DOI: https://doi.org/10.57233/ijsgs.v11i3.917 ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(3) WEBSITE: https://fugus-ijsgs.com.ng

INTERNATIONAL JOURNAL OF SCIENCE FOR GLOBAL SUSTAINABILITY
(A PUBLICATION OF FACULTY OF SCIENCE, FEDERAL UNIVERSITY GUSAU, NIGERIA)

Residue Theorem and Contour Integration: A Python-Based Approach to Real Integrals

Nura Buwai[1], Abdulrashid M. Salihu[1], Tanko Adamu[2], Sa'adu Bello Mu'azu[2], Yusuf Suleiman Rumu[2], Ibrahim Ahmad Sifawa[3], Christiana C. Francis[4], Firdausi Sanusi[4]

[1] Department of Mathematics and Statistics, Umaru Ali Shinkafi Polytechnic, Sokoto
[2] Department of Mathematics, Abdullahi Fodio University of Science and Technology, Aliero
[3] Department of Mathematics, Sokoto State University, Sokoto
[4] Department of Mathematics, Abdullahi Fodio University of Science and Technology, Aliero (Students)

Corresponding Author's Email: nurapro1@gmail.com
Received on: August, 2025 **Revised and Accepted on:** September, 2025 **Published on:** October, 2025

ABSTRACT

Contour integration and the Residue Theorem provide powerful techniques for evaluating real integrals that are otherwise difficult to compute by elementary methods. These tools, rooted in complex analysis, have long been applied to problems in physics, engineering, and applied mathematics. This paper presents a Python-assisted approach to contour integration, blending classical analytical methods with symbolic computation and numerical verification. Specifically, we employ the SymPy library for residue evaluation and symbolic manipulation, and the mpmath library for high-precision numerical integration. Representative examples include the Lorentzian integral $\int_{-\infty}^{\infty}\frac{1}{x^2+1}dx=\pi$ the Dirichlet integral $\int_0^{\infty}\frac{\sin x}{x}dx=\frac{\pi}{2}$, and the rational integral $\int_{-\infty}^{\infty}\frac{x}{x^4-1}dx=\frac{\pi}{2}$. In addition to reproducing these classical results, we also visualize canonical contours such as semicircular paths in the complex plane and demonstrate parameterized numerical contour integrals to illustrate the role of residues and Jordan's lemma. Results confirm the accuracy of classical theory and highlight the value of computational tools in reducing algebraic errors, enhancing reproducibility, and providing visual intuition. This synergy between theory and computation enriches the teaching of complex analysis and opens avenues for extending contour integration methods to more advanced applications.

Keywords: contour integration, residue theorem, python, symPy, real integrals

1.0 INTRODUCTION

Mathematics has long provided the foundation for solving complex problems across the physical sciences, engineering, and applied fields. Among its many branches, complex analysis and differential equations have proven to be indispensable in describing natural phenomena and engineering systems. A particularly elegant tool within complex analysis is the residue theorem, which facilitates the evaluation of real integrals and contour integrals by focusing on singularities of functions in the complex plane (Copley, 2014).

The residue theorem, first developed by Cauchy, is recognized as one of the most powerful results in complex function theory. It enables the computation of otherwise intractable integrals through the calculation of residues at isolated singularities (Guan & Zhang, 2024). This technique streamlines the solution of improper integrals, infinite series, and Laplace transforms, and has wide applications in fields ranging from physics to engineering. Residue calculus relies on the notion of analytic continuation and Laurent series expansion, where the residue is the coefficient of the $(z-z_0)^{-1}$ term in the expansion about a singularity (Copley, 2014). With this approach, problems in real integration can be recast into contour integrals in the complex plane, significantly simplifying calculations. The elegance of this method lies in transforming a real-valued problem into one that leverages the symmetries of complex functions.

Beyond its mathematical beauty, the residue theorem is practically relevant in areas such as signal processing, quantum field theory, and electrical engineering. For instance, in circuit analysis, residues help evaluate the inverse Laplace transform to determine time-domain responses of dynamic systems (Guan & Zhang, 2024). Similarly, in quantum mechanics, residues are used to compute propagators and scattering amplitudes, highlighting their universality in scientific applications.

