Journal of Kirkuk University – Scientific Studies , vol.3, No.1,2008

Approximate Solution of the System of Nonlinear Fredholm Integral Equations of the Second Kind Using Spline Function

Rostam K. Saeed ***and*** ***Kawa M. Aziz***
College of Science - University of Salahaddin-Erbil

Abstract

This study deals with introducing spline function to find the approximate solution of the system of nonlinear Fredholm integral equations of the second kind. The benefit of spline functions was demonstrated by presenting several examples.

Introduction

In this paper we consider the numerical solution of the systems of nonlinear Fredholm integral equation of the second kind (SNFIES),

$$u_i(x) = f_i(x) + \int_a^b k_i(x,t,u_1(t),u_2(t),...,u_m(t))\,dt, \text{ for } i=1, 2,\dots, m, \qquad \dots (1)$$

Where $u_i(x)$, i=1, 2, …, m are the unknown functions which are to be determined in the interval [a, b], the functions $f_i(x)$ and the kernels $k_i(x,t,u_1(t),u_2(t),...,u_m(t))$ are given, by using spline function. Suppose that the system (1) has a unique solution. However, the necessary and sufficient conditions for existence and uniqueness of the solution of system (1) could be found in **(**Delves & Muhamed, 1985), **(**Jumaa, 2005).Spline function is a function that consists of polynomial pieces joined together with certain smoothness condition, which is used for data interpolation and can be used for finding derivatives and integrals of function. (Cheney & Kinciad, 2004**)** Splines can be any degree: linear splines are simply straight line segments connecting two points. Linear splines yield first-order approximation polynomials. The slopes (i.e., first derivatives) and curvature (i.e., second derivatives) are discontinuous at every data point. Quadratic splines yield second-order approximating polynomials. The slopes of the quadratic splines can be forced to be continuous at each data point, but the curvatures are still discontinuous. Cubic splines yield a third–degree polynomial connecting each pair of data points. The slopes and curvatures of the cubic splines can be forced to be continuous at each data point. (Hoffman, 2001) The name Spline comes from the thin flexible rod, called a spline, use by draftsmen to draw smooth curves through a series of discrete points. The spline is placed over the points and either weighted or pinned at each point. Due to the flexure properties of a flexible rod (typically of rectangular cross section), the slope and curvature of the rod are continuous at each point. A smooth curve is then traced along the rod, yielding a spline curve. (Ahlberg, et. al, 1967)Some authors and researchers used spline functions to find the approximate solution of the integral equations, such as (Deputat, et. al, 2005;Saveljeva, 2006 and Hoffman, 2001). Also, (Mustafa, 2004) and (Jumaa, 2005) used spline functions to solve a system of linear and nonlinear Volterra integral equations of the second kind respectively. In this paper, linear, quadratic and cubic spline functions are used for solving the system of non-linear Fredholm integral equation of the second kind.

Spline Functions(Cheney&Kinciad,2004;Mathews&Fink, 1999)

A function $S(x)$ is called a spline function of degree k if:

1- The domain of S is an interval [a, b],

2- $S, S', S'', ..., S^{(k-1)}$ are all continuous functions on [a, b],

3- There is a partitioning of the interval a=$x_0 < x_1 < x_2 < \ldots < x_n$=b, such that $S(x)$ is a polynomial of degree at most k on each subinterval $[x_i, x_{i+1}]$, i=0, 1,…, n-1.Such a function is somewhat complicated to define in explicit terms, we are forced to write

$$S(x)=\begin{cases} S_0(x) & x\in[x_0,x_1] \\ S_1(x) & x\in[x_1,x_2] \\ \vdots & \\ S_{n-1}(x) & x\in[x_{n-1},x_n] \end{cases},$$

Where $S_i(x)$ is a polynomial of degree at most k on each subinterval $[x_i, x_{i+1}]$ for i=1, 2,…, n-1. In this section three types of spline functions have been considered, they are:

1-First-Degree Spline Function (LSF):

A simple example for the spline functions is the spline of degree one, whose pieces are linear polynomials joined together to achieve continuity. The linear spline function $L(x)$ consists of n functions which are a linear polynomial on each subinterval, $[x_i, x_{i+1}]$.

$$L(x)=\begin{cases} L_0(x) & x\in[x_0,x_1] \\ L_1(x) & x\in[x_1,x_2] \\ \vdots & \\ L_{n-1}(x) & x\in[x_{n-1},x_n] \end{cases} \quad \ldots(2)$$

Where $L_i(x) = a_i\, x + b_i$ for i=0, 1,…, n-1.

