A Parallel Krylov-Type Method for Nonsymmetric Linear Systems

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

[1] Division of Computer Science, University of Texas at San Antonio,
6900 North Loop 1604 West, San Antonio, TX 78249, USA
atc@cs.utsa.edu
[2] Department of Computational Mathematics and Cybernetics, Moscow State University,
Vorobjevy Goru, 119899, Moscow, Russia

Abstract. Parallel Krylov (**S-step** and **block**) iterative methods for linear systems have been studied and implemented in the past. In this article we present a parallel Krylov method based on **block s-step** method for nonsymmetric linear systems. We derive two new averaging algorithm to combine several approximations to the solution of a single linear system using the block method with multiple initial guesses. We implement the new methods with ILU preconditioners on a parallel computer. We test the accuracy and present performance results.

1 Introduction

The s-Step Orthomin forms, at each iteration, s independent direction vectors using repeated matrix-vector products of the coefficient matrix with a single residual vector ([4]). Then the solution is advanced simultaneously using the s direction vectors. The **orthogonal s-Step GCR/Orthomin** was introduced in [4]. In the Orthogonal s-Step GCR/Orthomin (OSGCR(s)/OSOmin(k,s)) ([5]), a Modified Gram-Schmidt method (**MGS**) is used to orthonormalize the direction vectors within each block. We note that OSGCR(s) is the same as OSOmin(0,s).

An alternative approach to the s-step methods, in terms of parallel properties, is offered by the block methods. The block methods use a number of **linearly independent** *initial residual* vectors. This number is called the **blocksize** of the method. The residual vectors are used to compute simultaneously a block of direction vectors which are orthonormalized via MGS (much like in OSOmin). These direction vectors are then used to advance the solution and compute the new residual vector. Several authors have studied the block methods see for example [2] , [3] , [8] , [9] , [10] , [12] , [14] and the references therein.

A block OSGCR/OSOmin for the solution of nonsymmetric linear systems if obtained by turning the OSGCR/OSOmin into a block algorithm as in the references above. We want to use this method to solve a linear system with a single right handside on a parallel computer. One can use many initial guesses

B. Monien, V.K. Prasanna, S. Vajapeyam (Eds.): HiPC 2001, LNCS 2228, pp. 104–114, 2001.

and then combine the final solution approximations into one by an averaging scheme. Such a scheme was mentioned in [6], for the Conjugate Gradient method.

The convergence study of the s-step methods can be found in [4], [5]. Here we simply derived block algorithms in order to increase the parallelism of the methods. The block algorithms are 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(k,s) method we obtain methods mathematically equivalent to other widely used Krylov methods (e.g. OSOmin(k,1) is equivalent to Omin(k), OSGCR(s) is equivalent to Odir(s) and GMRES(s)) (see [4]). Thus, we only test the block OSOmin method for various k, s and we expect these comparisons to hold for other widely used methods.

The article follows the following structure. In section 2, the block s-step method is presented. In section 3, new solution averaging block methods for solving a linear system with a single right hand side are proposed. In section 4, a partial differential equation problem discretization which generates a large sparse matrix and a parallel preconditioner are described. In section 5, implementation and test results are presented. In section 6, we draw conclusions.

2 Block OSGCR/OSOmin

For integers i, k, s and $1 \leq i, k$, let $j_i = 1$ for OSGCR(s) $j_i = \max(1, i-k+1)$ for OSOMin(k,s). In OSGCR and OSOmin, each iteration generates a *block* of s direction vectors, which are denoted by the *matrix* $P_i = [p_i^1, \ldots, p_i^s]$. Firstly, AP_i is obtained from the column vectors $[Ar_i, A^2 r_i, \ldots, A^s r_i]$, by simultaneously $A^T A$ -orthogonalizing them against the preceding blocks of direction vectors and then orthogonalizing them amongst themselves. Then the direction vectors P_i are formed using the same linear combinations (as in AP_i) starting with the vectors $[r_i, Ar_i, \ldots, A^{s-1} r_i]$. The norm of the residual $\|r_{i+1}\|_2$ is minimized simultaneously in all s new direction vectors in order to obtain x_{i+1}. All orthogonalizations apply the Modified Gramm Schmidt algorithm (MGS) [5].

