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

S. Halayka*

Monday 26th January, 2026 16:07

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

In [1], we presented a short tutorial for C++ programmers on isotropic Newtonian gravitation. In [1] we built an isotropic gravitational field through the use of pseudorandomly generated field lines. In [1] we used 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, 3]. 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.

3 C++ code

The following code uses the Newtonian or Schwarzschild gravitation:

```
void worker_thread(
        long long unsigned int start_idx,
        long long unsigned int end_idx,
        unsigned int thread_seed,
        const real_type emitter_radius,
        const real_type receiver_distance,
        const real_type receiver_distance_plus,
        const real_type receiver_radius,
        real_type& result_count,
        real_type& result_count_plus)
```

```
{
        // Thread-local random number generator
        std::mt19937 local_gen(thread_seed);
        std::uniform_real_distribution<real_type> local_dis(0.0, 1.0);

        real_type local_count = 0;
        real_type local_count_plus = 0;

        // Update progress every N iterations to reduce atomic overhead
        const long long unsigned int progress_update_interval = 10000;
        long long unsigned int local_progress = 0;

        for (long long unsigned int i = start_idx; i < end_idx; i++)
        {
                vector_3 location = random_unit_vector(local_gen, local_dis);
                location.x *= emitter_radius;
                location.y *= emitter_radius;
                location.z *= emitter_radius;

                vector_3 surface_normal = location;
                surface_normal.normalize();


                // A) Newtonian gravitation
                //vector_3 normal =
                //      surface_normal;

                // B) Schwarzschild gravitation, classical
                //vector_3 normal =
                //      random_cosine_weighted_hemisphere(
                //              surface_normal, local_gen, local_dis);

                // C) Schwarzschild gravitation, quantum
                vector_3 r = random_unit_vector(local_gen, local_dis);
                r.x *= emitter_radius;
                r.y *= emitter_radius;
                r.z *= emitter_radius;
                vector_3 normal = (location - r).normalize();


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

                local_count_plus += intersect(
                        location, normal,
                        receiver_distance_plus, receiver_radius);

                // Update global progress periodically
                local_progress++;
                if (local_progress >= progress_update_interval)
                {
                        global_progress.fetch_add(
                                local_progress, std::memory_order_relaxed);

                        local_progress = 0;
```

```
                }
        }

        // Add any remaining progress
        if (local_progress > 0)
        {
                global_progress.fetch_add(
                        local_progress, std::memory_order_relaxed);
        }

        result_count = local_count;
        result_count_plus = local_count_plus;
}
```

Here we see that there are at least 2 different ways of looking at the same Schwarzschild solution. That is, solution B) uses a cosine weighted approximation method. Solution C) uses a quantum, exact method.

Note that the code is highly parallelizable, and so it takes advantage of C++ standard multithreading.

See Fig. 3 for a plot showing the strength of the Schwarzschildian and Newtonian gravitation.

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`

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. See Fig. 4 for a plot of the onset of repulsive gravitation mean distance d based on field line count n, where $d \approx n/18580 + 7347$.

4 Conflict of interest

There are no conflicts of interest to report.

5 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] Misner et al. Gravitation. (1970)

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.

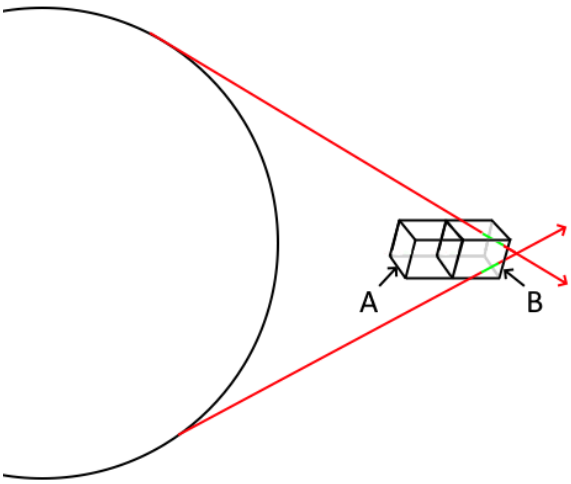

Figure 2: Axis-aligned bounding box A and box B are separated by the value epsilon $\epsilon = 2r_r$. Box B contains more field lines than box A, resulting in repulsive gravitation. Note that the field lines are not normal to the surface.