In the subinterval $[x_k, x_{k+1}]$ the Lagrange interpolation polynomial is used to represent this piecewise linear curve **(**Cheney & Kinciad, 2004**)**:

$$L_k(x) = y_k \frac{x - x_{k+1}}{x_k - x_{k+1}} + y_{k+1} \frac{x - x_k}{x_{k+1} - x_k} \text{ for } x_k \le x \le x_{k+1}, \qquad \ldots(3)$$

2-Quadratic Spline Function (QSF):

A quadratic spline function is continuously differentiable piecewise quadratic function which consists of n separated polynomial over each subinterval $[x_k, x_{k+1}]$ i.e.

$$Q(x) = \begin{cases} Q_0(x) & x \in [x_0, x_1] \\ Q_1(x) & x \in [x_1, x_2] \\ \vdots & \\ Q_{n-1}(x) & x \in [x_{n-1}, x_n] \end{cases} \qquad \ldots(4)$$

$Q(x)$ is a piecewise quadratic function and continuously differentiable on the entire interval $[x_0, x_n]$ and which interpolates the table; that is $Q(x_i) = y_i$ for $0 \le i \le n$.We can write

$$Q_i(x) = \frac{Q_i'(x_{i+1}) - Q_i'(x_i)}{2(x_{i+1} - x_i)}(x - x_i)^2 + Q_i'(x_i)(x - x_i) + Q_i(x_i), \qquad \ldots(5)$$

Where

$$Q_i'(x_{i+1}) = -Q_i'(x_i) + 2\left(\frac{Q_i(x_{i+1}) - Q_i(x_i)}{x_{i+1} - x_i}\right), \qquad 0 \le i \le n-1 \qquad \ldots(6)$$

This equation can be used to obtain the values of $Q_i'(x_1), Q_i'(x_2), \ldots, Q_i'(x_n)$, starting with an arbitrary value for $Q_i'(x_0)$.

3-Cubic Spline Function (CSF):

Suppose that (x_k, y_k), k=0, 1,…, n are n+1 distinct points, where $a = x_0 < x_1 < \ldots x_n = b$. The function $S(x)$ is called a cubic spline if there exist n cubic polynomials $S_k(x)$ with coefficients a_k, b_k, c_k and d_k that satisfy the following properties:

I. $S(x) = S_k(x) = a_k\,(x - x_k)^3 + b_k\,(x - x_k)^2 + c_k\,(x - x_k) + d_k$

for $x \in [x_k, x_{k+1}]$ and k=0, 1, …, n-1. …(7)

II. $S(x_k) = y_k$ for k=0, 1, …, n. …(8)

III. $S_k(x_{k+1}) = S_{k+1}(x_{k+1})$ for k=0, 1, …, n-2. …(9)

IV. $S_k'(x_{k+1}) = S_{k+1}'(x_{k+1})$ for k=0, 1, …, n-2. …(10)

V. $S_k''(x_{k+1}) = S_{k+1}''(x_{k+1})$ for k=0, 1, …, n-2. …(11)

Property I states that $S(x)$ consists of n piecewise cubic polynomials. Property II states that the piecewise cubics interpolate the given set of data points. Property III and IV require that the piecewise cubics represent a smooth continuous function. Property V states that the second derivative of the resulting function is also continuous. There are $4n$ unknowns which are the a_k, b_k, c_k and d_k for k=0,1,…,n-1.

By using the above properties we get

$$y_k = d_k \qquad \text{… (12)}$$

Suppose $M_k = S''(x_k)$, k=0, 1, …, n-1.

Hence we write

$$b_k = \frac{M_k}{2}, \qquad \text{… (13)}$$

$$a_k = \frac{M_{k+1} - M_k}{6h_k}. \qquad \text{…(14)}$$

and

$$c_k = \frac{y_{k+1} - y_k}{h_k} - \frac{2h_k M_k + h_k M_{k+1}}{6}. \qquad \text{…(15)}$$

We get

$$h_{k-1} M_{k-1} + 2(h_{k-1} + h_k) M_k + h_k M_{k+1} = 6\left(\frac{y_{k+1} - y_k}{h_k} - \frac{y_k - y_{k-1}}{h_{k-1}}\right), \qquad \text{…(16)}$$

for k=1, 2, …, n-1.

Take $M_0 = 0$ and $M_n = 0$. This makes the end cubics approach linearity at extremities. This condition, called a natural spline, matches precisely to the drafting device. This technique is used very frequently.

