On the quantum Schwarzschild gravitational theories consisting of field lines, for C++ programmers

S. Halayka*

Friday 16$^{\text{th}}$ January, 2026 14:48

Abstract

This paper contains a short introduction to isotropic Schwarzschild gravitation. It is found that with non-normal (e.g. cosine weighted) messenger particle emission comes increased gravitational strength – matching that from the Schwarzschild solution. It is also found that there can be repulsive gravitation.

1 Introduction

First see [1] for a short tutorial for C++ programmers on isotropic Newtonian gravitation. In [1] we build an isotropic gravitational field through the use of pseudorandomly generated field lines. In [1] we use a sphere as the receiver.

In this paper, we find a match between the numerical gravitation and the gravitational time dilation from Schwarzschild's general relativity [2–4]. Here, we use an axis-aligned bounding box (AABB) as the receiver.

See Fig. 1 for an example of Newtonian gravitation, where the field lines are normal to the surface of the emitter. See Fig. 2 for an example of Schwarzschild gravitation, where the field lines are not necessarily normal to the surface of the emitter.

In this paper we use the natural Planck units, where $c = G = \hbar = k = 1$.

2 Method

Where r_e is the emitter's Schwarzschild radius, r_r is the receiver AABB radius (e.g. half of the AABB side length), and 1e11 and 0.01 are arbitrary constants:

$$r_e = \sqrt{\frac{1\text{e}11 \log(2)}{\pi}}, \tag{1}$$

$$r_r = r_e \times 0.01. \tag{2}$$

The event horizon area is:

$$A_e = 4\pi r_e^2. \tag{3}$$

*sjhalayka@gmail.com

The entropy (e.g. field line count) is:

$$n_e = \frac{A_e}{4\log(2)} = 1\text{e}11. \tag{4}$$

Where R is the distance from the emitter's centre, the derivative is:

$$\alpha = \frac{\beta(R+\epsilon) - \beta(R)}{\epsilon}. \tag{5}$$

Here β is the get intersecting line density function. The gradient strength is:

$$g = \frac{-\alpha}{2r_r^3}. \tag{6}$$

From this we can get the Newtonian acceleration a_N, where $r_e \ll R$:

$$a_N = \frac{gR\log 2}{8M_e} = \sqrt{\frac{n_e \log 2}{4\pi R^4}} = \frac{M_e}{R^2}. \tag{7}$$

We can also get a general relativistic acceleration a_S, where $r_e < R$. In this case, the same gradient-based expression is applied, but is not restricted by the weak-field condition:

$$a_S = \frac{gR\log 2}{8M_e}. \tag{8}$$

At close proximity, where $r_e \approx R$, the metric produced is related to the Schwarzschild metric – curved space, curved time:

$$t = \sqrt{1 - \frac{r_e}{R}}, \tag{9}$$

$$\frac{\partial t}{\partial R} = \frac{r_e}{2tR^2}. \tag{10}$$

$$a_S \approx \frac{\partial t}{\partial R}\frac{2}{\pi} = \frac{r_e}{\pi t R^2}. \tag{11}$$

At far proximity, where $r_e \ll R$, $t \approx 1$, and $\frac{\partial t}{\partial R} \approx 0$, the metric produced is Newtonian – curved space, practically flat time.

In general relativity, acceleration is also dependent on the kinematic time dilation of the gravitated body – internal process forms a resistance to gravitation. Where the speed is $v \approx c$, like that for neutrinos, the gravitational attraction is twice of that predicted by Newtonian gravity because of a lack of said resistance.

3 C++ code

The following code uses the Newtonian or Schwarzschild gravitation:

```
double get_intersecting_line_density(
        const long long unsigned int n,
        const double emitter_radius,
        const double receiver_distance,
        const double receiver_distance_plus,
        const double receiver_radius)
```

```cpp
{
	double count = 0;
	double count_plus = 0;

	generator.seed(static_cast<unsigned>(0));

	for (long long unsigned int i = 0; i < n; i++)
	{
		if (i % 10000000 == 0)
			cout << double(i) / double(n) << endl;

		glm::dvec3 location = random_unit_vector();

		location.x *= emitter_radius;
		location.y *= emitter_radius;
		location.z *= emitter_radius;

		glm::dvec3 surface_normal =
			glm::normalize(location);

		// A) Newtonian gravitation
		//glm::dvec3 normal =
		//	surface_normal;

		// B) Schwarzschild gravitation
		//glm::dvec3 normal =
		//	random_cosine_weighted_hemisphere(surface_normal);

		// C) Schwarzschild gravitation using a useful trick
		//glm::dvec3 normal =
		//	glm::normalize(
		//		surface_normal + random_unit_vector());

		// D) Emulate Quantum Graphity to get
		// Schwarszschild gravitation
		glm::dvec3 normal =
			glm::normalize(
				location -
				random_unit_vector() * emitter_radius);

		if (dot(normal, surface_normal) < 0)
			normal = -normal;

		count += intersect(
			location, normal,
			receiver_distance, receiver_radius);

		count_plus += intersect(
			location, normal,
			receiver_distance_plus, receiver_radius);
	}

	return count_plus - count;
}
```

The cosine weighting is given by:

```
vector_3 random_cosine_weighted_hemisphere(const vector_3& normal)
{
	glm::dvec2 r = glm::vec2(dis(generator), dis(generator));

	glm::dvec3 uu =
		glm::normalize(
			glm::cross(normal, glm::dvec3(0.0, 1.0, 1.0)));

	glm::dvec3 vv = glm::cross(uu, normal);

	double ra = sqrt(r.y);
	double rx = ra * cos(2.0 * pi * r.x);
	double ry = ra * sin(2.0 * pi * r.x);
	double rz = sqrt(1.0 - r.y);
	glm::dvec3 rr = glm::dvec3(rx * uu + ry * vv + rz * normal);

	return glm::normalize(rr);
}
```

4 Conclusion

It's worth noting that gravitation, if cosine weighted, is a diffuse process similar to electromagnetic lighting in path tracing from computer graphics. For instance, see Fig. 3 for an image produced using path tracing.

Experimentation with the receiver size is necessary to understand the intricacies of the code. For instance, where $r_r = r_e \times 0.01$, it is found that gravitation can be repulsive.

The full code for this paper can be found at:

https://github.com/sjhalayka/schwarzschild_falloff_field_lines

Note that while Quantum Graphity [5] uses gravitons, it is practically the same as using field lines, and so there is repulsion in some cases as well.

It should be noted that repulsive gravitation plagues all non-normal (e.g. cosine weighted) theories. It should also be noted that getting rid of the field lines altogether, by replacing them with randomly emitted gravitons, does not solve the problem of repulsive gravitation.

5 Conflict of interest

There are no conflicts of interest to report.

6 Data availability

The C++ code is freely available, providing a way to reproduce the data.

References

[1] Halayka. Newtonian gravitation from scratch, for C++ programmers. (2024)

[2] ‘t Hooft. Dimensional reduction in quantum gravity. (1993)

[3] Susskind. The World as a Hologram. (1994)

[4] Misner et al. Gravitation. (1970)

[5] Konopka et al. Quantum Graphity: a model of emergent locality (2008)

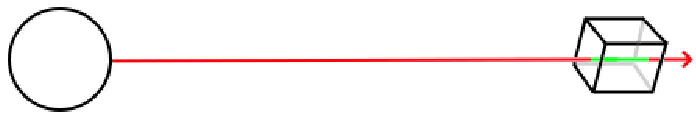

Figure 1: This figure shows an axis-aligned bounding box and an isotropic emitter, looking from slightly above. An example field line (red) and intersecting line segment (green) are given. The bounding box is filled with these green intersecting line segments. It is the gradient of the density of these line segments that forms the gravitational acceleration. Note that the field line is normal to the surface.

