Homework 1 - Advanced Computational Physics

Desani Ramadhan - 100065797

21 January 2025

Question 1: Numerical solutions of ODEs [20 points]

1. Write your own **rk2** method. Design your method for a general ODE; this means making the derivative function $f(t, x)$ a separate method.
2. Use your **rk2** solver in a program that solves the equation of motion (a second-order ODE) for a charged particle moving in a variable external electric field within an anharmonic trapping potential given by:

$$m\frac{d^2x}{dt^2} = F_{ext}(x,t) - kx^{p-1}$$

   where $p = 4$ and $F_{ext} = -qEx^2$; assume $q = m = k = 1$ and $E = 15$ for simplicity and assume that the motion starts at $x(0) = 0$ with the particle velocity $x'(0) = 3$.
3. Plot the position of the particle $x(t)$ for the first 10 seconds of motion ($t = 0$ to 10). Find the turning points of the motion (min and max of the position, $x(t)$).
4. Solve the same equation using the **rk4** and **rk45** solvers (write your own code and/or use existing implementations in SciPy). Plot the three solutions (rk2, rk4, and rk45) in one figure and the relative error of rk2 vs rk4 (or rk45) in another figure.

Question 2: Euler's method [10 points]

Write a Python program to solve a 1D motion of a rectangular object of a surface area $A = 100$ cm^2 (regardless of the orientation) and mass $m = 0.1$ kg in a free fall (near the surface of the Earth) with the air resistance present. Assume the motion is due to the gravitational force $F_g = -mg$ and the drag (stopping force due to the air resistance), $F_D = \frac{1}{2}\rho v^2 C_D A$, where ρ is the density of air and $C_D = 0.7$ is the drag coefficient.

a. What is the approximate terminal velocity for the motion of this object?
b. Plotl $y(t)$ as a function of t with and without drag for the duration of the motion if the object is dropped from $y(t = 0) = 100$ m with zero initial velocity.

Question 3: Spherical harmonics [10 points]

Write a Python program that plots first 5 spherical Bessel functions of the first and second kind.

Answer

Question 1

This section is for question no 1, sub-question no 1, 2, and 3

The general solution of 1^{st} order ordinary differential equation ("ODE") of $\frac{dx}{dt} = f(x,t)$ using Runge-Kutta method is:

$$x_{i+1} = x_i + \phi(t_i, x_i, h) \text{ .. } (1)$$

where $\phi(t_i, x_i, h)$ is the slope and h is the step. For the 2^{nd} order Runge-Kutta method ("rk2"), we can further extend the slope to be:

$$x_{i+1} = x_i + (ak_1 + bk_2)h \text{ .. } (2)$$

where $k_1 = f(x_i, t_i)$ and $k_2 = f(x_i + \alpha h, t_i + \beta k_1 h)$.

If we expand x_{i+1} into the second-order Taylor series and solve with equation (2), we will get the result as follows:

$$a = 1 - b \text{ and } \alpha = \beta = \tfrac{1}{2b} \text{ .. } (3)$$

Since there is infinite number of values of b, there are infinite number of rk2. However, there are 3 most commonly used versions:

- **Heun Method with single corrector ($b = 1/2$)**

  $$x_{i+1} = x_i + (\tfrac{k_1}{2} + \tfrac{k_2}{2})h \text{ .. } (4)$$

  where $k_1 = f(x_i, t_i)$ and $k_2 = f(t_i + \frac{h}{2}, x_i + k_1 \frac{h}{2})$

- **Midpoint Method ($b = 1$)**

  $$x_{i+1} = x_i + k_2 h \text{ .. } (5)$$

  where $k_1 = f(x_i, t_i)$ and $k_2 = f(t_i + h, x_i + k_1 h)$

- **Ralston's Method ($b = 2/3$)**

  $$x_{i+1} = x_i + (\tfrac{k_1}{3} + \tfrac{2k_2}{3})h \text{ .. } (6)$$

  where $k_1 = f(x_i, t_i)$ and $k_2 = f(t_i + \frac{3h}{4}, x_i + \frac{3k_1 h}{4})$

