4.4 Simulations for the induced acceleration

Earlier, while describing the theory of Maxwellian gravity, the force according to the $\frac{1}{c^2}\frac{2\gamma+1}{\gamma+1}\vec{v}\times\vec{E}\times\vec{v}$ term discovered by chance was introduced, and simulations showed that this force explained the perihelion advance of Mercury's orbit. Now that a new $-\frac{\vec{u}}{c^2}(\vec{u}\cdot\vec{E})$ dependent acceleration has been discovered, it is necessary to check whether this acceleration has any effect on the orbit.

First, download the program from GitHub (https://github.com/kycgit/gsimm) as did in the previous Maxwellian gravity simulation chapter, and then run common-lisp from that folder. Then run the following command.

```
CL-USER> (load "gsimm2.lisp")
To load "trivial-arguments":
Load 1 ASDF system:
trivial-arguments
; Loading "trivial-arguments"

T
CL-USER> (in-package va)
#<PACKAGE "VECTOR-ARITHMETIC">
VA>
```

In the previous case, it was sufficient to simulate the effect of only one factor, but now, since the effect of various types of factors must be simulated individually, it is too much to repeat the whole explanation if all of the factors are programmed. So, I modified the program to define each factor as a simple function and call it whenever necessary to simulate it. The newly added parts are as follows.

```
(defun 3/2*vXaXv/c2 (a v) [v X* a X* v .* 1.5d0 %c ./ ^ 2.0d0])
(defun a*{v.v}/c^2 (a v) [v .* v .* a %c ./ ^ 2d0])
(defun -v{v.a}/c^2 (a v) [-1 .* v .* a .* v %c ./ ^ 2.0d0])
(defun total (a v) [v X* a X* v .* 0.5d0 v .* a .- .* v %c ./ ^ 2.0d0])
(defun 1+GM/rc2 (m r) [1 + /(%G * m sqrt v.v r %c %c)])
(defun 2X1+GM/rc2 (m r) [1 + /(%G * m sqrt v.v r %c %c) ^ 2])
(defun 3X1+GM/rc2 (m r) [1 + /(%G * m sqrt v.v r %c %c) ^ 3])
(defun 1/gamma (v) [1 v.v v %c sqrt - ./ ^ 2])

(defun next-T (idt af mf gf)
  (labels ((cp (x) (make-pobj :m (pobj-m x) :r (pobj-r x) :v (pobj-v x) :a 0v)))
    (let* ((ndt (make-tpsy :t 0 :psy (mapcar #'cp (tpsy-psy idt))))
           (i (car (tpsy-psy ndt))) (j (cadr (tpsy-psy ndt)))
           (dr [pobj-r i .- pobj-r j])
           (mm (if mf [pobj-m i * funcall(mf pobj-m i dr)] (pobj-m i)))
           (ag [dr v.v dr ./ ^ 1.5d0 .* %G .* mm])
           (br [v.v pobj-r cadr tpsy-psy ndt]) (nts [*ts*  br / df-r * ^ 0.75d0]))
      (setf (tpsy-t ndt) (+ (tpsy-t idt) nts))
```

```
      (setf (pobj-a j)
            [ag .+ if(af funcall (af ag pobj-v j) 0v) .* if(gf funcall (gf pobj-v j) 1)])
      (setf (pobj-r j) [pobj-r j pobj-v j .+ .* nts 0.5d0 .* pobj-a j nts .+ .* ^ 2])
      (setf (pobj-v j) [pobj-v j pobj-a j .+ .* nts ])
      ndt)))

(defun sim-T (af mf gf dtime sptime tstime planet ec outfile)
  (initparameters dtime planet)
  (init-seed planet ec)
  (let (sdata (nxtt sptime) (xin *seed*))
    (setq sdata
          (loop while (< nxtt tstime)
                do (loop for x = xin then (next-T x af mf gf)
                         while (< (tpsy-t x) nxtt)
                         finally (setq xin x nxtt [nxtt + sptime]))
                collect xin))
    (push *seed* sdata)
    (with-open-file
        (stream outfile
                :direction :output ; opens for output
                :if-exists :supersede   ;:overwrite ; replace existing file
                :if-does-not-exist :create)
      (loop for i in sdata do
        (let ((x (pobj-r (cadr (tpsy-psy i)))))
          (format stream "~A, ~A~%" (aref x 0) (aref x 1))))
      (close stream))))

(defun sim-thT (dtime el planet ec)
  (initparameters dtime planet)
  (init-seed planet ec)
  (setq el [el * getf(getf(*p-data* planet) :ap) ^ 2])
  (let (sdata (rdata 0v))
    (loop for x = *seed* then (next-T x #'total #'3X1+GM/rc2 #'1/gamma)
          while (< el
                   (let ((rx (aref (pobj-r (cadr (tpsy-psy x))) 0))
                         (ry (aref (pobj-r (cadr (tpsy-psy x))) 1)))
                     [+(rx ^ 2 ry ^ 2)]))
          finally (setq sdata x))
    (loop for x = sdata then (next-T x #'total #'3X1+GM/rc2 #'1/gamma)
          while (> el
                   (let ((rx (aref (pobj-r (cadr (tpsy-psy x))) 0))
                         (ry (aref (pobj-r (cadr (tpsy-psy x))) 1)))
                     [+(rx ^ 2 ry ^ 2)]))
          finally (setq sdata x))
    (loop for x = sdata then (next-T x #'total #'3X1+GM/rc2 #'1/gamma)
          while (< el
                   (let ((rx (aref (pobj-r (cadr (tpsy-psy x))) 0))
                         (ry (aref (pobj-r (cadr (tpsy-psy x))) 1)))
                     [+(rx ^ 2 ry ^ 2)]))
          do (setq rdata [rdata .+ pobj-r cadr tpsy-psy x]))
    (format T "24pi.. is ~d~%" (24pi3.. (getf (getf *p-data* planet) :ap) ec))
    (format T "simul result is ~d~%~%" [atan(aref(rdata 1) / aref(rdata 0))])))
```

The simple functions in the front part are expressed as simple functions to easily add the influence of various other factors, and the following are the simulation programs modified so that the functions can be used. It would be convenient to explain the detailed introduction and usage of the additional items with examples of actual use.

