35006 - Numerical Analysis Final Project

Zachary Zerafa - 24557656

October 29, 2025

1 Introduction

Often mathematical problems arise in the form of *closed contour integrals*, of which one can prove is equal to the sum of a set of quantities called *residues*, caled by $2\pi i$. A *residue calculator* would offer tremendous efficiency for those who need to calculate a nontrivial contour integral, or compute several contour integrals on some domain; a user can use this calculator to cherrypick the related values to sum and cale by $2\pi i$.
This report discusses the theory behind such a program, the numerical methods implemented to create said program, and analyzes it's error, computational complexity, and robustness.

2 Assumptions

It is assumed that all functions that the residue calculator works on are *meromorphic*; this is the 'sufficiently nice' condition required for the theory of residues to be useful. This means that the algorithm developed relies on the security that the function is complex differentiable on the entire domain specified to the algorithm (this also implies that it is integrable by the properties of complex analysis).
We discuss the general theory related to residues to help elucidate the different problems in numerical analysis that this method will have to solve. As preliminary notation, we will assume that $f : \Omega \to \mathbb{C}$ is a meromorphic function on the complex rectangle Ω (i.e $\Omega = \{x + iy : x \in [a, b], y \in [c, d]\}$). The numerical method will be denoted as a function M, whose input is a meromorphic function, tolerance, bracket refinement integer, and a complex rectangle, and whose output is a set of ordered pairs of a pole and its residue respectively.

3 Algorithmic design

To extract residues, 3 primary obstacles have been identified.

- Determining which brackets contain poles (Bracketing method)
- Computing a contour integral around a pole (Numerical integration method)
- Converging arbitrarily close to a pole (Zero finding method)

3.1 Complex differentiation

The residue calculator draws upon methods that required the derivative to calculate; it will be necessary to implement numerical differentiation. Complex differentiation is defined in an identical way to real differentiation.

Definition 3.1 (Complex differentiation)**.**

$$f\text{is complex differentiable at } z_0 \iff f'(z_0) = \lim_{z\to 0} \frac{f(z_0) - f(z)}{z_0 - z}$$

Since meromorphic functions are complex differentiable, we can be sure that we take the derivative at any point (and just as importantly, any direction in the complex plane.

3.1.1 Implementation (how the code works)

This method uses a 5-point balanced differentiation method to calculate the derivative.

$$f'(z) = \frac{-f(x+2h) + 8f(z+h) - 8f(z-h) + f(z-2h)}{12h} + O(h^4)$$

At one stage, the Cauchy integral formula was considered as an implementation; it yielded extreme accuracy in comparison to the current implementation and allowed the calculation of derivatives to given tolerances, but it had severe impacts on computational complexity and was therefore scrapped.

3.2 Complex Trapezoidal integration

3.2.1 Implementation (how the code works)

A standard trapezoidal rule algorithm that exponentially increases points was implemented. The trapezoidal rule proved to be much more efficient than the Gaussian-Legendre integration algorithm for the bracketing procedure, reducing runtimes by 50% in some cases!

3.3 Complex Gaussian-Legendre integration

3.3.1 Implementation (how the code works)

Originally, contour integrals were programmed to have circular paths. Though this was functional, it was ultimately decided to use rectangular paths as o avoid superfluous applications of the exponential function in an attempt to reduce computational complexity, as well as to better suit the geometry of the brackets.

3.4 Bracketing procedure with Cauchy's argument principle as bracketing method criteria

A rectangular domain is required input for this method, and similarly to the bracketing of zeros this method will attempt to bracket poles. Instead of leveraging the intermediate value theorem as the criteria for determining poles, the following is much more robust.

Theorem 3.1 (Cauchy's argument principle)**.** Let U be a subset of the domain Ω, then the following integral equals the amount of zeros enclosed in U minus the amount of poles enclosed in U.

$$U \subseteq \Omega \implies \frac{1}{2\pi i} \oint_{\partial U} \frac{f'(z)}{f(z)} dz = |Z_U| - |P_U|$$

This principle will swap with the intermediate value theorem as our new sufficient condition for poles used in the bracketing procedure.

3.4.1 Implementation (how the code works)

A complex rectangle is split, and integrals formed according to the Cauchy argument principle. 4 contour integrals for the borders of the bracket are taken by means of the Trapezoidal rule where tolerance is set to 0.1. Such a low tolerance is sufficient due to the fact these integrals always result in integers. Brackets are identified to have poles if the integral is a negative integer.

3.5 Complex 'pole' Newton-Raphson method

Fortunately, the problem of finding a pole of f is logically equivalent to finding a zero of $\frac{1}{f}$, hence it is sufficient to find a zero finding algorithm that is compatible with a complex neighborhood.

