An Improved GPU-based Ray-shooting Code for Gravitational Microlensing

Wenwen Zheng1,2, Xuechun Chen1,2, Guoliang Li1, and Hou-Zun Chen1,2

1 Purple Mountain Observatory, Chinese Academy of Sciences, Nanjing, Jiangsu, 210023, People’s Republic of China; guoliang@pmo.ac.cn
2 School of Astronomy and Space Science, University of Science and Technology of China, Hefei, Anhui, 230026, People’s Republic of China

Received 2022 February 14; revised 2022 March 31; accepted 2022 April 19; published 2022 June 1

Abstract

We present an improved inverse-ray-shooting code based on graphics processing units (GPUs) to generate microlensing magnification maps. In addition to introducing GPUs to accelerate the calculations, we also invest effort into two aspects: (i) A standard circular lens plane is replaced by a rectangular one to reduce the number of unnecessary lenses as a result of an extremely prolate rectangular image plane. (ii) An interpolation method is applied in our implementation, achieving significant acceleration when dealing with the large number of lenses and light rays required by high-resolution maps. With these applications, we have greatly reduced the running time while maintaining high accuracy: The speed was increased by about 100 times compared with an ordinary GPU-based inverse-ray-shooting code and a GPU-D code when handling a large number of lenses. If a high-resolution situation with up to 10,000² pixels, resulting in almost 10¹¹ light rays, is encountered, the running time can also be reduced by two orders of magnitude.

Unified Astronomy Thesaurus concepts: Gravitational microlensing (672); Gravitational lensing (670)

1. Introduction

Gravitational lensing studies on the source light deflected by a foreground lens have become a booming field since the first discovery of a multi-imaged quasar (Walsh et al. 1979). In the interim, Chang & Refsdal (1979) realized that the compact mass distributions within lens galaxies may further impact the light and split the image into subimages, the angular separation of which is too small to resolve; this is called the cosmological microlensing effect. A decade later, unambiguous evidence for the brightness change of one image of the quasar system Q2237+0305 was presented by Irwin et al. (1989), representing the first detected microlensing event. Since then, microlensing has been developed as a powerful tool for several areas, such as probing the inner structures of quasars, e.g., the accretion disc (Mortonson et al. 2005; Morgan et al. 2010; Chartas et al. 2016) and the quasar broad-line region (Sluse et al. 2012; Guerras et al. 2013; Fian et al. 2018), and constraining the compact matter distribution of lens galaxies (Wyithe et al. 2000; Wyithe & Turner 2001; Congdon et al. 2007; Mediavilla et al. 2009, 2017; Jiménez-Vicente & Mediavilla 2019).

An essential part of microlensing studies is generating magnification maps of the source plane. The standard method for producing microlensing magnification patterns is via inverse ray shooting (IRS), which was first described by Kayser et al. (1986) and Schneider & Weiβ (1986). One shoots a large number of light rays from the observer (image plane) to the source plane and then calculates the deflection angle caused by hundreds to millions of microlenses. However, brute-force IRS contains a large number of operations, which is proportional to the number of light rays ($N_{\text{rays}}$) and microlenses ($N_a$), resulting in catastrophic time consumption when millions of lenses are required in the parameter space while obtaining a magnification map with satisfactory resolution and accuracy.

Great effort has been expended to deal with this problem in two ways: (i) to perform the computation in parallel and (ii) to reduce the number of lenses ($N_a$) or light rays ($N_{\text{rays}}$) involved in the calculations. For the first one, different parallel strategies based on both CPUs (Chen et al. 2017) and graphics processing units (GPUs) have been applied to the IRS method, between which GPUs have shown great computational potential (Thompson et al. 2010). For the second, Wambsganss (1999) introduced a hierarchical tree method to reorder the lenses by their distance to individual light rays and group distant lenses into larger cells, allowing the $N_a$ directly involved in the calculations to be reduced. In addition, the inverse polygon mapping (IPM) method described in Mediavilla et al. (2006, 2011) suggests mapping regular polygons instead of individual light rays so that the magnification is related to the ratio of the overlapped area of the mapped polygon and source pixel to the original area. With this approach, the required number of $N_{\text{rays}}$ is greatly reduced to maintain the same accuracy.

Improved algorithms are still coming up. Garsden & Lewis (2010) implemented a CPU-based tree code on supercomputers. Alpay (2019) presented a GPU-parallel tree code. Weisenbach et al. (2021) introduced another GPU-based code that calculates the Taylor coefficients of the four corners of sparse squares and shoots more rays within the squares to use those coefficients to accelerate the computations; all lenses are used in the meantime. Shalypin et al. (2021) brought up a new nonparallel method that combines the IPM algorithm with a Poisson solver for a deflection potential to reduce $N_a$ and $N_{\text{rays}}$ simultaneously.

