Int. J. Math. And Appl., 8(1)(2020), 7–32
ISSN: 2347-1557
Available Online: http://ijmaa.in/

International Journal of *Mathematics And its Applications*

Numerical Solution of Linear and Nonlinear Integral and Integro-Differential Equations using Biorthogonal Spline Wavelet Transform Method

R. A. Mundewadi[1,*] and B. A. Mundewadi[2]

1 Department of Mathematics, M.E.S College of Arts, Commerce and Science, Bangalore, Karnataka, India.

2 Department of Mathematics, Government First Grade College for Women's, Bagalkot, Karnataka, India.

Abstract: Biorthogonal spline wavelet method is proposed for the numerical solution of Linear and nonlinear integral and integro-differential equations. Biorthogonal spline wavelet filter coefficients have the prolongation and restriction operators. The performance of the proposed method is better than the existing ones in terms of super convergence with low computational time. Some of the test problems are demonstrated for the applicability and efficiency of the scheme.

MSC: 65D30, 45B05.

Keywords: Biorthogonal spline wavelets; Filter coefficients; Multigrid Method; Full-approximation scheme; Integral equations; Integro-differential equations.

1. Introduction

Integral and integro-differential equations emerge normally in numerous applications in different fields of science and engineering and furthermore have been concentrated broadly both at the hypothetical and functional level. Specific applications of integral and integro-differential equations can be found in the mathematical modelling of spatiotemporal developments, epidemic modelling [1] and various biological and physical problems. Analytical solutions of integral and integro-differential equations, however, either do not exist or it is often hard to find. It is precisely due to this fact that several numerical methods have been developed for finding approximate solutions of integral and integro-differential equations [2–4].
Multigrid method is well known among the fastest solution method. Particularly, for elliptic problems, they have been proved to be highly accurate. In classical multigrid method pioneered by Brandt [10], a solution to

$$Lu = f$$

is sought, where L is self-adjoint operator on a fine grid Ω^h (with grid spacing h) by using standard relaxation (such as GaussSeidel) method, to approximate errors on the coarse grids Ω^{2h}, Ω^{4h},..., a hierarchy of grids with grid spacing that is increased by a factor of two. The details of the various cycles are not important, for our discussion, except to say that, it is the residue that is passed from the fine grids to the coarser grids. Vectors from fine grids are transferred to

* *E-mail: rkmundewadi@gmail.com*

coarser grids with Restriction operator, while vectors are transferred from coarse grids to the finer grids with a Prolongation operator. An introduction of multigrid method is found in Wesseling [5]. Multigrid Tutorial (Briggs [6] and Trottenberg et al. [7]), is helpful to get the basic ideas of multigrid techniques. The multigrid method is largely applicable in increasing the efficiency of iterative methods used to solve large system of algebraic equations resulted from discretization of the differential equations and integral equations are applicable to solve numerically. Most importantly, differential equations produce sparse matrix equation upon discretization, while integral equations typically result in dense matrices yielding more expensive numerical solution. Multigrid method was initially developed to solve differential equations more efficiently than other existing numerical techniques [8–11]. Multigrid method has been applied for the numerical solution of different types of integral equations. Hackbusch [12, 13] given the multigrid techniques and the integral equations from both theoretical and computational points of views. Schippers [14, 15] used multigrid methods for boundary integral equations. Gspr [16] has given a new approach a fast multigrid solution of boundary integral equations. Lee [17] has solved multigrid method for nonlinear integral equations. Paul [18], applied the multigrid algorithm for solving integral equations. Wavelet based numerical methods for solving integral and integro-differential equations introduced in [39–42]. In the historical three decades the development of effective iterative solvers for nonlinear systems of algebraic equations has been a significant research topic in numerical analysis, computational science and engineering. Brandt [10] was one of the first to introduce nonlinear multigrid method, which seeks to use concepts from the linear multigrid iteration and apply them directly in the nonlinear setting. Applying multigrid method directly to the nonlinear problems by employing the method so-called Full Approximation Scheme (FAS). In FAS, a nonlinear iteration, such as the nonlinear Gauss-Seidel method is applied to smooth the error and the residual is passed from the fine grids to the coarser grids. For a detailed treatment of FAS is given in Briggs et al. [6]. An introduction of FAS is found in Hackbusch and Trottenberg [19], Wesseling [5] and Trottenberg et al. [7]. Many authors applied the FAS for some class of differential equations. The full-approximation scheme (FAS) is largely applicable in increasing the efficiency of the iterative methods used to solve nonlinear system of algebraic equations. FAS are a well-founded numerical method for solving nonlinear system of equations for approximating given differential equation. Subsequently, the development of multiresolution analysis and the fast wavelet transforms by Avudainayagam and Vani [20] led to extensive research in wavelet multigrid schemes to solve certain differential equations arising in fluid dynamics. Lee [21] has introduced a multigrid method for solving the nonlinear Urysohn integral equations.

Wavelet analysis is a new branch of mathematics and widely applied in signal analysis, image processing and numerical analysis etc. The wavelet methods have proved to be very effective and efficient tool for solving problems of mathematical calculus. In recent years, these methods have attracted the interest of researchers of structural mechanics and many papers in this field are published. In most of the papers the Daubechies wavelets are applied. These wavelets are orthogonal, sufficiently smooth and have a compact support. Their shortcoming is that an explicit expression is lacking. This obstacle makes the differentiation and integration of these wavelets very complicated. For evaluation of such integrals the connection coefficients are introduced, but this complicates the course of the solution to a great extent [22]. Biorthogonal wavelet basis were introduced by Cohen-Daubechies-Feauveau in order to obtain wavelet pairs that are symmetric, regular and compactly supported. Unfortunately, this is incompatible with the orthogonality requirement that has to be dropped altogether. Biorthogonal wavelets build with splines are especially attractive because of their short support and regularity. So it is called a Biorthogonal Spline Wavelets [23]. In the biorthogonal case, rather than having one scaling and wavelet function, there are two scaling functions, that may generate different multiresolution analysis, and accordingly two different wavelet functions. But biorthogonal wavelet based multigrid schemes are found to be effective [24]. Biorthogonal wavelet based multigrid schemes provide some remedy in such challenging cases. Sweldens [25] highlights effectively the construction of biorthogonal wavelet filters for the solution of large class of ill-conditioned system. In this paper, we developed the

biorthogonal spline wavelet full-approximation transform method (BSWFATM) for the numerical solution of nonlinear integral and integro-differential equations using discrete biorthogonal spline wavelet transform (DBSWT) matrix. This matrix designed and implemented by Ruch and Fleet [26, 27] for decomposition and reconstruction of the given signals and images. Using these decomposition and reconstruction matrices we introduced restriction and prolongation operators respectively in the implementation of biorthogonal spline wavelet full-approximation transform method (BSWFATM).

2. Properties of Biorthogonal Wavelets

Discrete Biorthogonal Spline wavelet transform (DBSWT) matrix:

Let us consider the (5, 3) biorthogonal spline wavelet filter pair, We have

$$\tilde{c} = (\tilde{c}_{-1}, \tilde{c}_0, \tilde{c}_1) = \left(\frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{4}\right)$$

and

$$c = (c_{-2}, c_{-1}, c_0, c_1, c_2) = \left(\frac{-\sqrt{2}}{8}, \frac{\sqrt{2}}{4}, \frac{3\sqrt{2}}{4}, \frac{\sqrt{2}}{4}, \frac{-\sqrt{2}}{8}\right)$$

To form the highpass filters, We have

$$d_k = (-1)^k \tilde{c}_{1-k} \quad and \quad \tilde{d}_k = (-1)^k c_{1-k}$$

The highpass filter pair d and for the (5, 3) biorthogonal spline filter pair.

$$d_0 = \frac{\sqrt{2}}{4}, d_1 = \frac{-\sqrt{2}}{2}, d_2 = \frac{\sqrt{2}}{4} \quad and \quad \tilde{d}_{-1} = \frac{\sqrt{2}}{8}, \tilde{d}_0 = \frac{\sqrt{2}}{4}, \tilde{d}_1 = \frac{-3\sqrt{2}}{4}, \tilde{d}_2 = \frac{\sqrt{2}}{4}, \tilde{d}_3 = \frac{\sqrt{2}}{8}$$

In this paper, we use the filter coefficients which are, low pass filter coefficients: $c_{-2}, c_{-1}, c_0, c_1, c_2$ and High pass filter coefficients: d_0, d_1, d_2 for decomposition matrix.

Low pass filter coefficients: $\tilde{c}_{-1} = d_2, \tilde{c}_0 = -d_1, \tilde{c}_1 = d_0$ and High pass filter coefficients: $\tilde{d}_{-1} = -c_2, \tilde{d}_0 = c_1, \tilde{d}_1 = -c_0, \tilde{d}_2 = c_{-1}, \tilde{d}_3 = -c_{-2}$ for reconstruction matrix.

