Name: Nguyễn Đình Khánh Ngân

ID: ITCSIU22236

Lab 1

Part 1:

• Compute the first 4 terms of the Taylor Series for ex around x=0.

```
#Part 1
import math as m

def solve(x):
  return 1 + x + (x**2)/m.factorial(2) + (x**3)/m.factorial(3)
x = 1
exp_approxiate = solve(x)
exp_real = m.exp(x)
print('part 1')
print(f'For x = {x}')
print(f'Taylor approximate = {exp_approxiate}')
print(f'Taylor real = {exp_real}')
```

• Run the code and record the output (approximation and actual value) in your report.

```
part 1
For x = 1
Taylor approximate = 2.6666666666666665
Taylor real = 2.718281828459045
```

Part 2:

• Write a function to compute the Taylor Series approximation of sin(x).

```
#Part 2
import matplotlib.pyplot as plt
import math as m

#function compute Taylor Series approximate of sin(x)
def sin_T(x, n):
  t = 0.0
  for i in range(n):
    t += ((-1)**i * x**(2*i+1))/m.factorial(2*i+1)
  return t
```

- Run the code and copy the results into your report.

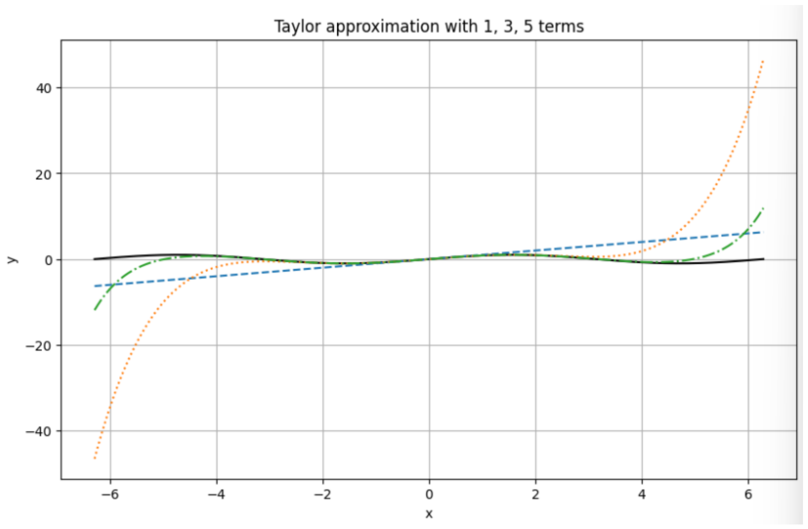
```
r = m.pi/2
appr = sin_T(r, 3)
print('part 2')
print(f'For x = {r}')
print(f'Taylor approximate = {appr}')
```

```
part 2
For x = 1.5707963267948966
Taylor approximate = 1.0045248555348174
```

- Plot the actual sin(x) against its Taylor approximations with 1, 3, and 5 terms.

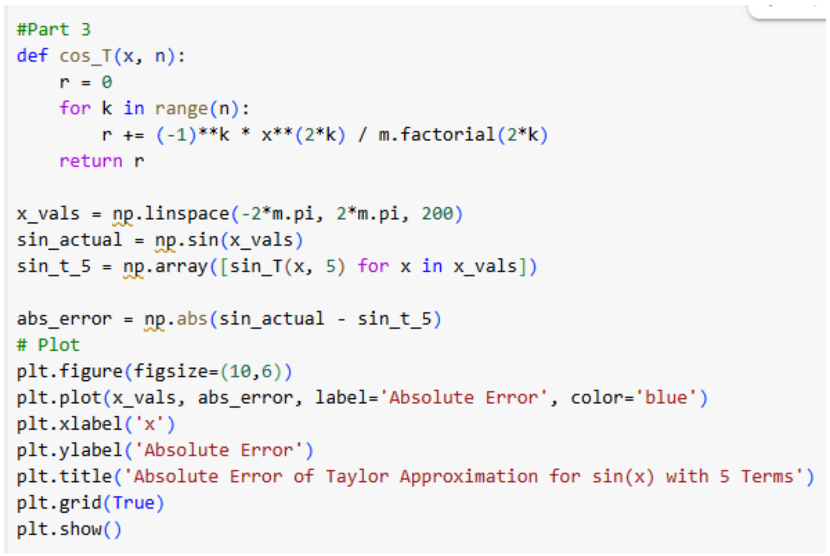
```
import matplotlib.pyplot as plt

# Prepare x values
x_vals = np.linspace(-2*m.pi, 2*m.pi, 400)
y_actual = np.sin(x_vals)
y_1 = np.array([sin_T(x, 1) for x in x_vals])
y_3 = np.array([sin_T(x, 3) for x in x_vals])
y_5 = np.array([sin_T(x, 5) for x in x_vals])
# Create the plot
plt.figure(figsize=(10, 6))
plt.plot(x_vals, y_actual, label="sin(x) actual", color='black')
plt.plot(x_vals, y_1, label="Taylor 1 term", linestyle="--")
plt.plot(x_vals, y_3, label="Taylor 3 terms", linestyle=":")
plt.plot(x_vals, y_5, label="Taylor 5 terms", linestyle="-.")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Taylor approximation with 1, 3, 5 terms")
plt.grid(True)
plt.show()
```

