AAS 15-568

A GPU-ACCELERATED COMPUTATIONAL TOOL FOR ASTEROID DISRUPTION MODELING AND SIMULATION

Ben J. Zimmerman* and Bong Wie†

This paper presents a two-dimensional hydrodynamic simulation tool for studying the effectiveness of hypervelocity kinetic-energy impactors (KEIs) and nuclear subsurface explosions for disrupting (i.e., dispersively pulverizing) hazardous asteroids. High-order methods on GPUs (Graphics Processing Units) are employed for hydrodynamic simulations of such complex physical problems. Because high-order method schemes are compact (many operations per element), they are highly parallelized and are ideal for the architecture of GPUs. This paper focuses on the implementation of such numerical methods with GPUs as applied to the asteroid disruption problem. Three cases are compared for disrupting a reference 2D 100-m asteroid model of a nominal density of 2000 kg/m^3. They are: i) a single, 5000-kg KEI with 10-km/s impact speed, ii) five 1000-kg KEIs in parallel, and iii) a 100-kt nuclear subsurface explosion subsequent to a smaller 500-kg KEI.

INTRODUCTION

The impact threat of near-Earth objects (NEOs) is a growing concern, which was spurred by Chelyabinsk event (caused by a 17-m meteorite) in Russia on February 15, 2013 and a near miss by asteroid 2012 DA$_{14}$ ($\sim$30 m diameter) on the same day. Non-nuclear methods, such as kinetic impactors and gravity tractors, will require a substantial amount of mission lead time in order to guarantee sufficient deflection distances. While these methods are appealing, they are not applicable to short warning cases. The best option for disrupting (i.e., dispersively pulverizing) asteroids or comets with short warning times is the use of a nuclear explosive device (NED), as concluded in [1]. However, if a less amount of energy from the NED than as expected is actually coupled to the asteroid body, non-ideal fragmentation of the target NEO can occur [2]. One solution to increase the energy coupling efficiency is to employ a subsurface explosion, which can transfer a significant amount of energy to the target body.

The concept of blending a hypervelocity kinetic-energy impactor (KEI) with a subsurface nuclear explosion was developed in a NASA Innovative Advanced Concepts (NIAC) study, resulting in a Hypervelocity Asteroid Intercept Vehicle (HAIV) mission concept [3–6]. The HAIV concept uses a hypervelocity KEI to generate an initial crater and a follower vehicle carrying an NED. The follower enters the generated crater and the NED is then detonated, resulting in a subsurface nuclear explosion. The benefit of subsurface or penetrated nuclear explosions was discussed in [7]. Depending on the amount of energy emitted from the NED along with its buried depth, it can be up

*Graduate Research Assistant, Asteroid Deflection Research Center, Department of Aerospace Engineering, 1200 Howe Hall, Ames, IA 50011.

†Vance Coffman Endowed Chair Professor, Asteroid Deflection Research Center, Department of Aerospace Engineering, 1200 Howe Hall, Ames, IA 50011.

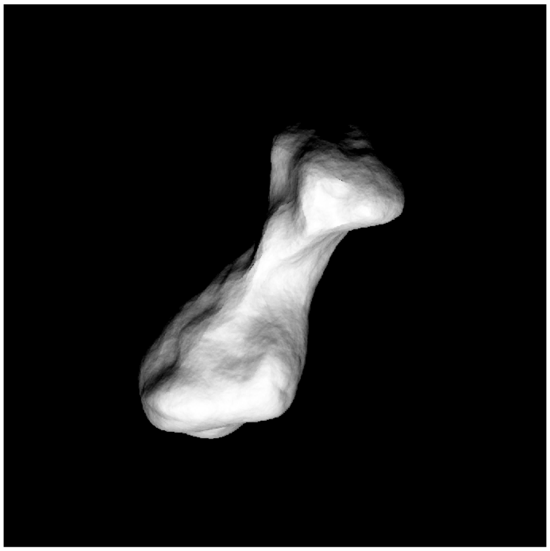


Figure 1. A computer-generated image of 216 Kleopatra asteroid.

to 35 times more efficient at coupling energy to the target body when compared to a surface contact explosion [7].

When sufficient mission lead times are not available, a kinetic impactor will be incapable of deflecting a target asteroid because the momentum transfer to the target from the impactor is not large enough to alter the target's orbital trajectory away from hitting Earth. Hence, disruption via nuclear explosives becomes an appealing solution for pulverization or vaporization of the target [1]. However, if the imparted energy from kinetic impactors is large enough, disruption of small asteroids ($<$ 150 m) using a massive kinetic-energy impactor may become feasible. In this paper, we consider a large 5000-kg KEI as well as a new Multiple Kinetic-Energy Impactor Vehicle (MKIV) proposed in [8]. The MKIV system concept stems from a concept of large 3D projectile arrays described in [9].

A multiple-HAIV system or a massive MKIV system [8] can be very effective for disrupting an asteroid shaped like 216 Kleopatra shown in Figure 1. A single massive KEI aimed at the center could split the body in two, yielding multiple fragments which can still be on an Earth-impacting trajectory. Distributed impactors striking over the surface of the body would be ideal in this situation. In this paper, three cases will be compared for disrupting a reference 2D 100-m asteroid model of a nominal density of 2000 kg/m^3. They are: i) a single, 5000-kg KEI with 10-km/s impact speed, ii) five 1000-kg KEIs in parallel, and iii) a 100-kt nuclear subsurface explosion subsequent to a smaller 500-kg KEI.

Numerical simulation of these cases requires high resolution and large computational power. To accomplish this, we apply high-order numerical methods with Graphics Processing Units (GPUs). The numerical method of choice is the the discontinuous Galerkin (DG) method [10–12], due to its high-accuracy and efficiency. Unlike popular Eulerian framework approaches, such as Finite Volume (FV) and Finite Element (FE) methods, DG allows local solution reconstruction (a solution polynomial is built within each element) whose solution at the next time step only involves information from direct neighbors. Due to this compactness, the method is highly parallelizable. NVIDIA's Compute Unified Device Architecture (CUDA) is used to hasten the solution generation, and arrive at the final required time by orders of magnitude faster than CPU simulations. The application of high-order methods with GPU CUDA has already been investigated in [13, 14], where significant increases in computing speed has been observed. Since our research group does not have access to a large computational CPU cluster, CUDA is of high interest. All simulations are partitioned and run across four NVIDIA Tesla K20 GPUs to enhance computing speeds.

Numerical modeling of the hypervelocity kinetic-energy impact, followed by subsurface nuclear explosion is conducted with the two-dimensional Euler equations. Upon kinetic-energy impact, the pressure imparted by the impactor is much greater than the target materials strength, hence the material behaves like a compressible fluid [15] and the Euler equations can be used. The equations are solved in an Eulerian framework, and the stiffened equation of state is implemented for the impactor and asteroid target. It is known that this equation of state is not sufficient to characterize the asteroids material properties completely. More accurate equations of states are currently being investigated and implemented into the model.

