Math Crowdmark Assignment 3 Q3_FarwaRehan

November 9, 2025

[5]:
```
#importing sympy so that code to be written will function and run as needed
import sympy as sp

#using the central difference formula in order to estimate the derivatives of␣
 ↪f'(c)
def central_derivative_definition(f, c):
    return (f(c + 1e-8) - f(c - 1e-8)) / (2 * 1e-8)

#approximating f(x) around a point c using linear, first-order,approximation
def linear_approximation(f, c, E, search_range=2.0, step=1e-4):
    derivative_f = central_derivative_definition(f, c)
    f_at_c = f(c)
    def L(x):
        return f_at_c + derivative_f * (x - c)
    def error_function(x):
        return abs(f(x) - L(x)) - E

#finding x-values/x-intercepts on both the left and the right side
    def roots(start, direction):
        x = start #searching outwards in numeric value until required value is␣
 ↪found
        while abs(x - c) < search_range:
            if error_function(x) * error_function(x + direction * step) < 0:
                a, b = x, x + direction * step
                for i in range(50):
                    middle_value = (a + b) / 2
                    if error_function(a) * error_function(middle_value) < 0:
                        b = middle_value
                    else:
                        a = middle_value
                return (a + b) / 2
            x += direction * step
        return None

    x1 = roots(c, -1)
    x2 = roots(c, +1)
#returning results
```

```
    if x1 and x2:
        return {
            "f_prime": derivative_f,
            "x1": x1, "x2": x2,
            "err1": abs(f(x1) - L(x1)),
            "err2": abs(f(x2) - L(x2)),
         }

#prompting user for needed values
expression = input("Enter function f(x): ").strip()
c = float(input("Enter center point c: "))
E = float(input("Enter target error E: "))
print("For f(x) = ", expression)
print("For the centre point (c): ", c)
print("For the target error value (E): ", E)

x = sp.symbols('x')
math_expression = sp.sympify(expression)
f = sp.lambdify(x, math_expression, "math")

result = linear_approximation(f, c, E)
#printing results if found
if result:
    print("\nResults:")
    print(f"f'(c)   {result['f_prime']:.6f}")
    print(f"x1   {result['x1']:.6f}")
    print(f"x2   {result['x2']:.6f}")
    print(f"|f(x1)-L(x1)|   {result['err1']:.6f}")
    print(f"|f(x2)-L(x2)|   {result['err2']:.6f}")
else:
    print("\nNo valid x1 & x2 found within search range.")
```

```
Enter function f(x):  sin(x)
Enter center point c:  0.7853981634
Enter target error E:  0.05

For f(x) =  sin(x)
For the centre point (c):  0.7853981634
For the target error value (E):  0.05

Results:
f'(c)   0.707107
x1   0.377867
x2   1.142701
|f(x1)-L(x1)|   0.050000
|f(x2)-L(x2)|   0.050000
```

[ ]: