AAS 11-403

COMPARISON OF FRAGMENTATION/DISPERSION MODELS FOR ASTEROID NUCLEAR DISRUPTION MISSION DESIGN

Brian D. Kaplinger* **and Bong Wie†**

This paper considers the problem of developing statistical orbit predictions of near-Earth object (NEO) fragmentation for nuclear disruption mission design and analysis. The critical component of NEO fragmentation modeling is developed for a momentum-preserving hypervelocity impact of a spacecraft carrying nuclear payload. The results of the fragmentation process are compared to static models and results from complex hydrodynamic code simulations, developing benchmark initial conditions for orbital prediction algorithms. The problem is examined in a way that enables high-performance GPU acceleration of the resulting computational system, and the mission design fidelity is improved to allow for high throughput self-gravity and collision models of NEO fragments. Improvements to model efficiency are demonstrated using a range of orbits to assess disruption mission effectiveness.

INTRODUCTION

Asteroids have impacted the Earth in the past and threaten to do so in the future. While the most likely near-term threat is that of a low-altitude airburst, the expected energy of an event such as Tunguska would be devastating in a highly populated area. Additionally, though the population of catastrophic impactors has been well surveyed, it is estimated that thousands of bodies over 140 m in diameter remain undiscovered [1]. Many methods have been suggested for the mitigation of this threat, but most require substantial lead time in order to be effective. A study by the National Research Council suggests that nuclear explosive devices may be the only option for late warning cases [2]. Previous simulations show that disruption, once thought to be undesirable, may substantially reduce the amount of mass remaining on impact trajectories. This method could be available with as little as 10 days of lead time between intercept and the predicted impact date for an orbit like that of the asteroid Apophis [3].

This paper addresses a simulation framework for a disrupted near-Earth object (NEO) dispersing along the orbit. Initial simulations use a spherical NEO model with a dense granite core and an outer shell of tuff material, similar to the description in [1], the only difference being that a meshless particle description is used, with the mass distributed to best approximate the desired body. A formulation for adapting the procedure in [4] is given for a general NEO shape, allowing a wide range of initial conditions with a momentum-preserving fragmentation model. After simulation of the NEO breakup, individual fragments are identified and their resulting velocities mapped to a rotating orbital coordinate system. The trajectory of each fragment is then predicted using an improved version of the algorithms presented in [5], with special attention paid to the probability density of

*Graduate Research Assistant, Asteroid Deflection Research Center, Iowa State University.
†Vance Coffman Chair Professor, Asteroid Deflection Research Center, Iowa State University.

fragment given uncertain initial conditions. The results are then compared to fragmentation models in [6] and resulting from complex hydrodynamic code simulations.

A major bottleneck in determining appropriate mitigation methods for NEOs has been a lack of experimental data on the efficacy of each approach, forcing a reliance on simulations to determine mission effectiveness. As we move from the concept stage into true mission planning for effective NEO threat mitigation, we must depart from simulation of a few sample cases and instead use actual mission parameters to integrate modeling and simulation into the mission design cycle. This paper presents the development of simulation tools designed to be implemented as part of the mission design procedure for nuclear fragmentation and dispersion of an NEO. A brief history of general purpose GPU computing will be given, followed by the particulars of high-level language access to the GPU for this simulation. Motivation for the parallelization of the presented model lies in the decoupled nature of each hydrodynamic particle, relying only on information for its immediate neighbors. Improvements of the fragmentation model are shown to result in 60% cost savings for the simulation and a speedup of over 300x compared to serial CPU implementation. The adaptation of previously presented models to the memory and compute capability of the GPU architecture will be described, as well as steps taken to optimize performance in the presence of GPU limitations.

Past work [5,7] showed that a large amount of data can be processed using GPU simulation. Initial work was focused mostly on prediction of relative impacting mass. Disruption at different times along a given orbit can have a large effect on the resulting shape of the debris cloud. This paper looks at the fragmentation model to better address how uncertainty in the NEO breakup affects orbital prediction, using the model developed in [4]. This has allowed for a revolution in computing on a budget, allowing hundreds of complex simulations to be tested. While new high-performance computing (HPC) technology is shown to solve old problems faster, this paper also addresses the identification of new problems that were previously intractable without the use of a supercomputer or dedicated cluster.

SIMULATION MODEL

This section presents the equations of motion and target model used in the fragmentation and dispersion simulations. Two primary reference targets are used, to emphasize the differences between material composition. Both are 100 meters in diameter, but have different bulk densities and material strength properties. The first target is a rubble-pile asteroid, with a bulk density of 1.91 g/cm^3. This is a likely target for demonstrating the behavior of more porous material. The second target is a single granite boulder with a bulk density of 2.63 g/cm^3. A linear model for material strength is used in this target with a yield strength of 14.6 MPa and a shear modulus of 35 MPa, resulting in a more granulated fragmentation and slower dispersion velocities. Real asteroid targets are expected to fall within these two extremes, with variances for composition, distribution of mass, and orientation. A Smoothed Particle Hydrodynamics (SPH) model [4] is used for the asteroid fragmentation simulation under 3 initial conditions: a subsurface explosion of 100 kt buried at a 5 m depth, a surface blast of 100 kt surrounded by a 1 m thick aluminum impactor, and a standoff blast at 10 m above the surface. We assume an isotropic Weibull distribution of implicit flaws in the NEO material and conduct Monte Carlo simulation to establish a mean response of the target NEO to the fragmentation process. Resulting coherent masses are propagated through a model of solar system dynamics until the predetermined date of impact. Masses remaining on impact trajectories undergo a simulation of reentry into Earth's atmosphere, resulting in final tallies of mass missing the Earth, fragments on capture trajectories, airburst events, and impacts of reduced-mass fragments.

