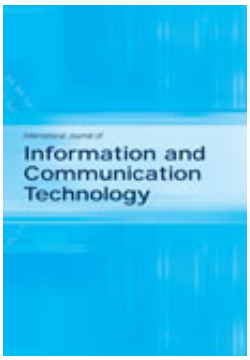
INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research



International Journal of Information and Communication Technology

ISSN online: 1741-8070 - ISSN print: 1466-6642
https://www.inderscience.com/ijict

Method of target damage probability distribution simulation and evaluation based on GPU parallel computing

Xiaoyun Lei, Yaping Tan, Lihua Zhu

Article History:

Received:	30 July 2024
Last revised:	12 September 2024
Accepted:	14 September 2024
Published online:	10 October 2024

Method of target damage probability distribution simulation and evaluation based on GPU parallel computing

Xiaoyun Lei and Yaping Tan

School of Information Technology,
Jiangsu Open University,
Nanjing, Jiangsu Province, China
Email: leixy@jsou.edu.cn
Email: tanyp@jsou.edu.cn

Lihua Zhu*

School of Mechanical Engineering,
Nanjing University of Science and Technology,
Nanjing, Jiangsu Province, China
Email: njust_academic@163.com
*Corresponding author

Abstract: For the projectiles with proximity-fused warheads used to destroy the target, the most vulnerable information is vital to destroy targets efficiently. So, it is important to quickly locate the most vulnerable position of the target in combat. An algorithm combining GPU parallel computing and numerical simulation is proposed. For a specific target, a slicing processing method is used to define the spatial position parameters, and the damage probability distribution in the space near the target is established to determine the most vulnerable position. In the example, the computation speed of the method is about 3.6 times higher than that of the CPU serial calculation method. It could quickly locate the most vulnerable position of a certain ballistic missile, namely, when the projectile was situated in the plane of 1/5 projectile length from the bottom of the target, the target would be damaged with the highest probability of 90%.

Keywords: damage probability; vulnerable position; GPU parallel computing; numerical simulation.

Reference to this paper should be made as follows: Lei, X., Tan, Y. and Zhu, L. (2024) 'Method of target damage probability distribution simulation and evaluation based on GPU parallel computing', *Int. J. Information and Communication Technology*, Vol. 25, No. 7, pp.37–56.

Biographical notes: Xiaoyun Lei graduated from the Nanjing University of Science and Technology in 2021. She works in Jiangsu Open Univer-sity. Her research interests include intelligent ammunition technology and robotics.

Yaping Tan graduated from Nanjing University of Science and Technology in 2022 and now works at Jiangsu Open University. Her research interests include terminal effects and impact dynamics.

Lihua Zhu graduated from Southeast University in 2016. She now works in Nanjing University of Science and Technology. Her research interests include robotics and remote sensing technology.

1 Introduction

For a target, the damage assessment is also called the vulnerability study. In the early research, the anti-damage ability of targets was explored through a large number of shooting tests (Starks, 1990; Guber et al., 1967). With the continuous development of computer technology, the simulation evaluation method of target vulnerability has developed into the main basic research method (Muuss et al., 1983; Wasmund, 2001). Domestic simulation evaluation methods began in the 1980s, mainly from two aspects of mathematics and image. The mathematical method is based on a large amount of known prior information. It generally simulates the entire process of ammunition acting on the target or analyses the actual drill and combat process, and executes the entire damage process from top to bottom. Due to the integration of a large amount of prior information, the calculation is very large, resulting in a serious delay in the acquisition of damage information. These kinds of methods are more suitable for the design stage of ammunition and protective armour, or post-war assessment. For example, Kong et al. proposed an evaluation method combining machine learning (Tekwa, 2023) and fuzzy hierarchical analysis (Tekwa, 2023; Zhang et al., 2021; Wang et al., 2020; Kong et al., 2023; Deng et al., 2022). This method focuses on improving the accuracy of the detection of the key components of the target. The fuzzy hierarchical analysis is difficult to solve the problem of repeatability evaluation, and the real-time performance is hard to be guaranteed. Chen et al. proposed a method based on Bayesian network models that can make inferences about uncertain damage situations (Deng et al., 2022; Chen et al., 2022). Jian et al. (2018) proposed a method combining fuzzy reasoning and Monte Carlo, but due to the large complexity of the parameters of the model, there is also a problem of information lag. The image-based evaluation method can basically meet the requirements of actual damage effect measurement, but accurate image registration still requires manual intervention, and it is still difficult to achieve a real-time evaluation of moving targets (Li et al., 2024; Liu et al., 2024; Xu et al., 2024). Foreign simulation and evaluation methods for target vulnerability have reached a high level, such as LMP-3 in Sweden (Gyllenspetz and Zabel, 1981), VAREA and VAST in the United States (Shah and Mehtre, 2015), and SLAMS in Canada (Shewchenko, 2012). A relatively perfect damage information database has been formed, and the efficiency of information acquisition has been improved.

To sum up, this paper proposes a simple damage assessment simulation method based on GPU parallel computing in order to achieve the real-time solution of the optimal detonating azimuth, distance and other combat information of warhead, and to solve the problem of time delay in the vulnerability assessment (Zhang et al., 2017; Fu et al., 2016) based on mathematical method. Firstly, the three-dimensional equivalent model and damage element information database of the target are designed. Then, a multi-threaded parallel computing software framework is constructed for the intersection calculation of projectile and target and the identification of effective damage element. The intersection algorithm is motivated by the principle of intersection between vector and surface

element. At the same time, the damage criterion of kill element penetrating target based on THOR theory is established. Since there are a large number of the same computational processes in the intersection calculation, we introduced the SPMD model (referring to the Single program multiple data flow model), and used the principle of multi-threaded parallel computing to create a kernel function to parallelise the cyclic intersection calculation, so as to accelerate the computation process. At last, according to a slicing processing of the spatial position information near the target, the spatial damage probability of the target in different spatial positions is simulated and computed with multithreaded parallel computing method, which can be used to quickly locate the most vulnerable position information of the target. It can provide important target intelligence support for the campaign actions.

