Bernstein Polynomials Solving One Dimensional Delay Volterra Integro Differential Equations

Shymaa Hussain Salih

Received on: 8/6/2010
Accepted on: 7/10/2010

Abstract
The main purpose of this paper lies briefly in submitting least square method for solving linear delay Volterra integro differential equation of the second kind containing three types (Retarded, Neutral and mixed) with the aid Bernstein polynomials as basis functions to compute the approximated solutions of delay volterra integro differential equations. Three examples are given for determining the results of this method.

Keywords: Delay Volterra integro differential equation of the second kind, least square method, Bernstein polynomials, least square error.

Introduction
Delay integro differential equation is an equation involving one or more unkown function $U(x)$ togethet with both differential and integral operations on $U(x)$. This means that it is an equation containing derivative of the unknown function $U(x)$ which appears outside the integral sign.

Recall form of linear Volterra Integro-differential equation of the second kind [1]:

$$\frac{dU(x)}{dx} = f(x) + \lambda \int_{0}^{x} K(x, y) U(y) dy$$

..(1)

where $K(x, y)$ is the kernel function and $f(x)$ is any continuous function and $\lambda$ is a scalar parameter.
Three types of linear delay Volterra integro differential equations are defined

1. Retarded integro-differential equation when delay comes in the unknown function $u(x)$ involved in the integrand sign.

$$\frac{du(x)}{dx} = f(x) + \int_{0}^{x} K(x,t)u(y - \tau)dy, 0 \leq x$$

2. Neutral integro-differential equation when delay comes in the derivative of $u(x)$ outside the integral.

$$\frac{du(x)}{dx} = f(x) + \int_{0}^{x} K(x,t)u(y)dy, 0 \leq x$$

3. Mixed integro-differential equation

$$\frac{du(x)}{dx} = f(x) + \int_{0}^{x} K(x,t)u(y - \tau_2)dy, 0 \leq x$$

Where $\tau, \tau_1, \tau_2$ are positive integers called delay or time lags.

Numerical Solution: The discrete form for the exact solution $U(x)$ for equation (1) can be written in the form:

$$U_N(x) = \sum_{i=0}^{N} a_i B_{i,N}(x) \quad \ldots(2)$$

where $B_{i,N}(x) = \binom{N}{i} (1-x)^{N-i}$

for $i=0,1,2,...,n$, where the combination

$$\binom{N}{i} = \frac{N!}{i! (N-i)!}$$

There are $N+1$ Nth degree Bernstein polynomials. The Bernstein polynomials of degree 1 are:

$$B_{0,1}(x) = 1 - x$$
$$B_{1,1}(x) = x$$

For the graph of these functions when $0 \leq x \leq 1$ see [2].

The Bernstein polynomials of deg.2 are

$$B_{0,2}(x) = (1-x)^2$$
$$B_{1,2}(x) = 2x(1-x)$$
$$B_{2,2}(x) = x^2$$

The graph of which can be found in [3]. Finally, the Bernstein polynomials of degree 3 may be calculated as:

$$B_{0,3}(x) = (1-x)^3$$
$$B_{1,3}(x) = 3x(1-x)^2$$
$$B_{2,3}(x) = 3x^2(1-x)$$
$$B_{3,3}(x) = x^3$$

and their graph has shown in [3, 4] as well. Some necessary characters of Bernstein polynomials: (for the proof see [5,6]) Therefore we could write the function

$$U_N(x) = U_N(x) = a_0B_{0,N}(x) + a_1B_{1,N}(x) + \ldots + a_NB_{N,N}(x)$$

$$= a_0 \sum_{k=0}^{N} (-1)^k \binom{N}{k} x^k + \ldots +$$
$$+ a_N \sum_{k=N}^{k-N} (-1)^k \binom{N}{k} x^k$$

Bernstein Polynomials Solving One Dimensional Delay Volterra Integro Differential Equations

\[ = a_0 + d_1 \left[ \sum_{k=0}^{N} (-1)^{k-1} \binom{N}{k} \right] x^k + \ldots + \]

\[ + d_N \left[ \sum_{k=0}^{N} (-1)^{k-N} \binom{N}{k} \right] x^N \]

This system can be written in a matrix form as follows

\[ U_N(x) = \begin{bmatrix} 1 & x & x^2 & \ldots & x^N \end{bmatrix} \begin{bmatrix} a_{0x} & 0 & \cdots & 0 \\ 0 & a_{1x} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{Nx} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{bmatrix} \]

