On Computational Efficiency of Nine Iterative Algorithms for Hammerstein Equations

Keywords: Fredholm Integral equations, Picard Iteration, Ishikawa iteration, Argawal iteration, Newton-Raphson method, Nonstandard Trapezoidal Rule.

Abstract

Many fixed point iterative methods have been proposed. Their authors provide, theoretically, the rate of convergence which indicates how much the error changes after one iteration step. Of practical importance are the CPU time required by each method and their accuracy. This has never been investigated. In this work we investigate nine iterative processes and compare their CPU time and accuracy. The schemes are the Picard, Mann, Ishikawa, Noor, Argawal, Abbas-Nazir, Thakur, Ullah, and S* methods. By applying a recently proposed fourth order quadrature rule to the integral equation and replacing the analytical operator with the discretized one, the iterative schemes are derived. They are tested on five Hammerstein equations. The results show that (i) all the methods converge to the exact solution at the correct discretization order of convergence, (ii) the Picard scheme is the fastest and also has good accuracy, and (iii) the S* and Abbas-Nazir schemes are the least efficient. It is concluded that for a contractive problem the Picard scheme is sufficient but if the operator is non-expansive, then the Mann scheme is sufficient.

2020 Mathematics Subject Classifications: 65R20, 45D05.

1 Introduction

In mathematics and its applications, it is common to seek the solution of problems of the form:

$$u(x) = (Tu)(x), \tag{1}$$

where u is an unknown function of $x \in [a, b] \subset \mathbb{R}$ and T is an operator which may be differential, integral, intro-differential, or partial differential operator. The problem (1) is called a fixed point problem, and the solution is called a fixed point of T. Under suitable assumptions on T the solution can be found using appropriate methods. We are interested in the case where T is a contraction for which the problem has a unique solution. Several iterative methods have been proposed to find the fixed point.

For example, the Picard iteration [1] is a very important method that has been widely investigated and applied to several problems. In the case of problems involving non-expansive operators in which the Picard iteration may not converge [2, 3], the Mann [4] and Krasnoselkij [5] schemes also exist to compute the fixed points. There are other methods, including the Ishikawa [6], which is a two-step method with two operator evaluations. Another two-step method is that of Argawal and collaborators [7]. This method is represented as if it has three operator evaluations but it actually has only two. This is why recent study [8] has found that the schemes of Argawal and coworkers and that of Ishikawa have approximately equal efficiency and even accuracy. Noor [9] proposed a three-step method with three operator evaluations, see also [10] for another such method. In 2016, Thakur and colleagues [11] developed another three-step method with three operator evaluations, while [12] proposed a three-step method with four operator evaluations. Hybrid iterative processes are also investigated in [13], while the so called S* method which involves four steps and eight operator evaluations is proposed in [3].

The authors of the above methods also prove their theoretical rate of convergence which gives an indication of how much the error changes after each iteration step. With these, they claim that their methods are 'faster' than the earlier methods. Bearing in mind that what actually matters in practice is the CPU time taken to arrive at a desired accuracy level, not necessarily the number of steps, we are interested to measure the actual CPU time of some of these iterative methods. This is the main goal of this study - to assess the methods for accuracy and efficiency. We will focus on the iterative methods developed in [1, 4, 6, 7, 9, 10, 11, 12] and [3].

We will derive discrete versions of those iterative schemes and apply them to solve nonlinear Hammerstein integral equations of the second kind for which T is of the form

$$Tu(x) := \int_a^b p(x, y)q(y, u(y))dy, \tag{2}$$

where $p, q : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. Integral equations have several applications like in physical sciences [14, 15, 16], finance [17], antenna wire modelling [18, 19], image processing [20], heat transfer [21], physical sciences [14, 15, 16], optimal control and economics [22, 23, 24, 25], epidemiology [26, 27], among others [28, 29]. These equations have been investigated theoretically, see [30]

on mixed Volterra-Fredholm equations,[31, 32, 33, 34] on functional integral equations, and [35, 36] on Volterra equations.

Integral equations have also been investigated numerically, see CAS wavelets [37, 38], collocation methods [39, 40], variational iteration methods [41, 42, 43, 44], Linear programming [45], Taylor series [46], and Trepezoidal-Newton type method [47, 36, 48]. See also [49, 50, 26, 27] and [51]. Recently, there have been growing interest in using fixed point methods to solve integral equations, see [29, 52, 53, 54, 55, 56] for example. In [8], three fixed point methods were compared with Newton method for solving nonlinear Fredholm equations.

The present work extends the work in [8] by investigating the performance of nine fixed point methods with regards to their computational costs on five nonlinear integral equations. In section 2, we present the nine discrete fixed point iteration schemes for solving (1)-(2). Numerical experiments are presented and discussed in section 3, conclusions are made in section 4.

2 Nine Discrete Fixed Point Iterative Schemes

Let $N \in \mathbb{Z}^+, N \geq 3$, $h := \frac{b-a}{N-1}, x_i = a+ih : i = 0, 1, \cdots, N$, and $\Omega_h = \{x_i = a + ih : i = 0, 1, \cdots, N\}$. Define the grid functions

$$\eta_j^N = \begin{cases} \frac{3h}{8}, & \text{if } j = 0, N, \\ \frac{7h}{6}, & \text{if } j = 1, N-1, \\ \frac{23h}{24}, & \text{if } j = 2, N-2, \\ h, & \text{if } 2 < j < N-2, \\ 0, & \text{otherwise} \end{cases}$$

and

$$u_{n,i} \approx u_n(x_i),\ g_i = g(x_i),\ \text{for all } i,$$

where $u_n(x)$ is the approximation of $u(x)$ after the n-th iteration of a given fixed point iterative process. At each iteration level, n and for the grid

function u_n, we define the grid function

$$\begin{aligned} H_i^{u_n} =& g_i + \frac{3h}{8}p(x_i,x_0)q(x_0,u_{n,0}) + \frac{7h}{6}p(x_i,x_1)q(x_1,u_{n,1}) \\ &+ \frac{23h}{24}p(x_i,x_2)q(x_2,u_{n,2}) + h\sum_{j=3}^{N-3} p(x_i,x_j)q(x_j,u_{n,j}) \\ &+ \frac{23h}{24}p(x_i,x_{N-2})q(x_{N-2},u_{n,N-2}) \\ &+ \frac{7h}{6}p(x_i,x_{N-1})q(x_{N-1},u_{n,N-1}) \\ &+ \frac{3h}{8}p(x_i,x_N)q(x_N,u_{n,N}) \\ &= \sum_{j=0}^{N} \eta_j^N p(x_i,x_j)q(x_j,u_{n,j}). \end{aligned} \tag{3}$$

