6.4 유도 가속에 대한 시뮬레이션

앞서 맥스웰 중력 이론을 기술하면서 우연히 발견한 $\frac{1}{c^2}\frac{2\gamma+1}{\gamma+1}\vec{v}\times\vec{E}\times\vec{v}$ 항에 따르는 힘을 소개하였었고, 이 힘이 수성 궤도의 근일점 전진을 설명하는 것을 시뮬레이션으로 보였었다. 지금 또다시 새로운 $-\frac{\vec{u}}{c^2}(\vec{u}\cdot\vec{E})$ 에 따르는 가속도를 발견하였으니 이 가속도는 혹시 궤도에 어떤 영향을 미치지는 않는지 한번 확인해 볼 필요가 있을 것이다.

먼저 앞서 맥스웰 중력 시뮬레이션에서 했던 대로 github(https://github.com/kycgit/gsimm) 에서 프로그램을 다운 받은 후 그 저장 폴더에서 common-lisp 을 실행한다. 그리고 다음 명령을 실행한다.

```
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>
```

이전의 경우에는 한 가지 요인이 미치는 영향만 시뮬레이션 해보는 것으로 충분했지만 이제는 보다 여러 종류의 요인이 미치는 영향을 각기 시뮬레이션을 해보아야 하기 때문에, 각 요인들에 대하여 모두 프로그램을 짜면 전체적으로 반복 설명해야 할 것이 너무 길어져서 각 요인들을 간단한 함수로 정의하고 그것을 필요할 때마다 호출하여 시뮬레이션 하도록 프로그램을 손 보았다. 새로이 추가된 부분들은 다음과 같다.

```
(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))]))
```

앞 부분의 간단한 함수들은 여러가지 다른 요인들의 영향을 쉽게 추가하기 위해 간단한 함수로 표현한 것들이고, 이후는 그 함수들을 사용할 수 있도록 수정한 시뮬레이션 프로그램이다. 추가 사항에 대한 자세한 소개와 사용법은 실제 사용 예와 함께 설명하는 것이 편리 하리라.

유도 가속도의 영향을 보기 위하여 다음의 명령을 실행한다.

```
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
```

이 명령은 유도 가속도의 영향을 반영하는 코드로서,

```
(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))
```

$-\frac{\vec{u}}{c^2}(\vec{u}\cdot\vec{E})$ 의 영향을 반영하는 코드이다.

추가된 코드의 "next-T" 함수의

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

"funcall" 부분을 통해 af 인자가 지시하는 "-v{v.a}/c^2" 함수가 호출되게 되며, 이는 시뮬레이션 호출 시에

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

"#' -v{v.a}/c^2" 인자로 표기되어 전달되었던 요인으로서 최종적으로 더해지게 된다. "nil"은 해당 함수는 없다는 의미로서 앞서 가속도 계산 시에 "gf" 인자가 "nil"로 없다는 의미이며, "if"문의 뒤쪽 표현 식이 실행되게 된다. "nil"이 아닐 때에는 해당 함수가 호출되어 적용되게 된다.

"next-T" 함수가 요구하는 인자는 "af", "gf" 외에 "mf" 가 하나 더 있으며 이는 "next-T" 함수의 아래 부분,

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

중력의 원천 태양의 질량을 불러오는 부분에서 태양의 질량에 곱해지게 되는 함수를 결정한다. "nil" 일 때는 곱하지 않는다는 의미이다. 계산의 나중에 가속도에 곱해도 되지만 굳이 태양 질량에 곱한 의미는 차후에 설명할 것이다. 종합하여 설명하면, "sim-T" 함수는 맥스웰 중력의 장에서 설명한 비슷한 함수의 사용법에 (sim-T af mf gf …) 형태로 "af", "mf", "gf" 인자가 추가된 함수이며, 이들은 각각, "af" 는 계산된 가속도에 더해지는 요인을 표현한 것이며, "mf"는 모든 다른 요인이 작용하기 전에 미리 곱해지는 영향이며, "gf"는 모든 다른 요인이 계산된 이후 나중에 곱해지는 요인을 표현하는 인자이다. 각 요인들이 필요한 이유는 차차 설명하기로 하고, 방금의 시뮬레이션 결과부터 살펴 보기로 한다.

gnuplot을 실행하고 다음 명령을 입력하면,

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

다음의 그래프를 보게 된다.

