Direct Calculation of Self-Gravitational Force for Infinitesimally Thin Gaseous Disks Using Adaptive Mesh Refinement

Yao-Huan Tseng,1, 2 Hsien Shang,1, 2 and Chien-Chang Yen3,1, 2

1Institute of Astronomy and Astrophysics, Academia Sinica, Taipei 10617, Taiwan, R.O.C.
2Academia Sinica, Theoretical Institute for Advanced Research in Astrophysics
3Department of Mathematics, Fu Jen Catholic University, New Taipei City, Taiwan

Submitted to ApJ Supplement

ABSTRACT

Yen et al. (2012) advanced a direct approach for the calculation of self-gravitational force to second order accuracy based on uniform grid discretization. This method improves the accuracy of \(N\)-body calculation by using exact integration of kernel functions and employing the Fast Fourier Transform (FFT) to reduce complexity of computation to nearly linear. This direct approach is free of artificial boundary conditions, however, the applicability is limited by the uniform discretization of grids. We report here an advancement in the direct method with the implementation of adaptive mesh refinement (AMR) and maintaining second-order accuracy, which breaks the barrier set by uniform grid discretization. The adoption of graphic process units (GPUs) can significantly speed up the computation and make application of this method possible to astrophysical systems of gaseous disk galaxies and protoplanetary disks.

Keywords: gravitation - methods: numerical

1. INTRODUCTION

Self-gravity of gas is significant in many areas of astrophysics. Dynamical evolution of the center region of galaxies, such as the starburst rings (Lin et al. 2013; Seo & Kim 2014), the sub-parsec scale within the Active Galactic Nuclei (AGN), inner bars and spirals (Kim et al. 2012; Lee & Shu 2012; Lee 2014), and bright and young stars formed along spiral arms (Elmegreen et al. 2014), are consequences of self-gravitating interactions of gas. Such phenomena exist because the interstellar gas has to be transferred to the central region at a typical scale of 10-kpc. Meanwhile, the self-gravity of gas play an important role to the evolution of gas in disk galaxies. A disk is regarded to be thin relative to the physical size scale when the ratio of lengths of vertical and radial directions is around 0.001–0.01. An infinitesimally thin gaseous disk represents the disk as a distribution of two dimensional surface density in three dimensional space.

The potential \(\Phi\) for a given distribution of gaseous density \(\rho\) in three-dimensional space satisfies the Poisson equation:

\[
\nabla \cdot (\nabla \Phi(x, y, z)) = 4\pi G \rho(x, y, z),
\]

where \(\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})\), or as an integral

\[
\Phi(x, y, z) = -G \iiint K(\bar{x} - x, \bar{y} - y, \bar{z} - z) \rho(\bar{x}, \bar{y}, \bar{z}) \, d\bar{x} \, d\bar{y} \, d\bar{z}
\]
where \( K(x,y,z) = \frac{1}{\sqrt{x^2 + y^2 + z^2}} \) and \( G \) is the gravitational constant. The density \( \rho(x,y,z) \) for a gaseous disk is associated with a surface density \( \sigma(x,y) \) by

\[
\rho(x,y,z) = \sigma(x,y) \delta(z),
\]

where \( \delta \) is the Dirac function. Furthermore, the potential in mid-plane, \( \Phi(x,y,0) \), is associated with a singular kernel integral through

\[
\Phi(x,y,0) = -G \int \int K(\bar{x} - x, \bar{y} - y, 0) \sigma(\bar{x}, \bar{y}) \, d\bar{x} \, d\bar{y}.
\]

It is well known multigrid method can solve the potential, which satisfies Eq. (1) on finite region with some prescribed boundary conditions with linear complexity. In three dimension, the linear complexity is \( O(N^3) \) where \( N \) is the number of zones in one dimension. In the current situation, the boundary is open and the values at boundary are required to be evaluated. The complexity will reach \( O(N^5) \) using direct approach and it can be reduced to \( O(N^4) \) if the density is replaced by surface density \( \sigma(\bar{x}, \bar{y}) \) defined in Eq. (2). Another evaluation of potential to deal with vacuum boundary is developed by James (1977) and it is extended to cylindrical coordinates (Moon et al. 2019). Once the potential obtained, one can take the numerical derivatives to get the forces. Therefore, the second order of accuracy for force calculation should be guaranteed by more than second order of accuracy for potential (Gupta et al. (1997); Lai & Tseng (2007)).

A self-gravity calculation can use the differential equation or the integral form. An integral form is more efficient than a differential equation for an infinitesimally thin gaseous disk. A direct calculation of the force in the \( x \)-direction is to compute the derivative

\[
-\frac{\partial}{\partial x} \Phi(x,y,0) = -\int \int \frac{\partial}{\partial x} K(\bar{x} - x, y - y, 0) \sigma(\bar{x}, \bar{y}) \, d\bar{x} \, d\bar{y},
\]

and similarly, the calculation of \( y \)-force and \( z \)-force are to compute the \( y \)- and \( z \)-partial derivative of the potential, respectively. A direct method for self-gravitational force calculation as shown in Eqn (3) has been proposed in Yen et al. (2012). This approach can be regarded as an improvement to the \( N \)-body calculation on order of accuracy using exact integration of the kernel functions. A fast computation for the method based on uniform grid discretization was developed by employing the fast Fourier transform (FFT), which reduces the complexity of computation to nearly linear. This approach has also been shown to second order accuracy. It was fast, and free of artificial boundary conditions. Such properties of the method have been developed and demonstrated for the uniform grid. Moreover, it has been extended for the nested-grids in Cartesian coordinates by Wang et al. (2016). Unfortunately, the order of accuracy for this approach on polar coordinate using logarithmic discretization in radial direction will be reduced due to the fact that no analytic formula exists for the integral of logarithmic function. For preserving second order of accuracy, current work is limited to the Cartesian coordinates.

Adaptive mesh refinement (AMR) in hydrodynamic simulations has become a highly desirable target for the generalization of self-gravitational force calculations. Here, we consider computation of self-gravity with AMR by preserving the order of accuracy. The technique of FFT itself in this case, however, is unable to accelerate the computation for the increased complexity. To carry out the full fast calculation with a direct method, one would need to rely on recent improvement of computational power. More precisely, we employ the graphic process units (GPU) in this new development to reduce the computational time. In short, we generalize Wang et al. (2016) to accommodate the grid generation in AMR with the computational capability of GPUs at state-of-art level. Meanwhile, we develop two processes of level refinement: independent and variable, for which the computations will be carried out and benchmarked.

This paper is organized as follows. We first outline generalization of Yen et al. (2012) into the AMR grid in Section 2. We carry out simulations using both uniform and AMR grid discretization with independent and variable level refinement approaches to investigate accuracy, order of accuracy, and the computational time in reality in Section 3. We conclude this work in the last section.