It can be shown that $H_i^{u_n}$ is a fourth-order approximation of $Tu_n(x)$ at $x_i \in \Omega_h$, see [8]. From the definition of $H_i^{u_n}$, we obtain H_i^g as

$$\begin{aligned} H_i^g =& g_i + \frac{3h}{8}p(x_i,x_0)q(x_0,g_0) + \frac{7h}{6}p(x_i,x_1)q(x_1,g_1) \\ &+ \frac{23h}{24}p(x_i,x_2)q(x_2,g_2) + h\sum_{j=3}^{N-3} p(x_i,x_j)q(x_j,g_j) \\ &+ \frac{23h}{24}p(x_i,x_{N-2})q(x_{N-2},g_{N-2}) \\ &+ \frac{7h}{6}p(x_i,x_{N-1})q(x_{N-1},g_{N-1}) \\ &+ \frac{3h}{8}p(x_i,x_N)q(x_N,g_N) \\ &= \sum_{j=0}^{N} \eta_j^N p(x_i,x_j)q(x_j,g_j) \end{aligned} \tag{4}$$

which is obtained by replacing u_n with g in (3). For example $u_{n,j}$ is replaced with g_j for any $j = 0, 1, \cdots, N$.

In what follows we set, [6]:

$$\alpha_n = \beta_n = \gamma_n = \frac{1}{\sqrt{n}}.$$

We are now ready to derive the iterative schemes.

2.0.1 The Schemes

The numerical scheme corresponding to each iterative method is derived by replacing the analytical operator $Tu_n(x)$ in the method with the discretized

form of it, namely $H_i^{u_n}$. This leads to the following discrete versions of the nine iterative methods.
The discrete Picard scheme [1, 57] is given as

$$u_{n+1,i} = H_i^{u_n}, \qquad \text{for } n \geq 0. \tag{5}$$

The Mann scheme [4] is

$$u_{n+1,i} = (1-\alpha_n)u_{n,i} + \alpha_n H_i^{u_n}, \qquad \text{for } n \geq 0. \tag{6}$$

The Ishikawa scheme is, [6]

$$\begin{aligned} u_{n+1,i} &= (1-\alpha_n)u_{n,i} + \alpha_n H_i^{v_n}, \\ v_{n,i} &= (1-\beta_n)u_{n,i} + \beta_n H_i^{u_n}, \qquad \text{for } n \geq 0. \end{aligned} \tag{7}$$

The scheme derived from iterative process of Argawal and colleagues [7] is:

$$\begin{aligned} u_{n+1,i} &= (1-\alpha_n)H_i^{u_n} + \alpha_n H_i^{v_n}, \\ v_{n,i} &= (1-\beta_n)u_{n,i} + \beta_n H_i^{u_n}, \qquad \text{for } n \geq 0. \end{aligned} \tag{8}$$

The Noor scheme [9] is:

$$\begin{aligned} u_{n+1,i} &= (1-\alpha_n)u_{n,i} + \alpha_n H_i^{v_n}, \\ v_{n,i} &= (1-\beta_n)u_{n,i} + \beta_n H_i^{w_n}, \\ w_{n,i} &= (1-\gamma_n)u_{n,i} + \gamma_n H_i^{u_n}, \qquad \text{for } n \geq 0. \end{aligned} \tag{9}$$

The scheme derived from the iterative process of Abbas and Nazir [10] is given by

$$\begin{aligned} u_{n+1,i} &= (1-\alpha_n)H_i^{v_n} + \alpha_n H_i^{w_n}, \\ v_{n,i} &= (1-\beta_n)H_i^{u_n} + \beta_n H_i^{w_n}, \\ w_{n,i} &= (1-\gamma_n)u_{n,i} + \gamma_n H_i^{u_n}, \qquad \text{for } n \geq 0. \end{aligned} \tag{10}$$

From the iterative process of Thakur and coworkers [11], we derive the scheme

$$\begin{aligned} u_{n+1,i} &= (1-\alpha_n)H_i^{w_n} + \alpha_n H_i^{v_n}, \\ v_{n,i} &= (1-\beta_n)w_{n,i} + \beta_n H_i^{w_n}, \\ w_{n,i} &= (1-\gamma_n)u_{n,i} + \gamma_n H_i^{u_n}, \qquad \text{for } n \geq 0. \end{aligned} \tag{11}$$

The iterative process of Ullah and Arshad [12] leads to the following scheme:

$$\begin{aligned} u_{n+1,i} &= H_i^{v_n} \\ v_{n,i} &= H_i^{(1-\alpha_n)w_n + \alpha_n H^{w_n}}, \\ w_{n,i} &= (1-\beta_n)u_{n,i} + \beta_n H_i^{u_n}, \qquad \text{for } n \geq 0. \end{aligned} \tag{12}$$

Finally, the S* iterative process [3] leads to the scheme:

$$\begin{aligned} u_{n+1,i} &= H_i^{(1-\alpha_n)v_n+\alpha_n H^{v_n}}, \\ v_{n,i} &= H_i^{(1-\beta_n)w_n+\beta_n H^{w_n}}. \\ w_{n,i} &= H_i^{(1-\gamma_n)x_n+\gamma_n H^{x_n}} \\ x_{n,i} &= H_i^{(1-\eta_n)u_n+\eta_n H^{u_n}}, \qquad \text{for } n \geq 0. \end{aligned} \tag{13}$$

Remark 2.1. *Each of the schemes in* (5)-(13) *above is initialized as follows*

$$u_{0,i} = H_i^g, \text{ for all } i, \tag{14}$$

where H_i^g *is given in* (4).

These schemes are implemented in a python code and their accuracy and efficiency are assessed in the next section.

3 Numerical Experiments

Several examples are now provided to demonstrate the convergence of the algorithms presented above and compare their accuracy and efficiencies. Errors are computed in maximum norm and experimental order of convergence are computed according to the formula in [53, 56], see also [58] for application of the idea. The convergence results of the Picard, Mann and Ishikawa schemes are displayed in Table 1, those of Argawal, Noor and Abbas-Nazir are displayed in Table 2 and those of Thakur, Ullah and S-star are in shown in Table 3. The results in these three Tables are computed for example 1 below. One can see that all the nine algorithms converge to the exact solution with fourth-order of accuracy which corresponds to the theoretical order of accuracy of the quadrature rule implemented above. Another remarkable observation is that all the methods have comparable accuracy; none has a significantly better or worse accuracy than the others. This confirms that the algorithms are correctly formulated and implemented, hence can be used to solve more problems and also compare their efficiency.

