Performance Analysis of the First Order Linear Stationary Iterative Methods in Solving Third Order Newton-Cotes Quadrature System

Mohana Sundaram Muthuvalu and Jumat Sulaiman

***Abstract*—**In this paper, application of the first order linear stationary iterative methods is extended to solve third order composite closed Newton-Cotes quadrature (3-CCNC) system. The performances of tested iterative methods for the 3-CCNC system are comparatively studied by their application to solve the second kind linear Fredholm integral equations. The derivation and implementation of the methods are presented. In addition, simulation results of three test problems are included to verify the performance of the methods.

***Keywords*—**Fredholm integral equations, Newton-Cotes quadrature, Stationary iterative method, Dense linear system.

I. Introduction

The aim of this paper is to compare the performance of the first order linear stationary iterative methods for solving nonsingular system arising from discretization of the second kind linear Fredholm integral equations of the form

$$\varphi(x)-(\kappa\varphi)(x)=f(x),\ x\in[\alpha,\beta] \tag{1}$$

Where

$$(\kappa\varphi)(x)=\int_{\alpha}^{\beta}K(x,t)\varphi(t)\,dt\,. \tag{2}$$

The function $f(x)\in L^2[\alpha,\beta]$ is given, $K(x,t)\in L^2([\alpha,\beta]\times[\alpha,\beta])$ is the kernel of the integral equation and $\varphi(x)$ is the solution to be determined. It is assumed that the $f(x)$ and $K(x,t)$ are continuous and problem (1) have a unique solution.

There is a vast literature on numerical methods for solving problem (1), for instance refer [1-10]. The applications of numerical methods for problem (1) mostly lead to dense linear system. In this paper, performance of five different first order linear stationary iterative methods i.e. Jacobi, backward Gauss-Seidel (BGS), forward Gauss-Seidel (FGS), backward Successive Over-Relaxation (BSOR) and forward Successive Over-Relaxation (FSOR) methods in solving third order composite closed Newton-Cotes quadrature (3-CCNC) system associated with the numerical solution of problem (1) are investigated.

The rest of this paper is organized as follows. An implementation of the 3-CCNC scheme in discretizing problem (1) is presented in Section II followed by the formulation of the tested first order linear stationary iterative methods in Section III. Numerical performance of the tested first order linear stationary iterative methods and concluding remarks are summarized in Section IV and V respectively.

II. 3-CCNC System

In this section, an implementation of the 3-CCNC scheme for discretizing problem (1) is discussed. Let interval $[\alpha,\beta]$ be divided uniformly into even N subintervals and the discrete set of points of x and t given by $x_i=\alpha+ih$ $(i=0,1,2,\cdots,N-2,N-1,N)$ and $t_j=\alpha+jh$ $(j=0,1,2,\cdots,N-2,N-1,N)$ respectively, where the constant step size, h is defined as follows

$$h=\frac{\beta-\alpha}{N}\,. \tag{3}$$

Before further discussion, the following notations will be used for simplicity

$$\left.\begin{aligned} K_{i,j}&=K(x_i,t_j)\\ \hat{\varphi}_i&=\hat{\varphi}(x_i)\\ \hat{\varphi}_j&=\hat{\varphi}(t_j)\\ f_i&=f(x_i)\end{aligned}\right\} \tag{4}$$

M. S. Muthuvalu is with the Department of Fundamental and Applied Sciences, Faculty of Science and Information Technology, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia. (Corresponding author: mohana.muthuvalu@petronas.com.my)

J. Sulaiman is with the School of Science and Technology, Universiti Malaysia Sabah, Jalan UMS, 88400 Kota Kinabalu, Sabah, Malaysia (e-mail: jumat@ums.edu.my).

An implementation of the 3-CCNC scheme reduced problem (1) to

$$\hat{\varphi}_i - \sum_{j=0}^{N} w_j K_{i,j}\,\hat{\varphi}_j = f_i \tag{5}$$

for $i = 0,1,2,\cdots,N-2,N-1,N$, where solution $\hat{\varphi}$ is an approximation of the exact solution φ to (1) and w_j is the weights of 3-CCNC scheme that satisfy the following conditions

$$w_j = \begin{cases} \frac{3}{8}h, & j = 0, N \\ \frac{3}{4}h, & j = 3,6,9,\cdots,N-3 \\ \frac{9}{8}h, & otherwise \end{cases}. \tag{6}$$

Following the conventional process, (5) can be written as the following matrix form

$$A\hat{\varphi} = f \tag{7}$$

where $A = \left(a_{i,j}\right) \in \mathrm{R}^{(N+1)\times(N+1)}$ is a real nonsingular matrix with

$$a_{i,j} = \begin{cases} 1 - w_j K_{i,j}, & i = j \\ -w_j K_{i,j}, & i \neq j \end{cases},$$

$$\hat{\varphi} = \begin{bmatrix} \hat{\varphi}_0 & \hat{\varphi}_1 & \cdots & \hat{\varphi}_{N-1} & \hat{\varphi}_N \end{bmatrix}^T$$

and

$$f = \begin{bmatrix} f_0 & f_1 & \cdots & f_{N-1} & f_N \end{bmatrix}^T.$$

III. First Order Linear Stationary Iterative Methods

Let

$$A = M - N \tag{8}$$

be a splitting of A that is $M, N \in \mathrm{R}^{(N+1)\times(N+1)}$ with $\det(M) \neq 0$. Based on (8), the first order linear stationary iterative method for solving linear system (7) has the form

$$M\hat{\varphi}^{(k+1)} = N\hat{\varphi}^{(k)} + f, \; k = 0,1,2,\cdots \tag{9}$$

or equivalently

$$\hat{\varphi}^{(k+1)} = T\hat{\varphi}^{(k)} + c, \; k = 0,1,2,\cdots \tag{10}$$

with $\hat{\varphi}^{(k)} \in \mathrm{R}^{N+1}$ the k-th approximation to the solution $\hat{\varphi}$ of (7), $T = M^{-1}N$ is called the iteration matrix of the method and $c = M^{-1}f$. Let $\sigma(T)$ denote the eigenvalue of matrix T. It is well-known that (10) produces a sequence of vectors $\left\{\hat{\varphi}^{(k)}\right\}$, $k = 0,1,2,\cdots$ convergent to the unique solution $\hat{\varphi}$ of (7) if and only if $\rho(T) < 1$ (where $\rho(T) = \max\left\{\left|\lambda_j\right|; \lambda_j \in \sigma(T)\right\}$) for an arbitrary initial datum $\hat{\varphi}^{(0)}$. It can be shown that the smaller $\rho(T)$ implies faster convergence of (10).