The matrix formulation of the discrete biorthogonal spline wavelet transforms (DBSWT) plays an important role in both biorthogonal spline wavelet transforms method (BSWTM) and biorthogonal Spline wavelet full-approximation transform method (BSWFATM) for the numerical computations. As we already know about the DBSWT matrix and its applications in the wavelet method and is given in [26] as,

Decomposition matrix:

$$D_W = \begin{pmatrix} c_{-1} & c_0 & c_1 & c_2 & 0 & 0 & \cdots & 0 & 0 & c_{-2} \\ d_1 & d_2 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & d_0 \\ 0 & c_{-2} & c_{-1} & c_0 & c_1 & c_2 & \cdots & 0 & 0 & 0 \\ 0 & d_0 & d_1 & d_2 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots & 0 & 0 & 0 \\ c_1 & c_2 & 0 & 0 & \cdots & \cdots & 0 & c_{-2} & c_{-1} & c_0 \\ 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & d_0 & d_1 & d_2 \end{pmatrix}_{N \times N}$$

Reconstruction matrix:

$$R_W = \begin{pmatrix} \tilde{c}_0 & \tilde{c}_1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \tilde{c}_{-1} \\ \tilde{d}_0 & \tilde{d}_1 & \tilde{d}_2 & \tilde{d}_3 & 0 & 0 & \cdots & 0 & 0 & \tilde{d}_{-1} \\ 0 & \tilde{c}_{-1} & \tilde{c}_0 & \tilde{c}_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \tilde{d}_{-1} & \tilde{d}_0 & \tilde{d}_1 & \tilde{d}_2 & \tilde{d}_3 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & \tilde{c}_{-1} & \tilde{c}_0 & \tilde{c}_1 \\ \tilde{d}_2 & \tilde{d}_3 & 0 & 0 & \cdots & \cdots & 0 & \tilde{d}_{-1} & \tilde{d}_0 & \tilde{d}_1 \end{pmatrix}_{N\times N}$$

Biorthogonal Spline Wavelet operators: Using the above matrices, we introduced biorthogonal spline wavelet restriction and biorthogonal spline wavelet prolongation operators respectively. i.e.,

Biorthogonal spline wavelet restriction operator:

$$BSWT_R = \begin{pmatrix} c_{-1} & c_0 & c_1 & c_2 & 0 & 0 & \cdots & 0 & 0 & c_{-2} \\ d_1 & d_2 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & d_0 \\ 0 & c_{-2} & c_{-1} & c_0 & c_1 & c_2 & \cdots & 0 & 0 & 0 \\ 0 & d_0 & d_1 & d_2 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & d_0 & d_1 & d_2 & 0 & \cdots & 0 \end{pmatrix}_{\frac{N}{2}\times N}$$

Biorthogonal spline wavelet prolongation operator:

$$BSWT_P = \begin{pmatrix} \tilde{c}_0 & \tilde{c}_1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \tilde{c}_{-1} \\ \tilde{d}_0 & \tilde{d}_1 & \tilde{d}_2 & \tilde{d}_3 & 0 & 0 & \cdots & 0 & 0 & \tilde{d}_{-1} \\ 0 & \tilde{c}_{-1} & \tilde{c}_0 & \tilde{c}_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \tilde{d}_{-1} & \tilde{d}_0 & \tilde{d}_1 & \tilde{d}_2 & \tilde{d}_3 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots & 0 & 0 & 0 \\ 0 & \cdots & 0 & \tilde{c}_{-1} & \tilde{c}_0 & \tilde{c}_1 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & \tilde{d}_{-1} & \tilde{d}_0 & \tilde{d}_1 & \tilde{d}_2 & \tilde{d}_3 & 0 & \cdots & 0 \end{pmatrix}_{\frac{N}{2}\times N}$$

Modified Discrete Biorthogonal Spline wavelet transform (MDBSWT) matrix:

Here, we developed MDBSWT matrix from DBSWT matrix in which by adding rows and columns consecutively with diagonal element as 1, which is built as,

New decomposition matrix:

$$MD_W = \begin{pmatrix} c_{-1} & 0 & c_0 & 0 & c_1 & 0 & c_2 & 0 & \cdots & 0 & 0 & 0 & c_{-2} & 0 \\ 0 & 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 & 0 \\ d_1 & 0 & d_2 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 & 0 & d_0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \ddots & \ddots & \vdots \\ c_1 & 0 & c_2 & 0 & \cdots & 0 & 0 & 0 & c_{-2} & 0 & c_{-1} & 0 & c_0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & d_0 & 0 & d_1 & 0 & d_2 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 & 1 \end{pmatrix}_{N\times N}$$

New reconstruction matrix:

$$MR_W = \begin{pmatrix} \tilde{c}_0 & 0 & \tilde{c}_1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & \tilde{c}_{-1} & 0 \\ 0 & 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 \\ \tilde{d}_0 & 0 & \tilde{d}_1 & 0 & \tilde{d}_2 & 0 & \tilde{d}_3 & \cdots & 0 & 0 & \tilde{d}_{-1} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots & \cdots & \cdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & 0 & \tilde{c}_{-1} & 0 & \tilde{c}_0 & 0 & \tilde{c}_1 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 1 & 0 & 0 \\ \tilde{d}_2 & 0 & \tilde{d}_3 & \cdots & 0 & 0 & \tilde{d}_{-1} & 0 & \tilde{d}_0 & 0 & \tilde{d}_1 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 & 1 \end{pmatrix}_{N\times N}$$

Modified Biorthogonal Spline wavelet operators:

Using the above matrices, we introduced a new biorthogonal spline wavelet restriction and prolongation operators respectively as,

New biorthogonal spline wavelet restriction operator:

$$MBSWT_R = \begin{pmatrix} c_{-1} & 0 & c_0 & 0 & c_1 & 0 & c_2 & 0 & \cdots & 0 & 0 & 0 & c_{-2} & 0 \\ 0 & 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 & 0 \\ d_1 & 0 & d_2 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 & 0 & d_0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_{-2} & 0 & c_{-1} & 0 & c_0 & 0 & c_1 & 0 & c_2 & 0 & \cdots & 0 \\ 0 & 0 & d_0 & 0 & d_1 & 0 & d_2 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & d_0 & 0 & d_1 & 0 & d_2 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}_{\frac{N}{2}\times N}$$

New biorthogonal spline wavelet prolongation operator:

$$MBSWT_P = \begin{pmatrix} \tilde{c}_0 & 0 & \tilde{c}_1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & \tilde{c}_{-1} & 0 \\ 0 & 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 \\ \tilde{d}_0 & 0 & \tilde{d}_1 & 0 & \tilde{d}_2 & 0 & \tilde{d}_3 & \cdots & 0 & 0 & \tilde{d}_{-1} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 \\ 0 & 0 & \tilde{c}_{-1} & 0 & \tilde{c}_0 & 0 & \tilde{c}_1 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots & \cdots & \cdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \tilde{d}_{-1} & 0 & \tilde{d}_0 & 0 & \tilde{d}_1 & 0 & \tilde{d}_2 & \tilde{d}_3 & \cdots & 0 \end{pmatrix}_{\frac{N}{2}\times N}$$

3. Biothogonal Spline Wavelet Transform Method of Solution

In the computation of numerical analysis obtain approximate solution containing some error. In this section, we have two cases to solve for the numerical solution of linear and nonlinear integral and integro-differential equations.

Case 1: Multigrid (MG) Method

In this case, we solve linear problems to approximate solution containing some error. There are many approaches to minimize the error. Some of them are multigrid (MG) method, biorthogonal spline wavelet transform method (BSWTM) and modified

biorthogonal spline wavelet transform method (MBSWTM). Multigrid method is explained in the [17]. Now, we discuss about BSWTM and MBSWTM using the Biorthogonal Spline Wavelet operators and as given in section 2, instead of multigrid R and P operators. Now we are discussing about the method of solution as follows:

Biorthogonal Spline Wavelet Transform Method (BSWTM): Consider the Volterra integral equations of the second kind,

$$u(t) = f(t) + \int_0^t k(t,s)u(s))ds0 \leq t, s \leq 1, \tag{1}$$

Consider the Fredholm integral equation of the second kind,

$$u(t) = f(t) + \int_0^1 k(t,s)u(s))ds0 \leq t, s \leq 1, \tag{2}$$

where *f(t)* and the kernels *k(t,s)* are assumed to be in $L^2(R)$ on the interval $0 \leq t, s \leq 1$. After discretizing the integral equation through the trapezoidal discretization method (TDM) [28], we get system of algebraic equations. Through this system we can write the system as

$$Au = b \tag{3}$$

where A is $N \times N$ coefficient matrix, b is $N \times 1$ matrix and is matrix to be determined. Solving the system of equation (3) through the iterative method, we get the approximate solution v of u. i.e., $u = e + v \Rightarrow v = u - e$, where e is $N \times 1$ matrix error to be determined. From equation (3), we get the approximate solution v of u. Now we find the residual as

$$r_{N\times 1} = [b]_{N\times 1} - [A]_{N\times N}\,[v]_{N\times 1} \tag{4}$$

We reduce the matrices in the finer $\left(N^{th} = 2^J\right)$ level to coarsest level using Biorthogonal spline wavelet restriction operator $(BSWT_R)$ and then construct the matrices back to finer level from the coarsest level using Biorthogonal spline wavelet prolongation operator$(BSWT_P)$. From (4),

$$r_{N/2\times 1} = [BSWT_R]_{N/2\times N}\,[r]_{N\times 1} \tag{5}$$

and

$$[A]_{N/2\times N/2} = [BSWT_R]_{N/2\times N}\,[A]_{N\times N}\,[BSWT_P]_{N\times N/2} \tag{6}$$

Residual equation becomes,

$$[A]_{N/2\times N/2}\,[e]_{N/2\times 1} = [r]_{N/2\times 1}$$

where $e_{N/2\times 1}$ is to be determined. Solve $e_{N/2\times 1}$ with initial guess $'0'$. From (5),

$$r_{N/4\times 1} = [BSWT_R]_{N/4\times N/2}\,[r]_{N/2\times 1} \tag{7}$$

and

$$[A]_{N/4\times N/4} = [BSWT_R]_{N/4\times N/2}\,[A]_{N/2\times N/2}\,[BSWT_P]_{N/2\times N/4}$$

Then residual equation becomes,

$$[A]_{N/4\times N/4}\,[e]_{N/4\times 1} = [r]_{N/4\times 1}$$

Solve $e_{N/4\times1}$ with initial guess $'0'$. Continue the procedure up to the coarsest level, we have,

$$r_{1\times1} = [BSWT_R]_{1\times2}\,[r]_{2\times1} \tag{8}$$

and

$$[A]_{1\times1} = [BSWT_R]_{1\times2}\,[A]_{2\times2}\,[BSWT_P]_{2\times1}$$

Residual equation is,

$$[A]_{1\times1}\,[e]_{1\times1} = [r]_{1\times1}$$

Solve $e_{1\times1}$ exactly. Now correct the solution

$$u_{2\times1} = [e]_{2\times1} + [BSWT_P]_{2\times1}\,[e]_{1\times1}$$

Solve $[A]_{2\times2}\,[u]_{2\times1} = [r]_{2\times1}$ with initial guess $u_{2\times1}$. Correct the solution

$$u_{4\times1} = [e]_{4\times1} + [BSWT_P]_{4\times2}\,[u]_{2\times1}$$

Solve $[A]_{4\times4}\,[u]_{4\times1} = [r]_{4\times1}$ with initial guess $u_{4\times1}$. Continue the procedure up to the finer level. Correct the solution

$$u_{N\times1} = [v]_{N\times1} + [BSWT_P]_{N\times N/2}\,[u]_{N/2\times1}$$

Solve $[A]_{N\times N}\,[u]_{N\times1} = [b]_{N\times1}$ with initial guess $u_{N\times1}$. $u_{N\times1}$ is the required solution of system (3).

