Phys 304: Homework 9

Mary Smith*
Haverford College Department of Physics
(Dated: April 6, 2024)

[Put abstract here]

1. PROBLEM 1: COMETARY ORBITS

Many comets travel in highly elongated orbits around the Sun. For much of their lives, they are far out in the solar system, moving very slowly, but on rare occasions their orbit brings them close to the Sun for a fly-by and for a brief period of time they move incredibly quickly.

The differential equation obeyed by a comet is straightforward to derive. The force between the Sun, with mass$M$ at the origin, and a comet of mass m with position vector $\mathbf{r}$ is GMm/r^2 in the direction $-\mathbf{r}/r$ (the direction towards the Sun), and hence Newton's second law means that

$$m\frac{d^2r}{dt^2} = -(\frac{GMm}{r^2})\frac{\mathbf{r}}{r}. \tag{1}$$

Canceling the m and taking the x component, this gives

$$\frac{d^2x}{dt^2} = -GM\frac{x}{r^3}, \tag{2}$$

and similarly for the y and z coordinates. However, if the axes are oriented such that a single plane is perpendicular to the z-axis, the z coordinate can be disregarded and there are just two second-order equations to solve:

$$\begin{aligned}\frac{d^2x}{dt^2} &= -GM\frac{x}{r^3}\\ \frac{d^2y}{dt^2} &= -GM\frac{y}{r^3},\end{aligned} \tag{3}$$

where $r = \sqrt{x^2+y^2}$. The pseudocode for Python programming for this problem can be found in Figure 1.

1.1. Part (a): Generating the first-order equations

The two second-order equations in Equation 3 can be simplified into two first-order equations that will be solved in the following sections. Setting dx/dt equal to v_x, and similarly dy/dt equal to v_y, the two equations in Equation 3 can be expressed as four first-order equations:

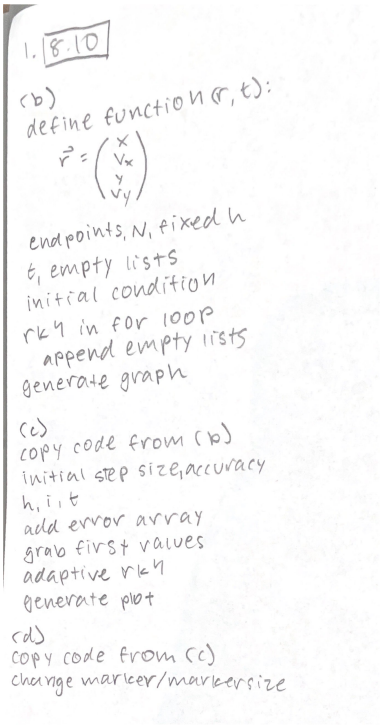


FIG. 1: [The pseudocode for Problem 1, cometary orbits.]

$$\begin{aligned}\frac{dx}{dt} &= v_x\\ \frac{dv_x}{dt} &= -GM\frac{x}{r^3}\\ \frac{dy}{dt} &= v_y\\ \frac{dv_y}{dt} &= -GM\frac{y}{r^3}.\end{aligned} \tag{4}$$

1.2. Part (b): Solving with a fixed step size

A Python program was written according to the pseudocode in Figure 1, using the fourth-order Runge-Kutta method with a fixed step size. As an initial condition, a comet was taken at coordinates x=4 billion kilometers and y=0, with initial velocity $v_x = 0$ and $v_y = 500ms^{-1}$.

In order to do this, a function was defined of $(\mathbf{r}, t$, where $\mathbf{r}$ is the vector of $(\frac{dx}{dt}, \frac{dv_x}{dt}, \frac{dy}{dt}, \frac{dv_y}{dt})$. A fixed step

*Electronic address: masmith@haverford.edu

size h was chosen in order to accurately calculate at least two full orbits of the comet. Then the fixed 4th order Runge-Kutta method was used to append to empty lists for the values of $\mathbf{r}$, given the initial conditions provided. The graph in Figure 2 was then generated. A fixed step size of $h = 31,557.6$ was used to create this graph. The ellipse doesn't perfectly overlap on the right side of the graph, because the fixed step size doesn't account for the speed of the comet increasing as it gets closer to the Sun. The calculation takes about 5 seconds to run.