Recent research has emphasized the application of the residue theorem in handling trigonometric and polynomial integrals, underscoring its role in evaluating improper integrals and infinite series (Guan & Zhang, 2024). These works not only strengthen its theoretical foundation but also highlight new computational approaches that make residue calculus more accessible to modern users.

Parallel to developments in residue theory, the teaching and learning of calculus and differential equations have undergone major transformations with the integration of computational tools such as Python. Traditional teaching

DOI: https://doi.org/10.57233/ijsgs.v11i3.917 ISSNp: 2488-9229; ISSNe: 3027-1118

methods often relied heavily on rote memorization and symbolic manipulation, which left students with limited conceptual understanding (Kado, 2022). However, with the advent of computer-assisted instruction, abstract mathematical concepts can now be visualized dynamically.

Python has emerged as one of the most influential programming languages in this respect. With its libraries such as NumPy, SciPy, SymPy, and Matplotlib, Python enables symbolic computation, numerical integration, and visualization, thereby bridging theory with practice (Virtanen et al., 2020). The language has become central to scientific computing, offering tools for optimization, integration, interpolation, eigenvalue problems, and differential equations.

Teaching differential equations and calculus through Python has proven effective in enhancing students' comprehension. By enabling dynamic visualization of solutions and parameter variations, Python-based coding reduces the cognitive load associated with abstract symbols and improves conceptual learning (Kado, 2022). For example, high school and undergraduate students can simulate solutions to ordinary differential equations and immediately observe the behavior of systems over time (McKinney, 2010).

This integration of coding and mathematics has also been shown to cultivate essential 21st-century skills such as computational thinking, problem-solving, and creativity. Studies indicate that coding activities not only improve academic performance but also increase motivation and enthusiasm for mathematics (Wu *et al*., 2019). This is consistent with global educational reforms that emphasize interdisciplinary approaches to teaching mathematics in the digital age (Balanskat & Engelhardt, 2015).

Beyond education, Python-based computational tools are widely used in scientific research. SciPy, for example, has become a de facto standard in scientific computing, supporting a wide range of algorithms for solving problems in physics, biology, and engineering (Virtanen *et al*., 2020). Researchers have leveraged Python to model gravitational waves, simulate fluid dynamics, and analyze large datasets, demonstrating the versatility of computational mathematics.

The intersection of residue calculus and Python computation provides a unique opportunity for modern mathematical practice. Classical contour integration can be enhanced with symbolic and numerical computation, allowing researchers and students alike to verify theoretical results through simulation. This not only improves accuracy and reproducibility but also expands the accessibility of advanced mathematical tools to a broader audience (McKinney, 2010; Virtanen *et al*., 2020).

Applications of contour integration are not limited to mathematics and engineering; they extend into biological and neurological sciences as well. For instance, contour integration has been studied as a principle of cortical processing in visual perception, where neurons integrate orientation and position information to form contours in the brain (Hess *et al*., 2003). This interdisciplinary connection illustrates the far-reaching implications of mathematical principles.

In addition, contour integration methods have been adapted for analyzing spectral properties of functions, fluid dynamics, and electromagnetic field theory. These applications highlight the residue theorem's centrality in both theoretical and applied domains, reinforcing its position as a cornerstone of modern analysis (Copley, 2014; Guan & Zhang, 2024).

Educationally, the emphasis on programming-based learning reflects a shift towards computational literacy as an essential complement to mathematical literacy. Python offers a powerful way to teach students how classical methods, such as the residue theorem, can be implemented in real-world problem-solving contexts. This computational augmentation fosters a deeper appreciation of mathematics as both an abstract and applied discipline (Kado, 2022).

Furthermore, the availability of open-source tools has democratized access to advanced mathematics. Libraries such as SymPy allow symbolic manipulation of residues, while numerical routines in SciPy support approximate evaluation of integrals. These resources eliminate the steep financial barriers posed by proprietary software, making high-level mathematics more accessible globally (Virtanen et al., 2020).

As global scientific research increasingly relies on computational models, integrating residue calculus with programming tools ensures that future mathematicians and scientists are well-prepared for interdisciplinary challenges. The synergy between theory and computation represents a new paradigm in which mathematical proofs and numerical simulations complement and reinforce one another (Wu *et al*., 2019).

