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.
Introduction

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

\[ u(x, t) = f(x, t) + \int_{c}^{d} \int_{\Omega} N(x, t, y, s) u(y, s) \, dy \, ds \quad \ldots \quad (1) \]

Where \( f \) is the given function in domain \( D \subset \Omega \times [c, d] \) (\( \Omega \) - 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 \( \Gamma = \{(x, t, y, s); x, y \in \Omega \text{ and } c < y < t < d\} \). It will be assumed that the functions \( N; D \times D \rightarrow \mathbb{R} \) and \( f; D \rightarrow \mathbb{R} \) 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:

\[ T_k(x) = \cos(k \cos^{-1} x), \quad k = 0, 1, 2, \ldots \]

Or by series expansion as:

\[ T_k(x) = \frac{1}{2^n k!} \sum_{r=0}^{n} (-1)^r \binom{n}{r} \binom{k-r-1}{r} (2x)^{k-2r}, \quad k = 1, 2, 3, \ldots \quad \ldots \ldots (2) \]

Where the bracket \( \lfloor x \rfloor \) 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

\[ \int_{-1}^{1} T_m(x) T_n(x) w(x) \, dx = \begin{cases} 0 & m \neq n \\ \frac{\pi}{2} & m = n = 0 \\ \frac{\pi}{n} & m = n > 0 \\ 0 & n = 0 \neq m \end{cases} \]

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, \([a, b] \):

\[ x = \frac{1}{2} (b + a) + \frac{1}{2} (b - a) \sin \theta, \quad a \leq x \leq b \]

To shift the number \( x \) in the interval \([a, b]\) into corresponding number \( \theta(x) \) 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:

\[ p_k(x) = (x - a)^k, \quad k = 0, 1, 2, \ldots \quad \ldots \ldots (3) \]

The Method

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

\[ u_k(x, t) = \sum_{\varepsilon=0}^{\infty} \beta_{\varepsilon} \phi_{\varepsilon}(x, t) \]

with basis functions \( \{\phi_{\varepsilon}\} \). [7]. In practice, we take \( \phi_{\varepsilon}(x, t) = \phi_{\varepsilon}(x) \phi_{\varepsilon}(t) \), where \( \varepsilon = 0, 1, \ldots, n \); \( f = 0, 1, \ldots, m \) and \( \{\phi_{\varepsilon}\} \) is a linearly independent in \( L^2(\Omega) \). \( \{\phi_{\varepsilon}\} \) is a linearly independent in \( L^2([a, b]) \).

Then we seek an approximate solution in the formula:

\[ u_{approx}(x, t) = \sum_{\varepsilon=0}^{n} \sum_{f=0}^{m} \beta_{\varepsilon f} \phi_{\varepsilon}(x) \phi_{f}(t), \quad (x, t) \in D \]

This equation shows a mathematical description of the tensor product \( \phi \)-surface, where the control points \( \beta_{\varepsilon f} \) are undermined constants coefficients, which control the shape of the \( \phi \)-surface.
The solution \( \mathbf{u} \) of the mixed integral equation (1) will be approximated by an element \( u_{\text{sum}} \) as in equation (4) submitted it, to obtain:

\[
\sum_{i=0}^{n} \sum_{j=0}^{m} \beta_{ij} \varphi_i(x) \varphi_j(t) = f(x, t)
\]

\[
+ \sum_{i=0}^{n} \sum_{j=0}^{m} \beta_{ij} \int \int N(x, t, y, s) \varphi_i(y) \varphi_j(s) dy ds
+ E_{\text{sum}}(x, t)
\]  

(5)

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

\[
X_{\ell}(x, t) = \varphi_i(x) \varphi_j(t)
\]

\[
- \int \int N(x, t, y, s) \varphi_i(y) \varphi_j(s) dy ds
\]

Thus equation (5) becomes:

\[
E_{\text{sum}}(x, t, \beta) = \sum_{i=0}^{n} \sum_{j=0}^{m} \beta_{ij} X_{\ell}(x, t) - f(x, t)
\]  

(6)

The main point here is how to find the coefficients \( \beta_{ij} \)’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 \( E_{\text{sum}} \) on the domain \( \mathcal{D} \). The least-squares technique used to minimized \( \| E_{\text{sum}} \|_2 \) is equivalent to that used to minimized \( \| E_{\text{sum}} \|_2 \), using the definition of \( (L_2) \) norm, [5], to get:

\[
\min \| E_{\text{sum}} \|_2 = \min \left[ \int \int \left| \sum_{i=0}^{n} \sum_{j=0}^{m} \beta_{ij} X_{\ell}(x, t) - f(x, t) \right|^2 dx dt \right]
\]

