DS623 PE06 – Taylor Polynomial Visualization

Topic: Differentiating Univariate Functions and Taylor Approximations
This script visualizes f(x) = sin(x) + cos(x) and its Taylor approximations T_0, T_1, T_3, T_5 on [-4, 4] using step size 0.001.
It optionally finds the x values that yield min/max of f(x).
School: City University of Seattle
Student: Verónica Elze

🤖 Mission Briefing: We are Robotic Unit EL-ZE. Our directive is to mimic the curve f(x) = sin(x) + cos(x) using polynomial approximations!
Engage mathematical processors and deploy the Taylor series up to orders 0, 1, 3, and 5.

Imports

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from math import factorial, sin, cos
import pandas as pd
```

Define f(x) = sin(x) + cos(x)

🤖 Directive 1: Define the original signal pattern (function)

In [2]:
```
def f(x):
    return np.sin(x) + np.cos(x)
```

Define Taylor series approximation centered at x0 = 0

🤖 Directive 2: Deploy Taylor Module Generator
This bot builds a Taylor copycat of f(x), using up to n terms, all starting from x = 0.

In [3]:
```
def taylor_series(f, x, n):
    taylor_approx = np.zeros_like(x)
    for k in range(n + 1):
        if k % 4 == 0:
            coeff = 1
        elif k % 4 == 1:
            coeff = -1 if k % 2 == 1 else 1
        elif k % 4 == 2:
            coeff = -1
        else:
            coeff = 1 if k % 2 == 1 else -1

        taylor_approx += coeff * x**k / factorial(k)
    return taylor_approx
```

Define x values

🤖 Directive 3: Set scanning range for x-values (our domain of operation)

In [4]:
```
x_vals = np.arange(-4, 4.001, 0.001)
f_vals = f(x_vals)
```

Compute Taylor approximations

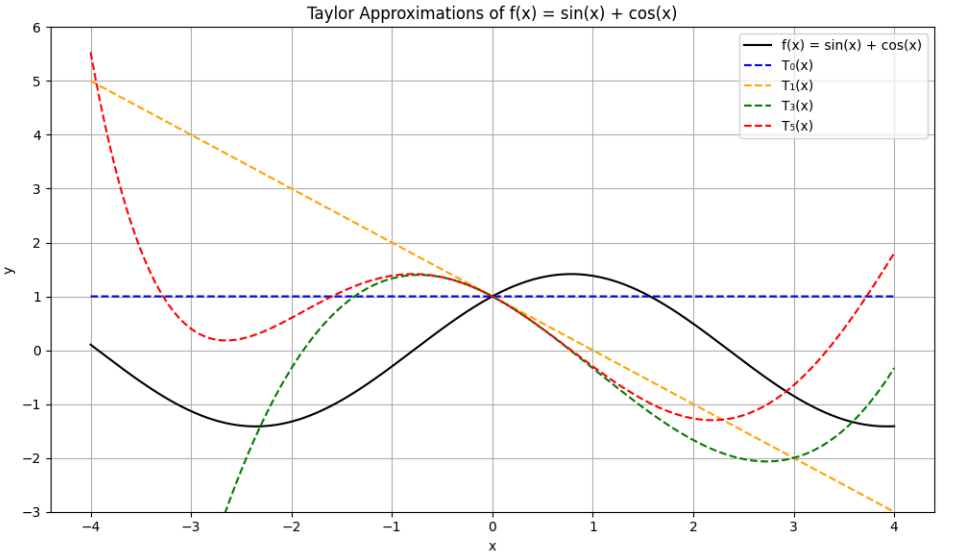
🤖 Directive 4: Compute Taylor approximations with increasing order of intelligence

In [5]:
```
t0 = taylor_series(f, x_vals, 0)
t1 = taylor_series(f, x_vals, 1)
t3 = taylor_series(f, x_vals, 3)
t5 = taylor_series(f, x_vals, 5)
```

Plotting

🤖 Directive 5: Visualize approximations and compare to original pattern

In [6]:
```
plt.figure(figsize=(10, 6))
plt.plot(x_vals, f_vals, label='f(x) = sin(x) + cos(x)', color='black')
plt.plot(x_vals, t0, '--', label='T₀(x)', color='blue')
plt.plot(x_vals, t1, '--', label='T₁(x)', color='orange')
plt.plot(x_vals, t3, '--', label='T₃(x)', color='green')
plt.plot(x_vals, t5, '--', label='T₅(x)', color='red')
plt.ylim([-3, 6])
plt.title("Taylor Approximations of f(x) = sin(x) + cos(x)")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



Optional: Find max and min points

🤖 Directive 6: Analyze terrain highs and lows in our operation range

In [7]:
```
x_max = x_vals[np.argmax(f_vals)]
x_min = x_vals[np.argmin(f_vals)]
```

Display extrema

🤖 Report findings to mission control

In [8]:
```
extrema_df = pd.DataFrame({
    "Label": ["x_max", "x_min"],
    "x_value": [x_max, x_min]
})
```

In [9]:
```
print("📡 Extrema Analysis Report:\n")
print(extrema_df)
```

```
📡 Extrema Analysis Report:

   Label  x_value
0  x_max    0.785
1  x_min    3.927
```

OpenAI. (2025). ChatGPT's assistance with Taylor series implementation [Large language model]. https://openai.com/chatgpt