2. DIRECT METHOD BASED ON AMR

The proposed direct method of force calculation based on AMR is more flexible for grid discretization, conceptually simpler, and straightforward to implement. Let us start with a surface density distribution \( \sigma(x,y) \), which is assumed
to vanish outside of a rectangular region $D$, and
\[
\sigma = \sum_{i=1}^{N} \sigma_i(x, y),
\]
(4)
where $\sigma_i(x, y) = \sigma(x, y)\chi_{D_i}$, $\chi_{D_i}$ is a characteristic function of the domain $D_i$,

\[
\chi_{D_i}(x, y) = \begin{cases} 
1, & (x, y) \in D_i, \\
0, & (x, y) \notin D_i,
\end{cases}
\]
with the sub-rectangle $D_i$ mutually disjoint with each other except at the boundary and $\bigcup_{i=1}^{N} D_i = D$. The positive integer $N$ is the total number of zones generated by the AMR grids. From Eqs. 3 and 4, it follows that

\[
-\frac{\partial}{\partial x} \Phi(x, y, z) = \sum_{i=1}^{N} \int_{D_i} \frac{\partial}{\partial x} (K(\bar{x} - x, \bar{y} - y, z)) \sigma_i(\bar{x}, \bar{y}) \, d\bar{x} \, d\bar{y}.
\]
(5)

In Yen et al (2012), the restricted surface function $\sigma_i$ on $D_i$ is approximated by a Taylor polynomial of order 1, and it is given as

\[
\sigma_i(x, y) \approx \sigma_i^0 + \delta_x^i (x - x_i^c) + \delta_y^i (y - y_i^c)
\]
(6)
in the subrectangle region $D_i$, with $(x_i^c, y_i^c)$ being the center of the sub-rectangle $D_i$. Substituting equations (6) into (5) gives us

\[
-\frac{\partial}{\partial x} \Phi(x, y, z) \approx -\sum_{i=1}^{N} \int_{D_i} \frac{\partial}{\partial x} (K(\bar{x} - x, \bar{y} - y, z)) \left[ \sigma_i^0 + \delta_x^i (x - x_i^c) + \delta_y^i (y - y_i^c) \right] \, d\bar{x} \, d\bar{y}
\]

\[
= -\sum_{i=1}^{N} \sigma_i^0 \int_{D_i} \frac{\partial}{\partial x} (K(\bar{x} - x, \bar{y} - y, z)) \, d\bar{x} \, d\bar{y}
\]

\[
-\sum_{i=1}^{N} \delta_x^i \int_{D_i} \frac{\partial}{\partial x} (K(\bar{x} - x, \bar{y} - y, z)) (\bar{x} - x_i^c) \, d\bar{x} \, d\bar{y}
\]

\[
-\sum_{i=1}^{N} \delta_y^i \int_{D_i} \frac{\partial}{\partial x} (K(\bar{x} - x, \bar{y} - y, z)) (\bar{y} - y_i^c) \, d\bar{x} \, d\bar{y}.
\]

We define $F_i^x(x, y, z) = -\frac{\partial}{\partial x} \Phi(x, y, z)$ and we obtain

\[
F_i^x(x, y, z) \approx F_i^{x,0}(x, y, z) + F_i^{x,x}(x, y, z) + F_i^{x,y}(x, y, z)
\]

where

\[
F_i^{x,0}(x, y, z) = -\sum_{i=1}^{N} \sigma_i^0 K_i^{x,0}(x, y, z),
\]

\[
F_i^{x,x}(x, y, z) = -\sum_{i=1}^{N} \delta_x^i K_i^{x,x}(x, y, z),
\]

\[
F_i^{x,y}(x, y, z) = -\sum_{i=1}^{N} \delta_y^i K_i^{x,y}(x, y, z),
\]
and

\[ K_i^{x,0}(x,y,z) = \iint_{D_i} \frac{(\vec{x} - x)}{((\vec{x} - x)^2 + (\vec{y} - y)^2 + z^2)^{3/2}} \, d\vec{x} \, d\vec{y}, \]

\[ K_i^{x,x}(x,y,z) = \iint_{D_i} \frac{(\vec{x} - x_i^r)(\vec{x} - x)}{((\vec{x} - x)^2 + (\vec{y} - y)^2 + z^2)^{3/2}} \, d\vec{x} \, d\vec{y}, \]

\[ K_i^{x,y}(x,y,z) = \iint_{D_i} \frac{(\vec{y} - y_i^c)(\vec{x} - x)}{((\vec{x} - x)^2 + (\vec{y} - y)^2 + z^2)^{3/2}} \, d\vec{x} \, d\vec{y}. \]

We further decompose the sub-rectangle domain \( D_i = [x_i^r, x_i^c] \times [y_i^d, y_i^u] \). The full expression of force kernels that are summarized as below and was formulated in Yen et al. (2012) when \( z = 0 \), using \([x_i^r, x_i^c]\) and \([y_i^d, y_i^u]\):

\[ K_i^{x,0}(x,y,z) = -\ln((\bar{y} - y) + \sqrt{(\bar{x} - x)^2 + (\bar{y} - y)^2 + z^2})|x_i^r y_i^d|, \]

\[ K_i^{x,x}(x,y,z) = (x - x_i^c)K_i^{x,0}(x,y,z) + \left[ (\bar{y} - y)\ln((\bar{x} - x) + \sqrt{(\bar{x} - x)^2 + (\bar{y} - y)^2 + z^2}) \right] \frac{x_i^r y_i^d}{x_i^c y_i^u}, \]

\[ K_i^{x,y}(x,y,z) = (y - y_i^c)K_i^{x,0}(x,y,z) - \left[ \sqrt{(\bar{x} - x)^2 + (\bar{y} - y)^2 + z^2} \right] \frac{x_i^r y_i^d}{x_i^c y_i^u}, \]

\[ K_i^{y,0}(x,y,z) = -\ln((\bar{x} - x) + \sqrt{(\bar{x} - x)^2 + (\bar{y} - y)^2 + z^2})|y_i^u x_i^r|, \]

\[ K_i^{y,x}(x,y,z) = (x - x_i^c)K_i^{y,0}(x,y,z) - \left[ \sqrt{(\bar{x} - x)^2 + (\bar{y} - y)^2 + z^2} \right] \frac{y_i^u x_i^r}{y_i^c x_i^u}, \]

\[ K_i^{y,y}(x,y,z) = (y - y_i^c)K_i^{y,0}(x,y,z) + \left[ (\bar{x} - x)\ln((\bar{y} - y) + \sqrt{(\bar{x} - x)^2 + (\bar{y} - y)^2 + z^2}) \right] \frac{y_i^u x_i^r}{y_i^c x_i^u}, \]

\[ K_i^{z,0}(x,y,z) = \arctan\left( \frac{\sqrt{(\bar{x} - x)^2 + (\bar{y} - y)^2 + z^2}}{|xy|} \right) \frac{y_i^u x_i^r}{y_i^c x_i^u}, \]

\[ K_i^{z,x}(x,y,z) = (x - x_i^c)K_i^{z,0}(x,y,z) + zK_i^{x,0}(x,y,z), \]

\[ K_i^{z,y}(x,y,z) = (y - y_i^c)K_i^{z,0}(x,y,z) + zK_i^{y,0}(x,y,z). \]

This full expression can be evaluated if the position \((x,y)\) does not coincide with the position \(\bar{x} = x\) or \(\bar{y} = y\) on the plane, \(z = 0\). For uniform grid discretization, the center of the zone, \(\bar{x}\) and \(\bar{y}\) will never fall on the interface; however, this situation may not be completely avoidable in the context of nested grid (Wang et al. 2016) or the AMR. When such situation occurs, the Kernel \(K_i^{x,0}(x,y,0)\) can be modified to be

\[ -\text{sgn}(|\bar{x} - x|)\ln((\bar{y} - y) + \sqrt{(\bar{x} - x)^2 + (\bar{y} - y)^2}) - (1 - \text{sgn}(|\bar{x} - x|))\text{sgn}(\bar{y} - y)\ln|\bar{y} - y|, \]

which can still be used for the uniform grids but evaluating it would take more computational time.