Now, let consider D, $-L$ and $-U$ be the diagonal, strictly lower triangular and strictly upper triangular parts of A respectively. Thus, each stationary iterative method that is considered in this paper is based on the splittings of A as follows

i) Jacobi
$M = D$, $N = L + U$

ii) Backward Gauss-Seidel
$M = D - U$, $N = L$

iii) Forward Gauss-Seidel
$M = D - L$, $N = U$

iv) Backward Successive Over-Relaxation
$M = \frac{1}{\omega}(D - \omega U)$, $N = \frac{1}{\omega}\left[\omega L - (\omega - 1)D\right]$

v) Forward Successive Over-Relaxation
$M = \frac{1}{\omega}(D - \omega L)$, $N = \frac{1}{\omega}\left[\omega U - (\omega - 1)D\right]$

where ω is a relaxation parameter. As can be seen, unlike the Jacobi method, the BGS, FGS, BSOR and FSOR methods depends on the ordering of the unknowns. FGS and FSOR methods begins the update of $\hat{\varphi}$ with the first component, whereas for BGS and BSOR methods with the last component. The performance of the BSOR and FSOR methods can be very often drastically improved with the proper choice of the relaxation parameter, ω. It is noted that when $\omega = 1$, BSOR and FSOR methods become the expression of BGS and FGS methods respectively.

By determining values of D, $-L$ and $-U$, iterative methods of Jacobi, BGS, FGS, BSOR and FSOR can be applied directly to solve linear system (7), which lead to

i) Jacobi expression:

$$\hat{\varphi}_i^{(k+1)} = \frac{1}{a_{i,i}}\left(f_i - \sum_{j=0}^{i-1} a_{i,j}\hat{\varphi}_j^{(k)} - \sum_{j=i+1}^{N} a_{i,j}\hat{\varphi}_j^{(k)} \right)$$

ii) BGS expression:

$$\hat{\varphi}_i^{(k+1)} = \frac{1}{a_{i,i}}\left(f_i - \sum_{j=0}^{i-1} a_{i,j}\hat{\varphi}_j^{(k)} - \sum_{j=i+1}^{N} a_{i,j}\hat{\varphi}_j^{(k+1)} \right)$$

iii) FGS expression:

$$\hat{\varphi}_i^{(k+1)} = \frac{1}{a_{i,i}}\left(f_i - \sum_{j=0}^{i-1} a_{i,j}\hat{\varphi}_j^{(k+1)} - \sum_{j=i+1}^{N} a_{i,j}\hat{\varphi}_j^{(k)} \right)$$

iv) BSOR expression:

$$\hat{\varphi}_i^{(k+1)} = (1-\omega)\hat{\varphi}_i^{(k)} + \frac{\omega}{a_{i,i}}\left(f_i - \sum_{j=0}^{i-1} a_{i,j}\hat{\varphi}_j^{(k)} - \sum_{j=i+1}^{N} a_{i,j}\hat{\varphi}_j^{(k+1)} \right)$$

iv) FSOR expression:

$$\hat{\varphi}_i^{(k+1)} = (1-\omega)\hat{\varphi}_i^{(k)} + \frac{\omega}{a_{i,i}}\left(f_i - \sum_{j=0}^{i-1} a_{i,j}\hat{\varphi}_j^{(k+1)} - \sum_{j=i+1}^{N} a_{i,j}\hat{\varphi}_j^{(k)} \right)$$

for $i = 0,1,2,\cdots,N-2,N-1,N$. The tested first order linear stationary iterative methods are performed by using all equations until the solution satisfied a specified convergence criterion i.e. maximum iteration error norm, $\left\| \hat{\varphi}^{(k+1)} - \hat{\varphi}^{(k)} \right\|_\infty \le \varepsilon$ where ε is the convergence criterion.

IV. Simulation Results and Analysis

To study the performance of the methods, the following three second kind linear Fredholm integral equations which will generate nonsingular matrix A by using 3-CCNC scheme were used as the test problems.

Test Problem 1 [11]

$$\varphi(x) - \int_0^1 \left(4xt - x^2\right)\varphi(t)\,dt = x, \; x \in [0,1]$$

and the exact solution is given by

$$\varphi(x) = 24x - 9x^2.$$

Test Problem 2 [7]

$$\varphi(x) - \int_0^1 \left(x^2 + t^2\right)\varphi(t)\,dt = x^6 - 5x^3 + x + 10, \; x \in [0,1]$$

with the exact solution

$$\varphi(x) = x^6 - 5x^3 + \frac{1045}{28}x^2 + x + \frac{2141}{84}.$$

Test Problem 3 [9]

$$\varphi(x) - \int_0^{\frac{\pi}{2}} \left(\frac{1}{2}xt\right)\varphi(t)\,dt = \sin(x) - \frac{x}{2}, \; x \in \left[0, \frac{\pi}{2}\right]$$

and the exact solution is of the form

$$\varphi(x) = \sin(x).$$

For the numerical simulations, the following criteria are considered to make a comparative analysis

k	Number of iterations
CPU	CPU time (in seconds) when the converged solution is obtained
$RMSE$	Root mean squared error [2]

The value of initial datum, $\hat{\varphi}^{(0)}$ is set to be zero for all the test problems and experimental values of ω for BSOR and FSOR methods are chosen within ± 0.01 to be an optimal value by a trial and error process. It was found that when the N is taken the same, CPU time required for each iteration step is the same for the Jacobi, BGS, FGS, BSOR and FSOR iterative methods. Thus, the ratio of iterations between two iterative methods equals to the ratio of CPU times. In view of this, the CPU time of BGS, FGS, BSOR and FSOR methods are computed directly from the CPU time of Jacobi method and the ratio of iterations. All simulations described in this paper are performed using C programming language on a PC with Intel(R) Core(TM) 2 (1.66Hz, 1.67Hz) and 1022MB RAM. Throughout the simulations, the convergence test considered $\varepsilon = 10^{-12}$ and carried out on several different N. The simulation results of the tested iterative methods for test problems 1 to 3 are recorded in Tables 1 to 3. Meanwhile, convergence histories of the iterative methods are plotted in Figures 1 to 3.

