A three-dimensional computational framework for impact fracture analysis of automotive laminated glass

Shunhua Chen\textsuperscript{a}, Mengyan Zang\textsuperscript{a,}\textsuperscript{*}, Wei Xu\textsuperscript{a,\textsuperscript{b}}

\textsuperscript{a} School of Mechanical & Automotive Engineering, South China University of Technology, Guangzhou 510641, People’s Republic of China
\textsuperscript{b} School of Mechanical Engineering, Kanazawa University, Kanazawa, 920-1192, Japan

Received 18 November 2014; received in revised form 19 May 2015; accepted 8 June 2015
Available online 14 June 2015

Abstract

This work is concerned with a three-dimensional modelling of impact fracture behaviours of laminated glass. A cohesive model that is able to account for path-dependent behaviours is extended to three dimensions to predict the onset and evolution of fracture. A compact and efficient adjacency-based topological data structure termed TopS is utilized to store and manipulate mesh information. In view of contact characteristics, a global contact search algorithm is developed, which includes a regular search and an adaptive search. The regular search is performed by recourse to a linear contact algorithm named LC-Grid. The adaptive search algorithm is proposed to efficiently update new contacts during fracture. Additionally, a local contact search algorithm is developed, which facilitates the judgements of node–face and edge–edge contacts with a unified inside–outside approach. Afterwards, the cohesive model is coupled with the frictional contact algorithm to account for the unilateral and frictional sliding effects of cohesive elements under mixed mode loading. Finally, the impact fracture behaviours of a laminated glass plate are simulated. The simulation results compare well with the experimental outcomes, which validates the capability of the proposed method in the impact fracture analysis of laminated glass.

© 2015 Elsevier B.V. All rights reserved.

Keywords: Laminated glass; Cohesive model; Contact algorithm; Impact fracture; Coupling model

1. Introduction

Recent years have witnessed mounting concern over the pedestrian protection all over the world. In a pedestrian–car accident, the head injury of a pedestrian caused during impact with automotive laminated glass is one of the leading causes of pedestrian’s death [1]. Consequently, there is a pressing need to investigate the impact fracture mechanics of laminated glass such as the characteristics of energy absorption during impact to reduce the head injury of the pedestrian. Moreover, the impact fracture patterns of automotive laminated glass are conducive to traffic reconstruction [2].
Automotive laminated glass is normally composed of two pieces of glass and one PVB film. It is considered to be safety glass in the sense that the broken pieces of glass are still bonded to the transparent plastic interlayer, PVB, in case of glass breakage and that it possesses good energy absorption characteristics due to the deformation of the PVB film. Up to now, both laboratory experiments and numerical simulations have been carried out to investigate the mechanical properties of laminated glass. Xu et al. [3,4] conducted a series of experiments of laminated glass including quasi-static and dynamic tests. Specifically, the impact fracture behaviours such as the propagations of radial and circular cracks of laminated glass under dynamic loading were intensively studied by using a drop-weight device combined with a high-speed photography device [4]. For other dynamic experiments of laminated glass, please refer to [5–7] for more details.

By way of contrast, it still remains challenging to simulate correctly all the observed features of dynamic fracture and fragmentation of laminated glass under impact. Various numerical methods have been adopted for the application of interest here. Among them, the most common one is the element deletion method (sometimes also called element erosion method) [6–12] which is available in some commercial finite element solvers such as LS-DYNA, ABAQUS. Though this method is simple and easy to implement, it suffers from some drawbacks. When a finite element meets the fracture criterion, the mass of the element is removed from the global mass matrix to represent the failure of the element, which may result in inaccurate results especially for large-size elements. Moreover, its reliability to predict the crack path is still questionable [13] and the simulation is highly mesh-dependent as well. Recently, an energy-based element deletion method referred as eigenerosion was proposed to overcome these shortcomings [14,15]. Other researchers adopted the continuum damage mechanics-based finite element method (FEM) as an alternative [16,17]. However, this method that idealizes the net effect of fracture as a degradation of the material is unable to capture the discrete nature of cracks [18]. In view of the inherent superiority of the non-continuum-based discrete element method (DEM) in dealing with multi-fracture and fragmentation phenomena, Zang et al. [19] simulated the impact fracture behaviours of laminated glass by using the three-dimensional DEM. On the basis of this work, Lei et al. [20] proposed a three-dimensional combined discrete element (DE)/finite element (FE) method by using a penalty function method. In this coupling algorithm, the glass is discretized into a series of spherical DEs, while the PVB film a series of FEs because it is prone to large deformation. Later, Xu et al. [21] developed a four-point combined DE/FE method which yielded a better result than the Lei’s algorithm [20] in the elastic range by constraining the rotation freedoms of DEs about their linked points. Then they simulated the impact fracture processes of a laminated glass beam by employing an extrinsic cohesive fracture model in the DE domain. Considering the drawback of poor computational efficiency of the DEM and the merit of mature ability of the FEM to solve continuum problems, Xu et al. [22] proposed a two-dimensional adaptive combined DE/FE method and simulated the crack propagation sequences of a laminated glass beam. In this algorithm, the solving domain is initially discretized into FEs, then the FEs will be replaced by clusters of DEs when a defined criterion is met during calculations; afterwards, an extrinsic cohesive fracture model is employed to simulate the occurrence and propagation of fracture in the DE domain. In the series of works of the Zang’s research group [19–22] mentioned above, the adaptive combined DE/FE method is certainly most accurate and computationally efficient, however it is still limited to two-dimensional plane stress analysis and its ability to capture the propagation and branching of multi-fracture as well needs to be proved. Additionally, Munjiza et al. [23] developed a new shell element for glass fracture simulations in the context of the combined DE/FE method. Recently, Xu et al. [24] have characterized the propagations of radial and circumferential cracks and performed a parametric study of a laminated glass plate under low-velocity impact by using the extended finite element method (XFEM). In this paper, in order to simulate the complex fracture behaviours of laminated glass under low-velocity impact where cracks may propagate, branch and even link up to form fragments, we resort to the cohesive zone model (CZM) whose pioneer works can be traced to those of Barenblatt [25,26] and Dugdale [27]. Several comparative studies between the XFEM and the CZM were performed in [13,28]. Though the crack paths predicted by the CZM tend to be somewhat mesh-dependent, the CZM appears to be more applicable to multi-fracture problems than the XFEM as the use of level set functions in the XFEM is cumbersome in multi-fracture cases [13].

Apart from the CZM, another two important aspects that should be noted in the impact fracture simulations are the data structure and the contact algorithm. The data structure which consists of topological entities and adjacency relationships for query should balance query efficiency and memory storage. Storing all the data structure information decidedly makes the query very efficient, while it will result in huge storage space for large-scale simulations. We have found that the adjacency-based topological data structure termed TopS [29,30] is complete and very compact. It has also been extended to cohesive zone-based fracture simulations [31]. TopS offers a good compromise in terms
of memory storage versus computational cost, which is of linear complexity in the adaptive insertions of cohesive elements. To address the same issues, Mota et al. [32] came up with a different data structure by using graphs which is as well linear in the cohesive element insertions. The TopS data structure is also used to store and manipulate the data information in the contact algorithm in this paper.

Regarding the global contact search algorithm, we have decided to utilize a cell-based global contact algorithm (LC-Grid) proposed by Chen et al. [33]. Compared with other global contact search algorithms such as the well-known position code algorithm [34], LC-Grid is linear in computational complexity. Moreover, it is as well linear in memory requirement with respect to other algorithms (e.g. [35]) with linear computational complexity which demand considerable memory space. Because new contacts are adaptively generated along with the propagations of cracks, we develop an adaptive contact search algorithm basing on LC-Grid, which exploits the advantages of the linear properties of LC-Grid. The developed global contact algorithm is efficient and particularly suitable for cohesive zone-based fracture and fragmentation simulations. As for the local contact algorithm in fracture simulations, it can go back to the work of Camacho and Ortiz [18] at least, where a node-to-segment algorithm was utilized for two-dimensional problems. Afterwards, Kane et al. [36] proposed a novel non-smooth local contact algorithm for the contact interactions among angular bodies by using the intersection area in two dimensions (or the intersection volume in three dimensions) as the constraint function. However, it is limited to triangular and tetrahedral elements and not straightforward to extend to quadrilateral and hexahedral elements. To overcome this drawback, Haikal et al. [37] developed a non-smooth algorithm basing on oriented volumes. However, the edge–edge contact is not taken into consideration which is crucial to the simulations involving impacts between non-smooth bodies [38]. Later, Johnson et al. [39] came up with a linear programming approach for the collision detections of convex bodies. In this work, we develop a robust local contact search algorithm which includes node–face and edge–edge contacts for fracture and fragmentation simulations. The judgements of these two contact types are achieved by using a unified inside–outside approach [40]. The proposed local search algorithm is simple and easy to implement. Though this algorithm is applied to hexahedral elements in this paper, it is quite general and easy to extend to other element types such as triangular and tetrahedral elements.

