Lab 07
Yinshi Liu and Landon Wang
Contribution:
Yinshi Liu did the entirety of Q1, Landon Wang did the entirety of Q2. This report is compiled by Yinshi Liu.
Q1a
Pseudocode (Adaptive portion)

1. Outline the initial conditions at t = 0 (h = 0, a = 0, b = 10, δ), setup empty arrays.
2. Calculate the time evolution term:
   Calculate $r(t+h)$ using the following method:

$$k_1 = hf(r,t) \tag{1.1}$$

$$k_2 = hf\left(r + \frac{1}{2}k_1, t + \frac{1}{2}h\right) \tag{1.2}$$

$$k_3 = hf\left(r + \frac{1}{2}k_2, t + \frac{1}{2}h\right) \tag{1.3}$$

$$k_4 = hf(r + k_3, t + h) \tag{1.4}$$

$$r(t+h) = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{1.5}$$

   which is the RK4 formulas from eqn. 8.33, Pg. 344.
   Repeat the process once more to obtain $r(t+h+h)$. Append the results to the respective arrays.
   Calculate $r(t+2h)$ using the RK4 method, and compute the error terms for x and y using the following formula:

$$\epsilon_x = \frac{1}{30}|x(t+2h) - x(t+h+h)| \tag{1.6}$$

$$\epsilon_y = \frac{1}{30}|y(t+2h) - y(t+h+h)| \tag{1.7}$$

   which corresponds to eqn. 8.54 in Newman.
   Calculate the Euclidean error ϵ and proportion ρ using:

$$\epsilon = \sqrt{\epsilon_x^2 + \epsilon_y^2} \tag{1.8}$$

$$\rho = \frac{h\delta}{\sqrt{\epsilon_x^2 + \epsilon_y^2}} \tag{1.9}$$

   Compare the proportion ρ to 1.
   If ρ is smaller than 1, then the accuracy of the result did not reach the target accuracy. Decrease the value of h using:

$$h' = h\rho^{\frac{1}{4}} \tag{1.10}$$

   Limit the change of step size by a maximum of a factor of 0.5. Hence, eqn. 1.10 becomes:

$$h' = h\min\left(\rho^{\frac{1}{4}}, 0.5\right) \tag{1.11}$$

   Discard the result of $r(t+h+h)$ and update the value of $r(t+h')$ using the RK4 method. Update time by h'.

If $\rho \geq 1$, then the calculated accuracy is better than target accuracy. Keep the result of $r(t+h+h)$, update h' using eqn. 1.10, and limit the change of h' to a factor of 2:

$$h' = h \min\left(\rho^{\frac{1}{4}}, 2\right) \tag{1.12}$$

update time by $2h$.
Iterate for all time values.

