J. Math. Study
doi: 10.4208/jms.v48n1.15.02

Vol. **48**, No. 1, pp. 18-29
March 2015

Self-adaptive Extrapolated Gauss-Seidel Iterative Methods

Guo-Yan Meng[1], Rui-Ping Wen[2,*]

[1] *Department of Mathematics, Xinzhou Teacher University, Xinzhou 034000, Shanxi Province, P.R. China.*
[2] *Department of Mathematics, Taiyuan Normal University, Taiyuan 030012, Shanxi Province, P.R. China.*

Received 19 November, 2014; Accepted (in revised version) 14 January, 2015

Abstract. In this paper, we consider a self-adaptive extrapolated Gauss-Seidel method for solving the Hermitian positive definite linear systems. Based on optimization models, self-adaptive optimal factor is given. Moreover, we prove the convergence of the self-adaptive extrapolated Gauss-Seidel method without any constraints on optimal factor. Finally, the numerical examples show that the self-adaptive extrapolated Gauss-Seidel method is effective and practical in iteration number.

AMS subject classifications: 65F10, 65F50, 15A06

Key words: Hermitian positive definite, Gauss-Seidel iteration, self-adaptive, extrapolated, linear systems.

1 Introduction

Consider the large sparse system of linear equations

$$Ax=b, \tag{1.1}$$

where $A=(a_{ij})\in\mathbb{C}^{n\times n}$ is a known nonsingular matrix and $x,b\in\mathbb{C}^n$ are vectors. The splitting iterative method is one of the important way for solving the linear systems (1.1). For any splitting $A=M-N$ with a nonsingular matrix M, the basic splitting iterative method can be expressed as

$$x^{(k+1)}=M^{-1}Nx^{(k)}+M^{-1}b,\quad k=0,1,\cdots. \tag{1.2}$$

Let $T=M^{-1}N$ and $c=M^{-1}b$. Then (1.2) can be also written as

$$x^{(k+1)}=Tx^{(k)}+c,\quad k=0,1,\cdots. \tag{1.3}$$

*Corresponding author. *Email addresses:* wenrp@tynu.edu.cn (R.-P. Wen)

Consider the usual splitting of A,

$$A = D - L - U, \tag{1.4}$$

where D is a diagonal matrix, $-L$ and $-U$ are the strictly lower and the strictly upper triangular parts of A, respectively. Taking $M = D - L$ in (1.2), the iteration (1.3) yields the classical Gauss-Seidel iterative method, and the iteration matrix of the Gauss-Seidel method is given by

$$T = (D - L)^{-1}U. \tag{1.5}$$

In order to accelerate the convergence of the Gauss-Seidel iterative method for solving the linear systems (1.1), Davod [4] presented the generalized Gauss-Seidel (GGS) iterative method, i.e., using the splitting $A = T_m - E_m - F_m$, where

$$T_m = (t_{ij}) = \begin{cases} -a_{ij}, & \text{for } |i-j| \le m, \\ 0, & \text{otherwise.} \end{cases}$$

So we can see that the GGS method is essentially block-type Gauss-Seidel method.

Gunawardena et. al [7] first introduced the modified Gauss-Seidel method with preconditioned technique. In particular, when coefficient matrix is an M-matrix, or a Z-matrix, or an H-matrix, many researchers [9, 13, 15, 20, 21, 23, 24] presented also the modifications and improvements of preconditioners to solve the linear systems (1.1). Recently, Kohno et. al [12], Kotakemori et. al [14] and Shen et. al [22] extended Gunawardena, Jain and Snyders' works to more general cases and employed new different modified Gauss-Seidel methods by using the general preconditioner P with various parameters. But these modifications and improvements of Gauss-Seidel iterative method are only theoretical and the numerical examples in those references are only small size problems, and the amount of calculation is rapidly increased with the problem size. And it is the most troubling that it is difficult in choosing many proper parameters for these methods.

