

NUMERICAL ALGORITHMS
FOR ESTIMATING
LEAST SQUARES PROBLEMS

Petko Ivanov Yanev

Ph.D. Thesis

Institut d'informatique
Université de Neuchâtel
Switzerland

2005

IMPRIMATUR POUR LA THESE

**NUMERICAL ALGORITHMS FOR ESTIMATING
LEAST SQUARES PROBLEMS**

Petko Ivanov YANEV

UNIVERSITE DE NEUCHATEL

FACULTE DES SCIENCES

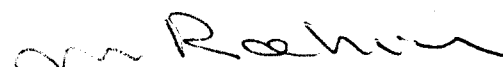
La Faculté des sciences de l'Université de
Neuchâtel, sur le rapport des membres du jury

MM. P. Kropf (directeur de thèse),
E. Kontoghiorghes (co-directeur de thèse),
P. Felber, M. Gill (Genève) et
M. Clint (Belfast)

autorise l'impression de la présente thèse.

Neuchâtel, le 12 avril 2005

La doyenne:



Prof. M. Rahier

To Desi

Acknowledgments

First, I would like to thank my Ph.D. supervisor and friend Prof. Erricos John Kontoghiorghes for his guidance, help and support during these years of Ph.D. study. He introduced me to the world of research and taught me to express my ideas and the results I obtained from the research in a scientific manner. I am also grateful to Prof. Dimitar Mekerov (FMI, Plovdiv University), who encouraged and helped me to start this Ph.D. and always supported me during my studies.

I extend my sincere thanks to the Department of Computer Science, University of Neuchatel for hosting me and providing me with an excellent working environment. I would like to acknowledge and thank as well the Swiss National Science Foundation, which financially supported this research through the projects 2000-061875.00 and 200020-100116.

I would like to thank many of my friends and colleagues, in particular Dr. Paolo Foschi, Dr. Cristian Gatu and Marc Hofmann for being always ready to discuss and help me with all the problems I encountered and for their encouragement.

Last, but not least, I would like to thank my wife Desi, my brother Zlatan and my parents Ivan and Maria for their great support, love and understanding.

Abstract

The solution of least squares estimation problems is of great importance in the areas of numerical linear algebra, computational statistics and econometrics. The design and analysis of numerically stable and computationally efficient methods for solving such least squares problems is considered. The main computational tool used for the estimation of the least squares solutions is the QR decomposition, or the generalized QR decomposition. Specifically, emphasis is given to the design of sequential and parallel strategies for computing the main matrix factorizations which arise in the estimation procedures. The strategies are based on block-generalizations of the Givens sequences and efficiently exploit the structure of the matrices.

An efficient minimum spanning tree algorithm is proposed for computing the QR decomposition of a set of matrices which have common columns. Heuristic strategies are also considered. Several computationally efficient sequential algorithms for block downdating of the least squares solutions are designed, implemented and analyzed. A parallel algorithm based on the best sequential approach for downdating the QR decomposition is also proposed. Within the context of block up-downdating, efficient serial and parallel algorithms for computing the estimators of the general linear and seemingly unrelated regression models after been updated with new observations are proposed. The algorithms are based on orthogonal factorizations and are rich in BLAS-3 computations. Experimental results which support the theoretical derived complexities of the new algorithms are presented. The comparison of the new algorithms with the corresponding LAPACK routines is also performed. The parallel algorithms utilize efficient load balanced distribution over the processors and are found to be scalable and efficient for large-scale least squares problems.

It is expected that the proposed block-algorithms will facilitate the solution of computationally intensive statistical problems and the estimation of large scale linear models on serial and parallel computers.

Contents

1	Introduction	1
1.1	Least Squares and QR Decomposition	1
1.2	Computing the QR decomposition	3
1.3	Structure of the thesis	7
2	QRD of a Set of Matrices with Common Columns	9
2.1	Introduction	10
2.2	Computing the QR decomposition of RS_i	11
2.3	The Minimum Spanning Tree algorithm	13
2.4	Numerical results	18
2.5	Conclusion	20
3	Block Downdating of LS solutions	23
3.1	Introduction	24
3.2	Block-downdating of the LS solutions	25
3.3	Strategies for computing the factorization (3.12)	29
3.4	Numerical results and conclusion	35
4	Parallel downdating of the QRD	39
4.1	Introduction	39
4.2	Block downdating of the QRD	41
4.3	Parallel downdating of the QRD	43
4.4	Conclusions	49
5	Estimating the general linear model	51
5.1	Introduction	52

5.2	Serial block Givens strategy	53
5.3	Parallel algorithm	57
5.4	Conclusion	60
6	Estimating the updated-observations SUR models	63
6.1	Introduction	64
6.2	Updating the SUR model with new observations	66
6.3	Sequential strategies for computing the UGQRD (6.14)	68
6.4	Parallel algorithm for computing the UGQRD	72
6.5	Conclusions	77
7	Conclusions and future research	79
7.1	Future research	81
	Bibliography	83

List of Figures

1.1	Column-based Givens sequence for computing the QRD of $A \in \mathbb{R}^{4 \times 3}$	5
2.1	Computing the QRD of $A \in \mathbb{R}^{12 \times 8}$ using Givens rotations.	12
2.2	Computing the QRD of RS_i , where $R \in \mathbb{R}^{12 \times 12}$, $k_i=6$ and $\lambda_i = (1, 2, 5, 6, 10, 12)$	13
2.3	The Graph $\Gamma(V, E, n)$ with all and reduced number of edges, where $ V = 9$ and $n = 6$	15
2.4	The Graph $\Gamma(V, E, n)$ with the artificial nodes \tilde{R}_9 and \tilde{R}_{10} , where $ V = 10$, $ E = 9$ and $n = 6$	16
3.1	The orthogonal $C^{(j)}$, ($j = 1, \dots, 4$).	32
3.2	The orthogonal $\tilde{C}^{(j)}$, ($j = 1, \dots, 4$).	34
4.1	The cyclic distribution of the matrices on 4 processors, when $g = 16$	44
4.2	The modified cyclic distribution of the matrices on 4 processors, when $g = 16$	47
5.1	The row-block cyclic distribution of the matrices on 4 processors, when $k = 18$	58
6.1	Computing the UGQRD (6.14), where $G = 3$ and $T^{(s)} = 2$	70
6.2	Theoretical complexity of the i th step ($i = 1, \dots, G$) of the UQRD (6.16), where $G = 80$, $k = 40$, $v = 20$ and $T^{(s)} = 50$	74
6.3	Distribution scheme for 4 processors, where $k_i = k$ ($i = 1, \dots, G$) and $G = 8$	75

List of Tables

2.1	Theoretical complexity and execution time of the modified heuristic method and that of re-triangularizing the G matrices one at a time, where the total number of distinct columns of all matrices is n	19
3.1	Theoretical complexities (Mflops) and execution times (sec).	36
3.2	Execution times of the downdating methods.	37
4.1	Execution times, communication times and efficiencies of Algorithm 7. . . .	46
4.2	Execution times (sec.) and efficiencies of Algorithm 8.	49
5.1	Execution times (sec.) and theoretical results of Algorithm 9 and LAPACK. . . .	57
5.2	Execution times (sec.) and efficiency of Algorithm 10.	60
6.1	Execution times and theoretical complexities of Algorithm 11, $k_i = k$, $i = 1, \dots, G$ and $v = 20$	72
6.2	Execution times and theoretical complexities of Algorithm 11, $v = 20$ and $G = 5$	73
6.3	Execution times (sec.) and efficiencies of Algorithm 12 for $k_i = k$ ($i = 1, \dots, G$) and $G = 32$	77
6.4	Execution times (sec.) and efficiencies of Algorithm 12 for $k_i = k$ ($i = 1, \dots, G$) and $T^{(s)} = 50$	78
6.5	Execution times (sec.) and efficiencies of Algorithm 12 for $k_i = ik$ ($i = 1, \dots, G$).	78

List of Algorithms

1	The optimal MST algorithm.	17
2	The heuristic MST algorithm.	18
3	Downdating the least squares problem (3.3).	28
4	The third Strategy for computing (3.12).	32
5	The computation of factorization (3.27).	35
6	The sequential block downdating of the QRD.	43
7	The parallel block downdating of the QRD with cyclic distribution on p processors.	45
8	The modified parallel block downdating of the QRD with cyclic distribution.	48
9	The sequential block Givens algorithm for solving the GLLSP (5.2).	56
10	The parallel algorithm for solving the GLLSP (5.2) on p processors.	59
11	Sequential algorithm for solving the UGQRD (6.14).	70
12	The parallel algorithm for solving the UGQRD (6.14) on p processors.	76

Chapter 1

Introduction

The linear Least Squares (LS) problem is a computational problem of great importance, which arises in various applications, such as statistics, econometrics, optimization and signal processing to name but a few. Efficient methods for solving LS problems have been developed during the last fifty years of active research [5, 21, 22, 32]. The orthogonal factorizations are very important in the LS computations due to the property that the orthogonal matrices preserve the Euclidean norm of a vector after multiplication. Specifically, the QR decomposition (QRD) is often associated with the solution of LS problems. The QRD is one of the main computational tools in numerical linear algebra [5, 20, 32]. It provides more accurate solution than the LU and other similar decompositions and involves less computations than the singular value decomposition.

In this thesis sequential and parallel algorithms employing Givens rotations and Householder transformations for computing various LS and QRD-related problems are proposed. The algorithms, in general, utilize block-generalizations of the sequential and parallel Givens sequences. They exploit efficiently the structure of the computed matrices and are rich in BLAS-3 operations.

1.1 Least Squares and QR Decomposition

The LS problem can be formulated as

$$\hat{x} = \underset{x}{\operatorname{argmin}} \|Ax - y\|_2, \quad (1.1)$$

where $A \in \mathbb{R}^{m \times n}$ is the data matrix of full column rank, $y \in \mathbb{R}^m$ is the response vector, $x \in \mathbb{R}^n$ is the unknown vector and $\|\cdot\|$ denotes the Euclidean norm. Let the QRD of the augmented $(n+1) \times (n+1)$ matrix $\tilde{A} \equiv (A \ y)$ be given by

$$Q^T \tilde{A} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} \equiv \begin{pmatrix} n & 1 \\ R & u \\ 0 & s \\ 0 & 0 \end{pmatrix} \begin{matrix} n \\ 1 \\ m-n-1 \end{matrix}, \quad (1.2)$$

where $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{n \times n}$ is upper triangular and non-singular. The LS estimator of x in (1.1) is computed by solving the triangular system

$$R\hat{x} = u. \quad (1.3)$$

The Generalized Linear Least Squares Problem (GLLSP) has the form

$$\operatorname{argmin}_{u,x} u^T u \quad \text{subject to} \quad y = Ax + Cu \quad (1.4)$$

and is often associated with the computation of the best linear unbiased estimator (BLUE) of the General Linear Model (GLM)

$$y = Ax + \varepsilon, \quad \varepsilon \sim (0, \sigma^2 \Omega). \quad (1.5)$$

Here $y \in \mathbb{R}^m$ is the response vector, $A \in \mathbb{R}^{m \times n}$ is the full rank exogenous data matrix, $x \in \mathbb{R}^n$ are the coefficients to be estimated and $\varepsilon \in \mathbb{R}^m$ is the noise with zero mean and variance-covariance matrix $\sigma^2 \Omega$. Without loss of generality it is assumed that $\Omega = CC^T$ is non-singular. The random vector u is defined by $Cu = \varepsilon$, i.e. $u \sim (0, \sigma^2 I_m)$.