We will write the code to solve the problem for sub-question no. 2 for each method to solve the following differential equation:

$$m\tfrac{d^2x}{dt^2} = F_{ext}(x,t) - kx^{p-1} \text{ .. } (7)$$

where $p = 4, F_{ext} = -qEx^2, q = m = k = 1$ and $E = 15; x(0) = 0 and x'(0) = 3$. If we simplify further, we will have 2^{nd} order ODE as below:

$$\tfrac{d^2x}{dt^2} = -15x^2 - x^3 \text{ .. } (8)$$

We further separate the problem into two $1st$ order ODE as below:

$$\tfrac{dv}{dt} = -15x^2 - x^3 \text{ and } \tfrac{dx}{dt} = v \text{ .. } (9)$$

where $x(0) = 0$ and $v(0) = 3$

Python Output

- **rk2 method with steps (h) = 0.01**

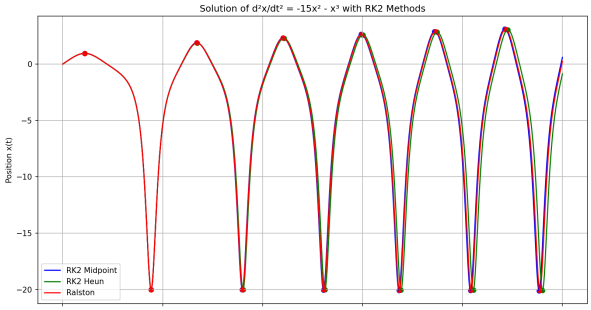

Figure 1a. x(t) plot for 10s with h = 0.01

Table of Turning Points

t_max	x_max_midpoint	x_max_heun	x_max_ralston	t_min	x_min_midpoint	x_min_heun	x_min_ralston
0.44	0.95	0.95	0.95	1.77	-20.02	-19.99	-20.01
2.68	1.90	1.87	1.89	3.59	-20.02	-20.01	-20.02
4.41	2.33	2.29	2.32	5.22	-20.05	-20.01	-20.02
5.97	2.64	2.60	2.63	6.73	-20.07	-20.04	-20.05
7.44	2.89	2.84	2.88	8.16	-20.08	-20.04	-20.05
8.85	3.10	3.05	3.09	9.53	-20.09	-20.07	-20.09

Figure 1b. Table of turning points with h = 0.01

- **rk2 method with steps (h) = 0.001**

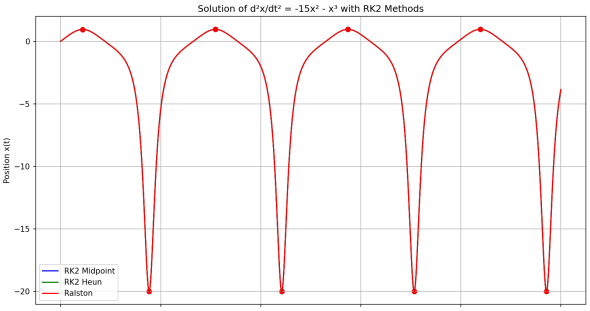

Figure 2a. x(t) plot for 10s with h = 0.001

Table of Turning Points

t_max	x_max_midpoint	x_max_heun	x_max_ralston	t_min	x_min_midpoint	x_min_heun	x_min_ralston
0.44	0.95	0.95	0.95	1.77	-20.00	-20.00	-20.00
3.10	0.95	0.95	0.95	4.42	-20.00	-20.00	-20.00
5.74	0.96	0.95	0.96	7.07	-20.00	-20.00	-20.00
8.39	0.96	0.96	0.96	9.71	-20.00	-20.00	-20.00

Figure 2b. Table of turning points with h = 0.001

The figures above show the plot of the differential equation solution of $x(t)$ for 10 s using 3 different rk2 methods.

From the two graphs, we can inferred that the Heun, Midpoint and Ralston methods produce similar results. However, stability is only achieved when the step is set to h = 0.001 instead of h = 0.01.