### 3. Numerical Simulations

In this section, we carry out simulations involving self-gravitational force calculation in thin disks based on the AMR formulation derived above. The force is calculated at the center of each zone. We will demonstrate that our updated direct method for the calculation of self-gravitational force fits well under the framework of AMR, with two refinement approaches. One is the Independent Level Refinement (ILR) in which each level refined is independent of the value of the level, and the other is the Variable Level Refinement (VLR) in which each of the refinement taking place is variable with the value of the level.

For simplicity, the domain of consideration is confined to a rectangle for the coarsest grid discretized. Each refinement subdivides each parent zone into four sub-rectangles. We record each cell by a tuple of 9 elements, \((x_c, y_c, L, P, EL, EL, EL, EL, EL)\).
LN, DN, RN, UN), whose center \((x_c, y_c)\) is located at level \(L\) for each cell, with parent \(P\) and the end of leaf \(EL\). The elements LN, DN, RN, and UN, are the nearby nodes on the sides of left, down, right, and up. This generation of grid allows an easier implementation into the hydrodynamical simulations.

In the hydrodynamical simulation, one needs to consider the grid zones at the same and different levels for the calculation of flux. Here we concentrate on the self-gravitational force calculation, which is only restricted to the leaf nodes (zones). We apply our approach to an AMR grid generated in the sense of Berger & Oliger (1984), as shown in Figure 1. The generation of an uniform grid is initiated at the coarsest level \(L = 1\), where the coarsest grid is divided into \((1, 2, 3, \ldots, 16)\), a total of 16 cell zones. The first refinement is taking place at the cell zones 6 and 11 to the level of \(L = 2\), where the cell zone 6 is divided into zones \((17, 18, 19, 20)\) and zone 11 subdivided into zones \((21, 22, 23, 24)\). For \(L = 3\) refinement within zones 17 and 19, they are referred to as zones \((25, 26, 27, 28)\) and \((29, 30, 31, 32)\), respectively. We report the complete list of recorded data in Table 10 of the Appendix A.

**Figure 1.** Illustration of the adaptive mesh refinement to \(L = 3\) level starting from a coarsest level of 16 zones.

We first address the preservation of second order accuracy in the complexity of computation on an AMR grid. We adopt the \(D_2\) disk in the family of finite disk with surface density \(\Sigma_n(R; \alpha) = (1 - R^2/\alpha^2)^{n-1/2}\) from Schulz (2009), for which \(n = 2\). The \(D_2\) disk has the following surface density:

\[
\Sigma_{D_2}(R; \alpha) = \begin{cases} 
\sigma_0(1 - R^2/\alpha^2)^{3/2} & \text{for } R < \alpha, \\
0 & \text{for } R > \alpha. 
\end{cases}
\]  

The corresponding potential in explicit form at the mid-plane \(z = 0\) as shown in Eqn. (29) of Schulz (2009) is,

\[
\Phi_{D_2}(R, 0; \alpha) = \begin{cases} 
-\frac{3\pi^2\sigma_0 R G}{64\alpha^3} (8\alpha^4 - 8\alpha^2 R^2 + 3R^4) & \text{for } R \leq \alpha, \\
-\frac{3\pi^2\sigma_0 R G}{32\alpha^3} \left[ (8\alpha^4 - 8\alpha^2 R^2 + 3R^4) \sin^{-1}(\frac{\alpha}{R}) + 3\alpha(2\alpha^2 - R^2) \sqrt{R^2 - \alpha^2} \right] & \text{for } R > \alpha,
\end{cases}
\]  

and the radial force can be found by differentiating the potential with respect to \(R\) (Eqn. [30] of Schulz 2009) as

\[
F_{R,D_2}(R, 0; \alpha) = \begin{cases} 
\frac{3\pi^2\sigma_0 R G}{10\alpha^3} (4\alpha^2 - 3R^2) & \text{for } R \leq \alpha, \\
\frac{3\pi^2\sigma_0 R G}{8\alpha^3} \left[ R(4\alpha^2 - 3R^2) \sin^{-1}(\frac{\alpha}{R}) - \alpha(2\alpha^2 - 3R^2) \sqrt{1 - \alpha^2/R^2} \right] & \text{for } R > \alpha.
\end{cases}
\]  

With the help of the potential-density pair, we compare the numerical results with the analytic calculations. The \(D_2\) disk is not smooth at its boundary, therefore it is suitable for a study based on the AMR approach to demonstrate the
resulting improvement. This is a feature that can not be explicitly shown by smooth functions either with a uniform or adaptively refined grid setting. Since the $D_2$ disk is symmetric, another case is a $D_2.2$ disk which consists of two $D_2$ disks which have the centers located at $(0, 0.5)$ and $(0, -0.5)$ of $D_2$, respectively, with $\alpha = 0.25$ breaks the property of symmetry. It is regarded as a general situation.

Without loss of generality, order of accuracy and computational complexity, we set the computational domain at $\Omega = [-1, 1] \times [-1, 1]$, $\sigma_0 = G = 1$ and $\alpha = 0.25$. We denote $\text{AMR}(L, D_2)$ as the simulation for the level $L$ and the $D_2$ disk. In particular, when $L = 1$, it is referred as the uniform grid discretization. Similarly, for the $D_{2.2}$ disk, it is denoted as $\text{AMR}(L, D_{2.2})$.

We first demonstrate accuracy for simulations of $\text{AMR}(L, D_2)$ and $\text{AMR}(L, D_{2.2})$, to be followed by demonstration of order of accuracy with the size of $N_k = 2^k$ zones in one dimension for $k = 7, 8, \ldots, 11$. The results for $D_2$ and $D_{2.2}$ with $L = 1$ reproduce the entries of Table 1 and Table 2 of Yen et al. (2012) for $k = 7$ to 10, which are shown below for comparison. Yen et al. (2012) concluded previously that the accuracy is nearly second order in the $L^1$, $L^2$ and $L^\infty$ norms. In the sense of AMR, we can express the norms as

$$|u|_1 \equiv \sum_i |u_i| \Delta A_i, \quad |u|_2 \equiv \left( \sum_i (u_i)^2 \Delta A_i \right)^{1/2}, \quad |u|_\infty \equiv \max_i |u_i|$$

where $\Delta A_i$ is the area of sub-region $D_i$. The order of accuracy $O_{\alpha,\beta}$ in the bottom portion of Table 1 and Table 2 are defined as $\log_2(L_{\alpha,\beta}(N_k)/L_{\alpha,\beta}(N_{k+1}))$ where $\alpha = x, y, R$, and $\beta$ equals 1, 2, $\infty$.