Hydrodynamic Equations

For the purposes of the present simulation study, a meshless hydrodynamics model was desired. This approach would eliminate the need for storing and updating a grid, simplify calculations for large deformations, and allow for contiguous memory access to local field properties. The SPH formulation [8,9] was chosen to satisfy the first two goals, while the latter will be discussed with regards to the GPU implementation. The core idea of SPH is to approximate a field property $f(x)$ by using a mollifier W (also known as an approximate identity) with compact support:

$$\langle f(x) \rangle = \int_{\Omega} f(s) W(x-s) ds, \quad W \in C_0^1(\mathbb{R}^n), \quad \Omega = \text{supp}(W) \tag{1}$$

where the brackets indicate the SPH approximation [9], allowing the field variables to be computed as a sum over the nearest neighbor particles representing the flow. In the present formulation, W is taken as the cubic spline kernel [8,9], with a variable isotropic domain of support with radius h. Changing h in space and time allows for the simulation to respond to changes in flow conditions with a change in local resolution [8,9]. A mass m is assigned to each particle representative in the model, as well as initial position and velocity components (x^β and v^β) in each β direction. Material properties such as density, ρ, and specific energy, e, complete the state description. Similar to the above integral relationship, derivatives and integrals of field functions can be approximated, resulting in the following set of equations [8-10] involving the kernel derivative (a scalar valued function of vector position **x**):

$$\frac{Dx_i^\alpha}{Dt} = v_i^\alpha \tag{2}$$

$$\frac{D\rho_i}{Dt} = \sum_{j=1}^{N} m_j \left(v_i^\beta - v_j^\beta \right) \frac{\partial W(\mathbf{x}_j - \mathbf{x}_i)}{\partial x^\beta} \tag{3}$$

$$\frac{Dv_i^\alpha}{Dt} = -\sum_{j=1}^{N} m_j \left(\frac{\sigma_i^{\alpha\beta}}{\rho_i^2} + \frac{\sigma_j^{\alpha\beta}}{\rho_j^2} + \Pi_{ij} \right) \frac{\partial W(\mathbf{x}_j - \mathbf{x}_i)}{\partial x^\beta} + F_i^\alpha \tag{4}$$

$$\frac{De}{Dt} = \frac{1}{2} \sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij} \right) \left(v_i^\beta - v_j^\beta \right) \frac{\partial W(\mathbf{x}_j - \mathbf{x}_i)}{\partial x^\beta} + \frac{1}{\rho_i} S_i^{\alpha\beta} \epsilon_i^{\alpha\beta} + H_i \tag{5}$$

where repeated indices in a product indicate implied summation over all possible values, $\sigma^{\alpha\beta}$ is the stress tensor, P is the pressure, $S^{\alpha\beta}$ is the deviatoric (traceless) stress tensor, $\epsilon^{\alpha\beta}$ is the local strain rate tensor, F represents external forces, and H represents energy sources. Π_{ij} represents the Monaghan numerical viscosity [9,11] used to resolve shocks, accommodate heating along the shock, and resist unphysical material penetration. The material strength model for the solid target uses an elastic-perfectly plastic description of strength [8-10], where the hydrodynamic stress is determined as

$$\sigma_i^{\alpha\beta} = -P_i \delta^{\alpha\beta} + (1-\eta) S_i^{\alpha\beta}, \quad \eta \in [0,1] \tag{6}$$

where η is a material damage indicator, to be discussed later. It should be noted that fully damaged material ($\eta = 1$) is relieved of all stress due to deformation and behaves as a cohesionless fluid

[10,12]. The rubble-pile target is treated in this manner by default. In this elastic-plastic model, the components of the deviatoric stress tensor $S^{\alpha\beta}$ evolve using the following equation based on Hooke's law [8,13]:

$$\frac{DS_i^{\alpha\beta}}{Dt} = 2G_s\left(\epsilon_i^{\alpha\beta} - 3\delta_i^{\alpha\beta}\epsilon_i^{\gamma\gamma}\right) + S_i^{\alpha\gamma}R_i^{\beta\gamma} + R_i^{\alpha\gamma}S_i^{\gamma\beta} \tag{7}$$

where $R^{\alpha\beta}$ is the local rotation rate tensor, G_s is the shear modulus, and the SPH approximation for these terms is given by

$$\epsilon_i^{\alpha\beta} = \frac{1}{2}\sum_{j=1}^{N}\frac{m_j}{\rho_j}\left[(v_j^\alpha - v_i^\alpha)\frac{\partial W(\mathbf{x}_j - \mathbf{x}_i)}{\partial x^\beta} + (v_j^\beta - v_i^\beta)\frac{\partial W(\mathbf{x}_j - \mathbf{x}_i)}{\partial x^\alpha}\right] \tag{8}$$

$$R_i^{\alpha\beta} = \frac{1}{2}\sum_{j=1}^{N}\frac{m_j}{\rho_j}\left[(v_j^\alpha - v_i^\alpha)\frac{\partial W(\mathbf{x}_j - \mathbf{x}_i)}{\partial x^\beta} - (v_j^\beta - v_i^\beta)\frac{\partial W(\mathbf{x}_j - \mathbf{x}_i)}{\partial x^\alpha}\right] \tag{9}$$

To complete this system, we use the following equations governing the change of support radius h [8,9], and the fracture damage ratio η [10]. The latter is limited in accordance with the number of material flaws activated in the structure.

$$\frac{Dh_i}{Dt} = -\frac{1}{n}\frac{h_i}{\rho_i}\frac{D\rho_i}{Dt}, \quad \frac{D}{Dt}\eta^{1/3} = \frac{c_g}{r_s} \tag{10}$$

where c_g is the crack growth rate, here assumed to be 0.4 times the local sound speed [10], and r_s is the radius of the subvolume subject to tensile strain. In the present model, the latter term is estimated by interpolation based on the strain rate tensor of neighbor particles. An equation of state remains to complete the mechanical system. We use the Tillotson equation of state [14] in the solid asteroid and in the aluminum penetrator used to deliver the surface explosive. This is modified to include porosity, and an irreversible crush strength, for the "rubble pile" target [12,15]. We assume a power law distribution for number of implicit flaws in a volume of material with respect to local tensile strain (a Weibull distribution), and assign flaws with specific activation thresholds to each SPH particle [10]. The maximum damage allowed to accumulate in a volume is described by

$$\eta_i^{\max} = \left(\frac{n_i}{n_i^{\text{tot}}}\right)^{1/3}, \quad \epsilon_i = \frac{\sigma_i^t}{(1-\eta_i)E} \tag{11}$$

where n_i is the number of active flaws ($\epsilon > \epsilon^{\text{act}}$) and n^{tot} is the total number of flaws assigned to a particle, which can vary widely, but is always at least one. Equation (11) also gives the relationship for the local scalar strain, as a function of the maximum tensile stress σ^t, the local damage, and the Youngs modulus E.

Orbit Propagation

Statistics representing the fragmented system are collected and stored as cumulative density functions for the needed variables, similar to those shown in Fig. 1. A representative fragment system of 10,000 to 100,000 fragments is created from these statistics using inverse transform sampling.

