AAS 11-266

OPTIMIZED GPU SIMULATION OF A DISRUPTED NEAR-EARTH OBJECT INCLUDING SELF GRAVITY

Brian D. Kaplinger[*] and Bong Wie[†]

This paper focuses on the development of a simulation model for disrupting a Near-Earth Object (NEO) on terminal approach with the Earth. The problem is simulated numerically using a Graphics Processing Unit (GPU) architecture, and is designed to highlight the benefits of this technology. A high-fidelity model, including mutual gravitation and collisions between NEO fragments, is developed and tested for the GPU. The unique limitations of this computational infrastructure are presented, as well as optimization strategies applied to the present model. The results of this project reflect a new range of high-performance computing options available to the planetary defense research community.

INTRODUCTION

This section describes the motivation behind the Near-Earth Object (NEO) disruption problem, and the desire to accelerate present simulation models using Graphics Processing Units (GPUs). It also presents the concerns and limitations of modeling self-gravity in a fragmented NEO system.

Asteroid Deflection

With thousands of known NEOs crossing the orbit of Earth, and over 1100 of these considered potentially hazardous at this time [1], there has been significant scientific interest in developing the capability to deflect an NEO from an impacting trajectory. Conventional wisdom, and a substantial amount of planetary defense literature, has held that the best way to accomplish this goal is to slowly push the NEO onto another trajectory, or to use an impulsive force (nuclear standoff explosion or kinetic impactor) at least a decade in advance of impact [2, 3]. Two main motives drive research into high-energy, last minute options for asteroid deflection: a) Many bodies on impacting trajectories may not be detected early enough to spend decades deflecting their orbits, and b) The expense and risk of an interplanetary mission to deflect an NEO will require that an object be demonstrated to pose a substantial and imminent threat before action is taken.

While the “late notice” motive is becoming less relevant as more NEOs are catalogued, there are still small bodies that are detected within weeks of closest approach [4], and it would be hard to prove that no more such bodies exist. The “late decision” motive is a political and fiscal reality that planetary defense researchers will have to consider. Even if a viable method of deflection is demonstrated and awaits deployment, a potential target must be shown to impact the Earth considering all relevant uncertainties. As the long-term ephemerides of a small body in the solar system contain many uncertainties that could account for several Earth radii of displacement [5, 6], we expect that a decision to deflect an NEO will come within the last few orbits before impact (possibly after a close approach with the Earth alters the body trajectory).

The asteroid 99942 Apophis is one such example. Initially given a much higher probability of impact on April 13, 2036, the current estimated probability of impact is 1 out of 250,000 after many additional observations. There is still much uncertainty of what will happen after Apophis passes within 36,000 km of

[*] Graduate Assistant, Asteroid Deflection Research Center, Iowa State University, 2271 Howe Hall, Room 2355, Ames, IA, 50011-2271, bdkaplin@iastate.edu.

[†] Vance Coffman Endowed Chair Professor, Asteroid Deflection Research Center, Iowa State University, 2271 Howe Hall, Room 2355, Ames, IA, 50011-2271, bongwie@iastate.edu.

the Earth on April 13, 2029 [7]. If a body like Apophis was forced into a resonant orbit after a close approach with the Earth and was confirmed to be on an impacting trajectory, we would only have a short window in which to act (7 years in the Apophis scenario). This has been shown to be problematic for mission design, with reasonable launch windows only giving a few weeks of time for a deflection to take place [8].

Past research has shown that the use of a kinetic impactor or nuclear explosive device will possibly exceed the gravitational binding energy of many NEOs during a deflection attempt [9]. Additionally, it has been shown that fragmenting an Apophis-like body on an impacting trajectory can reduce the amount of impacting mass remaining on impacting trajectories to 1% in as little as 15 days [6,10], a scenario in which some amount of impacting mass is inevitable. Modeling of atmospheric reentry for a fragmented body has suggested that lowering the individual masses results in substantial reduction of ground impacts, with many fragments burning up or being partially ablated by the atmosphere [6]. Therefore, it is desired that any viable last-minute option lower the impacting mass below this threshold, allowing the atmosphere to have an increased effect. For this reason, we investigate the timing of NEO fragmentation and the scaling of initial explosive energy to model NEO disruption and determine a desired course of action in a “late notice” or “late decision” scenario.

GPU Computing

As the Central Processing Unit (CPU) of the modern personal computers and workstations evolved, there was a substantial problem overcoming thermal issues that accompanied additional computing power. One solution to these thermal issues was to design multi-core CPUs capable of handling several computing threads in parallel [11]. Additionally, as graphics for games and commercial applications grew more demanding, a separate Graphics Processing Unit (GPU) was implemented to allow the screen rendering to be performed by an alternate processing chip. In most graphic applications, either pixels or sections of the display can be calculated separately from each other. Due to this fact, the GPU evolved into a many-core parallel structure, trading core clock speed for number of parallel computing threads. At the time of writing, retail GPU chips are available with up to 512 cores and over 1 trillion floating point operations per second (FLOPs) of theoretical computing power [11,12]. For comparison, a standard workstation processor averages 10-20 GFLOPs. Due to this leap in technology, the GPU has been considered by many the ideal commercially-available massively parallel architecture.