Execute the following command to see the effect of the induced acceleration.

```
VA> (time(sim-T #'-v{v.a}/c^2 nil nil
            [0.005 * minute] hour [0.4 * year]
            :Mercury 0.99992 "simT--vva-099992.txt"))
Evaluation took:
  474.817 seconds of real time
  473.768418 seconds of total run time (473.156799 user, 0.611619 system)
  [ Real times consist of 2.886 seconds GC time, and 471.931 seconds non-GC time. ]
  [ Run times consist of 2.867 seconds GC time, and 470.902 seconds non-GC time. ]
  99.78% CPU
  1,753,495,486,152 processor cycles
  792,483,127,088 bytes consed
```

This is the code that reflects the effect of induced acceleration,

```
(defun -v{v.a}/c^2 (a v) [-1 .* v .* a .* v %c ./ ^ 2.0d0])

VA> (macroexpand '[-1 .* v .* a .* v %c ./ ^ 2.0d0])
(./ (.* (.* (.* -1 V) A) V) (EXPT %C 2.0))
```

This code reflects the influence of $-\frac{\vec{u}}{c^2}(\vec{u}\cdot\vec{E})$.

```
(setf (pobj-a j)
          [ag .+ if(af funcall (af ag pobj-v j) 0v) .* if(gf funcall (gf pobj-v j) 1)])

(time(sim-T #'-v{v.a}/c^2 nil nil
            [0.005 * minute] hour [0.4 * year]
            :Mercury 0.99992 "simT--vva-099992.txt"))
```

The ”funcall” part of the ”next-T” function in the added code causes the ”-v{v.a}/c^2” function directed by the af argument to be called.

That is the factor that has been passed in as the ”#’-v{v.a}/c^2” argument when calling the simulation, and is finally summed. ”nil” means that there is no corresponding function, the ”gf” factor was ”nil” in the previous acceleration calculation, and the expression after the ”if” statement would be executed. If it is not ”nil”, a corresponding function is called and applied.

The ”next-T” function requires three arguments, ”af,” ”gf,” and additionally, one more argument, ”mf.” This ”mf” argument plays a role in the ”next-T” function’s next section,

```
(mm (if mf [pobj-m i * funcall(mf pobj-m i dr)] (pobj-m i)))
```

where it determines the function that gets multiplied to the mass of the Sun, the source of gravity. When ”nil”, it signifies no multiplication. Although it could be multiplied to the acceleration later in the calculation, the specific meaning of multiplying to the Sun’s mass will be explained later. To sum up, the ”sim-T” function extends the usage of a similar function described in the chapter on Maxwellian gravity. It takes the form (sim-T af mf gf …) with added arguments ”af,” ”mf,” and ”gf.” These arguments respectively represent a factor adding to the calculated acceleration, a factor multiplying to beforehand before all other factors take effect, and a factor multiplying to later after all other factors have been computed. The reasons for each of these factors will be explained gradually, but for now, I will start by examining the results of the simulation.

If run gnuplot and type the following command,

```
gnuplot> plot "simA--vva-099992.txt" lw 3 with lines
```

We will see the following graph.

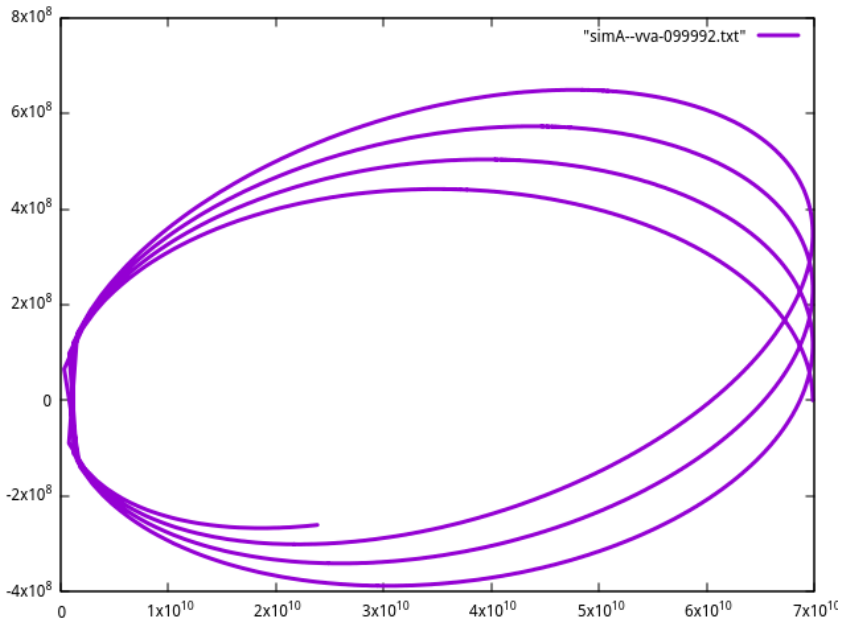

Figure 45: Effect of induced acceleration

I see an unexpected advance in orbit. It can be seen that exactly 1/3 of the actual orbital advance occurs. As a part of introducing a new acceleration, while a customary verification, an unforeseen and serious issue arose. While discussing Maxwellian gravity before, I found the deflection force of type $\frac{1}{c^2}\frac{2\gamma+1}{\gamma+1}\vec{v}\times\vec{a}_g\times\vec{v}$ as the cause of the advance of the orbit, and I thought the deflection force