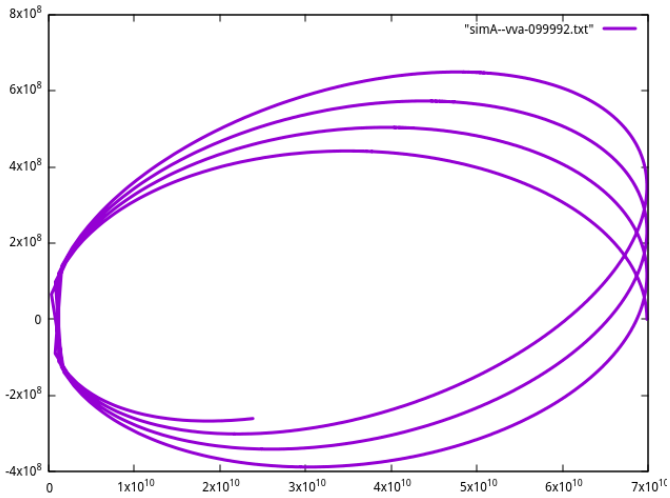

Figure 87: 유도 가속의 효과

나로서는 예상치 않았던 궤도의 전진을 보게 된다. 실제 궤도 전진의 정확히 1/3 이 발생함을 볼 수 있다. 새로운 가속도를 도입하는 김에 기계적으로 확인한 것인데 뜻밖의 심각한 문제가 생겼다. 앞서 맥스웰 중력을 논하면서 나는 궤도의 전진을 만드는 원인으로서 $\frac{1}{c^2}\frac{2\gamma+1}{\gamma+1}\vec{v}\times\vec{a}_g\times\vec{v}$ 형식의 편향력을 찾아 내었었고, 그 편향력은 $\frac{1}{c^2}\frac{2\gamma+1}{\gamma+1}\vec{v}\times\vec{a}_g\times\vec{v}=\frac{1}{c^2}\vec{v}\times\vec{a}_g\times\vec{v}+\frac{1}{c^2}\frac{\gamma}{\gamma+1}\vec{v}\times\vec{a}_g\times\vec{v}$ 로서 순수 편향력 항인 $\frac{1}{c^2}\vec{v}\times\vec{a}_g\times\vec{v}$ 과 위그너 회전에 기인한 항인 $\frac{1}{c^2}\frac{\gamma}{\gamma+1}\vec{v}\times\vec{a}_g\times\vec{v}$ 의 합으로 구성된 것으로 생각하였고. 이 위그너 회전 항은 느린 속도에서 $\frac{\gamma}{\gamma+1}=\frac{1}{2}$ 로서 전체 편향력의 1/3 정도를 담당하는 것으로 추론 하였었다. 그런데 여기에 새로운 1/3 궤도 전진 효과가 추가되면 전체 회전이 3/2이 아닌 2가 되어 지나치게 커진다. 기존 계산의 무언가는 잘못 들어간 것으로서 빠져야만 했다.

사실 나는 이 문제를 맞닥뜨리자 거의 즉시 한 가지 변명을 만들어 내었고 얼버무렸었다. 그러나 한 3~4일 후 결국 편치 않은 마음에 다시 생각해 보았고 다행히 올바른 답을 찾아내게 되었다. 다시 처음부터 생각해보자 떠오른 것은 또 다시 bac-cab 규칙 이었다. 바로 $\frac{1}{c^2}\vec{v}\times\vec{E}\times\vec{v}=\frac{1}{c^2}\vec{E}(\vec{v}\cdot\vec{v})-\frac{1}{c^2}\vec{v}(\vec{E}\cdot\vec{v})$ 이란 사실이다. 여기서, $\frac{1}{c^2}\vec{v}\times\vec{E}\times\vec{v}$ 이 2/3의 궤도 회전을 만들어 내는데 $-\frac{1}{c^2}\vec{v}(\vec{E}\cdot\vec{v})$ 이 1/3의 궤도 회전을 만들어 내는 것이다. 그렇다면 $\frac{1}{c^2}\vec{E}(\vec{v}\cdot\vec{v})$ 는 얼마 만큼의 회전을 만들어 낼 것인가를 확인하여 보았다.

```
(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))
```