The solution of the GLLSP (1.4) requires the computation of the generalized QR decomposition (GQRD) of the augmented matrix $\tilde{A} \equiv (A \ y)$ and C [35]. That is, computing the QRD of \tilde{A} in (1.2) and the RQ decomposition

$$(Q^T C)\Pi = U = \begin{pmatrix} n & 1 & m-n-1 \\ U_{1,1} & r & U_{1,2} \\ 0 & \delta & g \\ 0 & 0 & U_{2,2} \end{pmatrix} \begin{matrix} n \\ 1 \\ m-n-1 \end{matrix}. \quad (1.6)$$

Here \tilde{R} and U are upper triangular matrices of order n and m , respectively, and $Q, \Pi \in \mathbb{R}^{m \times m}$ are orthogonal [5, 20]. Pre-multiplying both sides of the constraints in the GLLSP (1.4) with the orthogonal matrix Q^T , the problem becomes equivalent to

$$\operatorname{argmin}_{u,x} \|\Pi^T u\|^2 \quad \text{subject to} \quad Q^T y = Q^T Ax + Q^T C \Pi \Pi^T u, \quad (1.7)$$

which after the factorization (1.6) can be written as

$$\operatorname{argmin}_{v_1, v, v_2, x} (\|v_1\|^2 + v^2 + \|v_2\|^2) \quad \text{subject to} \quad \begin{cases} \tilde{y} = Rx + U_{1,1}v_1 + rv + U_{1,2}v_2, \\ \eta = \delta v + gv_2, \\ 0 = U_{2,2}v_2. \end{cases} \quad (1.8)$$

Note that, here the vector $u^T \Pi$ is partitioned as $(v_1^T \ v \ v_2^T)$. From the last constraint it follows that $v_2 = 0$ and consequently, the second constraint can be used to derive $v = \eta/\delta$. Thus, the GLLSP (1.8) is reduced to

$$\operatorname{argmin}_{v_1, x} \|v_1\|^2 \quad \text{subject to} \quad \tilde{y} = Rx + U_{1,1}v_1 + rv. \quad (1.9)$$

In order to minimize the objective function of (1.9) the vector v_1 is set to zero. Finally, the BLUE of the GLM (1.5) is obtained by solving the triangular system

$$R\hat{x} = \tilde{y} - \eta r/\delta. \quad (1.10)$$

1.2 Computing the QR decomposition

The QRD of a matrix $A \in \mathbb{R}^{m \times n}$ ($m > n$) is given by

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}, \quad (1.11)$$

where $Q \in \mathbb{R}^{m \times m}$ is orthogonal, i.e. it satisfies $Q^T Q = Q Q^T = I_m$, and $R \in \mathbb{R}^{n \times n}$ is upper triangular. The matrix A is assumed to be of full-column rank. The two most significant methods for forming the QRD are known as Householder transformations and Givens rotation methods.

A Householder matrix (or Householder transformation) is a symmetric orthogonal $m \times m$ matrix of the form

$$H = I_m - 2 \frac{hh^T}{\|h\|^2},$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$ for some θ , that is $c^2 + s^2 = 1$ [20]. The transformation $G_{i,j}^{(k)}$ when applied to the left of a matrix annihilates the k th element in the j th row. The Givens rotations are important, because they can annihilate the elements of a matrix or a vector more selectively than the Householder transformations and affects only the i th and the j th rows during the computations. Specifically, for $A \in \mathbb{R}^{m \times n}$, the Givens transformation $\tilde{A} = G_{i,j}^{(k)} A$ results in the element $\tilde{a}_{j,k}$ being zero and modifies the p th ($p = i, j$) row of A as follows:

$$\tilde{A}_{p,:} = \begin{cases} cA_{i,:} + sA_{j,:}, & \text{if } p = i, \\ -sA_{i,:} + cA_{j,:}, & \text{if } p = j, \\ A_{p,:}, & \text{otherwise.} \end{cases}$$

Note that, in order for $a_{j,k}$ to become zero, the values of c and s are given, respectively, by $c = a_{i,k}/t$ and $s = a_{j,k}/t$, where $t^2 = a_{i,k}^2 + a_{j,k}^2$.

Consider the computation of the QRD (1.11) as a sequence of $n(2m - n - 1)/2$ Givens rotations. The orthogonal matrix Q is defined as the product of the Givens matrices

$$Q = (G_{m-1,m}^{(1)} \cdots G_{1,2}^{(1)})(G_{m-1,m}^{(2)} \cdots G_{2,3}^{(2)}) \cdots (G_{m-1,m}^{(n)} \cdots G_{n,n+1}^{(n)}).$$

This sequence is referred as column-based, because it annihilates the elements below the main diagonal from bottom to top starting with the first column. Figure 1.1 illustrates the column-base annihilation Givens sequence, where $m = 4$ and $n = 3$.

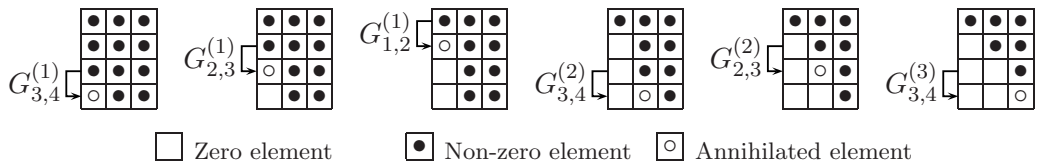


Figure 1.1: Column-based Givens sequence for computing the QRD of $A \in \mathbb{R}^{4 \times 3}$.

A block generalization of the Householder and Givens sequences can be used for developing efficient sequential and parallel algorithms which are rich in BLAS-3 operations. Instead of a sequence of Givens rotation, a single block Givens transformation can be used

1.3 Structure of the thesis

Each chapter of the thesis is self contained¹. Chapter 2 investigates algorithms for computing the QRD of a set of matrices which have common columns. The problem is tackled using a directed weighted graph to express the relationship (the common columns) among the matrices. Each node in the graph represent the triangular factor derived from the QRD of a matrix from the set. An edge between two nodes exists if and only if the columns of one of the matrices is a subset of the columns of the other. The weight of an edge is the computational cost of deriving the triangular factor of the subset matrix given the triangular factor of the larger matrix. The problem is shown to be equivalent to finding the minimum spanning tree (MST) of the graph. An algorithm for computing the QRDs using the MST is proposed. Alternative heuristic strategies are also considered. The theoretical and numerical results are given.

Chapter 3 presents five computationally efficient algorithms for block downdating of the least squares problem. The algorithms are block generalization strategies of the Givens sequences. They exploit the specific structure of the matrices and are rich in BLAS-3 computations. The theoretical complexities and execution times of the algorithms are derived. The experimental results confirm the theoretical ones.

Chapter 4 considers parallel algorithms for downdating the QRD. An efficient algorithm, which is a block version of sequential Givens strategy, is proposed. The algorithm takes advantage of the structure of the computed matrices. Furthermore, it utilizes an efficient load-balanced distribution of the matrices over the processors which does not require inter-processor communication. The presented theoretical and experimental results shows that the parallel strategy is scalable and efficient for large-scale downdating problems.

In Chapter 5 efficient sequential and parallel algorithms for estimating the general linear model (GLM) are proposed. The algorithms use the GQRD as a main computational tool and exploit efficiently the triangular structure of the Cholesky factor and the variance-covariance matrix of the model. The sequential block algorithm is an adaptation of a known Givens strategy. Specifically, the GLM is solved recursively, computing a series of smaller and smaller GLLSPs. The parallel version of the serial algorithm proposes an efficient distribution of the matrices over the processors which requires low inter-processor communication. The theoretical complexities of both, the sequential and the parallel algorithm are derived and analyzed. Experimental results are presented which confirm the theoretical

¹Each chapter has been published, or is submitted for publication in a refereed international journal.

analysis. The parallel algorithm is scalable and efficient for computing large-scale general linear estimation problems.

Chapter 6 presents computationally efficient sequential and parallel algorithm for estimating the seemingly unrelated regressions model after been updated with new observations. The proposed algorithms are based on orthogonal transformations and are rich on BLAS-3 computations. The structure of the computed matrices is efficiently exploited. A load-balanced distribution is proposed for the parallel algorithm which leads to low inter-processor communications. Theoretical and experimental results are presented and analyzed. The results show the efficiency and scalability of the parallel algorithm. The last chapter concludes the thesis and provides future research directions.

Chapter 2

Algorithms for computing the QR decomposition of a set of matrices with common columns

Abstract:

The QR decomposition of a set of matrices which have common columns is investigated. The triangular factors of the QR decompositions are represented as nodes of a weighted directed graph. An edge between two nodes exist if and only if the columns of one of the matrices is a subset of the columns of the other. The weight of an edge denotes the computational complexity of deriving the triangular factor of the destination node from that of the source node. The problem is equivalent to construct the graph and find the minimum cost for visiting all the nodes. An algorithm which computes the QR decompositions by deriving the minimum spanning tree of the graph is proposed. Theoretical measures of complexity are derived and numerical results from the implementation of this and alternative heuristic algorithms are given.

¹This chapter is a reprint of the paper: P. Yanev, P. Foschi and E.J. Kontoghiorghes. Algorithms for computing the QR decomposition of a set of matrices with common columns. *Algorithmica*, 39:83–93, 2004.

2.1 Introduction

Computationally intensive methods for deriving the least-squares estimators of seemingly unrelated regression and simultaneous equation models have been proposed [24]. These estimation methods require the QR decompositions of a set of matrices which have common columns. These columns correspond to exogenous factors that occur in more than one econometric relationship of the model. Consider the QR decomposition (QRD) of the full column rank matrix $A_i \in \mathbb{R}^{m \times k_i}$:

$$Q_i^T A_i = \begin{pmatrix} R_i \\ 0 \end{pmatrix} \begin{matrix} k_i \\ m - k_i \end{matrix}, \quad (i = 1, \dots, G), \quad (2.1)$$

where $Q_i \in \mathbb{R}^{m \times m}$ is orthogonal and $R_i \in \mathbb{R}^{k_i \times k_i}$ is upper triangular. The exogenous matrices with common columns can be expressed as

$$A_i = A S_i, \quad (i = 1, \dots, G), \quad (2.2)$$

where $A \in \mathbb{R}^{m \times n}$ and $S_i \in \mathbb{R}^{n \times k_i}$ is a selection matrix [13, 24, 30]. It is often the case that $n \ll \sum_{i=1}^G k_i$, i.e. the number of distinct factors is much less than the total number of factors occurring in the whole model.

