TAYLOR SERIES AND POLYNOMIALS

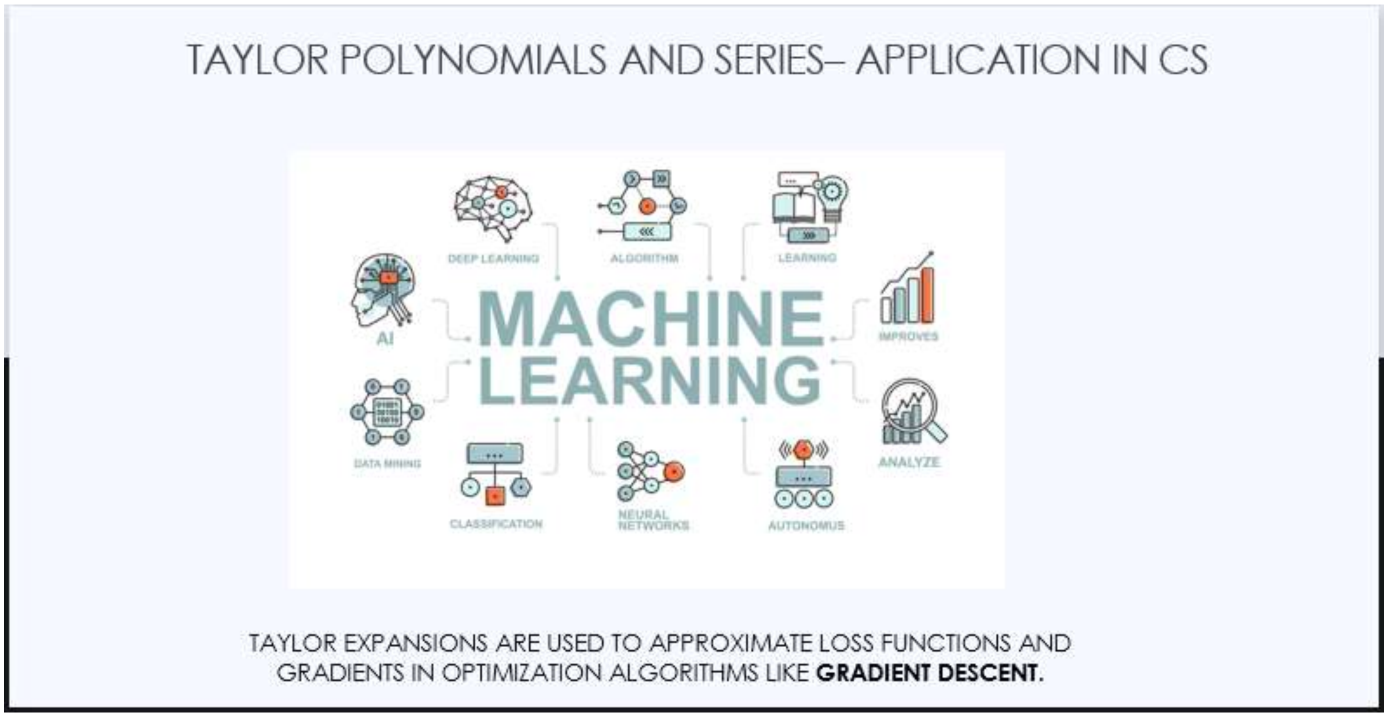
Imagine you're trying to figure out how a curve behaves, like a roller coaster track, but you only know one point on the track. The Taylor series is like building a really good approximation of the whole curve using just that one point and the information about how steep the curve is at that point (its slope), how the steepness changes (curvature), and so on.

- It's like creating a recipe for the curve based on:
- 1. Where you are (the point on the curve).
- 2. How steep it is (first derivative).
- 3. How quickly that steepness changes (second derivative), and so on.

With each extra piece of information, you get a better and better idea of what the curve looks like near that point. The more terms you add, the more accurate the approximation becomes.