This paper is organized in the following manner. Section 1 describes the mathematical modeling used for simulations, including the governing equations, equation of state, and numerical method. Section 2 outlines the implementation into the GPU architecture and presents speed-up results. Section 3 presents and discusses the results of three cases.

NUMERICAL MODELING

This section outlines the mathematical model used in the hydrodynamic simulations. The governing equations and numerical discretization are outlined. Additionally, two one-dimensional problems are simulated to show proper implementation of the model.

Governing Equations

As mentioned previously, the velocity is extremely high in hypervelocity problems, such that most materials behave similar to fluids. The two-dimensional Euler equations are chosen for numerical modeling, and are governed by conservation of mass, momentum, and energy, as follows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} = 0 \tag{1}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \left(\rho u^2 + p\right)}{\partial x} + \frac{\partial (\rho u v)}{\partial y} = 0 \tag{2}$$

$$\frac{\partial \rho v}{\partial t} + \frac{\partial (\rho u v)}{\partial x} + \frac{\partial \left(\rho v^2 + p\right)}{\partial y} = 0 \tag{3}$$

$$\frac{\partial e}{\partial t} + \frac{\partial \left[u(e+p)\right]}{\partial x} + \frac{\partial \left[v(e+p)\right]}{\partial y} = 0 \tag{4}$$

where ρ is the density, u is the x-direction velocity, v is the y-direction velocity, p is the pressure, and e is the total energy per unit volume. To close the system, an appropriate equation of state (EOS) must be defined. For this study, a stiffened EOS is considered as

$$p = \rho e(\gamma - 1) - \gamma p_o \tag{5}$$

where γ is the ratio of specific heats and p_o is a pressure constant. Both γ and p_o depend on the material to be modeled, and can be found via experiments and empirical curve fitting [16]. The speed of sound in the stiffened EOS model is given as follows:

$$c = \sqrt{\frac{\gamma(p + p_o)}{\rho}} \tag{6}$$

If p_o is set to 0, the stiffened equation of state reduces to the ideal gas equation of state. However, simply defining a new equation of state is not sufficient. Our solver uses an Eulerian framework, and we must track the γ and p_o variables, the states along material interfaces. Similar work for the stiffened EOS was completed in [17]. The equation of state is integrated into the state variables, further complicating the method and increasing the computational cost of the approach. Two additional state equations are added to the governing model. One equation tracks only the γ term as

$$\frac{\partial}{\partial t}\left(\frac{1}{\gamma - 1}\right) + \frac{\partial}{\partial x}\left(\frac{u}{\gamma - 1}\right) + \frac{\partial}{\partial y}\left(\frac{v}{\gamma - 1}\right) = 0 \tag{7}$$

And the other equation tracks the p_o term as

$$\frac{\partial}{\partial t}\left(\frac{\gamma p_o}{\gamma - 1}\right) + \frac{\partial}{\partial x}\left(\frac{u\gamma p_o}{\gamma - 1}\right) + \frac{\partial}{\partial y}\left(\frac{v\gamma p_o}{\gamma - 1}\right) = 0 \tag{8}$$

The governing equations can be written in the form of a hyperbolic equation as

$$\frac{\partial \boldsymbol{q}}{\partial t} + \vec{\nabla} \cdot \vec{\boldsymbol{F}}(\boldsymbol{q}) = 0 \tag{9}$$

where $\vec{\nabla} \cdot \vec{\boldsymbol{F}}(\boldsymbol{q}) = \dfrac{\partial \boldsymbol{f}(\boldsymbol{q})}{\partial x} + \dfrac{\partial \boldsymbol{g}(\boldsymbol{q})}{\partial y}$. where $\boldsymbol{q}$ is the state variable vector and $\boldsymbol{f}$ and $\boldsymbol{g}$ are flux vectors defined as

$$\boldsymbol{q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \\ \dfrac{1}{\gamma - 1} \\ \dfrac{\gamma p_o}{\gamma - 1} \end{bmatrix}, \qquad \boldsymbol{f}(\boldsymbol{q}) = \begin{bmatrix} \rho u \\ p + \rho u^2 \\ \rho uv \\ u(e + p) \\ \dfrac{u}{\gamma - 1} \\ \dfrac{u\gamma p_o}{\gamma - 1} \end{bmatrix}, \qquad \boldsymbol{g}(\boldsymbol{q}) = \begin{bmatrix} \rho v \\ \rho uv \\ p + \rho v^2 \\ v(e + p) \\ \dfrac{u}{\gamma - 1} \\ \dfrac{v\gamma p_o}{\gamma - 1} \end{bmatrix} \tag{10}$$

Now, the solver can track the interfaces of different fluids/materials throughout the domain. The above formulation is known to generate small errors in the pressure field, as discussed in [18, 19]. A new methodology is currently being implemented to remove these errors, and will be presented in future work. With the governing equations defined, we must discretize the equations with an appropriate method. The discontinuous Galerkin method is chosen for implementation, and is briefly discussed next.

Numerical Method

The computational domain is discretized with non-overlapping quadrilateral elements. Each element has index i and volume V_i. The elements are transformed into standard elements [20], where a set of solution points and flux points are defined. Figure 2 illustrates the locations of solution points

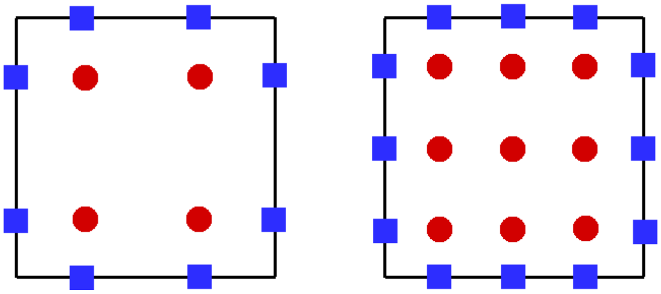


Figure 2. Solution points (circles) and interpolation points (squares) for P^1 and P^2 DG.

and flux points in a quadrilateral element. The solution points contain the state variable information, while the flux points provide coupling across elements. The number of solution points within an element can be increased, providing a higher order interpolation polynomial and improving accuracy of the method. The figure demonstrates a second order scheme (left element) and a third order scheme (right element). Every element in the domain is discretized with the same number of points, maintaining a constant order of accuracy.

