International Journal of Electronics and Electrical Engineering

Volume 3 | Issue 2 Article 5

October 2014

PIECEWISE CONSTANT SOLUTION OF NON LINEAR VOLTERRA INTEGRAL EQUATION

S. CHARAN TEJA
Electrical Engg. Deptt., National Institute of Technology, Kurukshetra (Haryana), India, s.charanteja@gmail.com

MONIKA MITTAL
Electrical Engg. Deptt., National Institute of Technology, Kurukshetra (Haryana), India, monika_mittalkkr@rediffmail.com

PIECEWISE CONSTANT SOLUTION OF NON LINEAR VOLTERRA INTEGRAL EQUATION

S. CHARAN TEJA, MONIKA MITTAL[2]

Electrical Engg. Deptt., National Institute of Technology, Kurukshetra (Haryana), India.
E-mail: s.charanteja@gmail.com, monika_mittalkkr@rediffmail.com

Abstract - In this paper, modification in the computational methods, for solving Non-linear Volterra integral equations, is presented. Here, two piecewise constant methods are considered for obtaining the solutions. The first method is based on Walsh Functions (WF) and the second method is via Block Pulse Functions (BPF). Comparison between the two methods is presented by calculating the errors vis-à-vis exact solution. Computational efficiency of BPF is established by profiling the computations for two examples with MATLAB 7.10 Profiler..

I. INTRODUCTION

The Nonlinear Volterra Integral equation is widely used for solving many problems in mathematical physics and engineering. Using the Solution of volterra equations, the necessary and sufficient conditions for mean square stability, stochastic stability and mean exponential stability are obtained which can be equivalently tested in terms of a matrix eigen value computation, an LMI feasibility problem, and a Nyquist criterion condition and assess performance of the impulsive system by computing a second moment Lyapunov exponent.

In this paper, two computational methods Block Pulse functions and Walsh functions for solving nonlinear Non linearVolterra integral equations are presented and some modifications are proposed. The Volterra equation can be solved by using Taylor polynomial solutions[1], an adaptive method for the solution of the volterra equation[2], and many numerical methods[3]. Most of the numerical methods are not computationally efficient in solving the nonlinear volterra equation. To reduce the time and computational effort piecewise constant solution methods can be applied. The main advantage of using Piecewise constant solution is that, it reduces the integro-differential equation into a simple matrix algebraic equation and hence make it very easy to solve complicated non linear volterra integral equation. Hence they provide a more computational efficient alternative in comparison to other numerical methods given in[1-3].

The paper is organized as follows: firstly, some special properties of the Walsh functions are given followed by the solution of volterra equation using Walsh function. In the next section, some properties of the Block Pulse Functions are given followed by the solution the volterra equation using Block Pulse Function. Simulation results are presented next and comparison between the two methods for a illustrative example is done for same resolution.

II. WALSH FUNCTIONS:

Walsh function may be generated using Rademacher functions[4] using the relation.

$$w_n(t) = [r_q(t)]^{d(q)}\,[r_{q-1}(t)]^{d(q-1)} \ldots\ldots$$

where wn(t) is the (n+1)-th member of {wi(t)} ordered in a particular way, and
q=|[log2n]|+1
There are many kinds of ordering of Walsh function. We chose only one particular form called the Payley from here. In a m-set of Walsh function, m=2k, where k is a positive integer. Thus
w0(t) = r0(t),
w1(t) = r1(t),
w2(t) = r2(t),
w3(t) = r2(t) r1(t),…
The system of Walsh Function is orthonormal and complete.
A 4-set of Walsh Function is shown in below.

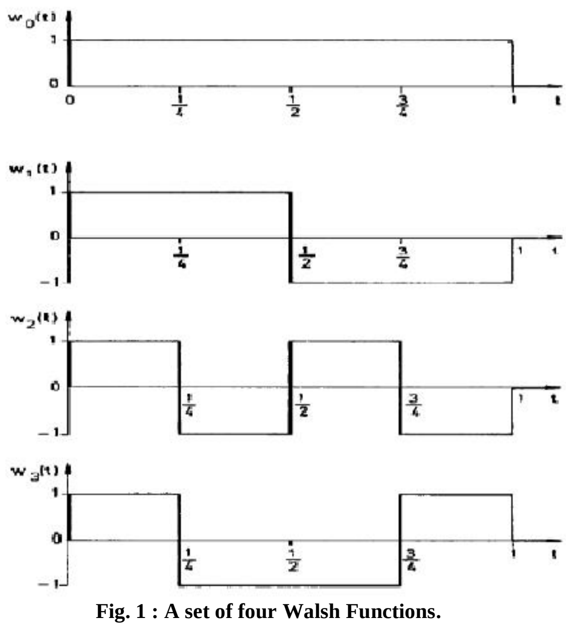

Fig. 1 : A set of four Walsh Functions.

