NUMERICAL SOLUTION FOR VOLTERRA-FREDHOLM INTEGRAL EQUATION OF THE SECOND KIND BY USING LEAST SQUARES TECHNIQUE

Shazad Shawki Ahmed
Department of Mathematics, College of Science, University of Sulaimani, Sulaimani -Iraq.

Abstract

In this paper we investigate the numerical solution of an important class of mixed linear integral equations, called Volterra-Fredholm integral equations which are used in technology, mechanics and mathematical physics.

The basic concepts are: First, approximates the unknown function by a tensor product (Algebraic or Chebyshev)-surface and substituting it in the Volterra-Fredholm integral equations. Second, apply least-square technique for minimizing the error terms on the given domain. Third, obtain a system of linear algebra equations which we solve for control points.

An algorithm is illustrated by several numerical examples with comparison tables and written computer programs in MatLab (V 7.1) for the given algorithm.

الحل العددي لمعادلة فولتيرا- فريدهولم التكاملية من النوع الثاني بأستخدام تقنية المربعات الصغرى

شازاد شوفي أحمد

قسم الرياضيات، كلية العلوم، جامعة السليمانية، السليمانية- العراق.

الخلاصة

الهدف في هذا البحث أيجاد الحل العددي لصنف مهم من مزيج للمعادلات الخطية التي تدعى فولتيرا- فريدهولم التكاملية ، التي تستخدم في المجالات التكنولوجية ' الميكانيكية و الفيزياء الرياضية.

الخطوات الاساسية هي: أولاً ' تقريب الدالة بواسطة الضرب التنسري (Tensor Product) (جبري-اً أو جبنشيف)-سطوحياً و تعويضها في معادلة فولتيرا- فريدهولم التكاملية. ثانياً ' تطبيـق تقنيـة المربعـات الصغرى لتصغير مربع الخطأ في المجال المعطاة . و ثالثاً الحصول علي نظام للمعادلات الجبرية الخطية و التي نحلها لايجاد قيم الثوابت (نقاط السيطرة).

وأخيراً أعطينا خوارزمية كفوئه و قمنا بتوضيحها بواسطة العديد من الامثله العددية مع جداول المقارنة و كتبنـا البـرامج الحاسـوبية باسـتخدام (MatLab V 7.1) للخوارزميـة أعـلاه .

Introduction

In this paper we consider linear two-dimensional Volterra-Fredholm (V-F) integral equation of second kind:

$$u(x,t) = f(x,t) + \int_c^t \int_\Omega N(x,t,y,s)u(y,s)\,dy\,ds \quad \dots (1)$$

Where f is the given function in domain $D - \Omega \times [c,d]$ (Ω- a compact subset of Euclidean space $\mathbb{R}^k$ or compact manifold) and u is unknown function in D. The given kernel N is defined in domain $\Gamma = \{(x,t,y,s): x,y \in \Omega \text{ and } c \le s \le t \le d\}$. It will be assumed that the functions $N: D \times D \to \mathbb{R}$ and $f: D \to \mathbb{R}$ are bounded and continuous. On virtue this property the prove of existence and uniqueness of solution for equation (1) in space $L^2(D)$ are given in [1].

The consider integral equation in mixed type plays a very important role in mechanics and technology, with special attention paid to large sense of power engineering. Some initial-boundary problems for a number of partial differential equations in physics can be reduced to consider integral equation [1].

Numerical results for double integral equation had been treated by many authors and different methods are used by: H.Brunner (Collocation method), Lechoslaw Hacia (Galerkin method), Valise Carutasu (Spline functions) and (Taylor's expansion method)

using by Shazad Shawki Ahmed, [2, 1, 3, and 4].

In this paper we propose a new procedure for solving the mixed integral equation of type (V-F), using algebraic and orthogonal (Chebyshev) polynomials with the aid of least-squares techniques for two variables.

Preliminaries [5, 6]

The Chebyshev polynomial of the first kind and k-th degree $T_k(x)$ for interval $[-1,1]$ are even and odd functioning of x defined by the relation:

$$T_k(x) = \cos(k\cos^{-1}x), \quad k = 0,1,2,\dots$$

Or by series expansion as:

$$T_0(x) = 1,$$

$$T_k(x) = \frac{k}{2}\sum_{r=0}^{\lfloor k/2 \rfloor} (-1)^r \frac{(k-r-1)!}{r!(k-2r)!}(2x)^{k-2r}$$

$$k = 1,2,3,\dots \quad \dots (2)$$

Where the bracket $\lfloor . \rfloor$ denotes the largest integer not greater than the number it embraces. The Chebyshev polynomials $T_k(x)$ are an important set of orthogonal functions over the interval $[-1,1]$ with weighting function $w(x) = 1/\sqrt{1-x^2}$, that is

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}dx = \begin{cases} 0 & n \neq m \\ \pi/2 & n = m \neq 0 \\ \pi & n = m = 0 \end{cases}$$

Although the T_k are defined only on the interval $[-1,1]$, a simple change of variable allows the expansion to be used to represent a function between two arbitrary limits, $[a,b]$:

$$\theta_x = \frac{x - \frac{1}{2}(b+a)}{\frac{1}{2}(b-a)}, \qquad a \le x \le b$$

To shift the number x in the interval $[a,b]$ into corresponding number θ_x in the interval $[-1,1]$. Therefore use equation (2) to find $T_k(\theta_x)$.

