# Iteration and Error

David Tran and Spencer Kelly

January 28, 2024

## Abstract

The following series of labs explore different algorithms in numerical analysis, as well as their applications. We offer an exploratory look on each of the algorithms, including notes on its implementation, properties of the algorithms, and the practical effects of its implementation on real-world systems.

This lab introduces the concept of iteration in numerical algorithms through use of Taylor series. Through computations, we show how well the Taylor series approximates a function, and additionally quantify the error in its approximation. We observe and discuss the effects of truncation and precision or rounding errors which occur due to the limitations of hardware, and show how we can measure and differentiate between the two.

## Introduction

In the field of numerical analysis, one of the first important concepts introduced is the idea of approximating otherwise complex and computationally expensive to evaluate functions using much simpler and efficient methods. In this lab, the first of four, we delve into the approximations of functions using their Taylor series and their associated Taylor remainders, exposing the reader to introductory concepts in functional approximation.

Taylor series are a powerful tool in mathematics, and provide us with a means of representing a smooth function using an infinite series of polynomial terms. These series allow us to work with otherwise difficult functions, and have applications in a variety of fields including physics, engineering, and quantitative finance, where they aid in modeling complex systems, solving differential equations, and the analysis of complex functions, to name a few.

For the purposes of computation, it is not realistic to use the infinite taylor series for a given function. In this lab, we introduce the truncated (finite terms) Taylor series for a single function. We will compare the truncated Taylor series to the function in which it approximates, and analyze the difference between the two, over various domains. We will also analyze the Taylor remainder, which represents the error associated with truncation to finite terms. By analyzing the differing number of terms before truncation and plotting the Taylor remainder, we can observe how the degree of truncation effects the accuracy of the approximation, providing insight into the tradeoff between efficiency and accuracy in the field of numerical analysis.

The field of numerical analysis in some form dates back over 2000 years ago in ancient Egypt, and there is evidence to suggest that Babylonians may have used numerical methods to calculate the square root of 2 to within 5 decimal places. Though using more modern methods, and with the help of the computational software MATLAB, this lab serves as an introduction to methods of computation and approximation in numerical analysis that have been of great use across many fields and civilizations.

## 1 Taylor series and error

### 1.0 Problem 0

Let $f(x) = e^{-x}$. Note that $f^{(1)}(x) = -e^{-x}, f^{(2)}(x) = e^{-x}$, and in general, $f^{(n)}(x) = (-1)^n e^{-x}$. So, $f^{(n)}(0) = (-1)^n$, and the Taylor series expansion around $x_0 = 0$ is given by

$$\begin{aligned} T_n(x) &= \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!} x^k \\ &= \sum_{k=0}^{n} \frac{(-1)^k}{k!} x^k \end{aligned}$$

Similarily, the remainder term is

$$\begin{aligned} R_n &= \frac{f^{(n+1)}(z)}{(n+1)!}(x - x_0)^{n+1} \\ &= \frac{(-1)^{n+1} e^{-c}}{(n+1)!} x^{n+1}. \end{aligned}$$

for some $0 \leq c \leq x$.

### 1.1 Problem 1

a. We have

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \dots$$

so

$$\begin{aligned} F(x) &= \frac{e^{-x} - 1 + x}{x^2} \\ &= \frac{1}{2!} - \frac{x}{3!} + \frac{x^2}{4!} - \dots \\ &= \sum_{k=0}^{\infty} (-1)^k \frac{x^k}{(k+2)!} \end{aligned}$$

and its Taylor series up to $n + 1$ terms is

$$T_n(x) = \sum_{k=0}^{n} (-1)^k \frac{x^k}{(k+2)}$$

To find the remainder term $R_n$, we use the fact that $R_n = F(x) - T_n(x)$.

b. We know that the Taylor remainder is in general written as

$$\frac{f^{k+1}(c)}{(k+1)!}(x - x_0)^{k+1}$$

So that, in this case, we end up with our taylor series of $n+1$ terms having remainder term:

