Institute for Theoretical Physics at the University of Zurich

Relativistic Simulation of Gravitational Lensing

Programming project for Computational Science 2

Cédric Huwyler

Supervised by
Dr. Saha Prasenjit

May 2009

Contents

1 Introduction

This is the documentation to my Computational Science 2 project. Its goal is to simulate gravitational lensing not in semi-classical ways, but to involve full general relativity. For this reason, the Hamiltonian equations of motion for photons in a Schwarzschild metric around a black hole are established and are then integrated with a Runge-Kutta integrator. The implementation has to cope with several problems, such as numerical integration at the Schwarzschild radius singularity, interpolation of photon coordinates to equal time coordinates, multiprocessing, etc. Additional features such as an anaglyphic, perspective-projected plot viewable with coloured 3D glasses, visualisation of redshift, image storage, photon statistics and benchmarking are also realized. Finally, the results of the simulation are summarized in chapter 4 of this documentation.

2 Theoretical Background: General Relativity

2.1 The Schwarzschild metric

In general relativity, spacetime curvature is given by a symmetric metric tensor $g_{\mu\nu}$. The invariant length element is then given as $ds^2 = g_{\mu\nu}x^\mu x^\nu$. The metric tensor can be expressed as

$$g = g_{\mu\nu} dx^\mu \otimes dx^\nu$$

and is the solution to the Einstein equations

$$G_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$$

where $G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu}$ and $R_{\mu\nu}$ are the so called Einstein and Ricci tensor respectively. They contain information about spacetime geometry. The left-hand side is ruled by the energy-momentum tensor $T_{\mu\nu}$, telling us about energy and momentum densities. The Einstein equations can only be solved for restricted geometries. For the region around a star or a black hole, such (spherically symmetric) solutions are the Schwarzschild metric (non-rotating, uncharged stars), the Kerr metric (rotating, uncharged stars), the Reissner-Nordstrøm metric (non-rotating, charged stars) and the Kerr-Newman metric (rotating, charged stars). As we want to look at non-rotating and uncharged stars, the Schwarzschild solution is the one we are going to use:

$$g_{\mu\nu} = F(r)dt^2 - \frac{1}{F(r)}dr^2 - r^2 d\theta^2 - r^2 \sin^2\theta d\phi^2 \tag{1}$$

where $F(r) = 1 - \frac{2M}{r}$. Note that I use the signature (1,-1,-1,-1) for the metric: this makes time run into positive direction in the integration.

One finds quickly that this solution not only contains a singularity at $r = 0$ but also one at $r = 2M$, which as we will see later makes a numerical integration around this Schwarzschild radius very problematic. At first glance the Reissner-Nordstrøm metric with $F(r) = 1 - \frac{2M}{r} + \frac{Q^2}{r^2}$ could solve this problem by setting Q so that $F(r) = 0$ has no real solutions. This condition is $Q^2 > M^2$ which means that such a heavily charged black hole will be repulsive for small r and thus is not what we want.

2.2 Hamiltonian formulation

In general relativity, the Lagrangian of a system is expressed as the length of the four-velocity vector:

$$L = g_{\mu\nu}\dot{x}^\mu\dot{x}^\nu \equiv \dot{x}_\mu\dot{x}^\mu \tag{2}$$

Because lightlike particles satisfy $\dot{x}_\mu\dot{x}^\mu = 0$, this Lagrangian vanishes (L=0) for light. To find the Hamiltonian formulation, we define the canonical momentum

$$p_\mu = \frac{\partial L}{\partial \dot{x}^\mu} \tag{3}$$

There are two ways to set p_μ and $\dot{x}_\mu$ into relation. The straightforward one is to do the derivation in (3) explicitly:

$$p_\mu = \frac{\partial}{\partial \dot{x}^\mu} g_{\alpha\beta}\dot{x}^\alpha\dot{x}^\beta = 2g_{\mu\nu}\dot{x}^\nu$$