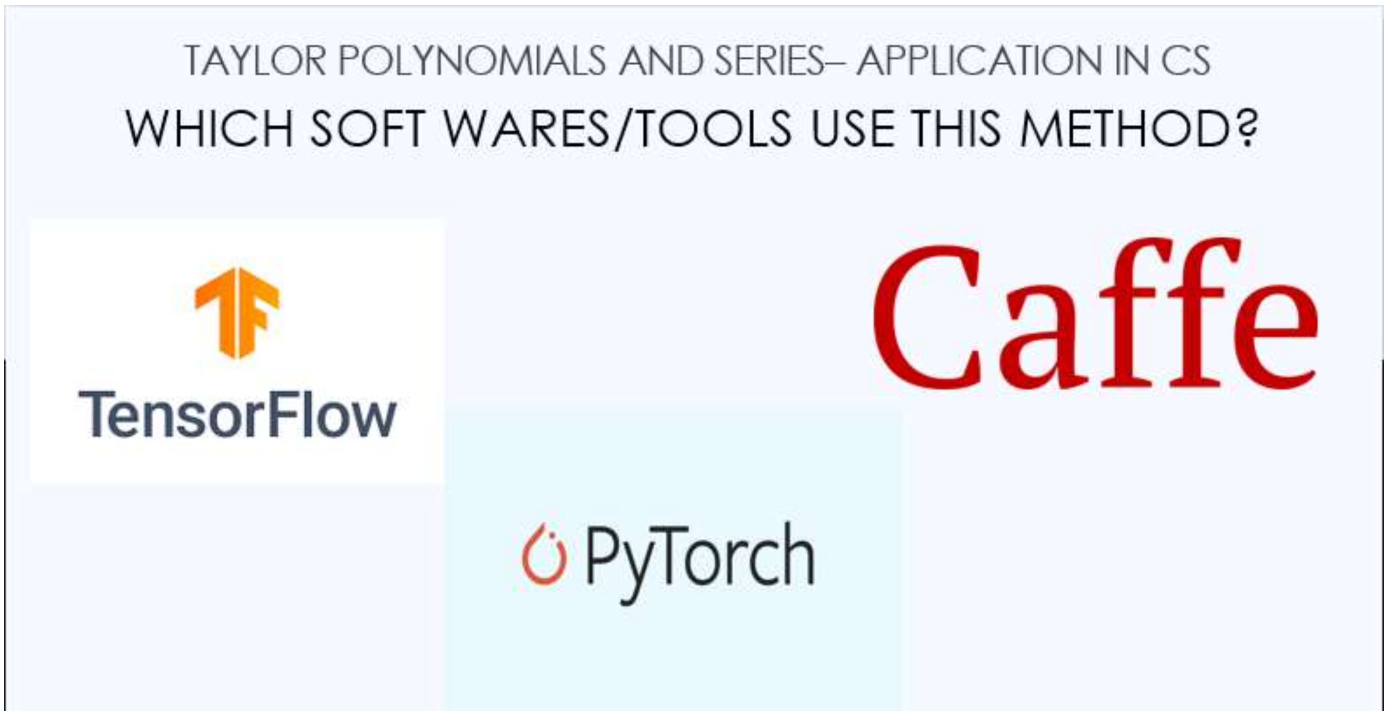
In short, the Taylor series lets you approximate complex curves or functions using simpler terms—essentially breaking down a complicated shape into small, easy-to-understand pieces, one step at a time. It's used in everything from computer graphics to engineering, where precise calculations are needed but exact solutions are difficult.

APPLICATION IN CS



TAYLOR POLYNOMIALS AND SERIES– APPLICATION IN CS

TAYLOR EXPANSIONS ARE USED TO APPROXIMATE LOSS FUNCTIONS AND GRADIENTS IN OPTIMIZATION ALGORITHMS LIKE **GRADIENT DESCENT.**



TAYLOR POLYNOMIALS AND SERIES– APPLICATION IN CS

WHICH SOFT WARES/TOOLS USE THIS METHOD?

TensorFlow

Caffe

PyTorch

MATHEMATICAL REPRESENTATION

Taylor series and polynomials are of basic importance and used extensively in numerical analysis, in approximating a differentiable function and its integral etc.

Taylor Series

Let f and its higher order derivatives be continuous on [a, b] and x_0 be a fixed point in [a, b]. Then, for $x \in$ [a, b], $f(x)$ can be expressed as the infinite series (called Taylor's Series)

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + \frac{f^{(n+1)}(x_0)}{(n+1)!}(x - x_0)^{n+1} + \cdots$$

Taylor Polynomial

If the Taylor's series is truncated to the term $\frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$, then the polynomial obtained is called the Taylor's polynomial

$$f(x) \approx P_n(x) = f(x_0) + f^{'}(x_0)(x - x_0) + \frac{f^{''}(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

***Theorem* (Taylor's Theorem)**

Suppose $f \in C^n[a, b]$, that $f^{(n+1)}$ exists on $[a, b]$, and $x_0 \in [a, b]$. For every $x \in [a, b]$, there exists a number $\xi(x)$ between x_0 and x with