Let us assume that we are to solve a single linear system $Ax = f$ of dimension n with b righthand sides. We next present the block OSGCR(s)/OSOmin(k,s) (BOSGCR(b,s)/BOSOmin(b,k,s)). We note that BOSGCR(b,s) is the same as BOSOMIN(b,0,s). The following notation facilitates the description of the algorithm. We omit here the iteration subscripts which appear in the algorithm. Let b denote the block size, for example the number of initial solutions. The matrices F (right hand sides), X (solution vectors), R (residual vectors) are of dimensions $n \times b$. The matrices P (direction vectors), AP (matrix times direction vectors), Q, S (used in the orthonormalization of AP and similar transformations on P, respectively) are of dimension $n \times bs$. And U is upper triangular of dimension $bs \times bs$. The (parameter) matrix of steplengths α is of dimension $bs \times b$. Note that if (the b righthand sides are distinct) then $f_i \neq f_j$, for $i \neq j$; else (for a single righthand side $f_i = f$, for $i = 1, \ldots, b$.

Algorithm 1 BOSGCR(b,s)/BOSOmin(b,k,s)
Initialization
Set $F = [f_1, \ldots, f_b]$, Compute the initial residuals $R_1 = F - AX_1$ and set $Q_0 = S_0 = 0$.
Iterations
For $i = 1, 2, \ldots$ **until** convergence **do**
1. Compute $P_i = [R_i, AR_i, \ldots, A^{s-1}R_i]$
   and (set) $AP_i = [AR_i, A^2R_i, \ldots, A^sR_i]$
2. Orthogonalize AP_i against the matrices
   $Q_{i-1}, \ldots, Q_{j_i}$ and update P_i
   If $(i > 1)$ **then**
   $AP_i := AP_i - \sum_{j=j_i}^{i-1} Q_j(Q_j^T AP_i)$
   $P_i := P_i - \sum_{j=j_i}^{i-1} S_j(Q_j^T AP_i)$
   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
   $\alpha_i = Q_i^T R_i$
   $R_{i+1} = R_i - Q_i \alpha_i$
   $X_{i+1} = X_i + S_i \alpha_i$

EndFor
5. If(b distinct righthand sides) **then** exit **Else** Combine the b solutions X_{i+1} to obtain a single solutions $\bar{x}$

3 Solution Averaging Algorithms

In this section we discuss how a single system could be solved using Algorithm1. In order to apply some block method, in which b vectors are iterated, to solving of a single right hand side system

$$Ax = f \tag{1}$$

one must obtain b linearly independent initial guesses. Output of block iterations gives us b approximate solutions. It is reasonable that on input we have only one initial guess $\hat{x}_0$, and 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 easy to construct, it would be natural to obtain b initial guesses, choosing b different parameter values. For an application of this approach see the end of the next section.

We now derive a new Solution Averaging algorithm which extracts a single approximate solution with a optimal from b final iterates. Let

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

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

be final block solution and block residual vector respectively. Since the right hand side is the same we can write

$$R_i = fe^T - AX_i, \tag{4}$$

where $e^T = [1, 1, \ldots, 1]$ is a unit vector of size b. For brevity we shall omit the index i. Multiplying the last equation by some b -vector one can obtain

$$\frac{1}{e^T\xi}R\xi = f - A(\frac{1}{e^T\xi}X\xi) \tag{5}$$

So, a residual $\bar{r} = f - A\bar{x}$ of a vector

$$\bar{x} = \frac{1}{e^T\xi}X\xi \tag{6}$$

equals to

$$\bar{r} = \frac{1}{e^T\xi}R\xi. \tag{7}$$

Now we are ready to formulate an optimization problem for an averaging vector ξ.