Multiplying both sides with the inverse $g^{\mu\nu}$ results in

$$\dot{x}^\mu = \frac{1}{2}g^{\mu\nu}p_\nu \tag{4}$$

The second, more interesting way is to find $\dot{x}^\mu$ by integrating (3):

$$\int p_\mu \partial\dot{x}^\mu = \int \partial L + H$$

Here H is an integration constant. Thus we have gained the relation between Hamiltonian and Lagrangian formulation, the Legendre transform

$$H = p_\mu\dot{x}^\mu - L \tag{5}$$

For light we have $L = 0$. By plugging in the first result (4) we find

$$H = \frac{1}{2}g^{\mu\nu}p_\mu p_\nu \tag{6}$$

By using the Lagrangian (2) and relation (4) one easily finds that if $L = 0$ then also $H = 0$. Now we have the tools to find equations of motion by using Hamilton's equations.

2.3 Equations of motion

Because we have spherical symmetry, the metric (1) takes a diagonal form:

$$g_{\mu\nu} = \begin{pmatrix} F(r) & 0 & 0 & 0 \\ 0 & -F^{-1}(r) & 0 & 0 \\ 0 & 0 & -r^2 & 0 \\ 0 & 0 & 0 & -r^2 \sin^2\theta \end{pmatrix}$$

And the inverse metric is

$$g^{\mu\nu} = \begin{pmatrix} F^{-1}(r) & 0 & 0 & 0 \\ 0 & -F(r) & 0 & 0 \\ 0 & 0 & -r^{-2} & 0 \\ 0 & 0 & 0 & -r^{-2} \sin^{-2}\theta \end{pmatrix}$$

We can thus derive an expression for the Hamiltonian in spherical coordinates:

$$H = g^{tt}p_t^2 + g^{rr}p_r^2 + g^{\theta\theta}p_\theta^2 + g^{\phi\phi}p_\phi^2 = \frac{p_t^2}{F(r)} - F(r)p_r^2 - \frac{p_\theta^2}{r^2} - \frac{p_\phi^2}{r^2 \sin^2\theta}$$

Note that we drop the $\frac{1}{2}$ for we are not interested in quantitative results but rather in qualitative ones. For our purposes it's better to work in cartesian coordinates. Doing the rather lengthy coordinate transformation, we end up with

$$H = \frac{p_t^2}{F(r)} - p_x^2 - p_y^2 - p_z^2 - \frac{F(r) - 1}{r^2}(xp_x + yp_y + zp_z)^2 \tag{7}$$

where $F(r) \equiv F(\sqrt{x^2 + y^2 + z^2})$

For lightlike particles moving on a null geodesic it is necessary that $H = 0$. As p_t can be seen as the 'speed of time' which can be re-scaled arbitrarily, we can easily choose p_t such that it satisfies (7):

$$p_t = \sqrt{-F(r)H|_{p_t=0}}$$

Hamilton's equations of motion are

$$\dot{q}_\mu = \frac{\partial H}{\partial p_\mu}, \qquad \dot{p}_\mu = -\frac{\partial H}{\partial q_\mu}$$

It is to be mentioned that the dot denotes a derivation by an affine parameter λ and not by the time which is now a separate coordinate. $\left(\dot{q} \equiv \frac{\partial q}{\partial \lambda}\right)$

The derivatives can be obtained as

$$\begin{aligned}
\dot{t} = \frac{\partial H}{\partial p_t} &= 2\frac{p_t}{F(r)} \\
\dot{x}_i = \frac{\partial H}{\partial p_{x_i}} &= -2\left[p_{x_i} + \frac{F(r)-1}{r^2}(xp_x + yp_y + zp_z)x_i\right] \\
\dot{p}_t = -\frac{\partial H}{\partial t} &= 0 \\
\dot{p}_{x_i} = -\frac{\partial H}{\partial x_i} &= \frac{p_t^2}{F^2(r)}\partial_{x_i}F + \frac{(\partial_{x_i}F)r^2 - 2(F(r)-1)x_i}{r^4}(xp_x + yp_y + zp_z)^2 \\
&\quad + \frac{2(F-1)}{r^2}(xp_x + yp_y + zp_z)p_{x_i}
\end{aligned} \tag{8}$$

the derivatives of the metric function $F(r) = 1 - \frac{2M}{r}$ are

$$\partial_{x_i}F(r) = 2M\frac{x_i}{r^3} \tag{9}$$

The goal of the simulation is to integrate these equations of motion for a source of photons. I will talk about that in the next chapter.

