Attached written notes are my pseudo code for the problems.

8.10

a)

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

$$\frac{d^2y}{dt^2} = \frac{dV_y}{dt} = -GM\frac{y}{r^3}$$

$$\frac{dx}{dt} = V_x$$
$$\frac{dV_x}{dt} = -GM\frac{x}{r^3}$$
$$\frac{dy}{dt} = V_y$$
$$\frac{dV_y}{dt} = -GM\frac{y}{r^3}$$

b) Use 4th order Runge-Kutta method

pseudo code:

```
import everything
G = 6.67 E-11
m = 1.989 E 30.
def f(r,t):
    x = r[0]
    Vx = r[1]
    y = r[2]
    Vy = r[3]
    fx = Vx
    fy = Vy
    fvx = - GM x/r^3
    fvy = - GM y/r^3
    return(     )

a = 0
b =
N =
h = (b-a)/N

tpoints = np.arange(a, b, h).
x_values = [ ]
y        = [ ]
Vx       = [ ]
Vy       = [ ]

r = np.array([4e12, 0, 0, 500], float)

for t in tpoints:
    x_values.append(r[0]).
    y           ...
    Vx          ...
    Vy          ...
    K1 =
    K2 =
```

$k_3 =$

$k_4 =$

$r + = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).$

plt. plot ( ).

c). We change the "else" statement in the "if" loop and add the r_1 and r_2 into the while loop.

```
while t < b:
    r1 = np.copy (r)
    r2 = np.copy (r)

  ...
    if :
  ...
    else:
        h = h · e^(1/4)
        r2 = np.copy(r).
        k1 = h x f (r2, t).
        k2 = h x f (r2 + 0.5 · k1 , t + 0.5h)
        k3 = h x f (r2 + 0.5 k2 , t + 0.5h)
        k4 = h x f (r2 + k3 , t+h)
        r2 + = (k1 + 2k2 + 2k3 + k4) / 6.

        k1 = h x f (r2 , t)
        k2 = h x f (r2 + 0.5k1 , t + 0.5h)
        k3 = h x f (r2 + 0.5k2 , t + 0.5h)
        k4 = h x f (r2 + k3 , t+h).
        r2 + = (k1 + 2k2 + 2k3 + k4) / 6

plt. plot ( ).
```

d). plt. scatter ( ).

8.14

a.
$$\hat{H}|\psi\rangle = E|\psi\rangle$$
$$i\hbar\frac{\partial}{\partial t}\psi(x) = E\psi(x)$$
$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V(x)\psi(x) = E\psi(x)$$
$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} = (E-V(x))\psi(x)$$
$$\frac{d^2\psi}{dx^2} = (V(x)-E)\cdot\frac{2m}{\hbar^2}\psi(x)$$

$$\frac{d\psi}{dx} = \phi$$
$$\frac{d\phi}{dx} = (V(x)-E)\cdot\frac{2m}{\hbar^2}\psi(x)$$

Pseudocode:

```
import all libraries
import all constants

def V(x):
    return V0 x^2 / a^2

def f(r, x, E):
    ψ = r[0]
    φ = r[1]
    fψ = φ
    fφ = (V(x)-E)·2m/ħ^2 ψ(x)
    return array([fψ, fφ], float)

def solve(E):
    ψ = 0
    φ = 1
    r = array([ψ, φ], float)

    for x in (-10a, 10a, h):
        k1 =
        k2 =
        k3 =
        k4 =
        r += (k1+2k2+2k3+k4)/6

    return r[0]
```

from squarewell.py

```
E = [ ]

for n in range(3): # set the loop for each energy state.
    E1 =        # initial guess
    E2 =        # initial guess
    ψ2 = Solve(E2)

    target = e/1000
    while (E1-E2) > target:
```

```
            ψ1, ψ2 = ψ2, solve(E2)
            E1, E2 = E2, E2 - ψ2 (E2-E1)/(ψ2-ψ1)
        E.append(E2)
    print(      )
```

b).

```
    def V(x):
        return V0 x^4/a^4
```

All the other parts are the same.

c).

```
for n in range(len(E)):
        E = E(n)
        ψ = 0
        φ = 1
        ψ_values = []
        φ_values = []
        r = array([ψ, φ])

    # inside the loop
    for x in arange(-5a, 5a, h):
        ψ_values.append(r[0])
        φ_values.append(r[1])
        k1 =
        k2 =
        k3 =
        k4 =
        r+ =
    # transform the list to an array to make calculation
      easier
```

trapezoid method

```
    t = 1/2 h ψ_values[0] + 1/2 h ψ_values[-1]
    for k in range(1, len(ψ_values)):
        t+ = ψ_values[k]·h

    Normalization = 1/t

    plt.plot( )
```

PHYS 304 AS5

Xiyue Shen*
Haverford College Department of Physics
(Dated: April 12, 2024)

I asked for an extension on this homework. Yang and Melanie helped me to figure out some problems. I have three python files.

