On the quantum decomposition of the planet Mercury's orbit path using post-Newtonian gravitation

S. Halayka*

Thursday 6th June, 2024 20:56

Abstract

By quantizing the kinematic and gravitational time dilation using various step sizes, one obtains a set of weighted orbit paths. The precession associated with each weighted orbit path combines to provide the same answer as the classical analytical solution.

1 Introduction

In a previous paper [1], we introduced a method of numerical simulation for the four Solar System tests of general relativity, in particular the precession of Mercury's orbit. There was a catch: one had to quantize the kinematic and gravitational time dilation by casting the relevant floating point variables from double-precision to single-precision. A first, this was taken to be a bug, but after careful consideration, it turns out to be a feature of reality. This paper will demonstrate how to decompose the orbit path of Mercury, to numerically obtain the relativistic orbit precession.

The C++ code for this paper is available. It uses the Boost multiprecision floating point library, so that we may specify an arbitrary number of mantissa bits.

2 Quantization of time dilation

In this paper, we will be quantizing the kinematic and gravitational time dilation by casting them to a lesser-precision floating point number.

The non-exponent bit count n includes the number of mantissa bits m, plus one sign bit. We generally used $n = 100$, except for the kinematic and gravitational time dilation, which uses a lesser, various precision (e.g. $n = 24$).

For subnormal numbers such as those used here, the smallest step size that can be represented is $\epsilon = 2 \times 2^{-b}$, where b is the largest exponent value (e.g. $2^7 - 1 = 127$ in single-precision floating point numbers, $2^{10} - 1 = 1023$ for doubles). For instance, we use 1023 where $n = 100$, and 127 otherwise (e.g. where $n = 24$).

As for m, it governs how many places there are after the decimal point.

See Figs. 1 and 2.

*sjhalayka@gmail.com

3 Steps in spacetime

In this paper, we use Euler integration, with a time step of $\delta_t = 0.01$.

Where ℓ_s denotes the Sun's location at the origin, ℓ_o denotes the orbiter Mercury's location, and $\vec{d}$ denotes the direction vector that points from the orbiter toward the Sun:

$$\vec{d} = \ell_s - \ell_o, \tag{1}$$

$$\hat{d} = \frac{\vec{d}}{||\vec{d}||}, \tag{2}$$

the Newtonian acceleration vector is:

$$\vec{g}_n = \frac{\hat{d}GM}{||\vec{d}||^2}. \tag{3}$$

One parameter is closely related to the kinematic time dilation:

$$\alpha = 2 - \sqrt{1 - \frac{||\vec{v}_o||^2}{c^2}}. \tag{4}$$

Another parameter is the gravitational time dilation from the Schwarzschild solution:

$$\beta = \sqrt{1 - \frac{R_s}{||\vec{d}||}}. \tag{5}$$

Finally, the semi-implicit Euler integration for velocity and then location is:

$$\vec{v}_o(t + \delta_t) = \vec{v}_o(t) + \delta_t \alpha \vec{g}_n, \tag{6}$$

$$\ell_o(t + \delta_t) = \ell_o(t) + \delta_t \beta \vec{v}_o(t + \delta_t). \tag{7}$$

Note that Newtonian gravity is the result where $\alpha = \beta = 1$.

See Figs. 3 and 4.

4 Classical analytical calculation of Mercury's orbit precession

The classical orbit precession is:

$$\delta_p = \frac{6\pi GM}{c^2(1 - e^2)a} \left(\frac{1}{\pi \times 180 \times 3600} \right) \left(\frac{365}{88} \times 100 \right) = 42.937 \tag{8}$$

arcseconds per Earth century, where $e = 0.2056$ is the eccentricity, and $a = 5.7909 \times 10^{10}$ is the semi-major axis.

See Fig. 5.

5 Quantum path decomposition

Here we report the precession angle δ_p in terms of arcseconds per Earth century, after one full orbit. There is a highly visible pattern, which we shall point out.

We use an initial location that is 6.9817079×10^{10} metres from the Sun (e.g. the aphelion). We use various initial speeds.

Where initial speed is 20000 metres per second, and the analytical solution is 162.09:

Bits n	Angle δ_p	Weight
21		1
22		1
23	53.16	1

If we add bit counts 21, 22, and 23 together, we get arcseconds per Earth century.

Where initial speed is 25000 metres per second, and the analytical solution is 103.7:

Bits n	Angle δ_p	Weight
22		1
23		1
24		1

If we add bit counts 22, 23, and 24 together, we get .

Where initial speed is 30000 metres per second, and the analytical solution is 72.04:

Bits n	Angle δ_p	Weight
23	41.45	1
24	26.78	1

If we add bit counts 23 and 24 together, we get 68.23.