Early simulations and computing programs written for execution on the GPU were done using graphics processing languages. These approaches cleverly harnessed the computing power available by casting a problem as one involving textures and pixels. DirectCompute, OpenGL, and other languages were very powerful methods for moving computation from the CPU to the GPU, freeing resources and allowing problems to scale more easily [13]. However, for general scientific computing this method had a steep learning curve. In 2006, NVIDIA launched the Compute Unified Device Architecture (CUDA), which allowed compilation of GPU executables to be created through extensions to the C language. This approach is highly portable, and an example of the versatility of general scientific computing on the GPU [12]. The CUDA approach will be the method discussed in this paper.

Self-Gravity

The mutual gravitational force among bodies in a fragmented NEO system has been a subject of interest in recent years. Once considered too small to be significant, this force (as well as friction and dissipative collisions) has been shown to be important in considering the reaggregation of fragments into a rubble-pile [14]. Past results [10,16] have shown that a fragmented NEO with low initial fragment velocities is substantially affected by mutual gravitational forces. A complication of this effect is that models including self-gravity are far more expensive in terms of computational resources. Results from smaller systems [10] have been presented in planetary defense research, but adequate modeling of realistic systems requires use of High-Performance Computing (HPC). GPU acceleration of a fragmented NEO system has allowed for compression of elapsed time on the order of several hundred. A speedup of over 500x compared to serial implementation is shown in this paper, indicating the ability to address problems previously considered intractable.

PROBLEM FORMULATION

This section describes the underlying problem being solved. A model of the fragmentation due to the subsurface detonation of an explosive is briefly outlined. We present previous results describing the behavior of this model to scaling of explosive energy and timing along the orbit. The general formulation of the orbit propagation model used is discussed, followed by the model used for atmospheric reentry.

Subsurface Explosion Model

The hypothetical deflection mission target presented in this paper is an NEO with a total mass of 2.058E10 kg and a diameter of 270 meters. A full description of the construction of this Apophis-like model can be found in [15]. A nuclear explosion equivalent to 300 kT was simulated in a cylindrical region below the surface of the body, creating a shock that propagates through the body resulting in fragmentation and dispersal. The core region was modeled with a linear strength model and a yield strength of 14.6 MPa. The mass-averaged speed of the fragments after 6 seconds was near 50 m/s with peak near 30 m/s, as shown in Figure 1. Figure 1 also shows the axisymmetric distribution of body fragments after the completion of the subsurface explosion simulation.

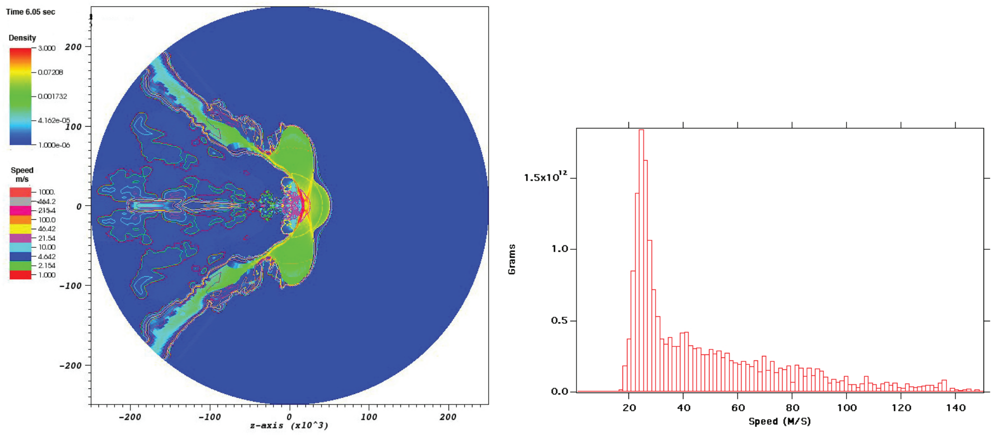


Figure 1. Distribution of Fragments and Velocities for 300 kT Subsurface Explosion [10].

A three-dimensional fragment distribution was constructed by Dr. David Dearborn at Lawrence Livermore National Laboratory from the hydrodynamics model by rotating the state variables about the axis of symmetry, resulting in particle state values for a reasonable NEO fragmentation. To track the dispersion of these fragments along the orbital trajectory before impact, the relative position axes were aligned such that the highest momentum projectiles coincided with the desired deflection direction at the time of the explosion. A velocity scaling parameter allows the testing of distributions with fragment velocities different than the predicted 50 m/s maximum velocity from the explosion simulation.

Scaling of initial fragment velocities was done to model critical and sub-critical breakup of an NEO due to high-velocity impact and other impulsive events. Figure 2 indicates that slower initial velocity fields increase the relative effect of total impacting mass over 15 days. It should be noted, however, that in this work and [16] some fragments that would otherwise hit Earth did not do so after accounting for self-gravity. While the total impacting mass increased, the individual fragments involved in the collision were changed. Since this model does not have isotropic distribution of initial asteroid mass, it is possible for lower mass fragments to be exchanged for higher mass impacts, thus lowering the expected impacting mass. It should be noted that these cases are statistical in nature and are designed to exhibit expected system behavior for a disrupted NEO rather than any given deflection capability.