| $\text{AMR}(1, D_2)$ - $N_k$ | $L_{x,1}$ | $L_{x,2}$ | $L_{x,\infty}$ | $L_{R,1}$ | $L_{R,2}$ | $L_{R,\infty}$ |
|-----------------------------|---------|---------|------------|---------|---------|-------------|
| 128                         | 8.48E-4 | 9.31E-4 | 5.26E-3    | 1.32E-3 | 1.31E-3 | 5.91E-3     |
| 256                         | 2.16E-4 | 2.44E-4 | 1.93E-3    | 3.38E-4 | 3.44E-4 | 1.99E-3     |
| 512                         | 5.62E-5 | 6.88E-5 | 9.48E-4    | 8.80E-5 | 9.70E-5 | 9.84E-4     |
| 1024                        | 1.43E-5 | 1.82E-5 | 3.28E-4    | 2.24E-5 | 2.57E-5 | 3.47E-4     |
| 2048                        | 3.55E-6 | 4.77E-6 | 1.24E-4    | 5.57E-6 | 6.73E-6 | 1.36E-4     |

| $\text{AMR}(1, D_2)$ - $N_k/N_{k+1}$ | $O_{x,1}$ | $O_{x,2}$ | $O_{x,\infty}$ | $O_{R,1}$ | $O_{R,2}$ | $O_{R,\infty}$ |
|---------------------------------------|----------|----------|----------------|----------|----------|---------------|
| 128/256                               | 1.97     | 1.93     | 1.45           | 1.96     | 1.93     | 1.57          |
| 256/512                               | 1.94     | 1.93     | 1.03           | 1.94     | 1.93     | 1.02          |
| 512/1024                              | 1.98     | 1.92     | 1.53           | 1.98     | 1.92     | 1.50          |
| 1024/2048                             | 2.00     | 1.93     | 1.40           | 2.00     | 1.93     | 1.35          |

Table 1. $\text{AMR}(L, D_2)$ simulation for a $D_2$ disk with $L = 1$ (uniform discretization). $L_{\alpha,\beta}$ represents the the errors of forces between numerical and analytic solutions for the $x$-direction ($\alpha = x$) and radial direction ($\alpha = R$) in the norms of $\beta = 1, 2, \infty$. The top portion shows the errors for $N_k = 2^k$, where $k = 7, 8, \ldots, 11$. The bottom portion shows the order of accuracy in the 1- and 2-norms are of nearly second order for both $x$ and $R$ directions, but roughly about 1.3 for the $\infty$-norm.

3.1. Accuracy of Computation and Order of Accuracy on AMR

We demonstrate the cases of adaptive mesh refinement for $\text{AMR}(L, D_2)$ and $\text{AMR}(L, D_{2.2})$ with $L = 2$ and 3. At the same time, we also investigate constraints on the refinement. We adopt two different strategies of refinement to illustrate accuracy of the method and the required time of computation under the respective conditions using two approaches of refinement. The first is the independent level refinement (ILR), where we set the distance between the edges of the next level less than 0.05 of the coarser grid for each refinement. This approach is more suitable for preserving the numerical accuracy and order of accuracy, but it may require more computational time. The other is called Variable Level Refinement (VLR), where we set the distance between the edges of next grid level less than $0.2 \times d^{d-L}$ for each of the level $L > 1$ with $L = 1$ being the coarsest. Here, $d = \frac{1}{2}$ is used. The grid structures are shown in Figure 2 for $N_k = 32, k = 5$ and $L = 3$ for the $D_2$ disk on the left and the $D_{2.2}$ disk on the right for two refinement strategies ILR on the top and VLR on the bottom.

The results of $\text{AMR}(L, D_2)$ for refinement levels $L = 2$ and $L = 3$ with the ILR are shown in Table 3. The first and third sub-tables are the errors between the numerical and analytic solutions, $N_k = 2^k$, $k = 7, 8, \ldots, 11$, for $L = 2$ and $L = 3$, respectively. The order of accuracy is listed in the second and fourth sub-tables according to $L = 2$ and
### Table 2. Data for AMR($L,D_{2,2}$) simulation for $D_{2,2}$ disk with the uniform $L = 1$ discretization. $L_{\alpha,\beta}$ calculates errors between the numerical and analytic solutions for the $x$-direction ($\alpha = x$) and the radial direction ($\alpha = R$) in $\beta = 1, 2, \infty$ norms, respectively. The top portion of the table shows the errors for $N_k = 1, 2, \infty$-norm. Such trends are the same as those shown in Table 1.

$L = 3$, respectively. It is shown to be of nearly second order in both the $x$ and $R$ directions for the 1 and 2 norms. However, it is only about 1.4 for the $\infty$-norm. Although the orders of accuracy are almost the same for both the uniform and AMR grids, the results obtained for the AMR converge better than those obtained for the uniform grids. This situation is shown in Figure 3. Similar results hold for the case of AMR($L, D_{2,2}$) for $L = 2$ and $L = 3$ with ILR, which are summarized in Table 4.

We also investigate the Variable Level Refinement approach for the $D_2$ disk and $D_{2,2}$ disk. The results of AMR($L,D_2$) for $L = 2$ and $L = 3$ with VLR are shown in Table 5. The first and third sub-tables show the errors, $N_k = 2^k$, for $k = 7, 8, \ldots, 11$, respectively. The order of accuracy is listed in the second and fourth sub-tables according to $L = 2$ and $L = 3$, respectively. From the Table 5, the values for the order of accuracy fall between 0.8 and 3 among the 1, 2, $\infty$-norms. This shows that the convergence can go beyond the second order, which is the phenomenon of super-convergence. Although the order of accuracy fluctuates more for the AMR grid with VLR, the majority of results with the AMR are better converged than some of the uniform grids. Similar situation holds for the AMR($L, D_{2,2}$) simulation for $L = 2$ and $L = 3$ with VLR in Table 6, but with a narrower range of order of accuracy than those of AMR($L,D_2$).

#### 3.2. Comparison of Run-Time on GPUs and on CPUs

The direct $N$-body method puts a strict demand on the computing power to deal with the calculation of self-gravitational force. Parallel computing offers a great advantage in term of performance because it deploys the same calculation to lots of processing units to run in the same amount of time. MPI and OpenMP are popular parallel schemes for the standard computers.

Until about a decade ago did the potential of using graphical processing units (GPU) for general purpose applications become viable. In 2006, Nvidia Corporation introduced a general purpose parallel programming architecture that uses their parallel computing engine GPUs. The platform, Compute Unified Device Architecture, abbreviated as CUDA, is a software layer that gives direct access to the virtual instruction set of Nvidia GPUs and parallel computational elements. In addition, it solves complex computational problems in a more efficient way with much better performance than a CPU does, especially on direct $N$-body calculation like the one illustrated in this paper. OpenACC is another different parallel programming model which is user-driven, and directives-based on heterogeneous hardware platforms and architectures. Just like OpenMP, user can add few directives into the source code and specify how a compiler should process these parts with GPU. The OpenACC specifications supports Fortran, C, C++ programming language and a variety of different architectures including AMD HD GPUs, Nvidia GPUs and Intel Xeon Phi. Although the performance of OpenACC is not good as CUDA, OpenACC is easier to implement into an existing code.