Table 1. Numerical results for test problem 1

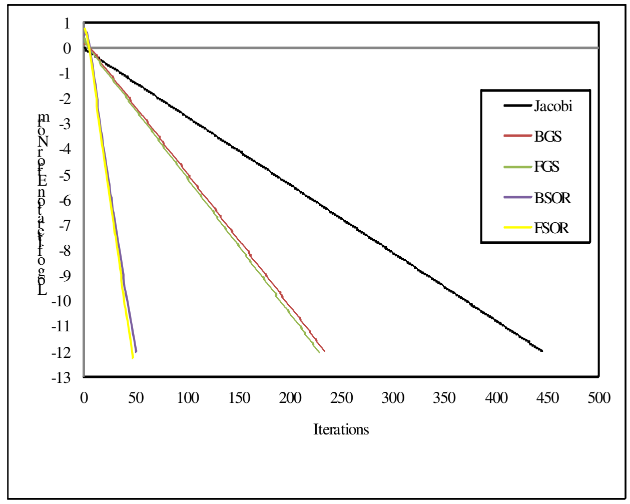
N	Methods	k	CPU	$RMSE$
120	Jacobi	446	0.83	1.0463×10^{-11}
	BGS	234	0.44	4.8017×10^{-12}
	FGS	229	0.43	5.1177×10^{-12}
	BSOR	51 (ω =1.56)	0.09	1.4177×10^{-13}
	FSOR	48 (ω =1.54)	0.09	1.1781×10^{-13}
240	Jacobi	450	3.11	1.0250×10^{-11}
	BGS	236	1.63	4.7567×10^{-12}
	FGS	231	1.60	5.0696×10^{-12}
	BSOR	51 (ω =1.56)	0.35	1.5936×10^{-13}
	FSOR	48 (ω =1.54)	0.33	3.2299×10^{-13}
480	Jacobi	452	6.72	1.0151×10^{-11}
	BGS	237	3.52	4.7364×10^{-12}
	FGS	232	3.45	5.0485×10^{-12}
	BSOR	51 (ω =1.56)	0.76	1.9492×10^{-13}
	FSOR	48 (ω =1.54)	0.71	5.2596×10^{-13}
960	Jacobi	453	26.89	1.0105×10^{-11}
	BGS	238	14.13	4.4523×10^{-12}
	FGS	232	13.77	5.3554×10^{-12}
	BSOR	52 (ω =1.56)	3.09	1.6052×10^{-13}
	FSOR	48 (ω =1.54)	2.85	6.6403×10^{-13}
1920	Jacobi	453	95.57	1.0393×10^{-11}
	BGS	238	50.21	4.5836×10^{-12}
	FGS	233	49.16	4.8834×10^{-12}
	BSOR	52 (ω =1.56)	10.97	1.6493×10^{-13}
	FSOR	49 (ω =1.55)	10.34	5.0660×10^{-14}
3840	Jacobi	454	379.83	9.9180×10^{-12}
	BGS	238	199.12	4.6486×10^{-12}
	FGS	233	194.93	4.9567×10^{-12}
	BSOR	52 (ω =1.56)	43.50	1.6401×10^{-13}
	FSOR	49 (ω =1.55)	40.99	4.9942×10^{-14}
7680	Jacobi	455	1550.58	9.3994×10^{-12}
	BGS	239	814.48	4.1561×10^{-12}
	FGS	234	797.44	4.4197×10^{-12}
	BSOR	52 (ω =1.56)	177.21	1.3260×10^{-13}
	FSOR	49 (ω =1.55)	166.99	4.9390×10^{-14}

Table 2. Numerical results for test problem 2

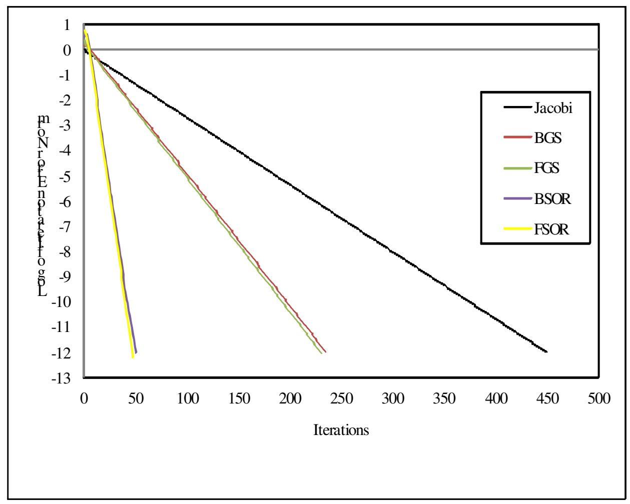
N	Methods	k	CPU	$RMSE$
120	Jacobi	123	0.41	2.5497×10^{-07}
	BGS	66	0.22	2.5497×10^{-07}
	FGS	65	0.22	2.5497×10^{-07}
	BSOR	27 (ω =1.29)	0.09	2.5497×10^{-07}
	FSOR	26 (ω =1.29)	0.09	2.5497×10^{-07}
240	Jacobi	123	1.08	1.5886×10^{-08}
	BGS	67	0.59	1.5887×10^{-08}
	FGS	65	0.57	1.5887×10^{-08}
	BSOR	27 (ω =1.29)	0.24	1.5888×10^{-08}
	FSOR	26 (ω =1.29)	0.23	1.5888×10^{-08}
480	Jacobi	124	5.06	9.8986×10^{-10}
	BGS	67	2.73	9.9096×10^{-10}
	FGS	66	2.69	9.9073×10^{-10}
	BSOR	27 (ω =1.29)	1.10	9.9146×10^{-10}
	FSOR	26 (ω =1.29)	1.06	9.9141×10^{-10}
960	Jacobi	124	19.16	6.0241×10^{-11}
	BGS	67	10.35	6.1381×10^{-11}
	FGS	66	10.20	6.1140×10^{-11}
	BSOR	27 (ω =1.29)	4.17	6.1897×10^{-11}
	FSOR	26 (ω =1.29)	4.02	6.1841×10^{-11}
1920	Jacobi	124	64.28	2.1580×10^{-12}
	BGS	67	34.73	3.3197×10^{-12}
	FGS	66	34.21	3.0747×10^{-12}
	BSOR	27 (ω =1.29)	14.00	3.8407×10^{-12}
	FSOR	26 (ω =1.29)	13.48	3.7798×10^{-12}
3840	Jacobi	125	258.00	1.1161×10^{-12}
	BGS	67	138.29	3.3317×10^{-13}
	FGS	66	136.22	5.6647×10^{-13}
	BSOR	27 (ω =1.29)	55.73	2.2100×10^{-13}
	FSOR	26 (ω =1.29)	53.66	1.6569×10^{-13}
7680	Jacobi	124	1015.03	1.7285×10^{-12}
	BGS	67	548.44	5.5597×10^{-13}
	FGS	66	540.26	7.9662×10^{-13}
	BSOR	27 (ω =1.29)	221.01	5.2906×10^{-14}
	FSOR	26 (ω =1.29)	212.83	1.0232×10^{-13}