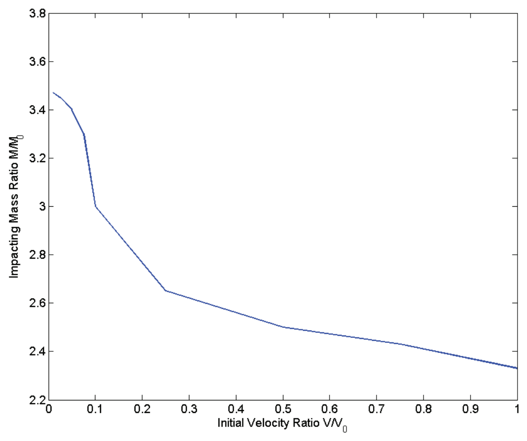


Figure 2. Effect of Initial Velocity on Relative Importance of Mutual Gravitation [10].

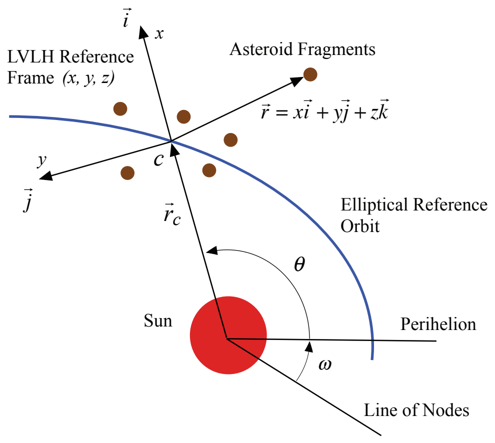
Orbit Propagation Model

In order to have a nominal impacting trajectory that is independent of platform and integration scheme, the present model computes the orbital parameters at the start of the simulation. These parameters are functions of the lead time before impact and are based on an estimated post-2029 orbit of the asteroid Apophis, assuming passage through a resonant keyhole orbit. Since the orbit is calculated at run time using the adaptive method described in [6, 10], it also allows for changes to the orbital perturbations used in the model. The orbit used for a 15 day lead time impact is shown in Table 1.

Table 1. Orbital Parameters for 15 Day Impact Trajectory

Orbital Parameter	Value
Semimajor Axis	1.1082428 AU
Eccentricity	0.189928428
Inclination	2.18995362 deg
Longitude of Right Ascension	203.18642266 deg
Argument of Perihelion	69.929774 deg
Initial Mean Anomaly	296.74684241 deg
Epoch	64781 MJD
Miss Distance on Target Date	4.738466849E-011 Earth Radii

As described in [6], the present model computes the change of each fragment relative to the mean motion around the sun. Therefore, the nonlinear relative equations of motion are integrated to update the state of each fragment compared to a rotating coordinate system in orbit around the Sun (shown in Figure 3). Including the gravitational perturbations of the major planets, minor planets, and Earth's moon, the equations of motion are:



$\vec{i}$
x
LVLH Reference Frame (x, y, z)
Asteroid Fragments
$\vec{r} = x\vec{i} + y\vec{j} + z\vec{k}$
y
$\vec{j}$
c
$\vec{r}_c$
Elliptical Reference Orbit
θ
Sun
Perihelion
ω
Line of Nodes

Figure 3. Rotating Local Vertical Local Horizontal (LVLH) Frame.

$$
\begin{aligned}
\ddot{x}_i &= 2\dot{\theta}\Big(\dot{y}_i - \frac{\dot{r}_c}{r_c}y_i\Big) + \dot{\theta}^2 x_i + \frac{\mu}{r_c^2} - \frac{\mu}{r_d^3}(r_c + x_i) + \frac{\mu_E(x_E - x_i)}{r_{Ei}^3} + (F_x)_i \\
\ddot{y}_i &= -2\dot{\theta}\Big(\dot{x}_i + \frac{\dot{r}_c}{r_c}x_i\Big) + \dot{\theta}^2 y_i - \frac{\mu}{r_d^3}y_i + \frac{\mu_E(y_E - y_i)}{r_{Ei}^3} + (F_y)_i \\
\ddot{z}_i &= -\frac{\mu}{r_d^3}z_i + \frac{\mu_E(z_E - z_i)}{r_{Ei}^3} + (F_z)_i
\end{aligned}
\tag{1}
$$

where x_i, y_i, z_i, r_c, and θ are defined as shown in Figure 3; $r_d = |\vec{r}|$; μ and μ_E are the solar gravitational parameter and Earth's gravitational parameter.. Finally, r_{Ei} is the relative distance between each fragment and the Earth and F_i is the combined acceleration due to self-gravity and other gravitational perturbations. This term contains the effects of all major party gravitation besides that of Earth, and the positions of these bodies are computed at each time step as discussed in [6]. A fixed time step of 60 seconds is used in these computations to allow for all arrays to remain in GPU memory, rather than transferring the data to the host to determine an adaptive step size. The reason for this change will be discussed later in this paper.