The main method used to compute (2.1) is by forming the QRDs of A_1, \dots, A_G one at a time, without taking into account that the matrices may share common columns. Let the QRD of A be given by

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} n \\ m - n \end{matrix}, \quad (2.3)$$

where $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{n \times n}$ is upper triangular. Thus, the upper triangular factor R_i in (2.1) can be derived by computing the QRD

$$\tilde{Q}_i^T R S_i = \begin{pmatrix} R_i \\ 0 \end{pmatrix} \begin{matrix} k_i \\ n - k_i \end{matrix}, \quad (i = 1, \dots, G), \quad (2.4)$$

where $\tilde{Q}_i \in \mathbb{R}^{n \times n}$ is orthogonal [19, 20, 23]. The orthogonal matrix Q_i in (2.1) is defined by

$$Q_i = Q \begin{pmatrix} \tilde{Q}_i & 0 \\ 0 & I_{m-n} \end{pmatrix}. \quad (2.5)$$

Notice that the QRDs in (2.4) are equivalent to the re-triangularization of a set of upper-triangular matrices after deleting columns.

Sequential and parallel strategies which compute the QRD of RS_i have been proposed [23, 24, 30]. These strategies use Givens rotations and exploit the non-full structure of RS_i . However, the occurrence of common columns among RS_1, \dots, RS_G has not been exploited. The purpose of this work is to propose and investigate sequential factorization strategies that take advantage of this possibility when $n \ll \sum_{i=1}^G k_i$. The algorithms are based on Givens rotations [20].

A Givens rotation in plane (i, j) that reduces to zero the element $b_{j,k}$ when it is applied from the left of $B = [b_{i,j}] \in \mathbb{R}^{m \times n}$ will be denoted by $G_{i,j}^{(k)}$, where $1 \leq i, j \leq m$ and $1 \leq k \leq n$. The rotation $G_{i,j}^{(k)} B$ affects only the i th and j th rows of B . The changes in these rows can be written as

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} b_{i,:} \\ b_{j,:} \end{pmatrix} = \begin{pmatrix} \tilde{b}_{i,1} & \dots & \tilde{b}_{i,k} & \dots & \tilde{b}_{i,n} \\ \tilde{b}_{j,1} & \dots & \tilde{b}_{j,k} & \dots & \tilde{b}_{j,n} \end{pmatrix}, \quad (2.6)$$

where $b_{j,k} \neq 0$, $c^2 + s^2 = 1$, $c = b_{i,k}/\tau$, $s = b_{j,k}/\tau$, $\tau^2 = b_{i,k}^2 + b_{j,k}^2$, $\tilde{b}_{i,k} = \tau$ and $\tilde{b}_{j,k} = 0$. If $b_{j,k} = 0$, then $G_{i,j}^{(k)} \equiv I_m$. Standard column notation is used to denote sub-vectors and sub-matrices [20]. The construction of a Givens rotation requires 6 flops denoted by t . The same time is required to apply the rotation to a 2-element vector. Thus, nt flops are needed to compute (2.6). Notice that the rotation is not applied to the pair of elements $b_{i,k}$ and $b_{j,k}$ used in the construction of the rotation. These are set to τ and zero, respectively.

In the next section Givens' sequences for computing the QRD of RS_i ($i = 1, \dots, G$) are presented. Section 3 proposes an efficient algorithm for computing the QRDs of RS_1, \dots, RS_G , which are represented as nodes of a directed graph. Within this context the Minimum Spanning Tree (MST) terminology is used. That is, the problem of deriving the MST in a graph is equivalent to that of finding the tree which consists of the shortest paths for visiting all nodes, starting from the root node. Numerical results are presented and the performance of the algorithm is evaluated. In section 4 conclusions are offered.

2.2 Computing the QR decomposition of RS_i

There are many equivalent strategies for computing the QR decomposition using Givens rotations [20]. Consider the case where the elements of a matrix below the main diagonal are annihilated column-by-column and from bottom to the top with zero elements being preserved throughout the annihilation process. Furthermore, let the Givens rotations be between adjacent planes. The number of Givens rotations required to compute (2.3) is

given by $\sum_{i=1}^n (m - i) = n(2m - n - 1)/2$ and Q^T is defined by

$$Q^T = \prod_{i=1}^n \prod_{j=1}^{m-i} G_{m-j, m-j+1}^{(i)} . \tag{2.7}$$

●	●	●	●	●	●	●	●
11	●	●	●	●	●	●	●
10	21	●	●	●	●	●	●
9	20	30	●	●	●	●	●
8	19	29	38	●	●	●	●
7	18	28	37	45	●	●	●
6	17	27	36	44	51	●	●
5	16	26	35	43	50	56	●
4	15	25	34	42	49	55	60
3	14	24	33	41	48	54	59
2	13	23	32	40	47	53	58
1	12	22	31	39	46	52	57

Figure 2.1: Computing the QRD of $A \in \mathbb{R}^{12 \times 8}$ using Givens rotations.

Figure 2.1 shows the annihilation pattern corresponding to this Givens' sequence, where $m = 12$ and $n = 8$. An entry i ($i = 1, \dots, 60$) indicates that the element is reduced to zero by the i th rotation. The complexity of computing the QRD (2.3) using this strategy is given by

$$\begin{aligned} C(m, n) &= t \sum_{i=1}^n (m - i)(n - i + 1) \\ &= tn(3m(n + 1) - n^2 - 3n - 2)/6 . \end{aligned} \tag{2.8}$$

Thus, the complexity of computing the QRDs of A_1, \dots, A_G simultaneously is given by

$$T_1(m, k, G) = \sum_{i=1}^G C(m, k_i) , \tag{2.9}$$

where $k = (k_1, \dots, k_G)$.

Let S_i in (2.2) be expressed as $S_i \equiv (e_{\lambda_{i,1}} \dots e_{\lambda_{i,k_i}})$ with $\lambda_i = (\lambda_{i,1}, \dots, \lambda_{i,k_i})$, where $e_{\lambda_{i,j}}$ is the $\lambda_{i,j}$ th column of the unit matrix I_n , $i = 1, \dots, G$ and $j = 1, \dots, k_i$ [23, 24, 30]. Then, the number of Givens rotations needed to compute the QRD (2.4) is given by $\sum_{j=1}^{k_i} (\lambda_{i,j} - j)$

and the orthogonal matrix \tilde{Q}_i^T is defined as:

$$\tilde{Q}_i^T = \prod_{n=1}^{k_i} \prod_{j=1}^{\lambda_{i,n}-n} G_{\lambda_{i,n}-j, \lambda_{i,n}-j+1}^{(n)}. \quad (2.10)$$

Figure 2.2 shows the Givens' sequence when re-triangularizing RS_i , where $n = 12$, $k_i = 6$ and $\lambda_i = (1, 2, 5, 6, 10, 12)$.

●	●	●	●	●	●
	●	●	●	●	●
		●	●	●	●
		2	●	●	●
		1	4	●	●
			3	9	●
				8	15
				7	14
				6	13
				5	12
					11
					10

Figure 2.2: Computing the QRD of RS_i , where $R \in \mathbb{R}^{12 \times 12}$, $k_i=6$ and $\lambda_i = (1, 2, 5, 6, 10, 12)$.

The complexity of computing the QRD (2.4) is given by

$$C_i(\lambda_i, k_i) = t \sum_{j=1}^{k_i} (\lambda_{i,j} - j)(k_i - j + 1). \quad (2.11)$$

Thus, the total complexity of computing (2.3) followed by re-triangularization of RS_1, \dots, RS_G one at a time is given by

$$T_2(\lambda_i, k_i, G) = C(m, n) + \sum_{i=1}^G C_i(\lambda_i, k_i). \quad (2.12)$$

2.3 The Minimum Spanning Tree algorithm

The triangular factors R, R_1, \dots, R_G can be represented as nodes N_0, N_1, \dots, N_G of a weighted directed graph. An edge between two nodes N_i and N_j (denoted by $E_{i,j}$) exists and is directed from N_i towards N_j , if and only if R_i contains all the columns of R_j

($i, j = 0, 1, \dots, G$ and $i \neq j$). The weight of $E_{i,j}$ is denoted by $C_{i,j}$, the complexity of computing R_j given R_i . The goal is to construct the graph and to find Minimum Spanning Tree of the graph which provides the minimum computational cost for deriving R_1, \dots, R_G [31, 40].

To determine the MST the properties of the graph need to be explored. Let $\Gamma(V, E, n)$ be a graph with the sets of nodes and edges denoted by V and E , respectively, and n denotes the number of columns of the matrix R . The graph $\Gamma(V, E, n)$ can be divided into n levels L_1, \dots, L_n . The matrices with k columns belong to the level L_k ($k = 1, \dots, n$). Notice that R belongs to level L_n and level L_{n-1} can have at most n nodes (matrices). In general, there are at most $C_k^n = n!/k!(n-k)!$ nodes in the level L_k ($k = 1, \dots, n$). Therefore, the maximum number of nodes in $\Gamma(V, E, n)$ is

$$|V|_{max} = \sum_{i=0}^{n-1} C_i^n = 2^n - 1 . \quad (2.13)$$

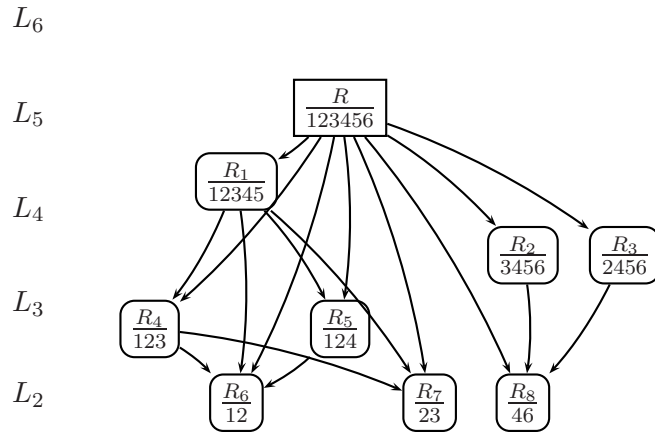
Now, from the k th level exists a maximum of $C_k^n(2^{(n-k)} - 2)$ edges. Thus, the maximum number of edges in the graph is

$$\begin{aligned} |E|_{max} &= \sum_{i=0}^{n-2} C_i^n (2^{(n-i)} - 2) \\ &= 3^n - 2^{(n+1)} + 1 . \end{aligned} \quad (2.14)$$

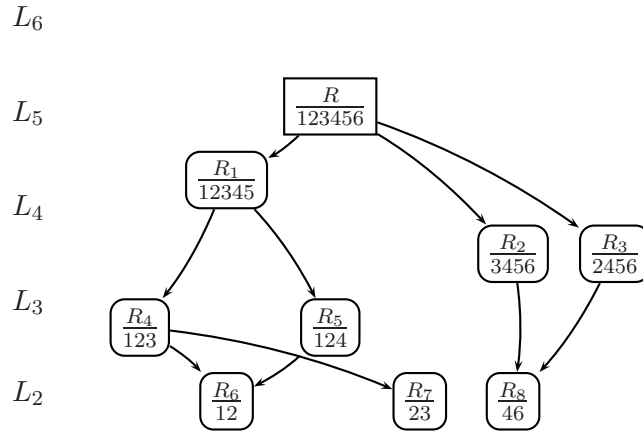
Let $E_{i,j}$ exist and let $p_{i,j}$ denote the position of the j th column of R_j in R_i . Notice that $p_{i,j} \geq j$ for every j . Then, the cost of the edge $C_{i,j}$ is given by

$$C_{i,j} = t \sum_{j=1}^{k_j} (p_{i,j} - j)(k_j - j + 1) . \quad (2.15)$$

Now, let $R_s \in L_p$, $R_h \in L_q$ and $R_i \in L_r$, where $E_{s,i}$ and $E_{h,i}$ exist, and $p \neq q \neq r$. If there is a path from R_s to R_h , then $C_{h,i} \leq C_{s,i}$. Therefore, $E_{s,i}$ can be deleted from the graph. A path from R_s to R_h exists if and only if the node R_h can be reached from the node R_s . When this rule is applied the number of the edges to be computed is reduced. Figures 2.3(a) and 2.3(b) illustrate the graph $\Gamma(V, E, 6)$ with all and with the reduced number of edges, respectively. The matrices R and R_i ($i = 1, \dots, G$) are denoted by square and round frames, respectively. The indexes of columns for each matrix are shown by a sequence of digits in the frames.



