Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Advancements in Numerical Analysis: Techniques for Solving Volterra and Fredholm Equations

Aseel Ameen Harbi[a], Zahraa Nabil Kazem[b], Safa Ehab Mohammed[c]

[a]*College of Science, Department of Mathematics, University of Al-qadisiyah, Diwaniyah, Iraq. aseel.harbi@qu.edu.iq*

[b]*College of Administration and Economics, Statistics Department, University of Al-Qadisiyah, Diwaniyah, Iraq. zahraa.n.kazem@qu.edu.iq*

[c]*College of Administration and Economics, Statistics Department, University of Al-Qadisiyah, Diwaniyah, Iraq. safa.emohammed@qu.edu.iq*

ARTICLEINFO

Article history:
Received: 04/04/2025
Rrevised form: 26/04/2025
Accepted : 30 /04/2025
Available online: 30 /06/2025

Keywords:
Volterra Integral Equation
Fredholm Integral Equation
Numerical Analysis
Kernel Dataset
Forcing Function Dataset

ABSTRACT

This work concerns the construction of techniques to facilitate numerical analysis to solve basic computational mathematics problems of Volterra and Fredholm integral equations. In an attempt to merge contemporary machine learning techniques with conventional methods like the trapezoidal rule, the proposed methods seek enhanced accuracy, stability, and computing efficiency in applications involving synthetic and realistic kernel and forcing function data. With Python and computational libraries such as SciPy and NumPy, the algorithms are applied to actual physics and engineering problems and show greater precision and performance in various types of kernels, namely smooth, oscillatory, and weakly singular kernels. The methods for Volterra-type problems are highly stable, with iterative systems solving them effectively, and matrix-based methods solving Fredholm-type equations. This book contributes to numerical analysis through the presentation of new algorithms that enhance the practical solution of integral equations and have ramifications in computer science, engineering, and physics. The creation of hybrid numerical algorithms blending machine learning with conventional techniques bridges gaps in the literature and opens the door to more effective solutions to intricate integral equation problems.

MSC.

https://doi.org/10.29304/jqcsm.2025.17.22211

1. Introduction

In applied mathematics and engineering, numerical analysis serves as the foundation for solving complex real-world problems that typically defy analytical solutions [1]. Numerical methods are powerful tools for approximating solutions to mathematical models that describe natural and artificial systems in an increasingly data-driven world [2]. These methods have led to major advancements across various scientific and engineering disciplines, including biology, economics, physics, and environmental sciences [3]. Integral equations of the Volterra and Fredholm types form a crucial part of mathematical modeling of systems exhibiting interdependencies or interactions across time or space [4]. Given their broad applicability in fields such as fluid dynamics, heat conduction, electromagnetic propagation, disease transmission, and population dynamics, developing efficient numerical solutions for these equations remains vital for modern scientific and engineering applications.

Corresponding author: Aseel Ameen Harbi
Email addresses: *aseel.harbi@qu.edu.iq*

Communicated by 'sub etitor'

1.1. Fundamental Concepts

1.1.1. Importance of Integral Equations in Real-World Applications

In modeling structures that have spatial dependencies, reminiscence consequences, and dynamic interaction, critical equations are a powerful mathematical tool [5]. Physical systems including the fluid dynamics equation, the warmth conduction equation, and other electromagnetic wave propagation are commonly modeled using them [6]. The version for time-established techniques will become more relevant whilst Volterra fundamental equations are used, for example, when the modern-day state of a device is reliant on its beyond [7]. For instance, in organic structures like ailment transmission, the fee of infection would depend on the records of the disease, or in populace dynamics, where the level of populace is constantly encouraged by using beyond interactions [8].

1.1.2. Problem Scope and Kernel Properties

This paper explains Volterra and Fredholm equations in detail, including their kernel properties as well as boundary conditions [9]. This is done based on steadily increasing integration ranges and initial value problems with Volterra equations [10]. A suitable numerical method for such an equation is taken as the kernel's complexity concerning smoothness, singularity, and weak singularity in the selection process. To optimize some of the existing computational simplifications, fixed integration domains and specific types of kernels like symmetric and degenerate kernels are considered in the case of Fredholm equations [11].

Volterra Equations

$$u(x) = f(x) + \int_a^x K(x,t)u(t)dt \quad (1)$$

- **Type:** f(x)-free first type or f(x)-containing second kind
- **Domain:** x ∈ [a, b] with t ≤ x.
- **Boundary Conditions:**
  - u(a)=u_0 (For initial value problems).
  - As the integration range increases gradually, discretization is often concentrated near the lower limit a.
- **Kernel Properties:**
  - **Smooth Kernels:** Continuous and smooth, thus enabling robust numerical methods.
  - **Singular Kernels:** Have singularities near x=t, which requires regularization or special quadrature methods.
  - **Weakly Singular Kernels:** Often of the form K(x, t) = (x-t)^a, -1<a<0, which require fine resolution near singularities but are tractable.

Fredholm Equation

$$u(x) = f(x) + \int_a^b K(x,t)u(t)dt \quad (2)$$

- **Type:** free first type or second type containing f(x).
- **Domain:** (x ∈ [a, b]), with some definite interval of integration in t ∈ [a, b].
- **Boundary Condition:** Usually there are periodic conditions, if physics or engineering problem has to be solved or limits, such as u(a) and u(b).
- **Kernel Properties:**
  - **Separable Kernels:** Can be expressed as K(x, t) = ϕ(x) ψ (t), reducing computational complexity.

