Journal of Physics: Conference Series **454** (2013) 012014 doi:10.1088/1742-6596/454/1/012014

Studying the core-cusp problem in cold dark matter halos using N-body simulations on GPU clusters

Go Ogiya[1], Masao Mori[1], Yohei Miki[1], Taisuke Boku[1] and Naohito Nakasato[2]

[1] University of Tsukuba, Tsukuba, Ibaraki, Japan
[2] University of Aizu, Aizu-Wakamatsu, Fukushima, Japan

E-mail: ogiya@ccs.tsukuba.ac.jp

Abstract. The discrepancy in the mass–density profile of dark matter halos between simulations and observations, the core–cusp problem, is a long–standing open question in the standard paradigm of cold dark matter cosmology. Here, we study the dynamical response of dark matter halos to oscillations of the galactic potential which are induced by a cycle of gas expansion and contraction in galaxies driven by supernova feedback. We developed a fast tree–code for PC clusters with GPU which displays high performance and high scalability. We perform large scale N–body simulations to follow the dynamical evolution of dark matter halos under the effect of oscillating gravitational potential. Furthermore, we compare the results of simulations with an analytical model of the resonance between particles and density waves to understand the physical mechanism of the cusp-core transition.

1. Introduction

The core–cusp problem remains one of the main unresolved contradictions between observation and theory predicted by the standard paradigm of the cold dark matter (CDM) cosmology. ΛCDM–cosmological N–body simulations have always predicted a steep power-law mass–density profile at the center of CDM halos [1], [2], [3]. However, recent observations of lower mass galaxies have revealed that the density profile of the dark matter (DM) halo is nearly constant around the center [4], [5], [6].

Mashchenko et al. [7] proposed that periodic variations in the galactic potential driven by recurrent star burst events may flatten the central DM density profile. Bursts of star formation induce a large–scale outflow from the galactic center. In time, the mass–loss leads to gas depletion and temporarily terminates star formation. Subsequently, the expelled gas falls back toward the galactic center, loses a large amount of internal energy by radiative cooling, and, once sufficient cold gas has accumulated, a starburst arises again. This cycle of expansion and contraction of the interstellar gas could lead to a cyclic change in the gravitational potential around the center of galaxies and have an impact on the density profile of DM halos. Some cosmological hydrodynamic simulations exhibit this starburst–outflow cycle and also observed a cusp–core transition at the center of CDM halos [7], [8], [9]. However, the physical connection between these two phenomena is still unclear.

We investigate the dynamical response of DM halos with a central cusp to a recurring change of the baryon gravitational potential with collisionless N–body simulations. To this end, we have developed a fast tree–code on PC clusters with GPU, which displays high performance and

high scalability parallel processing. In section 2, we briefly describe the tree method. In section 3, we devise a data distribution procedure in memory space and propose an expedient method to reduce the frequency of warp branches. This implementation speeds up the kernel function more than three times compared with the methods proposed in previous work.

The results of our DM simulations presented in section 4 show that the cycle of expansion and contraction of the interstellar gas is an effective mechanism for flattening the central cusp in the mass density profile of the CDM halo. The recurrence frequency of star formation is one of the important factors in determining the dynamical response of DM halos. In section 5, we compare these results with an analytical model of the resonance between DM particles and the density waves of the interstellar gas that induces the recurring change of the gravitational potential. This analysis indicates that the resonance is effective in the cusp to core transition of the CDM halos and that this theory can resolve the core–cusp problem.

2. The tree method for collisionless gravitational systems

DM halos are collisionless self–gravitational systems in which the gravity of the whole system is more important than encounters of nearby particles. To follow the dynamical evolution of such systems, N–body simulations are a suitable numerical method. N–body simulations with large number of particles, N, must be performed to avoid artificial two–body relaxation and to achieve a sufficiently high resolution that ensures the softening length is much smaller than the resolution limit of observations of density profiles of DM halos.

The most straightforward and accurate algorithm to compute gravity among particles is the "direct" method. The computational cost is $O(N^2)$, since we would have to compute gravitational acceleration between all pairs of particles. Therefore this algorithm is unfavorable to solve problems with large N. Hereafter, we call the particles for which the gravitational acceleration is computed the "i–particles" and call the particles that interact with i–particles the "j–particles". The tree method reduces the cost to $O(N \log N)$ by treating a group of j–particles that are sufficiently far from the i–particle as a single heavy j–particle [10].