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

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

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

\[
\sum_{i=0}^{n} \sum_{j=0}^{m} \beta_{ij} u_{\ell r} = f_{\ell r} \quad \ldots (7)
\]

for each \( \ell = 0, 1, ..., n \); \( r = 0, 1, ..., m \)

with:

\[
\alpha_{\ell r} = \int \int X_{\ell}(x, t) X_{r}(x, t) dx dt \quad \ldots (7a)
\]

\[
f_{\ell r} = \int \int X_{\ell}(x, t) f(x, t) dx dt \quad \ldots (7b)
\]

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

\[
A\beta = F \quad \ldots (8)
\]

where \( A \) and \( F \) are constant matrices with dimensions \( p \times q \) and \( p \times 1 \) respectively. The vector \( \beta \) is an \( n \times m \) block-column vector whose rows are the unknown control points \( \beta_{00} \) for \( 0 \leq \ell \leq n \) and \( 0 \leq j \leq m \), such that:

\[
p = (n + 1)(m + 1) \quad \text{and} \quad q = (n + 1)(m + 1)
\]

with \( F = [f_{\ell r}] \) and \( B = [\alpha_{\ell r}] \) for each \( q = 0, 1, ..., p \). On the other hand, \( q = ij \) (as a component) where \( j = 0, 1, ..., m \); \( i = 0, 1, ..., n \). Furthermore:

\[
A = [A_{\ell i}] \quad \text{and} \quad A_{\ell i} = [\alpha_{\ell r}]
\]

where

\[
\begin{cases}
\ell = 0, 1, ..., n \quad ; \quad r = 0, 1, ..., m \\
i = 0, 1, ..., n \quad ; \quad j = 0, 1, ..., m
\end{cases}
\]

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

\[
\begin{bmatrix}
A_{00} & A_{01} & \ldots & A_{0n} \\
A_{10} & A_{11} & \ldots & A_{1n} \\
\vdots & \vdots & \ddots & \vdots \\
A_{n0} & A_{n1} & \ldots & A_{nm}
\end{bmatrix}
\begin{bmatrix}
B_0 \\
B_1 \\
\vdots \\
B_m
\end{bmatrix}
= \begin{bmatrix}
F_0 \\
F_1 \\
\vdots \\
F_m
\end{bmatrix}
\]

(9)
where, for all \( t, \vartheta = 0, 1, \ldots, n \),
\[
A_{ij} = \begin{bmatrix}
A_{00} & A_{01} & \cdots & A_{0n} \\
A_{10} & A_{11} & \cdots & A_{1n} \\
\vdots & \vdots & \ddots & \vdots \\
A_{n0} & A_{n1} & \cdots & A_{nn}
\end{bmatrix} \quad \text{and}
\]
\[
B_t = \begin{bmatrix}
B_{t0} & B_{t1} & \cdots & B_{tn}
\end{bmatrix}
\]

Note that, the diagonal blocks in matrix \( A \) are square of order \((mn + 1)\) and non-singular. The block Jacobi-iterative method \[8\] for the solution of the system \((9)\) is given by:
\[
A_{ij}B_t^{(k+1,i)} = F_i - \sum_{j=2}^{n} A_{ij}B_j^{(k,i-1)}, \quad i = 0, 1, \ldots, n
\]

Consequently, in the \( t \)-th of the total \((n + 1)\) phases of the \( k \)-th iteration of the block Jacobi-iterative method, equation \((11)\) for \( B_t^{(k,i)} \) 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 \( \beta_{ij} \)'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]:
\[
\|e\|_{\infty} = \sup\{|e(x, t)|: (x, t) \in D\} \quad \text{as} \quad n, m \to \infty
\]

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 \( t = 0, 1, \ldots, n \) and \( j = 1, 2, \ldots, m \):

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

b. Compute \( a_{er} \) and \( f_{er} \) for each \( x_{ij}(x, t) \) 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 \( \beta_{ij} \)'s to get the \( u_{nm}(x, t) \), using block Jacobi-iterative as in \((11)\) with Gauss-elimination method.