3 Implementation

The simulation code consists of three main parts: The integration of the photon paths according to the equations in (8), the interpolation of the photon space coordinates to the ones at equal time coordinate and (perspective) drawing of the photon positions. These main parts are supported by additional components: multiprocessing, redshift visualisation, source patterns, image storage, photon statistics and benchmarking tools.

3.1 Numerical Integration

The photon path integrator integrates the photon paths up to an observer time coordinate provided by a special observer time 'pacemaker' and is designed to run as multiple threads (See section 3.4). The actual numerical integration is done with a fifth order adaptive stepsize Runge-Kutta integrator provided by the qgd base library.

An important feature of this integrator is a method to avoid the singularity problem: As mentioned before, the Schwarzschild metric has a singularity at $r = 2M$. It is very hard to avoid this singularity, especially when we look at large Schwarzschild radii. When approaching this radius, the Runge-Kutta integrator fails by making its step size smaller and smaller for an infinite time, causing the current thread to hang. My solution to this problem is a simple one: As we are not interested in what happens to the photons *within* the Schwarzschild radius (once they passed it they can't escape the black hole anyway) but in what happens far away from it, we can simply drop those photons from our simulation and stop to calculate their paths. For this purpose a flag array is introduced: a boolean for each photon which tells whether or not it still exists. As soon as this flag is set to false for a photon, the simulation stops integrating, interpolating and drawing it. A photon is removed once the Runge-Kutta integrator has adapted its step size more than a thousand times.

3.2 Time Interpolation

As the photon paths are all integrated with respect to an affine parameter λ (comparable to the proper time), the integrator produces photons with various time coordinates when it enters the relativistic region around the black hole. We would like to see the photon paths as an outside observer and are interested in their time evolution: we expect from all photons to have the same time coordinate. Therefore we need to interpolate the spatial coordinates of all photons back/forward to the ones they would have at a given observer time.

Using the integrated eight-dimensional photon phase space vector z and the previous, non-integrated one $\widetilde{z}$, we can interpolate the vectors linearly to find the ones at equal time which is the observer time t_{obs}. The vector we look for is

$$Z(\lambda) = \widetilde{z} + \lambda \Delta z \tag{10}$$

with $\Delta z = z - \widetilde{z}$. As the time coordinate of this vector should be equal to t_{obs}, we can derive an expression for the parameter λ:

$$Z_t(\lambda) \equiv t_{obs} = \widetilde{z}_t + \lambda \Delta z_t \;\Rightarrow \lambda = \frac{t_{obs} - \widetilde{z}_t}{\Delta z_t} \tag{11}$$

Thus the vector of a photon interpolated to the coordinate time t_{obs} is

$$Z_{\text{interpolated}} = \widetilde{z} + \frac{t_{obs} - \widetilde{z}_t}{\Delta z_t} \Delta z \tag{12}$$

These spatial coordinates are now ready to be plotted on the screen.

3.3 Perspective drawing and 3D glasses

Integration and time interpolation leave us with three-dimensional coordinates. To display a three-dimensional image, a perspective projection of these coordinates is performed and drawn as an anaglyphic image which can be viewed with coloured 3D glasses.