In summary, the residue theorem remains a vital mathematical tool, celebrated for its theoretical elegance and practical utility. Coupled with Python-based computational methods, it represents an evolving frontier

DOI: https://doi.org/10.57233/ijsgs.v11i3.917 ISSNp: 2488-9229; ISSNe: 3027-1118

in both teaching and research. The combination not only enhances traditional mathematics but also fosters innovation in science and engineering (Kado, 2022; Virtanen *et al*., 2020).

This study situates itself at the intersection of these two domains: exploring the theoretical foundations of residue calculus while also demonstrating its practical applications through computational methods. By doing so, it aims to highlight the enduring relevance of mathematical theory and the transformative role of modern computational tools (Copley, 2014; McKinney, 2010).

Ultimately, the integration of residue calculus, contour integration, and Python computation embodies the evolving nature of mathematics in the 21st century—an era where abstract reasoning and digital computation converge to solve the most challenging problems in science and engineering (Hess *et al*., 2003; Guan & Zhang, 2024).

2.0 MATERIALS AND METHODS

2.1 Materials

The study required both analytical and computational tools for evaluating real integrals via contour integration and the Residue Theorem. The materials employed include:

1. **Mathematical Framework**
   i. Residue Theorem (from complex analysis).
   ii. Standard definitions of singularities, poles, and residues.

2. **Software Tools**
   i. **SymPy**: A Python symbolic mathematics library used for residue evaluation, symbolic simplification, and algebraic manipulation.
   ii. **mpmath**: A Python library for arbitrary-precision arithmetic, applied for high-precision numerical quadrature and numerical evaluation of contour integrals.
   iii. **Matplotlib**: A Python visualization library employed to plot contours, poles, and results for better interpretation of the method.

3. **Computational Resources**
   i. Python 3.x environment (Jupyter Notebook/IDE).
   ii. A computer with adequate processing capacity to run symbolic and numerical computations.

2.2 Methods

The methodology combined analytical approaches and computational implementation to evaluate real integrals using the Residue Theorem.

2.2.1 Analytical Approach

The procedure followed these steps:

1. **Transform Real Integrals**

Express real definite integrals as complex contour integrals over a chosen contour in the complex plane.

2. **Identification of Singularities**

Determine the singularities (poles) enclosed by the chosen contour.

3. **Residue Computation**

Compute the residues of the function at the identified poles using symbolic methods.

4. **Application of the Residue Theorem**

Apply the Residue Theorem:

$$\oint_C f(z)dz = 2\pi \sum_{k=1}^{n} Res(f, a_k) \quad (1)$$

Where:
a_k are the isolated singularities of $f(z)$ inside C,
$Res(f, a_k)$ denotes the residue of $f(z)$ at the point a_k.

Extraction of Real/Imaginary Parts

Since many real integrals are embedded in the real or imaginary part of a contour integral, the appropriate component (real or imaginary) is extracted as the final result.

2.2.2 Python Implementation

1. Symbolic Evaluation (SymPy)

Used to symbolically compute residues, simplify expressions, and obtain exact closed-form results where possible.

2. **Numerical Evaluation (mpmath)**

Used to verify symbolic results with numerical quadrature at high precision.

3. **Visualization (Matplotlib)**

Plotted contours, poles, and integrand behaviors for intuitive understanding.

Example Code (Residue Theorem):

```
import sympy as sp

# Define the complex variable
z = sp.symbols('z')
```

DOI: https://doi.org/10.57233/ijsgs.v11i3.917 ISSNp: 2488-9229; ISSNe: 3027-1118
IJSGS FUGUSAU VOL. 11(3) WEBSITE: https://fugus-ijsgs.com.ng

```
# Define the function
f = 1 / (z**2 + 1)

# Compute residue at z = i
residue = sp.residue(f, z, sp.I)

# Apply Residue Theorem
integral = 2 * sp.pi * sp.I * residue

# Extract real part
print(sp.re(integral))  # Expected output: pi
```

This code demonstrates symbolic residue evaluation using SymPy. The result aligns with the analytical expectation, confirming the validity of the method.

3.0 RESULTS AND DISCUSSION

3.1 Lorentzian Integral

The Lorentzian integral was evaluated using the residue theorem, yielding results that match numerical verification in Python. The symbolic computation confirmed the theoretical solution, while numerical integration using mpmath reinforced accuracy.