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & 0 & 0 & \cdots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \cdots & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & 0 & 0 \\ 0 & 0 & & & & \vdots \\ \vdots & \vdots & & & & 0 \\ 0 & 0 & \cdots & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ \\ \vdots \\ M_{n-1} \end{pmatrix} = 6 \begin{pmatrix} \frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0} \\ \frac{y_3-y_2}{h_2} - \frac{y_2-y_1}{h_1} \\ \frac{y_4-y_3}{h_3} - \frac{y_3-y_2}{h_2} \\ \vdots \\ \\ \frac{y_n-y_{n-1}}{h_{n-1}} - \frac{y_{n-1}-y_{n-2}}{h_{n-2}} \end{pmatrix} \qquad \text{…(17)}$$

Solutions of a SNFIES Using Spline Functions

In this section, we use spline functions: linear, quadratic and natural cubic to find numerical solution of a **SNFIES.**

To apply the spline functions divide the intervals [*a*, *b*] into *n* subintervals as follows: $a = x_0 < x_1 < ... < x_n = b$; $x_k = x_{k-1} + h$, *k*=1, 2,…, *n* where $h = \frac{b-a}{n}$. Since $k_i(x,t,u_1(t),u_2(t),...,u_m(t))$ are continuous in the interval $[x_0, x_n]$ we can write equation (1) as follows:

$$u_i(x) = f_i(x) + \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} k_i(x,t,u_1(t),u_2(t),...,u_m(t))\,dt \text{; for } i=1, 2, \ldots, m \quad \ldots(18)$$

Note:

In this paper, evaluate the integrals directly if it's possible or numerically by using composite trapezoid method.

1- Using Linear Spline Function (LSF):

To solve equations (1), suppose in the interval $[x_k, x_{k+1}]$

$$u_i(x) = L_{ik}\frac{x - x_{k+1}}{x_k - x_{k+1}} + L_{i,k+1}\frac{x - x_k}{x_{k+1} - x_k}, \quad \ldots(19)$$

where $L_{ik} = u_i(x_k)$ and $L_{i,k+1} = u_i(x_{k+1})$; for *i*=1, 2, …, *m* ; *k*=0,1, …, *n*-1.
Substituting equation (19) into equation (18) we get:

$$u_i(x) = f_i(x) + \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} k_i(x,t,L_{1k}\frac{x_{k+1}-t}{h} + L_{1,k+1}\frac{t-x_k}{h}, L_{2k}\frac{x_{k+1}-t}{h} + L_{2,k+1}\frac{t-x_k}{h},...,$$

$$L_{mk}\frac{x_{k+1}-t}{h} + L_{m,k+1}\frac{t-x_k}{h})dt; \text{ for } i=1, 2, \ldots, m. \quad \ldots(20)$$

In equation (20) replace *x* by $x_0, x_1, ..., x_n$ for each *i*; we get V equations in the V unknowns where V=*m*(*n*+1). Solve the resulting nonlinear system by using modified Newton-Raphson method to find the unknowns. Substituted the values of $L_{10}, L_{11}, ... L_{1n}, L_{20}, L_{21}, ..., L_{2n}, ..., L_{m0}, L_{m1}, ..., L_{mn}$ in the equation (19), we get the approximation solution of (1).

The Algorithm LSF:

Step (1): Input *n* (number of subintervals). Set $h = \frac{b-a}{n}$, $x_k = x_{k-1} + h$; for *k*=1, 2, ..., *n*, and divide the interval [*a*, *b*] into *n* subintervals $a = x_0 < x_1 < ... < x_n = b$.

Step (2): In (20) put $x = x_k$ we get

$$L_{ik} = f_i(x_k) + \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} k_i(x_k, t, L_{1k}\frac{x_{k+1}-t}{h} + L_{1,k+1}\frac{t-x_k}{h}, L_{2k}\frac{x_{k+1}-t}{h}$$

$$+ L_{2,k+1}\frac{t-x_k}{h}, ..., L_{mk}\frac{x_{k+1}-t}{h} + L_{m,k+1}\frac{t-x_k}{h})dt, \text{ for } i=1, 2, \ldots, m.$$

Step(3): Solve a nonlinear system in step (2) to find the unknowns $L_{10}, L_{11},...L_{1n}, L_{20}, L_{21},..., L_{2n},..., L_{m0}, L_{m1},..., L_{mn}$ by using modified Newton-Raphson method, and put the resulting values in equation (19) to get the approximation solution of (1).