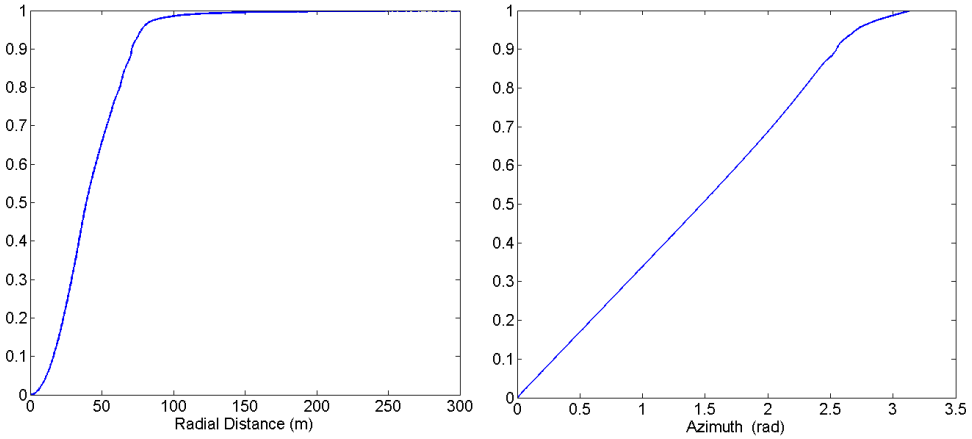

Figure 1. Cumulative Density Functions for Disrupted Asteroid.

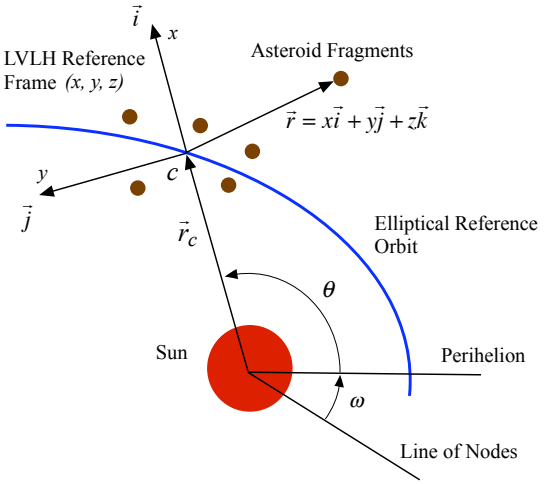

Figure 2. Rotating Local-Vertical-Local-Horizontal (LVLH) Frame.

The debris cloud is given global coordinates in a Local-Vertical-Local-Horizontal (LVLH) reference frame about the center of mass, as shown in Fig. 2. Since the hydrodynamic model is axisymmetric, and has a definite direction of maximum momentum along the axis of symmetry, a desired deflection direction must be chosen. In the present paper, deflections along the 3 LVLH axes are considered: radial ($\pm x$), transverse ($\pm y$), and normal ($\pm z$) axes. These are then integrated to predict an ephemeris for a 48 hour period surrounding the nominal time of impact. Since the LVLH reference frame is computationally beneficial for self-gravity and collision modelling among fragments [5], we use the nonlinear relative equations of motion for this frame to govern fragment trajectories [5,7]:

$$\ddot{x}_i = 2\dot{\theta}\left(\dot{y}_i - \frac{\dot{r}_c}{r_c}y_i\right) + \dot{\theta}^2 x_i + \frac{\mu}{r_c^2} - \frac{\mu}{r_d^3}(r_c + x_i) + \frac{\mu_E}{r_{Ei}^3}(x_E - x_i) + F_i^x \tag{12}$$

$$\ddot{y}_i = -2\dot{\theta}\left(\dot{x}_i + \frac{\dot{r}_c}{r_c}x_i\right) + \dot{\theta}^2 y_i - \frac{\mu}{r_d^3} + \frac{\mu_E}{r_{Ei}^3}(y_E - y_i) + F_i^y \tag{13}$$

$$\ddot{z}_i = -\frac{\mu}{r_d^3}z_i + \frac{\mu_E}{r_{Ei}^3}(z_E - z_i) + F_i^z \tag{14}$$

where x, y, z, r_c, and θ are defined as shown in Fig. 2, r_d is the length of the relative coordinate vector, μ and μ_E are gravitational parameters for the sun and the Earth, r_{Ei} is the distance from each fragment to Earth, and (F^x, F^y, F^z) are the combined acceleration components due to 3rd body gravitational terms (solar system major body model [7]), self gravity, and collision corrections. The threading structure for computing the values for self gravity terms is described in [5], while collisions are predicted using a Sort-and-Search algorithm [16], resulting in post-collision changes to position and velocity of fragments. An elastic spherical collision model is assumed for the fragments, with a coefficient of restitution of 0.5.

Uncertainty Analysis

In order to test the response of orbital dispersion with respect to uncertain initial fragment positions and velocities, a Gaussian noise is added to the mapping around the nominal center of mass. A standard deviation of 10% is assumed, resulting in deviations from the hydrodynamic simulations up to $\pm$ 30%. For a given orbit, 1000 random perturbations are integrated to impact, resulting in an average system behavior and a standard deviation representative of the uncertainty due to the initial conditions.

This procedure is completed for a database of 906 orbits chosen to impact at a fixed date. The orbital parameters for the nominal trajectory are sampled from a (a, e, i) space that represents the distribution of known NEOs, as shown in Fig. 3. For each of 6 deflection directions, the Monte Carlo procedure described above results in a characteristic behavior of a disrupted NEO on the range of orbits tested.

DISRUPTION MISSION PROFILES

This section outlines the initial conditions for three method of NEO deflection using nuclear explosive devices. In all cases, a 100 m diameter target asteroid is modeled with an energy source of 100 kt. Thermal emission is omitted from the subsurface and surface explosions due to absorbtion by surrounding material in the time scale of interest.

Subsurface Explosion Setup

For this simulation, the explosive is modeled as a cylindrical energy source buried at a depth of 5 meters. As shown for the solid target in Fig. 4, The blast wave compresses the NEO, reducing it to fragments, and disperses it primarily along the axis of the explosion. The resulting fragment distribution for a case like this has a peak between 20-70 m/s, with a tail of high-speed ejecta like that shown in Fig. 4.