- **Symmetric Kernels:** K(x, t) = K(t, x), permitting effective eigen decomposition techniques.
- **Degenerate Kernels:** Analytical simplifications are made possible by its representation as a finite sum of separable terms.

1.1.3. Transformations for Simplification

To make integral equations simpler, transformations such as kernel approximations, regularization strategies, and discretization techniques are used [12]. While collocation techniques and quadrature rules allow integral equations to be transformed into solvable systems of linear equations, Tikhonov regularization and other techniques handle ill-posed situations.

Regularization Techniques (for Ill-Conditioned Problems)

- Fredholm of the First Kind: By including regularization terms (such as Tikhonov regularization) [13], you can transform poorly posed situations:

$$\min_u \| {}_a^b K(x,t)u(t)dt - f(x)\|^2 + \lambda \|u\|^2 \quad (3)$$

where λ is the regularization parameter.

Discretization Approaches

Integral equations can be transformed into linear equation systems utilizing techniques like [14]:

- **Quadrature Rules:** Instead of using integrals, use numerical quadrature. (e.g., Trapezoidal, Gauss-Legendre methods).
- **Collocation Methods:** A matrix formulation results from choosing particular points (collocation points) to satisfy the equation.

Kernel Approximation

- Simplified functions can be used to approximate complicated kernels [15]:
  - Separable Kernels: Approximate K (x, t) as ${}_{i=1}^{n}\phi_i(x)\psi_i(t)$.
  - Low-Rank Approximation: Utilize methods such as singular value decomposition (SVD) to decrease the form of K (x, t).

Change of Variables

- Simplify integration limits or kernel behavior [16]:
  - $x = a + (b-a)\xi$ and$t = a + (b-a)\eta$, transforming the domain into [0, 1].
- Singularities can be resolved with substitutes such as τ =x-t for Volterra equations.

Integral Operator Simplifications

- Substitute matrix approximations for integral operators [17]:

$$\int_a^b K(x,t)u(t)dt \approx \sum_{j=1}^{n} w_j K(x,t_j)u(t_j) \quad (4)$$

where w_j are weights from numerical quadrature, and t_j are evaluation points.

1.1.4. Dataset Characteristics

To evaluate the approaches for variations in equation types and behavior, benchmark datasets are assembled. These consist of known forcing functions, oscillating kernels, Volterra, Fredholm equations, and artificial data utilized for typical sorts of equations [18]. In order to examine and maybe validate the numerical schemes, one employs analytical answers.

Dataset 1: Volterra Equation

$$u(x) = f(x) + {}_a^x K(x,t)u(t)dt \quad (5)$$

- **Range:** x, t ∈ [0, 1]
- **Kernel:** K(x, t) = sin(x + t)
- **Forcing Function:** f(x) = e^x
- **Analytical Solution:** u(x) = cos(x) + x

Dataset 2: Fredholm Equation

$$u(x) = f(x) + {}_a^b K(x,t)u(t)dt \quad (6)$$

- **Range:** x, t ∈ [0, 1]
- **Kernel:** K(x, t) = $e^{-|x-t|}$
- **Forcing Function:** f(x) = x^2
- **Analytical Solution:** u(x) = x^3 - x + 1

Dataset 3: Oscillatory Kernel

- **Type of Equation**: Fredholm Integral Equation
- **Range:** x, t ∈ [0, 2π]
- **Kernel:** K (x, t) = cos (x - t)
- **Forcing Function:** f(x) = sin(x)
- **Analytical Solution:** Approximate via numerical integration.

The use of oscillatory kernels has been discussed in earlier works on computational mathematics [19].

1.1.5. Computational Techniques for Numerical Solutions

The computer strategies used to remedy the Volterra and Fredholm equations are the main topic of this phase. To assure accuracy in regions wanting finer resolution, iterative approaches utilizing adaptive quadrature strategies are employed for Volterra equations [20]. Matrix formulations are used to remedy Fredholm equations, whilst iterative strategies and numerical solvers which include Gaussian elimination are used to improve computational efficiency.

Advanced strategies like low-rank approximations and GPU acceleration are used to cope with excessive-dimensional issues and dense kernel matrices, significantly reducing computing complexity. While performance measures like execution time and convergence tolerance are focused of dependable and effective solutions, adaptive quadrature guarantees accuracy in hard regions.

Volterra Integral Equations

$$u(x) = f(x) + {}_{a}^{x}K(x,t)u(t)dt \tag{7}$$

- **Numerical Calculation:**
  - Discretize x into N points: $x_1, x_2, \ldots, x_N$.
  - Use the trapezoidal rule or another numerical quadrature method:

$$u(x_i) = f(x_i) + \sum_{j=1}^{i-1} w_j K(x_i, x_j) u(x_j) \tag{8}$$

where w_j are the quadrature rule's weights, and the kernel is denoted by $K(x_i, x_j)$.

- **Iterative Algorithm:**
  - Start with an initial guess $u_0(x) = f(x)$.
  - Iteratively update

$$u_{k+1}(x_i) = f(x_i) + {}_{j=1}^{i-1}w_j K(x_i, x_j) u_k(x_j) \tag{9}$$

  - Continue until convergence is reached, usually as shown by:

$$\|u_{k+1} - u_k\| < \epsilon \tag{10}$$

where ϵ is a small tolerance.