Therefore, we find the anharmonic plot with a **minimum position of** $x(t) = -20$ and **maximum position of** $x(t) = 0.95$.

This section is for question no 1, sub-question no 4

In this sub-section we will solve the problem using 3 different methods. We will compare the result obtained from the rk2 method with the other 2 methods. The rk2 method will be using midpoint method.

- 4^{th} order Runge-Kutta method ("rk4")
  The formula for the rk4 method as follows:

  $$x_{i+1} = x_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k4)h \ ..(10)$$

  where:

  $$k_1 = f(x_i, t_i); k_2 = f(t_i + \frac{h}{2}, x_i + \frac{k_1 h}{2}); k_3 = f(t_i + \frac{h}{2}, x_i + \frac{k2h}{2}); k_4 = f(t_i + h, x_i + k_3 h)$$

- Runge-Kutta Fehlberg ("rk45")
  We also will use Runge-Kutta Fehlberg ("rk45") method which use 4^{th} and 5^{th} order Runge-Kutta method.

  The formula for the rk45 method as follows:

  4^{th} order formula:

  $$x_{i+1}(4) = x_i + h\left(\frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5\right) \ ..(11),$$

  5^{th} order formula:

  $$x_{i+1}(5) = x_i + h\left(\frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6\right)..(12)$$

  where:
  $k_1 = f(x_i, t_i)$,

  $k_2 = f\left(t_i + \frac{1}{4}h, x_i + \frac{1}{4}hk_1\right)$,

  $k_3 = f\left(t_i + \frac{3}{8}h, x_i + \frac{3}{32}hk_1 + \frac{9}{32}hk_2\right)$,

  $k_4 = f\left(t_i + \frac{12}{13}h, x_i + \frac{1932}{2197}hk_1 - \frac{7200}{2197}hk_2 + \frac{7296}{2197}hk_3\right)$,

  $k_5 = f\left(t_i + h, x_i + \frac{439}{216}hk_1 - 8hk_2 + \frac{3680}{513}hk_3 - \frac{845}{4104}hk_4\right)$

  $k_6 = f\left(t_i + \frac{1}{2}h, x_i - \frac{8}{27}hk_1 + 2hk_2 - \frac{3544}{2565}hk_3 + \frac{1859}{4104}hk_4 - \frac{11}{40}hk_5\right)$.

  The local truncation error is estimated as:

  $$E = x_{i+1}(5) - x_{i+1}(4). \ ..(13)$$

  The step size is adjusted based on the error:

  $$h_{new} = h \cdot \left(\frac{tolerance}{|E|}\right)^{1/5}. \ ..(14)$$

Python Output

- **rk2 VS rk4 VS rk45 method with steps (h) = 0.01**

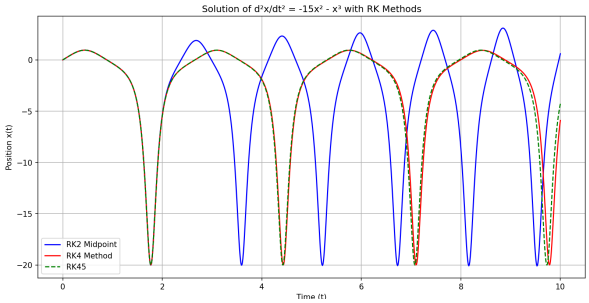

Figure 3a. x(t) plot for 10s rk2 vs rk4 vs rk45 with h = 0.01

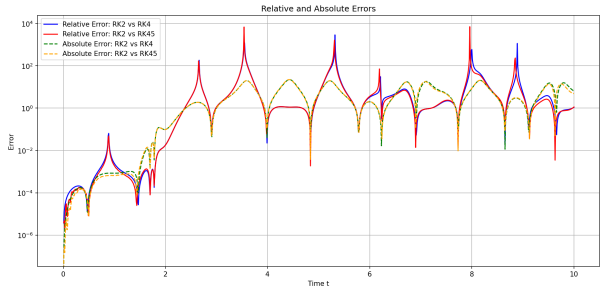

Figure 3b. Relative and Absolute Errors rk2 vs rk4 and rk2 vs rk45 with h = 0.01

- **rk2 VS rk4 VS rk45 method with steps (h) = 0.001**

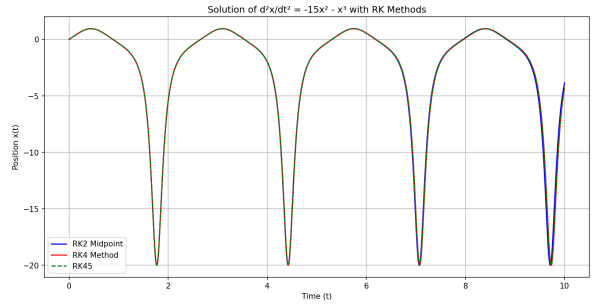

Figure 4a. x(t) plot for 10s rk2 vs rk4 vs rk45 with h = 0.001

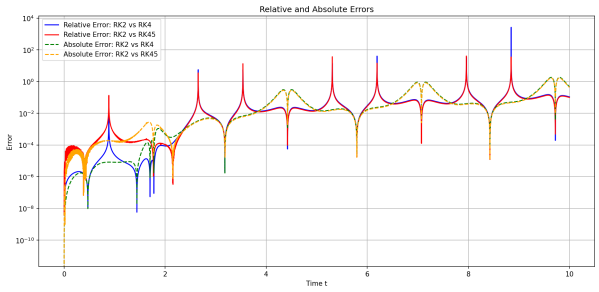

Figure 4b. Relative and Absolute Errors rk2 vs rk4 and rk2 vs rk45 with h = 0.001

Based on the results above, we compare the calculation with step sizes h = 0.01 and h=0.001, we can infer that:

- The rk45 method closely aligns with rk4 in both cases, as these are higher-order methods compared to rk2. This demonstrates that rk4 and rk45 produce more accurate results than rk2 for larger steps (h = 0.01). However, rk2 also produce accurate and similar result with rk4 and rk45 for smaller steps (h = 0.001).
- For larger steps (h = 0.01), the relative and absolute errors for rk2 compared to rk4 and rk45 is higher than smaller steps (h = 0.001). Some relative errors point are very large due to the nature of the relative errors formula for a very small denominator value.

$$\text{relative error} = \left|\frac{x_a - x_b}{xb}\right|$$

where x_a = output from rk2 and x_b = output from rk4 or rk45.

Hence, we can look into the absolute error to see the difference. As we can see in the smaller step, the abslute error of rk2 compared to rk4 and rk45 is below 10^1 which indicate the accuracy achieved by rk2 in the smaller step.

$$\text{absolute error} = |x_a - x_b|$$

where x_a = output from rk2 and x_b = output from rk4 or rk45.

- In conclusion, rk2 is less accurate than rk4 and rk45 and require smaller steps to be more accurate. For more efficient calculation we can use higher order Runge-Kutta method like rk4 and rk45 to allow larger steps which reduce the computation time and cost.

Question 2

This section is for question no 2, sub-question no a

Terminal velocity happens when the net force is zero and the object stops accelerating resulting in constant speed which then called terminal velocity.

In our case, we can find the terminal velocity (v_t) when the net force is zero:

$$F_{net} = F_{drag} + F_{gravity}$$
$$0 = \frac{1}{2}\rho v_t^2 C_d A - mg$$
$$v_t = \sqrt{\frac{2mg}{\rho C_d A}} \text{ ..(15)}$$

Where:

$$m = 0.1;\ A = 0.01;\ g = 9.8;\ \rho = 1.225 \text{ (air density)};\ C_d = 0.7$$

We then can compute the value using simple python code as below:

m = 0.1
A = 0.01
g = 9.8
rho = 1.225
$C_d = 0.7$
$v_t = ((2 * m * g)/(rho * C_d * A)) * *0.5$
$print(f"Terminal Velocity : v_t : .2f m/s")$

Which give us the value of the terminal velocity (v_t) = 15.12 m/s

This section is for question no 2, sub-question no b

We will plot y(t) with and without the drag force assuming the object is dropped from y(t=0) = 100 m and the initial velocity v(t=0) = 0 m/s.

