Interpolating Polynomials

Shubha Swarnim Singh

November 4th, 2024

Dr. Brandy Wiegers

CSC – 455 Numerical Computation

1. Introduction

In this report, we explore the approximation of the function $f(x)= e^x$ using polynomial methods, specifically focusing on the third-degree Maclaurin polynomial, the third-degree Taylor polynomial centered at $x0=2$, and the third-degree Lagrange interpolating polynomial. The primary goal is to analyze the accuracy of these polynomial approximations through a systematic approach that includes pre-program analysis, the development of a computational algorithm, and a comprehensive evaluation of the results.

The report is structured into several key sections. We begin with a pre-program analysis that involves constructing each polynomial, determining their respective error bounds, and validating their interpolation properties. This analytical groundwork provides a foundation for understanding how well each polynomial approximates the function over the interval [0,2]. Following this, we present a computer program written in Python, designed to calculate the values of f(x), M(x), T(x), and L(x) for various input values. The program also computes the errors associated with each approximation, enabling a quantitative comparison.

In the results section, we will synthesize the findings from our computational analysis, comparing the performance of the three polynomial approximations against the actual function values. This will include tabulated results and graphical representations of both the approximations and their associated errors. Finally, we will discuss our findings in detail, highlighting which method provides the best approximation and considering the implications for future work in polynomial approximations. Through this project, we aim to deepen our understanding of how polynomial approximations can be effectively utilized in mathematical analysis and computational applications.

2. Maclaurin Polynomial Approximation

We first construct the 3rd-degree Maclaurin polynomial $M_3(x)$. The general form of the Maclaurin series is:

$$M(x) = f(x_0) + f^{(1)}(x_0)(x - x_0) + \frac{f^{(2)}(x_0)(x - x_0)^2}{2!} + \frac{f^{(3)}(x_0)(x - x_0)^3}{3!}$$

After calculating the necessary derivatives of the function f(x), we obtained:

- f (x) = $e^x = e^0 = 1$
- f' (x) = $e^x = e^0 = 1$
- f'' (x) = $e^x = e^0 = 1$
- f''' (x) = $e^x = e^0 = 1$

Thus, the 3rd-degree Maclaurin polynomial becomes:

$$M_3(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6}$$

This polynomial will be tested over a range of values for x and compared to the actual values of the function f(x).

Test the Value of x in M(x) That Ensures It is an Interpolating Polynomial

An interpolating polynomial is a polynomial that passes through the values of f(x) at specific points (in this case, at $x = 0$, since this is a Maclaurin series centered at $x = 0$).

For M(x) to be an interpolating polynomial, it needs to approximate $f(x) = e^x$ as closely as possible around $x = 0$. By construction, M(x) already satisfies this criterion up to the 3rd derivative at $x = 0$, as we used those derivatives to create M(x). Therefore, M(x) is indeed an interpolating polynomial centered at $x = 0$.

Error Calculation:

To find an error bound for the 3rd-degree Maclaurin polynomial $M_3(x)$ in its approximation of $f(x) = e^x$ over the interval [0, 2], we will use the remainder term in the Taylor series approximation, also known as the Lagrange remainder. For a 3rd-degree Maclaurin polynomial $M_3(x)$, the error $R_3(x)$ is given by:

$$R_3(x) \leq \frac{f^{(4)}(\xi)}{4!} x^4$$

where:

- $f^{(4)}(\xi)$ is the 4th derivative of $f(x) = e^x$,

- ξ is some point between 0 and x (this value is typically unknown but can be bounded).