In this work, we present a GPU-based IRS code implemented with a rectangular lens plane and combined with an interpolation algorithm to improve the efficiency. We call it GPU-PMO (GPU-Parallel Microlensing Optimizer) hereafter. This paper is organized as follows: Section 2 outlines the main procedures of our work, focusing especially on the modified rectangular lens plane, the setting of the “protective border,” and the interpolation method. Section 3 presents the runtime and accuracy comparison between our GPU-PMO code, the
ordinary GPU-based IRS code, and the widely used GPU-D code. In Section 4, we provide a short summary of this work.

2. Methods

2.1. Lens Equation

The core idea of the IRS method is to select a small rectangular “window” on the image plane (usually within a larger circular lens plane) and trace back a large number of light rays from the image window to the source plane based on the lens equation:

\[
\beta = \frac{1 - \kappa - \kappa_s - \gamma}{\kappa - \kappa_s + \gamma} \theta - \frac{N_s}{\kappa_s} \sum_{i=1}^{N_s} \frac{m_i (\theta - \theta_i)}{|\theta - \theta_i|^2},
\]

where \( \beta \) refers to the light positions in the source plane, and \( \theta \) and \( \theta_i \) are, respectively, the light positions and lens positions on the image/lens plane. \( N_s \) stands for the number of microlenses while \( m_i \) denotes their mass. In addition, \( \kappa \) represents the dimensionless surface mass density of the mass sheet of the lens plane and \( \gamma \) represents the external shear; both can be obtained from the strong-lens model.

Because the lens plane can be regarded as a mixture of continuously distributed matter (generally considered to be dark matter) and compact objects such as stars and black holes, \( \kappa \) can be divided into two parts: a smooth part \( \kappa_s \) and a compact part \( \kappa_c \), i.e.,

\[
\kappa = \kappa_s + \kappa_c.
\]

The lens Equation (1) is converted into angular coordinates and scaled by the Einstein angle \( \theta_E \) of a point mass:

\[
\theta_E = \sqrt{\frac{4G \langle M \rangle}{c^2 D_{ls} D_{ls}^{-1}}},
\]

where \( \langle M \rangle \) stands for the average mass of the microlenses, and \( D_{ls}, D_{ls}^{-1} \) and \( D_l \) refer to the angular diameter distances from the lens to the source, from the lens to the observer, and from the source to the observer, respectively.

If Equation (1) is rewritten in the form

\[
\beta = \theta - \alpha(\theta),
\]

we obtain the equation for the deflection angle:

\[
\alpha(\theta) = \begin{pmatrix} \kappa_s + \gamma \\ 0 \end{pmatrix} \theta + \frac{N_s}{\kappa_s} \sum_{i=1}^{N_s} \frac{m_i (\theta - \theta_i)}{|\theta - \theta_i|^2},
\]

which clearly reveals the essence of microlensing: The light rays are affected by both smooth matter and compact objects (microlenses) in their path, resulting in the deflection angles. The first term of Equation (5) represents the contribution from smooth matter and external shear; the second term refers to the cumulative contribution to the deflection angle by each microlens. The main procedure of the IRS method is to solve the deflection angle of each light ray, which is the most time-consuming part. The bright side is that the calculation of each light ray is relatively independent, which is perfectly suitable for GPU parallel processing.

2.2. Modified Rectangular Lens Plane

In our implementation, we choose a rectangular plane that acts as both an image plane and a lens plane, as shown in Figure 3, rather than a rectangular image plane embedded in a larger circular lens plane. Changing a circular lens plane into a rectangular one does not just change the shape, but also the lens equation.

Notably, ignoring the shape of the lens plane, the general form of the deflection angle can be written as

\[
\alpha(\theta) = \begin{pmatrix} \kappa + \gamma \\ 0 \end{pmatrix} \theta + \frac{N_s}{\kappa_s} \sum_{i=1}^{N_s} \frac{m_i (\theta - \theta_i)}{|\theta - \theta_i|^2} + \alpha_{\kappa_0},
\]

where the first term refers to the influence of the local convergence (\( \kappa \), from both the compact and smooth mass) and shear (\( \gamma \)) caused by strong lensing. The second term indicates the contribution from the individual microlenses, whose mass surface mass density is \( \kappa_s \). One may find that the mass sheet with a density of \( \kappa_s \) is redundant, thus we add a third term \( \alpha_{-\kappa_0} \), which represents a negative mass sheet that contains only smooth matter with the same \( \kappa_0 \) to cancel out the “extra mass” from the second term. The sum of the second and the third terms can be regarded as the perturbation caused by compact objects.