The efficiency and errors computed by each method for the problems on grids with $5, 20, 100, 120$ and 150 points are tabulated in Tables 4-8. Each example is solved 50 times by each method and the average CPU time and computed error are recorded. The 50 times run is indicated in the Tables under the column name, "**No. of Runs**". The "**Picard Factor**" of a method is the ratio of the average CPU time of the method to that of the Picard method for the same problem and over equal number of runs. This indicates how much the method is faster or slower than the Picard scheme.

Example 1

This example is derived from the method of manufactured solutions (MMS) [59, 60, 61]:

$$p(x,y) = \frac{xy^3}{1+x^2},\ q(y,u) = y^2 + sin(u),$$
$$g(x) = \frac{-x/3 + (1+x^2)\sin^{-1}(x^2)}{1+x^2},\ x \in [0,1]$$

which has the exact solution $u(x) = \sin^{-1}(x^2)$.

Table 4 shows the efficiency and errors for this example. The following can be observed from this Table,

(a) on a grid of 5 points, (i) all the nine methods computed the solution with equal error of 3.98598×10^{-3}. (ii) the Picard scheme is the most efficient, followed by the Mann/Krasnoselskij scheme, then Argawal and Ishikawa schemes respectively. (iii) The S-star scheme is the least efficient, followed by the Abbas-Nazir and Thakur schemes in that order.

(b) On 20-points grid, (i) the Picard and Mann schemes computed the solution with slightly smaller error of 1.20598×10^{-5}, while other methods computed with 1.20599×10^{-5}. (ii) the Picard, Mann and Ishikawa schemes are the most efficient, in that order, while S-star, Abbas-Nazir and Noor are the least efficient methods in this case.

(c) on a grid of 100 points, (i) the Picard and Mann schemes also computed slightly better accuracy of 1.75732×10^{-8} while the others computed the solution with an error of 1.75819×10^{-8}. They also had better accuracy on the finer meshes of 120 and 150 mesh points. (ii) the Picard and Mann schemes also took the least CPU times, while S-star, Abbas-Nazir and Ullah schemes took the most CPU times.

(d) We can summarize the finding from this example by concluding that the Picard scheme, followed by the Mann scheme performed best, whereas the S-star and Abbas-Nazir schemes performed least for this example.

Example 2

This example is consrtucted from MMS [62, 63, 64]

$$p(x,y) = x^2y^2,\ q(y,u) = \frac{u^4}{1+y^2},$$
$$g(x) = x^2(x - 36979/45045 + \pi/4),\ x \in [0,1]$$

which has the exact solution $u(x) = x^3$.

The performance of the methods on example 2 is tabulated in Table 5. The following can be observed:

(a) on all the grids, with 5, 20, 100, 120, 150 grid points, (i) the Picard and Mann schemes performed best in terms of CPU times. (ii) The S-star and Abbas-Nazir schemes performed least.

(b) on grids with 100, 120 and 150 grid points, the Picard and Mann schemes are slightly more accurate than the others.

Example 3

The third example is a general Fredholm equation:

$$u(x) = g(x) + \int_0^1 k(x, y, u(y))\, dy,$$

where

$$\begin{aligned} k(x, y, u) &= \frac{x^2 + y^2 + u^3/3 - u^2}{12}, \\ g(x) &= -x^2/12 + \sqrt(2)\sin(x + pi/4) - \sin(1)/24 + \cos(3)/216 \\ &\quad + \sin(3)/216 - \cos(2)/24 + \cos(1)/24 + 11/216,\ x \in [0, 1] \end{aligned}$$

This has exact solution as $u(x) = \sin(x) + \cos(x)$.
Table 6 records the results computed for example 3. The following are observed:

(a) on all the grids, the Picard and Mann schemes consumed the least CPU time while S-star and Abbas-Nazir schemes took most CPU times,

(b) on the grid with 5 points the Picard and Mann schemes computed slightly higher error, but

(c) on all the finer grids, with 20, 100, 120, 150 mesh points, the Picard and Mann schemes had better accuracy than the others.

Example 4

Next, we consider, [65, 23]:

$$\begin{aligned} p(x, y) &= e^{x^4 + y^4},\ q(y, u) = u^3(y)/10, \\ g(x) &= x + \frac{1}{40}(1 - e^1)e^{x^4}, \quad x \in [0, 1]. \end{aligned}$$

with the exact solution $u(x) = x$. The performance of the methods on this example are summarized in Table 7 and the following are observed:

(a) on all the meshes, the Picard and Mann schemes performed best while Abbas-Nazir and S-star are the least efficient,

(b) on the grid with 5 points, all the methods had equal accuracy, but

(c) on the finer grids (with 20, 100, 120, and 150 mesh points), the Picard and Mann schemes had better accuracy.

Example 5

Our final example is the problem, [65, 23]:

$$p(x,y) = sin(x + e^y)/20,\ q(y, u(y)) = e^{u(y)},$$
$$g(x) = x + \frac{1}{20}\left(cos(x + e^1) - cos(1 + x)\right) \quad x \in [0, 1].$$

This has the exact solution $u(x) = x$. The results for this problem is displayed in Table 8. We observe that

(a) on all the grids, the Picard and Mann schemes took the least CPU time than the others,

(b) on the grid of 5 points, the error of the Picard and Mann schemes are a little higher than the others, but

(c) on the finer grids with 20, 100, 120, 150 grid points, the Picard and Mann schemes are a little more accurate.