3. Plot the results.

Code Output
Figure 1.1

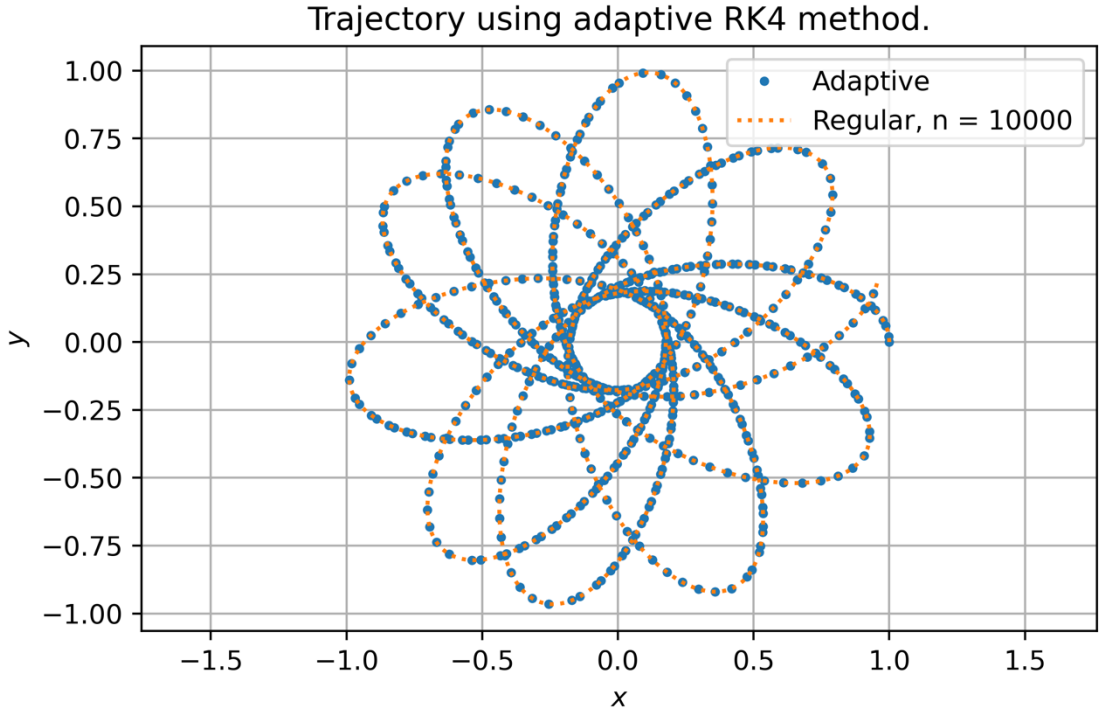


Figure 1.1 compares the trajectory using adaptive RK4 and regular RK4 with n = 10000. The trajectories are nearly identical to each other, which means that the adaptive method is as accurate as the regular approach with way less points (n = 691 vs. n = 10000). The effect of varying time step is demonstrated with the spacing of each points, where some areas are covered with more points than other areas. A further analysis on the distribution of time step sizes can be found in Q1c.

Q1b
Code Output

```
time_adaptive = 0.20391607284545898 s
time_regular = 0.7383291721343994 s
```

As expected, the adaptive method takes shorter time to compute compared to regular RK4.

Q1c
Code Output
Figure 1.2

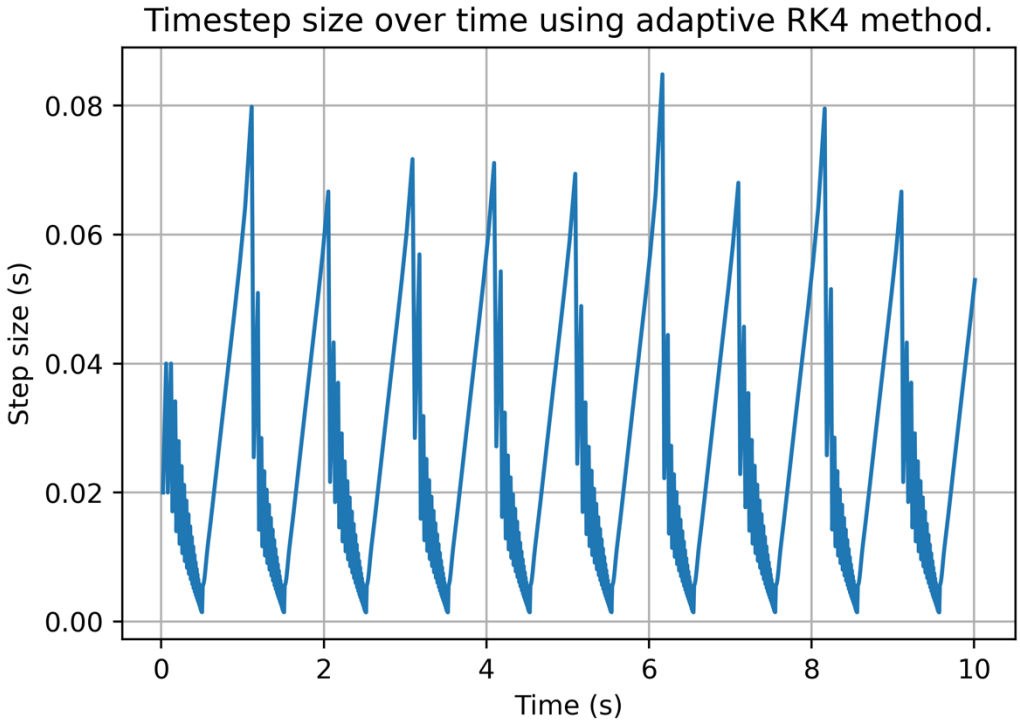
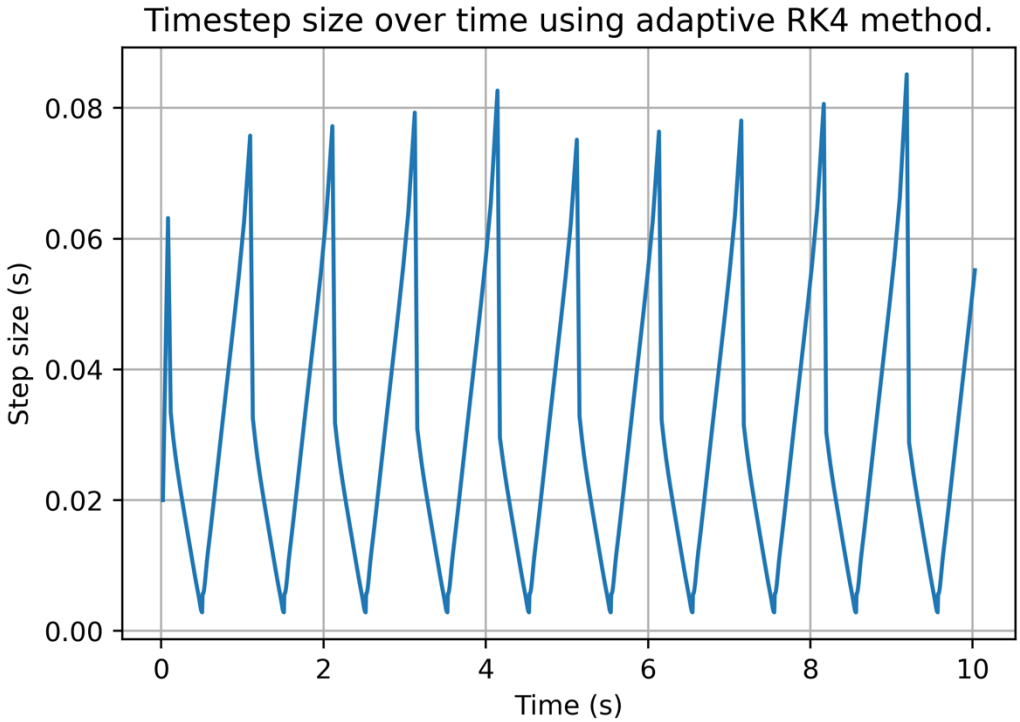


