NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2010; **17**:3–15
Published online 13 March 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla.643

Block s-step Krylov iterative methods

Anthony T. Chronopoulos[1,*,†] and Andrey B. Kucherov[2]

[1]*Department of Computer Science, University of Texas at San Antonio, 6900 North Loop 1604 West, San Antonio, TX 78249, U.S.A.*
[2]*Synopsys Inc., 700 East Middlefield Road, Mountain View, CA 94043, U.S.A.*

SUMMARY

Block (including s-step) iterative methods for (non)symmetric linear systems have been studied and implemented in the past. In this article we present a (combined) block s-step Krylov iterative method for nonsymmetric linear systems. We then consider the problem of applying any block iterative method to solve a linear system with one right-hand side using many linearly independent initial residual vectors. We present a new algorithm which combines the many solutions obtained (by any block iterative method) into a single solution to the linear system. This approach of using block methods in order to increase the parallelism of Krylov methods is very useful in parallel systems. We implemented the new method on a parallel computer and we ran tests to validate the accuracy and the performance of the proposed methods. It is expected that the block s-step methods performance will scale well on other parallel systems because of their efficient use of memory hierarchies and their reduction of the number of global communication operations over the standard methods. Copyright © 2009 John Wiley & Sons, Ltd.

Received 10 October 2008; Revised 6 February 2009; Accepted 8 February 2009

KEY WORDS: block; s-step; iterative methods

1. INTRODUCTION

Several authors have studied the block methods for solving linear systems (see, for example, [1–6] and references therein). In general, the block methods are used to solve linear systems with a single coefficients matrix and a number of right-hand sides. This number is called the *blocksize* of the method. However, these block methods can also be used in solving a linear system with a single right-hand side if a number of *linearly independent initial residual* vectors are provided. The residual vectors are used to compute simultaneously direction vectors that are orthonormalized (e.g. via modified Gram–Schmidt (*MGS*)). These direction vectors are then used to advance the solutions and compute the new residual vectors. At the end all the solutions are combined to obtain

*Correspondence to: Anthony T. Chronopoulos, Department of Computer Science, University of Texas at San Antonio, 6900 North Loop 1604 West, San Antonio, TX 78249, U.S.A.
†E-mail: atc@cs.utsa.edu

a single solution. This approach was studied in the past for symmetric and positive-definite linear systems in [4].

The s-step Krylov methods for solving nonsymmetric linear systems with a single right-hand side were proposed and studied in the past [7, 8]. These methods exhibit better parallel properties than the standard methods. These methods form, at each iteration, s independent direction vectors using repeated matrix–vector products of the coefficient matrix with a single residual vector [7]. Then the solution is advanced simultaneously using the s direction vectors. The *orthogonal s-Step GCR/Orthomin* was introduced in [8]. In the Orthogonal s-Step GCR/Orthomin (OSGCR(s)/OSOmin(s,k)) (where s, k are positive integers, [8]), MGS is used to orthonormalize the direction vectors within each block. The convergence study of the s-step methods can be found in [7, 8].

Here, we attempt to enhance further the parallel properties of the s-step method by proposing block s-step methods. At first, a new block OSGCR/OSOmin algorithm for the solution of nonsymmetric linear systems with multiple right-hand sides is obtained by turning the OSGCR/OSOmin into a block algorithm. Then we use many initial guesses and combine the final solution approximations into a single solution. We also propose an entirely new method that can be used to optimally combine the many solutions of a block method with many initial guesses into a single solution.

The new block OSGCR/OSOmin is expected to exhibit similar convergence properties in reference to (the class of) the matrix of coefficients as in the s-step methods, assuming that the initial residual vectors are linearly independent. By varying the parameters k, s in the OSGCR(s)/OSOMin(s,k) method one obtains methods mathematically equivalent to other well-known Krylov methods (e.g. OSOmin(1,k) is equivalent to Omin(k) (or Orthomin(k)), OSGCR(s) is equivalent to Odir(s) and GMRES(s) (see [7, 9] and references therein). Thus, we only test the block OSOmin method for various k, s and we expect these comparisons to hold for other well-known methods.

Finally, we present a parallel implementation of our algorithms on a model problem arising in the discretization of elliptic partial differential equations using a parallel preconditioning to accelerate the convergence. We present runs on a parallel computer to validate the convergence and performance of our algorithms.

Notations: (1) The norms used are the 2-norms and the associated inner products throughout the article unless otherwise stated. (2) We use English upper case letters to denote matrices, English or Greek lower case letters to denote scalars and English letters with a bar to denote vectors.

The article follows the following structure. In Section 2, a block s-step method is presented and studied. In Section 3, a new solution averaging method for combining several solutions to a linear system into a single solution is proposed and studied. In Sections 4 and 5, a partial differential equation problem discretization that generates a large sparse matrix and a parallel preconditioner are described. In Section 6, implementation and test results are presented. In Section 7, we draw conclusions.

2. A BLOCK S-STEP ITERATIVE METHOD FOR NONSYMMETRIC LINEAR SYSTEMS

We first review the OSGCR/OSOMIN methods [7]. Then we present the block versions of the methods and we obtain their complexities.