Gravity Model

Since the simulation is predominantly meshless, a temporary grid is created between the minimum and maximum coordinate values in the LVLH frame. In the current model, the fragments are regarded as spherical so the grid spacing is generally at least 2.0 times the maximum particle radius. In most cases, the grid was allowed to be up to 20 times the particle radius, as grid size determined the precision of self-gravity perturbations. Referring to the two-dimensional grid in Figure 4, the colored grid cells adjacent to the grid containing the current fragment of interest are the cells in which self-gravity has the form:

$$
\begin{aligned}
(F_x)_i &= \sum_j \frac{Gm_j(x_j - x_i)}{r_{ij}^3} + \sum_k A_k \\
(F_y)_i &= \sum_j \frac{Gm_j(y_j - y_i)}{r_{ij}^3} + \sum_k A_k \\
(F_z)_i &= \sum_j \frac{Gm_j(z_j - z_i)}{r_{ij}^3} + \sum_k A_k
\end{aligned}
\tag{2}
$$

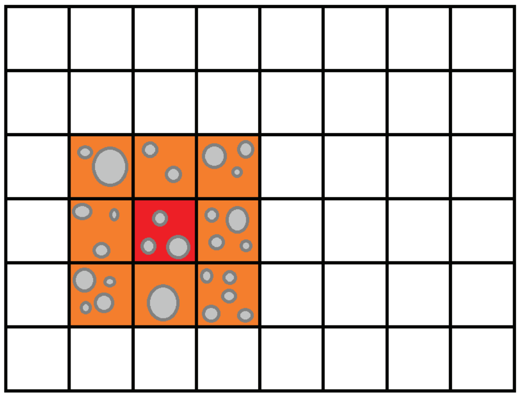


Figure 4. Temporary Grid Creation and Evaluation.

where G is the universal gravitational constant, m_j is the mass of the gravitating body, r_{ij} is the relative distance between each fragment and the gravitating body being considered, and A_k are the cell averages for all nonadjacent cells. Outside of this region, an averaged fragment containing the total mass contained within the cell is modeled at the centroid of the cell. Using this method, the expense of the self-gravity calculation is reduced while retaining an aggregate force for many small masses. Some methods for self-gravity neglect the force between bodies if it is sufficiently small. However, this cutoff is not well defined, and a large number of small fragments at a considerable distance produce an effect similar to a much larger body at the average distance of these fragments. Therefore, it would substantially alter the system dynamics to neglect this force rather than average it. The radius of cells in which self-gravity is considered is a user set parameter for the present model, usually set between 1 and 3. A block diagram of the process logic for a computing thread block containing fragment i is shown in Figure 5.

Collision Model

Collisions between bodies are a concern of any aggregation model. With a self-gravity model attracting each fragment to all others, there must be a physically realistic way of limiting the distance between two bodies that cannot occupy the same space. However, checking for collisions through brute force can be as expensive as the self-gravity model. A model excluding interacting pairs in adjacent cells is employed similar to that discussed in [17]. A Sort and Sweep algorithm checks for colliding pairs along each coordinate, excluding fragments from further consideration. When an overlapping pair of fragments are detected, they are backed up along the normal connecting their centroids. An inelastic collision process with a coefficient of restitution of 0.5 predicts the post-collision velocity, as shown in Figure 6.

Reentry Model

A density model was constructed based on a static Russian GOST atmosphere [18], with the coefficients listed in [6]. A static night-time atmosphere neglecting solar effects is assumed for altitudes above 120 km, using a solar flux value of $F_0 = 150$. This density then allows for a drag term to be added to the equation of motion, decelerating the fragment as it enters the atmosphere:

$$\dot{\mathbf{v}}_i = \dot{\mathbf{v}}_{ECI} - \frac{\rho(\mathbf{v}_i \cdot \mathbf{v}_i) S C_d}{2m_i} \hat{\mathbf{e}}_v \tag{3}$$

where $\dot{\mathbf{v}}_{ECI}$ is the acceleration of the fragment in the Earth Centered Inertial (ECI) frame, ρ is the density as a function of altitude, S is the exposed area, $\hat{\mathbf{e}}_v$ is the unit vector in the velocity direction, and $C_d = 1.7$ is

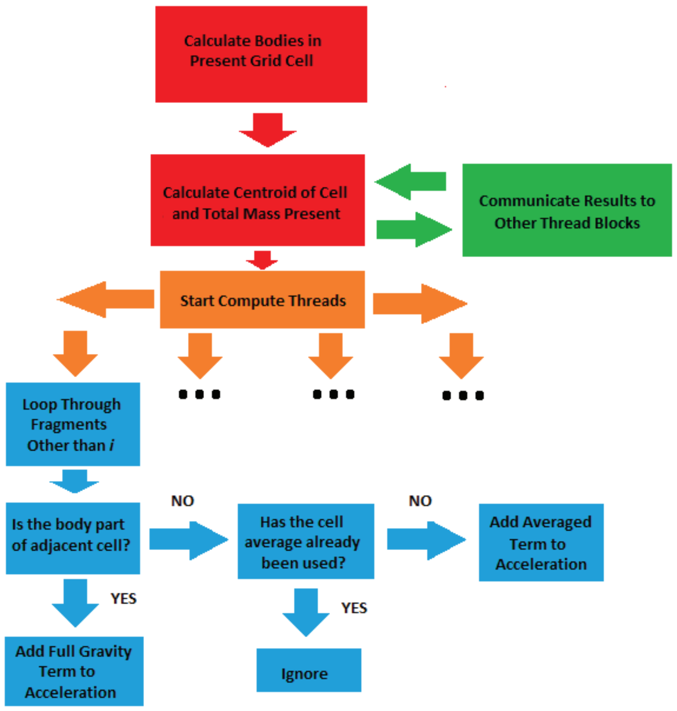

Calculate Bodies in Present Grid Cell
Calculate Centroid of Cell and Total Mass Present
Communicate Results to Other Thread Blocks
Start Compute Threads
Loop Through Fragments Other than *i*
Is the body part of adjacent cell?
NO
Has the cell average already been used?
NO
Add Averaged Term to Acceleration
YES
Add Full Gravity Term to Acceleration
YES
Ignore