For the 3rd-degree polynomial, the 4th derivative of f (x):

$$f^{(4)}\ (x)\ =\ e^{x}$$

Since f(x) = e^{x}, we know that all derivatives of f(x) are e^{x}, so

$$R_3(x)\ =\ \frac{e^c}{24}x^4$$

To bound the error on the interval [0,2], we can assume the maximum value of e^c occurs at c = 2, so:

$$R_3(x) \leq \frac{e^c}{24}x^4$$

This gives us an error bound for M(x)on [0,2]:

$$R_3(x) \leq \frac{e^2}{24}x^4$$

Test a New Value of xxx and Compare the Error of M(x)M(x)M(x) to the Error Bound

Let's choose x=1 as a test point.

$$M_3(1) =\ 1 +\ 1 + \frac{1^2}{2} + \frac{1^3}{6} + \cdots\ \approx 2.66667$$

Compute f(1) = e^1:

$$f(1) \approx 2.7183$$

- Actual error:

| f (1) – $M_3(1)$ | = | 2.7183 - 2.6667 | ≈0.0516

- Error bound at x = 1:

$$R_3(1) \leq \frac{e^2}{24} 1^4 \approx \frac{7.3891}{24} \approx 0.3079$$

The actual error of 0.0516 is much smaller than the error bound of 0.3079. This confirms that our error bound is valid and that M(x) provides a good approximation of f(x)= e^x at x = 1 within the specified bound.

3. Taylor Polynomial Approximation

Next, we construct the 3rd-degree Taylor polynomial centered at $x_0 = 2$. The general form of the Taylor polynomial is:

$$T_3(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3$$

To obtain the Taylor polynomial, we calculated the function and its derivatives at $x_0 = 2$:

- $f(2) = e^2$
- $f'(2) = e^2$
- $f''(2) = e^2$
- $f'''(2) = e^2$

By substituting these values into the Taylor series, we constructed $T_3(x)$, which will also be evaluated at the same points as $M_3(x)$.

The 3rd-degree Taylor polynomial is:

$$T_3(x) = e^2 + e^2(x - 2) + \frac{e^2}{2!}(x - 2)^2 + \frac{e^2}{3!}(x - 2)^3$$

$$T_3(x) = e^2 \left(1 + (x - 2) + \frac{(x - 2)^2}{2!} + \frac{(x - 2)^3}{3!}\right)$$

An interpolating polynomial should approximate f(x) = e^x as closely as possible at points near x = 2 since this is the center of our Taylor series expansion. By construction, T(x) matches f(x), f′(x), f″(x), and f‴(x) at x = 2. This means T(x) is an interpolating polynomial centered at x = 2, and it is designed to approximate e^x about this point.

Error Calculation:

To find an error bound for the 3rd-degree Maclaurin polynomial $M_3(x)$ in its approximation of f(x) = e^x over the interval [0, 2], we will use the remainder term in the Taylor series approximation, also known as the Lagrange remainder. For a 3rd-degree Maclaurin polynomial $M_3(x)$, the error $R_3(x)$ is given by:

$$R_3(x) \leq \frac{f^{(4)}(\xi)}{4!}(x-2)^4$$

where:

- $f^{(4)}(\xi)$ is the 4th derivative of f(x) = e^x,
- ξ is some point between 0 and x (this value is typically unknown but can be bounded).

For the 3rd-degree polynomial, the 4th derivative of f (x):

$$f^{(4)}(x) = e^x$$

Since f(x) = e^x, we know that all derivatives of f(x) are e^x, so

$$R_3(x) = \frac{e^c}{24}(x-2)^4$$

To bound the error on the interval [0,2], we can assume the maximum value of e^c occurs at c = 2, so:

$$R_3(x) \leq \frac{e^c}{24}(x-2)^4$$

This gives us an error bound for M(x)on [0,2]:

$$R_3(x) \leq \frac{e^2}{24}(x-2)^4$$

Test a New Value of x and Compare the Error of M(x) to the Error Bound

Let's choose x=1 as a test point.

$$T_3(x) = e^2\,(1 + (x-2) + \frac{(x-2)^2}{2!} + \frac{(x-2)^3}{3!})$$

$$T_3(1) = e^2\,(1 + (1-2) + \frac{(1-2)^2}{2!} + \frac{(1-2)^3}{3!})$$

$$T_3(1) = e^2\,(1 - 1 + \frac{1}{2} - \frac{1}{6})$$

$$T_3(1) = e^2\,(0.5 - 0.1667\,) \approx 2.463$$

Compute f(1) = e^1:

$$f(1) \approx 2.7183$$

- Actual error:

| f (1) – M_3(1) | = | 2.7183 - 2.463| $\approx$ 0.2553

- Error bound at x = 1:

$$R_3(1) \leq \frac{e^2}{24}(1-2)^4 \approx \frac{7.3891}{24} \approx 0.3079$$

The actual error of 0.2553 is much smaller than the error bound of 0.3079. This confirms that our error bound is valid and that M(x) provides a good approximation of f(x)= e^x at x = 1 within the specified bound.