The specific form of the third term, which also consists of a “convergence part” and a “shear part,” is associated with the shape of the lens plane. For a circular lens plane (mass sheet), the deflection angle is given by \( \alpha \propto M/\theta = -\kappa_0 \pi \theta \), which is a combination of two vector components: \( \alpha_1 \propto -\kappa_0 \pi \theta_1 \) and \( \alpha_2 \propto -\kappa_0 \pi \theta_2 \), where \( M \) and \( \theta \) denote the mass inside an angular radius \( \theta \). Accordingly, the “shear part” would be

\[
\frac{\gamma_1}{\gamma_2} = \frac{\partial \alpha_1}{\partial \theta_1} - \frac{\partial \alpha_2}{\partial \theta_2} = 0;
\]

therefore, for a circular lens plane, \( \alpha_{-\kappa_0} \) contains only the part of “convergence”:

\[
\alpha_{-\kappa_0} = \begin{pmatrix} \kappa_0 \\ 0 \end{pmatrix} \theta.
\]

Then, we deduce Equation (5) by combining Equation (9) and Equation (6). It is worth noting that Equation (1) and Equation (5) only default to the circular lens plane, which is commonly used; for lens planes with other shapes such as the rectangular one, the “shear part” of \( \alpha_{-\kappa_0} \) is no longer zero.

In this work, we obtained the modified \( \alpha_{-\kappa_0} \) suitable for a rectangular lens plane; please see the Appendix for more details. An example of the deflection angle perturbation for \( \kappa_0 = 0 \) is shown in Figure 1, which is the sum of the second (compact mass) and third terms (negative mass sheet \( \alpha_{-\kappa_0} \)) of Equation (6).

The subsequent question to address might be: Why bother using a rectangular lens plane instead of a circular one? As we know, Equation (1) requires a circular lens plane, which is the circle circumscribed around a rectangular image with a side length ratio of \( [1 - \kappa - \gamma] / [1 - \kappa + \gamma] \). The areas of the lens circle and rectangular image are not significantly different when \( |\gamma| \ll |1 - \kappa| \). However, the area of the circumscribed circle will be dramatically larger than its inscribed rectangle.
when the image plane is extremely long and narrow, and the circumscribed circle will be filled with lots of “wasted lenses,” resulting in excess calculations. The modified lens plane shares the same rectangle as the image plane, solves the problem well, and leads to an easy way to set a “protective border” upon the source plane and corresponding image plane as shown below.

2.3. The “Protective Border”

Another problem comes afterward: light rays from the image plane will not completely fall on the target source plane due to the compact objects in lens galaxies, which leads to the loss of magnification accuracy (Katz et al. 1986; Wambsganss 1999). Our approach is to add a “protective border” around the target source plane as mentioned by Katz et al. (1986), resulting in an enlarged corresponding image/lens plane to ensure that 98% of the light rays fall on the target source plane to maintain accuracy. We follow Katz et al. (1986) to determine the size of the “protected border”: We calculate the probability density distribution of the deflection angle perturbation (as shown in Figure 1), but with a varying number of lenses, presented in Figure 2. The deflection angle is scaled by \( \kappa \), here \( \kappa = \sqrt{\kappa_0} \), and different colors denote different numbers of lenses from \( 10^2 \) to \( 10^7 \). Here we only show \( 10^2, 10^3, 10^4 \), and \( 10^6 \) in Figure 2 for clarity. Solid lines represent the probability density calculated in our realization; dashed lines in the left panel represent the analytical results in Figure 1 of Katz et al. (1986), based on their Equation (13):

\[
\rho_N(\phi) = \frac{1}{2\pi\kappa_0} \int_0^{10^7} dx \exp \left( \frac{-x^2}{2\kappa_0^2} \ln \left( \frac{3.05N^{1/2}}{x} \right) \right).
\]

Considering the term \( \alpha_\kappa \kappa_0 \), in Equation (6), we find an improved Fourier transform of \( \rho_N \):

\[
\tilde{\rho}_N \approx \exp \left( -\frac{Nk^2}{2} \ln \frac{1.454}{k} \right);
\]

this form is consistent with that of Katz’s Equation (9) but with the numerator changed from 3.05 to 1.454. Thus, we replot Equation (10) with this change in the right panel of Figure 2. Please refer to our follow-up work for more details.