Figure 5. Block Diagram for Self-Gravity Process Logic (Contains Fragment i).

the ballistic drag coefficient for a cylinder (the shape model assumed) [19]. Pressure stress and mass loss due to ablation are modeled using material parameters:

$$\dot{m}_i = -\frac{S}{Q} \min\left(\frac{1}{2} C_H \rho v_i^3, \sigma T^4\right) \tag{4}$$

where Q is the heat of ablation (assumed to be 1E7 J/m^3), C_H is the coefficient of heat transfer (assumed constant at 0.1), σ is the Stefan-Boltzmann constant, and T is the temperature of thermal ionization of the surrounding gas (25,000 K). This equation governs the ablative mass loss until the mean pressure in the cylinder $p = 0.25 C_d \rho v^2$ exceeds the yield strength of the material, at which point the energy deposition implies burnup of the fragment.

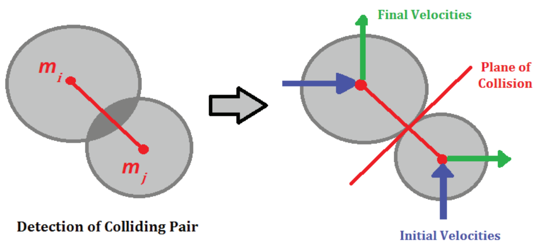


Figure 6. Collision Detection and Evaluation Process for Interacting Pair.

COMPUTATIONAL APPROACH

This section address the computational approach used to solved the problem. Each state variable update for a fragment is conducted in parallel at each time step. The various sets of hardware used for computation and comparison are given. We present our model for host and GPU memory, and how we limited explicit communication along the hardware bus. Approaches for limiting bandwidth use are discussed, followed by integrating multi-core CPU computation with the GPU executables.

Hardware and Implementation

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 [12]. These cost estimates are used to determine hardware performance on the various systems. A summary of the hardware used is shown in Table 2 (Note: all CPUs are Intel brand, and all GPUs are NVIDIA brand).

Table 2. 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

The implementation of the simulation is conducted in two ways. The first version uses CUDA extensions to the C language, and bindings for these kernels into existing Fortran 90 code. The second version uses CUDA Fortran, developed by the PGI group [20].

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 model is predominantly bandwidth-limited for small data sets. The final algorithm is shown in block diagram form in Figure 7.

Memory Model and Explicit Communication

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. Figure 8 shows an overview of this computational memory model.

The explicit communication needed in the simulation is shown in the following psuedocode:

```
*Transfer state variable arrays Host to Device
Begin loop through time steps
    Calculate planetary positions at subintervals on host
```

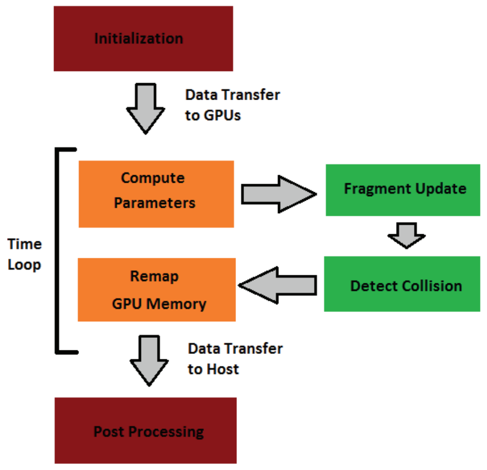


Figure 7. Block Diagram of Simulation Procedure.

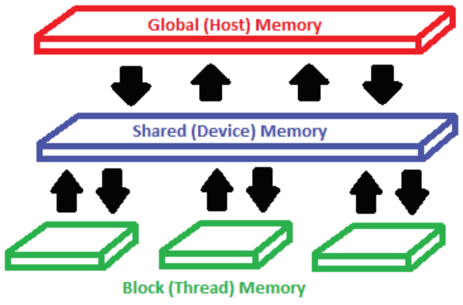


Figure 8. Visualization of Memory Model.

```
        *Transfer planet position arrays Host to Device
        Launch GPU kernels to update subset of state variables
        Calculate closest approach to Earth on GPU
        *Transfer updated array slices to Host
        BARRIER
        *Transfer fully updated arrays to each device
    End time step loop
    *Transfer final array slices back to Host, including close approach
    Postprocessing on Host using all device data
```

Bandwidth Use and Serial Computation

One of the primary limitations of GPU acceleration is the PCI bus connecting the Host to the Device. Communication between these two sets of memory, or between individual GPUs, is therefore very expensive [12, 20]. Therefore, the use of communication bandwidth should be minimized to achieve optimal performance. For example, an early implementation of a numerical integrator may be a subroutine that reads state variables from Host memory, computes the updated state on the GPU, and returns the next state to the Host. Unless the computation of the updated state is extremely intensive, this approach will not yield a high speedup over a CPU implementation. For smaller problems, the approach discussed above is preferred. Though the device code may be more complicated, this method was found to be an order of magnitude faster for an NEO system with 18,220 fragments. With 511,744 fragments, the limitation from data transfer is less pronounced, though leaving arrays in device memory yielded a simulation that ran 5 times faster.

