Numerical and Simulation Methods for Solving of Non-Linear Fredholm Integro-Differential Equations

Mojtaba Moradi\textsuperscript{a}, Zaynab Ayati\textsuperscript{a}, MohammadAli Mirzazadeh\textsuperscript{a}

\textsuperscript{a}Faculty of Technology and Engineering East of Guilan, University of Guilan, Iran.

ABSTRACT
This paper introduces a new Monte Carlo algorithm for solving integro-differential equations. Finally, comparing the results of this algorithm with numerical Gauss-Legendre method is considered.

© 2014, Darbose. All rights reserved.

1. Introduction

Consider the following Fredholm integral equation:

\[ u(x) = f(x) + \int_{D} \ldots \int_{D} k(x, y_1, \ldots, y_m) \prod_{i=1}^{m} u(y_i) \prod_{i=1}^{m} dy_i \]  \hspace{1cm} (1.1)

where \( D \subset \mathbb{R} \) and \( m \) is a natural number greater than or equal to 2, \( f(x) \in L_2(D) \) and the kernel \( K(x, y_1, \ldots, y_m) \) belongs to \( L_2(D \times D \times \ldots \times D) \equiv L_2(D^{m+1}) \). It is assumed that this equation has an iterative solution corresponding to the iteration process:

\[ u_l(x) = f(x) + \int_{D} \ldots \int_{D} |K(x, y_1, \ldots, y_m)| \prod_{i=1}^{m} u_j(y_i) \prod_{i=1}^{m} dy_i, \quad l = 1, 2, \ldots \]  \hspace{1cm} (1.2)

\[ u_0(x) = f(x) \]

If the method of successive approximations

\[ u_l(x) = |f(x)| + \int_{D} \ldots \int_{D} |K(x, y_1, \ldots, y_m)| \prod_{i=1}^{m} u_j(y_i) \prod_{i=1}^{m} dy_i, \quad l = 1, 2, \ldots \]  \hspace{1cm} (1.3)

\[ u_0(x) = f(x) \]

Email addresses: mmoradi@guilan.ac.ir (Mojtaba Moradi), zainab.ayati@guilan.ac.ir (Zaynab Ayati), mirzazadehs2@guilan.ac.ir (MohammadAli Mirzazadeh)
converge, then with simulating the branching processes, we can establish an unbiased estimator for \( <g, u> \) i.e. its mathematical expectation is the inner product of \( g \) and \( u \):

\[
J(u) = <g, u> = \int_D g(x)u(x)dx, \quad \text{where } g(x) \in L_2(D).
\] (1.4)

With setting \( \delta(x - x_0) = g(x) \), \( \delta \) is Dirac delta function, in 1.4 we obtain \( u(x_0) \) \([1, 2, 7]\).

For explaining the branching process, suppose that a particle such as bacteria generates the same kind of particles as itself. Each of these generated particles, independent of other particles, either dies out or generates the new particles as the same kind of its mother particle. The initial particle, that is called zero seed and denote by \( x_0 \) generates \( m \geq 0 \) particles, where we show with \( x_{01}, x_{02}, \ldots, x_{0m} \). In the next generation, each of these particles acts as their parent particle. It means that either it generates \( m \geq 0 \) particles of the same kind of itself or dies out. We consider the following statistical structure for the above explained branching process.

Suppose that each particle starts its generation with the initial density function \( p_0(x) \geq 0 \) where \( x \in D \) and \( \int_D p_0(x)dx = 1 \). In the next step of generation, either this particle dies out with probability \( h(x_0) \) where \( 0 \leq h(x_0) < 1 \), or generates \( m \) particles \( x_{01}, x_{02}, \ldots, x_{0m} \), with probability \( p_m(x_0) = 1 - h(x_0) \). Suppose that \( p(x_0, x_{01}, x_{02}, \ldots, x_{0m}) \) be its transition probability where, for all \((x_0, x_{01}, x_{02}, \ldots, x_{0m}) \in D,

\[
p(x_0, x_{01}, x_{02}, \ldots, x_{0m}) \geq 0
\]

\[
\int_D \ldots \int_D p(x_0, x_{01}, x_{02}, \ldots, x_{0m}) \prod_{i=1}^{m} dx_{0i} = 1
\]