Where initial speed is 38858.47 metres per second (e.g. the actual speed at aphelion), and the analytical solution is 42.9:

Bits n	Angle δ_p	Weight
24	46.35	42.9 / 46.35

See Eq. 8. Note that a single-precision floating point number has 24 mantissa and sign bits.

Where initial speed is 42500 metres per second, and the analytical solution is 35.8:

Bits n	Angle δ_p	Weight
26	51.7	35.8 / 51.7

6 Conclusion

Here we have quantized the kinematic and gravitational time dilation. The result is a set of weighted orbit paths that add up to provide the same answer as the classical analytical solution.

References

[1] Halayka. On simulating the four Solar System tests of general relativity using two-parameter post-Newtonian gravitation with Euler integration. (2024)

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

```
cout << setprecision(30) << endl;

const cpp_bin_float_100_ one = 1.0;
const cpp_bin_float_100_ pi_divided_by_four = atan(one);

cout << cpp_bin_float_2_(pi_divided_by_four) << endl;
cout << cpp_bin_float_3_(pi_divided_by_four) << endl;
cout << cpp_bin_float_4_(pi_divided_by_four) << endl;
cout << cpp_bin_float_5_(pi_divided_by_four) << endl;
cout << cpp_bin_float_6_(pi_divided_by_four) << endl;
cout << cpp_bin_float_7_(pi_divided_by_four) << endl;
cout << cpp_bin_float_8_(pi_divided_by_four) << endl;
cout << cpp_bin_float_9_(pi_divided_by_four) << endl;
cout << cpp_bin_float_10_(pi_divided_by_four) << endl;
cout << cpp_bin_float_11_(pi_divided_by_four) << endl;
cout << cpp_bin_float_12_(pi_divided_by_four) << endl;
cout << cpp_bin_float_13_(pi_divided_by_four) << endl;
cout << cpp_bin_float_14_(pi_divided_by_four) << endl;
cout << cpp_bin_float_15_(pi_divided_by_four) << endl;
cout << cpp_bin_float_16_(pi_divided_by_four) << endl;
cout << cpp_bin_float_17_(pi_divided_by_four) << endl;
cout << cpp_bin_float_18_(pi_divided_by_four) << endl;
cout << cpp_bin_float_19_(pi_divided_by_four) << endl;
cout << cpp_bin_float_20_(pi_divided_by_four) << endl;
cout << cpp_bin_float_21_(pi_divided_by_four) << endl;
cout << cpp_bin_float_22_(pi_divided_by_four) << endl;
cout << cpp_bin_float_23_(pi_divided_by_four) << endl;
cout << cpp_bin_float_24_(pi_divided_by_four) << endl;
```

Figure 1: Code, showing the quantization of a constant $\pi/4$. Here n runs from 2 to 24.

```
0.75
0.75
0.8125
0.78125
0.78125
0.7890625
0.78515625
0.78515625
0.78515625
0.78515625
0.785400390625
0.785400390625
0.785400390625
0.785400390625
0.785400390625
0.785400390625
0.785396575927734375
0.7853984832763671875
0.7853984832763671875
0.785398006439208984375
0.7853982448577880859375
0.78539812564849853515625
0.785398185253143310546875
```

Figure 2: Output from the code.

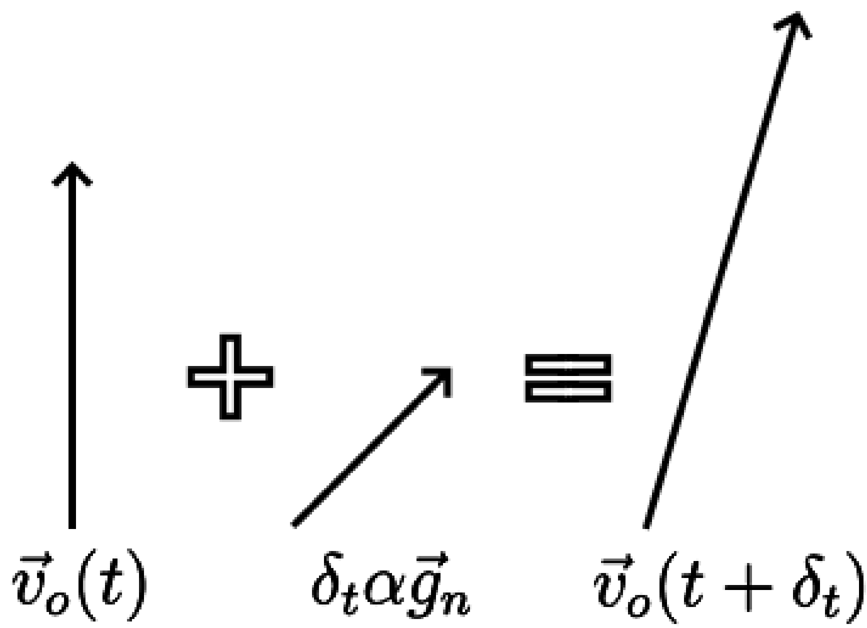

Figure 3: A diagram of the Euler integration of velocity.