Using operator forms, equation (3) can be written as \( L[U] = f(x) \)

Using operator forms this equation (1) can be written as \( L[U] = f(x) \)

Where the operator \( L \) is defined for each type of delay integro–differential equation as:

1. Retarded integro-differential equation
   \[ L[u(x)] = \int_{0}^{x} k(x, \tau) u(t-\tau) d\tau \]

2. Neutral integro-differential equation
   \[ L[u(x)] = \frac{du(x)}{dx} - \int_{0}^{x} k(x, \tau) u(t-\tau) d\tau \]

3. Mixed integro-differential equation
   \[ L[u(x)] = \frac{du(x)}{dx} - \int_{0}^{x} k(x, \tau) u(t-\tau_1) d\tau - \int_{0}^{x} k(x, \tau) u(t-\tau_2) d\tau \]

The unknown function \( U(x) \) is approximated by the form

\[ U_N(x) = \sum_{i=0}^{N} a_i B_i(x) \]

Substituting equation (6) in equation (5)

\[ L[U_N] = f(x) + E_N(x) \]

Where

\[ L[U_N] = \sum_{i=0}^{N} \left[ \frac{d}{dx} B_i(x) \right] f(x) \]

\[ L[U_N(x)] = \sum_{i=0}^{N} \left[ \frac{d}{dx} B_i(x) \right] f(x) \]

\[ L[U_N(x)] = \sum_{i=0}^{N} \left[ \frac{d}{dx} B_i(x) \right] f(x) \]

\[ L[U_N(x)] = \sum_{i=0}^{N} \left[ \frac{d}{dx} B_i(x) \right] f(x) \]

This system can be written in a matrix form as follows

\[ U_N(x) = \begin{bmatrix} 1 & x & x^2 & \ldots & x^N \end{bmatrix} \begin{bmatrix} a_{0x} & 0 & \cdots & 0 \\ 0 & a_{1x} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{Nx} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{bmatrix} \]

Using operator forms, equation (3) can be written as \( L[U] = f(x) \)

Using operator forms this equation (1) can be written as \( L[U] = f(x) \)

Where the operator \( L \) is defined for each type of delay integro–differential equation as:

1. Retarded integro-differential equation
   \[ L[u(x)] = \int_{0}^{x} k(x, \tau) u(t-\tau) d\tau \]

2. Neutral integro-differential equation
   \[ L[u(x)] = \frac{du(x)}{dx} - \int_{0}^{x} k(x, \tau) u(t-\tau) d\tau \]

3. Mixed integro-differential equation
   \[ L[u(x)] = \frac{du(x)}{dx} - \int_{0}^{x} k(x, \tau_1) u(t-\tau_1) d\tau - \int_{0}^{x} k(x, \tau_2) u(t-\tau_2) d\tau \]

The unknown function \( U(x) \) is approximated by the form

\[ U_N(x) = \sum_{i=0}^{N} a_i B_i(x) \]

Substituting equation (6) in equation (5)

\[ L[U_N] = f(x) + E_N(x) \]

Where

\[ L[U_N] = \sum_{i=0}^{N} \left[ \frac{d}{dx} B_i(x) \right] f(x) \]

\[ L[U_N(x)] = \sum_{i=0}^{N} \left[ \frac{d}{dx} B_i(x) \right] f(x) \]

\[ L[U_N(x)] = \sum_{i=0}^{N} \left[ \frac{d}{dx} B_i(x) \right] f(x) \]

\[ L[U_N(x)] = \sum_{i=0}^{N} \left[ \frac{d}{dx} B_i(x) \right] f(x) \]

\[ L[U_N(x)] = \sum_{i=0}^{N} \left[ \frac{d}{dx} B_i(x) \right] f(x) \]
Bernstein Polynomials Solving One Dimensional Delay Volterra Integro Differential Equations


\[ L[U_N(x)] = \sum_{i=0}^{N} c_i \left( \frac{dB_i(x-t_1)}{dx} - \int_0^x k(x,t)B_i(t-t_2)dt \right) \]

For which we have the residue equation
\[ E_N(x) = L[U_N(x)] - f(x) \]