DOI: 10.1002/nla

First, let us consider a linear system with a single right-hand side: $A\bar{x} = \bar{f}$. Let us consider the OSGCR and OSOmin are applied to a linear system with a single right-hand side. For positive integers i, k, s, we denote by $j_i = 1$ for OSGCR(s) and $j_i = \max(1, i-k+1)$ for OSOmin(s,k) the index parameters used in updating the direction vectors [7, 8, 10]. At iteration i, let $\bar{r}_i$ denote the residual error vector. At iteration i, the method generates s direction vectors, which are denoted by the *matrix* $P_i = [\bar{p}_i^1, \ldots, \bar{p}_i^s]$. First, AP_i is obtained from the vectors $[A\bar{r}_i, A^2\bar{r}_i, \ldots, A^s\bar{r}_i]$, by simultaneously $A^{\mathrm{T}}A$ -orthogonalizing them against the preceding blocks of direction vectors and then orthogonalizing them among themselves. Then the direction vectors P_i are formed using the same linear combinations (as in AP_i) starting with the vectors $[\bar{r}_i, A\bar{r}_i, \ldots, A^{s-1}\bar{r}_i]$. The norm of the residual $\|\bar{r}_{i+1}\|_2$ is minimized simultaneously in all s new direction vectors in order to obtain new solution approximation $\bar{x}_{i+1}$. All orthogonalizations apply the Modified Gram–Schmidt algorithm (MGS) [8].

We next present the block OSGCR(s)/OSOmin(s,k) (BOSGCR(s,b)/BOSOmin(s,b,k)), where b is the blocksize. The methods (OSGCR(s)/OSOmin(s,k)) have been introduced and analyzed in the past ([7, 8] and references therein). OSGCR is a full orthogonalization method and it is implemented as a restarted method and OSOmin(s,k) is a truncated method. These methods are the *s-step* extensions of GCR (which is a full orthogonalization and it is implemented as a restarted method) and Omin(k) (or Orthomin(k)) (which is a truncated method) ([9, 10] and references therein).

The *block* method to be derived is either due to b right-hand sides or due to b (linearly independent) initial residuals. The algorithm is a combination of block and s-step iterative method. For $b = 1$, the methods are the s-step methods.

Let us assume that we are to solve a single linear system with coefficient matrix A and b right-hand sides. We denote the b right-hand sides by the *block vector* (which is a matrix of size $n \times b$):

$$F = [\bar{f}^{(1)}, \bar{f}^{(2)}, \ldots, \bar{f}^{(b)}] \tag{1}$$

Similarly, we denote the solution ith iterate by

$$X_i = [\bar{x}_i{}^{(1)}, \bar{x}_i{}^{(2)}, \ldots, \bar{x}_i{}^{(b)}] \tag{2}$$

and the residual ith iterate by

$$R_i = [\bar{r}_i^{(1)}, \bar{r}_i^{(2)}, \ldots, \bar{r}_i^{(b)}] \tag{3}$$

If we consider a single right-hand side then, we can write

$$R_i = F - AX_i = \bar{f}\bar{e}^{\mathrm{T}} - AX_i \tag{4}$$

where $\bar{e}^{\mathrm{T}} = [1, 1, \ldots, 1]$ is a unit vector of size b.

The following notation facilitates the description of the algorithm. We omit (in the notation) the iteration subscripts that appear in the algorithm. The matrices F (right-hand sides), X (solution vectors), R (residual vectors), as described above are of dimensions $n \times b$.The matrices P (direction vectors), AP (matrix times direction vectors), Q, S (are used in the orthonormalization of AP and similar transformations applied to P, respectively) are of dimension $n \times sb$. And U is an upper triangular matrix of dimension $sb \times sb$. The matrix of steplengths G is of dimension $sb \times b$. We denote by j_i the lowest index of preceding directions vectors in the orthogonalization step of the algorithm. Thus, for BOSGCR(s,b): $j_i = 1$ and for BOSOmin(s,b,k): $j_i = \max(1, i-k+1)$.

DOI: 10.1002/nla

This means that BOSGCR(s,b) stores all the preceding direction vectors, where as BOSOmin(s,b,k) stores only the k preceding direction vectors. Thus, BOSGCR(s,b) is a full orthogonalization and it is implemented as a restarted method whereas BOSOmin(s,b,k) is a truncated method. Following the literature [7, 8, 10] we formulate both BOSGCR(s,b) and BOSOmin(s,b,k) in a single algorithm. Note that if the b right-hand sides are distinct then $\bar{f}_i \neq \bar{f}_j$, for $i \neq j$; else for a single right-hand side $\bar{f}_i = \bar{f}$, for $i = 1, \ldots, b$. We denote by ε the error tolerance.

Algorithm 1 BOSGCR(s,b)/BOSOmin(s,b,k)
Initialization
Compute $R_1 = F - AX_1$ and set $Q_0 = S_0 = 0$.
Iterations
For $i = 1, 2, \ldots$ **until** $\|R_i\|_2 < \varepsilon$ **do**

1. Compute $P_i = [R_i, AR_i, \ldots, A^{s-1}R_i]$
   and $AP_i = [AR_i, A^2R_i, \ldots, A^sR_i]$
2. **If** $(i > 1)$ **then**
   $AP_i := AP_i - \sum_{j=j_i}^{i-1} Q_j (Q_j^{\mathrm{T}} AP_i)$ (orthogonalize AP_i against $Q_{j_i}, \ldots, Q_{i-1}$)
   $P_i := P_i - \sum_{j=j_i}^{i-1} S_j (Q_j^{\mathrm{T}} AP_i)$ (update P_i)
   EndIf
3. Compute QR-decomposition (via MGS) of AP_i and obtain S_i
   $AP_i = Q_i U_i$
   $P_i = S_i U_i$
4. Compute the steplengths/residuals/solutions
   $G_i = Q_i^{\mathrm{T}} R_i$
   $R_{i+1} = R_i - Q_i G_i$
   $X_{i+1} = X_i + S_i G_i$

EndFor

We note that (1) BOSGCR(s,b) is the same as BOSOMIN(s,b,0); (2) for $b = 1$ the BOSGCR/BOSOmin algorithms are identical to OSGCR/OSOmin.

