Research Report

KSTS/RR-12/002
October 16, 2012

An efficient implementation of the block Gram-Schmidt method

by

Yoichi Matsuo
Takashi Nodera

Yoichi Matsuo
Keio University

Takashi Nodera
Keio University

Department of Mathematics
Faculty of Science and Technology
Keio University

KSTS/RR-12/002
October 16, 2012

An Efficient Implementation of the Block Gram-Schmidt Method

Yoichi MATSUO* Takashi NODERA†

Abstract

The Block Gram-Schmidt method (BGS) computes the QR factorization rapidly by partitioning matrix X into columns, which are then orthogonalized into blocks with the dimension of block-size m. However, if the wrong block-size is chosen, the computation time increases significantly. Moreover since the optimal block-size m is not consistent when employing BGS, block-size m must be redetermined during each calculation.

Recent developments in supercomputers have made parallelization mandatory when solving large scale problems. One of the numerous parallelization procedures is the Column-Wise Distribution method. In this paper, we have chosen to explore this particular method of parallelization, and have also endeavored to determine optimal block-size in this scenario.

Our new scheme for determining optimal block size m achieves this through focusing on the relationship between computation time and complexity. Numerical experiments were implemented to evaluate the effectiveness of our proposed algorithms, and the results suggest that our scheme is effective.

key words. block Gram-Schmidt algorithm, optimal block size, parallel computing
AMS(MOS) subject classifications. 65F10, 65M12

1 Introduction

In the fields of science and technology, it is often necessary to find approximate solutions for large scale problems, which are derived from partial difference equations by using finite difference or finite element discretization. The QR factorization is arguably one of the most important processes in a linear algebraic computation, and there are numerous studies on this subject matter [1, 2, 6, 7, 9, 10, 11]. It is frequently used to solve least squares problems and eigenvalue problems which arise in signal processing, structural mechanics or magnetohydrodynamics [9].

There are several algorithms for computing the QR decomposition. One of these algorithms is the Classical Gram-Schmidt orthogonalization (CGS). The CGS generates

*School of Fundamental Science and Technology, Graduate School of Science and Technology, Keio University, 3-14-1 Hiyoshi, Kohoku, Yokohama, 223-8522, JAPAN, Mail address: matsuo@math.keio.ac.jp

†Department of Mathematics, Faculty of Science and Technology, Keio University, 3-14-1 Hiyoshi, Kohoku, Yokohama, 223-8522, JAPAN, Mail address: nodera@math.keio.ac.jp

the orthogonal vector y from the given vector X. Using the orthogonal matrix Q, the CGS algorithm is as follows:

$$y = \left(I - QQ^T\right) x = x - QQ^T x \equiv x - Qr. \tag{1.1}$$

The CGS is sensitive to the round-off error, and the reorthogonalization process is employed to address this issue. A lot of iterative methods for solving eigenvalue problems and large scale linear equations use this process. For example, the GMRES which is a Krylov subspace method, relies on the reorthogonalization process to maintain accuracy.

Alongside accuracy, speed-up is important. The Block Gram-Schmidt orthogonalization (BGS) enables us to compute the QR factorization quickly. A study by Stewart [8] and Matsuo *et al.* [5] illustrates how the computation time of the QR factorization can be shortened by employing the BGS. Stewart [8] also proposed a new method where a vector was replaced with one with components with random values, when X is the ill-conditioned matrix. However, if matrix X has a significantly large condition number, an orthogonal vector with reorthogonalization is not always generated, and it would break down when $y = 0$. In such a case, it becomes necessary to replace this with a new random vector, which is:

$$x = \frac{x_{\text{new}}}{\|x_{\text{new}}\|_2} \times \|x\|_2, \tag{1.2}$$

and this would prevent the breakdown from occurring. This makes it possible to consistently generate an orthgonal matrix, but compromises the numerical precision of the factorization of X.

Recent developments in supercomputers have made it mandatory to use parallelization when employing various methods for solving large scale problems. There is a small body of literature regarding block versions of the Gram-Schmidt algorithms, mostly in the area of parallel computing [3, 10]. The algorithms in these studies block matrix Q, so that it can be distributed over a system of processors.