2- Using Quadratic Spline Functions (QSF):

In each subintervals $[x_k, x_{k+1}]$, suppose that

$$u_i(x) = \frac{Q_i'(x_{k+1}) - Q_i'(x_k)}{2h}(x - x_k)^2 + Q_i'(x_k)(x - x_k) + Q_i(x_k), \qquad \ldots(21)$$

for i=1, 2, …, m; where $Q_i(x_k) = u_i(x_k)$, k=0, 1 , …, n-1.
Substituting equation (21) into equation (18) we get

$$\begin{aligned} u_i(x) = f_i(x) + \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} k_i\Big(x,t,& \frac{Q_1'(x_{k+1}) - Q_1'(x_k)}{2h}(t - x_k)^2 + Q_1'(x_k)(t - x_k) + Q_1(x_k), \\ & \frac{Q_2'(x_{k+1}) - Q_2'(x_k)}{2h}(t - x_k)^2 + Q_2'(x_k)(t - x_k) + Q_2(x_k)+,..., \\ & \frac{Q_m'(x_{k+1}) - Q_m'(x_k)}{2h}(t - x_k)^2 + Q_m'(x_k)(t - x_k) + Q_m(x_k)\Big)dt; \qquad \ldots(22) \end{aligned}$$

for i=1, 2, …,m.

Where $Q_i'(x_k)$, for i=1, 2,…, m and k=1, 2, …, n is given by the equation (6). Replacing x by $x_0, x_1,..., x_n$ in equation (22) respectively for each i=1, 2, …, m we get a nonlinear system of V equations in the V unknowns $Q_1(x_0)$, $Q_1(x_1),\ldots,Q_1(x_n), Q_2(x_0), Q_2(x_1),\ldots,Q_2(x_n),\ldots,Q_m(x_0), Q_m(x_1), Q_m(x_n)$.

Solve the resulting nonlinear system by using modified Newton-Raphson method to get the approximation solution of equation (1).

Note:

In this subsection, for QSF we choose

$$Q_i'(x_0) = \begin{cases} f_i'(x_0) & \text{if } \left.\dfrac{\partial k_i(x,t,u_1(t),u_2(t),...,u_m(t))}{\partial x}\right|_{x=x_o} = 0 \\ 0 & \text{otherwise} \end{cases} \qquad \ldots(23)$$

for i=1, 2, …, m.

The Algorithm QSF:

Step (1): Input n (number of subintervals). Set $h = \dfrac{b-a}{n}$; and $x_k = x_{k-1} + h$, for k=1, 2, ..., n, and divide the interval [a, b] into n subintervals $a = x_0 < x_1 < ... < x_n = b$.

Step (2): In the equation (6) compute $Q_i'(x_{k+1})$,for *k*=0, 1,.., *n*-1; *i*=1, 2, .., *m*.

Step (3): In equation (22) replacing *x* by x_k for *k*=0, 1, …, *n*-1, we get a nonlinear system of V equations in the V unknowns $Q_1(x_0)$, $Q_1(x_1)$, $\ldots, Q_1(x_n)$, $Q_2(x_0)$, $Q_2(x_1), \ldots, Q_2(x_n), \ldots$, $Q_m(x_0)$, $Q_m(x_1)$, …, $Q_m(x_n)$.

Step (4): Solve the resulting nonlinear system in step (3) by using modified Newton-Raphson method, then substitute the results in the equation (21) to get the approximation solution of the equation (1).

3- Using Cubic Spline Functions (CSF):

In each subintervals [x_k, x_{k+1}], suppose

$$u_i(x) = a_{ik}(x - x_k)^3 + b_{ik}(x - x_k)^2 + c_{ik}(x - x_k) + S_{ik} \qquad \ldots(24)$$

where

$$a_{ik} = \frac{M_{i,k+1} - M_{ik}}{6h},$$

$$b_{ik} = \frac{M_{ik}}{2}$$

$$c_{ik} = \frac{S_{i,k+1} - S_{ik}}{h} - \frac{2hM_{ik} + hM_{i,k+1}}{6},$$

and

$S_{ik} = S_i(x_k) = u_i(x_k)$ for *k*=0, 1, …, *n*; *i*=1, 2, …,*m*.

For finding the values of $M_{i0}, M_{i1}, \ldots, M_{1n}$, rewrite the system (17) as follows

$$\begin{pmatrix} 4h & h & 0 & 0 & \cdots & 0 \\ h & 4h & h & 0 & \cdots & 0 \\ 0 & h & 4h & h & 0 & \vdots \\ 0 & & & & & 0 \\ \vdots & & & & & 0 \\ 0 & 0 & \cdots & 0 & h & 4h \end{pmatrix} \begin{pmatrix} M_{i1} \\ M_{i2} \\ M_{i3} \\ \\ \vdots \\ M_{i,n-1} \end{pmatrix} = \frac{6}{h} \begin{pmatrix} S_{i2} - 2S_{i1} + S_{i0} \\ S_{i3} - 2S_{i2} + S_{i1} \\ S_{i4} - 2S_{i3} + S_{i2} \\ \\ \vdots \\ S_{in} - 2S_{i,n-1} + S_{i,n-2} \end{pmatrix}, \qquad \ldots(25)$$

where $M_{i0} = M_{in} = 0$ and $h = h_0 = h_1 = \ldots = h_n$, then solve the system (25) .