With regard to parameters that have a significant effect on the convergence rate of the algorithm, Hadjidimos [8] considered the extrapolation and relaxation methods, Hadjidimos and Yeyios [10] proposed the extrapolated Gauss-Seidel method

$$x^{(k+1)} = [(1-\alpha)I + \alpha T]x^{(k)} + \alpha c, \tag{1.6}$$

where T is given by (1.5). Galanis et. al [6] presented methods that compute optimal relaxation factor in the different case for p-cyclic matrices. Bai et. al [1] and Chen et. al [2] considered the optimal convergence factor and the contraction factors of the GSOR method, respectively. Zhang et. al [27] presents a global relaxed non-stationary multisplitting multi-parameter method, while Migallón et. al [19] proposed non-stationary multisplitting with general weighting matrices.

But the optimal parameters or the weighting matrices of all above-mentioned methods are determined in advance, they are not known to be good or bad, this influences the efficiency of these iterative methods. Fortunately, the papers [26] and [18] have applied

the optimal model to compute the weight factors. Based on this strategy and the extrapolated methods [17], we consider the self-adaptive extrapolated Gauss-Seidel method for the Hermitian positive definite systems.

The rest of the paper is organized as follows. Next section is the preliminaries. In Section 3, We focus on to discuss the new algorithm and its convergence. Section 4 is devoted to the numerical experiments.

2 Preliminaries

In this section, we give some of the notations and lemmas which will be used in what follows.

As usual, $\mathbb{C}^{n\times n}$ is used to denote the $n\times n$ complex matrix set, and $\mathbb{C}^n$ the n-dimensional complex vector set. The matrix A^H denotes the conjugate transpose of A. Similarly the conjugate transpose of a vector x is denoted by x^H. A matrix $A\in\mathbb{C}^{n\times n}$ is called Hermitian positive definite (or semidefinite), if it is Hermitian and for all $x\in\mathbb{C}^n, x\neq 0$, it holds that $x^HAx>0$(or $x^HAx\geq 0$). If A is Hermitian positive definite, the norm

$$\|x\|_A=\|A^{1/2}x\|_2=(Ax,x)^{1/2} \tag{2.1}$$

is called the A-norm [5] or energy norms of the vector x.

Lemma 2.1. *(Convergence,* [25]*) Let $A=D-E-E^H$ be an $n\times n$ Hermitian matrix with D is Hermitian and positive definite and $D-E$ is nonsingular. Then, the Gauss-Seidel iterative method is convergent if and only if A is positive definite.*

Lemma 2.2. *(Extrapolation,* [10,17]*) The sufficient conditions for the convergence of the extrapolated Gauss-Seidel method (EGS)* (1.6) *are:*

1. *the original Gauss-Seidel method is convergent,*
2. $0<\alpha<2/(1+\rho(T))$*, where T is given by* (1.5)*.*

Lemma 2.3. ([11]) *If a nonsingular A is Hermitian positive definite, then there exists a unique Hermitian positive definite B such that $B^2=A$.*

Lemma 2.4. *(Cauchy-Schwarz inequality) If $(\cdot,\cdot)$ is an inner product on a vector space V over the field $\mathbb{C}$, then*

$$|(x,y)|^2\leq(x,x)(y,y),\quad for all\ x,y\in V. \tag{2.2}$$

Equality occurs if and only if x and y are linear dependent, that is, $x=\alpha y$ or $y=\alpha x$ for some $\alpha\in\mathbb{C}$.

Lemma 2.5. ([11]) *Every vector norm $\|\cdot\|$ on $\mathbb{C}^n$ is a uniformly continuous function.*

3 Self-adaptive EGS method

In this section, we give the self-adaptive extrapolated Gauss-Seidel method.

