Numerical analysis of Taylor Series

Shashvat Jain
(2020PHY1114)(20068567054)

January 15, 2022

Lab Report for Assignment No. 1

College Roll No :	2020PHY1114
University Roll NoName:	20068567054
Unique Paper Code:	32221401
Paper Title:	Mathematical physics III Lab
Course and Semester :	B.Sc.(H) Physics Sem IV
Due Date:	Jan 15,2022
Date of Submission:	Jan 14,2022
Lab Report File Name:	mp3A1_2020PHY1114.pdf
Partner's Name:	Harsh Saxsena
Partner's College Roll No.:	2020PHY1162

1 Theory

Any one-variable infinitely differentiable real-valued function $f(x) : A \to B$ where $A, B \subseteq \mathbb{R}$ might be expanded as an infinite power series function with parameter $x_0 \in A$, This series function is also termed as **Taylor series** representation of f because of its procurement from the **Taylor's Theorem.**

$$\begin{aligned} f(x) = T(x, x_0) &= f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3 + \cdots \\ &= \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \end{aligned} \tag{1}$$

Taylor series representation of a function with the parameter $x_0 = 0$ is called the **Maclaurin series**.

$$f(x) = T(x, 0) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!}(x)^n \tag{2}$$

The point on the line $x = x_0$ is called the center of taylor series. The value of the function and its derivatives must be known at the center and the radius of convergence of the series is determined about this point.

Radius of Convergence of a power series: Every power series has a radius of convergence R which is the distance of its center from the nearest singularity(point of divergence). If $R > 0$, then the power series $\sum_{n=0}^{\infty} c_n(x - x_0)^n$ converges for all $|x - a| \leq R$ and diverges for $|x - a| > R$. If the series converges for all x, then we write $R = \infty$.

Taylor series representation for a function of two variables $f(x, y) : \mathbb{R}^2 \to \mathbb{R}$ about $(x, y) = (x_0, y_0)$ is given by the Taylor theorem as follows,

$$\begin{aligned} f(x, y) = T((x, y), (x, y_0)) = f(x_0, y_0) + f_x|_{x_0,y_0}(x - x_0) + f_y|_{x_0,y_0}(y - y_0) + + f_x x|_{x_0,y_0}(x - x_0)^2 \\ + f_{yy}|_{x_0,y_0}(y - y_0)^2 + f_{xy}|_{x_0,y_0}(x - x_0)(y - y_0) + \cdots \end{aligned} \tag{3}$$

The functions $\exp(x), \sin(x), \cos(x) : \mathbb{R} \to \mathbb{R}$ are defined by the following Maclaurin series expansions.

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} \qquad \text{for all } x \tag{4}$$

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \qquad \text{for all } x \tag{5}$$

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \qquad \text{for all } x \tag{6}$$

2 Algorithm

Algorithm 1 Calculate partial sums for taylor series for exponential function

procedure EXP(x, a, n)
  Input: x points , a about which we calculate series,n as no. of terms to include in partial temrs
  Output: Returns y the approx value of e^x
  $exp = 0$ ▷ Initialize sum
  for $k = 0, 1, 2, 3 \ldots n$ **do**
    $exp = exp + \frac{(x-a)^k}{k!}$
  end for
  return exp
end procedure

Algorithm 2 Calculate n-partial sums for taylor series for Sin function

procedure SINSERIES(x, a, n)
Input: x points , a about which we calculate series,n as no. of terms to include in partial terms
Output: Returns y the approx value of $\sin(x)$
$sin = 0$ ▷ Initialize sum
for $k = 0, 1, 2, 3 \ldots n$ **do**
$sin = sin + \frac{(-1)^k (x-a)^{2k+1}}{(2k+1)!}$
end for
return sin
end procedure