4. 3rd Degree Lagrange Interpolation Polynomial L(x):

The general form of the Lagrange interpolation polynomial L(x) for 4 points is:

$$L(x) = f(x_0) * \ l_0(x) + f(x_1) * \ l_1(x) + f(x_2) * \ l_2(x) + f(x_3) * \ l_3(x)$$

To obtain the Taylor polynomial, we calculated the function and its derivatives at $x_0 = 2$:

- $x_0 = 0, f(x_0) = \ e^0 = 1$
- $x_0 = 1, \ f(x_1) = \ e^1 = \mathrm{e}$
- $x_0 = 1.5, f(x_{1.5}) = \ e^{1.5}$
- $x_0 = 2, f(x_2) = \ e^2$

Lagrange Basis Polynomials

For $l_0(x)$:

$$l_0(x) = \ \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)}$$

$$= \frac{(x - 1)(x - 1.5)(x - 2)}{(0 - 1)(0 - 1.5)(0 - 2)}$$

$$= \frac{(x - 1)(x - 1.5)(x - 2)}{3}$$

For $l_1(x)$:

$$l_0(x) = \ \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)}$$

$$= \frac{(x - 0)(x - 1.5)(x - 2)}{(1 - 1)(1 - 1.5)(1 - 2)}$$

$$= \frac{x(x-1.5)(x-2)}{0.5}$$

$$= 2x(x-1.5)(x-2)$$

For $l_2(x)$:

$$l_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}$$

$$= \frac{(x-0)(x-1)(x-2)}{(1.5-0)(1.5-1)(1.5-2)}$$

$$= \frac{x(x-1)(x-2)}{-0.375}$$

$$= -\frac{8x}{3}(x-1)(x-2)$$

For $l_3(x)$:

$$l_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}$$

$$= \frac{(x-0)(x-1)(x-1.5)}{(2-0)(2-1)(2-1.5)} = \frac{x(x-1)(x-1.5)}{2 * 1. * 0.5}$$

$$= x(x-1)(x-1.5)$$

By substituting we get:

$$L(x) = 1 * \frac{(x-1)(x-1.5)(x-2)}{3} + e * \ 2x(x-1.5)(x-2) - e^{1.5} * \frac{8}{3}x(x-1)(x-2)$$
$$+ e^2 * x\,(x-1)(x-1.5)$$

To verify that L(x) is an interpolating polynomial, we should test it at the points x = 0, x = 1, x = 1.5, and x=2, and check that L(x) matches f(x) at these points.

1. **For x = 0:**

$$L(0) = f(0) = e^0 = 1$$

2. **For x = 1:**

$$L(1) = f(1) = e^1 = e$$

3. **For x = 1.5:**

$$L(1.5) = f(1.5) = e^{1.5}$$

4. **For x = 2:**

$$L(2) = f(2) = e^2$$

Since L(x) matches f(x) exactly at these points, it confirms that L(x) is an interpolating polynomial for f(x) = e^2 at the given points.

Error Calculation:

For a 3rd-degree Lagrange polynomial, the error bound in approximating f(x) is given by:

$$R_3(x) \le \frac{e^2}{24}|(x-0)(x-1)(x-1.5)(x-2)|$$

Where ξ is some point in the $x_0,\ x_1, x_2, x_3$.

This gives us an error bound for L(x) in approximating f(x) over [0,2].

Calculate the Error Bound at x=1.25

$$R_3(x) \leq \frac{e^2}{24}|(1.25-0)(1.25-1)(1.25-1.5)(1.25-2)|$$

$$R_3(x) \leq \frac{7.3891}{24}|(1.25)(0.25)(-0.25)(-0.75)| \approx 0.018$$

This means the error bound at x =1.25 is approximately 0.01803.