2 Calculation method for target damage probability distribution in three-dimensional space

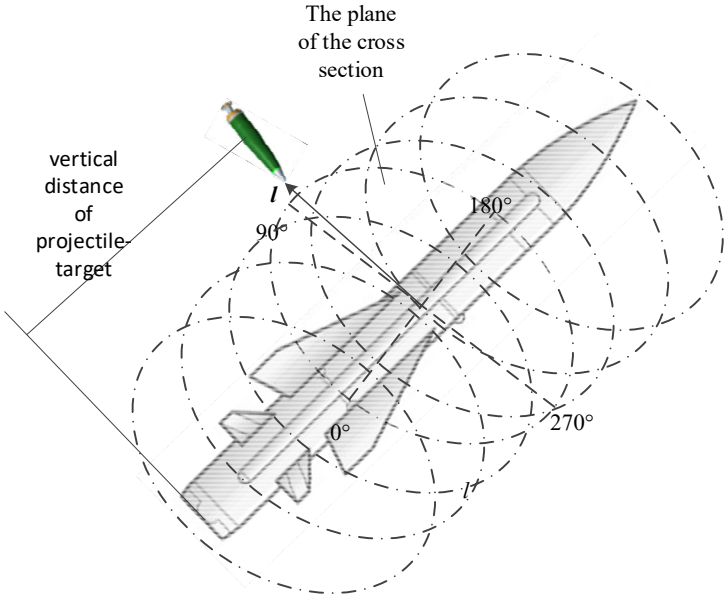
For the establishment of a damage probability model in the target space, it will be calculated by the damage assessment method and numerical simulation. The probability model will vary with the differences between the vulnerable parts of different targets in performance and location, and it may not have strong spatial distribution regularity in space. Therefore, the spatial damage probability for a specific target was analysed, and a database could be established for multiple targets to access the spatial damage probability model in the form of a database. For a target in the air, as shown in Figure 1, the space near the target was divided into layers by the cross section of the target passing through the geometric centre, and the influence law of the projectile-target orientation on the damage probability of the target was studied. The attitude of the projectile had a certain influence on the formation of the damage field, but in this study, the spherical damage field formed by natural fragments was considered, and the influence of attitude was temporarily ignored. In the figure is the position vector of the projectile and target; For the definition of the orientation of the projectile relative to the target centre on the sectional plane, when viewed from the head of the target, the orientation was 0° in the vertical direction and 90°–360° (0°) in the clockwise direction respectively. Aiming at the sectional plane, the variation law of the damage probability of several sliced planes within a certain projectile-target distance was explored, and the spatial damage probability distribution model of the target was established.

2.1 Intersection calculation and effective damage element judgment method

In this study, the number of effective fragments suffered by the target was calculated and counted by GPU parallel numerical calculation, and then the damage of each component was calculated. Finally, the overall damage probability of the target was solved according to the complete failure damage tree. According to the static fragment field information of the projectile calculated by AUTODYN simulation, the information including the mass, volume, velocity vector, and position vector of the damage element at the time of explosion was derived in an XLS format. The target was modelled using 3D software (such as UG and ProE, etc.), and the target model was exported in the form of an STL file in the ASCII code format. The STL file contained the geometric information of the target, that is, the vertex coordinates and the external normal vector of the triangular surface

element (SE), which was an ideal file format for calculating the intersection between the fragment and the target.

Figure 1 Diagram of division of space around the target (see online version for colours)



Given the attitude of the projectile and the target upon intersection, the damage probability of each component of the target was related to the following parameters: the projectile-target position vector, the target velocity vector, and the projectile velocity vector.

$$P = p\left(l, V_t, V_m\right) \tag{1}$$

2.1.1 Intersection calculation algorithm

With the known projectile-target intersection information, it was necessary to transform the projectile and the target into the same coordinate system for damage calculation when calculating the number of effective damage fragments. In the simulation calculation of AUTODYN, the information of the projectile system in the static fragmentation field could be obtained, and the relative velocity with the projectile needed to be synthesised before the damage assessment to obtain the dynamic damage field of the projectile. The following coordinate system was defined: the ground coordinate system was used to determine various ballistic parameters of the projectile and the target, such as the positions of the target and the projectile at the intersection point, velocity, and attitude angle. The ground coordinate system (g) is represented by O-$x_g y_g z_g$. The projectile body coordinate system (m) was used to describe the scattering characteristics of fragments, expressed by O-$x_m y_m z_m$. The target coordinate system (t) was used to describe the radiation characteristics and distribution position of various physical fields of the target, as well as the position and distribution of the key parts of the target. Assuming that the geodetic coordinate system coincided with the projectile coordinate system, then the

projectile was transformed from the projectile coordinate system to the geodetic coordinate system and the target was transformed from the target coordinate system to the geodetic coordinate system. According to the above two transformations, the target could be transformed into the projectile coordinate system, and similarly, the projectile could be transformed into the target system. The direction of the velocity vector V_m of the projectile in the ground coordinate system could be determined by two angles: the yaw angle and trajectory angle of the projectile. The trajectory angle θ_m is the included angle between the longitudinal axis of the projectile and the horizontal plane (assuming that the velocity of the projectile is consistent with the longitudinal axis of the projectile), and the yaw angle φ_m is the included angle between the projection of the longitudinal axis of the projectile on the horizontal plane and the O-x_g axis. With known θ_m and φ_m, the three components of the projectile velocity in the ground coordinate system could be determined by the following coordinate transformation, and the coordinate transformation matrix is M_{xg}.

$$M_{xg} = \begin{bmatrix} \cos\varphi_m \cos\theta_m & -\sin\varphi_m & \cos\varphi_m \sin\theta_m \\ \sin\varphi_m \cos\theta_m & \cos\varphi_m & \sin\varphi_m \sin\theta_m \\ -\sin\theta_m & 0 & \cos\theta_m \end{bmatrix} \tag{2}$$

Then, the three components of the projectile velocity in the ground coordinate system are as follows:

$$\begin{bmatrix} V_{mxg} \\ V_{myg} \\ V_{mzg} \end{bmatrix} = M_{xg} \begin{bmatrix} V_m \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

Similarly, with the known heading angle φ_t and trajectory angle θ_t of the target, the component of the target velocity vector (V_t, 0, 0) in the ground coordinate system is the same as the above formula.

For the convenience of research, the following assumptions were made in the process of coordinate system transformation:

a The angle of attack, sideslip angle, and roll angle of the projectile are zero, that is, the longitudinal axis of the projectile is consistent with the velocity direction of the projectile.

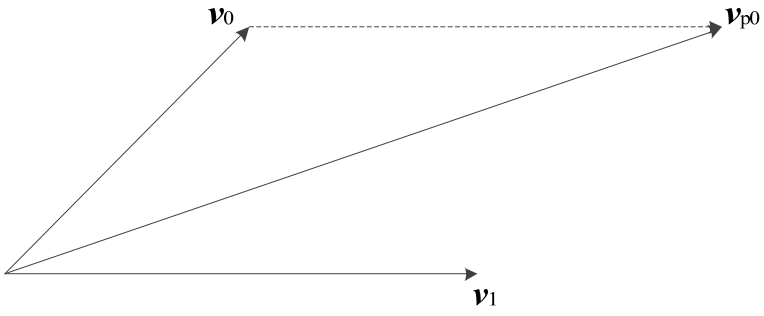
b The angle of attack, sideslip angle, and roll angle of the target are zero, that is, the longitudinal axis of the target is consistent with its velocity direction.