(a) The Graph $\Gamma(V, E, 6)$, where $|E| = 17$.



(b) The Graph $\Gamma(V, E, 6)$, where $|E| = 10$.

Figure 2.3: The Graph $\Gamma(V, E, n)$ with all and reduced number of edges, where $|V| = 9$ and $n = 6$.

In order to determine the MST, the cost $C_{i,j}$ of each edge is computed and, for each node, the incoming edge with minimum cost is selected. If more than one incoming edge with equal weights exist, then one of them is selected randomly. The correctness of this algorithm follows from the acyclic property of $\Gamma(V, E, n)$ [1]. The time required to derive $C_{i,j}$ depends on the time to compute $p_{i,1}, \dots, p_{i,k_j}$ and calculate the summation (2.15). At most k_i comparisons are necessary to determine $p_{i,1}, \dots, p_{i,k_j}$. A single comparison and the summation of (2.15) requires one and $5k_j$ flops, respectively. The total time needed to

compute $C_{i,j}$ is $k_i + 5k_j \leq 6k_i \equiv k_it$. Let

$$T_{\text{EDGE}} = \max_{i=1,\dots,G} (k_it) \tag{2.16}$$

and the upper bound of the time needed for deriving the MST of a graph with $|E|$ nodes be given by

$$T_{\text{MST}} \leq |E|T_{\text{EDGE}} + |E|. \tag{2.17}$$

Here $|E|T_{\text{EDGE}}$ is the maximum time needed to compute the costs of all edges and $|E|$ is the maximum number of comparisons that could be done. Then, the complexity of computing the matrices R_1, \dots, R_G using the MST approach is:

$$T_3(k_i, m, n, p_i, G) = C(m, n) + \sum_{i=1}^G \sum_{j=1}^{k_i} (p_{i,j} - j)(k_i - j + 1) + T_{\text{MST}}, \tag{2.18}$$

where $C(m, n)$ is given by (2.8) and corresponds to the complexity of computing the QRD (2.3) and $p_i = (p_{i,1} \cdots p_{i,k_i})$.

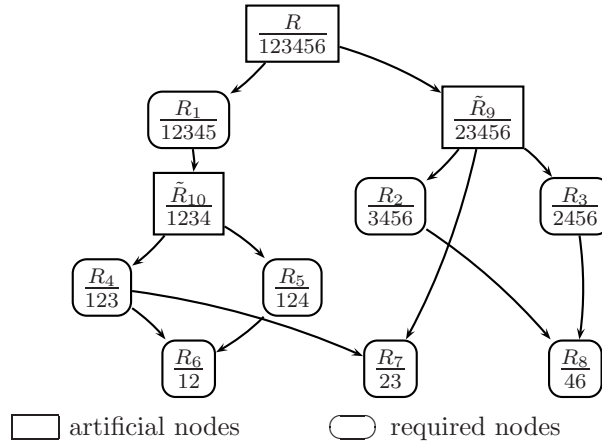


Figure 2.4: The Graph $\Gamma(V, E, n)$ with the artificial nodes \tilde{R}_9 and \tilde{R}_{10} , where $|V| = 10$, $|E| = 9$ and $n = 6$.

The MST approach reduces the complexity in the specific case where the columns of some of the matrices are subsets of the columns of other matrices. In order to exploit the possibility of common columns occurring in R_1, \dots, R_G new nodes (hereafter called artificial nodes) are added in the graph $\Gamma(V, E, n)$. An artificial node is the conjunction of the columns of two or more matrices. The QRD of these matrices might be more quickly

derivable given the QRD of the artificial node. Figure 3 illustrates the graph $\Gamma(V, E, 6)$, where the two artificial nodes \tilde{R}_9 and \tilde{R}_{10} are denoted by square frames. Thus, the problem becomes one of finding the optimal tree which covers R_1, \dots, R_G in the graph that includes all artificial nodes. Algorithm 1 computes this optimal tree.

Algorithm 1 The optimal MST algorithm.

- 1: Construct the full graph consisting of R_1, \dots, R_G and all possible artificial nodes.
 - 2: Find all the edges and their corresponding weights.
 - 3: For all subgraphs which include R_1, \dots, R_G find the MST of each of them.
 - 4: Compute the complexity of each MST and choose the minimum one.
-

Now, let the full graph generated by the Algorithm 1 be denoted by $\Gamma_F(V_F, E_F, n)$, where $|V_F| = 2^G + 1$ and the maximum number of edges is given by $|E_F|_{\text{MAX}} = 2^G(2^G + 1)/2$. The number of all subgraphs which include the matrices R_1, \dots, R_G is $2^{(2^G - G)}$. Thus, an upper bound for the total complexity of this algorithm is

$$C(G) = 2^{(2^G - G)}T_{\text{MST}} + 2^G(2^G + 1)T_{\text{EDGE}}/2 + T_s, \quad (2.19)$$

where T_{EDGE} and T_{MST} are given by (2.16) and (2.17), respectively. The time to compute the complexities of each MST and to derive the minimum one is denoted by T_s .

Algorithm 1 implements the optimal strategy for computing R_1, \dots, R_G , given R . However, this optimal strategy has a double exponential complexity. Thus, it is not computationally feasible. To reduce the computational cost of Algorithm 1 a heuristic approach can be considered. The heuristic algorithm (Algorithm 2) computes the MST of the initial matrices R_1, \dots, R_G and then searches for artificial nodes which can reduce the weight of the tree. An artificial node is added to the MST if and only if it reduces the complexity between an existing node and its children. Then, a new MST is reconstructed and the procedure is repeated. A maximum of 2^G artificial nodes can be constructed from the G initial matrices. Each of these artificial nodes is evaluated to determine whether it should be included into the tree or not. Thus, the complexity of finding the MST using this heuristic approach is exponential $O(2^G)$ and computationally expensive.

Algorithm 2 can be modified to reduce its high complexity. The artificial nodes are constructed from the columns of those two child nodes which have the maximum number of common columns. In this way not all 2^G artificial nodes are considered. The total number

Algorithm 2 The heuristic MST algorithm.

- 1: Find the MST of R_1, \dots, R_G .
 - 2: **for** each node with more than one outgoing edge **do**
 - 3: Construct all artificial nodes from the columns of the child nodes.
 - 4: Compute the weights of the incoming and outgoing edges of the new artificial node.
 - 5: Add the new artificial node to the tree if it reduces the cost.
 - 6: Re-construct the MST until no more artificial nodes can be added.
 - 7: **end for**
-

of computed matrices is \bar{G} , where $\max(\bar{G}) = 2G$. Thus, the complexity of determining the MST is polynomial $O(kG^2)$. The total complexity of the modified heuristic method is

$$T_4(k_i, m, n, p_i, \bar{G}) = C(m, n) + \sum_{i=1}^{\bar{G}} \sum_{j=1}^{k_i} (p_{i,j} - j)(k_i - j + 1) + O(kG^4). \quad (2.20)$$

2.4 Numerical results

The modified heuristic approach is most efficient in the two cases, where there are many artificial nodes or none, but the columns of some matrices are subsets of the columns of others. The performance of the algorithms is considered in these two cases. First, when R_i is a sub-matrix of R_j for all $i = k, k + 1, \dots, G, j = 0, 1, \dots, G, (i \neq j)$, where $1 < k < G/2$ and the MST containing R_1, \dots, R_G can not be optimized. In this case no artificial nodes can be determined and the solution is optimal. Second, when the columns of none of the initial matrices R_1, \dots, R_G are subset of the columns of other initial matrix, but where they have most of their columns common. Here many artificial nodes can be determined, but the solution may not be the optimal. Tables 1(a) and 1(b) show the execution times of the modified heuristic method in these two cases, respectively. The performance of computing the QRDs (2.4) one at a time is also reported. Comparisons between the two methods are made also using their theoretical measures of complexity.

The constructed MST of each of the matrices in Table 1(a) is a binary tree. In this case no artificial nodes can be determined. Thus the MST strategy for factorizing the matrices R_1, \dots, R_G is optimal. Furthermore, the execution time of the *modified* heuristic algorithm is the same as that of Algorithm 2. In Table 1(b) the matrices R_1, \dots, R_G have a large number of common columns, but none of them is a sub-matrix of another matrix. In this

Table 2.1: Theoretical complexity and execution time of the modified heuristic method and that of re-triangularizing the G matrices one at a time, where the total number of distinct columns of all matrices is n .

		Execution times			Theoretical Complexity	
G	n	Retriang. method	Heuristic method	<u>Retriang.</u> Heuristic	<u>Retriang.</u> Heuristic	
14	1120	4.39	3.09	1.42	1.70	
14	2560	54.29	36.90	1.47	1.70	
14	2880	85.68	56.05	1.52	1.70	
14	3200	120.87	82.60	1.46	1.70	
30	1440	10.69	6.63	1.61	1.85	
30	2240	35.70	22.25	1.60	1.85	
30	2560	56.68	37.29	1.52	1.85	
30	3040	120.46	74.31	1.62	1.85	
62	1920	25.10	16.38	1.53	1.93	
62	2560	65.66	42.89	1.53	1.93	

(a) No artificial nodes exist in the graph. The MST is a binary tree which consists of exactly G matrices.