For simplify to distinguish of the elements of the branches we consider the following indices. If we show \( x_0 \) as the initial particle, then the next generated particles show by \( x_{01}, x_{02}, \ldots, x_{0m} \). Further, each generated particle behaves as the initial one, for the next branches. Each particle of these branches also has its mother index to identify it with other generated particles, with the same step. For example, \( x_{02} \) shows the second generated particle with its mother particle \( x_0 \). In the next generation of branches each particle has its parents indices for example \((01) \) plus an extra index such as \( i = 1, 2, \ldots, m \) see Figure 1.

Moreover, consider \( A \) be the all particles that can generate the same particles in the next steps and \( B \) denotes the all died particles of the above explained branches. In this paper, we employ the branching processes that each of its particles either can generate \( m \) particles or dies out. We present \( \Gamma \) as the
branching process. In this stage we want to construct the unbiased estimator of \(<g, u>\). If we have the only particle \(x_0\), we set 
\[\Theta_g(\Gamma) = \frac{g(x_0)}{p_0(x_0)}\]
and if the \(\Gamma\) consist other points, \(\Theta_g(\Gamma)\) constructed simultaneously with the construction of \(\Gamma\). When we have a transition from point \(x_t\) to \(x_{t1}, \ldots, x_{tm}\) we multiply \(\Theta_g(\Gamma)\) to 
\[K(x_t, x_{t1}, \ldots, x_{tm}) \over p(x_t, x_{t1}, \ldots, x_{tm})\]
otherwise if the particle \(x_t\) stay without generation, we multiply \(\Theta_g(\Gamma)\) to 
\[f(x_t) \over h(x_t)\]

The details of the above discussion is in the following theorem.

**Theorem:** Under the above assumptions we have:

\[E(\Theta_g(\Gamma)) = <g, u> = \int_D g(x)u(x)dx\]

where

\[\Theta_g(\Gamma) = \frac{g(x_0)}{p_0(x_0)} \prod_{x_t \in A} \frac{K(x_t)}{p(x_t)} \prod_{x_t \in B} \frac{f(x_t)}{h(x_t)}\]

with

\[p(\Gamma) = p_0(x_0) \prod_{x_t \in A} p(x_t) \prod_{x_t \in B} h(x_t)\]

and \(K(x_t)\), denotes

\[K(x_t) = K(x_t, x_{t0}, x_{t1}, \ldots, x_{tm-1})\]

and

\[p(x_t) = p_m(x_t)p(x_t, x_{t0}, x_{t1}, \ldots, x_{tm-1}).\]

**Proof:** Suppose that the sub tree \(\gamma\) is fixed and it has \(n_1 + n_2 + 1\) points, such that \(x_0, \ldots, x_{n_1}\) and \(x_{n_1+1}, \ldots, x_{n_1+n_2}\). It is clear that this can not be considered as a restriction of the generality. Consider the following density function

\[p_\gamma = p_\gamma(x_0, x_1, \ldots, x_{n_1}, x_{n_1+1}, \ldots, x_{n_1+n_2}) = p_0(x_0) \prod_{i=0}^{n_1} p(x_i) \prod_{i=1}^{n_2} h(x_{n_1+i})\]

and the probability to obtain sub tree \(\gamma' = \delta\gamma\) is \(P(\gamma' = \delta\gamma) = \int_D \cdots \int_D p_\gamma dx_0 \cdots dx_{n_1+n_2}\).

The random variable \(\Theta_g(\gamma) = g(x_0) \prod_{x_t \in A} K(x_t) \prod_{x_t \in B} f(x_t) / h(x_t)\) has the following condition density function

\[p_\gamma(x_0, x_1, \ldots, x_{n_1+n_2} | \gamma = \gamma') = \frac{p_\gamma(x_0, x_1, \ldots, x_{n_1+n_2})}{p(\gamma = \gamma')}\]

where \(K(x_t) = k(x_t, x_{t0}, x_{t1}, \ldots, x_{t,m-1})\) and \(p(x_t) = p_m(x_t)p(x_t, x_{t0}, x_{t1}, \ldots, x_{t,m-1})\).

Now, we calculate the mathematical expectation:

\[\Phi \text{ Darbose}\]
\[ E(\Theta_g(\gamma)) = E(\Theta_g(\gamma')|\gamma = \gamma') = \int_D \cdots \int_D \Theta_g(\gamma)p_{\gamma'}(x_0, x_1, \ldots, x_{n_1+n_2} | \gamma = \gamma')dx_0 \cdots dx_{n_1+n_2} \]
\[ = \int_D \cdots \int_D g(x_0) \prod_{t=0}^{n_1} K(x_t) \prod_{t=1}^{n_2} \frac{f(x_{n_1+t})}{p(\gamma = \gamma')} \frac{dx_0 \cdots dx_{n_1+n_2}}{\frac{<g,u_1>}{p(\gamma = \gamma')}} \]  
(1.5)

where
\[ u_1^t = u_1^t(x_0) = \int_D \cdots \int_D g(x_0) \prod_{t=0}^{n_1} K(x_t) \prod_{t=1}^{n_2} f(x_{n_1+t})dx_0 \cdots dx_{n_1+n_2} \]

One can choose \( n \) sub trees of the full tree \( \Gamma \). Note that some of the sub trees may be chosen many times, since \( n \) is a large number. The \( n_\gamma \) is the number of the trees which correspond to \( \gamma \). The following expression holds:
\[ \frac{n_\gamma}{n} \approx \frac{p(\gamma = \gamma')}{n} \]

On the other hand
\[ <u_1^t> \approx \frac{p(\gamma = \gamma')}{n_\gamma} \sum_{t=1}^{n_\gamma} (\Theta_g(\gamma))_t \approx \frac{1}{n} \sum_{t=1}^{n_\gamma} (\Theta_g(\gamma))_t \]  
(1.6)

Clearly
\[ E(\Theta_g(\gamma)) = \sum_{t=0}^{N} E(\Theta_g(\gamma')|\gamma = \gamma')p(\gamma = \gamma') \]

where \( N \) is the number of all sub trees of the \( \Gamma \). Using 1.5 one can get
\[ E(\Theta_g(\gamma')) = \sum_{t=0}^{N} <g, u_1^t> = <g, u_1> \]

This completes the proof of the theorem.

We note that if \( l \to \infty \), then the mathematical expectation of the random variable is:
\[ \lim_{l \to \infty} E(\Theta_g(\gamma')) = <g, u> = \int_D g(x)u(x)dx \]  
(1.7)

where \( u(x) \) is the solution of 1.1.

Besides, we propose with probability one that all the branches have a finite number of generations and the arithmetic mean over particles which are born in any generation is also finite. If we simulate \( n \) branches \( \Gamma \) as above explained, we can estimate \( u(x_0) \), with taking the average of estimation of \( u(x_0) \) as Monte Carlo solution where \( g(x) = \delta(x - x_0) \).

If we consider the probability transition probability of Markov chain determined in our branching processes as:
\[ p(x, y_1, y_2, \ldots, y_m) = \frac{|k(x, y_1, y_2, \ldots, y_m)|}{\sum_{y_1, \ldots, y_m} |k(x, y_1, y_2, \ldots, y_m)|dy_1 \cdots dy_m} \]  
(1.8)

with initial transition probability \( p_0 = \frac{|g(x)|}{\int |g(x)|dx} \) reduces the variance and is called the Almost Optimal density function for employed Markov chains [4].

Darbose
2. Integro-Differential Equations

In this section, we develop the above algorithm for solving Fredholm integro-differential equations. We focus our discussion on the integro-differential equations with separable kernel \( k(x, y_1, y_2, \ldots, y_m) \) i.e. we can write the kernel \( k(x, y_1, y_2, \ldots, y_m) \) as a finite summation as below:

\[
K(x, y_1, y_2, \ldots, y_m) = \sum_{j=1}^{n} v_j(x)w_j(y_1, y_2, \ldots, y_m)
\]  

(2.1)

It is trivial that we can covert each non-separable kernel to a separable kernel using Taylor extension[8]. Without losing the generality of discussion we consider those kernels that strictly determined by a phrase as follow and finally we extend the results to other cases:

\[
K(x, y_1, y_2, \ldots, y_m) = v(x)w(y_1, y_2, \ldots, y_m)
\]  

(2.2)

It is well-known that the standard form of the Fredholm integro-differential equations in this non-linear case is:

\[
u^{(n)}(x) = f(x) + \int_{0}^{1} \ldots \int_{0}^{1} K(x, y_1, \ldots, y_m) \prod_{i=1}^{m} u(y_i) \prod_{i=1}^{m} d(y_i)
\]  

(2.3)

where \( u^{(k)}(0) = b_k, 0 \leq k \leq n - 1 \), \( u^{(n)}(x) \) shows the \( n \)th derivative of the function \( u(x) \) and \( b_k, 0 \leq k \leq n - 1 \) are real constants. Using 2.2 in 2.3 we have

\[
u^{(n)}(x) = f(x) + v(x) \int_{0}^{1} \ldots \int_{0}^{1} w(y_1, y_2, \ldots, y_m) \prod_{i=1}^{m} u(y_i) \prod_{i=1}^{m} d(y_i)
\]  

(2.4)

With obtaining the integration of 2.4 from 0 to \( x \), \( n \) times, we have:

\[
u(x) = b_0 + b_1 x + \frac{1}{2!} b_2 x^2 + \ldots + \frac{1}{(n-1)!} b_{n-1} x^{n-1} + \int_{0}^{x} \int_{0}^{x} \ldots \int_{0}^{x} f(x_n) dx_{n-1} \ldots dx_1 \\
+ \left( \int_{0}^{x} \int_{0}^{x} \ldots \int_{0}^{x} w(x_n) dx_{n-1} \ldots dx_1 \right) \int_{0}^{1} \int_{0}^{1} \ldots \int_{0}^{1} u(y_1, y_2, \ldots, y_m) \prod_{i=1}^{m} d(y_i)
\]

If we set,

\[
A(x) = \int_{0}^{x} \int_{0}^{x_1} \ldots \int_{0}^{x_{n-1}} f(x_n) dx_{n-1} \ldots dx_1 \\
B(x) = \int_{0}^{x} \int_{0}^{x_1} \ldots \int_{0}^{x_{n-1}} v(x_n) dx_{n-1} \ldots dx_1 \\
F(x) = A(x) + b_0 + b_1 x + \frac{1}{2!} b_2 x^2 + \ldots + \frac{1}{(n-1)!} b_{n-1} x^{n-1} \\
K(x, y_1, y_2, \ldots, y_m) = B(x) w(y_1, y_2, \ldots, y_m)
\]

We rewrite the above equation as:

\[
u(x) = F(x) + \int_{D} \ldots \int_{D} K(x, y_1, y_2, \ldots, y_m) \prod_{i=1}^{n} u(y_i) \prod_{i=1}^{n} d(y_i)
\]  

(2.5)

Darbose
Now, we can use the explained Monte Carlo method for solving the 2.5. The advantage of Monte Carlo method for solving 2.3 is that we do not need to have the strict form of the \( A(x) \) and \( B(x) \) functions, since we just need to have the functions \( K(x, y_1, y_2, ..., y_m) \) and \( f(x) \) at \( x_0 \). Specially, when we do not have \( A(x) \) and \( B(x) \) we can use the Monte Carlo method (alternatively other numerical methods) to obtain the values of \( A(x_0) \) and \( B(x_0) \). This is the advantage of the Monte Carlo methods since it has a fast estimation of the desired solution, especially when we can not use classical methods [2].