$$\frac{1}{c^2}\frac{2\gamma+1}{\gamma+1}\vec{v}\times\vec{a}_g\times\vec{v}=\frac{1}{c^2}\vec{v}\times\vec{a}_g\times\vec{v}+\frac{1}{c^2}\frac{\gamma}{\gamma+1}\vec{v}\times\vec{a}_g\times\vec{v}$$

is composed of the sum of $\frac{1}{c^2}\vec{v}\times\vec{a}_g\times\vec{v}$, a pure bias term, and $\frac{1}{c^2}\frac{\gamma}{\gamma+1}\vec{v}\times\vec{a}_g\times\vec{v}$, a term due to Wigner rotation. This Wigner rotation term was inferred to be responsible for about 1/3 of the total deflection force as $\frac{\gamma}{\gamma+1}=\frac{1}{2}$ at low speeds. However, if a new 1/3 orbit advance effect is added here, the total rotation becomes 2 instead of 3/2, which becomes excessively large. It became evident that something from the previous calculation had been inaccurately included and needed to be left out.

In fact, when I came across this problem, I almost immediately came up with an excuse and evaded it. However, after 3 or 4 days, I finally thought about it again because I was not comfortable, and fortunately I found the right answer. Thinking again from the beginning, what came to my mind was the bac-cab rule again. It is the fact that

$$\frac{1}{c^2}\vec{v}\times\vec{E}\times\vec{v}=\frac{1}{c^2}\vec{E}(\vec{v}\cdot\vec{v})-\frac{1}{c^2}\vec{v}(\vec{E}\cdot\vec{v})$$

. Here, $\frac{1}{c^2}\vec{v}\times\vec{E}\times\vec{v}$ produces a 2/3 orbital rotation, and $-\frac{1}{c^2}\vec{v}(\vec{E}\cdot\vec{v})$ creates a 1/3 orbital rotation. If so, what I had to check was how much rotation would be made by $\frac{1}{c^2}\vec{E}(\vec{v}\cdot\vec{v})$.

```
(defun a*{v.v}/c^2 (a v) [v .* v .* a %c ./ ^ 2d0])
VA> (macroexpand '[v .* v .* a %c ./ ^ 2d0])
(./ (.* (.* V V) A) (EXPT %C 2.0))
```

The function in the above code will be executed, and the simulation is performed with the following command.

```
VA> (time(sim-T #'a*{v.v}/c^2 nil nil
            [0.005 * minute] hour [0.4 * year]
            :Mercury 0.99992 "simT-avv-099992.txt"))
Evaluation took:
  454.356 seconds of real time
  453.813166 seconds of total run time (453.387037 user, 0.426129 system)
  [ Real times consist of 2.396 seconds GC time, and 451.960 seconds non-GC time. ]
  [ Run times consist of 2.482 seconds GC time, and 451.332 seconds non-GC time. ]
```

```
99.88% CPU
1,677,933,855,494 processor cycles
755,866,108,960 bytes consed
```

If graph the result with the following command of gnuplot,

```
gnuplot> plot "simT-avv-099992.txt" lw 3 with lines, "simT--vva-099992.txt" lw 3 with lines
```

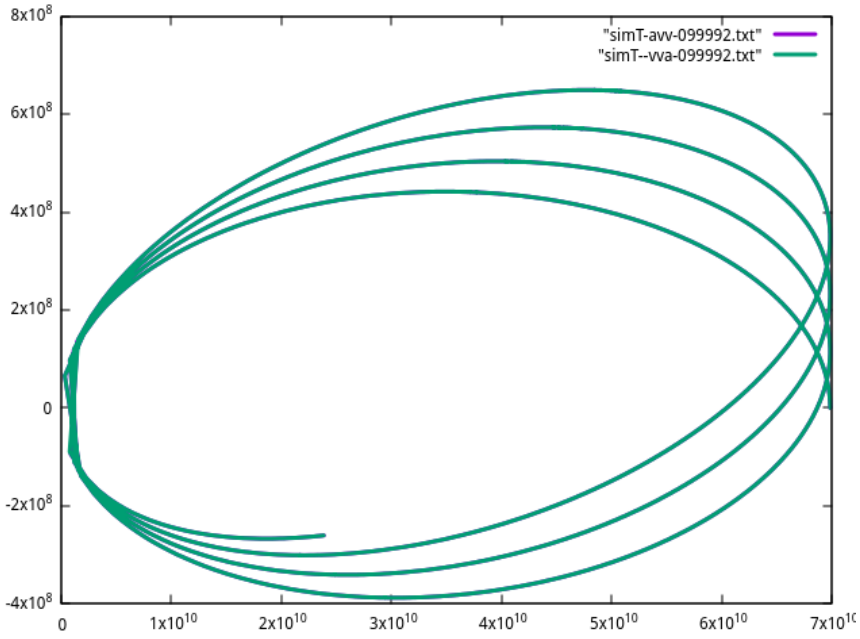

Figure 46: Comparison of the orbital advance

It can be seen that the effect of $\frac{1}{c^2}\vec{E}(\vec{v}\cdot\vec{v})$ and the effect of $-\frac{1}{c^2}\vec{v}(\vec{E}\cdot\vec{v})$ are precisely identical and generates a perihelion advancing effect of 1/3 of the total and 1/2 of $\frac{1}{c^2}\vec{v}\times\vec{E}\times\vec{v}$. The difference between the two is exceedingly minute, on the order of the simulation's margin of error.

