Numerical Investigation of Rotational Resonance of Mercury
A Case-study of Scheme Simulation

By

Victor Afigbo

May 2021

University of Idaho, Moscow Campus

A project submitted in partial fulfillment for the requirement of Numerical Methods

Abstract

In this numerical project, we are going to write a sample program for the spin-orbit problem for Mercury, and numerically investigate this resonance dynamics. We are going to show that the resonant condition for Mercury satisfies the equation given below:

$$Df(t) = n*(1 - e^2)^{1/2}\left(\frac{a}{R(t)}\right)^2 [1]; \text{ where } \frac{a}{R(t)} = \frac{(1 + e*\cos(f(t)))}{(1-e^2)}$$

Introduction

Basically, Mercury seems to be the smallest planetary body in our solar system, as well as the closest to our Sun [2]. The Sun appears at least three times as large when viewed from the surface of Mercury, compared to the Earth[2]. This is also applicable to the brightness of the Sun, which is also seven times as bright, going by initial analysis [2].

The planet, Mercury, rotates in a unique manner when compared to other planets in the Solar System[3]. It is locked with the Sun in a 3:2 spin–orbit resonance. This spin-orbit resonance is relative to the fixed stars, and means that, it rotates axially exactly three times for every two revolutions it makes around the Sun [4].

As seen from the Sun, in a frame of reference that rotates with the orbital motion, it appears to rotate only once every two years, Mercurian year to be precise. An observer on Mercury would have the luxury of seeing just a day, every two Mercurian years [5].

Mercury's orbit is the most eccentric orbit in the Solar System. Its orbital eccentricity is 0.21 and its distance from the sun falls within the range from 46-70 million kilometers. It only takes 88 days for Mercury to orbit around the Sun at 47.8 km/sec (29.7 miles/sec). A typical year on Mercury is equivalent to 88 days om planet Earth [6]. Owing to its changing distance from the Sun, and its 3:2 spin-orbit resonance, Mercury's temperature has strong variation followed by a complex surface structure [6].

Sequel to its discovery in the '60s, that Mercury has a rotation period that is precisely 2/3 times its orbital period, it is obvious that this resonant behavior in the spin-orbit model problem plays a vital role in the description of the extrinsic properties of planet Mercury. We are going to numerically perform slight perturbations on some collected data for Mercury and see how far off the rotation rate can be and still be trapped in this spin-orbit resonance. If the mismatch in angular velocity is too great, Mercury's rotation is no longer resonantly locked to its orbit[1].

