

Mini project 2 - Convergence of Taylor Series for the Error Function

Jyoutir 40360629

October 15, 2024

OBJECTIVE: This report investigates the convergence behavior of the Taylor series approximation for the error function $\text{erf}(z)$, focusing on how the number of terms required for convergence depends on the value of z . By comparing the absolute difference between the Taylor polynomial approximation and the reference values provided by Python's `math.erf()` function [1], the report quantifies the accuracy of the approximation. This analysis shows the balance between computational efficiency and accuracy in numerical calculations involving series expansions.

SUMMARY OF PROCEDURE: The error function, $\text{erf}(z)$, is defined as follows:

$$\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt \quad (1)$$

To approximate $\text{erf}(z)$, its Taylor series expansion about the origin is used:

$$P_N(z) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^N \frac{(-1)^n z^{2n+1}}{n!(2n+1)} \quad (2)$$

Here, N represents the number of terms in the approximation.

The accuracy of this approximation was evaluated by computing the absolute difference between $P_N(z)$ from Equation 2 and the reference value of $\text{erf}(z)$ obtained from Python's `math.erf()` function. For increasing values of N , the process was repeated until the difference fell below a specified tolerance ϵ .

For each value of $z \in \{0.1, 1.0, 2.0, 3.0\}$, the convergence was examined using tolerances $\epsilon = 10^{-15}$ for $z = 0.1, 1.0, 2.0$ and $\epsilon = 10^{-13}$ for $z = 3.0$. These tolerances account for limitations in floating-point precision and increased computational error for larger z values. The absolute differences were plotted against the number of terms N on a semi-logarithmic graph to illustrate the convergence behavior for each z .

RESULTS AND DISCUSSION: Figure 1 shows the convergence of the Taylor series approximation for $\text{erf}(z)$ at different values of z . The y-axis represents the absolute difference between the Taylor approximation and the reference value from `math.erf()`, plotted on a logarithmic scale, while the x-axis represents the number of terms, N , in the Taylor series.

Several trends are evident as both the polynomial order and the value of z increase:

- For small values of z (e.g., $z = 0.1$), the series converges rapidly, requiring fewer than 10 terms to achieve high accuracy. This occurs because higher-order terms become negligibly small quickly due to the small magnitude of z .
- At $z = 1.0$, the series converges more slowly, with around 15 terms needed to reach the same accuracy as for $z = 0.1$, as higher-order terms contribute more significantly.
- For $z = 2.0$, about 30 terms are required for similar precision, with the larger z^{2n+1} terms outweighing the factorial growth in the denominator, slowing convergence.
- The slowest convergence is observed at $z = 3.0$, where over 40 terms are necessary. Here, series terms decrease more gradually, and floating-point precision errors become noticeable.

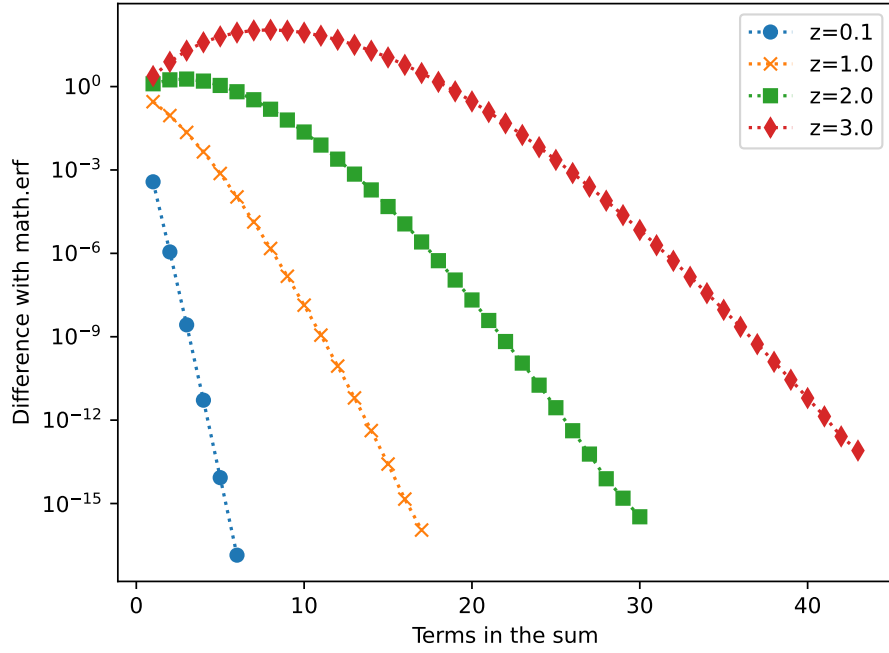


Figure 1: Convergence of Taylor polynomial for $\text{erf}(z)$, as defined in Equation 1, for different values of z . The figure shows the absolute difference between the Taylor approximation from Equation 2 and the standard $\text{erf}(z)$ as a function of the number of terms in the Taylor series.

These trends can be explained by the remainder term from Taylor's theorem [2]:

$$R_N(z) = \frac{f^{(N+1)}(\xi)}{(N+1)!} z^{N+1} \quad (3)$$

where ξ is between 0 and z . As z increases, the term z^{N+1} grows, and although the factorial $(N+1)!$ in the denominator increases rapidly, the numerator dominates for larger z and small N , leading to a larger remainder $R_N(z)$. The higher derivatives $f^{(N+1)}(\xi)$ of e^{-t^2} include polynomials of degree $2N+2$ multiplied by $e^{-\xi^2}$, which increase with N , affecting convergence.

Thus, for larger z , more terms are needed to reduce the remainder term to given tolerance, hence slower convergence.

Conclusively, Taylor series for $\text{erf}(z)$ converges rapidly for small z but slows for larger z , requiring more terms for the same accuracy. This behaviour aligns with Taylor's theorem, where the remainder term increases with both z and N . My results show the trade-off between computational efficiency and precision, particularly for larger values of z , and shows the importance of choosing appropriate numerical methods.

REFERENCES

- [1] Python Software Foundation. *Python 3.10.8 Documentation - math.erf*. <https://docs.python.org/3/library/math.html#math.erf>. Accessed: 15th October 2024.
- [2] Wikipedia. *Taylor's theorem*. https://en.wikipedia.org/wiki/Taylor%27s_theorem. Accessed: 15th October 2024.