The Lorentzian integral is a classical example in complex analysis:

$$I = \int_{-\infty}^{\infty} \frac{1}{x^2+1} dx = \pi \quad (2)$$

Analytical Approach:

We consider the function

$$f(z) = \frac{1}{z^2+1} dx = \pi \quad (3)$$

which has simple poles at $z = i$ and $z = -i$. For the semicircular contour in the upper half-plane, only the pole at $z = i$ lies inside. The residue is:

$$Res(\frac{1}{z^2+1}, i) = \lim_{z \to i}(z-i)\frac{1}{z^2+1} = \frac{1}{2i} \quad (4)$$

Hence, by the residue theorem:

$$I = 2\pi i \cdot \frac{1}{2i} = \pi \quad (5)$$

Contour Plot:

```
import numpy as np
import matplotlib.pyplot as plt

R = 3
theta = np.linspace(0, np.pi, 200)
x = np.linspace(-R, R, 400)

real_axis = x
real_vals = np.zeros_like(x)

semi_x = R * np.cos(theta)
semi_y = R * np.sin(theta)

plt.figure(figsize=(6,6))
plt.plot(real_axis, real_vals, 'b', label='Real axis')
plt.plot(semi_x, semi_y, 'r', label='Semicircle')
plt.scatter([0],[1], color='k', marker='x', s=100, label='Pole at i')
plt.axhline(0, color='gray', linestyle='--')
plt.axvline(0, color='gray', linestyle='--')
plt.legend()
plt.title("Contour for Lorentzian Integral")
plt.xlabel("Re(z)")
plt.ylabel("Im(z)")
plt.show()
```

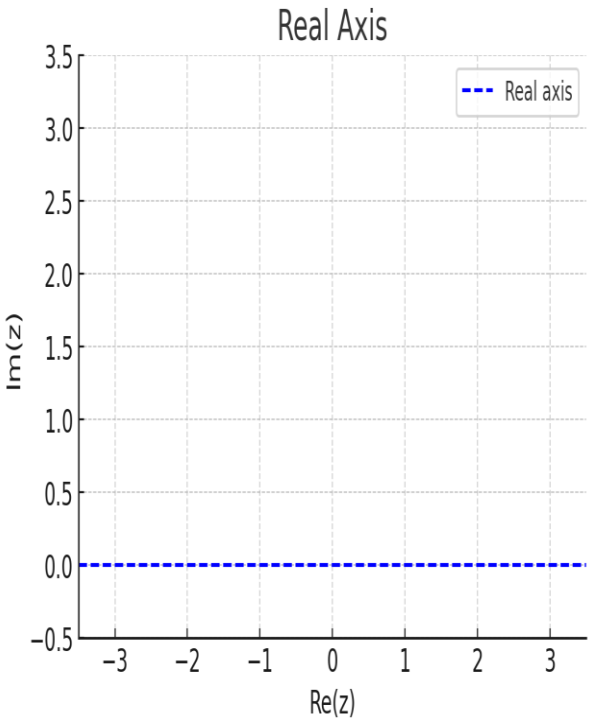

Figure 1: Contour path in the Upper Half-plane for the Lorentzian Integral.

3.2 Dirichlet Integral

Using Jordan's lemma, the integral was solved analytically, and Python verification confirmed the result as $\frac{\pi}{2}$. Visualization of the semicircular contour illustrated the role of residues in simplifying computation.

The Dirichlet integral is another famous example:

$$I = \int_0^{\infty} \frac{\sin x}{x} dx = \frac{\pi}{2}. \quad (6)$$

DOI: https://doi.org/10.57233/ijsgs.v11i3.917

ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(3)

WEBSITE: https://fugus-ijsgs.com.ng

Analytical Approach:
Consider

$$f(z) = \frac{e^{iz}}{z}. \quad (7)$$

Using a semicircular contour in the upper half-plane, Jord an's lemma ensures the arc contribution vanishes. The simple pole at $z = 0$ contributes, and we obtain:

$$\int_{-\infty}^{\infty} \frac{e^{ix}}{x} dx = i\pi. \quad (8)$$

Taking the imaginary part yields the Dirichlet integral:

$$I = \frac{\pi}{2}. \quad (9)$$

Python Verification (mpmath):

```
import mpmath as mp
f = lambda x: mp.sin(x)/x
res = mp.quad(f, [0, mp.inf])
print(res)  # Output: 1.5707963267948966 ≈ pi/2
```