We assume that the symmetric positive definite matrix A has the following splitting, as (1.4)

$$A=D-L-L^T, \tag{3.1}$$

and the iteration matrix of the Gauss-Seidel method is given by

$$T=(D-L)^{-1}L^T. \tag{3.2}$$

Algorithm 3.1.(*The Self-adaptive Extrapolated GS(SEGS) Iterative Method*)

Let $x^{(0)}\in\mathbb{C}^n$ be an arbitrary initial guess. For $k=0,1,2,\ldots$ until the sequence of iterations $\{x^{(k)}\}_{k=0}^{\infty}\subset\mathbb{C}^n$ converges, compute $x^{(k+1)}$ by the following scheme:

$$\widetilde{x}^{(k)}=Tx^{(k)}+(D-L)^{-1}b, \tag{3.3}$$

$$x^{(k+1)}=\alpha\widetilde{x}^{(k)}+(1-\alpha)x^{(k)}, \tag{3.4}$$

where α is the solution of the following optimization model

$$\min_{\alpha}\frac{1}{2}(x^{(k)})^HAx^{(k)}-(x^{(k)})^Hb. \tag{3.5}$$

Alternatively, (3.4) can be rewritten as:

$$x^{(k+1)}=x^{(k)}+\alpha_k(D-L)^{-1}r^{(k)}, \tag{3.6}$$

where

$$r^{(k)}=b-Ax^{(k)}, \tag{3.7}$$

and α_k is obtained by the optimization model (3.5). By simplified derivation, (3.5) yields the desired value of α_k

$$\begin{aligned}\alpha_k&=\frac{\left((D-L)^{-1}r^{(k)}\right)^Hr^{(k)}}{\left((D-L)^{-1}r^{(k)}\right)^HA(D-L)^{-1}r^{(k)}}\\&=\frac{\left((D-L)^{-1}r^{(k)},r^{(k)}\right)}{\left((D-L)^{-1}r^{(k)},A(D-L)^{-1}r^{(k)}\right)}.\end{aligned} \tag{3.8}$$

Accordingly, the iteration matrix G_{α_k} at the k-th step for the Algorithm 3.1 is

$$\begin{aligned} G_{\alpha_k} &= \alpha_k T + (1-\alpha_k) I \\ &= \alpha_k (D-L)^{-1} L^H + (1-\alpha_k)(D-L)^{-1}(D-L) \\ &= (D-L)^{-1}\Big(\alpha_k L^H + (1-\alpha_k)(D-L)\Big) \\ &= I - \alpha_k (D-L)^{-1} A. \end{aligned} \tag{3.9}$$

If α_k is independent on the iteration index k, Algorithm 3.1 will be the extrapolated Gauss-Seidel method. From Lemma 2.1 and Lemma 2.2, we know that the extrapolated Gauss-Seidel method of the Hermitian positive definite matrix is convergent with $0<\alpha<2/(1+\rho(T))$. But our Algorithm 3.1 will be proved convergence without any constraints on α.

Theorem 3.1. *Let A be a Hermitian positive definite matrix. Then Algorithm* 3.1 *is convergent with respect to $\|\cdot\|_A$ for any choice of the initial guess $x^{(0)}$.*

Proof. Let x^* be the unique solution of linear system of equations (1.1), $x^{(k)}$ be the solution generated by the Algorithm 3.1 at the k-th step, and

$$\varepsilon^{(k)} = x^{(k)} - x^*, \ \ k = 0,1,2,\cdots$$

be the error at the k-th step of iteration. And the following can be obtain from (3.7),

$$r^{(k)} = b - Ax^{(k)} = Ax^* - Ax^{(k)} = -A\varepsilon^{(k)}. \tag{3.10}$$

Similar to (3.10),

$$r^{(k+1)} = -A\varepsilon^{(k+1)}. \tag{3.11}$$

For simplification, we define

$$z^{(k)} = (D-L)^{-1} r^{(k)}. \tag{3.12}$$

Due to the properties of the matrix A and the definition of A-norm (2.1), we have