As shown in Figure 2, our modified equation fits the numerical results better compared with Katz’s Equation (13). The broadening of the probability density distribution increases with the number of lenses, even for \( 10^4 \) lenses. However, the distribution curves decay rapidly when the deflection angles are larger than \( 4\theta_0 \); the density is below \( 10^{-4} \) at \( 10\theta_0 \). As a result, we add a “protective border” with the length of \( 10\theta_0 \) (20\theta_0 for the side length in each direction) around the target source plane; the size of the corresponding image plane will then be calculated based on this “enlarged” source plane. This process can ensure that more than 98% of the total required light rays fall into the target source plane and achieve a guaranteed accuracy.

2.4. Interpolation Algorithm

As mentioned above, most calculation time costs are for computing deflection angles for individual light rays. Using GPU to parallel-compute light rays just opens the outer loop; however, for each light ray, the inner loop that adds up the contribution of each lens to the deflection angle would significantly slow down the speed, especially when the number of lenses gets larger (Bate et al. 2010).

In this work, we use an interpolation algorithm that is a tree-like method (Wambsganss 1999) to improve computational efficiency. It is mainly about treating lenses differently based on their distance to the individual light rays. We divide the lenses into distant ones and nearby ones as the farther a lens is from the light ray, the less influence it has. Thus, for “distant lenses,” we use interpolation to approximate their contribution to the deflection angle instead of simply summing up the lenses.

We achieve this by setting up three levels of grids in the image/lens plane, as shown in Figure 3. The gray rectangle denotes the image/lens plane, which is organized as the gray level-1 grid to show the lenses (yellow stars). The level-1 grid in the center is chosen to be an example containing a beam of sampled light rays (green points, also called level-3 grid points; in practice, the image plane will be filled with these light rays). This central level-1 grid is further divided into blue level-2 grids (level-2 grid points are marked in red), and eight level-1 grids around it are marked out by purple dotted lines. For target light rays, the lenses inside the purple dashed lines will be treated as the “nearby lenses” and will be accumulated precisely. By contrast, the outside ones will be regarded as the “distant lenses” and will be handled by the subsequent interpolation procedures. The deflection angle will be the sum of the contributions from these two parts.

For “distant lenses”, their contribution will be accumulated upon the red level-2 grid points (for further optimization, the contribution from the negative mass sheet will also be calculated in this step). Then, eight Taylor coefficients up to fourth order of the level-2 grid points can be calculated directly, similar to Appendix A.2 of Wambsganss (1990). Thus, the deflection angle of the target light rays contributed by “distant lenses” and the negative mass sheet can be easily obtained from the local Taylor expansion during the calculation of level-3 grids.

Figure 4 shows the flowchart of the GPU-PMO code, i.e.,

1. **Input parameters:**
   - \( \kappa, \gamma, f_* \): obtained from the strong-lens model,
   - \( f_* = \kappa_0/\kappa \);
   - \( S_{\text{scale}} \) (side length of the source plane, scaled with \( \theta_0 \)), \( N_{\text{pix}} \) (number of source pixels), \( N_{\text{av}} \) (average light rays per pixel): related to specific scientific requirements;
   - \( M_{\text{min}}, M_{\text{max}}, \alpha \): parameters of the microlens mass when following a certain mass distribution.

![Figure 1](image1.png)

**Figure 1.** Deflection angle perturbation in the lens plane. \( \kappa = \kappa_0 = 0.6, \gamma = 0.1, N_s = 238 \), with an equal mass of \( 1 M_\odot \).

![Figure 2](image2.png)

![Figure 3](image3.png)

![Figure 4](image4.png)
2. Calculate the size of the “protective border”:

\[ P_{\text{scale}} = 10^{k} \]

then the side length of image/lens plane will be:

\[ L_x = \frac{(S_{\text{scale}} + 2P_{\text{scale}})}{1 - \kappa - \gamma}, \]
\[ L_y = \frac{(S_{\text{scale}} + 2P_{\text{scale}})}{1 - \kappa + \gamma}. \]

3. Generate lenses:

Generate random positions and masses for lenses following certain mass distributions.

4. Lenses packed into level-1 grids:

The resolution of the level-1 grid:

\[ \Delta x_1 = \min[L_0, \min(L_x, L_y)/10], \]

where

\[ L_0 = \sqrt{(L_x \cdot L_y)/N_0} \]

represents the mean side length of the area occupied by each lens. We choose the smaller of \( L_0 \) or 1/10 of the short side length.

5. Calculate the deflection angle of level-2 grid points:

The contribution of the distant lenses and negative mass sheet (\( \alpha_{-\kappa, \gamma} \) contains complex calculations) are calculated in this step.