Equation (9) is integrated over an element's volume and multiplied by an arbitrary weighting function w, as follows:

$$\int_{V_i} \left[\frac{\partial \boldsymbol{q}}{\partial t} + \vec{\nabla} \cdot \vec{\boldsymbol{F}}(\boldsymbol{q}) \right] w dV = 0 \tag{11}$$

By integrating by parts, approximating $\boldsymbol{q}$ on each element by $\boldsymbol{q}_i$, and using polynomials P^k, we rewrite Equation (11) as

$$\int_{V_i} \frac{\partial \boldsymbol{q}_i}{\partial t} w dV + \int_{\partial V_i} w \vec{\boldsymbol{F}}(\boldsymbol{q}_i) \cdot \boldsymbol{n} dS - \int_{V_i} \vec{\nabla} w \cdot \vec{\boldsymbol{F}}(\boldsymbol{q}_i) dV = 0 \tag{12}$$

The solution and flux polynomial is approximated over the $k+1$ Gauss-Legendre points in each element, as follows:

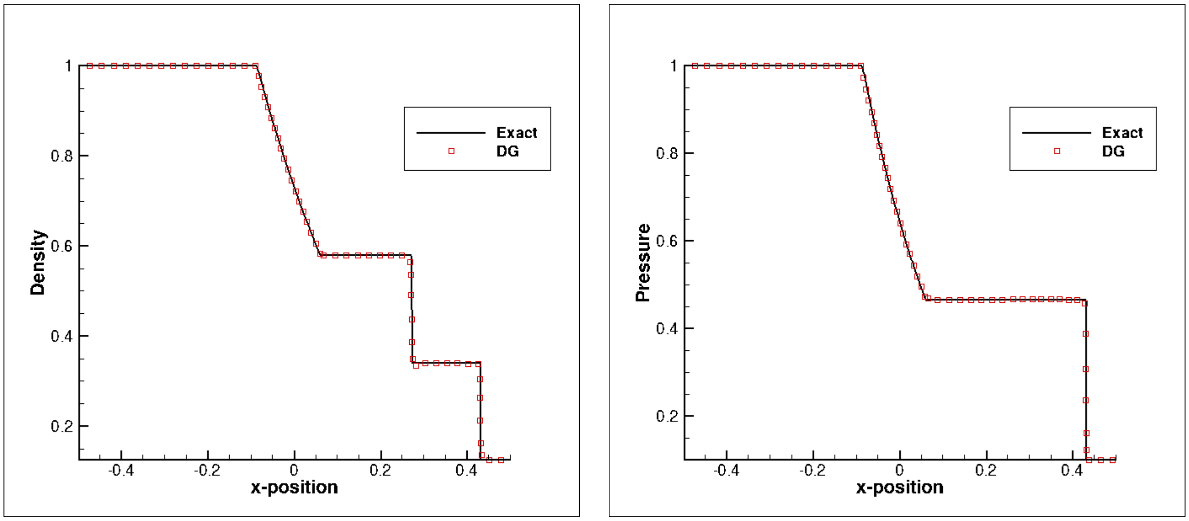
$$\boldsymbol{q}_i = \sum_{j=1}^{k+1} \boldsymbol{q}_{i,j} \phi_j, \quad \vec{\boldsymbol{F}}(\boldsymbol{q}_{i,j}) = \sum_{j=1}^{k+1} \vec{\boldsymbol{F}}_{i,j} \phi_j \tag{13}$$

In Equation (13), ϕ_j is the basis function at solution point j. If the basis and weighting function are the same, then the process is Galerkin. Both the basis and weighting functions are chosen as Lagrange polynomials in this paper. To provide element coupling flux points are set on element interfaces, and a common Riemann flux is calculated from left and right solution states. In this paper, only the Rusanov interface flux [21] is considered.

Numerical approximations near shocks or solution discontinuities can produce oscillations and non-physical results for any approximation greater than 1^{st} order. This is better known as Gibbs phenomenon. To deal with this numerical issue, slope limiting is implemented with the DG method [12]. Inside an element i, we define an approximate solution $\boldsymbol{q}_i^h$ which can overwrite the current solution if a discontinuity exists. This approximate solution is computed as follows:

$$\boldsymbol{q}_{i,j}^h = \bar{\boldsymbol{q}}_i + \boldsymbol{q}_{i,j}^x \alpha(x) + \boldsymbol{q}_{i,j}^y \beta(y) \tag{14}$$

where $\bar{\boldsymbol{q}}_i$ is the solution average in element i and α and β are orthogonal basis functions. See the paper by Cockburn and Shu [12] for more information. The functions $\boldsymbol{q}_{i,j}^x$ and $\boldsymbol{q}_{i,j}^y$ are limited



(a) Density field

(b) Pressure field

Figure 3. One-dimensional shock tube ($t = 0.2$)**.**

functions of the solution computed via the difference of means. For example, $\boldsymbol{q}^x_{i,j}$ is computed as follows:

$$\boldsymbol{q}^x_{i,j} = minmod(\overline{\boldsymbol{q}}_{i+1} - \boldsymbol{q}^+_{i,j}, \boldsymbol{q}^-_{i,j} - \overline{\boldsymbol{q}}_{i-1}) \tag{15}$$

where $\boldsymbol{q}^+_{i,j}$ and $\boldsymbol{q}^-_{i,j}$ are the solution values at the left and right interface points respectfully. The *minmod* function is defined as

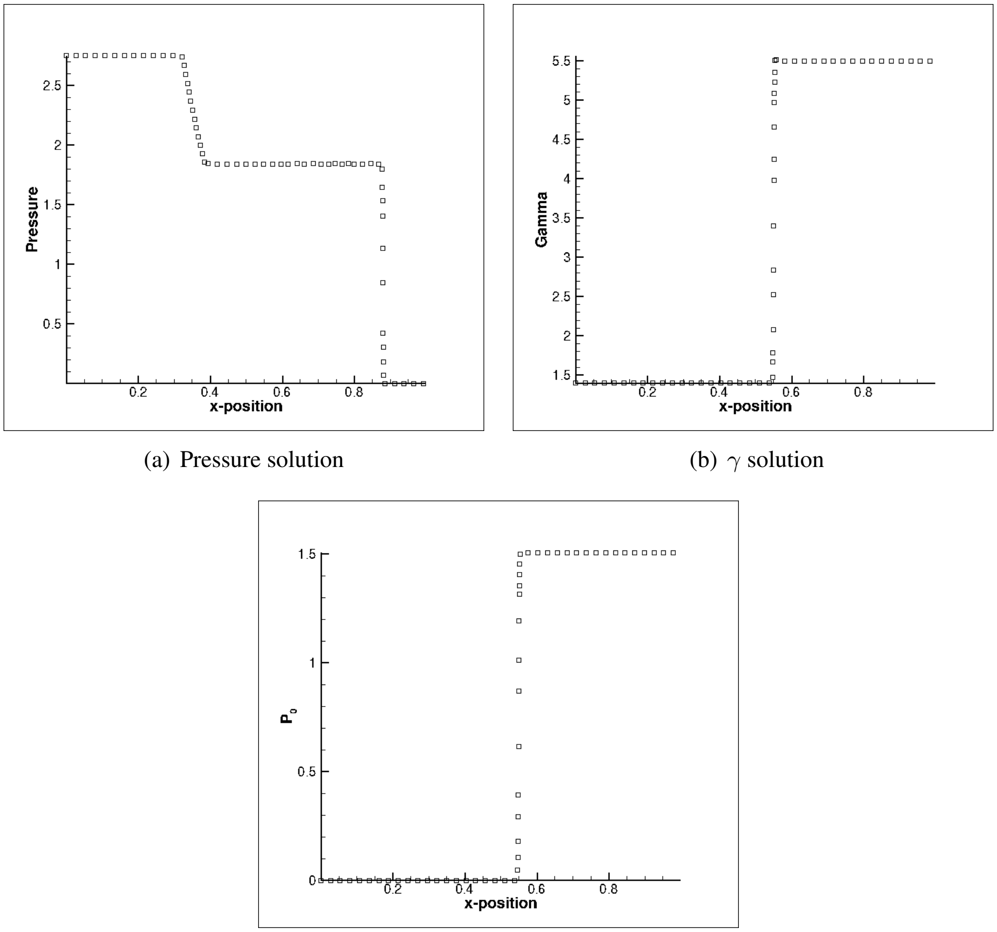
$$mimod(a,b) = \begin{cases} s \cdot \text{min}(a,b), & \text{if } s = \text{sign}(a) = \text{sign}(b) \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