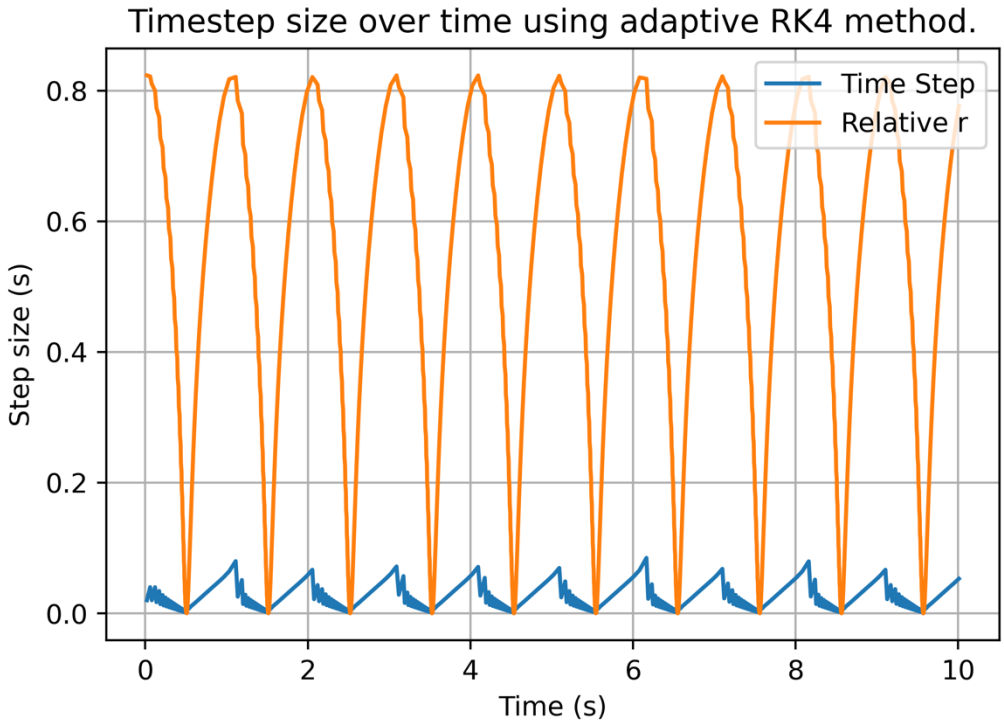
Figure 1.3 (Removed time step change limits)