		Execution times			Theoretical Complexity	
G	n	Retriang. method	Heuristic method	<u>Retriang.</u> Heuristic	<u>Retriang.</u> Heuristic	
16	500	0.67	0.44	1.52	1.60	
16	1000	3.97	2.56	1.55	1.60	
16	1500	11.79	7.53	1.57	1.60	
16	2000	27.44	17.88	1.55	1.60	
28	1500	8.80	5.66	1.55	1.67	
28	2000	18.88	11.99	1.57	1.67	
28	2500	35.39	21.91	1.61	1.67	
28	3000	62.71	37.71	1.66	1.67	
40	2400	27.37	18.05	1.52	1.72	
40	3200	62.92	38.31	1.64	1.72	

(b) The MST consists of $3G/2$ nodes from which $G/2$ are artificial.

example $G/2$ artificial nodes are constructed. Thus, the MST consists of $3G/2$ nodes. An artificial node is constructed from two matrices if they have at least half of their columns in common. Notice that, in both cases, the heuristic method executes in less than $2/3$ of the time required by re-triangularization of R_1, \dots, R_G one at a time. The discrepancy between the theoretical and actual performance of the heuristic algorithm is due to the implementation overheads.

2.5 Conclusion

Strategies for computing the QR decomposition (QRD) of the set of matrices A_1, \dots, A_G which have common columns have been considered. The first strategy computes the QRD of each matrix independently and does not exploit the relationship that may exist among the matrices. The second strategy expresses the matrix A_i as AS_i , where A consists of all the distinct columns of A_1, \dots, A_G and S_i is a column-selection matrix. Initially it computes the triangular factor R of the QRD of A . Then it derives the QRD of A_i by re-triangularizing RS_i ($i = 1, \dots, G$). This re-triangularization is equivalent to the multiple-column downdating of the QRD [19, 24]. The second strategy is found to have better complexity than the first.

The remaining novel strategies use a weighted directed graph to express the relationship (common columns) among the matrices. The nodes represent the triangular factors R_1, \dots, R_G derived from the QRDs of A_1, \dots, A_G , respectively. An edge between two nodes exist if the columns of one of their corresponding matrices is a subset of the columns of the other. The weight of an edge is the computational cost of deriving the triangular factor of the subset matrix given the QRD of the larger matrix. The Minimum Spanning Tree (MST) of this graph provides efficient strategies of computing the QRDs of A_1, \dots, A_G when the columns of some of them are subset of the columns of others. If no such matrices exist, then the MST is equivalent to the second strategy which derives R_1, \dots, R_G one at a time. This is offset by adding new (artificial) nodes which correspond to matrices constructed from the conjunction of columns of two or more matrices.

The algorithm for deriving the MST of the graph that includes all artificial nodes has double exponential complexity and is thus computationally intractable. A heuristic approach that reduces the complexity of the algorithm to polynomial time has been proposed. The performance of the heuristic method has been investigated in two cases, where

it is most efficient. The numerical results indicate the superiority of this method compared to that of the second strategy which re-triangularizes RS_1, \dots, RS_G one at a time.

The re-triangularization of the matrices RS_i ($i = 1, \dots, G$) has been performed using Givens rotations. Householder transformations and block versions of Given rotations can also be used [11, 26, 27, 47]. Furthermore, in some econometric models the data matrices A_1, \dots, A_G may have special structure and properties [10, 12, 13, 14, 24]. In such cases the efficient re-triangularization of RS_i ($i = 1, \dots, G$) will require special algorithms. This will result in the edges of the directed graphs having different costs. However, the derivation of the MST and heuristic strategies for factorizing the matrices will remain the same. Currently the adaptation of the proposed strategies to compute subset regression models is under investigation [15, 16].

Acknowledgments

The authors are grateful to Maurice Clint for his constructive comments and suggestions.

Chapter 3

Efficient algorithms for block downdating of least squares solutions

Abstract:

Five computationally efficient algorithms for block downdating of the least squares solutions are proposed. The algorithms are block versions of Givens rotations strategies and are rich in BLAS-3 operations. They efficiently exploit the triangular structure of the matrices. The theoretical complexities of the algorithms are derived and analyzed. The performance of the implementations confirms the theoretical results. The new strategies are found to outperform existing downdating methods.

¹This chapter is a reprint of the paper: P. Yanev and E.J. Kontoghiorghes. Efficient algorithms for block downdating of least squares solutions. *Applied Numerical Mathematics*, 49:3–15, 2004.

3.1 Introduction

Consider the least squares (LS) problem

$$\hat{x} = \operatorname{argmin}_x \|Ax - b\|_2, \quad (3.1)$$

where $A \in \mathbb{R}^{m \times (n-1)}$ is of full column rank, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^{(n-1)}$ and $\|\cdot\|$ denotes the Euclidean norm. Let $\tilde{A} = (A \ b)$ be partitioned as

$$\tilde{A} \equiv \begin{pmatrix} \tilde{A}_1 \\ \tilde{A}_2 \end{pmatrix} \equiv \begin{pmatrix} A_1 & b_1 \\ A_2 & b_2 \end{pmatrix} \begin{matrix} d \\ m-d \end{matrix}. \quad (3.2)$$

The downdating least squares problem can be defined as the solution of the least squares problem

$$\hat{x}_2 = \operatorname{argmin}_x \|A_2x - b_2\|_2 \quad (3.3)$$

after (3.1) has been solved. Here it is assumed without loss of generality that the first d observations are deleted, A_2 has full column rank and $m > d + n$.

Let the QR decomposition (QRD) of \tilde{A} be given by:

$$Q^T \tilde{A} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} = \begin{pmatrix} R & u \\ 0 & p \\ 0 & 0 \end{pmatrix} \begin{matrix} n-1 \\ 1 \\ m-n \end{matrix}, \quad (3.4)$$

where $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{(n-1) \times (n-1)}$ is upper triangular. The LS estimator of x in (3.1) is obtained from the solution of the triangular system

$$R\hat{x} = u. \quad (3.5)$$

Thus, the downdating problem can also be seen as equivalent to computing the QRD of \tilde{A}_2

$$Q_2^T \tilde{A}_2 = \begin{pmatrix} \tilde{R}_2 \\ 0 \end{pmatrix} = \begin{pmatrix} R_2 & u_2 \\ 0 & p_2 \\ 0 & 0 \end{pmatrix} \begin{matrix} n-1 \\ 1 \\ m-d-n \end{matrix} \quad (3.6)$$

and solving $R_2\hat{x}_2 = u_2$ after (3.4) and (3.5) have been computed. The orthogonal matrix Q_2 and the upper triangular matrix \tilde{R}_2 in (3.6) are of order $(m - d)$ and n , respectively.

Different sequential strategies for solving the downdating least squares problem have been proposed [5, 19, 20, 24, 28, 32, 33]. These mainly consider Givens rotations for downdating the LS solution by single observation, or the straightforward use of the QRD for downdating the block of observations. In the latter case the structure of the matrices is not exploited [8]. In this work sequential algorithms for block-downdating the LS solution are proposed. These new methods are rich in BLAS-3 operations and take advantage of the initial triangular structure of the matrices. The evaluation of the various methods is performed using theoretical measures of complexity and experimental results.

It is assumed that the orthogonal matrix Q in (3.4) is available. The algorithms have been implemented on Intel Pentium III, 800 MHz processor. The performance of the algorithms has been evaluated and the execution times are reported in seconds. The complexity of the algorithms in number of flops (floating point operations) are also reported. The next section considers various LS block-downdating strategies. In section 3 the performance of the algorithms is evaluated. Finally, in section 4 conclusions are drawn.

3.2 Block-downdating of the LS solutions

The downdating LS problem can be solved in three stages. Let Q in (3.4) be partitioned as

$$Q^T \equiv \begin{pmatrix} & d & m-d \\ Q_{1,1}^T & Q_{1,2}^T \\ Q_{2,1}^T & Q_{2,2}^T \\ Q_{3,1}^T & Q_{3,2}^T \end{pmatrix} \begin{matrix} n \\ d \\ m-d-n \end{matrix}. \quad (3.7)$$

The first stage computes the QRD

$$H^T \begin{pmatrix} Q_{2,1}^T \\ Q_{3,1}^T \end{pmatrix} = \begin{pmatrix} Z \\ 0 \end{pmatrix} \begin{matrix} d \\ m-d-n \end{matrix} \quad (3.8a)$$

and the product

$$H^T \begin{pmatrix} Q_{2,2}^T \\ Q_{3,2}^T \end{pmatrix} = \begin{pmatrix} \tilde{Q}_{2,2}^T \\ \tilde{Q}_{3,2}^T \end{pmatrix}. \quad (3.8b)$$

Here H is orthogonal of order $(m-n)$ and $Z \in \mathbb{R}^{d \times d}$ is upper triangular. The second stage computes the row-permuted QRD

$$G^T \begin{pmatrix} Q_{1,1}^T \\ Z \end{pmatrix} = \begin{pmatrix} 0 \\ D \end{pmatrix} \quad (3.9a)$$

and the products

$$G^T \begin{pmatrix} R \\ 0 \end{pmatrix} = \begin{pmatrix} B \\ E \end{pmatrix}_d \quad \text{and} \quad G^T \begin{pmatrix} Q_{1,2}^T \\ \tilde{Q}_{2,2}^T \end{pmatrix} = \begin{pmatrix} \tilde{Q}_{1,2}^T \\ 0 \end{pmatrix}_d. \quad (3.9b)$$

Here G is a $(d+n) \times (d+n)$ orthogonal matrix and, by construction, $D \in \mathbb{R}^{d \times d}$ is upper triangular, but it will be shown that $|D| = I_d$. That is, D is diagonal with entries ± 1 . Finally, the third stage computes the QRD of B , i.e.

$$\tilde{Q}^T B = \tilde{R}_2 \quad (3.10)$$

with orthogonal $\tilde{Q} \in \mathbb{R}^{n \times n}$ and upper triangular $\tilde{R}_2 \in \mathbb{R}^{n \times n}$. In summary,

$$\begin{aligned} \begin{pmatrix} \tilde{Q}^T & 0 \\ 0 & I_{m-n} \end{pmatrix} \begin{pmatrix} G^T & 0 \\ 0 & I_{m-d-n} \end{pmatrix} \begin{pmatrix} I_n & 0 \\ 0 & H^T \end{pmatrix} Q^T \tilde{A} &= \begin{pmatrix} 0 & \tilde{Q}^T \tilde{Q}_{1,2}^T \\ D & 0 \\ 0 & \tilde{Q}_{3,2}^T \end{pmatrix} \begin{pmatrix} \tilde{A}_1 \\ \tilde{A}_2 \end{pmatrix} \\ &\equiv \begin{pmatrix} \tilde{R}_2 \\ E \\ 0 \end{pmatrix}. \end{aligned}$$

Here it can be seen that $(0 \ D^T \ 0)^T$ is the image under an orthogonal map of an orthogonal set of d column vectors – the first d columns of Q^T – and hence, D has orthogonal columns. Since D is by construction upper triangular, its diagonal elements must be ± 1 . Hence, $E = D\tilde{A}_1$, i.e. $DE = \tilde{A}_1$ is the deleted block of observations, and the orthogonal Q_2^T in the QRD (3.6) is given by

$$Q_2^T = \begin{pmatrix} \tilde{Q}^T \tilde{Q}_{1,2}^T \\ \tilde{Q}_{3,2}^T \end{pmatrix}.$$

If Q in (3.4) is not available, but A_1 and R are known, then $Q_{1,1}$ and Z in (3.9a) can be obtained from the solution of the triangular system $A_1 = Q_{1,1}R$ and the Cholesky factorization $Z^T Z = I_d - Q_{1,1}Q_{1,1}^T$, respectively [19, 24, 28].