Modified Biorthogonal Spline Wavelet Transform Method (MBSWTM): As explained in the above, the same procedure is applied for modified biorthogonal spline wavelet transform method (MBSWTM). Here, we use the modified biorthogonal spline wavelet operators and as given in section 2, instead of multigrid R and P operators.

Case 2: Full-Approximation Scheme (FAS)

In this case, we solve Nonlinear problems to approximate solution containing some error. There are many approaches to minimize the error. Some of them are Full-Approximation Scheme (FAS), Biorthogonal Spline Wavelet Full-Approximation transform method (BSWFATM) and modified Biorthogonal Spline Wavelet Full-Approximation transform method (MBSWFATM). Full-Approximation Scheme is explained in the [19]. Here, we use the Biorthogonal Spline Wavelet operators and as given in section 2, instead of multigrid R and P operators.

Biorthogonal Spline Wavelet Full-Approximation Transform Method (BSWFATM): Consider the Nonlinear Fredholm integral equation of the second kind,

$$u(t) = f(t) + \int_0^1 k(t,s,u(s))ds0 \le t,s \le 1, \tag{9}$$

Consider the Nonlinear Volterra integral equation of the second kind,

$$u(t) = f(t) + \int_0^t k(t,s,u(s))ds0 \le t,s \le 1, \tag{10}$$

where *k(t, s, u(s))* is a nonlinear function defined on $[0,1]\times[0,1]$. The known function *k(t, s, u(s))* is called the kernel of the integral equation, while the unknown function *u(t)* represents the solution of the integral equation. After discretizing

the integral equation through the trapezoidal discretization method (TDM) [28], we get the system of nonlinear equations of the form,

$$i.e., A(u) = b \tag{11}$$

where A is $N \times N$ coefficient matrix, I is the identity matrix, b is $N \times 1$ matrix and u is $N \times 1$ matrix to be determined. This has N equations with N unknowns. Solving the system of Equation (11) through the iterative method that is Gauss Seidel (GS), we get approximate solution v of u. i.e., $u = e + v \Rightarrow v = u - e$, where e is $(N \times 1)$ matrix error to be determined. Now, we are deliberating about the Biorthogonal Spline Wavelet Full-Approximation Transform Method (BSWFATM) of solutions given by Briggs et. al [6] is as follows the procedure. From the system Equation (11), we get the approximate solution v for u. Now we find the residual as

$$r_{N\times 1} = b_{N\times 1} - A(v)_{N\times 1} \tag{12}$$

Reduce the matrices in the finer level to coarsest level using Biorthogonal Spline Wavelet Restriction operator and then construct the matrices back to finer level from the coarsest level using Biorthogonal Spline Wavelet Prolongation operator as given in section 2. Next,

$$r_{N/2\times 1} = [BSWT_R]_{N/2\times N}\,[r]_{N\times 1} \tag{13}$$

Similarly,

$$v_{N/2\times 1} = [BSWT_R]_{N/2\times N}\,[v]_{N\times 1} \tag{14}$$

and

$$A\left(v_{N/2\times 1} + e_{N/2\times 1}\right) + A\left(v_{N/2\times 1}\right) = r_{N/2\times 1} \tag{15}$$

Solve Equation (15) with initial guess 0, we get $e_{N/2\times 1}$. Next,

$$r_{N/4\times 1} = [BSWT_R]_{N/4\times N/2}[r]_{N/2\times 1}$$

Similarly,

$$v_{N/4\times 1} = [BSWT_R]_{N/4\times N/2}[v]_{N/2\times 1}$$

and

$$A(v_{N/4\times 1} + e_{N/4\times 1}) + A(v_{N/4\times 1}) = r_{N/4\times 1} \tag{16}$$

Solve Equation (16) with initial guess 0, we get $e_{N/4\times 1}$. Next, the procedure is continue up to the coarsest level, we have,

$$r_{1\times 1} = [BSWT_R]_{1\times 2}[r]_{2\times 1}.$$

Similarly,

$$v_{1\times 1} = [BSWT_R]_{1\times 2}[v]_{2\times 1}.$$

and

$$A(v_{1\times 1} + e_{1\times 1}) + A(v_{1\times 1}) = r_{1\times 1}. \tag{17}$$

Solve Equation (17) we get, $e_{1\times 1}$. Next, Interpolate error up to the finer level, i.e.

$$e_{2\times 1} = [BSWT_P]_{2\times 1}[e]_{1\times 1},$$

$$e_{4\times1} = [BSWT_P]_{4\times2}[e]_{2\times1},$$

and so on we have,

$$e_{N\times1} = [BSWT_P]_{N\times N/2}[e]_{N/2\times1}.$$

Lastly, Correct the solution with error, $u_{N\times1} = [v]_{N\times1} + [e]_{N\times1}$. This is the required solution of the given integral equation. **Modified Biorthogonal Spline Wavelet Full-Approximation Transform Method (MBSWFATM):** As explained in the above, the same procedure is applied for modified biorthogonal spline wavelet full-approximation transform method (MBSWFATM). Here, we use the modified biorthogonal spline wavelet operators $MBSWT_R$ and $MBSWT_P$ as given in section 2, instead of multigrid R and P operators.

4. Method of Implementation

In this section, we implemented MG, BSWTM, MBSWTM, FAS, BSWFATM and MBSWFATM for the numerical solution of linear and nonlinear integral and integro-differential equations and subsequently presented in tables and figures, here error analysis is considered as $E_{max} = max|u_e - u_a|$, where u_e and u_a are exact and approximate solutions respectively.

Test Problem 4.1. *Let us consider the linear Volterra integral equation [29],*

$$u(t) = sin(t) - \int_0^t k(t-s)u(s))ds0 \leq t, s \leq 1, \tag{18}$$

which has the exact solution $u(t) = \frac{1}{2}(sin(t) + sinh(t))$. *After discretizing the Equation (18) through the trapezoidal discretization method (TDM), we get a system of linear algebraic equations of the form, (for* $N = 8$*).*

$$[A]_{8\times8}[u]_{8\times1} = [b]_{8\times1} \tag{19}$$

Solving Equation (19) through the iterative method, we get the approximate solution v of u. i.e., $u = e + v \rightarrow v = u - e$, where e is $(8 \times 1$ matrix) error to be determined. The implementation of the problem is given as the BSWTM is discussed in section 3, as follows, From Equation (19), we find the residual as

$$r_{8\times1} = [b]_{8\times1} - [A]_{8\times8}[v]_{8\times1}$$

We get $r_{8\times1} = [\ 0\ \ 7.75e-09\ \ 2.84e-09\ \ -7.59e-09\ \ -3.90e-09\ \ 1.03e-08\ \ -5.74e-09\ \ 1.06e-09\ ]$. We reduce the matrices in the finer level to coarsest level using Restriction operator $'BSWT_R'$ and then construct the matrices back to finer level from the coarsest level using Prolongation operator $'BSWT_P'$. From Equation (20),

$$r_{4\times1} = [BSWT_R]_{4\times8}[r]_{8\times1} \tag{20}$$

and

$$[A]_{4\times4} = [BSWT_R]_{4\times8}\,[A]_{8\times8}\,[BSWT_P]_{8\times4}$$

Residual equation becomes,

$$[A]_{4\times4}[e]_{4\times1} = [r]_{4\times1},$$

where $e_{4\times1}$ to be determine. Solve with initial guess 0. We get $e_{4\times1} = [\ 1.25e-09\ \ 7.71e-09\ \ 6.25e-09\ \ -7.33e-09\ ]$. From Equation (21),

$$r_{2\times1} = [BSWT_R]_{2\times4}[r]_{4\times1} \tag{21}$$

and

$$[A]_{2\times2} = [BSWT_R]_{2\times4}\,[A]_{4\times4}\,[BSWT_P]_{4\times2}$$

Then residual equation becomes,

$$[A]_{2\times2}[e]_{2\times1} = [r]_{2\times1}.$$

Solve $e_{2\times1}$ with initial guess 0. We get $e_{2\times1} = [\ 5.20e-09\ \ 7.72e-09\ ]$. From Equation (22),

$$r_{1\times1} = [BSWT_R]_{1\times2}[r]_{2\times1} \tag{22}$$

and

$$[A]_{1\times1} = [BSWT_R]_{1\times2}\,[A]_{2\times2}\,[BSWT_P]_{2\times1}$$

and Residual equation is,

$$[A]_{1\times1}[e]_{1\times1} = [r]_{1\times1}.$$

Solve $e_{1\times1}$ exactly. We get $e_{1\times1} = 1.56e-08$. From $e_{1\times1}$, now correct the solution

$$u_{2\times1} = [e]_{2\times1} + [BSWT_P]_{2\times1}[e]_{1\times1}$$

Solve $[A]_{2\times2}[u]_{2\times1} = [r]_{2\times1}$ with initial guess $u_{2\times1}$. We get $u_{2\times1} = [\ 5.20e-09\ \ 7.72e-09\ ]$. Correct the solution from $u_{2\times1}$,

$$u_{4\times1} = [e]_{4\times1} + [BSWT_P]_{4\times2}[u]_{2\times1}$$

Solve $[A]_{4\times4}[u]_{4\times1} = [r]_{4\times1}$ with initial guess $u_{4\times1}$. We get $u_{4\times1} = [\ 1.25e-09\ \ 7.71e-09\ \ 6.25e-09\ \ -7.33e-09\ ]$. From $u_{4\times1}$ correct the solution,

$$u_{8\times1} = [v]_{8\times1} + [BSWT_P]_{8\times4}[u]_{4\times1}$$

Solve $[A]_{8\times8}[u]_{8\times1} = [f]_{8\times1}$ with initial guess $u_{8\times1}$, where $u_{8\times1}$ is the required solution of Equation (18). The numerical solutions of Equation (18) is obtained through the method as explained in section 3 compared with the exact and existing method are shown in table 1 and in the figure 1 for $N = 64$. Maximum error and CPU time are shown in table 2.