$$\text{Find} \quad \xi = argmin_\xi ||\bar{r}||. \tag{8}$$

By direct computations we get

$$||\frac{1}{e^T\xi}R\xi||^2 = \frac{1}{(e^T\xi)^2}\xi^T R^T R\xi. \tag{9}$$

Then using the Cauchy-Schwarz inequality

$$(e^T\xi)^2 \leq (\xi^T(R^T R)\xi)(e^T(R^T R)^{-1}e) \tag{10}$$

we obtain

$$||\frac{1}{e^T\xi}R\xi||^2 \geq \frac{\xi^T R^T R\xi}{(\xi^T R^T R\xi)(e^T(R^T R)^{-1}e)} \tag{11}$$

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

$$min||\bar{r}|| = \frac{1}{\sqrt{e^T(R^T R)^{-1}e}} \tag{12}$$

which is attainable if and only if

$$\xi = c(R^T R)^{-1}e, \quad c \neq 0 \tag{13}$$

for any nonzero scalar c. We choose $c = 1$ for simplicity. An equation $R^T R\xi = e$ should be solved with a special care, since usually columns of matrix R are very close to linearly dependent vectors and $R^T R$ is badly conditioned. QR-decomposition with column pivoting is an appropriate tool for factorizing the matrix $R^T R$. We apply MGS with column pivoting to compute $R = Q_R W$, where $Q_R^T Q_R = I_{b\times b}$, W is an upper triangular $b \times b$-matrix, and then find ξ from the equation $W^T W\xi = e$.

Therefore we obtain a Solution Averaging algorithm as follows

Algorithm 2 Solution Averaging I
For b solution guesses $X = [x^{(1)}, \ldots, x^{(b)}]^T$ compute
residuals $R = [r^{(1)}, \ldots, r^{(b)}]^T = fe^T - AX$
where: $e = [1, \ldots, 1]^T$
Iterate with Algorithm 1 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 $W^T W\xi = e$,
Compute:
an average solution: $\bar{x} = \frac{1}{e^t\xi} X\xi$
an average residual: $\bar{r} = \frac{1}{e^t\xi} R\xi$
an estimate for the average residual:
$||\bar{r}|| = \frac{1}{\sqrt{(e^T\xi)}}$

Another algorithm for combining the residuals can be derived from an averaging scheme proposed for the CG method with multiple initial guesses [6]. There, it was mentioned (without providing an algorithm) that the average residual can be obtained by optimally chosing coefficients $\bar{\xi}$ of the expansion

$$\bar{r} = r^{(b)} + \sum_{m=1}^{b-1} (r^{(m)} - r^{(b)})\bar{\xi}[m] \tag{14}$$

This condition implies that the 2-norm of the average solution $\bar{r}_i$, must be minimized. We use QR Decomposition with column pivoting to solve this problem and we obtain and implement the following residual averaging algorithm. We omit the iteration subscripts for simplicity.

Algorithm 3 Solution Averaging II
For b solution guesses $X = [x^{(1)}, \ldots, x^{(b)}]^T$ compute
residuals $R = [r^{(1)}, \ldots, r^{(b)}]^T = fe^T - AX$
where: $e = [1, \ldots, 1]^T$
Iterate with Algorithm 1 and compute R and X
Compute QR-decomposition $\bar{R} = Q_{\bar{R}}\bar{W}$
where: $\bar{R} = [r^{(1)} - r^{(b)}, \ldots, r^{(b-1)} - r^{(b)}]^T$
(by applying MGS with column pivoting)
Back-Solve a $(b-1) \times (b-1)$ system
$\bar{W}\bar{\xi} = -Q_{\bar{R}}^T r^{(b)}$
Compute: an average residual/solution:
$\bar{r} = r^{(b)} + \sum_{m=1}^{b-1} (r^{(m)} - r^{(b)})\bar{\xi}[m]$
$\bar{x} = x^{(b)} + \sum_{m=1}^{b-1} (x^{(m)} - x^{(b)})\bar{\xi}[m]$

