ps2_solutions

October 28, 2019

```
In [1]: from __future__ import division
        import sympy
        from sympy import *
        from sympy import init_printing
        from sympy.physics.vector import dynamicsymbols
        init_printing()
```

Problem 1. Mercury's precession. Consider a test particle orbiting in a potential $V(r)$ on a nearly circular orbit.

(a) Derive the Lagrange equations of motion for r and φ.

(b) The 'unperturbed' orbit is taken to be circular, with

$$r_0 = a \tag{1}$$
$$\varphi_0 = nt \tag{2}$$

Determine n in terms of V (and/or its derivatives).

(c) Linearize the Lagrange equations by writing $r = r_0 + r'$, where $r' = \mathrm{Re}(Re^{-i\omega t})$, and similarly $\varphi = \varphi_0 + \mathrm{Re}(\Phi e^{-i\omega t})$. (Note that R and Φ are complex constants.) Determine the frequency ω. What is the physical meaning of ω (i.e. what happens at that frequency), and what is its value for the Keplerian potential $V = -k/r$? [Make sure your answer agrees with what you know about Keplerian orbits.]

(d) From here on, we will assume a nearly Keplerian potential

$$V(r) = -\frac{k}{r} + \epsilon(r) \tag{3}$$

where $\epsilon(r)$ is a small correction. In that case, your solution above describes a precessing ellipse. Calculate the precession rate. [Make sure your answer agrees with what you know should happen for $\epsilon = \mathrm{const}/r$.]

(e) Because of general relativity, Mercury is not in a purely Keplerian potential, but has the following ϵ:

$$\epsilon_{\mathrm{GR}} = -\frac{k^2 a}{c^2 r^3}, \tag{4}$$

with $k \equiv GM$, a is considered to be constant when taking derivatives of this potential, and we drop terms $\mathcal{O}(e^2)$, consistent with our linearization. What is Mercury's precession rate due to GR? Your answer should be $\sim 40''/\text{century}$.

(f) Mercury also precesses due to the potential induced by the other planets. Jupiter's mass, averaged over its orbit, introduces the following ϵ:

$$\epsilon_\mathrm{J} = -\frac{GM_\mathrm{J}}{4a_\mathrm{J}^3}r^2, \tag{5}$$

where M_J and a_J are Jupiter's mass and semimajor axis; we assume that Jupiter's orbit is circular; and we keep the lowest term in the expansion of r/a_J (i.e. the quadrupole term). What is Mercury's precession due to the combined effect of Venus through Saturn? Your answer should be $\sim$ 10x larger than the GR value. However, it's around 20% off from the real answer, largely because Venus is so close that the quadrupole formula is inexact.

Solution. Part (a):

In [2]:
```
a = symbols("a",positive=True)
m, T, t, L, l = symbols('m T t L l')
r = dynamicsymbols("r")
rdot = diff(r,t)
rddot = diff(rdot,t)
r_symbol = symbols(r'r')
rdot_symbol = symbols(r'\dot{r}')
rddot_symbol = symbols(r'\ddot{r}')
t = symbols("t")
phi = dynamicsymbols(r'\varphi')
phidot = diff(phi,t)
phiddot = diff(phidot,t)
phi_symbol = symbols(r'\varphi')
phidot_symbol = symbols(r'\dot\varphi')
phiddot_symbol = symbols(r'\ddot\varphi')
V = dynamicsymbols(r'V').subs(t,r)
V_symbol = symbols(r"V")
Vprime = diff(V,r).evalf()
Vdprime = diff(Vprime,r).evalf()
Vprime_symbol = symbols(r"V'")
V_symbol = symbols(r'V')
alias = {phiddot: phiddot_symbol, rddot: rddot_symbol,
         phidot: phidot_symbol, rdot: rdot_symbol,
         phi: phi_symbol, r: r_symbol}
```

In [3]:
```
T = m*rdot**2/2+m*r**2*phidot**2/2
L = T - m*V
relational.Eq(symbols(r"L"),L.subs(alias))
```

Out[3]:

$$L = \frac{\dot\varphi^2 m}{2}r^2 + \frac{\dot r^2 m}{2} - mV(r)$$

```
In [4]: #radial equation of motion
        eom1 = diff(diff(L,rdot),t) - diff(L,r)
        relational.Eq(eom1.subs(alias)) #.subs(Vprime.evalf(),Vprime_symbol)
```

Out[4]:

$$\ddot{r}m - \dot{\varphi}^2 mr + m\frac{d}{dr}V(r) = 0$$

Setting $\ddot{r} = 0$, we obtain the cirular orbit frequency, $n \equiv \dot{\varphi}$ is:

```
In [5]: nsq_solution = solve(eom1.subs(rddot,0),phidot**2)[0]
        relational.Eq(symbols(r'n^2'),Subs(nsq_solution.subs(alias).subs(m*r_symbol,m*a),
                                           r_symbol,a))
```

Out[5]:

$$n^2 = \frac{1}{r}\frac{d}{dr}V(r)\bigg|_{r=a}$$

where the derivative is evaluated at $r = a$.

```
In [6]: #azimuthal equation of motion
        l_expression = diff(L,phidot)
        eom2 = diff(l_expression,t) - diff(L,phi)
        relational.Eq(eom2.subs(alias)) #.subs(Vprime.evalf(),Vprime_symbol)
```

Out[6]:

$$\ddot{\varphi}mr^2 + 2\dot{\varphi}\dot{r}mr = 0$$

This form of φ-equation of motion is less useful, however, than the conservation of angular momentum:

```
In [7]: relational.Eq(l,l_expression.subs(alias))
```

Out[7]:

$$l = \dot{\varphi}mr^2$$

which is a constant. Solving for $\dot{\varphi}$, we obtain:

```
In [8]: phidot_solution = solve(l_expression-l,phidot)[0]
        relational.Eq(symbols(r'\dot\varphi'),phidot_solution.subs(alias))
```

Out[8]:

$$\dot{\varphi} = \frac{l}{mr^2}$$

Now, plugging this into the radial equation of motion, we obtain:

```
In [9]: eom1_vs_l = eom1.subs(phidot,phidot_solution)
        relational.Eq(eom1_vs_l.subs(alias))
```

Out[9]:

$$\ddot{r}m - \frac{l^2}{mr^3} + m\frac{d}{dr}V(r) = 0$$

On a circular orbit, $\ddot{r} = 0$, therefore:

In [10]:
```
lsq_solution=solve(eom1_vs_l.subs(rddot,0),l**2)[0]
relational.Eq(l**2,Subs(lsq_solution.subs(alias).subs(m*r_symbol**3,m*a**3),
                        r_symbol,a))
```

Out[10]:

$$l^2 = a^3m^2\frac{d}{dr}V(r)\Big|_{r=a}$$

Thus, the frequency of radial oscillations is:

In [11]:
```
rddot_solution = solve(eom1_vs_l,rddot)[0]
omegasq_solution = -rddot_solution.diff(r).subs(l**2,lsq_solution)
relational.Eq(symbols(r"\omega^2"),
              Subs(omegasq_solution.subs(alias),r_symbol,a))
```

Out[11]:

$$\omega^2 = \frac{d^2}{dr^2}V(r) + \frac{3}{r}\frac{d}{dr}V(r)\Big|_{r=a}$$

In [12]:
```
relational.Eq(symbols(r"n^2")-symbols(r"\omega^2"),
              Subs((nsq_solution-omegasq_solution).subs(alias).subs(m*r_symbol,m*a),
                   r_symbol,a))
```

Out[12]:

$$-\omega^2 + n^2 = -\frac{d^2}{dr^2}V(r) - \frac{2}{r}\frac{d}{dr}V(r)\Big|_{r=a}$$

When $\epsilon = 0$, we obtain $\omega = n$, as is expected for a Keplerian orbit. For $\epsilon \neq 0$, we have $n^2 - \omega^2 \approx (n-\omega)2n_0$:

In [13]:
```
n0 = symbols(r"n_0")
relational.Eq(symbols(r"n")-symbols(r"\omega"),
              ((nsq_solution-omegasq_solution)/(2*n0)).\
              subs(Vdprime,symbols(r"\epsilon''")).\
              subs(Vprime,symbols(r"\epsilon'")).subs(alias))
```