In below we describe the Monte Carlo algorithm for calculating of the \( u(\xi) \) where \( u(x) \) is the solution of 2.3.

**Algorithm**

1. Take the \( p_m(x) \) from user.
2. Set
   \[
   A(x) = \int_0^x \int_0^{x_1} \cdots \int_0^{x_{n-1}} f(x_n) dx_n ... dx_2 dx_1 \\
   B(x) = \int_0^x \int_0^{x_1} \cdots \int_0^{x_{n-1}} v(x_n) dx_n ... dx_2 dx_1 \\
   p(x, y_1, y_2, ..., y_m) = \frac{|v(x)w(y_1, ..., y_m)|}{\prod_{i=1}^m dy_i} \\
   D(x) = b_0 + b_1 x + \frac{b_2}{2!} x^2 + ... + \frac{b_{n-1}}{(n-1)!} x^{n-1}
   \]
3. Set \( \Theta(\Gamma) = 1 \).
4. Generate a random number \( \alpha \in [0, 1] \) with uniform probability density function, and if \( \alpha \leq p_m(\xi) \) go to 6, else go to next step.
5. Obtain
   \[
   \Theta(\Gamma) = \Theta(\Gamma) \times \frac{A(\xi) + D(\xi)}{1 - p_m(\xi)}.
   \]
   In this step, we say that the point \( \xi \) dies out.
6. Generate \( m \) random numbers \( \xi_1, ..., \xi_m \), with probability density 1.8 and obtain:
   \[
   \Theta(\Gamma) = \Theta(\Gamma) \times \frac{v(\xi)w(\xi_1, ..., \xi_m)}{p_m(\xi)p(\xi_1, ..., \xi_m)}
   \]
   then repeat the steps 4,5 and 6 for \( \xi, \xi_1, ..., \xi_m \) consequently
7. Stop the algorithm whenever the all points die out.

### 3. Numerical Method for Integral equation

In numerical methods for evaluating the integral \( \int_a^b f(x) dx \), we use the following approximation

\[
\int_a^b f(x) dx \approx \sum_{i=0}^n w_i f(x_i) \tag{3.1}
\]

Darbouse
where \( x_i \in (a, b) \) and \( w_i \) is real constant. In Fredholm non linear integral equation 1.1 or 1.3, we employ the following equation

\[
\begin{align*}
    u(x) &= f(x) + \sum_{i_1=0}^{n} w_{i_1} \sum_{i_2=0}^{n} w_{i_2} \cdots \sum_{i_m=0}^{n} w_{i_m} K(x, y_{i_1}, \ldots, y_{i_m}) \prod_{j=1}^{m} u(y_{i_j})
\end{align*}
\]  

(3.2)

If we set \((n+1)\) points, \(x_i, i = 0, 1, \ldots, n\) in the above equation, we have

\[
\begin{align*}
    u(x_i) &= f(x_i) + \sum_{i_1=0}^{n} w_{i_1} \sum_{i_2=0}^{n} w_{i_2} \cdots \sum_{i_m=0}^{n} w_{i_m} K(x_i, y_{i_1}, \ldots, y_{i_m}) \prod_{j=1}^{m} u(y_{i_j})
\end{align*}
\]  

(3.3)

If we select all points of the upper and lower of limits of the integrals, i.e. \(i_j, j = 1, \ldots, n\) with the same procedure and \(x_i, i = 1, \ldots, n\) is selected on these points, then we reach to a non linear system of equation involving \((n+1)\) unknown variables in \((n+1)\) equation. In numerical method by solving these systems, we estimate the unknown \(x_i, i = 1, \ldots, n\) for approximating the integral 2.5. We note that using such methods such as Trapezoidal or Simpson rule we need the values of the integro function on subintervals with equal length. Also, in Gauss-Legendre method the corresponding integral should be converted from \([a, b]\) to \([-1, 1]\) and the values of \(x_i\) and \(w_i\) are known.