$PHYS\_304\_AS9\_8.10a\&b\_Xiyue\_Shen$ and $PHYS\_304\_AS9\_8.10c\&d\_Xiyue\_Shen$ files are the comet trajectory problem, numbered as 8.10 in the textbook.

$PHYS\_304\_AS9\_8.14\_Xiyue\_Shen$ is the file for a quantum harmonic oscillator labeled as 8.14 in the textbook.

1. EXERCISE

1.1. Exercise 8.10: Cometary orbits

1.1.1. part a

For this part, we have two second-order differential equations. Following the trick that we have done for the simple harmonic oscillation, we have split the second-order differential equations into two first-order equations by introducing a new parameter. In this case, I introduced one parameter for each differential equation. Let $\frac{dx}{dt}$ equal to a new parameter. Then, the differential form of this new parameter is equal to the original equations. The form looks like this,

$$\frac{dx}{dt} = v_x \tag{1}$$
$$\frac{dv_x}{dt} = -GM\frac{x}{r^3} \tag{2}$$
$$\tag{3}$$

Similarly, for the y direction, we have,

$$\frac{dy}{dt} = v_y \tag{4}$$
$$\frac{dv_y}{dt} = -GM\frac{y}{r^3} \tag{5}$$
$$\tag{6}$$

,where $r = \sqrt{x^2 + y^2}$

*Electronic address: xshen2@brynmawr.edu; URL: Optionalhomepage

1.1.2. part b

For this part, we will code the four differential equations using the fourth-order Runge-Kutta method. It's very similar to what we have done in the simple harmonic oscillator problem. The difference here would be adding two more parameters in every array, such as the initial condition array. The function must be modified to include all the functions shown in part a by the pseudocode. One thing worth noticing is that we want to do more than two full cycles by correctly setting the time step. The time step I have found is $h = 10^4 s$ since we have a very high initial condition. The plot is shown in figure 1

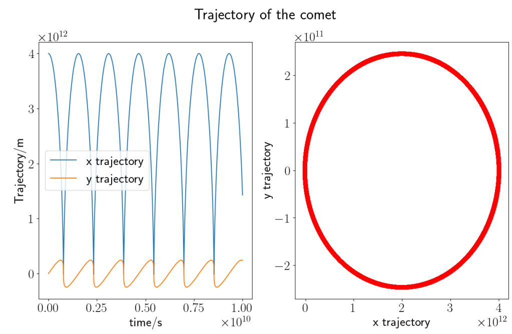

FIG. 1: [The trajectory of the comet with initial conditions $x = 4 \times 10^{12} m$, $y = 0$, $v_x = 0$, and $v_y = 500 ms^{-1}$]

In figure 1, we have two subplots. The left one has a time scale on the x-axis. I made this plot mainly to see how many cycles we go through. Also, we have to tell if the energy is conserved through this plot. Both x and y amplitudes don't decay with time, which indicates a conservation of energy. The right one is a $X - Y$ plot, which shows a circle, as expected by the problem.

1.1.3. part c and d

In this part, we are supposed to use the adaptive size to calculate the comet trajectory with target accuracy is δ = 1 km/day, which is 3.17×10^{-5} m/s. I followed the steps in "pendulum_adaptive.py" and adjust the else loop a bit, as shown in the pseudocode attached. The basic idea here is that we have an error value equal to $\sqrt{dx^2 + dy^2}$, where dx and dy are the values difference between the one big step and two small steps methods. Since we have two coordinates, I took the square root of the two combined errors. The new step size is,

