An Approach to Decrease Fill-in in Sparse Orthogonalizations on a MIMD Computer *

J. Touriño and R. Doallo
E-mail: {juan, doallo}@udc.es
Dep. Electronica y Sistemas. University of La Coruña
Campus de Elviña, s/n. 15071 La Coruña, Spain

Abstract

Modified Gram-Schmidt, Householder transformations and Givens plane rotations are popular methods in linear algebra based on orthogonalization. The major advantage of the orthogonal factorization is its stability. We present a general heuristic strategy of fill-in control in the rank-revealing sparse QR decomposition, which is not very costly for the execution times, using these three algorithms. This strategy is based on column pivoting and it maintains accuracy in the results, as we show experimentally on the AP1000 for a set of sparse matrices from the Harwell-Boeing collection.

1 Introduction

QR orthogonalization appears in several applications of linear algebra: linear systems of equations, least squares problems [11], eigenvalue calculation [6, 8], which are necessary to solve in many areas, such as fluid dynamics, circuit simulation, structural analysis $\cdots$ Therefore, it is necessary high-quality software for those scientific numeric computations; for instance, a sparse QR factorization implementation in *Matlab* is described in [10].

Since sequential algorithms for computing this factorization have a high arithmetic complexity, the development of parallel algorithms is of considerable interest. Our algorithms take into account the sparse nature of the matrices; although a sparse problem could be treated with a parallel program for dense factorization, the cost of ignoring sparsity, both in storage and time, would erase the benefits provided by parallel processing.

*This work was supported by the Ministry of Education and Science of Spain under project CICYT TIC96-1125-C03

2 Orthogonalization Algorithms

Decomposition of a matrix $A \in \Re^{m\times n}$ into the product of an orthogonal matrix $Q \in \Re^{m\times n}$ ($Q^T = Q^{-1}$) and an upper triangular matrix $R \in \Re^{n\times n}$, is the aim of orthogonalization algorithms. Modified Gram-Schmidt (MGS), Householder and Givens are common methods for orthogonalization; they are widely described in [5].

Modified Gram-Schmidt algorithm is the only one that obtains explicitly the matrix Q, which is necessary in some applications such as eigenvalue calculation. It has better numerical properties than its classical version, Classical Gram-Schmidt, which presents a loss of orthogonality among the computed columns of Q.

Householder transformations or reflections are applied to the original matrix A; each transformation zeroes the subdiagonal elements of one column of the matrix and, after n iterations, the upper triangular matrix R is obtained.

Givens rotations annihilates entries more selectively: each rotation involves two rows so that the first element of the second row is zeroed. Thus, the appropiate entries of matrix A are annihilated to obtain matrix R.

Our algorithms incorporate column pivoting to take into account rank-deficiency situations; thus, the rank of the matrix can be calculated accurately when column swappings are performed during the decomposition (rank-revealing factorization). Basically, these algorithms consist of as many iterations as columns (n) of the original matrix A (except for rank-deficiency matrices). The column pivoting strategy used in each iteration is the following:

$$\text{Find } p,\ k \leq p \leq n, \text{ such that} \\ norm_p = \max_{k \leq j \leq n} norm_j \qquad (1)$$

where $norm_j$ is the euclidean norm of column j, p is the index of the pivot column and k is the index of the current iteration.

The design of the corresponding parallel algorithms used for obtaining the experimental results in the next sections are described in [1], [2] and [14], for MGS, Householder and Givens, respectively. We assume a mesh topology and a cyclic distribution of data (double precision floating point entries were used), which are stored in one-dimensional, column-oriented doubly linked lists.