- Equation of motion with the drag force $F_d = \frac{1}{2}\rho v_t^2 C_d A$:

$$F_{net} = F_{drag} + F_{gravity}$$
$$ma = \frac{1}{2}\rho v^2 C_d A - mg$$
$$a = \frac{\rho v^2 C_d A}{2m} - g$$
$$\frac{d^2x}{dt^2} = \frac{\rho v^2 C_d A}{2m} - g \text{..(16)}$$

  We can further write in the form of two 1^{st} order ODE as below:

$$\frac{dv}{dt} = \frac{\rho v^2 C_d A}{2m} - g \text{ and } \frac{dx}{dt} = v \text{ ..(17)}$$

  We will solve the differential equation using Euler Method as below:

$$y_{i+1} = y_i + f(t_i, y_i)h \text{ ..(18)}$$

  Where $f(t_i, y_i)$ or $\frac{dy_i}{dt_i}$ is the slope and h is the number of steps.

- Equation of motion without the drag force $F_d = \frac{1}{2}\rho v_t^2 C_d A$:

  From classical mechanics, we know the function of position over time for a free fall object (zero initial velocity) as below:

$$y(t) = \frac{1}{2}gt^2 \quad ..(19)$$

Python Output

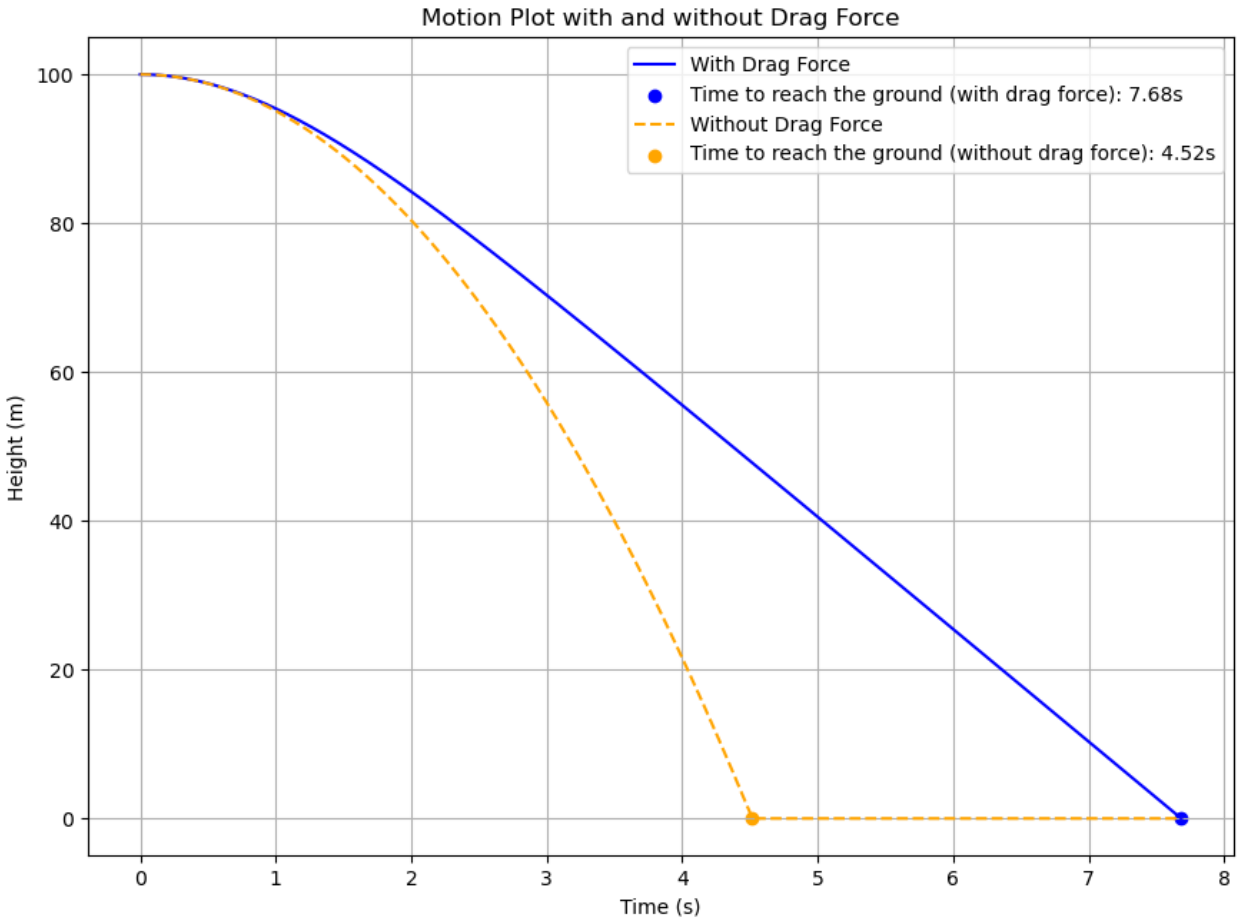

Figure 4. Position (y(t)) plot as a function of time t with and without the drag force (Fd)

As can be seen in the graph, the time for the object to hit the ground is longer when we include the drag force or air resistance in the calculation compared to the equation without the drag force.