$$f(x) = P_n(x) + R_n(x)$$

Where

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

and

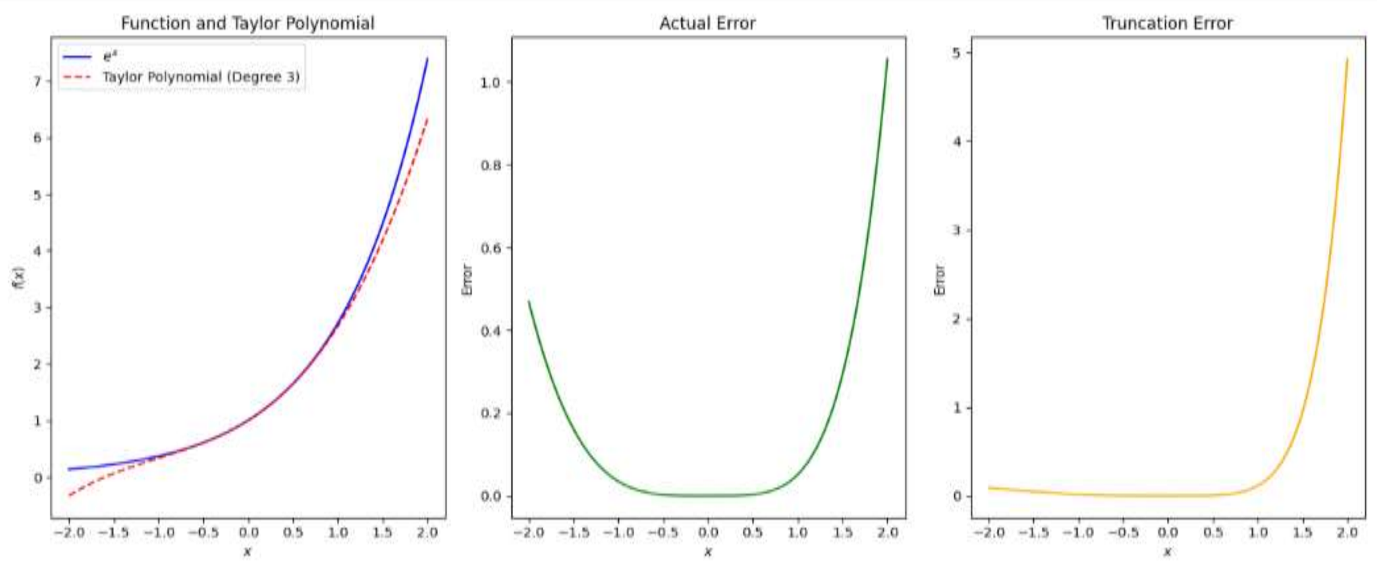
$$R_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)^{n+1}.$$

Here $P_n(x)$ is the ***n*th Taylor polynomial** for f about x_0, and $R_n(x)$ is the **remainder term** (or **truncation error**) associated with $P_n(x)$. Since the number $\xi(x)$ in the truncation error $R_n(x)$ depends on the value of x at which the polynomial $P_n(x)$ is being evaluated, it is a function of the variable x. However, we should not expect to be able to explicitly determine the function $\xi(x)$. Taylor's Theorem simply ensures that such a function exists, and that its value lies between x and x_0.

In the case $x_0 = 0$, the Taylor polynomial is often called a **Maclaurin polynomial**, and the Taylor series is often called a **Maclaurin series**.

The term **truncation error** in the Taylor polynomial refers to the error involved in using a truncated, or finite, summation to approximate the sum of an infinite series.

GRAPHICAL REPRESENTATION



EXAMPLES

Example 1. Let $f(x) = \cos x$ and $x_0 = 0$. Determine

(a) the second, third and fourth Taylor polynomial for f about x_0; and use these polynomials to approximate $f(0.02)$ by finding the actual error.

(b) an upper bound for $|f(0.02) - P_n(0.02)|$ using the error formula $R_2(x)$, compare it to the actual error.

Solution. Since $f \in C^\infty(\mathrm{R})$, Taylor's Theorem can be applied for any $n \geq 0$. Also,

$$f'(x) = -\sin x, \qquad f''(x) = -\cos x, \qquad f'''(x) = \sin x, \qquad \text{and} \quad f^{(4)}(x) = \cos x,$$

So

$$f(0) = 1, \qquad f'(0) = 0, \qquad f''(0) = -1, \quad f'''(0) = 0, \qquad and \quad f^{(4)}(0) = 1.$$

(a) For n = 2 and $x_0 = 0$, we have

$$P_2(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 = 1 - \frac{1}{2}x^2$$

To approximate $f(0.02)$,

we use $P_2(x)$ at x = 0.02,

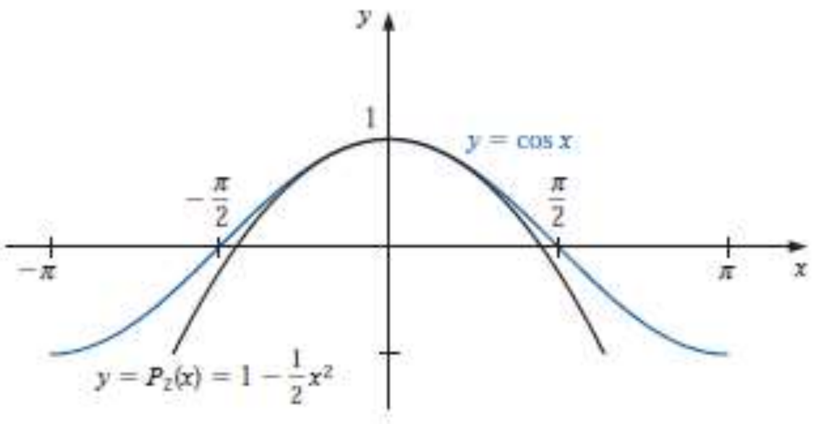
Implies $P_2(0.02)$ =0.9998

While **exact value** is

$$\cos 0.02 = 0.9998000067,$$

Therefore, **Actual error**

$$= 6.67 \times 10^{-9} .$$



For n = 3 and $x_0 = 0$, we have

$$P_3(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 = 1 - \frac{1}{2}x^2$$