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

This tells us that the bound for the truncation error can be written as

$$\frac{(|x|)^{n+1}}{((n+1)+2)!}$$

### 1.2 Problem 2

We compute and plot $F(x)$ from above with the following.

```matlab
% Evaluate F(x) using the first n terms of its Taylor series.
function T = FTaylor(x, n)
    % Initialize sum as 0
    T = 0;
    % Loop over terms in series
    for k = 0:n
        T = T + (-1)^k .* x.^k ./ factorial(k + 2);
    end
end
```

FTaylor.m

```matlab
% Plot F and its Taylor series approximation up to the nth degree term
function plotF(x, n)
    F_values = F(x);
    FTaylor_values = FTaylor(x, n);

    figure;
    plot(x, F_values, 'b', 'LineWidth', 2, 'DisplayName', 'F(x)');
    hold on;
    plot(x, FTaylor_values, 'r--', 'LineWidth', 2, 'DisplayName', 'Taylor Series')
    ;
    xlabel('x');
    ylabel('y');
    title('F(x) and its Taylor Series');
    legend('Location', 'Best');
    grid on;
    hold off;
end
```

plotF.m

giving the graph in Figure 1:

### 1.3 Problem 3

Plotting the graph from Problem 2 using instead a range of $(-10^{-7}, 10^{-7})$ yields Figure 2. Observe the absolute and relative errors over the interval $(-10^{-6}, 10^{-6})$ in Figure 3. We use the truncation error term in Problem 1 as an upper bound on truncation error. If we plot this alongside the absolute and relative errors as in Figure 4, we see the absolute and relative errors are well above the theoretical truncation error. Also note that the truncation error decreases as we approach 0

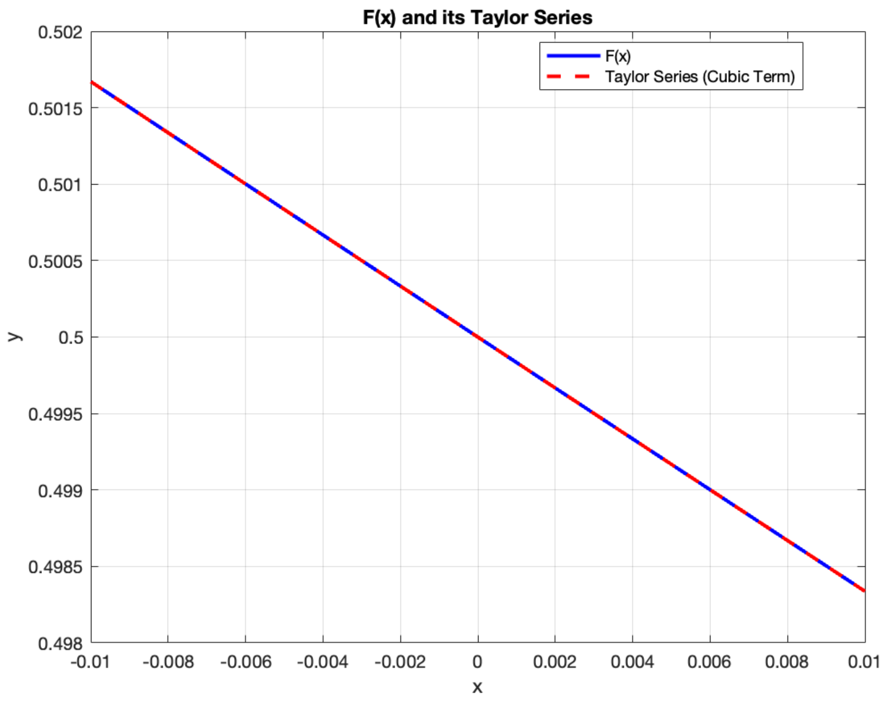


Figure 1: $F(x)$ and its Taylor Series from $(-0.01, 0.01)$

and increases as we move away from it, since our Taylor series expansion is centered at $x = 0$. Because the absolute and relative errors are above the truncation error, the problem is roundoff error rather than truncation error. This is because the order of magnitude of the values we are working with $(10^[-6])$ are so small that we approach the machine epsilon when we take powers of these values. The limited precision of floating-point numbers mean that we can't represent these values as precisely. Indeed, as we get small enough, we can *underflow*: where the number is smaller than our floating-point representation can represent. This lack of floating-point precision is what is making the roundoff errors so large.

