Project Proposal

CUDA-Based N-Body Simulation

Students: mohammad babakhani-bita bagheri

---

1. Project Overview and Objectives

This project implements a **from-scratch N-body simulation using CUDA**, focusing on both **physically reasonable behavior** and **GPU performance scalability** for systems with all-to-all interactions (**$O(N^2)$** per timestep).

The simulator models a simplified **gravitational system inspired by the solar system**, extended with a comet and many small bodies. The objectives are:

- correct modeling of gravitational interactions and time evolution,
- efficient CUDA-based pairwise force computation,
- clear visual output of system behavior,
- and quantitative analysis of performance scaling with increasing N.

2. Simulated System and Features

2.1 Body Representation

Each simulated object is a point mass defined by:

- Position: $\mathbf{r_i = (x_i, y_i, z_i)}$
- Velocity: $\mathbf{v_i = (v_{xi}, v_{Y\,i}, v\_z_i)}$
- Mass: $\mathbf{m_i}$

Bodies conceptually include a dominant **Sun**, several **planets**, a **comet**, and many **minor bodies**. All objects follow identical physical rules; categories are used only for initialization and visualization.

2.2 Interaction Model (Forces)

The only interaction between bodies is **gravitational attraction**, modeled using Newtonian gravity with softening.

For each body i, the total acceleration is:

$$\mathbf{a}i = \sum j \neq iG\,m_j \frac{\mathbf{r}_j - \mathbf{r}_i}{\left(\|\mathbf{r}_j - \mathbf{r}_i\|^2 + \epsilon^2\right)^{3/2}}$$

Each object therefore experiences forces from **all other objects**. The softening parameter **ε** prevents numerical instability during close encounters. Units and constants may be normalized for numerical convenience.

2.3 Collision and Close-Encounter Handling

Bodies are treated as **point masses**.

- **No physical collisions** (merging or bouncing) are modeled.
- During close approaches, the softening term limits force magnitude, allowing bodies to pass through each other while experiencing strong gravitational deflection.

This choice simplifies implementation and keeps the focus on performance.

3. Time Evolution

The simulation advances in discrete timesteps **Δt** using a **Leapfrog (Velocity-Verlet)** integrator.

At each timestep:

1. Accelerations are computed from all pairwise interactions.
2. Velocities are updated by half a timestep.
3. Positions are updated.
4. Accelerations are recomputed.
5. Velocities are finalized.

This method offers improved stability over Euler integration with minimal additional cost.

4. CUDA-Based Computational Approach

4.1 Parallelization Strategy

- **One CUDA thread per body**.
- Each thread computes the total force acting on its assigned object.
- The entire simulation loop executes on the GPU.

4.2 Kernel Structure

Three kernels are implemented:

1. **Acceleration kernel ($O(N^2)$)** for pairwise gravitational interactions.
2. **Velocity update kernel ($O(N)$)**.
3. **Position update kernel ($O(N)$)**.

4.3 Memory Optimization

To reduce global memory traffic:

- Bodies are processed in **tiles**.
- Each block loads a tile into **shared memory**.
- Threads accumulate interactions from the shared tile before proceeding.

This improves memory reuse and bandwidth efficiency.

---

5. Visualization and Output

5.1 Visual Output

The simulation provides a visual representation showing:

- Stable planetary orbits around the Sun,
- A comet on an elongated orbit being deflected by planets,
- Clouds or belts of minor bodies.

Objects are rendered as points or small spheres, optionally color-coded by type.

5.2 GUI and Interaction

A lightweight visualization interface allows:

- real-time or near-real-time observation,

- camera zoom and pan,
- optional toggles (e.g., show/hide minor bodies or trails).

The GUI is designed for inspection and does not interfere with performance measurements.

---

6. Expected Outputs

The project produces:

- animated visualization of system evolution,
- trajectory plots or logged position data,
- performance measurements per timestep.

7. Evaluation and Metrics

7.1 Performance Metrics

- Time per timestep (CUDA events),

$$Interactionspersecond : \text{IPS} \approx \frac{N(N-1)}{t_{\text{step}}}$$

7.2 Correctness Indicators

- Long-term stable planetary orbits,
- Clear comet deflection near massive bodies,
- Absence of numerical instability.

8. Scope Limitations

- $O(N^2)$ complexity limits maximum N (≈16k–32k),
- numerical stability depends on Δt and ε,
- visualization is diagnostic rather than a full-featured GUI.