위 코드의 함수를 실행할 것이며, 다음의 명령으로 시뮬레이션을 수행한다.

```
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
```

결과를 gnuplot 의 다음 명령을 통해 그래프로 그려보면,

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

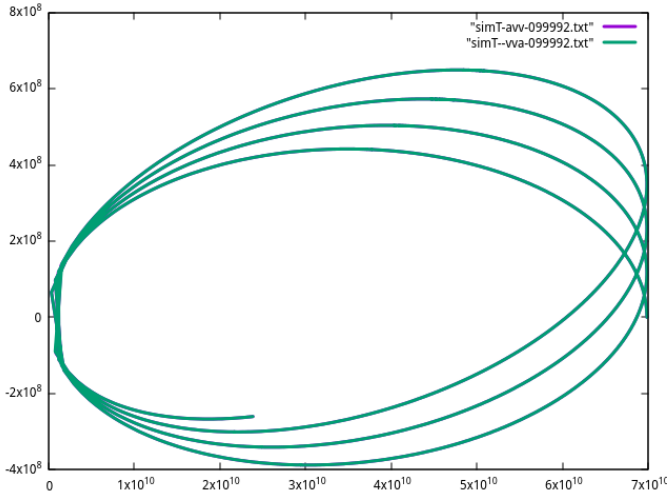

Figure 88: 궤도전진 비교

$\frac{1}{c^2}\vec{E}(\vec{v}\cdot\vec{v})$ 의 효과와 $-\frac{1}{c^2}\vec{v}(\vec{E}\cdot\vec{v})$ 의 효과가 정확히 일치하며, 전체의 1/3, $\frac{1}{c^2}\vec{v}\times\vec{E}\times\vec{v}$ 의 1/2 씩의 근일점 전진 효과를 만들어내는 것을 볼 수 있다. 둘의 차이는 시뮬레이션 오차 정도로 극히 미미하다.

이것으로 내가 애초에 $\frac{1}{c^2}\vec{v}\times\vec{E}\times\vec{v}$ 꼴의 편향력으로 생각 했던 것은 사실은 힘이 아니라 $-\frac{1}{c^2}\vec{v}(\vec{E}\cdot\vec{v})$ 꼴의 상대론적인 가속 효과와 $\frac{1}{c^2}\vec{E}(\vec{v}\cdot\vec{v})$ 꼴의 어떤 현상의 합이라는 추측이 자연 스러워 졌다.

그렇다면 이것은 새로운 힘 $\frac{1}{c^2}\vec{E}(\vec{v}\cdot\vec{v})$ 의 존재를 의미할 것인가? 나는 그 보다는 $\vec{v}\cdot\vec{v}=v^2$ 의 형태에 주목하였다. 이 v 제곱 항은 에너지나 γ 인자 관련하여 자주 등장하는 항으로서, 이 항의 등장이 의미하는 것은 어떤 힘이라기 보다는 어떤 상대론 적인 효과가 원인일 수 있다고 생각하였다. 사실 예전에 이런 저런 시뮬레이션을 할 때 상대론 적인 효과들이 어느 정도 궤도의 회전을 일으키는 것은 본 적이 있었다. 그러나 그 당시에는 그 양이 미미하여 정확히 어느 정도인지 확인하지 않았었는데, 이제 실제 궤도 전진의 1/3 정도를 일으키는 방법이 단 하나가 아닌 것을 알게 된 이상 이런 특정한 양 만큼의 회전이 흔한 현상은 아닌지 확인해 볼 필요를 느꼈다.

가장 먼저 생각난 것은 태양의 질량이 가까이 다가갈 수록 더 무겁게 보여야 한다는 사실 이었다. 자유 공간에서 어떤 질량을 가지고 있던 어떤 물체가 중력장에 들어서면 중력장에 의하여 운동에너지를 얻게 되는데, 그 만큼 상대론 적인 관성 질량이 증가 하겠지만 중력장 외부에서 볼 때에는 그 질량에 변화가 없어야 한다. 이후 그 물체가 중력원과 충돌하여 빛을 방출하고 나면 중력으로 얻게 된 에너지만큼 빛으로 내보내게 된다. 그리고, 그 에너지 만큼 중력장 바깥에서는 그 물체의 질량이 줄어든 것으로 보게 된다. 하지만 중력장 안에 들어가서 그 물체를 곁에서 측정하면 원래의 질량으로 측정 될 것이다. 즉 중력장 안에서는 중력원의 질량이 멀리서 볼 때 보다 더 커 보이게 되는 효과가 있다.

그것이 어느 정도일 것 인지에 대해서는 여러가지 공식을 고려할 수 있지만 나는 가장 단순한 형태를 생각하였다.

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

무한대 거리에서 m_∞ 의 질량을 가진 것으로 보이는 물체는 r거리로 다가가면 m_r 질량으로 보일 것이라는 공식이다. 이러한 공식은 앞서 맥스웰 중력의 장에서 설명한 것처럼 여러가지 가능성이 있지만, 태양 주위 정도의 약한 중력장에서는 전부 이 형태로 근사될 것이므로 나는 이 형태를 선택하였다. 이것 외에 다른 어떤 형태가 옳을 것인가 하는 문제에 대하여 나는 지금 시점에서는 논할 생각이 없다. 이것보다는 좀 더 그럴듯한 이유가 될 더 정교한 계산을 예전에 했었던 기억은 어렴풋이 나지만 그것을 재구성 하는 데에는 며칠 걸릴 것이고, 그러한 계산을 한다 하더라도 그것은 자체적인

모순은 없다는 것을 확인하는 정도로서 좀 더 정교한 주장이 될 뿐 확정적인 정도는 아니었던 것으로 기억한다. 나를 포함한 모든 인간은 어느 정도 그럴듯한 이유따윈 얼마든지 만들어낼 수 있는 존재이다. 필요한 것은 어느 정도 그럴듯한 이유가 아니라 다른 물리 현상과 연계되어 반드시 그래야만 하는 필연적인 이유이다. 그런것을 아직 발견하지 못한 이상은 태양 주위 정도의 약한 중력 하에서는 근사적으로 반드시 이래야 할 것이라고 확신할 수 있는 이 공식을 사용하는 것이 적당할 것이다.
이것을 표현하는 코드는,

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

이다. 실행은,

```
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
```

이며, 결과를 확인해 보면,

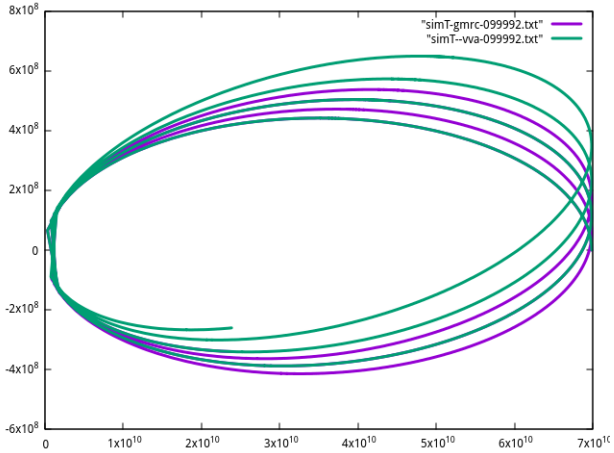

Figure 89: 중력장 증가 효과

보라색 선이 중력장 증가 효과로 인한 궤도의 전진이다. 녹색선인 유도 가속 효과의 정확히 절반, 총 궤도 회전량의 1/6 임이 확인된다. 이것으로는 부족하다. 그렇다면 이것은 잘못된 추측인가? 그렇지 않다. 동일한 공식에 의한 궤도 전진 효과가 더 있다.

중력장 내의 시간 지연 현상 역시 근일점 회전의 원인이 된다. 앞서 관성계 간의 가속도 변환 공식 $\vec{a}' = \gamma^2((\gamma-1)(\vec{a}\cdot\hat{v})\hat{v} + \vec{a})$ 을 통해 시간이 천천히 흐르는 측에서는 빨리 흐르는 측의 가속도를 시간이 흐르는 속도의 제곱만큼 크게 보게 된다는 것을 확인 하였었다. 시간이 느리게 흐르는 입장에선 보다 짧은 시간 동안 보다 빠른 속도 변화를 관찰하게 되기 때문이다. 그 현상이 중력에 의한 시간 지연에서도 적용된다면 중력원에 가까운 곳에서는 역제곱 법칙 이상으로 중력이 강해지는 것으로 느끼게 될 것이므로 궤도가 회전하는 원인이 된다. 중력장 바깥에서 볼 때, 물체가 중력원에 접근함에 따라 시간이 느려지는 효과가 있는데 중력의 효과가 역제곱의 법칙 그대로 적용 된다면 물체의 입장에서는 중력원에 접근 할수록 역제곱 법칙 이상으로 중력이 강해 지는 것으로 느끼게 될 것이다. 그 공식은 마침 질량 증가에 의한 증력장 증가 효과에서 사용한 공식과 동일하게 사용할 수 있다.

시간 지연 효과를 표현하는 코드는

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

로서, 이것은 실제 시뮬레이션을 해보지 않아도 조금 전의 중력 증가 효과의 두배 정도 일 것은 확실하다. 사실 질량 증가 효과를 $m_r = m_\infty\left(1 + \frac{Gm_\infty}{rc^2}\right)$ 꼴을 고집한 것은 같은 공식을 사용하고 싶었던 이유가 컸다. 시뮬레이션으로 확인해보면,

```
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

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

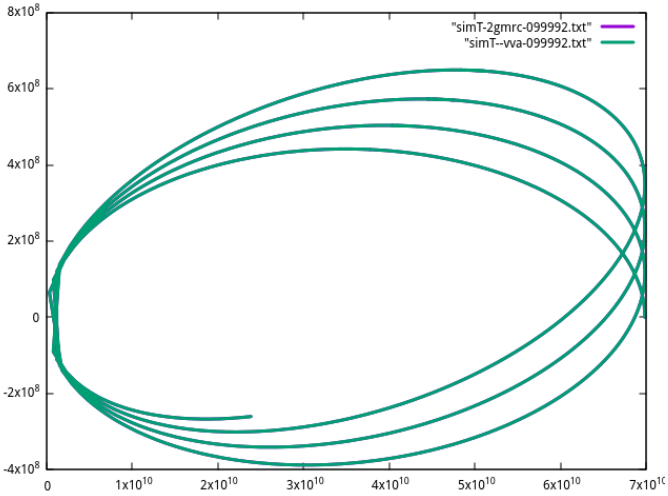

Figure 90: 시간지연 효과

예상대로 정확히 전체 회전량의 1/3 임을 볼 수 있다.

그런데 중력장 증가 효과와 시간 지연 효과를 합치면 1/2 로서 위그너 회전과 유도 가속도의 영향까지 전부 합하면 전체 회전 량이 1/6만큼 넘치는 것을 알 수 있다. 뭔가 역회전 -1/6 요인이 필요하다.

그에 필요한 마지막 요인으로서 중력 질량과 관성 질량의 등가원리 폐기를 제시한다.

이 장의 말미에서 좀 더 자세히 설명할 예정 이지만, 실은 앞서 다루었던 장의 상대론 적인 일관성 유지 문제에서 중력과 작용하는 것은 상대론 적으로 증가하는 관성 질량이 아닌 고유한 정지 질량임을 이미 확인할 수 있었다. 그 현상은 타원 궤도의 경우 속도가 빨라지는 태양과 가까운 지역에서 일정한 중력 질량에 비해 관성 질량은 더 커지게 되므로 중력의 영향을 줄이는 궤도 역회전 효과를 낼 것이 확실하다. 그 영향이 정확히 실제 회전량의 1/6 인지 확인해 볼 것이다.

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

단순한 1/γ 인자를 계산하는 함수이다. 이를 마지막에 곱해주는 gf 인자로 지정한다.