Substituting (24) into the equation (18) we get:

$$u_i(x) = f_i(x) + \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} k_i(x, t, a_{1k}(t - x_k)^3 + b_{1k}(t - x_k)^2 + c_{1k}(x - x_k) + S_{1k},$$
$$a_{2k}(t - x_k)^3 + b_{2k}(t - x_k)^2 + c_{2k}(t - x_k) + S_{2k} + ,\ldots, a_{mk}(t - x_k)^3 + b_{mk}(t - x_k)^2 +$$
$$c_{mk}(t - x_k) + S_{mk})\,dt; \text{ for } i=1, 2, \ldots, m \qquad \ldots(26)$$

In equation (26), put $x = x_k$, for *k*=0, 1, …, *n* to get:

$$S_{ik} = f_i(x_k) + \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} k_i(x_k, t, a_{1k}(t-x_k)^3 + b_{1k}(t-x_k)^2 + c_{1k}(x-x_k) + S_{1k},$$
$$a_{2,k}(t-x_k)^3 + b_{2,k}(t-x_k)^2 + c_{2,k}(t-x_k) + S_{2,k} + ,...,$$
$$a_{mk}(t-x_k)^3 + b_{mk}(t-x_k)^2 + c_{mk}(t-x_k) + S_{mk})\,dt. \qquad \ldots(27)$$

Solve a nonlinear system (27) of V equations in the V unknowns $S_{10}, S_{11}, ..., S_{1n}, S_{20}, S_{21}, ..., S_{2n}, ..., S_{m0}, S_{m1}, ..., S_{mn}$ by using modified Newton-Raphson method. Substitute the values in the equation (24), to get the approximation solution of (1).

The Algorithm CSF:

Step(1): Input *n* (number of subintervals). Set $h = \dfrac{b-a}{n}$; $x_k = x_{k-1} + h$, *k*=1, 2,…, *n*, and divide the interval [*a, b*] into *n* subintervals $a = x_0 < x_1 < ... < x_n = b$.

Step(2): Put $M_{i1} = M_{in} = 0$; for *i*=1, 2, …, *m*.

Step(3): Solve a linear system (25) to find the values of M_{i1}, M_{i2}, $..., M_{i,\,n-1}$ with respect to $S_{i0}, S_{i1}, ..., S_{in}$; for *i*=1, 2,…, *m*.

Step(4): In the equation (26), put $x = x_k$ for *k*=0,1,…,*n* to get the equation (27).

Step(5): Solve the resulting nonlinear system in step (4) by using modified Newton-Raphson method, and then substitute the values of S_{10}, $S_{11}, ..., S_{1n}, S_{20}, S_{21}, ..., S_{2n}, ..., S_{m0}, S_{m1}, ..., S_{mn}$ in the equation (24) to get an approximation solution of (1).

Illustrative Examples

In this section, three examples presented for conforming the methods. In each example, we calculate the least-square error and the running times of the program for each method.

Note:

1. In modified Newton-Raphson method we choose tolerance (ε) $=10^{-7}$.

2. We choose the length of each subintervals *h*=0.1, i.e. *n*=10.

3. In **QSF,** use the equation (23) to choose the initial value $Q_i'(x_0)$.

Example 1: (**Babolian et. al, 2004**)

Consider the following SNFIES:

$$u_1(x) = x - \frac{5}{18} + \int_0^1 \frac{1}{3}(u_1(t) + u_2(t))dt$$

$$u_2(x) = x^2 - \frac{2}{9} + \int_0^1 \frac{1}{3}(u_1^2(t) + u_2(t))dt,$$

with the exact solutions $u_1(x) = x$ and $u_2(x) = x^2$

Solution: When applying the algorithms LSF, QSF and CSF, we get the following results:

Table (1) shows a comparison between the exact solutions *x* and the numerical solution $u_1(x)$, of Example 1.