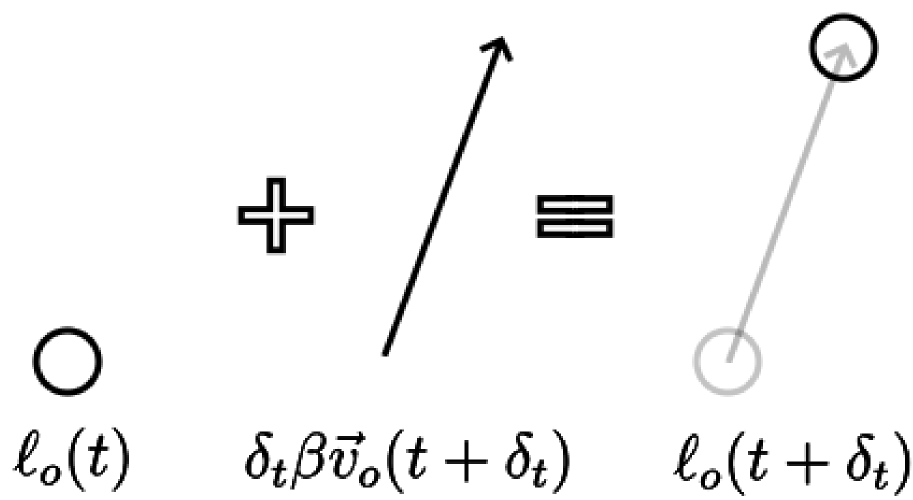

Figure 4: A diagram of the Euler integration of location.

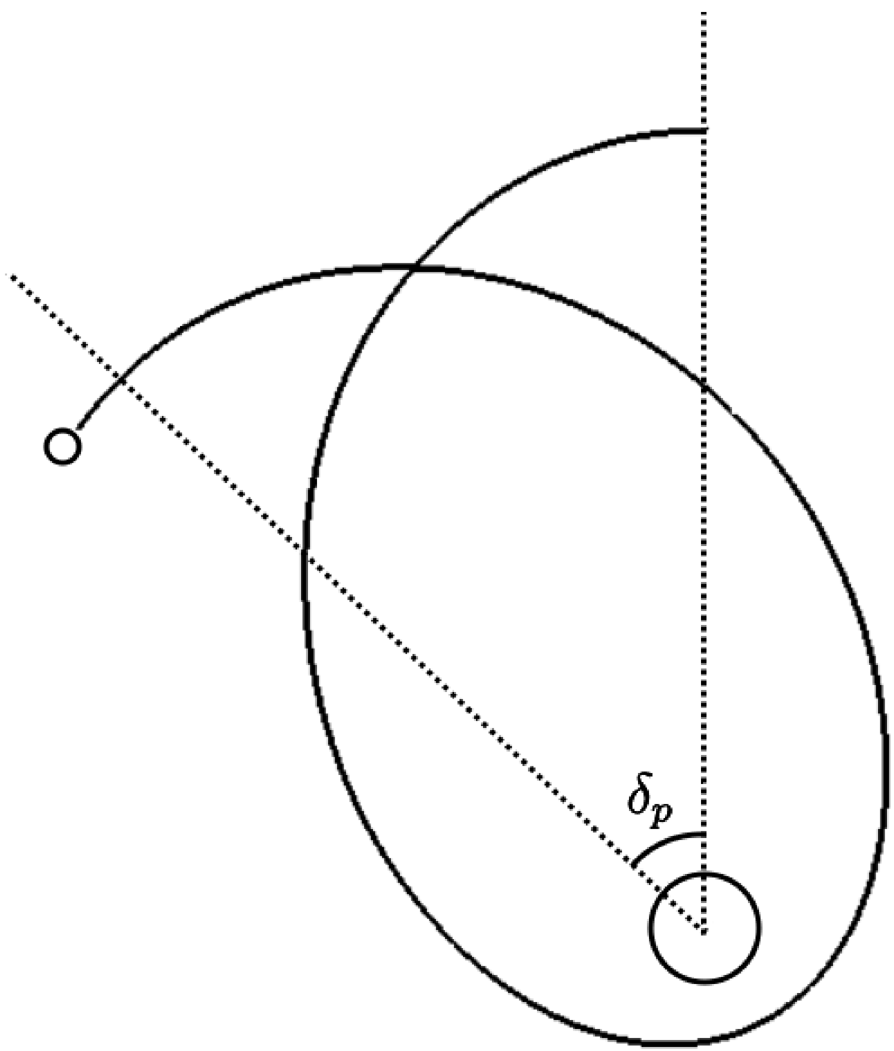

Figure 5: A diagram showing precession, where the orbit does not quite form a closed ellipse.