This is logical as there should be air resistance in the actual world and due to the Newton's third law, there will be drag force from the air resistance in reaction with the gravity force of the falling object which makes the time for the object to hit the ground is longer.

Question 3

There are two kinds of spherical Bessel function:

- First Kind ($J_n(x)$)

  The formula of the first kind spherical bessel function as below:

  $$J_n(x) = (-x)^n (\frac{1}{x}\frac{d}{dx})^n \frac{sinx}{x} \quad ..(20)$$

  Hence, we can write down the first two of the first kind spherical Bessel function for n = 0 and n = 1 as below:

  $J_0(x) = \frac{sinx}{x}$
  $J_1(x) = \frac{sinx}{x^2} - \frac{cosx}{x}$

  As for the next J_n function, we can find through the recursive formula as below:

  $$J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x) \quad ..(21)$$

  For our computation purpose, since we want to find $J_2(x), J_3(x)$ and $J_4(x)$, while $n_0 = 2$ in our computation formula (because we have defined the formula for $n = 0$ and $n = 1$ and we want to start the iteration to find $J_2(x)$), we can modify the indices of our recursive formula as below:

  $$J_n(x) = \frac{2(n-1)}{x} J_{n-1}(x) - J_{n-2}(x) \quad ..(22)$$

  Based on the recursive formula on equation (22) we can plot the first 5 first kind spherical Bessel function as below:

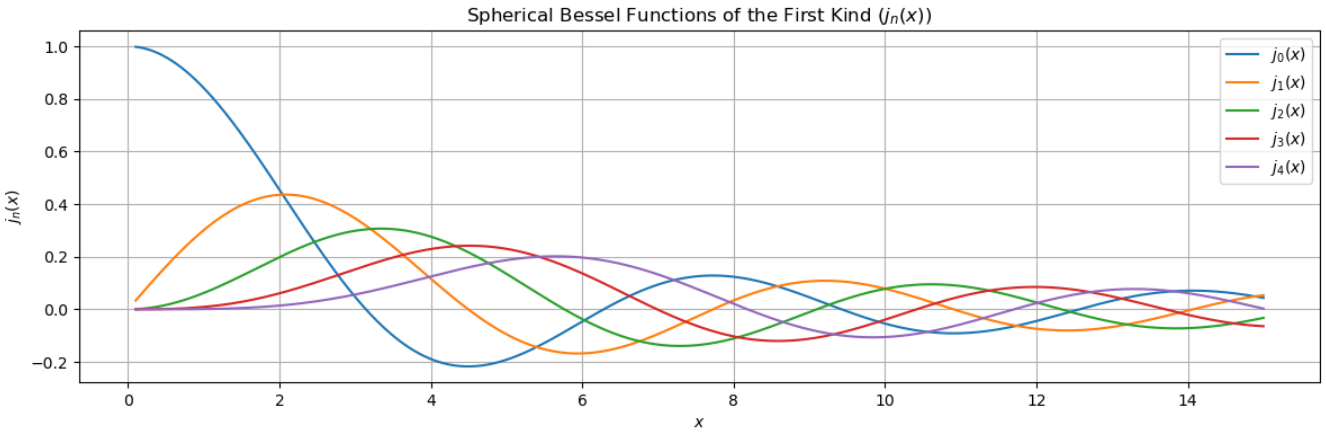

Figure 5. First 5 spherical Bessel Function - First Kind

- Second Kind ($Y_n(x)$)
  The formula of the second kind spherical bessel function or spherical Neumann function as below:

$$Y_n(x) = (-x)^n \left(\frac{1}{x}\frac{d}{dx}\right)^n \frac{cosx}{x} \quad ..(23)$$

Hence, we can write down the first two of the second kind spherical Bessel function for n = 0 and n = 1 as below:

$Y_0(x) = -\frac{cosx}{x}$
$Y_1(x) = -\frac{cosx}{x^2} - \frac{sinx}{x}$

As for the next Y_n function, we can find through the recursive formula as below:

$$Y_{n+1}(x) = \frac{2n}{x} Y_n(x) - Y_{n-1}(x) \quad ..(24)$$

For our computation purpose, since we want to find $Y_2(x), Y_3(x)$ and $Y_4(x)$, while $n_0 = 2$ in our computation formula (because we have defined the formula for $n = 0$ and $n = 1$ and we want to start the iteration to find $Y_2(x)$), we can modify the indices of our recursive formula as below:

$$Y_n(x) = \frac{2(n-1)}{x} Y_{n-1}(x) - Y_{n-2}(x) \quad ..(25)$$

Based on the recursive formula on equation (25) we can plot the first 5 second kind spherical Bessel function as below:

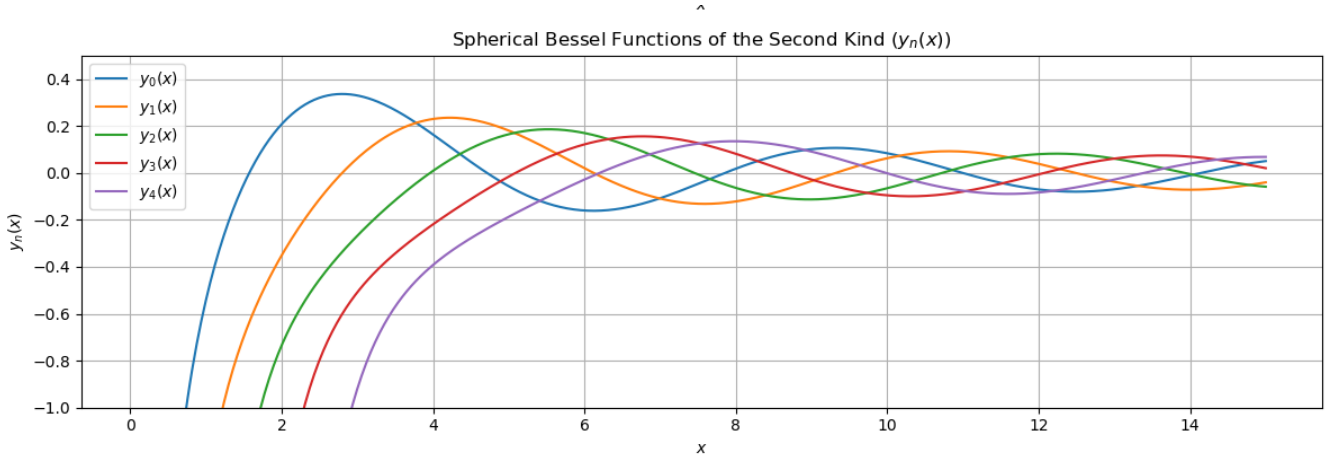

Figure 6. First 5 spherical Bessel Function - Second Kind