As time evolves, the step size changes in a semi-regular manner. With the exception of the first few seconds, where the adaptive routine has yet to fully take effect, the size of time steps would decrease to a certain minimum (~0.001s) in an oscillation like manner, the increase at a steady rate to a local maximum (0.07s – 0.08s). The process would then repeat itself for each cycle,

with minor variation in values. The oscillatory decrease of time step sizes is due to the limit on time step changes we implemented in Q1a.
The effect of removing time step limits can be demonstrated in figure 1.3, where we observed that the oscillatory declines are gone. The initial few points also changed more erratically before resuming the pattern. The general shapes of both figures are similar.

Figure 1.4



As Figure 1.4 has shown, the change in time step sizes corresponds closely with the change of particle's position. As the particle gets closer to the center, it's speed would get faster, which means that the curve would change at a more rapid rate. At this point, the size of the time steps also gets shorter. As the particle moves away from the center, the speed would decrease, which represents the curve would change at a lower rate. Similarly, the size of the time step would increase. In other words, the size of the time step depends on the speed of the particle.

Q2

The time independent Schrodinger's equation states that

$$-\frac{\hbar}{2m}\nabla^2\psi(\mathbf{x}) + V(\vec{x})\psi(\vec{x}) = E\psi(\vec{x}) \tag{2.1}$$

For hydrogen atoms, the $\psi(\mathbf{x})$ follows

$$\psi(\mathbf{x}) = R(r)Y_l^m(\theta,\phi) \tag{2.2}$$

Where Y is the spherical harmonic which we don't really care.

The important thing to note is that $R(r)$ follows the second order ODE

$$\frac{d}{dr}\left(r^2\frac{dR}{dr}\right) - \frac{2mr^2}{\hbar^2}[V(r) - E]R = l(l+1)R \tag{2.3}$$

And for hydrogen atom,

$$V(r) = -\frac{e^2}{4\pi\epsilon_0 r} \tag{2.4}$$

a)

For this part, we are tasked to find R using the shooting method with RK4. As we mentioned earlier, the equation 2.3 is in second order, we need to separate into a set of two coupled ODEs

Let

$$\frac{dR}{dr} = S \tag{2.5}$$

We can now write equation 2.3 as

$$\frac{d}{dr}(r^2S) - \frac{2mr^2}{\hbar^2}[V(r) - E]R = l(l+1)R$$

$$\implies \quad \frac{dS}{dr} = \frac{R}{r^2}\left(l(l+1) + \frac{2mr^2}{\hbar^2}[V(r) - E]\right) - \frac{2S}{r} \tag{2.6}$$

Now, we modify the squarewell.py with the potential following eq. 2.4 and the Schrodinger's equation for our specific case following eq. 2.6

b)

Now, with a working function and working code, we start to find the impact on the initial conditions.

First, we have the initial condition suggested in the part a:

```
Initial Conditions: r_max = 20a, h = 0.002a, target = e/1000
E_1s = -13.212665021995116 eV
E_2s = -4.588819331853651 eV
E_2p = -3.2499903733645454 eV
Time taken for E_1s calculation: 1.0851304531097412
```

This is a reference for any further tuning. Notice that E_1s is fairly accurate to the theoretical value of -13.6eV, whereas E_2s and E_2p are not the same, suggested by the theory, while far apart from the theoretical value of -3.4eV.

Now, we change the r_max to increase 5 times, which gives a further distance in space, which provides a better boundary accuracy:

```
Initial Conditions: r_max = 100a, h = 0.002a, target = e/1000
E_1s = -19.945657904707303 eV
E_2s = -3.782316995733819 eV
E_2p = -3.4415697418139386 eV
```

```
Time taken for E_1s calculation: 10.816528081893921
```

We observed a significant increase in calculation time and we have the most acuate E_2p number we can get over any setting tested here. However, E_1s apparently is way off from the theoretical and E_2s did improved its accuracy but still has a long way to go. This is likely due to the fact that we set R(h) = 0 which impact the 1s state the most.

We are also interested on decrease the step size h:

```
Initial Conditions: r_max = 20a, h = 0.0004a, target = e/1000
E_1s = -13.214860257240693 eV
E_2s = -4.58946312760813 eV
E_2p = -3.249988156837012 eV
Time taken for E_1s calculation: 5.512851238250732
```

Noticed we also decreased the step size 5 times, but the effect on calculation time is significantly less compared to the changes on r_max. noticed the numbers output are very close to the original result, we can simply conclude the decrease in h is pretty much useless.