```
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
```

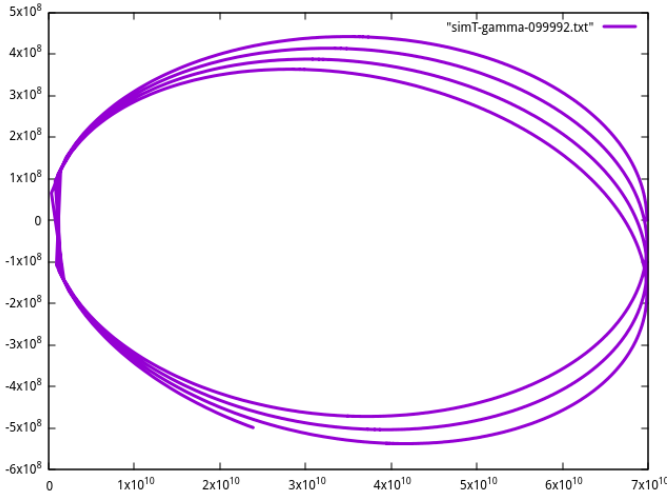

Figure 91: 관성질량 증가 효과

예상대로 역회전이고, 그리고 그 양은 정확히 -1/6 임을 확인할 수 있다.

알고보니 궤도에 영향을 끼칠만한 요인들은 그 영향이 약한 중력에서는 모조리 실제 회전량의 1/3, 1/6, -1/6 등으로 정수 비례 관계에 있었음을 알게 되었다. 24π…로 복잡한 공식으로 보였던 거버-아인슈타인 공식은 실은 웬만한 궤도에 영향을 줄만한 요인들이 전부 실제량의 정수 비례로서 약간의 수학적인 잔 재주만 동원하면 이들 중의 하나만 가지고도 미리 관측된 실제 값을 만들어 내어버리는 함정에 빠지기 쉬운 문제였다.

이제 모든 영향들을 다 더해서 실제 궤도 회전량이 시뮬레이션 되는지를 최종 확인해 볼 것이다.

```
(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])
```

"total"은 위그너 회전 $\frac{1}{2c^2}\vec{v}\times\vec{a}\times\vec{v}$ 과 유도 가속도 $-\frac{\vec{v}}{c^2}(\vec{v}\cdot\vec{a})$ 의 합으로서 계산된 가속도에 더해진다.

"3X1+GM/rc2"은 중력장 내에서의 중력 질량 증가 효과와 시간 지연 효과 $1+\frac{Gm_\infty}{rc^2}$ 를 모두 곱하여 3 제곱으로 표현한 항으로 질량에 곱해진다.

"1/gamma"는 1/γ 로서 관성질량 증가 효과이며 마지막에 곱해진다.

```
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

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

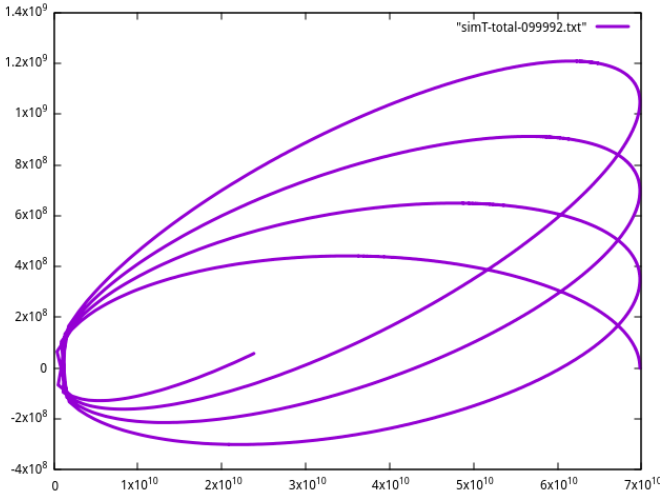

Figure 92: 전체 효과

전체 효과를 모두 적용하면 익숙한 회전이 확인 된다. 이 회전을 처음 발견하였던 $\frac{3}{2c^2}\vec{v}\times\vec{a}\times\vec{v}$ 과 비교하면,

```
(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
```