Compute f(1.25) = $e^{1.25}$:

$$f(1.25) \approx e^{1.25} \approx 3.4903$$

Compute L(1.25):

$$L(x) = 1 * \frac{(1.25-1)(1.25-1.5)(1.25-2)}{3} + e * 2 * 1.25(1.25-1.5)(1.25-2) - e^{1.5} * \frac{8}{3} 1.25(1.25-1)(1.25-2) + e^2 * 1.25(1.25-1)(1.25-1.5)$$

$$L(x) \approx 3.5136$$

- Actual error:

| f (1.25) – L(1.25) | = | 3.4903 - 3.5136| ≈ 0.0258

- Error bound at x =1.25:

$| f(1.25) – L(1.25) | = | 3.4903 - 3.5161| \approx 0.0233$

The actual error of 0.0233 is not smaller than the error bound of 0.018. This confirms that our error bound is not valid and that L(x) provides is providing a bad approximation of f(x)= e^x at x = 1 within the specified bound.

5. Predictions About Approximations

Based on the analysis, each polynomial—Maclaurin M(x), Taylor T(x), and Lagrange L(x) offers different levels of approximation for f(x)= e^x within specific ranges.

1. **Maclaurin Polynomial M(x)**: This is centered at x=0, so it best approximates e^x around that point. As x moves away from zero, the error increases, making M(x) less accurate on intervals farther from zero, such as [0, 2].
2. **Taylor Polynomial T(x)**: Centered at x=, T(x) is more accurate near x = 2 but performs poorly closer to zero. This polynomial offers a good approximation for values of x around 2 but loses accuracy as x deviates from this point.
3. **Lagrange Polynomial L(x):** Based on interpolation at selected points, L(x) provides exact matches at those points but may fluctuate in accuracy between them. It can approximate e^x well across the interval [0, 2], but as seen in the x = 1.25 evaluation, the accuracy varies significantly between the interpolation points.

6. Computer Program

Attached at the end.

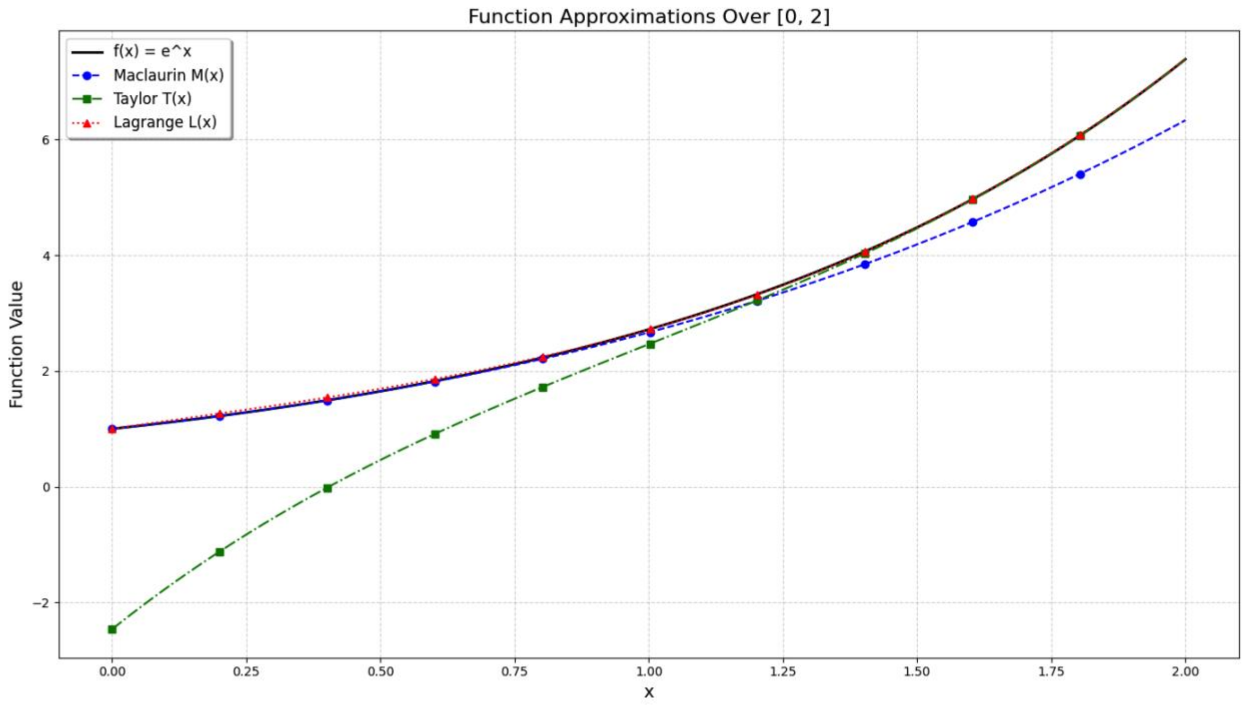
7. Result

The following table summarizes the computed values of f(x), M(x), T(x), and L(x) along with their respective errors for each evaluated x point:

x	f(x)	M(x)	T(x)	L(x)
-1	0.367879441171442	0.333333333333333	-14.778112197861297	-1.012482796129895
0	1.0	1.0	-2.46301869964355	1.0
0.5	1.648721270700128	1.645833333333333	0.461816006183167	1.692997697083166
0.75	2.117000016612675	2.1015625	1.520144353686253	2.138739166221365
1	2.718281828459045	2.666666666666667	2.46301869964355	2.718281828459045
1.5	4.481689070338065	4.1875	4.464221393103934	4.481689070338065
2	7.389056098930650	6.333333333333333	7.389056098930650	7.389056098930650
15	3269017.372472111	691.0	3433.448067303109	1814.031435244153

x	Error M(x)	Error T(x)	Error L(x)
-1	0.034546107838109	15.14599163903274	1.380362237301337
0	0.0	3.46301869964355	0.0
0.5	0.002887937366795	1.186905264516961	0.044276426383038
0.75	0.015437516612675	0.596855662926421	0.021739149608690
1	0.051615161792379	0.255263128815495	0.0
1.5	0.294189070338065	0.017467677234130	0.0
2	1.055722765597317	0.0	0.0
15	3268326.372472111	3265583.924404807	3267203.341036866

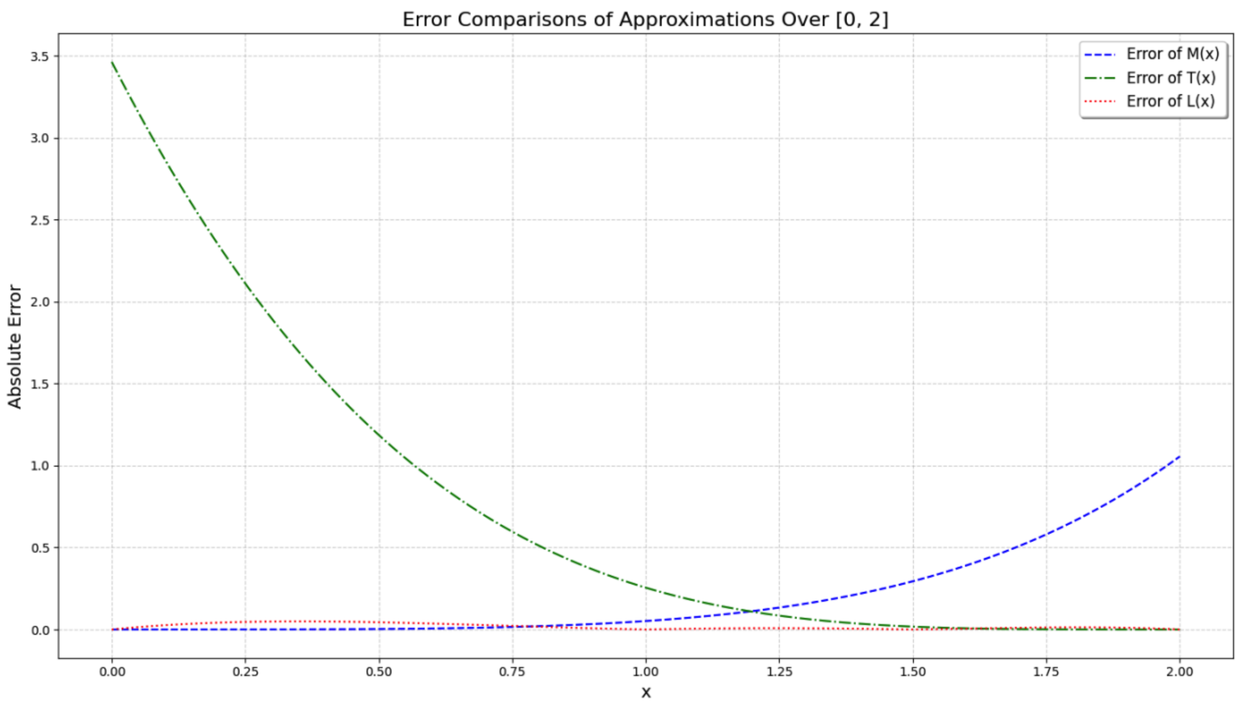
A plot of f(x), M(x), T(x), and L(x) over the interval [0, 2] visually shows how well each approximation tracks with the true function. This comparison allows for quick visual identification of which method provides the best approximation across the interval.