At last, in this section, we defined also the algebraic polynomial of k-th degree for interval $[a,b]$ by the relation:

$$P_k(x) = (x-a)^k, \qquad k = 0,1,2,\dots \quad \dots (3)$$

The Method

The approximate solution of equation (1) proposed in the form:

$$u_k(x,t) = \sum_{r=0}^{k} \beta_r \psi_r(x,t)$$

with basis functions $\{\psi_r\}$, [7]. In practice, we take $\psi_r(x,t) = \phi_i(x)\phi_j(t)$, where $i = 0,1,\dots,n$; $j = 0,1,\dots,m$ and $\{\phi_i\}$ is a linearly independent in $L^2(\Omega)$, $\{\phi_j\}$ is a linearly independent in $L^2([c,d])$.

Then we seek an approximate solution in the formula:

$$u_{nm}(x,t) = \sum_{i=0}^{n}\sum_{j=0}^{m} \beta_{ij}\phi_i(x)\phi_j(t), (x,t) \in D \quad \dots (4)$$

This equation shows a mathematical description of the tensor product ϕ-surface, where the control points β_{ij} are undermined constants coefficients, which control the shape of the ϕ-surface.

The solution u of the mixed integral equation (1) will be approximated by an element u_{nm} as in equation (4) submitted it, to obtain:

$$\sum_{i=0}^{n}\sum_{j=0}^{m}\beta_{ij}\phi_i(x)\phi_j(t) = f(x,t) + \sum_{i=0}^{n}\sum_{j=0}^{m}\beta_{ij}\int_c^t\int_\Omega N(x,t,y,s)\phi_i(y)\phi_j(s)dy\,ds + E_{nm}(x,t) \quad \ldots(5)$$

Where E_{nm} is the error involved which depends on $(x,t)\in D$ and on the way that control points are chosen. Define:

$$\chi_{ij}(x,t) = \phi_i(x)\phi_j(t) - \int_c^t\int_\Omega N(x,t,y,s)\phi_i(y)\phi_j(s)dy\,ds$$

Thus equation (5) becomes:

$$E_{nm}(x,t,\beta) = \sum_{i=0}^{n}\sum_{j=0}^{m}\beta_{ij}\chi_{ij}(x,t) - f(x,t) \quad \ldots(6)$$

The main point here is how to find the coefficients β_{ij}'s of the approximate solution (4) such that error is minimized.

The general least-squares techniques insist on minimizing the norm of the error function E_{nm} on the domain D. The least-squares technique used to minimized $\|E_{nm}\|_2$ is equivalent to that used to minimized $\|E_{nm}\|_2^2$, using the definition of (L_2) norm, [5], to get:

$$\min\|E_{nm}\|_2^2 = \min\left[\iint_D |E_{nm}(x,t)|^2 dx dt\right] = \min\underbrace{\left[\int_c^d\int_\Omega\left|\sum_{i=0}^{n}\sum_{j=0}^{m}\beta_{ij}\chi_{ij}(x,t) - f(x,t)\right|^2 dx\,dt\right]}_{I[\beta]}$$

In order to find this minimum, we compute the partial derivative of $I[\beta]$ with respect to all control points $\beta_{\ell r}$ and setting each of these derivatives to zero, i.e.

$$\frac{\partial I[\beta]}{\partial\beta_{\ell r}} = 0 \quad for\ each\ \ell = 0,1,\ldots,n;\ and\ r = 0,1,\ldots,m$$

At last putting the resulting equations as a system of algebraic linear equations:

$$\sum_{i=0}^{n}\sum_{j=0}^{m}\beta_{ij}a_{\ell rij} = f_{\ell r} \quad \ldots(7)$$
$$for\ each\ \ell = 0,1,\ldots,n;\ r = 0,1,\ldots,m$$

with:

$$\left.\begin{aligned} a_{\ell rij} &= \int_c^d\int_\Omega \chi_{\ell r}(x,t)\chi_{ij}(x,t)dx\,dt \\ f_{\ell r} &= \int_c^d\int_\Omega \chi_{\ell r}(x,t)f(x,t)dx\,dt \end{aligned}\right\} \quad \ldots(7a)$$

$$\chi_{ij}(x,t) = \phi_i(x)\phi_j(t) - \int_c^t\int_\Omega N(x,t,y,s)\phi_i(y)\phi_j(s)dy\,ds \quad \ldots(7b)$$