In this paper, we employ both Fortran and CUDA languages in the simulations for comparison of performance. The GPU computations were run entirely on Nvidia Tesla P100 with CUDA. Tesla P100 is powered by the Pascal GPU microarchitecture and was released in 2016. Each Tesla P100 has 3584 Nvidia CUDA cores with a GPU clock speed
of 1189 MHz. The double-precision performance is 4.7 teraFLOPS according to the Nvidia website. The Fortran CPU code runs on Intel Xeon E5-2698 operating at 2.2GHz. The Xeon E5-2698 is a 64-bit icosahedral x86 microprocessor introduced by Intel in 2016. Although Xeon E5-2698 has 20 cores, each Fortran benchmark runs on a single core.

A direct method as illustrated here is a time intensive one. It also requires investigations on how realistically capabilities of the methodology can be carried with available computational resources. For this purpose, we compare simulations carried out with the Fortran and CUDA platforms. The AMR(1,$D_2$) and AMR(1,$D_{2,2}$) were carried out with both Fortran and CUDA platforms. Table 7 shows that the CUDA platform is faster than the Fortran one by about 263 times, i.e, $CT(\text{Fortran}) = 263 \times CT(\text{CUDA})$, where $CT$ stands for the computational time.
We demonstrated a direct method of second order accuracy for self-gravitational force calculation based on AMR by numerical simulations. The AMR grids can further improve the accuracy but preserving the order of accuracy, as shown in Figures 3 and 4 for the disk $D_2$ and the disk $D_{2,2}$, respectively. The values of errors are from Table 1 and Table 2 for uniform grids, from Table 3 and Table 4 for AMR with ILR, and from Table 5 and Table 6 for AMR with VLR. The slopes of the lines on the $\log_2(\text{error})$-$\log_2(N_k)$ plots give the order of accuracy while the absolute $\log_2(\text{error})$ position give the accuracy of the computation. These profiles in Figures 3 and 4 demonstrate that the $L = 3$ refinement indeed gives the most accurate calculations among the simulations in this work for $D_2$ and $D_{2,2}$ disks, and likewise, the $L = 2$ refinement is better than the $L = 1$ uniform grid. Profiles of errors in the $x$, $y$ and radial directions for the

| $L_x=1$ | $L_x=2$ | $L_x=\infty$ | $L_R=1$ | $L_R=2$ | $L_R=\infty$ |
|---|---|---|---|---|---|
| $N_k=128$ | 1.67E-04 | 1.58E-04 | 1.90E-03 | 2.55E-04 | 2.22E-04 | 1.96E-03 |
| $N_k=256$ | 4.33E-05 | 4.29E-05 | 9.52E-04 | 6.71E-05 | 6.04E-05 | 9.84E-04 |
| $N_k=512$ | 1.10E-05 | 1.12E-05 | 3.28E-04 | 1.72E-05 | 1.57E-05 | 3.46E-04 |
| $N_k=1024$ | 2.80E-06 | 2.90E-06 | 1.24E-04 | 4.38E-06 | 4.09E-06 | 1.36E-04 |
| $N_k=2048$ | 7.04E-07 | 7.53E-07 | 4.91E-05 | 1.10E-06 | 1.06E-06 | 4.93E-05 |

Table 3. The simulation of AMR($L, D_2$) for $D_2$ disk with $L = 2$ and 3 independent level refinement (ILR). $L_{\alpha,\beta}$ represents the domain of accuracy in the second and fourth sub-tables according to Table 2 for uniform grids, from Table 3 and Table 4 for AMR with ILR, and from Table 5 and Table 6 for AMR with VLR. The behavior is similar for VLR as reported in Table 9.

This can be seen directly in Table 7, the column of Fortran/CUDA at $N_k = 512$. Therefore, GPU can rise to the task for simulations in practice and we compare the performance solely in the simulations with AMR grids for the two refinement strategies, ILR and VLR, as below. Tables 8 and 9 summarize computational times for AMR($L, D_2$) and AMR($L, D_{2,2}$) based only on the CUDA platform. On average, the computational time for the AMR, CT(AMR), is about 8 times of the computational time for the uniform grids (UG, $L = 1$), CT(UG), from the column AMR/UG. The complexity of computational times for self-gravity is related to the number of leaf zones. In Table 8, the $EL/(N_k^2)$ ratios for $N_k = 2^k$, $k = 7, 8, \ldots, 11$ and $D_2$ using ILR are fixed at 1.12 and 1.36 for $L = 2$ and $L = 3$, respectively. The $EL$ is the number of the leaf zones which are interacting with each other by gravity for the AMR. The $N_k^2$ is the zone number of the uniform grid. The higher $EL/(N_k^2)$ ratio, the more gravity calculation for the AMR. Likewise for $D_{2,2}$, the ratios are 1.19 and 1.57 for $L = 2$ and $L = 3$, respectively. The computational time increases exponentially with $N_k$. The behavior is similar for VLR as reported in Table 9.

4. DISCUSSION AND CONCLUSION

We demonstrated a direct method of second order accuracy for self-gravitational force calculation based on AMR by numerical simulations. The AMR grids can further improve the accuracy but preserving the order of accuracy, as shown in Figures 3 and 4 for the disk $D_2$ and the disk $D_{2,2}$, respectively. The values of errors are from Table 1 and Table 2 for uniform grids, from Table 3 and Table 4 for AMR with ILR, and from Table 5 and Table 6 for AMR with VLR. The slopes of the lines on the $\log_2(\text{error})$-$\log_2(N_k)$ plots give the order of accuracy while the absolute $\log_2(\text{error})$ position give the accuracy of the computation. These profiles in Figures 3 and 4 demonstrate that the $L = 3$ refinement indeed gives the most accurate calculations among the simulations in this work for $D_2$ and $D_{2,2}$ disks, and likewise, the $L = 2$ refinement is better than the $L = 1$ uniform grid. Profiles of errors in the $x$, $y$ and radial directions for the
Table 4. AMR($L,D_{2,2}$) simulation for $D_{2,2}$ disk with $L = 2$ and 3 independent level refinement (ILR). $L_{\alpha,\beta}$ represents the errors of forces between numerical and analytic solutions for $x$-direction ($\alpha = x$) and radial direction ($\alpha = R$) in $\beta = 1, 2, \infty$ norm. The first and third sub-tables show the errors, $N_k = 2^k$, $k = 7, 8, \ldots, 11$, for $L = 2$ and $L = 3$, respectively. The order of accuracy in the second and fourth sub-tables according to $L = 2$ and $L = 3$, respectively, show that the 1- and 2-norm are nearly of second order for both $x$ and $R$ directions, but roughly about 1.4 for $\infty$-norm.

$L = 3$ AMR($3, D_{2,2}$) with ILR using $N_k = 2048$ are shown in Figure 5. It illustrates that most errors occur near the edges of the disks.