In this paper, we have focused on the efficient implementation and speed-up of the Block Gram-Schmidt method. Section 2 is an outline of the BGS and an explanation is provided as to why this method was best suited for our schemes. Section 3, is a proposal of the new schemes for determining optimal block-size automatically. In section 4, the BGS was parallelized and adapted to these schemes. Section 5, analyzes data from our numerical experiments and this is followed by a conclusion that our methods are effective.

2 The Block Gram-Schmidt Orthogonalization

2.1 Algorithm

The Block Gram-Schmidt algorithm is a natural generalization of the CGS. In this algorithm, the matrix X which is orthogonalized is partitioned by columns into blocks. Each block is sequentially orthogonalized and included into the block of matrix Q, where X has p columns and the block-size is m. The BGS step is as follows:

$$R_{12} = Q^T X_{\text{block}}, \tag{2.1}$$

$$\widehat{Y} = X_{\text{block}} - QR. \tag{2.2}$$

When this procedure is employed, the use of the orthonormal matrix Q is reduced compared to the CGS. However, in general, each column of matrix $\widehat{Y}$ is not mutually orthogonal to those of matrix Q in equations (2.1) and (2.2).

Let $Y_{1:k}$ be the submatrix of column Y. Let the submatrix $Y_{1:k}$ be composed of rows 1 to k of column Y. Let $\widehat{y}_{k+1}$, and y_{k+1} be the $k+1$th column vectors of $\widehat{Y}$ and Y. Consider the following Gram-Schmidt step:

$$\begin{aligned} r &= Y_{1:k}^T\, \widehat{y}_{k+1}, \\ y_{k+1} &= \widehat{y}_{k+1} - Y_{k:1} r. \end{aligned}$$

Iterating the above step, we have the following equation:

$$\widehat{Y} = Y R_{22}. \tag{2.3}$$

From equations (2.2) and (2.3), matrix X_{block} satisfies the following recurrence relation:

$$X_{\text{block}} = Q R_{12} + Y R_{22}. \tag{2.4}$$

In order to implement a reorthogonalization process successfully, the first cycle of the orthogonalization must produce a matrix $\widehat{Y}$ satisfying the following equation:

$$X_{\text{block}} = Q R_{12} + \widehat{Y} R_{22},$$

where R_{22} is an upper triangular matrix. The second cycle must produce a matrix Y with orthonormal columns satisfying:

$$Y = Q S_{12} + Z S_{22}, \tag{2.5}$$

where S_{22} is a lower triangular matrix. Combining equations (2.4) and (2.5), the following equation is obtained:

$$X_{\text{block}} = Q\left(R_{12} + S_{12} R_{12}\right) + Z S_{22} R_{22}. \tag{2.6}$$

Based on the above equation (2.6), the BGS with reorthogonalization is denoted in Algorithm 1.

Since the matrix products of the BGS steps are equations (2.1) and (2.2), we can use level 3 BLAS (basic Linear Algebra Subprogram) because the BGS will have a shorter computation time. However in general, if the value of m is enlarged and the computation time is shortened, the computation time increases after a certain threshold. This is why it is necessary to pay attention to optimal block-size.

2.2 The Complexity of the Block Gram-Schmidt Algorithm

Let X be an $n \times n$ matrix, let the orthogonal matrix Q be $n \times h$, let m be the block-size (to simplify, divide n by m), and let one multiplication be one unit. Assume that every column will require reorthogonalization. Now, consider the computational complexity for one iteration. Firstly, equations (2.1) and (2.2) require the following multiplications:

$$(2nmh) \times 2 = 4nmh. \tag{2.7}$$

Algorithm 1: The Block Gram-Schmidt Method

```
**Data**: $X \in \mathbb{R}^{n\times n}, X_{\text{block}} \in \mathbb{R}^{n\times m}$
**Result**: $Q, R$
**begin**
    $K \longleftarrow P/m$;
    **for** $k = 0 : K-1$ **do**
        $X_{\text{block}} \longleftarrow X[:; km : (k+1)m]$;
        $R_{12} \longleftarrow Q^T X_{\text{block}}$;
        $\widehat{Y} \longleftarrow X_{\text{block}} - QR_{12}$;
        **for** $l = 1 : m$ **do**
            $r \longleftarrow Y^T_{1:l-1}\widehat{y}_l$;
            $y \longleftarrow Y_{1:(l-1)}r$;
            $R_{22}[1 : l-1; l] \longleftarrow r$;
            $R_{22}[k : k] \longleftarrow \|y\|$;
        **end**
        **if** $(\widehat{y}_k < 1/2\|x_k\|)$ **then**
            Reorthogonalization;
        **end**
        $S_{12} \longleftarrow Q^T\widehat{Y}$;
        $\widehat{Y} \longleftarrow \widehat{Y} - QS_{12}$;
        **for** $l = 1 : m$ **do**
            $r \longleftarrow Y^T_{1:(l-1)}\widehat{y}_l$;
            $y \longleftarrow Y_{1:(l-1)}r$;
            $S_{22}[1 : l-1; l] \longleftarrow r$;
            $S_{22}[k : k] \longleftarrow \|y\|$;
        **end**
        $Q[:; km : (k+1)m] \longleftarrow Y$;
        $R_{12} \longleftarrow S_{12}R_{22} + R_{12}$;
        $R_{22} \longleftarrow S_{22}R_{22}$;
        $R \longleftarrow R_{12} + R_{22}$;
    **end**
**end**
```

Secondly, the number of multiplications of equation (2.3) requires:

$$\sum_{k=1}^{m-1} 2nk \times 2 = 2nm\,(m-1)\,. \tag{2.8}$$

Finally, to generate R from R_{12}, R_{22}, S_{12}, and S_{22}, the following is required:

$$m^2h + m^3. \tag{2.9}$$

From equations (2.7), (2.8) and (2.9), the computational complexity of one iteration of the BGS step is:

$$4nmh + 2nm\,(m-1) + (h+m)\,m^2. \tag{2.10}$$

Therefore, the entire complexity of the BGS S_{BGS} results in the following computational complexity:

$$\begin{aligned} S_{\text{BGS}} &= \sum_{k=1}^{n/m-1} (4nm^2k + 2nm\,(m-1) + (k+m)\,m^2), \\ &= -m^3 + \frac{1}{2}nm^2 + \frac{1}{2}n^2m + 2n^2\,(n-1)\,. \end{aligned} \tag{2.11}$$

Here, under the same conditions, the CGS requires:

$$S_{\text{GS}} = 2n^2\,(n-1)\,, \tag{2.12}$$

and the following is obtained:

$$S_{\text{BGS}} - S_{\text{GS}} = -m^3 + \frac{1}{2}nm^2 + \frac{1}{2}n^2m. \tag{2.13}$$

The computational complexity increases when the BGS with reorthogonalization is applied. However, in practice, we can compute the factorization of matrix X more quickly by employing the BGS.

2.3 Computation Time

The computation time of the BGS increases constantly. We can make sure of this by examining the computation complexity of the BGS. Let us consider function $f(h)$, which is obtained from (2.10):

$$f(h) = 4nmh + 2nm\,(m-1) + (h+m)\,m^2. \tag{2.14}$$

$f(h)$ is a linear function and it is possible to predict that the amounts of increase in the complexity of the BGS will be constant. Figure 1 illustrates the computation time of the QR decomposition when BGS $m = 50$. The test matrix BCSSTK15 is from the Matrix Market [4], which is a nonsymmetric real matrix of 3948×3948 which arises from eigenvalue problems. This figure also shows that the graph is very similar to those with a linear function.

The BGS is characterized by another property. Where the BGS is concerned, the optimal block-size is not consistent. There is no unique block-size for all calculations. Figure 2 illustrates the relationship between computation time and block-size. The test matrix is the same as the one in Figure 1. It shows that the computation time of the BGS changes according to block-size.