After the projectile explodes, the scattering velocity vector of its fragment field will change under the influence of the projectile's transport velocity vector. Let the transport velocity vector of the fragment be v_1, and the initial velocity vector of a fragment after static explosion of the projectile be v_0. The dynamic scattering angle and dynamic initial velocity v_{p0} of the fragment could be calculated by the following formula:

$$\varphi = \arctan \frac{v_0 \sin\varphi_0}{v_0 \sin\varphi_0 + v_1} \tag{4}$$

$$v_{p0} = \sqrt{v_0^2 + v_1^2 + 2v_0 v_1 \cos\varphi} \tag{5}$$

Figure 2 Composition of velocities of fragment



The projectile-target intersection is a dynamic intersection process, that is, the projectile continues to move along its own trajectory while the fragments fly to the target. Therefore, the intersection result is calculated by relative motion during the period from the formation of fragments to hitting the target, that is, the target is transformed into the projectile coordinate system, and the intersection result of fragments relative to the target is calculated under the projectile system, assuming that the target moves in a straight line at a uniform velocity in this short period.

When the damage element flies to the target at a resultant velocity, its velocity actually attenuates continuously in the air. Generally, the following drag formula is adopted:

$$v_p = v_{p0} e^{-\frac{C_x A \rho x}{2m_p}} \tag{6}$$

where v_{p0} is the initial fragment velocity (m/s), v_p is the attenuated fragment velocity (m/s), x is the fragment flight distance (m), ρ is the air density (kg/m^3), m_p is the fragment mass (kg), A is the average windward area (m^2), and c_x is the fragment resistance coefficient, which is related to the fragment shape and velocity. The average windward area A of the fragment is associated with the fragment mass and shape, generally expressed by equation (7). Therein, K is the fragment shape coefficient (m^2/kg$^{2/3}$).

$$A = \mathrm{K}m_p^{\frac{2}{3}} \tag{7}$$

Table 1 Empirical values of c_x and K of typical steel fragments

Fragment shape	*Spherical shape*	*Square shape*	*Long strip shape*	*Irregular shape*
c_x	0.97	1.56	1.3	1.5
K (10^{-3}m^2/kg$^{2/3}$)	3.07	3.09	3.3–3.8	4.5–5

The drag coefficient c_x and shape coefficient K of fragments are mainly obtained by experiments. Table 1 lists the values of drag coefficient c_x and shape coefficient K of various steel fragments. Among them, influenced by the shape change, the shape coefficient K of rhombic, elongated, and irregular shapes is a range.

In the process of fragment-target intersection, whether the fragment hits the target can be determined by judging whether the velocity vector of the damage element intersects with the surface element of the target component. In case of intersection, it means that the damage element hits the target component. By using the geometric characteristic data of

the target component and the coordinate transformation theory of analytic geometry, the intersection of the vector and the surface element can be judged, and then the coordinates of the geometric intersection point, impact angle, and other data can be obtained. The penetration equation can be used to judge whether the damage element can penetrate the target component and obtain the residual mass and velocity of the damage element. On this basis, whether the motion of the damage element ends is decided and whether the penetration damage succeeds is judged.

The basic algorithm for the intersection judgment is as follows:

Algorithm: Intersection judgment

It is known that:	The STL file of the target takes a triangle as the basic unit;
	The coordinate system of both projectile and target has been transformed to the same coordinate system;
	The three vertices of the triangular surface elements that constitute the target are A, B, and C respectively, the unit normal vector is n, the unit vector of the fragment velocity vector is m, and the initial position point of the fragment is P_0;
Step 1:	The included angle between two vectors is calculated as $\theta = \text{acos}(mn/(\|m\|\|n\|))$
Step 2:	The intersection point between the straight line and plane is: $Cross\_P = m(nA - nP_0)/mn + P_0$
Step 3:	Whether the intersection point $Cross\_P$ is inside the triangular surface element or the sideline is judged:

$$\begin{cases} n_{AC} = C - A \\ n_{AB} = B - A \\ n_{AP} = Cross\_P - A \\ tmp = \dfrac{1}{\left[(n_{AC}, n_{AC})(n_{AB}, n_{AB}) - (n_{AC}, n_{AB})(n_{AC}, n_{AB})\right]} \\ u = \left[(n_{AB}, n_{AB})(n_{AC}, n_{AP}) - (n_{AC}, n_{AB})(n_{AB}, n_{AP})\right] \cdot tmp \\ v = \left[(n_{AC}, n_{AC})(n_{AB}, n_{AP}) - (n_{AC}, n_{AB})(n_{AC}, n_{AP})\right] \cdot tmp \end{cases}$$

If:	$(u < 0 \,\|\|\, u > 1) \,\|\|\, (v < 0 \,\|\|\, v > 1)$ is true, the condition is not satisfied (|| indicates logic OR);
If:	$(u + v <= 1)$ is true, the condition is satisfied, and the distance d between the fragment position and intersection point and the target angle λ in this case can be calculated accordingly.
	$d = \|\|Cross\_P - P_0\|\|$
	$\lambda = (\pi - \theta)$

The above intersection method is used to calculate the intersection information of a single fragment and a surface element. When the whole damage field intersects with all surface elements of the whole target, the calculation amount is amazing. In this study, therefore, the idea of parallel computing was followed, and rapid resolving was performed through the GPU parallel computing method (Hong et al., 2019; Huang and Xie, 2021; Ma et al., 2022).

The fragments (0, 2 … M) of the damage field, each component of the target, and the triangular surface element of each component are numbered, and the component number corresponds to the position (row, column) of the component in the damage tree, and the

surface element number contained in the component is 0, …, *N*. The damage field is digitised into an M'8 matrix, and the attribute parameters of each damage element are the row elements of the matrix, including the coordinate component value, velocity vector component value, volume, and mass of the damage element. The STL file of each component of the target can be read and then digitised into an N'12 matrix, and each row contains the geometric information of a triangular surface element, including the normal vector component value of the surface element and the coordinate component values of three vertices.

GPU accelerated parallel computing was performed using NVIDIA's CUDA C programming technology, which could greatly improve the repeated computing efficiency. The CUDA application program can directly call the bottom-layer CUDA driver to call GPU hardware for parallel computing. If the commonly used serial calculation algorithm is adopted, the code of the geometric intersection calculation part is as follows. It can be seen that the whole iterative loop computing process will be repeated (N + 1)' (M+1)' (T + 1) times, where T + 1 is the number of components included in the target. And there is no dependence between each iterative computing and the next-round computing. Therefore, the tasks or data can be divided and loops can be parallelised.

Figure 3 Part of code for intersection calculation in serial computing method

```
for t=0:T
    //Load the geometry information matrix of the first part
    Data_surface=Surfaces.DIRS(t).name;
    //intersection judgment
    for m=0:M
      l=Line(m,:);%%The row number is the fragment number
        //N+1 is the number of facet elements of the current part
      for n=0:N
        p=Plane(n,:);
          //Surface element j of k part intersect with fragment i
        [flag,theta,distance,point]=sub_intersection(p,l);
        If flag~=0
          EffectAct(end+1,:)=[mn flag theta distance point];
        end
      end
    end
    。 。 。 //Process
  //Intersection result
geo_jiaohui=[ geo_jiaohui;temp2];
end
```

In CUDA, the loop is parallelised by creating kernel functions. At this time, the loop control variable will no longer represent the number of loops, but a variable used to represent the currently allowed threads. CUDA provides a special variable – thread index (thread ID) to identify each thread, which can be used as a subscript to access the array. Each thread performs the operation of intersection judgment, with the same code but

different data, that is, the SPMD (Single program multiple data stream) model in CUDA is adopted.