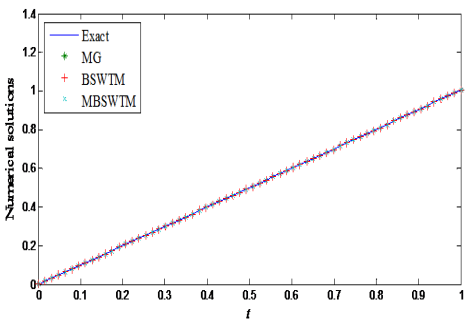

Figure 1. **Comparison of numerical solutions with exact solution of test problem 4.1, for N=64.**

t	MG	BSWTM	MBSWTM	Exact
0	0.0000	0.0000	0.0000	0
0.1428	0.1423	0.1423	0.1423	0.1428
0.2857	0.2847	0.2847	0.2847	0.2857
0.4285	0.4271	0.4271	0.4271	0.4286
0.5714	0.5698	0.5698	0.5698	0.5719
0.7142	0.7132	0.7132	0.7132	0.7158
0.8571	0.8577	0.8577	0.8577	0.8609
1	1.0043	1.0043	1.0043	1.0083

Table 1. **Numerical results of the test problem 4.1, for $N = 8$**

N	Method	E_{max}	Setup time	Running time	Total time
	MG	8.72e-04	0.2599	0.0296	0.2895
16	BSWTM	8.72e-04	0.0470	0.0107	0.0578
	MBSWTM	8.72e-04	0.0462	0.0030	0.0492
	MG	2.04e-04	0.1722	0.0321	0.2043
32	BSWTM	2.04e-04	0.0975	0.0098	0.1073
	MBSWTM	2.04e-04	0.0648	0.0029	0.0677
	MG	4.95e-05	0.1848	0.0334	0.2182
64	BSWTM	4.95e-05	0.1385	0.0109	0.1494
	MBSWTM	4.95e-05	0.0894	0.0032	0.0926
	MG	1.21e-05	0.3431	0.0117	0.3548
128	BSWTM	1.21e-05	0.2226	0.0310	0.2536
	MBSWTM	1.21e-05	0.1798	0.0041	0.1839

Table 2. **Maximum error and CPU time (in seconds) of the test problem 4.1**

Test Problem 4.2. *Next, consider the linear Fredholm integral equation [30],*

$$u(t) = 0.9t^2 + \int_0^1 0.5t^2 s^2 u(s) ds 0 \leq t, s \leq 1, \tag{23}$$

which has the exact solution $u(t) = t^2$. The numerical solutions of Equation (24) is obtained through the method as explained in section 3 compared with the exact and existing method are shown in table 3 and in the figure 2 for $N = 64$. Maximum error and CPU time are shown in table 4.

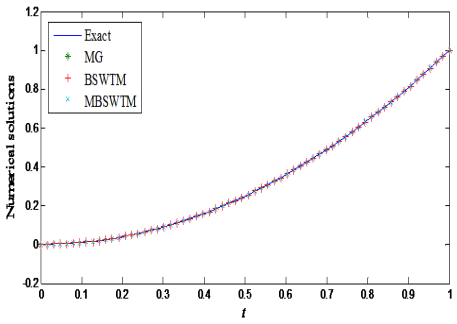

Figure 2. **Comparison of numerical solutions with exact solution of test problem 4.2, for N=64.**

t	MG	BSWTM	MBSWTM	Exact
0	0.0000	0.0000	0.0000	0
0.0666	0.0044	0.0044	0.0044	0.0044
0.1333	0.0177	0.0177	0.0177	0.0177
0.2000	0.0400	0.0400	0.0400	0.0400
0.2666	0.0711	0.0711	0.0711	0.0711
0.3333	0.1112	0.1112	0.1112	0.1111
0.4000	0.1601	0.1601	0.1601	0.1600
0.4666	0.2179	0.2179	0.2179	0.2177
0.5333	0.2846	0.2846	0.2846	0.2844
0.6000	0.3602	0.3602	0.3602	0.3600
0.6666	0.4448	0.4448	0.4448	0.4444
0.7333	0.5382	0.5382	0.5382	0.5377
0.8000	0.6405	0.6405	0.6405	0.6400
0.8666	0.7517	0.7517	0.7517	0.7511
0.9333	0.8718	0.8718	0.8718	0.8711
1	1.0008	1.0008	1.0008	1

Table 3. **Numerical results of the test problem 4.2, for** $N = 16$

N	Method	E_{max}	Setup time	Running time	Total time
	MG	8.23e-04	0.2650	0.0290	0.2939
16	BSWTM	8.23e-04	0.0521	0.0089	0.0611
	MBSWTM	8.23e-04	0.0476	0.0047	0.0523
	MG	1.92e-04	0.1890	0.0303	0.2193
32	BSWTM	1.92e-04	0.0680	0.0095	0.0774
	MBSWTM	1.92e-04	0.0571	0.0031	0.0602
	MG	4.66e-05	0.2028	0.0309	0.2337
64	BSWTM	4.66e-05	0.1504	0.0104	0.1608
	MBSWTM	4.66e-05	0.0953	0.0037	0.0991
	MG	1.14e-05	0.3496	0.0128	0.3624
128	BSWTM	1.14e-05	0.2404	0.0324	0.2728
	MBSWTM	1.14e-05	0.2150	0.0043	0.2192

Table 4. **Maximum error and CPU time (in seconds) of the test problem 4.2**

Test Problem 4.3. *Next, consider the linear Volterra-Fredholm integral equation [31],*

$$u(t) = t - 2exp(t) + exp(-t) + 1 + \int_0^t sexp(t)u(s))ds + \int_0^1 exp(t+s)u(s))ds 0 \leq t, s \leq 1, \tag{24}$$

which has the exact solution $u(t) = exp(-t)$*. After discretizing the Equation (25) through the trapezoidal discretization method (TDM), we get a system of linear algebraic equations of the form, (for* $N = 8$*).*

$$[A]_{8\times 8}[u]_{8\times 1} = [b]_{8\times 1} \tag{25}$$

Solving Equation (26) through the iterative method, we get the approximate solution v of u. i.e., $u = e + v \rightarrow v = u - e$, where e is (8×1 matrix) error to be determined. The implementation of the problem is given as the BSWTM method is discussed in section 3, as follows, From Equation (26), we find the residual as

$$r_{8\times 1} = [b]_{8\times 1} - [A]_{8\times 8}[v]_{8\times 1} \tag{26}$$

We get $r_{8\times 1} = [\ -2.67e-07\ \ 1.82e-07\ \ 8.04e-07\ \ -1.36e-06\ \ 9.50e-07\ \ -3.31e-07\ \ 4.64e-08\ \ -1.15e-08\ ]$. We reduce the matrices in the finer level to coarsest level using Restriction operator $'BSWT'_R$ and then construct the matrices

back to finer level from the coarsest level using Prolongation operator $'BSWT'_P$. From Equation (27),

$$r_{4\times1} = [BSWT_R]_{4\times8}[r]_{8\times1} \tag{27}$$

and

$$[A]_{4\times4} = [BSWT_R]_{4\times8}\,[A]_{8\times8}\,[BSWT_P]_{8\times4}$$

Residual equation becomes,

$$[A]_{4\times4}[e]_{4\times1} = [r]_{4\times1}$$

where $e_{4\times1}$ to be determine. Solve with initial guess 0. We get $e_{4\times1} = [\ 8.70e-08\ \ 9.01e-09\ \ 1.21e-06\ \ -1.54e-06\ ]$. From Equation (28),

$$r_{2\times1} = [BSWT_R]_{2\times4}[r]_{4\times1} \tag{28}$$

and

$$[A]_{2\times2} = [BSWT_R]_{2\times4}\,[A]_{4\times4}\,[BSWT_P]_{4\times2}$$

Then residual equation becomes,

$$[A]_{2\times2}[e]_{2\times1} = [r]_{2\times1}.$$

Solve $e_{2\times1}$ with initial guess 0. We get $e_{2\times1} = [\ 4.18e-07\ \ 3.71e-07\ ]$. From Equation (29),

$$r_{1\times1} = [BSWT_R]_{1\times2}[r]_{2\times1} \tag{29}$$

and

$$[A]_{1\times1} = [BSWT_R]_{1\times2}\,[A]_{2\times2}\,[BSWT_P]_{2\times1}$$

and Residual equation is,

$$[A]_{1\times1}[e]_{1\times1} = [r]_{1\times1}.$$

Solve $e_{1\times1}$ exactly. We get $e_{1\times1} = 1.36e-06$. From $e_{1\times1}$, now correct the solution

$$u_{2\times1} = [e]_{2\times1} + [BSWT_P]_{2\times1}[e]_{1\times1}$$

Solve $[A]_{2\times2}[u]_{2\times1} = [r]_{2\times1}$ with initial guess $u_{2\times1}$. We get $u_{2\times1} = [\ 4.18e-07\ \ 3.71e-07\ ]$. Correct the solution from $u_{2\times1}$,

$$u_{4\times1} = [e]_{4\times1} + [BSWT_P]_{4\times2}[u]_{2\times1}$$

Solve $[A]_{4\times4}[u]_{4\times1} = [r]_{4\times1}$ with initial guess $u_{4\times1}$. We get $u_{4\times1} = [\ 8.70e-08\ \ 9.01e-09\ \ 1.21e-06\ \ -1.54e-06\ ]$. From $u_{4\times1}$ correct the solution,

$$u_{8\times1} = [v]_{8\times1} + [BSWT_P]_{8\times4}[u]_{4\times1}$$

Solve $[A]_{8\times8}[u]_{8\times1} = [f]_{8\times1}$ with initial guess $u_{8\times1}$ where $u_{8\times1}$ is the required solution of Equation (25). The numerical solutions of Equation (25) is obtained through the method as explained in section 3 compared with the exact and existing method are shown in table 5 and in the figure 3 for $N = 64$. Maximum error and CPU time are shown in table 6.