The tree method consists of 2 parts, tree–construction and tree–traversal. In the first part, the tree–construction, we construct an oct–tree of particles. We split the cube containing all particles recursively into 8 (in 3D) smaller cubes (figure 1), and we link all cubes to their daughter– and sister–cubes (figure 2). We continue the recursion until every cube contains less than or equal to $N_{\rm crit}$ particles. We set the parameter $N_{\rm crit} = 4$. After the linking operation, we compute the mass and position of each cube. The mass of the cube is the sum of that of the contained particles, and its position is the center of mass of the contained cubes. In the second part, the tree–traversal, we walk the tree constructed in the previous part for each i–particle. A cube is sufficiently far from an i–particle if its angular size, θ, is smaller than a critical size $\theta_{\rm c}$ =0.6.

When we reach a cube for which $theta > theta_{\rm c}$, we walk to a daughter–cube. If we reach a cube for which $theta < theta_{\rm c}$ we walk to a sister–cube. In the latter case, j–particles contained in the cube are treated as a single heavy j–particle, reducing the cost of computation.

The tree method to solve collisionless systems reduces the computational cost to $O(N \log(N))$ and produces results of comparable accuracy to the direct method if $\theta_{\rm c}$ is sufficiently small. N–body simulations are manageable because they are inherently boundaryless and scale–free. Thus, N–body simulations utilizing the tree method are commonly used in computational astrophysics.

3. Implementation and performance of our tree code on GPU clusters

3.1. Overview of the code

We speed–up our tree code by optimizing it for GPU clusters. Following previous work, we let CPU cores compute the tree–construction and let GPUs compute the tree–traversal [11], [12]. We order particle data along a Peano–Hilbert curve, a type of space–filling curve, in

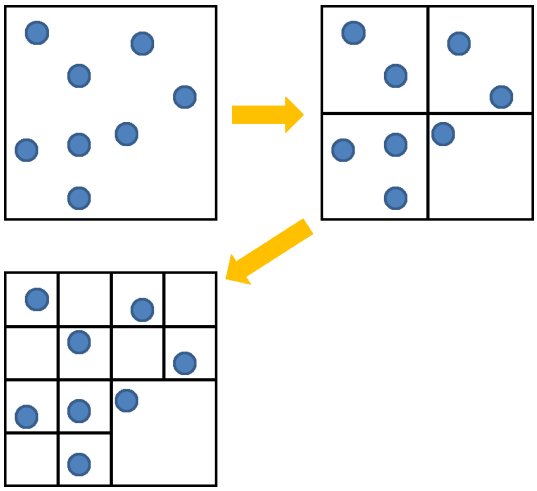

Figure 1. Recursive splitting of cubes into 8 (in 3D) smaller cubes until all cubes contain less than $N_{\rm crit}$ particles (blue dots). In this schematic diagram, $N_{\rm crit}$ is 1, whereas in our simulations $N_{\rm crit} = 4$.

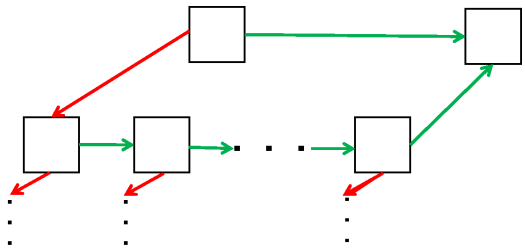

Figure 2. Links between cubes. Red and green arrows indicate links to daughter– and sister–cube, respectively.

memory space superior to Morton ordering, to achieve high cache hit rate. Furthermore, we have parallelized our code using MPI. Each MPI process becomes the host of a GPU, owns partial particle data, constructs a partial tree composed of its i–particles, and communicates necessary tree data with other processes.

"Warp branches" are an important issue in GPU computing. We explain the problem for the case of the NVIDIA Fermi architecture. In this architecture of GPUs, 32 cores work concurrently with the same instruction stream. In other words, GPU cores work in the manner of SIMD. This cluster of 32 cores is called a warp. When two or more different instructions emerge among the 32 cores, a large overhead arises because the instruction can only be processed sequentially within a warp. Such warp branches occur during the tree–traversal part whenever we step to a daughter– or sister–cube. Warp branches become less frequent if we bundle routes for given i–particles in the tree structure.