The mechanism of the damage process of cohesive elements is important especially for mixed mode loading [41]. Recently, an increasing interest has been devoted to combining cohesive damage with frictional contact [41–48]. Among them, Snozzi et al. [41] adopted a transition function for the frictional force to give a soft transition from the crack onset to pure frictional sliding. Alfano et al. [42] decomposed the cohesive interface into two parts, i.e. the undamaged and damage parts, and assumed that the friction only occurred on the damage part. Benedetti et al. [45] introduced some friction terms which were damage levels related into the traction–separation law. Raous et al. [48] developed a model accounting for the unilateral contact, friction and adhesion. In this paper, we also couple the cohesive model with the frictional contact algorithm, which brings benefits to the numerical treatments of cohesive elements under mixed mode loading.

The paper is organized in the following way. Section 2 presents the formulation of the principle of virtual work, and then extends a cohesive fracture law that accounts for path-dependent behaviours to three dimensions. Section 3 introduces the compact topological data structure, TopS, which is applied to the cohesive fracture simulation and the contact algorithm. Section 4 describes the developed contact algorithm including a global contact algorithm and a local contact algorithm. Section 5 introduces the coupling of the cohesive model and the frictional contact algorithm. We validate the computational efficiency of the adaptive contact search algorithm and the effectiveness of the local contact search algorithm via numerical examples in Sections 6.1 and 6.2; in Section 6.3, we simulate the impact fracture behaviours of a laminated glass plate and compare the simulation results with the experimental outcomes. Finally, a conclusion is drawn in Section 7.

2. Cohesive zone models

2.1. Principle of virtual work

We consider a boundary value problem for a general body with developing cracks due to impact, as shown in Fig. 1. The problem consists of a domain $\Omega$ with a boundary $\Gamma$. General mixed boundary conditions are taken into consideration. Traction $\mathbf{t}$ and displacements $\mathbf{u}$ are specified along the subsets of the body boundary, $\Gamma_t$ and $\Gamma_u$, respectively. The governing equations without considering cracks in the framework of the updated Lagrangian
description can be defined as:

\[
\begin{align*}
\text{div} \sigma + \rho b &= \rho \ddot{u}, \quad \text{in} \ \Omega \ (\text{equilibrium}) \\
\sigma n &= t, \quad \text{on} \ \Gamma_t \ (\text{traction boundary condition}) \\
u &= \overline{u}, \quad \text{on} \ \Gamma_u \ (\text{displacement boundary condition})
\end{align*}
\] (1)

where \( \text{div} \) is the divergence operator; \( \rho \) is the mass density; \( b \) is the body force vector; \( \ddot{u} \) is the second order partial derivative of the displacement vector \( u \) with respect to time \( t \); and \( \sigma \) represents the Cauchy stress tensor and \( n \) is the normal vector to \( \Gamma_t \).

Under these boundary conditions, the weak form of the equilibrium equation (i.e. the principle of virtual work) at time \( t \) is given by:

\[
\int_{\Gamma_t} \overline{t} \delta u^T d\Gamma_t - \int_{\Gamma_c} T \delta \Delta^T d\Gamma_c - \int_{\Omega} \sigma \delta \varepsilon^T d\Omega + \int_{\Gamma_t} \rho (b - \ddot{u}) \delta u^T d\Gamma_t = 0
\] (2)

where the first term represents the external work acting on the boundary \( \Gamma_t \) by tractions \( \overline{t} \); \( \delta u \) is the virtual displacement and \( \delta \varepsilon \) is the virtual strain.

In addition, cohesive tractions \( T \) of the crack tip act on the separating surface \( \Gamma_c \). Considering the existence of the cohesive surface, the expression of the principle of virtual work is rewritten as follows:

\[
\int_{\Gamma_t} \overline{t} \delta u^T d\Gamma_t - \int_{\Gamma_c} T \delta \Delta^T d\Gamma_c - \int_{\Omega} \sigma \delta \varepsilon^T d\Omega + \int_{\Omega} \rho (b - \ddot{u}) \delta u^T d\Omega = 0
\] (3)

where the second term denotes the work done by the cohesive tractions \( T \); \( \delta \Delta \) represents the virtual displacement jump along the separation surface \( \Gamma_c \).

2.2. General aspects of cohesive zone models

A large body of literature has reported the successful applications of the cohesive zone models in various fields such as the delamination of laminates whose fracture paths are known a priori (e.g. [49–51]) and the brittle fracture simulations where cracks are able to propagate along arbitrary paths (e.g. [18,52–55]). The fundamental idea of cohesive zone models is to idealize fracture as a gradual process of separation in the small region ahead of the crack front. The gradual process is then governed by a phenomenological traction–separation law which is characterized by a cohesive strength, i.e. the peak stress of the traction–separation curve, and a toughness, i.e. the area under the traction–separation curve.

According to the differences of CZMs in finite element implementations, CZMs can be classified into two groups: intrinsic (also called initially elastic) models (e.g. [53]) and extrinsic (also called initially rigid) models (e.g. [18]). For intrinsic cohesive models, cohesive elements are inserted at all the inter-element boundaries prior to the beginning of simulations, while they are inserted adaptively during calculations in extrinsic cohesive models. The traction–separation curve of the intrinsic approach is assumed to have an elastic response when compared
Fig. 2. The traction–separation law expressed in terms of the surface traction $t$ and the opening displacement $\delta$ is characterized by the cohesive strength $\sigma_{\text{max}}$ and the energy release rate $G_c$.

with the extrinsic curve, as shown in Fig. 2. Because intrinsic cohesive models suffer from artificial compliance, extrinsic cohesive models are more preferable than intrinsic cohesive models for the impact fracture simulations [54]. CZMs can also be classified as potential-based models [50,56] and non-potential-based models [57,58]. The classical Camacho–Ortiz model [18,59] is extrinsic and potential-based which is simple and widely used. However, this model dissipates constant energy regardless of the opening path of a growing crack, because this model is based on a potential [41,60]. Considering this, Snozzi et al. [41] developed a modified Camacho–Ortiz model by introducing a new independent parameter to define the ratio between the energy release rates of mode I and mode II in two dimensions. In this paper, we extend this model to three dimensions.

2.3. A three-dimensional cohesive model

When the effective traction $T_e$ exceeds the cohesive strength $T_{\text{max}}$, a cohesive element is adaptively inserted into the common surface of two elements:

$$ T_e \geq T_{\text{max}}. $$

(4)

The cohesive theories assume that the constitutive response of a cohesive crack surface is governed by a traction–separation law (TSL) which reads:

$$ T = T(\Delta). $$

(5)

The classical Camacho–Ortiz TSL is potential-based which implies that it is possible to obtain the traction vector $T$ by differentiating the free energy density function $\Psi$ with respect to the opening displacement vector $\Delta$:

$$ T = \frac{\partial \Psi}{\partial \Delta}. $$

(6)

We follow the work of Snozzi et al. [41] which introduces a new independent parameter to define the ratio between the energy release rates of mode I and mode II in two dimensions and extend it to three-dimensional problems. For an isotropic material which is of interest here, the energy release rates in the sliding plane, i.e. $G_{c,11}$ and $G_{c,111}$, are assumed to be equal. Thus the normal fracture energy $\psi_n$ is equal to the energy release rate $G_{c,1}$, and the tangential fracture energy $\psi_t$ is equal to $G_{c,11}$ or $G_{c,111}$. An independent parameter $\kappa$ defines the ratio between $\psi_t$ and $\psi_n$:

$$ \kappa = \frac{\psi_t}{\psi_n}. $$

(7)

With the help of the parameter $\kappa$, the work of separation in the tangential direction is restricted and related to that in the normal direction when the opening does not occur in the pure normal direction. Therefore, the modified TSL is capable of dissipating path-dependent fracture energy. The effective opening displacement $\Delta_{\text{eff}}$ in the Camacho–Ortiz
The Camacho–Ortiz linear irreversible cohesive law is expressed in terms of the cohesive traction \( t \) and the effective displacement \( \delta \). In view of the situations of crack opening, closure and reopening, a scalar opening displacement \( \Delta_{\text{eff}}^{(\text{max})} \) which denotes the maximum displacement in history is introduced.