Writing the resulting equations (7) in matrix from yields:

$$AB = F \quad \ldots(8)$$

where A and B are constant matrices with dimensions $p\times p$ and $p\times 1$ respectively. The vector B is an $p\times 1$ block-column vector whose rows are the unknown control points β_{ij} for $0\le i\le n$ and $0\le j\le m$, such that:
$p = (n+1)(m+1);\ n,m\in\mathbb{Z}^+$
with $F = [f_q]^T$ and $B = [\beta_q]^T$ for each $q = 0,1,\ldots,p$. On the other hand, $q = ij$ (as a component) where $j = 0,1,\ldots,m; i = 0,1,\ldots,n$. Furthermore:

$$A = [A_{\ell i}]\ \ and\ \ A_{\ell i} = [a_{\ell rij}]$$

where

$$\begin{cases} \ell = 0,1,\ldots,n\ ;\ \ r = 0,1,\ldots,m \\ i = 0,1,\ldots,n\ ;\ \ j = 0,1,\ldots,m \end{cases}$$

For more detail see Appendix (A). The system of equations (8) can be partitioned as follows:

$$\begin{bmatrix} A_{00} & A_{01} & \ldots & A_{0n} \\ A_{10} & A_{11} & \ldots & A_{1n} \\ \vdots & \vdots & & \vdots \\ A_{n0} & A_{n1} & \ldots & A_{nn} \end{bmatrix}_{(n+1)\times(n+1)} \begin{bmatrix} B_0 \\ B_1 \\ \vdots \\ B_n \end{bmatrix}_{(n+1)\times 1} = \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_n \end{bmatrix}_{(n+1)\times 1} \quad \ldots(9)$$

where, for all $i, \ell = 0,1,\dots,n$,

$$F_i = [f_{i0} \quad f_{i1} \quad \dots \quad f_{im}]^T;$$
$$B_i = [\beta_{i0} \quad \beta_{i1} \quad \dots \quad \beta_{im}]^T$$

and

$$A_{\ell i} = \begin{bmatrix} a_{\ell 0 0} & a_{\ell 0 1} & \dots & a_{\ell 0 m} \\ a_{\ell 1 0} & a_{\ell 1 1} & \dots & a_{\ell 1 m} \\ \vdots & \vdots & & \vdots \\ a_{\ell m 0} & a_{\ell m 1} & \dots & a_{\ell m m} \end{bmatrix}_{(m+1)\times(m+1)} \quad \dots(10)$$

Note that, the diagonal blocks in matrix A are square of order $(m+1)$ and non-singular. The block Jacobi-iterative method [8] for the solution of the system (9) is given by:

$$A_{ii}B_i^{(k)} = F_i - \sum_{\substack{j=0 \\ j\neq i}}^{n} A_{ij}B_j^{(k-1)}, \qquad i = 0,1,\dots,n \quad \dots(11)$$

Consequently, in the i-th of the total $(n+1)$ phases of the k-th iteration of the block Jacobi-iterative method, equation (11) for $B_i^{(k)}$ is solved. These sub-systems can be solved using Gauss-elimination procedure [8].

Finally, we will attempt to solve the system of linear algebraic equations in (7) for control points β_{ij}'s by above technique, substituting these values in (4) to get the approximation solution $u_{nm}(x,t)$ of equation (1) . Then the resulting method error $e(x,t) = u(x,t) - u_{nm}(x,t)$ satisfies, [2]:

$$\|e\|_\infty = \sup\{|e(x,t)| : (x,t) \in D\} \quad (as\ n,m \to \infty)$$

Because the calculation of error given in above, is not easy so we use the norm of matrix to study the quantitatively of the error in this approximation method, and we use the relative error which is $\|u - u_{nm}\|/\|u\|$, [5].The method presented above was implemented in the MatLab (V 7.1), (see Appendix B).

The Algorithm [LS (V-F) M]

Step 1: Input the number of terms (n,m) for approximation the function $u(x,t)$.

Step 2: For all $i = 0,1,\dots,n$ and $j = 0,1,\dots,m$:

a. Evaluate $\chi_{ij}(x,t)$, apply equation (7b).

b. Compute $a_{\ell rij}$ and $f_{\ell r}$ for each $\ell = 0,1,\dots,n$; $r = 0,1,\dots,m$; using equation (7a).

Step 3: Construct the block-matrices A and F, which are represented in system (9).