So, the directly computed function is more accurate. We know the closed form of $F(x)$, so we should use it, rather than approximate the function with the terms of a Taylor series, since, at values of such a small magnitude, it introduces more roundoff error.

### 1.4 Problem 4

Below in Figure 5 is the plot of the domain from problem 2 done using the function myfofxv2.m, which uses a taylor series expansion to plot x values between $-10^{-7}$ and $10^{-7}$. If the value falls not between those bounds, then the functions opts for the original form of the function.

Below in Figure 6 is the plot of the domain from problem 3, using the same function:

## 2 Summary and Conclusions

In this lab, we used the Taylor series the approximate a function whose closed-form we knew. We plotted the approximation alongside the closed form to observe how well the Taylor series was able to approximate, as well as where it fell short. We found both analytically and empirically the

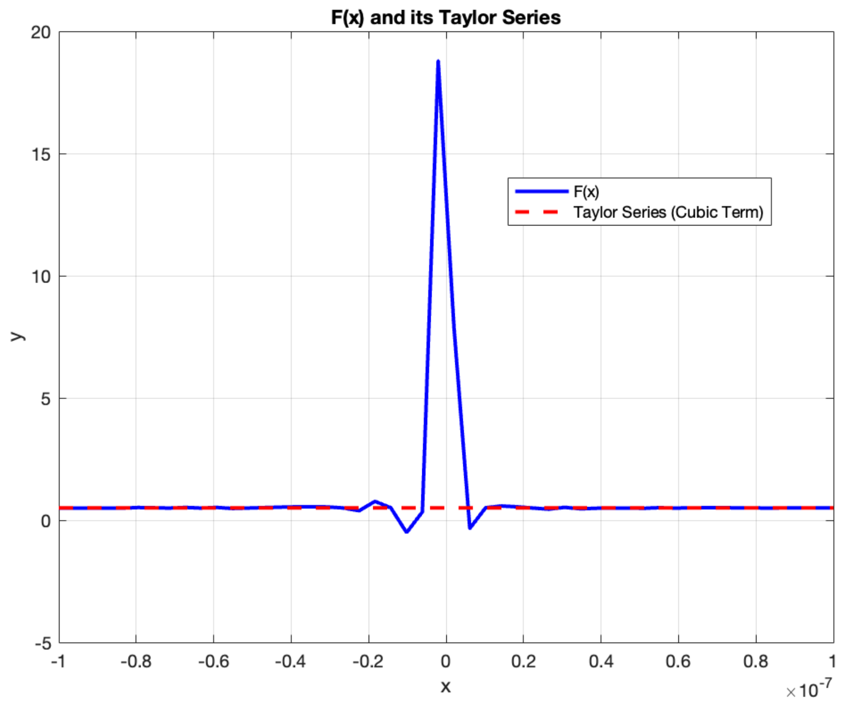


Figure 2: $F(x)$ and its Taylor Series from $(-10^{-7}, 10^{-7})$ up to cubic terms

