

# 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 (Roos, 2012). 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 (Kuhlen et al., 2012).

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/> during the writing of this report), 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

In this section the relevant theory and numerical methods are described. The method in short was to use a Poisson solver operating on a grid and an  $N$ -body integrator operating on particles, that are coupled via interpolation between the grid and the particles. This method is used in the context of massive collisionless particles interacting through gravity (e.g. Hockeny and

Eastwood, 1985).

## 2.1 Theory

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

$$\nabla^2 \phi = 4\pi G \rho, \quad (1)$$

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

$$\mathbf{F} = -\nabla \phi. \quad (2)$$

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.

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

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

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.

## 2.2 Numerical methods

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

**Algorithm 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 the successive over-relaxation (SOR) algorithm (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 example 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

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

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

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

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

```

function SOR( $\phi$ , tolerance, max_iterations)
  for  $n = 0, 1, \dots$  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

```

### 2.3 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 randomly distribute dark matter particles into a box and let them collapse by the effect of gravity. We used either Dirichlet boundary conditions with  $\rho = 0$  or periodic boundary conditions.

### 2.4 Verification of results

To verify the simulation results we test for energy conservation:

$$T + V = \text{constant} \quad (9)$$

$$\frac{1}{2} \sum_p m_p v_p^2 + M \sum_{i,j,k} \phi_{i,j,k} = \text{constant} \quad (10)$$

where  $T$  is the total kinetic energy,  $V$  is the total potential energy and  $M$  refers to the total mass of the particles. In the context of gravitational interactions only, the virial theorem is:

$$\langle T \rangle_\tau = -\frac{1}{2} \langle V \rangle_\tau \quad (11)$$

where we take a time average of the kinetic and potential energy over a time period  $\tau$ .

We could not use time steps, distances or masses with proper units because the Uintah interpolator had some problems with large numerical values (see section ??). Therefore the set of values in our simulation have arbitrary units. To test energy conservation and virial theorem we solved a scaling factor  $c$  from the virial theorem over some time interval  $\tau_1$ . The scaling factor relates the kinetic energy  $T$  and potential energy  $V$  of the system via  $\langle T \rangle_{\tau_1} = c \langle V \rangle_{\tau_1}$ .

Then we tested the virial theorem over some other time interval  $\tau_2$  as well as energy conservation to see whether they are satisfied when we use the scaling factor  $c$ .

We also compare the radial mass distribution of the obtained halo with the Navarro-Frenkel-White (NFW) profile, which is the radial mass distribution often obtained from simulations (Navarro et al., 1995). The profile is given by

$$\rho(r) = \frac{\rho_0}{\left(1 + \frac{r}{R_s}\right)^2}, \quad (12)$$

where  $r$  is distance from the center of the halo and  $(\rho_0, R)$  are parameters.

### 3 IMPLEMENTATION

In this section, we will often refer to Section 2.2 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 3.1, 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.

### 3.1 Uintah

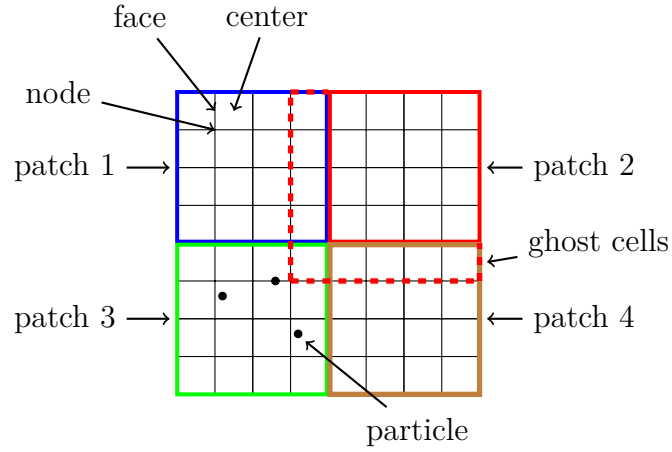
As mentioned in the title, we made a *fully parallel* poisson solver utilizing particle-in-cell approach.

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.

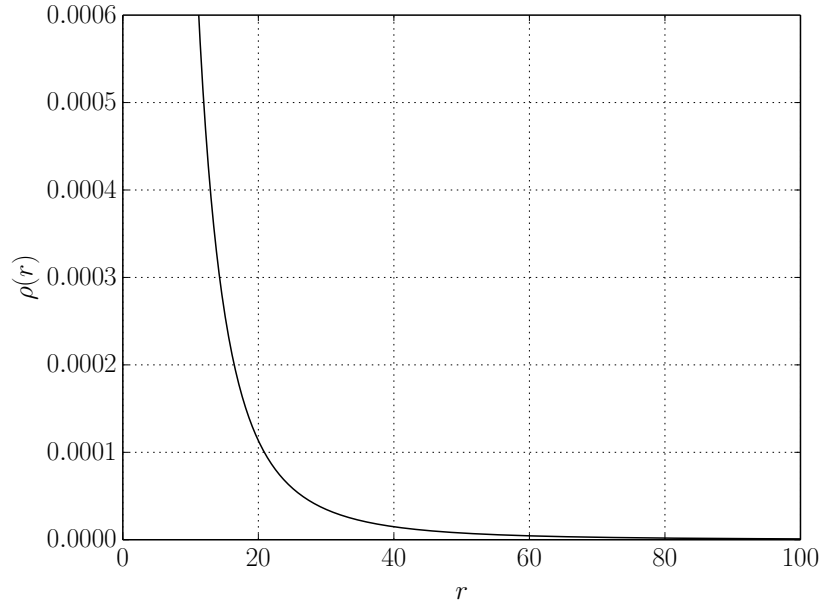


**Figure 1.** A two dimensional view of the grid with its various components labeled.

Throughout our simulation,

### 3.2 The program

## 4 RESULTS



**Figure 2.** The shape of the radial distribution of mass density suggested by NFW profile (Equation 12), choosing  $\rho_0 = 1$  and  $R_s = 1$ .

## 5 CONCLUSIONS

## REFERENCES

- Kuhlen, M. et al., (2012), Numerical Simulations of the Dark Universe: State of the Art and the Next Decade, arXiv:1209.5745 [astro-ph, physics:hep-ph]
- Roos, M., (2010), Dark Matter: The evidence from astronomy, astrophysics and cosmology, arXiv:1001.0316 [astro-ph]
- Hockney, R.W. and Eastwood, J.W., (1985), Computer Simulation Using Particles, CRC Press



Navarro, F.N. et al., (1995), The Structure of Cold Dark Matter Halos,  
arXiv:astro-ph/9508025