4 Test Problem and Preconditioning

We consider right preconditioning because it minimizes the norm of the residual error. Left preconditioning is similar (see [7]), but it minimizes the norm

of the preconditioned residuals. Let B^{-1} be the preconditioning matrix. The transformed system is

$$(AB^{-1})(Bx) = f, \tag{15}$$

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

$$AB^{-1} \approx I \tag{16}$$

or AB^{-1} has clustered eigenvalues. The preconditioner B must be easily invertible, so that the system $By = \phi$ is easy to solve. Following [5], in combining right preconditioning with the block OSGCR/OSOmin, we only need to modify *Step 1*, as follows

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

For a nonsymmetric linear system we consider the linear system arising from the discretization of the three-dimensional elliptic equation

$$\Delta u + \gamma(xu_x + yu_y + zu_z) = f(x, y, z) \tag{17}$$

on the unit cube with homogeneous Dirichlet boundary conditions. The degree of nonsymmetry of the problem is controlled by γ. We chose $\gamma = 50$. A discretization by centered finite differences on a uniform $n_h \times n_h \times n_h$ grid with mesh size $1/(n_h + 1)$ yields a linear system

$$Au_h = f_h^{(m)} \tag{18}$$

of the size $n = n_h^3$. As an example one could consider solving $b = 4$ linear systems. For $m = 1, 2, \ldots, b$, we could choose the following exact solutions

$$u_h^{[2m-1]} = w(x, y, z)\sin(\pi mxyz), \tag{19}$$

$$u_h^{[2m]} = w(x, y, z)\cos(\pi mxyz), \quad m = 1, 2 \tag{20}$$

where

$$w(x, y, z) = x(1 - x)y(1 - y)z(1 - z)\exp(xyz), \tag{21}$$

and x, y, z are taken at grid points. Righthand sides $f_h^{(m)}$ could be obtained by matrix-vector product $Au_h^{(m)}$, $\quad m = 1, 2, 3, 4$. In the tests (in the next section) we only consider the first linear system with $b = 4$ initial guesses.

We use 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. Firstly, 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 neighbour subdomains share 2 overlapped xy-planes. Secondly, 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. Thus we obtain the following restricted matrix on each subdomain Ω

$$\begin{gathered} A_\Omega y(j) = a_s(j)y(j-n_h) + a_w(j)y(j-1) + a_0(j)y(j) + \\ +a_e(j)y(j+1) + a_n(j)y(j+n_h) + a_b(j)y(j-n_h^2) + a_t y(j+n_h^2), \\ -n_h^2+1 \le j \le n_h^3/p + n_h^2, \end{gathered} \tag{22}$$

$$a_b(j) = 0, \quad -n_h^2+1 \le j \le 0; \tag{23}$$

$$a_t(j) = 0, \quad n_h^3/p+1 \le j \le n_h^3/p + n_h^2. \tag{24}$$

Thirdly, on each subdomain we construct an Incomplete LU decomposition (ILU) preconditioner B as follows $B = B_\Omega(\theta) = LDU$ where L/U - lower/upper triangular and D - diagonal matrices respectively, and $0 \le \theta \le 1$. 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. We show the sparsity patterns of these matrices by their action on a vector v in the matrix times vector product.