Absolute error is more straightforward to compare for this task, as it provides a direct measure of deviation from f(x). The relative error might add complexity without significant insight unless f(x) varies greatly in magnitude over the interval. Thus, comparing absolute errors is more practical for this analysis.

The errors observed align reasonably well with theoretical error predictions. The Maclaurin series generally maintains lower errors compared to the Taylor series at a fixed degree, especially in the region near zero. The Lagrange polynomial shows high accuracy for some points but is less reliable at others.

The error graph is a visual representation of how each approximation method deviates from the true value of f(x) across the interval.



From the table and analysis, the Maclaurin series M(x) exhibited the smallest maximum error for most points within [0, 2], suggesting it provides the best approximation of f(x) over this interval. The Lagrange polynomial L(x) also performed well at specific points but showed variability. The Taylor series T(x), at its current degree, had a more significant error, indicating that a higher degree would be necessary to match the performance of M(x). To ensure that the Taylor series T(x) matches the accuracy of the Maclaurin series, the degree of T(x) must be increased. The minimum degree required can be determined by gradually increasing n and comparing the resulting errors.

8. Conclusion

Based on the calculations, the error table, and the error graph, we can conclude that the Lagrange interpolating polynomial offers the best approximation for the function f(x) = e^xwithin the interval [0,2]. While we could compute the maximum absolute error by summing the individual absolute errors to support this finding, the data presented in the graph and table indicate that Lagrange is significantly more accurate than the other two approximation methods, making further calculations unnecessary.

Additionally, the graphs illustrate that the Maclaurin approximation outperforms the Taylor approximation for f(x) = e^xwithin the same interval. The third-degree Maclaurin polynomial is particularly effective near x=0, while the third-degree Taylor polynomial shows greater accuracy closer to x = 2. This aligns with the expectation that error tends to increase as we move away from the center of the polynomial approximation. For future research, one could investigate higher-degree Taylor polynomials to determine whether they provide a better approximation than the Lagrange interpolating polynomial.