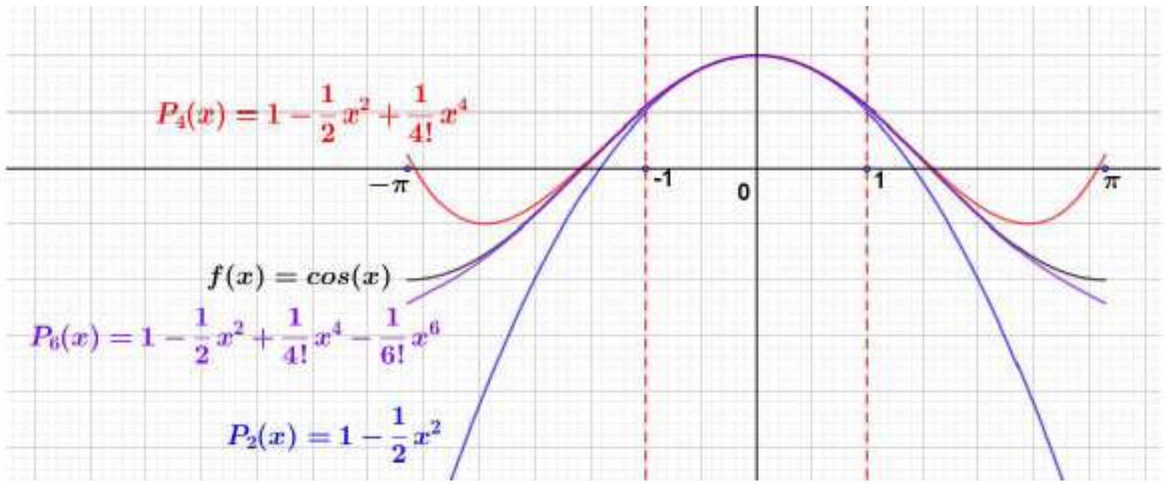
$$P_3(0.02) = 0.9998, \text{ same as } P_2(0.02).$$

For n = 4 and $x_0 = 0$, we have

$$P_4(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \frac{f^{(4)}(0)}{4!}x^4$$

$$= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4$$

$$\Longrightarrow P_4(0.02) = 0.99980001$$



(b) To find upper bound of error we use formula

$$R_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)^{n+1}.$$

where $\xi(x)$ is some (generally unknown) number between 0 and x.

Therefore, the upper bound of error for $P_2(x)$ will be

$$R_2(x) = \frac{f'''(\xi(x))}{3!}x^3 = \frac{\sin \xi(x)}{6}x^3$$

The error bound is much larger than the actual error. This is due to the poor bound we used for $|\sin \xi(x)|$.

As from calculus, we have $|\sin x| \le |x|$. Since $0 \le \xi < 0.02$, we could have used the fact that $|\sin \xi(x)| \le 0.02$ in the error formula, producing the bound $R_2(0.02) = 2.67 \times 10^{-8}$.

Similarly, the upper bound of error for $P_3(x)$ will be

$$R_3(x) = \frac{f^{(4)}(\xi(x))}{4!}x^4 = \frac{\cos \xi(x)}{24}x^4$$

Since, $|\cos \xi(x)| \le 1$, so $R_3(0.02) \le 6.67 \times 10^{-9}$.

Example 2.

Find Taylor series and polynomials of degree 4 and 5 for the function $\sin x$ about $x_0 = 0$ and use the polynomial to find the actual error at $x = 0.1$ and $x = 0.5$.

Solution:

$$f(x) = \sin x, \qquad f(0) = 0$$
$$f'(x) = \cos x, \qquad f'(0) = 1$$
$$f''(x) = -\sin x, \qquad f''(0) = 0$$
$$f'''(x) = -\cos x, \qquad f'''(0) = -1$$
$$f^{(4)}(x) = \sin x, \qquad f^{(4)}(0) = 0$$

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + + \frac{f^{(4)}(0)}{4!}x^4 + \cdots$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

or $\quad \sin x \approx P_4(x) = x - \frac{x^3}{3!} \quad$ and $\quad \sin x \approx P_5(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!}$

To approximate $f(0.05)$ at $x = 0.1$, we use $P_4(x)$ and $P_5(x)$ at x = 0.1, respectively

Implies $P_4(0.1) = 0.09983333$ and $P_5(0.1) = 0.09983341667$.

While **exact value** is $\sin 0.1 = 0.09983341665$,

Therefore, **Actual error** for $P_4(0.1) = 8.7 \times 10^{-8}$ and for $P_5(0.1) = 2 \times 10^{-11}$, resp.

* But if we check error at $x = 0.5$, with $P_5(0.5) = 0.4794271$

and **exact value** is $\sin 0.5 = 0.4794255,$

Therefore, **Actual error** $= 0.0000016,$ greater error at point farther from $x_0 = 0$.

Example 3. Find $P_2(x)$ for the function $f(x) = e^x \cos x$ about $x_0 = 0$.

(a) Use $P_2(0.5)$ to approximate $f(0.5)$ by finding the actual error.

(b) Find an upper bound of error $|f(0.5) - P_2(0.5)|$ using formula $R_2(x)$, and compare it to the actual error.

(c) Find an upper bound of error $|f(x) - P_2(x)|$ using $R_2(x)$ on the interval $[0,1]$.