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

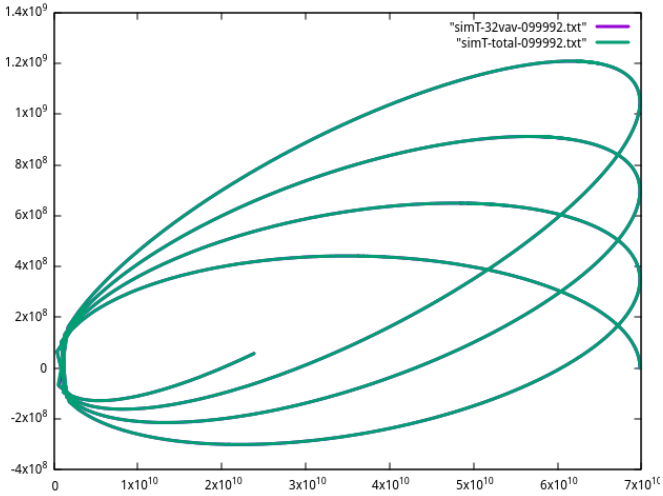

Figure 93: 비교

궤도의 최종 회전량 뿐만 아니라 중간의 궤적도 확대하여 시뮬레이션 오차를 확인하지 않는 한 완전히 일치하는 것을 볼 수 있다.

각기 다른 이심률에서 거버-아인슈타인 공식과의 비교를 해보면,

```
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
```

결과는 각각
(5.017394317114734e-5, 5.029979866876237e-6, 5.059859527929451e-7)
로서 예전의 같은 경우의 계산 결과
(5.0148822758972275e-5, 5.0303364610700276e-6, 5.061111510661374e-7)
들과 거의 흡사한 것을 확인할 수 있다.

마지막 실제 수성 궤도에 대한 시뮬레이션 결과가 가장 오차가 큰 것이 마음에 걸려 시뮬레이션 시간을 열 배 정도 늘려보았다. 두 번째 수성의 이심률을 열 배로 늘린 결과도 열 배 정도의 시간을 더 주어 보았다.

```
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
```

둘 다 많이 개선되는 것을 볼 수 있다. 특히 실제 수성의 이심률을 넣어서 약 18 시간 동안의 시뮬레이션 끝에 나온 결과는 위그너 회전 항에 γ인자 적용없는 상태로 거버-아인슈타인 공식과 현재의 관측 오차 한계 이내로 일치하는 것을 확인할 수 있다.

여러가지로 놀라운 결과이다. 새로운 항이 발견되었길래 그저 당연한 습관적인 시뮬레이션을 해보았는데 위그너 회전항과 동일한 궤도 전진을 보게 되었다. 그리고 그 새로운 항은 기존의 설명을 깨뜨리기에 궁리해 보았는데 결과적으로 놀랍게도 내가 처음 편향력이라고 생각했던 것의 절반의 정체는 새로이 발견한 유도 가속도 이었고, 나머지 절반은 잡다한 상대론적 효과들의 총합이었다는 것을 알게 되었다. 세 가지 상대론적 효과가 전부 1/3, 1/6, -1/6 만큼 궤도 회전에 대하여 기여하고 있었다는 뜻밖의 사실을 통해 자연의 예측 불가능성은 인간의 상상력 정도는 확실히 뛰어넘는 것을 다시 한번 체험할 수 있었다.

이 장을 시작하면서 애초에 계획한 편향력이 상대론 적으로 일관성 있는 힘 인가에 대한 확인은 그 편향력이 사실은 새로이 발견한 유도 장력과 상대론 적인 부수 효과들에 의한 현상임을 발견하는 것으로 저절로 해결이 되었다. 유도장력은 임의 방향 속도 차이에 대한 수치 계산을 통하여 상대론 적으로 항상 동일한 형태로 기술되는 상대론 적으로 일관적인 현상 임을 보였고 상대론 적인 부수 효과 들은 애초에 장이 아니며, 그 운동 자체는 관성계들 간의 가속도 변환 기법을 통해 얼마든지 다른 관성계 표현으로 옮길 수 있음을 보였다. 이들은 상대론 적으로 완전히 올바른 현상들이며, 특히 유도 가속은 그 것이 없이는 불완전한 상대론과 전자기학의 한 단원을 완성 시키는 힘이다. 그리고 한 가지 더 부연 하자면 나는 애초에 전자기력에서도 동일한 이심률의 타원 궤도라면 중력과 동일한 근일점 회전을 나타낼 것이라는 생각으로 시작한 생각이었는데 재미있게도 전자기력에 의한 궤도의 회전은 중력에 의한 효과의 딱 절반이란 것이 최종 결론이다.