$$Lv(j) = \frac{1}{d(j)}v(j) + a_w(j)v(j-1) + a_s(j)v(j-n_h) + a_b(j)v(j-n_h^2), \tag{25}$$

$$Dv(j) = d(j)v(j), \tag{26}$$

$$Uv(j) = \frac{1}{d(j)}v(j) + a_e(j)v(j+1) + a_n(j)v(j+n_h) + a_b(j)v(j+n_h^2), \tag{27}$$

$$-n_h^2+1 \le j \le n_h^3/p + n_h^2, \tag{28}$$

where

$$\begin{gathered} 1/d(j) = a_0(j) - a_w(j)d(j-1)(a_e(j-1) + \theta a_n(j-1) + \theta a_t(j-1)) - \\ -a_s(j)d(j-n_h)(\theta a_e(j-n_h) + a_n(j-n_h) + \theta a_t(j-n_h)) - \\ -a_b(j)d(j-n_h^2)(\theta a_e(j-n_h^2) + \theta a_n(j-n_h^2) + a_t(j-n_h^2)) \end{gathered} \tag{29}$$

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

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 $0 \le \theta < 1$ to generate initial multiple guesses. We compute b initial guesses as follows

$$x_0^{(m)} = B^{-1}(\theta_m)f, \qquad \theta_m = \frac{b-m}{b}, m = 1, 2, \ldots, b \tag{30}$$

5 Implementation and Test Results

We ran our tests on the CRAY T3E at the University of Texas Austin. We used MPI and Fortran 90 and BLAS 1-3 for our implementation.

Table 1. BOSGCR/BOSOMin , $b = 1$, $b = 4$.

Block Method	(k, s)	$Iter's$	$MError1$	$MRRES1$	$MError4$	$MRRES4$
Omin	$(4, 1)$	61	$1.5E-12$	$6.55E-11$	$3.5E-10$	$5.12E-8$
	$(8, 1)$	55	$1.9E-12$	$9.76E-11$	$8.1E-10$	$1.27E-7$
	$(16, 1)$	55	$1.0E-12$	$5.99E-11$	$4.5E-10$	$7.72E-8$
OSGCR	$(0, 4)$	15	$6.9E-13$	$4.88E-11$	$3.3E-12$	$5.84E-10$
	$(0, 8)$	8	$3.0E-13$	$1.63E-11$	$4.3E-11$	$1.10E-8$
OSOmin	$(3, 2)$	30	$1.2E-12$	$6.08E-11$	$9.8E-12$	$2.13E-9$
	$(1, 4)$	15	$1.0E-12$	$4.60E-11$	$1.2E-10$	$1.98E-8$
	$(1, 8)$	7	$8.7E-13$	$6.49E-11$	$7.9E-10$	$1.77E-7$

We see that the algorithms consist of the following operations: (i) Sparse Matrix x 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 $n = n_h^3 = 262,144$. The smallest number of PEs that have enough memory to run the problem is $p = 4$.

Convergence tests: We ran Algorithm 1 for $b = 1$ and $b = 4$ with the Solution Averaging methods I - II, on $p = 4$ PEs, 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 $b = 4$ respectively). We use (MRRES =) $Max_{m=1}^{b}(||r_i^m||_2/||r_0^m||_2) < \epsilon$, as stopping criterion. The results are in Table 1. The methods are: the standard Omin(k) or Orthomin(k) for $s = 1$; the OSGCR for $k = 0$; the OSOMIN(k,s) for $k = 1$, $s > 0$. *Iter* is the total number of iterations. Algorithms 2 and 3 gave the same results for this test problem. Our aim in this test was to obtain Maximum-True-Error of order 10^{-10}. We used $\epsilon = 10^{-10}$ at first in all the runs. Then we observed that we can use $\epsilon = 10^{-6}$ in the Algorithms 2, 3 with $b = 4$ (thus with fewer iterations) and obtain Maximum-True-Error less than 10^{-10}.