To achieve this, we propose a method in 2 steps: The first step is a "vectorization" step. We vectorize the operations of route selection in the tree structure and that of calculating gravitational acceleration for V i–particles, where V is the vector length or the number of particles per GPU core. We show a conceptual diagram of this step in figure 3. We bundle routes of V i–particles in a GPU core by measuring the distance to the cube from the nearest

particle in a core, which is circled red in figure 3. We then select the next step in the tree structure, a daughter– or the sister–cube, by using the distance to the present cube from the nearest particle among the bundled ones. If $\theta > \theta_c$ for the nearest particle, we take a step to a daughter–cube even if other particles are sufficiently far from the present cube. If $\theta < \theta_c$ for the nearest particle, we compute gravitational acceleration from that cube to the i–particles by using the respective distance between each i–particle and the cube. After that, the walk continues to a sister–cube. The frequency of warp branches drops with increasing vector length, V. However, the amount of work per GPU core is proportional to V. Thus it is not advisable to set V very large.

The second step is a "grouping" step, which alleviates the problem of vectorization described above to some degree. We bundle routes for i–particles among several cores in the same fashion as we bundle the routes for i–particle in a core (figure 4). Again, the nearest particle in the group is circled red. We select the next step in the tree-traversal for i–particles by the value of the distance to a cube from the nearest particle. By grouping, we can increase the number of bundled i–particles without increasing the number of particles owned by a GPU core. The number of bundled particles becomes $V \times G$, where G is the number of group members.

Vectorizing and grouping will decrease the frequency of warp branches, but will increase the amount of computation. This is because, by choosing the particle in a vector or group nearest to the cube, will reduce the number of particles for which $\theta < \theta_c$ and for which we could treat the cube as a single heavy j–particle. Thus we expect that there exist an optimal pair of values for the vector length and number of group members. The Peano–Hilbert ordering of particles is very important in our method since bundling routes for spatially distant particles results in a large amount of unnecessary computations.

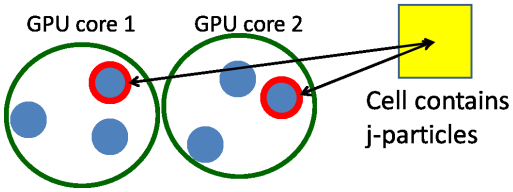

Figure 3. Conceptual diagram of vectorization. Blue dots represent i–particles. The red circled particles are the closest ones to the cube of j–particles. In this case, each GPU core holds 3 i–particles ($V = 3$).

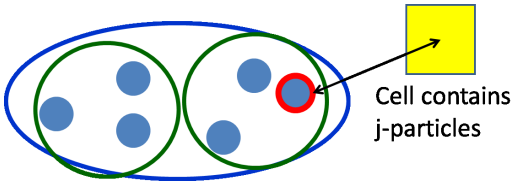

Figure 4. Conceptual diagram of grouping. The same i–particles are shown as figure 3. In this case, the number of group member, G, is 2, and 6 i–particles have been bundled into a group.

3.2. Computing environment for performance measurements

We measure the performance of our code on the GPU cluster HA-PACS, which was installed at the Center for Computational Sciences of the University of Tsukuba in 2012 [13]. HA–PACS has

4 NVIDIA Tesla M2090 GPU cards per node. As described above, each MPI process is the host of a GPU, and we launch up to four MPI processes per node. The specifications of HA–PACS are shown in Table 1.

Table 1. Compendium of HA–PACS

CPU	Intel(R) Xeon(R) CPU E5-2670 2.60GHz (8 cores/socket × 2 sockets = 16 cores/node)
GPU	NVIDIA Tesla M2090 (4 GPUs / node)
Main Memory	128 GB, DDR3 1600MHz, 4 channel / socket, 102.8 GB/s/node
OS	CentOS release 6.1 (Final)
CPU Compiler	GCC 4.4.5-6
GPU Toolkit	CUDA 4.0.17
Interconnection	Infiniband QDR × 2 rails
MPI	MVAPICH2 1.8

3.3. Performance of the code

In figure 5, we showed the results of performance measurements for the GPU part of the kernel function, which is essentially the tree–traversal part. In this test, we simply follow the evolution of particles initially distributed with the Navarro–Frenk–White (NFW) profile, which is often used for CDM halos [1] and a random velocity field. We set the number of particles, $N = 2^{23} \approx 8$ million, and the tolerance parameter of the tree method, $\theta_c = 0.6$. We study the effects of vectorization and grouping by comparing runs with different combinations of V and G. Compared with the case of no vectorization or grouping, $(V, G) = (1, 1)$, tree–traversal may become more than 3 times faster. As expected, there exists an optimal pair of values for the vector length and the number of group members. In this test, the optimal pair is $(V, G) = (4, 4)$. In general, the optimal pair values depends on the N–body problem and the GPU architecture.