Figure 5 Schematic design of thread block layout

thread0~15，thread block0	thread16~31，thread block0
thread32~47，thread block0	thread48~63，thread block0
thread64~79，thread block0	thread80~95，thread block0
thread96~111，thread block0	thread112~127，thread block0
thread128~143，thread block0	thread144~159，thread block0
thread160~175，thread block0	thread176~191，thread block0
thread192~207，thread block0	thread208~223，thread block0
thread224~239，thread block0	thread240~255，thread block0
thread0~15，thread block1	thread16~31，thread block1
thread32~47，thread block1	thread48~63，thread block1
thread64~79，thread block1	thread80~95，thread block1
thread96~111，thread block1	thread112~127，thread block1
thread128~143，thread block1	thread144~159，thread block1
thread160~175，thread block1	thread176~191，thread block1
thread192~207，thread block1	thread208~223，thread block1
thread224~239，thread block1	thread240~255，thread block1
thread224~239，thread blocki	thread240~255，thread blocki

As shown in the following figure, the surface element matrix (marked as *a*) and the fragment field velocity vector matrix (marked as *b*) are both two-dimensional arrays. In order to maximise the equipment utilisation, each thread block has 256 threads, and 32 threads (or integer multiples thereof) form a thread bundle, so a thread block has eight thread bundles. In addition, we suggest that the array span should be an integer multiple of the thread bundle size, if the array is not filled, and if judgment statement is used to

omit the calculation of the filled part in the calculation. We choose a rectangular layout, as shown in the figure, to read the data once, to access the memory continuously in the form of rows, and then to perform the next intersection calculation. For *a* and *b* matrices, preprocessing is needed to design the layout to meet the requirements of computing efficiency, and GPU parallel computing sacrifices the memory for a high computing speed. The rows *a* and *b* are complemented to 32 elements, the columns to an integer multiple of 8, and the sizes of the two matrices become M'32 and N'32. Moreover, leng_blocks = max{M', N'}' (T + 1), then the layout design code is:

```
dim3 threads_rect(32, 8);
dim3 blocks_rect(1,leng_blocks/8);
```

Therefore, the indexes of the array in two dimensions are:

```
idx = (blockIdx.x*blockDim.x) + threadIdx.x;
idy = (blockIdx.y*blockDim.y) + threadIdx.y;
```

BlockIdx.x is an index of thread blocks in a thread grid in the *x* direction. In this study, the *x* dimension index is always 0; BlockIdx.y is the index in the y direction, which is within 0-leng_blocks in this study; blockDim.x = 32, blockDim.y = 8, and threadIdx.x = 0~31; threadIdx.y = 0-7. In the array, idy corresponds to the row number and idx corresponds to the column number. There is no need to worry about the number of threads to open. Even if there are 64 million threads to process 64 million array elements, and each array element is a single-precision floating-point number, then each element occupies 4 bytes, totalling about 256 MB of data storage space. Now basically all GPUs support this size of space.

After the layout is designed, the intersection algorithm of the kernel function can be designed. The memory of two matrices is allocated on GPU, and the intersection results are also stored in the matrices and returned to CPU. Each row of the returned matrix includes eight elements (fragment number, component number, surface element number, intersection marks, target angle, projectile-target distance upon intersection, and coordinates of the intersection point).

The call of this kernel function is as follows, and the partial codes of main functions are displayed in Appendix A3.

```
kernel <<< blocks _ rect, threads _ rect >>> (a, b, c);
```

2.1.2 Calculation of spatial damage probability distribution

Most of the fragments that hit the target do not necessarily cause damage to the target, so the fragments with a damage effect need to be further discriminated by using other conditions in the damage element that hits the target. Generally speaking, fragments can cause mechanical damage to the target by kinetic energy, that is, forming holes or penetrating the target. Fragments penetrating the target usually cause damage to the target. THOR's formula (Chen et al., 2015; Cho et al., 2018; Wang et al., 2022) provides a simple method for preliminarily calculating the ultimate velocity of fragments penetrating a target with a specific thickness and material, as well as the residual velocity and residual mass of fragments after penetrating the target. For steel fragments, the parameters needed for calculation include the fragment mass, fragment impact velocity, and so on. The THOR equation does not include the secondary fragments formed by the

collapse or fracture of the target material. After counting the number of broken fragments on each component, the damage probability of each key component can be obtained according to the damage criterion. After obtaining the damage probability distribution of the key components of the target, the probability operation is carried out according to the logical connection relationship between the key components in the damage tree, that is, the damage probability of each subsystem (intermediate event) is calculated through the damage probability of the key component (bottom event), and then the damage probability corresponding to each functional system or damage level (top event) is obtained.

Let the average damage rate of a single effective damage element to the target be p_i, and assume that the target killing by each damage element as an independent event, then the probability of the target being damaged under the condition that i damage elements penetrate the target component is:

$$P_i = 1-(1-p_i)^i \tag{8}$$

For a specific damage tree, the logical relationship between its key components includes two connection modes: series connection and parallel connection, in which the series damage tree is a logical AND operation relationship, while the parallel damage tree is a logical OR operation relationship. For J mutually independent killing events E_1, E_2, …, EJ connected in series, the result event Q will only occur if and only if J killing events occur simultaneously. When damage events are mutually independent, the occurrence probability of the result event can be expressed as below:

$$P(Q) = P\left(\prod\nolimits_{k=1}^{J} E_k\right) = \prod\nolimits_{k=1}^{J} P(E_k) \tag{9}$$

For J mutually independent damage events E_1, E_2, …, EJ connected in parallel, the result event Q will occur as long as one damage event occurs, and the occurrence probability of the result event can be expressed as follows:

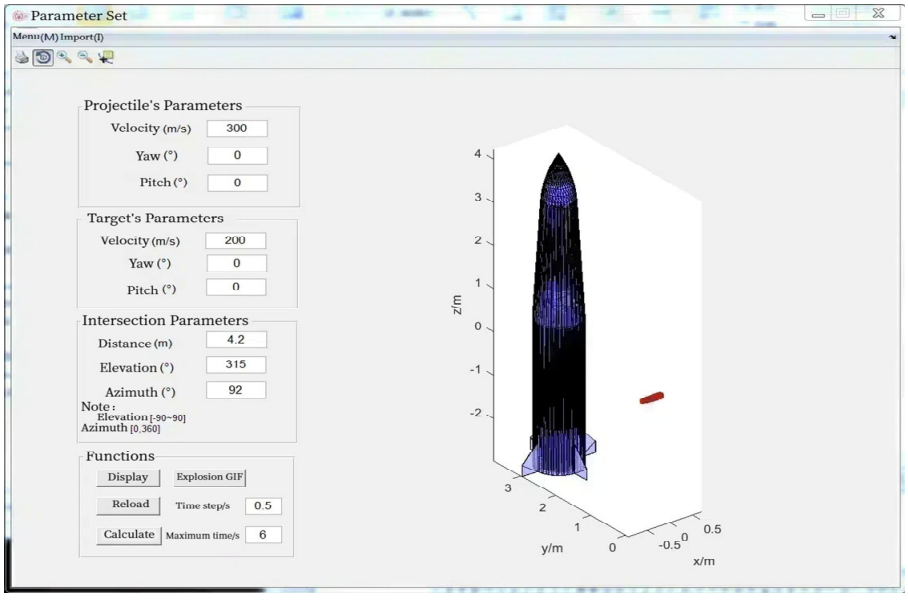
$$P(Q) = P\left(\sum\nolimits_{k=1}^{J} E_k\right) = 1-\prod\nolimits_{k=1}^{J}\left[1-P(E_k)\right] \tag{10}$$

3 Simulation analysis

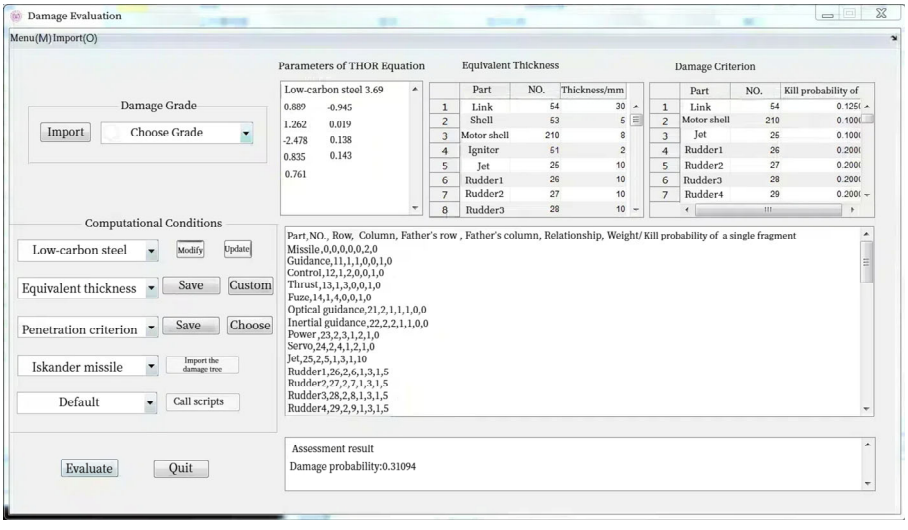
In this study, the damage probability calculation method was based on the 3D STL model of target components and the AUTODYN damage field simulation model (Pang et al., 2022; Thakur et al., 2022), and the parallel calculation was adopted to speed up the calculation process. On this basis, effective assumptions were made, and the number of damage elements that hit the target with damage effects was calculated through numerical simulation, parallel calculation, and statistics. Finally, the overall damage probability of the target was solved through the damage tree. Therefore, such series of complicated calculations with a large data size cannot be implemented just by a simple mathematical formula. Before completing the damage evaluation calculation of the target, a database of typical targets and damage fields was established by using SQL, as shown in the example in Appendix A2. An interactive simulation platform was compiled by using MATLAB and CUDA C (Defez et al., 2022; Zhang et al., 2011; Hou et al., 2017; Yamout et al., 2022; Schmid et al., 2022), as shown in Figure 6, where Figure 6(a) is the interface for

setting projectile and target parameters, and Figure 6(b) is the interface for damage assessment.

Figure 6 GUI of simulation platform (see online version for colours)



(a)

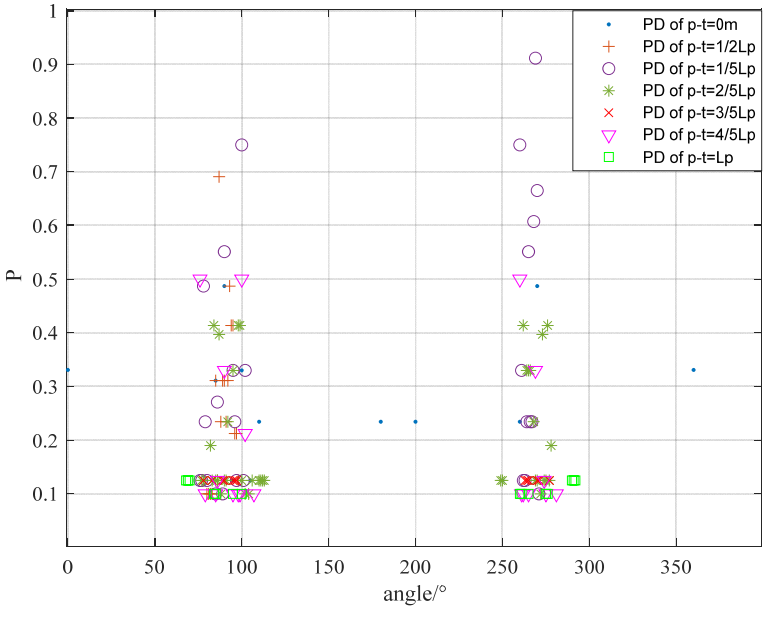


(b)

In order to determine the influence of the orientation of the projectile in the layered space near the target on the damage probability of the target, considering the spatial position of the target centre at a vertical distance of 5 m from the axis of the projectile, the complete damage probability of a single projectile to a target under different spatial intersection conditions was quickly calculated by using the accelerated parallel computing method.

The target was divided into 50 components, with a total of 67,487 surface elements, and the damage field contained about 780 damage elements. It took about 7 min to calculate an example by using the CPU serial computing method, and the average time ratio between them was about 3.6 after using the GPU parallel computing method (of course, this result is high correlated with the performance of the computer itself and the degree of program optimisation), indicating that parallel computing can indeed improve the computing efficiency of damage assessment to a certain extent, but the degree of improvement has a great relationship with computer performance, program structure optimisation, algorithm optimisation, and data access (Zhou et al., 2021; Kistler and Franz, 2001).

Figure 7 Diagram of damage probability distribution of a single projectile to a specific target (see online version for colours)



Note: 'PD of p-t' means perpendicular distance of projectile and target. Lp is the length of projectile.

Considering the relationship between the degree of damage and the relative position of the projectile and a specific target, the hierarchical spatial damage probability distribution of the target was determined. As shown in Figure 7, the target had different damage probabilities when it was near the orientations of 100° and 275°, especially when the vertical distance between the projectile and the target was 1/5 of the length of the projectile, that is, the projectile was located in the plane at 1/5 of the projectile length from the bottom of the target, and the damage probability of the target was the highest, which could reach 90%. When the warhead was detonated in other directions of the target, the damage probability of the target was almost zero. In theory, when only the kinetic energy of fragments was considered, the kinetic energy of fragments would be reduced because of the air drag in the process of flying to the target, so the probability of target damage would be further reduced if the warhead was detonated at a far distance from the target in the same orientation.