The TVB corrected *minmod* functions is employed in our formulation [12, 22, 23]. This allows the method to first detect areas of discontinuities and limit the solution only within these areas. Thus, smooth portions of the solution remain unchanged.

1D Test Cases

We present two example problems for the model in one-dimension. The first test case was proposed in Toro [24], where two regions of gas with differing densities, pressures, and velocities are separated by a membrane, which is removed at the beginning of the simulation, The left high-pressure region has initial conditions $(\rho, u, p, \gamma, p_0) = (1.0, 0.75, 1.0, 1.4, 0.0)$, while the right low-pressure region has the conditions $(\rho, u, p, \gamma, p_0) = (0.125, 0.0, 0.1, 1.4, 0.0)$. All the flow variables are in non-dimensional form. We use 400 elements with a P^2 reconstruction (3 points per element), which gives 1200 degrees of freedom. A three-stage Runge-Kutta method [25] is used for time integration. Figure 3 shows the density and pressure fields of the DG method and the exact solution at a non-dimensional time of $t = 0.2$. The rarefaction wave, contact discontinuity, and shock wave are all captured, and compare well with the exact solution.

We simulate a second test problem, one that contains differing specific heat ratios, γ, and pressure coefficients, p_0. Again, a membrane separates the two gases at location $x = 0.5$. Left of this separation, the initial state is given as $(\rho, u, e, \gamma, p_o) = (1.241, 0, 2.753, 1.4, 0)$, while to the right,



(a) Pressure solution

(b) γ solution

(c) P_0 solution

Figure 4. One-dimensional interface problem ($t = 0.1$).

the initial conditions are $(\rho, u, e, \gamma, p_o) = (0.991, 0, 3.059 \times 10^{-4}, 5.5, 1.505)$. The number of elements in the domain is increased to 800, and a P^2 reconstruction is employed per element. A three-stage Runge-Kutta method is used for time integration. The purpose of this case is to ensure that the method can evaluate the flow field variables while tracking the interface of the two gases. Figure 4 demonstrates the pressure solution along with γ and p_0 solutions at a time of $t = 0.1$. The results presented here match well with results from [17]. The interface is clearly captured from the γ and p_0 solution profiles. Small oscillations are apparent in the pressure solution which was discussed previously. Future work aims at completely removing these oscillations.

GPU APPLICATION

This section outlines the implementation of the DG approach with the added equation of state on GPUs. Due to the compactness of the DG method, it is efficient for application to GPU CUDA computing. Most of the computational work is completed locally, per element. This allows for a high level of parallelism, since GPU threads can work independently from one another. Two GPU kernels will be discussed, **gpu_couple** and **gpu_rhs** to demonstrate how the elements are coupled

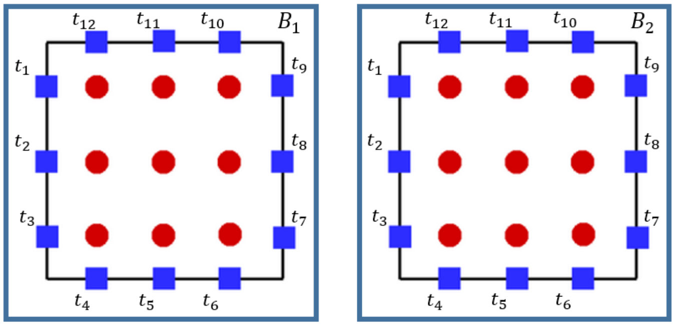


Figure 5. Flux point thread numbering for P^2 DG.

Algorithm 1 gpu_couple

```
  ▷ Each point on my face
  $i$ = threadIdx.x
  ▷ Current cell in block
  $tmpk$ = threadIdx.y
  ▷ Current global cell
  $k = \text{blockIdx}.x * \text{blockDim}.y + tmpk$
  **if** $k < \text{n}_c$ **then**
      ▷ Read the left solution state
      $q_L^l[(0...5)] = q_L^t[i + \text{n}_{fp} * ((0...5) + \text{n}_v * k)]$
      ▷ Read the right solution state
      $q_R^l[(0...5)] = q_R^t[i + \text{n}_{fp} * ((0...5) + \text{n}_v * k)]$
      ▷ Compute the interface flux
      $\text{InterfaceFlux}(q_L^l, q_R^l, f_n^l)$
      ▷ Store interface flux into global memory
      $f_n^g[i + \text{n}_{fp} * ((0...5) + \text{n}_v * k)] = f_n^l[(0...5)]$
  **end if**
```

and the right hand side is computed. The kernel **gpu_couple** is set-up in the following manner. Along each face, we define the threads as the flux points as shown in Figure 5. These threads along the faces will run in parallel, since the computations at these points don't depend on any other flux points within the element. As noted in the figure, each block contains one element. This can be extended to utilize more threads within each block by adding additional elements per block. Before the algorithm is presented, take note that all information is stored in one-dimensional arrays, and the superscripts l, s, t, and g denote local, shared, texture, and global memory respectfully. Algorithm 1 demonstrates the parallel CUDA code. The index i represents each point on an elements face, $tmpk$ is the current cell within one block, and k is the global cell identifier. Assume that we have interpolated the solution to the flux points on the element edges, and the left and right information are stored within textured memory located in q_L^t and q_R^t respectfully. The terms n_c, n_v, and n_{fp} indicate the number of cells (or elements), number of states, and number of flux points per element. The dimension of the blocks is $\vec{t} = [t_x, t_y] = [\text{n}_{fp}, \text{cpb}]$, where cpb $=$ cells per block. The dimension of the grid must then be $\vec{b} = [b_x] = [\text{n}_c/\text{cpb}]$. For example, in Figure 5, assume that only two elements reside in the domain. Then $\vec{t} = [12, 1]$ and $\vec{b} = [2]$. Of course, in actual simulations, the number of threads and blocks are drastically increased. Each thread updates all the solution

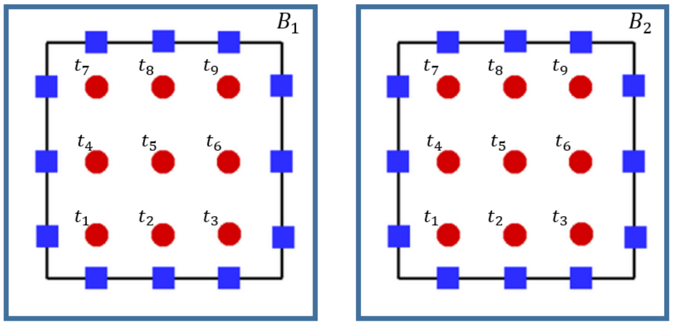


Figure 6. Solution point thread numbering for P^2 DG.