Table 3. Numerical results for test problem 3

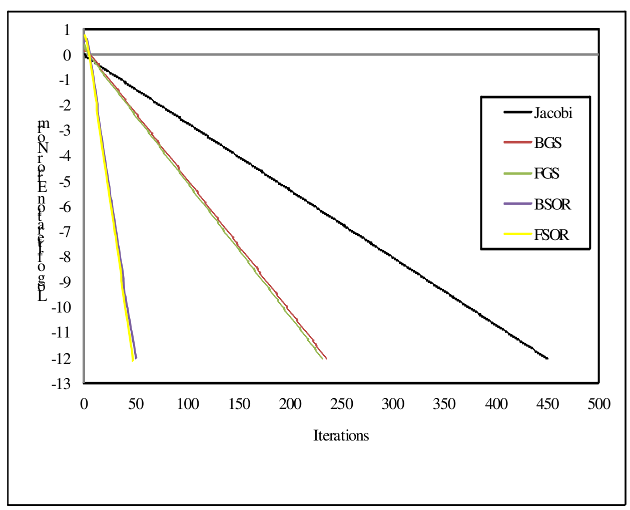
N	Methods	k	CPU	$RMSE$
	Jacobi	62	0.07	1.2835×10^{-03}
	BGS	35	0.04	1.2835×10^{-03}
	FGS	34	0.04	1.2835×10^{-03}
120	BSOR	18	0.02	1.2835×10^{-03}
		(ω =1.19)		
	FSOR	18	0.02	1.2835×10^{-03}
		(ω =1.18)		
	Jacobi	63	0.29	1.2796×10^{-03}
	BGS	35	0.16	1.2796×10^{-03}
	FGS	34	0.16	1.2796×10^{-03}
240	BSOR	18	0.08	1.2796×10^{-03}
		(ω =1.19)		
	FSOR	18	0.08	1.2796×10^{-03}
		(ω =1.19)		
	Jacobi	63	0.88	1.2776×10^{-03}
	BGS	36	0.50	1.2776×10^{-03}
	FGS	35	0.49	1.2776×10^{-03}
480	BSOR	18	0.25	1.2776×10^{-03}
		(ω =1.19)		
	FSOR	18	0.25	1.2776×10^{-03}
		(ω =1.19)		
	Jacobi	63	5.08	1.2766×10^{-03}
	BGS	36	2.90	1.2766×10^{-03}
	FGS	35	2.82	1.2766×10^{-03}
960	BSOR	18	1.45	1.2766×10^{-03}
		(ω =1.19)		
	FSOR	18	1.45	1.2766×10^{-03}
		(ω =1.19)		
	Jacobi	63	13.86	1.2761×10^{-03}
	BGS	36	7.92	1.2761×10^{-03}
	FGS	35	7.70	1.2761×10^{-03}
1920	BSOR	18	3.96	1.2761×10^{-03}
		(ω =1.19)		
	FSOR	18	3.96	1.2761×10^{-03}
		(ω =1.19)		
	Jacobi	63	51.92	1.2758×10^{-03}
	BGS	36	29.67	1.2758×10^{-03}
	FGS	35	28.84	1.2758×10^{-03}
3840	BSOR	18	14.83	1.2758×10^{-03}
		(ω =1.19)		
	FSOR	18	14.83	1.2758×10^{-03}
		(ω =1.19)		
	Jacobi	63	207.65	1.2757×10^{-03}
	BGS	36	118.66	1.2757×10^{-03}
	FGS	35	115.36	1.2757×10^{-03}
7680	BSOR	18	59.33	1.2757×10^{-03}
		(ω =1.19)		
	FSOR	18	59.33	1.2757×10^{-03}
		(ω =1.19)		