Walsh functions have many properties similar to those of the trigonometric functions. For example they form a complete, total collection of functions with respect to the space of square Lebesgue integrable functions. However, they are simpler in structure to the trigonometric functions because they take only the values 1 and -1. The Walsh (-Payley) system may be obtained, following Paley, from the Rademacher system.

For example consider that resolution (m)=4
The Walsh operational matrix is given below

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Since walsh function matrix contains no zeros so, the time taken for the computation will increase accordingly.

2.1 Properties of Walsh Fucntions:

For a functions of two independent variables, which are absolutely integrable with respect to both variables, tε[0,1),t" ε[0,1), an be expanded by double Walsh series[5], approximately.

$$h(t,t') = W^T_{(m)}(t)\, H_{(mxn)}\, W^T_{(n)}(t') \qquad (1)$$

$$H_{mxn} = [W_m(t')]^{-1}\, h(t,t')\, [W_n(t)]^{-1}$$

By using the above expression [5] the two independent variables expression can be obatined. Normally the matrix is obtained by using (1), but by using the proposed equation computational effort can be reduced.

where $m=2^d$, $n=2^c$, d and c are integers, "T" means transpose, the walsh vectors.

$$W^T_{(m)}(t)=[\, W_{(0)}(t),\, W_{(1)}(t),\ldots\ldots W_{(m-1)}] \qquad (2)$$

$$W^T_{(n)}(t')=[\, W_{(0)}(t'),\, W_{(1)}(t'),\ldots W_{(m-1)}(t')] \qquad (3)$$

$$H(mxn) = \begin{bmatrix} h00 & h01 & .. & h0,n-1 \\ h10 & h11 & .. & h1,n-1 \\ .. & .. & .. & .. \\ hm-1,0 & hm-1,1 & .. & hm-1,n-1 \end{bmatrix} \qquad (4)$$

Using the orthonormal property of the Walsh functions, the coefficients hij are evaluated as follows:

$$hij = \int_0^1 \int_0^1 h(t,t')W_i(t)W_j(t') \qquad (5)$$

The integration of the Walsh vector W(m)(t) has been given by C.F. Chen and Hsiao in term of operational matrix [6] P(mxm):

$$\int_0^t W(m)(t')dt' = P(mxm)W(m)(t) \qquad (6)$$

$$P_{mxm} = \begin{bmatrix} P(\frac{m}{2}X\frac{m}{2}) & \frac{-1}{2*m}I(\frac{m}{2}X\frac{m}{2}) \\ \frac{1}{2*m}I(\frac{m}{2}X\frac{m}{2}) & 0(\frac{m}{2}X\frac{m}{2}) \end{bmatrix} \qquad (7)$$

Thus, by using the above recursive algorithm the integration matrix of Walsh function can be computed. But this recursive nature required more computational time especially at higher resolutions „m".

For example consider m=4, then the operational matrix for integration using Walsh function is given by

$$P_{4x4} = \begin{bmatrix} 1/2 & -1/4 & -1/8 & 0 \\ 1/2 & 0 & 0 & -1/8 \\ 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \end{bmatrix}$$

Letting x(t') be expanded in Walsh series [6] as

$$x(t') = x_{(n)}^T\, W^T_{(n)}(t'), \qquad (8)$$

with $x_{(n)}^T(t)=[x_0, x_1,\ldots\ldots,x_{n-1}]$ (9)