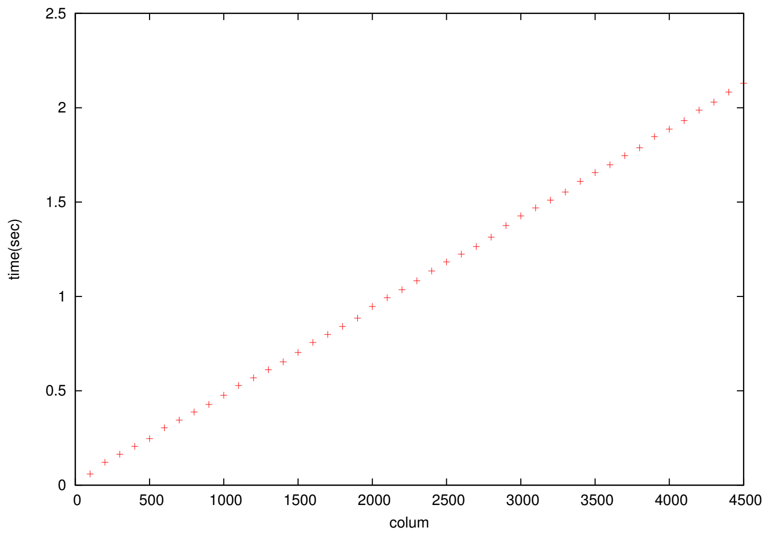

Figure 1: Computation Time of the BGS Steps

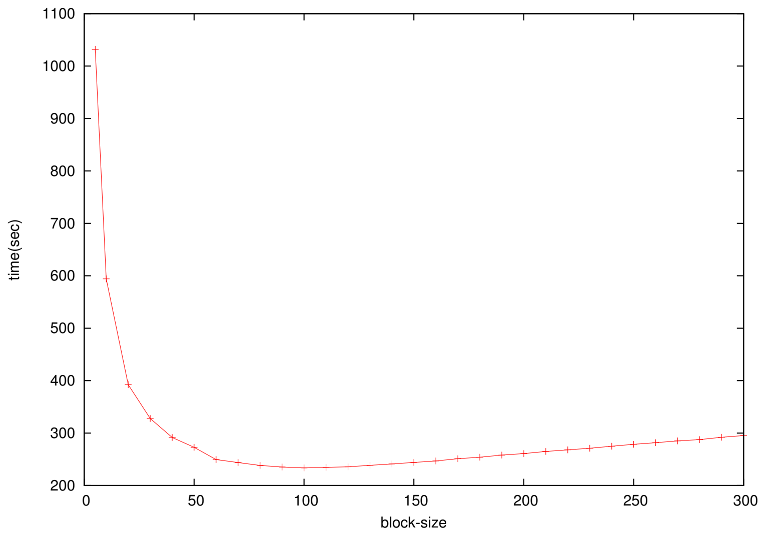

Figure 2: Relationship Between Block-size m and Computation Time

3 Optimal Block-Size

As has been mentioned above, the computation time of the BGS changes significantly depending on block-size. Conventionally, optimal block-size has been determined through trial and error. This is not an issue with a small-scale matrix, but when the matrix is large, the computation time for one iteration is lengthened so much that it becomes a problem. To address this issue, we have proposed two methods that allow us to determine optimal block-size automatically through exploiting the relationship between computational complexity and computation time [5].

In a study by Matsuo et al [5], scheme A, which is one of the two methods, compares the various computation times c_i for one iteration of the computational complexity of the multiplications:

$$c_i = \frac{(4nm_ih + 2nm_i(m_i - 1) + (h + m_i)m_i^2)}{t_i}. \tag{3.1}$$

Hence, block-size m as shown below is the optimal block-size:

$$m = \{m_i \mid \min_i c_i\}. \tag{3.2}$$

Scheme B determines block-size by approximating the cubic function with respect to m. As can be seen, the computation time is proportional to the computational complexity of the multiplications. From this, it can be estimated that the total computation T_i time is:

$$T_i = S_{\text{BGS}m_i} \times c_i. \tag{3.3}$$

In this scenario, we only need three different computation times from the BGS. If we take the h data along with the small value of m, we reduce the computation time and minimize the computational complexity of the multiplications.