We have parallelized the tree–construction and tree–traversal operations with MPI. N particles are sorted according to the Peano–Hilbert space–filling curve, and N/N_p particles are assigned to each MPI process, where N_p is the number of MPI processes or GPU cards.

We now check the performance of the whole code. The tree–construction, tree–traversal, and the construction of the minimum partial trees that are communicated to other processes (Locally Essential Tree; LET) [14] take up most of the computational time.

Figure 6 shows the performance for strong scaling as a function of N_p. We set the total number of particles, $N = 2^{25} \approx 32$ million. When $N_p < 10$, the scaling is good. However, the performance saturates for $N_p > 10$. We suspect the reason for the performance saturation of the kernel function (blue line) is the worsening load balance between MPI processes. Because we have sorted and assigned N/N_p particles to each process, some processes are assigned particles from the central part of the computational domain while others are assigned from the outskirts of the domain. The number of cubes encountered in the tree–traversal for these processes can differ in this test by a factor ~ 5. To improve the load balance, we need to change the way particles are assigned to processes. The time to construct the LET (magenta) is nearly constant, and becomes dominant when $N_p > 10$. Another issue is that, in the current implementation, some CPU cores remain idle. The idle CPUs could be used to accelerate the operation of construction of LET.

We show the performance for weak scaling in figure 7. Each process holds $N_1 = 2^{24} \sim 16$M particles. The total number of particles is $N = N_1 \times N_p$. The time to compute gravitational

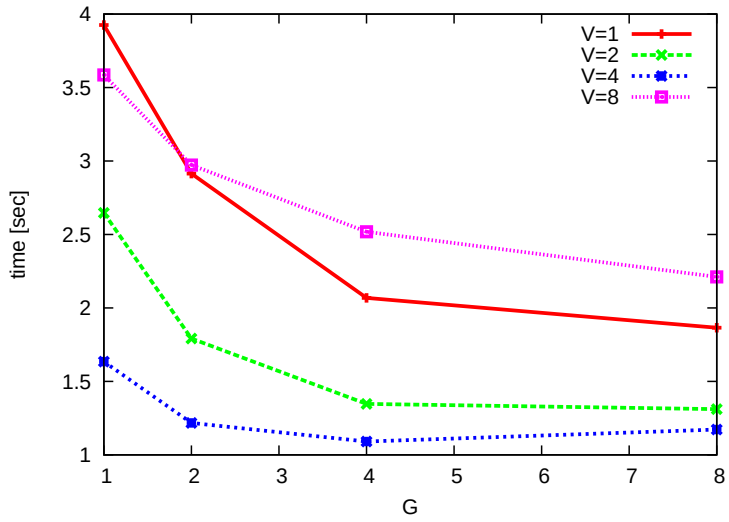

Figure 5. The dependence of the computational time of the kernel function on the vector length, V, and group size, G. Each line is the result for a different V. The best performance for this test calculations is obtained for $(V, G) = (4, 4)$.