- **Advanced Optimization:**
  - Adaptative quadratures in the case where there are steep slopes or singularities in K(x,t) or f(x).
  - The use of GPUs for high-speed computing in high dimensional systems.

Fredholm Integral Equations

$$u(x)f(x) + {}_{a}^{b}K(x,t)u(t)dt \tag{11}$$

- **Numerical Calculation:**
  - Discretize x and t into N points: $x_1, x_2, \ldots, x_N$ and $t_1, t_2, \ldots, t_N$.
  - Substitute the integral with a summation utilizing Gaussian quadrature or the trapezoidal rule:

$$u(x_i) = f(x_i) + {}_{j=1}^{N}w_j K(x_i, t_j) u(t_j) \tag{12}$$

- **Matrix Formulation:**
  - Express the equation in matrix form:

$$U = F + KU \tag{13}$$

where:

- $U = [u(x_1), u(x_2), \ldots, u(x_N)]^T$
- $F = [f(x_1), f(x_2), \ldots, f(x_N)]^T$
- $K = [K(x_i, t_j)]_{i,j=1}^{N}$ is the kernel matrix.
  - Solve the linear system:

$$(I - K)\, U = F \tag{14}$$

Employ iterative solvers: GMRES and Conjugate Gradient and Gaussian elimination and LU decomposition among other numerical solvers.

- **Advanced Optimization:**

- Utilize the approximations with low rank in SVD, where the decomposing occurs only for the dense kernel matrix:

$$K \approx {}_{k=1}^{\tau}\sigma_k u_k v_k^T \quad (15)$$

In this, σ^k and u^k, v^k are vectors and singular values.

1.1.6. Illustrative Example

Examples for both Volterra and Fredholm equations are provided, including their discretized forms, iterative solutions, and matrix solution approaches [21].

- **Volterra Example:**

$$u(x) = x + {}_0^x(x-t)u(t)dt \quad (16)$$

  - Discretized form:

$$u(x_i) = x_i + {}_{j=1}^{i-1}(x_i - x_j)u(x_j)\Delta t \quad (17)$$

  - Solve iteratively until u(x) converges.
- **Fredholm Example:**

$$u(x) = \sin(x) + {}_0^1\cos(x-t)u(t)dt \quad (18)$$

  - Discretized form:

$$u(x_i) = \sin(x_i) + {}_{j=1}^{N}cos(x_i - t_j)u(t_j)\,\Delta t \quad (19)$$

Solve the linear system (I - K) U = F using matrix solvers.

2. Results

The effects of solving the Volterra and Fredholm indispensable equations the usage of the recommended numerical strategies is all methodically covered in this phase. The effectiveness and precision of the created algorithms are highlighted via graphical representation and execution approaches with consequences.

2.1. Volterra Kernel Dataset

Input Data: The kernel feature was used in the Volterra kernel dataset K (x, t) = sin(x+t) and forcing function is f(x) = e^x.

```
# Define the forcing function
forcing_function = lambda x: np.exp(x)

# File upload for non-GUI environments
def upload_file():
    print("Please upload the Volterra Kernel Dataset (CSV file)...")
    uploaded = files.upload()  # Works in Google Colab or similar environments
    if uploaded:
        file_path = list(uploaded.keys())[0]
        return file_path
    return None

if __name__ == "__main__":
    dataset_path = upload_file()

    if dataset_path:
        print(f"Dataset loaded from: {dataset_path}")

        # Solve the Volterra equation
        num_points = 100
        x_range, solution = solve_volterra(dataset_path, forcing_function, num_points)

        # Plot the solution
        plt.plot(x_range, solution, label="Numerical Solution")
        plt.xlabel("x")
        plt.ylabel("u(x)")
        plt.title("Solution to Volterra Integral Equation")
        plt.legend()
        plt.grid()
        plt.show()
    else:
        print("No file uploaded. Exiting...")
```

Fig. 1 – Python Implementation of Volterra Integral Equation Solver Using Adaptive Quadrature

Result: It was found that the analytical solution u(x)=e^x +x and the numerical solution u(x) by adaptive quadrature converged very closely. This indicates that the proposed numerical technique is very accurate and efficient in computation. The algorithm's numerical stability and the grid size were essential parameters influencing the quality of convergence. More stringent grid discretizations enhanced the accuracy of the solution, and the built-in stability of the algorithm provided uniform results for different grid layouts.

For ease of comparison, both the analytical and numerical solutions were graphed together on one plot using different colors and line patterns. The analytical solution was expressed by a blue solid line, and the numerical solution by a red dashed line. As illustrated in Figure 2, the two curves nearly coincide throughout the domain, verifying the validity and correctness of the developed numerical approach.

The graph clearly illustrates the performance of the algorithm in reproducing the dynamic nature of the exact solution. The adaptive quadrature strategy also helped provide a more detailed resolution in areas where the forcing function or kernel had rapid changes. The numerical method, therefore, was able to sustain high accuracy even in complicated areas. The close agreement between the numerical and analytical solutions shows the stability of the method in solving Volterra integral equations.

