A Stable and Efficient Parallel Block Gram-Schmidt Algorithm

Denis Vanderstraeten

Katholieke Universiteit Leuven, Computer Science Dept.,
Celestijnenlaan, 200A, B-3001 Heverlee (Belgium)
Denis.Vanderstraeten@@cs.kuleuven.ac.be

Abstract. The Modified Gram-Schmidt (MGS) orthogonalization process — used for example in the Arnoldi algorithm — constitutes often the bottleneck that limits parallel efficiencies. Indeed, a number of communications, proportional to the square of the problem size, is required to compute the dot-products. A block formulation is attractive but it suffers from potential numerical instability.
In this paper, we address this issue and propose a simple procedure that allows the use of a Block Gram-Schmidt algorithm while guaranteeing a numerical accuracy similar to MGS. The main idea is to dynamically determine the size of the blocks. The main advantage of this dynamic procedure are two-folds: first, high performance matrix-vector multiplications can be used to decrease the execution time. Next, in a parallel environment, the number of communications is reduced. Performance comparisons with the alternative Iterated CGS also show an improvement for moderate number of processors.

1 Introduction

The numerical simulation of physical problems often requires the construction of an orthogonal basis of a matrix or a set of vectors. Among the existing techniques, the Gram-Schmidt algorithms have proven to be successful candidates. In particular, the Modified Gram-Schmidt algorithm is often used in GMRES to built an orthogonal basis of the Krylov subspace. Unfortunately, in a parallel environment, the number of communications grows like the square of the number of vectors.

Various attempts have been proposed to overcome this problem. The Classical Gram-Schmidt algorithm or a block formulation of MGS provide a substantial reduction of the number of communications but a reorthogonalization procedure must be introduced to ensure the numerical accuracy [5, 8, 9]. In the particular context of Krylov basis, much attention has also been devoted and we refer the reader to [10, 11, 12] for further readings.

Our work departs from other works by the assumptions that are made: (a) we do not assume that GMRES (or any Krylov method) is our target application, and (b) we wish to avoid any reorthogonalization step to limit the additional cost compared to MGS.

P. Amestoy et al. (Eds.): Euro-Par'99, LNCS 1685, pp. 1128–1135, 1999.

2 Gram-Schmidt Algorithms

The Modified Gram-Schmidt algorithm produces a QR factorization of a matrix $A = [a_1 \dots a_n] \in \mathbb{R}^{m \times n}$ with $m \geq n$ [3, 4]. The pseudo-code for MGS is given in Alg. 1 where q_i represents the ith column of Q.

```
   for i = 1 to n
1.      q_i = a_i
        for j = 1 to i - 1
2.          δ = q_j^T q_i
3.          q_i = q_i - q_j δ
        end
4.      q_i = q_i / ||q_i||_2
   end
```

Alg. 1. Modified Gram-Schmidt

```
   for k = 1 to p
1.      W_k = A_k
        for j = 1 to k - 1
2.          Δ = Q_j^T W_k
3.          W_k = W_k - Q_j Δ
        end
4.      Q_k = MGS(W_k)
   end
```

Alg. 2. Block Gram-Schmidt

On a parallel computer where the components of a_i and q_i are distributed over the processors, a synchronization, followed by the global accumulation of the local dot-products must be performed at step 2 of the algorithm. The application of MGS thus requires $n^2/2$ communications.

To reduce the number of communications, one can use the Classical Gram-Schmidt algorithm but its numerical accuracy is not guaranteed. To regain the accuracy, the Iterated Classical Gram-Schmidt algorithm (ICGS) performs two steps of Classical Gram-Schmidt if a loss of orthogonality between the vectors is detected [8]. The number of communications of ICGS is in $\mathcal{O}(n)$ but the computational work may be doubled compared to MGS.