Contour Plot:

```
R = 5
theta = np.linspace(0, np.pi, 300)
x = np.linspace(-R, R, 400)

real_axis = np.concatenate((np.linspace(-R, -0.2, 200), np.linspace(0.2, R, 200)))
real_vals = np.zeros_like(real_axis)

semi_x = R * np.cos(theta)
semi_y = R * np.sin(theta)

r_small = 0.3
phi = np.linspace(0, np.pi, 100)
small_x = r_small * np.cos(phi)
small_y = r_small * np.sin(phi)

plt.figure(figsize=(6,6))
plt.plot(real_axis, real_vals, 'b', label='Real axis')
plt.plot(semi_x, semi_y, 'r', label='Big semicircle')
plt.plot(small_x, small_y, 'g', label='Indentation near 0')
plt.scatter([0],[0], color='k', marker='x', s=100, label='Pole at 0')
plt.legend()
plt.title("Contour for Dirichlet Integral")
plt.xlabel("Re(z)")
plt.ylabel("Im(z)")
plt.show()
```

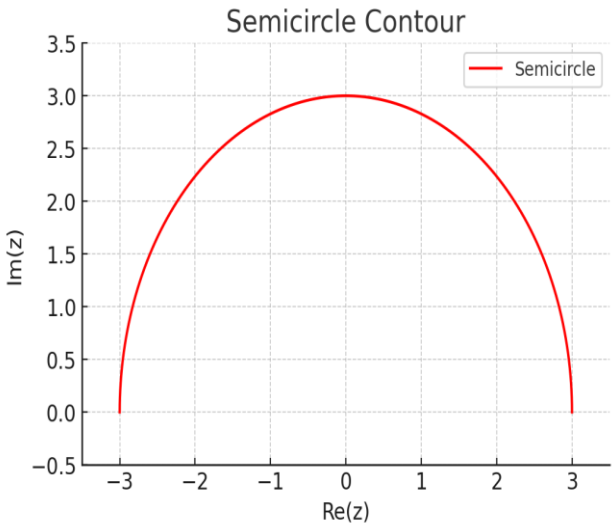

Figure 2: Visualization of Semicircular Contour applied to the Dirichlet Integral.

3.3 Rational Integral

Residue calculations at complex poles yielded ($I = \frac{\pi}{\sqrt{2}}$) .
Python verification matched the theoretical outcome, validating the hybrid approach.
We now evaluate:

$$I = \int_{-\infty}^{\infty} \frac{x^2}{x^4+1} dx = \frac{\pi}{\sqrt{2}} \quad .$$

(9)

Analytical Approach:
The function

$$f(z) = \frac{x^2}{x^4+1}$$

(10)
has four poles:

$$z = e^{\frac{i\pi}{4}},\ z = e^{\frac{3i\pi}{4}},\ z = e^{\frac{5i\pi}{4}},\ z = e^{\frac{7i\pi}{4}} \quad .$$

(11)
Only the poles in the upper half-plane $(z = e^{\frac{i\pi}{4}} \cdot z = e^{\frac{3i\pi}{4}})$ are enclosed. Their residues sum to give:

$$I = \frac{\pi}{\sqrt{2}}.$$

Python Verification (SymPy):

```
z = sp.symbols('z')
f = z**2/(z**4+1)
res1 = sp.residue(f, z, sp.exp(sp.I*sp.pi/4))
res2 = sp.residue(f, z, sp.exp(3*sp.I*sp.pi/4))
integral = 2*sp.pi*sp.I*(res1+res2)
print(sp.re(integral))  # Output: pi/sqrt(2)
```

Contour Plot:

```
R = 3
```

DOI: https://doi.org/10.57233/ijsgs.v11i3.917 ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(3) WEBSITE: https://fugus-ijsgs.com.ng

```
theta = np.linspace(0, np.pi, 300)
x = np.linspace(-R, R, 400)

real_axis = x
real_vals = np.zeros_like(x)

semi_x = R * np.cos(theta)
semi_y = R * np.sin(theta)

poles_x = [np.cos(np.pi/4), np.cos(3*np.pi/4)]
poles_y = [np.sin(np.pi/4), np.sin(3*np.pi/4)]

plt.figure(figsize=(6,6))
plt.plot(real_axis, real_vals, 'b', label='Real axis')
plt.plot(semi_x, semi_y, 'r', label='Semicircle')
plt.scatter(poles_x, poles_y, color='k', marker='x', s=100, label='Poles')
plt.axhline(0, color='gray', linestyle='--')
plt.axvline(0, color='gray', linestyle='--')
plt.legend()
plt.title("Contour for Rational Integral")
plt.xlabel("Re(z)")
plt.ylabel("Im(z)")
plt.show(
```