Table 1: Accuracy Assessment for Example 1

N	P-Error	P-EOC	Kras-Error	Kras-EOC	Ishi-Error	Ishi-EOC
5	3.98597895e-03	-	3.98597895e-03	-	3.98597923e-03	-
10	2.14745588e-04	4.2	2.14745588e-04	4.2	2.14745657e-04	4.2
20	1.20598259e-05	4.2	1.20598259e-05	4.2	1.20599184e-05	4.2
40	7.11484993e-07	4.1	7.11484993e-07	4.1	7.11585895e-07	4.1
80	4.30685774e-08	4.0	4.30685774e-08	4.0	4.31723663e-08	4.0
160	2.65002931e-09	4.0	2.65002931e-09	4.0	2.65878164e-09	4.0
320	1.56096025e-10	4.1	1.56096025e-10	4.1	1.64881220e-10	4.0
640	1.41953116e-12	6.8	1.41953116e-12	6.8	1.02164943e-11	4.0

Table 2: Accuracy Assessment for Example 1

N	Noor-Error	Noor-EOC	Arga-Error	Arga-EOC	Abbas-Error	Abbas-EOC
5	3.98597923e-03	-	3.98597923e-03	-	3.98597923e-03	-
10	2.14745662e-04	4.2	2.14745657e-04	4.2	2.14745657e-04	4.2
20	1.20599260e-05	4.2	1.20599184e-05	4.2	1.20599184e-05	4.2
40	7.11594280e-07	4.1	7.11585895e-07	4.1	7.11585895e-07	4.1
80	4.31810243e-08	4.0	4.31723663e-08	4.0	4.31723663e-08	4.0
160	2.65878164e-09	4.0	2.65878164e-09	4.0	2.65878164e-09	4.0
320	1.64881220e-10	4.0	1.64881220e-10	4.0	1.64881220e-10	4.0
640	1.02164943e-11	4.0	1.02164943e-11	4.0	1.02164943e-11	4.0

4 Conclusion

Nine discrete iterative methods have been proposed and implemented for Hammerstein equations. The methods were tested for accuracy and computational efficiency. Some of the results obtained include the following:

Table 3: Accuracy Assessment for Example 1

N	Thakur-Error	Thakur-EOC	Ullah-Error	Ullah-EOC	Sstar-Error	Sstar-EOC
5	3.98597923e-03	-	3.98597923e-03	-	3.98597923e-03	-
10	2.14745662e-04	4.2	2.14745662e-04	4.2	2.14745662e-04	4.2
20	1.20599260e-05	4.2	1.20599260e-05	4.2	1.20599260e-05	4.2
40	7.11594280e-07	4.1	7.11594280e-07	4.1	7.11594330e-07	4.1
80	4.31810243e-08	4.0	4.31810243e-08	4.0	4.31810765e-08	4.0
160	2.65878164e-09	4.0	2.65883449e-09	4.0	2.65883449e-09	4.0
320	1.64881220e-10	4.0	1.64934733e-10	4.0	1.64934733e-10	4.0
640	1.02164943e-11	4.0	1.02697850e-11	4.0	1.02697850e-11	4.0

(i) all the methods converge with the correct order of accuracy,

(ii) they all have comparable accuracy,

(iii) the Picard scheme is the fastest, followed by the Mann scheme,

(iv) the S-star and Abbas-Nazir schemes are the slowest,

(v) on very coarse grids, they all have comparable accuracy, but

(vi) the Picard and Mann schemes have slightly better accuracy on finer grids.

From the above results, we conclude that the Picard scheme should be the method of choice whenever the operator of the problem is a contraction. But if the operator is only non-expansive, then the Mann scheme is sufficient. We recommend that more iterative schemes should be investigated on this and other type of problems.

References

[1] Émile Picard. Memoire sur la theorie des equations aux derivees partielles et la methode des approximations successives. *Journal de Mathématiques pures et appliquées*, 6:145–210, 1890.

[2] Vasile Berinde. *Iterative Approximation of Fixed Points*. Springer, 2007.

[3] Shanza Hassan, Manuel De la Sen, Praveen Agarwal, Qasim Ali, and Azhar Hussain. A new faster iterative scheme for numerical fixed points estimation of suzukis generalized nonexpansive mappings. *Mathematical Problems in Engineering*, 2020:1–9, 2020.

[4] W.R. Mann. Mean value methods in iteration. 4, 506510 (1953). *Proc. Am. Math. Soc.*, 4:506–510, 1953.

[5] Mark Aleksandrovich Krasnosel'skii. Two remarks on the method of successive approximations. *Uspekhi matematicheskikh nauk*, 10(1):123–127, 1955.

[6] Shiro Ishikawa. Fixed points by a new interation method. *Proceedings of the American Mathematical Society*, 44(1):147, 1974.

[7] RP Agarwal, Donal O Regan, and DR2314666 Sahu. Iterative construction of fixed points of nearly asymptotically nonexpansive mappings. *Journal of Nonlinear and convex Analysis*, 8(1):61, 2007.

[8] Chinedu Nwaigwe, Azubuike Weli, and Dang Ngoc Hoang Thanh. Fourth-order trapezoid algorithm with four iterative schemes for nonlinear integral equations. `https://www.researchgate.net/publication/368830972_Fourth-Order_Trapezoid_Algorithm_with_Four_Iterative_Schemes_for_Nonlinear_Integral_Equations`, 2023. (Accessed on 19-January-2023).

[9] Muhammad Aslam Noor. New approximation schemes for general variational inequalities. *Journal of Mathematical Analysis and applications*, 251(1):217–229, 2000.

[10] Mujahid Abbas and Talat Nazir. Some new faster iteration process applied to constrained minimization and feasibility problems. 2014.

[11] Balwant Singh Thakur, Dipti Thakur, and Mihai Postolache. A new iteration scheme for approximating fixed points of nonexpansive mappings. *Filomat*, 30(10):2711–2720, 2016.

[12] K. Ullah and M. Arshad. New three step iteration process and fixed point approximation in banach space. *Journal of Linear and Topological Algebra*, 7(2):87–100, 2018.

[13] Godwin Amechi Okeke. Convergence analysis of the picard–ishikawa hybrid iterative process with applications. *Afrika Matematika*, 30(5):817–835, 2019.

[14] MA Abdou. On a symptotic methods for fredholm–volterra integral equation of the second kind in contact problems. *Journal of Computational and Applied Mathematics*, 154(2):431–446, 2003.

[15] Tien-Dung Le, Christian Moyne, MA Murad, and SA Lima. A two-scale non-local model of swelling porous media incorporating ion size correlation effects. *Journal of the Mechanics and Physics of Solids*, 61(12):2493–2521, 2013.

[16] Aline C Rocha, Marcio A Murad, Christian Moyne, Saulo P Oliveira, and Tien D Le. A new methodology for computing ionic profiles and disjoining pressure in swelling porous media. *Computational Geosciences*, 20(5):975–996, 2016.

[17] Jim Gatheral, Alexander Schied, and Alla Slynko. Transient linear price impact and fredholm integral equations. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 22(3):445–474, 2012.

[18] Amjad Alipanah. Solution of hallens integral equation by using radial basis functions. *MATHEMATICAL REPORTS*, 15(3):211–220, 2013.