(d) Approximate $\int_0^1 f(x)dx$ using $\int_0^1 P_2(x)\,dx$.

Solution. As, $f(x) = e^x \cos x,$

so $f'(x) = e^x(\cos x - \sin x),$ $\quad f''(x) = -2e^x \sin x,$

and $f'''(x) = -2e^x(\cos x + \sin x)$

So

$$f(0) = 1, \qquad f'(0) = 1, \qquad f''(0) = 0.$$

For n = 2 and $x_0 = 0$, we have

$$P_2(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 = 1 + x$$

(a) $$P_2(0.5) = 1.5$$

While **exact value** is $f(0.5) = 1.4469$ So, **Actual error** $= 0.0531$.

(b) To find upper bound of error for $P_2(x)$ will be

$$R_2(x) = \frac{f'''(\xi(x))}{3!}x^3 = -\frac{x^3}{3}e^{\xi(x)}(\cos\xi(x) + \sin\xi(x))$$

$$\text{so } |R_2(0.5)| = \left|-\frac{0.5^3}{3}\right| \left|e^{\xi(x)}(\cos\xi(x) + \sin\xi(x))\right|$$

$$\leq 0.103 \quad \text{taking} \quad \xi(x) = 0.5, \ \cos\xi(x) = 1 \ \text{ and } \ \sin\xi(x) = \xi(x).$$

(c) To find upper bound of error for $P_2(x)$ on the interval $[0,1]$

$$R_2(x) = \frac{f'''(\xi(x))}{3!}x^3 = -\frac{x^3}{3}e^{\xi(x)}(\cos\xi(x) + \sin\xi(x))$$

$$\leq \left|-\frac{1^3}{3}\right| \left|e^{\xi(x)}(\cos \xi(x) + \sin \xi(x))\right|$$

$$= 1.812 \qquad \text{taking} \quad \xi(x) = 1,\ \cos \xi(x) = 1,\ \sin \xi(x) = \xi(x)$$

(d) To approximate $\int_0^1 f(x)dx$, we use

$$\int_0^1 P_2(x)\,dx = \int_0^1 (1+x)\,dx = 1.5$$

While **exact value** is $\int_0^1 e^x \cos x \ dx = 1.378.$

Example 4. Let $f(x) = \sqrt{1-x}$ and $x_0 = 0$. Determine

(a) Third and fourth Taylor polynomial for f about x_0; and use these polynomials to approximate $f(0.2)$ by finding the actual error.

(b) an upper bound of error using formula $R_3(x)$, compare it to the actual error.

(c) Approximate $\int_0^{0.5} f(x)dx$ using $\int_0^{0.5} P_3(x)\,dx$.

Solution. As, $f(x) = \sqrt{1-x}$,

$$\text{so} \quad f'(x) = -\tfrac{1}{2}(1-x)^{-\frac{1}{2}}, \qquad f''(x) = -\tfrac{1}{4}(1-x)^{-\frac{3}{2}},$$

$$f'''(x) = -\frac{3}{8}(1-x)^{-5/2}, \qquad \text{and} \quad f^{(4)}(x) = -\frac{15}{16}(1-x)^{-7/2}$$

So

$$f(0) = 1, \qquad f'(0) = -\frac{1}{2}, \qquad f''(0) = -\frac{1}{4},$$

$$f'''(0) = -\frac{3}{8}, \qquad \textit{and} \qquad f^{(4)}(0) = -\frac{15}{16}.$$

(a) For n = 3 and $x_0 = 0$, we have

$$P_3(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3$$

$$= 1 - \tfrac{1}{2}x - \tfrac{1}{8}x^2 - \tfrac{1}{16}x^3$$

$$P_3(0.2) = 0.8945$$

While **exact value** is $\quad f(0.2) = \sqrt{1-0.2} = 0.8944272$

So, **Actual error** $= 7 \times 10^{-5}$.

For n = 4 and $x_0 = 0$, we have

$$P_4(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \frac{f^{(4)}(0)}{4!}x^4$$

$$= 1 - \tfrac{1}{2}x - \tfrac{1}{8}x^2 - \tfrac{1}{16}x^3 - \tfrac{5}{128}x^4$$

$$\Longrightarrow P_4(0.2) = 0.8944375.$$

Therefore, **Actual error** $= 1.03 \times 10^{-5}$.

(b) To find upper bound of error for $P_3(x)$ will be

$$R_3(x) = \frac{f^{(4)}\big(\xi(x)\big)}{4!}x^4 = -\frac{5}{128}(1-\xi(x))^{-7/2}x^4$$

$$\text{so } R_3(0.2) \le 1.365 \times 10^{-4} \qquad \text{taking} \quad \xi(x) = 0.2.$$