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

By using the basic functions \( \phi_k(x) \) developed in section \((3)\) we are going to apply the least-square mixed to the following cases:

1. Algebraic polynomials.
2. Orthogonal (Chebyshev) 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:
\[
u_{nm}(x, t) = \sum_{j=1}^{n} \sum_{i=1}^{m} \beta_{ij}(x - a)^i(t - c)^j, \quad (x, t) \in D
\]

Doing the same stages described in section \((3)\), obtain the system \((7)\) with their descriptions. Here, change each \( \phi_k(x) \) 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 \( \beta_{ij} \)'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:
\[
u_{nm}(x, t) = \sum_{i=0}^{n} \sum_{j=0}^{m} \beta_{ij}T_i(\theta_1)T_j(\theta_2) \quad \text{... (12)}
\]

with \( \theta_{\infty} = 2(x - a)/(x - b) - 1 \) and \( \theta_{\infty} = 2(t - a)/(t - b) - 1 \) and \( T_k(\theta) \) are the Chebyshev polynomials of the first kind with \( k \)-th degree.

Note that the primes \( 'n' \) 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 $\beta_{00}$ is multiplied by $\frac{1}{4}$ [4].

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

$$\sum_{j=0}^{n} \sum_{l=0}^{m} \beta_{il} a_{i} = f_{i},$$

for each $d = 0,1, ..., n$ and $r = 0,1, ..., m$,

where $a_{i}$ and $f_{i}$ are as the same in equation (7a), with

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

At last, applying the algorithm [LS (V-F) M] to founding all control points $\beta_{ij}$'s with above properties of $a_{i}$ 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:

$$u(x,t) = \frac{1}{2} - x^{2} + t + (1/6)t^{3} - \frac{1}{2}xft \cos(t) - \frac{1}{6}xft \cos(t) + \int_{0}^{1} \int_{0}^{1} (x \cos(t) - sy)u(y,s)dyds,$$

$$(x,t) \in D_{1} = [0,1] \times [0,1].$$

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

- **Algebraic polynomial:**

$$u_{approx}^{D}(x,t) = \sum_{i=0}^{n} \sum_{j=0}^{m} \beta_{ij} x^{i} t^{j}, \quad (x,t) \in D_{1},$$

- **Chebyshev polynomial:**

$$u_{approx}^{C}(x,t) = \sum_{i=0}^{n} \sum_{j=0}^{m} \beta_{ij} T_{i}(2x - 1) T_{j}(2t - 1), \quad (x,t) \in D_{1}.$$

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

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

**Table (1)**
Putting these values in approximation (14) and (15) respectively, we obtain:

\[
\begin{align*}
    u_{n,m}^d(x,t) &= 0.5 + \varepsilon - (0.31801 \varepsilon - 13) x + (0.33634 \varepsilon - 13) x^2 t - x^2 \\
    u_{n,m}^s(x,t) &= 0.5 + \varepsilon + (0.55925 \varepsilon - 15) x - (0.1119 \varepsilon - 14) x^2 t + (0.99666 \varepsilon - 15) x^2 t
\end{align*}
\]

where the exact solution is \((1/2 - x^2 + \varepsilon)\). 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:

- Algebraic polynomial:
  \[
  (2.61351 \text{ sec} - 014) \\
  (9.64568 \text{ sec} - 017) \\
  (10.60976 \text{ sec})
  \]

- Chebyshev polynomial:
  \[
  (1.243939 \text{ sec})
  \]

and with running time:

- Algebraic polynomial:
  \[
  \frac{1}{2} - x^2 + \varepsilon
  \]

- Chebyshev polynomial:
  \[
  \frac{1}{2} - x^2 + \varepsilon
  \]

Assume that the approximate solution is in the forms:

- **Algebraic polynomial:**
  \[
  u_{n,m}^d(x,t) = \sum_{i=0}^{N} \sum_{j=0}^{m} \beta_{ij}(x + 1)^i t^j, \\
  \forall (x,t) \in D_2 = [-1,1] \times [0,1]
  \]

- **Chebyshev polynomial:**
  \[
  u_{n,m}^s(x,t) = \sum_{i=0}^{N} \sum_{j=0}^{m} \beta_{ij} T_i(x) T_j(2t - 1), \\
  \forall (x,t) \in D_2 = [-1,1] \times [0,1]
  \]

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 \(\beta_{ij}\)'s of equations (16 and 17) for each \(i\) and \(j\), respectively.