[19] K Mei. On the integral equations of thin wire antennas. *IEEE Transactions on Antennas and Propagation*, 13(3):374–378, 1965.

[20] Yao Lu, Lixin Shen, and Yuesheng Xu. Integral equation models for image restoration: high accuracy methods and fast algorithms. *Inverse Problems*, 26(4):045006, 2010.

[21] WE Olmstead and Richard A Handelsman. Asymptotic solution to a class of nonlinear volterra integral equations. ii. *SIAM Journal on Applied Mathematics*, 30(1):180–189, 1976.

[22] Shouchuan Hu, Mohammad Khavanin, and WAN Zhuang. Integral equations arising in the kinetic theory of gases. *Applicable Analysis*, 34(3-4):261–266, 1989.

[23] Abdul Jerri. *Introduction to integral equations with applications*. John Wiley & Sons, 1999.

[24] Donal Oregan. Existence results for nonlinear integral equations. *Journal of Mathematical Analysis and Applications*, 192(3):705–726, 1995.

[25] Jan Prüss. *Evolutionary integral equations and applications*, volume 87. Birkhäuser, 2013.

[26] K Maleknejad and M Hadizadeh. A new computational method for volterra-fredholm integral equations. *Computers & Mathematics with Applications*, 37(9):1–8, 1999.

[27] Abdul-Majid Wazwaz. A reliable treatment for mixed volterra–fredholm integral equations. *Applied Mathematics and Computation*, 127(2-3):405–414, 2002.

[28] Abdul-Majid Wazwaz. *First Course In Integral Equations, A*. World Scientific Publishing Company, 2015.

[29] Sanda Micula. On some iterative numerical methods for mixed volterra–fredholm integral equations. *Symmetry*, 11(10):1200, 2019.

[30] Pakhshan M Hasan, Nejmaddin A Sulaiman, Fazlollah Soleymani, and Ali Akgül. The existence and uniqueness of solution for linear system

of mixed volterra-fredholm integral equations in banach space. *AIMS Mathematics*, 5(1):226–235, 2020.

[31] K Maleknejad, R Mollapourasl, and K Nouri. Study on existence of solutions for some nonlinear functional–integral equations. *Nonlinear Analysis: Theory, Methods & Applications*, 69(8):2582–2588, 2008.

[32] Hemant Kumar Pathak et al. Study on existence of solutions for some nonlinear functional-integral equations with applications. *Mathematical Communications*, 18(1):97–107, 2013.

[33] PM Fitzpatrick. Klaus deimling, nonlinear functional analysis. *Bulletin (New Series) of the American Mathematical Society*, 20(2):277–280, 1989.

[34] Józef Banaś and Beata Rzepka. On existence and asymptotic stability of solutions of a nonlinear integral equation. *Journal of Mathematical Analysis and Applications*, 284(1):165–173, 2003.

[35] Kendall Atkinson and Weimin Han. *Theoretical Numerical Analysis: A Functional Analysis Framework*, volume 39. Springer, 2005.

[36] Sohrab Bazm, Pedro Lima, and Somayeh Nemati. Analysis of the euler and trapezoidal discretization methods for the numerical solution of nonlinear functional volterra integral equations of urysohn type. *Journal of Computational and Applied Mathematics*, page 113628, 2021.

[37] R Ezzati and S Najafalizadeh. Numerical methods for solving linear and nonlinear volterra-fredholm integral equations by using cas wavelets. *World Applied Sciences Journal*, 18(12):1847–1854, 2012.

[38] SC Shiralashetti and Lata Lamani. Cas wavelets stochastic operational matrix of integration and its application for solving stochastic itô-volterra integral equations. *Jordan Journal of Mathematics and Statistics (JJMS)*, 14(3):555–580, 2021.

[39] Yadollah Ordokhani and Mohsen Razzaghi. Solution of nonlinear volterra–fredholm–hammerstein integral equations via a collocation method and rationalized haar functions. *Applied Mathematics Letters*, 21(1):4–9, 2008.

[40] Hermann Brunner. On the numerical solution of nonlinear volterra–fredholm integral equations by collocation methods. *SIAM Journal on Numerical Analysis*, 27(4):987–1000, 1990.

[41] Lan Xu. Variational iteration method for solving integral equations. *Computers & Mathematics with Applications*, 54(7-8):1071–1078, 2007.

[42] Shruti S Sheth, Dr Singh, et al. An analytical approximate solution of linear, system of linear and non linear volterra integral equations using variational iteration method. In *Proceedings of International Conference on Advancements in Computing & Management (ICACM)*, 2019.

[43] Sohrab Ali Yousefi, Ali Lotfi, and Mehdi Dehghan. Hes variational iteration method for solving nonlinear mixed volterra–fredholm integral equations. *Computers & Mathematics with Applications*, 58(11-12):2172–2176, 2009.

[44] A Hamoud and K Ghadle. On the numerical solution of nonlinear volterra-fredholm integral equations by variational iteration method. *Int. J. Adv. Sci. Tech. Research*, 3:45–51, 2016.

[45] Pakhshan MA Hasan and NA Suleiman. Numerical solution of mixed volterra-fredholm integral equations using linear programming problem. *Applied Mathematics*, 8(3):42–45, 2018.

[46] Zhong Chen and Wei Jiang. An approximate solution for a mixed linear volterra–fredholm integral equation. *Applied Mathematics Letters*, 25(8):1131–1134, 2012.

[47] Chinedu Nwaigwe and Deborah Ngochinma Benedict. Generalized banach fixed-point theorem and numerical discretization for nonlinear volterra-fredholm equations. *Journal of Computational and Applied Mathematics*, 425:115019, 2023.

[48] Chinedu Nwaigwe. Solvability and approximation of nonlinear functional mixed volterra-fredholm equation in banach space. *Journal of Integral Equations and Applications*, 34(4):489–500, 2022.

[49] Imran Aziz et al. New algorithms for the numerical solution of nonlinear fredholm and volterra integral equations using haar wavelets. *Journal of Computational and Applied Mathematics*, 239:333–345, 2013.

[50] Kendall E Atkinson. The numerical solution of integral equations of the second kind. *Cambridge Monographs on Applied and Computational Mathematics*, 1996.

[51] YH Youssri and RM Hafez. Chebyshev collocation treatment of volterra–fredholm integral equation with error analysis. *Arabian Journal of Mathematics*, 9(2):471–480, 2020.

[52] Sanda Micula. Numerical solution of two-dimensional fredholmvolterra integral equations of the second kind. *Symmetry*, 13(8):1326, 2021.