Substituting (6) in (7) we get
\[ E_{o(x)} = L[\sum_{i=0}^{N} a_i B_i(x)] - f(x) = \sum_{i=0}^{N} a_i L[B_i(x)] - f(x) \]

Obviously the weighting function setting its weighted integral equal to zero
\[ \int w_j E_N(x) dx = 0 \]

Inserting (8) in (9)
\[ \int w_j \left\{ \sum_{i=0}^{N} a_i L[B_i(x)] - f(x) \right\} = 0 \]

\[ \sum_{i=0}^{N} a_i \int w_j [L[B_i(x)] - f(x)] = \int w_j f(x) \]

Where
\[ L[B_j(x)] = \frac{dB_j(x)}{dx} - \int_0^x k(x,t)B_j(t-t_2)dt \]

Introducing Matrix K and vector H has
\[ K_j = \int w_j L[B_j(x)] dx \]
\[ H_j = \int w_j f(x) \]

Least square Method

The Least square method is one of the approximated methods used to solve delay volterra integro differential equations of the second kind In this method the weighting function is chosen as follow
\[ w_j = L[B_j(x)] \]

This leads to
\[ \int L[B_j(x)]L[B_j(x)] a_j = L[B_j(x)] f(x) \] \( i = 0,1,...,N \)

Hence (12) can be seen as a system of \((N+1)\) equations in the \((N+1)\) unknown \(a_i\), \(i=0,1,...,N\)

\[ A = \begin{bmatrix} \int L[B_0](x)\int L[B_0](x) dx & \int L[B_0](x)\int L[B_1](x) dx & \cdots & \int L[B_0](x)\int L[B_N](x) dx \\ \\ \\ \\ \int L[B_1](x)\int L[B_0](x) dx & \int L[B_1](x)\int L[B_1](x) dx & \cdots & \int L[B_1](x)\int L[B_N](x) dx \\ \\ \\ \\ \cdots & \cdots & \cdots & \cdots \\ \\ \int L[B_N](x)\int L[B_0](x) dx & \int L[B_N](x)\int L[B_1](x) dx & \cdots & \int L[B_N](x)\int L[B_N](x) dx \\ \\ \int L[B_0](x) f(x) dx & \int L[B_1](x) f(x) dx & \cdots & \int L[B_N](x) f(x) dx \end{bmatrix} \]

A computationally efficient way to calculate the value \(a_i\) is by solving the system

\[ KA = H \]

For the coefficint aj's which satisfies (6) the approximated solution of (1) will be given

PDF created with pdfFactory Pro trial version www.pdffactory.com
Algorithm (ABIF)

Step 1 select \( B_j \) Bernstein polynomial

Step 2 computed

\[ L(B_i) \text{ and } L(B_j), \quad i, j = 0, 1, \ldots, N \] (10)

Step 3 computed the Matrix K and H by using (11)

Step 4 solve the system (13) for coefficients \( a_j \)’s

Step 5 substitute \( a_j \)’s in transforming form to obtain the approximated solution of \( U(x) \).

Numerical Examples:

Example 1: Consider the Retarded Volterra integro differential equation of the second kind:

\[ u'(x-1) = \frac{f(x)}{1} + \int_0^x k(x,t)u(t-1)dt \]

where

\[ f(x) = 1 - \frac{x^3}{3} + \frac{x^2}{2} \]

and let the linear kernel be

\( k(x, y) = xy \)

and the exact solution is taken to be

\( u(x) = x \).

For \( h = 0.1 \) and \( x = x_i = a + ih, \quad i = 0, 1, \ldots, 10 \)

The tabulated result is obtained by applying the method involved in this paper i.e the implementation of Bernstein polynomial (ABIF); these numerical results are compared with the exact one in the same table below.

Example 2: Consider the Neutral Volterra integro equation of the second kind

\[ u'(x-1) = \frac{f(x)}{1} + \int_0^x k(x,t)u(t-1/2)dt \]

where \( f(x) = 2x - 2 - \frac{x^5}{4} + \frac{x^4}{3} - \frac{x^3}{8} \) and \( k(x,t) = (xt) \), and the exact solution is taken to be \( u(x) = x^3 \); the step size is taken to be \( h = 0.1 \). The application of Bernstein polynomial (ABIF) yields the results shown in the table below together with the exact solution at each point \( x \). The least square error and the consumed time.