Since $W_i(t)\, W_i(t)=W_{i\oplus j}(t)$, it is found that

$$W_{(n)}(t')W^T_{(n)}(t')= x_{(n)} = X_{(nxn)}\, W^T_{(n)}(t') \quad (10)$$

where $W_{(n)}(t')\, W^T_{(n)}(t')$ is called the Walsh product matrix. Here the coefficient Matrix $X_{(nxn)}$ corresponding to the coefficient vector $x_{(n)}$ is as follows [6]:

n=2

$$X(2x2) = \begin{bmatrix} X0 & X1 \\ X1 & X0 \end{bmatrix} \qquad (11)$$

n=4

$$X(4x4) = \begin{bmatrix} X0 & X1 & X2 & X3 \\ X1 & X0 & X3 & X2 \\ X2 & X3 & X0 & X1 \\ X3 & X2 & X1 & X0 \end{bmatrix} \qquad (12)$$

The quadratic form [6] $W^T_{(m)}(t)Q_{(mxn)}\, W^T_{(n)}(t)$ with

$$Q(mxn) = \begin{bmatrix} q00 & q01 & .. & q0,n-1 \\ q10 & q11 & .. & q1,n-1 \\ .. & .. & .. & .. \\ qm-1,0 & qm-1,1 & .. & qm-1,n-1 \end{bmatrix} \qquad (13)$$

Can be easily converted into Walsh series by a direct manipulation. That is

$$W^T_{(m)}(t)Q_{(mxn)}\, W^T_{(n)}(t) = y_r^T\, W^T_{(r)}(t) \quad (14)$$

where r=max[m,n] (15)

$$y_r^T=[y_0\, ,\, y_1,\ldots\ldots,y_{r-1}] \qquad (16)$$

$$yi = \Sigma\, qjk\,, \qquad j \oplus k = i \qquad (17)$$

Where $\oplus$ is modulo two addition.

For example, m=4=n, r=4

$$y(4) = \begin{bmatrix} q00 + q11 + q22 + q33 \\ q01 + q10 + q23 + q32 \\ q02 + q20 + q13 + q31 \\ q03 + q30 + q21 + q12 \end{bmatrix} = \begin{bmatrix} y0 \\ y1 \\ y2 \\ y3 \end{bmatrix} \quad (18)$$

This coefficient matrix can also be found using recursive formulation given in [7]. Now these integral operational matrix, coefficient matrix are used in walsh function method to obtain the solution of non linear volterra integral equation.

2.2 Walsh Function Method :

The general Volterra integral equation can be written as follows [7,8]:

$$y(x) = f(x) + \lambda \int_a^b K(x,t)y(t)dt$$

It is called fredholm equation as limits of integral are 0 to 1

$$y(x) = f(x) + \int_0^1 K(x,t)y(t)dt \qquad (19)$$

The Walsh approximates of y(x), f(x), K(x,t) are given by

$$y(x) = y_{(m)}^T W_{(m)}(x) \qquad (20)$$

$$f(x) = x_{(m)}^T W_{(m)}(x) \qquad (21)$$

where $y_{(m)}^T = [y_0, y_1, \dots\dots y_{m-1}]$ (22)

$x_{(m)}^T = [x_0, x_1, \dots\dots x_{m-1}]$ (23)

$$K(x,t) = W^T_{(m)}(x)\, H_{(mxn)}\, W^T_{(n)}(x) \qquad (24)$$

Substituting these approximates into (19), we have

$$y_{(m)}^T W_{(m)}(x) = x_{(m)}^T W_{(m)}(x) + \int_0^1 W^T_{(m)}(x) H_{(mxn)} W^T_{(n)}(t)\, W_{(m)}(t) y_{(m)} dt \qquad (25)$$

$$y_{(m)}^T W_{(m)}(x) = x_{(m)}^T W_{(m)}(x) + W_{(m)}(x)\, H_{(mxm)}\, Y_{(mxm)} \int_0^1 W_{(m)}(t)\, dt \qquad (26)$$

From the definition of the Walsh functions the integration of $W_{(m)}(t)$ from t=0 to t=1 is

$$\int_0^1 W_{(m)}(t)\, dt' = \begin{bmatrix} 1 \\ 0 \\ . \\ . \\ 0 \end{bmatrix} \qquad (27)$$

Substituting (27) into (26) gives

$$y_{(m)}^T W_{(m)}(x) = x_{(m)}^T W_{(m)}(x) + W_{(m)}^T(x)\, H_{(mxm)}\, Y_{(mxm)} \quad (28)$$

Therefore, the set of algebraic equations for y_i is

$$Y_{(m)} = x_{(m)} + H_{(mxm)}\, Y_{(mxm)} \, (29)$$

Hence by using the above algorithm the solution of the non linear volterra equation can be obtained.