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 resulting arrays can be quite large. Each model of GPU has a limited number of memory registers available to each computing block of threads [12]. 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 trades off directly with the added expense of calculating parameters on the Host at each timestep. For the present work, calculating planetary positions at each step was found to be preferable to using a large pre-calculated array. For some hardware, sufficient GPU memory was not available.

Integration with SMP Computation

Some of the systems used to test this work had multiple GPUs. This was used to the advantage of the program by launching several Shared Memory Parallel (SMP) threads on the Host CPU. Each thread, or team of threads, was assigned a GPU on which to launch compute kernels. The calculations were conducted on portions of the state variable arrays. Since no interaction among fragments was assumed, the GPUs did not have to communicate the states of the fragments they were responsible for. This was found to be an extremely effective setup for large data sets, and scaled almost linearly to the number of GPUs used with some overhead for data transfer to partial arrays on the GPUs.

RESULTS

This section discusses the present results. Earth distance histograms for sample cases including self-gravity are given, and we address the level of computation needed to obtain them. Computational speedup relative to CPU implementations are given for various codes used, and we briefly present our efforts at determining scalability of the present model. Finally, we address limitations unique to this type of architecture.

Self-Gravity Test Cases

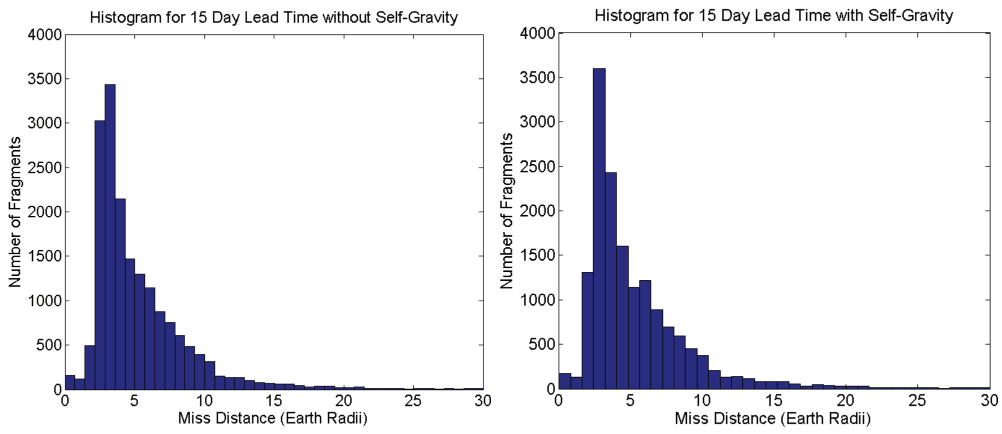


Figure 9. Comparison of Miss Distance Including Self-Gravity.

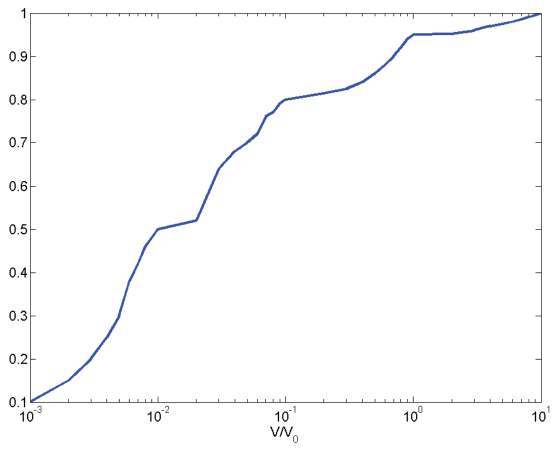


Figure 10. Ratio of Mean Radius Including Self-Gravity.

Test cases were run for a 15-day lead time including self-gravity. The baseline model calculated 196 impacts with Earth, with a total impacting mass of 1.18%. Including self-gravity, the ratio was a higher value of 1.66%, as expected. However, the total number of impacts predicted was 188. This indicates that some fragments that were predicted to hit the Earth were perturbed off of impacting trajectories, but some higher mass fragments were perturbed into orbits impacting the Earth. The miss distance histograms for these two cases can be observed in Figure 9. More complicated dynamics was observed, similar to that of isotropic distributions in [16]. However, this has occurred on a much shorter time scale. With 15 day lead time testing of fragments on impacting trajectories is not a good measure of system dynamics with slow initial velocity, as most particles will remain within 1 Earth Radius of the center of mass. The size of the debris cloud in these cases are measured using the median radial distance in Figure 10. As expected, the influence on median radial distance decreases with slower initial velocity. This indicates an exchange of momentum in the debris cloud that cannot be discounted. Again, this model statistically describes a fragmented NEO, so further study must be done using different input conditions to determine the effect on asteroid deflection mission planning. One case with subcritical input conditions was run, scaling the median initial velocity to 0.3 m/s. Reaggregation of these fragments similar to that of [14] was observed. These results are shown in Figure 11, where orange and red material reflects fragments falling inward towards the center of mass 22,000 seconds after disruption. As collisions dissipate the energy of the inward falling fragments, the velocities drop to near zero as a smaller body is recreated 40,000 seconds after disruption.