$$\begin{aligned}
\|\varepsilon^{(k+1)}\|_A^2 &= \left(A\varepsilon^{(k+1)},\varepsilon^{(k+1)}\right) = \left(-r^{(k+1)},G_{\alpha_k}\varepsilon^{(k)}\right)\\
&= \left(-r^{(k)}+\alpha_k A(D-L)^{-1}r^{(k)},\left(I-\alpha_k(D-L)^{-1}A\right)\varepsilon^{(k)}\right)\\
&= \left(-r^{(k)},\varepsilon^{(k)}\right)+\alpha_k\left(\left(r^{(k)},(D-L)^{-1}A\varepsilon^{(k)}\right)+\left(Az^{(k)},\varepsilon^{(k)}\right)\right)\\
&\quad -\alpha_k^2\left(Az^{(k)},(D-L)^{-1}A\varepsilon^{(k)}\right)\\
&= \|\varepsilon^{(k)}\|_A^2+\frac{\left(z^{(k)},r^{(k)}\right)}{\left(z^{(k)},Az^{(k)}\right)}\bullet\\
&\quad \left(\left(r^{(k)},-z^{(k)}\right)+\left(z^{(k)},-r^{(k)}\right)-\frac{\left(z^{(k)},r^{(k)}\right)}{\left(z^{(k)},Az^{(k)}\right)}\left(Az^{(k)},-z^{(k)}\right)\right)\\
&= \|\varepsilon^{(k)}\|_A^2-\frac{\left(z^{(k)},r^{(k)}\right)^2}{\left(z^{(k)},Az^{(k)}\right)}.
\end{aligned}\tag{3.13}$$

By Lemma 2.3, there exists a Hermitian positive definite matrix $B=A^{1/2}$, such that

$$\begin{aligned}
\frac{\|\varepsilon^{(k+1)}\|_A^2}{\|\varepsilon^{(k)}\|_A^2} &= 1-\frac{\left(z^{(k)},r^{(k)}\right)^2}{\left(z^{(k)},Az^{(k)}\right)\left(A\varepsilon^{(k)},\varepsilon^{(k)}\right)}\\
&= 1-\frac{\left(z^{(k)},-A\varepsilon^{(k)}\right)^2}{\left(z^{(k)},Az^{(k)}\right)\left(A\varepsilon^{(k)},\varepsilon^{(k)}\right)}\\
&= 1-\frac{\left(A^{1/2}z^{(k)},A^{1/2}\varepsilon^{(k)}\right)^2}{\left(A^{1/2}z^{(k)},A^{1/2}z^{(k)}\right)\left(A^{1/2}\varepsilon^{(k)},A^{1/2}\varepsilon^{(k)}\right)}.
\end{aligned}\tag{3.14}$$

And in (3.14) the vectors $A^{1/2}z^{(k)}$ and $A^{1/2}\varepsilon^{(k)}$ are linear independent. If not, there exists a real β, so

$$\begin{aligned}
&z^{(k)}=\beta\varepsilon^{(k)},\\
\Leftrightarrow\quad &(D-L)^{-1}r^{(k)}=\beta A^{-1}r^{(k)},\\
\Leftrightarrow\quad &\beta\det(D-L)=\det(A),\quad \text{or } r^{(k)}=0.
\end{aligned}$$

But on the account of (3.1) and (3.10) the equations (3.15) is, in general, not true, so we are done. This means that the equality does not hold for Cauchy-Schwarz inequality (2.2), namely

$$0<\frac{\left(A^{1/2}z^{(k)},A^{1/2}\varepsilon^{(k)}\right)^2}{\left(A^{1/2}z^{(k)},A^{1/2}z^{(k)}\right)\left(A^{1/2}\varepsilon^{(k)},A^{1/2}\varepsilon^{(k)}\right)}<1.\tag{3.15}$$