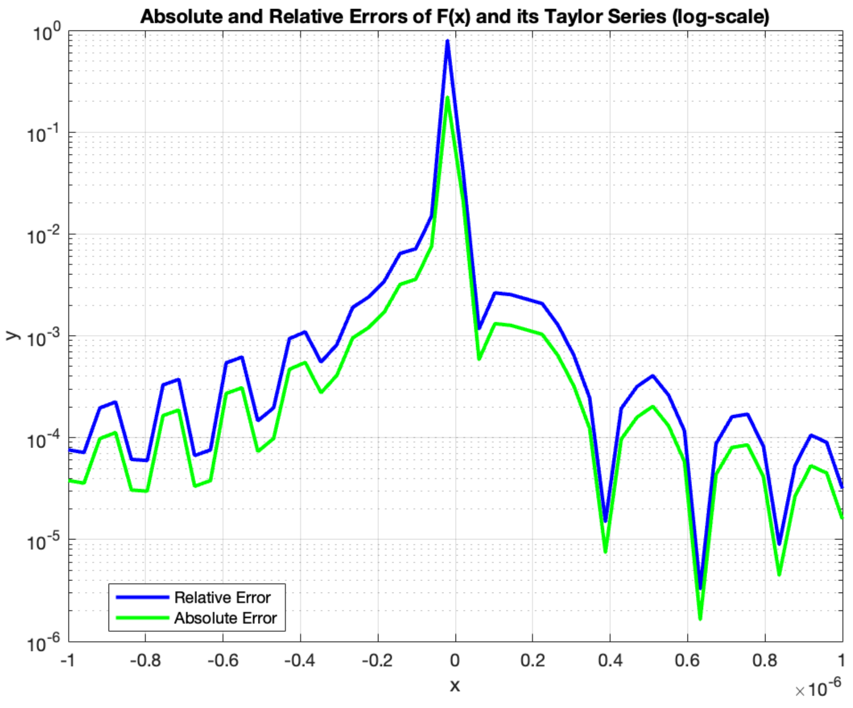


Figure 3: Error between $F(x)$ and its Taylor Series

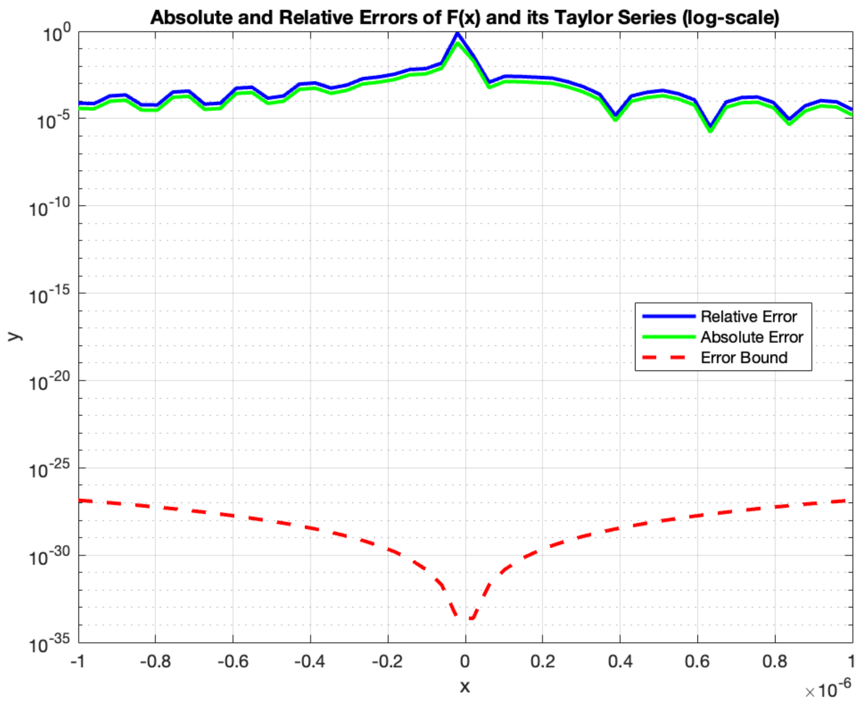


Figure 4: Error between $F(x)$ and its Taylor Series and truncation bound

expected error due to truncation of the series, and discovered that roundoff errors dominated the error over truncation errors at very small values of $x$.

The method of using a Taylor series to approximate a function is very applicable and often used to compute values for functions whose closed form we do not know. For example, in many programming languages, the trigonometric functions are defined in terms of their power series, and 3 to 4 terms are usually used as enough to achieve the maximum-allowed precision due to floating-point representation. For future work, it wouuld be good to explore the threshold at which roundoff error begins to dominate truncation error, as well as discover possible workarounds for approximating functions with Taylor series at very small input values.

