Computational Physics: PS 7

Marcus Hoskins

October 30, 2024

1 Discussion

My GitHub repo is `https://github.com/marcusHoskinsNYU/phys-ga2000`. For images that are blurry here, please see the ps-7 folder for individual image files.

1.1 Problem 1

1.1.1 Part (a)

Suppose that there is a test mass m_T located at L_1 of the system give. Then, Newton's second law on this system reads:

$$\sum F = m_T a \quad \Rightarrow \quad \frac{GMm_T}{r^2} - \frac{Gmm_T}{(R-r)^2} = m_T a_c,$$

where we have assumed circular orbits, that Earth can be assumed motionless, thus implying the only acceleration here is centripetal. Moreover, we assume that the positive r direction is along the vector pointing from Earth to the moon. Then, since $a_c = \omega^2 r$, and here m_T being in the L_1 Lagrange means that it orbits along perfectly with the moon, ω is that of the Moon. Thus, to calculate this frequency we consider only the Earth-Moon system, which has a force equation of:

$$\sum F = ma \quad \Rightarrow \quad \frac{GMm}{R^2} = ma_c = m\omega^2 R,$$

thus implying that $\omega^2 = \frac{GM}{R^3}$. Therefore, the equation of motion for a test mass in the L_1 Lagrange point is:

$$\frac{GM}{r^2} - \frac{Gm}{(R-r)^2} = \frac{GM}{R^3} r,$$

where we have canceled the common factor of m_T. We can then rewrite this equation of $r' = \frac{r}{R}$ and $m' = \frac{m}{M}$, which gives:

$$\frac{1}{(r')^2} - \frac{m'}{(1-r')^2} = r'.$$

Finally, this can be written as a degree 5 polynomial:

$$(r')^5 - 2(r')^4 + (r')^3 + (m'-1)(r')^2 + 2r' - 1 = 0.$$

System	L1 Location (in m)
Moon-Earth	326318427.12358475
Earth-Sun	148504433590.2818
Jupiter-Mass Planet and Sun at Earth Distance	139998787386.94373

Table 1: L_1 Lagrange Points for various systems, in meters.

1.1.2 Part (b)

The code for solving this equation using Newton's method is given in the repo. The results are given in table 1

1.2 Problem 2

The code for this is given in the repo. For a tolerance of 10^{-8} in my Brent implementation, and with a starting bracket of (0, 0.5, 2), my Brent implementation gives a zero of 0.29999999958600876, and the scipy function gives a value of 0.2999999996737109. As a note, my Brent implementation finished after 20 loops, and did not need to resort to a golden search.