ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 8, No. 4, 2013, *pp*. 243-255

Tau numerical solution of the Volterra-Fredholm Hammerstein integro-differential equations with the Bernstein multi-scaling functions

Yadollah Ordokhani 1+ , Solmaz Moosavi 1 and Mohsen Shahrezaee 2

1 Department Of Mathematics, Alzahra University, Tehran, Iran

2 Department Of Mathematics, Emam Hossein University, Tehran, Iran

(*Received January 20, 2013, accepted June 13, 2013*)

Abstract. This paper involves the development of the Tau method with Bernstein multi-scaling (BMS) functions basis for the numerical solution of the Volterra-Fredholm Hammerstein integro-differential equations (VFHIDEs). For this purpose at the beginning we define BMS functions and express briefly some properties of BMS functions and after function approximation by using BMS functions, will be presented. Then, the operator matrix representation for the differential and integral parts seeming in the equation using the operational Tau method base on BMS functions basis, will be displaced. The operational Tau method transforms the differential and integration parts of the desired VFHIDEs to some operational matrices. In fact, this method reduces VFHIDEs to a system of algebraic equations. Numerical examples demonstrate the validity and applicability of the proposed method with BMS functions basis.

Keywords: Bernstein multi-scaling functions , Operational Tau method , Hammerstein integro-differential equation, Algebraic equation, Fredholm, Volterra.

1. Introduction

Let us consider the general form of VFHIDE

$$Du(t)-\lambda_1\int_0^t k_1(t,s)G_1(s,u(s))ds-\lambda_2\int_0^1 k_2(t,s)G_2(s,u(s))ds = f(t) \quad 0\le t\le 1, \tag{1}$$

with n_d independent boundary conditions

$$\sum_{s=1}^{n_d}[c_{js}^{(1)}u^{s-1}(t_1)+c_{js}^{(2)}u^{s-1}(t_2)] = d_j, \quad j=1,2,3,...,n_d \tag{2}$$

where $f(t)$, $k_1(t,s)$ and $k_2(t,s)$ are given continuous functions. λ_1 , λ_2 , $c_{js}^{(1)}$ and $c_{js}^{(2)}$, are given constants and $t_1,t_2\in[0,1]$. u(t) is the unknown function to be determined and $G_1(s,u(s)),G_2(s,u(s))$ are analytic functions of the unknown function u(s). n_d is order of the differential operator D with polynomial coefficients $p_i(t)$

$$D=\sum_{i=0}^{n_d}p_i(t)\frac{d^i}{dt^i}, \quad p_i(t)=\sum_{j=0}^{\alpha_i}p_{ij}t^j,$$

where α_i is the degree of $p_i(t)$.

In this section, some numerical methods that discuss about solutions of Volterra-Fredholm integro-differential equations will be presented. Ordokhani [1] has used walsh functions operational matrix with Newton-Cotes nodes for solving Fredholm-Hemmerstein integro-differential equations. Arikoglu et al. [2] by using differential transform method obtained numerical solution of integro-differential equations. Babolian in [3], obtained solutions of nonlinear Volterra-Fredholm integro-differential equations by using direct computational method and triangular functions. With in [4], hybrid Legendre polynomials and Block-

+ Corresponding author. Tel.: +98-21-88048931; fax: +98-21-88048931.
E-mail address: ordokhani@alzahra.ac.ir.

Pulse functions are presented to approximate the solution of Volterra-Fredholm integro-differential equations. Saberi Nadjafi and Ghorbani in [5] have used his homotopy perturbation method for solving integral and integro-differential equations.

Also, In [6−12] different numerical methods exist for resolving linear and nonlinear integro-differential equations.

Recently, the authors, have used the operational Tau method for the numerical solution of linear and nonlinear Fredholm and Volterra integral and integro-differential equations of second kind. Authors [13−19], developed the Tau method to find numerical solutions of the Fredholm, Volterra and Fredholm-Volterra integral and integro-differential equations with arbitrary polynomial bases.

In this work, we are interested in solving VFHIDEs with an operational approach of the Tau method based on BMS functions. Because in the Tau method, we obtain a system of algebraic equations wherein its solution is easy. The paper is organized as follows: In Section 2, we define BMS functions and we give function approximation by using BMS functions. We drive matrix representation of differential, integral and supplementary conditions parts, in Section 3. Numerical examples are given in Section 4 to illustrate the accuracy of our method. Finally, concluding remarks are given in Section 5.

2. Basic definitions

2.1. Bernstein polynomials and their properties

For $m \geq 0,$ the Bernstein polynomials (B-polynomials) defined on the interval $[0,1]$ as follows [20]

$$B_{i,m}(t) = \binom{m}{i} t^i (1-t)^{m-i}, \quad that \quad \binom{m}{i} = \frac{m!}{i!(m-i)!},$$

where

i) $B_{i,m}(t) = 0,$ if $i < 0$ or $i > m.$

ii) $\{B_{i,m}(t), i = 0,1,...,m\}$ in Hilbert space $L^2[0,1]$, is a complete non orthogonal set [21].

2.2. BMS functions and function approximation

For $m \geq 1$ and any positive integer $k > 1,$ the BMS functions $\psi_{i,n}, i = 0,1,...,m$ and $n = 0,1,...,k-1$ are defined on the interval $[0,1)$ as [22]

$$\psi_{i,n}(t) = \begin{cases} B_{i,m}(kt-n), & \frac{n}{k} \leq t < \frac{n+1}{k}, \\ 0 & otherwise. \end{cases} \tag{3}$$

In equation (3), m is the order of B-polynomials on the interval $[0,1],$ n is the translation argument and t is the normalized time.

If $\phi(t)$ be a vector function of BMS functions on the interval $[0,1),$ as $\phi(t) = [\psi_{0,0}(t), \psi_{1,0}(t),..., \psi_{m-1,0}(t), \psi_{m,0}(t),..., \psi_{0,k-1}(t), \psi_{1,k-1}(t),..., \psi_{m-1,k-1}(t), \psi_{m,k-1}(t)]^T,$ then by taking integration of the cross product of two of these vector functions, a matrix of $k(m+1)\times k(m+1)$ dimensional will be resulted which will be indicated as follow

$$D = <\phi, \phi> = \int_0^1 \phi(t)\phi^T(t)dt. \tag{4}$$

This matrix is known by dual operational matrix of $\phi(t)$([22]).

A function f(t) defined over $[0,1]$ may be expanded in terms of BMS functions as

$$f(t) \cong \sum_{n=0}^{k-1}\sum_{i=0}^{m} f_{i,n}\psi_{i,n}(t) = F^T\phi(t),$$

where $\phi(t)$ is the vector function defined before and C is a $k(m+1)\times 1$ vector given by $F = [f_{0,0}, f_{1,0},..., f_{m-1,0}, f_{m,0},..., f_{0,k-1}, f_{1,k-1},..., f_{m-1,k-1}, f_{m,k-1}]^T,$ and can be obtained by [22]