Then, it follows from (3.15) and (3.14)

$$\frac{\|\varepsilon^{(k+1)}\|_A}{\|\varepsilon^{(k)}\|_A} < 1. \tag{3.16}$$

Consequently, by Lemma 2.5 there exist a positive number $\rho < 1$ such that

$$\frac{\|\varepsilon^{(k+1)}\|_A}{\|\varepsilon^{(k)}\|_A} \leq \rho, \tag{3.17}$$

which is equivalent to

$$\lim_{k\to\infty} \varepsilon^{(k+1)} = 0.$$

Hence, we have proved this theorem. □

Remark 3.1. For Algorithm 3.1, the optimization model in each iteration require the 'exact' solution. However, this may be very costly and impractical in actual implementations, particularly when the scale of the original problem is very large. To improve the computing efficiency of the Algorithm 3.1, similar to the Algorithm 4.2.1 of [16], we will employ a simple strategy to perform a new updating after k iteration steps.

4 Numerical experiments

In this section, we use three different numerical examples to show the feasibility and effectiveness of Algorithm 3.1 when it is used to find the solution of the linear systems (1.1) with Hermitian positive definite coefficient matrix. The results of all examples are compared with some of the better existing iterative methods. That is to say, we solve the system of linear equations (1.1) by Algorithm 3.1, the basic Gauss-Seidel(GS) method and the extrapolated Gauss-Seidel (EGS) method with some arbitrary values of α. Also, we solve the system of linear equations (1.1) by the GGS method presented by Davod [4].

For each example, we compare all the above methods from the view of the iteration numbers (denoted by IT) and in terms of the total CPU execution time (denoted by CPU). All timing results are reported in seconds. In the following tables, k stands for a new updating after k iteration steps.

In our implementations, the initial vector $x^{(0)}$ is set to zero and the iteration is terminated when the current iterate satisfied

$$\frac{\|b - Ax^{(k)}\|_2}{\|b\|_2} \leq 10^{-6}, \tag{4.1}$$

where $\|\cdot\|_2$ refers to L_2-norm. In addition, all codes were run in MATLAB [version 7.12.0.635 (R2011a)] in double precision and all experiments were performed on a personal computer with 3.20GHz central processing unit [Intel(R) Pentium(R)(TM)2 CPU G2130], 4G memory and Microsoft Window 7 operating system (6.1).

Table 1: IT and CPU for SEGS, EGS, GGS and GS methods for Example 4.1.

p		SEGS	EGS		GGS	GS
		$k=3$	$\alpha=1.2$	$\alpha=1.6$		
20	IT	83	351	262	254	422
	CPU	0.004764	0.017525	0.012783	0.333172	0.018299
40	IT	141	1220	914	880	1465
	CPU	0.016766	0.113716	0.090983	5.556272	0.136999
60	IT	341	2546	1908	1835	3056
	CPU	0.075822	0.486628	0.364732	28.617407	0.579725
80	IT	582	4294	3219	3093	5153
	CPU	0.226410	1.356569	1.009542	91.436310	1.652199
100	IT	896	6439	4828	4637	7727
	CPU	0.538224	3.297829	2.443553	225.282633	3.911060
120	IT	1216	8962	6721	6454	10755
	CPU	1.197484	7.282420	5.268566	465.063581	8.453618
140	IT	1874	11849	8886	8533	14220
	CPU	2.632422	13.892134	10.198586	865.481567	16.005318

Example 4.1 The test PDE problem we are considering in this Example is

$$-\Delta u \equiv -\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=f(x,y) \tag{4.2}$$

with $(x,y)\in\Omega$, where $\Omega=(0,1)\times(0,1)$ is a square region. For the test problem, only the matrix A, which is constructed from nine-point finite difference discretization of the given PDE (4.2), is of importance, so the right-hand side vector b is created artificially. Hence, the right-hand side function $f(x,y)$ in Examples 4.1 is not relevant. So the coefficient matrix

$$A=\text{tridiag}(D,G,D)\in\mathbb{R}^{q\times q},$$

where

$$D=\text{tridiag}(-4,20,-4)\in\mathbb{R}^{p\times p},G=\text{tridiag}(-1,-4,-1)\in\mathbb{R}^{p\times p}.$$

and the right-hand side vector b is chosen so that $b=Ae$ with e being the vector of all entries equal to 1. For this example, we set $p=q$.