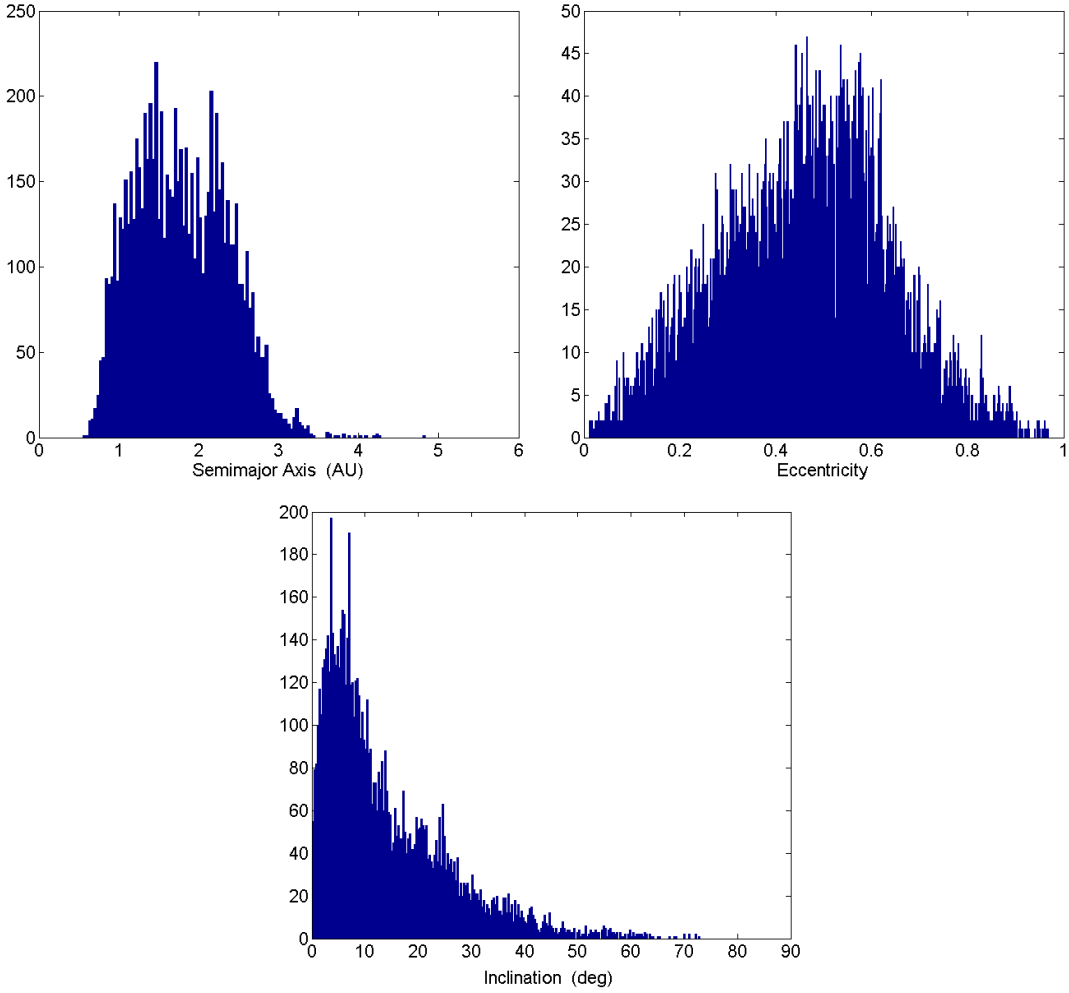

Figure 3. Histograms of Known NEO Population.

Surface Penetrator Model

Two main models for an explosion at the surface are used. One is a static explosion, which results in vastly different systems depending on the composition of the body. For a solid target, cratering and pitting is expected rather than disruption. Even dispersed rubble-pile asteroids have a far lower mean fragment velocity than a similar subsurface system. The second model, shown here, includes an aluminum penetrator impacting the surface at 6.1 km/s. The explosion thermal energy turns the high-mass impactor into a plasma, which burrows into the surface as it releases its energy. Slower dispersion velocity is observed than the subsurface case, but this approach is extremely beneficial from an engineering standpoint, as there is strong coupling between time-to-impact and a reduction in mission fuel cost [17]. The benefit to this method relative to a subsurface explosion is that it does not require a rendezvous, and therefore there are available launch windows for this type of mission right up until immediately before the impact date.

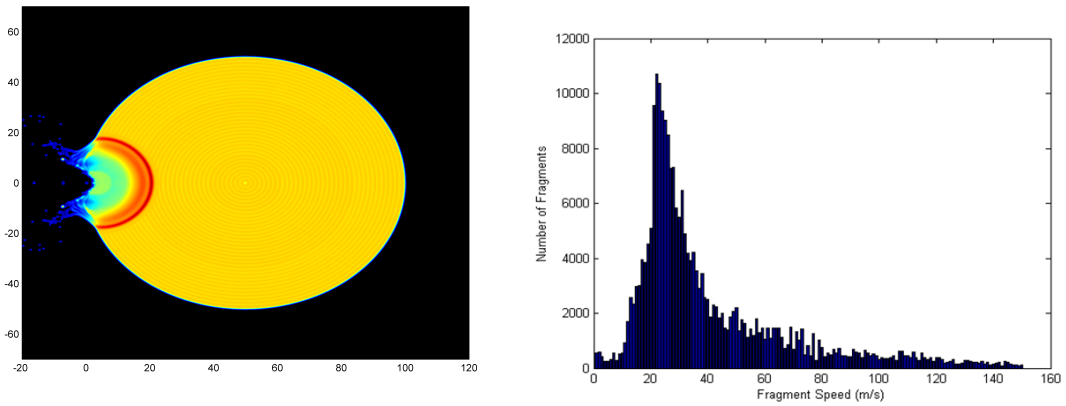

Figure 4. Subsurface Explosion and Resulting Fragment Velocities.

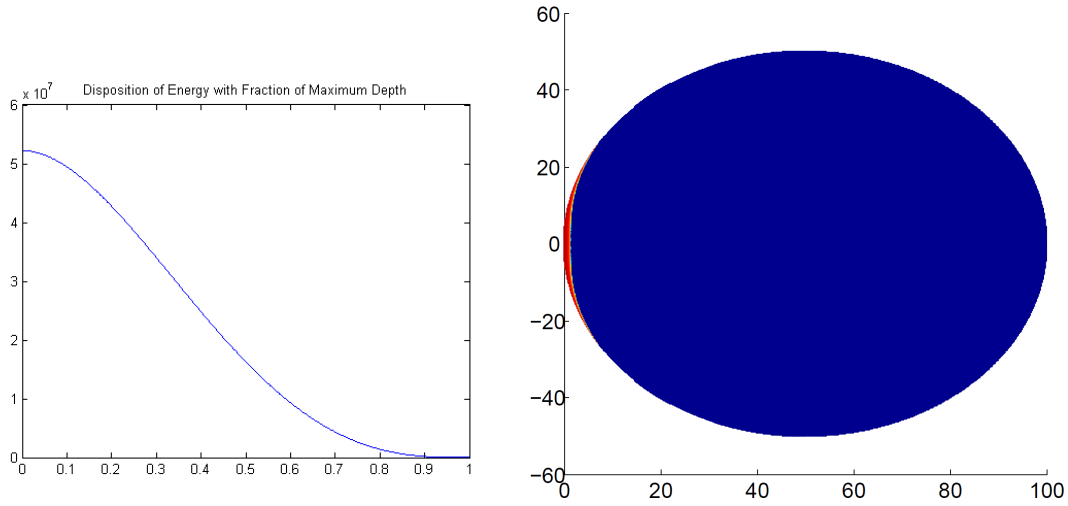

Figure 5. Radial Energy Deposition and Total Deposition Region.