3 Fill-in Decrease and Accuracy of the Results

3.1 Controlling Fill-in

In order to decrease fill-in, we could apply techniques for reordering the original matrix (for instance, [12] for Givens rotations and [9] for Cholesky factorization), many of which are sequential by nature. We have implemented a general heuristic criterion to preserve the sparsity by taking advantage of the column pivoting. This strategy is based on the Markowitz criterion for the LU factorization [3, Chapter 7]. We use the following new criterion to select the pivot column instead of expression (1):

$$\text{Find } p,\ k \leq p \leq n, \text{ such that} \\ \left\{ \epsilon \left(\frac{zero_p}{\max\limits_{k \leq j \leq n} zero_j} \right) + (1-\epsilon) \left(\frac{norm_p}{\max\limits_{k \leq j \leq n} norm_j} \right) \right\} \qquad (2) \\ \text{is maximum and } norm_p \geq \alpha$$

where $zero_j$ is the number of null elements in column j for MGS, and the number of null entries in the subdiagonal elements of column j, for Householder and Givens. A fill-in parameter ϵ, $0 \leq \epsilon \leq 1$, is introduced; for $\epsilon = 0$ the pivot column selection strategy is equivalent to the one shown in (1). Although we try to control fill-in as much as possible by choosing a pivot column with many null elements ($\epsilon \approx 1$), we shall keep a reasonable degree of numerical stability in the factorization by discarding as pivot columns those ones with norm close to zero, that is, $norm_p < \alpha$ (if $y \in \Re < \alpha$, y is considered zero; in our algorithms, $\alpha = 10^{-20}$).

Matrix	m×n	#A	% #A
SHL400	663×663	1712	0.39%
JPWH991	991×991	6027	0.61%
SHERMAN1	1000×1000	3750	0.37%
MAHINDAS	1258×1258	7682	0.49%

Table 1: *Harwell-Boeing test matrices*

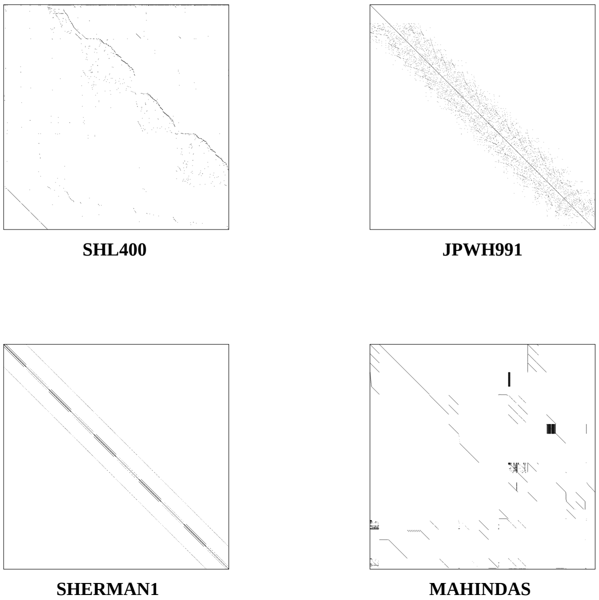

Figure 1: Sparse matrices patterns

We have selected four matrices from the Harwell-Boeing sparse matrix collection [4] with the purpose of testing this strategy. These matrices are shown in table 1, where $m \times n$ is the matrix size, $\#A$ is the number of nonzero entries and $\%\#A$ is the percentage of these entries. The corresponding patterns are illustrated in figure 1. The scientific fields in which the test matrices are used are linear programming problems, circuit physics modelling, oil reservoir simulation and economic modelling.

Matrix	$\epsilon = 0$		$\epsilon = 0.85$		$\epsilon = 1$		%Red
	#R	**% #R**	**#R**	**% #R**	**#R**	**% #R**	
SHL400	154644	70.26%	108159	49.14%	1712	0.78%	98.89%
JPWH991	422312	85.92%	223720	45.51%	174307	35.46%	58.73%
SHERMAN1	117581	23.49%	57421	11.47%	53560	10.70%	54.45%
MAHINDAS	552007	69.71%	105964	13.38%	32392	4.09%	94.13%

Table 2: *Fill-in decrease in MGS using 16 cells*

Matrix	$\epsilon = 0$		$\epsilon = 0.85$		$\epsilon = 1$		%Red
	#R	**% #R**	**#R**	**% #R**	**#R**	**% #R**	
SHL400	168786	76.68%	113609	51.61%	1712	0.78%	98.99%
JPWH991	439220	89.36%	338984	68.96%	259791	52.85%	40.85%
SHERMAN1	136290	27.23%	109016	21.78%	103176	20.61%	24.30%
MAHINDAS	607758	76.75%	215976	27.27%	62471	7.89%	89.72%

