C/C++ programming language with several
360 extensions that make it possible to leverage the power of the GPGPU in the
361 computations. The CUDA programming model uses the abstractions of “*threads*”,
362 “*blocks*” and “*grids*”⁴⁸ (Fig. 3). A greater degree of parallelism occurs within the
363 GPGPU device itself. Functions also known as “*kernels*” are launched on the GPGPU
364 device and are executed by many “*threads*” in parallel. A “*thread*” is just an execution
365 unit of a “*kernel*” that has a given “*thread index*” within a particular “*block*”. As shown
366 in Fig. 3, a “*block*” is a group of the threads, and a unique “*block index*” is given to each
367 “*block*”. The “*block index*” and “*thread index*” enable each thread to use its unique
368 “*index*” to globally access elements in the GPGPU data array such that the collection of
369 all threads processes the entire data set in parallel. A “*grid*” is just a group of “*blocks*”. A
370 system with a single “*grid*” is used in this study. The “*blocks*” can execute concurrently
371 or serially depending on the number of streaming processors available in a GPGPU
372 accelerator. Synchronization between “*threads*” within a “*block*” is possible, while no
373 synchronization is possible between “*blocks*”. At each “*thread*” level, the corresponding
374 code that “*threads*” execute is very similar to the CPU-based sequential code (see
375 Appendix 1 and Fig. A1), which is one of the advantages of the application of CUDA

376 C/C++. For example, in the Quadro GP100 accelerator (Pascal generation ⁴⁸) used in this
 377 paper, the number of streaming processors and CUDA cores ⁴⁸ are 56 and 3584,
 378 respectively. Thus, a high computational performance of the GPGPU parallelized code
 379 run on the GPU accelerator can be achieved, compared to that of ordinary CPU-based
 380 sequential code. The number of “*blocks*” per “*grid*” (N_{BpG}) and the number of “*threads*”
 381 per “*block*” (N_{TpB}) can be changed for the speed-up using GPGPU (Fig. 3). The current
 382 version of 2D Y-HFDEM IDE normally sets N_{TpB} to either 256 or 512, and N_{BpG} is
 383 automatically computed by dividing the total number of threads (N_{thread}) in each “*kernel*”
 384 by N_{TpB} , in which an additional block is needed if $N_{\text{thread}}/N_{TpB}$ is not the multiple of N_{TpB} .
 385 The value of N_{thread} is set to be equal to the total number of TRI3s, CE4s, contact couples
 386 or nodes depending on the purpose of each “*kernel*”.

387 In the GPGPU implementation of 2D Y-HFDEM IDE, the computation of each TRI3
 388 ($\mathbf{f}_{\text{int}}$ and $\mathbf{M}$), CE4 ($\mathbf{f}_{\text{coh}}$), contact couple ($\mathbf{f}_{\text{con}}$) or nodal equation of motion (Eq. (17)) is
 389 assigned to each GPGPU “*kernel*”, as shown in Fig. 4, and processed in a massively
 390 parallel manner. In Appendix A.1, an exemplary code of the GPGPU “*kernel*”
 391 programmed to solve Eq. (17) is shown. It is evident from this example that the CUDA
 392 code used in the “*kernel*” is very similar to the CPU-based sequential code, which also
 393 holds true for all of the computations shown in Fig. 4. Thus, most parts of the original
 394 sequential CPU-based code can be used with minimal modifications. To compute the
 395 contact force $\mathbf{f}_{\text{con}}$, a “triangle-to-triangle” (TtoT) contact interaction is used in the earliest
 396 versions of the Y-2D code⁵. This TtoT approach exactly considers the geometries of both
 397 contactor and target TRI3s, and the integration of the contact force distributed along the
 398 edges of the TRI3s is done analytically. Because this approach integrates the contact
 399 force exactly, it is precise although quite time consuming. As pointed out in the literature
 400 ³⁰, the contact interaction in 2D can be further simplified by “triangle-to-point” (TtoP)
 401 contact interaction kinematics, which makes the implementation simpler and more time
 402 efficient. However, the precision of the computed contact force using the TtoP approach
 403 is low unless a sufficient number of target points per TRI3 is used. Thus, in the Y-

404 HFDEM 2D IDE, the TtoT approach is applied (instead of the TtoP approach) to ensure
405 the precision of the computed contact force.

406 A flowchart of the 2D Y-HFDEM IDE is shown in Fig. 5. However, one of the
407 challenging problems in Fig. 5 is the achievement of efficient contact detection for
408 identifying each contacting couple only through the GPGPU without any sequential
409 computation. For example, in the case of a sequential CPU implementation, there are
410 powerful and efficient contact detection algorithms, such as the NBS (no binary search)
411 contact detection algorithm proposed by Munjiza ⁴, which can achieve the linear search in
412 which the required computation for contact detection is proportional to the number of
413 TRI3 candidates subjected to contact detection. However, such contact detection
414 algorithms are not straightforward to implement in the GPGPU-based code. In the 2D Y-
415 HFDEM IDE, considering that hybrid FDEM modeling requires a fine mesh that often
416 consists of TRI3s of a similar size, the following contact detection algorithm is
417 implemented. In this algorithm, the analysis domain comprising a massive number of
418 TRI3s is subdivided into multiple equal-sized (n_x , n_y) square subcells in the x and y
419 directions (Fig. 6) so that the largest TRI3 in the analysis domain is completely included
420 in a single subcell. In this way, the center of every TRI3 can have a single subcell to
421 which it belongs (Fig. 6). By using an integer coordinate (ix , iy) ($ix=0, \dots, n_x-1$, $iy=0, \dots, n_y-$
422 1) in the x and y directions for the location of each subcell, unique hash values h
423 ($=iy \times n_x + ix$) are assigned to each subcell. For example, the five TRI3s shown in red in
424 Fig. 6 are included in the blue subcell and have the same hash value. The subsequent
425 contact detection procedure is explained using a simplified example shown in Fig. 7,
426 where there are ten TRI3 candidates of similar size subjected to contact detection. First,
427 all of the TRI3s are mapped into the integer coordinate ($ix=0, 1$ and 2 , and $iy=0, 1$ and 2)
428 with $n_x = n_y = 3$ along with each hash value. In this way, the list L-1 is readily constructed.
429 Then, the IDs of the TRI3s in the list L-1 are sorted from smallest to largest according to
430 the hash values as keys, which generates the list L-2 in Fig. 7. For the key sorting by
431 hash, the radix sorting algorithm optimized for CUDA ⁵⁷ and implemented in the open-

432 source “*thrust*” library is used; thus, this procedure can also be processed in a massively
 433 parallel manner. Utilizing the list L-2 and GPGPU device shared memory ⁴⁸, the list L-3
 434 is further constructed in a GPGPU “*kernel*“, which makes it possible to identify the first
 435 and last indices of the particular hash value in the List L-2. As an example, let us consider
 436 the index = 2 in the arrays of L-2, i.e. “Sorted TRI3 ID” = 0 and “Sorted_*h*” = 3. By
 437 comparing the hash values *h* in the immediately left and right neighbor indices with that
 438 in the index = 2 (i.e., indices =1 and 3 of “Sorted_*h*” array) in L-2, it can be found that the
 439 hash value *h* in the index=1 (i.e. *h* = 0) is different from that in the index = 2 (i.e. *h* = 3).
 440 Thus, the indices = 1 and 2 in L-2 correspond to the last and first TRI3s in the subcells
 441 with *h* = 0 and 3, respectively. Thus, the “last index” for *h* = 0 and the “first index” for *h*
 442 = 3 in L-3 are set to “1” and “2”, respectively, as indicated using the green dash curves in
 443 L-2 and L-3 of Fig. 7. At the same time, for the hash values *h* = 1 and 2, these subcells
 444 are clearly empty with no TRI3 elements, which are filled in “E” (i.e. empty) in L-3. The
 445 same explanation is applicable to all the other indices in the arrays of L-2. In this way, L-
 446 3 is constructed. Therefore, the “first index” and the “last index” in L-3 indicate the first
 447 and last indices, respectively, in L-2 for each hash. Since this operation only requires to
 448 check the hash values of two adjacent array indices in L-2, the construction of L-3 can
 449 also be processed in a massively parallel manner. To further explain how the efficient
 450 contact detection is achieved using L-2 and L-3, let us consider a TRI3 with ID =8 in Fig.
 451 7. It is evident that the TRI3 with ID = 8 belongs to a sub-cell with *h*=4. Thus, the
 452 neighboring sub-cells around the subcell *h* = 4 are: “*h*-1-*n_x*”, “*h* -*n_x*”, “*h*+1-*n_x*”, “*h*-
 453 1”, “*h*”, “*h*+1”, “*h*-1+*n_x*“, “*h* +*n_x*“ and “*h*+1+*n_x*“, i.e. *h* = 0, 1, 2, 3, 4, 5, 6, 7 and 8,
 454 respectively, in this example. According to L-3, it is unnecessary to search the
 455 neighboring cells with *h* = 1, 2, 6 and 8 since these are the empty subcells. Then, for the
 456 remaining non-empty subcells with *h* = 0, 3, 4, 5 and 7, TRI3 with ID = 8 of “Sorted
 457 TRI3 ID List” in L-2 may contact with the TRI3s included in these non-empty subcells.
 458 As an example, for a subcell with *h* = 0, L-3 indicates that the “first index” = 0 and the
 459 “last index” = 1. In this case, it is only necessary to trace the indices from the “first

index” to the “last index” and then process contacts for the TRI3s with IDs corresponding to each index of “Sorted TRI3 ID List”. Since no sequential CPU procedure is involved in the above procedure, it becomes possible to achieve efficient contact detection using the GPGPU device only.

Therefore, the GPGPU-parallelized Y-HFDEM IDE can run completely parallel on the GPGPU device and no sequential processing is necessary (except for the input and output procedures). The data transfer from the GPGPU device to the host computer is always necessary for outputting the analysis results, the time of which is often negligible in the entire simulation time for most of the Y-HFDEM IDE simulations. The obtained results can be visualized in either OpenGL implemented in the Y-HFDEM IDE ³⁷ or the open-source visualization software, Paraview ⁵⁸.

Finally, it is worth mentioning that an efficient contact detection activation approach has been proposed and applied in many publications on the hybrid FDEM in the framework of ICZM (e.g., section 2.3.3.2 and Fig. 2.14 in ²⁹). In the efficient contact detection activation approach, along with the contact candidates prescribed by the user, only TRI3s in the vicinity of a newly created explicit fracture become contact candidates and are added into the contact detection list. One advantage of this approach is that the contact detection and contact force calculation are necessary only for initial material surfaces by the time the broken/failed CE4s are generated; thus, drastic time savings for the contact detection computation are possible. However, in the hybrid FDEM simulation of rocks under compressive-dominant loading conditions such as uniaxial compression, most TRI3s can overlap during compression even before the generation of explicit/macro fractures. In this case, if the amount of overlap is not negligible when broken CE4s are generated, the efficient contact detection activation approach results in the sudden application of a contact force such as a step function, i.e., a shock, which causes spurious numerical instability and results in unrealistic fragmentation. To avoid spurious numerical instability, an infinitesimally small Δt must be used, which makes the simulation impracticable. Although the efficient contact detection activation approach is

implemented in the Y-HFDEM IDE, for the numerical simulation of uniaxial compression discussed in subsection 3.2, almost all TRI3s are added into the contact detection at the early stage of simulation, which makes the computation become rather demanding but is essential for avoiding spurious numerical instability. Thus, the demanding contact detection activation approach is also implemented in the current code in addition to the efficient contact detection activation approach. It should be noted that every ICZM-based hybrid FDEM code needs to address this problem carefully to avoid any inaccurate simulation (although it has not been reported in the literature).

3. Numerical tests and code validation

This section aims to conduct verifications and validations via several numerical simulations. All numerical simulations in this section are conducted using the GPGPU-parallelized Y-HFDEM IDE developed in this study. When the numerical simulation results are presented, compressive stress is regarded as negative (cold color) while tensile stress is taken as positive (warm color).

3.1. Verification of the continuum deformation of GPGPU-parallelized hybrid

FDEM and the implementation of the local damping scheme

Without the insertion of CE4s, the hybrid FDEM acts as an explicit FEM to solve continuum deformation. To verify the ability of the developed code in the simulation of continuum deformations, a simple example of geostatic stress analysis under gravity is modeled using the GPGPU-parallelized Y-HFDEM IDE, and the numerical model is shown in Fig. 8. The computation of the geostatic stress field is very important in many rock engineering applications, such as the stability testing of underground excavations and slopes. Normally, geostatic stress analysis is conducted before the insertion of CE4s^{13,19}. In most explicit FEM schemes, a dynamic relaxation scheme based on artificial damping is applied to compute the geostatic stress field. The critical damping technique is one of the simplest approaches that has been used in many existing applications in the literature^{13,19} to compute the *in situ*/initial stress. However, it was noted during our code

development that the convergence rate of the critical damping technique is very poor. To solve this problem, we implemented local damping with a mass scaling technique into the GPGPU-parallelized Y-HFDEM IDE, which was initially proposed by Cundall⁵⁹ and implemented in Itasca's commercial FLAC software⁶⁰. In the local damping with mass scaling technique, the following Eq. (18) is used instead of the aforementioned Eq. (17):

$$\mathbf{M}^{\text{scale}} \partial^2 \mathbf{u} / \partial t^2 = \mathbf{f}_{\text{tot}} + \alpha \|\mathbf{f}_{\text{tot}}\| \text{sgn}(\mathbf{v}) \mathbf{1} \quad (18)$$

where $\mathbf{M}^{\text{scale}}$ is the scaled lumped mass, $\mathbf{f}_{\text{tot}}$ is the nodal out-of-balance-force, i.e., the right-hand side of Eq. (17), $\mathbf{v}$ is the nodal velocity, $\|\mathbf{f}_{\text{tot}}\|$ is the absolute value of each component of $\mathbf{f}_{\text{tot}}$, $\text{sgn}(\cdot)$ is the sign function automatically determined by the sign of $(\cdot)$ ^{59,60} and α is the local damping coefficient^{59,60}. The comprehensive details of the local damping scheme can be found in the literature^{59,60}.

By applying the gravitational acceleration $g = 9.806 \text{ (kg.m/s}^2\text{)}$ downward with the boundary conditions shown in Fig. 8, initial stress analyses under the plane strain condition are conducted with an overburden pressure of 0. Two analyses are conducted using critical damping and local damping schemes. The mechanical properties of the rock mass are listed in Table 1, where the density $\rho = 1800 \text{ (kg/m}^3\text{)}$, Young's modulus $E = 12.2 \text{ (GPa)}$ and Poisson's ration $\nu = 0.25$ (correspondingly Lamé constants $\lambda = \mu = 4.88 \text{ (GPa)}$). For the critical damping scheme, the critical viscous damping factor $\eta_{\text{crit}} = 3.0 \text{ (MPa.s)}$ and the largest time step $\Delta t = 25 \text{ (}\mu\text{s)}$ for the stable simulation are selected, while $\alpha = 0.8$ ^{59,60} and $\Delta t = 1 \text{ (s)}$ are used for the local damping scheme with mass scaling. The number of TRI3s and nodes in the model (Fig. 8) are 98314 and 49516, respectively.

Fig. 9(a) depicts the distribution of the vertical stresses σ_{yy} simulated using both the critical damping scheme and the local damping scheme with mass scaling. Because no differences can be identified between the final resultant stresses simulated using these two schemes, only the result of one scheme is displayed in Fig. 9(a). In addition, Fig. 9(b) shows the profiles of σ_{xx} and σ_{yy} along the vertical direction of the model from the top to the bottom. Considering theoretical stresses as $\sigma_{yy} = -\rho g(100 - y)$ and $\sigma_{xx} = \nu / (1 - \nu) \sigma_{yy}$, the obtained results are in good agreement with the theoretical values. Therefore, the

544 computations of each TRI3 ($\mathbf{f}_{\text{int}}$ and $\mathbf{M}$, $\mathbf{M}^{\text{scale}}$) in Eqs. (1)–(3), (17) and (18) based on the
545 GPGPU computation can be considered to be accurate.

546 In the case of explicit FEMs with artificial damping schemes, static stress equilibrium
547 is achieved when both the total kinetic energy of the system and the maximum value of
548 $\|\mathbf{f}_{\text{tot}}\|$ among all nodes converge to zero. Fig. 10 illustrates the variations of the total
549 kinetic energy of the system with respect to the calculation time steps during the
550 simulation of the initial stress analyses. Evidently, the simulation with the local damping
551 scheme with mass scaling can achieve the equilibrium stress state significantly faster than
552 that with the critical damping scheme. To reach the final resultant stress state in Fig. 9,
553 the local damping scheme with mass scaling requires approximately 6400 calculation
554 steps, while the critical damping scheme requires approximately 1000000 calculation
555 steps if the convergence criterion suggested in the literature^{59,60} is applied.

556

557 3.2. Verifications of the implementation of contact damping and contact friction

558 To assess the accuracy of the contact damping model implemented in section 2.1, a
559 simple impact test is modeled (Fig. 11(a)) using the GPGPU-parallelized Y-HFDEM
560 IDE. The model is same as that reported by Mahabadi et al.¹³, and the obtained results are
561 compared with those reported in their work. The model consists of a circular elastic body
562 with a radius of 0.1 (m) vertically impacting a fixed rigid surface. The elastic body is not
563 allowed to fracture in this model. Following the literature¹³, gravitational acceleration is
564 neglected, the density of the elastic body is 2,700 (kg/m³), and the kinetic energy of the
565 elastic body before the impact event is 4.1 (kJ). Because the Lamé constants λ and μ for
566 the elastic body are not available in the literature¹³, it is simply assumed that $\lambda = \mu = 5.0$
567 (GPa) and the damping coefficient $\eta = 0$. Thus, energy dissipation is only due to contact
568 damping. Parametric analyses are conducted by changing the exponent b and the
569 transition force T in Eq. (15), and the normal contact penalty P_{con_n} between the elastic
570 body and the rigid surface. The kinetic energy of the elastic body as a function of time is
571 monitored during the parametric analyses.

Fig. 11(b) compares five cases with b values equal to 1, 2, 5, 20 and 30 when $T = \infty$ and $P_{\text{con}_n} = 0.1$ (GPa). The case with $b = 1$ corresponds to elastic contact, and thus no energy dissipation due to contact occurs. The small decrease in the kinetic energy occurs after the impact event because a small amount of the kinetic energy is converted to the strain energy of the elastic body. By changing the values of b , the amount of kinetic energy dissipated from the system increases. This behavior is the same as that reported in the literature¹³ using a sequential hybrid FDEM. Cases with different values of P_{con_n} (= 0.1 (GPa) and 10 (GPa)) when $b = 2$ and $T = \infty$ show that the same b does not result in the same energy dissipation when P_{con_n} differs, which is a reasonable outcome because the maximum value of the nominal normal overlap o_{n_max} in Eq. (15) during the impact event does change when P_{con_n} changes. However, this detail is not reported or explained in the literature¹³. Likewise, cases with different values of T ($= \infty$ and 1 MPa) when $b = 2$ and $P_{\text{con}_n} = 0.1$ GPa show a different amount of energy dissipation, which can also be explained by the change in o_{n_max} . Thus, it is verified that the contact detection and computation of $\mathbf{f}_{\text{con}}$ are properly processed in the GPGPU code; however, this paper does not consider any contact damping in the following numerical simulations because the calibration of these parameters is beyond the scope of this paper.

To assess the accuracy of the contact friction model implemented in the GPGPU-parallelized Y-HFDEM IDE, a simple sliding test, suggested by Xiang et al.⁶¹, is modeled, and the obtained results are compared against those from theoretical analyses. The model includes a simple sliding rock square with a length of 5 cm and a fixed rigid base as shown in Fig. 12(a). The material parameters of the rock square are assigned according to those listed in Table 1. The rock square can slip along the horizontal plane with a friction coefficient of $\mu_{\text{fric}} = 0.5$. The sliding distance L is a function of initial velocity (v_i), gravity acceleration (g) and μ_{fric} , which can be determined theoretically through Eq. (19):

$$L = v_i^2 / (2\mu_{\text{fric}}g) \quad (19)$$

599 In the numerical modeling, the sliding rock square, which is assigned various initial
600 velocities from 2 m/s to 6 m/s, slows and stops at a distance due to the friction between
601 the rock square and the rigid base. As illustrated in Fig. 12(b), the obtained results from
602 the numerical simulation are in good agreement with those from the theoretical analyses.