The model can be redefined as:

\[
\Delta_{\text{eff}} = \sqrt{\frac{\eta^2}{\kappa^2} \Delta_t^2 + \langle \Delta_n \rangle^2}
\]

where \( \Delta_t = \sqrt{\Delta_{t1}^2 + \Delta_{t2}^2} \), \( \Delta_n \) is the normal displacement component, and \( \Delta_{t1} \) and \( \Delta_{t2} \) are the tangential displacement components; \( \eta \) is a parameter used in the Camacho–Ortiz model to represent the mixed mode effect; the operator \( \langle \cdot \rangle \) is defined as:

\[
\langle x \rangle = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}
\]

Note that the impenetrability condition can be enforced by coupling the cohesive zone model with a local contact algorithm when the normal opening displacement becomes negative, which will be addressed in Section 5.

In the linear irreversible cohesive law proposed by Ortiz et al. [18, 59] as shown in Fig. 3, the conditions of loading, unloading and reloading are taken into account. The cohesive traction vector of the modified Camacho–Ortiz TSL can be rewritten as:

\[
T = \frac{\sigma_{\text{max}}}{\Delta_{\text{eff}}^{(\text{max})}} \left( 1 - \frac{\Delta_{\text{eff}}}{\delta_n} \right) \left( \frac{\eta^2}{\kappa^2} \Delta_t + \Delta_n n \right)
\]

where \( \Delta_{\text{eff}}^{(\text{max})} \) represents the maximum displacement in history, which is equal to \( \Delta_{\text{eff}} \) under loading, however, represents the displacement at the turning point from loading to unloading under unloading and reloading; \( n \) is the unit outward normal; \( \Delta_t = \Delta - \Delta_n n \).

Thus the normal and tangent effective cohesive tractions, \( T_n \) and \( T_t \), can be obtained as follows:

\[
T_n = \sigma_{\text{max}} \left( 1 - \frac{\Delta_{\text{eff}}}{\delta_n} \right) \frac{\Delta_n}{\Delta_{\text{eff}}^{(\text{max})}}
\]

\[
T_t = \frac{\eta^2}{\kappa^2} \sigma_{\text{max}} \left( 1 - \frac{\Delta_{\text{eff}}}{\delta_n} \right) \frac{\Delta_t}{\Delta_{\text{eff}}^{(\text{max})}}.
\]

The two tangential cohesive tractions, \( T_{t1} \) and \( T_{t2} \), can be expressed by:

\[
T_{t1} = \frac{\Delta_{t1}}{\Delta_t} T_t
\]

\[
T_{t2} = \frac{\Delta_{t2}}{\Delta_t} T_t.
\]
2.4. Finite element implementation

The elements of interest in this paper are linear hexahedral elements. As shown in Fig. 4, a cohesive element is inserted between two brick elements and shares one face with each element respectively, i.e. the upper face 1–2–3–4 and the lower face 5–6–7–8. The displacement vector $\Delta$ at an arbitrary point on the cohesive surface can be expressed by:

$$\Delta = \sum_{i=1}^{4} \Delta_i = \sum_{i=1}^{4} N_i (x_i - x_{i+4}) $$  

(15)

where $\Delta_i$ is the opening displacement vector at the corner node $i$ of the cohesive surface and $x_i$ is the displacement vector of node $i$; and $N_i$ denotes the shape function at node $i$.

Consequently, the equivalent nodal forces for the eight nodes of the cohesive element due to the cohesive traction are:

$$f_i = - \int_{S_m} T N_i |v_{t1} \times v_{t2}| dS_m, \quad (i = 1, \ldots, 4) $$  

(16a)

$$f_j = \int_{S_m} T N_j |v_{t1} \times v_{t2}| dS_m, \quad (j = 5, \ldots, 8) $$  

(16b)

where the cohesive traction vector $T$ can be obtained through Eq. (10); and $v_{t1}$ and $v_{t2}$ are the two tangential vectors on the middle face $S_m$ respectively.

Several investigations [51,61,62] on the numerical integration schemes applied to cohesive elements have been conducted, which show that the Newton–Cotes integration technique is superior over the traditional Gaussian integration technique and that the full integration scheme is better than the reduced integration scheme. Thus, in this paper, we adopt the Newton–Cotes full integration scheme.

Upon discretization, the principle of virtual work Eq. (3) takes the form:

$$M \ddot{x} + f^{int}(x) = f^{ext}(t) $$  

(17)

where $M$ is the mass matrix; $x$ is the array of nodal coordinates; $f^{int}$ and $f^{ext}$ are the internal and external force arrays respectively.

An explicit central difference time stepping scheme incorporating the term of cohesive force vector [57] is employed in the present work.

3. TopS data structure

Normally, an integrated data structure comprises of topological entities and adjacency relationships for query, which is of vital importance for many applications such as fracture simulations. A good data structure should balance the conflicting requirements of storage and computational cost. Thus, this paper utilizes an adjacency-based topological data structure termed TopS proposed by Celes [29–31] which offers a good compromise in terms of compactness versus computational efficiency.
In order to balance storage and computational cost, TopS only explicitly stores two basic topological entities, i.e. element and node, and two adjacent relationships, i.e. the ordered group of all the elements adjacent to a certain element denoted by $E[E]$ and the ordered group of all the nodes incident to a certain element denoted by $E[N]$. On the contrary, the other entities including facet, edge and vertex are implicitly stored. The concepts of oriented entity-uses and element templates are introduced. The representations of implicit entities are created on-the-fly when required via entity-uses and the defined element template type. Likewise, the implicit adjacent relationships can be derived through oriented entity-uses when required. In addition, because the facet-use of each element is frequently used in the extraction of adjacent relationships, the reverse indices are stored to allow direct access to its facet mate, which significantly improves the query efficiency for three-dimensional elements while maintaining the storage space within reasonable limits. For more details, please refer to [29–31].

3.1. TopS for the fracture simulation

TopS has been extended to cohesive zone-based fracture simulations [31]. Three adjacent relationships that are of special interest in the impact fracture simulations are: the ordered group of two elements adjacent to a given facet denoted by $f[E]$, the cyclically ordered group of elements adjacent to a given edge denoted by $e[E]$ and the set of elements adjacent to a given vertex node denoted by $N[E]$. The topological operations due to the insertions of cohesive elements are addressed in detail in [31].

3.2. TopS for the contact algorithm

Apart from the cohesive fracture approach, the data information in the contact algorithm should be carefully treated as well. In this paper, two contact types, i.e. node–face and edge–edge contacts, are considered. The adjacency relationships that are fluently retrieved in the contact algorithm are $f[N]$ and $e[N]$, where $f[N]$ means the cyclically ordered bounding nodes of a certain face and $e[N]$ means two ordered bounding nodes of a certain edge. In TopS, the relationships of $f[N]$ and $e[N]$ can be easily retrieved by first accessing the associated facet-use and edge-use and then obtaining the bounding nodes from the defined element template, respectively.

4. Contact algorithm

A contact algorithm usually includes three parts: global search, local search and contact enforcement. The global search is to roughly find all the possible contact pairs. The local search is to judge whether the contact pair is in contact or not. In the impact fracture simulations of laminated glass, the contact interactions among cracks bear some characteristics: new contacts are adaptively generated along with the propagations of cracks, which requires an efficient global contact search algorithm; the contacts among cracks and fragments are typically non-smooth and multiple collisions among glass fragments are involved, in which the traditional node–face contact is not sufficient. In this paper, we develop an efficient global contact search algorithm and a robust local contact search algorithm. The global contact search algorithm includes a regular search and an adaptive search. The local contact search algorithm contains node–face and edge–edge contacts. If penetration occurs, a normal contact force is enforced by using the penalty function method and a frictional force is calculated by recourse to the Coulomb friction model.