Table 3: *Fill-in decrease in Householder using 16 cells*

Matrix	$\epsilon = 0$		$\epsilon = 0.85$		$\epsilon = 1$		%Red
	#R	**% #R**	**#R**	**% #R**	**#R**	**% #R**	
SHL400	154644	70.26%	120221	54.62%	1712	0.78%	98.89%
JPWH991	422312	85.92%	354161	72.05%	152785	31.08%	63.82%
SHERMAN1	118375	23.65%	75834	15.15%	43677	8.73%	63.10%
MAHINDAS	550768	69.55%	156007	19.70%	33715	4.26%	93.88%

Table 4: *Fill-in decrease in Givens using 16 cells*

Matrix	$\epsilon = 0$			$\epsilon = 0.85$			$\epsilon = 1$		
	MGS	**HOU**	**GIV**	**MGS**	**HOU**	**GIV**	**MGS**	**HOU**	**GIV**
SHL400	$7{\cdot}10^{-2}$	$2{\cdot}10^{-11}$	$5{\cdot}10^{-10}$	$7{\cdot}10^{-2}$	$3{\cdot}10^{-10}$	$4{\cdot}10^{-10}$	0	0	0
JPWH991	$6{\cdot}10^{-15}$	$1{\cdot}10^{-14}$	$1{\cdot}10^{-14}$	$6{\cdot}10^{-15}$	$1{\cdot}10^{-14}$	$1{\cdot}10^{-14}$	$7{\cdot}10^{-15}$	$8{\cdot}10^{-15}$	$2{\cdot}10^{-14}$
SHERMAN1	$6{\cdot}10^{-13}$	$8{\cdot}10^{-13}$	$2{\cdot}10^{-13}$	$5{\cdot}10^{-13}$	$3{\cdot}10^{-13}$	$4{\cdot}10^{-13}$	$2{\cdot}10^{-13}$	$4{\cdot}10^{-12}$	$5{\cdot}10^{-13}$
MAHINDAS	$8{\cdot}10^{-7}$	$7{\cdot}10^{-9}$	$5{\cdot}10^{-10}$	$7{\cdot}10^{-7}$	$3{\cdot}10^{-9}$	$8{\cdot}10^{-10}$	$7{\cdot}10^{-7}$	$1{\cdot}10^{-9}$	$6{\cdot}10^{-10}$

Table 5: *Numerical errors using 16 cells*

Fill-in in matrix R for MGS, Householder and Givens is shown in tables 2, 3 and 4, respectively, with ϵ=0, 0.85 and 1; $\#R$ and $\%\#R$ represent the number and percentage of nonzero entries of matrix R, and $\%Red$ is the % of reduction in the fill-in obtained with $\epsilon = 1$ with respect to $\epsilon = 0$. As we can see, for $\epsilon = 0$ (the common criterion of pivoting in dense orthogonalizations) fill-in is very high, but setting $\epsilon = 1$ fill-in is decreased to a large extent (by 77% -Modified Gram-Schmidt-, 63% -Householder- and by 80% -Givens- on average for the set of sparse matrices shown in table 1). Execution times are also substantially reduced with $\epsilon = 1$ because few nonzero elements appear and, therefore, computation savings are achieved; for instance, the execution time for MGS using the matrix *MAHINDAS* and 16 cells is 733 seconds for $\epsilon = 0$ and only 32 seconds for $\epsilon = 1$. A complete set of experimental results about running times in the *Fujitsu AP1000* is presented in [13].

3.2 Numerical Accuracy

Next, we will discuss the effect of the decrease of fill-in on the accuracy of the results obtained in solving Least Squares Problems (LSP) by means of sparse orthogonalization. Our algorithms were adapted to solve LSP; this is described in detail in [13]. LSP consists of calculating a vector $x \in \Re^n$ that minimizes $\|Ax - b\|_2$, where $b \in \Re^m$.