Algorithm 3 Calculate n-partial sums for taylor series for Cos function

procedure COSSERIES(x, a, n)
Input: x points , a about which we calculate series,n as no. of terms to include in partial terms
Output: Returns y the approx value of $\cos(x)$
$\cos = 0$ ▷ Initialize sum
for $k = 0, 1, 2, 3 \ldots n$ **do**
$\cos = \cos + \frac{(-1)^k (x-a)^{2k}}{(2k)!}$
end for
return cos
end procedure

3 Programming

First we defined the following functions for MySinSeries,MyCosSeries which take in **x**: the point where the series is to be calculated, **a**: the center of the Taylor series and **n**: the number of terms of the series to calculate.

```
import matplotlib.pyplot as plt
import math
import numpy as np
from numba import vectorize
import pandas as pd
plt.style.use("seaborn-dark-palette")

@vectorize
def exp(x,a,n):
  sum_ = 1
  for i in np.arange(1,n):
    sum_ += (x-a)**(i)/math.gamma(i+1)
  return(sum_)

@vectorize
def MySinSeries(x,a,n):
  sum_ = 0
  for i in np.arange(n):
    sum_ += (-1)**i*(x-a)**(2*i+1)/math.gamma(2*i+2)
  return(sum_)

@vectorize
def MyCosSeries(x,a,n):
  sum_ = 0
  for i in np.arange(n):
    sum_ += (-1)**i*(x-a)**(2*i)/math.gamma(2*i+1)
  return(sum_)

def get_n_sin(x,rtol = 0.5e-4):
```

```
    i_ = np.zeros(x.shape)
    y1_,max_rel_ = i_.copy(),i_.copy()
    for k in np.arange(len(x)):
      y0 = MySinSeries(x[k],0,1)
      for i in range(2,1000):
        y1 = MySinSeries(x[k],0,i)
        max_rel = np.max(np.abs((y1 - y0)/y1))
        if max_rel <= rtol :
          i_[k] = i ; y1_[k] = y1 ; max_rel_[k] = max_rel
          break
        elif max_rel == 0:
          i_[k] = 0 ; y1_[k] = y1 ; max_rel_[k] = 0
        y0 = y1.copy()
    return([i_,y1_,max_rel_])
```

Next we proceeded obtain the data and get the necessary plots using the following script.

```
    xs= np.linspace(-2*np.pi,2*np.pi,1000)
    m = np.arange(2,20,2)
    fig1,(ax1,ax2) = plt.subplots(1, 2)
    yj_sin = np.array([MySinSeries(xs,0,i) for i in m],dtype=float)

    for j in range(len(m)) :
        ax1.plot(xs,yj_sin[j],label=f"m={m[j]}")
    ax1.plot(xs,np.sin(xs),label="Numpy's sin(x) ")
    ax1.set_ylim([-10, 10])

    setaxis(ax1,"$\sin(x)$",["x","y"])
    ym_sin = MySinSeries(np.pi/4,0,m)
    ax2.plot(m,ym_sin,"-*",label=r"MySinSeries($\frac{\pi}{4},m$)")
    ax2.plot(m,np.sin(np.pi/4)*np.ones(m.shape),"-",label=r"Numpy's $\sin(\frac{\pi
}{4})$")
    ax2.legend()
    ax1.set_xlabel("x");ax1.set_ylabel("y")
    ax2.set_xlabel("m");ax2.set_ylabel(r"$\cos(\frac{\pi}{4})$")

    fig2,(ax12,ax22) = plt.subplots(1, 2)
    yj_cos = np.array([MyCosSeries(xs,0,i) for i in m],dtype=float)
    for j in range(len(m)) :
        ax12.plot(xs,yj_cos[j],label=f"m={m[j]}")
    ax12.plot(xs,np.cos(xs),label="Numpy's cos(x) ")
    ax12.set_ylim([-10, 10])
    setaxis(ax12,"$\cos(x)$")
    ax12.set_xlabel("x");ax12.set_ylabel("y")

    ym_cos= MyCosSeries(np.pi/4,0,m)
    ax22.plot(m,ym_cos,"-*",label=r"MyCosSeries($\frac{\pi}{4},m$)")
    ax22.plot(m,np.cos(np.pi/4)*np.ones(m.shape),"-",label=r"Numpy's $\cos(\frac{\pi
}{4})$")
    ax22.set_xlabel("m");ax22.set_ylabel(r"$\cos(\frac{\pi}{4})$")
    ax22.legend()
    xvec = np.arange(0,np.pi+0.1,np.pi/8)
    reltol = 0.5e-6
    n,calsin,relerror =get_n_sin(xvec,reltol)
    table = pd.DataFrame({"x": xvec , "MySinSeries(x)" :map(lambda x: f"{x:#.9g}",
calsin),"n":n ,"Numpy's sin(x)":map(lambda x: f"{x:#.9g}",np.sin(xvec))})
    table.to_csv("table.csv")
    fig0,ax0 = plt.subplots(1, 1)
    xs2 = np.linspace(0,2*np.pi)
    ax0.plot(xs2,np.sin(xs2),label= "Numpy's sin(x) continuous")
    ax0.scatter(xvec,list(map(lambda x: float(f"{x:#.3g}"),calsin)),label = "
MySinSeries() with 3 significant digits")
    ax0.set_xlabel("x");ax0.set_ylabel("y")
    print(table)
    plt.plot()
    plt.show()
```