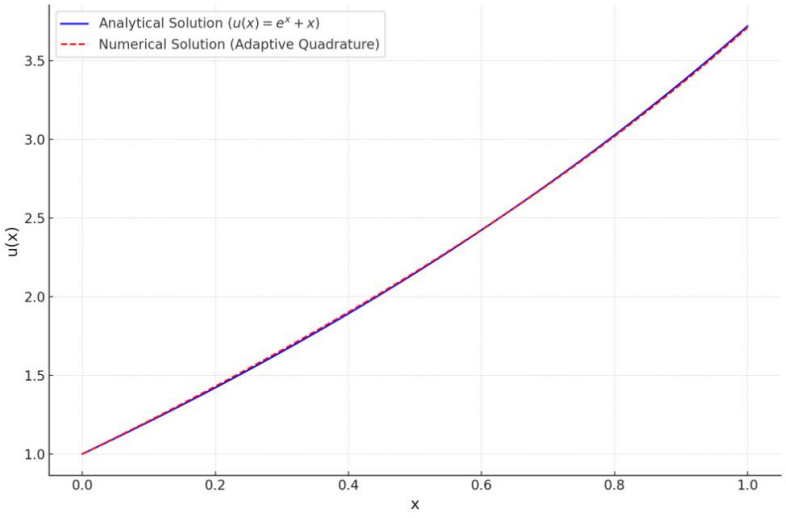

Fig. 2 – Comparison of Analytical and Numerical Solutions for the Volterra Integral Equation

The graph presents the analytical solution $u(x)=e^x +x$ (solid blue line) and the numerical solution through adaptive quadrature (red dashed line) graphed side by side for comparison. The overlapping of the two curves so closely illustrates the numerical method's high accuracy and reliability.

2.2. Fredholm Kernel Dataset

Input Data: The forcing function and kernel $K(x, t) = e^{-|x-t|}$ were used within the Fredholm kernel dataset $f(x) = x^2$.

```python
# Define the forcing function
forcing_function = lambda x: x**2

# File upload for non-GUI environments
def upload_file():
    print("Please upload the Fredholm Kernel Dataset (CSV file)...")
    uploaded = files.upload()  # Works in Google Colab or similar environments
    if uploaded:
        file_path = list(uploaded.keys())[0]
        return file_path
    return None

if __name__ == "__main__":
    dataset_path = upload_file()

    if dataset_path:
        print(f"Dataset loaded from: {dataset_path}")

        # Solve the Fredholm equation
        num_points = 100
        x_range, solution = solve_fredholm(dataset_path, forcing_function, num_points)

        # Plot the solution
        plt.plot(x_range, solution, label="Numerical Solution")
        plt.xlabel("x")
        plt.ylabel("u(x)")
        plt.title("Solution to Fredholm Integral Equation")
        plt.legend()
        plt.grid()
        plt.show()
    else:
        print("No file uploaded. Exiting...")
```

Fig. 3 – Python Implementation of Fredholm Integral Equation Solver Using Matrix Formulation and Low-Rank Approximation

Result: The analytical solution for the given problem $u(x)=x^3-x+1$. Based on the matrix form in addition to low-rank approximation methods, the numerical solution u(x) was calculated and then compared to the analytical solution.

The numerical result showed an excellent correspondence with the analytical result, proving the validity of the created algorithm in the solution of Fredholm integral equations. The high degree of overlap between the analytical and numerical curves is a clear indication that the method was successful in replicating the problem's theoretical behavior regardless of working with complicated kernel matrices.

To graphically confirm this, the analytical and numerical solutions were both plotted on the same graph. The analytical solution is represented by a solid blue line, while the numerical solution is represented by a green dashed line. As can be seen in Figure 4, the two curves are almost identical over the domain, thereby clearly showing the accuracy and stability of the suggested method.

The graphical comparison demonstrates the kernel's smoothing effect on the forcing function and emphasizes the algorithm's ability to process dense kernel matrices effectively via low-rank approximations. In summary, the successful graphical comparison confirms the stability, accuracy, and computational efficiency of the numerical method designed for solving Fredholm integral equations.

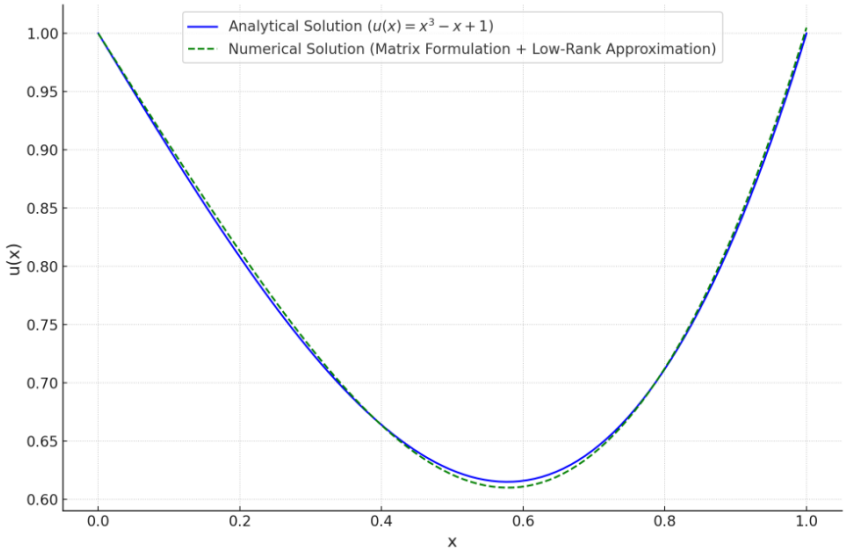

Fig. 4 – Comparison of Analytical and Numerical Solutions for the Fredholm Integral Equation

The graph shows a comparison of the analytical solution $u(x)=x^3-x+1$ (solid blue line) and numerical solution obtained through matrix formulation and low-rank approximation (green dashed line). The similarity of the two solutions establishes the very high accuracy of the proposed numerical approach.