We next present the storage and computational work for *a single iteration* (serial and parallel complexities) of Algorithm 1 in Tables I and II. Storage includes the matrix A and the matrices:

$$[\bar{x}_i^{(1)}, \bar{x}_i^{(2)}, \ldots, \bar{x}_i^{(b)}], \quad [\bar{r}_i^{(1)}, \bar{r}_i^{(2)}, \ldots, \bar{r}_i^{(b)}], \quad \{P_j\}_{j=j_i}^{j=(i-1)}, \quad \{AP_j\}_{j=j_i}^{j=(i-1)}$$

We only count vector operations on vectors of dimension (of the linear system) n. We use the following notation:

- **Ops** (Operations on vectors of dimension n, '+' or '*')
- **Dpr** (Dot products)
- **Mv** (Matrix times vectors)
- **Lc** (Linear combinations)
- **Stg** (Storage requirements for vectors besides the Matrix A).

For $i \leqslant k-1$, the BOSOMIN has same complexity and storage as BOSGCR.

We next obtain the vector operation for each iteration of Algorithm 1.

For Step 1: *sb* matrix times vectors are required.

DOI: 10.1002/nla

For Step 2: (i) *Dpr* ($Q_j^{\mathrm{T}} A P_i$) is a matrix–matrix multiply of matrix dimensions $(sb \times n) \times (n \times sb)$. So there are $(sb)^2$ Dpr or $2(sb)^2 n$ Ops. For BOSGCR, we have $2n \sum_{j=1}^{i-1} (sb)^2 = 2n(i-1)(sb)^2$ Ops. For BOSOMIN(s,b,k), after the first initial (k-1) iterations for which the complexity equals BOSGCR, we have $2n \sum_{j=i-k}^{i-1} (sb)^2 = 2nk(sb)^2$ Ops.

(ii) The *Lc* for AP_i have the same complexity as *Dpr*.

(iii) For P_i, we use the same Dpr but new Lc are computed, so the complexity is same as Lc of AP_i.

So for BOSGCR, the complexity for step 2 is $6n(i-1)(sb)^2$ Ops. For BOSOMIN with $k \leqslant i$, the complexity is $6nk(sb)^2$ Ops.

For Step 3: This is MGS. The $(sb) \times (sb)$ U matrix is upper triangular matrix consisting of entries that are dot products. To compute the nonzero entries of U, the complexity is $n(sb)^2/2$ Ops. We can compute Q by Lc: $Q = (AP)U^{-1}$ and this is a matrix–matrix multiply with dimensions $(n \times sb) \times (sb \times sb)$. Using the nonzero entries of U^{-1}, so the Lc cost is $n(sb)^2/2$ Ops. The computation of S is similar with Q, so the Lc cost is $n(sb)^2/2$ Ops. The total complexities for step 3 are $3n(sb)^2/2$ Ops.

For Step 4: (i) $\alpha_i = Q_i^{\mathrm{T}} R_i$ are Dpr and it is a matrix–matrix multiply with dimensions $(sb \times n) \times (n \times b)$. So total cost is $2sb^2 n$ Ops.

(ii) R_{i+1} is Lc and it is matrix–matrix multiply with dimension $(n \times sb) \times (sb \times b)$ so the cost is $2sb^2 n$.

(iii) X_{i+1} is similar to R_{i+1} and its cost is $2sb^2 n$ Ops.

So the total complexities for Step 4 is $6nsb^2$ Ops.

Counting (the serial) complexities in the steps of the Algorithm 1 yields the totals in Table I.

Parallel processing complexities are obtained from Table I for Ops divided by p (the number of processors used to run the algorithm in parallel). In addition, we must also consider the local and global (parallel processor system) communication operations. Table II contains the parallel complexity.

Remark

The parameters s and b affect the parallel performance of the methods. The s-step methods were shown to be well-suited to achieve high performance for parallel computers with architecture based on (cache or local memory) memory hierarchy [8]. The combination of block and s-step methods is expected to lead to high performance on the massively parallel (including multicore) computers due to the following properties: (a) They further enhance the efficient use of the memory hierarchy over the s-step methods; (b) They further reduce the number of global communication operations (due to dot products), because several dot products are performed simultaneously. Properties (a) and (b) are exploited in parallel programming by using BLAS-2 and BLAS-3

Table I. Number of Ops for the ith iteration of BOSGCR and BOSOmin.

Ops	OSGCR	OSOmin
Stg	$2n(i+1)sb$	$2n(k+1)sb$
Dpr	$(2i+1/2)n(sb)^2+2nsb^2$	$(2k+1/2)n(sb)^2+2nsb^2$
Mv	nsb	nsb
Lc	$(4i+1)n(sb)^2+4nsb^2$	$(4k+1)n(sb)^2+4nsb^2$

DOI: 10.1002/nla

Table II. Parallel complexities of BOSGCR and BOSOmin.

