Lec25-nonlinear-RyanSponzilli

November 21, 2024

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize
```

1 ASTR 310 Lecture 25 - nonlinear equations

1.0.1 Exercise 1: gas plus radiation pressure

A mixture of ionized hydrogen gas and radiation (such as at the center of the Sun) with a density ρ and temperature T has a pressure given by the following, where the first term is thermal (ideal gas) pressure and the second is radiation pressure.

$$P = \frac{2\rho kT}{m_p} + \frac{1}{3}aT^4.$$

Here $k = 1.38 \times 10^{-16}$ erg K^{-1}, $a = 7.56 \times 10^{-15}$ erg cm^{-3} K^{-4}, and $m_p = 1.67 \times 10^{-24}$ g.

Find the temperature corresponding to $\rho = 80$ g cm^{-3} and $P = 1.3 \times 10^{18}$ dyn cm^{-2} using Brent's method. You should get about 87×10^6 K.
Report what fraction of the total pressure is thermal (ideal gas) and what fraction is radiation pressure.

What are good starting guesses? A lower bound on the temperature is easy, but an upper bound requires more thought. You can find a strict upper limit by noting that since the temperature is always positive, the combined ideal gas and radiation pressure is always larger than either individual term alone.

(Aside: if you're wondering where the 2 came from, the ideal gas pressure has assumed pure H composition so that the mean mass per particle is $m_p/2$.)

[7 pts]

[17]:
```
def pressure(T):
    return (2*80*1.38e-16*T)/(1.67e-24) + (1/3)*7.56e-15*T**4 - 1.3e18

soln = scipy.optimize.brentq(pressure, 1e3, 1e8)
print(f"Root: {soln}")

P_therm = (2*80*1.38e-16*soln)/(1.67e-24)
P_rad = (1/3)*7.56e-15*soln**4
```

```
print(f"Thermal Fraction: {P_therm/(P_therm+P_rad)}")
print(f"Radiation Fraction: {P_rad/(P_rad+P_therm)}")
```

```
Root: 87269196.37557216
Thermal Fraction: 0.887565110996146
Radiation Fraction: 0.11243488900385401
```

1.0.2 Exercise 2: Moon missile

A projectile is launched from the surface of the Moon with a speed $v_0 = 10^5$ cm s^{-1} at an angle $\alpha = 45°$ from the surface. This is not enough to achieve orbit, so the projectile falls back to the Moon's surface. Ignoring the Moon's rotation, what is the angle θ_0 subtended by half of the trajectory?

Orbital parameters: (compute from given data)

$$E_0 = \frac{1}{2}v_0^2 - \frac{GM}{R}$$

$$a = -\frac{GM}{2E_0}$$

$$J_0 = Rv_0 \cos\alpha$$

$$e = \left(1 + \frac{2E_0 J_0^2}{G^2 M^2}\right)^{1/2}$$

Ellipse equation: solve with SciPy. Hint: $r = R$ at launch and impact, so θ is the only unknown.

$$\frac{r^2 \sin^2\theta}{a^2(1-e^2)} + \frac{(r\cos\theta - ae)^2}{a^2} = 1$$

Physical data: $ M = 7.348 ×10^{25} g; R = 1.737 ×10^8 cm; G = 6.673 ×10^{-8} cm^3 s^{-2} g^{-1} $

[6 pts]

[25]:
```
v0 = 1e5
alpha = 45
G = 6.673e-8
M = 7.348e25
R = 1.737e8

E0 = 0.5*v0**2 - (G * M / R)
a = -G*M/(2*E0)
J0 = R*v0*np.cos(np.pi/180*alpha)
e = (1 + (2*E0*J0**2)/(G**2*M**2))**0.5
```

```
def eq(theta):
    return (R**2 * np.sin(theta)**2)/(a**2*(1-e**2)) + ((R*np.cos(theta) -␣
 ↪a*e)**2)/a**2 - 1

soln = scipy.optimize.brentq(eq, 0, np.pi)
print(f"Angle subtended by half the trajectory: {soln*180/np.pi} degrees")
```

```
Angle subtended by half the trajectory: 12.147637163214025 degrees
```

1.0.3 Exercise 3: a nonlinear system of equations

Using `scipy.optimize.root`, solve the system of equations

$$9x^2 + 36y^2 + z^2 - 36 = 0$$
$$x^2 - 2y^2 - 20z = 0$$
$$x^2 - y^2 + z^2 = 0$$

for (x, y, z). Good starting guesses are $(\pm 1, \pm 1, 0)$. There are four roots. Try using an error tolerance of 10^{-10}.

Check carefully that you are actually getting four distinct roots – I had to tweak one of the starting guesses in order to pull out the fourth root. It might be helpful to plot the roots projected onto the (x,y) plane to verify that you're actually getting all four.

[7 pts]

[33]:
```
def f(vars):
    x, y, z = vars
    eq1 = 9*x**2 + 36*y**2 + z**2 - 36
    eq2 = x**2 - 2*y**2 - 20*z
    eq3 = x**2 - y**2 + z**2
    return eq1, eq2, eq3

soln1 = scipy.optimize.root(f, [-1, -1, 0], tol=1e-10)
print(soln1.x)

soln2 = scipy.optimize.root(f, [-1, 1, 0], tol=1e-10)
print(soln2.x)

soln3 = scipy.optimize.root(f, [1, -1, 0], tol=1e-10)
print(soln3.x)

soln4 = scipy.optimize.root(f, [1, 1, 0], tol=1e-10)
print(soln4.x)
```

```
[-0.89368798 -0.89458694 -0.04009467]
[-0.89368798  0.89458694 -0.04009467]
[ 0.89368798 -0.89458694 -0.04009467]
[ 0.89368798  0.89458694 -0.04009467]
```