Example 3: Consider the Mixed Volterra integro equation of the second kind

\[ u'(x-1) = \frac{f(x)}{1} + \int_0^x k(x,t)u(t-1/2)dt \]

where \( f(x) = 83422 \)

and \( k(x,t) = (xt) \), and the exact solution is taken to be \( u(x) = x^2 \); the step size is taken to be \( h = 0.1 \). The application of Bernstein polynomial (ABIF) yields the results shown in the table below together with the exact solution at each point \( x \). The least square error and the consumed time.

Conclusion: Bernstein Polynomial is introduced to find the approximate solution of delay Volterra integro differential equation of the second kind. Three numerical examples were submitted to illustrate the given idea with good approximate results were achieved. We conclude that:

1. In general, least square method with aid Bernstein polynomials have been applied to find the solution of linear delay Volterra integro differential equation and have proved their effectiveness from through finding accurate results.

2. A disadvantage of the Bernstein polynomials is its dependence upon a free parameter \( n \) that gives the smallest least square error.
References


Table (1) Presents a comparison the exact solution and approximate solution (Bernstein polynomial)

<table>
<thead>
<tr>
<th>x</th>
<th>Exact sol.</th>
<th>ABIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1000</td>
<td>0.1000</td>
</tr>
<tr>
<td>0.2</td>
<td>0.2000</td>
<td>0.2000</td>
</tr>
<tr>
<td>0.3</td>
<td>0.3000</td>
<td>0.3000</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4000</td>
<td>0.4000</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5000</td>
<td>0.5000</td>
</tr>
<tr>
<td>0.6</td>
<td>0.6000</td>
<td>0.6000</td>
</tr>
<tr>
<td>0.7</td>
<td>0.7000</td>
<td>0.7000</td>
</tr>
<tr>
<td>0.8</td>
<td>0.8000</td>
<td>0.8000</td>
</tr>
<tr>
<td>0.9</td>
<td>0.9000</td>
<td>0.9000</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>L.Sq.E</td>
<td></td>
<td>0.0000</td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td>0.1880sec</td>
</tr>
</tbody>
</table>

Table (2) Presents a comparison the exact solution and approximate solution (Bernstein polynomial)

<table>
<thead>
<tr>
<th>x</th>
<th>Exact sol.</th>
<th>ABIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1000</td>
<td>0.1000</td>
</tr>
<tr>
<td>0.2</td>
<td>0.2000</td>
<td>0.2000</td>
</tr>
<tr>
<td>0.3</td>
<td>0.3000</td>
<td>0.3000</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4000</td>
<td>0.4000</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5000</td>
<td>0.5000</td>
</tr>
<tr>
<td>0.6</td>
<td>0.6000</td>
<td>0.6000</td>
</tr>
<tr>
<td>0.7</td>
<td>0.7000</td>
<td>0.7000</td>
</tr>
<tr>
<td>0.8</td>
<td>0.8000</td>
<td>0.8000</td>
</tr>
<tr>
<td>0.9</td>
<td>0.9000</td>
<td>0.9000</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>L.Sq.E</td>
<td></td>
<td>0.0000</td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td>0.1880sec</td>
</tr>
</tbody>
</table>
Table (3)

Presents a comparison the exact solution and approximate solution (Bernstein polynomial)

<table>
<thead>
<tr>
<th>x</th>
<th>Exact sol.</th>
<th>ABIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.1</td>
<td>0.01000</td>
<td>0.0100</td>
</tr>
<tr>
<td>0.2</td>
<td>0.04000</td>
<td>0.0400</td>
</tr>
<tr>
<td>0.3</td>
<td>0.09000</td>
<td>0.0900</td>
</tr>
<tr>
<td>0.4</td>
<td>0.1600</td>
<td>0.1600</td>
</tr>
<tr>
<td>0.5</td>
<td>0.2500</td>
<td>0.2500</td>
</tr>
<tr>
<td>0.6</td>
<td>0.3600</td>
<td>0.3600</td>
</tr>
<tr>
<td>0.7</td>
<td>0.4900</td>
<td>0.4900</td>
</tr>
<tr>
<td>0.8</td>
<td>0.6400</td>
<td>0.6400</td>
</tr>
<tr>
<td>0.9</td>
<td>0.8100</td>
<td>0.8100</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>L Sq. E</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>0.1880sec</td>
<td></td>
</tr>
</tbody>
</table>