Ops	OSGCR	OSOmin
Stg	$2(i+1)sb*\frac{n}{p}$	$2(k+1)sb*\frac{n}{p}$
Dpr	$((2i+1/2)(sb)^2+2sb^2)*\frac{n}{p}$	$((2k+1/2)(sb)^2+2sb^2)*\frac{n}{p}$
Mv	$sb*\frac{n}{p}$	$sb*\frac{n}{p}$
Lc	$((4i+1)(sb)^2+4sb^2)*\frac{n}{p}$	$((4k+1)(sb)^2+4sb^2)*\frac{n}{p}$
Local Comm	$6sbn^{2/3}$	$6sbn^{2/3}$
Global Comm	$(sb)^2/2+(i+1/2)sb+1$	$(sb)^2/2+(k+1/2)sb+1$

[11] and by using a special implementation of the multiplication of a sparse matrix by many vectors [12].

We next discuss the main convergence properties of of (BOSGCR) BOSOmin(s,k). The convergence of (OSGCR) OSOmin(s,k) (with $1\leqslant s, k$) has been studied [7, 8]. In analogy to these results we give the following theorem which describes the convergence of (BOSGCR) BOSOmin(s,k).

Theorem 1
Let $1\leqslant s$ and assume that the degree of the minimal polynomial of all residual vectors $[\bar{r_i}^1, \bar{r_i}^2, \ldots, \bar{r_i}^b]$ is greater than s. Assume that for each iteration $i=1,\ldots$ (in Algorithm 1) (a *definiteness* condition) all diagonal elements of ${R_i}^{\mathrm{T}} A^j R_i$) are positive for some j (with $1\leqslant j\leqslant s$). Then (BOSGCR) BOSOmin(s,k) converge to the solution.

Proof
It is similar to the case $s=1$ which can be found in [8]. □

The condition ${R_i}^{\mathrm{T}} A^j R_i \neq 0$, for some j (with $1\leqslant j\leqslant s$), provides that one of the steplengths ${G_i}^j$ is not zero and thus progress toward the solution is made at the ith iteration of Algorithm 1.

3. USING BLOCK ITERATIVE METHODS TO SOLVE LINEAR SYSTEMS WITH A SINGLE RIGHT-HAND SIDE

In this section, we discuss how a linear system with a single right-hand side could be solved using block iterative methods. In parallel processing this approach has the advantage that it uses better parallel properties of the block methods to provide an overall faster solution of the linear system on parallel computers. This approach was studied for symmetric and positive-definite linear systems in [4]. We provide a new method to combine the block approximate solution vectors into a single approximate solution vector of the linear system. Although we implemented this method in conjunction with Algorithm 1, it can be used with any other block iterative methods.

More generally, in order to apply some block iterative method, in which b vectors are iterated, to solving a single right-hand side system

$$A\bar{x} = \bar{f} \tag{5}$$

DOI: 10.1002/nla

one must have b linearly independent initial residual vectors available. Then the output of the block method iterations produces b approximate solutions. It is also possible that on input we have only one initial guess and we want to get only one solution (approximation) at the end. If we use a preconditioned method, in which a preconditioning matrix depends upon some parameter and it is also easy to construct, then it would be possible to obtain b initial guesses, choosing b different parameter values. For an application of this approach, see the end of Section 5.

In order to apply BOSGCR/BOSOmin to the case of a single right-hand side we must simply add the following step to Algorithm 1: 5. Combine the b solutions X_{i+1} to obtain a single solution $\hat{\bar{x}}_{i+1}$.

We now derive a new solution averaging algorithm that extracts (optimally) a single approximate solution from b approximate solutions. Let

$$X_i = [\bar{x}_i^{(1)}, \bar{x}_i^{(2)}, \ldots, \bar{x}_i^{(b)}] \tag{6}$$

$$R_i = [\bar{r}_i^{(1)}, \bar{r}_i^{(2)}, \ldots, \bar{r}_i^{(b)}] \tag{7}$$

be the block approximate solutions and block residual vectors, respectively. For brevity, we shall omit the index i in the rest of this section.

Theorem 2
Let R be a matrix of full-rank. The optimal linear combination of the b solutions X to obtain a single approximate solution $\hat{\bar{x}}$ is given by

$$\hat{\bar{x}} = \frac{1}{\bar{e}^{\mathrm{T}}\bar{\xi}} X\bar{\xi} \tag{8}$$

where

$$\bar{\xi} = c(R^{\mathrm{T}}R)^{-1}\bar{e}, \quad c \neq 0 \tag{9}$$

for any nonzero scalar c and $\bar{e}^{\mathrm{T}} = [1, 1, \ldots, 1]$ a unit vector of size b.

Proof
Since the right-hand side is the same we can write

$$R = F - AX = \bar{f}\bar{e}^{\mathrm{T}} - AX \tag{10}$$

where $\bar{e}^{\mathrm{T}} = [1, 1, \ldots, 1]$ is a unit vector of size b. Multiplying the last equation by some b-dimensional vector $\bar{\xi}$ (to be determined) and scaling both sides by the scalar $\bar{e}^{\mathrm{T}}\bar{\xi}$ one can obtain

$$\frac{1}{\bar{e}^{\mathrm{T}}\bar{\xi}} R\bar{\xi} = \bar{f} - A\left(\frac{1}{\bar{e}^{\mathrm{T}}\bar{\xi}} X\bar{\xi}\right) \tag{11}$$