## Teamwork Statement

There were minimal issues encountered as we worked on this lab as a team. Both lab members were well-equipped with knowledge on the theory pertaining to Taylor series and their applications. Lab members were also experienced with the applications of Taylor series, and had experience with projects on numerical methods and their applications to other fields of work including modelling of stellar atmospheres, modelling of thermodynamic and electromagnetic systems, and the optimization of algorithms. Overall, there were little to no problems that arose in terms of teamwork on this project.

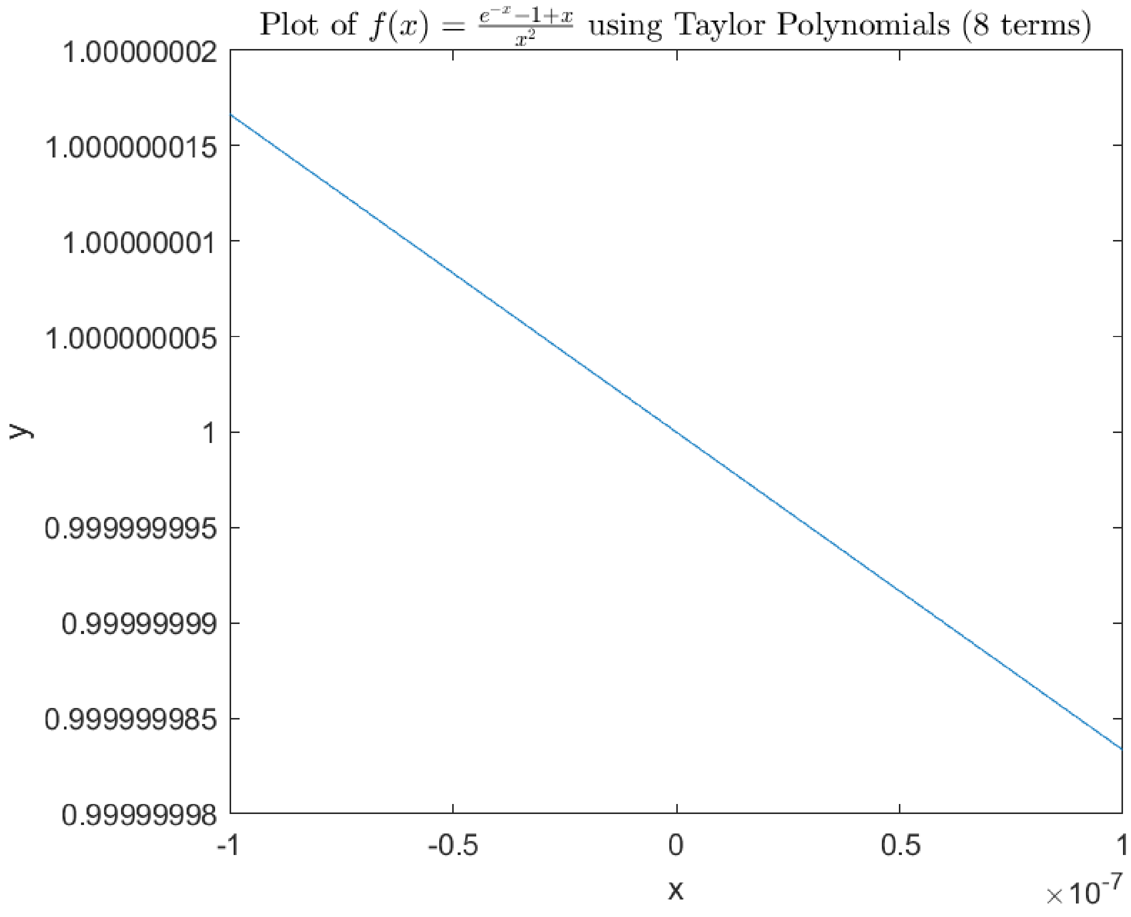


Figure 5: Plot of domain from Problem 2

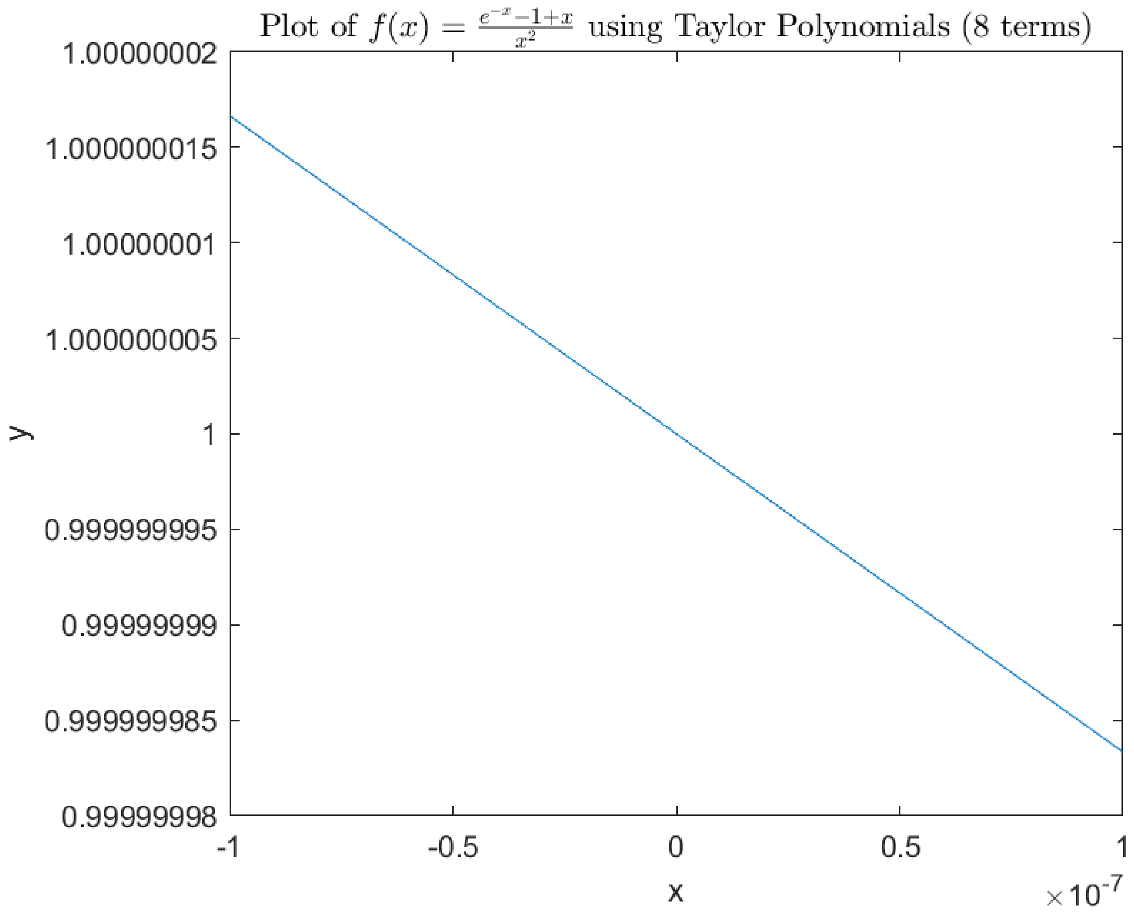


Figure 6: Plot of domain from Problem 3

## Code Appendix

```

function fx =  myfofxv2(x)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
if -10^(-6) < x & x < 10^(-6)
    fx = lab1bP2v2(x,8);
else
    fx = myfxv2(x);
end
end
```