Table 1. GPU speeds

Reconstruction	4 × Intel Xeon	4 × Tesla K20	Speed-Up
P^1	9.400	0.258	36.4
P^2	16.420	0.711	23.2
P^3	37.856	1.489	23.1

states at one point within an element. The result is written into global memory to be used in later kernels.

Next, consider the **gpu_rhs** kernel, which computes the flux derivative and prepares the solution to be updated towards the next time step. The thread and block assignment is similar to that shown in Figure 6. Each thread will compute the flux derivative for each solution state at the corresponding solution point. The term n_{sp} is the number of solution points per element, while F and G denote the functions for the flux (see equation 10). In addition, the terms c_x^t, c_y^t, and c_{sur}^t indicate the x and y derivative coefficients for the volume integration and the coefficient for the surface integration, respectfully. The solution state is read from texture memory into local memory. This local memory is used to compute the flux functions, which are stored into shared memory. Shared memory is used because each thread will require the flux information at other points within the block. Once the shared memory is written, CUDA requires the threads to synchronize, which ensures all data is written into the shared memory before continuing. The shared memory is used to preform the volume integration, which is written into local memory. Next, surface integration is completed by reading the appropriate flux information computed from **gpu_couple**. After the volume and surface integrations are completed, the solution is stored into the global rhs^g array. The next solution state can then be computed using a three-stage Runge-Kutta time-stepping scheme [25].

We demonstrate our solver's current speeds in Table 1. The CPU used was an Intel Xeon E5-2640 clocked at 2.5 GHz. For the GPU, we used a Tesla K20 card. A domain with 40,000 elements was given and initial condition, and the speed at 100 iterations was recorded. Table 1 shows the speed per iteration in seconds along with the speed-up factor of using GPUs. We observe 23 to 37 times faster computation times using four Tesla cards when compared to four Intel CPUs. Meaning, in order to meet the computing speed of the four Tesla cards for our solver, one would require 90-150 processors, or a small computing cluster.

Algorithm 2 gpu_rhs

```
▷ Each point in my cell
$i$ = threadIdx.x
▷ Current cell in block
$tmpk$ = threadIdx.y
▷ Current global cell
$k = \text{blockIdx}.x * \text{blockDim}.y + tmpk$
**if** $k < \text{n}_c$ **then**
    ▷ Read the solution state
    $q^l[(0...5)] = q^t[i + \text{n}_{sp} * ((0...5) + \text{n}_v * k)]$
    ▷ Compute the flux into shared memory
    $f^s[(0...5) + \text{n}_v * (i + \text{n}_{sp} * tmpk)] = \text{F}(q^l[(0...5)])$
    $g^s[(0...5) + \text{n}_v * (i + \text{n}_{sp} * tmpk)] = \text{G}(q^l[(0...5)])$
    ▷ Ensure everything is loaded
    __syncthreads
    ▷ Volume integration
    **for** $mm = 0$ to $(\text{n}_{sp} - 1)$ **do**
        $id = (0...5) + \text{n}_v * (mm + \text{n}_{sp} * tmpk)$
        $Vol^l[(0...5)] = Vol^l[(0...5)] + c^t_x[mm + i * \text{n}_{sp}] * f^s[id] + c^t_y[mm + i * \text{n}_{sp}] * g^s[id]$
    **end for**
    ▷ Surface integration
    **for** $mm = 0$ to $(\text{n}_{fp} - 1)$ **do**
        $id = mm + \text{n}_{fp} * ((0...5) + \text{n}_v * k)$
        $Sur^l[(0...5)] = Sur^l[(0...5)] - c^t_{sur}[mm + \text{n}_{fp} * (i + \text{n}_{sp} * k)] * f^t_n[id]$
    **end for**
    ▷ Update the right hand side
    $rhs^g[i + \text{n}_{sp} * ((0...5) + \text{n}_v * k)] = Vol^l[(0...5)] + Sur^l[(0...5)]$
**end if**
```

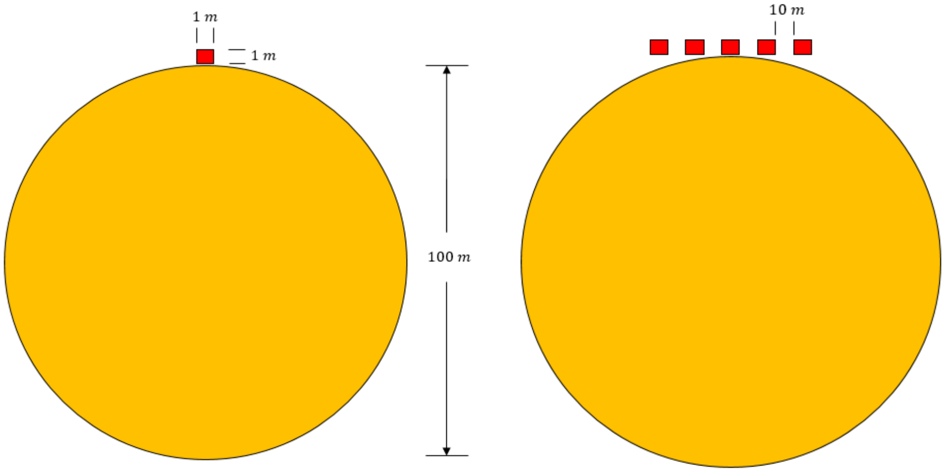
PROBLEM DESCRIPTION AND RESULTS

In this section, we introduce the problem and discuss the results obtained. The target body is a 2D circular 100-m diameter asteroid, while the impactor is modeled as a 1×1 m box. There are no porosity effects or varying density with the target asteroid or impactor. The asteroid body is modeled as granite, while the impactor is modeled as aluminum. The constants for γ and p_o are given in Table 2 and taken from [26,27]. The density is also shown in this table, but for the impactor, it varies from case to case to be presented later. All impactors shown in Figure 7 are traveling at 10 km/s. In all simulations, a P^1 reconstruction is employed, with a third-order Runge-Kutta time-stepping scheme. Each simulation contains 360,000 elements, which implies 1.44 million degrees of freedom per equation with the aforementioned reconstruction.