Programming Component

Here, we are going to implement the codes for the resonant phenomenon of Mercury's orbit, using Scheme, a primarily functional programming language[7]. This will be done in stages as the codes are bulky. Please see below for the codes. All mathematical expressions were taken from [1], chapter 2.

```
1
2   ;[8]
3   ; Set epsilon = 0.026 and e = 0.2.
4
5   ; (a.) We write a program for the spin-orbit problem so this resonance dynamics can be
6   ; investigated numerically. We will show that f satisfies
7   ; the equation:
8   ;       Df(t) = n*(1 - e^2)^(1/2) (a / R(t))^2   [1]
9   ; where
10  ;       a / R(t) = (1 + e*cos(f(t))) / (1-e^2).   [1]
11
12  ;; Step 1: Calculate the mean anomaly, M=n(t-tau)
13  (define ((M n tau) state)
14      (let ((t (time state)))
15              (* n (- t tau))))
16
17  ;; Step 2: Numerically solve Kepler's Equation, M=E-esinE, for E
18  (define ((Keplers_Equation e n tau t) E)
19      (let ((thisM ((M n tau) (up t))))
20          (- (- E thisM) (* e (sin E)))))
21
22  ((Keplers_Equation 0.5 0.001 0. -1000.) -1.)
23
24  (define ((Eanomaly e n tau) state)
25      (let ((t (time state)))
26          (let ((thisM ((M n tau) state)) )
27              (let ((offset (if (< thisM 0.) :pi 0.)))
28                  (ref
29                      (minimize (square (Keplers_Equation e n tau t))
30                          (- thisM :pi) (+ thisM :pi))
31                      0)))))
32
33  ((Eanomaly 0.5 0.001 0.) (up -20000.))
34
35  ;; Step 3: Using E to find r & f
36  (define ((f_Kepler e n tau) state)
37      (let ((E ((Eanomaly e n tau) state)))
38          (let ((flip (if (< ((principal-value :pi) E) 0.) -1. 1.)))
39              (* flip (acos (/ (- (cos E) e)
40                      (- 1. (* e (cos E))))))))))
41
42  ((f_Kepler 0.5 0.001 0.) (up 1000.))
43
44  (define ((r_Kepler e n tau a) state)
45      (let ((f ((f_Kepler e n tau) state)))
46          (/ (* a(- 1 (square e)))
47              (+ 1 (* e (cos f))))))
48
49  (define win2 (frame -2. 2. -2. 2.))
50  (graphics-operation win2 'resize-window 1000 1000)
51  (graphics-clear win2)
52
53  (define ((xy_Kepler e n tau a) t)
54      (let ((r ((r_Kepler e n tau a) (up t)))
55          (f ((f_Kepler e n tau) (up t))) )
56          (cons (* r (cos f))
57              (* r (sin f))) ))
58
59  ((xy_Kepler 0.5 0.001 0. 1.) 1000.)
60  (plot-parametric win2 (xy_Kepler 0.5 0.001 0. 1.) 0. 16280. 1.)
61
62  ; Define our constants for Mercury and the Moon
63  ; Mercury constants taken from:
64  ;       https://solarsystem.nasa.gov/planets/mercury/by-the-numbers/[8]
65  (define s_per_day 86400.)
66  (define M_Mercury 3.301e23)             ; kg
67  (define R_Mercury 2440000.)             ; Mean radius in meters
68  (define P_Mercury (* s_per_day 87.97))  ; seconds per cycle
69  (define n_Mercury (/ :2pi P_Mercury))   ; seconds per radian
```

```
(define a_Mercury 57909000000.)         ; meters
(define epsilon_Mercury 0.026)
(define e_Mercury 0.2)

(define M_Moon 7.348e22)            ; kg
(define R_Moon 1737100.)            ; Mean radius in meters
(define P_Moon (* s_per_day 27.3))  ; seconds per cycle
(define n_Moon (/ :2pi P_Moon))     ; seconds per radian
(define a_Moon 384000000.)          ; meters
(define epsilon_Moon 0.026)
(define e_Moon 0.0549)

(define win2 (frame 0. (* 50. P_Mercury) (* 2 :-pi) (* 2 :pi)))
(graphics-operation win2 'resize-window 1000 1000)

(define win3 (frame 0. (* 50. P_Moon) -1 1))
(graphics-operation win3 'resize-window 1000 1000)

; Solve Eq. 2.135
(define ((a_over_r e f) state)
    (/ (+ 1 (* e (cos f)))
        (- 1 (square e))))

; Solve Eq. 2.134
(define ((f_dot e n) state)
    (let ((f (ref (coordinate state) 1)))
        (* n (sqrt (- 1. (square e))) (square ((a_over_r e f) state)))))

(show-expression ((f_dot 'e 'n)
    (up 't
        (up 'theta 'f_state)