Out[13]:

$$-\omega + n = \frac{1}{2n_0}\left(-\frac{2\epsilon'}{r} - \epsilon''\right)$$

In [14]:
```
pomegadotgr_symbol, epilongr, c, G, M, Tmerc \
    = symbols(r"\dot\varpi_{\rm{GR}} \epsilon_{\rm{GR}} c G M T_{\rm{merc}}",
              positive=True)
epsilongr = -(G*M)**2*a/(c**2*r**3)
#precession rate
pomegadotgr = ((nsq_solution-omegasq_solution)/(2*n0)).\
                    subs(V,epsilongr).doit().subs(r,a)
relational.Eq(pomegadotgr_symbol,pomegadotgr)
```

Out[14]:

$$\dot{\varpi}_{\mathrm{GR}} = \frac{3G^2M^2}{a^4c^2n_0}$$

In [15]:
```
pomegadotgr = (pomegadotgr).subs(sqrt(G*M/a**3),n0).subs(n0,2*pi/Tmerc).doit()
relational.Eq(pomegadotgr_symbol,pomegadotgr)
```

Out[15]:

$$\dot{\varpi}_{\mathrm{GR}} = \frac{6\pi GM}{T_{\mathrm{merc}}ac^2}$$

In [16]:
```
#operate in cgs
second = 1
day = 86400*second
year = 365*day
cm = 1
km = 1e5*cm
arcsecond = 2*pi/360/(60*60)
pomegadotgr_cgs = pomegadotgr.subs({Tmerc:87.969*day,
                                    G: 6.67259e-8,
                                    M: 2e33,
                                    a: (46.00+69.82)*1e6*km/2,
                                    c: 3e10})
pomegadotgr_cgs
```

Out[16]:

$$2.021325311521 \cdot 10^{-14}\pi$$

In [17]:
```
#converting to arcseconds per century
pomegadotgr_arcsec_per_century = pomegadotgr_cgs*100*year/arcsecond
relational.Eq(pomegadotgr_symbol,pomegadotgr_arcsec_per_century)
```

Out[17]:

$$\dot{\varpi}_{\mathrm{GR}} = 41.3064457356339$$

Of course, keep in mind that the result is very approxiate, so we can only trust that the true answer is somewhere around 40″, and the rest of the digits are not reliable.

```
In [18]: pomegadot_planet_symbol, epilon_planet, c, G, Mp, Tp, ap \
             = symbols(r"\dot\varpi_{\rm{p}} \epsilon_{\rm{p}} c G \
                         M_{\rm{p}} T_{\rm{p}} a_{\rm{p}}",
                     positive=True)
         epsilon_planet = -G*Mp*r**2/(4*ap**3)
         #precessoin rate
         pomegadot_planet = ((nsq_solution-omegasq_solution)/
                               (2*symbols(r"n_0"))).subs(V,epsilon_planet).doit()
         relational.Eq(pomegadot_planet_symbol,pomegadot_planet)
```

Out[18]:

$$\dot{\varpi}_{\rm p} = \frac{3GM_{\rm p}}{4a_{\rm p}^3 n_0}$$

Let's simplify this a bit by eliminating everything in favor of masses and periods:

```
In [19]: pomega_planet_final = pomegadot_planet.subs(G/ap**3,(2*pi/Tp)**2/M).\
                                                  subs(n0,2*pi/Tmerc)
         relational.Eq(pomegadot_planet_symbol,pomega_planet_final)
```

Out[19]:

$$\dot{\varpi}_{\rm p} = \frac{3\pi M_{\rm p} T_{\rm merc}}{2MT_{\rm p}^2}$$

where M is solar mass, M_p is the mass of the planet, T_p is the period of the planet orbit, and $T_{\rm merc}$ is the period of Mercury orbit.

```
In [20]: pomegadot_planet_func = lambdify([Mp,M,Tp,Tmerc], pomega_planet_final,'numpy')
```

```
In [21]: import numpy as np
         Planet_names_array = ["Venus", "Earth", "Mars", "Jupiter", "Saturn"]
         Mp_array = np.array([4.8675, 5.9724, 0.64171, 1898.19, 568.34])*1e27
         Msun = 2e33
         Tp_array = np.array([224.701, 365.256, 686.980, 4332.589, 10759.22]) * 86400
         Tmercury = 87.969*86400
         pomega_all_planets = pomegadot_planet_func(Mp_array,Msun,Tp_array,Tmercury).sum()
         (pomega_all_planets*100*year/arcsecond).evalf()
```

Out[21]:

$$387.876412347518$$

That is, the net precession rate due to planets from Venus through Saturn is around 390″ per century