From this, it became natural to speculate that what I initially thought to be a bias force in the form of $\frac{1}{c^2}\vec{v}\times\vec{E}\times\vec{v}$ was not actually a force, but a combination of a combined relativistic acceleration effect in the form of $-\frac{1}{c^2}\vec{v}(\vec{E}\cdot\vec{v})$ and some phenomenon represented by $\frac{1}{c^2}\vec{E}(\vec{v}\cdot\vec{v})$.

Then, does this imply the existence of the new force $\frac{1}{c^2}\vec{E}(\vec{v}\cdot\vec{v})$? I, however, focused more on the form of $\vec{v}\cdot\vec{v}=v^2$. The term involving v squared often appears in relation to energy or the γ factor. Rather than signifying a certain force, the appearance of this term suggests a relativistic effect could be the cause. In fact, during various simulations in the past, I noticed that relativistic effects could induce some degree of orbital rotation. However, at that time, the amount seemed negligible, and I didn't precisely ascertain its magnitude. Now that I know

there is not just one way to generate about 1/3 of the actual orbital advancement, I felt the question of whether such specific amounts of rotation are a common occurrence.

The first thing that came to mind was the fact that the mass of the sun should appear heavier as it gets closer. When an object with a certain mass in free space enters a gravitational field, it gains kinetic energy from the gravitational field, and its relativistic inertial mass will increase to that extent, but there should be no change in the mass when viewed from outside the gravitational field. Then, when the object collides with the gravitational source and emits light, the same amount of energy gained by gravity is emitted as light. And, as much as that energy, outside the gravitational field, the mass of the object is seen to be reduced. However, if we enter the gravitational field and measure the object from the near, it will be measured at its original mass. In other words, within a gravitational field, there's an effect that causes the mass of the source of gravity to appear larger than when observed from a distance.

Various formulas can be considered for how much it will be, but I selected the simplest form.

$$m_r = m_\infty \left(1 + \frac{Gm_\infty}{rc^2}\right)$$

This formula states that an object that appears to have a mass of m_∞ at an infinite distance will appear to have a mass of m_r at the distance r. There are various possibilities for this formula, as explained in the chapter on Maxwellian gravity, but I chose this form because all of them should be approximated in this form in a weak gravitational field around the sun. As for the question of what form other than this would be more correct, I have no intention of discussing it at this point. I vaguely remember making more elaborate calculations once upon a time that would have been a more plausible reason than this, but it would take days to reconstruct it, and even if such calculations were made, they would be only more elaborate arguments to the extent of confirming that they do not contradict themselves. I remember that it was not definitive. All human beings, including me, are beings who can invent any number of plausible reasons. What is needed is not a somewhat plausible reason, but an inevitable reason that must be so in connection with other physical phenomena. Unless such a thing has been discovered yet, it would be appropriate to use this formula, which is approximately certain under the weak gravity around the sun.

The code to express this is

```
(defun 1+GM/rc2 (m r) [1 + /(%G * m sqrt v.v r %c %c)])
```

Execution is

```
VA> (time(sim-T nil #'1+GM/rc2 nil
            [0.005 * minute] hour [0.4 * year]
            :Mercury 0.99992 "simT-gmrc-099992.txt"))
Evaluation took:
  446.308 seconds of real time
  445.795506 seconds of total run time (445.269655 user, 0.525851 system)
  [ Real times consist of 2.586 seconds GC time, and 443.722 seconds non-GC time. ]
  [ Run times consist of 2.644 seconds GC time, and 443.152 seconds non-GC time. ]
  99.89% CPU
  1,648,212,175,925 processor cycles
  755,851,858,912 bytes consed
```