III. BLOCK PULSE FUNCTIONS:

Block pulse functions form a set of orthogonal functions with piecewise constant values and are usually applied as a useful tool in the analysis, synthesis, identification and other problems of control and systems science.[4]. The first four Block Pulse Functions are shown in fig.2

A m-set Block Pulse Function is defined as

$$B_i(t) = \begin{cases} 1, & \frac{i-1}{m} \le t < m, for\ all\ i = 1,2, \dots, m \\ & 0,\ \text{otherwise} \end{cases}$$

For example consider m=4 the Block Pulse Function Matrix is given by

$$B_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here it is observed that the block pulse function matrix contains identity matrix. So, it is easy for computation.

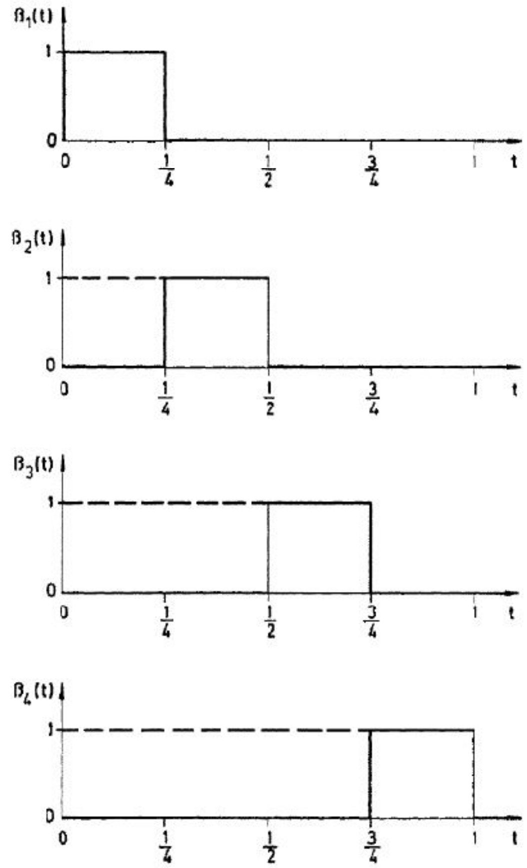

Fig. 2 : A set of four block Pulse functions

3.1 Properties of Block Pulse Functions:

The Block Pulse Functions are disjoint and orthonormal. That is,[4]

$$B_i(t)\, B_j(t) = \begin{cases} 0, i \neq j \\ Bi(t), i = j \end{cases}$$

$$(B_i(t)\, B_j(t)) = \begin{cases} 0, i \neq j \\ \frac{1}{m}, i = j \end{cases}$$

y(t) can be expanded approximately into a set of the block pulse functions as

$$y(t) = \sum_{i=1}^{m} Ci\, B_i(t) \qquad (30)$$

Where $C_i = m\int_0^1 y(t)B_i(t)dt$

$$= m\int_{(i-1)/m}^{i/m} y(t)dt \qquad (31)$$

is determined to minimize the integral square error

$$\epsilon = \int_0^1 (y(t) - \sum_{i=1}^{m} Ci\ Bi(t))^2 dt.$$

Ci‟s given by (31) are called coefficients of block pulse series. The integrations of the block pulse functions gives the results shown in Fig 2. are written in terms of the block pulse functions, we obtain