603

604 **3.3. Simulation of the rock failure process in standard laboratory rock mechanics**

605 **tests**

606 To verify the capabilities of the code in simulating the rock fracture process and
607 associated failure mechanism, this section models two standard laboratory rock
608 mechanics tests, i.e., the Brazilian test and the uniaxial compression test, using the
609 GPGPU-parallelized Y-HFDEM IDE. For this purpose, the failure processes of a
610 relatively soft limestone in these two tests are modeled. The physicomechanical
611 properties of the limestone and the numerical input parameters used in the numerical
612 simulation are listed in Table 1. The numerical input parameters are determined based on
613 the methodology suggested by Tatone and Grasselli¹⁸ while the physicomechanical
614 properties of the limestone are obtained from laboratory tests. The diameter and thickness
615 of the Brazilian disc (BD) are 51.7 (mm) and 25.95 (mm), respectively, and the numerical
616 model consists of 10,520 unstructured TRI3s (Fig. 13(a)). In the numerical model of the
617 uniaxial compression test, the height and diameter of the rock specimen are 129.5 (mm)
618 and 51.7 (mm), respectively. The numerical model consists of 44,214 unstructured TRI3s
619 (Fig. 13(b)). The average edge length h_{ave} of TRI3s in both models is 0.7 (mm). The rock
620 specimens are placed between two moving rigid platens with a constant velocity of 0.05
621 (m/s) to satisfy quasi-static loading conditions¹⁸. The friction coefficient μ_{fric} between
622 platens/rock and rock/rock are assumed to be 0.1 and 0.5, respectively⁶². Hereafter,
623 compressive stress is shown as negative (cold color) while tension stress is regarded as
624 positive (warm color).

625 Fig. 14 illustrates the screenshots of the rock failure process (Fig. 14 (a)-(c)) and the
626 associated indirect tensile stress versus axial strain curve (Fig. 14 (d)) obtained from the

627 numerical simulation of the Brazilian test using the GPGPU-parallelized Y-HFDEM IDE.
628 It should be noted that the screenshots of the rock failure process shown in Fig. 14 (a), (b)
629 and (c) are taken at the stress levels labeled using A, B and C, respectively, in the indirect
630 tensile stress versus axial strain curve depicted in Fig. 14 (d). It can be seen from Fig.
631 14(a) that the horizontal stress distribution σ_{xx} in the rock specimen is almost uniform
632 along its loading diameter. When the concentrated stress reaches the critical value (i.e.,
633 the tensile strength of CE4s for the rock), tensile failure develops in the model (Fig.
634 14(b)). With the two loading plates further moving toward each other, the splitting failure
635 of the BD occurs due to the propagation of the formed macroscopic fractures, which
636 coalesce with the microcracks initiated due to the resultant tensile stresses during the
637 stress propagation process.

638 As seen from Fig. 14, at the stress labeled by point A (Fig. 14 (d)), i.e., before the
639 peak stress, microcracks/damages are initiated and propagate near the loading areas (Fig.
640 14 (a)). Once the resultant stress (point B in Fig. 14 (d)) reaches the peak strength of the
641 rock, the macro-crack, i.e., the splitting crack, then appears around the central line of the
642 model due to the coalescence of microcracks (Fig. 14 (b)). Finally, the stress-strain curve
643 decreases rapidly during the postfailure stage (e.g., point C in Fig. 14 (d)) when the
644 splitting crack propagates along the sub-central line of the rock specimen dividing it into
645 two halves (Fig. 14 (c)). The modelled rock fracture process and the obtained indirect
646 tensile stress versus axial strain curve are in good agreement with those observed in
647 laboratory testing of a Brazilian disc specimen under a quasi-static load. Therefore, the
648 GPGPU-parallelized Y-HFDEM IDE can realistically model the splitting/tensile failure
649 process of rock in the Brazilian test.

650 Fig. 15 illustrates the stages of the rock fracture process in the uniaxial compression
651 test modeled using the GPGPU-parallelized Y-HFDEM IDE. Fig. 15 (a) illustrates the
652 stress build-up and evolution process before the onset of nonlinearity in the stress-axial
653 strain curve, i.e., point A in Fig. 16(a). As loading continues, unstable microcrack growth
654 commences and continues to the peak stress point of the stress-strain curve, i.e., point B

in Fig. 16(a). Then, the microcracks coalesce to form macroscopic cracks, which results in the rock specimen losing its bearing capacity, and correspondingly, the observed stress decreases with the strain increase. Finally, the formed macroscopic cracks propagate, resulting in the rock specimen completely losing its bearing capacity at point C in Fig. 16(a). Fig. 16 (b) compares the final fracture patterns obtained from the numerical simulation and the experimental tests. Clearly, the developed GPGPU-parallelized Y-HFDEM IDE can realistically model the failure process of rock under a uniaxial compression test in which shear failure is the dominant mechanism.

3.4. Simulation of dynamic rock fracture process

In this section, the dynamic fracture process in rock blasting is modeled using the GPGPU-parallelized Y-HFDEM IDE. Because the code is formulated based on the explicit FEM, the hybrid FDEM is also a powerful tool for the simulation of a dynamic rock fracture process and the rock fracture process. By default, the hybrid FDEM considers all boundaries reflection boundaries, i.e., free faces. However, the simulation of dynamic problems such as rock blasting often require a nonreflection, i.e., absorbing, boundary to satisfy the infinity condition. Correspondingly, the absorbing boundary is implemented into the GPGPU-parallelized Y-HFDEM IDE by viscous boundary tractions using an approach similar to that used in the literature^{63,64}. The normal and tangential boundary tractions (t_n , t_s) acting perpendicularly and tangentially, respectively, to the boundary are given by Eq. (20):

$$\begin{bmatrix} t_n \\ t_s \end{bmatrix} = -\rho \begin{bmatrix} V_p v_n \\ V_s v_s \end{bmatrix} \quad (20)$$

where V_p and V_s are the P- and S-wave speeds of the target boundary, respectively, and v_n and v_s are the particle velocities that are perpendicular and tangential to the boundary, respectively. The computation of the viscous boundary tractions in each edge of the target boundary is assigned to each GPGPU “thread” following the same concept shown in Fig. 4. In addition, the same features as those in the literature³⁹ are implemented in the

682 GPGPU-parallelized Y-HFDEM IDE, which includes the modeling of blast-induced
 683 pressure due to detonation phenomena and the effect of the loading rate on the fracture
 684 behaviors of rock. The blast-induced pressure at time t according to the literature ³⁹ is
 685 applied to multiple surface edges of TRI3s corresponding to the surfaces of blast holes
 686 and broken CE4s connected to each blast hole. The computations in each edge are also
 687 assigned to each GPGPU “*thread*” (following the same concept shown in Fig. 4) and
 688 processed completely in a parallel manner. However, the authors have recognized that the
 689 gas pressure model used in the literature ³⁹ should be improved further, which is beyond
 690 the scope of this paper, and thus not used in the following verification.

691 The same model used in the literature ⁶⁴ is utilized with the GPGPU-parallelized Y-
 692 HFDEM IDE to simulate the dynamic rock fracture analysis process (Fig. 17) because of
 693 its simplicity and validity. The model consists of a single blast hole (0.05 m in radius)
 694 within a rock disk (5 m in radius). The model includes 45,086 unstructured TRI3s,
 695 135,258 CE4s and 135,258 nodes. The same mechanical properties of rock as those in the
 696 literature ⁶⁴ are used (see Table 1 in ⁶⁴), and the simulation is conducted under the plane
 697 strain condition. It should be noted that the 2D DFPA code applied in the literature ⁶⁴ is
 698 developed based on the implicit FEM using the ECZM model, and its numerical model is
 699 constructed using structured mesh. Thus, the intact stress wave propagation exactly
 700 follows the constitutive behavior of an isotropic elastic body. However, the GPGPU-
 701 parallelized Y-HFDEM IDE is developed based on the ICZM, and thus the values of the
 702 penalty numbers (P_f , P_{tan} and $P_{overlap}$) of the CE4s need to be set carefully. For the
 703 dynamic rock fracture modeling reported in this study, the penalty numbers of $P_f = P_{tan} =$
 704 $P_{overlap} = 50$ times the Young modulus of rock ($E_{rock} = 60$ GPa) is used. Accordingly, a
 705 smaller $\Delta t = 0.4$ (ns) is used in the modeling to avoid numerical instability. The following
 706 pressure function $P(t)$ of Eq. (21), which is used in the literature ⁶⁴, is applied to the
 707 surface of the blast hole:

$$708 \quad P(t) = P_0 \left(e^{-\alpha t} - e^{-\beta t} \right) / \left(e^{-\alpha t_0} - e^{-\beta t_0} \right) \quad (21)$$

709 where t_0 is the rise time of the pressure and given by $t_0 = [1/(\beta - \alpha)] \log(\beta/\alpha)$, and β/α is the
710 controlling parameter of the pressure decay. Here, only the case of $\beta/\alpha = 1.5$ and $t_0 = 100$
711 (μs) is simulated. $P(t)$ is only applied to the initial surface of the blast hole following that
712 in the literature ⁶⁴, and thus the gas flow into the fractures is not considered. Two cases
713 are considered to demonstrate the effect of the implementation of the absorbing boundary
714 on the exterior boundary of the model shown in Fig. 17. The first case (case 1) considers
715 the exterior boundary an absorbing boundary, while the other case (case 2) considers it a
716 free boundary.

717 Fig. 18(a) compares the spatial distribution of the maximum principal stress (PS1)
718 and broken CE4s at selected time intervals for both case 1 and case 2. Because the results
719 of both cases are exactly the same by the time t_{arrive} , i.e., when the stress wave front
720 reaches the exterior boundary, only one case is shown before t_{arrive} . After the stress wave
721 front reaches the exterior boundary of the model, the results of both cases are illustrated
722 and compared with each other. The positive value (warm color) of PS1 corresponds to
723 tension, and broken CE4s are shown by black lines. Just after the commencement of the
724 pressure application to the blast hole, the stress wave starts to propagate radially from the
725 blast hole ($t = 200 \mu\text{s}$ in Fig. 18 (a)). The front of the PS1 wave shows compression (cold
726 color), and this means that both the radial and circumferential stress components are in
727 compression. This stress state results in shear fracturing in the vicinity of the blast hole,
728 i.e., the crushed zone. The tensile PS1 wave shown by the warm color follows this
729 compressive stress wave front, which causes the radially propagating tensile cracks
730 around the crushed zone. These tensile cracks extend further with the propagation of the
731 tensile PS1 wave ($t = 600 \mu\text{s}$) and then arrest because the gas flows in the cracks are not
732 considered. The obtained result is in good agreement with the reported results in the
733 literature ⁶⁴. After the compressive stress wave front reaches the external boundary, as
734 expected, no wave reflection is shown in case 1 at $t = 1300 \mu\text{s}$. However, the wave
735 reflection is shown in case 2 at $t = 1300 \mu\text{s}$, with the tensile stress wave propagating back
736 to the blast hole. It should be noted that the sign reversal of the reflective stress waves

737 occurs for both the compressive and tensile stress waves impinging on the free face.
738 Thus, the implemented absorbing boundaries show the expected behavior during rock
739 blasting modeling. Correspondingly, it can be concluded that the dynamic fracture
740 process analysis by the GPGPU-parallelized Y-HFDEM IDE has been verified.

741 To investigate the effect of the penalty numbers (P_f , P_{tan} and $P_{overlap}$) of CE4 on the
742 intact stress wave propagations, Fig. 19(b) shows the spatial distribution of PS1 and
743 broken CE4s at $t = 900 \mu s$ for three cases with various penalty numbers (i.e., $P_{overlap} = P_f$
744 $= P_{tan} = 50 E_{rock}$, $10 E_{rock}$ and E_{rock}). Evidently, the stress distributions clearly differ
745 among the three cases. The condition with the penalty numbers of E_{rock} even shows a
746 different fracture pattern than those with the penalty numbers of $50 E_{rock}$ and $10 E_{rock}$. By
747 computing the smallest distance between the stress wave front and the original blast-hole
748 surface, the apparent P-wave velocities for cases with the penalty numbers of $50 E_{rock}$, 10
749 E_{rock} and E_{rock} are found to be 4910 m/s, 4858 m/s and 4660 m/s, respectively, while the
750 expected (theoretical) P-wave velocity is 5000 m/s according to the theory of
751 elastodynamics. In addition, the apparent wave length of the stress wave becomes longer
752 when the values of the penalty numbers of CE4s decrease. In dynamic fracture process
753 analysis, stress wave propagation is the most important factor because it determines the
754 fracturing process, and many previous publications using the ICZM-based hybrid FDEM
755 for dynamic fracture process analyses simply set the penalty numbers of CE4s to E_{rock} or
756 close to E_{rock} . In the hybrid FDEM, the intact behavior of rocks should only be governed
757 by Eqs. (1) or (2) because either of these equations is the constitutive equation for
758 “continuum behavior”; moreover, the artificial behavior of CE4s controlled by the
759 penalty numbers should not affect the continuum behavior described by Eq. (1) or Eq. (2).
760 Otherwise, there is no meaning in specifying the elastic parameters as input parameters.
761 Hence, the condition of the penalty numbers of CE4s close to E_{rock} *must not be used* for
762 any quantitatively meaningful dynamic fracture process analysis. Therefore, any ICZM-
763 based hybrid FDEM simulations used for quantitative evaluation/prediction should set
764 penalty numbers for CE4s with the utmost care, and the applied penalty numbers of CE4s

765 must be validated before any dynamic fracture process analysis. The ECZM-based
766 method does not suffer from this artificial behavior of CE4s, and our future task includes
767 the implementation of the ECZM-based hybrid FDEM.

768

769 **4. Performance**

770 This section discusses the performance of the GPGPU-parallelized Y-HFDEM IDE,
771 mainly in terms of its improvement compared with the sequential implementation of the
772 Y-HFDEM IDE and its performance on several GPGPU accelerators. To accomplish this
773 goal, the modeling of the rock failure process in the uniaxial compression test, as
774 discussed in subsection 3.3, is selected as a benchmark because it is a computationally
775 demanding simulation. The model shown in Fig. 13 (b) comprises 44,214 unstructured
776 TRI3s, and the numbers of CE4s, nodes and initial contact couples are 66,810, 134,442
777 and 362,043, respectively. Because the performance of GPGPU-parallelized code is
778 significantly dependent on the applied GPGPU accelerators ⁴⁸, the GPGPU-parallelized
779 Y-HFDEM IDE is run using several NVIDIA® GPGPU accelerators, i.e., Quadro GP100,
780 GTX 1060, GTX 1050Ti, GTX 830M, TESLA K80(K40) and TESLA K20, to investigate
781 its performance. Each of the NVIDIA® GPGPU accelerators can be categorized based on
782 its generation. Sorting by date from newest to oldest, the Quadro GP100, GTX 1060 and
783 GTX 1050Ti belong to the “Pascal” generation, whereas GTX 830M is in the “Maxwell”
784 generation, and TESLA K80(K40) and TESLA K20 are in the “Kepler” generation. The
785 new “Volta” and “Turing” generations have been released recently, but this paper does
786 not consider these generations. The developed GPGPU-parallelized Y-HFDEM IDE can
787 be run in all these GPGPU accelerators without any modifications. At the same time, an
788 Intel® Core i7-440 CPU (3.40 GHz) is used to run our sequential CPU-based Y-HFDEM
789 IDE.

790 Fig. 19 shows the speed-up times of the GPGPU-based code relative to the sequential
791 CPU-based code running on a single thread. In Fig. 19, the vertical axis shows the
792 quotients of the total run time using each GPGPU accelerator divided by that using the

793 sequential CPU-based code (= 138.17 (hours)), which, thus, correspond to the speed-up
 794 times of the GPGPU-based code relative to the sequential CPU-based code. Clearly, all
 795 the GPGPU-based codes can achieve quicker times than the sequential CPU-based code,
 796 and the Quadro GP100 accelerator in the “Pascal” generation shows the best performance
 797 among them. However, the performance of GTX 830M in the “Maxwell” generation is
 798 very poor because the computational capability to perform double precision arithmetic in
 799 the GPGPU accelerator of this generation is very limited compared with that of the
 800 “Kepler” and “Pascal” generations ⁴⁸. Therefore, among the investigated GPGPU
 801 accelerators, the application of “Kepler” and “Pascal” generations are suited to achieve
 802 better performance, considering the given specifications of each accelerator ⁴⁸. In
 803 addition, because the performance of GPGPU accelerators have continued to significantly
 804 improve by generation, the GPGPU-parallelized Y-HFDEM IDE can easily achieve better
 805 speed-up times without requiring any changes if it is run on GPGPU accelerators of the
 806 newest generation, such as the Volta generation (e.g., GTX TITAN V, Quadro GV100
 807 and Tesla V100) ⁴⁸. This finding is very important because the selection of the proper
 808 GPGPU accelerator tends to be difficult for many researchers.

809 In addition, the relative speed-up times between GPGPU-based and sequential CPU-
 810 based codes can further increase when many more elements and nodes with more
 811 significant contact detections/force calculations are involved. In other words, keeping all
 812 the GPGPU cores busy is the most important factor in achieving the best performance of
 813 the GPGPU-based code. For example, the uniaxial compression model shown in Fig. 13
 814 (b) is simulated using six different average element lengths (h_{ave}), whose values are 0.15
 815 (mm), 0.2 (mm), 0.3 (mm), 0.4 (mm), 0.5 (mm) and 0.6 (mm). To make the condition of
 816 each case closer, the value of Δt is fixed to 0.5 (ns), with other conditions remaining the
 817 same as those in Table 1, excluding the critical damping coefficient η_{crit} , which is
 818 selected based on element size. Table 2 shows the number of TRI3s, CE4s and nodes, and
 819 the initial number of contact couples in each model. It is evident that mesh discretization

820 with $h_{ave} = 0.15$ (mm) results in tremendously massive computation. The runtime required
821 for 10,000 calculation time-steps is monitored here.

822 The list of actual runtimes required for 10,000 calculation time steps is shown at the
823 bottom of Table 2 for several values of h_{ave} , for the cases of the GPGPU-based code using
824 the Quadro GP100 accelerator and the sequential CPU-based code. The results show that
825 108,767 s (30.2 (hours)) are required to solve the 10,000 time steps in the sequential
826 CPU-based code for $h_{ave} = 0.15$ (mm), which means that solving the problem with this
827 level of fine discretization is too computationally expensive using the sequential code.
828 Based on the list in Table 2, Fig. 20 shows the speed-up times of the GPGPU-based code
829 using the Quadro GP100 accelerator relative to the sequential CPU-based code, in which
830 the horizontal axis represents the number of TRI3s for each h_{ave} . The relative speed-up
831 time increases when the mesh becomes finer, i.e., when the GPGPU becomes busier.
832 Notably, a relative speed-up time of 128.6 times is achieved for the finest mesh ($h_{ave} =$
833 0.15 (mm)). Thus, the GPGPU accelerator must be kept busy to achieve its best
834 performance. Even if different models and different architectures of the GPGPU
835 accelerators are used to run the CUDA-based GPGPU-parallelized Y-HFDEM IDE
836 discussed in this paper, the obtained speed-up time is quite competent compared with the
837 performance of the OpenCL-based GPGPU code “IRAZU” reported by Lisjak et al. ²².
838 Finally, considering that the Quadro GP100 can be installed in an ordinary workstation,
839 the demonstrated speed-up performances indicate that less space and time are required to
840 solve large-scale hybrid FDEM simulations by applying the GPGPU-parallelized Y-
841 HFDEM IDE. The presented speed-up list here can provide useful information for the
842 application of GPGPU parallelization to the hybrid FDEM simulations.