Checking the result,

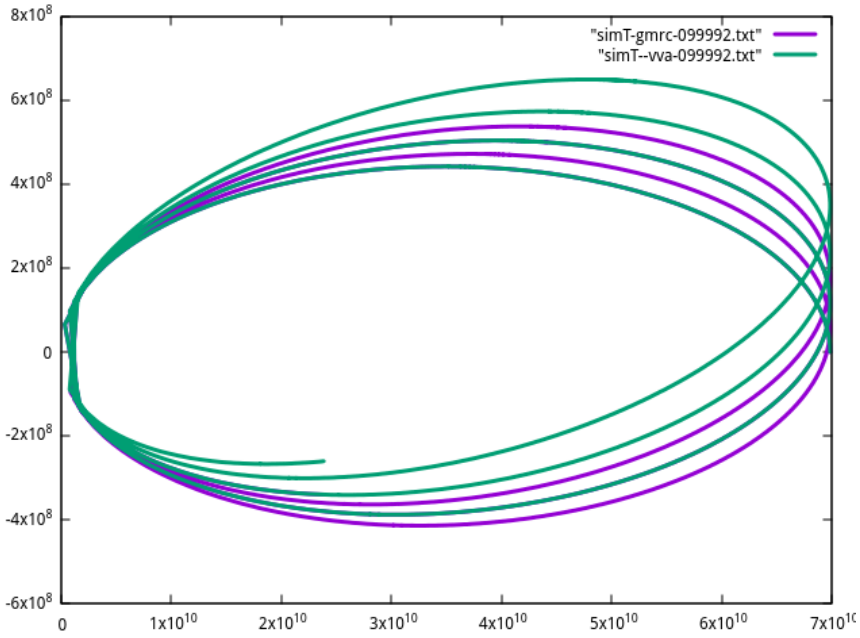

Figure 47: Effect of increasing the gravitational field

The violet line represents the advance of orbit due to the effect of increasing the gravitational field. It is confirmed that it is exactly half of the green line which is the induced acceleration effect, and 1/6 of the total orbital rotation amount. This is not enough. So, Does this mean the conjecture is incorrect? It is not. There is another additional orbital advance effect by the same formula.

The time dilation phenomenon within a gravitational field also becomes a cause for perihelion rotation. Through the previously discussed $\vec{a}' = \gamma^2((\gamma - 1)(\vec{a} \cdot \hat{v})\hat{v} + \vec{a})$ formula for the acceleration transformation between inertial frames, it was confirmed that on the side where time passes slowly, the acceleration on the side where time passes quickly looks much larger according to the square of the rate of time passage. This is because, from the perspective of slower time passage, faster changes in velocity are observed over shorter periods of time. If this phenomenon applies to time dilation caused by gravity as well, then as one gets closer to a gravitational source, there will be a perception that gravity becomes stronger than the inverse-square law would predict. Thus, this contributes to the cause of orbital rotation. When viewed from outside the gravitational field, as an object approaches a gravitational source, there's a slowing of time, and if the effect of gravity follows the inverse-square law just as it does outside, then from the object's point of view, gravity would appear to strengthen more than the inverse-square law predicts as it approaches the gravitational source. The formula can be used the same as the formula used for the effect of increasing the gravity field due to the increase in mass.

The code expressing the time delay effect is

```
(defun 2X1+GM/rc2 (m r) [1 + /(%G * m sqrt v.v r %c %c) ^ 2])
```

It is certain that this is about twice the effect of increasing gravitational field just before, even without actual simulation. In fact, the reason why I insisted on using the $m_r = m_\infty \left(1 + \frac{Gm_\infty}{rc^2}\right)$ form for the mass increase effect was largely because I wanted to use the same formula. If check the simulation,

```
VA> (time(sim-T nil #'2x1+GM/rc2 nil
            [0.005 * minute] hour [0.4 * year]
            :Mercury 0.99992 "simT-2gmrc-099992.txt"))
Evaluation took:
  451.972 seconds of real time
  451.466825 seconds of total run time (451.004237 user, 0.462588 system)
  [ Real times consist of 2.816 seconds GC time, and 449.156 seconds non-GC time. ]
  [ Run times consist of 2.795 seconds GC time, and 448.672 seconds non-GC time. ]
  99.89% CPU
  1,669,136,630,525 processor cycles
  761,956,138,480 bytes consed
```

```
gnuplot> plot "simT-2gmrc-099992.txt" lw 3 with lines, "simT--vva-099992.txt" lw 3 with lines
```

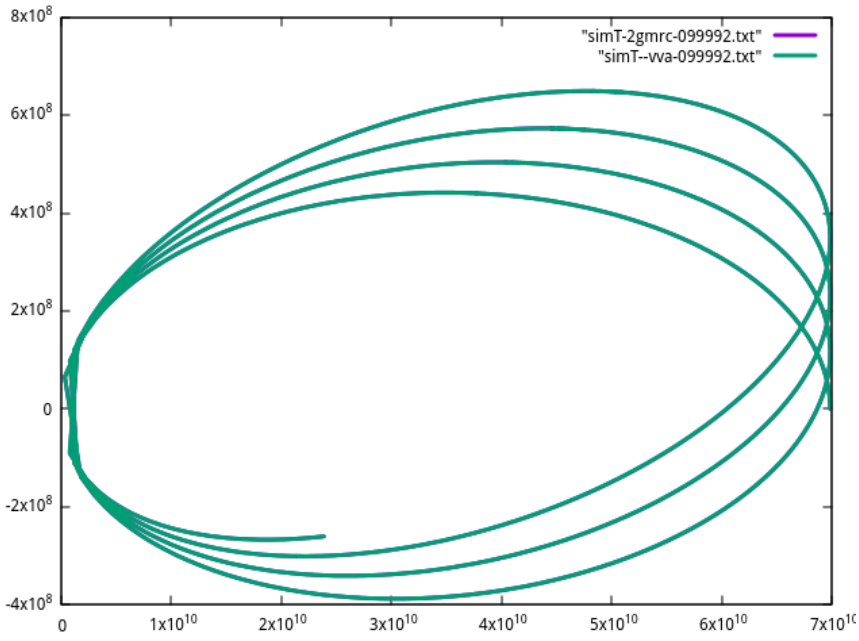

Figure 48: Time delay effect

As expected, it can be seen that it is exactly 1/3 of the total rotation amount.