Step 4: Solving the system (9) for control points β_{ij} ; $i - 0,1,\dots,n$ and $j = 0,1,\dots,m$, using block Jacobi-iterative as in (11) with Gauss- elimination method.

Step 5: Substituting all β_{ij}'s into equation (4) to obtain the approximate solution $u_{nm}(x,t)$ of $u(x,t)$.

By using the basic functions $\phi_k(k = i \ or \ j)$ developed in section (3) we are going to apply the least-square mixed to the following cases:

(1) **A**lgebraic polynomials.

(2) Orthogonal (**C**hebyshev) polynomials.

1. Algebraic Polynomials

In this section the solution of mixed type (V-F) integral equations will be found using algebraic polynomial (3) accompanied with least-squares techniques.

Here, the unknown function $u(x,t)$ in equation (1) is approximated by the form:

$$u_{nm}(x,t) = \sum_{i=0}^{n}\sum_{j=0}^{m} \beta_{ij}(x-a)^i(t-c)^j \ ; \quad (x,t) \in D$$

Doing the same stages described in section (3), obtain the system (7) with their descriptions. Here, change each $\phi_i(x)$ and $\phi_j(t)$ in (7b) by $(x-a)^i$ and $(t-c)^j$ respectively. At last, applying the algorithm [LS (V-F) M] to founding all control points β_{ij}'s to get the approximate solutions for mixed integral equation (1).

2. Chebyshev Polynomials

Here, the unknown function $u(x,t)$ in equation (1) is approximated by a tensor-product Chebyshev polynomial expansion, that is, a polynomial of the form:

$$u_{nm}(x,t) = {\sum_{i=0}^{n}}' {\sum_{j=0}^{m}}' \beta_{ij}T_i(\theta_x)T_j(\theta_t) \quad \dots(12)$$

with $\theta_x = 2\left(\frac{x-a}{b-a}\right) - 1$; $\theta_t = 2\left(\frac{t-a}{b-a}\right) - 1$ and $T_k(k = i \ or \ j)$ are the Chebyshev polynomials of the first kind with k-th degree .

Note that the primes $(')$ on the summation signs mean that the zeroth row and column of the matrix of coefficients are each

multiplied by $\frac{1}{2}$, and hence that β_{00} is multiplied by $\frac{1}{4}$ [4].

Substituting $u_{nm}(x,t)$ into equation (1) and applying the same steps described in section (3), we conclude the following system:

$$\sum_{i=0}^{n}{}' \sum_{j=0}^{m}{}' \beta_{ij} a_{\ell r i j} = f_{\ell r}$$

$$for\ each \quad \ell = 0,1,\dots,n \ ; \ r = 0,1,\dots,m$$

where $a_{\ell r i j}$ and $f_{\ell r}$ are as the same in equation (7a), **for all** ℓ **and** r, with

$$a_{\ell r i j} = \begin{cases} \frac{1}{4} a_{\ell r i j} & if \quad i = j = 0 \\ \frac{1}{2} a_{\ell r i j} & if \quad i = 0 \ or \ j = 0 \\ a_{\ell r i j} & O.W. \end{cases}$$

$$\chi_{ij}(x,t) = T_i(\theta_x) T_j(\vartheta_t) - \int_c^t \int_\Omega N(x,t,y,s) T_i(\theta_y) T_j(\theta_s) dy\, ds$$

At last, applying the algorithm [LS (V-F) M] to founding all control points β_{ij}'s with above properties of $a_{\ell r i j}$ to get the approximate solutions for mixed integral equation (1).

Numerical Experiment

For numerical verification of the above method we consider the following examples:

Example (1):

Consider the following Volterra-Fredholm integral equation:

$$u(x,t) = 1/2 - x^2 + t + (1/6)t^3 - (1/2)xt^2\cos(t) - (1/6)\,xt\cos(t) + \int_0^t \int_0^1 (x\cos(t) - sy)u(y,s)dy\,ds \ , \quad (x,t) \in D_1 = [0,1]\times[0,1] \quad \dots(13)$$

Let us assume that $u(x,t)$ is approximated by:

- **A**lgebraic polynomial:

$$u_{nm}^{A}(x,t) = \sum_{i=0}^{n}\sum_{j=0}^{m} \beta_{ij} x^i t^j \ , \qquad (x,t) \in D_1 \quad \dots(14)$$

- **C**hebyshev polynomial:

$$u_{nm}^{C}(x,t) = \sum_{i=0}^{n}{}' \sum_{j=0}^{m}{}' \beta_{ij} T_i(2x-1)\, T_j(2t-1) \ , \quad (x,t) \in D_1 \quad \dots(15)$$