So, the residual $\hat{\bar{r}} = \bar{f} - A\hat{\bar{x}}$ of the approximate solution vector

$$\hat{\bar{x}} = \frac{1}{\bar{e}^{\mathrm{T}}\bar{\xi}} X\bar{\xi} \tag{12}$$

equals

$$\hat{\bar{r}} = \frac{1}{\bar{e}^{\mathrm{T}}\bar{\xi}} R\bar{\xi} \tag{13}$$

DOI: 10.1002/nla

Now we are ready to formulate an optimization problem for a solution averaging vector $\bar{\xi}$

$$\text{Find} \quad \bar{\xi} = \text{argmin}_{\bar{\xi}} \|\hat{\bar{r}}\|^2 \tag{14}$$

By direct computations we get

$$\left\| \frac{1}{\bar{e}^{\mathrm{T}}\bar{\xi}} R\bar{\xi} \right\|^2 = \frac{1}{(\bar{e}^{\mathrm{T}}\bar{\xi})^2} \bar{\xi}^{\mathrm{T}} R^{\mathrm{T}} R \bar{\xi} \tag{15}$$

Then by using the Cauchy–Schwarz inequality

$$(\bar{e}^{\mathrm{T}}\bar{\xi})^2 \leqslant (\bar{\xi}^{\mathrm{T}}(R^{\mathrm{T}}R)\bar{\xi})(\bar{e}^{\mathrm{T}}(R^{\mathrm{T}}R)^{-1}\bar{e}) \tag{16}$$

we obtain

$$\left\| \frac{1}{\bar{e}^{\mathrm{T}}\bar{\xi}} R\bar{\xi} \right\|^2 \geqslant \frac{\bar{\xi}^{\mathrm{T}} R^{\mathrm{T}} R \bar{\xi}}{(\bar{\xi}^{\mathrm{T}} R^{\mathrm{T}} R \bar{\xi})(\bar{e}^{\mathrm{T}}(R^{\mathrm{T}}R)^{-1}\bar{e})} \tag{17}$$

It is seen now that a minimal value of $\|\hat{\bar{r}}\|^2$ equals

$$\frac{1}{\bar{e}^{\mathrm{T}}(R^{\mathrm{T}}R)^{-1}\bar{e}} \tag{18}$$

which is attainable if and only if

$$\bar{\xi} = c(R^{\mathrm{T}}R)^{-1}\bar{e}, \quad c \neq 0 \tag{19}$$

for any nonzero scalar c. We choose $c = 1$ for simplicity. The equation $R^{\mathrm{T}}R\bar{\xi} = \bar{e}$ should be solved with a special care, since usually columns of matrix R may be nearly linearly dependent vectors and thus $R^{\mathrm{T}}R$ may have a large condition number. QR-decomposition with column pivoting is an suitable tool for factorizing the matrix $R^{\mathrm{T}}R$. MGS with column pivoting could be used to compute (the QR-decomposition) $R = Q_R W$, where $Q_R^{\mathrm{T}} Q_R = I_{b\times b}$ (the identity matrix) and W an upper triangular $b \times b$-matrix. Then we find $\bar{\xi}$ from the equation $W^{\mathrm{T}}W\bar{\xi} = \bar{e}$. □

Now using Theorem 2, we obtain a solution averaging algorithm as follows.

Algorithm 2 (solution averaging for block iterative methods)
For b initial solution guesses $X_1 = [\bar{x}^{(1)}, \ldots, \bar{x}^{(b)}]^{\mathrm{T}}$ compute
residuals $R = [\bar{r}^{(1)}, \ldots, \bar{r}^{(b)}]^{\mathrm{T}} = \bar{f}\bar{e}^{\mathrm{T}} - AX$
where: $\bar{e} = [1, \ldots, 1]^{\mathrm{T}}$
Iterate with the block iterative algorithm and compute R and X
Compute the QR-decomposition $R = Q_R W$
(by applying MGS with column pivoting)
Back-Solve two $b \times b$ linear systems $W^{\mathrm{T}}\bar{\eta} = \bar{e}$, $W\bar{\xi} = \bar{\eta}$,
Compute:
an average solution: $\hat{\bar{x}} = \frac{1}{\bar{e}^{\mathrm{T}}\bar{\xi}} X\bar{\xi}$
an average residual: $\hat{\bar{r}} = \frac{1}{\bar{e}^{\mathrm{T}}\bar{\xi}} R\bar{\xi}$
an estimate for the average residual:
$\|\hat{\bar{r}}\| = \frac{1}{\sqrt{(\bar{e}^{\mathrm{T}}\bar{\xi})}}$

DOI: 10.1002/nla

We note that the block iterative algorithm, in Algorithm 2, could be BOSGCR/BOSOmin (i.e. Algorithm 1) or any other block iterative algorithm.

4. A LARGE SPARSE TEST PROBLEM

We consider the discretization of an elliptic partial differential equation boundary value problem on a cubic region by the method of finite differences. This is a standard elliptic problem and the right-hand side function is constructed so that the analytic solution is known

$$\Delta u+\gamma(xu_x+yu_y+zu_z)=f(x,y,z) \tag{20}$$

If this problem is discretized using the centered difference scheme on a uniform $n_x\times n_y\times n_z$ grid (where $n_x=n_y=n_z=n_h$) with mesh size $h=1/(n_h+1)$, we obtain a linear system of equations $[A\bar{u}_h=\bar{f}_h]$ of size $n=n_h^3$. If we use natural ordering of the grid points in the x, y, z directions, then the matrix A is a block five diagonal matrix of the form

$$A=[\tilde{C}_k,\tilde{D}_k,\tilde{B}_k],\quad 1\leqslant k\leqslant n_h \tag{21}$$

where $\tilde{C}_k,\tilde{D}_k,\tilde{B}_k$ are matrices of size n_h^2; and $\tilde{C}_1=\tilde{B}_{n_h}=0$. The matrices $\tilde{C}_k,\tilde{B}_k$ are diagonal matrices and $\tilde{D}_k$ are five diagonal matrices. For fixed k, each row of submatrices $[\tilde{C}_k,\tilde{D}_k,\tilde{B}_k]$ contains coefficients of equations for grid points in 1 horizontal plane of the 3D space domain (i.e. fixed $z=k*h, x, y$ vary). Thus, the matrix has nonzeros only on seven diagonals. The matrix is *large*. For example, if $n_h=100$, then the dimension of A is $n=10^6$.