3.1 The New Scheme

As has been discussd in paper [5], we have proposed two schemes which can estimate optimal block-size based on computation time and complexity. However, since the increase of computation time for each step cannot be calculated, sometimes these schemes will find a bad block-size which is either very large or small. In this study, we have improved this method by implementing two steps of the BGS to compute sample points. Based on computation time t_i and the complexity of the two steps, the total computation time T_i for each m_i is estimated. The critical point of this method can be summarized as follows: Let m_i be $m_i = 2^i,\ i = 1, \cdots,\ 4$ and $X_{\text{block}} \in \mathbb{R}^{n\times m}$ and $Q \in \mathbb{R}^{n\times h}$ be the chosen matrix. First the BGS with m_i for two steps is executed to extract some sample points. The practical relationship between t_i and m can almost be classified as a linear function. Next, the increase in computation time against one column is approximated:

$$a = (t_{i0} - t_{i1})/m_i. \tag{3.4}$$

In this manner, the computation time of the the BGS is approximated:

$$T_i := \frac{1}{2}n^2a + t_{i0} - a(h - m_i). \tag{3.5}$$

The algorithm of the new scheme is illustrated in Algorithm 2.

4 The Parallel Block Gram-Schmidt Orthogonalization

4.1 Parallelization

The **K** supercomputer, which is located in the RIKEN Advanced Institute for Computational Science, is being used to solve large scale problems in various fields including data

Algorithm 2: The New BGS Method for Estimating Optimal Block Size

```
Data: $X \in \mathbb{R}^{n\times n}$
Result: $m$
begin
    $m_1 \longleftarrow 1$;
    for $i = 1 : 5$ do
        $m_i \longleftarrow m * 2$;
        for $j = 0 : 1$ do
            $start \longleftarrow gettimeofday()$;
            The Block Gram-Schmidt Method;
            $end \longleftarrow gettimeofday()$;
            $t_{ij} \longleftarrow end - start$;
        end
        $a \longleftarrow (t_{i0} - t_{i1})/m_i$;
        $r[i] \longleftarrow \frac{1}{2}n^2a + t_{i0} - a(h - m)$;
        for $j$=5:1 do
            $A[ij] = m_i^{j-1}$;
        end
    end
    solve $Ax = b$;
    $f(m) \longleftarrow x_1m^4 + x_2m^3 + x_3m^2 + x_4m + x_5$;
    solve $opt - m \longleftarrow \min_{m\in[0,\frac{1}{2}N]} f(m)$;
end
```

mining and climate prediction. When dealing with large scale problems in these fields, it is integral to parallelize the BGS to speed-up computation time. Vanderstraeten [10] and Gudula [3] developed a parallelized BGS orthogonalization referred to as the PBGS. The PBGS involves less communication and high parallelization compared to the Parallel Modified Gram-Schmidt method. However, like the CGS, the BGS suffers from low accuracy. To address these shortcomings, we have proposed a new scheme that focuses on the speed-up of the BGS without compromising accuracy.

There are a number of existing parallelization procedures, and and the Column-Wise Distribution method is one of them. Other methods are also easily implemented. In the following section, the BGS was parallelized employing Column-Wise Distribution and the optimal block-size was defined.