The density results for the two cases (a single 5000-kg KEI and five KEIs in parallel) are shown in Figure 8. From the density contours, it can be seen that the single-impactor case has a high-dense region in the asteroid's center. There are low-dense regions around the outside of the body, indicating damage from spall. An ideal collision of a 5000-kg KEI traveling at 10 km/s with a 2D 100-m (diameter) circular model of an asteroid with a nominal density of 2000 kg/m^3 will cause a 3.18-m/s ΔV of the center-of-mass of the target body. From our simulation results, the lowest

Table 2. Material parameters

Body	ρ (kg/m^3)	γ	p_o (Pa)
Outside	0.1	1.4	0
Impactor	-	3.8	1.4E8
Asteroid	2000.0	2.6	1.42E10



(a) Single kinetic-energy impactor

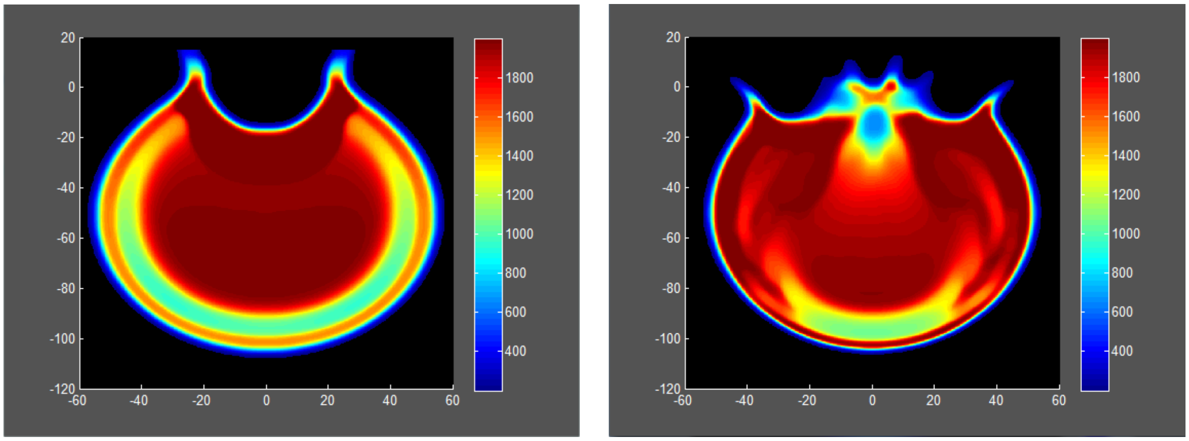
(b) Multiple kinetic-energy impactors

Figure 7. Illustration of ideal 2D hydrodynamic simulation models.

velocity magnitude was measured to be 3.25 m/s, which is in good agreement with the estimated ΔV. The results from the case of five KEIs show three main regions of high density. One near the asteroid center, which is much smaller than the single-impactor case, and two others near the crater openings. The velocity magnitude contours, shown in Figure 9, illustrate that the high-dense region near the center of the single-impactor case is indeed moving slowly. However, only one slow-moving region exists in the multiple-impactor case, and it is smaller in size than the single-impactor case.

A very interesting result can be seen in the velocity histogram plots shown in Figure 10. For both plots, the x-axis shows the velocity magnitude values in units of m/s, while the y-axis represents the percentage of particles at a specific velocity. It is observed that the single-impactor case has a high amount of particles (roughly 27%) moving very slowly. These particles are represented by the high-density region as shown in Figure 8. The multiple-KEI histogram plot shows roughly 7% of particles moving very slow, and most particles appear to be in the 0-50 m/s region.

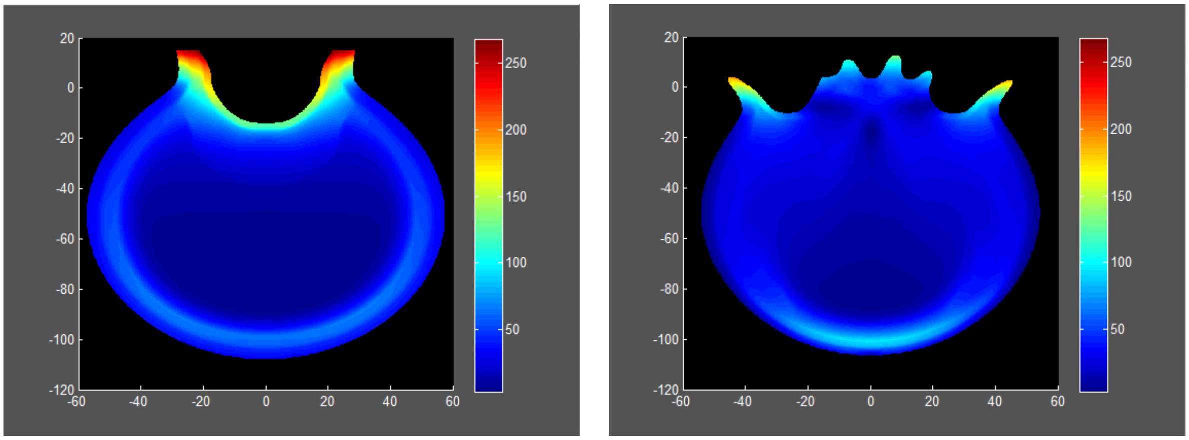
We now consider an example of HAIV mission of employing a 500-kg impactor and a 100-kt nuclear device. The 500-kg impactor first strikes the asteroid body, generating a crater. At 1 millisecond after the impactor has hit, the 100-kt device was inserted into the crater and detonated. The asteroid is still governed by the stiffened EOS; however, the nuclear device is modeled using the ideal gas EOS, with a constant specific heat ratio of $\gamma = 1.4$. It is initialized as a small circle (radius $r < 1.0$ m) within the generated crater. Energy equivalent to 100-kt is initialized inside the device radius; i.e., no energy loss from the nuclear device is assumed. Density contours for the target body are shown in Figure 11. The elapsed time in the figures is the time after the nuclear



(a) Single kinetic-energy impactor

(b) Multiple kinetic-energy impactors

Figure 8. Density contours at 0.23 seconds (density in units of kg/m^3).