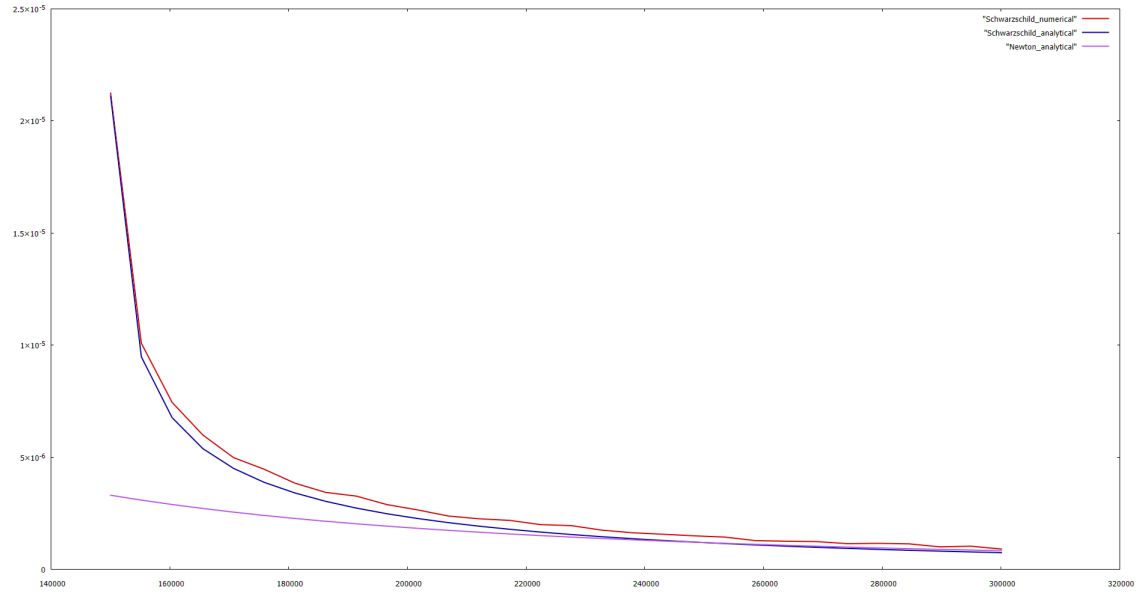

Figure 3: A plot showing the strength of the Schwarzschildian and Newtonian gravitation.

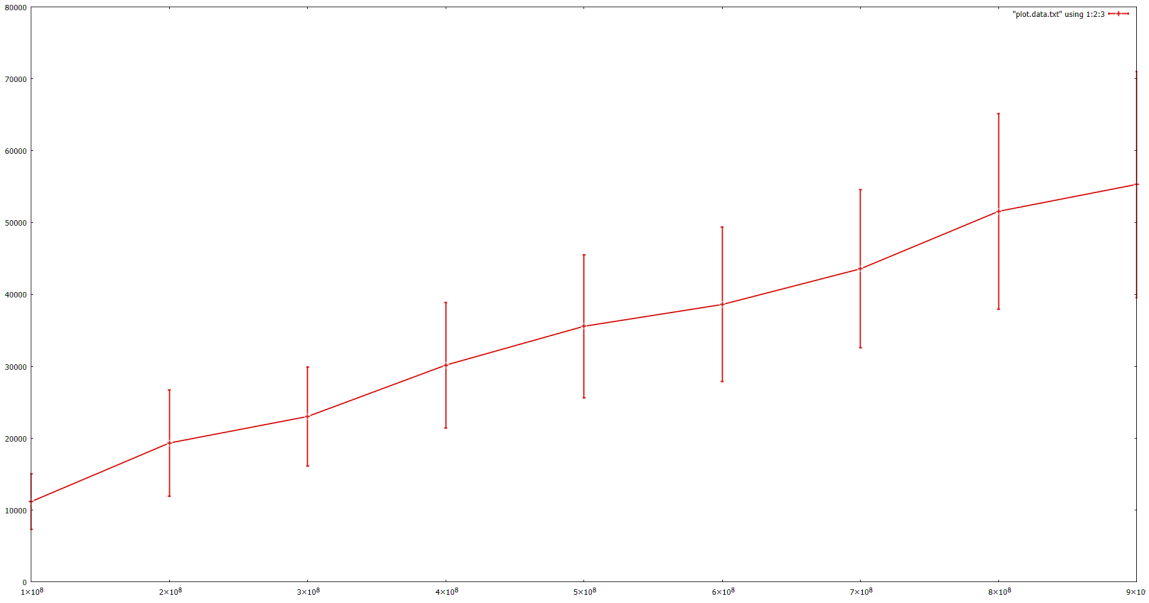

Figure 4: A plot of the onset of repulsive gravitation mean distance d based on field line count n, where $d \approx n/18580 + 7347$. That is, the plot forms an approximately straight line. Although not explicitly shown here, the standard deviations also form an approximately straight line.