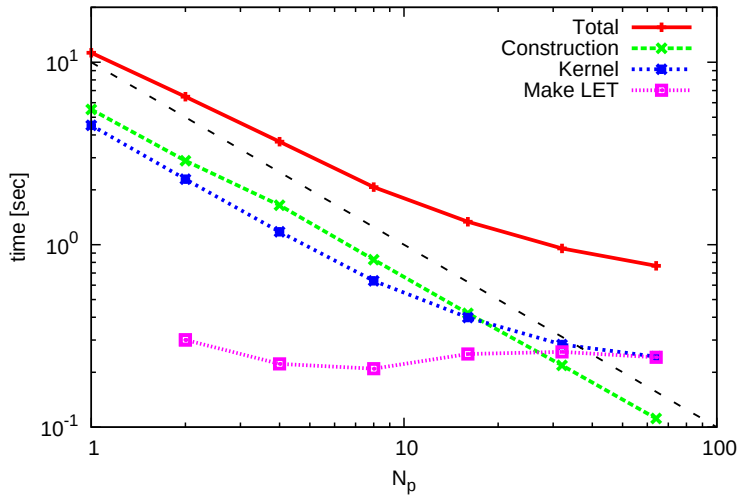

Figure 6. Strong scaling of different components of our code. Horizontal and vertical axes are number of MPI processes, $N_{\rm p}$ and computational time in the unit of second. Each line represents the time for tree–construction (green), kernel function (blue), and making LET (magenta). The red line is the total time. The black dashed one shows a scaling $\propto N_{\rm p}^{-1}$.

acceleration (red) is proportional to $\log N_{\rm p}$. This can be explained as follows: The computational cost of the tree method is $O(N \log N) = O(N_1 N_{\rm p} \log N_1 N_{\rm p})$. Through parallelization, it becomes $O(N_1 \log N_1 N_{\rm p})$. Since N_1 is a constant, the computational cost $\propto \log N_{\rm p}$. In other words, the depth of the tree structure is proportional to $\log N \propto \log N_{\rm p}$. In future work, we will enable the computation of the kernel function and the construction of the LET to occur in parallel through MPI communication to improve performance.

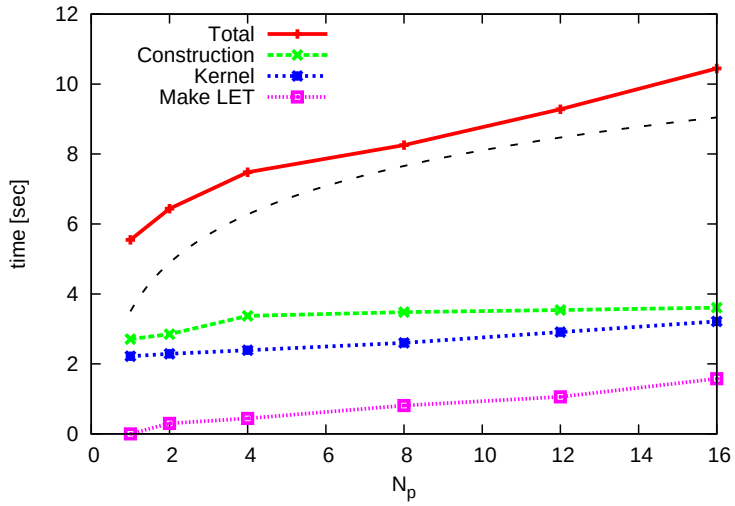

Figure 7. Weak scaling of different component of our code. The black line represents a scaling $\propto \log N_{\rm p}$. Other lines and axes have the same meaning as in figure 6.

4. Results of N–body simulations with oscillating galactic potential

We investigate the dynamical response of DM halos to oscillations of the galactic potential using N–body simulations.

4.1. Set up

The N–body system represents a DM halo. Initially, the halo profile follows the NFW model in virial equilibrium state. The NFW profile has a central cusp. To express the baryon component, we add an external Hernquist potential [15] to the N–body system and change its scale length in an oscillating manner to represent the gas contraction and expansion cycle driven by periodic starbursts. We choose a scale length typical for dwarf galaxies, 1kpc and an oscillation amplitude of 1kpc, that represent massive outflows. Fundamental parameters adopted in our simulations are listed in Table 2. We investigate three different values for the oscillation period of the external potential, T, which are chosen to reflect the star formation histories of observed local dwarf galaxies [16] [17] [18].

Table 2. Fixed parameters in our N–body simulation.

Number of particles	$2^{24} \sim$ 16M, $2^{27} \sim$ 128M
Softening (resolution)	0.004 kpc
Mass of DM halo	$10^9 M_\odot$
Scale length of DM halo	2 kpc
Baryonic mass	$1.7 \times 10^8 M_\odot$

4.2. Results

Figure 8 demonstrates the resultant density profile of DM halos after 10 oscillation periods. The red, blue, and magenta lines show the results for $T = 10$, 30 and 100 Myr, respectively. The yellow line is the result of a high resolution run for $T = 30$ Myr, in which we used 8 times the number of particles, which demonstrates good convergence of the solution. The black dashed

line represents the initial NFW model, which has a central cusp. It is clear that the cusp has transitioned to a core, and that the resultant core scale depends on the oscillation period of the external potential, T. We have confirmed that the resultant core structure is stable during at least the first 100 oscillation periods.