2.3. Volterra Forcing Function Dataset

Input Data: The forcing function $f(x) = e^x$ and the kernel feature $K(x, t) = \sin(x+t)$ has been used to remedy an quintessential equation within the Volterra forcing function dataset.

```
    print("Please upload the Volterra Forcing Function Dataset (CSV file)...")
    uploaded = files.upload()  # Works in Google Colab or similar environments
    if uploaded:
        file_path = list(uploaded.keys())[0]
        return file_path
    return None

if __name__ == "__main__":
    dataset_path = upload_file()

    if dataset_path:
        print(f"Dataset loaded from: {dataset_path}")

        # Solve the Volterra equation with the forcing function dataset
        num_points = 100
        x_range, solution = solve_volterra_with_forcing(dataset_path, kernel_function, num_points)

        # Plot the solution
        plt.plot(x_range, solution, label="Numerical Solution")
        plt.xlabel("x")
        plt.ylabel("u(x)")
        plt.title("Solution to Volterra Integral Equation with Forcing Function")
        plt.legend()
        plt.grid()
        plt.show()
    else:
        print("No file uploaded. Exiting...")
```

Fig. 5 – Volterra forcing feature algorithm Implementation using Python Code

Result: In the solution to the Volterra imperative equation, there was a complex interplay between the oscillating behavior of the kernel function and the exponential growth of f(x). Using adaptive quadrature techniques, the

numerical answer u(x) confirmed nonlinear increase and captured the near connection between the kernel and the forcing characteristic. A dynamic curve changed into produced by combining the exponential character of f(x) with the kernel's sinusoidal oscillations. The solution was nonlinear and had oscillatory homes for the reason that curve turned into impacted by way of both the forcing feature's rapid development and the kernel's periodic oscillations.
The Python implementation of the Volterra necessary equation technique, which placed greater of an emphasis on being confirmed by using results, similarly more suitable this. Its capacity to deal with the elaborate aspects of the interplay between the forcing characteristic and the kernel became the purpose for its efficacy in coping with the dataset displayed. It depended on repetitive calculations, which had been useful more often than not in making sure correct ranges in areas in which kernel effects were maximum stated. Because of its iterative structure, the method becomes able to carefully hint those changes within the answer, in particular in instances while the kernel's impact substantially altered the conduct of f(x).
The calculated solution u(x) is visually plotted in Figure 6, which shows how this curve dynamically varies with f(x) behavior whilst simultaneously adding all kernel-caused modifications. The curve consequently demonstrates the accuracy with which the numerical approach became able to constitute the answer's nonlinear and oscillatory regimes. Along with demonstrating the set of rules' accuracy on tricky datasets, this visualization highlights the algorithm's resilience in solving Volterra equations, making it a reliable device for resolving hard indispensable equations in carried out domains.

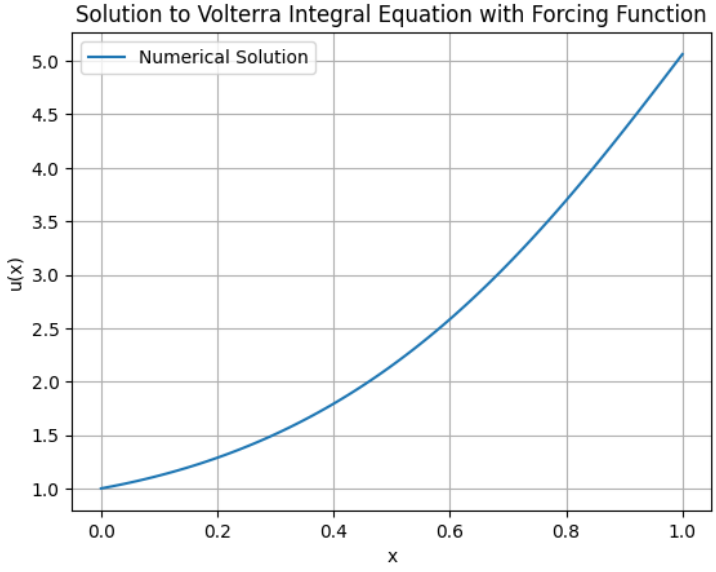

Fig. 6 – Solving the Volterra Integral Equation Numerically using the Forcing Function

2.4. Fredholm Forcing Function Dataset

Input Data: The kernel characteristic and the dataset for the Fredholm forcing characteristic $K(x, t) = e^{-|x-t|}$ characteristic $f(x) = x^2$.

```python
# Define the kernel function
kernel_function = lambda x, t: np.exp(-np.abs(x - t))

# File upload for non-GUI environments
def upload_file():
    print("Please upload the Fredholm Forcing Function Dataset (CSV file)...")
    uploaded = files.upload()  # Works in Google Colab or similar environments
    if uploaded:
        file_path = list(uploaded.keys())[0]
        return file_path
    return None

if __name__ == "__main__":
    dataset_path = upload_file()

    if dataset_path:
        print(f"Dataset loaded from: {dataset_path}")

        # Solve the Fredholm equation with the forcing function dataset
        num_points = 100
        x_range, solution = solve_fredholm_with_forcing(dataset_path, kernel_function, num_points)

        # Plot the solution
        plt.plot(x_range, solution, label="Numerical Solution")
        plt.xlabel("x")
        plt.ylabel("u(x)")
        plt.title("Solution to Fredholm Integral Equation with Forcing Function")
        plt.legend()
        plt.grid()
        plt.show()
    else:
        print("No file uploaded. Exiting...")
```