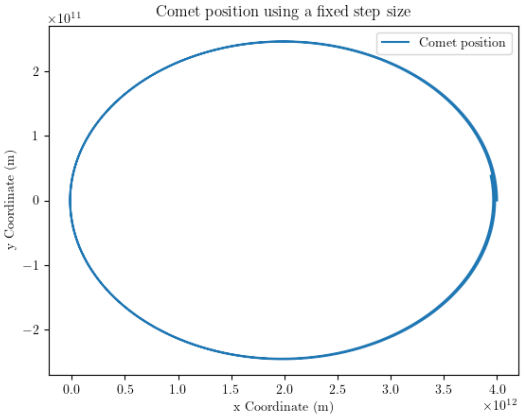
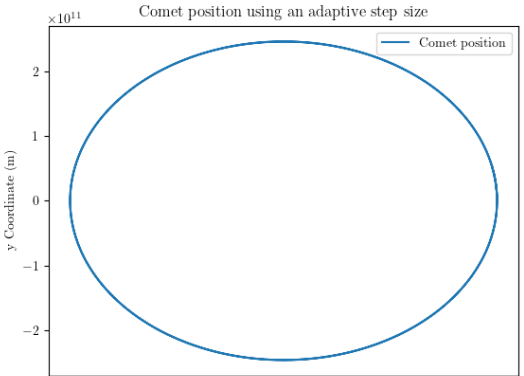


FIG. 2: [The position of a comet orbiting the Sun, approximated using the fixed 4th order Runge-Kutta method.]

1.3. Part (c): Solving with an adaptive step size

An adaptive step size method is extremely useful for cases like this, because for the large periods of time when the comet is moving slowly, long time-steps can be used so that the program runs quickly, but short time steps are crucial in the brief but fast-moving period close to the Sun. The Python program was then modified according to the pseudocode in Figure 1 in order to do the calculation with an adaptive step size. An accuracy of $\delta = 1$ kilometer per year in the position of the comet was set and the trajectory was once again plotted.

In order to do this, the first initial values for the comet were grabbed. Then for the times of interest to this problem, a while loop was created. Inside this while loop, the arrays for different steps were created and then one large step and two small steps were taken according to the adaptive Runge-Kutta 4th order. If the ρ value was sufficiently large, the $\mathbf{r}$ vector was set to be the small step, with an increase in t and h. Else, $\mathbf{r}$ was approximated for an h values based on ρ. This was used to generate the graph in Figure 3. The ellipse in this graph is perfectly aligned. The program runs almost instantly and is extremely accurate when compared to the plot in Figure 2.



1.4. Part (d): The position at each step

The Python program was then further modified according to the pseudocode in Figure 1. Adjusting the marker and the marker size, dots can be placed on the graph showing the position of the comet at each Runge-Kutta step around a single orbit. This can be found in Figure 4. The steps, represented by dots in the figure, get closer together when the comet is close to the Sun and further apart when it is far out in the solar system.

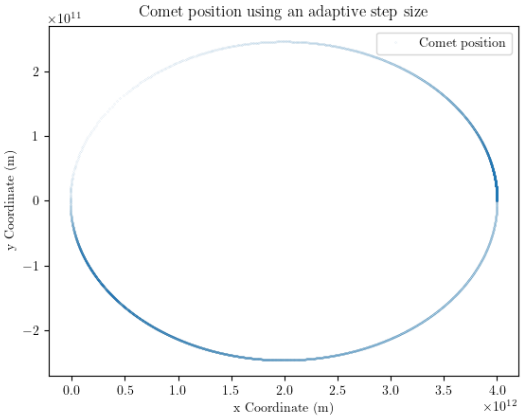


FIG. 4: [The position of a comet orbiting the Sun, showing the step size of the adaptive Runge-Kutta 4th order approximation of trajectory.]

2. PROBLEM 2: QUANTUM OSCILLATORS

Could not figure out the graph for this section in time. I am submitted what I currently have, then will submit the full problem set when I can figure out my graph.