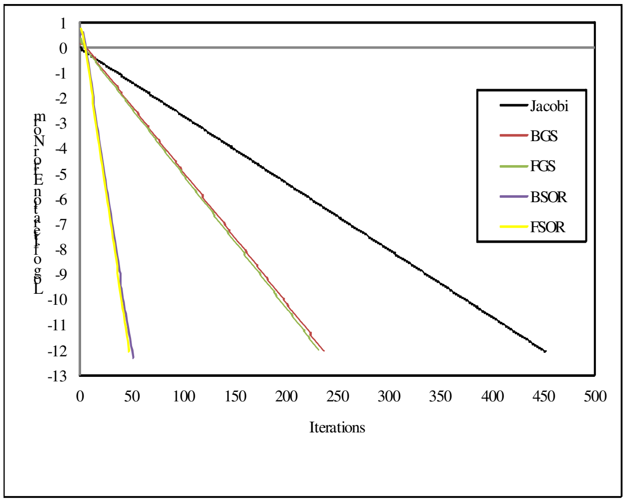
a) $N=120$



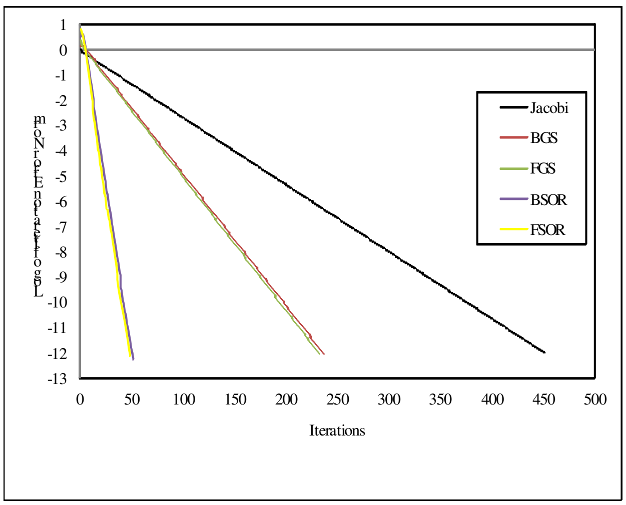
b) $N=240$



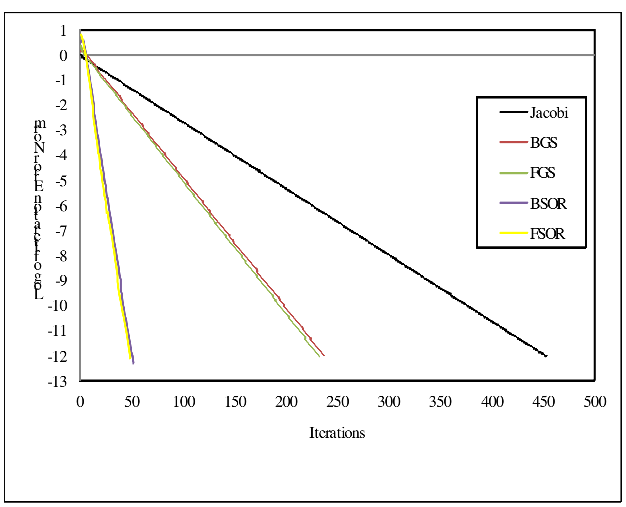
c) $N=480$



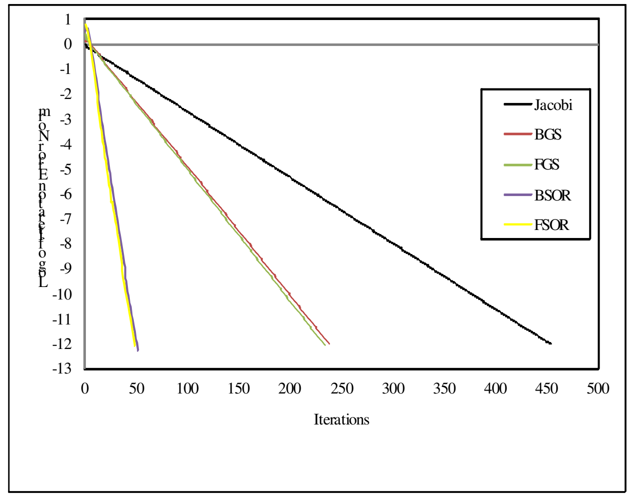
d) $N = 960$



e) $N = 1920$

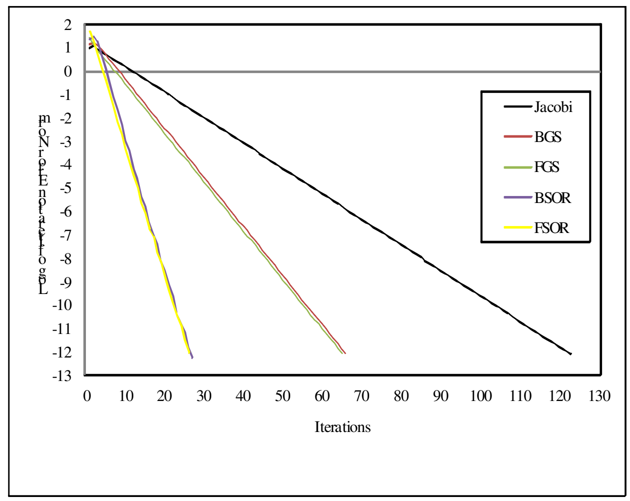


f) $N = 3840$

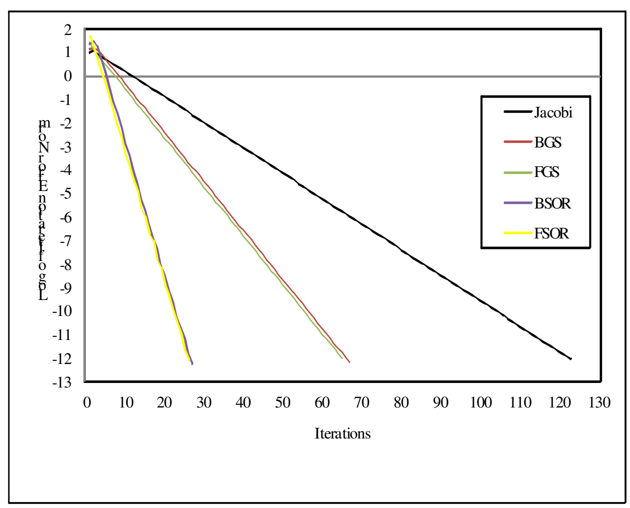


g) $N = 7680$

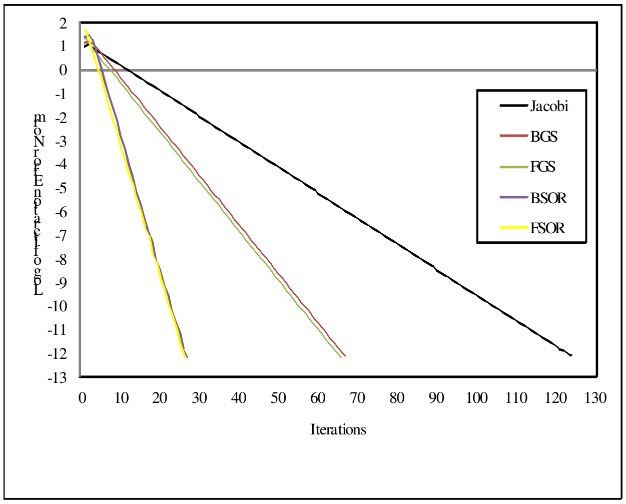
Fig. 1. a)-g) show the convergence histories for test problem 1