x	Exact solution	Approximate solution by		
		LSF	QSF	CSF
0.05	0.05	0.051667151	0.049999996	0.050096223
0.15	0.15	0.151667590	0.149999997	0.150096224
0.25	0.25	0.251667590	0.249999997	0.250096224
0.35	0.35	0.351667590	0.349999997	0.350096224
0.45	0.45	0.451667590	0.449999997	0.450096224
0.55	0.55	0.551667591	0.549999998	0.550096225
0.65	0.65	0.651667591	0.649999998	0.650096225
0.75	0.75	0.751667591	0.749999998	0.750096225
0.85	0.85	0.851667591	0.849999998	0.850096225
0.95	0.95	0.951667591	0.949999998	0.950096225
L.S.E		2.7808e-005	2.0278e-018	9.2592e-008
R.T		00:00:17	00:00:34	00:00:20

Table (2): shows a comparison between the exact solutions x^2 and the numerical solution $u_2(x)$ of Example 1.

x	Exact solution	Approximate solution by		
		LSF	QSF	CSF
0.05	0.0025	0.006668105	0.002499997	0.003511282
0.15	0.0225	0.026668517	0.022499997	0.022351061
0.25	0.0625	0.066668517	0.062499997	0.062661835
0.35	0.1225	0.126668517	0.122499997	0.122578962
0.45	0.2025	0.206668517	0.202499997	0.202599681
0.55	0.3025	0.306668518	0.302499998	0.302599681
0.65	0.4356	0.426668518	0.422499997	0.422578963
0.75	0.5625	0.566668518	0.562499998	0.562661836
0.85	0.7225	0.726668518	0.722499998	0.722351063
0.95	0.9025	0.906668518	0.902499999	0.903511284
L.S.E		1.7376e-004	7.9943e-019	2.1744e-006
R.T		00:00:17	00:00:34	00:00:20

Example 2:

Consider the **SNFIES**:

$$u_1(x) = \sin(x) - 0.5e + \frac{7}{8} + \int_0^1 0.5(\frac{t^2}{t+e^t})u_2^2(t)\ dt$$

$$u_2(x) = x + e^x + \cos(1) - 1 + \int_0^1 u_1(t)\ dt\,,$$

with the exact solution $u_1(x) = \sin(x)$ and $u_2(x) = x + e^x$.

Solution:

When applying the algorithms LSF, QSF and CSF, we get the following results:

Table (3): shows a comparison between the exact solution $\sin(x)$ and the numerical solution $u_1(x)$ of Example 2.

x	Exact solution	Approximate solution by		
		LSF	QSF	CSF
0.05	0.049979169	0.056713619	0.056755265	0.056776071
0.15	0.149438132	0.156048285	0.156256015	0.156234994
0.25	0.247403959	0.253891680	0.254180210	0.254200844
0.35	0.342897807	0.349266186	0.349715841	0.349694487
0.45	0.434965534	0.441218852	0.441741931	0.441762864
0.55	0.522687228	0.528830917	0.529505403	0.529482018
0.65	0.605186405	0.611226992	0.611962936	0.611990572
0.75	0.681638760	0.687583801	0.688457059	0.688407829
0.85	0.751280405	0.757138412	0.758057050	0.758180367
0.95	0.813415504	0.819195859	0.820233907	0.819826884
L.S.E		3.8810e-004	4.6490e-005	4.5799e-004
R.T		00:00:13	00:00:44	00:00:17

Table (4): shows a comparison between the exact solutions x^2 and the numerical solution $u_2(x)$ of Example 2.

x	Exact solution	Approximate solution by		
		LSF	QSF	CSF
0.05	1.101271096	1.108999223	1.107706493	1.108141727
0.15	1.311834242	1.319700602	1.318228101	1.318125246
0.25	1.534025416	1.542044547	1.540460996	1.540471546
0.35	1.769067548	1.777255517	1.775461609	1.775472610
0.45	2.018312185	2.026686749	2.024747987	2.024726183
0.55	2.283253017	2.291833801	2.289647324	2.289672107
0.65	2.565540829	2.574349520	2.571976902	2.571930360
0.75	2.867000016	2.876060584	2.873394623	2.873502406
0.85	3.189646851	3.198985786	3.196083257	3.195727052
0.95	3.535709659	3.545356236	3.542104631	3.543365414
L.S.E		7.3667e-004	4.0895e-005	4.3039e-004
R.T		00:00:13	00:00:44	00:00:17

Example 3:

Consider the following system

$$u_1(x) = \frac{x}{2} - \frac{1}{4} + \int_0^1 [x\,u_1(t) + t\,u_2(t)]\;dt$$

$$u_2(x) = x^2 - \frac{2}{9} + \int_0^1 \frac{1}{3}(u_3^2(t) + u_2(t))\;dt$$

$$u_3(x) = x - \frac{2}{15} + \int_0^1 \frac{1}{5}(u_1(t)u_3(t) + u_2(t))\;dt\,,$$

with the exact solution $u_1(x) = x$, $u_2(x) = x^2$ and $u_3(x) = x$.