843

844 5. Conclusion and future work

845 This paper developed a GPGPU-parallelized Y-HFDEM IDE based on the authors’
846 formal CPU-based sequential hybrid FDEM code. The algorithm of the GPGPU-
847 parallelized hybrid FDEM was first given in detail so that this paper can provide a basis

848 for further improvement and progress of any hybrid FDEM codes that were reviewed in
849 the introduction section on the basis of GPGPU parallelization. It should be noted that a
850 new contact detection algorithm that differs from that in the sequential CPU code was
851 implemented in the GPGPU-parallelized Y-HFDEM IDE because the contact detection
852 algorithm in the sequential code is not suitable for GPGPU parallelization. A number of
853 new features that were unavailable in the original CPU-based sequential code were
854 implemented into the GPGPU-parallelized Y-HFDEM IDE to achieve improvements in
855 rock engineering applications, which include the implementation of efficient geostatic
856 stress analysis through the local damping scheme with mass scaling, contact damping,
857 contact friction and the absorbing boundary. Then, a series of numerical simulations were
858 conducted using the GPGPU-parallelized Y-HFDEM IDE, and the obtained results were
859 compared with those from either theoretical analysis or the literature to calibrate the
860 implementations. Finally, GPGPU-parallelized Y-HFDEM IDE was applied in modeling
861 the rock failure process in the Brazilian test, the uniaxial compression test and rock
862 blasting to demonstrate its application in rock engineering. Through this study, the
863 following conclusions can be drawn:

864 – The developed GPGPU-parallelized Y-HFDEM IDE can work well with the
865 implementation of the aforementioned algorithm, and its precision is successfully verified
866 through a series of numerical simulations.

867 – By conducting the fracture process analyses of rock due to quasi-static loading (the
868 uniaxial compression test and Brazilian test) and dynamic loading (blasting), the obtained
869 results successfully demonstrated the capability of the developed GPGPU-parallelized Y-
870 HFDEM IDE for various types of loading configurations. From the obtained results, it
871 can be concluded that for dynamic simulation including stress wave propagation in rock,
872 the correct selection of penalty terms for CE4s using the ICZM-based approach is very
873 important.

874 – The GPGPU-parallelized Y-HFDEM IDE can run on various GPGPU accelerators
875 of different generations. However, the comparison of the runtimes from the simulation of

876 the uniaxial compression test concludes that GPGPU accelerators of different generations
877 perform in drastically different ways, and thus the proper selection of the GPGPU
878 accelerators is suggested. Remarkably, the GPGPU-parallelized Y-HFDEM IDE running
879 on the Quadro GP100 GPGPU accelerator achieves a speed-up of 128.6 times compared
880 with the authors' formal sequential CPU-based code. It must be emphasized that this
881 performance is obtained using a single GPGPU accelerator.

882 Therefore, the GPGPU-parallelized Y-HFDEM IDE developed in this study is a
883 valuable and powerful numerical tool for rock engineering applications. However, further
884 work is needed. The following are the highlights of our future tasks.

885 – The main purpose of this paper was to explain the newly developed GPGPU-
886 parallelized Y-HFDEM IDE, and we intentionally selected the simpler examples for the
887 verifications and validations of the developed code because complex problems make the
888 verifications very challenging. Thus, we will apply the developed GPGPU-based code in
889 wide range of rock engineering problems, such as mechanical rock cutting and rock
890 blasting, in the next phase.

891 – This study focused on the development and verification of 2D GPGPU-based Y-
892 HFDEM IDE; however, the authors have already developed the prototype of the 3D
893 version of the GPGPU-based hybrid FDEM based on both ICZM and ECZM. The
894 verification and validation of the 3D GPGPU-parallelized Y-HFDEM IDE are in active
895 progress and will be presented in a separate study.

896 – Because multiple GPGPU accelerators can reside in a single ordinary workstation or
897 even in a GPGPU cloud/cluster, another important task, which is in active progress, is to
898 implement the code using multiple GPGPU accelerators to solve much larger problems
899 using the GPGPU-based hybrid FDEM. In this case, the application of MPI is
900 indispensable for the multiple GPGPU accelerators to communicate with each other.

901 – For the blasting simulation, although we implemented the approach used in An et al.
902 ³⁹ to model the blast-induced pressure, this approach must be improved to realize a more
903 precise blasting simulation. To achieve a better blasting simulation, we are currently

904 working on coupling the GPGPU-based hybrid FDEM with GPGPU-based smoothed
905 particle hydrodynamics ⁶⁵ to model the expansion of the blast-induced gas and its
906 interaction with rock, including newly created fracture surfaces. The development is still
907 in active progress.

908

909 **Acknowledgments**

910 The first author of this work was supported by JSPS KAKENHI (Grant Number
911 JP18K14165) for Grant-in-Aid for Young Scientists. The corresponding author is
912 thankful for the support of the AJF (No. 17/20470), AAS (No. RI8) and IRGS (No.
913 L0018929) grants of Australia and the National Science Foundation grants of China (No.
914 51574060 and No. 51079017). All authors would like to thank the anonymous reviewers
915 for their helpful and constructive comments that significantly contributed to improving
916 the final version of the paper.

917

918 **Appendix A.1**

919 Fig. A.1 shows an example GPGPU “kernel” for solving Eq. (17), i.e., the 2D
920 mechanical solver for nodes with x and y degrees of freedom. To enhance the readability
921 of the code, the assignment of the boundary condition is not shown and only the update of
922 the nodal velocity and the coordinate in the x direction is shown in this example. The
923 name of the “kernel”, i.e., the name of this function, is “Ysd2MEC_GPU”. As is evident
924 from this example, the appearance of the CUDA C/C++ code is very similar to the
925 sequential CPU-based C/C++ code. Variables with names ending in “DEV” are global
926 data stored on GPGPU global memory ⁴⁸ and their meanings are given in the example
927 code. The speed of GPGPU global memory access is relatively slow, and thus access
928 should be minimized to improve performance. This “kernel” is launched by
929 “Ysd2MEC_GPU<<< N_{BpG} , N_{TpB} >>>” from the host C/C++ code, which is again very
930 similar to ordinary C/C++ except that the specification of (N_{BpG}, N_{TpB}) is necessary (see
931 subsection 2.2). When the “kernel” is launched, each CUDA thread is assigned its unique

932 thread ID (*threadIdx.x*) and block ID (*blockIdx.x*) and N_{TpB} is automatically stored in
 933 “*blockDim.x*” (see Figs. 3 and 4). Thus, “*threadIdx.x + blockIdx.x * blockDim.x*” in the
 934 code can be considered a unique node ID “*inopo*”. If each node ID is within the range of
 935 the total number of existing nodes (*current_num_nodes_CST*), the mechanical solver is
 936 processed for the node IDs in completely parallel manner. Note that variables with names
 937 ending in “CST” are stored in the GPGPU constant memory ⁴⁸, which can achieve faster
 938 memory access than the GPGPU global memory. However, because memory size is
 939 limited, GPGPU constant memory is used to store the constant values, such as
 940 mechanical properties and topological data, that do not change throughout the simulation.
 941 Finally, using the “*register*” ⁴⁸ for the declaration of local variables in each “*thread*” can
 942 achieve the fastest memory access. However, the total size of the register memory in each
 943 “*thread*” is quite limited, and defining too many local variables using the “*register*”
 944 keyword results in poor performance. In the developed code, we minimized the variables
 945 declared with the “*register*” keyword and used them repeatedly for several meanings.
 946 Thus, the readability of the actual code was slightly compromised.

947

948 References

- 949 1. Mohammadnejad M, Liu H, Chan A, Dehkhoda S, Fukuda D. An
 950 overview on advances in computational fracture mechanics of rock.
 951 *Geosystem Engineering*. 2018;1-24.
- 952 2. Lisjak A, Grasselli G. A review of discrete modeling techniques for
 953 fracturing processes in discontinuous rock masses. *Journal of Rock*
 954 *Mechanics and Geotechnical Engineering*. 2014;6(4):301-314.
- 955 3. Lei Q, Latham J-P, Tsang C-F. The use of discrete fracture networks for
 956 modelling coupled geomechanical and hydrological behaviour of fractured
 957 rocks. *Computers and Geotechnics*. 2017;85:151-176.
- 958 4. Munjiza AA. *The combined finite-discrete element method*. John Wiley &
 959 Sons; 2004.
- 960 5. Munjiza A. *The Combined Finite-Discrete Element Method*. Wiley; 2004.
- 961 6. Xiang J, Munjiza A, Latham JP. Finite strain, finite rotation quadratic
 962 tetrahedral element for the combined finite–discrete element method.
 963 *International Journal for Numerical Methods in Engineering*.
 964 2009;79(8):946-978.
- 965 7. Munjiza A, Xiang J, Garcia X, Latham JP, D’Albano GGS, John NWM.
 966 The Virtual Geoscience Workbench, VGW: Open Source tools for
 967 discontinuous systems. *Particuology*. 2010;8(2):100-105.

- 968 8. Lukas T, Schiava D'Albano GG, Munjiza A. Space decomposition based
969 parallelization solutions for the combined finite–discrete element method
970 in 2D. *Journal of Rock Mechanics and Geotechnical Engineering*.
971 2014;6(6):607-615.
- 972 9. Rockfield. Rockfield Software Ltd. 2005;
973 <http://www.rockfield.co.uk/elfen.htm>.
- 974 10. Elmo D, Stead D. An Integrated Numerical Modelling–Discrete Fracture
975 Network Approach Applied to the Characterisation of Rock Mass Strength
976 of Naturally Fractured Pillars. *Rock Mechanics and Rock Engineering*.
977 2010;43(1):3-19.
- 978 11. Hamdi P, Stead D, Elmo D. Damage characterization during laboratory
979 strength testing: A 3D-finite-discrete element approach. *Computers and*
980 *Geotechnics*. 2014;60:33-46.
- 981 12. Rogers S, Elmo D, Webb G, Catalan A. Volumetric Fracture Intensity
982 Measurement for Improved Rock Mass Characterisation and
983 Fragmentation Assessment in Block Caving Operations. *Rock Mechanics*
984 *and Rock Engineering*. 2015;48(2):633-649.
- 985 13. Mahabadi OK, Lisjak A, Munjiza A, Grasselli G. Y-Geo: New Combined
986 Finite-Discrete Element Numerical Code for Geomechanical Applications.
987 *International Journal of Geomechanics*. 2012;12(6):676-688.
- 988 14. Lisjak A. *Investigating the influence of mechanical anisotropy on the*
989 *fracturing behaviour of brittle clay shales with application to deep*
990 *geological repositories*: Civil Engineering, University of Toronto; 2014.
- 991 15. Lisjak A, Figi D, Grasselli G. Fracture development around deep
992 underground excavations: Insights from FDEM modelling. *Journal of*
993 *Rock Mechanics and Geotechnical Engineering*. 2014;6(6):493-505.
- 994 16. Lisjak A, Grasselli G, Vietor T. Continuum–discontinuum analysis of
995 failure mechanisms around unsupported circular excavations in anisotropic
996 clay shales. *International Journal of Rock Mechanics and Mining*
997 *Sciences*. 2014;65:96-115.
- 998 17. Mahabadi O, Kaifosh P, Marschall P, Vietor T. Three-dimensional FDEM
999 numerical simulation of failure processes observed in Opalinus Clay
1000 laboratory samples. *Journal of Rock Mechanics and Geotechnical*
1001 *Engineering*. 2014;6(6):591-606.
- 1002 18. Tatone BSA, Grasselli G. A calibration procedure for two-dimensional
1003 laboratory-scale hybrid finite–discrete element simulations. *International*
1004 *Journal of Rock Mechanics and Mining Sciences*. 2015;75:56-72.
- 1005 19. Lisjak A, Liu Q, Zhao Q, Mahabadi OK, Grasselli G. Numerical
1006 simulation of acoustic emission in brittle rocks by two-dimensional finite-
1007 discrete element analysis. *Geophysical Journal International*.
1008 2013;195(1):423-443.
- 1009 20. Mahabadi O, Lisjak A, He L, Tatone B, Kaifosh P, Grasselli G.
1010 Development of a new fully-parallel finite-discrete element code: Irazu.
1011 Paper presented at: 50th US Rock Mechanics/Geomechanics
1012 Symposium 2016.
- 1013 21. Lisjak A, Kaifosh P, He L, Tatone BSA, Mahabadi OK, Grasselli G. A
1014 2D, fully-coupled, hydro-mechanical, FDEM formulation for modelling
1015 fracturing processes in discontinuous, porous rock masses. *Computers and*
1016 *Geotechnics*. 2017;81:1-18.

- 1017 22. Lisjak A, Mahabadi OK, He L, et al. Acceleration of a 2D/3D finite-
1018 discrete element code for geomechanical simulations using General
1019 Purpose GPU computing. *Computers and Geotechnics*. 2018;100:84-96.
- 1020 23. Xiang J, Latham J-P, Farsi A. Algorithms and Capabilities of Solidity to
1021 Simulate Interactions and Packing of Complex Shapes. 2016; Singapore.
- 1022 24. Latham J, Guo L, Wang X, Xiang J. Modelling the evolution of fractures
1023 using a combined FEM-DEM numerical method. 2012.
- 1024 25. Guo L, Xiang J, Latham J-P, Izzuddin B. A numerical investigation of
1025 mesh sensitivity for a new three-dimensional fracture model within the
1026 combined finite-discrete element method. *Engineering Fracture*
1027 *Mechanics*. 2016;151:70-91.
- 1028 26. Lei Q, Latham J-P, Xiang J. Implementation of an Empirical Joint
1029 Constitutive Model into Finite-Discrete Element Analysis of the
1030 Geomechanical Behaviour of Fractured Rocks. *Rock Mechanics and Rock*
1031 *Engineering*. 2016;49(12):4799-4816.
- 1032 27. Guo L, Latham J-P, Xiang J. A numerical study of fracture spacing and
1033 through-going fracture formation in layered rocks. *International Journal*
1034 *of Solids and Structures*. 2017;110:44-57.
- 1035 28. Solidity. Solidity Project. 2017; <http://solidityproject.com/>.
- 1036 29. Guo L. *Development of a three-dimensional fracture model for the*
1037 *combined finite-discrete element method*, Imperial College London; 2014.
- 1038 30. Lei Z, Rougier E, Knight EE, Munjiza A. A framework for grand scale
1039 parallelization of the combined finite discrete element method in 2d.
1040 *Computational Particle Mechanics*. 2014;1(3):307-319.
- 1041 31. Rougier E, Knight EE, Broome ST, Sussman AJ, Munjiza A. Validation of
1042 a three-dimensional Finite-Discrete Element Method using experimental
1043 results of the Split Hopkinson Pressure Bar test. *International Journal of*
1044 *Rock Mechanics and Mining Sciences*. 2014;70:101-108.
- 1045 32. Yan C, Zheng H. A coupled thermo-mechanical model based on the
1046 combined finite-discrete element method for simulating thermal cracking
1047 of rock. *International Journal of Rock Mechanics and Mining Sciences*.
1048 2017;91:170-178.
- 1049 33. Yan C, Zheng H. FDEM-flow3D: A 3D hydro-mechanical coupled model
1050 considering the pore seepage of rock matrix for simulating three-
1051 dimensional hydraulic fracturing. *Computers and Geotechnics*.
1052 2017;81:212-228.
- 1053 34. Yan C, Jiao Y-Y. A 2D fully coupled hydro-mechanical finite-discrete
1054 element model with real pore seepage for simulating the deformation and
1055 fracture of porous medium driven by fluid. *Computers & Structures*.
1056 2018;196:311-326.
- 1057 35. Yan C, Jiao Y-Y, Zheng H. A fully coupled three-dimensional hydro-
1058 mechanical finite discrete element approach with real porous seepage for
1059 simulating 3D hydraulic fracturing. *Computers and Geotechnics*.
1060 2018;96:73-89.
- 1061 36. Yan C, Jiao Y-Y, Yang S. A 2D coupled hydro-thermal model for the
1062 combined finite-discrete element method. *Acta Geotechnica*. 2018.
- 1063 37. Liu HY, Kang YM, Lin P. Hybrid finite-discrete element modeling of
1064 geomaterials fracture and fragment muck-piling. *International Journal of*
1065 *Geotechnical Engineering*. 2015;9(2):115-131.
- 1066 38. Liu HY, Han H, An HM, Shi JJ. Hybrid finite-discrete element modelling
1067 of asperity degradation and gouge grinding during direct shearing of rough

- 1068 rock joints. *International Journal of Coal Science & Technology*.
1069 2016;3(3):295-310.
- 1070 39. An HM, Liu HY, Han H, Zheng X, Wang XG. Hybrid finite-discrete
1071 element modelling of dynamic fracture and resultant fragment casting and
1072 muck-piling by rock blast. *Computers and Geotechnics*. 2017;81:322-345.
- 1073 40. Mohammadnejad M, Liu H, Dehkhoda S, Chan A. Numerical
1074 Investigation of Dynamic Rock Fragmentation in Mechanical Cutting
1075 Using Combined FEM/DEM. 3rd Nordic Rock Mechanics Symposium -
1076 NRMS 2017; 2017/11/3/, 2017; Helsinki, Finland.
- 1077 41. Richard T. *Determination of rock strength from cutting tests*. Master's
1078 thesis: University of Minnesota, Minneapolis, MN, USA Google
1079 Scholar1999.
- 1080 42. Zhou Y, Lin J-S. Modeling the ductile–brittle failure mode transition in
1081 rock cutting. *Engineering Fracture Mechanics*. 2014;127:135-147.
- 1082 43. Guo N, Zhao J. Parallel hierarchical multiscale modelling of hydro-
1083 mechanical problems for saturated granular soils. *Computer Methods in
1084 Applied Mechanics and Engineering*. 2016;305:37-61.
- 1085 44. Guo N, Zhao J. A coupled FEM/DEM approach for hierarchical multiscale
1086 modelling of granular media. *International Journal for Numerical
1087 Methods in Engineering*. 2014;99(11):789-818.
- 1088 45. Wu H, Guo N, Zhao J. Multiscale modeling and analysis of compaction
1089 bands in high-porosity sandstones. *Acta Geotechnica*. 2018;13(3):575-599.
- 1090 46. Wu H, Zhao J, Guo N. Multiscale Insights Into Borehole Instabilities in
1091 High-Porosity Sandstones. *Journal of Geophysical Research: Solid Earth*.
1092 2018;123(5):3450-3473.
- 1093 47. Zhou Y, Lin J-S. On the critical failure mode transition depth for rock
1094 cutting. *International Journal of Rock Mechanics and Mining Sciences*.
1095 2013;62:131-137.
- 1096 48. NVIDIA. Cuda C Programming Guide. 2018;
1097 <http://docs.nvidia.com/cuda/>.
- 1098 49. Zhang L, Quigley SF, Chan AHC. A fast scalable implementation of the
1099 two-dimensional triangular Discrete Element Method on a GPU platform.
1100 *Advances in Engineering Software*. 2013;60-61:70-80.
- 1101 50. Batinić M, Smoljanović H, Munjiza A, Mihanović A. GPU based parallel
1102 FDEM for analysis of cable structures. *Grđevinar*. 2018;69(12.):1085-
1103 1092.
- 1104 51. Liu Q, Jiang Y, Wu Z, Xu X, Liu Q. Investigation of the Rock
1105 Fragmentation Process by a Single TBM Cutter Using a Voronoi Element-
1106 Based Numerical Manifold Method. *Rock Mechanics and Rock
1107 Engineering*. 2018;51(4):1137-1152.
- 1108 52. Li X, Zhang Q, Li J, Zhao J. A numerical study of rock scratch tests using
1109 the particle-based numerical manifold method. *Tunnelling and
1110 Underground Space Technology*. 2018;78:106-114.
- 1111 53. Liu W, Zhu X, Jing J. The analysis of ductile-brittle failure mode
1112 transition in rock cutting. *Journal of Petroleum Science and Engineering*.
1113 2018;163:311-319.
- 1114 54. Zhang Z, Paulino G, Celes W. Extrinsic cohesive modelling of dynamic
1115 fracture and microbranching instability in brittle materials. *International
1116 Journal for Numerical Methods in Engineering*. 2007;72(8):893-923.