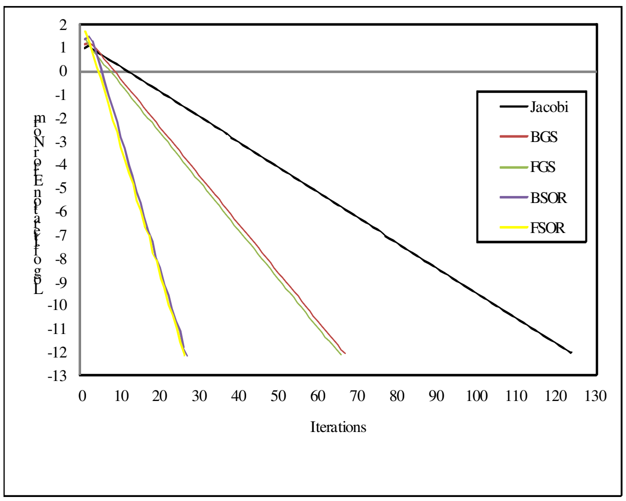
a) $N = 120$



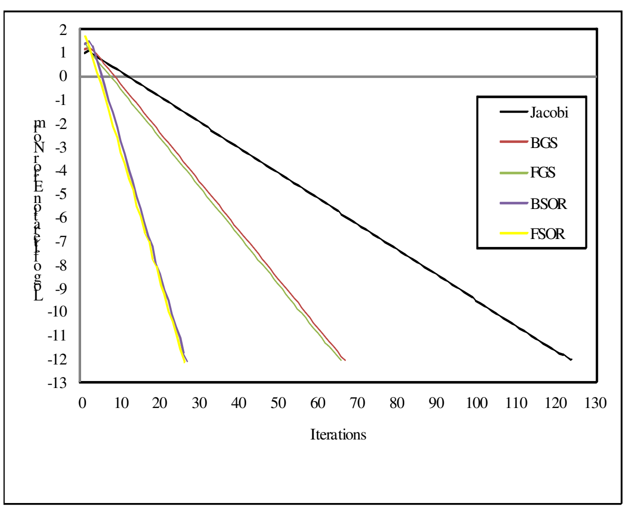
b) $N = 240$



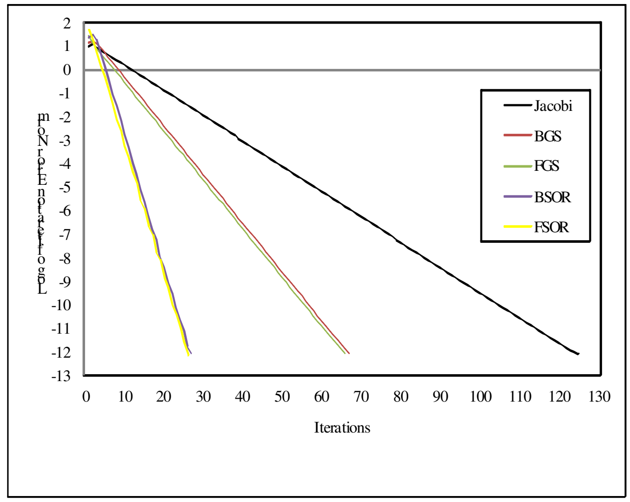
c) $N = 480$



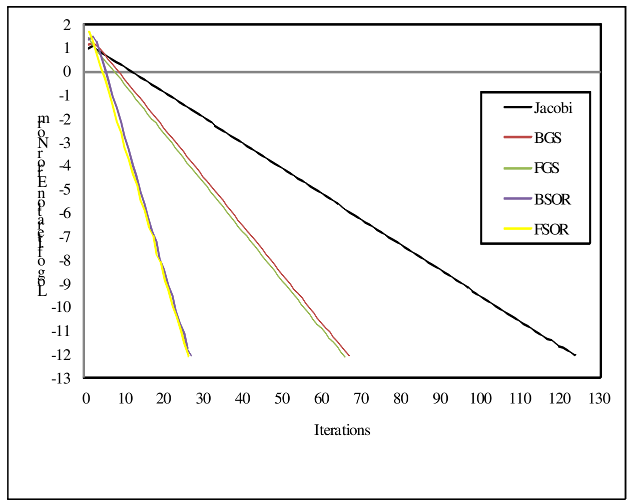
d) $N = 960$



e) $N = 1920$

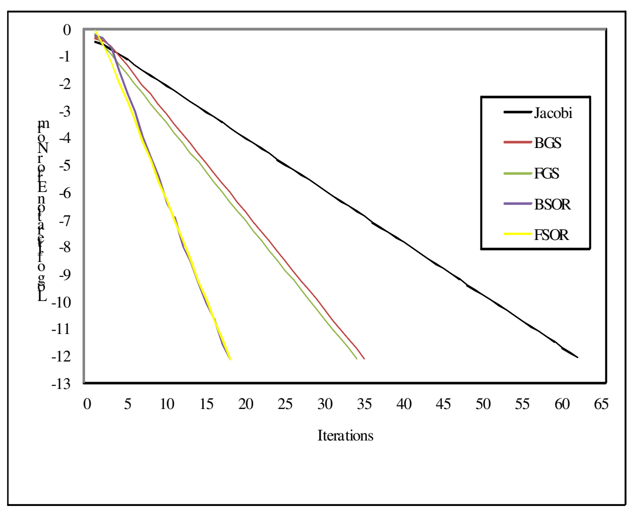


f) $N = 3840$

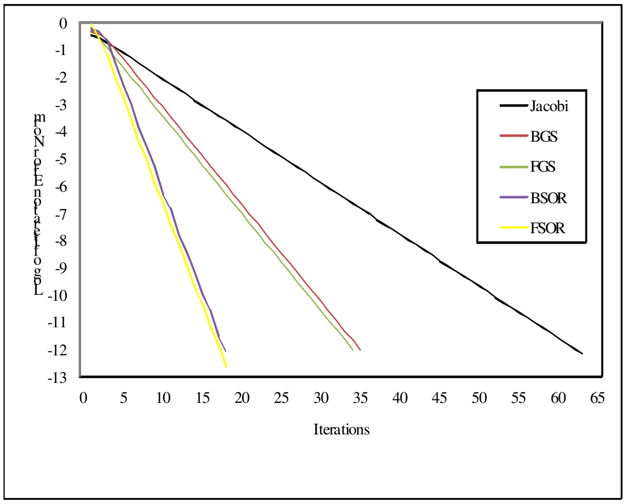


g) $N = 7680$

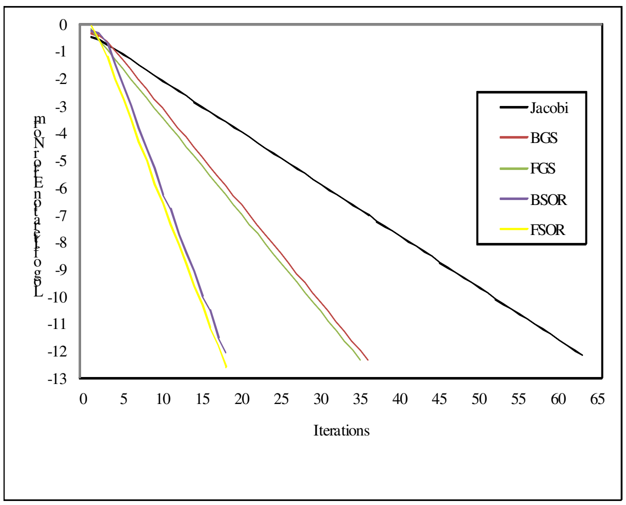
Fig. 2. a)-g) show the convergence histories for test problem 2