The computation of (3.8a) and (3.8b) can be obtained efficiently using standard QRD methods and software, e.g. LAPACK, which are based on Householder transformations [2, 20]. Now, consider the computation of (3.9a), i.e. the second stage of the downdating process. Let $Q_{1,1}^T$ and R be partitioned, respectively, as

$$Q_{1,1}^T = \begin{pmatrix} d \\ W^{(1)} \\ W^{(2)} \\ \vdots \\ W^{(g)} \end{pmatrix} \begin{matrix} n_1 \\ n_2 \\ \vdots \\ n_g \end{matrix} \quad \text{and} \quad R = \begin{pmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,g} \\ & R_{2,2} & \cdots & R_{2,g} \\ & & \ddots & \vdots \\ & & & R_{g,g} \end{pmatrix} \begin{matrix} n_1 \\ n_2 \\ \vdots \\ n_g \end{matrix}, \quad (3.11)$$

where $n = \sum_{i=1}^g n_i$. The factorization (3.9a) is obtained in g steps block by block. Starting from $Z^{(0)} = Z$ and $E_g^{(0)} = 0$, the i th ($i = 1, \dots, g$) step computes the row-permuted QRD

$$G_i^T \begin{pmatrix} W^{(i)} \\ Z^{(i-1)} \end{pmatrix} = \begin{pmatrix} 0 \\ Z^{(i)} \end{pmatrix} \begin{matrix} n_i \\ d \end{matrix} \quad (3.12a)$$

and the product

$$G_i^T \begin{pmatrix} R_{i,i} & \cdots & R_{i,g} \\ 0 & \cdots & E_g^{(i-1)} \end{pmatrix} = \begin{pmatrix} \widehat{R}_{i,i} & \cdots & \widehat{R}_{i,g} \\ E_i^{(i)} & \cdots & E_g^{(i)} \end{pmatrix}, \quad (3.12b)$$

where G_i is an orthogonal matrix of order $(d + n_i)$, $Z^{(i)}$ is upper triangular and $\widehat{R}_{i,i}$ is full. Thus, in (3.9a) $|D| = |Z^{(g)}| \equiv I_d$, $E \equiv (E_1^{(g)} \cdots E_g^{(g)})$ and the matrix B has the block-triangular structure

$$B = \begin{pmatrix} \widehat{R}_{1,1} & \widehat{R}_{1,2} & \cdots & \widehat{R}_{1,g} \\ & \widehat{R}_{2,2} & \cdots & \widehat{R}_{2,g} \\ & & \ddots & \vdots \\ & & & \widehat{R}_{g,g} \end{pmatrix} \begin{matrix} n_1 \\ n_2 \\ \vdots \\ n_g \end{matrix}. \quad (3.13)$$

The factorization (3.10) is obtained by computing the QRDs

$$\widetilde{Q}_i^T \widehat{R}_{i,i} = \widetilde{R}_{i,i} \quad (3.14a)$$

and

$$\tilde{Q}_i^T \begin{pmatrix} \hat{R}_{i,i+1} & \cdots & \hat{R}_{i,g} \end{pmatrix} = \begin{pmatrix} \tilde{R}_{i,i+1} & \cdots & \tilde{R}_{i,g} \end{pmatrix}, \quad (3.14b)$$

where the orthogonal \tilde{Q}_i and upper triangular $\tilde{R}_{i,i}$ are of order n_i and $i = 1, \dots, g$. Thus, \tilde{Q}^T and \tilde{R}_2 in (3.10) are given, respectively, by $\tilde{Q}^T = \text{diag}(\tilde{Q}_1^T, \dots, \tilde{Q}_g^T)$ and

$$\tilde{R}_2 = \begin{pmatrix} \tilde{R}_{1,1} & \tilde{R}_{1,2} & \cdots & \tilde{R}_{1,g} \\ & \tilde{R}_{2,2} & \cdots & \tilde{R}_{2,g} \\ & & \ddots & \vdots \\ & & & \tilde{R}_{g,g} \end{pmatrix}. \quad (3.15)$$

Algorithm 3 summarizes the steps of the block strategy for computing the downdating LS solution. The standard colon notation will be used to denote subvectors and submatrices [20].

Algorithm 3 Downdating the least squares problem (3.3).

- 1: Let Q be partitioned as in (3.7)
 - 2: Compute the QRD $H^T \begin{pmatrix} Q_{2,1}^T \\ Q_{3,1}^T \end{pmatrix} = \begin{pmatrix} Z \\ 0 \end{pmatrix}$
 - 3: Let $Z^{(0)} = Z$ and $E_g^{(0)} = 0$
 - 4: Let \tilde{R} in (3.4) and $Q_{1,1}^T$ in (3.7) be partitioned as in (3.11)
 - 5: **for** $i = 1, \dots, g$ **do**
 - 6: Compute the row-permuted QRD $G_i^T \begin{pmatrix} W^{(i)} \\ Z^{(i-1)} \end{pmatrix} = \begin{pmatrix} 0 \\ Z^{(i)} \end{pmatrix}$
 - 7: $G_i^T \begin{pmatrix} R_{i,i:g} \\ E_{i:g}^{(i-1)} \end{pmatrix} = \begin{pmatrix} \hat{R}_{i,i:g} \\ E_{i:g}^{(i)} \end{pmatrix}$
 - 8: **end for**
 - 9: **for** $i = 1, \dots, g$ **do**
 - 10: Compute the QRD $\tilde{Q}_i^T \hat{R}_{i,i} = \tilde{R}_{i,i}$
 - 11: $\tilde{Q}_i^T \hat{R}_{i,i+1:g} = \tilde{R}_{i,i+1:g}$
 - 12: **end for**
 - 13: Let $\tilde{R}_2 = [\tilde{R}_{i,j}] \equiv \begin{pmatrix} R_2 & u_2 \\ 0 & p_2 \end{pmatrix}$ as in (3.6)
 - 14: Solve the triangular system $R_2 \hat{x}_2 = u_2$
-

3.3 Strategies for computing the factorization (3.12)

The main operation of the downdating LS Algorithm 3 is the loop 5-8 which corresponds to the factorizations (3.12a) and (3.12b). Hereafter, the particularly interesting case where $n \gg d$ is considered. The theoretical complexity for computing (3.12) directly using the standard LAPACK routines (hereafter called Strategy 1) is given by

$$T_{1,i} = 2d^3(5/3 + 3i) + 2d^2i \equiv O(2d^3(5/3 + 3i)). \quad (3.16)$$

Let $W^{(i)}$ and $Z^{(i)}$ in (3.12a) be partitioned, respectively, as

$$W^{(i)} = \begin{pmatrix} v_1 & \cdots & v_k \\ W_1^{(i)} & \cdots & W_k^{(i)} \end{pmatrix}_{n_i} \quad \text{and} \quad Z^{(i)} = \begin{pmatrix} v_1 & \cdots & v_k \\ Z_{1,1}^{(i)} & \cdots & Z_{1,k}^{(i)} \\ \vdots & \ddots & \vdots \\ Z_{k,k}^{(i)} \end{pmatrix} \begin{matrix} v_1 \\ \vdots \\ v_k \end{matrix}, \quad (3.17)$$

where $Z_{j,j}^{(i)}$ ($j = 1, \dots, k$) is upper triangular and $n_i = \sum_{j=1}^k v_j$. For simplicity it will be assumed that $n_i = d$ ($i = 1, \dots, g$) and $v_j = v$ ($j = 1, \dots, k$), i.e. $n = gd$, $d = kv$ and v is the block size of the partitioned matrices. The updating QRD (3.12a) is obtained in k steps. For $W_j^{(i,0)} \equiv W_j^{(i)}$ and $W_j^{(i,k)} \equiv 0$, the j th step ($j = 1, \dots, k$) computes the updating QRD of smaller-in-dimension matrices

$$D_j^T \begin{pmatrix} W_j^{(i,j-1)} & W_{j+1}^{(i,j-1)} & \cdots & W_k^{(i,j-1)} \\ Z_{j,j}^{(i-1)} & Z_{j,j+1}^{(i-1)} & \cdots & Z_{j,k}^{(i-1)} \end{pmatrix} = \begin{pmatrix} 0 & W_{j+1}^{(i,j)} & \cdots & W_k^{(i,j)} \\ Z_{j,j}^{(i)} & Z_{j,j+1}^{(i)} & \cdots & Z_{j,k}^{(i)} \end{pmatrix} \begin{matrix} kv \\ v \end{matrix}, \quad (3.18)$$

where D_j^T is orthogonal of order $(k+1)v$, $Z_{j,j}^{(i)} \in \mathbb{R}^{v \times v}$ is upper triangular and recall $d = kv$. The orthogonal D_j^T can be applied to the corresponding rows of the matrices in (3.12b) without explicitly computing G_i^T . That is, it computes

$$D_j^T \begin{pmatrix} R_{i,i}^{(j-1)} & \cdots & R_{i,g}^{(j-1)} \\ 0 & \cdots & E_g^{(j-1,j)} \end{pmatrix} = \begin{pmatrix} R_{i,i}^{(j)} & \cdots & R_{i,g}^{(j)} \\ E_i^{(j,j)} & \cdots & E_g^{(j,j)} \end{pmatrix}, \quad (3.19)$$

where $R_{i,q}^{(0)} = R_{i,q}$, $R_{i,q}^{(k)} = \widehat{R}_{i,q}$ in (3.12) and $E_q^{(j,j)}$ denotes the j th block row of the matrix $E_q^{(i)}$ ($q = i, \dots, g$) during the j th ($j = 1, \dots, k$) step of the computation. The theoretical complexity of this method (Strategy 2) is given by

$$T_{2,i} = 2d^3(1 + 2i) + d^2(1 + 2i)(1 + v) + dv(v/3 - 1) \equiv O(2d^3(1 + 2i)). \quad (3.20)$$

An alternative approach (Strategy 3) is to construct explicitly G_i^T and then compute (3.12b). The orthogonal $G_i^T = \tilde{D}_k^T \cdots \tilde{D}_1^T$, where

$$D_j^T = \begin{pmatrix} kv & v \\ D_{1,1}^{(j)} & D_{1,2}^{(j)} \\ D_{2,1}^{(j)} & D_{2,2}^{(j)} \end{pmatrix} \begin{matrix} kv \\ v \end{matrix}, \quad (3.21a)$$

with $D_{2,2}^{(j)}$ lower triangular [24, pages 49-53] and

$$\tilde{D}_j^T = \begin{pmatrix} kv & (j-1)v & v & (k-j)v \\ D_{1,1}^{(j)} & 0 & D_{1,2}^{(j)} & 0 \\ 0 & I_{(j-1)v} & 0 & 0 \\ D_{2,1}^{(j)} & 0 & D_{2,2}^{(j)} & 0 \\ 0 & 0 & 0 & I_{(k-j)v} \end{pmatrix} \begin{matrix} kv \\ (j-1)v \\ v \\ (k-j)v \end{matrix}. \quad (3.21b)$$