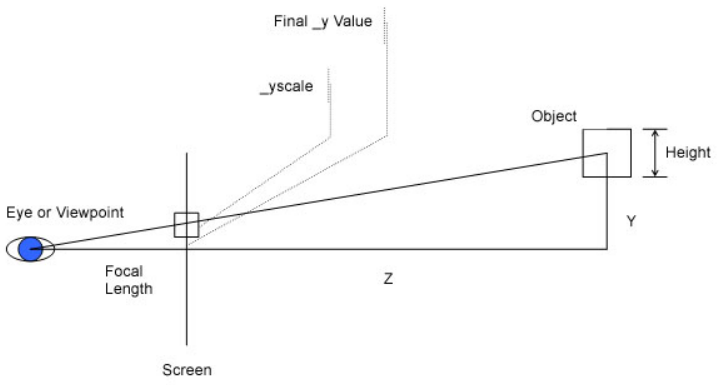

Figure 1: Perspective projection[1]

A sketch of the setup of a perspective projection is shown in figure 1. The relation is

$$Px_i = x_i \cdot \frac{eye}{eye + Z} \tag{13}$$

where *eye* stands for the distance of the observer to the screen, Z is the distance of the screen to the observed object and Px_i is the projected value of x_i.

[1]Image taken from `http://www.bit-101.com/tutorials/perspective.html`

The anaglyphic image can be created by slightly rotating the points by a small angle to the left and to the right and colour them cyan and red respectively. This creates the illusion of a three-dimensional object. For the rotation I use quaternions from the qgd library.

3.4 Multiprocessing

As the simulation gets more interesting with an increasing number of photons, it is very important that it runs as fast as possible, preferably on a super computer. Also most of today's personal computers are equipped with a dual core processor. Thus it is desirable that the integration can be splitted into multiple threads. This is very trivial as the photons do not interact with each other and every photon path can be integrated with no knowledge about the other paths.

For curiosity of the author, two threading models have been implemented: the standard threading model using the *Thread* class and the *Executor Service* from the concurrency package[1]. Curiously, the first one has proven to run faster on a laptop.

Additionally, there is an option to disable threading completely. This is mainly thought to test what speed enhancement we get with multiprocessing.

3.5 Redshift visualisation

The frequency of light gets shifted to the blue region of the color spectrum when approaching a gravitational source while going away from the source makes it redshifted. The relation between original frequency and redshifted frequency is given by the relation

$$\frac{f_0}{\tilde{f}} = 1 + z = \sqrt{\frac{g_{00}(\vec{x})}{g_{00}(\vec{x_0})}} \tag{14}$$

where g is the metric, f_0 is the frequency of the light emitted by the light source and $\tilde{f}$ its frequency redshifted by the black hole. $\vec{x}$ represents the current position of the photon and $\vec{x_0}$ the position of the light source.

With respect to the Schwarzschild metric, the redshifted frequency is

$$\tilde{f} = f_0 \sqrt{\frac{1 - \frac{2M}{r_{source}}}{1 - \frac{2M}{r}}} \tag{15}$$

Here r_{source} and r are the distances to the black hole from light source and current photon position respectively.

[1] See `http://java.sun.com/docs/books/tutorial/essential/concurrency/`

The visible color spectrum ranges from frequencies of 400 Hz (red) to 789 Hz (violet). The frequency of the light source should therefore be set closely to red, as the photons first get blueshifted until they have passed the black hole and then get redshifted again. If the frequency of a photon exceeds the visible spectrum, its color is set to black.

3.6 Source patterns

The generation of different patterns for the light source is possible: Adjustable parameters are number of photons, light frequency, number of pulses, gap between pulses and speed/position of the light source. The generation of photons is possible either in full three dimensions or just on a plane. In 3D the photons can either be distributed equally (better effect with coloured glasses) or randomly onto the angles φ and θ.

3.7 Tools

3.7.1 Image storage and animation

As a feature to turn on and off, all generated 2D plots can be stored into GIF images. It takes a long time to write the images (see benchmarking tool) and therefore this option is only recommended for large photon numbers. Once the simulation is over, the animation can be done by a provided shell script which uses *gifsicle*[2] to animate the GIFs and *mplayer*[3] to play it.