myfofxv2.m

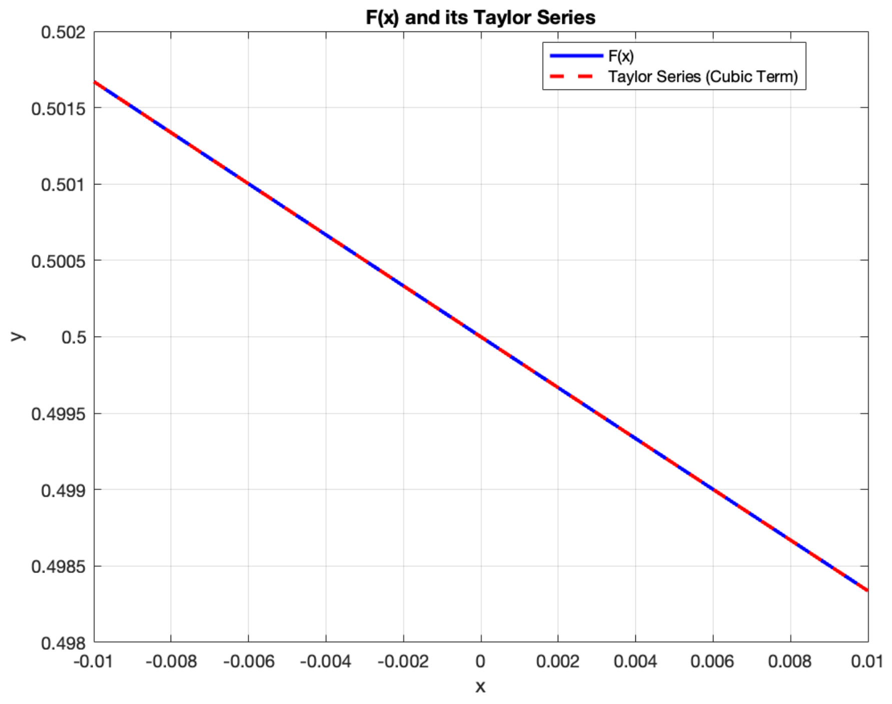
```
% Plot the relative and absolute error between F and its Taylor series approx.
    with n terms
function plotFTaylorError(x, n, showErrorBound)
    % Evaluate F(x) and its Taylor series up to the quartic term
    F_values = F(x);
    FTaylor_values = FTaylor(x, n);

    absolute_error = abs((F_values - FTaylor_values));
    relative_error = absolute_error ./ F_values;
    error_bound = abs(x .^ (n + 1) ./ (factorial(n + 3)));

    figure;
    plot(x, relative_error, 'b', 'LineWidth', 2, 'DisplayName', 'Relative Error');
    hold on;
    plot(x, absolute_error, 'g', 'LineWidth', 2, 'DisplayName', 'Absolute Error');
    hold on;
    if showErrorBound
        plot(x, error_bound, 'r--', 'LineWidth', 2, 'DisplayName', 'Error Bound');
    end
    xlabel('x');
    ylabel('y');
    set(gca, 'YScale', 'log');

    title('Absolute and Relative Errors of F(x) and its Taylor Series (log-scale)'
    );
    legend('Location', 'Best');
    grid on;
    hold off;
end
```