Another approach towards the reduction of communication cost is to focus on a block formulation [9]. To this end, the matrix $A = [A_1 \dots A_p]$ is partitioned in p blocks $A_k \in \mathbb{R}^{m \times s}$ of s consecutive columns (we assume for simplicity that n is a multiple of s). We partition Q accordingly. The Block Gram-Schmidt algorithm (BGS) is given in Alg. 2. Although BGS achieves better parallelism than MGS, it is not as stable. Indeed, Jalby and Philippe [9] show that the matrix $\hat{Q}$, computed by BGS, satisfies

$$||\hat{Q}^T\hat{Q} - I||_2 \leq \rho \boldsymbol{u} \max_{k=1\dots p} \kappa_2(W_k)\ \kappa_2(A), \tag{1}$$

where W_k is the matrix used in step 4 of Alg. 2 and $\kappa_2(\cdot)$ denotes the condition number. Hence, if any block W_k has a condition number close to $\kappa_2(A)$, the bound (1) is of the order $\kappa_2^2(A)$ and BGS may produce an inaccurate result. A simple remedy to this problem consist in applying MGS (or CGS) twice to every block W_k. This is referred as B2GS in [9].

In the next section, we present a simple modification of BGS to ensure numerical stability still benefiting from the desirable parallel properties of BGS but without introducing a reorthogonalization step.

3 Dynamic Block Gram-Schmidt (DGS)

We have implicitly assumed that all blocks have the same size. This is not a requirement and allowing different block sizes gives room for improvement of BGS. Indeed, if we are able to find a decomposition of A such that every block W_k satisfies

$$\kappa_2(W_k) \leq \tau \qquad\qquad k = 1 \ldots p, \tag{2}$$

where the parameter τ is a small threshold, then, we will have an algorithm with essentially the same bound as MGS.

We proceed iteratively to determine the block sizes. Assume that the size of the first $p-1$ blocks have been determined and that the current block $W_p \in \mathbb{R}^{m\times(s-1)}$ satisfies (2). The next vector w_s is first orthogonalized with respect to Q_k, $(k = 1 \ldots p-1)$. Then, if the extended matrix $W_p^* = [W_p \; w_s]$ still satisfies (2), we can safely add w_s in the current block. Otherwise, a new block W_{p+1} is created with w_s as unique column.

Usually, the block Q_p is created along with the iterations and we should avoid keeping a copy of W_p for memory reasons. However, step 4 of Alg. 2 produces a factorization $W_p = Q_p R_p$. Since,

$$\kappa_2(W_p) = \kappa_2(R_p), \tag{3}$$

if suffices then to keep the matrix R_p. Strictly speaking, in finite precision arithmetic, (3) is not exact but $\kappa_2(R_p)$ provides a sufficient approximation.

Computing the condition number is an expensive operation, even for small matrices. Instead of computing it exactly, we use the algorithm proposed by Bischof [2] that estimates the condition number of a triangular matrix. The cost of this algorithm is in $\mathcal{O}(s^2)$ for a matrix of order s. The main idea is to exploit the relation

$$||R^{-1}||_2 = \max_{R^T x = d, d\neq 0} \frac{||d||_2}{||x||_2}, \tag{4}$$

together with the fact that R is constructed column by column. Assume that we have evaluated a vector $x \in \mathbb{R}^{s-1}$ with small (preferably the smallest) norm and a unit-norm vector d such that $R^T x = d$. The orthogonalization of w_s leads to the creation of an extended matrix

$$R^* = \begin{pmatrix} R & r \\ 0 & \rho \end{pmatrix}. \tag{5}$$

We need to determine x^* and d^* solution of ${R^*}^T x^* = d^*$ and such that the norm of x^* is small. If we choose $d^* = [cd, s]^T$ with $c = \cos\theta$ and $s = \sin\theta$, then the solution x^* is easily expressed in terms of x by

$$x^* = \begin{pmatrix} cx \\ (s - cr^T x)/\rho \end{pmatrix}. \tag{6}$$

The minimization of $||x^*||_2$ reduces to the one-dimensional problem of finding the optimal θ. It requires only a few dot-products and the computational cost is in $\mathcal{O}(s)$.