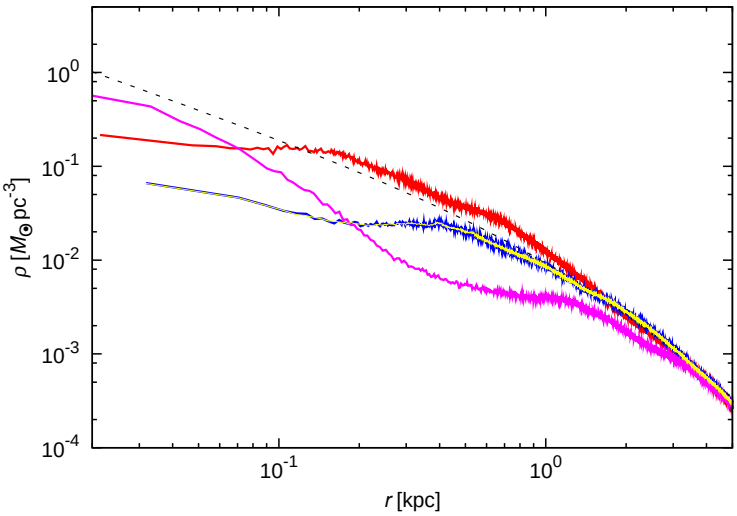

Figure 8. Density profiles of DM halos after 10 oscillation periods. The horizontal and vertical axes are the distance r from the galactic center and mass-density of the DM halo, respectively. The red, blue, and magenta lines show the results for $T = 10$, 30 and 100 Myr, respectively. Vertical lines are the predictions of the core scale by our analytical model (see section 5).

We also analyze the temporal Fourier spectrum of radial velocities of the modified DM halos. In figure 9, we show the Fourier spectrum of the radial velocity of the system, $\hat{v}_\mathrm{r}$, as a function of r for the Fourier frequencies, $\omega = 2\pi/T$. Comparing the location of the peaks of the spectra in this figure with the core length scale in figure 8, we find that the two match very well. We now turn to the physics of the mechanism that accelerates the DM particles on core length scales and creates the core structure from an initially cuspy profile.

5. Linear analysis of resonance

What is the physical reason for the transition from DM cusp to core in our simulations? To investigate this, we performed a linear analysis of the resonance between the density waves and the particles. We approximate the particle system by a fluid and consider the situation in which the equilibrium system, labeled 0, is perturbed by the external force (subscript ex), inducing a change in some physical quantities (subscript ind). We focus on a particular group of particles of some constant density, ρ_0, and velocity, v_0, in the equilibrium state. We assume the external force has a sinusoidal form,

$$-\frac{\partial \Phi_\mathrm{ex}}{\partial x} = A \sin\left(kx - \Omega t\right), \tag{1}$$

where A, k, and Ω are the oscillation strength, which is a positive constant, the wavenumber, and the frequency of the external force, respectively.

We solve the linearized Euler's equation and the equation of continuity, and derive the following solutions for the induced velocity and induced density enhancement;

$$v_\mathrm{ind}(t, r) \quad = \quad \frac{A}{\Omega - k v_0} \cos\left(kr - \Omega t\right), \tag{2}$$

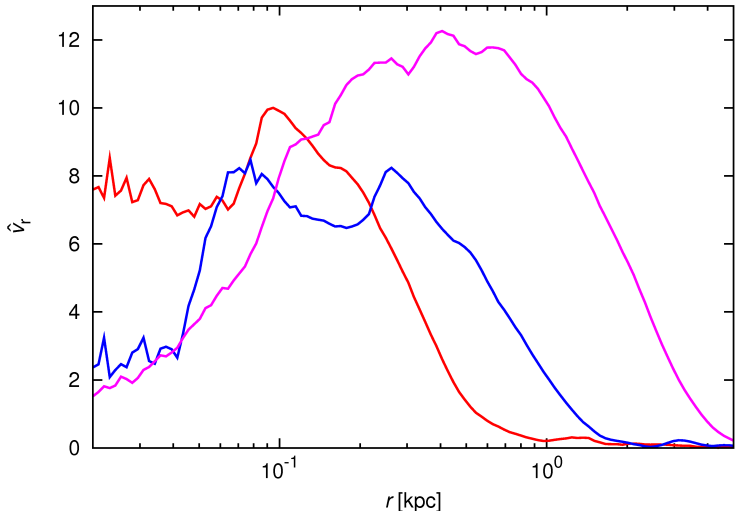

Figure 9. Fourier spectrum of the radial velocity of the system, $\hat{v}_{\rm r}$ as a function of r. Each line represents the spectrum of same run as figure 8.