To process (V-F) integral equation (13) we apply the algorithm [LS (V-F) M], here we take $n = 2 \quad and \quad m = 1$, so we obtain:

$$A^A = \left[\begin{array}{ccc:ccc} 0.8132228817 & 0.4324889226 & 0.3871913439 & 0.2025702236 & 0.2516541969 & 0.1804887865 \\ 0.4324889226 & 0.3166121068 & 0.2164684188 & 0.1566157048 & 0.1443869317 & 0.1088470203 \\ 0.3871913439 & 0.2164684188 & 0.2675661790 & 0.1432566585 & 0.2080342884 & 0.1071785887 \\ \hdots & \hdots & \hdots & \hdots & \hdots & \hdots \\ 0.2025702236 & 0.1566157048 & 0.1432566585 & 0.1053506851 & 0.1094949545 & 0.07924806766 \\ 0.2516541969 & 0.1443869317 & 0.2080342884 & 0.1094949545 & 0.1666474778 & 0.08820824368 \\ 0.1804887865 & 0.1088470203 & 0.1071785887 & 0.07924806766 & 0.08820824368 & 0.06379208280 \end{array}\right]$$

$$A^C = \left[\begin{array}{ccc:ccc} 0.2033057204 & 0.02587748178 & -0.0194200970 & -0.0158567568 & -0.1355371469 & -0.017251655 \\ 0.01298874089 & 0.1748578091 & 0.01986801198 & 0.00143813789 & -0.0086258273 & -0.116571878 \\ -0.0097100484 & 0.01986801198 & 0.1678611111 & 0.0001648210 & 0.00647886566 & -0.013245841 \\ \hdots & \hdots & \hdots & \hdots & \hdots & \hdots \\ -0.0039641892 & 0.00719068943 & 0.0000771605 & 0.1111772487 & 0.00264279280 & -0.0004793793 \\ -0.0677685735 & -0.0086258273 & 0.0064788657 & 0.0052865866 & 0.2229668268 & 0.00576055161 \\ -0.0048129136 & -0.0582859364 & -0.006622671 & -0.0004793793 & 0.00287527575 & 0.1573758094 \end{array}\right]$$

and

$$F^A = [0.5874461665 \quad 0.3884696364 \quad 0.2070298024 \ : \ 0.1484058621 \quad 0.1035665524 \quad 0.08088314488]^T$$

$$F^C = [0.5874461665 \quad 0.1894931062 \quad -0.1738865618 \ : \ -0.0099292627 \quad -0.2402598838 \quad -0.06316486875]^T$$

Next, we substitute these values in equation (8) to construct the block-matrices in linear system (9). Table (1) presents the control points β_{ij} 's for Algebraic (14) and Chebyshev (15) polynomials.

Table (1)

Method \ β_{ij}	β_{00}	β_{01}	β_{10}	β_{11}	β_{20}	β_{21}
Algebraic	0.5	1.0	$-0.31801e-13$	$0.33834e-13$	-1.0	$-0.5255e-13$
Chebyshev	2.5	1.0	-1.0	$-0.3047e-16$	-0.25	$0.6228e-16$

Putting these values in approximation (14) and (15) respectively, we obtain:

$$u_{nm}^{A}(x,t) = 0.5 + t - (0.31801e-13)\,x + (0.33834e-13)\,xt - x^2 - (0.5255e-13)\,x^2t$$

and

$$u_{nm}^{C}(x,t) = 0.5 + t + (0.55925e-15)\,x - (0.1119e-14)\,xt - x^2 + (0.9966e-15)\,x^2t$$

where the exact solution is $(1/2 - x^2 + t)$. We could all these manipulations in such the program; see Appendix (B), which can find the relative error for it with running time (R.T.). In this example the error for algebraic and Chebyshev approximations are: $(2.61351e-014)$ and $(8.9456e-017)$ with running time: $(1.060976\ sec)$ and $(1.243939\ sec)$ respectively.

Example (2):

Consider the following mixed (V-F) integral equation:

$$u(x,t) = x - \sin(t) - 2e^{-x}(t\cos(t) - \sin(t)) + \int_0^t \int_{-1}^{1} se^{-x}u(y,s)\,dy\,ds,$$

whose exact solution is $u(x,t) = x - \sin(t)$, $\forall (x,t) \in D_2 = [-1,1] \times [0,1]$

Assume that the approximate solution is in the forms:

- **A**lgebraic polynomial:

$$u_{nm}^{A}(x,t) = \sum_{i=0}^{n}\sum_{j=0}^{m} \beta_{ij}(x+1)^i t^j, \quad (x,t) \in D_2 \qquad \ldots(16)$$

- **C**hebyshev polynomial:

$$u_{nm}^{C}(x,t) = {\sum_{i=0}^{n}}' {\sum_{j=0}^{m}}' \beta_{ij}T_i(x)\,T_j(2t-1), \quad (x,t) \in D_2 \qquad \ldots(17)$$

Take $n-1$ *and* $m-3$, then apply algorithm [LS (V-F) M] to find the approximate solution (16 and 17) of consider problem by running the program (in appendix B). So, table (2) present the control points β_{ij}'s of equations (16 and 17) for each i *and* j, respectively.