Standoff Energy Deposition

For a standoff blast, additional physics must be considered. An energy deposition strategy is required that does not directly compute X-ray and neutron scattering in the target. For this, a ray-tracing algorithm is used with radial energy deposition at the surface as shown in Fig. 5 for neutrons. This is derived from a Monte Carlo scattering result from TART, a DOE neutron deposition code, in NEO analog materials [18]. A 10% neutron yield is assumed for these simulations, and a maximum deposition depth of 1.5 m to compare to deposition predicted for chondritic materials [19]. The overall deposition region (shown as the logarithm of deposited energy) is also shown in Fig. 5. A modified SPH node representation is created that resembles an ablative modeling grid used in high-energy deposition physics. This distribution is shown in Fig. 6, and has a minimum smoothing scale of 0.1 cm with a maximum local change rate of 10% up to 0.2 m resolution. Also in Fig. 6, the resulting ablation provides an effect similar to that of a rocket, but also disrupts the rubble-pile target completely.

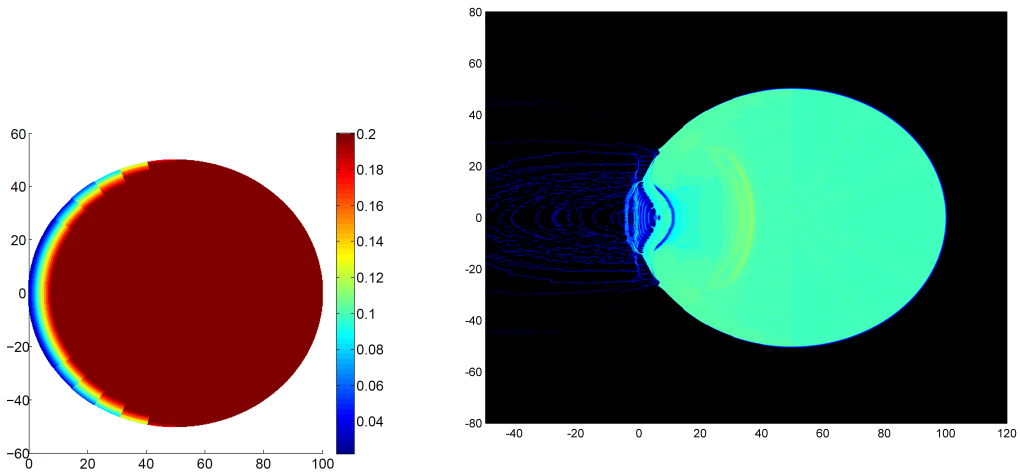

Figure 6. SPH Nodes and Resulting Ablation for Standoff Model.

COMPUTATIONAL APPROACH

This section address the computational approach used to solve the disruption problem. Each state variable update for a fragment is conducted in parallel at each time step. A variety of hardware was available for this project, with a substantial difference in performance. This allowed us to get reasonable estimates on the computational cost of this simulation, in comparison to LINPACK performance numbers. Performance can vary based on the type of arrays used, and the number of threads dedicated to each GPU calculation. These factors are determined by the CUDA Compute Capability (CUDA CC), which is a property of the GPU [20]. These cost estimates are used to determine hardware performance on the various systems. A summary of the hardware used is shown in Table 1 (Note: all CPUs are Intel brand, and all GPUs are NVIDIA brand).

Table 1. Hardware for Benchmark Systems

System	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5
CPU	1x Core2 Q6600	1x Core2 Q6600	1x Xeon X5550	2x Xeon E5520	2x Xeon X5650
CPU Cores	4	4	4	8	12
CPU TPEAK	9.6 GFLOPs	9.6 GFLOPs	12.8 GFLOPs	21.36 GFLOPs	32.04 GFLOPs
GPU	1x 8800GTS	1x GTX470	1x GTX480	4x Tesla c1060	4x Tesla c2050
GPU Cores	112	448	480	960	1792
GPU TPEAK	84 GFLOPs	324 GFLOPs	385 GFLOPs	336 GFLOPs	2060 GFLOPs
CUDA CC	CC 1.0	CC 2.0	CC 2.0	CC 1.3	CC 2.0

Each thread on the GPU calculates the state variable change for one fragment, with the GPU kernel limited to one time step. This is necessary because the positions of the planets and other gravitating bodies must be calculated and transferred to the GPU at each time step. Additionally, the positions of fragments at each integration substep are shared among multiple GPUs and CPU threads. For this reason, the present hydrodynamics model is predominantly bandwidth-limited for small data sets. While grid information is not retained, one of the disadvantages of the SPH

hydrocode is that neighboring particles must be calculated at each time step. Our approach in this model is to create a bounding volume for each SPH particle and perform the same Sort and Sweep in parallel as used to detect collisions in the orbital model [16]. We retain the information for neighbors connected by material strength, as well as carrying neighbor information through the correction step of the integrator. This results in a 28% performance improvement over recalculating neighbors at both the prediction and correction steps, while allowing for a variable time step based on the Courant condition [8,9]:

$$\delta t = \min_i \frac{h_i}{c_i} \tag{15}$$

where c is the local sound speed. While the reduction operation to determine the new time step can be done in parallel, all GPU threads must have position information for all particles to determine neighbors. This requirement could be eliminated through clever domain decomposition, but there is a tradeoff between associating a mesh to the model and taking advantage of contiguous memory sections of particles. Load balancing would also require additional communication between GPUs, which has an impact on performance, as PCI-E bandwidth is one of the limiting factors in GPU acceleration [20]. Our memory model for this simulation includes a shared host memory, distributed device memory for each GPU, and data transfers between them handled through explicit array transfer. Each block of compute threads on the GPU takes the data it needs from the global device memory when the kernel reaches its block. This is an important factor, because the varying compute capabilities have different limitations on this block memory, changing the number of threads that may be used in the calculation. Constants are transferred to all GPU memories implicitly using a pointer to the host constant value. While modern dedicated compute GPUs have a high amount of onboard memory, it usually is far less than system memory. Though it may seem advantageous to calculate parameters for every time step before the start of the simulation, the arrays resulting from this approach are quite large. Each model of GPU has a limited number of memory registers available to each computing block of threads [20]. Therefore, the use of several large arrays can actually slow down the simulation in some cases, by lowering the number of threads below the maximum allowed by the architecture. This is addressed in the present code by utilizing asynchronous data transfers and kernel launches to split the work into streams. This allows the CPU to calculate new parameters needed for the next time step while the GPU is updating the current step.