[53] Chinedu Nwaigwe and Azubuike Weli. Ishikawa-collocation method for nonlinear fredholm equations with non-separable kernels. *Journal of Advances in Mathematics and Computer Science*, 38(3):1–11, 2023.

[54] Sanda Micula. Iterative numerical methods for a fredholm–hammerstein integral equation with modified argument. *Symmetry*, 15(1):66, 2023.

[55] Chinedu Nwaigwe and Sanda Micula. Fast and accurate numerical algorithm with performance assessment for nonlinear functional volterra equations. `https://www.researchgate.net/publication/361722935_Fast_and_Accurate_Numerical_Algorithm_with_Performance_Assessment_for_Nonlinear_Functional_Volterra_Equations`, 2023. (Accessed on 19-January-2023).

[56] Azubuike Weli and Chinedu Nwaigwe. Computational analysis of two numerical solvers for functional fredholm equations. *International Journal of Mathematical and Computational Methods*, 8:1–8, 2023.

[57] Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta mathematicae*, 3(1):133–181, 1922.

[58] Chinedu Nwaigwe, Jonathan Oahimire, and Azubuike Weli. Numerical approximation of convective brinkman-forchheimer flow with variable permeability. *Applied and Computational Mechanics*, 17(1), 2023.

[59] Chinedu Nwaigwe. An unconditionally stable scheme for two-dimensional convection-diffusion-reaction equations. `https://www.researchgate.net/publication/357606287_An_Unconditionally_Stable_Scheme_for_Two-Dimensional_Convection-Diffusion-Reaction_Equations`, 2022. (Accessed on 19-January-2023).

[60] Chinedu Nwaigwe and Oluwole Daniel Makinde. Finite difference investigation of a polluted non-isothermal non-newtonian porous media flow. *Diffusion Foundations*, 26(4):145–156, 2019.

[61] Chinedu Nwaigwe and Azubuike Weli. Analysis of two finite difference schemes for a channel flow problem. *Asian Research Journal of Mathematics*, 15:1–14, 2019.

[62] Azubuike Weli and Chinedu Nwaigwe. Numerical analyses of channel flow with velocity-dependent suction and nonlinear heat source. *Journal of Interdisciplinary Mathematics*, 23(5):987–1008, 2020.

[63] Chinedu Nwaigwe. Analysis and application of a convergent difference scheme to nonlinear transport in a brinkman flow. *International Journal of Numerical Methods for Heat & Fluid Flow*, 30(10):4453–4473, 2020.

[64] Chinedu Nwaigwe, Azubuike Weli, and Oluwole Daniel Makinde. Computational analysis of porous channel flow with cross-diffusion. *American Journal of Computational and Applied Mathematics*, 9(5):119–132, 2019.

[65] Huda O Bakodah and Mohamed Abdalla Darwish. Solving hammerstein type integral equation by new discrete adomian decomposition methods. *Mathematical Problems in Engineering*, 2013, 2013.

Table 4: CPU times for Example 1

Method	No. of Runs	Avg. CPU Time	Error	Picard Factor
Results for 5 *Mesh Points.* Maximum Error is 3.98598e-03				
picard	50	0.00154	3.98598e-03	1.0
krasnoselskij	50	0.0016	3.98598e-03	1.0368
ishikawa	50	0.00193	3.98598e-03	1.2554
noor	50	0.00229	3.98598e-03	1.4839
argawal	50	0.00192	3.98598e-03	1.2464
abbasNazir	50	0.00286	3.98598e-03	1.8556
thakur	50	0.00229	3.98598e-03	1.4845
ullahAshad	50	0.00225	3.98598e-03	1.4596
sstar	50	0.00297	3.98598e-03	1.9289
Results for 20 *Mesh Points.* Maximum Error is 1.20599e-05				
picard	50	0.01437	1.20598e-05	1.0
krasnoselskij	50	0.01445	1.20598e-05	1.0052
ishikawa	50	0.01596	1.20599e-05	1.1105
noor	50	0.01916	1.20599e-05	1.3335
argawal	50	0.01599	1.20599e-05	1.1126
abbasNazir	50	0.02392	1.20599e-05	1.6646
thakur	50	0.01911	1.20599e-05	1.33
ullahAshad	50	0.01906	1.20599e-05	1.3265
sstar	50	0.02538	1.20599e-05	1.7658
Results for 100 *Mesh Points.* Maximum Error is 1.75819e-08				
picard	50	0.30877	1.75732e-08	1.0
krasnoselskij	50	0.3089	1.75732e-08	1.0004
ishikawa	50	0.37039	1.75819e-08	1.1995
noor	50	0.37276	1.75819e-08	1.2072
argawal	50	0.37	1.75819e-08	1.1983
abbasNazir	50	0.5563	1.75819e-08	1.8016
thakur	50	0.37129	1.75819e-08	1.2025
ullahAshad	50	0.4949	1.75819e-08	1.6028
sstar	50	0.74098	1.75819e-08	2.3997
Results for 120 *Mesh Points.* Maximum Error is 8.44525e-09				
picard	50	0.43594	8.43647e-09	1.0
krasnoselskij	50	0.43516	8.43647e-09	0.9982
ishikawa	50	0.52422	8.44519e-09	1.2025
noor	50	0.5248	8.44519e-09	1.2038
argawal	50	0.52247	8.44519e-09	1.1985
abbasNazir	50	0.78585	8.44519e-09	1.8027
thakur	50	0.52504	8.44519e-09	1.2044
ullahAshad	50	0.69836	8.44525e-09	1.602
sstar	50	1.04836	8.44525e-09	2.4048
Results for 150 *Mesh Points.* Maximum Error is 3.44540e-09				
picard	50	0.67351	3.43660e-09	1.0
krasnoselskij	50	0.67361	3.43660e-09	1.0001
ishikawa	50	0.80986	3.44535e-09	1.2025
noor	50	0.81206	3.44535e-09	1.2057
argawal	50	0.81036	3.44535e-09	1.2032
abbasNazir	50	1.2148	3.44535e-09	1.8037
thakur	50	0.80988	3.44535e-09	1.2025
ullahAshad	50	1.0786	3.44540e-09	1.6015
sstar	50	1.61777	3.44540e-09	2.402