        (up 'thetadot 'f_statedot))))

; b. Show that the 3:2 resonance is stable by numerically integrating the system when
; the rotation is not exactly in resonance and observing that the angle theta - (3/2)nt
; oscillates.

; Eq. 2.109, the averaged equation for motion in 1:1 resonance
(define ((thetadotdot_one_one_resonance n a epsilon e) state)
    (let ((t (time state))
        (theta (ref (coordinate state) 0))
        (thetadot (ref (velocity state) 0))
        (f (ref (coordinate state) 1)))
        (* -1 1/2 (square n) (square epsilon)
            (sin (* 2 (- theta f))) (cube ((a_over_r e f) state))  )))

; Eq. 2.118, the averaged equation for motion in 3:2 resonance
(define ((thetadotdot_three_two_resonance n a epsilon e) state)
    (let ((t (time state))
        (theta (ref (coordinate state) 0))
        (thetadot (ref (velocity state) 0))
        (f (ref (coordinate state) 1)))
        (* -1 1/2 (square n) (square epsilon) (/ (* 7 e) 2)
            (sin (- (* 2 theta) (* 3 f))))))

; Run the numbers on the averaged equation for motion for the moon
((thetadotdot_one_one_resonance n_Moon a_Moon epsilon_Moon e_Moon)
    (up 0. (up 0.1 0.) (up (* 1.01 n_Moon) n_Moon)))

; Run the numbers on the averaged equation for motion for the Mercury
((thetadotdot_three_two_resonance n_Mercury a_Mercury epsilon_Mercury e_Mercury)
    (up 0. (up 0.1 0.) (up (* 1.01 n_Mercury) n_Mercury)))

; Run the numbers on the averaged equation for motion on the moon
(define ((theta-state-derivative n a epsilon e) state)
    (up 1.
        ; replace P_Moon with P_Moon if trying to fix aritherror
        (up
            (* 1. (ref (velocity state) 0))
```

```
            (* 1. ((f_dot e n) state)))
        (up
            (* (square 1.) ((thetadotdot_one_one_resonance n a epsilon e) state))
            0.0)))

; Run the numbers on the averaged equation for motion on Mercury
(define ((theta-three-two-state-derivative n a epsilon e) state)
    (up 1.
        ; replace P_Mercury with P_Mercury if trying to fix aritherror
        (up
            (* 1. (ref (velocity state) 0))
            (* 1. ((f_dot e n) state)))
        (up
            (* (square 1.) ((thetadotdot_three_two_resonance n a epsilon e) state))
            0.0)))

((theta-state-derivative n_Moon a_Moon epsilon_Moon e_Moon)
    (up 0. (up 0. 0.1) (up (* 1.01 n_Moon) n_Moon) ))

((theta-three-two-state-derivative n_Mercury a_Mercury epsilon_Mercury e_Mercury)
    (up 0. (up 0. 0.1) (up (* 1.01 n_Mercury) n_Mercury) ))

; Show subsolar for the 1:1 resonance
(define ((show-subsolar win) state)
    (let (
        (t (time state))
        (theta (ref (coordinate state) 0))
        (f (ref (coordinate state) 1)))
        (let ((phi (- theta f)))
            (graphics-operation win2 'set-foreground-color "white")
            (plot-point win t ((principal-value :pi) phi))))))

; Show subsolar for the 3:2 resonance
(define ((show-subsolar-three-two win) state)
    (let (
        (t (time state))
        (theta (ref (coordinate state) 0))
        (f (ref (coordinate state) 1)))
        (let ((phi (- (* 2 theta) (* 3 f))))
            (graphics-operation win2 'set-foreground-color "white")
            (plot-point win t ((principal-value :pi) phi))))))

;; need to run this separately; can't just cut-and-paste or the window doesn't come out
right
(graphics-clear win2)
(graphics-clear win3)

((evolve theta-state-derivative n_Moon a_Moon epsilon_Moon e_Moon)
    (up 0.
        (up 0.0 0.0)
        (up (* 1.01 n_Moon) 0.))
    (show-subsolar win3)
    (* 0.1 s_per_day)
    (* 50. P_Moon)
    1.e-7)

((evolve theta-three-two-state-derivative n_Mercury a_Mercury epsilon_Mercury e_Mercury)
    (up 0.
        (up 0.0 0.0)
        (up (* 1.01 n_Mercury) 0.))
    (show-subsolar-three-two win2)
    (* 0.1 s_per_day)
    (* 50. P_Mercury)
    1.e-7)
;;[8]
```