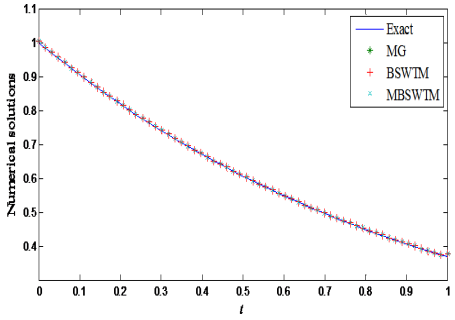

Figure 3. **Comparison of numerical solutions with exact solution of test problem 4.3, for N=64.**

t	MG	BSWTM	MBSWTM	Exact
0	1.0274	1.0274	1.0274	1
0.1428	0.8984	0.8984	0.8984	0.8668
0.2857	0.7644	0.7644	0.7644	0.7514
0.4285	0.6587	0.6587	0.6587	0.6514
0.5714	0.5673	0.5673	0.5673	0.5647
0.7142	0.4881	0.4881	0.4881	0.4895
0.8571	0.4195	0.4195	0.4195	0.4243
1	0.4586	0.4586	0.4586	0.3678

Table 5. **Numerical results of the test problem 4.3, for $N = 8$**

N	Method	E_{max}	Setup time	Running time	Total time
	MG	4.46e-02	0.2630	0.0294	0.2924
16	BSWTM	4.46e-02	0.0516	0.0109	0.0625
	MBSWTM	4.46e-02	0.0493	0.0051	0.0544
	MG	2.16e-02	0.1717	0.0305	0.2022
32	BSWTM	2.16e-02	0.0717	0.0083	0.0800
	MBSWTM	2.16e-02	0.0618	0.0059	0.0677
	MG	1.05e-02	0.2494	0.0313	0.2807
64	BSWTM	1.05e-02	0.1890	0.0225	0.2115
	MBSWTM	1.05e-02	0.1154	0.0045	0.1198
	MG	5.23e-03	0.3877	0.0096	0.3973
128	BSWTM	5.23e-03	0.2529	0.0345	0.2874
	MBSWTM	5.23e-03	0.1911	0.0054	0.1965

Table 6. **Maximum error and CPU time (in seconds) of the test problem 4.3**

Test Problem 4.4. *Next, consider the Nonlinear Fredholm integral equations [32],*

$$u(t) = -sin(4t) - t^3\left(\frac{-367}{4096}cos(4)sin(4) + \frac{11357}{98304} - \frac{2095}{32768}cos^2(4)\right) + \int_0^1 t^3 s^5 u^2(s))ds, 0 \le t \le 1, \tag{30}$$

which has the exact solution $u(t) = sin(-4t)$. After discretizing the Equation (31) through the trapezoidal discretization method (TDM), we get a system of nonlinear algebraic equations of the form, (for $N = 8$).

$$[A]_{8\times 8}[u]_{8\times 1} = [b]_{8\times 1} \tag{31}$$

Solving Equation (32) through the iterative method, we get the approximate solution v of u. i.e., $u = e + v \rightarrow v = u - e$, where e is (8×1 matrix) error to be determined. The implementation of MBSWFATM is discussed in section 3 as follows, From Equation (32), we find the residual as

$$r_{8\times 1} = [b]_{8\times 1} - [A]_{8\times 8}[v]_{8\times 1} \tag{32}$$

We get $r_{8\times1} = [\ 0\ \ 0\ \ -3.98e-09\ \ -3.69e-07\ \ -5.75e-06\ \ -1.50e-05\ \ 8.31e-05\ \ 6.44e-04\ ]$. We reduce the matrices in the finer level to coarsest level using Restriction operator $MBSWT_R$ and then construct the matrices back to finer level from the coarsest level using Prolongation operator $MBSWT_P^T$. From Equation (33),

$$r_{4\times1} = [MBSWT_R]_{4\times8}[r]_{8\times1} \tag{33}$$

Similarly, $v_{4\times1} = [MBSWT_R]_{4\times8}][v]_{8\times1}$ and

$$A(v_{4\times1} + e_{4\times1}) + A(v_{4\times1} = r_{4\times1}.) \tag{34}$$

Solve Equation (35) with initial guess 0, we get $e_{4\times1}$. From Equation (34),

$$r_{2\times1} = [MBSWT_R]_{2\times4}[r]_{4\times1} \tag{35}$$

Similarly, $v_{2\times1} = [MBSWT_R]_{2\times4}][v]_{4\times1}$ and

$$A(v_{2\times1} + e_{2\times1}) + A(v_{2\times1} = r_{2\times1}.) \tag{36}$$

Solve Equation (37) with initial guess 0, we get $e_{2\times1}$. From Equation (36),

$$r_{1\times1} = [MBSWT_R]_{1\times2}[r]_{2\times1} \tag{37}$$

Similarly, $v_{1\times1} = [MBSWT_R]_{1\times2}][v]_{2\times1}$ and

$$A(v_{1\times1} + e_{1\times1}) + A(v_{1\times1} = r_{1\times1}.) \tag{38}$$

Solve Equation (39) we get, $e_{1\times1}$. From $e_{1\times1}$, Interpolate error up to the finer level, i.e.

$$e_{2\times1} = [MBSWT_P^T]_{2\times1}[e]_{1\times1}$$
$$e_{4\times1} = [MBSWT_P^T]_{4\times2}[e]_{2\times1}$$

and lastly we have,

$$e_{8\times1} = [MBSWT_P^T]_{8\times4}[e]_{4\times1}. \tag{39}$$

We get $e_{8\times1} = [\ 1.43e-06\ \ 7.32e-11\ \ 3.56e-05\ \ -3.67e-07\ \ 1.03e-05\ \ 0\ \ -7.15e-07\ \ 0\ ]$. From Equation (40) correct the solution with error $u_{8\times1} = v_{8\times1} + e_{8\times1}$. Lastly, we get $u_{8\times1}$ is the required solution of Equation (31). The numerical solutions of the given equation is obtained through the present method as explained in section 3 and are presented in comparison with the exact solution are shown in the table 7 and the figure 4, for $N = 64$. Maximum error analysis and CPU time are shown in table 8.

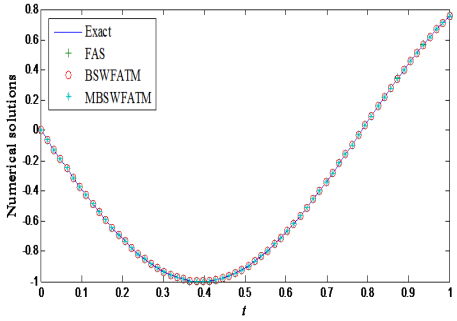

Figure 4. **Comparison of numerical solutions with exact solution of test problem 4.4, for N=64.**

N	Method	E_{max}	Setup time	Running time	Total time
	FAS	2.45e-03	0.0161	0.0915	0.1075
16	BSWFATM	1.98e-03	0.0245	0.0753	0.0999
	MBSWFATM	1.98e-03	0.0102	0.0701	0.0803
	FAS	5.54e-04	0.0285	0.4081	0.4366
32	BSWFATM	3.17e-04	0.0148	0.3372	0.3520
	MBSWFATM	3.17e-04	0.0080	0.3222	0.3302
	FAS	1.28e-04	0.2280	3.0487	3.2767
64	BSWFATM	8.45e-05	0.0148	2.7879	2.8027
	MBSWFATM	8.45e-05	0.0075	1.8799	1.8873

Table 7. **Maximum error and CPU time (in seconds) of the test problem 4.4**

t	FAS	BSWFATM	MBSWFATM	EXACT
0	0.0000	0.0000	0.0000	0
0.0666	-0.2635	-0.2631	-0.2635	-0.2635
0.1333	-0.5084	-0.5083	-0.5081	-0.5084
0.2000	-0.7173	-0.7173	-0.7173	-0.7173
0.2666	-0.8755	-0.8755	-0.8754	-0.8756
0.3333	-0.9718	-0.9718	-0.9718	-0.9719
0.4000	-0.9994	-0.9994	-0.9994	-0.9995
0.4666	-0.9562	-0.9562	-0.9562	-0.9565
0.5333	-0.8454	-0.8454	-0.8454	-0.8459
0.6000	-0.6748	-0.6748	-0.6748	-0.6754
0.6666	-0.4564	-0.4564	-0.4564	-0.4572
0.7333	-0.2056	-0.2056	-0.2056	-0.2067
0.8000	0.0598	0.0597	0.0597	0.0583
0.8666	0.3212	0.3210	0.3210	0.3193
0.9333	0.5598	0.5595	0.5595	0.5578
1	0.7592	0.7587	0.7587	0.7568

Table 8. **Numerical results of the test problem 4.4, for $N = 16$**

Test Problem 4.5. *Next, consider the Nonlinear Fredholm-Hammerstein integral equations [33],*

$$u(t) = tln(t+1) - \frac{55}{108}t + \frac{1}{3}ln2\left(\frac{8}{3}t + 2 - tln2\right) - \frac{241}{576} + \frac{1}{2}\int_0^1 (t-s)u^2(s))ds, 0 \leq t \leq 1, \tag{40}$$

which has the exact solution $u(t) = tln(t+1)$. The numerical solutions of Equation (41) is obtained through the present method as explained in section 3 and are presented in comparison with the exact solution are shown in the table 9 and the figure 5, for $N = 64$. Maximum error analysis and CPU time are shown in table 10.

t	FAS	BSWFATM	MBSWFATM	EXACT
0	-0.0003	-0.0003	-0.0003	0
0.0666	0.0039	0.0039	0.0039	0.0043
0.1333	0.0163	0.0163	0.0163	0.0166
0.2000	0.0361	0.0361	0.0361	0.0364
0.2666	0.0627	0.0627	0.0627	0.0630
0.3333	0.0956	0.0956	0.0956	0.0958
0.4000	0.1343	0.1343	0.1343	0.1345
0.4666	0.1784	0.1784	0.1784	0.1787
0.5333	0.2277	0.2277	0.2277	0.2279
0.6000	0.2817	0.2817	0.2817	0.2820
0.6666	0.3403	0.3403	0.3403	0.3405
0.7333	0.4031	0.4031	0.4031	0.4033
0.8000	0.4700	0.4700	0.4700	0.4702
0.8666	0.5407	0.5407	0.5407	0.5409
0.9333	0.6151	0.6151	0.6151	0.6152
1	0.6930	0.6930	0.6930	0.6931