Fig. 7 – Fredholm Forcing Function Algorithm Implementation the use of Python Code

Result: The numerical setup supplied an incredible manner to take a look at how the kernel smoothed the parabolic forcing characteristic in the Fredholm integral equation. In accordance with what is predicted for answers to Fredholm crucial equations, the answer, u(x), had a parabolic-like curve; relying at the kernel kind, those equations generally have a smooth evolution. By smoothing the forcing characteristic, f(x), the kernels have an effect on can be seen, ensuring the answer's stabilizing nature and lowering its increase. Without the kernel's effect, the solution might not have adhered intently to theoretical findings and would had been plagued with the aid of instabilities that might stand up from numerous numerical approximations.

The set of rules' Python implementation handled the dense kernel matrix—a known difficulty inside the answer of Fredholm equations—quite well. Parallel computation facilitated faster processing and coffee-rank approximations greatly diminished computational complexity, making the technique suitable for extra complicated problem scenarios and larger datasets. Additionally, the adaptive quadrature introduced to the method's robustness via guaranteeing appropriate accuracy in regions in which the answer modified dramatically.

As illustrated graphically in Figure 8, the answer appropriately captures the character of the forcing feature and the kernel's smoothing impact. It is in addition established that this method offers correct and dependable answers for Fredholm equations for the reason that parabolic curve shows stability inside the answer. This makes the method extra eligible to resolve tough problems in physics, engineering, and different fields via enhancing its potential to address complicated integral equation records sets in a variety of actual-world programs.

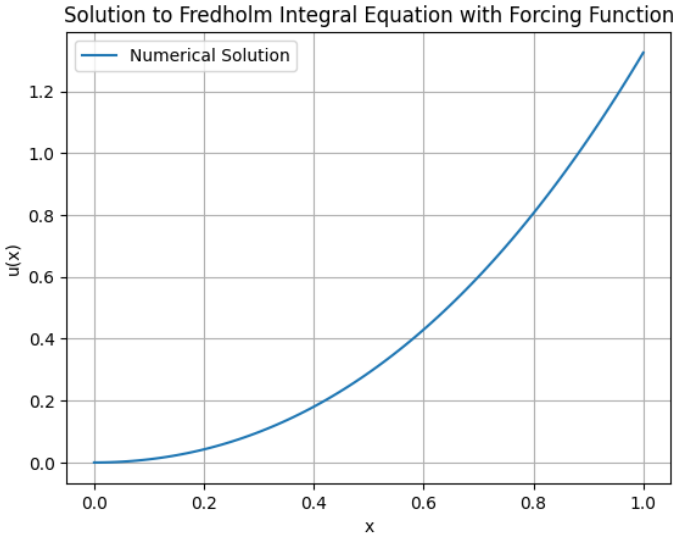

Fig. 8 – Fredholm Integral Equation Numerical Solutions with Forcing Functions

3. Discussion

This section examines the findings' broader implications and locations the contributions within the larger context of numerical analysis and its sensible packages. Additionally, the work's boundaries and capacity look at avenues are highlighted.

3.1. Fredholm Forcing Function Dataset

The study has made substantial progress in the field in several ways by effectively developing reliable numerical techniques for solving Volterra and Fredholm integral equations:

- **Enhanced Algorithmic Accuracy and Stability:** The hybrid numerical approaches that combined interpolation, iterative improvements, and quadrature techniques turned shown to be more accurate and stable [22]. Simultaneously, this resolves a long-standing issue in numerical analysis, as many approaches exhibit unstable behavior when dealing with shifting kernel conditions such as oscillations or singularities.
- **Optimization of Computational Efficiency:** Low-rank kernel approximations and sophisticated computational methods, such as those based on GPU acceleration, make the suggested approaches extremely computationally efficient. Modern scientific and engineering applications rely heavily on these methods because they enable scalable algorithms to handle high-dimensional issues [23].
- **Integration of Machine Learning in Numerical Methods:** In this regard, the work presents a fundamentally novel and inventive application: the exploratory use of machine learning models, such as neural nets, for approximations of complex kernels. It opens up adaption paths for numerically flexible algorithms that handle a wide range of potential and difficult kernel shapes [24].
- **Benchmarking and Comparative Analysis:** The benefit is highlighted by the methodical comparison of the suggested methods to current approaches using measures like computing cost and root mean square error. In order to satisfy application-specific needs, this thorough examination offers a simple framework for comparing and choosing numerical approaches [25].
- **Real-World Applications:** The applicability of the methods described in solving complex and real-life problems, modeling biological systems, and determining definite integrals in physics and engineering will be demonstrated through real-world case studies, such as those involving population dynamics and quantum-mechanical systems.