Theorem 1 *Let $C^{(j)} = \tilde{D}_j^T \cdots \tilde{D}_1^T$ and $C^{(0)} = I_{2kv}$. The orthogonal matrix $C^{(j)}$ has the structure*

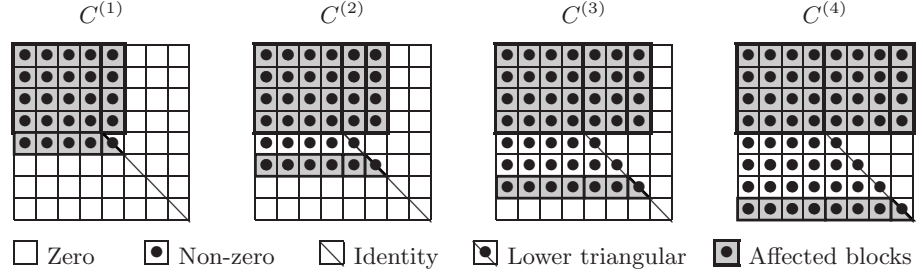
$$C^{(j)} = \begin{pmatrix} kv & (j-1)v & v & (k-j)v \\ C_{1,1}^{(j)} & C_{1,2}^{(j)} & C_{1,3}^{(j)} & 0 \\ C_{2,1}^{(j)} & C_{2,2}^{(j)} & 0 & 0 \\ C_{3,1}^{(j)} & C_{3,2}^{(j)} & C_{3,3}^{(j)} & 0 \\ 0 & 0 & 0 & I_{(k-j)v} \end{pmatrix} \begin{matrix} kv \\ (j-1)v \\ v \\ (k-j)v \end{matrix}, \quad (3.22)$$

where $C_{2,2}^{(j)}$ and $C_{3,3}^{(j)}$ are lower triangular.

PROOF. The proof is by induction. For $j = 1$ the matrix

$$C^{(1)} \equiv \tilde{D}_1^T = \begin{pmatrix} kv & v & (k-1)v \\ D_{1,1}^{(1)} & D_{1,2}^{(1)} & 0 \\ D_{2,1}^{(1)} & D_{2,2}^{(1)} & 0 \\ 0 & 0 & I_{(k-1)v} \end{pmatrix} \begin{matrix} kv \\ v \\ (k-1)v \end{matrix}, \quad (3.23)$$

has the structure (3.22). Assuming that $C^{(j)}$ has the structure defined in (3.22). It will be

Figure 3.1: The orthogonal $C^{(j)}$, ($j = 1, \dots, 4$).

Here, $\widehat{C}_{2,2}^{(k)}$ is lower triangular and this can be exploited when it comes to the matrix multiplication (3.12b). This strategy to compute (3.12) is summarized by Algorithm 4. The steps 7-10 correspond to the computation of (3.12b). The complexity of Algorithm 4 (i.e. Strategy 3) is given by

$$T_{3,i} = d^3(3 + 7i) + 2d^2(2 + 3v) + 4dv^2/3 \equiv O(d^3(3 + 7i)). \quad (3.26)$$

Algorithm 4 The third Strategy for computing (3.12).

- 1: Let $C^{(0)} = I_{2kv}$
 - 2: **for** $j = 1, \dots, k$ **do**
 - 3: Compute the QRD $D_j^T \begin{pmatrix} W_j^{(i,j-1)} \\ Z_{j,j}^{(i-1)} \end{pmatrix} = \begin{pmatrix} 0 \\ Z_{j,j}^{(i)} \end{pmatrix}$
 - 4: $D_j^T \begin{pmatrix} W_{j+1:k}^{(i,j-1)} \\ Z_{j,j+1:k}^{(i-1)} \end{pmatrix} = \begin{pmatrix} W_{j+1:k}^{(i,j)} \\ Z_{j,j+1:k}^{(i)} \end{pmatrix}$
 - 5: Compute $C^{(j)} = \widetilde{D}_j^T C^{(j-1)}$ as in (3.24)
 - 6: **end for**
 - 7: $\widehat{R}_{i,i} = C_{1,1}^{(k)} R_{i,i}$
 - 8: $E_i^{(i)} = \widehat{C}_{2,1}^{(k)} R_{i,i}$
 - 9: $\widehat{R}_{i,i+1:g} = C_{1,1}^{(k)} R_{i,i+1:g} + \widehat{C}_{1,2}^{(k)} E_{i+1:g}^{(i-1)}$
 - 10: $E_{i+1:g}^{(i)} = \widehat{C}_{2,1}^{(k)} R_{i,i+1:g} + \widehat{C}_{2,2}^{(k)} E_{i+1:g}^{(i-1)}$
-

An alternative approach is to compute the QRD of $W^{(i)}$ prior to the computation of (3.12a). That is, (3.12) is computed in three stages:

$$Q_*^T \begin{pmatrix} d & n_i & \dots & n_g \\ W^{(i)} & R_{i,i} & \dots & R_{i,g} \end{pmatrix} = \begin{pmatrix} d & n_i & \dots & n_g \\ U^{(i)} & \check{R}_{i,i} & \dots & \check{R}_{i,g} \end{pmatrix}, \quad (3.27a)$$

$$\tilde{G}_i^T \begin{pmatrix} U^{(i)} \\ Z^{(i-1)} \end{pmatrix} = \begin{pmatrix} 0 \\ Z^{(i)} \end{pmatrix} \quad (3.27b)$$

and

$$\tilde{G}_i^T \begin{pmatrix} \check{R}_{i,i} & \cdots & \check{R}_{i,g} \\ 0 & \cdots & E_g^{(i-1)} \end{pmatrix} = \begin{pmatrix} \hat{R}_{i,i} & \cdots & \hat{R}_{i,g} \\ E_i^{(i)} & \cdots & E_g^{(i)} \end{pmatrix}. \quad (3.27c)$$

Here \tilde{G}_i is orthogonal of order $2d$ and in (3.12), $G_i^T = \tilde{G}_i^T \text{diag}(Q_*^T, I_d)$.

Now, let $U^{(i)}$ be conformally partitioned as $Z^{(i)}$ in (3.17). That is,

$$U^{(i)} = \begin{pmatrix} v_1 & \cdots & v_k \\ U_{1,1}^{(i)} & \cdots & U_{1,k}^{(i)} \\ & \ddots & \vdots \\ & & U_{k,k}^{(i)} \end{pmatrix} \begin{matrix} v_1 \\ \vdots \\ v_k \end{matrix}, \quad (3.28)$$

where $U_{j,j}^{(i)}$ ($j = 1, \dots, k$) is upper triangular, $U_{1:j,j}^{(i,0)} \equiv U_{1:j,j}^{(i)}$ and $U_{1:j,j}^{(i,k)} \equiv 0$. The updating QRD (3.27a) is obtained in k steps. The j th step ($j = 1, \dots, k$) computes the updating QRD of smaller-in-dimension matrices

$$D_j^T \begin{pmatrix} U_{1:j,j}^{(i,j-1)} & U_{1:j,j+1}^{(i,j-1)} & \cdots & U_{1:j,k}^{(i,j-1)} \\ Z_{j,j}^{(i-1)} & Z_{j,j+1}^{(i-1)} & \cdots & Z_{j,k}^{(i-1)} \end{pmatrix} = \begin{pmatrix} 0 & U_{1:j,j+1}^{(i,j)} & \cdots & U_{1:j,k}^{(i,j)} \\ Z_{j,j}^{(i)} & Z_{j,j+1}^{(i)} & \cdots & Z_{j,k}^{(i)} \end{pmatrix}, \quad (3.29)$$

where D_j^T is orthogonal of order $(j+1)v$ and $Z_{j,j}^{(i)} \in \mathbb{R}^{v \times v}$ is upper triangular.

Two approaches for computing (3.12) can be considered. The first one applies D_j^T to the corresponding rows of the matrices in (3.27b) without deriving \tilde{G}_i^T explicitly. The complexity of this approach (Strategy 4) is given by

$$\begin{aligned} T_{4,i} &= 2d^3(1+2i) + d^2(1+2v+4i+4iv) + dv(2v/3-1) \\ &\equiv O(2d^3(1+2i)). \end{aligned} \quad (3.30)$$

Notice that Strategies 2 and 4 have the same order of complexity. The second approach constructs $\tilde{G}_i^T = \tilde{D}_k^T \cdots \tilde{D}_1^T$. Now,

$$D_j^T = \begin{pmatrix} (j-1)v & v & v \\ D_{1,1}^{(j)} & D_{1,2}^{(j)} & D_{1,3}^{(j)} \\ D_{2,1}^{(j)} & D_{2,2}^{(j)} & D_{2,3}^{(j)} \\ D_{3,1}^{(j)} & D_{3,2}^{(j)} & D_{3,3}^{(j)} \end{pmatrix} \begin{matrix} (j-1)v \\ v \\ v \end{matrix}, \quad (3.31a)$$

with $D_{3,2}^{(j)}$ and $D_{3,3}^{(j)}$ lower triangular [24, pages 49-53] and

$$\tilde{D}_j^T = \begin{pmatrix} (j-1)v & v & (k-1)v & v & (k-j)v \\ D_{1,1}^{(j)} & D_{1,2}^{(j)} & 0 & D_{1,3}^{(j)} & 0 \\ D_{2,1}^{(j)} & D_{2,2}^{(j)} & 0 & D_{2,3}^{(j)} & 0 \\ 0 & 0 & I_{(k-1)v} & 0 & 0 \\ D_{3,1}^{(j)} & D_{3,2}^{(j)} & 0 & D_{3,3}^{(j)} & 0 \\ 0 & 0 & 0 & 0 & I_{(k-j)v} \end{pmatrix} \begin{matrix} (j-1)v \\ v \\ (k-1)v \\ v \\ (k-j)v \end{matrix}. \quad (3.31b)$$

Furthermore, as in Theorem 1, it can be proved that $\tilde{C}^{(j)}$ has the structure

$$\tilde{C}^{(j)} = \begin{pmatrix} (j-1)v & v & (k-j)v & (j-1)v & v & (k-j)v \\ \tilde{C}_{1,1}^{(j)} & \tilde{C}_{1,2}^{(j)} & 0 & \tilde{C}_{1,3}^{(j)} & \tilde{C}_{1,4}^{(j)} & 0 \\ \tilde{C}_{2,1}^{(j)} & \tilde{C}_{2,2}^{(j)} & 0 & \tilde{C}_{2,3}^{(j)} & \tilde{C}_{2,4}^{(j)} & 0 \\ 0 & 0 & I_{(k-j)v} & 0 & 0 & 0 \\ \tilde{C}_{3,1}^{(j)} & 0 & 0 & \tilde{C}_{3,2}^{(j)} & 0 & 0 \\ \tilde{C}_{4,1}^{(j)} & \tilde{C}_{4,2}^{(j)} & 0 & \tilde{C}_{4,3}^{(j)} & \tilde{C}_{4,4}^{(j)} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{(k-j)v} \end{pmatrix} \begin{matrix} (j-1)v \\ v \\ (k-j)v \\ (j-1)v \\ v \\ (k-j)v \end{matrix}, \quad (3.32)$$

where $\tilde{C}_{3,1}^{(j)}$, $\tilde{C}_{3,2}^{(j)}$, $\tilde{C}_{4,2}^{(j)}$ and $\tilde{C}_{4,4}^{(j)}$ are lower triangular. Figure 3.2 shows the affected submatrices during the computation of $\tilde{C}^{(1)}, \dots, \tilde{C}^{(k)} \equiv \tilde{G}_i^T$, where $k = 4$ and a square denotes an $v \times v$ submatrix.