RESULTS

In order to address the effectiveness of different fragmentation methods, we compare the mass remaining on impacting trajectories (including the uncertainty from the Monte Carlo process) against other methods for each orbit. For example, Fig. 7 shows the relative impacting mass for the surface penetrator in both the solid and the rubble-pile targets. On average across the orbits tested, the impacting mass was 10% higher for the solid target compared to the rubble target for deflections in the radial direction. Estimates like this will eventually allow for tabular look-up of performance for various methods without direct computation. It was also found that impacting mass for the solid target was 20% higher than the rubble target in the transverse direction.

No strong correlation was found for the semimajor axis or eccentricity of the NEO orbit with only 15 days of lead time, however, deflections on orbits with high inclination were more effective, as shown in Fig. 8. for the subsurface case. Ejecta velocities for the dynamic surface burst (at 6.1 km/s) were within the 10% assumed noise range compared to a static buried explosive, as shown

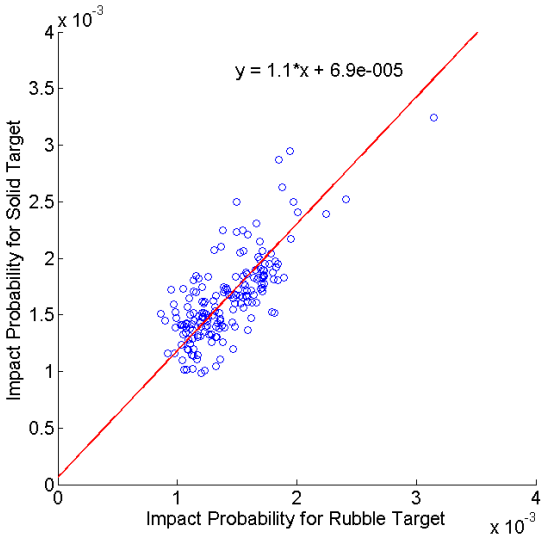

Figure 7. Relative Performance for Surface Impactor.

in Fig. 9. Thus, an emphasis might be placed on hypervelocity intercept and guidance technology rather than a rendezvous mission. One possible interceptor design includes an aluminum impactor followed by an explosive. With both interceptors impacting at 6.1 km/s, the resulting ejecta speed is on average 25% higher than the single surface blast, with a standard deviation of 5.3%. Figure 10 shows the relative velocities for these cases, which results in 20% lower impacting mass on most orbits tested.

Computational Optimization

A single computational node was used to determine optimal distribution of MPI and OpenMP processes across the current worker topology being considered. This system has 2 sockets populated with Intel Xeon X5650 six-core CPUs at 2.66 Ghz. Intel HyperThreading technology is enabled, resulting in 24 logical processors visible to the operating system. Additionally, the default level of OpenMP threading is 24. There are 4 NVIDIA Tesla C2050 GPU cards, each connected on a dedicated PCI-E x16 bus. System RAM is 32 GB, while each GPU has 3 GB GDDR5 for a total GPU work unit of 12 GB (11.2 GB with ECC enabled). Fourteen multiprocessors on each card result in 448 shader cores each, limited to a maximum kernel launch of 1024 threads per thread block. This new "Fermi" GPU architecture has a theoretical peak performance of 515 Mflops in double precision, representing a game-changing leap forward in GPU double precision computing, as shown by real-world results [5].

While grid information does not need to be stored for this model, the drawback is that neighboring particles need to be determined at each time step. Since the integration scheme is a second order predictor-corrector scheme, particle information is needed at both steps. The first change made to the standard scheme was to retain the neighbor ID information for the corrector step. Only the kernel and kernel derivative values at the new neighbor predicted position need to be computed. This reduced time-to-solution by 30.2% compared to a two-stage neighbor finding algorithm. Results for both cases were compared, and while ending state values could be slightly different the distribution remained the same, and the method conserved energy slightly better through the end of the simula-

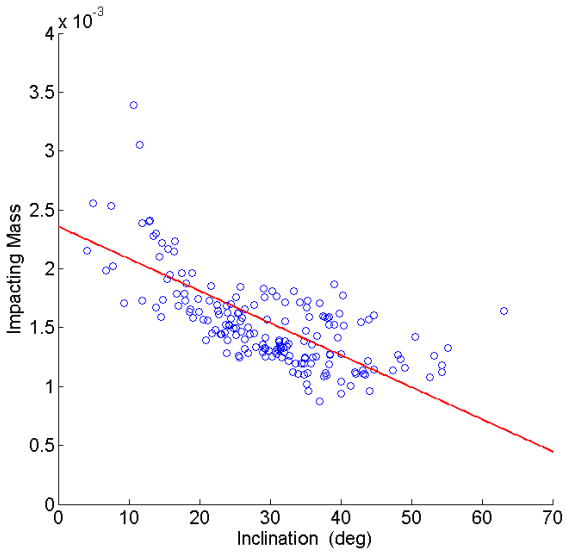

Figure 8. Impacting Mass for Subsurface Explosion on Orbits with Varying Inclination.

tion. A possible beneficial side effect of this approach is the reduction of importance of neighbor changes in a prediction step, which might help damp out numerical instabilities and allow for larger time step changes. This is something to be tested in the future. Also, while brute force computation of neighbor particles was the original approach, a Sort-and-Sweep method reduced this time by 36% for the present target model. This method scales as $N \log N$ rather than N^2 [16].

Neighbor information arrays were stored in a column-major format by particle, allowing stride 1 access to the ID number, kernel value, and kernel derivative values for each neighbor of a particle. Additionally, loop unrolling and inlining for simple functions were implemented, and optimization flags were passed in the build step. For the GPU model, utilizing asynchronous kernel launches to continue computation without synchronization resulted in an 8% performance increase. The theoretical load on each process should be equal, since each has the same number of particles for which a state update needs to be computed. However, in areas of quickly changing density (for example the expanding shock wave), the number of average neighbors for a particle goes up dramatically. This is controlled in 2 ways to aid load balancing. First, the ID assignment scheme works outward in a radial manner, while making sure that mirroring particles on opposite sides of the primary axis are adjacent in memory. Second, the evolution of h strives to keep the number of neighboring particles near the starting value, resulting in an equal computational burden. For the GPU model, a load factor was developed, dividing the minimum time to complete a section between synchronizations by the maximum time. Sampling this load factor allows one to better understand the efficiency of the code section. At a time of 1.2 ms, an example chosen because of the high energy of this point of the simulation, a vertical distribution of particle IDs resulted in a load factor efficiency of around 0.68. The present method has improved this portion to a median of 0.87.