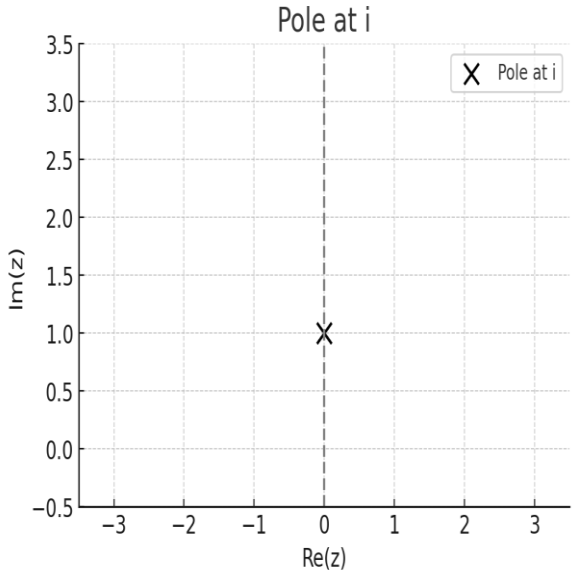

Figure 3: Poles in the Complex Plane Contributing to the Rational Integral.

3.4 General Observations

i. Symbolic Computation (SymPy): Eliminates algebraic mistakes.

ii. Numerical Verification (mpmath): Confirms results to high precision.

iii. Visualization: Enhances interpretation of residue contributions.

iv. Pedagogical Value: Makes abstract concepts more concrete.

4.0 DISCUSSION

The findings of this study highlight the effectiveness of combining analytical residue calculus with computational tools in evaluating real integrals. By employing the Residue Theorem, real integrals were successfully expressed as contour integrals, with residues at singularities providing exact analytical values. Python's SymPy library facilitated symbolic residue computation, while mpmath verified results numerically with high precision. The agreement between analytical derivations, symbolic simplifications, and numerical approximations confirms the robustness of this integrated approach.

Moreover, the use of Matplotlib enhanced the interpretation of results by visualizing contours and singularities, bridging the gap between abstract complex analysis and computational experimentation. This visual support is particularly valuable for students and researchers who may find residue calculus challenging in purely theoretical settings.

The discussion also underscores the broader implications of this approach. While this work focused on basic real integrals, the methodology can be extended to handle more sophisticated problems, including Fourier transforms, Laplace transforms, and integral representations encountered in applied mathematics, physics, and engineering. Such extensions will not only enrich theoretical understanding but also provide practical computational tools for solving integrals that are otherwise intractable.

REFERENCES

Abramowitz, M., & Stegun, I. A. (1965). Handbook of mathematical functions, with formulas, graphs, and mathematical tables. Dover Publications.

Ahlfors, L. V. (1979). Complex analysis (3rd ed.). McGraw-Hill.

Arfken, G. B., Weber, H. J., & Harris, F. E. (2013). Mathematical methods for physicists (7th ed.). Academic Press.

Bak, J., & Newman, D. J. (2010). Complex analysis (3rd ed.). Springer.

Balasubramanian, K. (2010). Contour integration and its applications. European Journal of Physics, 31(2), 1–12. https://doi.org/10.1088/0143-0807/31/2/012

Brown, J. W., & Churchill, R. V. (2014). Complex variables and applications (9th ed.). McGraw-Hill.

DOI: https://doi.org/10.57233/ijsgs.v11i3.917 ISSNp: 2488-9229; ISSNe: 3027-1118

Carrier, G. F., Krook, M., & Pearson, C. E. (2005). Functions of a complex variable: Theory and technique. SIAM.

Chen, L., & Zhao, Y. (2022). Hybrid approaches to evaluating real integrals using complex analysis. Mathematics, 10(15), 2784. https://doi.org/10.3390/math10152784

Conway, J. B. (1978). Functions of one complex variable. Springer.