$$h' = h(\frac{30h\delta}{error})^{1/4} = h\rho^{1/4} \quad (7)$$

,where $\rho = \frac{30h\delta}{error}$

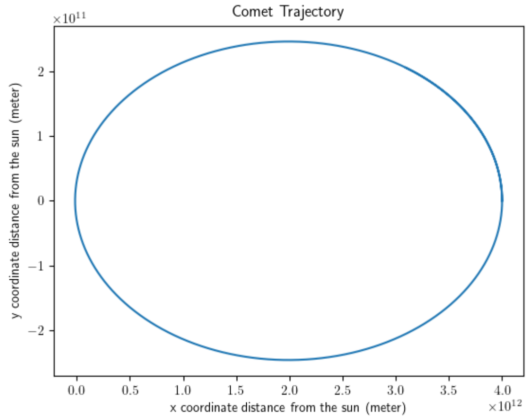

FIG. 2: [Comet trajectory with adaptive step size

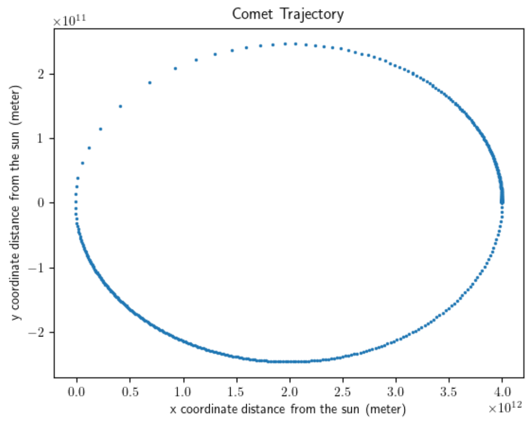

FIG. 3: [Comet trajectory with adaptive stepsize in scatter plot]

Figure 2 and figure 3 are what we have for part c and part d. In figure 3, I plotted every 20 point for a clear view. I expected the scatter plot to show a symmetric behavior, which means the points distribution on the top left and bottom right should exhibit the same density. One possible explanation here is that our initial conditions give a nonzero x coordinate value and a 0 y coordinate value. In this case, the error array dx and dy will have different estimates in the symmetric zone. So, this may break the expected symmetry.

1.2. Exercise 8.14: Quantum oscillators

1.2.1. part a

For this part, we are given a harmonic potential $V(x) = V_0 x^2/a^2$. The one-dimensional, time-independent Schroedinger equation is,

$$-\frac{\hbar^2}{2m}\frac{d^2\psi(x)}{dx^2} + V(x)\psi(x) = E\psi(x) \quad (8)$$

In this equation, $\hbar$ is the Planck constant over 2π, $\psi(x)$ is the wave function, m is the atom's mass, and E is the corresponding state energy. We can rewrite the equation to get the second-order differential equation on one side,

$$\frac{d^2\psi(x)}{dx^2} = \frac{2m}{\hbar}(V(x) - E)\psi(x) \quad (9)$$

To split it into two first-order differential equations, let $\frac{d\psi}{dx} = \frac{d\phi}{dx}$. Then we have,

$$\frac{d\psi}{dx} = \phi \quad (10)$$

$$\frac{d\phi}{dx} = \frac{2m}{\hbar}(V(x) - E)\psi \quad (11)$$

To form the code, I followed the *squarewell.py* example using the secant method. The pseudocode is shown in the previous attachment. In the main program, to solve the energy, I put them in a loop for each energy state. The result shows that the ground state energy is 138.03 eV, the first excited energy state is 414.07 eV, and the second excited state energy is 690.12 eV. To see if this result is correct, each state's energy difference is constant, around 276 eV.

1.2.2. part b

For this part, we are given a different potential, $V(x) = V_0\frac{x^4}{a^4}$. In the code, I changed the potential equation and the initial guess while keeping the rest the same as part a. The energy I got for the first three states are 205.31 eV, 735.69 eV, and 1443.57 eV. I tested this as the correct answer by plotting the wavefunction, which is part c.

1.2.3. part c

For this part, we are gonna solve the wavefunction and make a plot of the wavefunction. I set up a loop to run the solution for each energy state. Then, using the fourth-order Runge-Kutta method, we can solve the wavefunction and store the values. After this, a trapezoidal method is used to integrate for normalization. The normalization factor is $N = 1/\int|\psi|^2dx$. The plot is shown in figure 4

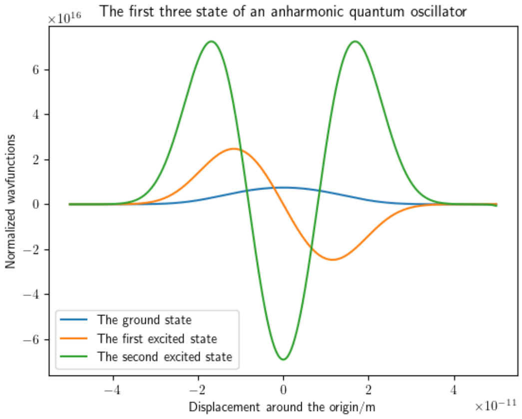

FIG. 4: [The first three wave states of the anharmonic quantum oscillator]

2. SURVEY

I spent roughly 8 hours on this homework. I'm now more familiar with adaptive stepsize differential equation problems. I like the comet trajectory problem most. It took me a very long time to figure out the break in the code. I practiced my troubleshooting skills. I think this set is about the right length.

3. UNGRADED PART

I have done all of the required and ungraded work.