- 1117 55. Xianqun H, Chaoshui X. Discrete element modelling of rock cutting: from
1118 ductile to brittle transition. *International Journal for Numerical and*
1119 *Analytical Methods in Geomechanics*. 2015;39(12):1331-1351.
- 1120 56. An B, Tannant DD. Discrete element method contact model for dynamic
1121 simulation of inelastic rock impact. *Computers & Geosciences*.
1122 2007;33(4):513-521.
- 1123 57. Satish N, Harris M, Garland M. Designing efficient sorting algorithms for
1124 manycore GPUs. Paper presented at: Parallel & Distributed Processing,
1125 2009. IPDPS 2009. IEEE International Symposium on 2009.
- 1126 58. Ayachit U. The paraview guide: a parallel visualization application. 2015.
- 1127 59. Cundall PA. Distinct element models of rock and soil structure. *Analytical*
1128 *and Computational Methods in Engineering Rock Mechanics*. 1987:129-
1129 163.
- 1130 60. Karekal S. Modeling Rock Chipping Process In Linear Drag Cutting
1131 Mode. ISRM International Symposium - EUROCK 2012; 2012/5/28/
1132 2012; Stockholm, Sweden.
- 1133 61. Xiang J, Munjiza A, Latham J-P, Guises R. On the validation of DEM and
1134 FEM/DEM models in 2D and 3D. *Engineering Computations*.
1135 2009;26(6):673-687.
- 1136 62. Mahabadi OK. *Investigating the Influence of Micro-scale Heterogeneity*
1137 *and Microstructure on the Failure and Mechanical Behaviour of*
1138 *Geomaterials*: Civil Engineering, University of Toronto; 2012.
- 1139 63. Lysmer J, Kuhlemeyer RL, Institute of T, et al. *Finite dynamic model for*
1140 *infinite media*. Berkeley, Calif.: Dept. of Civil Engineering, Univ. of
1141 California, Institute of Transportation and Traffic Engineering, Soil
1142 Mechanics Laboratory; 1969.
- 1143 64. Cho SH, Kaneko K. Influence of the applied pressure waveform on the
1144 dynamic fracture processes in rock. *International Journal of Rock*
1145 *Mechanics and Mining Sciences*. 2004;41(5):771-784.
- 1146 65. Benz W, Asphaug E. Simulations of brittle solids using smooth particle
1147 hydrodynamics. *Computer physics communications*. 1995;87(1-2):253-
1148 265.
- 1149

1150 Table Titles

1151 **Table 1.** Rock mechanical properties and numerical parameters

1152 **Table 2.** Model details for several h_{ave} values.

1153

1154 Figure Captions

1155 **Fig. 1** Modeling of transition from continuum to discontinuum behavior of rock in 2D Y-
1156 HFDEM IDE. (a) Assembly of TRI3s and CE4s and (b) tensile/shear softening curves in
1157 ICZM.

1158 **Fig. 2.** Elastic–inelastic power function model implemented in 2D Y-HFDEM IDE (after
1159 ⁵⁶ with modification).

1160 **Fig. 3.** The concept of the CUDA programming model using the abstractions of
1161 “*threads*”, “*blocks*” and “*grid*”.

1162 **Fig. 4.** The concept of massively parallel computation for each CUDA “*kernel*” for a
1163 particular purpose.

1164 **Fig. 5.** Flowchart of GPGPU-based 2D Y-HFDEM IDE.

1165 **Fig. 6.** The concept of hash values assigned to each square subcell (left) and TRI3s
1166 included in a particular subcell (right).

1167 **Fig. 7.** Efficient contact detection algorithm used in the developed GPGPU-based 2D Y-
1168 HFDEM IDE.

1169 **Fig. 8.** A model of initial stress analysis under gravity.

1170 **Fig. 9.** The result of initial stress analysis using critical and local damping schemes. (a)
1171 Stress distribution computed and (b) the profiles of σ_{xx} and σ_{yy} along the vertical direction
1172 of the model from the top to the bottom.

1173 **Fig. 10.** Comparison of change in total kinetic energy of the system with respect to
1174 calculation time-steps between local and critical damping schemes.

1175 **Fig. 11.** A simple test for verification of contact damping model. (a) Numerical model
1176 similar to ¹³ and (b) time history of total kinetic energy.

1177 **Fig. 12.** Contact friction verification: (a) model configuration, (b) comparison between
1178 numerical and theoretical results.

1179 **Fig. 13.** Verification models for fracture process of rock under quasi-static loading. (a)
1180 Brazilian indirect tensile test and (b) uniaxial compression test.

1181 **Fig. 14.** The results of the numerical simulation of the Brazilian test. (a) Extension of
1182 shear microcracks before the peak stress and corresponding horizontal stress distribution,
1183 (b) tensile microcracks at the peak stress and corresponding horizontal stress distribution,
1184 (c) postfailure fracture pattern and corresponding horizontal stress distribution and (d)
1185 Brazilian indirect tensile stress versus axial strain.

1186 **Fig. 15.** Fracture process in uniaxial compression test. (a) Initiation and propagation of
 1187 shear microcracks into model before the peak stress, (b) unstable crack propagation at the
 1188 peak stress and (c) postfailure fracture pattern.

1189 **Fig. 16.** Comparison of numerical simulation and experiment for uniaxial compression
 1190 test. (a) Plot of axial stress versus axial strain and (b) final fracture patterns from
 1191 numerical simulation and experimental test.

1192 **Fig. 17.** A numerical model with a single blast-hole used in the dynamic fracture process
 1193 analysis (after ⁶⁴).

1194 **Fig. 18.** The results of dynamic fracture process analysis for a single blast-hole model. (a)
 1195 Spatial distribution of PS1 and broken CE4s at selected time intervals and (b) the results
 1196 of dynamic fracture process analysis using different penalty numbers for CE4s.

1197 **Fig. 19.** Relative speed-up of the GPGPU-based code using different GPGPU accelerators
 1198 to the sequential CPU-based code.

1199 **Fig. 20.** Change in relative speed-up with respect to the number of TRI3s used in FDEM
 1200 mesh for the simulation of uniaxial compression test.

1201 **Fig. A.1.** An example of GPGPU “kernel” in 2D Y-HFDEM IDE.

1202

1203 **Table 1**

<i>Parameter</i>	<i>Unit</i>	<i>Value</i>
Density (ρ)	kg/m ³	1800
Young’s modulus (E)	GPa	12.2
Poisson’s ratio (ν)	-	0.25
Tensile strength (T_s)	MPa	1.77
Cohesion (c)	MPa	5
Internal Friction angle of intact rock (ϕ)	°	25
Mode I fracture energy (G_{fl})	J/m ²	16
Mode II fracture energy (G_{fII})	J/m ²	160
Normal contact penalty number (P_{n_con})	GPa	1220

Tangent contact penalty number ($P_{\text{tan_con}}$)	GPa/m	1220
Fracture penalty numbers ($P_f, P_{\text{tan}}, P_{\text{overlap}}$)	GPa	12200
Average element size (h_{ave})	mm	0.7
Critical viscous damping factor (η)	kg/m.s	5.60E+03

1204 **Table 2**

$h_{\text{ave}} / (\text{mm})$	0.15	0.2	0.3	0.4	0.5	0.6
The number of nodes	3,253,194	1,893,901	723,894	410,322	303,537	181,038
The number of TRI3s	1,084,398	540,754	241,298	136,774	86,570	60,346
The number of CE4s	1,624,686	809,872	306,990	204,355	129,316	90,040
The initial number of contact couples.	6,970,705	4,348,700	1,943,205	1,096,737	686,954	476,433
Simulation time (sec) for sequential CPU-based code /10000 steps	108,767	42,021	19,343	10,515	9,146	6,231
Simulation time (sec) for GPUGU-based code (Quadro GP100) /10000 steps	846	392	214	146	134	126

1205

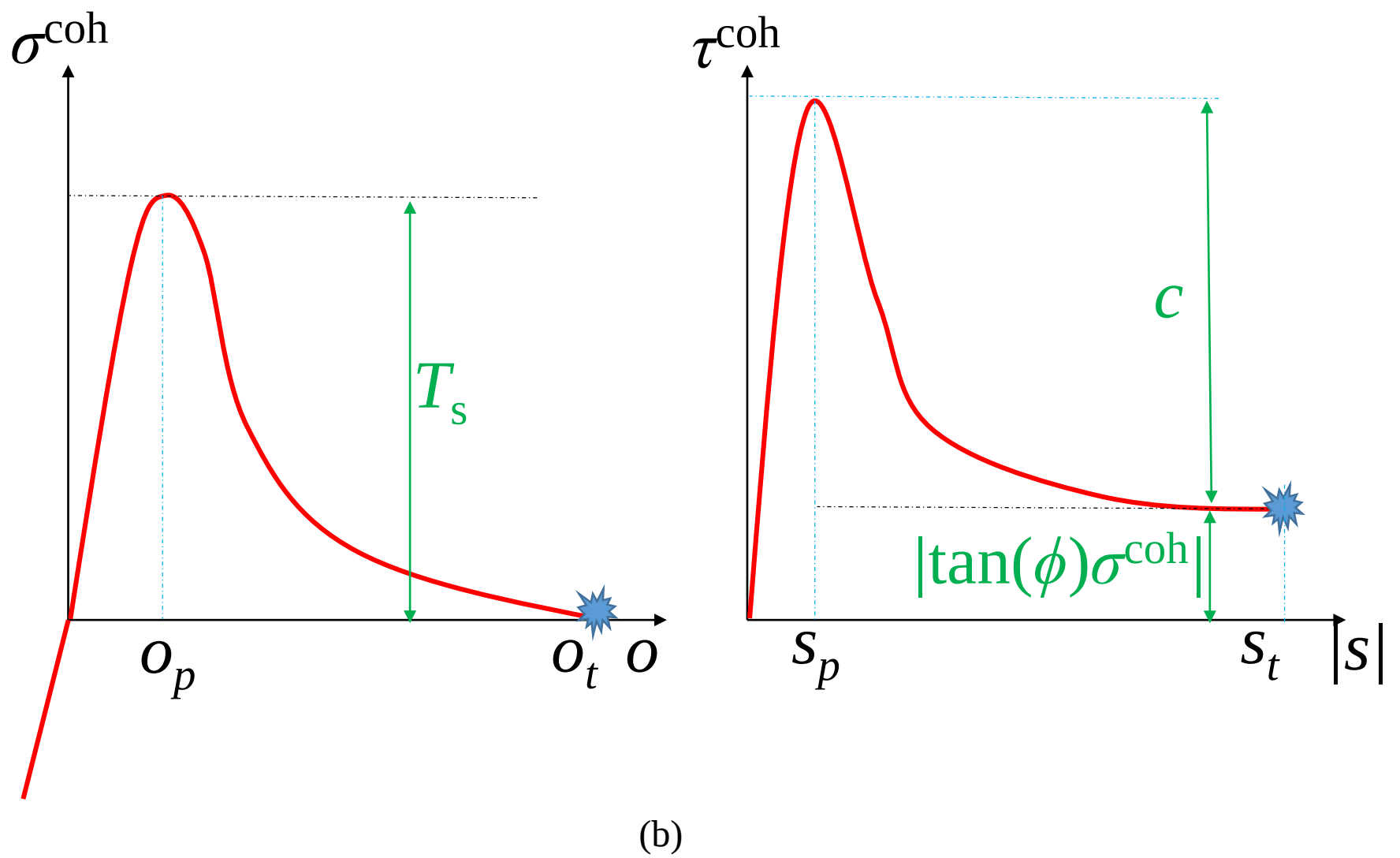
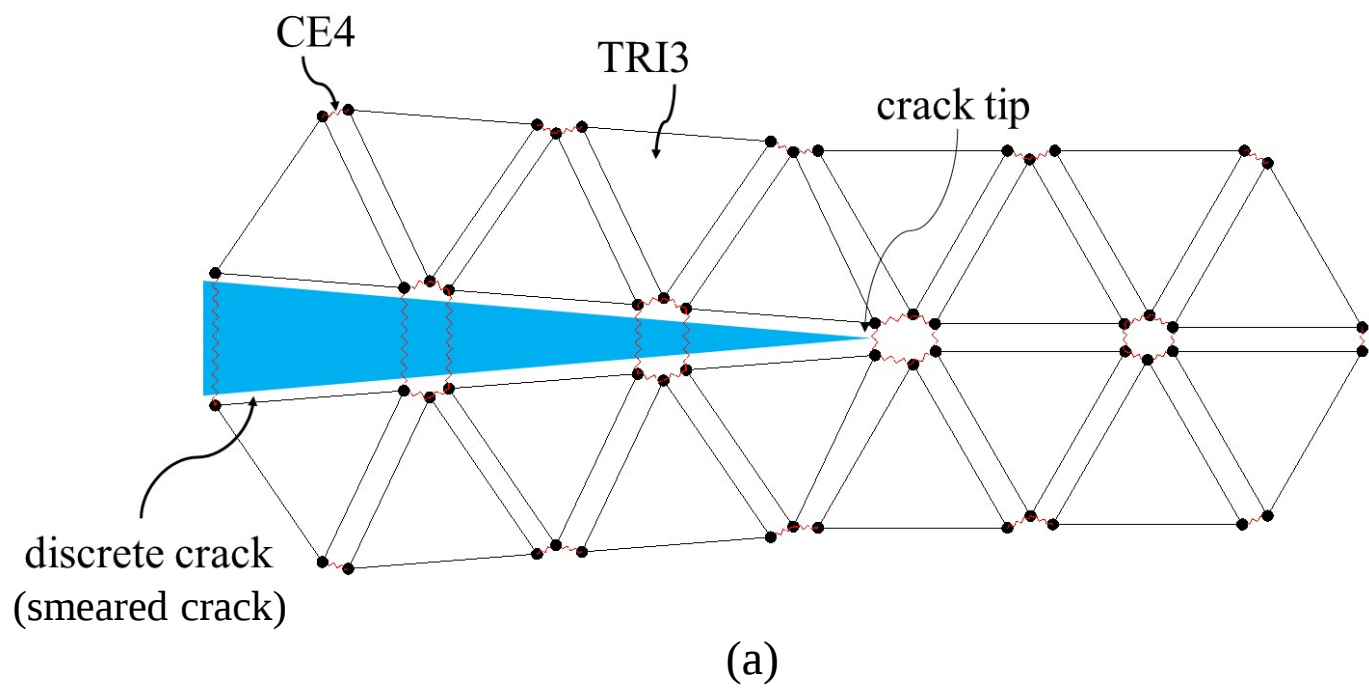


Fig. 1.

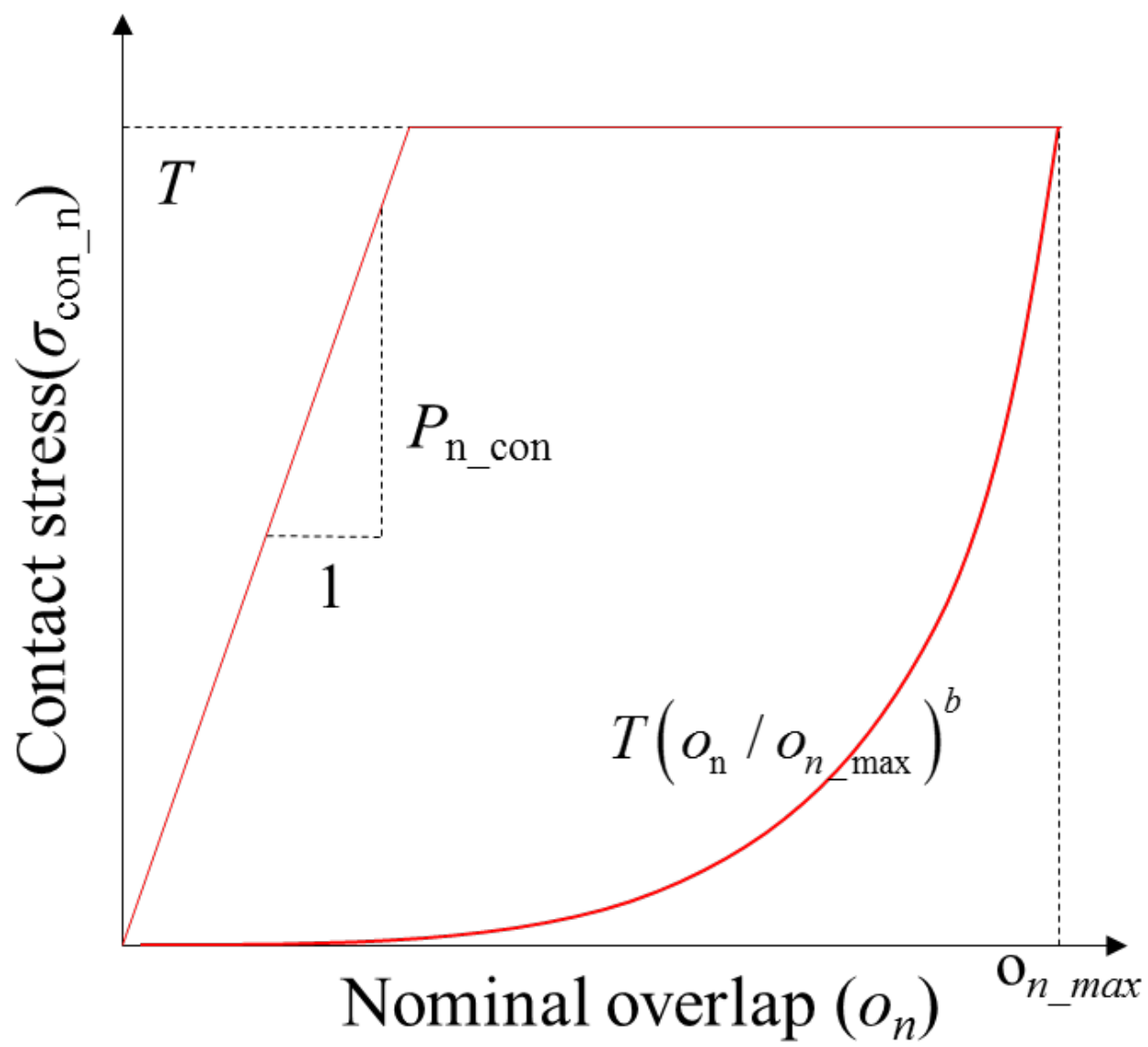


Fig. 2.

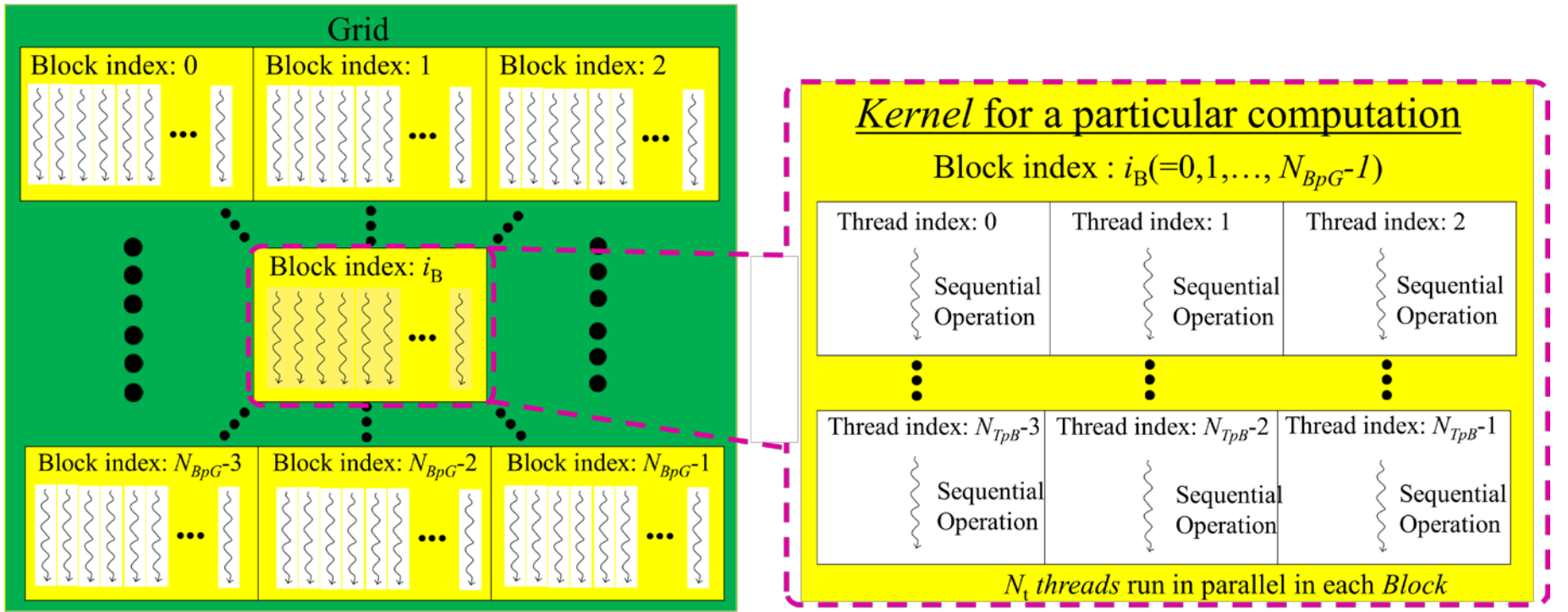


Fig. 3.