We take the implementation of the original traditional serial computing algorithm framework (Figure 3) on Intel i5-7300HQ CPU + GeForce RTX2080Ti GPU as a benchmark to evaluate the performance of the GPU-accelerated computing method. The following Table 2 lists the data in terms of running time, intersection calculation rate, and system power consumption. It can be seen that compared with i5 CPU, the speed of GPU-accelerated method has obvious advantages, and the system power consumption is lower than that of i5 CPU. The i5 CPU is more than 5 times of the GPU-accelerated computing method in power consumption, while the calculation rate of GPU-accelerated method is increased by more than 3.6 times, especially the runtime per intersection calculation process is decreased by 82.9%. GPU-accelerated method has obvious performance improvement.

Table 2 Performance comparison between serial computational processing method and GPU-accelerated computing method runs on Intel i5-7300HQ CPU + GeForce RTX2080Ti GPU

	Intel i5 CPU	*GPU-accelerated computing*
Runtime per probability calculation process (ms)	581.98	158.41
Runtime per intersection calculation process (us)	121.88	20.80
Power (W)	45.0	8.90

In conclusion, the results of simulation analysis under certain conditions show that the proposed method can locate the ideal optimal damage information to a certain extent and improve the computational efficiency, which is consistent with the purpose of the method, namely to quickly obtain the most vulnerable position of the target under different dynamic intersection conditions. From the established database information, the coverage of calculation parameters for damage assessment is relatively comprehensive. However, there is indeed a problem that the actual situation is not considered in detail. Because this paper focuses more on the effectiveness of the method in improving the computational efficiency, some equivalence is made, such as, the equivalent material and the equivalent thickness of functional damage of the components, etc. These equivalents are reasonable and consistent with the principles of damage assessment, and the calculation process of the damage probability is also a standard calculation method. Therefore, although there are some differences between the calculated results and the actual results, it can still provide some reference for the design of combat strategy. In future work, we need to further improve the method in terms of assessment accuracy.

4 Conclusions

In order to rapidly position of the most vulnerable position of the target and solve the problem of time delay in the methods of vulnerability assessment based on mathematical model, a simple target vulnerability evaluation simulation method based on multi-thread parallel computing framework is proposed. According to the designed damage information database and the sliced space position definition method, the spatial damage probability distribution information of the target can be solved timely to obtain the optimal explosion location of the warhead. The example shows that, compared with the serial calculation method, the efficiency can be improved by 3.6 times, and the most

vulnerable position of the target can be quickly provided for the ballistic end point to achieve the maximum damage probability of the target. Of course, prior database information is still indispensable in our method, and the human-computer interaction function of the whole software system needs to be further improved.

Acknowledgements

The research received support from the National Natural Science Foundation of China (Grand No. 12302471) and Natural Science Research of Jiangsu Higher Education Institutions of China (No. 1020220767 and No. 22KJD590001).

References

Chen, G., Yao, L., Wang, G., Shang, X., Chen, W., Yan, Y. and Ming, Z. (2022) 'A human-machine consensus formation method for robust decision making in battlefield situation assessment', *Acta Armamentarii*, Vol. 43, No. 11, p.2953.

Chen, J., Yuan, B.H., Xiao, C. and Chen, Y.J. (2015) 'An experimental evaluation method of energy release characteristics of reactive materials', *Chinese Journal of Explosives & Propellants*, Vol. 38, No. 3, pp.49–53.

Cho, A.H., Park, K. and Kim, G.I. (2018) 'Estimation of penetration equation parameters by comparing numerical analysis and experimental results', *Journal of Mechanical Science and Technology*, Vol. 32, No. 12, pp.5755–5765.

Defez, E., Ibáñez, J., Peinado, J., Alonso-Jordá, P. and Alonso, J.M. (2022) 'New Hermite series expansion for computing the matrix hyperbolic cosine', *Journal of Computational and Applied Mathematics*, Vol. 408, No. C, p.114084.

Deng, L., Yang, P., Liu, W. and Wang, J. (2022) 'Application in damage effect evaluation of early warning radar of cloudy Bayesian network based on Dempster Shafer/analytic hierarchy process method', *Acta Armamentarii*, Vol. 43, No. 4, p.814.

Fu, J.P., Guo, G., Feng, S., Chen, Z.G., Zhao, T.Y. and Hou, X.C. (2016) 'Damage assessment method and application of blast-fragmentation warhead against ground target', *Acta Armamentaria*, Vol. 37, No. S1, pp.7–12.

Guber, W., Nagel, R., Goldstein, R. et al. (1967) 'A geometric description technique suitable for computer analysis of both nuclear and conventional vulnerability of armored military vehicles', *Physics*, S 2624-8174, Corpus ID: 107670884 [online] https://api.semanticscholar.org/ CorpusID:107670884 (accessed 21 May 2024).

Gyllenspetz, I.M. and Zabel, P.H. (1981) 'Comparison of US and Swedish aerial target vulnerability assessment methodologies', *Försvarets forskningsanst* [online] https://api.semanticscholar.org/ CorpusID:107877327 (accessed 21 May 2024).

Hong, C., Jie, H., Yi, L. and Sen, L. (2019) 'Domain decomposition based SPH parallel computing method study and its application', *Journal of System Simulation*, Vol. 30, No. 10, pp.3717–3723.

Hou, K., Liu, W., Wang, H. and Feng, W.C. (2017) 'Fast segmented sort on GPUs', *Proceedings of the International Conference on Supercomputing*, pp.1–10.

Huang, X. and Xie, K. (2021) 'Research and implementation of a high performance distributed object-oriented simulation engine', *Journal of System Simulation*, Vol. 33, No. 9, pp.2215–2226.

Jian, L.J., Li, Y.Z. et al. (2018) 'Damage evaluation research considering fuzzy inference and Monte-Carlo method', *Aero Weaponry*, Vol. 2018, No. 6, pp.78–83, DOI: 10.19297/j.cnki.41-1228/tj.2018.06.013.

Kistler, T. and Franz, M. (2001) 'Continuous program optimization: design and evaluation', *IEEE Transactions on Computers*, Vol. 50, No. 6, pp.549–566.

Kong, X.X., Qin, W.Y. et al. (2023) 'Damage assessment algorithm based on deep learning and fuzzy analytic hierarchy process', *Acta Aeronautica et Astro nautica Sinica*, Vol. 40, No. X, pp.1–18, DOI: 10.7527/S1000-6893.2023.29503.