Performance

Pure MPI scalability for up to 12 processes was tested on the present hardware, resulting in near linear scaling and a total parallel speedup of 8.9 for MPI. Including OpenMP in a Hybrid parallel scheme, a total parallel speedup of 11.9 is achieved, showing near perfect expected scalability across

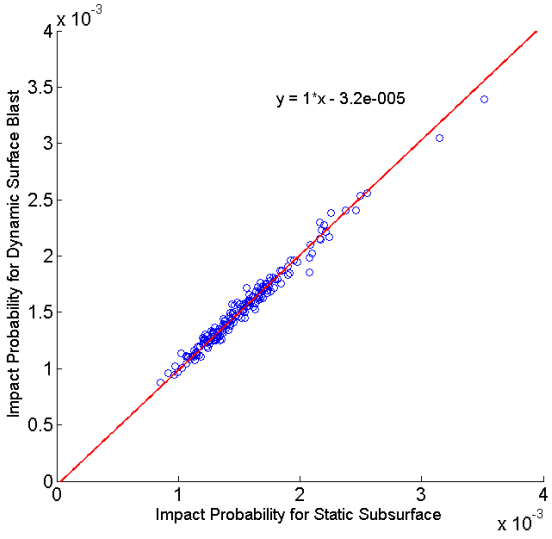

Figure 9. Impacting Mass Comparison for Subsurface and Dynamic Surface Cases.

a single node as shown in Fig. 9. Thus, each additional planned node might add almost 12x speedup for host computation, minus internode communication overhead. As shown in Fig. 9, when the binding option is passed to the Hydra process manager to set 1-2 MPI processes per socket, and an OpenMP thread level of 6 is set, the best performing speedup for the system is obtained. This corresponds to a value of 11.2 for 12 computational threads and 11.9 for 24 computational threads. Performance improvement using ¿ 12 threads is predominantly dependent on the HyperThreading hardware implementation. This is shown to only have an improvement over 12 threads when the shared thread level is 4, 6, 8, or 12. However, good performance with 12 threads among these hybrid schemes was limited to an OpenMP level of 6 and 12. While the default OpenMP maximum thread level for this system is 24, benefits from this technology are implementation dependent, so the preferred setup for future system programming is 1 MPI process per socket with an OpenMP threading level of 6 unless improvement from additional MPI processes can be demonstrated.

GPU acceleration performance for this method is a substantial improvement over a larger CPU-only cluster. Since the threading structure of the GPU is limited to SIMD kernel launches of multiple threads on a multiprocessor, serial performance for comparison is measured on the host CPU. Fig 9 shows then relationship between the number of GPUs used in the state update process and the parallel speedup. At least 1 MPI thread is needed per GPU. In fact, using the currently supported CUDA Fortran toolkit (version 4.0), binding between CPU thread and GPU control requires that additional threading use a shared memory approach such as OpenMP. In a previous test, GPU speedup for this architecture ranges from 50x to 120x for a 50 m diameter target problem. Since the GPU approach works well for data-parallel problems, one would expect that increasing the scale of the problem would yield better performance. In fact, using the current solid target standoff model (3.1M particles) maximum speedup on a single node is increased to 357.9x, as shown in Fig. 9. Since the neighbor search problem is substantially increased, the parallel structure of the GPU is far preferred to the hybrid CPU programming model.

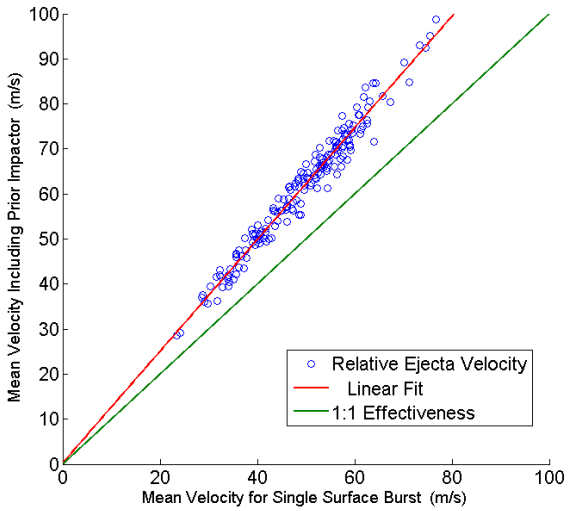

Figure 10. Mean Ejecta Velocity for Single and Double Impactor Cases.

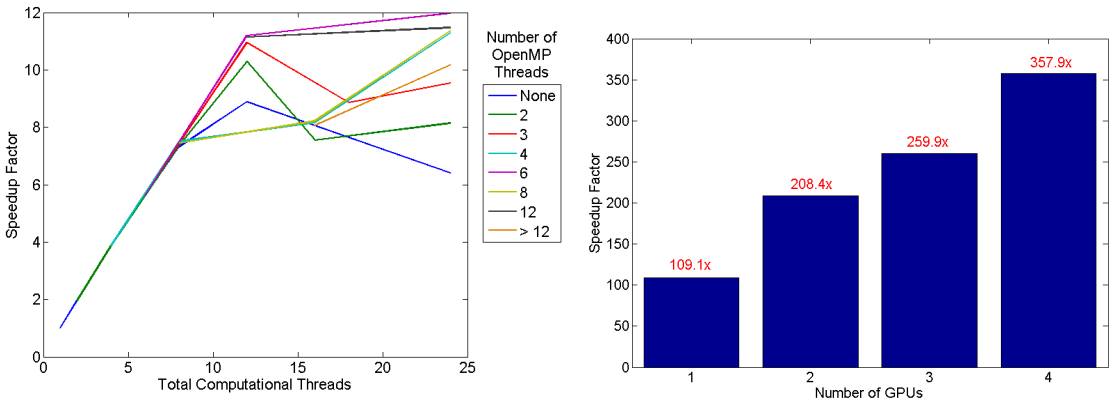

Figure 11. Comparison of Single-Node Performance on CPU and GPU.