4.2 Optimal Block-size for the Parallel Block Gram-Schmidt Orthogonalization

We have parallelized the new schemes in section 3.1 for the PBGS, and have proposed a new method which will be referred to as PBGS-m. The PBGS-m is different from the PBGS in that the PBGS-m can determine optimal block-size automatically. However, it should be noted that optimal m is not consistent in this scenario either. The critical point of this method can be summarized as follows: Let m_i be $m_i = 2^i,\ i = 1, \cdots,\ 4$ and $X_{\text{block}} \in \mathbb{R}^{n\times m}$ and $Q \in \mathbb{R}^{n\times h}$ be the matrix. In the first step, one PE sends X_{block} to

Algorithm 3: PBGS-m Method

```
Data: $X \in \mathbb{R}^{n\times n}$
Result: $m$
begin
    $m_1 \longleftarrow 1$;
    for $i = 1 : 5$ do
        $m_i \longleftarrow m * 2$;
        for $j = 0 : 1$ do
            if myrank==0 then
                $start \longleftarrow gettimeofday()$;
            end
            The Parallel Block Gram-Schmidt Method;
            if myrank==0 then
                $end \longleftarrow gettimeofday()$;
                $t_{ij} \longleftarrow end - start$;
            end
        end
        if myrank==0 then
            $a \longleftarrow (t_{i0} - t_{i1})/m_i$;
            $r[i] \longleftarrow \frac{1}{2}n^2a + t_{i0} - a(h - m)$;
            for j=5:1 do
                $A[ij] \longleftarrow m_i^{j-1}$;
            end
        end
    end
    if myrank==0 then
        solve $Ax = b$;
        $f(m) \longleftarrow x_1m^4 + x_2m^3 + x_3m^2 + x_4m + x_5$;
        solve $opt - m := \min_{m\in[0,\frac{1}{2}N]} f(m)$;
    end
end
```

the other PEs, and the PBGS with m_i for two steps as sample points, is executed. The practical relationship between t_i and m can almost be classified as a linear function. In the next step, the computation time of the BGS is approximated:

$$a = (t_{i0} - t_{i1})/m_i, \quad T_i := \frac{1}{2}n^2a + t_{i0} - a(h - m_i).$$

From the computation time t_i and the complexity of the two steps, the total computation time T_i for each m_i in each PE is estimated according to the method proposed by Matsuo *et al.* [5]. One PE gathers the information of T_i. Then using the information of samples T_i and m_i, the fastest computation time and optimal m are approximated. Please refer to Algorithm 3 for the PBGS-m algorithm.

Table 1: BGS versus BGS with Scheme A, B, C: BCSSTK

Name	Matrix Size	m_{TRIAL}	t_m	m_{A}	$t_{m_{\text{A}}}$	m_B	$t_{m_{\text{B}}}$	m_C	$t_{m_{\text{C}}}$
BCSSTK02	112×112	60	0.0008	32	0.0013	41	0.0014	51	0.0009
BCSSTK06	420×420	40	0.065	128	0.076	41	0.066	15	0.077
BCSSTK15	3948×3948	80	35.01	1024	64.93	43	39.27	53	37.3316
BCSSTK19	11948×11948	100	872.63	1024	1137.3	43	1035.5	55	959.9

5 Numerical Experiments

This section evaluates the Block Gram-Schmidt orthogonalization and Parallel Block Gram-Schmidt orthogonalization with an optimal block-size.

- m_{TRIAL}: Block-size is determined by trial and error
- m_{A}: Block-size of the BGS with Scheme A from Matsuo *et al.* [5]
- m_{B}: Block-size of the BGS with Scheme B from Matsuo *et al.* [5]
- m_{C}: Block-size of the BGS with the new scheme
- m_{D}: Block-size of the PBGS-m (in section 4.2)
- t: Time (sec)
- PE: Number of Processor Elements

Algorithms were programmed in C-language with double precision.

5.1 The Block Gram-Schmidt Method with Optimal Block-size

In this subsection, we have compared the QR decomposition of a conventional BGS versus a BGS with schemes A, B and C [5], to illustrate the effectiveness of this new method. The algorithms were run on the *sun fire X2250* with a 4G byte main memory.

5.1.1 Example 1

The test matrices selected for this experiment were BCSSTK02, 06, 15 and 19 from the Matrix Market [4]. These are nonsymmetric real matrices which arise from eigenvalue problems. The data from these numerical experiments are shown in Table 1. In Table 1, the figures for m_{A} became very large as the matrix sizes were large. In contrast, m_{B} resulted in similar values for each problem. m_{A}, m_{B} and m_{C} took the nearest value m to block-size m_{OPT} for BCSSTK02, 15 and 19. The computation times for $t_{m_{\text{C}}}$ were also faster than that of $t_{m_{\text{A}}}$ and $t_{m_{\text{B}}}$. It can be concluded that the performance of Scheme C was the best in this experiment.