In Table 1 we list the iteration numbers and the CPU times in seconds with respect to SEGS, EGS, GGS and GS methods.

Example 4.2 Consider the following system of linear equations, for which $A=(a_{k,j})\in\mathbb{C}^{n\times n}$ is defined as follows:

$$a_{k,j}=\begin{cases} 8, & \text{for } j=k;\\ -1, & \text{for } \max\{1,k-4\}\le j\le\{n,k+4\},k\neq j;\\ 0, & \text{otherwise.}\end{cases}$$

Table 2: Comparison of computational results for Example 4.2.

n		SEGS	EGS		GGS	GS
		$k=3$	$\alpha=1.3$	$\alpha=1.7$		
200	IT	379	5857	4477	5711	7616
	CPU	0.016880	0.195555	0.151929	0.413341	0.257282
300	IT	1208	13059	9985	12374	16979
	CPU	0.052650	0.487455	0.375210	0.838990	0.627948
400	IT	1958	23112	17672	22535	-
	CPU	0.094912	0.959881	0.737526	1.835402	-
500	IT	4058	-	27538	-	-
	CPU	0.223778	-	1.257733	-	-
600	IT	6155	-	-	-	-
	CPU	0.358920	-	-	-	-
800	IT	14774	-	-	-	-
	CPU	0.974303	-	-	-	-
1000	IT	26174	-	-	-	-
	CPU	1.998696	-	-	-	-

Notes. the symbol "$-$" denotes that the iteration is failing.

Table 3: Comparison of computational results for Example 4.3.

m		SEGS	EGS		GGS	GS
		$k=2$	$\alpha=1.2$	$\alpha=1.5$		
40	IT	244	1077	861	650	1294
	CPU	0.025882	0.090070	0.069360	3.753191	0.106702
60	IT	715	2383	1905	1433	2861
	CPU	0.135437	0.384121	0.310215	23.252445	0.464395
80	IT	1379	4201	3360	2524	5043
	CPU	0.420499	1.101609	0.879608	72.220400	1.305668
100	IT	2303	6531	5223	3922	7838
	CPU	1.096738	2.599714	2.053889	185.670557	3.041264
120	IT	3443	9372	7496	5626	11248
	CPU	2.443417	5.284361	4.215272	414.458203	6.519183
140	IT	4741	12725	10179	7638	15271
	CPU	4.835817	10.409126	8.386267	839.549724	12.475579
160	IT	6359	16590	13270	9957	19909
	CPU	9.122510	18.628376	14.842852	1446.824142	22.187892

We chose the right-hand side vector $b=(1,1,\cdots,1)^T$. The above coefficient matrix is ill-conditioned if n is large.

The following Table 2 shows the iteration numbers and the CPU times in seconds

Table 4: Comparison of computational results for Example 4.4.

m		SEGS	EGS		GGS	GS
		$k=2$	$\alpha=1.2$	$\alpha=1.4$		
16	IT	155	343	293	7	413
	CPU	0.009617	0.015375	0.014176	0.010085	0.019080
32	IT	487	1391	1191	7	1670
	CPU	0.057962	0.135159	0.118799	0.052460	0.164740
64	IT	2181	5556	4762	7	6669
	CPU	0.820887	1.658372	1.421410	0.223723	2.076621
96	IT	4337	12454	10674	7	14946
	CPU	3.657569	7.898573	6.892787	0.581035	9.832173
128	IT	7790	22069	18916	7	26484
	CPU	11.980302	25.932632	22.352691	0.935597	32.644396
160	IT	12109	-	29483	7	-
	CPU	32.774894	-	58.186521	1.752083	-
192	IT	17903	-	-	7	-
	CPU	68.103404	-	-	3.301699	-

Notes. the symbol "$-$" denotes that the iteration is failing.

with respect to SEGS, EGS, GGS and GS methods.