Li, H., Ma, G.R., Liu, Y.D., and Zhang, H.M. (2024) 'A remote sensing imagery-based model for assessment of building damage induced by large-equivalent explosions', *Explosion and Shock Waves*, Vol. 44, No. 3, p.031407.

Liu, Y., Liu, M., Lv, Z., Yan, J., Chu, S., Shi, Z. and Huang, F. (2024) 'Research on the development status and trend of ammunition damage assessment technology based on mapping knowledge domains', *Transactions of Beijing institute of Technology*, Vol. 44, No. 3, pp.219–230, DOI: 10.15918/j.tbit1001-0645.2023.100.

Ma, L., Zhang, X., Lei, X. and Bao, T. (2022) 'Design and implementation of a hybrid solver on CPU and GPU multi-target machines', *Journal of System Simulation*, Vol. 34, No. 4, pp.670–678.

Muuss, M.J., Applin, K.A., Suckling, J.R., MOSS, G., Weaver, E. and Stanley, C. (1983) *GED: An Interactive Solid Modeling System for Vulnerability Assessments*, PN, USA, DOI: 10.21236/ada126657.

Pang, S.L., Chen, X., Xu, J.S., Zhaori, G.T. and Du, H.Y. (2022) 'Analysis on damage characteristics and detonation performance of solid rocket engine charge subjected to jet', *Defence Technology*, Vol. 18, No. 9, pp.1552–1562.

Schmid, R.F., Pisani, F., Cáceres, E.N. and Borin, E. (2022) 'An evaluation of fast segmented sorting implementations on GPUs', *Parallel Computing*, Vol. 110, No. 2022, p.102889.

Shah, S. and Mehtre, B.M. (2015) 'An overview of vulnerability assessment and penetration testing techniques', *Journal of Computer Virology and Hacking Techniques*, Vol. 11, No. 2015, pp.27–49.

Shewchenko, N. (2012) 'A vulnerability/lethality model for the combat soldier, a new paradigm – basis and initial development', *Personal Armour Systems Symposium*, Nuremberg--Fürth, Germany, September.

Starks, M.W. (1990) 'Vulnerability science, a response to a criticism of the ballistic research haboratoey's vulnerability modeling strategy', *Journal: Engineering*, Vol. 1, No. AD-A224785, pp.1–24, DOI: 10.21236/ada224785,

Tekwa, K. (2023) 'Review of Wang & Sawyer. Machine learning in translation', *Babel*, Vol. 70, No. 3, pp.450–453, DOI: 10.1075/babel.00341.tek.

Thakur, N., Bharj, R.S., Kumar, P., Sharma, P., Sharma, S. and Bahl, S. (2022) 'Determination of the material model parameters for high pressure densification induced glass inferred through analytical comparison and its ballistic performance as used in composite laminates', *Physica Scripta*, Vol. 98, No. 1, p.015225.

Wang, M., Wang, X., Liu, Q., Shen, F. and Jin, J. (2020) 'A novel multi-dimensional cloud model coupled with connection numbers theory for evaluation of slope stability', *Applied Mathematical Modelling*, Vol. 77, No. 1, pp.426–438.

Wang, Y., Yin, J., Zhang, X. and Yi, J. (2022) 'Study on penetration mechanism of shaped-charge jet under dynamic conditions', *Materials*, Vol. 15, No. 20, p.7329.

Wasmund, T.L. (2001) 'New model to evaluate weapon effects and platform vulnerability: AJEM', *Wstiac Newsletter*, Vol. 2, No. 2001, pp.1–3.

Xu, Y.B., Yan, J.R. et al. (2024) 'Autonomous attack decision of bomb swarm on ground target based on visual damage assessment without communication', *Acta Armamentarii*, pp.1–14 [online] http://kns.cnki.net/kcms/detail/11.2176.tj.20240228.1319.006.html (accessed 16 June 2024).

Yamout, P., Barada, K., Jaljuli, A., Mouawad, A.E. and El Hajj, I. (2022) 'Parallel vertex cover algorithms on GPUs', *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May, pp.201–211.

Zhang C.Y, Shu, J.S. et al. (2017) 'Current status and development of evaluation technology for target damage effect', *Aerospace Technology*, Vol. 2017, No. 6, pp.68–72+77.

Zhang Z.T., Zhang, L. et al. 2021) 'Damage effectiveness evaluation of air defense weapon system based on cloud model', *Journal of Information Engineering University*, Vol. 22, No. 4, pp.497–501.

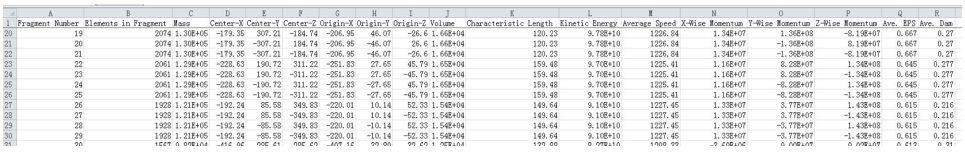
Zhang, K., Li, J., Chen, G. and Wu, B. (2011) 'GPU accelerate parallel Odd-Even merge sort: an OpenCL method', *Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp.76–83.

Zhou, K., Meng, X., Sai, R., Grubisic, D. and Mellor-Crummey, J. (2021) 'An automated tool for analysis and tuning of GPU-accelerated code in HPC applications', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 33, No. 4, pp.854–865.

Appendix A

1 Format of data

Damage field information obtained by AUTODYN simulation (part) (see online version for colours):