Table 5: CPU times for Example 2

Method	No. of Runs	Avg. CPU Time	Error	Picard Factor
Results for 5 *Mesh Points.* Maximum Error is 1.86076e-02				
picard	50	0.00143	1.86076e-02	1.0
krasnoselskij	50	0.00153	1.86076e-02	1.0664
ishikawa	50	0.00157	1.86076e-02	1.0954
noor	50	0.00166	1.86076e-02	1.1567
argawal	50	0.00159	1.86076e-02	1.1052
abbasNazir	50	0.00233	1.86076e-02	1.6253
thakur	50	0.00169	1.86076e-02	1.1803
ullahAshad	50	0.00174	1.86076e-02	1.2148
sstar	50	0.00256	1.86076e-02	1.7819
Results for 20 *Mesh Points.* Maximum Error is 1.34658e-04				
picard	50	0.00984	1.34658e-04	1.0
krasnoselskij	50	0.00995	1.34658e-04	1.0109
ishikawa	50	0.0108	1.34658e-04	1.0973
noor	50	0.01348	1.34658e-04	1.3692
argawal	50	0.01082	1.34658e-04	1.0994
abbasNazir	50	0.01624	1.34658e-04	1.6495
thakur	50	0.01351	1.34658e-04	1.3721
ullahAshad	50	0.01437	1.34658e-04	1.4595
sstar	50	0.02153	1.34658e-04	2.1868
Results for 100 *Mesh Points.* Maximum Error is 2.52100e-07				
picard	50	0.19173	2.51867e-07	1.0
krasnoselskij	50	0.19247	2.51867e-07	1.0039
ishikawa	50	0.20993	2.52064e-07	1.095
noor	50	0.26247	2.52100e-07	1.3689
argawal	50	0.20993	2.52064e-07	1.0949
abbasNazir	50	0.31627	2.52064e-07	1.6496
thakur	50	0.26376	2.52100e-07	1.3757
ullahAshad	50	0.28077	2.52100e-07	1.4644
sstar	50	0.42	2.52100e-07	2.1906
Results for 120 *Mesh Points.* Maximum Error is 1.22342e-07				
picard	50	0.27397	1.22109e-07	1.0
krasnoselskij	50	0.27426	1.22109e-07	1.0011
ishikawa	50	0.29856	1.22307e-07	1.0898
noor	50	0.37373	1.22342e-07	1.3641
argawal	50	0.29871	1.22307e-07	1.0903
abbasNazir	50	0.44882	1.22307e-07	1.6382
thakur	50	0.3737	1.22342e-07	1.364
ullahAshad	50	0.39832	1.22342e-07	1.4539
sstar	50	0.598	1.22342e-07	2.1828
Results for 150 *Mesh Points.* Maximum Error is 5.04261e-08				
picard	50	0.42176	5.01932e-08	1.0
krasnoselskij	50	0.42249	5.01932e-08	1.0017
ishikawa	50	0.46204	5.03905e-08	1.0955
noor	50	0.57649	5.04260e-08	1.3668
argawal	50	0.46063	5.03905e-08	1.0921
abbasNazir	50	0.69203	5.03905e-08	1.6408
thakur	50	0.58538	5.04260e-08	1.3879
ullahAshad	50	0.61517	5.04261e-08	1.4586
sstar	50	0.9223	5.04261e-08	2.1868

Table 6: CPU times for Example 3

Method	No. of Runs	Avg. CPU Time	Error	Picard Factor
Results for 5 *Mesh Points.* Maximum Error is 6.24627e-06				
picard	50	0.00138	6.24627e-06	1.0
krasnoselskij	50	0.00142	6.24627e-06	1.0224
ishikawa	50	0.00171	6.24601e-06	1.2384
noor	50	0.00203	6.24601e-06	1.4634
argawal	50	0.00172	6.24601e-06	1.2408
abbasNazir	50	0.00253	6.24601e-06	1.8259
thakur	50	0.00203	6.24601e-06	1.4687
ullahAshad	50	0.00201	6.24601e-06	1.4546
sstar	50	0.00264	6.24601e-06	1.9033
Results for 20 *Mesh Points.* Maximum Error is 2.99765e-08				
picard	50	0.01071	2.99562e-08	1.0
krasnoselskij	50	0.01077	2.99562e-08	1.0057
ishikawa	50	0.01189	2.99765e-08	1.1106
noor	50	0.01424	2.99751e-08	1.3298
argawal	50	0.01186	2.99765e-08	1.1077
abbasNazir	50	0.01782	2.99765e-08	1.6646
thakur	50	0.01423	2.99751e-08	1.3287
ullahAshad	50	0.0142	2.99751e-08	1.3264
sstar	50	0.01887	2.99751e-08	1.7619
Results for 100 *Mesh Points.* Maximum Error is 4.96982e-11				
picard	50	0.18341	2.96467e-11	1.0
krasnoselskij	50	0.1837	2.96467e-11	1.0015
ishikawa	50	0.20357	4.96982e-11	1.1099
noor	50	0.24424	4.83151e-11	1.3316
argawal	50	0.20346	4.96982e-11	1.1093
abbasNazir	50	0.30564	4.96982e-11	1.6664
thakur	50	0.24413	4.83151e-11	1.331
ullahAshad	50	0.24369	4.83151e-11	1.3287
sstar	50	0.32583	4.83076e-11	1.7765
Results for 120 *Mesh Points.* Maximum Error is 2.46855e-11				
picard	50	0.26321	4.64184e-12	1.0
krasnoselskij	50	0.26245	4.64184e-12	0.9971
ishikawa	50	0.29267	2.46855e-11	1.1119
noor	50	0.35097	2.33030e-11	1.3335
argawal	50	0.29189	2.46855e-11	1.109
abbasNazir	50	0.43842	2.46855e-11	1.6657
thakur	50	0.3504	2.33030e-11	1.3313
ullahAshad	50	0.35133	2.33030e-11	1.3348
sstar	50	0.46819	2.32954e-11	1.7788
Results for 150 *Mesh Points.* Maximum Error is 1.09307e-11				
picard	50	0.40479	9.10583e-12	1.0
krasnoselskij	50	0.40341	9.10583e-12	0.9966
ishikawa	50	0.45076	1.09307e-11	1.1136
noor	50	0.54036	9.54892e-12	1.3349
argawal	50	0.44984	1.09307e-11	1.1113
abbasNazir	50	0.67472	1.09307e-11	1.6669
thakur	50	0.54072	9.54892e-12	1.3358
ullahAshad	50	0.54039	9.54892e-12	1.335
sstar	50	0.72127	9.54115e-12	1.7818