- Save the plot as sin_taylor.png and include it in your report.



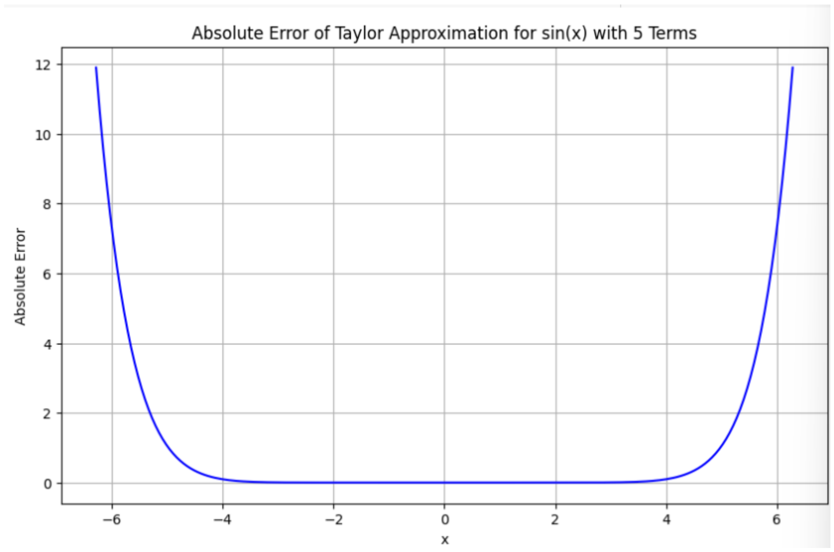
Part 3:

• Compute and plot the absolute error of the Taylor approximation for sin(x) with 5 terms.



```python
#Part 3
def cos_T(x, n):
    r = 0
    for k in range(n):
        r += (-1)**k * x**(2*k) / m.factorial(2*k)
    return r

x_vals = np.linspace(-2*m.pi, 2*m.pi, 200)
sin_actual = np.sin(x_vals)
sin_t_5 = np.array([sin_T(x, 5) for x in x_vals])

abs_error = np.abs(sin_actual - sin_t_5)
# Plot
plt.figure(figsize=(10,6))
plt.plot(x_vals, abs_error, label='Absolute Error', color='blue')
plt.xlabel('x')
plt.ylabel('Absolute Error')
plt.title('Absolute Error of Taylor Approximation for sin(x) with 5 Terms')
plt.grid(True)
plt.show()
```

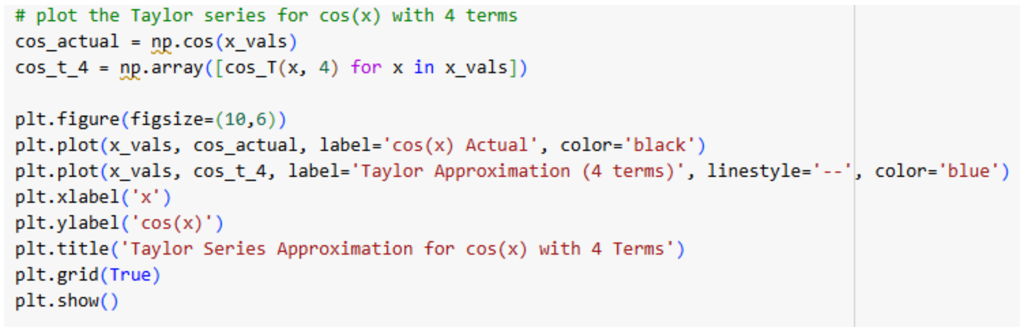
• Save the plot as sin_error.png and include it in your report.