3.7.2 Photon statistics

The statistics plot is mainly intended for diagnostics and draws some interesting photon properties:

- Hamiltonian (should be zero, see section 2.2)
- Metric function $F(r)$ (see section 2.1)
- Time coordinates (should all be equal in the interpolated version)
- Momentum in x direction
- Derivation of the momentum in x direction with respect to the affine parameter
- Derivation of the x coordinate with respect to the affine parameter

[2] `http://www.lcdf.org/gifsicle/`
[3] `http://www.mplayerhq.hu`

3.7.3 Benchmarking

As the goal is to make this simulation very fast for a large amount of photons, it is very important to know how fast it is (measured in integrations per second) and what part of the program uses what percentage of the time of a run. This job is done by a separate class and can be turned on and off for reasons of performance. The aim is for the integrator to use as much runtime as possible compared with making threads, interpolating and plotting.

4 Results

The main goal of this project was to succesfully integrate the Hamiltonian equations of motion for photons in a gravitational field and carry out the simulation with appropriate initial conditions. The simulation was not designed to find any specific quantity, it is still held very general.

4.1 Examples

Interesting results are shown in the following plots. Note that you can find them on the webpage given in appendix A, where they are colored, in a better quality and also exist in fully animated versions.

Figure 3 shows a lensing situation at a 'weak' black hole: light close to the black hole gets slightly deflected while light even closer to it is accelerated in such a way that it is sent back into the direction of the source. The sender thus sees its own image (although very distorted) in the black hole and so, technically, a black hole could act as a *mirror*. In this example, source, black hole and observer are located on the same line. If the light source is shifted orthogonally to this line, the observer will see the same wavefront from the source two times, seeming to come from different locations.

The situation shown in figure 4 is even more interesting. The black hole mass is now rather large (yellow circle: Schwarzschild radius, green circle: photon radius), in fact, it is critically large: the photons sent back into the direction of the source are lensed again, sent back to the observer, lensed again, etc. If there was an infinite amount of photons (of course there is not), this would go on and on forever. The observer sees the single wavefront emitted by the light source several times, originating from several locations.

Figure 5 depicts the same situation as in figure 3, except that the source now emits multiple wavefronts. Scientifically, this plot is not any more interesting than the example in figure 3, but as stars usually emit more than one wavefront of photons, we get closer to reality.

In figure 6, the source is not only pulsed but also moving. From a scientific point of view, this is not of much use now, but it is very nice to look at and even closer to reality than the previous example. The ability to create pulsed and moving sources could be used for more complex calculations in a later, more applied stage of the program.

Finally, figure 7 shows an example of the 3D plot. This is just to illustrate the ability of the program to generate perspective projections and anaglyphic images; to really watch it in 3D with coloured glasses, the reader is encouraged to run the program. This is a very nice gadget, but of real scientific interest are only the examples in figure 3 and 4.

4.2 Magnification

Theory says that in the first lensed wavefront, there should be a magnification greater than unity. I tried to find a way to show that with this simulation, but failed. Basically, the magnification should be in relation with the density of photons: technically, this is the distance of nearest neighbours. I tried to implement that in the following way:

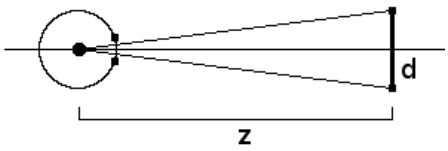

Figure 2: My approach to find an expression for the magnification

First we look at a situation without a black hole. The distance of nearest neighbours for the photons emitted on a circle with radius r is $\frac{2\pi r}{N}$, where N is the number of photons. At $r = 1$ the distance is thus $\frac{2\pi}{N}$. We can now find the nearest neighbour distance d at a distance of z (observer) to the source by using intercept theorems:

$$\frac{d}{\frac{2\pi}{N}} = \frac{z}{1} \tag{16}$$

and thus

$$d = \frac{2\pi z}{N} \tag{17}$$

in the absence of a gravitational field.

In this approach, in a lensing situation the nearest neighbour distance can be measured at the observer and compared to what it should be (d); it should eventually be smaller than the unlensed distance (higher photon density), but I could not verify that in my simulation which always produced larger nearest neigbour distances in lensed situations than in the ones without a lens.