3.5.1 Implementation (how the code works)

By using a '+' is used instead of '-', the method converges to the roots of $\frac{1}{f}$ rather than f; this modification gears the algorithm towards poles rather than zeros.

$$z_{n+1} = z_n + \frac{f(z_n)}{f'(z_n)}$$

The numerical derivative is again taken with $h = 10^{-15}$.

4 Analysis

4.1 Test cases

We will consider a pair of functions that are mermomorphic on the following domain.

$$\Omega = \{z : \Re(z) \in [-3.5, 3], \Im(z) \in [-3, 3]\}$$

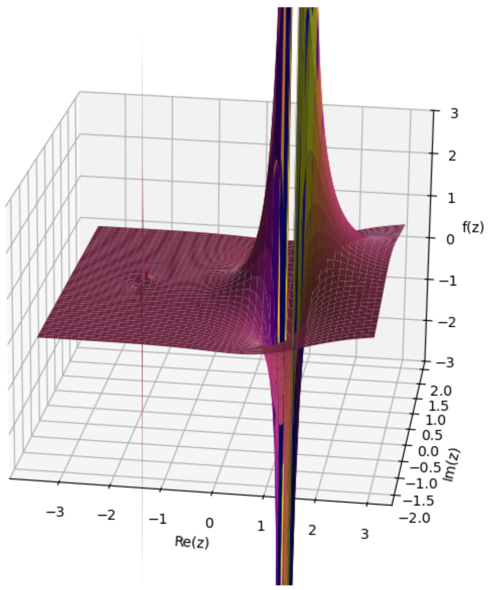

Figure 1: $f(z)$

4.1.1 An elementary function

$$f(z) = \frac{e^z}{z^3 - 3z + 2} = \frac{e^z}{(z-1)(z+2)}$$

Analytically, the pole-residue pairs for Ω are as such.

$$\text{Res}(f, \Omega) = \{(1, \frac{2e}{9}), (-2, \frac{e^{-2}}{9})\}$$

Numerically, the program estimates this set as the following.

$M(f, 10^{-9}, 50, \Omega) = \{$
$(-1.9999999999620535 - 2.140737409649829 \times 10^{-25}i, 0.01503725366679566 + 8.9024879476904 \times 10^{-19}i),$
$(1.000000000361795 - 5.3191420110654575 \times 10^{-6}i, 0.6040626284983213 - 3.990954244586713 \times 10^{-13}i)$
$\}$

4.1.2 Tangent function

$$\tan(z)$$

Analytically, the pole-residue pairs for Ω are as such.

$$\text{Res}(\tan, \Omega) = \{(\frac{\pi}{2}, -1), (-\frac{\pi}{2}, -1)\}$$

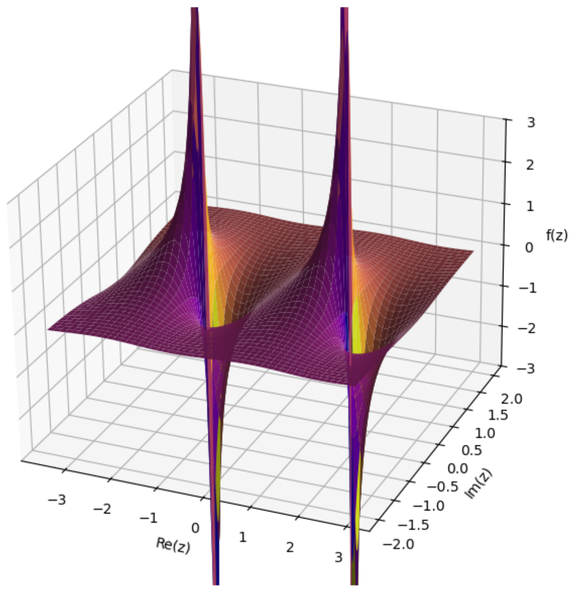

Figure 2: $\tan(z)$

Numerically, the program estimates this set as the following.

$M(\tan, 10^{-9}, 50, \Omega) = \{$
$(-1.5707963268491374 - 2.1092192356766435 \times 10^{-24}i, -0.9999999999714314 + 7.617825129923067 \times 10^{-}$
$(1.570796326839195 - 2.824047436646904 \times 10^{-25}i, -1.0000000000254874 - 2.965993452951992 \times 10^{-17}i)$
$\}$

4.2 Computational complexity

The most expensive operation in the whole algorithm is the bracketing procedure. This means that the algorithm runs in the $O(n^2)$.
Note that an amortised analysis of integration for the Cauchy principle argument shows that 2 iterations of the trapezoidal rule are usually required to meet the tolerance of $1e-1$. Hence for practical purposes it can be interpreted as some constant amount of steps ($O(1)$).