The pseudo-code for the Dynamic Block Gram-Schmidt algorithm is given in Alg. 3. The parameter s_{max} is introduced to limit the sizes of the blocks. Note also that the function MGS needs only to orthogonalize w_s with respect to Q_p and returns $Q_p^* = [Q_p\ q_s]$.

```
p = 1, s = 1, Q_1 = [ ]
for i  = 1 to n
      w_s = a_i
      for k = 1 to p − 1                    Orthogonalize wrt. the previous blocks
            d   = Q_k^T w_s
            w_s = w_s − Q_k d
      end
      [Q_p^*, R_p] = MGS([Q_p w_s])         Update the MGS of the current block
      if (κ_2(R_p) > τ or s ≥ s_max) then
            Q_{p+1} = [q_s]                 Create a new block
            s       = 2
            p       = p + 1
      else
            Q_p = Q_p^*                     Increase the current block size
            s   = s + 1
      end
end
```

Alg. 3. Dynamic Block Gram-Schmidt

4 Complexity Analysis

On a parallel computer, every vector v is decomposed into P sub-vectors $v^{(k)}$ of size m/P that are distributed to the P processors. A parallel dot-product $v^T w$ requires the accumulation of the P local dot-products ${v^{(k)}}^T w^{(k)}$. As usual, a point-to-point communication of size s between two processors is modelled by

$$T_{sr}(s) = \alpha + \beta s.$$

where α is the latency and β^{-1} denotes the bandwidth. The communication graph for the accumulation of the local dot-products is represented by a binary tree resulting in a total accumulation time, for s concurrent dot-products, of

$$T_{acc}(s) = 2 \log_2 P(\alpha + \beta s).$$

Let s denote the average block size of DGS. A count of the communications gives a total communication cost of

$$T^{(comm)} = \frac{n}{2} \log_2 P \left((s + \frac{n}{s})\alpha + (n+1)\beta \right). \tag{7}$$

The algorithm DGS behaves like MGS but with a reduced startup time. If $s = 1$ or $s = n$, we obtain the communication cost of MGS. In addition, it is easy to show that the communication cost is minimized for $s = \sqrt{n}$.

Using DGS also gives a gain in terms of computational cost. Indeed, matrix-vector products with a high flop rate can be used instead of the vector-vector operations of MGS. If we model the cost of evaluating the condition number by δs^2, the computational cost of DGS writes

$$T^{(comp)} = (\gamma_1(s+1) + \gamma_s(n-s))\, mn + \delta ns, \tag{8}$$

where γ_s^{-1} denotes the flop rate to perform a matrix-vector product with a matrix of size $m \times s$, and γ_1 is the cost of a dot-product (this corresponds to the BLAS2 `dgemv` and the BLAS1 `ddot` operations). The first term corresponds to the orthogonalization and the second represents the overhead due to the evaluation of the condition number. It can be neglected.

The above analysis highlights the two potential improvements of DGS compared to MGS: a higher flop rate associated with the reduction of the number of communications. On a specific platform, an estimate of the parameters α, β and γ_s enables a detailed analysis of the execution time and the determination of the optimal block size. Remember however, that the size of the blocks is primarily set by the problem itself and stiff problems may require blocks of size 1.

5 Numerical Experiments

5.1 Loss of Orthogonality

Table 4 presents accuracy results obtained with MGS, BGS and DGS. The parameters were set as $\tau = 10$ and $s_{max} = 8$. For comparison purposes, we also show results produced by B2GS and ICGS using reorthogonalization.

Three matrices are considered: (a) a random matrix of size 1024×512 with entries uniformly distributed in $[-1, 1]$, (b) a Lauchli matrix of size 65×64 with $\epsilon = 10^{-4}$, and (c) a Hilbert matrix of size 20×10. The orthogonality is measured by $||Q^TQ - I||_2$ and compared with a "reference" orthogonal matrix Q_{ref} obtained by applying MGS twice.

	Random		Lauchli		Hilbert	
	$\|\|Q^TQ-I\|\|_2$	$\|\|Q-Q_{ref}\|\|_2$	$\|\|Q^TQ-I\|\|_2$	$\|\|Q-Q_{ref}\|\|_2$	$\|\|Q^TQ-I\|\|_2$	$\|\|Q-Q_{ref}\|\|_2$
MGS	8.7e-15	6.8e-15	3.8e-13	3.8e-13	2.4e-6	2.4e-6
BGS	8.9e-15	7.0e-15	6.5e-9	6.5e-9	6.0e-4	6.0e-4
DGS	1.1e-14	7.8e-15	3.8e-13	3.8e-13	3.5e-6	3.9e-6
B2GS	9.6e-15	6.6e-15	3.8e-13	3.8e-13	2.4e-6	3.3e-6
ICGS	1.6e-14	8.2e-15	2.9e-16	3.8e-16	1.3e-14	3.7e-6

Table 4. Loss of orthogonality of Q computed by the various Gram-Schmidt algorithms

As expected, B2GS gives results of the same order as MGS while ICGS produces an output that is within the machine accuracy.