As an example one could consider solving $b=4$ linear systems using the block methods. For $m=1,2,\ldots,b$, we could choose the following exact (function) solutions:

$$u_h^{[2m-1]}=w(x,y,z)\sin(\pi mxyz),\quad m=1,2 \tag{22}$$

$$u_h^{[2m]}=w(x,y,z)\cos(\pi mxyz),\quad m=1,2 \tag{23}$$

where

$$w(x,y,z)=x(1-x)y(1-y)z(1-z)\exp(xyz) \tag{24}$$

and x,y,z are taken at grid points. The right-hand sides $\bar{f}_h^{(m)}$ could be obtained by matrix–vector products $A\bar{u}_h^{(m)}$, $m=1,2,3,4$.

The sparse nonsymmetric matrix A is a seven diagonal matrix due to the 7-point discretization operator. We use the following notations (shown in the subscripts of the matrix coefficients): o (for origin), e/w (for east/west), n/s (for north/south) and t/b (for top/bottom). Thus, $\bar{a_o}$ is the main diagonal, $\bar{a_b},\bar{a_s},\bar{a_w}$ are the (nonzero) subdiagonals and $\bar{a_t},\bar{a_n},\bar{a_e}$ are the are the (nonzero) superdiagonals of A. We note the block structure of matrix introduces zero entries on the super/sub-diagonals at a small number of index locations j (where: $1\leqslant j\leqslant n_h^3$). These locations correspond to the problem discretization at the domain boundaries. The main diagonal consists of nonzeros for this model problem, but the matrix may not be diagonally dominant. For simplicity of notation we assume that the index range of the diagonals is: $-n_h^2+1\leqslant j\leqslant n_h^3+n_h^2$. Then for $1\leqslant j\leqslant n_h^3$, the

DOI: 10.1002/nla

matrix times vector product is given by:

$$A\bar{y}(j) = \bar{a_s}(j)\bar{y}(j-n_h)+\bar{a_w}(j)\bar{y}(j-1)+\bar{a_o}(j)\bar{y}(j) \\ +\bar{a_e}(j)\bar{y}(j+1)+\bar{a_n}(j)\bar{y}(j+n_h)+\bar{a_b}(j)\bar{y}(j-n_h^2)+\bar{a_t}\bar{y}(j+n_h^2) \tag{25}$$

where

$$\bar{a_b}(j)=0, \quad -n_h^2+1\leqslant j\leqslant 0 \tag{26}$$

$$\bar{a_t}(j)=0, \quad n_h^3+1\leqslant j\leqslant n_h^3+n_h^2 \tag{27}$$

$$\bar{a_s}(j)=0, \quad -n_h+1\leqslant j\leqslant 0, \quad \bar{a_w}(j)=0, \quad j=0 \tag{28}$$

$$\bar{a_e}(j)=0, \quad j=n_h^3+1, \quad \bar{a_n}(j)=0, \quad n_h^3-n_h+1\leqslant j\leqslant n_h^3 \tag{29}$$

5. A PARALLEL PRECONDITIONING

We consider right preconditioning because it minimizes the norm of the residual error. Left preconditioning is similar (see [13, 14]), but it minimizes the norm of the preconditioned residuals. Let K^{-1} be the preconditioning matrix. The transformed system is

$$(AK^{-1})(K\bar{x})=\bar{f} \tag{30}$$

which is then solved by the iterative process. Either K is a close approximation to the A, i.e.

$$AK^{-1}\approx I \tag{31}$$

or AK^{-1} has clustered eigenvalues. The preconditioner K must be easily invertible, so that the linear system $K\bar{y}=\bar{z}$ is easy to solve. Following [8], in combining right preconditioning with BOSGCR/BOSOmin, we only need to modify *Step 1* of Algorithm 1, as follows:

1. Compute $P_i=[K^{-1}R_i, K^{-1}(AK^{-1})R_i, \ldots, K^{-1}(AK^{-1})^{s-1}R_i]$,
   $AP_i=[AK^{-1}R_i, (AK^{-1})^2R_i, \ldots, (AK^{-1})^sR_i]$.

In analogy to [15], we will consider a parallel incomplete decomposition preconditioner based on a domain overlapping. Let p be the number of processors. We split the domain into p overlapping subdomains. First, we split a computational domain along the z-axis into p subdomains, each of them contains $(n_h/p+2)$ xy-planes. So, each two neighbor subdomains share 2 overlapped xy-planes. We note that the entries of A corresponding to one xy-plane are in a single row block segment: $[\tilde{C}_k, \tilde{D}_k, \tilde{B}_k]$. Second, we consider a restriction of the original operator on each subdomain. The loss of connection between subdomains is partially compensated for by introducing overlapping planes.