Comparing the achieved accuracy and computational time with $N_k$, the ILR approach behaves well in terms of the order of accuracy. That is, the slopes of lines for ILR simulations in Figure 3 are almost identical for the levels of $L = 1, 2, 3$. On the other hand, the computational times of VLR are always less than those of the ILR at $N_k = 512$ for $L = 2, 3$. Although the order of accuracy for VLR fluctuates between 0.88 and 3 for the $D_2$ disk and between 1.11 and 2.94 for the $D_2$ disk, the accuracy of VLR is still relatively better than that of ILR. For situation that requires better refinement, the VLR is better suited than the ILR.

The speedup of approximately two orders of magnitude by CUDA is significant in allowing the direct method to become a realistic option on GPUs. In astrophysical simulations, a single galaxy simulation may require $512 \times 512$ grid zones with $10^4$ iterations. Under such conditions, the computational time required is about 10 days ($10^4 \times 8$ seconds). To reduce the computational time to 8 hours, 12 cards of P100 GPUs have to be deployed at once. Further improvement of the performance on GPU can be made through optimization of algorithms based on different GPU architectures as a followup to the calculations reported in this paper.

For the case of uniform grid discretization, the double sum of the potential function has been represented as a convolution form of the surface density and the fundamental kernel (Yen et al 2012). A direct calculation requires the computational complexity of $O(N^4)$ if the number $N$ is the number of zones in one direction. The fast computation of Yen et al. (2012) can reduce the complexity $O(N^4)$ to nearly linear $O((N \log_2 N)^2)$ with the help of FFT. Such a comparison with the studies of multi-grid and spectral method has been given in Yen et al. (2012) for the uniform grid. Here in this work, we compare the current approach that has been built on the extension of grid structure from...
the uniform discretization to one with adaptive mesh refinement. This generalization has been shown to preserve the order of accuracy successfully. The FFT technique can not be employed here to reduce computational time for AMR.

In this research, it should be emphasized that the formula $K_{y,0}^{x,0}$ should be modified for AMR. To reduce the computational time (Bellemann et al 2008), various directions can still be explored. The most direct is to improve the availability of powerful computational facility. Building a tree structure and implementing an approach which is similar to the traditional $N$-body calculation will be considered next. Finally, the framework of Cartesian coordinate calculation can also be generalized to Polar coordinates calculation, following Yen et al. (2012) if the second order of accuracy in the second and fourth sub-tables according to Eqn (4.13) in Yen et al. (2012) is always positive for AMR.

The authors acknowledge the support of the Theoretical Institute for Advanced Research in Astrophysics (TIARA) based in Academia Sinica Institute of Astronomy and Astrophysics (ASIAA). C.C.Y. thanks the Institute of Astronomy
| AMR(2,D_{2,2})-N_{k} | L_{x,1} | L_{x,2} | L_{x,\infty} | L_{y,1} | L_{y,2} | L_{y,\infty} | L_{R,1} | L_{R,2} | L_{R,\infty} |
|----------------------|--------|--------|-------------|--------|--------|-------------|--------|--------|-------------|
| 128                  | 6.20E-04 | 7.31E-04 | 3.31E-03 | 5.42E-04 | 7.03E-04 | 3.26E-03 | 6.28E-04 | 7.28E-04 | 3.15E-03 |
| 256                  | 1.13E-04 | 1.55E-04 | 1.03E-03 | 9.84E-05 | 1.49E-04 | 1.06E-03 | 1.15E-04 | 1.55E-04 | 1.10E-03 |
| 512                  | 2.30E-05 | 3.82E-05 | 4.69E-04 | 2.07E-05 | 3.76E-05 | 4.76E-04 | 2.34E-05 | 3.82E-05 | 4.70E-04 |
| 1024                 | 5.71E-06 | 1.10E-05 | 1.97E-04 | 5.41E-06 | 1.10E-05 | 1.98E-04 | 5.77E-06 | 1.10E-05 | 1.97E-04 |
| 2048                 | 1.56E-06 | 3.25E-06 | 7.28E-05 | 1.52E-06 | 3.24E-05 | 7.29E-05 | 1.56E-06 | 3.25E-06 | 7.32E-05 |

| AMR(2,D_{2,2})-N_{k}/N_{k+1} | O_{x,1} | O_{x,2} | O_{x,\infty} | O_{y,1} | O_{y,2} | O_{y,\infty} | O_{R,1} | O_{R,2} | O_{R,\infty} |
|------------------------------|--------|--------|-------------|--------|--------|-------------|--------|--------|-------------|
| 128/256                     | 2.46   | 2.24   | 1.59        | 2.46   | 2.24   | 1.62        | 2.45   | 2.23   | 1.52        |
| 256/512                     | 2.30   | 2.02   | 1.14        | 2.25   | 1.99   | 1.16        | 2.30   | 2.02   | 1.23        |
| 512/1024                    | 2.01   | 1.80   | 1.25        | 1.94   | 1.77   | 1.27        | 2.02   | 1.80   | 1.25        |
| 1024/2048                   | 1.87   | 1.76   | 1.44        | 1.83   | 1.76   | 1.44        | 1.89   | 1.76   | 1.43        |

| AMR(3,D_{2,2})-N_{k} | L_{x,1} | L_{x,2} | L_{x,\infty} | L_{y,1} | L_{y,2} | L_{y,\infty} | L_{R,1} | L_{R,2} | L_{R,\infty} |
|----------------------|--------|--------|-------------|--------|--------|-------------|--------|--------|-------------|
| 128                  | 2.87E-04 | 4.14E-04 | 1.73E-03 | 2.36E-04 | 3.75E-04 | 1.75E-03 | 2.91E-04 | 4.14E-04 | 1.86E-03 |
| 256                  | 3.87E-05 | 7.30E-05 | 4.15E-04 | 3.08E-05 | 6.66E-05 | 4.20E-04 | 3.95E-05 | 7.32E-05 | 4.20E-04 |
| 512                  | 5.52E-06 | 1.63E-05 | 1.87E-04 | 4.93E-06 | 1.60E-05 | 1.97E-04 | 5.80E-06 | 1.64E-05 | 1.95E-04 |
| 1024                 | 1.85E-06 | 3.53E-06 | 7.89E-05 | 1.80E-06 | 5.55E-06 | 8.03E-05 | 1.90E-06 | 5.54E-06 | 7.91E-05 |
| 2048                 | 5.26E-07 | 1.85E-06 | 3.21E-05 | 5.14E-07 | 1.85E-06 | 3.23E-05 | 5.27E-06 | 1.85E-06 | 3.21E-05 |

| AMR(3,D_{2,2})-N_{k}/N_{k+1} | O_{x,1} | O_{x,2} | O_{x,\infty} | O_{y,1} | O_{y,2} | O_{y,\infty} | O_{R,1} | O_{R,2} | O_{R,\infty} |
|------------------------------|--------|--------|-------------|--------|--------|-------------|--------|--------|-------------|
| 128/256                     | 2.89   | 2.50   | 2.06        | 2.94   | 2.49   | 2.06        | 2.88   | 2.50   | 2.15        |
| 256/512                     | 2.81   | 2.16   | 1.15        | 2.64   | 2.06   | 1.09        | 2.77   | 2.16   | 1.11        |
| 512/1024                    | 1.55   | 1.56   | 1.24        | 1.45   | 1.53   | 1.29        | 1.61   | 1.57   | 1.30        |
| 1024/2048                   | 1.84   | 1.58   | 1.30        | 1.81   | 1.59   | 1.31        | 1.85   | 1.58   | 1.30        |