It should be expected that reduce the secant algorithm target size will not increase the time taken as we only need to calculate a few more terms to get a better result compared to a larger target. Anyway, here is the result:

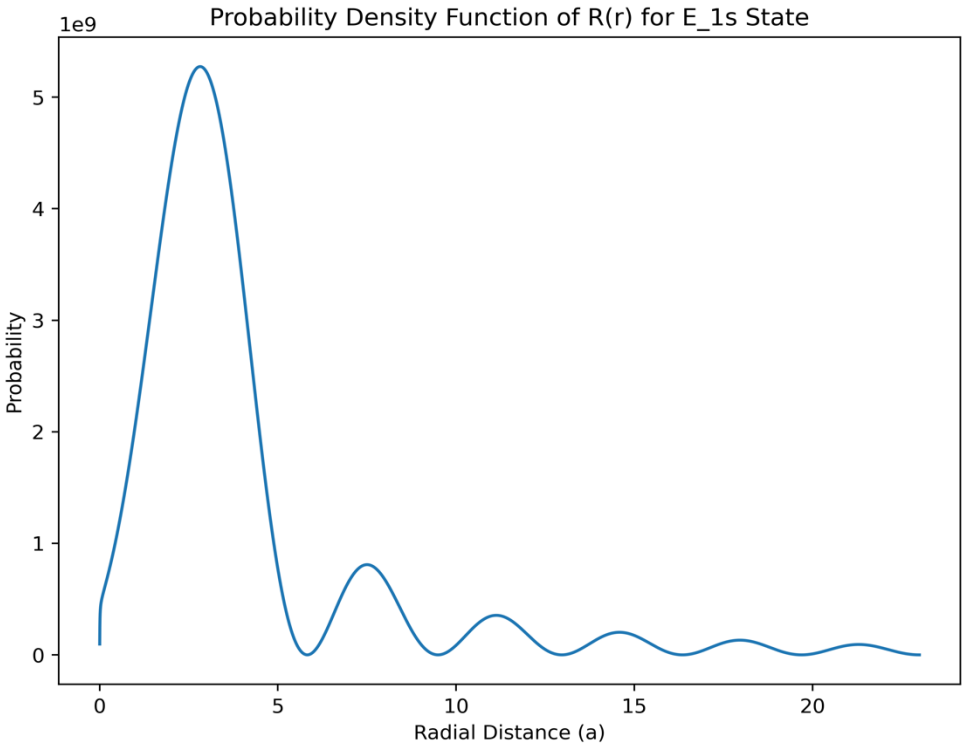
```
Initial Conditions: r_max = 20a, h = 0.002a, target = e/100000
E_1s = -13.212664864036508 eV
E_2s = -4.588819308363811 eV
E_2p = -3.2499903733645454 eV
Time taken for E_1s calculation: 1.2920348644256592
```

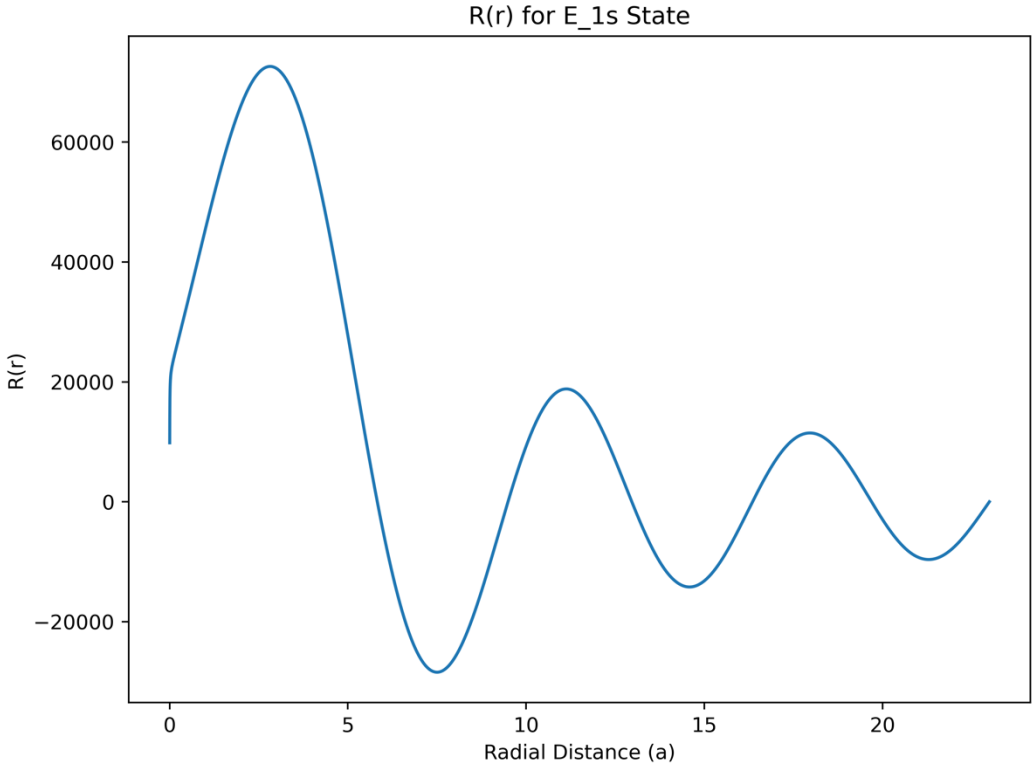
Similar situation like the decrease in h, decrease in target is fairly useless, it did not even increase the calculation time so much!