However, the sum of the gravitational field increase effect and the time delay effect is 1/2, and it can be seen that the total amount of rotation exceeds by amount of 1/6 when the Wigner rotation and the induced acceleration are added together. This indicates the need for a reverse rotation factor of -1/6.

As the final factor necessary for this, I propose the rejection of the equivalence principle of gravitational mass and inertial mass.

It will be explained in more detail at the end of this chapter, but in reality, in the relativistic consistency problem of the previous section, it has already been confirmed that it is the inherent rest mass, not the relativistically increasing inertial mass, that acts with gravity. In the case of an elliptical orbit, the inertial mass becomes larger compared to the constant gravitational mass in the area close to the sun where the speed increases, so it is certain to reduce the effect of gravity and produce an orbital reversal effect. I will check if the effect is exactly 1/6 of the actual amount of rotation.

```
(defun 1/gamma (v) [1 v.v v %c sqrt - ./ ^ 2])
```

This is a simple function that calculates the 1/γ factor. Specify this as the gf factor to be multiplied at the end.

```
VA> (time(sim-T nil nil #'1/gamma
            [0.005 * minute] hour [0.4 * year]
            :Mercury 0.99992 "simT-gamma-099992.txt"))
Evaluation took:
  447.032 seconds of real time
  446.456864 seconds of total run time (445.964453 user, 0.492411 system)
  [ Real times consist of 2.503 seconds GC time, and 444.529 seconds non-GC time. ]
  [ Run times consist of 2.627 seconds GC time, and 443.830 seconds non-GC time. ]
  99.87% CPU
  1,650,878,611,561 processor cycles
  737,526,151,536 bytes consed

plot "simT-gamma-099992.txt" lw 3 with lines
```

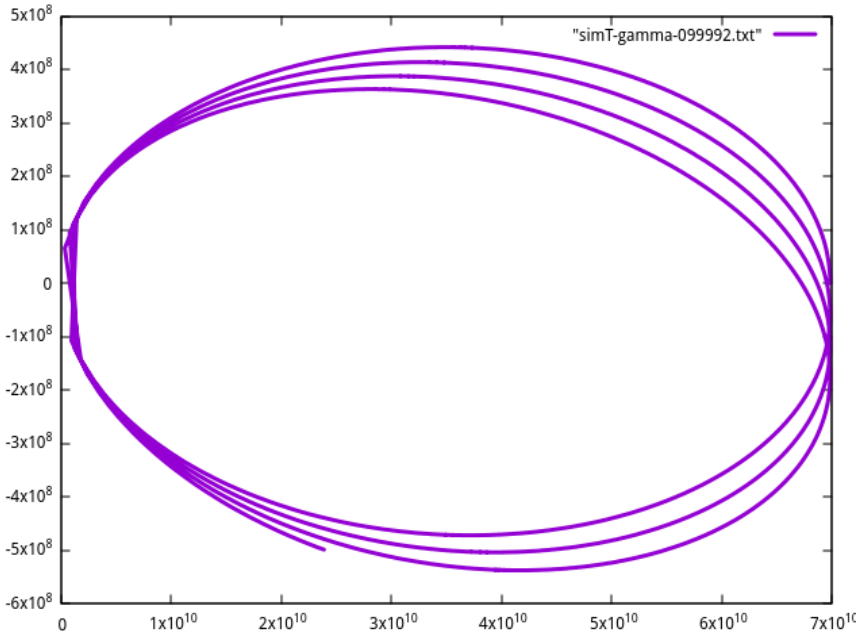

Figure 49: Effect of increasing inertial mass

As expected, it is a reverse rotation, and the amount is confirmed to be exactly -1/6.

As it turned out, the factors that could affect the orbit were all in integer proportionality, such as 1/3, 1/6, and -1/6 of the actual amount of rotation, in the case of weak gravity. The Gerber-Einstein formula, which seemed to be a complicated formula with 24π…, is in fact all

the factors that can affect the orbit are integer proportion to the actual amount. And with little mathematical tricks, even one of these can be used to produce the actual value that is observed in advance. It was an easy problem to fall into such a trap.

Now finally, by adding up all the effects, I will check whether the actual orbital rotation is simulated.

```
(defun total (a v) [v X* a X* v .* 0.5d0 v .* a .- .* v %c ./ ^ 2.0d0])
(defun 3X1+GM/rc2 (m r) [1 + /(%G * m sqrt v.v r %c %c) ^ 3])
(defun 1/gamma (v) [1 v.v v %c sqrt - ./ ^ 2])
```

The ”total” is the sum of the Wigner rotation $\frac{1}{2c^2}\vec{v} \times \vec{a} \times \vec{v}$ and the induced acceleration $-\frac{\vec{v}}{c^2}(\vec{v} \cdot \vec{a})$, and added to the computed acceleration.

The ”3X1+GM/rc2” is a term all multiplied by the gravitational mass increasing effect and the time delay effect $1 + \frac{Gm_\infty}{rc^2}$ in the gravitational field which is expressed as a 3rd power of one of them, and the mass is multiplied by this term.