편향력의 상대론 적인 일관성을 살펴보려 한 것이 원래 의도였는데 그보다 훨신 오래된 문제인 전자기력 자체의 상대론 적인 일관성 문제를 마주하고 해결하였다. 그 과정에서 편향력 자체는 보다 근원적인 형태로 분해하여 유도 전기력과 상대론 적인 부수 효과로 인한 현상임을 밝혀

내었고 그로 인해 자연히 해결되었다. 그러나, 여전히 그 형태의 현상은 위그너 회전에 의한 힘의 꼴에 남아있다. 그 문제를 다루어 볼 것이다.

6.5 위그너 편향력

앞서 맥스웰 중력의 장에서 나는 위그너 회전에 의한 힘을 장에 의한 힘에 통합하는 도박을 했었다. 나의 그러한 도박의 근거를 다시 요약하자면 위그너 회전의 영향이 가속도의 원인과 상관없이 가속도 자체에 대하여 적용된다면 광속에 극히 가까운 소립자들의 질량을 자기장 내에서의 회전 반경으로 측정하는 현재의 방법에서 큰 문제가 생겼을 테니 이미 발견 되고도 남았을 것이라는 추측이었다. 고속에서 자기장으로 측정 되는 질량이 절반으로 줄어드는 이상 현상이 있는데도 발견되지 못했을 가능성은 없다. 물리학이 쇠퇴했다고 나는 생각 하지만 그 정도는 아니라고 본다. 쇠퇴는 주로 이론 물리학에 치중되어있고 실험 물리학의 기술이 쇠퇴했을 리는 없다. 다른 가능성으로는 위그너 회전은 편향력으로 표현되지 않는다는 생각도 가능하다. 그러나, 그 생각은 무책임하다. 회전의 물리 단위는 1/시간 이고 거기에 속도가 작용하면 거리/시간제곱 으로서 가속도의 단위이다. 가속도의 물리 단위로 계산되고 반드시 존재할 수 밖에 없는 어떤 물리량은 이보다 더 명확한 다른 어떤 이유가 없는 한 가속도로 해석하는 것이 내가 아는 물리학의 논리 체계이다. 설혹 나중에 어떤 예상치 못한 이유가 발견되어 수정하는 한이 있더라도 보다 지금 현재는 있는 그대로 받아 들이는 것이 옳다. 이전에 내가 편향력으로 판단 했다거나 실은 유도 가속과 기타 상대론 적인 부수 효과의 합으로 판명된 현상을 돌이켜보자. 편향력으로 판단했던 것은 경솔한 오류였을까? 나는 그렇게 생각하지 않는다. 변경된 것은 그 근본 원인에 대한 해석일 뿐 편향력이라는 현상 자체는 부정된 것은 아니다. 타워 궤도 튜브의 비유에 따르면 지금도 구슬이 튜브 벽에 가하는 힘은 편향력으로 계산했던 바로 그 힘일 것이다. 실은 나는 계산할 필요를 느낄 수 없었지만 관련 식들의 형태를 보건데 장담 할만하다. 현상에 대한 묘사로서는 전혀 틀리지 않았다. 위그너 편향 역시 그와 비슷하리라 본다. 결국 그렇다면 마지막 가능성은, 위그너 편향 역시 유도 가속과 비슷한 방법으로 존재하고 있으며, 그 현상은 가속도라는 다른 현상의 결과에 연결된 것이 아니라 그 원인인 장의 표현에 직접 결합되어 있다고 볼 수 밖에 없다는 생각이다. 장에 직접 결합 되어 있다면 알려진 모든 힘에서도 마찬가지 여야만 할 것은 뻔하다. 그런 표현으로서 나는 $\frac{1}{c^2}\frac{\gamma}{\gamma+1}\vec{v}\times\vec{E}\times\vec{v}$ 꼴의 편향을 제시 하고자 한다. 물론 수성의 궤도 전진이라는 현상이 이 위그너 편향이 반드시 존재 해야만 할 이유이고, 궤도의 전진이 만약 실제의 2/3 이었다면 포기할 만한 생각이기는 하다.