4 Discussion

The results show us that the calculated value of $\sin(x)$ approaches the so-called true value (Numpy's approximation) as more and more terms are taken into account.

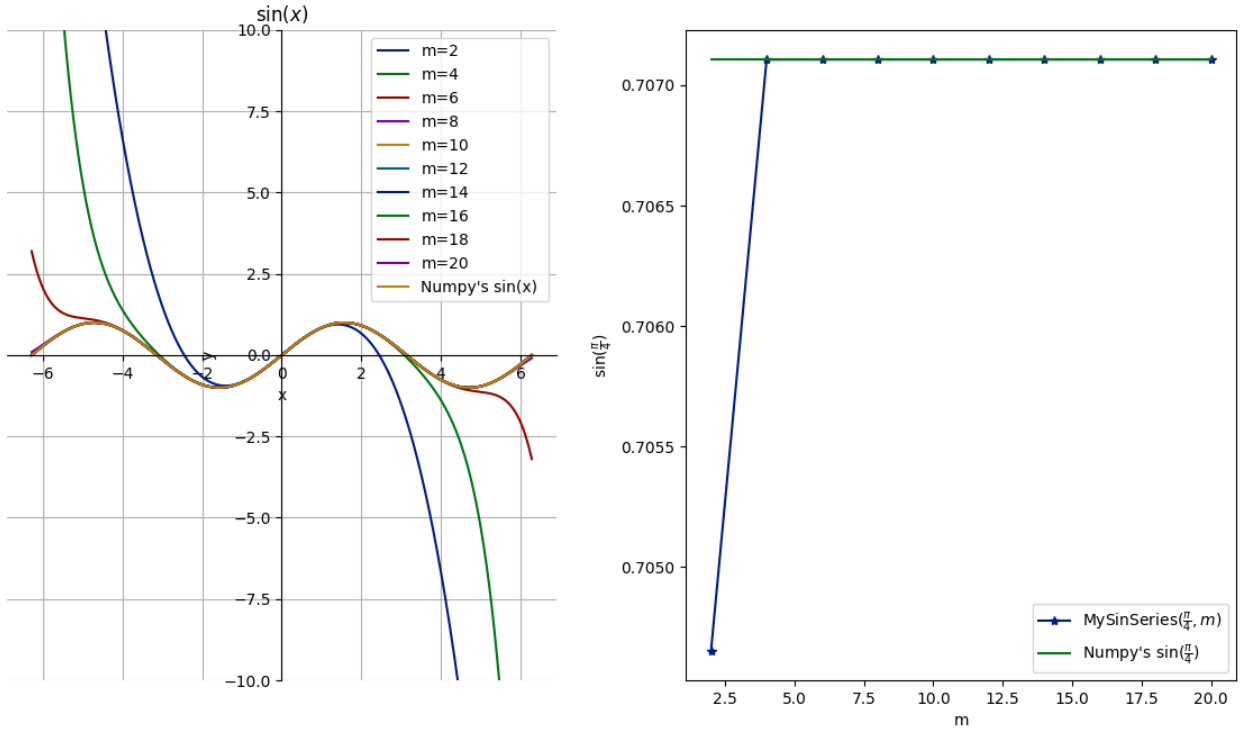


Figure 1: Graphs generated for MySinSeries: The graph on the