Analysis

Implementing the codes above, we arrive at the results below.

a) Using the lines of codes for a:

$$\frac{e^2 n(\cos(f_{state}))^2\sqrt{1.-e^2}+2en\cos(f_{state})\sqrt{1.-e^2}+n\sqrt{1.-e^2}}{1+e^4-2e^2}$$

This satisfies the expected relationships of f for equations 2.134 and 2.135 [1]. This equation shows that Mercury's rotation is locked in resonance to its orbit [9][10].

b) Using the lines of codes for b:

Our $\ddot{\theta}$ resonance for Mercury (see lines 128 – 134 of the code) was calculated to be:

$$\ddot{\theta} = -3.2122\text{e}-17$$

This was on the same order of magnitude as the value of -5.64e-16, calculated in the example of the moon.

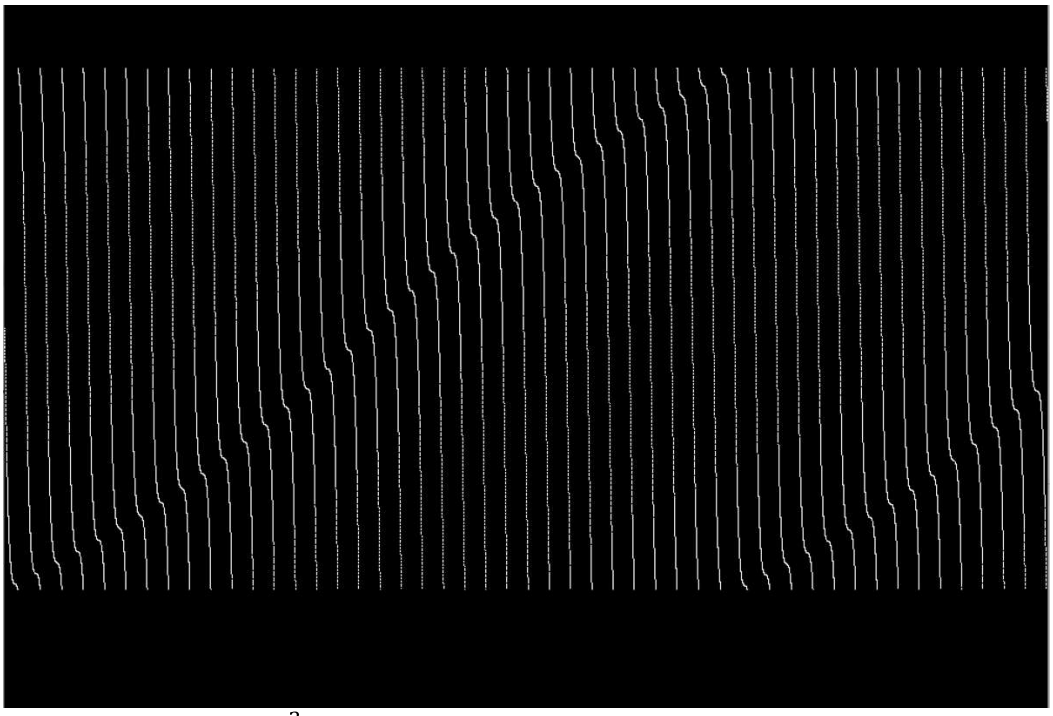
Our state derivative for Mercury was:

$$\begin{pmatrix} 1. \\ \begin{pmatrix} 8.349354630907662e-7 \\ 1.2634662750138744e-6 \end{pmatrix} \\ \begin{pmatrix} 4.778207326997884e-17 \\ 0. \end{pmatrix} \end{pmatrix}$$

This, again, seemed to be on the same order of magnitude as the example of the moon:

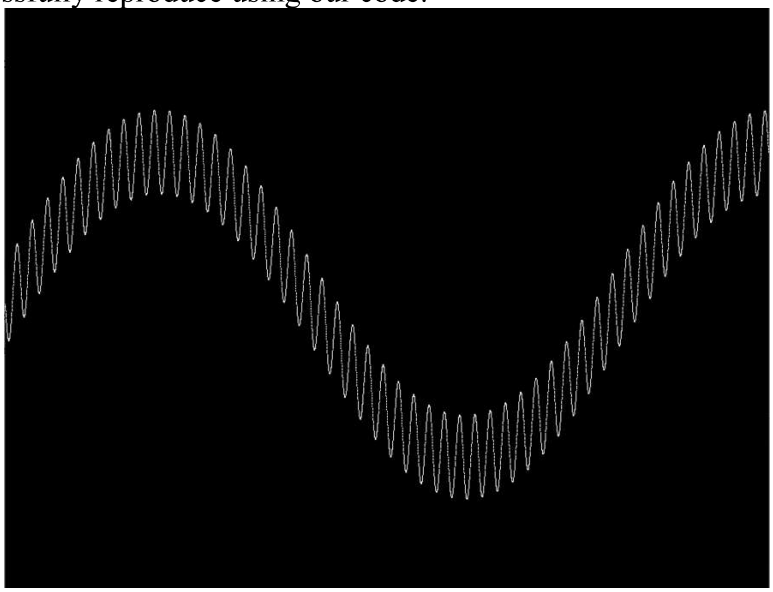
$$\begin{pmatrix} 1. \\ \begin{pmatrix} 2.6904495490144578e-6 \\ 2.9762308906165214e-6 \end{pmatrix} \\ \begin{pmatrix} 5.64004289385849e-16 \\ 0. \end{pmatrix} \end{pmatrix}$$

The plot generated to show the orbital characteristics of Mercury is below.



The plot of angle $\theta - \frac{3}{2}f$ for Mercury for 50 orbital periods on a scale of -2π to 2π radians. The x-axis is orbital periods, y-axis is the angle $\theta - \frac{3}{2}f$ in radians[9][10].

We can see the beginning of an oscillating ripple like the oscillation-of-oscillations we see in the moon example, which we were able to successfully reproduce using our code:



We were unable to find out why our graph failed to create true oscillations and instead generated discontinuous behavior. A problem in the show-subsolar-three-two function seems likely given that our intermediate steps all seemed reasonable and within expected bounds, but we were unable to successfully debug that function, owing to some difficulty in compatibility between the software and our personal computer [9][10].

Summary

Based on the achieved results, we can say that the 3:2 resonance for Mercury exists, and it is stable. Its stability is achieved by numerically integrating of the oscillating planetary system when the rotation is not exactly in resonance.

References

[1] Sussman, G.J.; Wisdom, J. “Structure and interpretation of classical mechanics (Second ed.).” (2015).

[2]NASA Solar System Exploration Overview | Mercury – NASA Solar System Exploration

[3] Elkins-Tanton, Linda T. ”Uranus, Neptune, Pluto, and the Outer Solar System”. Infobase Publishing. p. 51. ISBN 978-1-4381-0729-5. Extract of page 51. (2006).

[4] "Animated clip of orbit and rotation of Mercury". Sciencenetlinks.com.(2021).

[5] Wikipedia contributors, "Mercury (planet)," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Mercury_(planet)&oldid=1022721159 (accessed May 15, 2021).

[6] Planet Facts. “Orbit and Rotation of Mercury”. Orbit & Rotation of Mercury: Planet Mercury’s Year, Day & Revolution (planetfacts.org).(2021).

[7] Wikipedia contributors, "Scheme (programming language)," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Scheme_(programming_language)&oldid=1022628318 (accessed May 15, 2021).

[8] Barnes, J. “Analytical Mechanics”. Department of Physics, University of Idaho, Moscow campus. (2021).

[9] Trotter, K. “Analytical Mechanics”. Department of Physics, University of Idaho, Moscow campus. (2021).

[10] Isiaka, L. “Analytical Mechanics”. Department of Physics, University of Idaho, Moscow campus. (2021).