Third, on each subdomain we construct an incomplete LU decomposition (ILU) preconditioner K as follows $K=K_\Omega(\theta)=\bar{L}\bar{D}\bar{U}$ where $\bar{L}$, $\bar{U}$ are unit lower/upper triangular and D are diagonal matrices, respectively, and $0\leqslant\theta\leqslant 1$. In our case we chose L and U with the same sparsity pattern as A. In this case, one can easily see that only the diagonal of the matrix is modified by the elimination.

DOI: 10.1002/nla

Applying the ILU(0) algorithm only the main diagonal entries of A change. Let $\tilde{\bar{a}}_o(j)$ be the new main diagonal entries. Then

$$\tilde{\bar{a}}_o(j) = a_o(j) - \bar{a_w}(j) * \frac{\bar{a_e}(j-1)}{\tilde{\bar{a}}_o(j-1)} - \bar{a_s}(j) * \frac{\bar{a_n}(j-n_h)}{\tilde{\bar{a}}_o(j-n_h)} - \bar{a_b}(j) * \frac{\bar{a_t}(j-{n_h}^2)}{\tilde{\bar{a}}_o(j-{n_h}^2)}$$

We show the sparsity patterns of these matrices by their action on a vector v in the matrix times vector product.

$$\bar{D}\bar{v}(j) = \tilde{\bar{a}}_o(j)v(j) \tag{32}$$

$$\bar{L}\bar{v}(j) = \bar{v}(j) + \bar{a_w}(j)\bar{v}(j-1) + \bar{a_s}(j)\bar{v}(j-n_h) + \bar{a_b}(j)\bar{v}(j-n_h^2) \tag{33}$$

$$\bar{U}\bar{v}(j) = \bar{v}(j) + \frac{1}{\tilde{\bar{a}}_o(j)} * (\bar{a_e}(j)\bar{v}(j+1) + \bar{a_n}(j)\bar{v}(j+n_h) + \bar{a_t}(j)\bar{v}(j+n_h^2)) \tag{34}$$

$$-n_h^2 + 1 \leqslant j \leqslant n_h^3 + n_h^2 \tag{35}$$

If we set $\bar{d}(j) = 1/\tilde{\bar{a}}_o(j)$, then the ILU (ILU($\theta$)) with $0 \leqslant \theta \leqslant 1$ computes $\bar{d}(j)$ such that:

$$\begin{aligned} 1/\bar{d}(j) = {} & \bar{a_o}(j) - \bar{a_w}(j)d(j-1)(\bar{a_e}(j-1) + \theta\bar{a_n}(j-1) + \theta\bar{a_t}(j-1)) \\ & - \bar{a_s}(j)\bar{d}(j-n_h)(\theta\bar{a_e}(j-n_h) + \bar{a_n}(j-n_h) + \theta\bar{a_t}(j-n_h)) \\ & - \bar{a_b}(j)\bar{d}(j-n_h^2)(\theta\bar{a_e}(j-n_h^2) + \theta\bar{a_n}(j-n_h^2) + \bar{a_t}(j-n_h^2)) \end{aligned} \tag{36}$$

with $j = -n_h^2 + 1, \ldots, n_h^3 + n_h^2$.

We use the parallel $ILU(\theta)$ preconditioner described above. We take $\theta = 1$ for the iteration process (which provides the fastest convergence), and use other values $0 \leqslant \theta < 1$ to generate initial multiple guesses. We compute b initial guesses as follows

$$\bar{x}_0^{(m)} = K^{-1}(\theta_m)\bar{f}, \quad \theta_m = \frac{b-m}{b}, \quad m = 1, 2, \ldots, b \tag{37}$$

6. IMPLEMENTATION AND TEST RESULTS

We ran our tests on a IBM SP at the University of California, San-Diego (SDSC), U.S.A. The IBM SP at SDSC is a Power-3 processor clustered SMP system. Each processor has peak performance of 1.5 GFLOPS. We used MPI and Fortran and BLAS 1–3 for our implementation [11].

We see that the algorithms consist of the following operations: (i) Sparse Matrix$\times$vector; (ii) Inner Products; (iii) Linear Combinations; (iv) MGS. Our implementation is outlined as follows: (1) Map A and vectors to a logical linear array of PEs (processors); (2) on each PE use BLAS-2 (e.g. SDOT, SGER for (ii)–(iv)); (3) use ALLREDUCE for global communication (e.g. for (ii)); (4) use local PE communication (e.g. for (i)). For our experiments $n_h = 64$ so the size of a matrix is $n = n_h^3 = 262,144$.

Convergence tests: The results are in Table III. We ran Algorithm 1 for $b = 1$ and $b = 4$ with the solution averaging method (Algorithm 2), for convergence. We report the number of iterations (Iter's). The Maximum Relative Residual Error in 2-norm (MRRES1 and MRRES4 for $b = 1$ and $b = 4$, respectively). The Maximum True Error in ∞-norm (MError1 and MError4 for $b = 1$ and

DOI: 10.1002/nla

Table III. BOSGCR(s,b,k) and BOSOMin(s,b,k), for $b=1$, $b=4$.

Method	(s,k)	Iter1	Iter4	MError1	MRRES1	MError4	MRRES4
BOSGCR	$(4,0)$	15	12	6.9E−13	4.88E−11	3.3E−12	3.20E−7
	$(8,0)$	8	5	3.0E−13	1.63E−11	4.3E−11	1.58E−7
BOSOmin	$(2,3)$	30	21	1.2E−12	6.08E−11	9.8E−12	8.04E−7
	$(4,1)$	15	10	1.0E−12	4.60E−11	1.2E−10	3.31E−7
	$(8,1)$	7	4	8.7E−13	6.49E−11	7.9E−10	4.62E−7

Table IV. $(s,b,k)=(8,4,0)$; execution times (seconds).