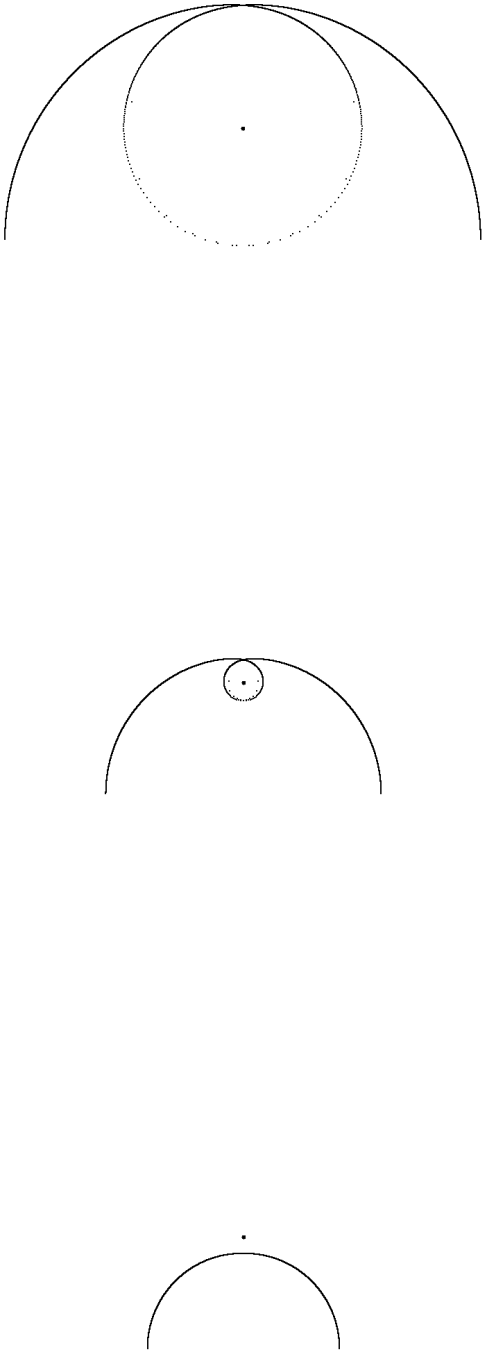

Figure 3: Lensing at a small black hole (M = 10000, 50000 photons)

Figure 4: Lensing at a critically large black hole (M = 214000, 50000 photons)

Figure 5: Pulsed source lensing at a small black hole (M = 20000, 5000 photons, 5 pulses)

Figure 6: Pulsed, moving source lensing at a large black hole (M = 560000, 5000 photons, 20 pulses, speed = 150000)

Figure 7: Anaglyphic, perspective projected 3D image (M = 200000, 2000 photons)

A Source code

The jar file, source code of this simulation and some nice animations can be found on

`http://www.physik.uzh.ch/~chuwyler/gravlens/`

A description of the purpose of the different java classes is given in table 1.

Table 1: Comments to the purpose of the classes of this simulation

main	GravLens2	main class
	Integrator	interface to the Runge-Kutta method (qgd library), designed for multithreading
	Interpolator	interpolates photon positions to a common coordinate time
	Observer	observer time 'pacemaker'
	Redshift	calculates redshifted photon frequencies
	SchwarzschildSolution	contains all the necessary physics for this simulation
	Source	creates different photon source patterns
tools	Benchmarker	measures runtime usage for different parts of the simulation
	ImageOutput	hooks to Panel2D and stores every step as a GIF image
	PatternGenerator	contains different source patterns (moving, pulsing, etc.)
	Pattern	pattern structure
	Plot	plots current properties of the photons
gui	BenchmarkPanel	provides a gui interface for Benchmarker
	ImageDialog	window to configure image storage
	LogSliderEntanglement	like SliderEntanglement but takes the slider on a logarithmic scale
	Plane2D	plot in two dimensions
	Plane3D	plot in three dimensions using perspective and an anaglyphic image
	SliderEntanglement	interconnects sliders and input fields
	SliderPanel	JPanel containing sliders and input fields to adjust properties of the simulation