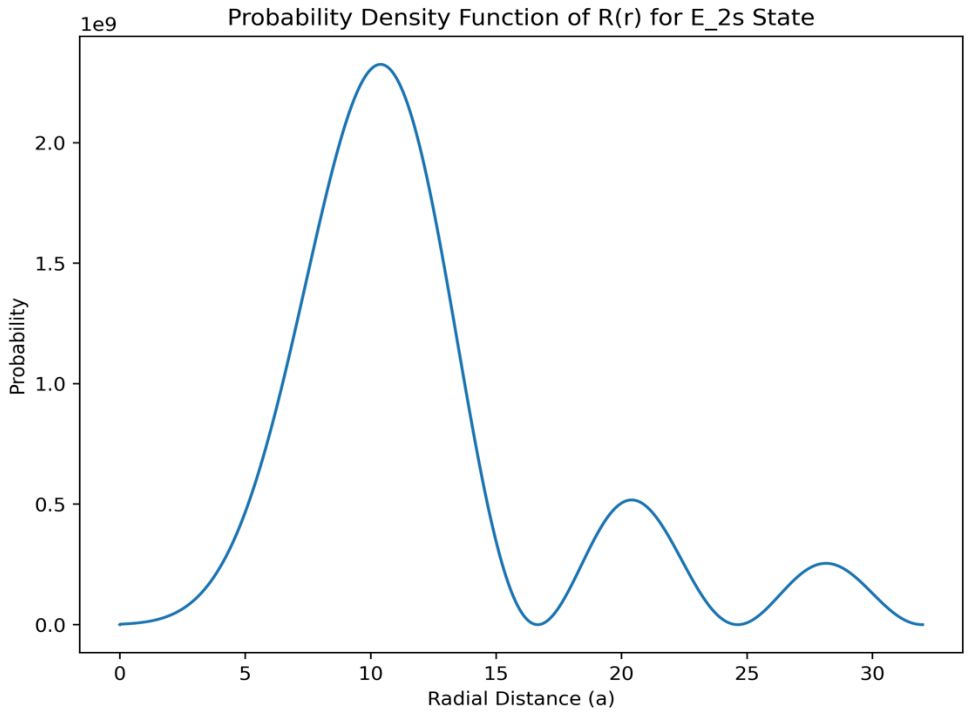
NOTE: code respect to this section is muted because the amount of time it takes to run everything.