The ”1/gamma” is the 1/γ which is the inertial mass increasing effect, and multiplied at the end.

```
VA> (time(sim-T #'total #'3X1+GM/rc2 #'1/gamma
            [0.005 * minute] hour [0.4 * year]
            :Mercury 0.99992 "simT-total-099992.txt"))
Evaluation took:
  718.235 seconds of real time
  717.423581 seconds of total run time (716.534989 user, 0.888592 system)
  [ Real times consist of 4.263 seconds GC time, and 713.972 seconds non-GC time. ]
  [ Run times consist of 4.416 seconds GC time, and 713.008 seconds non-GC time. ]
  99.89% CPU
  2,652,430,441,584 processor cycles
  1,176,641,252,688 bytes consed
```

```
gnuplot> plot "simT-total-099992.txt" lw 3 with lines
```

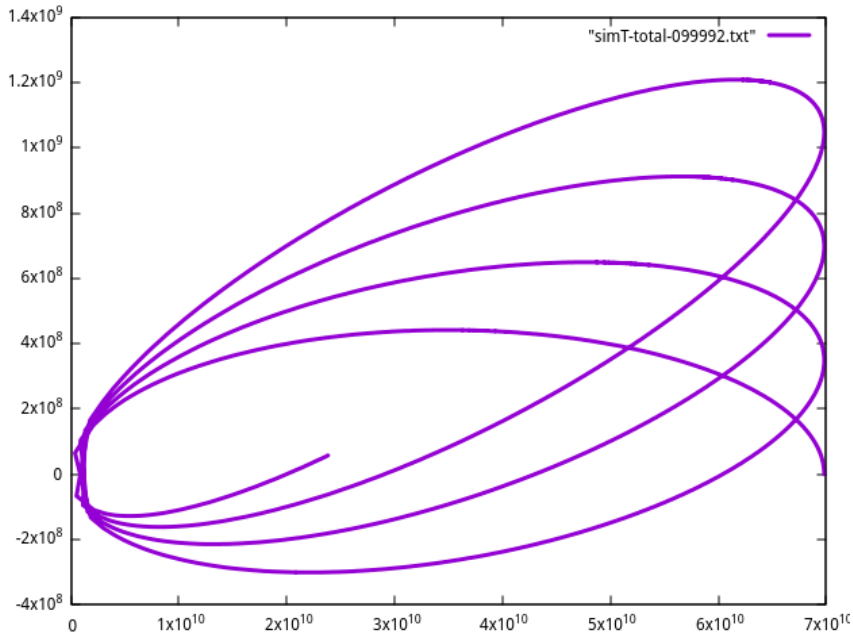

Figure 50: Total effect

When all effects are applied, the familiar rotation is confirmed. Comparing this rotation with the first discovered $\frac{3}{2c^2}\vec{v} \times \vec{a} \times \vec{v}$,

```
(defun 3/2*vXaXv/c2 (a v) [v X* a X* v .* 1.5d0 %c ./ ^ 2.0d0])
```

```
VA> (time(sim-T #'3/2*vXaXv/c2 nil nil
            [0.005 * minute] hour [0.4 * year]
            :Mercury 0.99992 "simT-32vav-099992.txt"))
Evaluation took:
  525.111 seconds of real time
  524.517250 seconds of total run time (523.901337 user, 0.615913 system)
  [ Real times consist of 3.016 seconds GC time, and 522.095 seconds non-GC time. ]
  [ Run times consist of 3.159 seconds GC time, and 521.359 seconds non-GC time. ]
  99.89% CPU
  1,939,221,533,452 processor cycles
  890,103,666,736 bytes consed
```

```
gnuplot> plot "simT-32vav-099992.txt" lw 3 with lines, "simT-total-099992.txt" lw 3 with lines
```

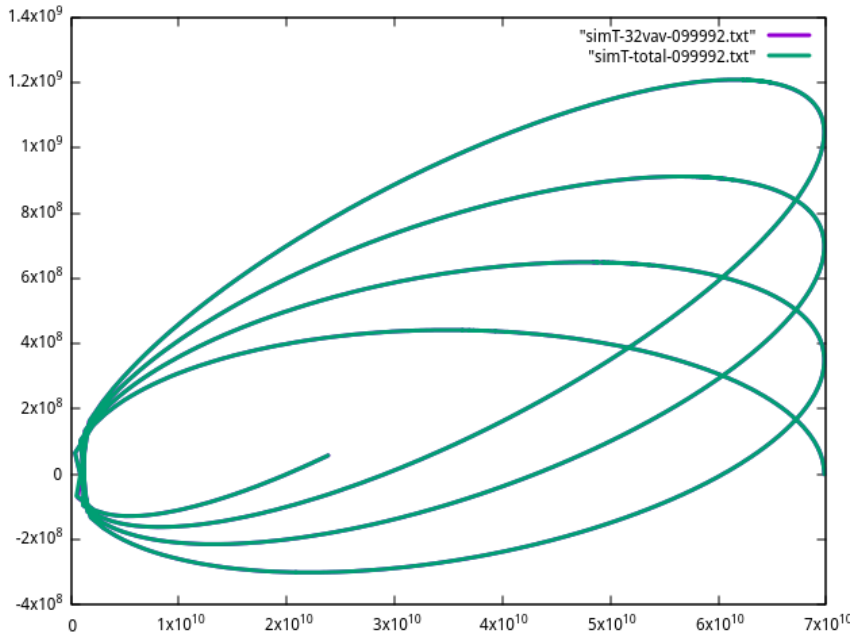

Figure 51: Comparison

Not only the final rotation amount of the orbit but also the middle trajectory can be seen to be completely consistent unless we zoom in to check the simulation error.