Table (2)

Method \ β_{ij}	*Algebraic Polynomial*	*Chebyshev Polynomial*
β_{00}	-0.9997394418	-1.799539750
β_{01}	-1.004922061	-0.8504228742
β_{02}	0.01962675087	0.05886470195
β_{03}	0.1438530528	0.009013204415
β_{10}	0.9999916901	2.000002848
β_{11}	0.0001592185453	0.000005099398626
β_{12}	-0.0004848490177	0.000006559726709
β_{13}	0.0003582178876	0.000011194309

Thus, we obtain the following approximate formulas:

$$u_{nm}^{A}(x,t) = -0.999739 - 1.004922\,t$$

$$+0.019627\,t^2 + 0.143853\,t^3$$
$$+0.999992\,(x+1) + 0.000159(x+1)t$$
$$-0.000485(x+1)t^2 + 0.000358(x+1)t^3$$

and

$$u^{C}_{nm}(x,t) = \frac{1}{4}(-1.7995398)$$
$$+\frac{1}{2}(-0.850423)\,T_1(2t-1)$$
$$+\frac{1}{2}(0.056865)\,T_2(2t-1)$$
$$+\frac{1}{2}(0.009013)\,T_3(2t-1)$$
$$+\frac{1}{2}(2.000003)\,T_1(x)$$
$$+0.0000051\,T_1(x)T_1(2t-1)$$
$$+0.0000066\,T_1(x)T_2(2t-1)$$
$$+0.000011\,T_1(x)T_3(2t-1)$$

Table (3), as well, exhibits the convergence of the approximated solutions. It presents the comparison between the exact solutions $u(x,t)$ and approximate solutions $u^{A.C}_{nm}(x,t)$ which depends on relative error and running time with different values of n and m. The error values $E_{nm}(x,t,\beta)$, $\forall(x,t) \in D_2$ are also included by applying the formula (6).

Table (3)

(n,m)		(1,3)	(1,5)	(1,7)	(1,9)
Algebraic	*Error*	0.00012972	2.79026289 e 007	4.910687692 e 009	2.221446461 e 006
	$E_{nm}(x,t,\beta)$	0.00982581	2.11695005 e − 005	2.648585951 e − 007	0.000168959
	R.T./sec	1.457584	3.069190	5.111184	7.963633
Chebyshev	*Error*	0.00012972	2.79014780 e − 007	3.224827632 e − 010	5.990197443 e − 011
	$E_{nm}(x,t,\beta)$	0.00982581	2.116914548 e 005	2.439781148 e 008	3.503084418 e 009
	R.T./sec	1.638185	3.836393	7.520051	13.431364

Conclusion

Algebraic and orthogonal polynomials with aid of least-squares technique are introduced to find the numerical solutions of linear Volterra-Fredholm integral equations. The approximate results are easily obtained by a few computations. Several examples are included for illustration. In practice, we conclude that:

- This method can be used even where there is no information about the exact solution (from the evaluation of error function E_{nm} in equation (6)).
- Numerical computations of this method, compared to the numerical schemes are simple and inexpensive.
- The solution is given by a function.
- We have shown that increasing the number of the basis functions one obtains better results.
- The choice algebraic polynomial is unwise since for large n and m it usually leads to ill-conditioning and we favors the choice Chebyshev polynomial, see last column in table (3).
- The disadvantage of the new methods is their dependence on a number of basis functions n and m which make the square matrix A in equation (8) with dimension $(n+1)(m+1) \times (n+1)(m+1)$ very large, which need large memory of computer with too much time to compute it.