Example 4.3 Consider linear equations with Hermitian positive definite coefficient matrix that rise in the reference [26], we choose $A=0.1\pi I+\mu K$ with $\mu=0.02$, and the matrix K possesses the form $K=I\otimes V+V\otimes E$ with

$$V=(m+1)^2\text{tridiag}(-1,2,1)\in\mathbb{R}^{m\times m},$$

where I is the identity matrix, and $\otimes$ is the Kronecker product symbol. Hence, A is an $n\times n$ block-tridiagonal matrix, with $n=m^2$. We take the right-hand side vector $b=(1,1,\cdots,1)^T$.

Table 3 shows the iteration numbers and the CPU times in seconds with respect to SEGS, EGS, GGS and GS methods.

Example 4.4 We consider the complex linear system $Ax=b$, with the Hermitian positive definite coefficient matrix $A=W+iT+(8-4i)I\in\mathbb{C}^{n\times n}$ being given by

$$W=\text{tridiag}(c,a,c)\in\mathbb{R}^{n\times n},\quad T=I\otimes V_c+V_c\otimes I\in\mathbb{R}^{n\times n},\tag{4.3}$$

with

$$V_c=V-e_1e_m^T+e_me_1^T\in\mathbb{R}^{m\times m}$$

and

$$\begin{aligned}
&V=\text{tridiag}(-1,2,1)\in\mathbb{R}^{m\times m},\\
&e_1=(1,0,\cdots,0)^T\in\mathbb{R}^m,\quad e_m=(0,\cdots,0,1)^T\in\mathbb{R}^m,\\
&a=(1,3,5,\cdots,2n-3,2n-1)^T\in\mathbb{R}^n,\\
&c=(-1,-2,\cdots,-(n-1))^T\in\mathbb{R}^{n-1}.
\end{aligned}$$

The right-hand side vector b is defined as $b=Ax_*$, with $x_*=(1,2,\cdots,n)^T\in\mathbb{R}^n$.

In Table 4 we report results for SEGS, EGS, GGS and GS methods.

We see from Tables $1-4$ that for all methods, the iteration numbers and the CPU times grow with problem size. However, this growth is slower for SEGS than for other methods, such as GS, GGS and EGS with fixed extrapolated factors. The reason is that the stationary of weighting factors is deleted, the range for finding the optimal weighting factors is extended. Hence, we show that our SEGS method with fixed initial guess $\alpha=1$ is practical and effective.

Acknowledgments

This work was supported by NSFC (No.11371275) and NSF of Shanxi Province (No.2014011006-1, 2012011015-6).

References

[1] Z.-Z. Bai, B. N. Parlett and Z.-Q. Wang. On generalized successive overrelaxation methods for augmented linear systems. Numer. Math., 2005, 102: 1–38.
[2] F. Chen, Y.-L. Jiang and B. Zheng. On contraction and semi-contration factors of GSOR method for augmented linear systems. J. Comput. Math., 2010, 28(6): 901–912.
[3] F. Chen, Q.-Q. Liu. On semi-convergence of modified HSS iteration methods. Numer. Algor., 2013, 64: 507C-518.
[4] K. S. Davod. Generalized Jacobi and Gauss-Seidel methods for solving linear system of equations. Numer. Math. J. Chinese Univ.(English Ser.), 2007, 16(2): 164–170.
[5] A. Frommer and D. B. Szyld. Weighted max norms, splitttngs, and overlapping additive Schwarz iterations. Numer. Math., 1999, 83(2): 259–278.
[6] S. Galanis, A. Hadjidimos, D. Noutsos and M. Tzoumas. On the optimum relaxation factor associated with p-cyclic matrices. Linear Algebra Appl., 1992, 162: 433–445.
[7] A. D. Gunawardena, S. K. Jain and L. Snyder. Modified iterative method for consistent linear systems. Linear Algebra Appl., 1991, 154-156: 123–143.
[8] A. Hadjidimos. A survey of the iterative methods for the solution of linear systems by extrapolation, relaxation and other techniques. Internatl. J. Comput. Math., 1987, 20: 37–51.
[9] A. Hadjidimos, D. Noutsos and M. Tzoumas. More on modifications and improvements of classical iterative schemes for M-matrices. Linear Algebra Appl., 2003, 364: 253–279.
[10] A. Hadjidimos and A. Yeyios. The principle of extrapolation in connection with the accelerated overrelaxation method. Linear Algebra Appl., 1980, 30: 115–128.