4. Numerical Results

For the numerical tests of the above algorithm, we consider the following Fredholm integral differential equation:

\[
\begin{align*}
    u(x) &= -36x^3 + 4 \int_{0}^{1} \int_{0}^{1} x^3yz u(y)u(z)dydz
\end{align*}
\]  

(4.1)

with initial conditions \(u(0) = 6\), has the unique solution: \(u(x) = 6\).

If we rewrite this equation as 2.5, we have

\[
\begin{align*}
    u(x) &= 6 - 9x^4 + 4 \int_{0}^{1} \int_{0}^{1} x^4yz u(y)u(z)dydz
\end{align*}
\]  

(4.2)
Table 1: Comparing Monte Carlo and Gauss-Legendre solutions, where exact solution is equal to 6.

<table>
<thead>
<tr>
<th>n</th>
<th>$x_i$</th>
<th>Gauss-Legendre solution</th>
<th>Monte Carlo Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-0.577350</td>
<td>6.0071204374</td>
<td>5.9556</td>
</tr>
<tr>
<td></td>
<td>+0.577350</td>
<td>7.3813236910</td>
<td>5.9906</td>
</tr>
<tr>
<td></td>
<td>-0.774597</td>
<td>5.9999997096</td>
<td>5.9835</td>
</tr>
<tr>
<td></td>
<td>+0.774597</td>
<td>5.9998844254</td>
<td>5.8862</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>6.00000000003</td>
<td>6.1201</td>
</tr>
<tr>
<td></td>
<td>-0.861136</td>
<td>6.0000001667</td>
<td>6.0038</td>
</tr>
<tr>
<td></td>
<td>+0.861136</td>
<td>6.0000028316</td>
<td>5.9766</td>
</tr>
<tr>
<td>4</td>
<td>-0.906180</td>
<td>5.99999999994</td>
<td>5.8246</td>
</tr>
<tr>
<td></td>
<td>-0.538469</td>
<td>5.9999996998</td>
<td>6.0489</td>
</tr>
<tr>
<td></td>
<td>+0.538469</td>
<td>5.999993843</td>
<td>6.0160</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>5.99999629384</td>
<td>5.9977</td>
</tr>
<tr>
<td></td>
<td>+0.906180</td>
<td>5.99999126576</td>
<td>5.8558</td>
</tr>
<tr>
<td>5</td>
<td>-0.932469</td>
<td>5.99999999996 ± .33088705079 e-7 i</td>
<td>5.8450</td>
</tr>
<tr>
<td></td>
<td>-0.661209</td>
<td>5.9999997549 ± .20960155836 e-4 i</td>
<td>5.9652</td>
</tr>
<tr>
<td></td>
<td>-0.238619</td>
<td>5.99999937489 ± .53465691481 e-3 i</td>
<td>5.9992</td>
</tr>
<tr>
<td></td>
<td>+0.238619</td>
<td>5.99999562178 ± .37447083479 e-2 i</td>
<td>5.9952</td>
</tr>
<tr>
<td></td>
<td>+0.661209</td>
<td>5.99998583412 ± .12116136281 e-1 i</td>
<td>6.0030</td>
</tr>
<tr>
<td></td>
<td>+0.932469</td>
<td>5.99997405840 ± .22187950667 e-1 i</td>
<td>5.8795</td>
</tr>
<tr>
<td>6</td>
<td>-0.949108</td>
<td>5.99999999925</td>
<td>6.3341</td>
</tr>
<tr>
<td></td>
<td>-0.741531</td>
<td>5.9999949855</td>
<td>5.8550</td>
</tr>
<tr>
<td></td>
<td>-0.405845</td>
<td>5.9998599798</td>
<td>5.9906</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>5.9988764515</td>
<td>6.0069</td>
</tr>
<tr>
<td></td>
<td>+0.405845</td>
<td>5.9956112426</td>
<td>6.0094</td>
</tr>
<tr>
<td></td>
<td>+0.741531</td>
<td>5.986648532</td>
<td>5.9179</td>
</tr>
<tr>
<td></td>
<td>+0.949108</td>
<td>5.9837843103</td>
<td>5.8793</td>
</tr>
<tr>
<td>7</td>
<td>-0.960290</td>
<td>5.99999999994 ± .395609227203 e-8 i</td>
<td>6.2472</td>
</tr>
<tr>
<td></td>
<td>-0.796666</td>
<td>5.99999996388 ± .271961157620 e-5 i</td>
<td>6.0243</td>
</tr>
<tr>
<td></td>
<td>-0.525532</td>
<td>5.999999982906 ± .806296125963 e-4 i</td>
<td>6.0511</td>
</tr>
<tr>
<td>8</td>
<td>-0.813435</td>
<td>5.999999060491 ± .707342753172 e-3 i</td>
<td>5.9991</td>
</tr>
<tr>
<td></td>
<td>-0.183435</td>
<td>5.999995855087 ± .312064596965 e-2 i</td>
<td>5.9917</td>
</tr>
<tr>
<td></td>
<td>+0.525532</td>
<td>5.999988554807 ± .861692180914 e-2 i</td>
<td>6.0406</td>
</tr>
<tr>
<td></td>
<td>+0.796666</td>
<td>5.99977980512 ± .165781575095 e-1 i</td>
<td>6.0116</td>
</tr>
<tr>
<td></td>
<td>+0.960290</td>
<td>5.99968795364 ± .23435238280 e-1 i</td>
<td>5.6664</td>
</tr>
</tbody>
</table>
In the computational parts of this example to comparing the numerical results with Monte Carlo solution, we selected the Gauss-Legendre method which is one of the most accurate numerical methods for obtaining the solution of integral equations.