Speedup and Scalability

Speedups of 25x to over 120x were observed for an NEO disruption with 18,220 fragments. These are shown in Figure 12. The benefits of this approach for large data sets are also shown in Figure 12, in which an NEO system with over 500,000 fragments had computational speedups of over 500x.

This problem was found to be highly scalable, with multiple GPUs allowing large data sets to be broken up into manageable pieces. The speedup factors using Machine 5 on various data set sizes is shown in Figure 13.

Limitations and Future Work

One limitation of the present work is the need to compute system parameters at each time step. Surmounting this difficulty would yield large gains in performance by eliminating data transfer dependency on serial computation at each step. The GPU computing model is not applicable to highly serial problems, and works best for distributed data sets like this fragmented NEO model. More applications of this technology continue to be discovered and applied.

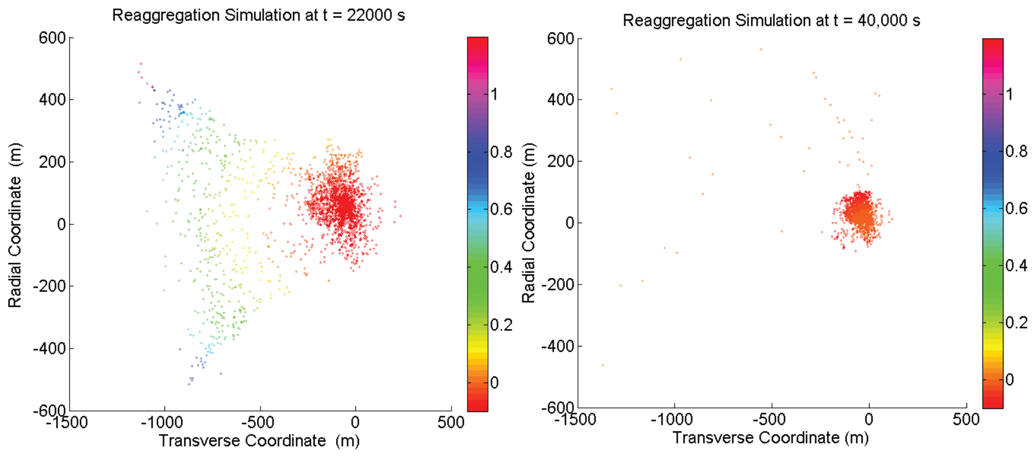


Figure 11. Reaggregation of Subcritical NEO Disruption.

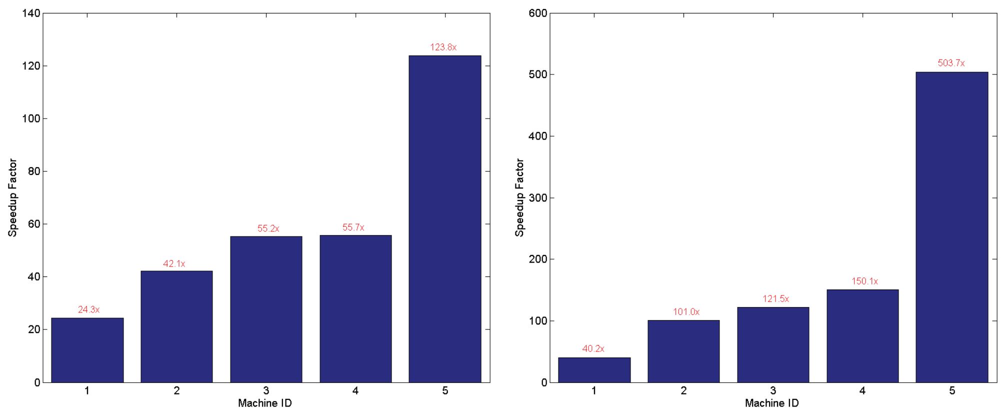


Figure 12. Computational Speedup Observed for Small and Large Data Sets.

Scalability for multiple GPU modeling appears to be bandwidth-limited for large data sets. Communication requirements at each time step for self-gravity models restrict favorable implementation of the simulation to a single node at this time. However, the larger the data set, the less time is spent in communication compared to the gravity and collision calculations.

Gravity and collision calculations are conducted by threads on contiguous sets of fragment ID numbers. No emphasis is placed on computing fragments that are physically close to one another, though this would be preferable for collision assessment. A study is currently being conducted on the implications of this computing strategy, and how neighbors would be determined in an efficient matter. Further work is planned on understanding the relationship between self-gravity and debris cloud shape under anisotropic mass and velocity conditions.

CONCLUSION

Using the computational power of the GPU to accelerate NEO deflection simulation has resulted in the ability to tackle large scale self-gravity simulations on a reasonable time scale. High speedup factors reduce most simulations to less than an hour, allowing researchers to more quickly check multiple cases.