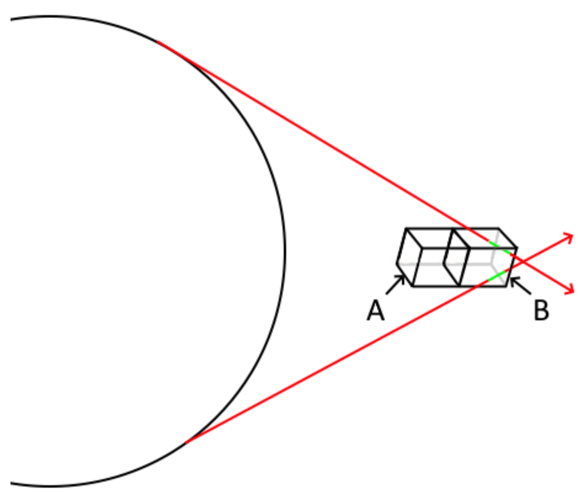

Figure 2: Axis-aligned bounding box A and box B are separated by the value epsilon $\epsilon = 2r_r$. Box B contains more field lines than box A, resulting in repulsive gravitation. Note that the field lines are not normal to the surface.

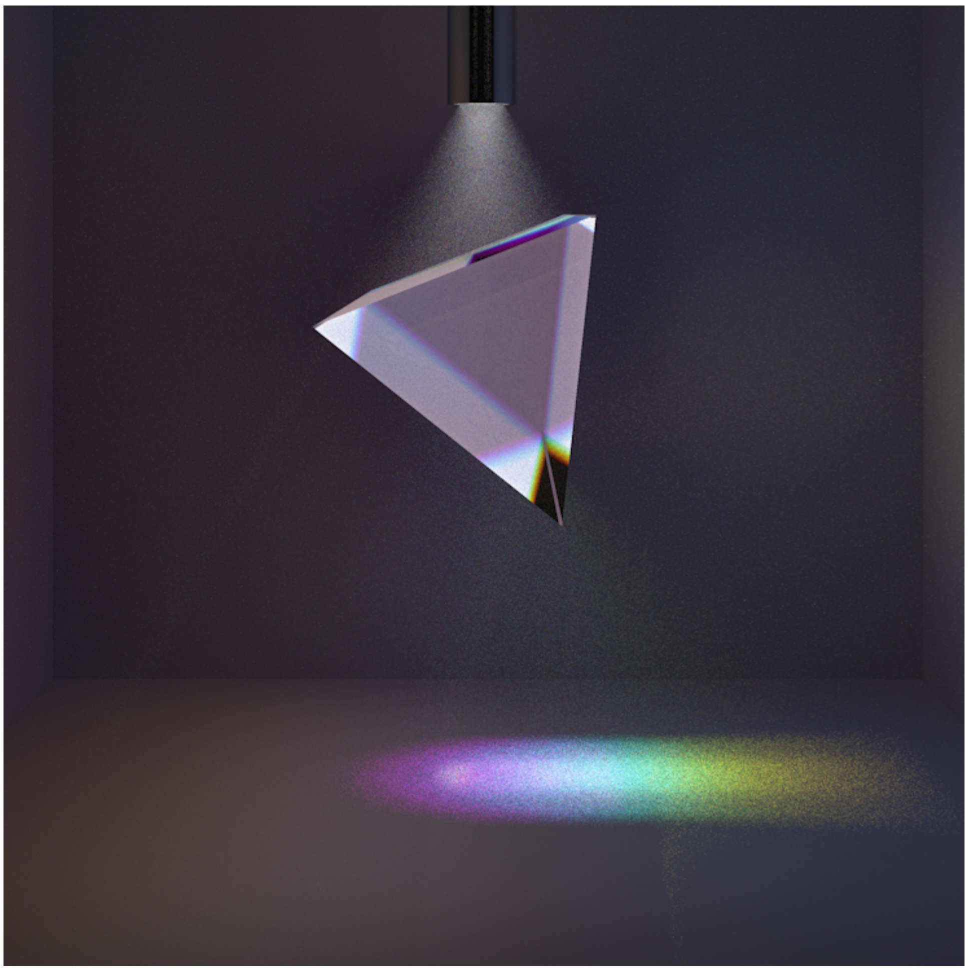

Figure 3: Backward path tracing in the Vulkan graphics API. The scene has a white barrel light that is dispersed by a prism into a spectrum of colours. There is some fog.