The resolution of the level-2 grid:

\[ \Delta x_2 = \Delta x_1/20; \]

apparently, the interpolation accuracy will be reduced if the level-2 grid is too sparse. On the other hand, it is not worth the expense of speed to deal with a dense grid. After a series of tests, we chose \( \Delta x_1/20 \) as a trade-off.

This step is implemented on GPU.
6. Calculate the deflection angle of all the light rays (level-3 grids):

The deflection angle consists of three parts, as described in Equation (6), which will be summed up in this step. The second term (i.e., the contribution from the individual microlenses) is the sum of the contribution from “nearby lenses” obtained from direct accumulation, and the distant lenses obtained from the local Taylor expansion of level-2 grid points. The contribution from the third term (i.e., the negative mass sheet $\kappa_-$) is calculated together with the “distant lenses.”

The resolution of the level-3 grid, which is also the resolution of image plane:

$$\Delta x_3 = \sqrt{(I_x \text{scale} \cdot I_y \text{scale})/N_{\text{rays}}},$$

where $N_{\text{rays}} = N_{\text{av}} \cdot N_{\text{pix}}$, refers to the total number of shooting light rays.

This step is implemented on GPU.

7. Calculate the magnification of each pixel:

$$\text{Mag}_{ij} = N_{ij} \cdot S_5,$$

where $N_{ij}$ represents the number of light rays that fell into each source pixel, and $S_1$ and $S_5$ represent the pixel area of the image plane and the source plane, respectively.

8. Integrate data and write them into the documents

3. Results

A sample magnification map generated with our GPU-PMO code executed by an Nvidia Tesla V100S GPU is shown in Figure 5.

To evaluate the performance of the GPU-PMO code, we run a series of timing and accuracy tests and compare it with the mature GPU-D code.

3.1. Speed

We download the latest GPU-D source code from the website mentioned in Vernardos et al. (2014, 2015). In addition, each performance was run under the same hardware condition with the GPU card Nvidia Tesla V100S GPU 32 GB.

It should be pointed out that GPU-based programs need to be optimized for specific hardware; however, we cannot ensure that the GPU-D code is optimized to fully fit our hardware. Therefore, we take an IRS program without the interpolation method we implemented as a substitute for the optimized GPU-D. In particular, we use the “double” data type in our implementation instead of the “float” type used in GPU-D; we consider it to be necessary when the image plane is extremely large. However, for comparison, both “double” type and “float” type are applied to the GPU-PMO code. Besides, in our implementation, a rectangular lens plane is used instead of the standard circular one that GPU-D uses; therefore, we adjust our realization to fit GPU-D: four codes share the same circular lens plane and random lenses generated by the GPU-D code.

Because the runtime $T$ is proportional to the number of $N_{\text{rays}}$ and $N_{\text{av}}$:

$$T \propto N_{\text{av}} \cdot N_{\text{pix}} \cdot N_{\text{av}} = N_{\text{rays}} \cdot N_{\text{av}},$$

we take $N_{\text{av}}$ and $N_{\text{rays}}$ as variables. Figure 6 and Figure 7 are the timing results as a function of $N_{\text{av}}$ and $N_{\text{pix}}$ (with fixed $N_{\text{av}}$), respectively. In Figure 6, four programs share the same parameters, with fixed $N_{\text{pix}}$ (4096$^2$) and $N_{\text{av}}$ (100), but $N_{\text{av}}$ varying from $10^2$ to $10^6$. The advantage of the interpolation algorithm becomes apparent when the number of lenses is greater than $10^3$. When $N_{\text{av}}$ is up to $10^5$, GPU-PMO (float type) can be almost 100 times faster than ordinary GPU-IRS and GPU-D. A similar conclusion can be drawn from Figure 7: The time consumption is reduced by about two orders of magnitude when $N_{\text{pix}}$ is up to 5000$^2$, which is a real boost because high-resolution maps are required with the development of lensed supernova and other high-precision research (Foxley-Marrable et al. 2018; Suyu et al. 2020).

3.2. Accuracy

We select a set of parameters from the speed comparison of Figure 6 to test their accuracy, where $\kappa = \kappa_\ast = 0.652$, $\gamma = 0$, $N_{\text{av}} = 100$, and $N_{\text{pix}} = 4096^2$ are kept fixed.
\(N_u = 100\), and \(S_{\text{scale}} = 50\theta_E\) with a resolution of \(1024^2\) and uniform mass distribution with \(\langle M \rangle = 0.1M_\odot\), corresponding to \(10^5\) lenses.