If A is square, LSP is equivalent to solving the linear equation system $Ax = b$; if A is maximum-rank, this system has one unique solution. We

Matrix	1×1 cell			2×2 cells		
	MGS	**HOU**	**GIV**	**MGS**	**HOU**	**GIV**
SHL400	1712	1712	1712	1712	1712	1712
JPWH991	174307	262294	174945	174307	260731	163097
SHERMAN1	53560	104238	51552	53560	102636	56290
MAHINDAS	32392	61412	34157	32392	61220	33785
	4×4 cells			*8×8 cells*		
	MGS	**HOU**	**GIV**	**MGS**	**HOU**	**GIV**
SHL400	1712	1712	1712	1712	1712	1712
JPWH991	174307	259791	152785	174307	259479	179857
SHERMAN1	53560	103176	43677	53560	102227	53706
MAHINDAS	32392	62471	33715	32392	62528	33672

Table 6: *Fill-in in matrix R using several mesh configurations* ($\epsilon = 1$)

Matrix	1×1 cell			2×2 cells		
	MGS	**HOU**	**GIV**	**MGS**	**HOU**	**GIV**
SHL400	0	0	0	0	0	0
JPWH991	$1{\cdot}10^{-14}$	$1{\cdot}10^{-14}$	$2{\cdot}10^{-14}$	$9{\cdot}10^{-15}$	$1{\cdot}10^{-14}$	$2{\cdot}10^{-14}$
SHERMAN1	$3{\cdot}10^{-13}$	$3{\cdot}10^{-12}$	$3{\cdot}10^{-12}$	$3{\cdot}10^{-13}$	$3{\cdot}10^{-12}$	$2{\cdot}10^{-12}$
MAHINDAS	$6{\cdot}10^{-7}$	$2{\cdot}10^{-9}$	$2{\cdot}10^{-9}$	$7{\cdot}10^{-7}$	$3{\cdot}10^{-9}$	$4{\cdot}10^{-10}$
	4×4 cells			*8×8 cells*		
	MGS	**HOU**	**GIV**	**MGS**	**HOU**	**GIV**
SHL400	0	0	0	0	0	0
JPWH991	$7{\cdot}10^{-15}$	$8{\cdot}10^{-15}$	$2{\cdot}10^{-14}$	$6{\cdot}10^{-15}$	$1{\cdot}10^{-14}$	$2{\cdot}10^{-14}$
SHERMAN1	$2{\cdot}10^{-13}$	$4{\cdot}10^{-12}$	$5{\cdot}10^{-13}$	$6{\cdot}10^{-13}$	$1{\cdot}10^{-12}$	$9{\cdot}10^{-13}$
MAHINDAS	$7{\cdot}10^{-7}$	$1{\cdot}10^{-9}$	$6{\cdot}10^{-10}$	$7{\cdot}10^{-7}$	$2{\cdot}10^{-9}$	$3{\cdot}10^{-9}$

Table 7: *Numerical errors using several mesh configurations* ($\epsilon = 1$)

have used the maximum-rank square matrices of table 1 to measure the accuracy of the results; vector b is chosen so that the exact solution of the system is $x = \vec{1}$. The numerical error of the solution is defined as $\max_{1 \le j \le n} |1 - x_{sol}|$, where x_{sol} is the solution of the equation system calculated by means of the orthogonalization.

Table 5 shows the numerical error previously defined for the three algorithms, with ϵ=0, 0.85 and 1, using 16 processing elements or cells. We note that with the sparsity preserve criterion ($\epsilon = 1$), not only do we maintain accuracy in the results with respect to $\epsilon = 0$ (dense matrices criterion), but also we notably improve them on some ocassions (for instance, see matrix *SHL400*).

3.3 AP1000 Scaling Tests

In this section, we are going to analyze the influence of the number of cells of the mesh configuration on the fill-in and on the numerical error. All the results (as well as the previous ones) were measured on the *Fujitsu AP1000* [7] MIMD distributed-memory computer.

As can be observed in table 6, sparsity fluctuates depending on the number of cells. The reason of this is that, when choosing the pivot column (according to expression (2)), it is possible a draw between two (or more) candidate columns. The pivot column selected by means of a reduction instruction varies depending on the mesh configuration, whereupon orthogonalization will be different. In Givens, we have the additional factor that the pairs of rows of A involved in each rotation are different for each mesh configuration and, consequently, fill-in is also different.