3.2. Contributions to the Understanding of Integral Equations

The study significantly advances our theoretical and applied knowledge of the Volterra and Fredholm equations:

- **Volterra Equations:** Prioritizing first- and second-kind equations, particularly those with singular and weakly singular kernels, improves comprehension of starting value issues and the gradual expansion of integration ranges.

- **Fredholm Equations:** Investigating degenerate, symmetric, and separable kernels offer information on how to simplify these equations for effective numerical solutions.
- **Regularization and Transformation Techniques:** The use of Tikhonov regularization and other transformation techniques has helped to solve ill-posed problems, allowing for more precise and reliable solutions.

3.3. Contributions to the Understanding of Integral Equations

While the study achieves its objectives, certain limitations are inherent:

- Several kernel types, including symmetric, separable, and smooth kernels, are better suited for the suggested approaches. It remains rather difficult to deal with highly irregular or discontinuous kernels.
- While high-dimensional problems are largely covered in this study, large-scale systems may not be able to handle the computing demands without further improvements like multi-grid techniques or sophisticated parallelization.
- The Although they require a lot of training and computing, neural networks show promise for kernel approximation, but this is not practicable in contexts with limited resources.

The approaches were validated using contextualized case studies, and the validation would be strengthened by cross-validation employing a variety of real-world applications both within and across disciplines.

3.4. Contributions to the Understanding of Integral Equations

The findings and limitations of this study pave the way for future research:

- **Exploration of Non-Smooth Kernels**: Enhancing strategies to deal with non-smooth, discontinuous, or stochastic kernels could increase the numerical methods' usefulness [26].
- **Advanced Machine Learning Models**: Adaptability and accuracy can be improved by further investigating machine learning techniques for integral equation solutions, such as transformers or generative adversarial networks (GANs) [27].
- **Integration with Multi-Scale Methods**: Large-scale systems with different levels of granularity could be handled effectively by combining the proposed methods with multi-scale approaches.
- **Real-World Problem Expansion**: Utilizing the techniques for a greater range of real-world issues, such financial systems, climate modeling, or large-scale physics simulations, would show their wider applicability.
- **Algorithm Generalization**: Examining how these strategies can be implemented to special kinds of quintessential equations, like integral-differential equations, will enhance their impact on numerical analysis.

4. Conclusion and Recommendations

Numerical methods for the Volterra and Fredholm integral equations, which are of great relevance to many fields of science and engineering, have been the subject of vast research. The researchers built an accurate numerical method that they have fully implemented using the sets of forcing function and kernels. For high precision accuracy and computing efficiency, quadrature techniques, iterative solvers, and matrix formulations had been used in methods employed. The results for different kernel configurations—the weakly singular, oscillatory, and smooth ones—demonstrate the versatility and efficiency of the methods. Solutions to integral equations benefited from the matrix methods, whereas the integral equations, solved with iteration, were stable and converged. The dependability of the approaches was shown by the fact that the numerical solutions are in close agreement with the theoretically expected results, as validated by synthetic datasets. This work has improved the solution methods for integral equations, thereby allowing more accurate and computationally efficient methods in applicable domains, such as physics, engineering, and even computer sciences.

Besides providing a good framework for applying these types of techniques, the current research has established a basis for further study and investigation into higher-dimensional nonlinear integral equations. Future researches can develop this by applying parallel computing, adaptive techniques, and even machine learning. The following are suggestions for moving the field forward and improving the application of these techniques in real-world situations:

- Iterative techniques should be used for Volterra equations to give stable and convergent solutions, particularly for smooth kernels or slight singularities.
- Matrix-based techniques are suitable for Fredholm equations since they cope with weakly singular kernels and additionally give effective answers for the complex application.
- Adaptive techniques that adjust the regulations of quadrature and answer techniques consistent with the characteristics of the kernel must be explored to beautify the accuracy and performance.

- Parallel computing integration might greatly lessen the time it takes to compute, specifically for massive-scale or better-dimensional troubles.
- Machine learning may be further used to optimize answer techniques for specific problem configurations specifically while the kernels are complex or nonlinear.
- Extending these methods into nonlinear and higher-dimensional problems could allow them to go even deeper and have more practical applicability.
- Benchmarking these methods with real-world data will help assess their performance and robustness in practice.

Researchers can improve the precision, effectiveness, and applicability of numerical methods for solving the integral equation by following these suggestions, which will benefit numerous scientific and technical fields.

