# Simulating a dark matter halo using a parallel grid method

*Otto Hannuksela* *Janne Lampilahti*

May 31, 2015

## 1 INTRODUCTION

Dark matter appears to be the dominant form of matter in the universe. While no direct observations of dark matter exist, it has been observed indirectly through its gravitational interaction with visible matter and radiation (**?**). The effect of dark matter on structure formation in the universe is the subject of an increasing scientific interest and a useful tool in the search for possible candidates of dark matter. Current progress in the investigation of structure formation is mainly driven by advances in computational methods and capabilities (**?**).

In this work we simulate a dark matter halo, neglecting baryonic matter, by using a particle based method and solving the Poisson's equation of gravity. The software used is *Uintah* available from `http://uintah.utah.edu/`, a parallel grid framework with a support for particle interactions and adaptive mesh refinement and which is aimed at solving partial differential equations. A major part of the project was to learn how to use the tools provided by Uintah.

## 2 METHODS

### 2.1 Theory

A distribution of mass density $\rho$ gives rise to a gravitational potential $\phi$ according to the Poisson's equation of gravity

$$\nabla^2\phi = 4\pi G\rho, \tag{1}$$

where $G \approx 6.674 \times 10^{-11}$ $\mathrm{Nm}^2/\mathrm{kg}^2$ is the gravitational constant. In our simulation we normalize $G = 1$. The corresponding force field can be solved from the gradient

$$\mathbf{F} = -\nabla\phi. \tag{2}$$

In our simulation a distribution of massive particles create the mass density. A new position and velocity for the particles after a time step $dt$ can be solved from the Newton's equation of motion.

$$\mathbf{v}(t+dt) = \int_t^{t+dt} \frac{\mathbf{F}(t')}{m} dt' + \mathbf{v}(t) \tag{3}$$

$$\mathbf{x}(t+dt) = \int_t^{t+dt} \mathbf{v}(t')dt' + \mathbf{x}(t) \tag{4}$$

The use of this theory assumes that we do not have relativistic speeds or masses and that the maximum grid size is small enough that expansion of the universe can be neglected.

One approach would be to try to calculate the particle-particle interactions but this is not computationally feasible since we would have $2^N$ interactions where $N$ is the particle number and we are using thousands of particles.

### Numerical methods

The simulation is set up with respect to a three dimensional grid that supports particles (Fig. 1).

The overall algortihm is expressed in Algortihm 1.

**Algortihm 1.** Main program.

set Dirichlet or periodic boundary conditions
set initial $\mathbf{x}_p$, $\mathbf{v}_p$ and $m_p$ for all particles $p$
interpolate initial $\rho$ from particle mass $m_p$
set initial guess for $\phi$
**loop**
    solve $\phi$ at the nodes using the SOR algorithm
    calculate $\mathbf{F} = -\nabla\phi$ with respect to each particle $p$
    **for** every particle $p$ **do**
        $\mathbf{v}_p(t + \Delta t) = (\mathbf{F}_p/m)\Delta t + \mathbf{v}_p(t)$
        $\mathbf{x}_p(t + \Delta t) = \mathbf{v}_p(t)\Delta t + \mathbf{x}_p(t)$
    **end for**
    interpolate new $\rho$ from particle mass $m_p$
    $t \leftarrow t + \Delta t$
**end loop**

The Poisson's equation of gravity is discretized to

$$\begin{aligned} 4\pi G\rho_{i,j,k} = &\frac{\phi_{i+1,j,k} - 2\phi_{i,j,k} + \phi_{i-1,j,k}}{(\Delta x)^2} + \\ &\frac{\phi_{i,j+1,k} - 2\phi_{i,j,k} + \phi_{i,j-1,k}}{(\Delta y)^2} + \\ &\frac{\phi_{i,j,k+1} - 2\phi_{i,j,k} + \phi_{i,j,k-1}}{(\Delta z)^2}. \end{aligned} \tag{5}$$

The gravitational potential $\phi$ is then solved at each node using using the successive over-relaxation (SOR) algortihm (Algorithm 2) . In the calculation of the potential gradient we obtain the potential values near the particles by interpolation and then calculate the gradient by numerical differentiation. For exmaple at the $p$th particle the gradient is calculated in the $x$ direction as

$$(\nabla\phi)(x_p) = \frac{\phi(x_p + dx) - \phi(x_p - dx)}{2dx}, \tag{6}$$

where $x_p$ is the particle's $x$ coordinate and $dx$ is a small distance.

The particle velocity and position are evolved over a small time step $dt$, assuming that the force remains constant. Essentially this means calculating

for each particle

$$= \mathbf{v}_p(t + \Delta t) = (\mathbf{F}_p/m)\Delta t + \mathbf{v}_p(t) \tag{7}$$

$$= \mathbf{x}_p(t + \Delta t) = \mathbf{v}_p(t)\Delta t + \mathbf{x}_p(t). \tag{8}$$

After this using the new positions the particle masses are interpolated back to the nodes to obtain a new mass density $\rho$.