4.1. Global contact search algorithm

4.1.1. Regular search

The regular search is performed by using a cell-based contact algorithm named LC-Grid which owns the merit of linear complexity in memory storage and computational efficiency [33]. In LC-Grid, only the node–face contact is considered, however, the edge–edge contact is also involved in this paper. In addition, we introduce a simple technique to avoid repetitive checking during the determination of edge–edge contact pairs. For completeness and subsequent reference, the main features of LC-Grid are summarized as follows, and for more details please refer to [33]:

(1) Space decomposition: The contact space is decomposed into a $n_x \cdot n_y \cdot n_z$ array of cubic boxes, where $n_x$, $n_y$ and $n_z$ are the numbers of the cubic boxes in $x$, $y$ and $z$ directions respectively. The magnitudes of $n_x$, $n_y$ and $n_z$ are
given by:

\[ n_x = \text{INT}\{ (x_{max} - x_{min})/L_c } \} + 1 \quad (18a) \]

\[ n_y = \text{INT}\{ (y_{max} - y_{min})/L_c } \} + 1 \quad (18b) \]

\[ n_z = \text{INT}\{ (z_{max} - z_{min})/L_c } \} + 1 \quad (18c) \]

where \( \text{INT}\{ \cdot \} \) is an operator which takes the integer part of a number; \( x_{min}, y_{min} \) and \( z_{min} \) are the minimum coordinates of the contact space and \( x_{max}, y_{max} \) and \( z_{max} \) are the maximum coordinates of the contact space; \( L_c \) is determined by a scale factor \( \omega_1 \) and a characteristic length \( L: L_c = \omega_1 \cdot L. \)

The characteristic length \( L \) is the average of the areas of segments:

\[ L = \sqrt{\frac{1}{M} \sum_{i=1}^{M} A_i} \quad (19) \]

where \( M \) is the total number of segments; \( A_i \) is the area of segment \( i \). The definition of the characteristic length is appropriate in the case that there is a large variation in the sizes of segments.

(2) **Buffer zone**: Because the regular search is computationally intensive, the concept of buffer zone is introduced to avoid performing it every time step. The definition of buffer zone is in Appendix A.

The performance frequency of the regular search is affected by the size of buffer zone \( \chi \) which is defined in Eq. (A.3). A regular search (at \( m \)th step) is performed once the cumulative maximum displacement among all the contact nodes starting from the last regular search (\( n \)th step) exceeds the size of buffer zone:

\[ \sum_{i=n}^{m} (2 \cdot v_{i_{max}} \cdot \Delta t_{i}) \geq \chi \quad (20) \]

where \( v_{i_{max}} \) is the maximum velocity among all the contact nodes at \( i \)th step with the time step of \( \Delta t_{i} \).

(3) **Positioning of contact entities**: All the contact entities, i.e. contact nodes, contact edges, and contact segments, are positioned to corresponding cells. The mapping procedures for these three contact entities are the same. In LC-Grid, a technique named linked list is used to efficiently store and retrieve the mapping information of contact entities. For each contact entity, this technique uses three one-dimensional arrays named Head lists with respective sizes of \( n_x, n_y \) and \( n_z \) and a one-dimensional array named Next list with a size of the number of the contact entity.

(4) **Determination of contact pairs**: The contact nodes and segments in the same cell are determined as potential node–face contact pairs; and the contact edges in the same cell are potential edge–edge contact pairs. For each node–face contact, a rough check is performed to reduce the storing memory of contact pair lists. Each kind of contact pair is stored by using two one-dimensional arrays of which one stores the contact counterpart and the other one records the address of the next counterpart.

Note that, for two contact edges in the same cell, they may already have been determined as a potential edge-edge contact pair in the previous cell. Thus, a simple technique named mask similar to the one addressed in [63] is introduced to avoid repetitive checking. Please see Appendix B for more details.

(5) **Migration**: As each segment or edge is assigned to a unique cell number in step (3), the segment or edge will be migrated into the next cell and checked with the contact nodes or edges in the next cell if the upper coordinates of its buffer zone exceed the lower boundaries of the next cell.

**4.1.2. Adaptive search**

During the impact fracture processes of laminated glass, new contacts are adaptively generated. Once new contacts are generated, a global search is supposed to be performed to roughly find all the new possible contact pairs. Nevertheless, it is time consuming to perform the regular global search (LC-Grid) for the updates of new contacts, because their number is usually not comparative to that of the whole contacts. Consequently, a simple and efficient adaptive search algorithm is developed to deal this problem. As shown in box 1, the global search algorithm in this paper can be outlined as:

With the help of buffer zone, the regular search does not need to be performed every time step. In other words, the contact pairs between two consecutive regular search remain unchanged. Accordingly, the adaptive search algorithm
The implementations of the global search algorithm

```plaintext
IF (Eq. (20) == .TRUE.) THEN
   CALL Regular-search()
ELSE IF (new contacts are generated) THEN
   CALL Adaptive-search()
END IF
```

updates the contact pairs for the new generated contact entities instead of the whole contact entities. Corresponded to the new contact entities, the contact entities that already exist in the current search are called origin entities. Besides the contact pairs between new entities, the contact pairs between new entities and origin entities also need to be updated. Thus, with respect to the node–face contact, three kinds of contact pairs, which include contact pairs between new nodes and new segments, between new nodes and origin segments, and between origin nodes and new segments, need to be updated. Likewise, two kinds of contact pairs, which involve contact pairs between new edges and between origin edges and new edges, need to be updated for the edge–edge contact. In the adaptive search, the contact pairs between new contact entities are updated by using LC-Grid to make full use of its linear properties, and the other contact pairs are updated by efficiently retrieving the origin contact entities during the updating of new contact entities.

The updating procedures are addressed as follows:

1. **Space decomposition and buffer zone**: The decomposed cells in the last regular search are used in the adaptive search instead of decomposing the contact space again. The size of buffer zone does not change either. Each new contact segment or new contact edge is assigned with corresponding buffer zone.

2. **Positioning of new contact entities**: The mapping procedures of new contact entities including new contact nodes, new contact segments and new contact edges are the same with the positioning steps addressed in step (3) of the regular search. The cells where new contact entities are generated are marked as new cells.

3. **Origin contact entities for new cells**: This step is to obtain origin contact entities which include origin nodes, origin segments and origin edges for new cells. The detailed procedure of this step can be found in Appendix C.

4. **Determination of contact pairs**: This step is to determine the contact pairs for node–face and edge–edge contacts in all the new cells. For each edge–edge contact, the judgement test addressed in step (4) of the regular search should be performed before recording it to the edge–edge linked list.

5. **Migration**: The procedures of migrations for new contact segments and edges are the same with those addressed in the regular search.

The implementations of the adaptive search algorithm based on LC-Grid are outlined in box 2 (Appendix D).

**Remark 1.** As addressed in [33], the computational cost of LC-Grid can be expressed by:

\[
T \propto (\alpha_1 N + \alpha_2 M + \alpha_3 E)
\]

(21)

where \(\alpha_1, \alpha_2, \alpha_3\) are constants independent of the number of contact entities.

In the adaptive search, the contact pairs between new contact entities are determined by using LC-Grid; some computational cost is needed in the processes of obtaining origin nodes from origin segments and new segments, which is roughly equal to the cost for positioning the corresponding origin nodes in LC-Grid; additional computational cost is required to obtain origin edges, which is coarsely equal to the cost to position and migrate the corresponding origin edges in LC-Grid as the sizes of the cell and the contact edges are not much different. Consequently, the computational cost in the adaptive search can be estimated by:

\[
T_a \propto (\alpha_1 N_{new} + \alpha_2 M_{new} + \alpha_3 E_{new} + \alpha_4 N_{origin} + \alpha_5 E_{origin})
\]

(22)

where \(\alpha_1, \ldots, \alpha_5\) are constants. It is evident that the adaptive search is efficient than LC-Grid for the updates of new contacts, especially when the number of new contact entities is not comparative to that of the whole contact entities. Note that, the computational cost for the operations of rough checks and the loops over all the cells is ignored in LC-Grid, however, this cost can usually not be neglected and may even be the most time-consuming part in the adaptive search for fracture simulations. Also, some computational cost to set the initial values for the arrays in the processes of obtaining origin nodes and origin edges cannot be neglected.
Fig. 5. The inside–outside judgement of a given node $p$ to a given surface $a-b-c-d$. In this figure, $n_a$ is the vertex normal of vertex $a$ which is the cross product of the vectors $n_{ab}$ and $n_{ad}$; then the surface normal $n_s$ can be calculated by averaging the four vertex normals; the vector $n_{abp}$ which is the cross product of $n_{ab}$ and $n_{ap}$ and the surface normal $n_s$ are used in the inside–outside judgement of node $p$ and edge $ab$; the judgements of node $p$ with the other three edges are carried out in the same way. Node $p$ is defined to be inside face $a-b-c-d$ provided that node $p$ is inside all the edges of the face. $\Phi_i$ is the area coordinate which is used to determine the projection point $o$ of node $p$ if node $p$ is inside the surface.