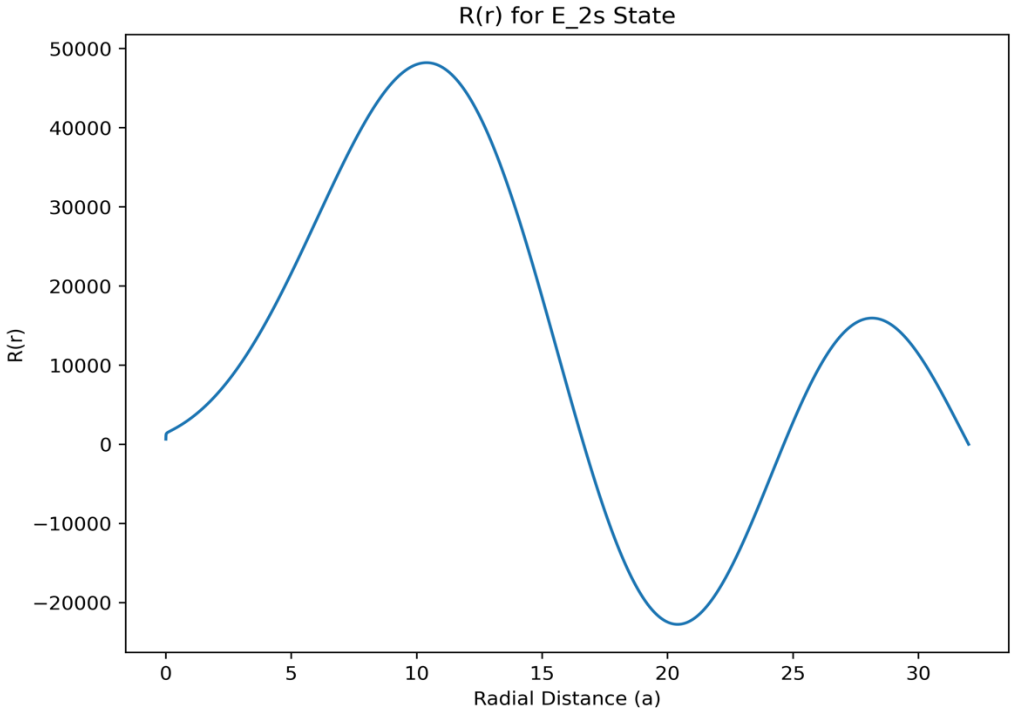
For further use, we tested and chose the r_max only and let the h and target unchanged. We noticed a cycle in the error of E when we increase r_max. We did not understand what is effecting the result, but we suspect it could be related to the secant algorithm or some strange behaviour of RK4.