Table 2. BOSGCR/BOSOMin, $b = 4$, $(k, s) = (16, 1)$; Execution times (secs)

p	Av	Prv	DotPr	LinComb	LocCom	GlobCom	T_p/T_{com}
64	$4.91E-2$	$4.77E-2$	$2.1E-3$	$5.2E-3$	$1.46E-2$	$6.7E-3$	$.1661/2.13E-2$
32	$9.85E-2$	$6.22E-2$	$3.5E-3$	$1.00E-2$	$1.46E-2$	$7.0E-3$	$.2862/2.16E-2$
16	.1934	$9.23E-2$	$5.9E-3$	$1.93E-2$	$1.47E-2$	$7.1E-3$	$.5218/2.18E-2$
8	.3815	.1522	$1.07E-2$	$3.81E-2$	$1.71E-2$	$5.9E-3$	$.9932/2.27E-2$
4	.7572	.2667	$2.05E-2$	$7.55E-2$	$1.53E-2$	$4.9E-3$	$1.9280/2.02E-2$

Table 3. BOSGCR/BOSOMin, $b = 4$, $(k, s) = (0, 8)$; Execution times (secs)

p	Av	Prv	DotPr	LinComb	LocCom	GlobCom	T_p/T_{com}
64	.3581	.3330	$1.22E-2$	$3.62E-2$	$9.79E-2$	$3.54E-2$	1.107/.1334
32	.7144	.4300	$2.11E-2$	$7.07E-2$	$9.68E-2$	$2.60E-2$	1.950/.1228
16	1.400	.6282	$3.55E-2$	.1365	.1005	$2.34E-2$	3.576/.1239
8	2.718	1.002	$8.04E-2$	.3216	.1041	$2.22E-2$	6.715/.1263
4	5.211	1.707	.1934	.7870	$9.66E-2$	$1.80E-2$	12.66/.1146

Performance tests: We ran the algorithms on $p = 4$, 8, 16, 32, 64 processors and measured the different computation/communication parts for each iteration. We report in Table 2-4 the times for: Matrix times vector product (Av), Preconditioning step (Prv), Dotproducts (DotPr), Linear Combinations (LinComb), Local Communication (LocCom), Global Communication (GlobCom), Parallel Execution Time (T_p), Total Communication Time (T_{com}). We plot the speedup in Figure 1, which shows that the algorithms are scalable.

Table 4. BOSGCR/BOSOMin , $b = 4$, $(k, s) = (1, 8)$; Execution times (secs)

p	Av	Prv	DotPr	LinComb	LocCom	GlobCom	T_p/T_{com}
64	.3593	.3306	$2.40E-2$	$8.73E-2$	$9.56E-2$	$3.71E-2$	1.683/.1328
32	.7193	.4295	$4.75E-2$	.1759	$9.66E-2$	$3.97E-2$	3.142/.1363
16	1.365	.6039	.1101	.4392	$9.72E-2$	$3.32E-2$	5.616/.1305
8	.2.589	.9397	.2601	1.055	$9.91E-2$	$3.15E-2$	10.16/.1306
4	5.277	1.713	.4605	1.912	$9.92E-2$	$2.84E-2$	20.91/.1276

6 Conclusions

We have derived two new parallel algorithms for solving linear systems with a single right hand side. We have implemented the algorithms with ILU preconditioning to approximate the solution of a large sparse system. We have studied the convergence and the parallel performance of the algorithms. Our results show that that methods are convergent and they are scalable.

Acknowledgement. Some reviewers' comments helped enhance the quality of presentation. This research was supported, in part, by research grants from (1) NASA NAG 2-1383 (1999-2000), (2) State of Texas Higher Education Coordinating Board through the Texas Advanced Research/Advanced Technology Program ATP 003658-0442-1999 (3) This research was also supported in part by NSF cooperative agreement ACI-9619020 through computing resources provided by the National Partnership for Advanced Computational Infrastructure at the University of California San Diego.