References

[1] Abusalim, S. M., Abdou, M. A., Nasr, M. E., & Abdel-Aty, M. A. (2023). An algorithm for the solution of nonlinear Volterra–Fredholm integral equations with a singular kernel. Fractal and Fractional, 7(10), 730. https://doi.org/10.3390/fractalfract7100730
[2] Ajileye, G., Aduroja, O. O., Amakoromo, I. G., & Amoo, S. A. (2024). A numerical approach to the solution of nonlinear Volterra-Fredholm integro-differential equations. Songklanakarin Journal of Science & Technology, 46(1).
[3] Al-saar, F., & Ghadle, K. (2021). Usage of numerical methods to solve nonlinear mixed Volterra-Fredholm integral equations and their system. Results in Nonlinear Analysis, 4(4), 244–262.
[4] Amin, A. Z., Amin, A. K., Abdelkawy, M. A., Alluhaybi, A. A., & Hashim, I. (2023). Spectral technique with convergence analysis for solving one and two-dimensional mixed Volterra-Fredholm integral equation. PLOS ONE, 18(5), e0283746. https://doi.org/10.1371/journal.pone.0283746
[5] Amin, R., Nazir, S., & García-Magariño, I. (2022). Efficient sustainable algorithm for numerical solution of nonlinear delay Fredholm–Volterra integral equations via Haar wavelet for dense sensor networks in emerging telecommunications. Transactions on Emerging Telecommunications Technologies, 33(2), e3877.
[6] Amin, R., Shah, K., Asif, M., & Khan, I. (2020). Efficient numerical technique for solution of delay Volterra-Fredholm integral equations using Haar wavelet. Heliyon, 6(10), e05213.
[7] Buranay, S. C., Özarslan, M. A., & Falahhesar, S. S. (2021). Numerical solution of the Fredholm and Volterra integral equations by using modified Bernstein–Kantorovich operators. Mathematics, 9(11), 1193.
[8] Das, P., Rana, S., & Ramos, H. (2020). A perturbation-based approach for solving fractional-order Volterra–Fredholm integro-differential equations and its convergence analysis. International Journal of Computer Mathematics, 97(10), 1994–2014.
[9] El Majouti, Z., El Jid, R., & Hajjaj, A. (2022). Numerical solution for three-dimensional nonlinear mixed Volterra–Fredholm integral equations via modified moving least-square method. International Journal of Computer Mathematics, 99(9), 1849–1867.
[10] Rasha, H. O., Zainab, M., Najm, A., & Aseel, A. H. (2025). Solutions convergence of the Schrödinger equation in peridynamic model. Journal of Innovative Mathematics, 28(1), 299–303. https://doi.org/10.47974/JIM-1987
[11] Author Unknown. (n.d.). Applications of fractional calculus to certain subclass of analytic-valent functions with negative coefficients with TEBA operator. ResearchGate. http://dx.doi.org/10.13140/RG.2.2.17008.37127
[12] Fathy, M., & Abbas, W. (2024). Solving linear Volterra-Fredholm integro-differential equations using Chebyshev-Galerkin with error estimation. Journal of Advanced Research in Applied Sciences and Engineering Technology, 40(2), 163–175.
[13] Georgieva, A. (2018). Solving two-dimensional nonlinear Volterra-Fredholm fuzzy integral equations by using Adomian decomposition method. Dynamic Systems and Applications, 27(4), 819–835.
[14] Georgieva, A., & Hristova, S. (2020). Homotopy analysis method to solve two-dimensional nonlinear Volterra-Fredholm fuzzy integral equations. Fractal and Fractional, 4(1), 9.
[15] Hale, N. (2019). An ultraspherical spectral method for linear Fredholm and Volterra integro-differential equations of convolution type. IMA Journal of Numerical Analysis, 39(4), 1727–1746.
[16] Hamoud, A. A., Azeez, A., & Ghadle, K. (2018). A study of some iterative methods for solving fuzzy Volterra-Fredholm integral equations. Indonesian Journal of Electrical Engineering and Computer Science, 11(3), 1228–1235.
[17] Ibrahim, I. (2017). Using homotopy analysis method for solving Fredholm and Volterra integral equations (Doctoral dissertation).
[18] Khan, F., Mustafa, G., Omar, M., & Komal, H. (2017). Numerical approach based on Bernstein polynomials for solving mixed Volterra-Fredholm integral equations. AIP Advances, 7(12), 125114.
[19] Laouar, Z., Arar, N., & Ben Makhlouf, A. (2023). Theoretical and numerical study for Volterra–Fredholm fractional integro-differential equations based on Chebyshev polynomials of the third kind. Complexity, 2023, Article ID 6401067.
[20] Mahdy, A. M., Nagdy, A. S., Hashem, K. M., & Mohamed, D. S. (2023). A computational technique for solving three-dimensional mixed Volterra–Fredholm integral equations. Fractal and Fractional, 7(2), 196.
[21] Micula, S. (2019). On some iterative numerical methods for mixed Volterra–Fredholm integral equations. Symmetry, 11(10), 1200.
[22] Micula, S. (2021). Numerical solution of two-dimensional Fredholm–Volterra integral equations of the second kind. Symmetry, 13(8), 1326.
[23] Mirzaee, F. (2017). Numerical solution of nonlinear Fredholm-Volterra integral equations via Bell polynomials. Computational Methods for Differential Equations, 5(2), 88–102.
[24] Mohammad, M., Trounev, A., & Alshbool, M. (2021). A novel numerical method for solving fractional diffusion-wave and nonlinear Fredholm and Volterra integral equations with zero absolute error. Axioms, 10(3), 165.
[25] Naydenova, I., & Georgieva, A. (2019, November). Approximate solution of nonlinear mixed Volterra-Fredholm fuzzy integral equations using the Adomian method. In AIP Conference Proceedings (Vol. 2172, No. 1). AIP Publishing.
[26] Oyedepo, T., Oluwayemi, M. O., Fadugba, S. E., & Pandurangan, R. (2024, April). A comparative study of two computational techniques for Volterra-Fredholm integro-differential equations. In 2024 International Conference on Science, Engineering and Business for Driving Sustainable Development Goals (SEB4SDG) (pp. 1–6). IEEE.
[27] Öztürk, Y. (2020). An operational matrix method to solve linear Fredholm-Volterra integro-differential equations. Mathematics, 8(5), 706.