$$\rho_{\rm ind}(t,r) \quad = \quad \frac{A\rho_0 k}{(\Omega - kv_0)^2}\cos\left(kr - \Omega t\right). \tag{3}$$

We find that the coefficients will diverge when the frequency of the external force $\Omega \equiv 2\pi/T$ equals kv_0. This is analogous to the resonance in the forced oscillation problem of harmonic oscillators. Thus, we conclude that DM halos are strongly affected by resonance between DM particles and density waves, and a core structure can form this way.

We may rewrite the resonance condition, $kv_0 \sim \Omega$, as

$$t_{\rm d}(r) \sim T, \tag{4}$$

where $t_{\rm d}(r)$ is the local dynamical time of the DM halo measured at r. Using this condition, we can predict the core scale created by resonance. The vertical lines in figure 8 are the predictions of this analytical model. They match well with the scales of resultant core.

6. Summary

To resolve the core-cusp problem of DM halos, we have studied the dynamical response of DM halos to oscillations of the gravitational potential. We have performed collisionless N–body simulations utilizing the tree algorithm. We speed–up our tree code by optimizing it for GPU clusters. We have proposed a method to reduce the frequency of Warp branches, and, as a result, the tree–traversal operation has been accelerated. From our simulations, we have arrived at the conclusion that resonance between DM particles and gas oscillation may play an important role to flatten the central cusp. Our analytical model predicts the core scales seen in the simulations quite accurately.

Acknowledgments

We greatly thank Alexander Wagner for checking our manuscript. Numerical simulations were performed with HA–PACS at the Center for Computational Sciences, University of Tsukuba. This work was supported by the Grant–in–Aid for Scientific Research (A)(21244013), and (C)(25400222).

References

[1] Navarro J F, Frenk C S and White S D M 1997 *The Astrophysical Journal* **490** 493
[2] Moore B, Quinn T, Governato F, Stadel J and Lake G 1999 *Monthly Notices of the Royal Astronomical Society* **310** 1147
[3] Ishiyama T, Makino J, Portegies Zwart S *et al.* 2011 *Preprint* astro–ph/1101.2020
[4] Moore B 1994 *Nature* **370** 629
[5] Burkert A 1995 *The Astrophysical Journal Lett.* **447** L25
[6] Oh S–H, de Blok W J G, Brinks E, Walter F and Kennicutt R C Jr. 2011 *The Astronomical Journal* **141** 193
[7] Mashchenko S, Wadsley J and Couchman H M P 2008 *Science* **319** 174
[8] Governato F, Brook C, Mayer L *et al.* 2010 *Nature* **463** 203
[9] Pontzen A and Governato F 2012 *Monthly Notices of the Royal Astronomical Society* **421** 3464
[10] Barnes J and Hut P 1986 *Nature* **324** 446
[11] Nakasato N 2012 *Journal of Comp. Sci.* Volume 3 **3** 132141
[12] Nakasato N, Ogiya G, Miki Y, Mori M and Nomoto K 2012 *Preprint* astro–ph/1206.1199
[13] HA–PACS, http://www.ccs.tsukuba.ac.jp/CCS/eng/research-activities/projects/ha-pacs/base-cluster
[14] Warren M and Salmon J 1993 *ACM/IEEE Conference on Supercomputing* (New York: ACM.) pp 1221
[15] Hernquist L 1990 *The Astrophysical Journal* **356** 359
[16] Mateo M L 1998 *Annual Review of Astronomy and Astrophysics* **36** 435
[17] Tolstoy E, Hill V and Tosi M 2009 *Annual Review of Astronomy and Astrophysics* **47** 371
[18] McQuinn K B W, Skillman E D, Cannon J M *et al.* 2010 *The Astrophysical Journal* **724** 49