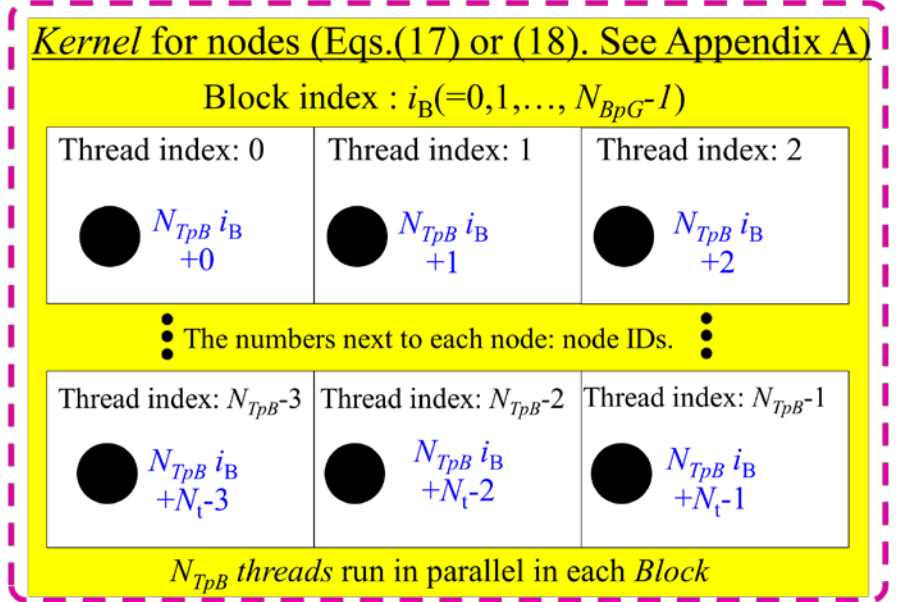
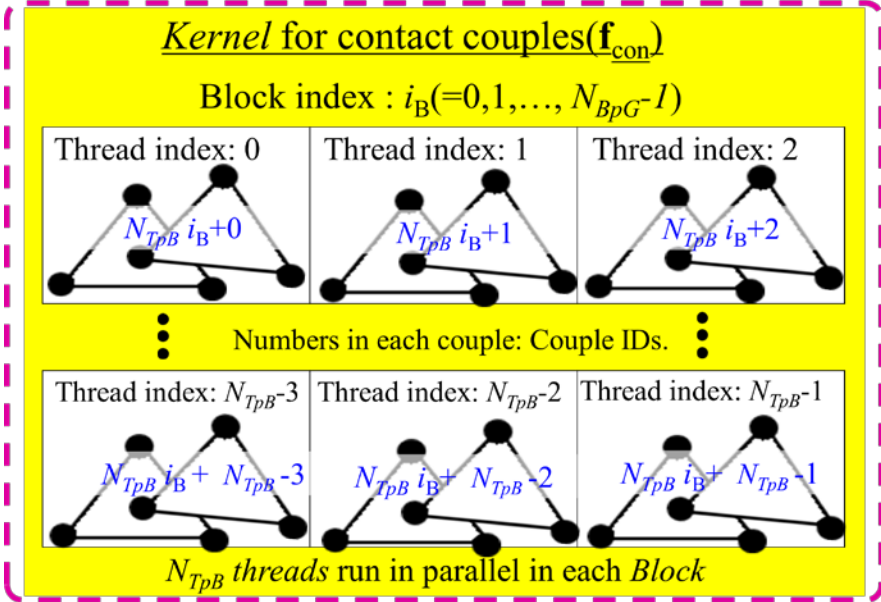
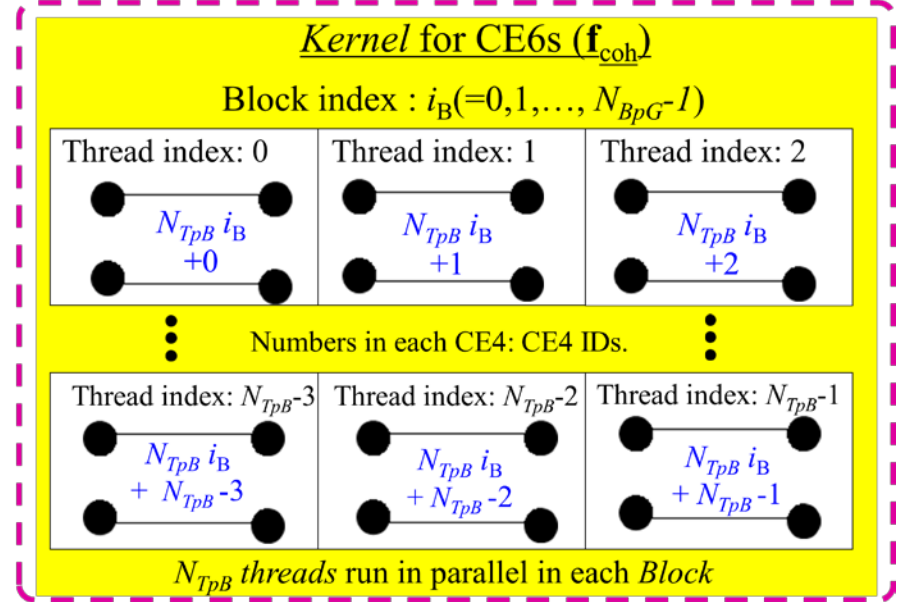
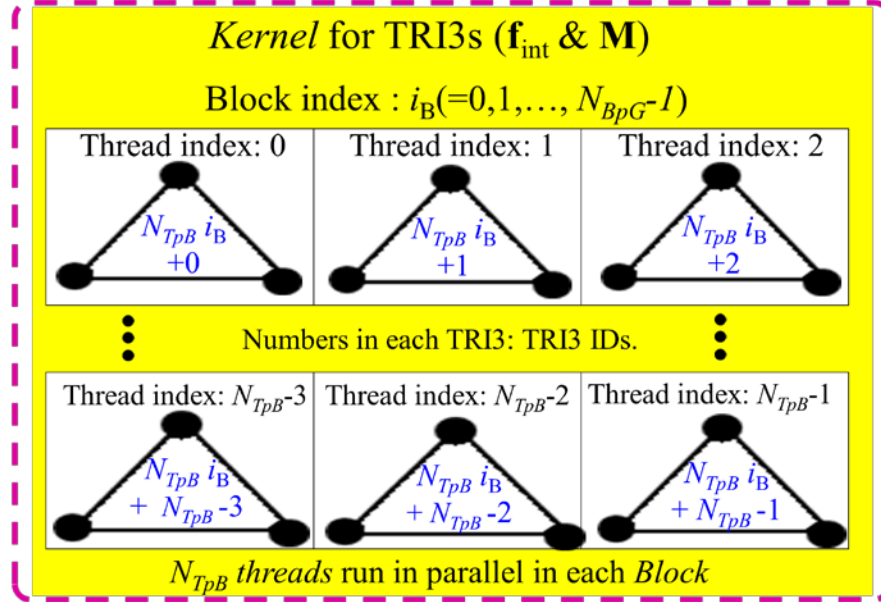


Fig. 4.

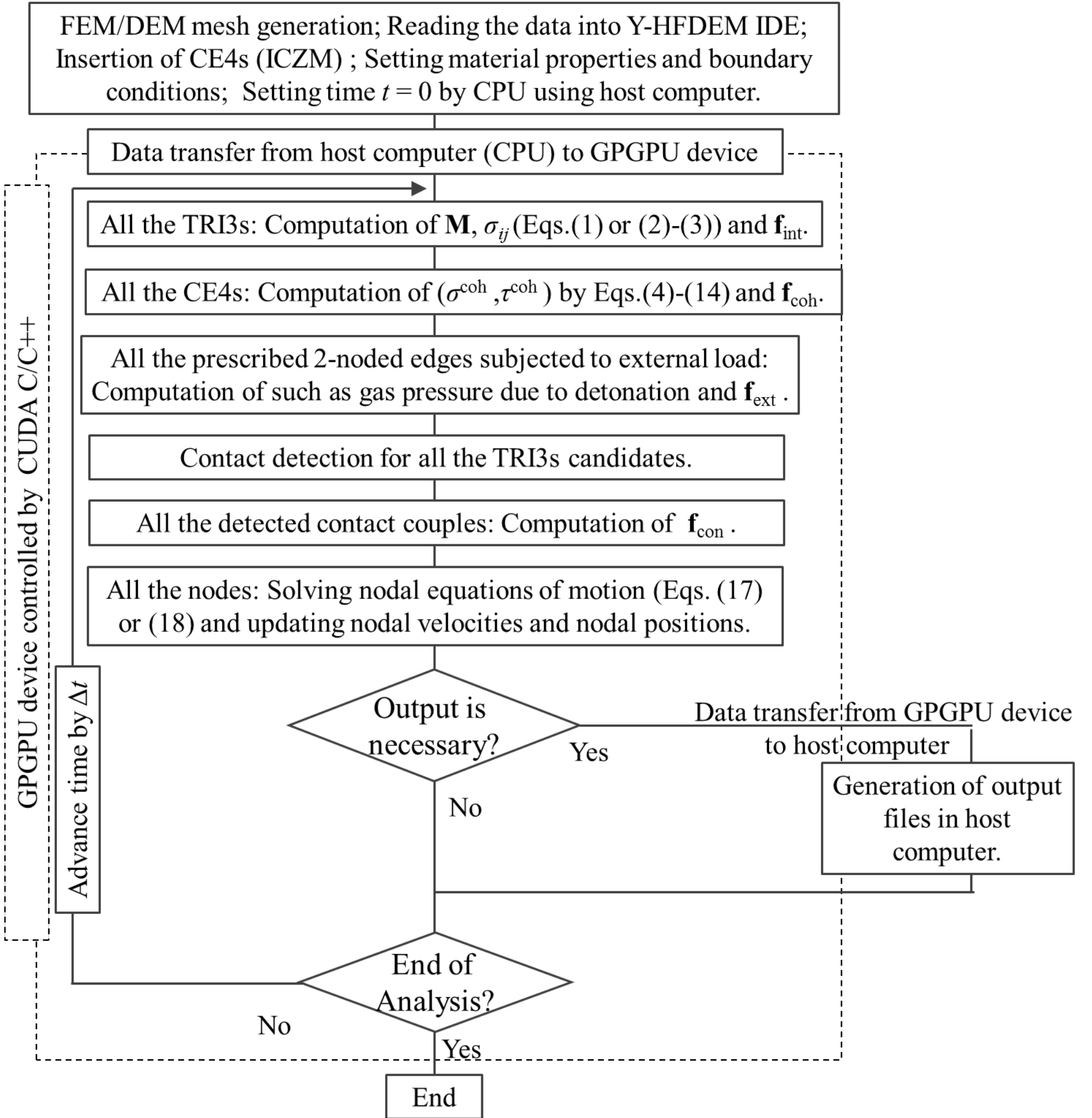


Fig. 5.

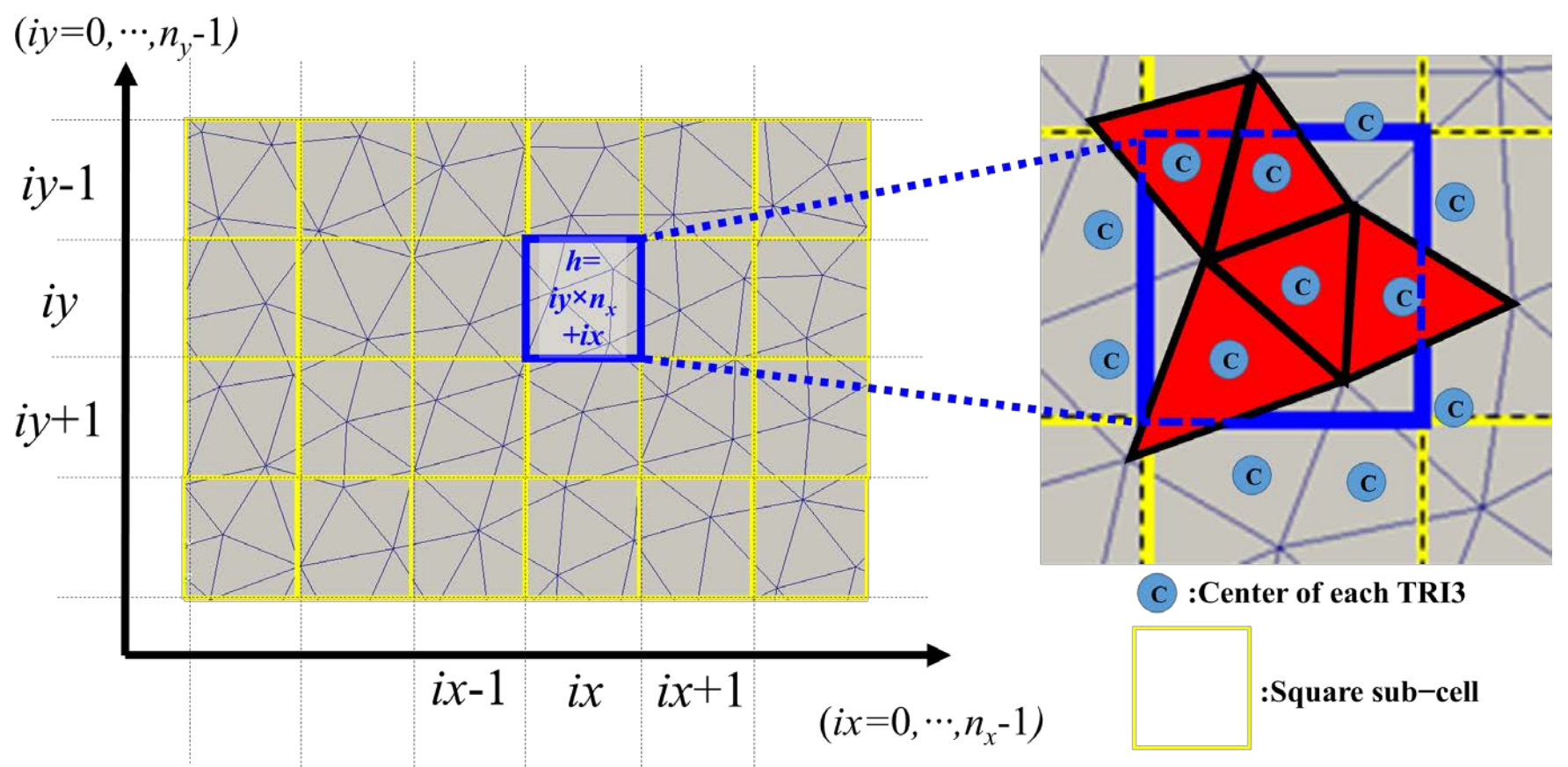
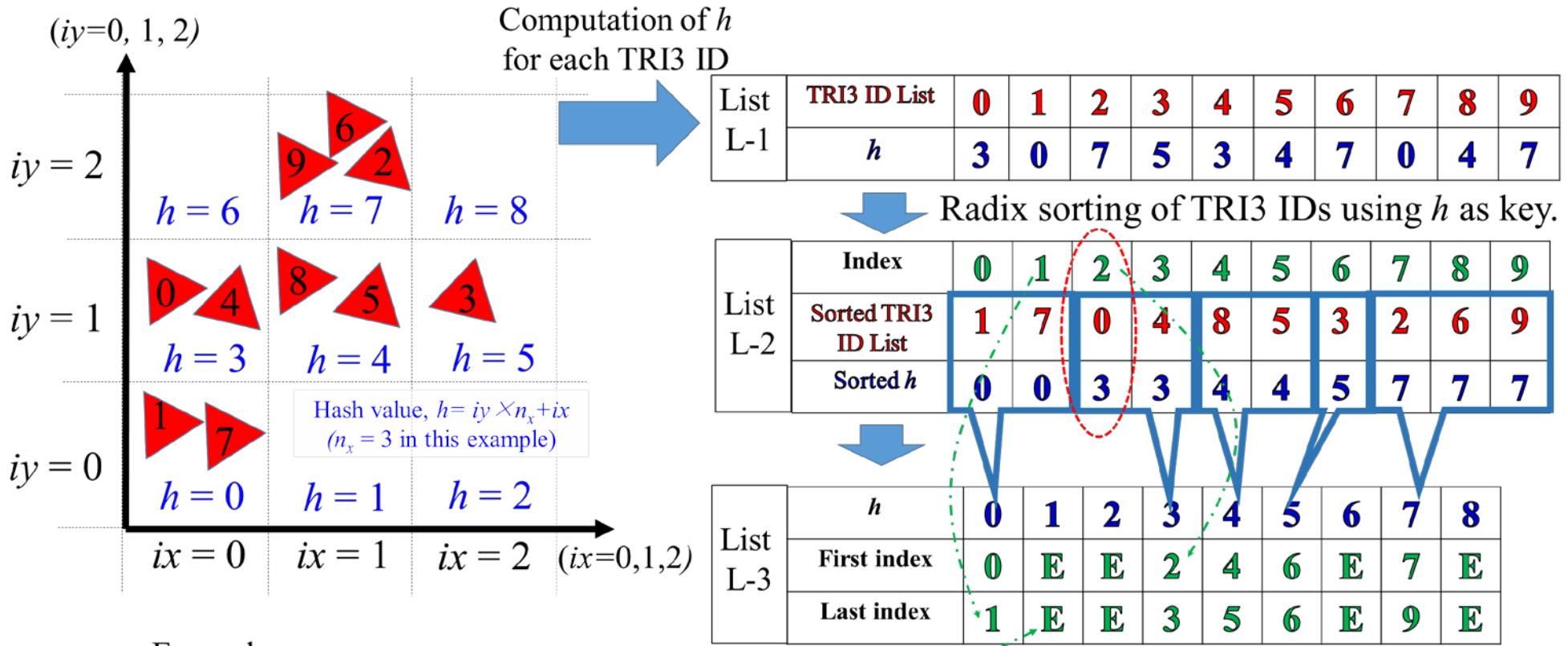


Fig. 6.



Example:

In the case of TRI3 ID = 5 ($ix=1, iy=1, h=4$)

→ Hash values of neighbour nine cells including $h=4$ are (0, 1, 2, 3, 4, 5, 6, 7, 8).

→ Using the lists L-2 and L-3, all the neighbour TRI3 IDs around TRI3 ID = 5 are readily available and these are subjected to contact detection

E: Empty (There are no TRI3s for the value of h)

Fig. 7.