The Random matrix is fairly well conditioned and all three algorithms produce accurate results. This is no longer the case for the ill-conditioned problems where BGS fails to construct an orthonormal basis. For the Lauchli matrix, the average block size is 7.11. Indeed, all the vectors are nearly aligned with the first vector, but their projections, parallel to this vector, turn out to be (almost) orthogonal. As a consequence, DGS begins by creating a block of size 1 while the subsequent blocks have size s_{max}. With the Hilbert matrix, with the exception of the first block of size 2, all blocks of DGS have size 1.

5.2 Parallel Performance

Parallel performance were measured on an IBM/SP2 with 8 processors using the vendor version of MPI. Results are presented for three different matrices of size 100 000 × 60, 100 000 × 120, and 200 000 × 60. These sizes are typical for the orthogonalization of Krylov bases arising in the solution of large linear systems with GMRES.

	100000 × 60				100000 × 120				200000 × 60			
s_{max}	$P=1$	2	4	8	$P=1$	2	4	8	$P=1$	2	4	8
MGS	1.00	0.91	0.74	0.44	1.00	0.92	0.73	0.46	1.00	0.92	0.83	0.51
DGS 1	1.00	0.91	0.73	0.44	0.99	0.91	0.74	0.46	1.00	0.94	0.83	0.59
DGS 2	1.11	1.02	0.85	0.64	1.10	1.03	0.61	0.60	1.12	1.06	0.85	0.66
DGS 4	1.70	1.53	1.31	0.80	1.77	1.59	1.37	1.00	1.72	1.58	1.43	1.12
DGS 8	1.94	1.75	1.49	1.08	2.15	1.88	1.67	1.15	1.97	1.77	1.58	1.14
DGS 16	1.85	1.63	1.33	0.97	2.22	1.91	1.49	1.14	1.87	1.69	1.30	1.15
B2GS	1.91	1.70	1.46	1.05	2.11	1.86	1.69	1.18	1.90	1.72	1.48	1.18
CGS	2.86	2.68	2.28	1.72	3.10	2.88	2.44	2.08	2.92	2.70	2.36	1.90

Table 5. Parallel efficiencies compared to the sequential MGS algorithm

Table 5 presents the efficiencies computed using the sequential time obtained with MGS. Efficiencies above 100% result from the reduction in communication time *combined* with the gain due to the more efficient BLAS2 operations.

Random matrices are usually well conditioned. No *single* reorthogonalization step is required for ICGS and this algorithm become thus identical to CGS. Depending on the matrices, the efficiency of ICGS can drop to half of the efficiency of CGS.

5.3 Application: GMRES

The dynamic block Gram-Schmidt procedure has been tested for the solution of an advection-diffusion problem defined on a square, discretized by a finite-element method. The resulting linear system has 160 000 unknowns. It is solved

iteratively with GMRES, preconditioned by a perfectly parallel block-ILU(0). The restart parameter is set to 30 vectors and convergence is attained when the *true* residual satisfies $||b - Ax||_2 \leq 10^{-4}||b||_2$.

Table 6 presents performance results obtained on an a 8-processors IBM/SP2 parallel computer, using the PETSc framework [1]. Both the total solution time of the linear system and the orthogonalization time are measured. The parallel efficiency is also measured.

	$P = 1$	$P = 2$		$P = 4$		$P = 8$	
	$T^{(tot)}$	$T^{(tot)}$	Eff.	$T^{(tot)}$	Eff.	$T^{(tot)}$	Eff.
Matvec	65	37	0.87	19	0.85	10	0.81
Precond	62	31	1.00	15	1.03	8.5	0.91
MGS	83	47	0.88	30	0.69	22	0.47
DGS	57	32	0.89	20	0.71	14	0.50
ICGS	95	50	0.95	28	0.84	20	0.59
Total (MGS)	288	153	0.94	92	0.78	63	0.57
Total (DGS)	263	140	0.93	83	0.79	55	0.60
Total (ICGS)	299	159	0.94	91	0.82	60	0.62

Table 6. Performance of the Dynamic Gram-Schmidt algorithm with GMRES

For 8 processors, DGS is 1.5 times faster than MGS, resulting in a reduction of 17% of the total computation time for the solution of the linear system. Of course, the global improvement depends on the relative cost of the orthogonalization compared to the other operations.