Due to the nature of the IRS method—the more light rays, the higher the accuracy of the magnification map—we use the map generated by the ordinary IRS code, which shoots back about \(4\times10^{10}\) rays as the criterion. Then, seven magnification maps with different accuracies (with different numbers of light rays, from \(10,240^2\) to \(71,680^2\)) are generated by GPU-PMO and GPU-D respectively, corresponding to seven colors in Figure 8. Then, we compare the magnification pixel by pixel to measure accuracy: Each pixel carries a different number of light rays \(N_{\text{rays}}\), which leads to different magnification distributions. Thereafter, we order 300 logarithmic bins by \(N_{\text{rays}}\) shown on the x-axis of Figure 8, then we obtain the magnification of the pixels and calculate the rms of the relative error within each bin; the logarithmic values are shown on the y-axis.

It is clear that for both implementations, the errors are decreasing with the increasing number of light rays. The error of GPU-PMO is smaller under all conditions of \(N_{\text{rays}}\). Additionally, the larger error of GPU-D may be due to the Poisson error brought about by the randomly generated light rays, leading to a logarithmic error slope close to \(-1/2\) (blue dotted line) as shown in Figure 8. For the ordinary IRS method with uniformly generated light rays, the slope is about \(-3/4\) (blue dashed–dotted line) as described in Mediavilla et al. (2011). In our realization, we chose to generate the light rays uniformly instead to reduce the error, resulting in a slope of around \(-2/3\) (blue solid line), larger than the Poisson error of \(-1/2\) and smaller than the theoretical approximation of \(-3/4\).

4. Summary

Cosmological microlensing plays an important role in multi-imaged quasar and even supernova systems. With upcoming new surveys, an increasing number of multi-imaged systems can be expected. Extensive simulations for these scenarios can give us a better understanding of the lens mass distributions, inner structures of the sources, and basic parameters of our universe. It implies that large numbers of magnification maps are required to deal with degenerate parameters and obtain statistical results with high confidence. An efficient and accurate strategy for generating magnification maps is essential. In this work, we proposed an optimized GPU-based code to achieve such a goal:

1. We modified the lens equation to use a rectangular lens plane in place of a standard circular one to greatly reduce unnecessary lenses when handling extreme parameters. In the meantime, we add a “protective border” to ensure a guaranteed accuracy of the magnification map.

2. We proposed a GPU-based version of an IRS microlensing simulation improved with an interpolation algorithm. We achieved this by setting up three levels of grids in the image plane to divide the lenses into “nearby lenses” and “distant lenses” for individual light rays; their deflection angles contributed by “nearby lenses” will be calculated precisely. On the other hand, the contribution from “distant lenses” will be handled with the level-2 grid points using the interpolation strategy, and all of the time-consuming parts were accomplished on the GPU in the meantime. This approach greatly reduced the runtime while maintaining accuracy.

3. We achieved excellent speed and accuracy in generating high-resolution and high-precision magnification maps. When compared with ordinary GPU-IRS and GPU-D, the GPU-PMO code can be about 100 times faster with a large number of \(N_u\). When dealing with a high-resolution situation, the time consumption can also be reduced by...
about two orders of magnitude. Additionally, we extended our code to a multiple-GPU computer in order to achieve better performance when faced with more challenging numerical situations.

This code is still being polished; readers are welcome to contact us to obtain the latest version.

The authors thank the anonymous referee for helpful comments. The authors also thank Xinzhong Er and Zuhui Fan for helpful discussions and suggestions. This work is supported by the NSFC (U1931210, No. 11673065, 11273061). We acknowledge the cosmology simulation database (CSD) in the National Basic Science Data Center (NBSDC) and its funding NBSDC-DB-10 (No. 2020000088). We acknowledge the science research grant from China Manned Space Project No. CMS-CSST-2021-A12.

Appendix

Rectangular Lens Plane Correction

Negative mass sheet \(\alpha_{-\kappa} \) for a rectangular lens/image plane: We obtain the deflection angle of a rectangular plane with a smooth mass distribution to a light ray by direct integration,

\[
\alpha_{-\kappa_{\theta=1}} = A \cdot \int_{\theta_2}^{\theta_2} \frac{\theta_1 - \theta'_1}{\theta_1 - \theta'_1^2 + (\theta_2 - \theta'_2)^2} d\theta_1' d\theta_2', \tag{A1}
\]

\[
A = \frac{\kappa_{\theta=1}}{\pi}, \quad \text{when } \theta_E = 1, \tag{A2}
\]