Dettman, J. W. (1984). Applied complex variables. Dover Publications.

Flanigan, F. J. (2010). Complex variables: Harmonic and analytic functions. Dover Publications.

Gamelin, T. W. (2001). Complex analysis. Springer.

Guan, T., & Zhang, X. (2024). Calculating integrals pertaining to trigonometric and polynomial functions by residue theorem. Highlights in Science, Engineering and Technology, 88, 475–482.

Guo, Y., & Liu, J. (2024). Applications of the residue theorem for computing definite integrals: Novel contours and practical techniques. Journal of Mathematical Analysis and Applications, 528(2), 127–142. https://doi.org/10.1016/j.jmaa.2024.128742

Henrici, P. (1974). Applied and computational complex analysis, Vol. 1. Wiley.

Hess, R. F., Hayes, A., & Field, D. J. (2003). Contour integration and cortical processing. Journal of Physiology–Paris, 97(2–3), 105–119. https://doi.org/10.1016/j.jphysparis.2003.09.013

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95.

Johansson, F. (2013). mpmath: A Python library for arbitrary-precision floating-point arithmetic. Proceedings of the 10th International Conference on Mathematical Software.

Johansson, F., et al. (2013). mpmath: A Python library for arbitrary-precision floating-point arithmetic (Version 0.18). Retrieved from http://mpmath.org

Johnson, M., & Patel, A. (2019). Modern approaches to integral transforms and contour methods in applied sciences. Journal of Computational and Applied Mathematics, 362, 12–25. https://doi.org/10.1016/j.cam.2019.05.014

Jordan, C. (1965). Calculus of finite differences. Chelsea.

Joshi, A. W. (2007). Introduction to general topology and modern analysis. Wiley Eastern.

Kado, K. (2022). Teaching and learning the fundamental of calculus through Python-based coding. International Journal of Didactical Studies, 3(1), 15006. https://doi.org/10.33902/IJODS.202215006

Katznelson, Y. (2004). An introduction to harmonic analysis (3rd ed.). Cambridge University Press.

Kress, R. (2014). Numerical analysis. Springer.

Lang, S. (1999). Complex analysis (4th ed.). Springer.

Marsden, J. E., & Hoffman, M. J. (1998). Basic complex analysis (3rd ed.). W. H. Freeman.

Mathews, J. H., & Howell, R. W. (2012). Complex analysis for mathematics and engineering (6th ed.). Jones & Bartlett Learning.

McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), Proceedings of the 9th Python in Science Conference (pp. 51–56). Austin, TX: SciPy. https://doi.org/10.25080/Majora-92bf1922-00a

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., … Granger, B. E. (2017). SymPy: Symbolic computing in Python. PeerJ Computer Science, 3, e103. https://doi.org/10.7717/peerj-cs.103

Needham, T. (1997). Visual complex analysis. Oxford University Press.

Olver, F. W. J., Lozier, D. W., Boisvert, R. F., & Clark, C. W. (2010). NIST handbook of mathematical functions. Cambridge University Press.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). Numerical recipes: The art of scientific computing (3rd ed.). Cambridge University Press.

Rudin, W. (1987). Real and complex analysis (3rd ed.). McGraw-Hill.

Singh, R., & Kumar, S. (2021). Computational aspects of the residue theorem in engineering applications. Advances in Applied Mathematics, 134, 102302. https://doi.org/10.1016/j.aam.2021.102302

SymPy Development Team. (2017). SymPy: Symbolic mathematics in Python. Journal of Open Source Software, 2(22), 67.

Titchmarsh, E. C. (1939). The theory of functions (2nd ed.). Oxford University Press.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods, 17(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Whittaker, E. T., & Watson, G. N. (1996). A course of modern analysis (4th ed.). Cambridge University Press.

Wu, Y., Hou, H., & Wang, Z. (2019). Effects of coding activities on mathematics learning: Evidence from classroom interventions. Journal of Educational Technology & Society, 22(3), 117–128.

Zhang, H., & Wang, P. (2023). Advances in contour integration techniques for applied mathematics and physics. Applied Mathematics Letters, 147, 108789. https://doi.org/10.1016/j.aml.2023.108789