left shows the different taylor series approximations corresponding to different values of m=2,4,...20. The graph on the right shows how the value of $\sin(\pi/4)$ varies with different values of m.

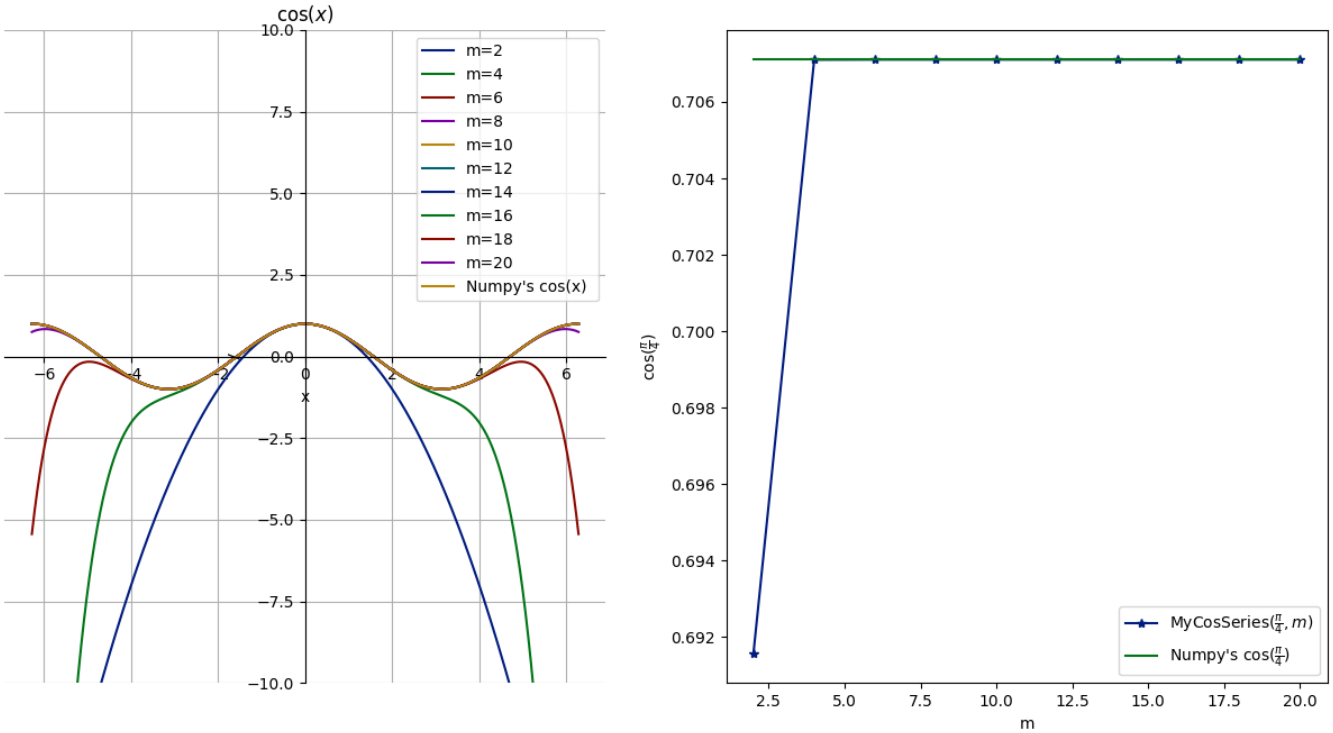
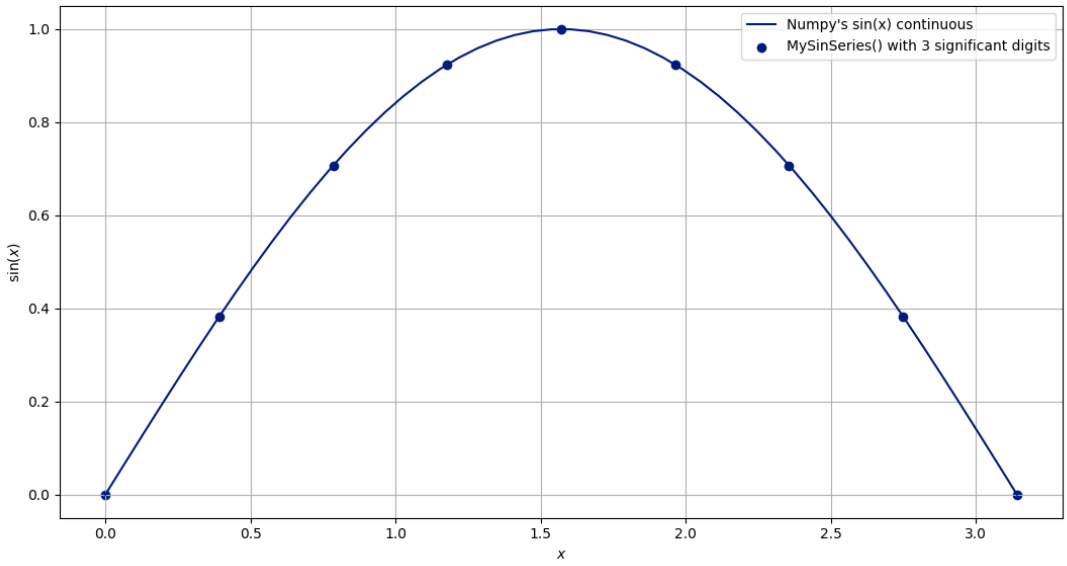