The integral region is the area of the image/lens plane, the \(X\)-axis direction (\(\theta'_1\)) is from \(a_1\) to \(a_2\), \(Y(\theta'_2)\) id from \(b_1\) to \(b_2\), then

\[
\alpha_{-\kappa_{\theta=1}} = A \cdot \int_{b_1}^{b_2} \int_{a_1}^{a_2} \frac{\theta_1 - \theta'_1}{\theta_1 - \theta'_1^2 + (\theta_2 - \theta'_2)^2} d\theta_1' d\theta_2', \tag{A3}
\]

\[
\begin{align*}
\alpha_{-\kappa_{\theta=1}} &= A \cdot \int_{b_1}^{b_2} \int_{a_1}^{a_2} \frac{\theta_1 - \theta'_1}{(\theta_1 - \theta'_1^2) + (\theta_2 - \theta'_2)^2} d\theta_1' d\theta_2' \\
&= A \cdot \frac{1}{2} \ln \left[ \frac{(\theta_1 - \theta_1^2) + (\theta_2 - \theta_2^2)^2}{(\theta_1 - \theta_2^2) + (\theta_2 - \theta_2^2)^2} \right] \\
&- \frac{1}{2} (\theta_2 - \theta_1) \ln \left[ \frac{(\theta_1 - \theta_2^2) + (\theta_2 - \theta_2^2)^2}{(\theta_1 - \theta_2^2) + (\theta_2 - \theta_2^2)^2} \right] \\
&+ \frac{1}{2} (\theta_2 - \theta_1) \ln \left[ \frac{(\theta_1 - \theta_2^2) + (\theta_2 - \theta_2^2)^2}{(\theta_1 - \theta_2^2) + (\theta_2 - \theta_2^2)^2} \right] \\
&- (\theta_1 - \theta_2) \arctan \left[ \frac{\theta_2 - \theta_1}{\theta_2 - \theta_1} \right] \\
&+ (\theta_1 - \theta_2) \arctan \left[ \frac{\theta_2 - \theta_2}{\theta_2 - \theta_2} \right] \\
&- (\theta_1 - \theta_2) \arctan \left[ \frac{\theta_2 - \theta_2}{\theta_2 - \theta_2} \right].
\end{align*}
\]

Similarly,

\[
\alpha_{-\kappa_{\theta=2}} = A \cdot \int_{\theta_1}^{\theta_2} \frac{\theta_2 - \theta'_2}{(\theta_1 - \theta_1' - \theta_2' + \theta_2')^2 + (\theta_2 - \theta'_2)^2} d\theta_1' d\theta_2',
\]

\[
A \cdot \frac{1}{2} \ln \left[ \frac{\theta_2 - \theta_1^2 + (\theta_2 - \theta_2^2)^2}{\theta_2 - \theta_1^2 + (\theta_2 - \theta_2^2)^2} \right] \\
- \frac{1}{2} (\theta_1 - \theta_1) \ln \left[ \frac{\theta_1 - \theta_1^2 + (\theta_2 - \theta_2^2)^2}{\theta_1 - \theta_1^2 + (\theta_2 - \theta_2^2)^2} \right] \\
+ \frac{1}{2} (\theta_1 - \theta_1) \ln \left[ \frac{\theta_1 - \theta_1^2 + (\theta_2 - \theta_2^2)^2}{\theta_1 - \theta_1^2 + (\theta_2 - \theta_2^2)^2} \right] \\
- (\theta_1 - \theta_2) \arctan \left[ \frac{\theta_2 - \theta_1}{\theta_2 - \theta_1} \right] \\
+ (\theta_1 - \theta_2) \arctan \left[ \frac{\theta_2 - \theta_2}{\theta_2 - \theta_2} \right] \\
- (\theta_1 - \theta_2) \arctan \left[ \frac{\theta_2 - \theta_2}{\theta_2 - \theta_2} \right].
\]

Accordingly, the potential for a rectangular area is

\[
\phi_{-\kappa_{\theta=1}} = A \cdot \frac{1}{2} \int_{a_1}^{a_2} \int_{b_1}^{b_2} \ln \left[ \frac{(\theta_1 - \theta_1^2) + (\theta_2 - \theta_2^2)^2}{(\theta_1 - \theta_2^2) + (\theta_2 - \theta_2^2)^2} \right] d\theta_1' d\theta_2',
\]

similar to the form of the equation in Appendix A of Abdelsalam et al. (1998). Bringing in the upper and lower limits of the integral, we get