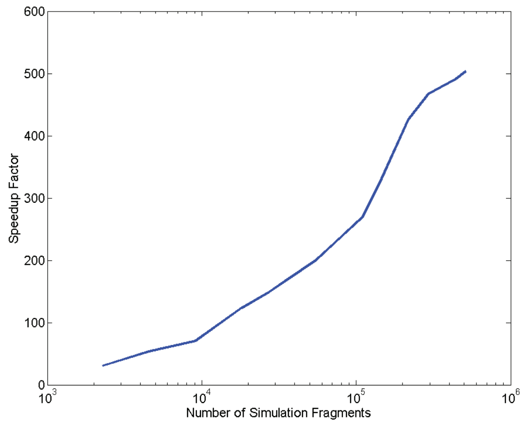


Figure 13. Computational Speedup for Varied Data Set Size (Machine 5).

Decreasing initial fragment velocity was shown to reduce the median debris cloud size under self-gravity including a dissipative collision model reducing the energy of the system. This indicates that a fragmented NEO under self-gravity leads to complicated dynamics, especially in the presence of Earth and the Moon, and cannot be neglected.

This technology has allowed larger data sets to be tested for NEO disruption simulation. A 511,744 fragment model of a 1 km body was tested and shown to have a computational speedup of over 500x compared to serial implementation. The GPU accelerated simulation was shown to scale well to number of NEO fragments, achieving maximum speedup near the 511,744 fragment benchmark. This project is an example of high fidelity computing on a budget, allowing almost 1/3 the performance of a $1.25 Million supercomputer for just over $14,000 (the cost of Machine 5 at time of writing). Reduction of the cost of HPC has allowed academic planetary defense research to verify and complement work at major research labs.

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 initial disruption model is courtesy Dr. David Dearborn (LLNL). The authors would like to thank Dr. Ramanathan Sugumaran (Director, ISGC) for his support of this research work, and Professor Z.J. Wang for testing on his GPU server.

REFERENCES

[1] NASA Jet Propulsion Laboratory, Near Earth Object Program, http://neo.jpl.nasa.gov/neo/pha.html

[2] National Research Council, *Defending Planet Earth: Near-Earth Object Surveys and Hazard Mitigation Strategies*, Final Report, Committee to Review Near-Earth Object Surveys and Hazard Mitigation Strategies, The National Academies Press, January 2010.

[3] Kaplinger, B.D., and Wie, B., "A Study on Nuclear Standoff Explosions for Deflecting Near-Earth Objects," 1st IAA Planetary Defense Conference, April 2009.

[4] NASA/JPL Near-Earth Object Program Office, "Two Small Asteroids to Pass Close by Earth on September 8, 2010," http://neo.jpl.nasa.gov/news/news169.html, September 7, 2010.

[5] Wie, B., "Solar Sailing Kinetic Energy Impactor Mission Design for Impacting and Deflecting Near-Earth Asteroids" White Paper No. 009, presented at NASA Workshop on NEO Detection, Characterization, and Threat Mitigation, Vail, CO, June 26-29, 2006.

[6] 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.

[7] Dachwald, B., Kahle, R., and Wie, B., "Solar Sailing KEI Mission Design Tradeoffs for Impacting and Deflecting Asteroid 99942 Apophis," AIAA-2006-6178, AIAA/AAS Astrodynamics Specialist Conference, Keystone, CO, August 21-24, 2006.

[8] 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.

[9] Sanchez, J.P., Vasile, M., and Radice, G., "On the Consequences of a Fragmentation Due to a NEO Mitigation Strategy," IAC-08-C1.3.10, 59th International Astronautical Congress, Glasgow, UK., 29 Sept - 3 Oct., 2008.

[10] Kaplinger, B.D., Wie, B., and Dearborn, D., "Efficient Parallelization of Nonlinear Perturbation Algorithms for Orbit Prediction with Applications to Asteroid Deflection," AAS 10-225, 20th AAS/AIAA Space Flight Mechanics Meeting, San Diego, CA, February 15-17, 2010.

[11] Kirk, D.B., and W.W. Hwu, *Programming Massively Parallel Processors: A Hands-On Approach*, Morgan Kaufmann: Burlington, MA, 2010.

[12] NVIDIA Corporation, *NVIDIA CUDA C Programming Guide*, v3.1, May 28, 2010.

[13] NVIDIA Corporation, *OpenCL Programming Guide for the CUDA Architecture*, v3.1, May 26, 2010.

[14] Korycansky, D.G., and Plesko, C.S., "Reaggregation Times of Potentially Hazardous Object Fragments After a Hazard Mitigation Impulse," #1456, 41st Lunar and Planetary Science Conference, The Woodlands, TX, March 1-5, 2010.

[15] 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.

[16] Kaplinger, B.D., and B. Wie, "Orbital Dispersion Simulation of Near-Earth Object Deflection/Fragmentation by Nuclear Explosions," IAC-09-C1.10.2, 60th International Astronautical Congress, Daejeon, Republic of Korea, October 12-16, 2009.

[17] Le Grand, S., "Broad-Phase Collision Detection with CUDA," *GPU Gems 3*, Ed. Hubert Nguyen, Addison-Wesley Professional, August 12, 2007.

[18] Vallado, D.A., *Fundamentals of Astrodynamics and Applications*, Third Edition, Microcosm Press, Hawthorne, CA, 2007.

[19] Chyba, C.F., Thomas, P.J., and Zahnle, K.J., "The 1908 Tunguska Explosion: atmospheric disruption of a stony asteroid," *Nature*, v361, p. 40-44, 1993.

[20] The Portland Group, *CUDA Fortran Programming Guide and Reference*, Release 2010.