5.1.2 Example 2

The test matrices selected for this experiment were CAVITY03, 06, 10 and 19 from the Matrix Market [4], which are symmetric real matrices which arise from the discretization

Table 2: BGS versus BGS with Schemes A, B, C: CAVITY

Name	Matrix Size	m_{TRIAL}	t_m	m_{A}	$t_{m_{\text{A}}}$	m_{B}	$t_{m_{\text{B}}}$	m_{C}	$t_{m_{\text{C}}}$
CAVITY03	317×317	90	0.035	128	0.041	41	0.037	55	0.038
CAVITY06	1182×1182	40	1.092	128	1.30943	1.175	13	1.601	
CAVITY10	2597×2597	80	10.57	256	15.25	46	11.47	55	11.00
CAVITY19	4562×4562	100	53.38	256	62.36	46	59.26	55	56.95

of partial differential equations. The results of the numerical experiments are shown in Table 2.

The results were similar to Example 1. m_{A} ended-up with large values as the size of the matrices were large. In contrast, m_{B} resulted in similar values for each problem. m_{A}, m_{B} and m_{C} took the nearest value m to the block-size m_{OPT} for CAVITY10 and 19. The computation times for $t_{m_{\text{C}}}$ were also faster than the computation times for $t_{m_{\text{A}}}$ and $t_{m_{\text{B}}}$. The performance of Scheme C was better than the other schemes in this experiment as well.

5.2 The Parallel Block Gram-Schmidt Method

In this subsection, numerical experiments were implemented in the environment below, to illustrate the effectiveness of the PBGS-m. The derived algorithm was programed in C-language using double precision and run on a *Six-Core AMD Opteron Processor 2439 SE* which has 12 processors and an *AuthenticAMD* with a 2.8GHz CPU.

5.2.1 Example 3

The performances of the PBGS and the PBGS-m were compared in this section. The test matrices were the same as those used in Example 1. BCSSTK15 and BCSSTK19 were selected from the Matrix Market [4]. These are nonsymmetric real matrices which arise from eigenvalue problems. Numerical experiments are shown in Table 3, Table 4 and Table 5. For each problem and each processor element, the computation time of the PBGS-m was faster than that of the BGS. For each problem, parallelization rendered block-size m_{D} large, in comparison to block-size m_{OPT}. This is why the matrix size orthogonalized for one step, became large when using several processors.

5.2.2 Example 4

The performances of the PBGS-m and the PBGS were compared with different block-sizes: $m = 50$, 100, 200 and 300. In this experiment, the total number of processor elements was eight. The test matrix was the same as that used in Example 1. BCSSTK15 from the Matrix Market [4], is a nonsymmetric real matrix which arises from eigenvalue problems. The results of the numerical experiments are shown in Table 6. In this table, speed-up refers to the ratio of speed-up of each method versus those of the PBGS-m. The data for the PBGS-m was not the fastest in Table 6. However, when $m = 50$, the

Table 3: BGS versus PBGS-m 1

Name	Matrix Size	m_{TRIAL}	t_m	$m_{\text{D(PE2)}}$	$t_{\text{D(PE2)}}$	$m_{\text{D(PE4)}}$	$t_{\text{D(PE4)}}$
BCSSTK15	3948×3948	80	35.01	150	28.40	210	18.15
BCSSTK19	11948×11948	100	872.6	180	524.0	230	489.3

Table 4: BGS versus PBGS-m 2

Name	Matrix Size	m_{TRIAL}	t_m	$m_{\text{D(PE6)}}$	$t_{\text{D(PE6)}}$	$m_{\text{D(PE8)}}$	$t_{\text{D(PE8)}}$
BCSSTK15	3948×3948	80	35.01	250	15.18	260	15.39
BCSSTK19	11948×11948	100	872.6	290	451.3	320	440.2