Remark 2. With respect to the memory storage, the storage in LC-Grid can be estimated by [33]:

$$ R = 3(n_x + n_y + n_z) + (N + E + M) + \beta_1 N + \beta_2 E $$

(23)

where the amount of $3(n_x + n_y + n_z)$ is used to store the $Head$ lists for three kinds of contact entities; the magnitude of $(N + E + M)$ is used for the $Next$ lists; the amount of $\beta_1 N + \beta_2 E$ is used to store the contact pair lists; and $\beta_1$ and $\beta_2$ are approximately to constants.

In the adaptive search, an additional amount of $N_{new}$ is utilized to store the mother node of each new contact node; another amount of $(2N_{origin}^t + E_{origin}^t)$ is used in the processes of obtaining origin nodes and edges. Therefore, the memory storage required in the adaptive search is given by:

$$ R_a = 3(n_x + n_y + n_z) + (N_{new} + E_{new} + M_{new}) + (\beta_1 + 1)N + \beta_2 E + N_{origin}^t + E_{origin}^t. $$

(24)

During the updates of new contacts in fracture simulations where, in most cases, the number of new contact entities is not comparable to that of the whole contact entities, the memory requirement in the adaptive search is roughly equivalent to that in LC-Grid.

4.2. Local contact search algorithm

Although only the node–face contact is sufficient for many applications, the edge–edge contact is crucial in fracture and fragmentation simulations [38]. We have found that the inside–outside approach for the node–face contact [40] is robust and efficient, and that this algorithm can be extended to check the contact status of the edge–edge contact as well. The local search algorithm addressed here is robust and efficient for the application of fracture simulations, while being easy to implement.

4.2.1. Node–face contact

Given a node $p$ and a segment $a-b-c-d$ as shown in Fig. 5 which are determined as a potential contact pair in the global search mentioned above, node $p$ is defined to be inside this segment provided that node $p$ is inside all the four edges of the segment.

Taking node $p$ and edge $ab$ as an example, the inside–outside approach is illustrated in detail in Appendix E.

4.2.2. Edge–edge contact

We have found that the inside–outside approach for the node–face contact can be readily extended to the judgement of the edge–edge contact. As shown in Fig. 6, edge $ab$ of element $E_A$ and edge $cd$ of element $E_B$ are a potential edge–edge contact pair determined in the global search. Note that each contact edge in an edge–edge contact pair should just belong to one element, that is to say, the two contact segments that are adjacent to a contact edge should belong to a same element.

The judgement procedures for edges $ab$ and $cd$ are illustrated in detail in Appendix F.
Fig. 6. The edge–edge contact judgement of edge $ab$ of element $E_A$ and edge $cd$ of element $E_B$. In this figure, an oriented virtual plane $c\cdot d\cdot e\cdot f$ is generated, where edges $de$ and $cf$ are parallel to edge $ab$ and edge $ef$ is parallel to edge $cd$; the length of edge $ab$ is equal to the lengths of edges $de$ and $cf$; the length of edge $cd$ is equal to the length of edge $ef$; the normal of the oriented plane $n_f$ is the cross product of the edge vectors $cd$ and $ab$; edge $ab$ is defined to be in contact with edge $cd$ provided that the vector dot product of edge normal $n_{ab}$ and $n_{cd}$ is less than zero and node $b$ of edge $ab$ is inside the oriented virtual plane $c\cdot d\cdot e\cdot f$; point $b'$ is the projection of node $b$ along surface normal $n_s$ onto the oriented virtual plane, subsequently, we can obtain the projection edge $a'b'$ of edge $ab$; and point $o$ is the contact point.

4.3. Contact enforcement

Once penetration occurs, a contact force is enforced to resist the penetration. The contact force can be decomposed into the normal and tangential components, the normal contact force $F_n$ and the frictional force $F_t$, respectively.

The normal contact force $F_n$ at the contact point is calculated by using the penalty function method [64]:

$$ F_n = k \cdot g \cdot n_s $$

(25)

where $g$ is the penetration depth; $n_s$ is the surface normal and the stiffness coefficient $k$ is given by:

$$ k = \varsigma \frac{K A^2}{V} $$

(26)

here, $\varsigma$ is the penalty factor; $K$ is the bulk modulus; $A$ is the segment area and $V$ is the element volume.

The Coulomb friction model is adopted to oppose the tangential relative motion of a contact pair at the contact point. The relative velocity $v_r$ at the contact point is given by:

$$ v_r = v_s - v_m $$

(27)

where, for a node–face contact, $v_s$ is the velocity of the slave node and $v_m$ is the velocity at the contact point which is the velocity interpolation of the four nodes in the master segment; however, for an edge–edge contact, $v_s$ and $v_m$ represent the velocities of the slave and master edges at the contact point respectively, which are the velocity interpolations of the two nodes of the corresponding edges.

Then the tangential unit $n_t$ can be obtained through the tangential component of $v_r$:

$$ v_t = v_r - (v_r \cdot n_s)n_s $$

(28)

$$ n_t = \frac{v_t}{|v_t|} $$

(29)

Thus, the friction force is given by:

$$ F_t = -\mu F_n n_t $$

(30)

where $F_n$ is the magnitude of the normal force; $\mu$ is the Coulomb friction coefficient.
5. Coupling of the cohesive model with frictional contact

During the damage processes of cohesive elements under mixed mode loading, a reliable cohesive model should not only account for the unilateral effects due to the closure of cracks but also consider the frictional sliding effects. Though the impenetrability condition in a cohesive element can be enforced simply by adding a penalty term when the normal opening displacement becomes negative, others prefer the contact algorithm as an alternative (e.g. [65–68]). The Newton–Cotes full integration scheme adopted for the cohesive force calculation in this paper, whose integration points are located at the element nodes, can also bring benefits to the continuous transition between cohesive resistance and unilateral contact enforcement. Depending on the sign of the normal opening displacement, an on–off switch between cohesive resistance and unilateral contact enforcement is automatically proceeded. That is to say, the cohesive force acts at the element node when the normal opening displacement is positive, while the contact enforcement functions when the normal opening displacement turns negative.

For the frictional sliding, a simple way is to neglect friction until the cohesive element is failure, while it may give rise to some physically inconsistent behaviours and numerical problems [45]. A simple approach proposed by Snozzi et al. [41] is used here to achieve a soft transition from cohesive fracture to pure frictional contact. The friction force is calculated once the cohesive element is under a compressive sliding state. Aiming to smooth the transition from cohesive fracture to pure frictional sliding, this approach introduces a coefficient function which depends on the amount of damage of the cohesive element to regulate the magnitude of the friction force. Thus, the magnitude of the friction force $F_{n}^{\text{new}}$ acting on contact entities can be redefined by:

$$F_{n}^{\text{new}} = F_{n} \cdot f(\Delta_{\text{eff}})$$

where $f(\Delta_{\text{eff}})$ is an exponential coefficient function which is expressed by using the effective opening displacement $\Delta_{\text{eff}}$ and the critical opening displacement $\delta_n$ in TSL: $f(\Delta_{\text{eff}}) = 1 - (1 - \frac{\Delta_{\text{eff}}}{\delta_n})^q$, and $q$ is an exponent.

**Remark 3.** In this cohesive–frictional model, the global contact search algorithm introduced in Section 4.1 is not needed, because the counterparts of these segments are determined by cohesive elements. However, along with the deformation and movement of finite elements, the segments that belong to the newly broken cohesive elements are likely to contact the other segments which may be defined at the beginning of calculations or belong to other broken cohesive elements. Consequently, in this paper, the global contact search procedure is performed to find new possible contact pairs when cohesive elements are broken.

6. Numerical examples

The numerical simulations in this section are performed on a personal computer with a processor of 3.09 GHz Intel Core (TM) i3-2100.