Figure 2: Graphs generated for MyCosSeries: The graph on the left shows the different taylor series approximations corresponding to different values of m=2,4,...20. The graph on the right shows how the value of $\cos(\pi/4)$ varies with different values of m.



(a) Plot of sin(x) vs x (upto 3 significant digits), the scattered plot is MySinSeries(x,n) with n varied to get the desired accuracy of 6 significant digits.

	x	MySinSeries(x)	n	Numpy's sin(x)
0	0.000000	0.00000000	0.0	0.00000000
1	0.392699	0.382683432	5.0	0.382683432
2	0.785398	0.707106783	5.0	0.707106781
3	1.178097	0.923879531	6.0	0.923879533
4	1.570796	1.00000000	7.0	1.00000000
5	1.963495	0.923879532	8.0	0.923879533
6	2.356194	0.707106775	8.0	0.707106781
7	2.748894	0.382683434	9.0	0.382683432
8	3.141593	2.46987570e-16	18.0	1.22464680e-16

(b) The data for the various calculated values of $\sin(x)$ using Numpy and MySinSeries upto accuracy of 6 significant digits.

Figure 3: Graphs for programming Question 2.

The graphs indicate that the value of MySinSeries() needs to be calculated for a larger value of n, that is more terms are required, for a fixed accuracy as we move away from the centre a of the expansion. This result is only natural as the partial sums give best approximations for points near the centre a. We also observe that the first 6 significant digits do not match for $x = \pi$, this is because the value is very close to zero resulting in a huge round-off errors in the relative error.