Table 5: BGS versus PBGS-m 3

Name	Matrix Size	m_{TRIAL}	t_m	$m_{\text{D(PE10)}}$	$t_{\text{D(PE10)}}$	$m_{\text{D(PE12)}}$	$t_{\text{D(PE12)}}$
BCSSTK15	3948×3948	80	35.01	280	14.83	310	15.55
BCSSTK19	11948×11948	100	872.6	380	392.6	420	376.3

Table 6: PBGS-m: BCSSTK15

Algorithm	m	t	Speed-up
PBGS (PE8)	50	35.67	1.96
PBGS (PE8)	100	21.22	1.17
PBGS (PE8)	200	15.62	0.86
PBGS (PE8)	300	16.20	0.89
PBGS-scheme D (PE8)	180	18.16	1.00

PBGS-m was twice as fast as the PBGS, and and when $m = 100$, the PBGS-m was also faster. When $m = 200$ and 300, the PBGS-m was slower by about 10-15%.

6 Conclusion

The Parallel Block Gram-Schmidt method can compute the QR factorization rapidly. However, determining optimal block-size has always been an issue. To address this issue, we have developed a new method that automatically determines block-size, and the results of our numerical experiments in section 5.2.1 and 5.2.2 show that the PBGS-m is an effective option.

In our paper, a number of ideas used in the development and extension of the Block Gram-Schmidt and Parallel Block Gram-Schmidt algorithms were explored, and four schemes on how to determine optimal block-size when applying the BGS and the PBGS were proposed.

The key role of our schemes was to determine optimal block-size automatically through the use of sample points. By applying this method, the BGS can be implemented more efficiently. The numerical experiments in section 5.1.1 and 5.1.2 also illustrate the effectiveness of our new schemes for almost any problem.

References

[1] Chang, X. W., "*On the Perturbation of the Q-factor of the QR factorization,*" Numer. Lin. Alg. Appl., Vol.19, pp. 607–619, 2012.

[2] Elden, L., and Park, H., "*Block Downdating of Least Squares Solutions,*" SIAM J. Matrix Anal. Appl., Vol. 15, pp. 1018–1034, 1994.

[3] Gudula, R. and Michael, S., "*Comparison of Different Parallel Modified Gram-Schmidt Algorithms,*" Euro-Par 2005, LNCS 3648, pp. 826–836, 2005.

[4] "*MATRIX MARKET,*" http://math.nist.gov/MatrixMarket/, Information Technology Laboratory of the National Institute of Standards and Technology, USA.

[5] Matsuo, Y., Nodera, T., "*The Optimal Block-Size for the Block Gram-Schmidt Orthogonalization,*" J. Sci. Tech, Vol. 49, pp. 348–354, 2011.

[6] Paige, C. C., and Elden, L., "*Loss and Recapture of Orthogonality in the Modified Gram-Schmidt Algorithm,*" SIAM J. Matrix Anal. Appl., Vol. 13, pp. 176–190, 1992.

[7] Qiaohua, L., "*Modified Gram-Schmidt-based Methods for Block Downdating the Cholesky Factorization,*" J. Comput. Appl. Math., Vol. 235, pp. 1897–1905, 2011.

[8] Stewart, G. W., "*Block Gram-Schmidt Orthogonalization,*" SIAM J. Sci. Comput., Vol. 31, pp. 761–775, 2008.

[9] ——, "*The Effect of Rounding Errors on an Algorithm for Downdating a Cholesky Factorization,*" J. Inst. Math. Appl., Vol. 23, pp. 203–213, 1979.

[10] Vanderstraeten, D., "*An Accurate Parallel Block Gram-Schmidt Algorithm without Reorthogonalization,*" Numer. Lin. Alg. Appl., Vol. 7, pp. 219–236, 2000.

[11] Yoo, K. and Park, H., "*Accurate Downdating of a Modified Gram-Schmidt OR Decomposition,*" BIT, Vol. 36, pp. 166–181, 1996.