Table 6. AMR(L,D_{2,2}) simulation for D_{2,2} disk with L = 2 and 3 with various level refinement (VLR). L_{\alpha,\beta} represents the errors of forces between numerical and analytic solutions for x-direction (\alpha = x) and radial direction (\alpha = R) in \beta = 1, 2, \infty norm. The first and third sub-tables show the errors, N_{k} = 2^k, k = 7, 8, \ldots, 11, for L = 2 and L = 3, respectively. The order of accuracy in the second and fourth sub-tables according to L = 2 and L = 3, respectively, dense in N_{k} for each norm, 1, 2, \infty-norm between 1.11 and 2.94. The result is the same as Table 5.

| AMR(1, D_{2})-N_{k} | Fortran (sec) | CUDA (sec) | Fortran/CUDA |
|----------------------|---------------|------------|--------------|
| 128                  | 1.58E+2       | 4.29E+0    | 3.68E+1      |
| 256                  | 2.71E+3       | 7.28E+0    | 3.72E+2      |
| 512                  | 4.11E+4       | 1.56E+1    | 2.63E+2      |
| 1024                 | -              | 1.71E+2    | *            |
| 2048                 | -              | 2.63E+3    | *            |

| AMR(1, D_{2,2})-N_{k} × N_{k} | Fortran (sec) | CUDA (sec) | Fortran/CUDA |
|-----------------------------|---------------|------------|--------------|
| 128 × 128                   | 1.62E+2       | 4.39E+0    | 3.69E+1      |
| 256 × 256                   | 2.68E+3       | 5.29E+0    | 5.07E+2      |
| 512 × 512                   | -              | 1.57E+1    | *            |
| 1024 × 1024                 | -              | 1.71E+2    | *            |
| 2048 × 2048                 | -              | 2.62E+3    | *            |

Table 7. The computational time for AMR(1,D_{2}) (top) and AMR(1,D_{2,2}) (bottom) simulation by Fortran and CUDA. The symbol “-” means that the computational time is more than one week and “*” means that no calculation. The result shows that CUDA is faster than Fortran about 263 times at N_{k} = 512.
Table 8. The computational time for AMR($L, D_{2}$) and AMR($L, D_{2}, 2$) for the level $L = 2$ and 3 simulations with ILR by CUDA. The $N_k$ is the number of zones in one dimension. The numbers in $L = 2$ or $L = 3$ are the total numbers of zones, $EL$ (leaf) is the corresponding number of leaf zones. CT means the computational time. The $EL/N_k^2$ is the ratio of the number of leaf zones divided by $N_k$. The ratios $CT(N_k)/CT(N_k-1)$ show that the computational time exponentially grows with $N_k$ for $k = 8, 9, 10, 11$. 

and Astrophysics, Academia Sinica, Taiwan for their constant support. C.C.Y. is supported by Ministry of Science and Technology of Taiwan, under grant MOST-107-2115-M- 030-005-MY2.

APPENDIX

A. DATA OF AMR ACCORDING TO FIGURE 1.

REFERENCES

Belleman R.G., Bédorf J., Portegies Z., & Simon F. 2008, New Astronomy, 13, 103B.
Berger M. J., Oliger J., 1984, JCoPh, 53, 484.
Elmegreen, D. M., Elmegreen, B. G., Erroz-Ferrer, S., et al. 2014, ApJ, 780, 32
Gupta, M. M., Kouatchou, J., & Zhang, J. 1997, JCoPh, 132, 226
Inutsuka, S.-i., Machida, M. N., & Matsumoto, T. 2010, ApJL, 718, L58
James, 1977, JCoPh, 25, 71
Kim, W.-T., Seo, W.-Y., & Kim, Y. 2012, ApJ, 758, 14
Lai, M.-C., & Tseng, J.-M. 2007, JCoAM, 201, 175
Lin, L.-H., Wang, H.-H., Hsieh, P.-Y., et al. 2013, ApJ, 771, 8
Lee, W.-K., & Shu, F. H. 2012, ApJ, 756, 45
Lee, W.-K. 2014, ApJ, 792, 122
Seo, W.-Y., & Kim, W.-T. 2014, ApJ, 792, 47
Shulz, E., 2009, ApJ, 693, 1310.
Wang, H.-H., Taam, R. E., Yen, C.-C., 2016, ApJS, 224, 16
| AMR($2$, $D_2$)-$N_k$ | $L = 1$; $N_k^2$ | $L = 2$ | $EL$(leaf) | $EL/N_k^2$ | CT(sec) | CT($N_k$)/CT($N_{k-1}$) | AMR/UG |
|----------------------|------------------|--------|------------|-------------|--------|-----------------------|--------|
| 128                  | 16384            | 26672  | 24100      | 1.47        | 4.61E+0 | –                     | 1.07   |
| 256                  | 65536            | 86032  | 80908      | 1.23        | 8.08E+0 | 1.75                  | 1.11   |
| 512                  | 262144           | 303344 | 293044     | 1.18        | 4.64E+1 | 5.74                  | 2.97   |
| 1024                 | 1048576          | 1130944| 1110352    | 1.06        | 5.99E+2 | 12.91                 | 3.50   |
| 2048                 | 4194304          | 4358832| 4317700    | 1.03        | 9.02E+3 | 15.06                 | 3.43   |
| AMR($3$, $D_2$)-$N_k$| $L = 1$; $N_k^2$ | $L = 3$ | $EL$(leaf) | $EL/N_k^2$ | CT(sec) | CT($N_k$)/CT($N_{k-1}$) | AMR/UG |
| 128                  | 16384            | 47168  | 39472      | 2.41        | 5.36E+0 | –                     | 1.25   |
| 256                  | 65536            | 127323 | 111008     | 1.71        | 1.25E+1 | 2.33                  | 1.72   |
| 512                  | 262144           | 385712 | 354820     | 1.35        | 6.98E+1 | 5.58                  | 4.47   |
| 1024                 | 1048576          | 1295472| 1233748    | 1.18        | 7.94E+2 | 11.37                 | 4.64   |
| 2048                 | 4194304          | 4688352| 464840     | 1.09        | 1.09E+4 | 13.73                 | 4.14   |
| AMR($2$, $D_{2,2}$)-$N_k$ | $L = 1$; $N_k^2$ | $L = 2$ | $EL$(leaf) | $EL/N_k^2$ | CT(sec) | CT($N_k$)/CT($N_{k-1}$) | AMR/UG |
| 128                  | 16384            | 32928  | 28792      | 1.76        | 4.97E+0 | –                     | 1.13   |
| 256                  | 65536            | 98400  | 90184      | 1.37        | 9.25E+0 | 1.86                  | 1.75   |
| 512                  | 262144           | 327904 | 311464     | 1.19        | 7.03E+1 | 7.60                  | 4.48   |
| 1024                 | 1048576          | 1180320| 1147384    | 1.09        | 2.01E+3 | 28.59                 | 11.75  |
| 2048                 | 4194304          | 4458080| 4392136    | 1.05        | 8.45E+4 | 42.04                 | 32.13  |
| AMR($3$, $D_{2,2}$)-$N_k$ | $L = 1$; $N_k^2$ | $L = 3$ | $EL$(leaf) | $EL/N_k^2$ | CT/sec | CT($N_k$)/CT($N_{k-1}$) | AMR/UG |
| 128                  | 16384            | 65792  | 53440      | 3.26        | 7.07E+0 | –                     | 1.61   |
| 256                  | 65536            | 164160 | 139504     | 2.13        | 1.59E+1 | 2.25                  | 3.01   |
| 512                  | 262144           | 459648 | 410272     | 1.57        | 1.27E+2 | 7.99                  | 8.09   |
| 1024                 | 1048576          | 1444096| 1345216    | 1.28        | 2.53E+3 | 19.92                 | 14.80  |
| 2048                 | 4194304          | 4985152| 4787440    | 1.14        | 1.28E+5 | 50.59                 | 48.86  |