References

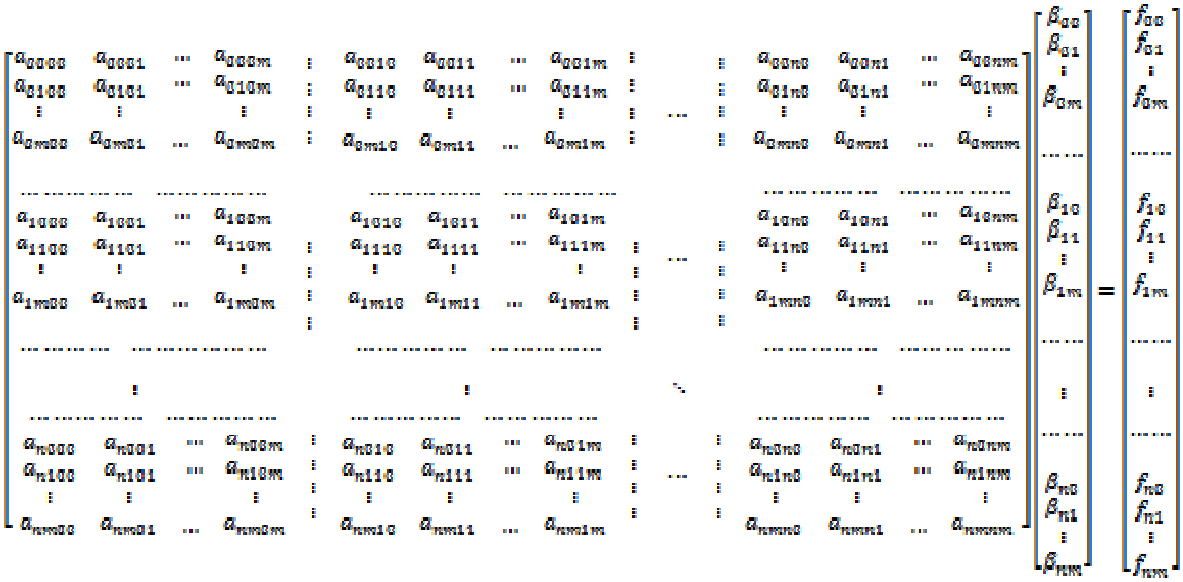
1. Lechoslaw H. **2002.** Computational Results for Integral Equations in Space-Time.

Computational Methods in Science And Technology **8**(1), 7-15.

2. Brunner,H. and Kautheu, J.P. **1989**. The Numerical Solution of Two-Dimensional Volterra Integral Equations by Collocation and Iterated Collocation. *IMA Journal of Numerical Analysis* **9**, 47-59.
3. Vasile C. **2001**. Numerical Solution of Two-dimensional non-Linear Fredholm Integral Equations of the Second Kind by Splint Functions. *General Mathematics*, Vol.**9**, No. 1.
4. Ahmed S.S. **2006**. Expansion Method for Solving Fredholm Integral Equations in Space-Time. *Journal of Kirkuk University*, Vol.**1** No. 1.
5. JAMES F.E. **2002**. *An Introduction to Numerical Methods and Analysis*. John Wiley and Sons; Inc; New York. pp. 20-60.
6. John H.M. and Kurtis D.F. **2004**. *Numerical Methods Using MatLab*; Fourth Edition; Pearson Education, Inc. pp. 1-100.
7. Delves, L.M. and Walsh, J. **1974**. *Numerical Solution of Integral Equations*. Oxford. pp. 25-75.
8. Alfio Q.; Riccardo S. and Fausto S. **2007**. *Numerical Mathematics*; Second Edition. Speringer Berlin Heidelberg. pp. 1-25.

Appendix (A):

For instance, in equation (7) thus:



Appendix (B):