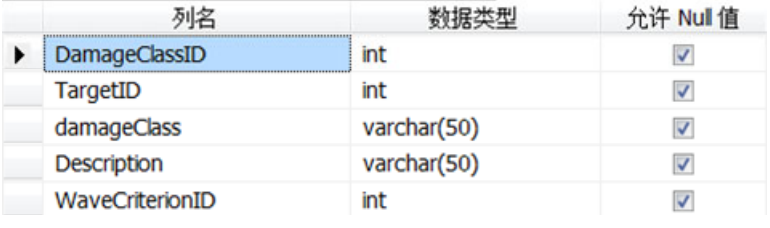
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Fragment Number	Elements in Fragment	Mass	Center-X	Center-Y	Center-Z	Origin-X	Origin-Y	Origin-Z	Volume	Characteristic Length	Kinetic Energy	Average Speed	X-Wise Momentum	Y-Wise Momentum	Z-Wise Momentum	Ave. EPS	Ave. Dam
20	19	2074	1.30E+05	-179.35	307.21	-184.74	-206.95	46.07	-26.6	1.66E+04	120.23	9.78E+10	1226.84	1.34E+07	1.36E+08	-8.19E+07	0.667	0.27
21	20	2074	1.30E+05	-179.35	-307.21	184.74	-206.95	-46.07	26.6	1.66E+04	120.23	9.78E+10	1226.84	1.34E+07	-1.36E+08	8.19E+07	0.667	0.27
22	21	2074	1.30E+05	-179.35	-307.21	-184.74	-206.95	-46.07	-26.6	1.66E+04	120.23	9.78E+10	1226.84	1.34E+07	-1.36E+08	-8.19E+07	0.667	0.27
23	22	2061	1.29E+05	-228.63	190.72	311.22	-251.83	27.65	45.79	1.65E+04	159.48	9.70E+10	1225.41	1.16E+07	8.28E+07	1.34E+08	0.645	0.277
24	23	2061	1.29E+05	-228.63	190.72	-311.22	-251.83	27.65	-45.79	1.65E+04	159.48	9.70E+10	1225.41	1.16E+07	8.28E+07	-1.34E+08	0.645	0.277
25	24	2061	1.29E+05	-228.63	-190.72	311.22	-251.83	-27.65	45.79	1.65E+04	159.48	9.70E+10	1225.41	1.16E+07	-8.28E+07	1.34E+08	0.645	0.277
26	25	2061	1.29E+05	-228.63	-190.72	-311.22	-251.83	-27.65	-45.79	1.65E+04	159.48	9.70E+10	1225.41	1.16E+07	-8.28E+07	-1.34E+08	0.645	0.277
27	26	1928	1.21E+05	-192.24	85.58	349.83	-220.01	10.14	52.33	1.54E+04	149.64	9.10E+10	1227.45	1.33E+07	3.77E+07	1.43E+08	0.615	0.216
28	27	1928	1.21E+05	-192.24	85.58	-349.83	-220.01	10.14	-52.33	1.54E+04	149.64	9.10E+10	1227.45	1.33E+07	3.77E+07	-1.43E+08	0.615	0.216
29	28	1928	1.21E+05	-192.24	-85.58	349.83	-220.01	-10.14	52.33	1.54E+04	149.64	9.10E+10	1227.45	1.33E+07	-3.77E+07	1.43E+08	0.615	0.216
30	29	1928	1.21E+05	-192.24	-85.58	-349.83	-220.01	-10.14	-52.33	1.54E+04	149.64	9.10E+10	1227.45	1.33E+07	-3.77E+07	-1.43E+08	0.615	0.216

2 The design of database

Database includes:

2.1 DamageClassTable1

The target damage level data table mainly stores all damage levels and damage level description information of the target (see online version for colours).



列名	数据类型	允许 Null 值
DamageClassID	int	☑
TargetID	int	☑
damageClass	varchar(50)	☑
Description	varchar(50)	☑
WaveCriterionID	int	☑

2.2 DamageCriterionTable1

Save the damage criteria and damage criteria of each component (see online version for colours).

列名	数据类型	允许 Null 值
DamageCriterionID	int	☐
DamageCriterion	varchar(50)	☑
LocalFile	varchar(50)	☑
ServerFile	varchar(50)	☑
Filepath	text	☑

2.3 *DamageTreeTable1*

The target damage data table mainly stores the target information, damage level, damage degree of system or component, parent node system, component ID, and the association relationship between the node and the component associated with the structure tree and each tree node (see online version for colours).

列名	数据类型	允许 Null 值
DamageTreeID	float	☐
[DamageClassID]	float	☑
Name	varchar(50)	☑
ParentName	varchar(50)	☑
NextRelation	float	☑
PartID	float	☑
PartPositionRow	float	☑
PartPositionColumn	float	☑
ParentPositionRow	float	☑
ParentPositionColumn	float	☑
AdjacentRelation	float	☑
ProbabilityPerFragment	float	☑

2.4 *EquDataTable1*

EquDataTable1 mainly stores the ownership of every part, vulnerable area proportion, Penetration equivalent ratio, equivalent material, functional damage equivalent thickness, structural dimension parameters (see online version for colours).

列名	数据类型	允许 Null 值
PartID	int	☑
EquSystemID	int	☑
EquShapeID	int	☑
PartName	varchar(50)	☑
EquMatrial	varchar(50)	☑
DamageRatio	float	☑
FuctionThickness	float	☑
EquRatio	float	☑
Parameter1	varchar(50)	☑
Parameter2	float	☑
Parameter3	float	☑
Parameter4	float	☑
Parameter5	float	☑
Parameter6	float	☑
Parameter7	float	☑
Parameter8	float	☑
Parameter9	float	☑
damageCriterionID	int	☑

2.5 EquMaterial1

Mainly save the calculation parameters of THOR formula (see online version for colours).

列名	数据类型	允许 Null 值
材料名称	varchar(50)	☑
c1	float	☑
alpha1	float	☑
beta1	float	☑
gamma1	float	☑
lamda1	float	☑
c2	float	☑
alpha2	float	☑
beta2	float	☑
gamma2	float	☑
lamda2	float	☑

```
#include <math.h>
#define pi 3.1415926535897932f
__global__ void kernel(int *c, const int *a, const int *b)
{
int j = (blockIdx.x*blockDim.x)+threadIdx.x;
int i = (blockIdx.y*blockDim.y)+threadIdx.y;
/* plane = {a[i][j+0] a[i][j+1] a[i][j+2];
a[i][j+3] a[i][j+4] a[i][j+5];
a[i][j+6] a[i][j+7] a[i][j+8];
a[i][j+9] a[i][j+10] a[i][j+11]}; */
float m[1][3],m[1][3],A[1][3],B[1][3],C[1][3],P[1][3];
float norm_n,norm_m,mn;
n = {a[i][j+0],a[i][j+1],a[i][j+2]};
norm_n = sqrt(a[i][j+0]*a[i][j+0]+a[i][j+1]*a[i][j+1]+a[i][j+2]*a[i][j+2]);
m = {b[i][j+0],b[i][j+1], b[i][j+2]};
norm_m = sqrt(b[i][j+0]*b[i][j+0]+b[i][j+1]*b[i][j+1]+b[i][j+2]*b[i][j+2]);
n[0] = n[0]/norm_n;
n[1] = n[1]/norm_n;
n[2] = n[2]/norm_n;
m[0] = m[0]/norm_m;
m[1] = m[1]/norm_m;
m[2] = m[2]/norm_m;
mn = n[0]*m[0]+n[1]*m[1]+n[2]*m[2];
A = {a[i][j+3],a[i][j+4],a[i][j+5]};
B = {a[i][j+6],a[i][j+7],a[i][j+8]};
C = {a[i][j+9],a[i][j+10],a[i][j+11]};
```

```
P = {b[i][j+3],b[i][j+4],b[i][j+5]};
int flag;
float theta,point[1][3];
if mn ! = 0
{
float phi,t,u,v;
float cross_point[1][3],uv[1][2];
phi = acos(mn);
t = (n[0]*A[0]+n[1]*A[1]+n[2]*A[2]-n[0]*P[0]+n[1]*P[1]+n[2]*P[2])/mn;
cross_point[0] = P[0]+m[0]*t;
cross_point[1] = P[1]+m[1]*t;
cross_point[2] = P[2]+m[2]*t;
uv = fcn_interCross(cross_point,A,B,C);
u = uv[0];
v = uv[1];
```