• Where is the error largest? Why?

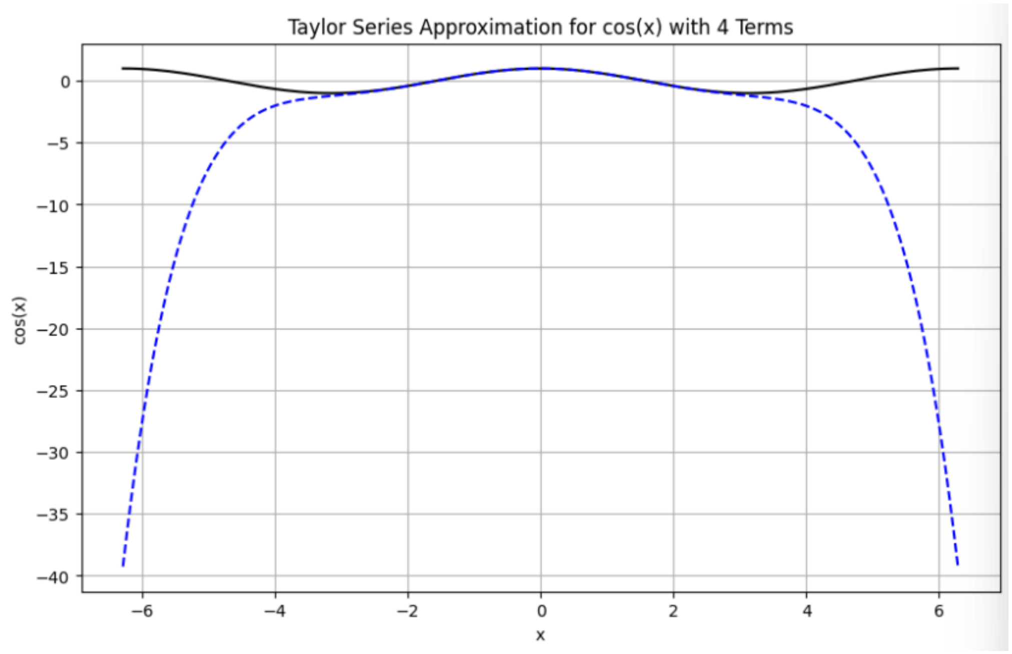
The error largest due to the endpoints of the interval, because the Taylor series is centered at x = 0. The approximate is mot accurate close to the center point, as x increase, higher – order terms become more significant. Therefore, as we move further away from zero point, the impact of these negligible terms increases, leading to larger absolute error

• Implement the Taylor Series for cos(x) with 4 terms around x=0.



```
# plot the Taylor series for cos(x) with 4 terms
cos_actual = np.cos(x_vals)
cos_t_4 = np.array([cos_T(x, 4) for x in x_vals])

plt.figure(figsize=(10,6))
plt.plot(x_vals, cos_actual, label='cos(x) Actual', color='black')
plt.plot(x_vals, cos_t_4, label='Taylor Approximation (4 terms)', linestyle='--', color='blue')
plt.xlabel('x')
plt.ylabel('cos(x)')
plt.title('Taylor Series Approximation for cos(x) with 4 Terms')
plt.grid(True)
plt.show()
```

• Run the code, record the results in your report, and save the plot as cos_taylor.png for inclusion.



• Write a short paragraph (5-7 sentences) in your report: Compare the convergence of the Taylor Series for sin(x) and cos(x).

The Taylor series for sin(x) converges very quickly near x=0, where its odd-powered terms effectively capture the function's behavior. However, as x increases, the approximation error grows because the higher order omitted terms become more significant. In contrast, Taylor series for cos(x) uses even-powered terms and exhibits a slightly smoother convergence over a broader interval due to its even symmetry. Both series provide approximations close to the center, but their accuracy diminishes as we move further from zero point. Overall, while both series converge rapidly near x=0, the cosine series tends to be more stable over a larger range, making it marginally more reliable for approximating cos(x) away from the center.