In this appendix we introduce a program in MatLab (V 7.1) for the method which is given in section-3 (i.e., algorithm [LS (V-F) M]):

```
clc
clear
format long g
syms x t y s
% [a1,b1];[a2,b2] is the boundary points of
integrals
a1=0 ;  b1=1 ;  a2=0 ;  b2=1 ;
% (n,m) is the number of terms in power
function or Chebyshev expansion
n=2 ;   m=1 ;
% To apply power function take (H=1) and
(H=2) for Chebyshev expansion
H=input('input 1: for using power function;
or   2: for Chebyshev expansion :  ');
% tic & toc is using for determine the time
in program
tic
q=(n+1)*(m+1);
A=zeros(q);
B=zeros(q,1);T2=zeros(n+1,m+1);Tp=sym(zeros(
n+1,m+1,q));Q=zeros(q,1);
[Ker,Gg]=NG(x,t,y,s);
[xt,ys]=IJXY(x,t,y,s,n,m,a1,b1,a2,b2,H);
p=ys*Ker;
T2=xt-int(int(p,y,a1,b1),s,a2,t);
BB=int(int(T2*Gg,x,a1,b1),t,a2,b2);
B(:)=double(BB');
L=1;
for i=0:n
    for j=0:m
        TP(:,:,L)=T2(i+1,j+1)*T2 ;
```

```
A=int(int(TP(:,:,L),x,a1,b1),t,a2,b2);
QQ=double(AA');
        Q(:)=QQ ; A(L,:)=Q ;
        L=L+1;
    end
end
switch H
    case 1
        A(:,:)=A(:,:);
    case 2
        A(:,1)=(1/4)*A(:,1);
        A(:,2:m+1)=(1/2)*A(:,2:m+1);
        for i=m+2:m+1:q
A(:,i)=(1/2)*A(:,i); end
    otherwise
end
%vpa(A,10),vpa(B,10)
cq=zeros(q,1);
c1=inv(A)*B;  cq=vpa(c1,10); cq'
switch H
    case 1
        cq(:,1)=cq(:,1);
    case 2
        cq(1,1)=(1/4)*cq(1,1);
        cq(2:m+1,1)=(1/2)*cq(2:m+1,1);
        for i=m+2:m+1:q
cq(i,1)=(1/2)*cq(i,1); end
    otherwise
end
Ls=1;f1=0;Lc=1;fc=0;
for i=0:n
    for j=0:m
        f1=f1+cq(Ls,1)*xt(i+1,j+1);
        fc=fc+cq(Lc,1)*T2(i+1,j+1);
        Ls=Ls+1;Lc=Lc+1;
    end
end
toc

%  The following steps using to find error's
for u and g by matrix-norm(2)
fv=vpa(f1,10);f=simplify(fv);
p1q=char(f);apq=inline(p1q,'x','t');
XT=char(fc-Gg);TX=inline(XT,'x','t');
exact=0.5-x^2+t
,ex=char(exact);exa=inline(ex,'x','t');
n0=100;m0=100; Erroru=0.0; Errorg=0.0;
DF=zeros(n0,m0);DG=zeros(n0,m0);DD=zeros(n0,
m0);
h0=(b1-a1)/(n0-1);k0=(b2-a2)/(m0-1);
for i0=1:n0
    xo=a1+(i0-1)*h0;
    for j0=1:m0
        t10=a2+(j0-1)*k0;
        DD(i0,j0)=exa(xo,t10);
DF(i0,j0)=apq(xo,t10);
        DG(i0,j0)=TX(xo,t10);
    end
end
Erroru=norm(DD-DF)/norm(DD);
Errorg=norm(DG);
pretty(simplify(f)),
[Erroru   Errorg]

% the subroutine of algebraic and Chebyshev
polynomials
function
[JIx,JIy]=IJXY(x1,t1,y1,s1,n1,m1,a11,b11,a22
,b22,HH)
format long g
JIx=sym(zeros(n1,m1));JIy=sym(zeros(n1,m1));
syms xx tt yy ss
switch HH
    case 1
        for i1=0:n1
            for j1=0:m1
                Ix=(x1-a11)^i1;  Jt=(t1-
a22)^j1;
                Iy=(y1-a11)^i1;  Js=(s1-
a22)^j1;
                JIx(i1+1,j1+1)=Ix*Jt;
JIy(i1+1,j1+1)=Iy*Js;
            end
        end
    case 2
        for i1=0:n1
            for j1=0:m1
                sx=0;sy=0;z1=floor(i1/2);
                st=0;ss1=0;z2=floor(j1/2);
                xx=2*((x1-a11)/(b11-a11))-1;
tt=2*((t1-a22)/(b22-a22))-1;
                yy=2*((y1-a11)/(b11-a11))-1;
ss=2*((s1-a22)/(b22-a22))-1;
                if i1==0
                    chebx=1; cheby=1;
                else
                    for r1=0:z1

sj=1;si=1;skx=1;sky=1;sj=factorial(i1-r1-1);

si=factorial(r1)*factorial(i1-2*r1);
                        sji=sj/si;si1=(-1);
                        for i=0:r1
                            si1=si1*(-1);
                        end
                        skx=(2*xx)^(i1-
2*r1); sky=(2*yy)^(i1-2*r1);
                        sx=sx+sji*si1*skx;
sy=sy+sji*si1*sky;
                    end
                    chebx=sx*i1/2;
cheby=sy*i1/2;
                end
                if j1==0
                    chebt=1; chebs=1;
                else
                    for r2=0:z2

sj=1;si=1;skt=1;sks=1;sj=factorial(j1-r2-1);

si=factorial(r2)*factorial(j1-2*r2);
                        sji=sj/si;sj1=(-1);
                        for i=0:r2
                            sj1=sj1*(-1);
                        end
                        skt=(2*tt)^(j1-
2*r2); sks=(2*ss)^(j1-2*r2);
                        st=st+sji*sj1*skt;
ss1=ss1+sji*sj1*sks;
                    end
                    chebt=st*j1/2;
chebs=ss1*j1/2;
                end
                JIx(i1+1,j1+1)=chebx*chebt;
JIy(i1+1,j1+1)=cheby*chebs;
            end
        end
    otherwise
end

function [M,Z]=NG(x,t,y,s)
format long g
% input the kernel N(x,t,y,s)

M=x*cos(t)-s*y ;
%M=s*exp(-x) ;

% input the function f(x,t)

Z=0.5-x^2+t+(1/6)*t^3-(1/2)*x*cos(t)*t^2-
(1/6)*x*t*cos(t)  ;
%Z=x-sin(t)-2*exp(-x)*(-sin(t)+t*cos(t))  ;
```