(c) To approximate $\int_0^{0.5} f(x)dx$, we use

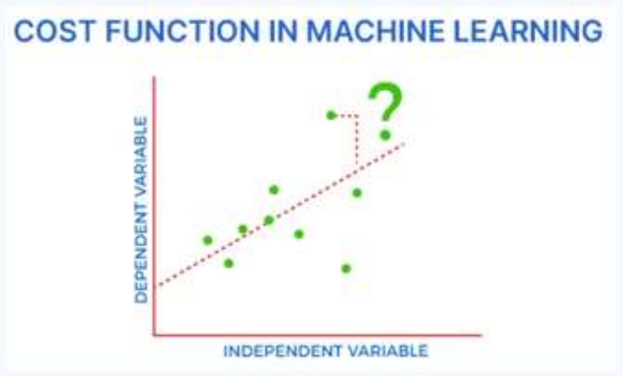
$$\int_0^{0.5} P_3(x)\,dx = \int_0^{0.5}(1 - \tfrac{1}{2}x - \tfrac{1}{8}x^2 - \tfrac{1}{16}x^3)\,dx = \; 0.431315$$

While **exact value** is $\int_0^{0.5} \sqrt{1-x}\,dx = 0.430964.$

Therefore, **Actual error** $= 3.51 \times 10^{-4}$.

PROBLEM STATEMENT

In machine learning, gradient-based optimization algorithms are widely used to train models. To accelerate convergence, some methods use approximations of functions through Taylor series expansions.
Suppose you are working on training a machine learning model, and you need to approximate the value of a complex cost function
$f(x) = e^x\cos(x)$ at a point $x = 0.5$ using a Taylor series expansion around $x_0 = 0$.
You have decided to use the 4th-degree Taylor polynomial $P_4(x)$ for this approximation.



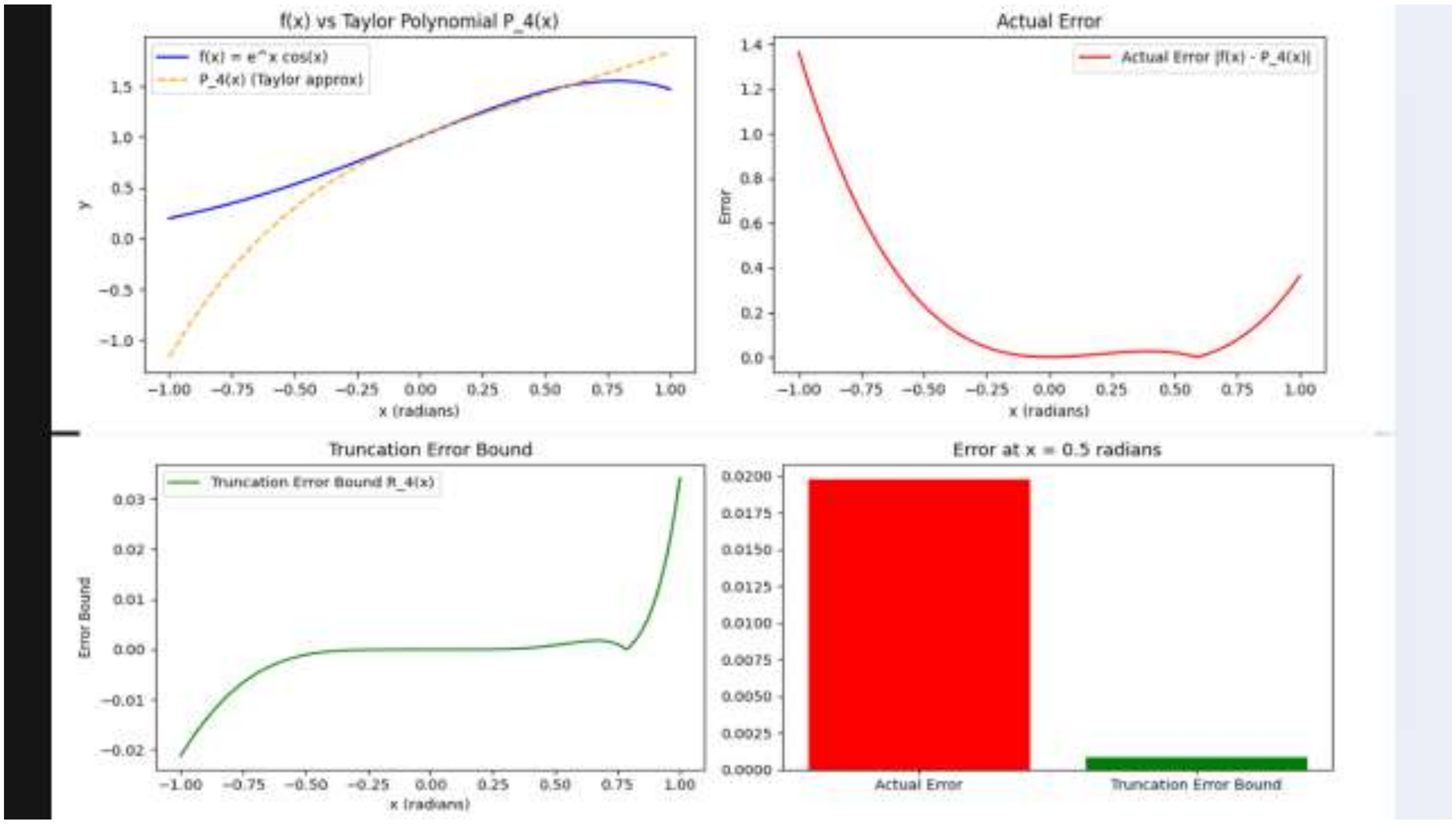
1. Compute the 4th-degree Taylor polynomial $P_4(x)$ for $f(x) = e^x \cos(x)$ about $x_0 = 0$.
2. Use $P_4(0.5)$ to approximate $f(0.5)$. Calculate the actual error by finding the difference between $f(0.5)$ and $P_4(0.5)$.
3. Determine an upper bound for the truncation error $|f(0.5) - P_4(0.5)|$ using the remainder term $R_4(x)$, and compare it to the actual error.