$$\int Bm(t)dt = H_{(mxm)}\, Bm(t) \qquad (32)$$

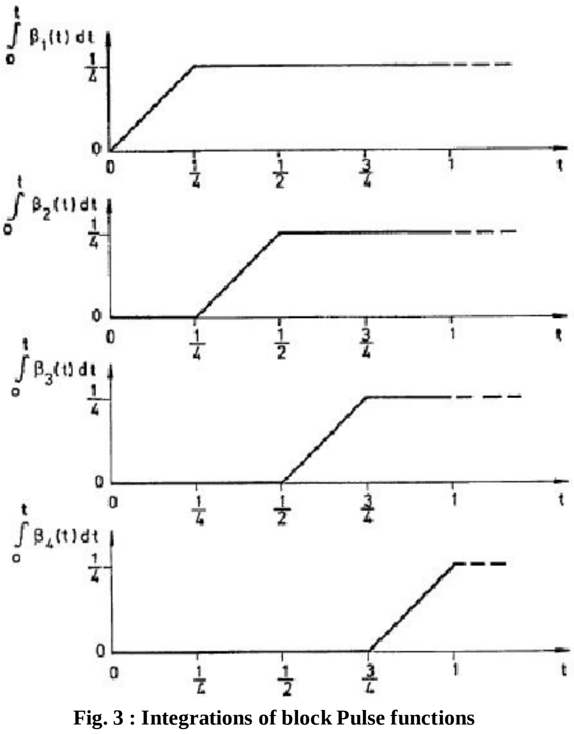

Fig. 3 : Integrations of block Pulse functions

Where Bm(t) denotes an m-vector, and H(mxm) is an mxm matrix given by [4]

$$H_{(mxm)} \triangleq \begin{bmatrix} h1 \\ h2 \\ . \\ . \\ .hm \end{bmatrix} = \frac{1}{m}\begin{bmatrix} 1/2 & 1 & 1 & .. & 1 \\ 0 & 1/2 & 1 & .. & 1 \\ . & . & . & .. & . \\ . & . & . & .. & . \\ 0 & 0 & 0 & .. & 1/2 \end{bmatrix} \quad (33)$$

Consider that resolution (m)=4, the operational matrix for integration using Block Pulse Function is given by

$$H_{(4x4)} = \frac{1}{4}\begin{bmatrix} 1/2 & 1 & 1 & 1 \\ 0 & 1/2 & 1 & 1 \\ 0 & 0 & 1/2 & 1 \\ 0 & 0 & 0 & 1/2 \end{bmatrix}$$

Here it can be observed that the integration matrix is not recursive and there are more number of zeros, so the time taken for the computation is less.

3.2 Block Pulse Function method (BPF):

Consider the following integral equation[7][8]

$$y(x) = f(x) + \lambda\int_a^b K(x,t)y(t)dt \qquad (34)$$

where a(x),f(x) and the kernel K(x,t) are given, and a, b and λ are constants.

Suppose a(x), y(x) and f(x) are all absolutely integrable in[0,1). Then, the approximate expansions of a(x),y(x) and f(x) into the block pulse series are given as follows:

$$a(x) = a^T B_m(x) \qquad (35)$$

$$y(x) = C^T B_m(x) \qquad (36)$$

$$f(x) = f^T B_m(x) \qquad (37)$$

where $a^T=[a_1, a_2, \ldots\ldots, a_m]$ and $f^T=[f_1, f_2, \ldots\ldots, f_m]$ are constants, and $C^T=[C_1, C_2, \ldots.. C_m]$ has to be determined.

$$B_i(x)\, B_j(x) = \begin{cases} 0, for\ i \neq j \\ Bi(x), for\ i = j \end{cases}$$

the multiplication of (35) and (36) yields

$$a(x)\, y(x) = C^T B_m(x)\, a^T B_m(x)$$
$$= C^T A_{(mxm)} B_m(x) \qquad (38)$$

$$\text{where} \quad A_{(mxm)} = \begin{bmatrix} a1 & 0 & 0 & .. & 0 \\ 0 & a2 & 0 & .. & 0 \\ . & . & . & .. & . \\ . & . & . & .. & . \\ 0 & 0 & 0 & .. & am \end{bmatrix} \quad (39)$$

Let K(x,t) be absolutely integrable in $x\varepsilon[0,1)$ and in $t\varepsilon[0,1)$. Then, it can be expanded approximately into the block pulse series with respect to x. That is

$$K(x,t) = \sum_{u=1}^{m} ki(t)\ B_i(x) \quad (40)$$

where $$k_i(t) = m\int_{(i-1)/m}^{i/m} K(x,t)dx.$$

Also, $k_i(t)$ can be expanded into the block pulse series as

$$k_i(t) = \sum_{j=1}^{n} Bj(t)k_{ji} \quad (41)$$

$$\text{where } k_{ji}=n\int_{(j-1)/n}^{j/n} ki(t)dt$$
$$= mn\int_{(j-1)/n}^{j/n}\int_{(i-1)/m}^{i/m} K(x,t)dx\,dt \quad (42)$$