Table 9. **Numerical results of the test problem 4.5, for** $N = 16$

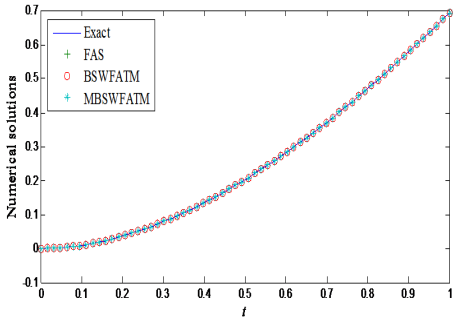

Figure 5. **Comparison of numerical solutions with exact solution of test problem 4.5, for** N**=64.**

N	Method	E_{max}	Setup time	Running time	Total time
	FAS	3.66e-04	0.0274	0.0462	0.0736
16	BSWFATM	3.66e-04	0.0178	0.0314	0.0492
	MBSWFATM	3.66e-04	0.0100	0.0289	0.0389
	FAS	8.57e-05	0.0252	0.0483	0.0735
32	BSWFATM	8.57e-05	0.0148	0.0447	0.0594
	MBSWFATM	8.57e-05	0.0100	0.0399	0.0500
	FAS	2.07e-05	0.0806	0.1031	0.1836
64	BSWFATM	2.07e-05	0.0148	0.0993	0.1141
	MBSWFATM	2.07e-05	0.0103	0.0954	0.1057
	FAS	5.10e-06	0.1789	0.2082	0.3871
128	BSWFATM	5.10e-06	0.0093	0.2195	0.2288
	MBSWFATM	5.10e-06	0.0063	0.1904	0.1966

Table 10. **Maximum error and CPU time (in seconds) of the test problem 4.5**

Test Problem 4.6. *Next, consider the Nonlinear Fredholm-Hammerstein integro-differential equation [34],*

$$u'(t) = 1 - \frac{1}{3}t + \int_0^1 tu^2(s))ds, u(0) = 0, 0 \leq t \leq 1, \tag{41}$$

which has the exact solution $u(t) = t$*. Integrating the Equation (42) w.r.t* t *and using the initial condition, we get*

$$u(t) = t - \frac{t^2}{6} + \frac{t^2}{2}\int_0^1 u^2(s)ds, \tag{42}$$

Solving this equation, we obtain the numerical solution through the present method as explained in section 3 and are presented in comparison with the exact solution are shown in the table 11 and the figure 6, for $N = 64$. Maximum error analysis and CPU time are shown in table 12.

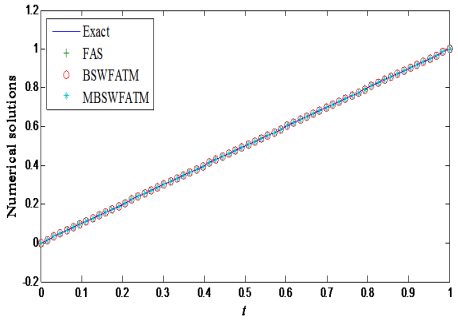

Figure 6. **Comparison of numerical solutions with exact solution of test problem 4.6, for N=64.**

t	FAS	BSWFATM	MBSWFATM	EXACT
0	0.0000	0.0000	0.0000	0
0.0666	0.0666	0.0666	0.0666	0.0666
0.1333	0.1333	0.1333	0.1333	0.1333
0.2000	0.2000	0.2000	0.2000	0.2000
0.2666	0.2666	0.2666	0.2666	0.2666
0.3333	0.3333	0.3333	0.3333	0.3333
0.4000	0.4000	0.4000	0.4000	0.4000
0.4666	0.4666	0.4666	0.4666	0.4666
0.5333	0.5333	0.5333	0.5333	0.5333
0.6000	0.6000	0.6000	0.6000	0.6000
0.6666	0.6666	0.6666	0.6666	0.6666
0.7333	0.7333	0.7333	0.7333	0.7333
0.8000	0.7998	0.7998	0.7998	0.8000
0.8666	0.8665	0.8665	0.8665	0.8666
0.9333	0.9332	0.9332	0.9332	0.9333
1	0.9998	0.9998	0.9998	1

Table 11. **Numerical results of the test problem 4.6, for $N = 16$**

N	Method	E_{max}	Setup time	Running time	Total time
	FAS	1.70e-03	0.0157	0.1053	0.1210
16	BSWFATM	1.70e-03	0.0205	0.0797	0.1002
	MBSWFATM	1.70e-03	0.0104	0.0668	0.0772
	FAS	9.71e-04	0.0286	0.4099	0.4385
32	BSWFATM	9.71e-04	0.0148	0.4074	0.4222
	MBSWFATM	9.71e-04	0.0100	0.4010	0.4110
	FAS	5.13e-04	0.0633	2.8380	2.9012
64	BSWFATM	5.13e-04	0.0150	2.4331	2.4481
	MBSWFATM	5.13e-04	0.006	1.5218	1.5281

Table 12. **Maximum error and CPU time (in seconds) of the test problem 4.6.**

Test Problem 4.7. *Next, consider the Nonlinear Volterra integral equations [35],*

$$u(t) = f(t) + \int_0^t ts^2u^2(s)ds, 0 \leq t \leq 1, \tag{43}$$

where $f(t) = \left(1 - \frac{11}{9}t + \frac{2}{3}t^2 - \frac{1}{3}t^3 + \frac{2}{9}t^4\right) ln(t+1) - \frac{1}{3}(t+t^3)(ln(t+1))^2 - \frac{11}{9}t^2 + \frac{5}{18}t^3 - \frac{2}{27}t^4$. *which has the exact solution* $u(t) = ln(t+1)$. *After discretizing the Equation (44) through the trapezoidal discretization method (TDM), we get system of nonlinear algebraic equations of the form (for* $N = 8$*),*

$$[A]_{8\times 8}[u]_{8\times 1} = [b]_{8\times 1} \tag{44}$$

Solving Equation (45) through the iterative method, we get the approximate solution v of u. i.e., $u = e + v \rightarrow v = u - e$, where e is $(8 \times 1$ matrix) error to be determined. The implementation of the problem is given as the MBSWFATM is discussed in section 3, as follows, From Equation (45), we find the residual as

$$r_{8\times 1} = [b]_{8\times 1} - [A]_{8\times 8}[v]_{8\times 1} \tag{45}$$

We get $r_{8\times 1} = [\,[0 \;\; 0 \;\; 3.52e-07 \;\; 1.14e-05 \;\; 1.29e-04 \;\; 8.29e-04 \;\; 3.66e-03 \;\; 4.29e-03\,]$. We reduce the matrices in the finer level to coarsest level using Restriction operator $MBSWT_R$ and then construct the matrices back to finer level from the coarsest level using Prolongation operator $MBSWT_P^T$. From Equation (46),

$$r_{4\times 1} = [MBSWT_R]_{4\times 8}[r]_{8\times 1} \tag{46}$$

Similarly,

$$v_{4\times 1} = [MBSWT_R]_{4\times 8}[v]_{8\times 1}$$

and

$$A(v_{4\times 1} + e_{4\times 1}) + A(v_{4\times 1}) = r_{4\times 1}. \tag{47}$$

Solve Equation (48) with initial guess '0', we get $e_{4\times 1}$. From Equation (47),

$$r_{2\times 1} = [MBSWT_R]_{2\times 4}[r]_{4\times 1} \tag{48}$$

similarly,

$$v_{2\times 1} = [MBSWT_R]_{2\times 4}[v]_{4\times 1}$$

and

$$A(v_{2\times 1} + e_{2\times 1}) + A(v_{2\times 1}) = r_{2\times 1}. \tag{49}$$

Solve Equation (50) with initial guess '0', we get $e_{2\times 1}$. From Equation (49),

$$r_{1\times 1} = [MBSWT_R]_{1\times 2} = [r]_{2\times 1} \tag{50}$$

similarly,

$$v_{1\times 1} = [MBSWT_R]_{1\times 2}[v]_{2\times 1}$$

and

$$A(v_{1\times 1} + e_{1\times 1}) + A(v_{1\times 1}) = r_{1\times 1}. \tag{51}$$

Solve Equation (52) we get, $e_{1\times 1}$. From $e_{1\times 1}$. Interpolate error up to the finer level, i.e.

$$e_{2\times 1} = [MBSWT_P^T]_{2\times 1}[e]_{1\times 1}, \qquad e_{4\times 1} = [MBSWT_P^T]_{4\times 2}[e]_{2\times 1},$$

and lastly we have,

$$e_{8\times1} = [MBSWT_P^T]_{8\times4}[e]_{4\times1}. \tag{52}$$

We get $e_{8\times1} = [\ 1.50e-05\ \ 0\ \ 1.75e-03\ \ 7.95e-06\ \ 5.03e-04\ \ 0\ \ -7.53e-06\ \ 0\ ]$. From Equation (53) Correct the solution with error $u_{8\times1} = v_{8\times1} + e_{8\times1}$. Lastly, we get $u_{8\times1}$ is the required approximate solution of Equation (44) are presented in comparison with the exact solution are shown in the table 13 and the figure 7 for $N = 64$. Maximum error analysis and CPU time is shown in table 14.