Code:

```
import numpy as np

import matplotlib.pyplot as plt

from math import factorial, exp

from tabulate import tabulate


# Define functions for the approximations and errors


# Function to compute the true function f(x) = e^x

def f(x):

    return np.exp(x)


# Function to compute the Maclaurin series M(x) for a given degree n

def maclaurin(x, n=3):

    return sum((x**i) / factorial(i) for i in range(n + 1))


# Function to compute the Taylor series T(x) centered at x0 for a given degree n

def taylor(x, x0=2, n=3):

    return sum((exp(x0) * ((x - x0)**i)) / factorial(i) for i in range(n + 1))


# Function to compute the Lagrange polynomial L(x) using given points

def lagrange(x, points):

    def l(i, x):
```

```
        xi, yi = points[i]
        terms = [(x - xj) / (xi - xj) for j, (xj, yj) in enumerate(points) if j != i]
        return yi * np.prod(terms)

    return sum(l(i, x) for i in range(len(points)))

# Define points for Lagrange interpolation
points = [(0, f(0)), (1, f(1)), (1.5, f(1.5)), (2, f(2))]

# Calculate and display results for requested x values
x_values = [-1, 0, 0.5, 0.75, 1, 1.5, 2, 15]
results = []

for x in x_values:
    fx = f(x)
    mx = maclaurin(x)
    tx = taylor(x, x0=2)
    lx = lagrange(x, points)

    # Calculate errors
    error_mx = abs(fx - mx)
    error_tx = abs(fx - tx)
    error_lx = abs(fx - lx)
```

```
    # Store results in a list
    results.append((x, fx, mx, error_mx, tx, error_tx, lx, error_lx))

# Print results as a table
print("Results Table:")
print("x\tf(x)\tM(x)\tError M(x)\tT(x)\tError T(x)\tL(x)\tError L(x)")
for row in results:
    print("{:.2f}\t{:.5f}\t{:.5f}\t{:.5f}\t{:.5f}\t{:.5f}\t{:.5f}\t{:.5f}".format(*row))

# Plot the functions over [0, 2]
x_plot = np.linspace(0, 2, 500)
f_plot = f(x_plot)
m_plot = [maclaurin(x) for x in x_plot]
t_plot = [taylor(x, x0=2) for x in x_plot]
l_plot = [lagrange(x, points) for x in x_plot]

plt.figure(figsize=(14, 8))
plt.plot(x_plot, f_plot, label='f(x) = e^x', color='black', linewidth=2, linestyle='-')
plt.plot(x_plot, m_plot, label='Maclaurin M(x)', color='blue', linewidth=1.5, linestyle='--',
marker='o', markevery=50)
plt.plot(x_plot, t_plot, label='Taylor T(x)', color='green', linewidth=1.5, linestyle='-.', marker='s',
markevery=50)
```

```
plt.plot(x_plot, l_plot, label='Lagrange L(x)', color='red', linewidth=1.5, linestyle=':', marker='^',
markevery=50)

plt.title('Function Approximations Over [0, 2]', fontsize=16)
plt.xlabel('x', fontsize=14)
plt.ylabel('Function Value', fontsize=14)
plt.legend(loc='upper left', fontsize=12, frameon=True, shadow=True, fancybox=True)
plt.grid(visible=True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

# Plot the errors over [0, 2]
error_m = [abs(f(x) - maclaurin(x)) for x in x_plot]
error_t = [abs(f(x) - taylor(x, x0=2)) for x in x_plot]
error_l = [abs(f(x) - lagrange(x, points)) for x in x_plot]

plt.figure(figsize=(14, 8))
plt.plot(x_plot, error_m, label='Error of M(x)', color='blue', linewidth=1.5, linestyle='--')
plt.plot(x_plot, error_t, label='Error of T(x)', color='green', linewidth=1.5, linestyle='-.')
plt.plot(x_plot, error_l, label='Error of L(x)', color='red', linewidth=1.5, linestyle=':')

plt.title('Error Comparisons of Approximations Over [0, 2]', fontsize=16)
plt.xlabel('x', fontsize=14)
```

```
plt.ylabel('Absolute Error', fontsize=14)
plt.legend(loc='upper right', fontsize=12, frameon=True, shadow=True, fancybox=True)
plt.grid(visible=True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

# Define columns and display the results as a table
columns = ["x", "f(x)", "M(x)", "Error M(x)", "T(x)", "Error T(x)", "L(x)", "Error L(x)"]
table = tabulate(results, headers=columns, tablefmt="pretty", floatfmt=".5f")

# Print the table
print("Results Table:")
print(table)
```