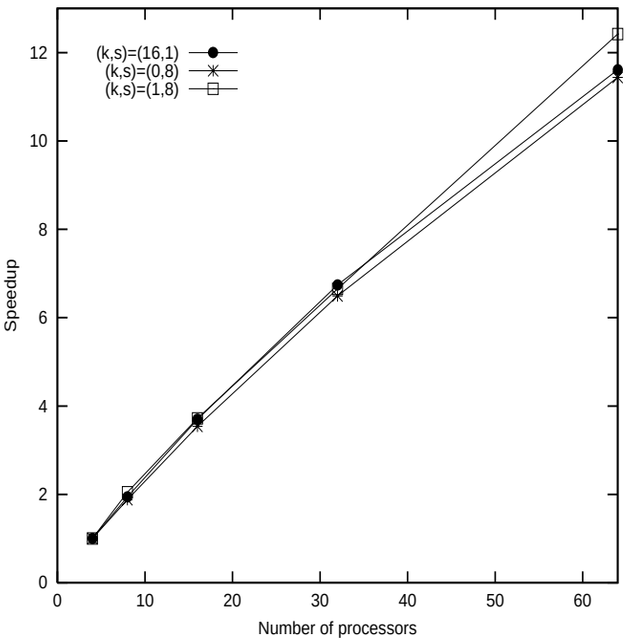

Fig. 1. BOSGCR/BOSOMin, $b = 4$; Speedup

References

[1] Axelsson, O.: Iterative Solution Methods. Cambridge University Press, (1996)
[2] Broyden, C.G.: Block Conjugate Gradient Methods. Optimization methods and Software, **2**(1993) 1-17
[3] Calvetti, D., Reichel L.: Application of a block modified Chebyshev algorithm to the iterative solution of symmetric linear systems with multiple right-hand side vectors. Numer. Math. **68**(1994), 3-16
[4] Chronopoulos, A.T.: S-step Iterative Methods for (Non)symmetric (In)definite Linear Systems. SIAM J. on Num. Analysis, No. 6, **28**(1991) 1776-1789.
[5] Chronopoulos, A.T., Swanson, C.D.: Parallel Iterative S-step Methods for Unsymmetric Linear Systems. Parallel Computing. Volume 22/5, (1996) 623-641
[6] Hackbush, W.: A parallel variant of the conjugate gradient method. Applied Mathematical Sciences, 95. Springer-Verlag, New-York, (1994) 111-119
[7] Meurant, G.: Computer Solution of Large Linear Systems. Elsevier (1999)
[8] Nikishin, A.A., Yeremin, A.Y.: Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers, I: General iterative scheme. SIAM J. Matrix Anal. Appl. **16**(1995) 1135-1153
[9] Papadrakakis M., Smerou, S.: A new implementation of the Lanczos method in linear problems. Internat. J. Numer. Methods Engrg., **29**(1990) 141-159
[10] Parlett, B.N.: A new look at the Lanczos and the block-Lanczos methods. SIAM J. Numer. Anal., **17**(1980) 687-706

[11] Radicati di Brozolo, G., Robert, Y.: Parallel conjugate gradient-like algorithms for sparse nonsymmetric systems on a vector multiprocessor. Parallel Computing, **11**(1989) 223-239

[12] Sadkane, M., Vital, B.: Davidson's Method for Linear Systems of Equations. Implementation of a Block Algorithm on a Multi-processor. Tech. Rept. TR/PA/91/60, CERFACS, Toulouse, (1991)

[13] Simon, H., Yeremin, A.: New Approach to Construction of Efficient Iterative Schemes for Massively Parallel Applications: Variable Block CG and BiCG Methods and Variable Block Arnoldi Procedure. Proc. of the 6th SIAM Conference on Parallel Proc. for Scientific Computing, ed. by R. Sincovec et al., SIAM, Philadelphia, (1993) 57-60

[14] Simoncini, V., Gallopoulos, E.: An iterative method for nonsymmetric systems with multiple right hand sides. SIAM J. Sci. Stat. Comput., **16**(1995) 917-933