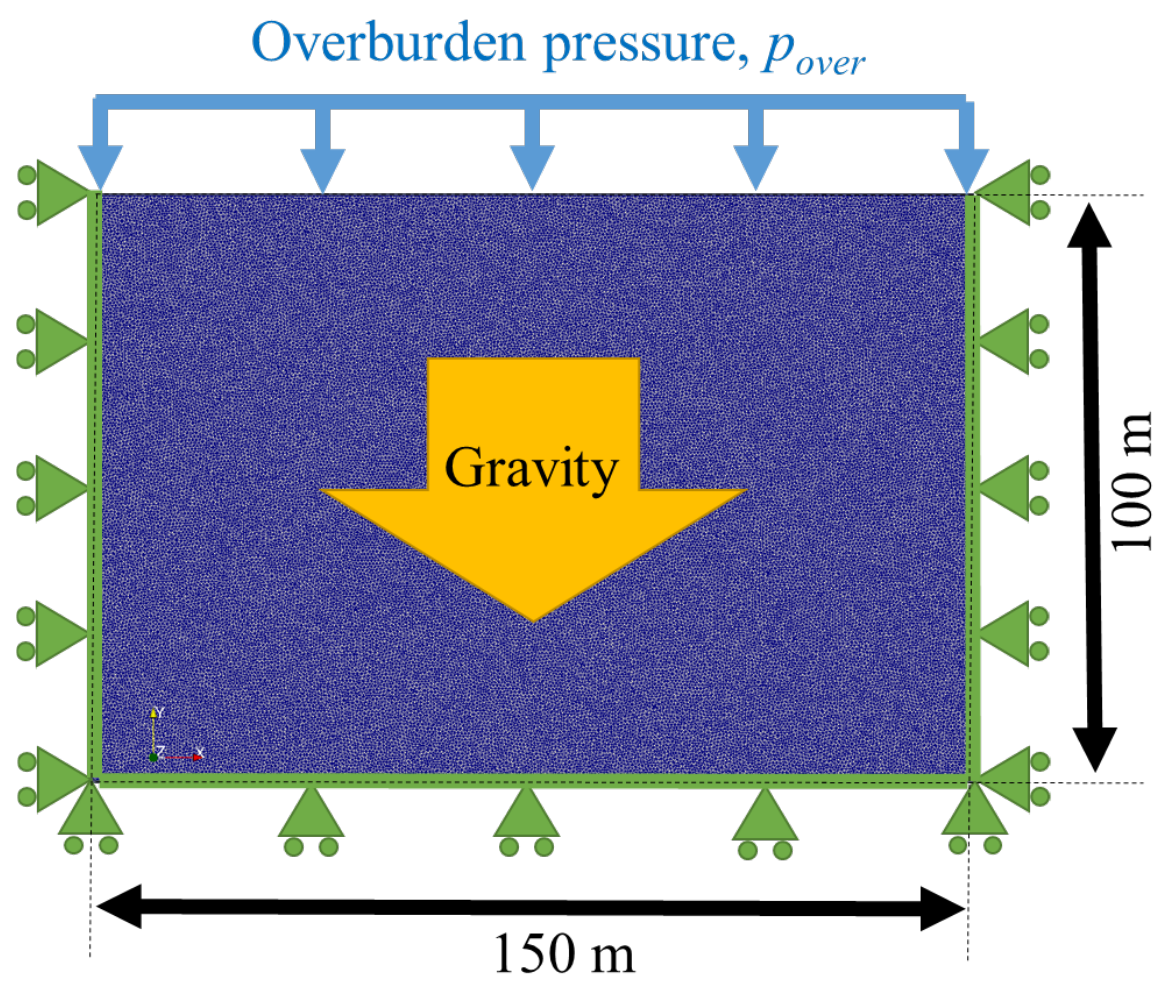
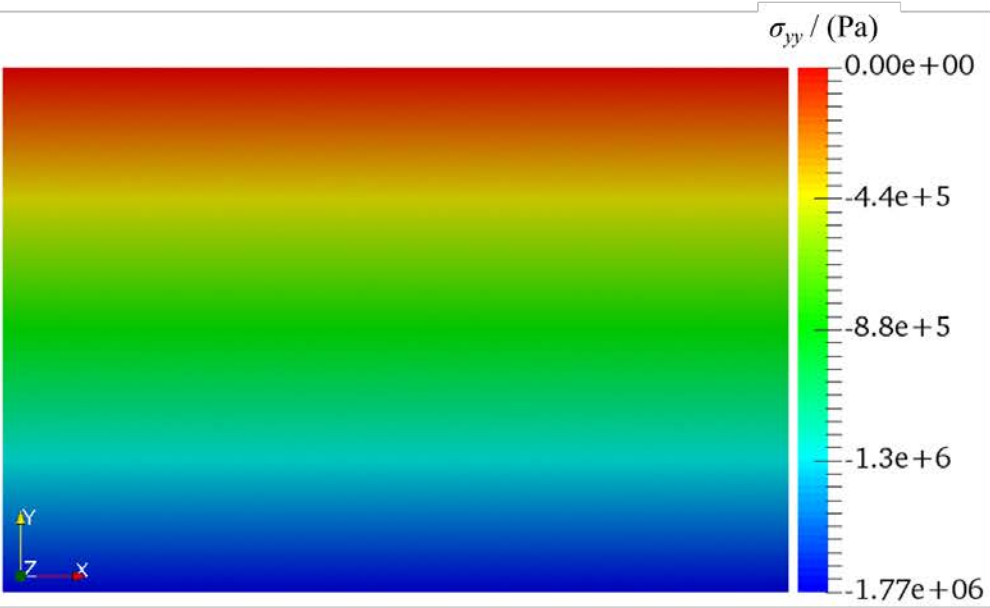
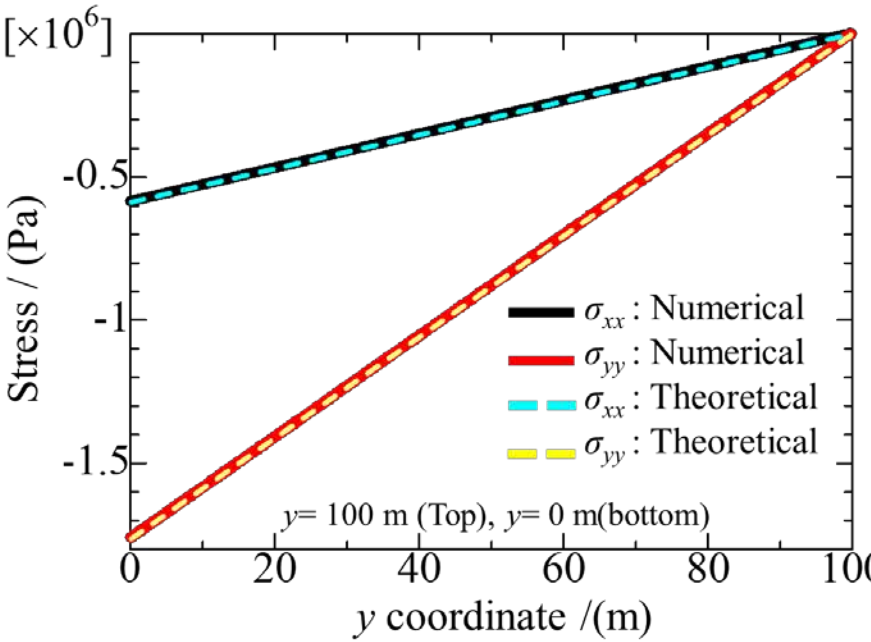


Fig. 8.



(a)



(b)

Fig. 9.

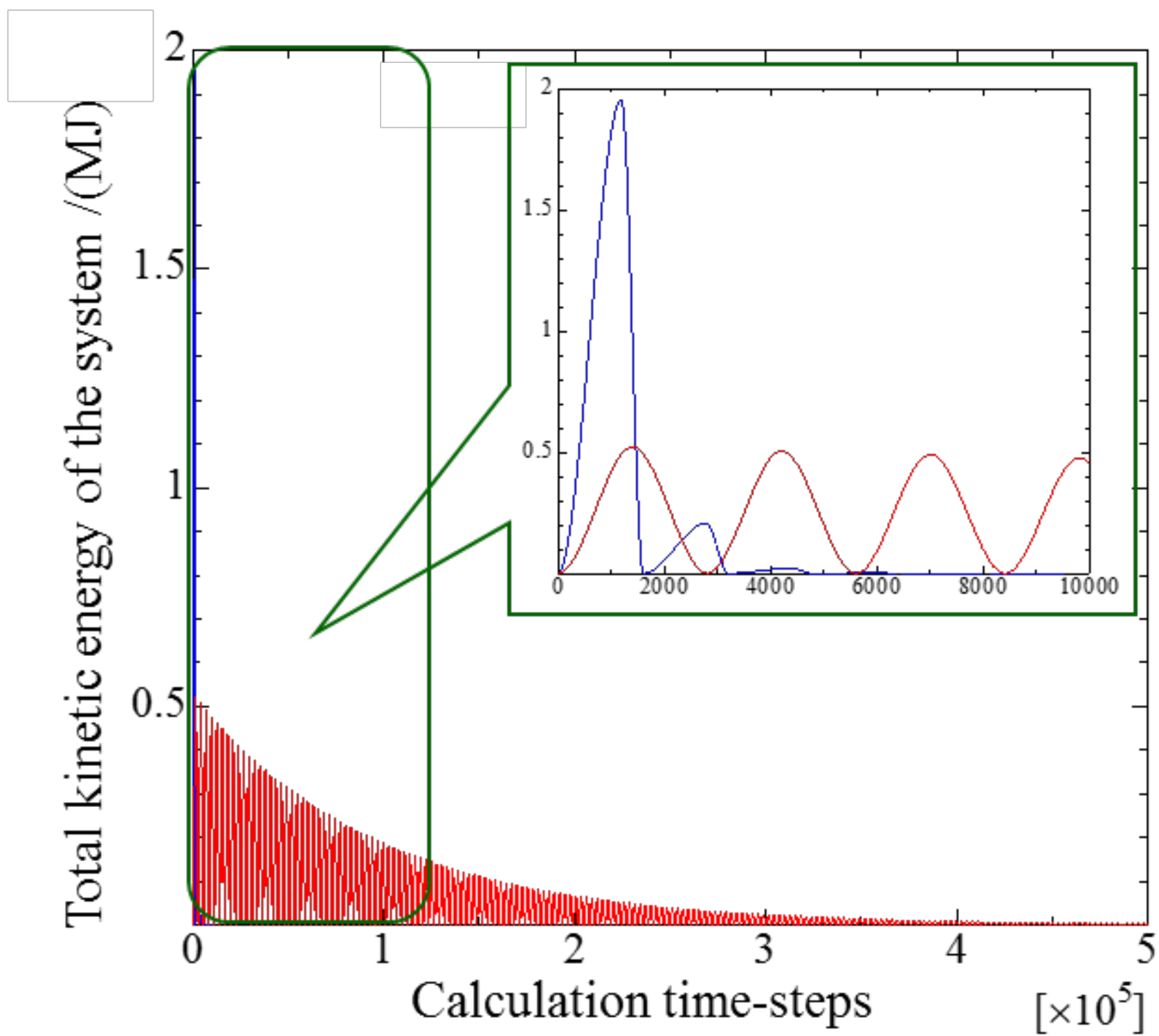
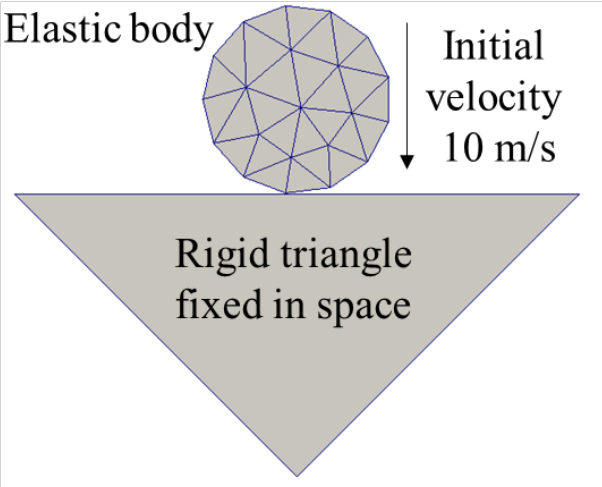
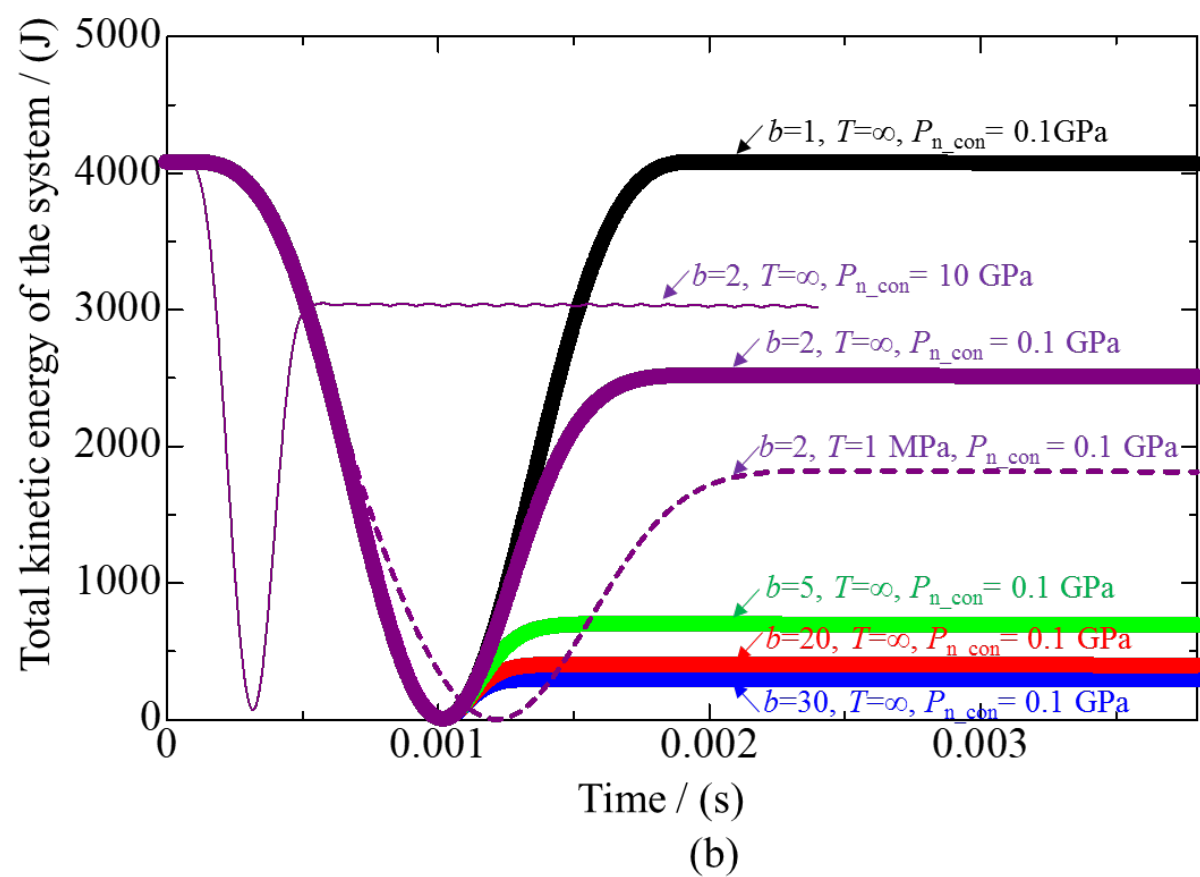


Fig. 10.

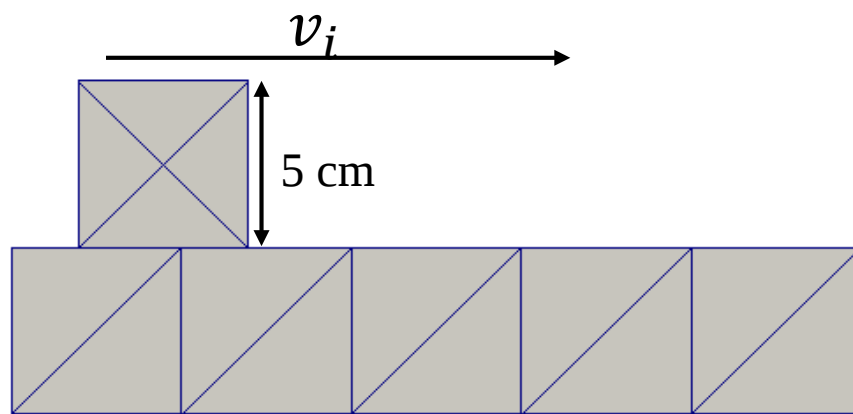


(a)

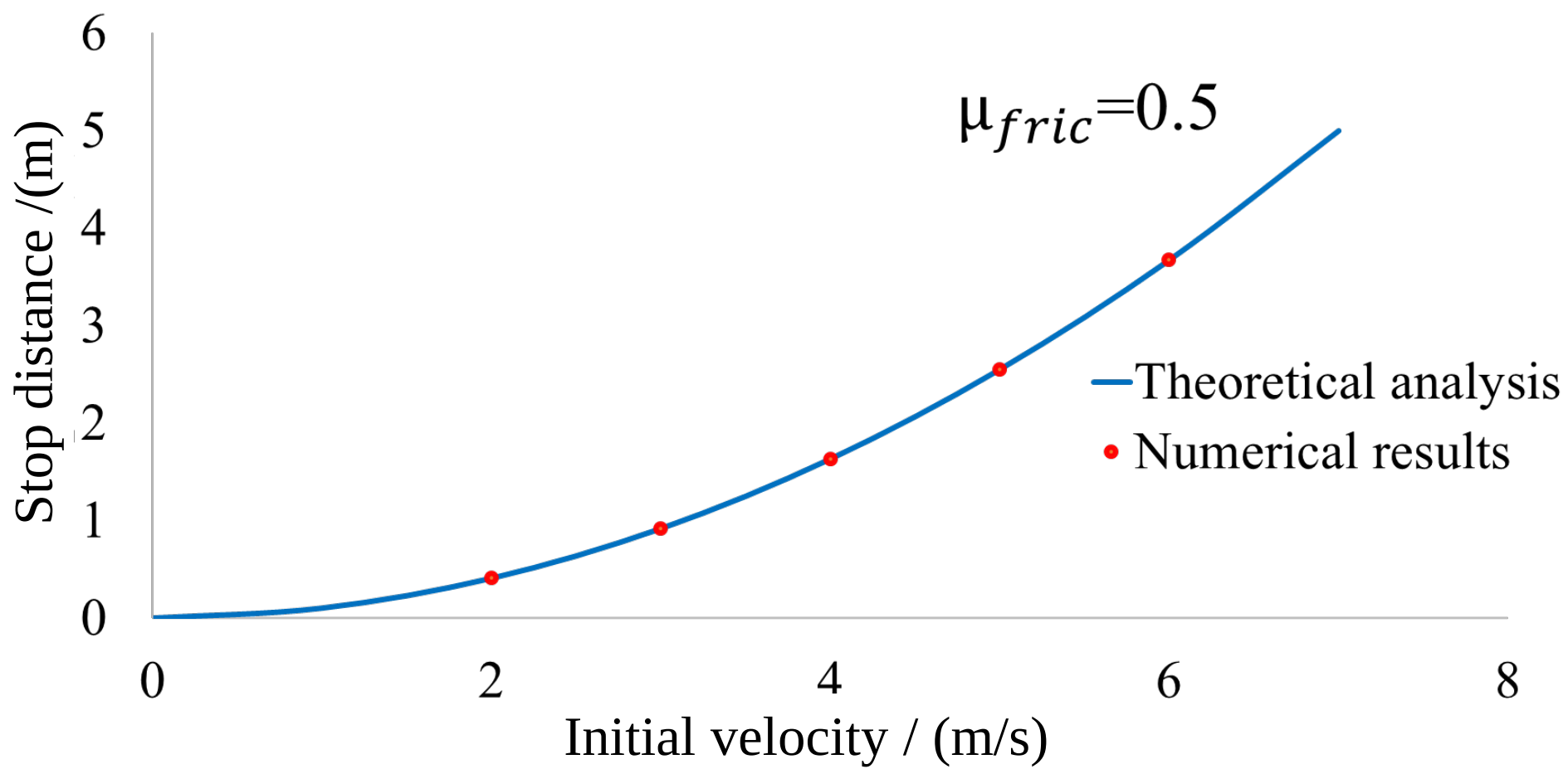


(b)

Fig. 11.



(a)



(b)

Fig. 12.