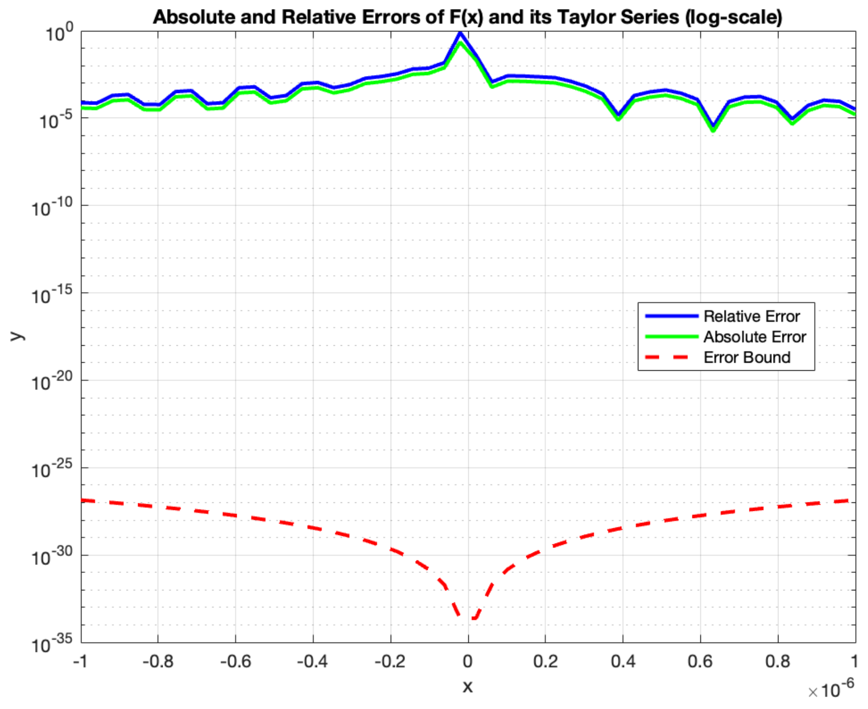
plotFTaylorError.m

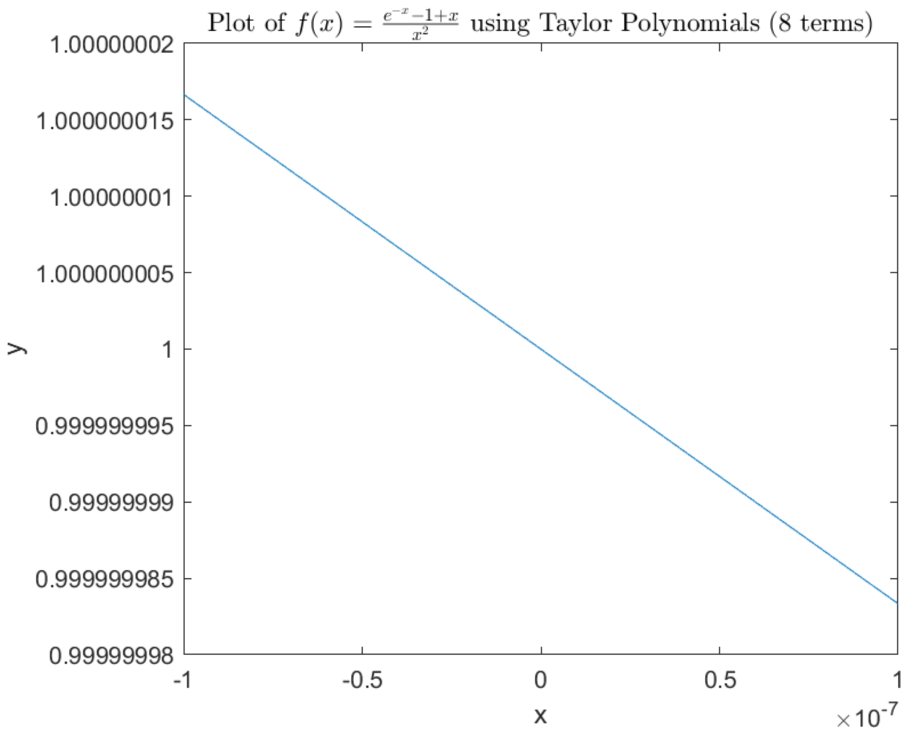
```
function y = myfxv2(x)
         y = (exp(-x)+x-1)./(x.^2);
end
```

myfxv2.m

## Plot Appendix







## References

[1] Encyclopædia Britannica, inc. *Historical background*. `https://www.britannica.com/science/numerical-analysis/Historical-background`.

[2] Garcia, A. L. *Numerical Methods for Physics*. CreateSpace, 2015.

[3] Leblanc, F. *An Introduction to Stellar Astrophysics*. John Wiley & Sons, 2010.

[4] King, G. C. *Vibrations and Waves*. John Wiley & Sons, 2010.

[5] Sauer, T. *Numerical Analysis*. Pearson, 2019.