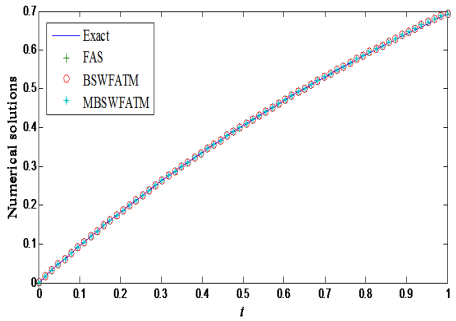

Figure 7. **Comparison of numerical solutions with exact solution of test problem 4.7, for N=64.**

t	FAS	BSWFATM	MBSWFATM	EXACT
0	0.0000	0.0000	0.0000	0
0.0666	0.0645	0.0650	0.0645	0.0645
0.1333	0.1251	0.1253	0.1259	0.1251
0.2000	0.1823	0.1822	0.1823	0.1823
0.2666	0.2364	0.2364	0.2366	0.2363
0.3333	0.2877	0.2877	0.2877	0.2876
0.4000	0.3367	0.3367	0.3366	0.3364
0.4666	0.3835	0.3835	0.3835	0.3829
0.5333	0.4284	0.4284	0.4284	0.4274
0.6000	0.4717	0.4716	0.4716	0.4700
0.6666	0.5137	0.5135	0.5135	0.5108
0.7333	0.5544	0.5542	0.5542	0.5500
0.8000	0.5946	0.5940	0.5940	0.5877
0.8666	0.6341	0.6331	0.6331	0.6241
0.9333	0.6729	0.6718	0.6718	0.6592
1	0.6958	0.6947	0.6947	0.6931

Table 13. **Numerical results of the test problem 4.7, for $N = 16$**

N	Method	E_{max}	Setup time	Running time	Total time
	FAS	1.36e-02	0.0158	0.0750	0.0908
16	BSWFATM	1.26e-02	0.0147	0.0733	0.0880
	MBSWFATM	1.26e-02	0.0101	0.0676	0.0778
	FAS	7.86e-03	0.0284	0.4068	0.4352
32	BSWFATM	7.50e-03	0.0148	0.4064	0.4211
	MBSWFATM	7.50e-03	0.0100	0.3366	0.3467
	FAS	4.19e-03	0.0833	3.2563	3.3396
64	BSWFATM	4.09e-03	0.0149	3.0753	3.0903
	MBSWFATM	4.09e-03	0.0102	2.2629	2.2732

Table 14. **Maximum error and CPU time (in seconds) of the test problem 4.7.**

Test Problem 4.8. *Next, consider the Nonlinear Volterra-Hammerstein integral equation [36],*

$$u(t) = \frac{-15}{56}t^8 + \frac{13}{14}t^7 - \frac{11}{10}t^6 + \frac{9}{20}t^5 + t^2 - t + \int_0^t (t+s)u^3(s)ds, 0 \le t \le 1, \tag{53}$$

which has the exact solution $u(t) = t^2 - t$. The numerical solutions of Equation (54) is obtained through the present method as explained in section 3 and are presented in comparison with the exact solution are shown in the table 16 and the figure 5, for $N = 64$. Maximum error analysis and CPU time are shown in table 15.

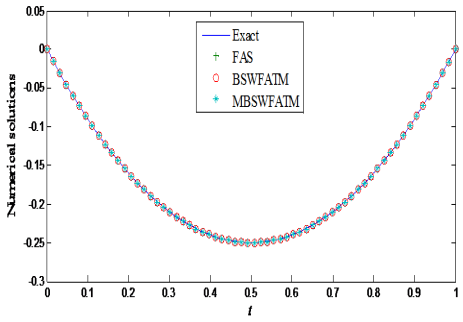

Figure 8. **Comparison of numerical solutions with exact solution of test problem 4.8, for N=64.**

N	Method	E_{max}	Setup time	Running time	Total time
	FAS	5.72e-04	0.0144	0.0949	0.1093
16	BSWFATM	5.59e-04	0.0135	0.0915	0.1050
	MBSWFATM	5.57e-04	0.0100	0.0737	0.0837
	FAS	2.81e-04	0.0208	0.4138	0.4346
32	BSWFATM	2.77e-04	0.0083	0.3506	0.3588
	MBSWFATM	2.77e-04	0.0060	0.3348	0.3408
	FAS	1.38e-04	0.0688	0.1026	0.1714
64	BSWFATM	1.37e-04	0.0098	0.0948	0.1045
	MBSWFATM	1.37e-04	0.0072	0.0962	0.1034
	FAS	6.87e-05	0.3143	0.4560	0.7703
128	BSWFATM	6.85e-05	0.0100	0.4471	0.4571
	MBSWFATM	6.85e-05	0.0137	0.4430	0.4567

Table 15. **Maximum error and CPU time (in seconds) of the test problem 4.8**

t	FAS	BSWFATM	MBSWFATM	EXACT
0	0.0000	0.0000	0.0000	0
0.0666	-0.0622	-0.0622	-0.0622	-0.0622
0.1333	-0.1155	-0.1155	-0.1155	-0.1155
0.2000	-0.1600	-0.1600	-0.1600	-0.1600
0.2666	-0.1957	-0.1957	-0.1957	-0.1955
0.3333	-0.2224	-0.2224	-0.2224	-0.2222
0.4000	-0.2403	-0.2404	-0.2403	-0.2400
0.4666	-0.2493	-0.2493	-0.2493	-0.2488
0.5333	-0.2494	-0.2494	-0.2494	-0.2488
0.6000	-0.2405	-0.2405	-0.2405	-0.2400
0.6666	-0.2227	-0.2227	-0.2227	-0.2222
0.7333	-0.1959	-0.1959	-0.1959	-0.1955
0.8000	-0.1602	-0.1602	-0.1602	-0.1600
0.8666	-0.1156	-0.1156	-0.1156	-0.1155

t	FAS	BSWFATM	MBSWFATM	EXACT
0.9333	-0.0622	-0.0622	-0.0622	-0.0622
1	-0.0000	-0.0000	-0.0000	0

Table 16. **Numerical results of the test problem 4.8, for** $N = 16$

Test Problem 4.9. *Next, consider the Nonlinear Volterra integro-differential equation [37],*

$$u'(t) = -1 + \int_0^t u^2(s)ds, 0 \leq t \leq 1, \tag{54}$$

which has the exact solution $u(t) = -t + \frac{t^4}{12} - \frac{t^7}{252} + \frac{t^10}{6048} - \frac{t^13}{157248}$.

We convert the Volterra integro-differential equation to equivalent Volterra integral equation by using the well-known formula, which converts multiple integrals into a single integral i.e.,

$$\int_0^t \int_0^t ... \int_0^t u(t)dt^n = \frac{1}{n-1}! \int_0^t (t-s)^{n-1} u(s)ds \tag{55}$$

Integrating Equation (55) on both sides from 0 to t and using the initial condition and also converting the double integral to the single integral, we obtain,

$$u(t) = f(t) + \int_0^t k(t,s)u^2(s)ds, \tag{56}$$

where $k(t,s) = (t-s)$ and $f(t) = -t$. The numerical solutions of Equation (57) is obtained through the present method as explained in section 3 and presented in table 17 for $N = 16$ and in figure 9 for $N = 64$. Maximum error analysis and CPU time is shown in table 18.

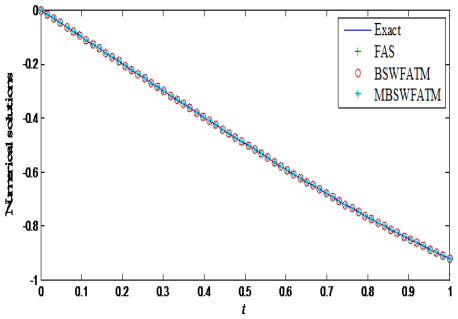

Figure 9. **Comparison of numerical solutions with exact solution of test problem 4.9, for** $N = 64$.

t	FAS	BSWFATM	MBSWFATM	EXACT
0	0.0000	0.0000	0.0000	0
0.0666	-0.0666	-0.0666	-0.0666	-0.0666
0.1333	-0.1333	-0.1333	-0.1333	-0.1333
0.2000	-0.1998	-0.1998	-0.1998	-0.1998
0.2666	-0.2662	-0.2662	-0.2662	-0.2662
0.3333	-0.3323	-0.3323	-0.3323	-0.3323
0.4000	-0.3979	-0.3979	-0.3979	-0.3978
0.4666	-0.4628	-0.4628	-0.4628	-0.4627
0.5333	-0.5267	-0.5267	-0.5267	-0.5266
0.6000	-0.5894	-0.5894	-0.5894	-0.5893
0.6666	-0.6505	-0.6505	-0.6505	-0.6504
0.7333	-0.7098	-0.7098	-0.7098	-0.7096

t	FAS	BSWFATM	MBSWFATM	EXACT
0.8000	-0.7668	-0.7668	-0.7668	-0.7666
0.8666	-0.8213	-0.8213	-0.8213	-0.8210
0.9333	-0.8727	-0.8727	-0.8727	-0.8724
1	-0.9207	-0.9207	-0.9207	-0.9204

Table 17. **Numerical results of the test problem 4.9, for** $N = 16$

N	Method	E_{max}	Setup time	Running time	Total time
	FAS	2.81e-04	0.0159	0.0349	0.0508
16	BSWFATM	2.81e-04	0.0148	0.0317	0.0465
	MBSWFATM	2.81e-04	0.0130	0.0275	0.0404
	FAS	6.56e-05	0.0233	0.0385	0.0618
32	BSWFATM	6.56e-05	0.0117	0.0356	0.0474
	MBSWFATM	6.56e-05	0.0080	0.0318	0.0398
	FAS	1.57e-05	0.0595	0.0763	0.1358
64	BSWFATM	1.57e-05	0.0107	0.0719	0.0825
	MBSWFATM	1.57e-05	0.0072	0.0689	0.0761
	FAS	3.69e-06	0.2930	0.3258	0.6188
128	BSWFATM	3.69e-06	0.0149	0.3280	0.3430
	MBSWFATM	3.69e-06	0.0102	0.3207	0.3309

Table 18. **Maximum error and CPU time (in seconds) of the test problem 4.9**

Test Problem 4.10. *Next, consider the Nonlinear Volterra-Fredholm Hammerstein integral equation [38],*

$$u'(t) = 1 + sin^2(t) + \int_0^1 u^2(s)ds, 0 \leq t \leq 1, \tag{57}$$

$$K(t,s) = \begin{cases} -3sin(t-s), 0 \leq s \leq t \\ 0, \qquad\qquad\quad t \leq s \leq 1 \end{cases}$$

which has the exact solution $u(t) = cost$. *The numerical solutions of Equation (58) is obtained through the present method as explained in section 3 and presented in comparison with the exact solution are shown in the table 19 and in the figure 10, for* $N = 64$. *Maximum error analysis and CPU time are shown in table 20.*

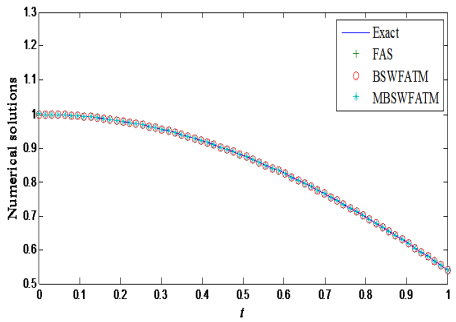

Figure 10. **Comparison of numerical solutions with exact solution of test problem 4.10, for** $N = 64$.