Table 9. The computational time for AMR($L, D_2$) and AMR($L, D_{2,2}$) for the level $L = 2$ and $3$ simulations with VLR by CUDA. $N_k$ is number of zones in one dimension. The numbers in $L = 2$ or $L = 3$ are the total numbers of zones, $EL$(leaf) is the corresponding number of leaf zones. CT stands for the computational time. The $EL/N_k^2$ is the ratio of the number of leaf zones divided by $N_k^2$. The ratios CT($N_k$)/CT($N_{k-1}$) shows that the computational time exponentially grows with $N_k$ for $k = 8, 9, 10, 11$

Wang, H.-H., Yen, D. C. C., & Taam, R. E. 2015, ApJS, 221, 4
Yen, C.-C., Taam, R. E., Yeh, K. H.-C., & Jea, K. C. 2012, JCoPh, 231, 8246
Zhang, H., Liu, H.-G., Zhou, J.-L., & Wittenmyer, R. A. 2014, RAA, 14, 433
Zhang, H., Yuan, C., Lin, D. N. C., & Yen, D. C. C. 2008, ApJ, 676, 639
Figure 3. The behavior of accuracy for AMR($L$, $D_2$) simulations. The dotted, dashed, and solid lines represent the level $L = 1, 2, 3$, respectively. The symbols “x”, “*” and “o” are for the cases of uniform grids, the AMR-ILR and AMR-VLR, respectively. The accuracy are shown in the 1, 2, and $\infty$-norms in the panels from left to right. The slopes of the fitted lines give the order of accuracy while their positions indicate their respective accuracy. The top row shows the force calculations in the $x$-direction, and the bottom row is for the radial force calculations. It shows that they have the same order of accuracy, but the accuracy from the $L = 3$ refinement is the best.
Figure 4. The behavior of accuracy for AMR($L, D_{2,2}$) simulations. The dotted (red), dashed (blue), and solid (cyan) lines label the level of refinement $L = 1, 2, 3$, respectively. The symbols “x”, “*” and “o” label the uniform grid, AMR-ILR and AMR-VLR, respectively. The accuracy of the 1, 2, and $\infty$-norms is shown in the columns from left to right. Slopes of the respective lines represent the order of accuracy. From top to bottom rows: $x$-direction force, $y$-direction force, and the radial force.
Figure 5. Contours of errors in $x$, $y$ and $R$ directions (from left to right) for the AMR($3$, $D_{2,2}$) simulations are shown with ILR using $N_k = 2048$. Larger errors occur near the edges of the disks.
| ID | $x_c$ | $y_c$ | L | P | EL | LN | DN | RN | UN |
|----|------|------|---|---|----|----|----|----|----|
| 1  | 0.13 | 0.13 | 1 | 0 | 1  | -1 | -1 | 2  | 5  |
| 2  | 0.38 | 0.13 | 1 | 0 | 1  | 1  | -1 | 3  | 6  |
| 3  | 0.63 | 0.13 | 1 | 0 | 2  | -1 | 4  | 7  |
| 4  | 0.88 | 0.13 | 1 | 0 | 1  | -1 | -1 | 8  |
| 5  | 0.13 | 0.38 | 1 | 0 | 1  | -1 | 1  | 6  | 9  |
| 6  | 0.38 | 0.38 | 1 | 0 | 0  | 5  | 2  | 7  | 10 |
| 7  | 0.63 | 0.38 | 1 | 0 | 1  | 6  | 3  | 8  | 11 |
| 8  | 0.88 | 0.38 | 1 | 0 | 1  | 7  | 4  | -1 | 12 |
| 9  | 0.13 | 0.63 | 1 | 0 | 1  | -1 | 5  | 10 | 13 |
| 10 | 0.38 | 0.63 | 1 | 0 | 1  | 9  | 6  | 11 | 14 |
| 11 | 0.63 | 0.63 | 1 | 0 | 0  | 10 | 7  | 12 | 15 |
| 12 | 0.88 | 0.63 | 1 | 0 | 1  | 11 | 8  | -1 | 16 |
| 13 | 0.13 | 0.88 | 1 | 0 | 1  | -1 | 9  | 14 | -1 |
| 14 | 0.38 | 0.88 | 1 | 0 | 1  | 13 | 10 | 15 | -1 |
| 15 | 0.63 | 0.88 | 1 | 0 | 1  | 14 | 11 | 16 | -1 |
| 16 | 0.88 | 0.88 | 1 | 0 | 1  | 15 | 12 | -1 | -1 |
| 17 | 0.31 | 0.44 | 2 | 6 | 0  | 5  | 19 | 18 | 10 |
| 18 | 0.31 | 0.31 | 2 | 6 | 1  | 17 | 20 | 7  | 10 |
| 19 | 0.44 | 0.31 | 2 | 6 | 0  | 5  | 2  | 20 | 17 |
| 20 | 0.44 | 0.44 | 2 | 6 | 1  | 19 | 2  | 7  | 18 |
| 21 | 0.56 | 0.69 | 2 | 11| 1  | 10 | 23 | 22 | 15 |
| 22 | 0.56 | 0.56 | 2 | 11| 1  | 21 | 24 | 12 | 15 |
| 23 | 0.69 | 0.56 | 2 | 11| 1  | 10 | 7  | 24 | 21 |
| 24 | 0.69 | 0.69 | 2 | 11| 1  | 23 | 7  | 12 | 22 |
| 25 | 0.28 | 0.47 | 3 | 17| 1  | 5  | 27 | 26 | 10 |
| 26 | 0.28 | 0.41 | 3 | 17| 1  | 25 | 28 | 18 | 10 |
| 27 | 0.34 | 0.41 | 3 | 17| 1  | 5  | 19 | 28 | 25 |
| 28 | 0.34 | 0.47 | 3 | 17| 1  | 27 | 19 | 18 | 26 |
| 29 | 0.41 | 0.34 | 3 | 19| 1  | 5  | 31 | 30 | 17 |
| 30 | 0.41 | 0.28 | 3 | 19| 1  | 29 | 32 | 20 | 17 |
| 31 | 0.47 | 0.28 | 3 | 19| 1  | 5  | 2  | 32 | 29 |
| 32 | 0.47 | 0.34 | 3 | 19| 1  | 31 | 2  | 20 | 30 |

**Table 10.** Data of AMR according to Figure 1.