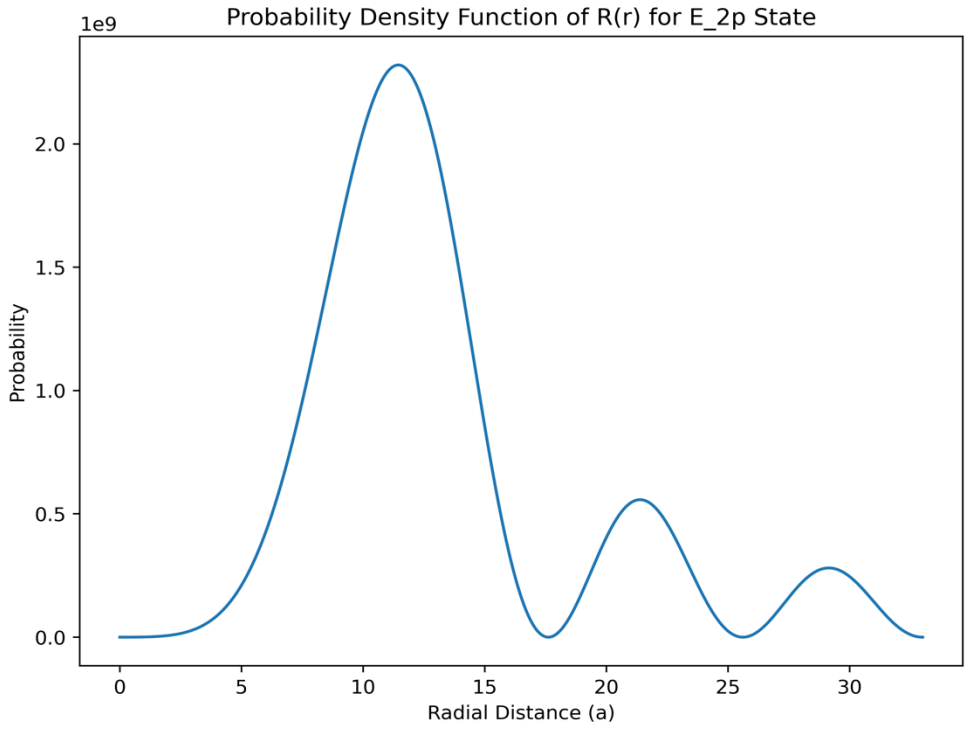
c)
FIGURE 2.1, 2.2, 2.3, 2.4, 2.5, 2.6

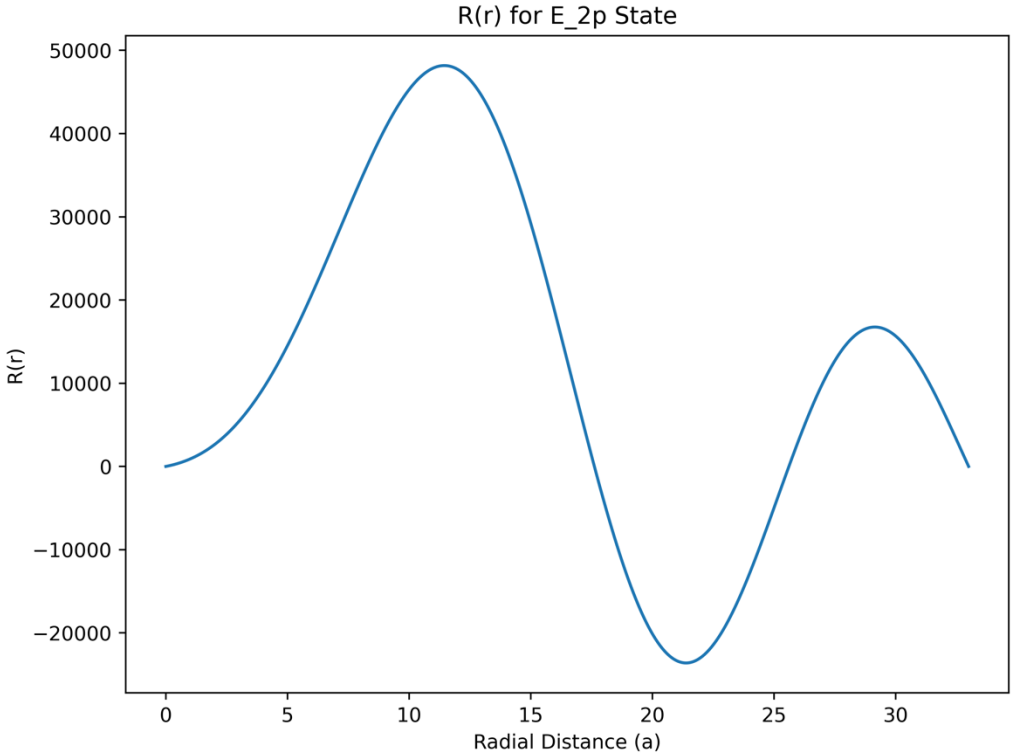

Probability Density Function of R(r) for E_1s State
1e9
Probability
Radial Distance (a)
0
1
2
3
4
5
0
5
10
15
20


R(r) for E_1s State
R(r)
Radial Distance (a)
60000
40000
20000
0
−20000
0
5
10
15
20


Probability Density Function of R(r) for E_2s State
1e9
2.0
1.5
1.0
0.5
0.0
Probability
0
5
10
15
20
25
30
Radial Distance (a)


R(r) for E_2s State
50000
40000
30000
20000
10000
0
−10000
−20000
R(r)
0
5
10
15
20
25
30
Radial Distance (a)


Probability Density Function of R(r) for E_2p State
1e9
2.0
1.5
1.0
0.5
0.0
Probability
0
5
10
15
20
25
30
Radial Distance (a)


R(r) for E_2p State
50000
40000
30000
20000
10000
0
−10000
−20000
R(r)
0
5
10
15
20
25
30
Radial Distance (a)

The three plots for three energy states showing the probability functions (fig.2.1, 2.3, 2.5), and three plots showing the eigenfunction R (fig. 2.2, 2.4, 2.6)

Compared to the theoretical values from the link in the lab manual, we notice that figure 2.1 resembles the theoretical state very well for the first peak. It is understandable for the smaller peaks at further distance because of the initial conditions we provided (ie. R(h)=0, S(h)=1) is very inaccurate.

The 2s and 2p states are indeed further away from the nucleus, which is expected. Notice that very minimal but exist shift between the 2s and 2p states. We have tested, it is not due to the small inconsistency of the eigenenergy we calculated previously. Therefore, we can conclude that l is the factor that made this difference. However, we noticed that 2p state is further away from the nucleus compared to 2s which contradicts the theory. Also, we do not see the supposed smaller peak at ~1a for the 2s state. We think the result for n=2 states are wildly inaccurate, but this is the best we can do for this setup.