CONCLUSION

The present SPH hydrocode suggests that a dynamic model of a hypervelocity surface burst yields results similar in spatial and temporal distribution to a static subsurface explosion. This gives additional launch windows for mission design, limits the fuel needed for a rendezvous burn, and avoids the need to bury the explosive payload. Additionally, the dynamic model should better predict system behavior when addressing high velocity penetrator architectures. This might give an option for realistically determining the limits of such a system for asteroid deflection missions. NEO orbital parameters such as semimajor axis and eccentricity were not found to be important for these time scale, but it was found that inclination was important in determining effectiveness of any given method.

All methods of disruption using a 100 kt nuclear energy source were quite effective for 100 m diameter targets for 15 days lead time, regardless of the orbit considered. Future work should consider larger bodies, a range of source energies, and lead times specific to the available mission

time for a given orbit. There was a slight advantage to the two impactor system analyzed compared to a single surface burst, in the form of higher coupled energy and a lower impacting mass on most orbits, including uncertainty.

New HPC technology utilizing GPU acceleration has resulted in orders of magnitude improvement in computational ability. Speedup of the GPU accelerated model compared to serial execution for the both target models has been demonstrated. While the 330,000 particles of the penetrator target are limited mostly by communication bandwidth, the 3.1 million particles in the standoff model are limited by computational speed and memory bandwidth for the threads on the GPU. A substantial speedup improvement, from 53x to 358x, is observed. This shows single node computational performance on the same order as a moderate cluster. The ability to run multiple cases to address statistical system behavior results in simulation being integrated into overall mission design. Mission effectiveness can be estimated in advance of a need for mission design, allowing new architectures and interchangeable components for a universal deflection plan. This paper outlined the development of software and hardware tools to aid the planning of NEO deflection mission design, and the current project strives to identify key technologies for effective implementation. This technology provides a useful reduction in time-to-solution comparable to 30 similar CPU-only nodes (which would cost $4,000 each) in a $14,000 form factor, showing a 8.6x improvement in cost-adjusted performance.

ACKNOWLEDGMENT

This research work was supported by a research grant from the Iowa Space Grant Consortium (ISGC) awarded to the Asteroid Deflection Research Center at Iowa State University. The authors would like to thank Dr. Ramanathan Sugumaran (Director, ISGC) for his support of this research work.

REFERENCES

[1] Boslough, M., "Airburst Warning and Response," IAA-PDC-2166721, *2nd IAA Planetary Defense Conference*, Bucharest, Romania, May 9-12, 2011.

[2] Committee to Review Near-Earth Object Surveys and Hazard Mitigation Strategies, National Research Council, *Defending Planet Earth: Near-Earth Object Surveys and Hazard Mitigation Strategies*, 152 pp, 2010.

[3] Kaplinger, B., and Wie, B., "Parameter Variation In Near-Earth Object Disruption Simulations Using GPU Acceleration," AAS-11-267, *21st AAS/AIAA Spaceflight Mechanics Meeting*, New Orleans, LA, Feb. 13-17, 2011.

[4] Kaplinger, B., Wie, B., and Dearborn, D., "Nuclear Fragmentation/Dispersion Modeling and Simulation of Hazardous Near-Earth Objects," IAA-PDC-2138266, *2nd IAA Planetary Defense Conference*, Bucharest, Romania, May 9-12, 2011.

[5] Kaplinger, B., and Wie, B., "Optimized GPU Simulation of a Disrupted Near-Earth Object Including Self-Gravity," AAS-11-266, *21st AAS/AIAA Spaceflight Mechanics Meeting*, New Orleans, LA, February 13-17, 2011.

[6] Wie, B. and Dearborn, D., "Earth-Impact Modeling and Analysis of a Near-Earth Object Fragmented and Dispersed by Nuclear Subsurface Explosions," AAS 10-137, *20th AAS/AIAA Space Flight Mechanics Meeting*, San Diego, CA, February 15-17, 2010.

[7] Kaplinger, B.D., Wie, B., and Dearborn, D., "Preliminary Results for High-Fidelity Modeling and Simulation of Orbital Dispersion of Asteroids Disrupted by Nuclear Explosives," AIAA-2010-7982, AIAA/AAS Astrodynamics Specialists Conference, Toronto, Ontario, Canada, August 2-5, 2010.

[8] Monaghan, J.J, "Smoothed Particle Hydrodynamics," *Rep. on Prog. in Physics*, vol. 68, pp. 1703-1759, July 2005.

[9] Liu, G.R., and Liu, M.B., *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*, Singapore: World Scientific Publishing, 2003.

[10] Benz, W., and Asphaug, E., "Simulations of Brittle Solids using Smooth Particle Hydrodynamics," *Computer Physics Communications*, vol. 87, pp. 253-265, 1995.

[11] Hiermaier, S., Konke, D., Stilp, A.J., and Thoma, K., "Computational Simulation of the Hypervelocity Impact of Al-Spheres on Thin Plates of Different Materials," *Intl. Journal of Impact Engineering*, vol. 20, pp. 363-374, 1997.

[12] Jutzi, M., Benz, W., and Michel, P., "Numerical Simulations of Impacts Involving Porous Bodies I. Implementing Sub-Resolution Porosity in a 3D SPH Hydrocode," *Icarus*, vol. 198, pp. 242-255, 2008.

[13] Randles, P.W., and Libersky, L.D., "Smoothed Particle Hydrodynamics: Some Recent Improvements and Applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 139, pp. 375-408, 1996.

[14] Tillotson, J.H., "Metallic Equations of State for Hypervelocity Impact," General Atomic Technical Report GA-3216, 1962.

[15] Schuster, S.H., and Isenberg, J. "Equations of State for Geologic Materials," Defense Nuclear Agency Technical Report DNA-2925Z, 1972.

[16] LeGrand, S., "Broad-Phase Collision Detection with CUDA," *GPU Gems 3*, ed. H. Nguyen, Addison-Wesley, 2007.

[17] Wagner, S., and Wie, B., "Analysis and Design of Fictive Post-2029 Apophis Intercept Mission for Nuclear Disruption," AIAA-2010-8375, *AIAA/AAS Astrodynamics Specialists Conference*, Toronto, Ontario, Canada, August 2-5, 2010.

[18] Miles, A.R., "Asteroid Deflection via Standoff Nuclear Explosions," *Asteroid Deflection Research Symposium*, Arlington, VA, October 23-24, 2008.

[19] Plesko, C.S., Weaver, R.P., and Huebner, W.F., "Energy Deposition in Hazard Mitigation by Nuclear Burst: Sensitivity to Energy Source Characteristics, Geometry, and Target Composition," Paper 2588, *42nd Lunar and Planetary Science Conference*, March 711, 2011.

[20] Kirk, D.B. and Hwu, W.W., *Programming Massively Parallel Processors*, Burlington: Morgan Kaufmann, 2010.