[11] R. A. Horn and C. R. Johnson. Matrix analysis. Cambridge University Press, England, 1986.
[12] T. Kohno, H. Kotakemori, H. Niki and M. Usui. Improving modified iterative methods for Z-matrices. Linear Algebra Appl., 1997, 267: 113–123.
[13] H. Kotakemori, K. Harada, M. Morimoto and H. Niki. A comparison theorem for the iterative method with the preconditioner $(I+S_{max})$. J. Comput. Appl. Math., 2002, 145: 373–378.
[14] H. Kotakemori, H. Niki and N. Okamoto. Accelerated iterative method for Z-matrices. J. Comput. Appl. Math., 1996, 75(1): 87–97.
[15] W. Li and W. Sun. Modified Gauss-Seidel type methods and Jacobi type methods for Z-matrices. Linear Algebra Appl., 2000, 317: 227–240.
[16] W.-W. Lin. Lecture notes of matrix computations. National Tsing Hua University, Hsinchu, Taiwan, 2008.
[17] M. M. Martins. On an accelerated overrelaxation iterative method for linear systems with strictly diagonally dominant matrix. Math. Comput., 1980, 35(152): 1269–1273.
[18] G.-Y. Meng, C.-L.Wang and X.-H. Yan. Self-adaptive non-stationary parallel multisplitting two-stage iterative methods for linear systems. Lecture Notes in Computer Science, 2012, 7696: 38–47.
[19] V. Migallón, J. Penadés and D. B. Szyld. Nonstationary multisplittings with general weighting matrices. SIAM J. Matrix Anal. Appl., 2001, 22: 1089–1094.
[20] H. Niki, K. Harada, M. Morimoto and M. Sakakihara. The survey of preconditioners used for accelerating the rate of convergence in the Gauss-Seidel method. J. Comput. Appl. Math., 2004(164–165): 587–600.
[21] H. Niki, T. Kohno and M. Morimoto. The preconditioned Gauss-Seidel method faster than the SOR method. J. Comput. Appl. Math., 2008, 218(1): 59–71.
[22] H. Shen, X. Shao, Z. Huang and C. Li. Precondition Gauss-Seidel iterative method for Z-matrices linear systems. Bull. Korean Math. Soc., 2011, 48(2): 303–314.
[23] L.-Y. Sun. Some extensions of the improved modified Gauss-Seidel iterative method for H-matrices. Numer. Linear Algebra Appl., 2006, 13(10): 869-876.
[24] M. Usui, H. Niki and T. Kohno. Adaptive Gauss-Seidel method for linear systems. Inter. J. Comput. Math., 1994, 51: 119–125.
[25] R. S. Varga. Matrix iterative analysis. Springer Berlin Heidelberg, Germany, 2009.
[26] C.-L. Wang, G.-Y. Meng and Y.-H. Bai. Practical convergent splittings and acceleration methods for non-hermitian positive definite linear systems. Adv. Comput. Math., 2012, doi: 10.1007/s10444-012-9278-8.
[27] L.-T. Zhang, T.-Z. Huang and T.-X. Gu. Global relaxed non-stationary multisplitting multiparameter methods Internatl. J. Comput. Math., 2008, 85(2): 211–224.