6.1. Computational efficiency of the adaptive search algorithm

Simple numerical simulations are performed here to verify the computational efficiency of the proposed adaptive search algorithm. A plate with a size of $120 \times 90 \times 2$ mm$^3$ is used for these tests. As shown in Fig. 7, the plate is divided into 30 subregions. Each subregion is discretized into 100 brick elements of the same size (10 elements in the $x$ or $y$ direction and 1 layer in the $z$ direction), as shown in Fig. 8. Three numerical tests are carried out, where the numbers of the new segments generated in every loop are different: in case 1, the number is 2; in case 2, it is 10; and in case 3, it is 20. The new segments are generated by inserting cohesive elements into inter-element boundaries. However, the cohesive strength and the energy release rate of each cohesive element are set to zero, which means that if there are nodes duplicated in a cohesive element, this cohesive element becomes contact segments once inserted. Moreover, the cohesive elements are randomly inserted to make the numerical cases more general. The total number of cohesive elements that are to be inserted is 5000. We focus on the computational cost of the global search algorithm in these tests, therefore no external loads are applied. The scale factors $\omega_1$ and $\omega_2$ are set to be 1.0 and 0.4, respectively. The number of the initial segments is 6220. We only consider node–face contact pairs here for simplicity.

The elapsed time for the updates of new segments by using LC-Grid and the adaptive search algorithm is listed in Table 1. The numerical results show that the adaptive search algorithm is much more efficient than LC-Grid for the
updates of new contacts. The main reason is that LC-Grid is performed by looping over all the contact segments for the updates of new contacts, which is computationally expensive especially when the number of loops is large. From Table 1 we can also see that the elapsed time in these three cases for 300 loops and 500 loops by using the adaptive search algorithm is almost the same. This is because the most time-consuming part in this algorithm is the cost for the loops over all the cells when the number of new segments is small. Thus, the proposed adaptive search algorithm is particularly suitable for fracture and fragmentation simulations.

The scale factor $\omega_1$ has a great influence on the computational efficiency of the adaptive search algorithm, because the number of the cells is determined by this factor. The results of the total computational time versus the scale factor $\omega_1$ in the three cases are shown in Fig. 9. Since the ideal range of $\omega_1$ in LC-Grid is 0.8–1.2 [33], the value of $\omega_1$ in the global search algorithm is also selected from this range.

The other scale factor $\omega_2$ has little influence on the computational cost of the adaptive global search. However, it should also be selected carefully in this paper, since it influences the computational time of the regular search and the local search. The ideal range of $\omega_2$ is 0.1–0.4 [33].
6.2. Effectiveness of the local contact search algorithm

In order to validate the effectiveness of the local contact search algorithm, two impact tests of cubic blocks are carried out, in which both node–face and edge–edge contacts are required. The material used in these two cases is assumed to be linear elastic and the parameters are: Young’s modulus $E = 74.0$ GPa, Poisson’s ratio $\nu = 0.2$ and the density $\rho = 2456$ kg/m$^3$. As shown in Fig. 10, in case 1, a single block impacts another one block which is constrained at one side with an initial velocity of $v_x = v_z = -3$ m/s; in case 2, a single block impacts a heap of blocks with an initial velocity of $v_x = v_z = -8$ m/s. The friction is not considered in these two cases.

During the impacts in case 1, an edge–edge contact occurs between these two blocks at the beginning (Fig. 11(a)); afterwards, the impactor rotates (Fig. 11(b)); and finally a node–face contact occurs (Fig. 11(c)) and then the impactor is pushed away (Fig. 11(d)). For the impacts in case 2, the impactor strikes one block in the heap through an edge–edge contact at the beginning (Fig. 12(a)); then the impactor rotates and multiple contacts occur in the heap (Fig. 12(b) and Fig. 12(c)); and finally the impactor hits against the block again via a node–face contact at 0.13 ms; the states of these blocks at 0.18 ms are shown in Fig. 12(b).

6.3. Impact fracture simulation of a laminated glass plate

6.3.1. Experimental set-up

A drop-weight experimental set-up is built to investigate the impact fracture behaviours of a laminated glass plate, as shown in Figs. 13 and 14. The set-up operates in the following way: a cylindrical impactor with a hemispherical
head is first adhesive to an electromagnet in the energized state to a height of 2.3 m, and then it is released into free fall due to the power outage of the electromagnet; the impactor drops and finally impacts a laminated glass specimen which is placed on a rubber sheet; additionally, an acceleration sensor which is mounted inside the impactor is used to measure the impact acceleration of the impactor. (see Figs. 13 and 14)
6.3.2. Finite element model and material parameters

The geometric model is shown in Fig. 15, which includes three parts, i.e. an impactor, a laminated glass plate and a supporter. Only one quarter of the model is analysed due to symmetry. The initial velocity of the impactor is 6.78 m/s and the mass is 35.5 g. Symmetric boundary conditions are applied to the planes of symmetry. This model includes 47,088 hexahedral elements and the top view of the meshes are shown in Fig. 16. The finite element meshes of each layer are identical and 9 layers are involved among which 3 layers are for the top-layer glass, 5 layers for the bottom-layer glass and 1 layer for the PVB film. The total physical time is 0.6 ms. The time step is $6 \times 10^{-5}$ s. In this finite element model, the faces of glass surfaces are set to be segments, and the faces on the impactor and supporter that are possible to be in contact with glass are also set to be segments. Thus, this finite element model includes 23,839 initial segments. The scale factors $\omega_1$ and $\omega_2$ are set to be 1.0 and 0.4 [33], respectively.

The material of the impactor is assumed to be rigid, and the materials of glass and the supporter are linear elastic. Because the PVB material is strongly rate-dependent [19], Young’s modulus $E$ of the PVB film with the unit of MPa is expressed by: $E = 5.88 + 29.21|\dot{\varepsilon}|^{0.176}$, where $\dot{\varepsilon}$ is the strain rate, and 5.88 MPa is the value measured at the strain rate of $4.25 \times 10^{-4}$ s$^{-1}$. The materials of this model are shown in Table 2. The energy release rates of the glass material are 10 N/m for mode I and 50 N/m for mode II and III [24]. The friction coefficient is 0.1 in this case [24].

6.3.3. Simulation results and discussion

The impact fracture processes of the laminated glass plate are shown in Fig. 17. A schematic diagram shown in Fig. 18 illustrates the impact fracture mechanism of laminated glass. At first, compressive loading waves are generated in the impact area of top-layer glass and then they spread in all directions. The compressive waves travel vertically down to the boundary between top-layer glass and the PVB film first. Due to the viscoelasticity of the PVB film
which reduces the amplitudes and speeds of the compressive waves, the compressive waves are reflected to be tensile waves, which leads to the fracture of glass. Thus, the cracks initiate firstly at the bottom of top-layer glass right below the impactor at 20 $\mu$s, as shown in Fig. 17. Then the cracks propagate to the areas where accumulated reflected tensile waves reach the fracture criterion on the boundary between top-layer glass and the PVB film. These cracks are called in-plane cracks here. Meanwhile, the in-plane cracks propagate upwardly in the thickness direction along with the travelling of reflected tensile waves, which forms the radial cracks. Since the PVB material is rate-dependent which will be more tough along with the increase of strain rates, especially in the area right below the impactor, the compressive waves eventually pass through the PVB film and arrive to the bottom-layer glass. Finally, the compressive waves reach the free boundary of the bottom-layer glass and are reflected to be tensile waves, which leads to the initiations of cracks in the bottom-layer glass at 30 $\mu$s. Then the cracks will propagate in the same way as the cracks in the top-layer glass do. Besides, the compressive waves also travel horizontally and are reflected to be tensile waves on the free boundary, which forms the circular cracks of glass.
Fig. 17. Impact fracture processes of the laminated glass plate. The pictures from the top to the bottom are captured at 20, 33, 120, 267 µs.

Fig. 18. Schematic diagram of the spreading of impact waves in the laminated glass plate in the \( xz \) plane. Note that this diagram is not a one quarter model.

Through the post-processing software, the one-quarter model is reflected to be the whole model. As shown in Fig. 19, the fracture patterns of the laminated glass plate in the simulation with reflection boundaries involve both radial and circular cracks, which are in good agreement with those captured in the experiment. In addition, the impact force in the simulation also compares well with that measured in the experiment, as shown in Fig. 20.