(a) Single kinetic-energy impactor

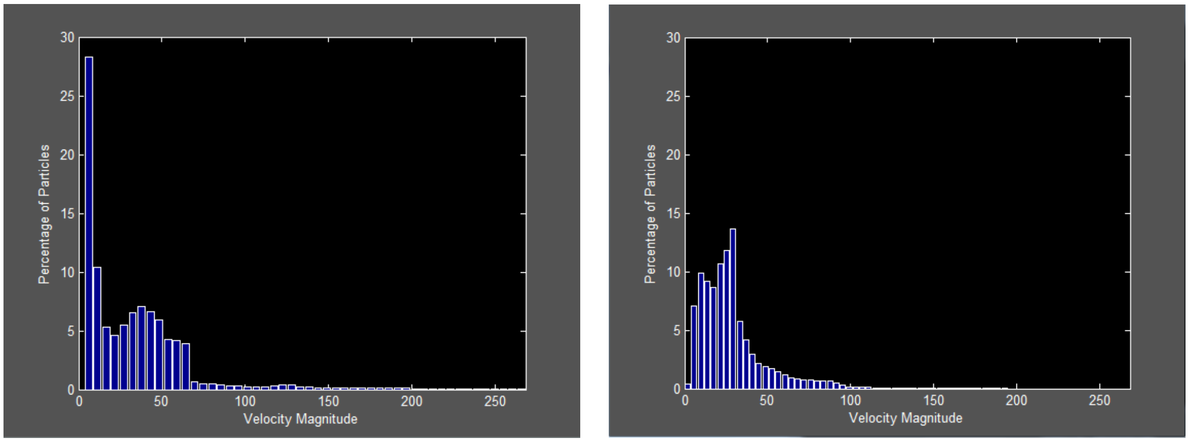
(b) Multiple kinetic-energy impactors

Figure 9. Velocity magnitude contours at 0.23 seconds (velocity in units of m/s).

device has detonated. It is easily observed that high damage occurs to the target body throughout the simulation.

The target body becomes highly deformed/damaged while the large energy shock travels through. The velocity histogram at the final time is shown in Figure 12. While there are some particles not yet moving (since the shock has not yet affected them), the most other particles have extremely high velocities (from 5-30 km/s). The HAIV concept appears to fully disrupt (i.e., dispersively pulverize) such small asteroid body in this simulation. However, note that such very large dispersal speeds here are due to an ideal 100-kt nuclear explosion energy directly coupled to an ideal 2D circular, 100-m hydrodynamic simulation model.

A further study using a more realistic 3D asteroid model as well as a more rigorous model of nuclear explosions [28,29] is necessary to evaluate the practical effectiveness of the various options for dispersively pulverizing small asteroids.



(a) Single kinetic-energy impactor

(b) Multiple kinetic-energy impactors

Figure 10. Velocity histogram at 0.23 seconds (velocity in units of m/s).

CONCLUSIONS

From the asteroid disruption simulations, both single-impactor and multiple-impactor cases have regions of high density within the target body. However, the regions in the multiple-KEI case are smaller in comparison. In addition, the single-impactor case has more slow moving particles than what was observed in the multiple-impactor case. More simulations are required in order to fully compare these two disruption techniques. The HAIV concept simulations show extremely high damage and large dispersal velocity profiles, clearly demonstrating the power of nuclear explosive devices, in comparison with a non-nuclear kinetic-energy impact approach. Our modeling is not yet complete. A new methodology is under development to remove the pressure oscillations observed in the one-dimensional test case. Once this oscillation issue is resolved, a better equation of state that better characterizes both the target body and kinetic impactor can be implemented in our GPU-accelerated hydrodynamic simulations.

REFERENCES

[1] *Defending Planet Earth: Near-Earth Object Surveys and Hazard Mitigation Strategies*, National Research Council, 2010.

[2] J. Sanchez, M. Vasile, and G. Radice, “On the Consequences of a Fragmentation due to a NEO Mitigation Strategy,” IAC-08-C1.3.10, *59th International Astronautical Congress*, 2008.

[3] B. Wie, “Hypervelocity Nuclear Interceptors for Asteroid Disruption,” *Acta Astronautica*, 90, 2013, pp. 146–155.

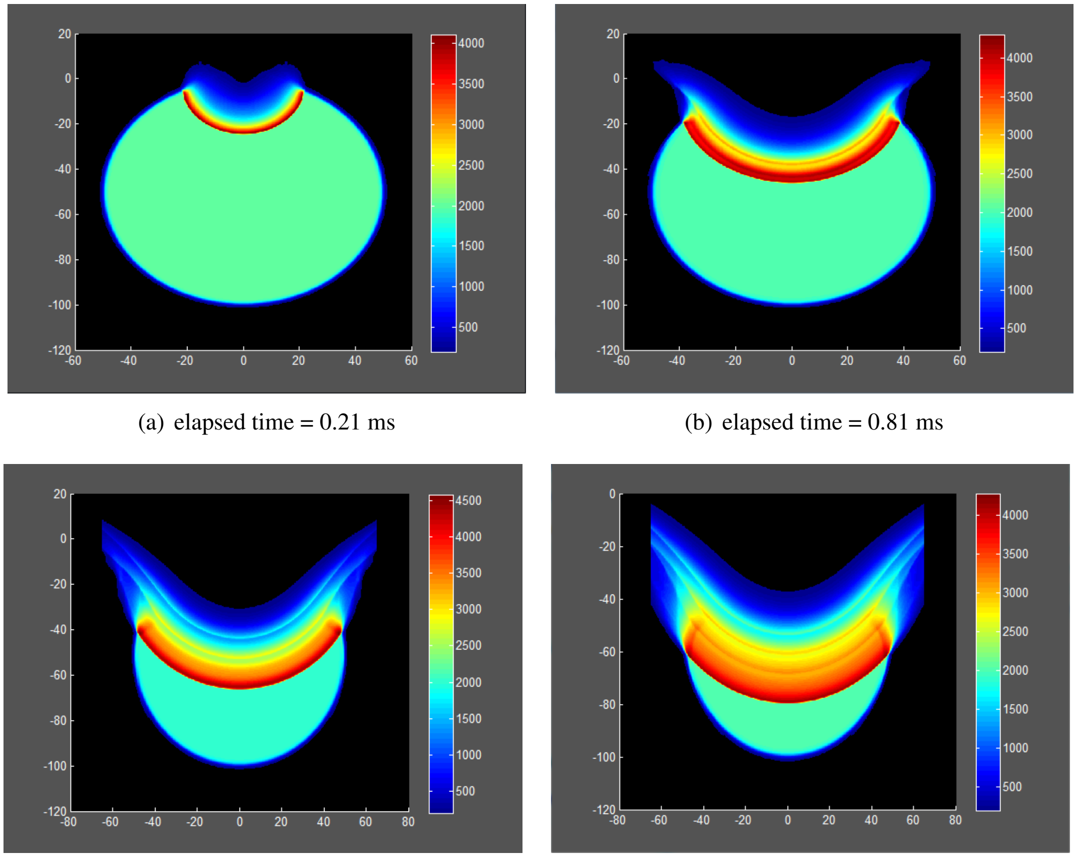
[4] A. Pitz, B. Kaplinger, G. Vardaxis, T. Winkler, and B. Wie, “Conceptual Design of a Hypervelocity Asteroid Intercept Vehicle (HAIV) and Its Flight Validation Mission,” *Acta Astronautica*, 94, 2014, pp. 42–56.