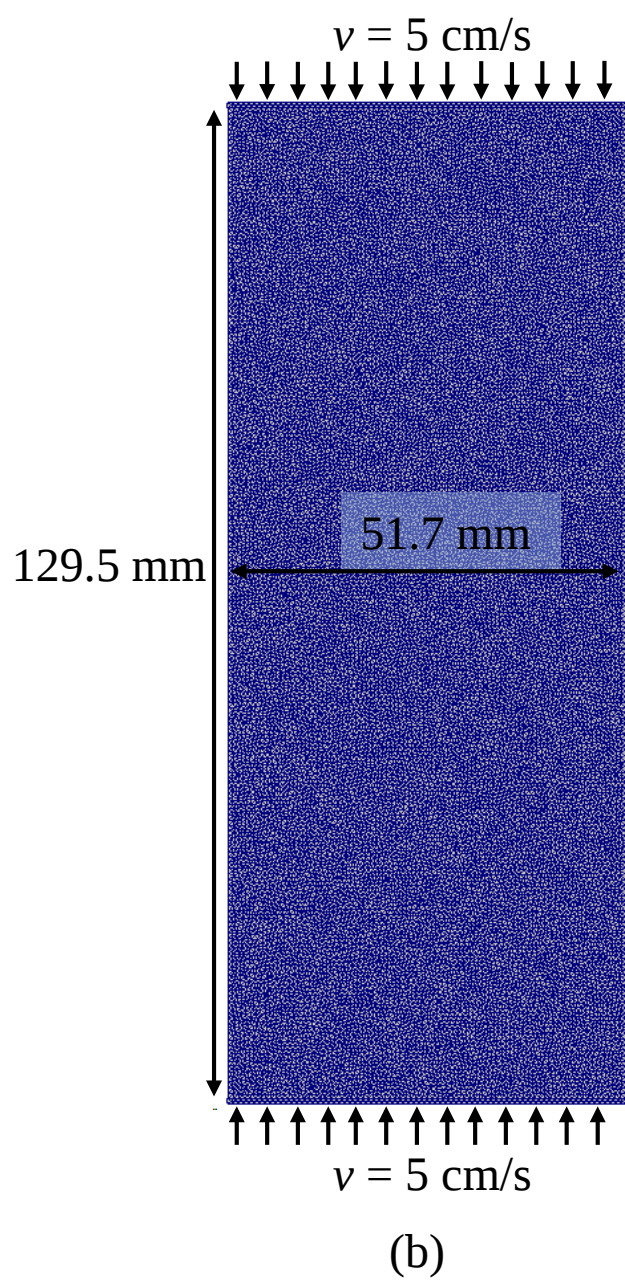
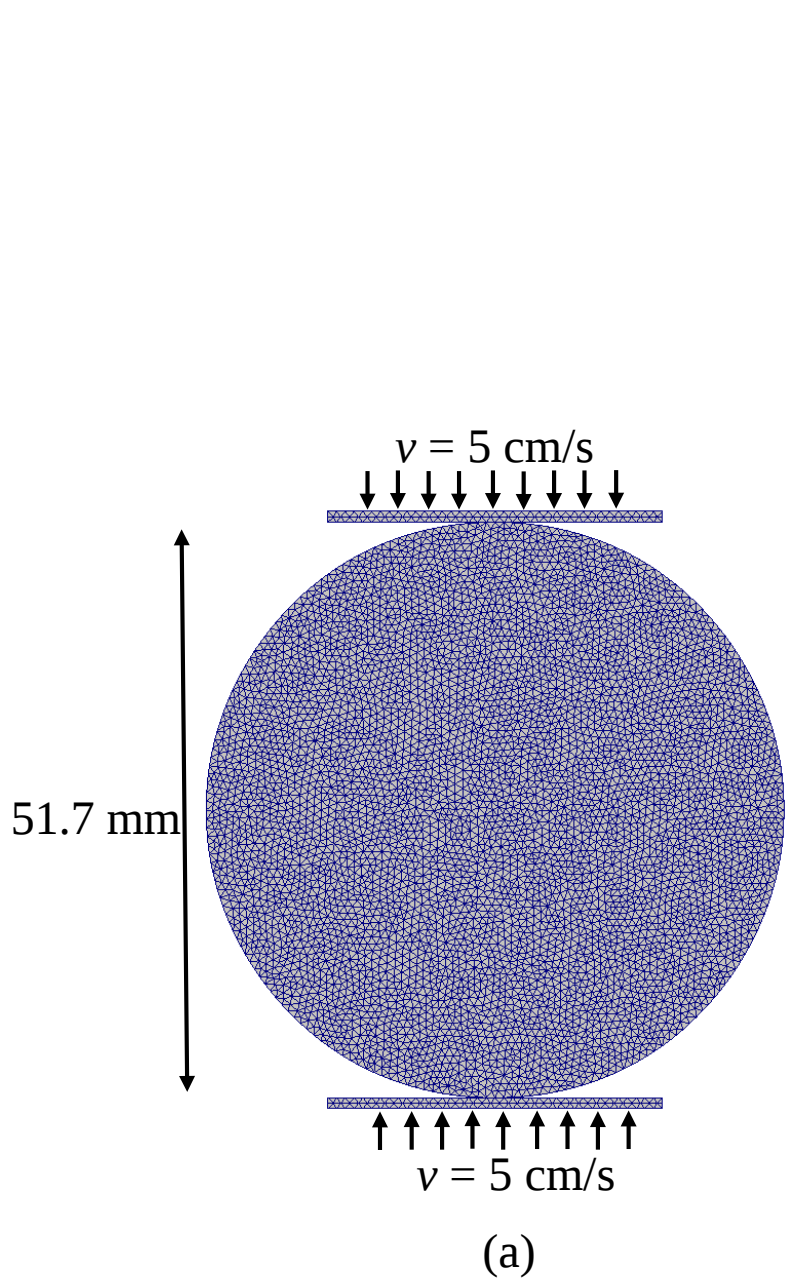
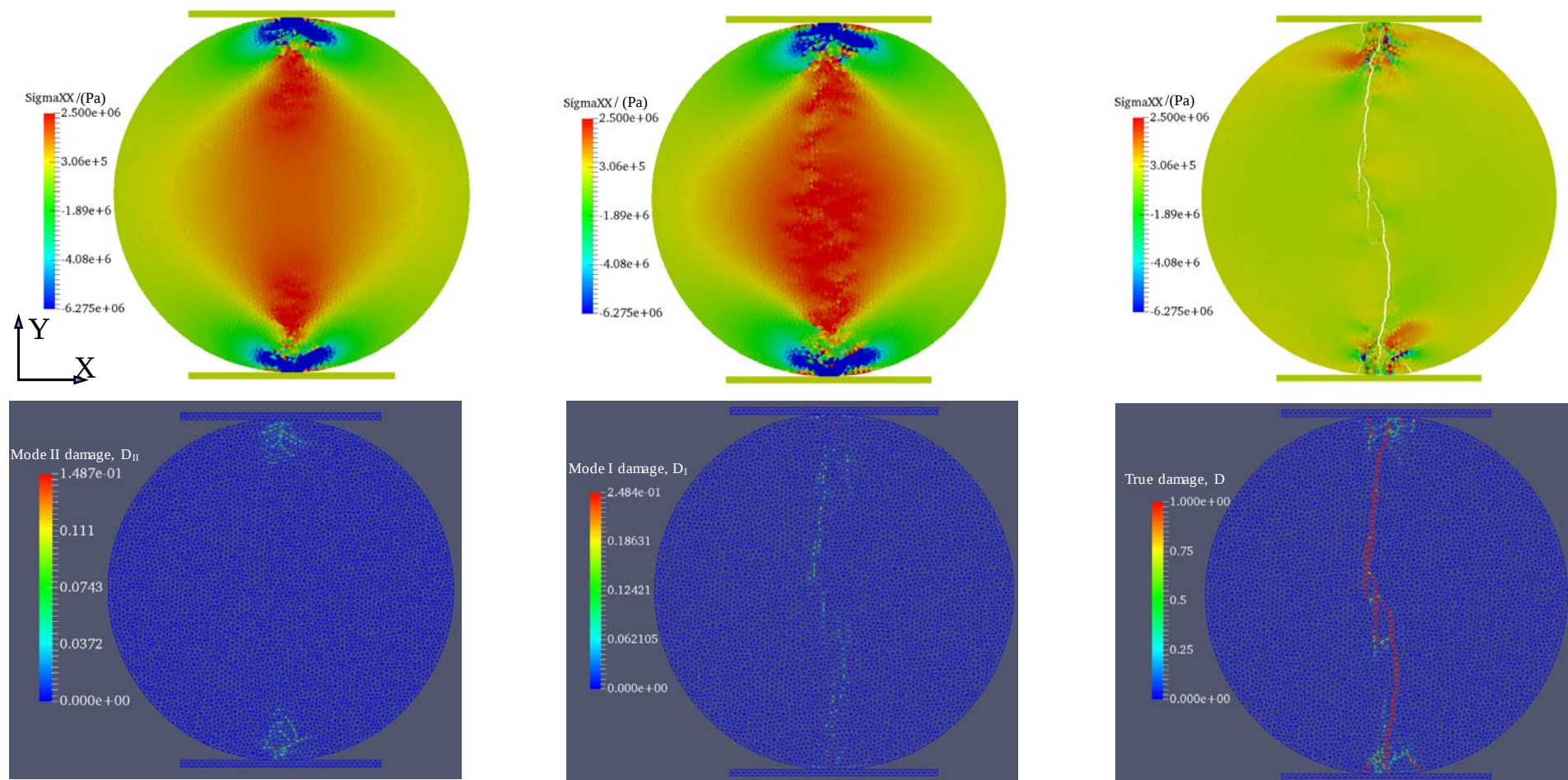


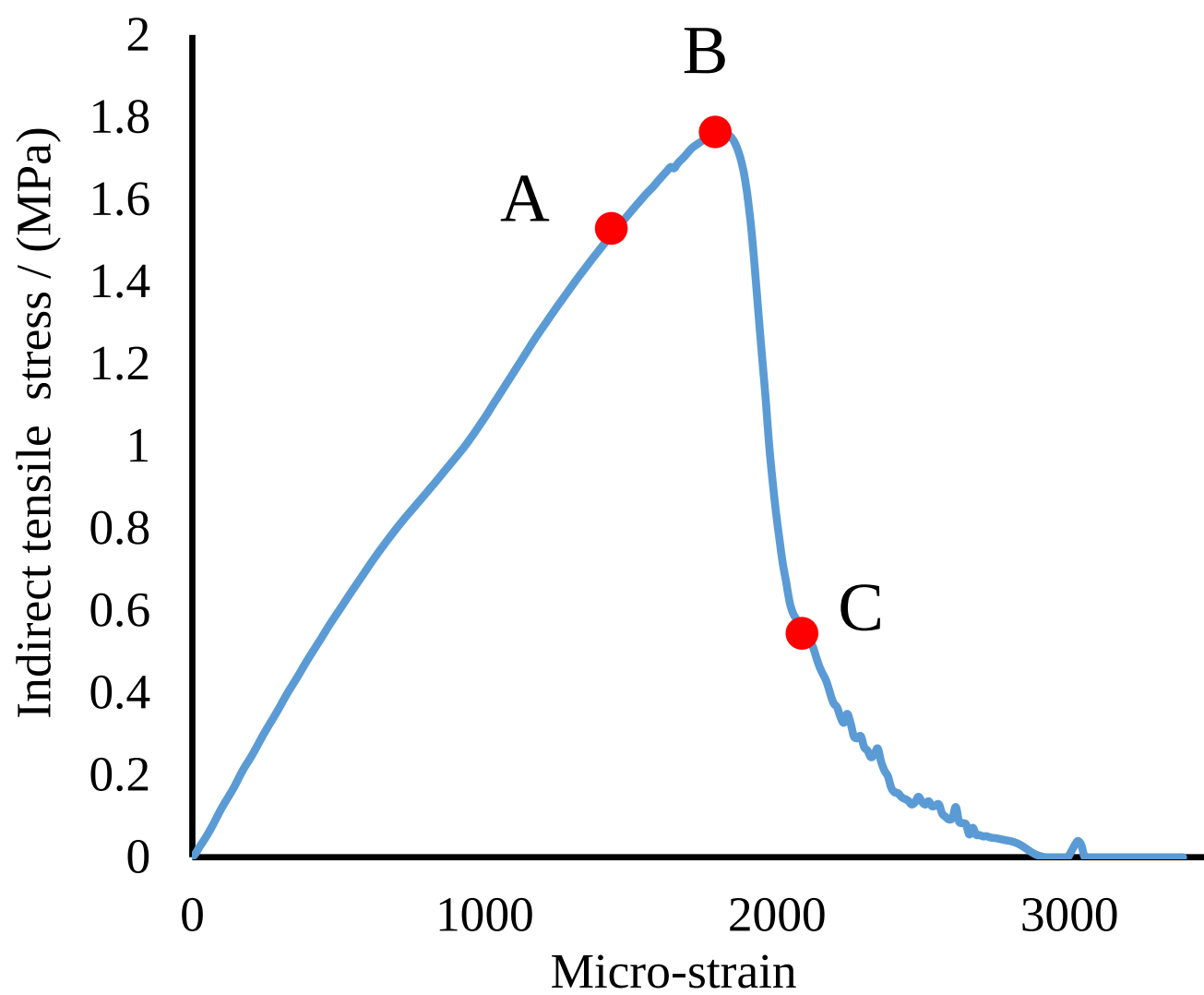
Fig. 13.



(a)

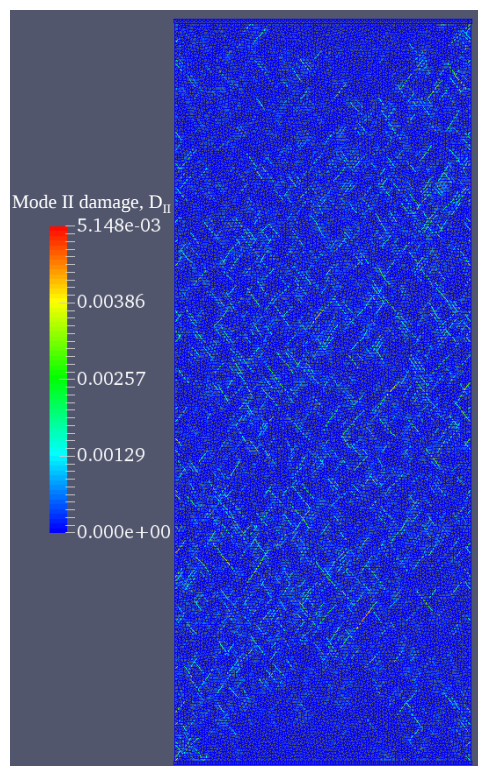
(b)

(c)

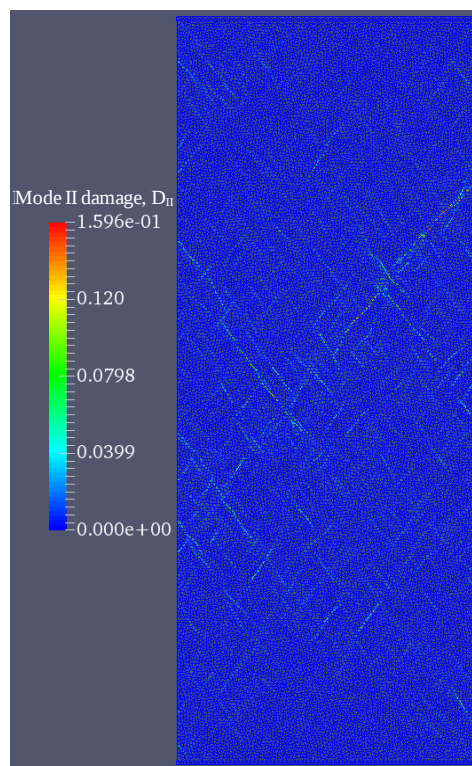


(d)

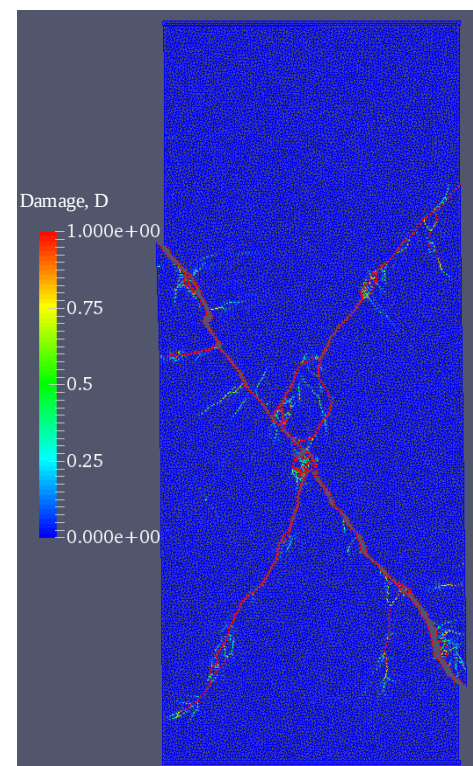
Fig. 14.



(a)

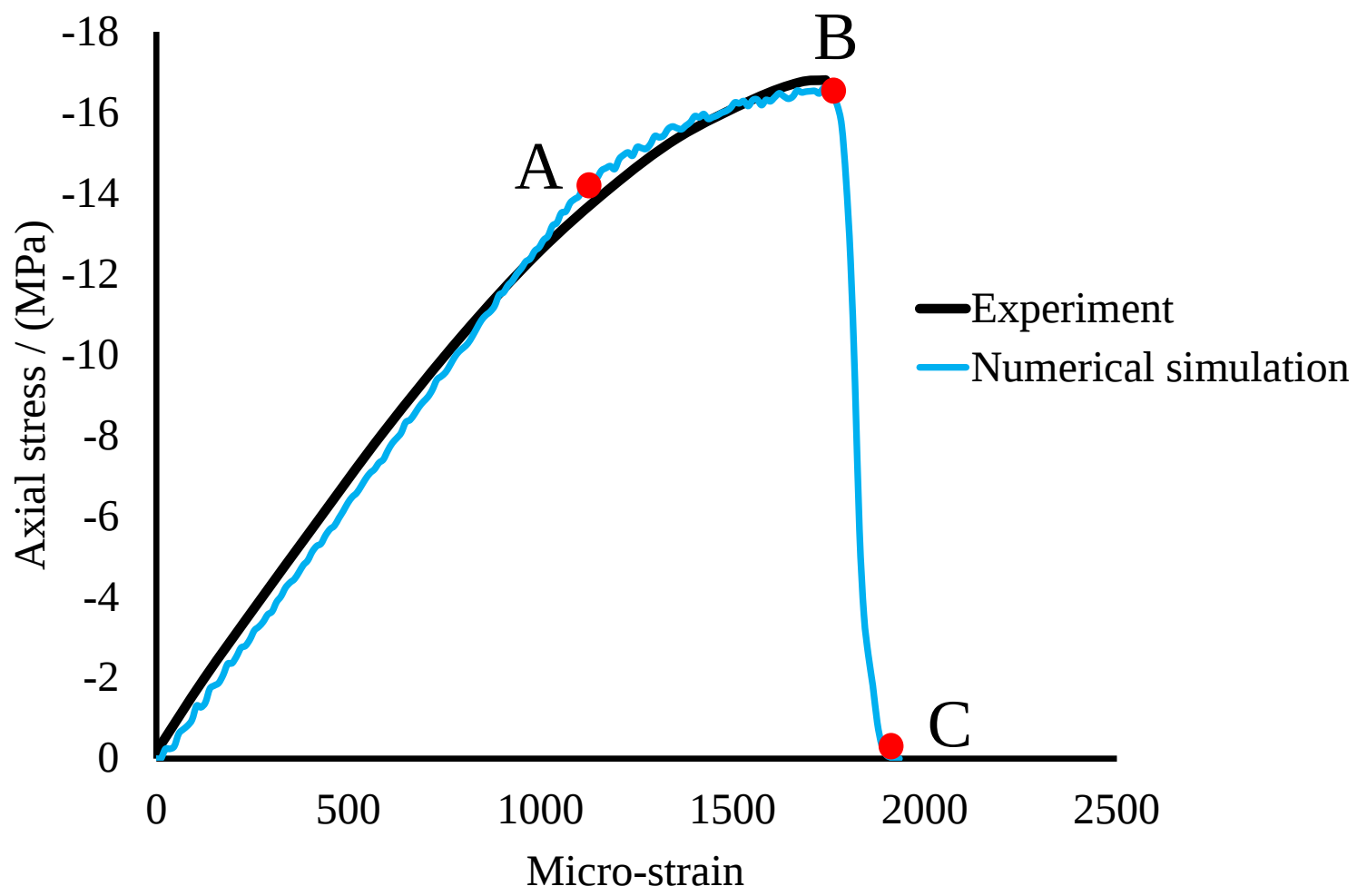


(b)

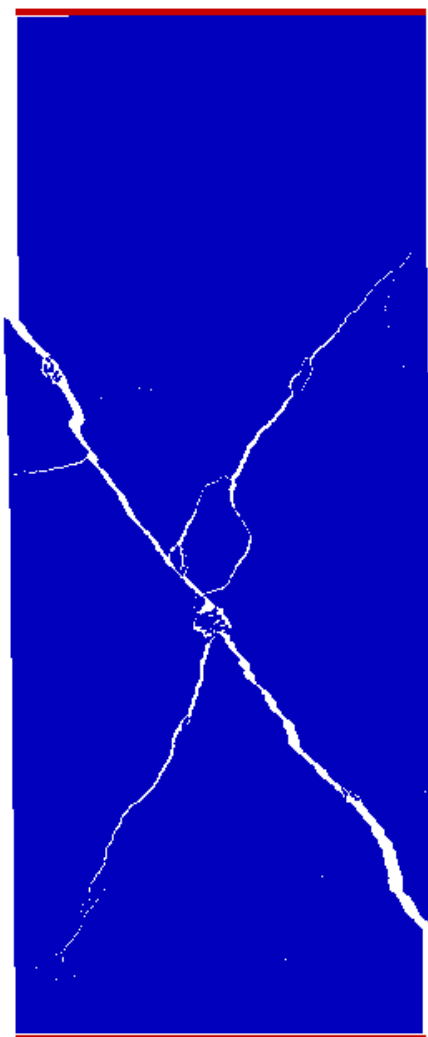


(c)

Fig. 15.