Solution:

When applying the algorithms LSF, QSF and CSF, we get the following results:

Table (5): shows a comparison between the exact solution x and the numerical solution $u_1(x)$ of Example 3.

x	Exact solution	Approximate solution by		
		LSF	QSF	CSF
0.05	0.05	0.051669159	0.025059012	0.050096332
0.15	0.15	0.151972642	0.175075872	0.150113847
0.25	0.25	0.252276126	0.225081492	0.250131362
0.35	0.35	0.352579609	0.375098353	0.350148876
0.45	0.45	0.452883093	0.425103973	0.450166391
0.55	0.55	0.553186576	0.575120833	0.550183906
0.65	0.65	0.653490060	0.625126454	0.650201421
0.75	0.75	0.753793544	0.775143315	0.750218936
0.85	0.85	0.854097027	0.825148935	0.850236452
0.95	0.95	0.954400511	0.975165796	0.950253967
L.S.E		4.9159e-005	0.006254352	3.3208e-007
R.T		00:00:18	00:00:23	00:01:03

Table (6): shows a comparison between the exact solution x^2 and the numerical solution $u_2(x)$ of Example 3.

x	Exact solution	Approximate solution by		
		LSF	QSF	CSF
0.05	0.0025	0.006368169	0.002612405	0.003493982
0.15	0.0225	0.026368169	0.022612405	0.022333761
0.25	0.0625	0.066368170	0.062612405	0.062644535
0.35	0.1225	0.126368170	0.122612405	0.122561662
0.45	0.2025	0.206368170	0.202612405	0.202582380
0.55	0.3025	0.306368170	0.302612405	0.302582381
0.65	0.4356	0.426368170	0.422612405	0.422561662
0.75	0.5625	0.566368170	0.562612405	0.562644536
0.85	0.7225	0.726368170	0.722612405	0.722333762
0.95	0.9025	0.906368170	0.902612405	0.903493983
L.S.E		1.0473e-004	1.2634e-007	2.0942e-006
R.T		00:00:18	00:00:23	00:01:03

Table (7) shows a comparison between the exact solution x and the numerical solution $u_3(x)$ of Example 3.

x	Exact solution	Approximate solution by		
		LSF	QSF	CSF
0.05	0.05	0.051068532	0.050224760	0.050061628
0.15	0.15	0.151068532	0.150224760	0.150061628
0.25	0.25	0.251068532	0.250224760	0.250061628
0.35	0.35	0.351068532	0.350224761	0.350061628
0.45	0.45	0.451068532	0.450224761	0.450061629
0.55	0.55	0.551068532	0.550224761	0.550061629
0.65	0.65	0.651068533	0.650224761	0.650061629
0.75	0.75	0.751068533	0.750224761	0.750061629
0.85	0.85	0.851068533	0.850224761	0.850061629
0.95	0.95	0.951068533	0.950224761	0.950061629
L.S.E		7.9923e-006	5.0517e-007	3.7981e-008
R.T		00:00:18	00:00:23	00:01:03

Table (8): shows a comparison between the solutions of Example 1 by using **QSF** with choosing two different initial values as follows:
1. $Q_i'(0)= f_i'(0)$, i=1, 2. 2. $Q_i'(0)=0$, i=1, 2.

X	Exact solution of $u_1(x)$	Approximate Solution of $u_1(x)$ where		Exact Solution of $u_2(x)$	Approximate Solution of $u_2(x)$ where	
		$Q_1'(0)=1$	$Q_i'(0)=0$		$Q_i'(0)=0$	$Q_i'(0)=0$
0.05	0.05	0.049999996	0.025666810	0.0025	0.002499997	0.003833625
0.15	0.15	0.149999997	0.175666810	0.0225	0.022499997	0.023833626
0.25	0.25	0.249999997	0.225666811	0.0625	0.062499997	0.063833626
0.35	0.35	0.349999997	0.375666811	0.1225	0.122499997	0.123833626
0.45	0.45	0.449999997	0.425666811	0.2025	0.202499997	0.203833626
0.55	0.55	0.549999998	0.575666811	0.3025	0.302499998	0.303833627
0.65	0.65	0.649999998	0.625666812	0.4356	0.422499997	0.423833626
0.75	0.75	0.749999998	0.775666812	0.5625	0.562499998	0.563833627
0.85	0.85	0.849999998	0.825666812	0.7225	0.722499998	0.723833627
0.95	0.95	0.949999998	0.975666812	0.9025	0.902499999	0.903833628
L.S.E		2.0278e-018	6.5878e-004		7.9943e-019	1.7785e-006

Table (9): shows a comparison between the solutions of Example 2 by using QSF with choosing two different initial values as follows:
1. $Q_i'(0)= f_i'(0)$, i=1, 2. 2. $Q_i'(0)=0$, i=1, 2.