<table>
<thead>
<tr>
<th>(\beta_{ij})</th>
<th>Algebraic Polynomial</th>
<th>Chebyshev Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\beta_{00})</td>
<td>-0.9997394418</td>
<td>-1.799539750</td>
</tr>
<tr>
<td>(\beta_{01})</td>
<td>-1.004922061</td>
<td>-0.850428742</td>
</tr>
<tr>
<td>(\beta_{02})</td>
<td>0.0196267987</td>
<td>0.05886470195</td>
</tr>
<tr>
<td>(\beta_{03})</td>
<td>0.1438350528</td>
<td>0.009013204415</td>
</tr>
<tr>
<td>(\beta_{10})</td>
<td>0.9999916901</td>
<td>2.000002648</td>
</tr>
<tr>
<td>(\beta_{11})</td>
<td>0.0001592185453</td>
<td>0.00000509398626</td>
</tr>
<tr>
<td>(\beta_{12})</td>
<td>-0.0004848490177</td>
<td>0.00000659726709</td>
</tr>
<tr>
<td>(\beta_{13})</td>
<td>0.0003582178876</td>
<td>0.000011194309</td>
</tr>
</tbody>
</table>

Thus, we obtain the following approximate formulas:

\[
    u_{d,m}^d(x,t) = -0.999739 - 1.004922 \quad t
\]
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 \( u^{approx}_{n,m}(x,t) \) which depends on relative error and running time with different values of \( n \) and \( m \). The error values \( E_{err}(x,t,\beta), V(x,t) \in \Omega_2 \) are also included by applying the formula (6).

<table>
<thead>
<tr>
<th>( (n,m) )</th>
<th>(13)</th>
<th>(15)</th>
<th>(17)</th>
<th>(19)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algebraic</strong></td>
<td><strong>Error</strong></td>
<td>0.00012972</td>
<td>2.79014780</td>
<td>3.224827632</td>
</tr>
<tr>
<td></td>
<td>( E_{err}(x,t,\beta) )</td>
<td>0.0008258</td>
<td>2.116914548</td>
<td>2.439781146</td>
</tr>
<tr>
<td></td>
<td>R.T./sec</td>
<td>1.638185</td>
<td>3.833933</td>
<td>7.520513</td>
</tr>
<tr>
<td><strong>Chebyshev</strong></td>
<td><strong>Error</strong></td>
<td>0.00012972</td>
<td>2.79014780</td>
<td>3.224827632</td>
</tr>
<tr>
<td></td>
<td>( E_{err}(x,t,\beta) )</td>
<td>0.0008258</td>
<td>2.116914548</td>
<td>2.439781146</td>
</tr>
<tr>
<td></td>
<td>R.T./sec</td>
<td>1.638185</td>
<td>3.833933</td>
<td>7.520513</td>
</tr>
</tbody>
</table>

**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 \( E_{err} \) 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 favor 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 \( (n+1)(n+1) \times (n+1)(n+1) \) very large, which need large memory of computer with too much time to compute it.

**References**


Appendix (A):
For instance, in equation (7) thus:

```matlab
A=zeros(q); % (n+1)*(m+1);
B=zeros(q,l); T2=zeros(n+1,m+1); % (H=2) for Chebyshev expansion
Tp=sym(zeros(n+1,m+1,q)); Q=zeros(q,l);
[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(f,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;
    end
end
```

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]):


A=double(TP(:,:,L),x,a1,b1); Q=double(AA';
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
    sx=0;sy=0;z1=floor(i1/2);
    st=0;s1=0;z2=floor(j1/2);
    tt=2^((x1-a22)/(b22-a22))-1;
    yy=2^((y1-a11)/(b11-a11))-1;
    if i1==0
      chebx=1; cheby=1;
    else
      for r1=0:z1
        sj=1;si=1;skx=1;sky=1;
        xx=2*((x1-a11)/(b11-a11))-1;
        ss=2^((s1-a22)/(b22-a22))-1;
        if j1==0
          cheby=1; cheb=1;
        else
          for r2=0:z2
            sj=1;si=1;skt=1;sks=1;
            tt=2*((t1-a22)/(b22-a22))-1;
            ss=2^((s1-a22)/(b22-a22))-1;
            if j1==0
              chebx=1; cheby=1;
            else
              for r2=0:z2
                sj=1;si=1;skt=1;sks=1;
                tt=2*((t1-a22)/(b22-a22))-1;
                ss=2^((s1-a22)/(b22-a22))-1;
                if j1==0
                  chebx=1; cheby=1;
                else
                  for r2=0:z2
                    sj=1;si=1;skt=1;sks=1;
                    tt=2*((t1-a22)/(b22-a22))-1;
                    ss=2^((s1-a22)/(b22-a22))-1;
                    if j1==0
                      chebx=1; cheby=1;
                    else
                      for r2=0:z2
...