(a)



(b)

Fig. 16.

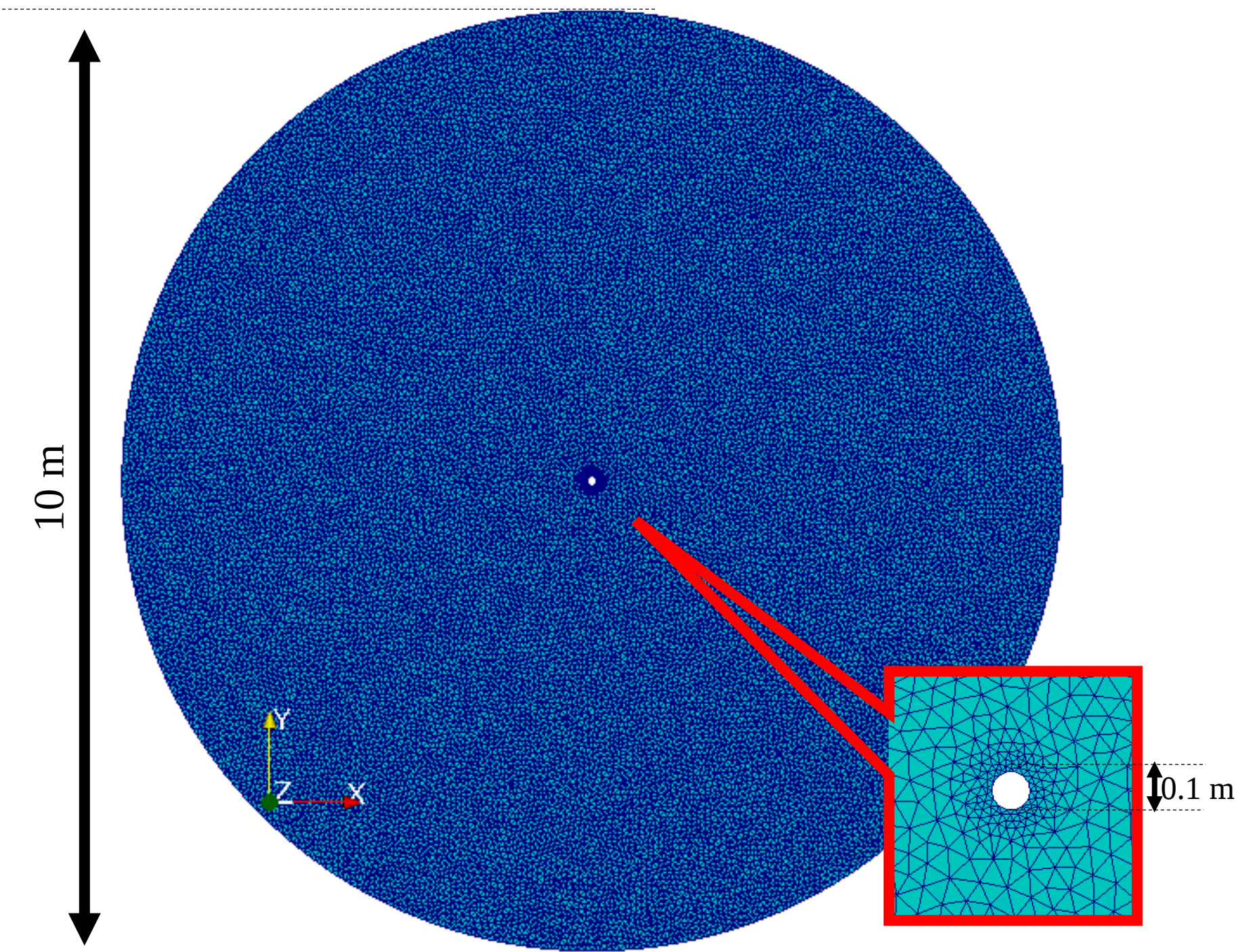
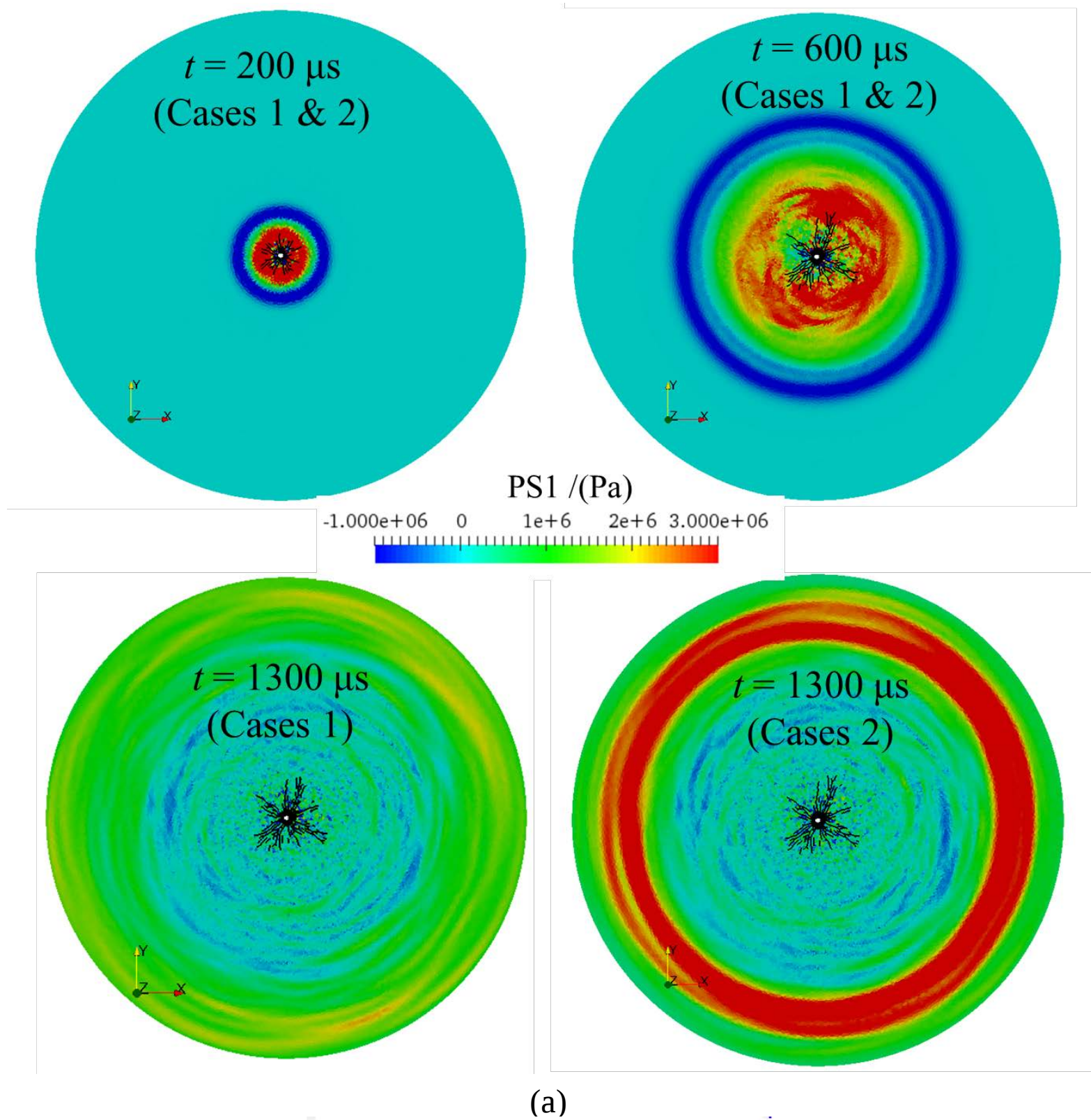
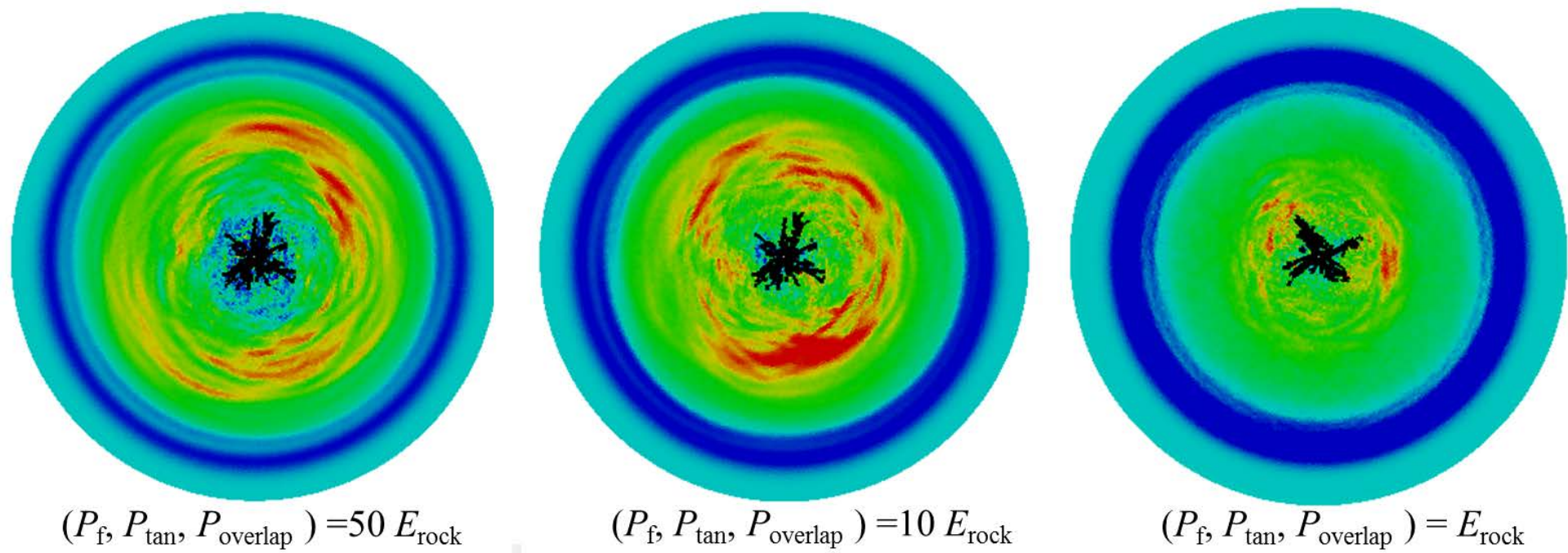


Fig. 17.



(a)



(b)

Fig. 18.

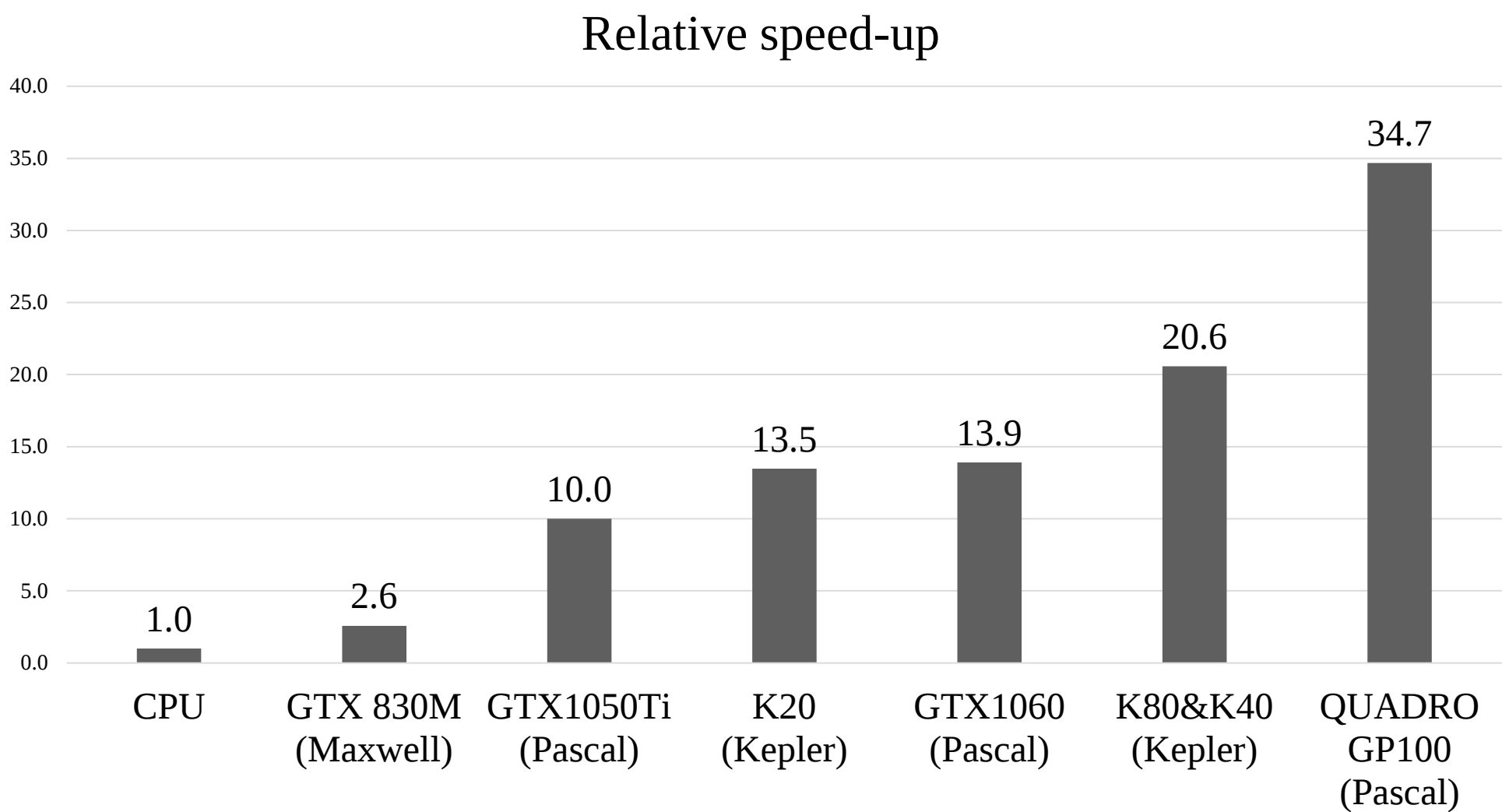


Fig. 19.

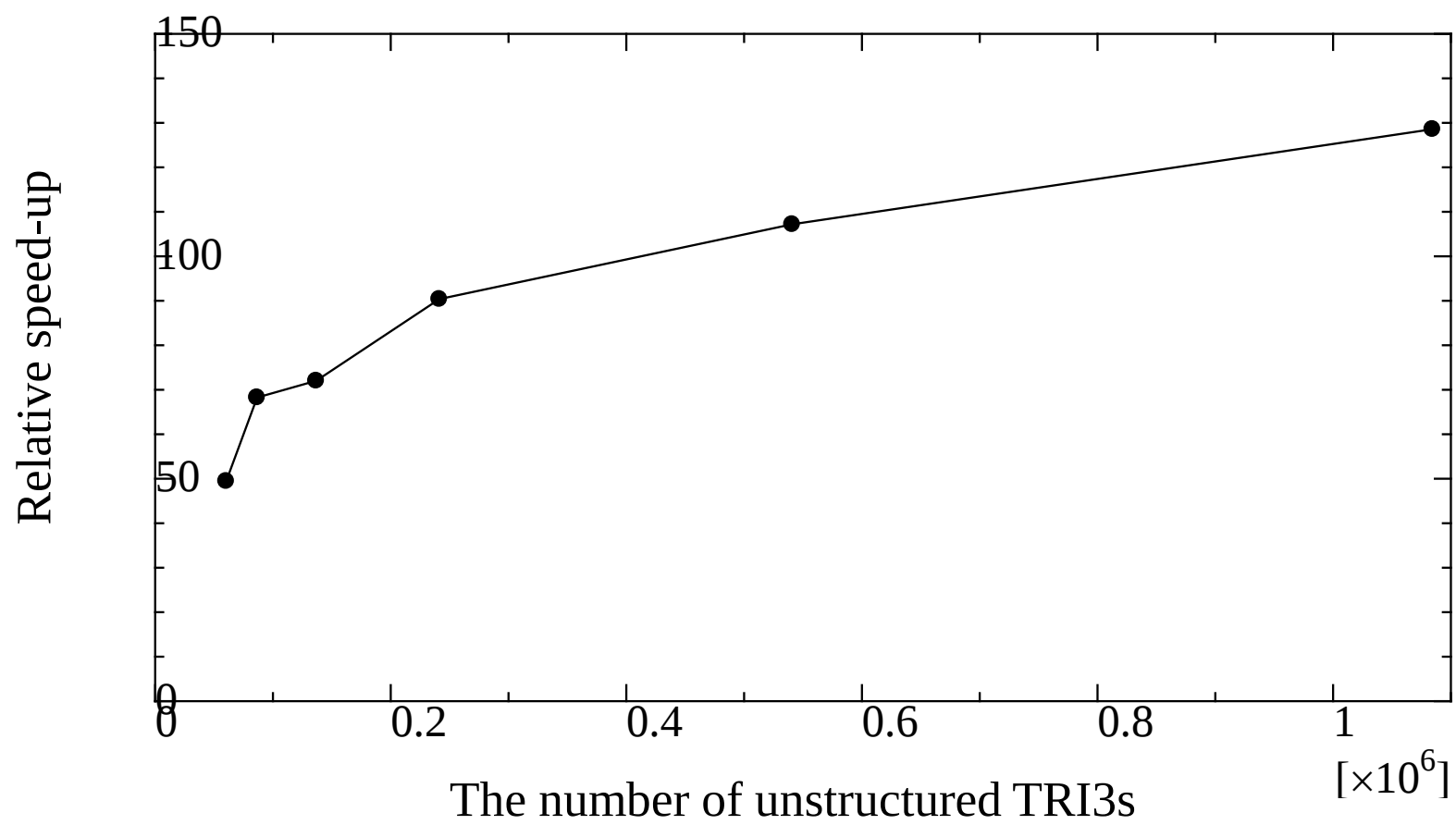


Fig. 20.

```

/* An example GPGPU kernel in 2-D Y-HFDEM IDE using CUDA C/C++.
for 2D mechanical solver for nodes with x & y degrees of freedom.
Assignment of boundary condition is not shown in this example.
Access speed to GPGPU device local memory shown by register is faster,
while that to GPGPU device global memory shown by variable names with
_DEV is much slower */

__global__ void Ysd2MEC_GPU( double *d1nmct_DEV, // nodal mass
                           double *d1nvcx_DEV, // nodal velocity for x
                           double *d1nvcy_DEV, // nodal velocity for y
                           double *d1nfcx_DEV, // current nodal force for x
                           double *d1nfcy_DEV, // current nodal force for y
                           double *d1nccx_DEV, // nodal coordinate for x
                           double *d1nccy_DEV // nodal coordinate for y )
{
    register int    inopo;
    register double dt, nodal_mass;
    register double aX,aY,fx,fy,vXnew,vYnew,vXpre,vYpre;;
    // threadIdx.x:thread id, blockIdx.x:block id, blockDim.x :number of threads per block
    // Thus, inopo in the next line corresponds to node ID.
    inopo = threadIdx.x + blockIdx.x * blockDim.x;
    // current_num_TRI3_contact_candidate_CST: The total number of nodes in the system
    if(inopo < current_num_nodes_CST)
    {
        // Get nodal mass from GPGPU device global memory
        nodal_mass = d1nmct_DEV[inopo];
        // As long as nodal_mass is positive, update the nodal positions and velocities
        if(nodal_mass>EPSILON) // EPSILON: Very small value
        {
            // x-direction
            // Get x nodal velocity in previous step from GPGPU device global memory
            vXpre = d1nvcx_DEV[inopo];
            // Get x nodal force in the previous step from GPGPU device global memory
            fx     = d1nfcx_DEV[inopo];
            // Get current time step increment from GPGPU device constant memory
            dt = dcstec_CST;
            // Compute nodal acceleration along x direction based on Eq.(15)
            aX = fx /nodal_mass;
            // Compute new nodal velocity along x direction
            vXnew = vXpre + aX * dt;
            // Update x coordinate of this node and store it into GPGPU device global memory
            d1nccx_DEV[inopo] += vXnew * dt;
            // Store new nodal velocity along x direction to GPGPU device global memory
            d1nvcx_DEV[inopo] = vXnew;
            // y-direction
            /* Similar to x-direction. → Omitted. */
        }
    }
}

```

Fig. A.1