\[
\phi_{-\kappa_{\theta=1}} = A \cdot \frac{1}{2} \int_{a_1}^{a_2} \int_{\theta_1}^{\theta_2} \ln \left[ \frac{(\theta_1 - \theta_1^2) + (\theta_2 - \theta_2^2)^2}{(\theta_1 - \theta_2^2) + (\theta_2 - \theta_2^2)^2} \right] \\
- (\theta_2 - \theta_1) \arctan \left[ \frac{\theta_1 - \theta_1}{\theta_2 - \theta_2} \right] \\
- (\theta_2 - \theta_1) \arctan \left[ \frac{\theta_1 - \theta_1}{\theta_2 - \theta_2} \right] \\
+ (\theta_1 - \theta_2) \arctan \left[ \frac{\theta_2 - \theta_1}{\theta_2 - \theta_1} \right] \\
- (\theta_1 - \theta_2) \arctan \left[ \frac{\theta_2 - \theta_1}{\theta_2 - \theta_1} \right] \\
+ 3(\theta_2 - \theta_1)(\theta_1 - \theta_2). \tag{A5}
\]
References

Abdelsalam, H. M., Saha, P., & Williams, L. L. R. 1998, MNRAS, 294, 734
Alpay, A. 2019, Teralens, GitHub, https://github.com/illuhad/teralens
Bate, N. F., Fluke, C. J., Barsdell, B. R., Garsden, H., & Lewis, G. F. 2010, NewA, 15, 726
Chang, K., & Refsdal, S. 1979, Natur, 282, 561
Chartas, G., Rhea, C., Kochanek, C., et al. 2016, AN, 337, 356
Chen, B., Kantowski, R., Dai, X., Baron, E., & Van der Mark, P. 2017, A&C, 19, 60
Congdon, A. B., Keeton, C. R., & Osmer, S. J. 2007, MNRAS, 376, 263
Fian, C., Guerras, E., Mediavilla, E., et al. 2018, ApJ, 859, 50
Foxley-Marrable, M., Collett, T. E., Vernardos, G., Goldstein, D. A., & Bacon, D. 2018, MNRAS, 478, 5081
Garsden, H., & Lewis, G. F. 2010, NewA, 15, 181
Guerras, E., Mediavilla, E., Jimenez-Vicente, J., et al. 2013, ApJ, 764, 160
Irwin, M. J., Webster, R. L., Hewett, P. C., Corrigan, R. T., & Jedrzejewski, R. I. 1989, AJ, 98, 1989
Jimenez-Vicente, J., & Mediavilla, E. 2019, ApJ, 885, 75
Katz, N., Balbus, S., & Paczynski, B. 1986, ApJ, 306, 2
Kayser, R., Refsdal, S., & Stabell, R. 1986, A&A, 166, 36
Mediavilla, E., Jimenez-Vicente, J., Munoz, J. A., Vives-Arias, H., & Calderon-Infante, J. 2017, ApJL, 836, L18
Mediavilla, E., Mediavilla, T., Munoz, J. A., et al. 2011, ApJ, 741, 42
Mediavilla, E., Munoz, J. A., Falco, E., et al. 2009, ApJ, 706, 1451
Mediavilla, E., Munoz, J. A., Lopez, P., et al. 2006, ApJ, 653, 942
Morgan, C. W., Kochanek, C. S., Morgan, N. D., & Falco, E. E. 2010, ApJ, 712, 1129
Mortonson, M. J., Schechter, P. L., & Wambsganss, J. 2005, ApJ, 628, 594
Schneider, P., & Weiβ, A. 1986, MPARp, 234, 75
Shalyapin, V. N., Gil-Merino, R., & Goicoechea, L. J. 2021, A&A, 653, A121
Sluse, D., Hutsemékers, D., Courbin, F., Meylan, G., & Wambsganss, J. 2012, A&A, 544, A62
Suyu, S. H., Huber, S., Cañameras, R., et al. 2020, A&A, 644, A162
Thompson, A. C., Fluke, C. J., Barnes, D. G., & Barsdell, B. R. 2010, NewA, 15, 16
Vernardos, G., Fluke, C. J., Bate, N. F., & Croton, D. 2014, ApJS, 211, 16
Vernardos, G., Fluke, C. J., Bate, N. F., Croton, D., & Vohl, D. 2015, ApJS, 217, 23
Walsh, D., Carswell, R. F., & Weymann, R. J. 1979, Natur, 279, 381
Wambsganss, J. 1999, PhD thesis, Munich Univ.
Wambsganss, J. 1999, JCoAM, 109, 353
Weisenbach, L., Schecter, P., & Pontula, S. 2021, ApJ, 922, 70
Wyithe, J. S. B., & Turner, E. L. 2001, MNRAS, 320, 21
Wyithe, J. S. B., Webster, R. L., & Turner, E. L. 2000, MNRAS, 315, 51