Table 7: CPU times for Example 4

Method	No. of Runs	Avg. CPU Time	Error	Picard Factor
Results for 5 *Mesh Points.* Maximum Error is 1.18205e-02				
picard	50	0.00332	1.18205e-02	1.0
krasnoselskij	50	0.0034	1.18205e-02	1.0232
ishikawa	50	0.00353	1.18205e-02	1.0618
noor	50	0.00407	1.18205e-02	1.2261
argawal	50	0.00351	1.18205e-02	1.0574
abbasNazir	50	0.00521	1.18205e-02	1.5691
thakur	50	0.00406	1.18205e-02	1.2227
ullahAshad	50	0.00381	1.18205e-02	1.1484
sstar	50	0.00455	1.18205e-02	1.3705
Results for 20 *Mesh Points.* Maximum Error is 7.61931e-05				
picard	50	0.02595	7.61925e-05	1.0
krasnoselskij	50	0.02596	7.61925e-05	1.0003
ishikawa	50	0.02913	7.61931e-05	1.1224
noor	50	0.02911	7.61931e-05	1.1217
argawal	50	0.02914	7.61931e-05	1.1227
abbasNazir	50	0.04372	7.61931e-05	1.6846
thakur	50	0.02909	7.61931e-05	1.121
ullahAshad	50	0.03228	7.61931e-05	1.2439
sstar	50	0.03867	7.61931e-05	1.4901
Results for 100 *Mesh Points.* Maximum Error is 1.45226e-07				
picard	50	0.49977	1.44638e-07	1.0
krasnoselskij	50	0.49921	1.44638e-07	0.9989
ishikawa	50	0.56161	1.45179e-07	1.1237
noor	50	0.65562	1.45225e-07	1.3118
argawal	50	0.56191	1.45179e-07	1.1243
abbasNazir	50	0.84302	1.45179e-07	1.6868
thakur	50	0.6555	1.45225e-07	1.3116
ullahAshad	50	0.62451	1.45222e-07	1.2496
sstar	50	0.74923	1.45226e-07	1.4992
Results for 120 *Mesh Points.* Maximum Error is 7.05710e-08				
picard	50	0.71536	6.99833e-08	1.0
krasnoselskij	50	0.7142	6.99833e-08	0.9984
ishikawa	50	0.80385	7.05242e-08	1.1237
noor	50	0.93812	7.05700e-08	1.3114
argawal	50	0.8056	7.05242e-08	1.1261
abbasNazir	50	1.21083	7.05242e-08	1.6926
thakur	50	0.94039	7.05700e-08	1.3146
ullahAshad	50	0.89496	7.05673e-08	1.2511
sstar	50	1.07267	7.05710e-08	1.4995
Results for 150 *Mesh Points.* Maximum Error is 2.91276e-08				
picard	50	1.09619	2.85398e-08	1.0
krasnoselskij	50	1.09752	2.85398e-08	1.0012
ishikawa	50	1.23293	2.90808e-08	1.1247
noor	50	1.43926	2.91265e-08	1.313
argawal	50	1.23749	2.90808e-08	1.1289
abbasNazir	50	1.85121	2.90808e-08	1.6888
thakur	50	1.44569	2.91265e-08	1.3188
ullahAshad	50	1.37603	2.91239e-08	1.2553
sstar	50	1.63943	2.91276e-08	1.4956

Table 8: CPU times for Example 5

Method	No. of Runs	Avg. CPU Time	Error	Picard Factor
Results for 5 *Mesh Points.* Maximum Error is 2.54076e-04				
picard	50	0.0023	2.54076e-04	1.0
krasnoselskij	50	0.00234	2.54076e-04	1.0166
ishikawa	50	0.00286	2.54075e-04	1.2437
noor	50	0.00342	2.54075e-04	1.4895
argawal	50	0.00288	2.54075e-04	1.255
abbasNazir	50	0.00426	2.54075e-04	1.8523
thakur	50	0.0034	2.54075e-04	1.4785
ullahAshad	50	0.00337	2.54075e-04	1.4673
sstar	50	0.0045	2.54075e-04	1.9599
Results for 20 *Mesh Points.* Maximum Error is 5.35560e-07				
picard	50	0.01949	5.35545e-07	1.0
krasnoselskij	50	0.01946	5.35545e-07	0.9985
ishikawa	50	0.02423	5.35560e-07	1.2436
noor	50	0.02905	5.35560e-07	1.491
argawal	50	0.02433	5.35560e-07	1.2487
abbasNazir	50	0.03645	5.35560e-07	1.8707
thakur	50	0.02909	5.35560e-07	1.4926
ullahAshad	50	0.0292	5.35560e-07	1.4983
sstar	50	0.03871	5.35560e-07	1.9868
Results for 100 *Mesh Points.* Maximum Error is 9.44677e-10				
picard	50	0.38211	9.30307e-10	1.0
krasnoselskij	50	0.38231	9.30307e-10	1.0005
ishikawa	50	0.47801	9.44635e-10	1.251
noor	50	0.57326	9.44677e-10	1.5002
argawal	50	0.47776	9.44635e-10	1.2503
abbasNazir	50	0.71852	9.44635e-10	1.8804
thakur	50	0.57338	9.44677e-10	1.5005
ullahAshad	50	0.57286	9.44677e-10	1.4992
sstar	50	0.76452	9.44677e-10	2.0008
Results for 120 *Mesh Points.* Maximum Error is 4.57105e-10				
picard	50	0.5356	4.42732e-10	1.0
krasnoselskij	50	0.53529	4.42732e-10	0.9994
ishikawa	50	0.66899	4.57063e-10	1.249
noor	50	0.80279	4.57105e-10	1.4989
argawal	50	0.66869	4.57063e-10	1.2485
abbasNazir	50	1.00378	4.57063e-10	1.8741
thakur	50	0.80231	4.57105e-10	1.498
ullahAshad	50	0.80231	4.57105e-10	1.498
sstar	50	1.07114	4.57105e-10	1.9999
Results for 150 *Mesh Points.* Maximum Error is 1.87850e-10				
picard	50	0.82695	1.73474e-10	1.0
krasnoselskij	50	0.82722	1.73474e-10	1.0003
ishikawa	50	1.03291	1.87808e-10	1.2491
noor	50	1.2396	1.87850e-10	1.499
argawal	50	1.03322	1.87808e-10	1.2494
abbasNazir	50	1.55019	1.87808e-10	1.8746
thakur	50	1.24041	1.87850e-10	1.5
ullahAshad	50	1.24052	1.87850e-10	1.5001
sstar	50	1.65244	1.87850e-10	1.9982