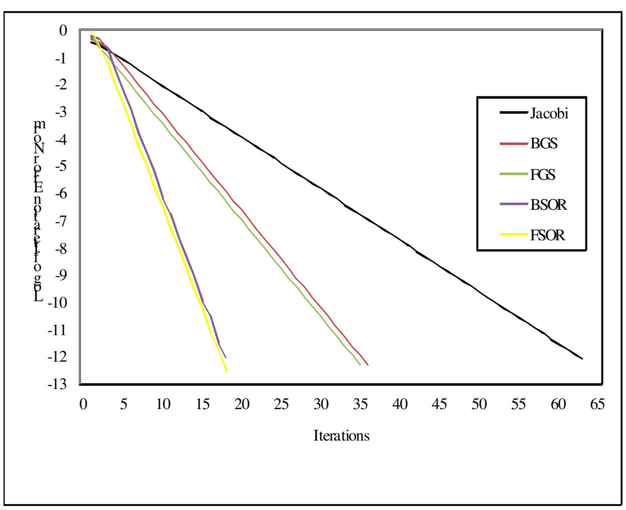
a) $N = 120$



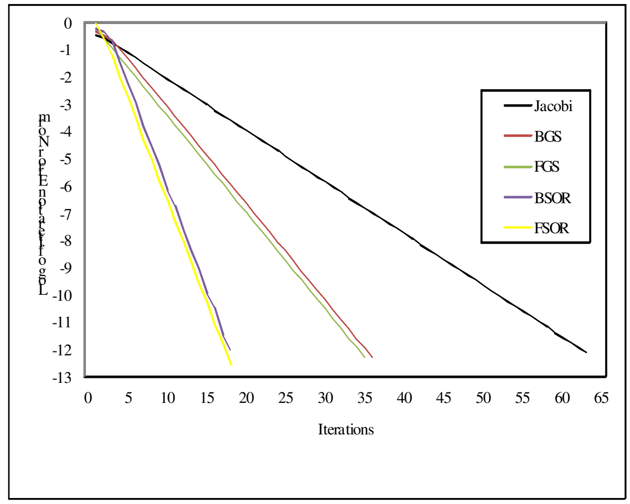
b) $N = 240$



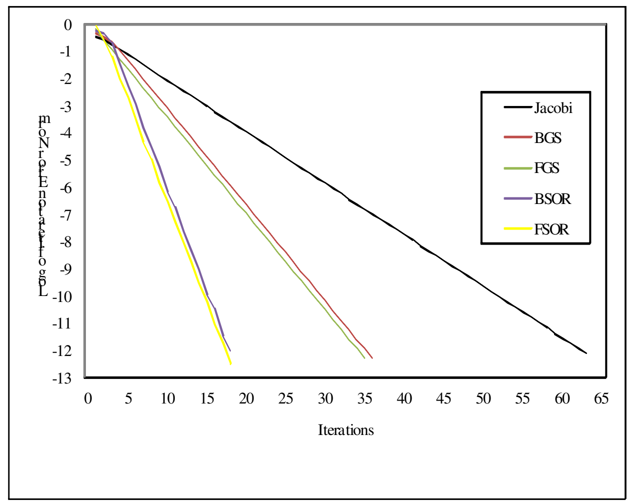
c) $N = 480$



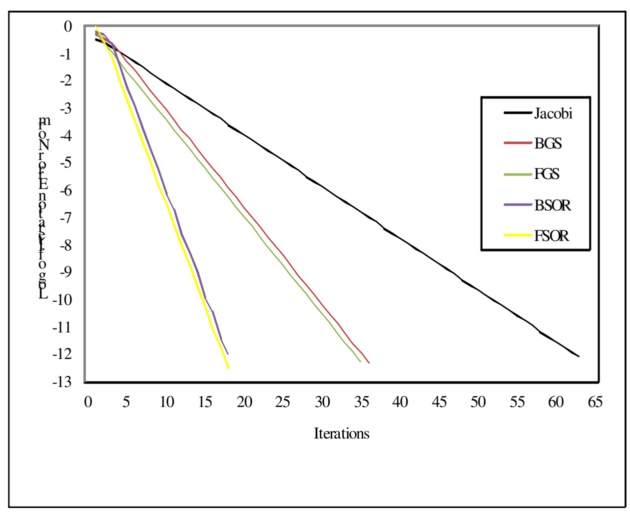
d) $N = 960$



e) $N = 1920$



f) $N = 3840$



g) $N = 7680$

Fig. 3. a)-g) show the convergence histories for test problem 3

Percentage gains in term of number of iterations and CPU time of BGS, FGS, BSOR and FSOR iterative methods compared to the Jacobi method are tabulated in Table 4.

Table 4. Percentage gains of the BGS, FGS, BSOR and FSOR methods compared with Jacobi method

		k	
Methods	Test Problem 1 (%)	Test Problem 2 (%)	Test Problem 3 (%)
BGS	47.46 - 47.58	45.52 - 46.40	42.85 - 44.45
FGS	48.56 - 48.79	46.77 - 47.20	44.44 - 46.04
BSOR	88.52 - 88.72	78.04 - 78.40	70.96 - 71.43
FSOR	89.18 - 89.41	78.86 - 79.20	70.96 - 71.43
		CPU	
Methods	Test Problem 1 (%)	Test Problem 2 (%)	Test Problem 3 (%)
BGS	46.98 - 47.62	45.37 - 46.40	42.85 - 44.83
FGS	48.19 - 48.80	46.34 - 47.23	42.85 - 44.83
BSOR	88.50 - 89.16	77.77 - 78.40	71.42 - 72.42
FSOR	89.15 - 89.44	78.04 - 79.21	71.42 - 72.42

V. Concluding Remarks

In the present paper, the performance of five conventional first order linear stationary iterative methods i.e. Jacobi, BGS, FGS, BSOR and FSOR for the solution of 3-CCNC system associated with the numerical solutions of the second kind linear Fredholm integral equations has been investigated. From the results obtained, it can be observed that FSOR method solved the test problems 1 and 2 with least number of iterations and lowest CPU time. Meanwhile performance of the FSOR method is comparable with BSOR method in solving test problem 3. In the aspect of accuracy, numerical solutions generated via FSOR method are slightly more precise relative to Jacobi, BGS, FGS and BSOR methods for test problem 1, as the N increases. Whereas, accuracy of numerical solutions obtained for test problems 2 and 3 are comparable for all the tested iterative methods. It also seems that the optimal value of relaxation parameter, ω lies in the range of $1 < \omega < 2$ for BSOR and FSOR methods. Overall, the FSOR iterative method is an efficient first order linear stationary iterative method among the tested methods for solving 3-CCNC system.

References

[1] C. Cattani, and A. Kudreyko, "Harmonic wavelet method towards solution of the Fredholm type integral equations of the second kind," *Applied Mathematics and Computation*, vol. 215, no. 12, pp. 4164-4171, 2010.