ANSWERS

(Solve it yourself!)

Function value at x = 0.5 : 1.4468890365841693
Taylor polynomial approximation at x = 0.5 : 1.4270833333333333
Actual error at x = 0.5: 0.019805703250836082
Upper bound of truncation error at x = 0.5: 0.0008547525434512426

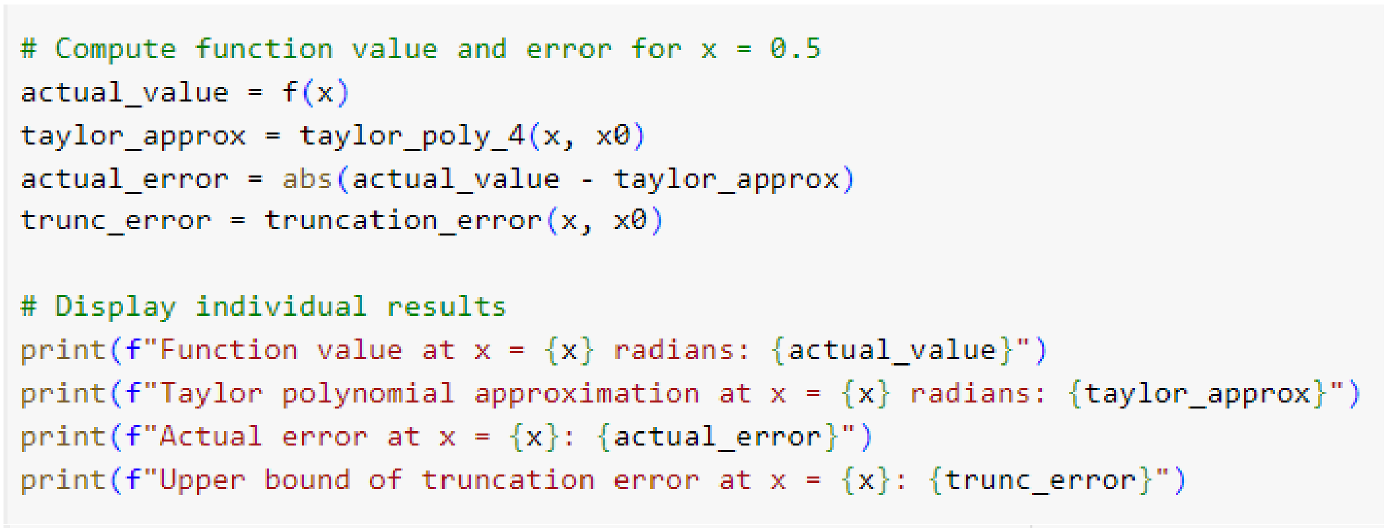
ERROR ANALYSIS



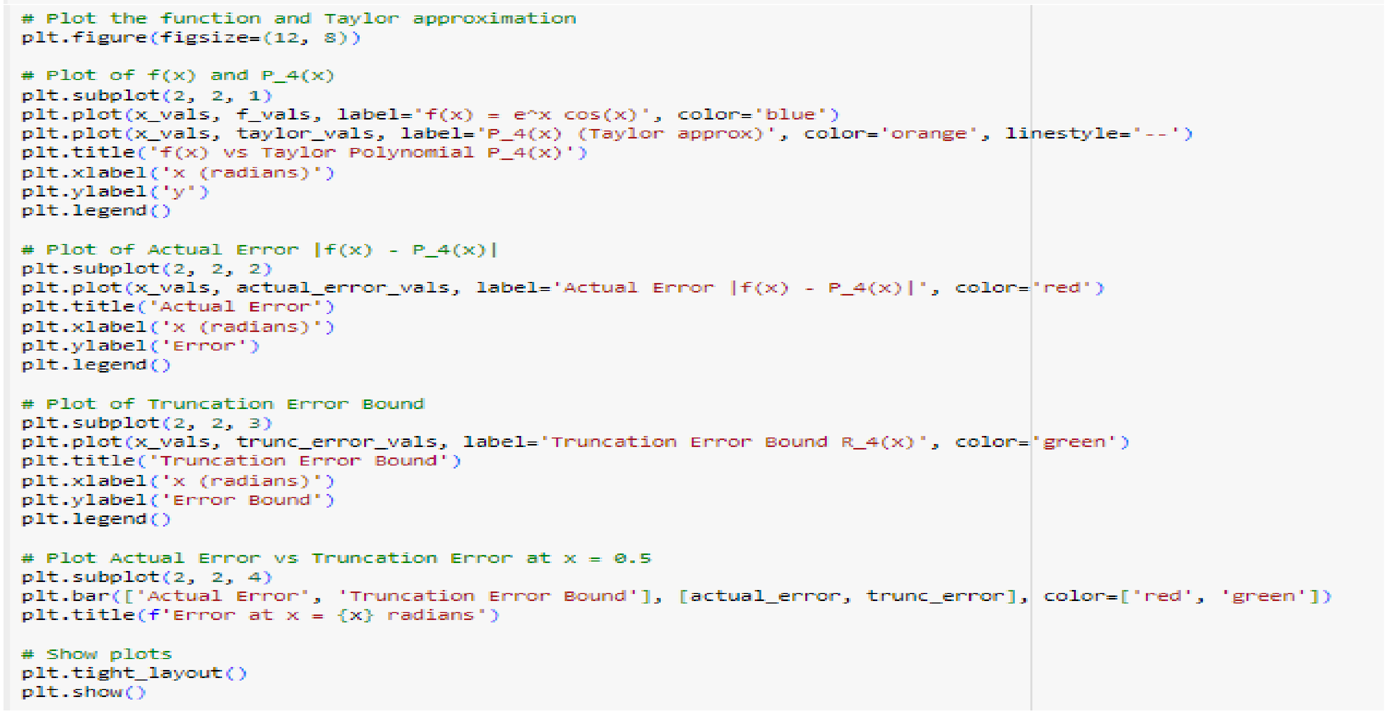
PYTHON CODE

```python
import numpy as np
import matplotlib.pyplot as plt

# Define the function and its derivatives
def f(x):
    return np.exp(x) * np.cos(x)

def f_prime(x):
    return np.exp(x) * (np.cos(x) - np.sin(x))

def f_double_prime(x):
    return np.exp(x) * (2 * np.sin(x) - np.cos(x))

def f_triple_prime(x):
    return np.exp(x) * (3 * np.cos(x) - 3 * np.sin(x))

def f_quadruple_prime(x):
    return np.exp(x) * (4 * np.sin(x) - 4 * np.cos(x))
```

```python
# Taylor polynomial of degree 4 centered at x0
def taylor_poly_4(x, x0):
    return (f(x0) +
            f_prime(x0) * (x - x0) +
            f_double_prime(x0) / 2 * (x - x0)**2 +
            f_triple_prime(x0) / 6 * (x - x0)**3 +
            f_quadruple_prime(x0) / 24 * (x - x0)**4)

# Upper bound for the truncation error
def truncation_error(x, x0):
    return np.exp(x) * abs(5 * np.cos(x) - 5 * np.sin(x)) / 120 * (x - x0)**5

# Set x0 and range of x (radians)
x0 = 0
x_vals = np.linspace(-1, 1, 400)  # x values in radians
x = 0.5  # x for the approximation

# Compute the function values and approximations
f_vals = f(x_vals)
taylor_vals = taylor_poly_4(x_vals, x0)
actual_error_vals = np.abs(f_vals - taylor_vals)
trunc_error_vals = truncation_error(x_vals, x0)
```



```