Substitute (41) into (40) gives

$$K(x,t) = \sum_{i=1}^{m}\sum_{j=1}^{n} B_j(t)\ k_{ji}\ B_i(x)$$
$$= B_{(n)}{}^{T}(t)\ K_{(nxm)}\ B_m(x) \quad (43)$$

where $K_{(nxm)} = [k_{ji}]_{(nxm)}$. For convenience, it is possible to set n=m. Substitute (36) and (43) with n=m into the last term of (34) and obtain

$$\lambda\int_a^b K(x,t)y(t)dt$$
$$= \lambda\int_a^b C^T B_m(t)\ B_{(m)}{}^{T}(t)\ K_{(nxm)}\ B_m(x)\ dt$$
$$=\lambda\ C^T \int_a^b B_{(mxm)}(t)dt\ K_{(mxm)}\ B_m(x)$$
$$=\lambda\ C^T\ (\phi_{(mxm)}(b) - \phi_{(mxm)}(a))K_{(mxm)}\ B_m(x) \quad (44)$$

The substitution of (37) ,(38) and (44) in (34) yields

$$C^T A_{(mx)}\ B_m(x)$$
$$= f^T\ B_m(x) + \lambda\ C^T\ (\phi_{(mxm)}(b) - \phi_{(mxm)}(a))K_{(mxm)}\ B_m(x)$$
$$C^T = f^T\ [A_{(mxm)} - \lambda((\phi_{(mxm)}(b) - \phi_{(mxm)}(a))\ K_{(mxm)}]^{-1} \quad (45)$$

Thus we obtain the coefficients of block pulse series expansion of the solution (34). It is seen that (45) is suitable for computer computation.

IV. SIMULATION RESULTS:

Consider an example of volterra equation of second kind[6].

$$y(t) = \frac{5}{6}t - \frac{1}{9} + \frac{1}{3}\int_0^1 (t + t')y(t')dt'$$

The exact solution (E.S.) is y(t) =t

Piecewise constant solutions are calculated using Walsh Functions using eqn.(29) and by using Block Pulse Function using eqn.(45) for resolution(m) = 4 are tabulated below:

Table 1:

t	WF	BPF	E.S	E.W	E.BPF
1/8	0.0243	0.1227	0.1250	0.1007	0.0023
3/8	0.2951	0.3725	0.3750	0.0799	0.0025
5/8	0.5229	0.6323	0.6250	0.1021	0.0027
7/8	0.7639	0.8720	0.8750	0.111	0.0030

Here, E.W is the error obtained with Walsh Function method, E.BPF is the error obtained with Block Pulse Function method.

From the above table it can be understood that the error obtained using Block Pulse Function is less when compared with the Walsh Function. This is also clearly visible in bar graph shown in fig.4

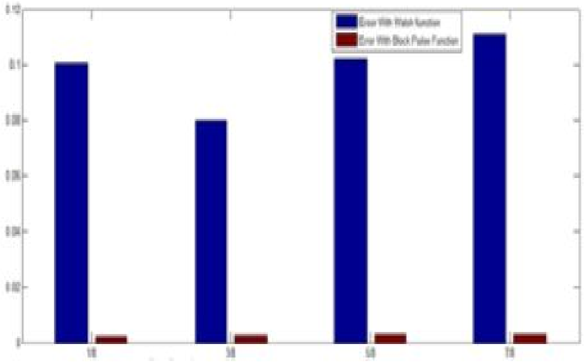

Fig. 4 Error Comparison Graph

If time consumed by WFM and BPFM is also done for same number of file calls using MATLAB 7.10 profiler shown in fig 5,fig 6.It is evident that BPF.M is fast as compare with WF.M for m=4. This is also clearly visible in bar graph shown in fig.7