[2] A. Golbabai, and S. Seifollahi, "An iterative solution for the second kind integral equations using radial basis functions," *Applied Mathematics and Computation*, vol. 181, no. 2, pp. 903-907, 2006.

[3] C. -H. Hsiao, "Hybrid function method for solving Fredholm and Volterra integral equations of the second kind," *Journal of Computational and Applied Mathematics*, vol. 230, no. 1, pp. 59-68, 2009.

[4] K. Maleknejad, and M. T. Kajani, "Solving second kind integral equations by Galerkin methods with hybrid Legendre and Block-Pulse functions," *Applied Mathematics and Computation*, vol. 145, no. 2-3, pp. 623-629, 2003.

[5] K. Maleknejad, M. T. Kajani, and Y. Mahmoudi, "Numerical solution of linear Fredholm and Volterra integral equation of the second kind by using Legendre wavelets," *Kybernetes*, vol.32, no. 9/10, pp. 1530-1539, 2003.

[6] K. Maleknejad, and T. Lotfi, "Numerical expansion methods for solving integral equations by interpolation and Gauss quadrature rules," *Applied Mathematics and Computation*, vol. 168, no. 1, pp. 111-124, 2005.

[7] M. S. Muthuvalu, and J. Sulaiman, "Half-Sweep Arithmetic Mean method with composite trapezoidal scheme for solving linear Fredholm integral equations," *Applied Mathematics and Computation*, vol. 217, no. 12, pp. 5442-5448, 2011.

[8] M. S. Muthuvalu, and J. Sulaiman, "Numerical solution of second kind linear Fredholm integral equations using QSGS iterative method with high-order Newton-Cotes quadrature schemes," *Malaysian Journal of Mathematical Sciences*, vol. 5, no. 1, pp. 85-100, 2011.

[9] S. Rahbar, and E. Hashemizadeh, "A computational approach to the Fredholm integral equation of the second kind," in *Proceedings of the World Congress on Engineering*, 2008, pp. 933-937.

[10] J. Saberi-Nadjafi, and M. Heidari, "Solving linear integral equations of the second kind with repeated modified trapezoid quadrature method," *Applied Mathematics and Computation*, vol. 189, no. 1, pp. 980-985, 2007.

[11] W. Wang, "A new mechanical algorithm for solving the second kind of Fredholm integral equation," *Applied Mathematics and Computation*, vol. 172, no. 2, pp. 946-962, 2006.

Mohana Sundaram Muthuvalu received the B.Sc (Hons.) and Ph.D degrees in Mathematics with Economics and Mathematics from Universiti Malaysia Sabah, in 2006 and 2012 respectively. His research interests are in numerical analysis in solving differential and integral equations. He has published some research articles in journals and conferences.