To estimate the value of $u(x)$ by Gauss-Legendre method for each $n \leq 8$, with initial probability density function $p_2(x) = \frac{x^2+1}{8}$, we simulate $N=3000$ random branches by Monte Carlo method.

Table (1) presents the solution of integro-differential equation 4.1 by two numerical Gauss-Legendre and Monte Carlo methods. When $n = 6$ the approximate solution given by Gauss-Legendre method is complex. It is important to know that for solving non-linear systems of the corresponding integral equation, we used MATLAB software, without involving to the concept of this method.

We note that for $n \leq 5$, the Gauss-Legendre method has better results than the Monte Carlo method, but for $n = 6$ and $n = 8$ the Gauss-Legendre method produces the complex solution. It seems that with increasing the dimension this will happen for the solution non-linear systems. One of the most important performances of the Monte Carlo method in comparison to Gauss-Legendre method evaluating 2.3 is that the $x_i$ using in 3.3 is applicable under some condition and 3.3 can not be used for every $x_i$. In fact using $x_i$ in Gauss-Legendre method is just fixed points. But in Monte Carlo method we do not have this restriction. For example the solution of integro-differential equation 4.1 at $x = \frac{\pi}{8}$ by Monte Carlo method, as we presented its computational results in figure (2).

5. Conclusion

The presented Monte Carlo algorithm for solving integro-differential equations has more accurate results than the Gauss-Legendre method. Specially, the computational results prove that the given by Gauss-Legendre method are produce the false results for some points, but the Monte Carlo method is still stable to have suitable results.

References