Comparing with the Gerber-Einstein formula at different eccentricities,

```
VA> (time(sim-thT [0.05 * minute] 0.9999999 :Mercury 0.9920563069))
24pi.. is 5.018881066308678e-5
simul result is 5.017394317114734e-5

Evaluation took:
  12.486 seconds of real time
  12.470078 seconds of total run time (12.426843 user, 0.043235 system)
  [ Real times consist of 0.073 seconds GC time, and 12.413 seconds non-GC time. ]
  [ Run times consist of 0.080 seconds GC time, and 12.391 seconds non-GC time. ]
  99.87% CPU
  46,118,766,698 processor cycles
  19,707,270,640 bytes consed

NIL
VA> (time(sim-thT [0.005 * minute] 0.9999999 :Mercury 0.920563069))
24pi.. is 5.018881066308663e-6
simul result is 5.029979866876237e-6

Evaluation took:
```

```
  92.485 seconds of real time
  92.358802 seconds of total run time (92.255654 user, 0.103148 system)
  [ Real times consist of 0.576 seconds GC time, and 91.909 seconds non-GC time. ]
  [ Run times consist of 0.593 seconds GC time, and 91.766 seconds non-GC time. ]
  99.86% CPU
  341,546,951,197 processor cycles
  144,971,876,992 bytes consed

NIL
VA> (time(sim-thT [0.00005 * minute] 0.9999999 :Mercury 0.20563069))
24pi.. is 5.018881066308666e-7
simul result is 5.059859527929451e-7

Evaluation took:
  6726.542 seconds of real time
  6717.647228 seconds of total run time (6710.227861 user, 7.419367 system)
  [ Real times consist of 44.927 seconds GC time, and 6681.615 seconds non-GC time. ]
  [ Run times consist of 45.657 seconds GC time, and 6671.991 seconds non-GC time. ]
  99.87% CPU
  24,840,978,194,961 processor cycles
  10,604,676,773,312 bytes consed
```

The results are (5.017394317114734e-5, 5.029979866876237e-6, 5.059859527929451e-7), respectively, and it can be seen that the calculation results are almost similar to the previous case (5.0148822758972275e-5, 5.0303364610700276e-6, 5.061111510661374e-7).

I was frustrated that the last simulation result for the actual orbit of Mercury had the largest error, so I increased the simulation time by about ten times. The second result of increasing the eccentricity of Mercury tenfold was also given ten times more time.

```
VA> (time(sim-thT [0.0005 * minute] 0.9999999 :Mercury 0.920563069))
24pi.. is 5.018881066308663e-6
simul result is 5.017730982221946e-6

Evaluation took:
  935.140 seconds of real time
  933.717519 seconds of total run time (932.739797 user, 0.977722 system)
  [ Real times consist of 6.170 seconds GC time, and 928.970 seconds non-GC time. ]
  [ Run times consist of 6.246 seconds GC time, and 927.472 seconds non-GC time. ]
  99.85% CPU
  3,453,449,704,426 processor cycles
  1,449,513,520,976 bytes consed

NIL
VA> (time(sim-thT [0.000005 * minute] 0.99999999 :Mercury 0.20563069))
24pi.. is 5.018881066308666e-7
simul result is 5.018808203222051e-7

Evaluation took:
```

```
66296.080 seconds of real time
66184.153402 seconds of total run time (66107.116280 user, 77.037122 system)
[ Real times consist of 386.245 seconds GC time, and 65909.835 seconds non-GC time. ]
[ Run times consist of 391.821 seconds GC time, and 65792.333 seconds non-GC time. ]
99.83% CPU
-152,395,993,960,830 processor cycles
1 page fault
106,033,400,113,984 bytes consed
```

Both show significant improvements. In particular, it can be confirmed that the result obtained after about 18 hours of simulation by inserting the actual eccentricity of Mercury agrees with the Gerber-Einstein formula within the current observation error limit without applying the γ factor to the Wigner rotation term.

This is a surprising result in many ways. A new term was discovered, so I did a customary simulation, which was just a routine. However, I found that it produced the same orbital advance as the Wigner rotation term. And since the new term breaks the existing explanation, I tried to find a new explanation. As a result, surprisingly, half of what I initially thought was the deflection force was the newly discovered induced acceleration, and the other half was the sum of miscellaneous relativistic effects. Through the unexpected revelation that all three relativistic effects were contributing exactly 1/3, 1/6, and -1/6 to the orbit rotation, I once again experienced the unpredictability of nature extending well beyond human imagination.

As I began this chapter, my initial plan to investigate whether the deflection force was a relativistically consistent force was resolved by itself by the discovery that the deflection force was in fact a phenomenon caused by the newly discovered induced acceleration and relativistic side effects. It was shown that the induced acceleration is a relativistically consistent effect that is always described in the same form relativistically through numerical calculation of the arbitrary velocity difference in any direction. And the relativistic side effects are not fields in the first place. Additionally, it was shown that the acceleration itself can be converted to other inertial system expressions through the acceleration conversion technique between inertial systems. These are consistent phenomena in terms of relativity, and induced acceleration in particular is a phenomenon that completes a chapter in relativity and electromagnetism that was incomplete without it. And to add one more thing, I originally started with the idea that an elliptical orbit with the same eccentricity in the electromagnetic force would exhibit the same perihelion rotation as gravity, but interestingly, the final conclusion is that the rotation of the orbit by the electromagnetic force is only half of the effect by gravity.

The original intention was to examine the relativistic consistency of the biasing force, but I encountered and solved the problem of the relativistic consistency of the electromagnetic force itself, which is much older. In the process, the deflection force itself was decomposed into a more fundamental form, and it was found that it was a phenomenon caused by induced