X	Exact Solution of $u_1(x)$	Approximate Solution of $u_1(x)$ where		Exact Solution of $u_2(x)$	Approximate Solution of $u_2(x)$ where	
		$Q_1'(0)=1$	$Q_i'(0)=0$		$Q_i'(0)=0$	$Q_i'(0)=0$
0.05	0.049979169	0.056755265	0.031755265	1.101271096	1.107706493	1.057706493
0.15	0.149438132	0.156256015	0.181256015	1.311834242	1.318228101	1.368228101
0.25	0.247403959	0.254180210	0.229180210	1.534025416	1.540460996	1.490460996
0.35	0.342897807	0.349715841	0.374715841	1.769067548	1.775461609	1.825461609
0.45	0.434965534	0.441741931	0.416741931	2.018312185	2.024747987	1.974747987
0.55	0.522687228	0.529505403	0.554505403	2.283253017	2.289647324	2.339647324
0.65	0.605186405	0.611962936	0.586962936	2.565540829	2.571976902	2.521976902
0.75	0.681638760	0.688457059	0.713457059	2.867000016	2.873394623	2.923394623
0.85	0.751280405	0.758057050	0.733057050	3.189646851	3.196083257	3.146083257
0.95	0.813415504	0.820233907	0.845233907	3.535709659	3.542104631	3.592104631
L.S.E		4.6490e-005	0.001012410		4.0895e-005	0.003180392

Conclusion

The scarcity of studies on the system of nonlinear integral equations, led to proposing a methods to obtain approximate solution of the system of nonlinear Fredholm integral equations of the second kind. In these methods we divided the intervals into *n* subintervals, in each subinterval the approximate solution is a polynomial of degree less than or equal to three. It is noticed that when *n* is increased, the solution has improved. Moreover, choosing initial solution $Q_i'(x_0)$ in equation (23) instead of an arbitrary value of $Q_i'(x_0)$ gives better solution.

References

- Ahlberg.J.H. Nilson.E. N.and WALSH.J.L., (1967): The theory of Splines and Their Applications, Academic Press,New York and London.
- Babolian, E., Biazar, J. and Vahidi, A. R.,(2004): The decomposition method applied to systems of Fredholm integral equations of the second kind, Applied Mathematics and Computation, Vol. 148, pp.443-452.
- Cheney, W. and Kincaid, D., (2004): Numerical Mathematics and Computing, Fifth Edition, Brooks /Cole Publishing company.
- Delves, L. M. and Mohamed, J. L., (1985): Computational Methods for Integral Equations, Cambridge University Press.
- Deputat, V., Oja, P. and Saveljeva, D., (2005): Quadratic Spline Sub domain Method for Volterra Integral Equation, Mathematical Modeling and Analysis, Vol. 10, pp. 335-344.
- Hoffman, J. D., (2001): Numerical Methods for Engineers and Scientists, second Edition, Marcel Dekker, Inc,.
- Jumaa, B. F., (2005): On Approximate Solutions to a system of Non-linear Volterra Integral Equations, PhD. thesis, University of Technology, School of Applied Sciences,.
- Mathews, J. H. and FINK, K. D., (1999): Numerical Methods Using MATLAB, Third Edition, a Viacom Company.
- Mustafa, M. M.,(2004): Numerical Algorithms for Treating System of Volterra Integral Equations Using Spline Functions, PhD. Thesis University Al-Mustansiriya, college of Science,.
- Saveljeva, D., (2006): Quadratic and Cubic Spline Volterra Integral Equations, PhD. Thesis, University of TARTU, Estonia.

الحَلّ التقريبي لنظامِ معادلاتِ فريدهولم اللاخطّية التكاملية من النوعِ الثانيِ باستعمال دالة سبلاين

روستم كريم سعيد و كاوه مصطفى عزيز
كلية العلوم ــ جامعة صلاح الدين/أربيل

الخلاصة

تم في هذه الدراسة استخدام دالة سبلاين لإيجاد الحَلِّ التقريبيِ لنظامِ معادلاتِ فريدهولم اللاخطّية التكاملية من النوعِ الثانيِ. وتم استخدام عدة أمثلة لبيان أهمية دالة سبلاين.