The numerical error also varies depending on the number of cells (see table 7) for the same reason explained above. Nevertheless, the variation in the numerical error due to the mesh configuration, within each algorithm, is not significant.

4 Conclusions

We have presented a comparison of the three most common algorithms for performing orthogonalizations. With little additional computations and communications, we have greatly decreased fill-in. This makes possible to use direct methods to solve sparse systems with reasonable execution times instead of iterative methods.

Givens and Householder algorithms provide us the most accurate numerical results, but the latter has the drawback of higher fill-in; so, the algorithm which we consider as most adequate for sparse orthogonalization is the one based on Givens plane rotations.

Acknowledgments

We gratefully thank the Imperial College/Fujitsu Parallel Computing Research Centre (IFPC) in London and the Fujitsu Parallel Computing Research Facilities (FPCRF) in Kawasaki for giving us access to the AP1000 machine.

References

[1] R. Doallo, B.B. Fraguela, J. Touriño, E.L. Zapata. Parallel Sparse Modified Gram-Schmidt QR Decomposition. In *Int'l Conference on High-Performance Computing and Networking, HPCN'96*, Springer-Verlag LNCS no. 1067, pages 646-653. Brussels, Apr 1996.

[2] R. Doallo, J. Touriño, E.L. Zapata. Sparse Householder QR Factorization on a Mesh. In *Fourth Euromicro Workshop on Parallel and Distributed Processing, PDP'96*, IEEE Computer Society Press, pages 33-39. Braga, Portugal, Jan 1996.

[3] I.S. Duff, A.M. Erisman, J.K. Reid. *Direct Methods for Sparse Matrices.* Clarendon Press, 1986.

[4] I.S. Duff, R.G. Grimes, J.G. Lewis. *User's Guide for the Harwell-Boeing Sparse Matrix Collection.* Technical Report TR-PA-92-96. CERFACS, Oct 1992.

[5] G.H. Golub, C.F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, second edition, 1989.

[6] J.B. Haag, D.S. Watkins. QR-Like Algorithms for the Nonsymmetric Eigenvalue Problem. *ACM Transactions on Mathematical Software*, 19(3):407-418, Sep 1993.

[7] H. Ishihata, T. Horie, T. Shimizu. Architecture for the AP1000 Highly Parallel Computer. *Fujitsu Sci. Tech. Journal*, 29(1):6-14, Mar 1993.

[8] L. Kaufman. A Parallel QR algorithm for the Symmetric Tridiagonal Eigenvalue Problem. *Journal of Parallel and Distributed Computing*, 23:429-434, 1994.

[9] B. Kumar, P. Sadayappan, C.H. Huang. On Sparse Matrix Reordering for Parallel Factorization. In *Int'l Conference on Supercomputing, ICS'94*, ACM Press, pages 431-438. Manchester, Jul 1994.

[10] P. Matstoms. Sparse QR Factorization in Matlab. *ACM Transactions on Mathematical Software*, 20(1):136-159, Mar 1994.

[11] P. Matstoms. Sparse QR Factorization with Applications to Linear Least Squares Problems. *Linköping Studies in Science and Technology. Dissertations*, no.337, Dep. of Mathematics, Linköping University. Sweden, 1994.

[12] Tz. Ostromsky, P.C. Hansen, Z. Zlatlev. A Parallel Sparse QR-Factorization Algorithm. In *Int'l Linear Algebra Year, Workshop on Direct Methods*, ftp.cerfacs.fr, /pub/algo/workshops/ILAY/Direct. CERFACS, 1995.

[13] J. Touriño, R. Doallo. *Parallel Sparse Orthogonalization on a MIMD Distributed-Memory Computer.* Technical Report, Dep. of Electronics and Systems, University of La Coruña. Spain, 1996.

[14] J. Touriño, R. Doallo, E.L. Zapata. Sparse Givens QR Factorization on a Multiprocessor. In *Second Int'l Conference on Massively Parallel Computing Systems, MPCS'96*, pages 551-556. Ischia, Italy, May 1996.