[5] B. Barbee, B. Wie, M. Steiner, and K. Getzandanner. “Conceptual Design of a Flight Demonstration Mission for a Hypervelocity Asteroid Intercept Vehicle,” *Acta Astronautica*, 106, 2015, pp. 139–159.

[6] B. Kaplinger, P. D. Premaratne, C. Setzer, and B. Wie, “GPU-Accelerated 3D Modeling and Simulation of a Blended Kinetic Impact and Nuclear Subsurface Explosion,” AIAA-2013-4548, 2013.

[7] *Effects of Nuclear Earth-Penetrator and Other Weapons,* National Academy of Sciences, 2005.

[8] B. Wie, B., Zimmerman, and P. Premaratne, “A Non-Nuclear MKIV (Multiple Kinetic-Energy Impactor Vehicle) System for Dispersive Pulverization of Small Asteroids with Short Warning Times, ” AAS 15-567, *AAS/AIAA Astrodynamics Specialist Conference*, August 10-13, 2015.



(a) elapsed time = 0.21 ms

(b) elapsed time = 0.81 ms

(c) elapsed time = 1.61 ms

(d) elapsed time = 2.21 ms

Figure 11. 2D simulation result for a small HAIV system of blending a 500-kg KEI and a 100-kt subsurface nuclear explosion. Density in units of kg/m^3.

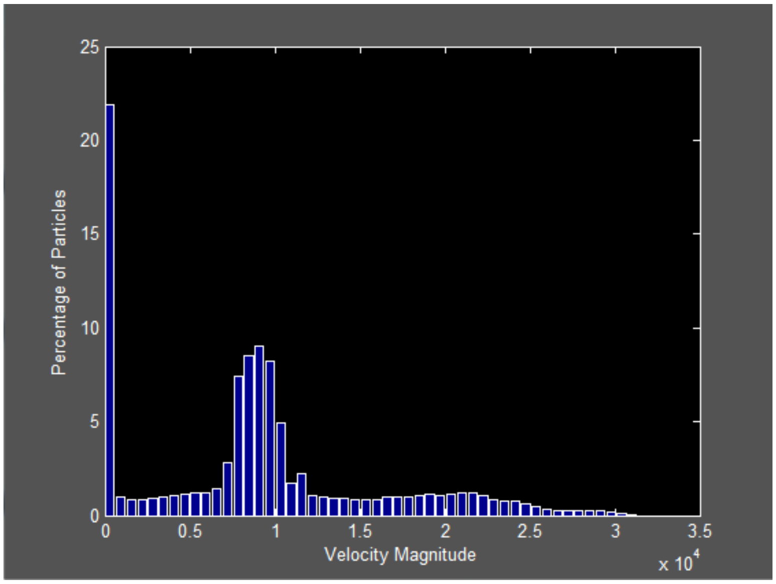


Figure 12. Velocity histogram of a HAIV mission example (velocity in units of m/s).

[9] L. Wood, R. Hyde, M. Ishikawa, and E. Teller, "Cosmic Bombardment V: Threat Object-Dispersing Approaches to Active Planetary Defense," *Proceedings of the Planetary Defense Workshop*, Lawrence Livermore National Laboratory, Livermore, CA, May 22-26, 1995.

[10] F. Bassi and S. Rebay, "High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations," *J. Computational Physics*, 138, 1997, pp. 251–285.

[11] C. Baumann and T.J. Oden, "A Discontinuous hp Finite Element Method for the Euler and Navier-Stokes Equations," *J. Numerical Methods in Fluids*, 31 (1), 1999, pp. 79–95.

[12] B. Cockburn and C.W. Shu, "The Runge-Kutta Discontinuous Galerkin Method for Conservation Laws V: Multidimensional Systems," *J. Computational Physics*, 141, 1998, pp. 199–224.

[13] M. Hoffmann, C.D. Munz, and Z. J. Wang, "Efficient Implementation of the CPR Formulation for the Navier-Stokes Equations on GPUs," *ICCFD*, 7-2603, 2012.

[14] B. J. Zimmerman and Z. J. Wang, "The Efficient Implementation of Correction Procedure via Reconstruction with GPU Computing," AIAA-2013-2692, 2013.

[15] R. F. Prater, "Hypervelocity Impact - Material Strength Effects of Crater Formation and Shock Propagation in Three Aluminum Alloys," *PhD Thesis*, AFML-TR-70-295, 1970.

[16] K. S. Holian, "T-4 Handbook of Material Properties Data Bases," Techical Report LA-10160-MS, Los Alamos National Laboratory, 1984.

[17] K. Shyue, "An Efficient Shock-Capturing Algorithm for Compressible Multicomponent Problems," *J. Computational Physics*, 142, 1998, pp. 208–242.

[18] S. Karni, "Multicomponent Flow Calculations by a Consistent Primitive Algorithm," *J. Computational Physics*, 112, 1994, pp. 31–43.

[19] R. Abgrall, "How to Prevent Pressure Oscillations in Multicomponent Flow Calculations: A Quasi Conservative Approach," *J. Computational Physics*, 125, 1996, pp. 150–160.

[20] O. C. Zienkiewicz and R. C. Tayler, *The Finite Element Method: the Basics*, 2000.

[21] V. V. Rusanov, "Calculation of Interaction of Non-Steady Shock Waves with Obstacles," *Journal of Computational Mathematical Physics*, USSR(1), 1961.

[22] C. W. Shu, "TVB Uniformly High-Order Sschemes for Conservation Laws," *Mathematical Computation*, 49, 1987.

[23] B. Cockburn and C. W. Shu, "TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Scalar Conservation Laws II: General Framework," *Mathematical Computation*, 52, 1989.

[24] E.F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*, Vol. 3, 2009.

[25] C. W. Shu, "Total-Variation-Diminishing Time Discretizations," *SIAM Journal on Scientific and Statistical Computing*, 9, 1988.

[26] R. Saurel, O. Le Metayer, J. Massoni, and S. Gavrilyuk, "Shock Jump Relations for Multiphase Mixtures with Stiff Mechanical Rrelaxation," *Shock Waves*, 16, 2007, pp. 209–232.

[27] R. Saurel and R. Abgrall, "A Simple Method for Compressible Multifluid Flows," *SIAM Journal on Scientific and Statistical Computing*, 21, 1999, pp. 1115-1145.

[28] R. Weaver, B. Barbee, B. Wie, and B. Zimmerman, "Los Alamos RAGE Simulations of the HAIV Mission Concept," IAA-PDC-15-03-14, *4th IAA Planetary Defense Conference,* Frascati, Roma, Italy, April 13-17, 2015.

[29] M. Bruck Syal, J.M. Owen, and P.L. Miller, "Nuclear and Kinetic Approaches to Asteroid Defense: New Numerical Insights," *46th Lunar and Planetary Science Conference,* 1673.pdf, 2015.