In order to make the comparisons above meaningful, the numerical simulations of the finite element model with a smaller time step of \( 4 \times 10^{-5} \) ms and a bigger time step of \( 8 \times 10^{-5} \) ms are performed. Meanwhile, the simulations of the laminated glass model with finer meshes and coarser meshes are carried out. The model with finer meshes includes 141,808 elements and the other model includes 20,828 elements. The top views of these two models are shown in Fig. 21. The time step in the mesh convergence study is set to be \( 6 \times 10^{-5} \) ms. The simulation results of the mesh and time step convergence tests are shown in Figs. 20 and 22. Minor variations exist in the observed fracture patterns. This is because different mesh and time step sizes slightly influence the stress states of the laminated glass model during simulations. Meanwhile, the fracture criterion of the extrinsic cohesive model used in this paper is
stress-based. Small variations also exist in the impact forces which are the consequences of the minor variations of fracture patterns.

Another one simulation is carried out, in which the new contacts are updated by using LC-Grid. The total computational time for the updates of new contacts by using LC-Grid is calculated, which is compared with that by using the adaptive search algorithm. The other conditions in these two simulations are the same. As shown in Fig. 23, the total calculation time for the updates of new contacts by using LC-Grid is about 115 s, while the other simulation consumes about 18 s, which also verifies the computational efficiency of the adaptive search algorithm (see Fig. 22).

7. Conclusions

A three-dimensional computational framework for the impact fracture analysis of laminated glass has been developed and the corresponding code has been implemented into an in-house explicit software CDFP [19–22] by using the Fortran 90/95 programming language.

For the application of interest here, the glass material dissipates different fracture energies in normal and tangential fracture processes. Thus, a two-dimensional cohesive fracture law accounting for path-dependent behaviours is extended to three-dimensional problems. As for the global search algorithm, we resort to a linear contact algorithm termed LC-Grid in the regular search process and develop an adaptive search algorithm basing on LC-Grid. When new contacts are generated, the adaptive search is performed to find new potential contact pairs instead of the regular search. Additionally, a local contact search algorithm including node–face and edge–edge contacts is developed for robust searching in fracture simulations. The judgements of these two contact types are performed by using a unified inside–outside approach. Simple numerical examples have verified the computational efficiency of the adaptive search algorithm and the effectiveness of the local search algorithm. Then the cohesive model is coupled with the frictional
contact algorithm to account for the unilateral and frictional sliding effects of cohesive elements under mixed mode loading.

The fracture behaviours of a laminated glass plate under drop-weight impact loading are simulated by using the proposed method. The simulation results are in good agreements with the experimental outcomes in terms of the fracture patterns and the impact force. Moreover, time step and mesh convergence tests have been carried out. Numerical examples have demonstrated the capacity of the proposed method to simulate the impact fracture behaviours of laminated glass. However, the impact fracture analysis of the automotive windshield glazing remains a remarkable computational task. The GPU parallel computing of this method is a good choice. Besides, the meshes of the laminated glass plate model are specially designed to somehow reduce the severe mesh bias introduced by using brick elements. Further works such as the adaptive remeshing method [69] are required to solve this problem.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Nos. 10972079 and 11172104) and the International S&T Cooperation Program of Guangdong Province (No. 2013B051000012).

Appendix A. Buffer zone

Normally, a contact segments or edge is represented by using an axis-aligned cuboid bounding box with the minimum coordinates of \((x_{\text{min}}^{c1}, y_{\text{min}}^{c1}, z_{\text{min}}^{c1})\) and the maximum coordinates of \((x_{\text{max}}^{c1}, y_{\text{max}}^{c1}, z_{\text{max}}^{c1})\). Another one axis-aligned cuboid bounding box is formed with the minimum coordinates of \((x_{\text{min}}^{c2}, y_{\text{min}}^{c2}, z_{\text{min}}^{c2})\) and the maximum coordinates of \((x_{\text{max}}^{c2}, y_{\text{max}}^{c2}, z_{\text{max}}^{c2})\) which are given by:

\[
\begin{align*}
   x_{\text{min}}^{c2} &= x_{\text{min}}^{c1} - \chi \\
   y_{\text{min}}^{c2} &= y_{\text{min}}^{c1} - \chi \\
   z_{\text{min}}^{c2} &= z_{\text{min}}^{c1} - \chi
\end{align*}
\]
Fig. 22. The fracture patterns of the laminated glass plate in time step and mesh convergence tests. The pictures from the top to the bottom correspond to the models with a time step of $4 \times 10^{-5}$ ms, a time step of $8 \times 10^{-5}$ ms, finer meshes and coarser meshes, respectively.

\[ x_{c2}^{\max} = x_{c1}^{\max} + \chi \]  
\[ y_{c2}^{\max} = y_{c1}^{\max} + \chi \]  
\[ z_{c2}^{\max} = z_{c1}^{\max} + \chi \]  

where $\chi$ is the size of buffer zone, which is also determined by a scale factor $\omega_2$ and the characteristic length $L$:

\[ \chi = \omega_2 \cdot L. \]  

Therefore, the buffer zone $(x^b, y^b, z^b)$ is the intersection of these two bounding boxes: $(x^b, y^b, z^b) \in (\Omega^c_1 \cap \Omega^c_2)$, as shown in Fig. A.24.
Fig. A.24. A certain contact segment with buffer zone. In this figure, the domain of buffer zone is the intersection of two axis-aligned cuboid bounding boxes; the bounding box 1 is determined by the coordinates of the contact segment, and the bounding box 2 is determined by the size of buffer zone and the coordinates of the bounding box 1. The construction of buffer zone for a contact edge is the same with that for a contact segment.

Fig. B.25. A simple technique is introduced to avoid repetitive checking in the judgements of edge–edge contacts. Five contact edges are distributed in four cells, respectively. Among them, edge $e_1$ is in cell $C_1$; edge $e_2$ in cells $C_1, C_2, C_3$ and $C_4$; edge $e_3$ in cells $C_2$ and $C_4$; edge $e_4$ in cells $C_3$ and $C_4$; and edge $e_5$ in cell $C_3$.

Appendix B. Mask

The technique named *mask* is introduced to avoid repetitive checking for edge–edge contacts. When the loop cycles to a certain cell $C_i$, two edges in this cell, $e_m$ and $e_n$, will not be determined as a potential edge–edge contact pair if one of the following conditions is satisfied:

\[
\begin{align*}
MAX\{x^m_{\text{min}}, x^n_{\text{min}}\} &< x^i_{\text{min}} \quad \text{(B.1a)} \\
MAX\{y^m_{\text{min}}, y^n_{\text{min}}\} &< y^i_{\text{min}} \quad \text{(B.1b)} \\
MAX\{z^m_{\text{min}}, z^n_{\text{min}}\} &< z^i_{\text{min}} \quad \text{(B.1c)}
\end{align*}
\]

where $MAX\{\cdot\}$ is the maximum operator; $x_{\text{min}}, y_{\text{min}}$ and $z_{\text{min}}$ with the superscripts of $m$ and $n$ are the minimum coordinates of buffer zones of edges $e_m$ and $e_n$ in $x$, $y$ and $z$ directions, respectively; and $x_{\text{min}}, y_{\text{min}}$ and $z_{\text{min}}$ with the superscript of $i$ are the minimum coordinates of cell $C_i$ in $x$, $y$ and $z$ directions, respectively.

As illustrated in Fig. B.25, looping from cells $C_1$ to $C_4$, the edges 2 and 3 in cell $C_4$ have already been determined before (in cell $C_2$) as the condition Eq. (B.1c) is satisfied; similarly, the edges 2 and 4 in cell $C_4$ are not potential contact edges as the condition Eq. (B.1a) is met.
Appendix C. Procedures to obtain origin entities

The procedures to obtain origin contact entities are as follows: (1) In the last global search (regular search or adaptive search), the potential contact segments for each contact node are stored in the linked list. The contact pairs stored in the linked list will not be erased until a new regular search is performed. Therefore, we can efficiently obtain all the origin segments for a certain new contact node by retrieving all the potential contact segments of its mother node. A one-dimensional array with a size of $N_{new}$ which is the number of new contact nodes is used to store the corresponding mother node for each new contact node.

(2) Given a new cell, the origin nodes in this cell can be easily obtained by looping over all the origin segments retrieved above and the new segments of this cell. However, not all the origin nodes in the origin segments or the new segments are in the given new cell. Thus, the corresponding cell ID of each origin contact node is calculated during looping, and a one-dimensional array with a size of $N_{origin}$ which is the number of the total origin contact nodes is used to store the calculated cell ID to avoid repetitive calculation. Another one-dimensional array with a size of $E_{origin}$ is required to store the indexes of the origin contact nodes in this cell that have been updated, because each contact node may be shared by several segments.