Method	p	Av	Prv	DotPr	LinComb	LocCom	GlobCom	$T_{\mathrm{p}}/T_{\mathrm{com}}$
BOSGCR	64	1.01	1.90	3.45	4.21	1.92	2.66	17.11/4.58
	8	2.41	2.87	4.23	23.90	1.79	1.33	49.28/3.13
	1	15.45	20.52	20.16	182.68	0.00	0.00	338.50/0.00

Table V. $(s,b,k)=(8,4,1)$; execution times (seconds).

Method	p	Av	Prv	DotPr	LinComb	LocCom	GlobCom	$T_{\mathrm{p}}/T_{\mathrm{com}}$
BOSOMIN	64	0.96	1.64	8.58	7.83	1.73	3.87	30.37/5.61
	8	2.78	3.25	24.62	54.81	2.03	2.46	123.78/4.49
	1	13.13	17.40	101.39	298.74	0.00	0.00	600.50/0.00

$b=4$, respectively). We use (MRRES=)$\mathrm{Max}_{m=1}^{b}(\|r_i^m\|_2/\|r_0^m\|_2)<\varepsilon$, as the stopping criterion. *Iter* is the total number of iterations. Our aim in this test was to obtain Maximum True Error of order 10^{-10}. We used a tolerance $\varepsilon=10^{-10}$ at first in all the runs. Then we observed that we can use a tolerance $\varepsilon=10^{-6}$ in the Algorithms 2, with $b=4$ (thus with fewer iterations) and obtain Maximum True Error less than 10^{-10}.

Performance tests: The results are in Tables IV and V. We ran the algorithms on $p=1,8,64$ PEs and measured the times for: Matrix times vector product (Av), Preconditioning step (Prv), Dotproducts (DotPr), Linear Combinations (LinComb), Local Communication (LocCom), Global Communication (GlobCom), Total Communication Time (T_{com}), Parallel Execution Time (T_{p}).

7. CONCLUSIONS

We present and study a block s-step Krylov iterative method for nonsymmetric linear systems. Such methods have enhanced parallel properties over the standard methods. We then consider applying block iterative methods to solve a linear system with a single right-hand side. We derive a new algorithm which combines the multiple solutions obtained by any block method into a single solution. We study the convergence and the parallel performance of the new methods. We have implemented our algorithms with a parallel preconditioning to approximate the solution of a large sparse linear system. Our results show that methods are convergent and they are scalable.

DOI: 10.1002/nla

ACKNOWLEDGEMENTS

The authors express their gratitude for the reviewers' comments which helped enhance the quality of presentation.

REFERENCES

1. Broyden CG. Block conjugate gradient methods. *Optimization Methods and Software* 1993; **2**(1):1–17.
2. Da Cunha RD, Becker D. Dynamic Block GMRES: an iterative method for block linear systems. *Advances in Computational Mathematics* 2007; **27**(4):423–448.
3. El Guennouni A, Jbilou K, Sadok H. A block version of BICGSTAB for linear systems with multiple right-hand sides. *Electronic Transactions on Numerical Analysis* 2003; **16**:129–142.
4. Hackbush W. A parallel variant of the conjugate gradient method. *Journal of Numerical Linear Algebra with Applications* 1992; **1**:133–147.
5. Simoncini V, Gallopoulos E. An iterative method for nonsymmetric systems with multiple right hand sides. *SIAM Journal on Scientific Computing* 1995; **16**(4):917–933.
6. Simoncini V, Szyld DB. Recent computational developments in Krylov Subspace Methods for linear systems. *Numerical Linear Algebra with Applications* 2007; **14**(1):1–59.
7. Chronopoulos AT. *S*-step iterative methods for (non)symmetric (in)definite linear systems. *SIAM Journal on Numerical Analysis* 1991; **28**(6):1776–1789.
8. Chronopoulos AT, Swanson CD. Parallel iterative *S*-step methods for unsymmetric linear systems. *Parallel Computing* 1996; **22**(5):623–641.
9. Chronopoulos AT, Kincaid D. On the Odir iterative method for nonsymmetric indefinite linear systems. *Numerical Linear Algebra with Applications* 2001; **8**(1–3):71–82.
10. Eisenstat SC, Elman HC, Schulz MH. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis* 1983; **20**(2):345–357.
11. Dongarra JJ, Duff IS, Sorensen DC, Van der Vorst HA. *Numerical Linear Algebra for High Performance Computers*. SIAM: Philadelphia, PA, U.S.A., 1998.
12. Baker AH, Dennis JM, Jessup ER. On improving linear solver performance: a block variant of GMRES. *SIAM Journal on Scientific Computing* 2006; **27**(4):1608–1626.
13. Axelsson O. *Iterative Solution Methods*. Cambridge University Press: Cambridge, 1996.
14. Meurant G. *Computer Solution of Large Linear Systems*. Elsevier: Amsterdam, 1999.
15. Radicati di Brozolo G, Robert Y. Parallel conjugate gradient-like algorithms for sparse nonsymmetric systems on a vector multiprocessor. *Parallel Computing* 1989; **11**(2):223–239.

DOI: 10.1002/nla