t	FAS	BSWFATM	MBSWFATM	EXACT
0 1.0000	1.0000	1.0000	1	
0.0666	0.9977	0.9977	0.9977	0.9977
0.1333	0.9911	0.9911	0.9911	0.9911
0.2000	0.9800	0.9800	0.9800	0.9800

t	FAS	BSWFATM	MBSWFATM	EXACT
0.2666	0.9646	0.9646	0.9646	0.9646
0.3333	0.9449	0.9449	0.9449	0.9449
0.4000	0.9209	0.9209	0.9209	0.9210
0.4666	0.8929	0.8929	0.8929	0.8930
0.5333	0.8610	0.8610	0.8610	0.8611
0.6000	0.8252	0.8252	0.8252	0.8253
0.6666	0.7857	0.7857	0.7857	0.7858
0.7333	0.7427	0.7427	0.7427	0.7429
0.8000	0.6965	0.6965	0.6965	0.6967
0.8666	0.6472	0.6472	0.6472	0.6473
0.9333	0.5950	0.5950	0.5950	0.5951
1	0.5401	0.5401	0.5401	0.5403

Table 19. **Numerical results of the test problem 4.10, for** $N = 16$

N	Method	E_{max}	Setup time	Running time	Total time
	FAS	1.60e-04	0.0160	0.0350	0.0510
16	BSWFATM	1.60e-04	0.0148	0.0312	0.0460
	MBSWFATM	1.60e-04	0.0100	0.0280	0.0380
	FAS	3.75e-05	0.0286	0.0485	0.0771
32	BSWFATM	3.75e-05	0.0149	0.0448	0.0597
	MBSWFATM	3.75e-05	0.0101	0.0400	0.0501
	FAS	9.10e-06	0.0790	0.1030	0.1820
64	BSWFATM	9.10e-06	0.0149	0.0994	0.1143
	MBSWFATM	9.10e-06	0.0101	0.0953	0.1054
	FAS	2.23e-06	0.2856	0.3512	0.6368
128	BSWFATM	2.23e-06	0.0167	0.3224	0.3391
	MBSWFATM	2.23e-06	0.0103	0.3268	0.3371

Table 20. **Maximum error and CPU time (in seconds) of the test problem 4.10**

5. Conclusion

The Spline wavelet transform methods is introduced for the numerical solution of integral and integro-differential equations. The biorthogonal spline wavelet (prolongation and restrictions) operators are used. The numerical solutions obtained agree well with the exact ones. Numerical results is observed when the calculation is refined by increasing the number N used. In this paper, the MG, FAS, BSWTM, BSWFATM, MBSWTM and MBSWFATM show the errors are same, but the CPU time changes. The standard MG, FAS, BSWTM and BSWFATM converge slowly with larger computational cost, as compared to MBSWTM and MBSWFATM ensure such slower convergence with lesser computational cost as given in tables and figures. Test problems are justified through the error analysis, as the level of resolution N increases, larger the accuracy increases. Hence, the new scheme is very convenient and efficient than the existing standard methods.

References

[1] H. R. Thiem, *A model for spatio spread of an epidemic*, J. Math. Biol., 4(1977), 337-351.

[2] A. M. Wazwaz, *Linear and Nonlinear Integral Equations Methods and Applications*, Springer, (2011).

[3] A. Jerri, *Introduction to integral equations with applications*, A wiley-interscience publication, Wiley, (1999).

[4] K. E. Atkinson, *The numerical solution of integral equations of the second kind*, Cambridge university press, (1997).

[5] P. Wesseling, *An introduction to Multigrid Methods*, Wiley, Chichester, (1992).

[6] W. L. Briggs, V. E. Henson and S. F. McCormick, *A Multigrid Tutorial*, SIAM, Philadelphia, (2000).

[7] U. Trottenberg, C. W. Oosterlee and A. Schuler, *Multigrid*, Academic Press, London, (2001).

[8] R. P. Fedorenko, *The speed of convergence of one iterative process*, USSR Comp. Math. Phys., 4(3)(1964), 227-235.

[9] N. S. Bakhvalov, *On the convergence of a relaxation method with natural constraints on the elliptic operator*, USSR Comp. Math. Phys., 6(5)(1966), 101-135.

[10] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31(1977), 333-390.

[11] A. Brandt, *Multi-level adaptive techniques for singular perturbation problems in Numerical Analysis of singular perturbation problems*, (P. W. Hemker and J. J. H. Miller, Eds.), Academic press, London, (1979).

[12] W. Hackbusch, *Multi-grid methods and applications*, Springer-Verlag, Berlin, (1985).

[13] W. Hackbusch, *Integral Equations: Theory and numerical treatment*, Inter. Ser. Numer. Math., Springer verlag, New york, (1995).

[14] H. Schippers, *Multigrid methods for boundary integral equations*, Numerische Mathematik, 46(3)(1985), 351-363.

[15] H. Schippers, *Application of multigrid methods for integral equations to two problems from fluid dynamics*, J. Comp. Phys., 48(1982), 441-461.

[16] C. Gaspar, *A fast multigrid solution of boundary integral equations*, Envir. Soft., 5(1)(1990), 26-28.

[17] H. Lee, *Multigrid method for nonlinear integral equations*, Korean J. Comp. Appl. Math., 4(1997), 427-440.

[18] S. Paul, *Numerical multigrid algorithm for solving integral equations*, Ball state university, Muncie, Indiana, (2014).

[19] W. Hackbusch and U. Trottenberg, *Multigrid Methods*, Springer-Verlag, Berlin, (1982).

[20] A. Avudainayagam and C. Vani, *Wavelet based multigrid methods for linear and nonlinear elliptic partial differential equations*, Appl. Math. Comp., 148(2004), 307320.

[21] H. Lee, *Multigrid method for nonlinear integral equations*, Korean J. Comp. Appl. Math., 4(1997), 427-440.

[22] G. Hariharan and K. Kannan, *An Overview of Haar Wavelet Method for Solving Differential and Integral Equations*, World Applied Sciences Journal, 23(12)(2013), 01-14.

[23] M. Unser, *Ten good reasons for using spline wavelets, Wavelets Applications in Signal and Image Processing V*, proc. Spie., 3169(1997) 422-431.

[24] A. Cohen, I. Daubechies and J. Feauvean, *Biorthogonal based of compactly supported wavelets*, Comm. Pure Appl. Math., 45(1992), 485-560.

[25] W. Sweldens, *The construction and application of wavelets in numerical analysis*, Ph.D Thesis, Department of Computer Science, Katholieke Universiteit Leuven, Belgium, (1994).

[26] D. K. Ruch and P. J. V. Fleet, *Wavelet theory an elementary approach with applications*, John Wiley and Sons, (2009).

[27] P. J. V. Fleet, *Discrete wavelet transformations an elementary approach with applications*, John Wiley and Sons, (2008).

[28] I. S. Kotsireas, *A Survey on Solution Methods for Integral Equations*, (2008).

[29] W. F. Blyth, R. L. May and P. Widyaningsih, *Volterra integral equations solved in fredholm form using walsh functions*, ANZIAM Journal, 45(2004), C269-C282.

[30] R. Farnoosh and M. Ebrahimi, *Monte Carlo method for solving Fredholm integral equations of the second kind*, Appl. Math. Comp., 195(2008), 309315.

[31] R. Ezzati, F. Mokhtaria and M. Maghasedi, *Numerical Solution of Volterra-Fredholm Integral Equations with The Help of Inverse and Direct Discrete Fuzzy Transforms and Collocation Technique*, Int. J. Indus. Math., 4(3)(2012), 221-229.

[32] M. Javidi and A. Golbabai, *Modified homotopy perturbation method for solving non-linear Fredholm integral equations*, Chaos Solitons Fractals, 40(2009), 1408-1412.

[33] J. Rashidinia and A. Parsa, *Analytical-Numerical Solution for Nonlinear Integral Equations of Hammerstein Type*, Inter. J. Math. Model. Comp., 2(1)(2012), 61-69.

[34] Y. Ordokhani and S. Davaei far, *Application of the Bernstein Polynomials for Solving the Nonlinear Fredholm Integro-Differential Equations*, J. Appl. Math. Bioinf., 1(2)(2011), 13-31.

[35] Y. Mahmoudi, *Wavelet Galerkin method for numerical solution of nonlinear integral equation*, Appl. Math. Comp., 167(2005), 1119-1129.

[36] B. Sepehrian and M. Razzaghi, *Solution of nonlinear volterra-hammerstein integral equations via single-term walsh series method*, Math. Prob. Eng., 5(2005), 547-554.

[37] S. M. El-Sayed and M. R. Abdel-Aziz, *A comparison of Adomians decomposition method and Wavelet-Galerkin method for solving integro-differential equations*, Appl. Math. Comp., 136(2003), 151-159.

[38] K. Parand and J. A. Rad, *Numerical solution of nonlinear VolterraFredholmHammerstein integral equations via collocation method based on radial basis functions*, Appl. Math. Comp., 218(2012), 5292-5309.

[39] S. C. Shiralashetti and R. A. Mundewadi, *Modified Wavelet Full-Approximation Scheme for the Numerical Solution of Nonlinear Volterra integral and integro-differential Equations*, Applied Mathematics and Nonlinear Sciences, 1(2)(2016), 529-546.

[40] S. C. Shiralashetti and R. A. Mundewadi, *Bernoulli Wavelet Based Numerical Method for Solving Fredholm Integral Equations of the Second Kind*, Journal of Information and Computing Science, 11(2)(2016), 111-119.

[41] S. C. Shiralashetti, H. S. Ramane, R. A. Mundewadi and R. B. Jummannaver, *A Comparative Study on Haar Wavelet and Hosaya Polynomial for the numerical solution of Fredholm integral equations*, Applied Mathematics and Nonlinear Sciences, 3(2)(2018) 447-458.

[42] R. A. Mundewadi and S. Kumbinarasaiah, *Numerical Solution of Abels Integral Equations using Hermite Wavelet*, Applied Mathematics and Nonlinear Sciences, 4(1)(2019) 181-192.