(3) Likewise, the origin edges in a given new cell can also be obtained by looping over the origin segments retrieved above and the new segments of this cell. The contact edges are checked whether they are in this given cell during looping. For a retrieved contact edge it is in this cell provided that:

\[ x_{min} < x_{max}, \quad y_{min} < y_{max}, \quad z_{min} < z_{max}; \]

where the superscripts $e$ and $c$ denote the edge and cell respectively, the subscripts $min$ and $max$ the minimum and maximum, and $x$, $y$, and $z$ the coordinates of buffer zones of the edge and cell boundary respectively. A one-dimensional array with a size of $E_{origin}$ which is the number of the total origin contact edges is allocated to store the indexes for the contact edges in this cell that have been updated, because each contact edge may be shared by several segments.

Appendix D. Implementations of the adaptive search algorithm

Appendix E. Inside–outside approach

Taking node $p$ and edge $\overline{ab}$ as an example, the inside–outside status can be checked as follows:

(1) Calculate the surface normal $n_s$:

\[ n_s = \sum_{i=1}^{4} n_i / |\sum_{i=1}^{4} n_i| \quad (E.1) \]

where $n_i$ is the cross product of its two connective edge vectors; take $n_1(n_a)$ as an example:

\[ n_1 = n_{ab} \times n_{ad}. \quad (E.2) \]

(2) Calculate the judgement vector $n_{abp}$:

\[ n_{abp} = n_{ab} \times n_{ap}. \quad (E.3) \]

(3) Node $p$ is inside edge $\overline{ab}$ provided that the projection $d_1$ of the judgement vector onto the mesh normal $n_s$ is positive:

\[ d_1 = n_{abp} \cdot n_s > 0. \quad (E.4) \]

If node $p$ is inside face $a-b-c-d$, the penetration depth can be calculated as follows:

(1) The projection point $o$ of node $p$ onto face $a-b-c-d$ along the surface normal $n_s$ can be obtained as:

\[ x_o = \sum_{i=1}^{4} N_i x_i \quad (E.5) \]

where $x_o$ is the position vector of point $o$, and $x_i$ is the position vector of node $i$ in surface $a-b-c-d$. $N_i$ is the shape function with respect to node $i$ of the surface segment and is given approximately as: $N_1 = \Phi_2 \Phi_3 / \Phi, N_2 = \Phi_3 \Phi_4 / \Phi,$
The implementations of adaptive global search

1. LOOP I: Loop over new contact segments, \( i = 1, 2, 3, \ldots, M_{\text{new}} \)
   1.1 Map them into layers and store them to the New-Segment-Z list
   END LOOP I

2. LOOP II: Loop over new contact edges, \( i = 1, 2, 3, \ldots, E_{\text{new}} \)
   2.1 Map them into layers and store them to the New-Edge-Z list
   END LOOP II

3. LOOP III: Loop over new contact nodes, \( i = 1, 2, 3, \ldots, N_{\text{new}} \)
   3.1 Map them into layers and store them to the New-Node-Z list
   END LOOP III

4. LOOP IV: Loop over non-empty layers, \( i_z = 1, 2, 3, \ldots, n_z \)
   4.1 (a) Retrieve all the new contact segments from the New-Segment-Z list
   (b) Map them into rows and store them to the New-Segment-Y list
   4.2 (a) Retrieve all the new contact edges from the New-Edge-Z list
   (b) Map them into rows and store them to the New-Edge-Y list
   4.3 (a) Retrieve all the new contact nodes from the New-Node-Z list
   (b) Map them into rows and store them to the New-Node-Y list
   4.4 LOOP V: Loop over non-empty rows, \( i_y = 1, 2, 3, \ldots, n_y \)
     4.4.1 (a) Retrieve all the new contact segments from the New-Segment-Y list
     (b) Map them into cells and store them to the New-Segment-X list
     4.4.2 (a) Retrieve all the new contact edges from the New-Edge-Y list
     (b) Map them into cells and store them to the New-Edge-X list
     4.4.3 (a) Retrieve all the new contact nodes from the New-Node-Y list
     (b) Map them into cells and store them to the New-Node-X list
     4.4.4 LOOP VI: Loop over non-empty cells
       (a) Retrieve new contact nodes from the New-Node-X list
       (a1) Obtain origin segments by retrieving the contact segments of its corresponding mother node
       (a2) Record new nodes and origin segments to the node–face pair list if the rough check is satisfied
       (a3) Obtain origin nodes in the origin segments
       (a4) Obtain origin edges in the origin segments
       (b) Retrieve new contact segments from the New-Segment-X list
       (b1) Obtain origin nodes in the new segments
       (b2) Obtain origin edges in the new segments
       (c) Record new nodes and new segments to the node–face list if the rough check is satisfied
       (d) Record origin nodes and new segments to the node–face list if the rough check is satisfied
       (f) Retrieve all the new contact edges from the New-Edge-X list
       (g) If the judgement test is satisfied, record the two new edges to the edge–edge contact list
       (h) If the judgement test is satisfied, record the new edges and the origin edges to the edge–edge contact list
       (i) Cell migrations for new contact segments and new contact edges
     END LOOP VI
   4.4.5 Row migrations for new contact segments and new contact edges
   END LOOP V
   4.5 Layer migrations for new contact segments and new contact edges
   END LOOP IV

\[ N_3 = \Phi_4 \Phi_1 / \Phi \text{ and } N_4 = \Phi_1 \Phi_2 / \Phi, \] with \( \Phi = (\Phi_1 + \Phi_3)(\Phi_2 + \Phi_4) \) and \( \Phi_i = d_i \) which is calculated through Eq. (E.4).

(2) The penetration depth \( g_n \) is given by:
\[ g_n = n_s \cdot (x_p - x_o) \] (E.6)
where \( x_p \) is the position vector of node \( p \); and \( g_n < 0 \) means there is penetration, while \( g_n \geq 0 \) means there is no penetration.

Appendix F. Procedures for the judgement of an edge–edge contact

The judgement procedures for edges \( \overrightarrow{ab} \) and \( \overrightarrow{cd} \) are as follows:
(1) The vector dot product of the edge normals \( n_{ab} \) and \( n_{cd} \) should be less than zero:
\[ n_{ab} \cdot n_{cd} < 0 \] (F.1)
where the edge normal is the summation of the normals of the two segment surfaces adjacent to the contact edge which can be calculated through Eq. (E.1).
Fig. F.26. The determination of the position of contact point o in an edge–edge contact. In this figure, edge $a'b'$ is the projection edge of edge $ab$ onto the oriented virtual plane $c$-$d$-$e$-$f$ with the surface normal of $n_f$; $\Phi_1$ and $\Phi_3$ are the area coordinates which are used to determine the position of contact point o.

(2) We define an oriented virtual plane to facilitate the edge–edge contact judgement by using the inside–outside approach. The normal $n_f$ of the oriented virtual plane $c$-$d$-$e$-$f$ is defined as:

$$n_f = cd \times ab$$  \hspace{1cm} (F.2)

where $ab$ and $cd$ are the edge vectors.

The edge $ab$ is defined to be in contact with edge $cd$ if node $b$ of edge $ab$ is inside the oriented virtual plane $c$-$d$-$e$-$f$, which is judged by using the inside–outside approach addressed in Appendix E. Note that the surface normal $n_s$ in Fig. 5 is given here by:

$$n_s = \frac{\text{sgn}(n_f \cdot n_{cd}) \cdot n_f}{|n_f|}$$  \hspace{1cm} (F.3)

where $\text{sgn}(\cdot)$ is the signum function.

(3) If the two edges are in contact, we should determine the contact point. In Fig. F.26, the area coordinates $\Phi_1$ and $\Phi_3$ can be obtained through Eq. (E.4); then the position $x_o$ of contact point o is given by:

$$x_o = \sum_{i=1}^{2} N_i x_i$$  \hspace{1cm} (F.4)

where $x_1$ and $x_2$ are the position vectors of nodes $c$ and $d$, respectively; and $N_i$ is the shape function, which satisfies: $N_1 = \Phi_1 / (\Phi_1 + \Phi_3)$ and $N_2 = 1 - N_1$.

As the penetration depth at the contact point o is equal to that at the point $b'$, we can obtain the penetration depth $g_e$ of the edge–edge contact through Eq. (E.6). $g_e < 0$ means there is penetration, while $g_e \geq 0$ means there is no penetration.

References