MATLAB 7.10 Profiler figures for example

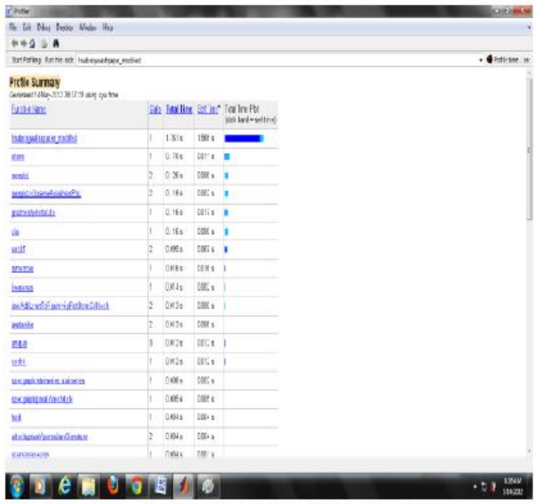

Fig. 5 MATLAB Profiler View of Block Pulse Functions

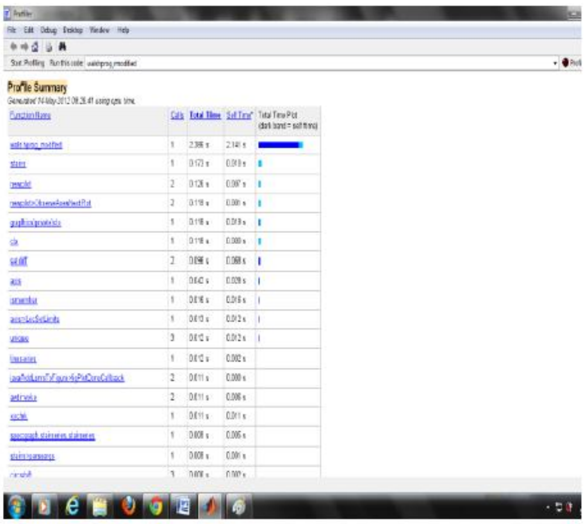

Fig. 6 MATLAB Profiler View of Walsh functions.

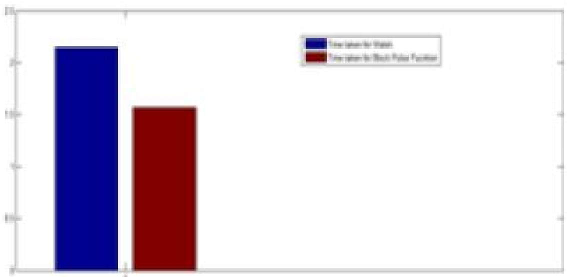

Fig. 7 Time Comparison Graph

From the above two MATLAB profiler figures it can be observed that the time consumed for the solution of Volterra equation using Block Pulse Function is less when compared with the Walsh functions for the same number of file calls. If the resolution (m) is increased the time taken for Walsh function increases as it contains recursive matrices.

V. CONCLUSION:

Two piecewise constant computational methods WF and BPF, are used to obtain solution of non-linear Volterra integral equation successfully. The two methods are compared by calculating the error vis-à-vis exact solution for m = 4. Computational efficiency of BPF is established by profiling the computations for same example with MATLAB 7.10 PROFILER establishing the faster computations in the case of BPFs for the same number of file calls. Also it has been demonstrated that smaller errors are obtained via BPF method. The computational advantage vis-à-vis WF method is more pronounced at higher resolutions.

The simplicity of BPFs make them more suitable for faster computer implementations and efficiently solving nonlinear Volterra integral equation.

REFERENCES:

[1] Mehmet Sezer "Taylor polynomial solutions of Volterra integral equations", Int. J. Math. Educ. Sci. Technol., 1994, VOL. 25, NO. 5, 625-633.

[2] Joshua H. Gordis, Beny Neta "An Adaptive Method for the Numerical Solution of Volterra Integral Equations".

[3] Ida Del Prete "Efficient Numerical Methods For Volterra Integral Equations Of Hammerstein Type", Dottorato Di Ricerca In Scienze Computazionali E Informatiche Ciclo Xviii.

[4] Ganti Prasada Rao "Piecewise Constant Orthogonal Functions and Their Application to Systems and Control", Springer-Verlag Berlin Heidelberg New York Tokyo 1983.

[5] Jiunn-Lin Wu " The Operational Matrix of Orthogonal Functions for Differential Equations", National Cheng Kung Universiy Taiwan, Taiwan, Republic of China December, 2003.

[6] Yen-Ping Shih, Wei-kong Chia, "Piecewise constant solutions of integral equations via Walsh functions", Journal of the Chinese Institute of Engineers,2011.

[7] F.C. Kung and S.Y. Chen "Solution of Integral Equations Using a Set of Block Pulse Functions", Cheng Kung University, Taiwan.

[8] A. Shshsavaran "Computational Method to Solve Nonlinear Integral Equations Using Block Pulse Functions by Collocation Method", Applied Mathematical Sciences, Col.5,2011, no. 65, 3211-3220.

◈◈◈