4.3 Error analysis

The integration methods calculates Cauchy principle arguments and residues to arbitrary precision, so this section concerns itself with the Newton-Raphson method's error since it is susceptible to propagation error from the numerical derivative. . Numerical differentiation is employed such that the errror is $O(h^4)$. Note that since the calculation of poles by the

Newton-Raphson relies on the numerical derivative, this error induces a small amount of propagation error to the Newton-Raphson method that is difficult to analyse.
One can show that the employed Newton-Raphson method has quadratic convergence, that is, if the method converges then the following holds regarding the error with some pole ζ.

$$\lim_{n\to\infty} \frac{|\zeta - z_{n+1}|}{|\zeta - z_n|^2} < \infty$$

Assuming convergence, the bottleneck of this method in terms of error minimisation stems from the use of a numerical derivative, which although allows the Newton-Raphson method to be used, suffers from roundoff error, which leads to propagated error in the Newton-Raphson method.

4.4 Advantages

4.4.1 Robustness of Cauchy's argument principle

The intermediate value theorem is the general 'kernel' of the bracketing method on $\mathbb{R}$. It can be extended to $\mathbb{C}$ and $\mathbb{R}^n$, however it is worth mentioning that if one tried to use the intermediate value theorem on $\frac{1}{f}$ to find poles, it would have the flaw that if $\frac{1}{f}$ quickly drops below zero and rises all within the same bracket, the pole won't be recogized.
Cauchy's principle argument gives an extremely precise criterion that returns an integer that determines the existence of poles. It even gives the order of the poles it detects, which although unused in this program could be used to give more information about each pole. Though Cauchy's argument principle isn't infallible, it avoids many of the issues present with the intermediate value theorem.

4.4.2 Efficient hybrid of Trapezoidal rule and Gaussian-Legendre quadrature

Originally Gaussian-Legendre quadrature was the sole numerical integration algorithm implemented in this program. However after some experimentation, it turns out that the Trapezoidal rule proved superior when haste was paramount and accuracy wasn't of major concern. The bracketing method requires many iterations of integration, all of which result to an integer; Gaussian-Legendre integration wasted many cycles in recomputing zeros and weights on these simple integrals.
The 'refined' trapezoidal rule was implemented in such a way that the tolerance could be specified to one decimal place, which was sufficient for this purpose. Under this scheme, the program's running time was cut in half for select test cases.

4.5 Disadvantages

4.5.1 Usage of numerical derivative

Cauchy's principle argument bracketing and the complex Newton-Raphson method are at the heart of this algorithm's functionality, hence it is necessary to include a numerical differentiation method as support.

The use of a numerical derivative induced a propagation of error when used in larger methods due to the fact that a numerically derived result is used in an estimation. Due to the nature of Python, roundoff error may place a bound on how close the program can calculate the derivative, regardless of how small the tolerance is made.
To manage the related risks of including numerical differentiation, derivatives have been computed with high tolerance and by the use of the 5-point balanced difference formula; a highly robust model for numerical differentiation.

4.5.2 Failure of Newton-Raphson method to converge

For functions that are pathological within the bracket, the Newton-Raphson method may fail to converge; estimates may escape the bracket or cycle endlessly. However, if this issue arises by the function changing convavity in the bracket, smaller brackets help tremendously to mitigate this phenomenon.

4.5.3 Failure of Cauchy's argument principle

Zeros in the same bracket as a pole may 'cancel out' the pole the from the Cauchy principle integral. For instance, if a bracket contains a zero and a pole of the same multiplicity, the Cauchy's argument principle will return 0; effectively ignoring the fact that a pole is present. Regardless of the criteria used for verifying poles, bracketing methods are not entirely robust.

4.5.4 Computational cost of Cauchy's argument principle

One critical disadvantage of using a Cauchy'S argument principle as the bracketing method criteria Is the fact that every bracket requires a contour integral to be done to verify whether there exist poles in the bracket. Even though the nature of these integrals requires little accuracy and can be optimised by choosing an appropriate integration method, it scales much worse than the Intermediate value theorem bracketing methods, even if it has an amortised analysis of $O(1)$.

4.5.5 Integrating directly on the pole

Once poles are located, the bracket of said pole provides the path of integration. If the pole falls directly on the bracket, the integral will fail to converge and crash the program. One can often increase the number of brackets slightly to avoid this issue.