**Algortihm 2.** Calculating potential $\phi$ using the SOR algorithm.

**function** SOR($\phi$, tolerance, max_iterations)
  **for** $n = 0, 1, \ldots$ max_iterations **do**
    error $\sigma \leftarrow 0$
    **for** every node $\phi_{i,j,k}$
      $\phi_{i,j,k}^{(n+1)} \leftarrow (1-\omega)\phi_{i,j,k}^{(n)} + \frac{\omega}{6}(\phi_{i+1,j,k}^{(n)} + \phi_{i-1,j,k}^{(n)} +$
        $\phi_{i,j+1,k}^{(n)} + \phi_{i,j-1,k}^{(n)} + \phi_{i,j,k+1}^{(n)} + \phi_{i,j,k-1}^{(n)} +$
        $h^3\rho_{i,j,k})$
      update $\sigma$
    **end for**
    **if** $\sigma \leq$ tolerance, **break**
  **end for**
  **return** $\phi$
**end function**

### 2.2 Initialization of the simultaion

The specific case we want to study with our simulation is a dark matter halo. Dark matter halos are structures composed of dark matter, believed to envelope galaxy disks. We initialize the halo by letting a sphere of randomly distributed dark matter particles to collapse by the effect of gravity.

### 2.3 Verification of results

To verify the simulation results we test for energy conservation, compliance with the virial theorem

$$\langle T \rangle_t = -\frac{1}{2}\langle V \rangle_t. \tag{9}$$

Previous $N$-body simulations suggest that dark matter halos follow the

Navarro-Frenk-White radial mass distribution

$$\rho(r) = \frac{\rho_0}{\frac{r}{R_s}\left(1 + \frac{r}{R_s}\right)^2}, \tag{10}$$

where $r$ is distance from the center of the halo and $(\rho_0, R)$ are parameters. As part of our verification procedure we fit this distribution to the simulated dark matter halo.

## IMPLEMENTATION

In this section, we will often refer to Section *Numerical methods* and describe each numerical method's implementation in detail. During the whole project, we made a lot of effort to make sure the algorithms fit the Uintah framework. Because of this, we will have extensive description of the Uintah framework in the Section *Uintah*, describing how the Uintah patches which are used in the Poisson solver work, how particle masses are interpolated to the grid and how the gradient of potential is calculated.

Additionally, we will discuss some of the limitations and bugs of Uintah software which drained a painstaking amount of time from us. For more information on Uintah framework, please refer to the *uintah_presentation.pdf* we made on the software.

### Uintah

As mentioned in the title, we made a *fully parallel* poisson solver utilizing particle-in-cell approach.

Before we start with the actual algorithms, let's go through the basics of Uintah. There are a few concepts to understand; *cells*, *patches*, *ghost cells*, *cell-centered variables*, *node-centered variables*, *face-centered variables* and *particles*.

We present an explanation for each of these accompanied by a Figure (Figure 1) demonstrating a visual description.

1. *Cells* are finite boxes in the grid. These are the basic elements of our simulation. In the case of our poisson solver, we discretize our poisson equation based on cells in such a way that our discretization length $h$ in Equation **??** is the length between *cells*.

2. *Patches* are collections of *cells* in our grid. The reason to have a concept of *patches* is that Uintah uses patches to distribute workloads for different processes. In Uintah, whenever we iterate through *cells*, we iterate through them patch-by-patch. This introduces some complications, such as how to handle the boundary between patches. The boundary issue will be further elaborated on.

3. *Ghost cells* are cells which are one or more step beyond the border of a *patch*. The whole concept of ghost cells is meaningless if we do not work in parallel. In a parallel simulation (employing more than one process), having ghost cells is of utmost importance because it is used to handle *MPI communication* between different processes' *patches*.

4. *Cell-centered variables* are variables which are defined at the center of each *cell*.

5. *Node-centered variables* are variables which are defined at the corners of each *cell*.

6. *Face-centered variables* are variables which are defined at the faces of each *cell*.

7. *Particles* are point variables defined at any arbitrary point in the simulation grid. In our simulation, cell/node/face-centered variables interact with each other by interpolating particle variables to cell/node/face-centered positions and by interpolating cell/node/face-centered variables to particle positions.

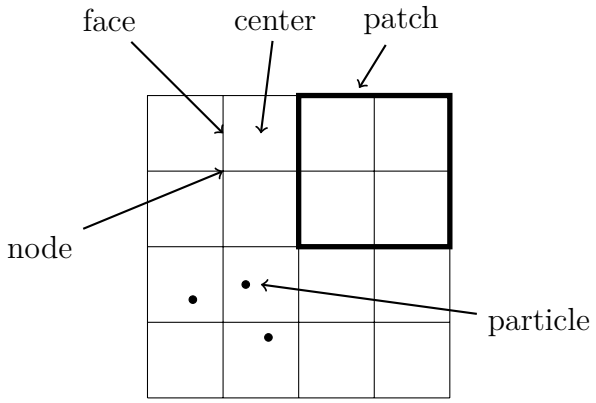


**Figure 1.** A two dimensional view of the grid with its various components and properties shown.

One of the biggest reasons we chose to use the Uintah framework was that it has a very advanced *parallel task system*. As already mentioned previously, in Uintah parallelization goes over patches. Every *process* has an N amount of patches which they are in charge of, and those patches are distributed according to load balancing settings. In our case, we chose to use a load balancing option that takes the number of particles into account.

The parallel task system takes care of most data communication and working with data dependencies. For each function, we must define a task which takes in the *function*, the function's *dependencies* and the function's *output*.

For instance, as we can see in the Equation **??**, for one poisson iteration, we need the *last iteration's* $\phi = \phi_{previous}$ value as well as the distribution of particle density $\rho$, and one poisson iteration calculates the next $\phi = \phi_{next}$ value.

In pseudocode, this would be:

$Task \leftarrow$new task($poissonIteration$)
$Task \leftarrow \phi_{previous}$
$Task \leftarrow \rho_{current}$
$Task$ computes $\phi_{next}$
$Scheduler \leftarrow Task$

**The program**

RESULTS

3 CONCLUSIONS