# Compute function value and error for x = 0.5
actual_value = f(x)
taylor_approx = taylor_poly_4(x, x0)
actual_error = abs(actual_value - taylor_approx)
trunc_error = truncation_error(x, x0)

# Display individual results
print(f"Function value at x = {x} radians: {actual_value}")
print(f"Taylor polynomial approximation at x = {x} radians: {taylor_approx}")
print(f"Actual error at x = {x}: {actual_error}")
print(f"Upper bound of truncation error at x = {x}: {trunc_error}")
```



```
# Plot the function and Taylor approximation
plt.figure(figsize=(12, 8))

# Plot of f(x) and P_4(x)
plt.subplot(2, 2, 1)
plt.plot(x_vals, f_vals, label='f(x) = e^x cos(x)', color='blue')
plt.plot(x_vals, taylor_vals, label='P_4(x) (Taylor approx)', color='orange', linestyle='--')
plt.title('f(x) vs Taylor Polynomial P_4(x)')
plt.xlabel('x (radians)')
plt.ylabel('y')
plt.legend()

# Plot of Actual Error |f(x) - P_4(x)|
plt.subplot(2, 2, 2)
plt.plot(x_vals, actual_error_vals, label='Actual Error |f(x) - P_4(x)|', color='red')
plt.title('Actual Error')
plt.xlabel('x (radians)')
plt.ylabel('Error')
plt.legend()

# Plot of Truncation Error Bound
plt.subplot(2, 2, 3)
plt.plot(x_vals, trunc_error_vals, label='Truncation Error Bound R_4(x)', color='green')
plt.title('Truncation Error Bound')
plt.xlabel('x (radians)')
plt.ylabel('Error Bound')
plt.legend()

# Plot Actual Error vs Truncation Error at x = 0.5
plt.subplot(2, 2, 4)
plt.bar(['Actual Error', 'Truncation Error Bound'], [actual_error, trunc_error], color=['red', 'green'])
plt.title(f'Error at x = {x} radians')

# Show plots
plt.tight_layout()
plt.show()
```