For this problem, about 80% of the Krylov vectors are reorthogonalized with ICGS. Following [8], modifying some parameters would decrease this percentage and produce a more competitive algorithm.

With GMRES, the value of τ has little influence as long as the matrix is reasonably well conditioned. Indeed, even for moderately large τ, the same number of iterations was needed to obtain convergence. First, restarting the iterations gives a smoothing effect on the loss of orthogonality. Next the linear iterations reach convergence to the desired value long before the effect of τ becomes apparent on the residual norm. This observation is in accordance with the results in [6, 7]. Indeed, these authors note that the important feature is the linear independence of the Krylov vectors and not the orthogonality. For this matrix, with the restart parameter and termination criterion, all three algorithms converge in the same number of iterations.

6 Concluding Remarks

We have presented an improved version of the Block Gram-Schmidt algorithm that enables its use on parallel computers while guaranteeing a numerical stability. The main idea is to dynamically determine the size of the blocks during the computation according to their condition number. An efficient algorithm for

estimating the condition number of small matrices makes the overall procedure very attractive.

In sequential mode, DGS already outperforms the widely used Modified Gram-Schmidt algorithm. Furthermore, in a parallel environment, an additional gain is achieved by the reduction in the number of communications.

Acknowledgements

This research is partially funded by the UIAP P4/02 Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science Technology and Culture. The author is a post-doctoral researcher funded by the *Fonds voor Wetenschappelijk Onderzoek*, Flanders, Belgium. The scientific responsibility rests with its author.

References

[1] S. Balay, W. D. Gropp, L. Curfman McInnes, and B. F. Smith. PETSc home page. http://www.mcs.anl.gov/petsc, 1998.

[2] C. H. Bischof. Incremental Condition Estimation. *SIAM J. Mat. Anal. Appl.*, No. 2, pp. 312–322, 1990.

[3] Å. Björck. Numerics of Gram-Schmidt Orthogonalization. *Linear Alg. Appl.*, pp. 297–316, 1994.

[4] Å. Björck and C. C. Paige. Loss and Recapture of Orthogonality in the Modified Gram-Schmidt Algorithm. *SIAM J. Mat. Anal. Appl.*, No. 1, pp. 176–190, 1992.

[5] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Steward. Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization. *Mathematics of Computations*, Vol. 30, No. 136, pp. 772–795, 1976.

[6] J. Drkošová, M. Rozložník, Z. Strakoš, and A. Greenbaum. Numerical Stability of GMRES. *BIT*, Vol. 35, pp. 309–330, 1995.

[7] A. Greenbaum, M. Rozložník, and Z. Strakoš. Numerical Behaviour of the Modified Gram-Schmidt GMRES Implementation. *BIT*, Vol. 37, No. 3, pp. 706–719, 1997.

[8] W. Hoffmann. Iterative Algorithms for Gram-Schmidt Orthogonalization. *Computing*, Vol. 41, pp. 335–348, 1989.

[9] W. Jalby and B. Philippe. Stability Analysis and Improvement of the Block Gram-Schmidt Algorithm. *SIAM J. Sci. Stat. Comput.*, No. 5, pp. 1058–1073, 1991.

[10] R. B. Sidje. Alternatives for Parallel Krylov Subspace Basis Computations. *Num. Lin. Alg. with Appl.*, Vol. 4, No. 4, pp. 305–331, 1997.

[11] E. De Sturler and H. A. van der Vorst. Reducing the Effect of Global Communication in GMRES(m) and CG on Parallel Distributed Memory Computers. *Applied Numerical Mathematics*, Vol. 18, pp. 441–459, 1995.

[12] H. F. Walker. Implementation of the GMRES Method Using Householder Transformation. *SIAM J. Sci. Stat. Comput.*, Vol. 9, No. 1, pp. 152–163, 1988.