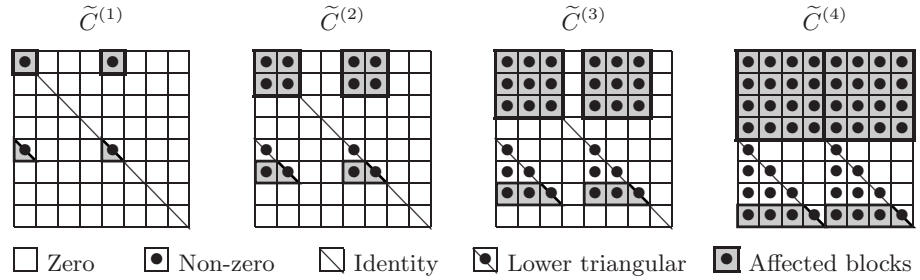


Figure 3.2: The orthogonal $\tilde{C}^{(j)}$, ($j = 1, \dots, 4$).

Thus, the orthogonal matrix \tilde{G}_i^T in (3.27b) has the structure

$$\tilde{G}_i^T \equiv \tilde{C}^{(k)} = \begin{pmatrix} kv & kv \\ \tilde{C}_{1,1}^{(k)} & \tilde{C}_{1,2}^{(k)} \\ \tilde{C}_{2,1}^{(k)} & \tilde{C}_{2,2}^{(k)} \end{pmatrix}_{kv}, \quad (3.33)$$

where

$$\bar{C}_{1,1}^{(k)} = \begin{pmatrix} \tilde{C}_{1,1}^{(k)} & \tilde{C}_{1,2}^{(k)} \\ \tilde{C}_{2,1}^{(k)} & \tilde{C}_{2,2}^{(k)} \end{pmatrix}, \quad \bar{C}_{1,2}^{(k)} = \begin{pmatrix} \tilde{C}_{1,3}^{(k)} & \tilde{C}_{1,4}^{(k)} \\ \tilde{C}_{2,3}^{(k)} & \tilde{C}_{2,4}^{(k)} \end{pmatrix}, \quad \bar{C}_{2,1}^{(k)} = \begin{pmatrix} \tilde{C}_{3,1}^{(k)} & 0 \\ \tilde{C}_{4,1}^{(k)} & \tilde{C}_{4,2}^{(k)} \end{pmatrix}$$

and

$$\bar{C}_{2,2}^{(k)} = \begin{pmatrix} \tilde{C}_{3,2}^{(k)} & 0 \\ \tilde{C}_{4,3}^{(k)} & \tilde{C}_{4,3}^{(k)} \end{pmatrix}.$$

Algorithm 5 summarizes the steps of this 4th strategy for computing (3.12), when reformulated as (3.27). The lines 9-10 compute (3.27c) by exploiting the triangular structure of $\bar{C}_{2,1}^{(k)}$ and $\bar{C}_{2,2}^{(k)}$. The complexity of Algorithm 5, i.e. Strategy 5, is given by

$$T_{5,i} = d^3(5/3 + 8i) + d^2(3 + 8v + 2i) + dv(1 + 4v) \equiv O(d^3(5/3 + 8i)). \quad (3.34)$$

Algorithm 5 The computation of factorization (3.27).

- 1: Let $\tilde{C}^{(0)} = I_{2kv}$
 - 2: Compute the QRD $Q_*^T W^{(i)} = U^{(i)}$
 - 3: $\check{R}_{i,i:g} = Q_{W_i}^T R_{i,i:g}$
 - 4: **for** $j = 1, \dots, k$ **do**
 - 5: Compute the QRD $D_j^T \begin{pmatrix} U^{(i,j-1)} \\ U_{1:j,j}^{(i,j-1)} \\ Z_{j,j}^{(i-1)} \end{pmatrix} = \begin{pmatrix} 0 \\ Z_{j,j}^{(i)} \end{pmatrix}$
 - 6: $D_j^T \begin{pmatrix} U^{(i,j-1)} \\ U_{1:j,j+1:k}^{(i,j-1)} \\ Z_{j,j+1:k}^{(i-1)} \end{pmatrix} = \begin{pmatrix} U^{(i,j)} \\ U_{1:j,j+1:k}^{(i,j)} \\ Z_{j,j+1:k}^{(i)} \end{pmatrix}$
 - 7: Compute $\tilde{C}^{(j)} = \tilde{D}_j^T \tilde{C}^{(j-1)}$ similar to (3.24)
 - 8: **end for**
 - 9: $\hat{R}_{i,i:g} = \bar{C}_{1,1}^{(k)} \check{R}_{i,i:g} + \bar{C}_{1,2}^{(k)} E_{i+1:g}^{(i+1)}$
 - 10: $E_{i,g}^{(i)} = \bar{C}_{2,1}^{(k)} \check{R}_{i,i:g} + \bar{C}_{2,2}^{(k)} E_{i+1:g}^{(i+1)}$
-

3.4 Numerical results and conclusion

The five strategies described here for computing (3.12) have been implemented and analyzed. A suitable block size v has been found to be 50 on the Intel Pentium III, 800 MHz processor. Table 3.1 shows the theoretical complexities and execution times of the

five strategies. The theoretical results confirm the performance of the implementations. Strategy 2 has the best performance. That is, the computation of (3.12) is best obtained by computing (3.18) and (3.19) for $j = 1, \dots, k$.

Table 3.1: Theoretical complexities (Mflops) and execution times (sec).

		Strategy 1		Strategy 2		Strategy 3		Strategy 4		Strategy 5	
d	i	T_1	Time	T_2	Time	T_3	Time	T_4	Time	T_5	Time
200	2	122.82	1.05	90.35	0.90	148.82	1.36	100.68	0.94	159.62	1.44
200	3	170.90	1.46	126.43	1.27	204.82	1.90	140.84	1.36	223.70	2.02
200	4	218.98	1.87	162.51	1.64	260.82	2.41	181.00	1.79	287.78	2.61
200	6	315.14	2.81	234.67	2.38	372.82	3.45	261.32	2.69	415.94	3.78
200	8	411.30	3.92	306.83	3.14	484.82	4.38	341.64	3.64	544.10	4.90
300	2	414.36	3.58	293.18	2.95	487.36	4.49	316.29	3.01	516.64	4.73
300	3	576.54	5.03	410.36	4.15	676.36	6.54	442.65	4.46	732.82	6.79
300	4	738.72	6.31	527.54	5.32	865.36	8.21	569.01	6.07	949.00	8.70
300	6	1063.08	11.40	761.90	8.57	1243.36	12.93	821.73	10.88	1381.36	13.59
300	8	1387.44	17.32	996.26	13.01	1621.36	21.08	1074.45	14.81	1813.72	22.36
400	2	981.97	8.40	681.11	6.85	1137.97	10.39	722.08	7.01	1199.80	11.18
400	3	1366.29	11.70	953.43	9.59	1585.97	15.29	1010.72	9.94	1712.12	15.88
400	4	1750.61	17.13	1225.75	15.18	2033.97	23.51	1299.36	16.10	2224.44	23.83
400	6	2519.25	28.09	1770.39	22.95	2929.97	35.26	1876.64	24.57	3249.08	36.73

Table 3.2 shows the execution times of three methods for solving the downdating problem for various values of m , n and d . The Standard LAPACK method corresponds to the conventional method in [8] which computes (3.7)–(3.10) using LAPACK routines, but without exploiting the structure of Z and R in (3.9a). The Givens method uses plane rotations to delete the d observations one at a time [19]. The new downdating method is Algorithm 3 which employs the 2nd strategy to compute (3.12).

The Givens method outperforms the conventional one when the number of deleted observations d is small compared to the number of variables n . Furthermore, if $d \ll n$, then the Givens method also outperforms the new downdating algorithm. However, for not very small d the proposed block-downdating algorithm is computationally the most efficient one. The numerical stability of the proposed algorithm should be investigated [9].

The parallelization of the new downdating algorithm using the various strategies for computing (3.12) is currently under investigation. The adaptation of the computationally

Table 3.2: Execution times of the downdating methods.

			ALGORITHMS			RATIO		
m	n	d	LAPACK	Givens	New Down.	$\frac{\text{LAPACK}}{\text{New Down.}}$	$\frac{\text{LAPACK}}{\text{Givens}}$	$\frac{\text{Givens}}{\text{New Down.}}$
1000	400	10	0.97	0.11	0.09	10.77	8.81	1.22
1000	400	20	1.09	0.23	0.17	6.41	4.73	1.35
1000	400	50	1.23	0.64	0.37	3.32	1.92	1.62
1000	400	100	1.77	1.55	0.94	1.88	1.14	1.64
1000	400	200	3.08	3.79	1.84	1.67	0.81	2.05
1000	600	10	3.34	0.24	0.22	15.18	13.91	1.09
1000	600	20	3.57	0.52	0.37	9.64	6.86	1.40
1000	600	50	3.82	1.31	0.86	4.44	2.91	1.52
1000	600	100	5.04	2.98	1.78	2.83	1.69	1.67
1000	600	200	7.43	7.66	3.84	1.93	0.96	1.99
1600	800	10	8.44	0.42	0.40	21.10	20.09	1.05
1600	800	20	8.56	0.92	0.68	12.58	9.30	1.35
1600	800	50	9.13	2.29	1.69	5.40	3.98	1.35
1600	800	100	10.86	5.40	3.37	3.22	2.01	1.60
1600	800	200	17.06	13.26	7.38	2.31	1.28	1.79
1600	800	400	31.11	35.55	16.90	1.84	0.87	2.08
2000	1200	10	24.62	0.96	1.13	21.78	25.64	0.84
2000	1200	20	34.20	2.06	2.17	15.76	16.60	0.94
2000	1200	50	36.29	4.98	4.83	7.51	7.28	1.03
2000	1200	100	40.30	11.28	8.19	4.92	3.57	1.37
2000	1200	200	51.48	25.88	15.37	3.34	1.98	1.68
2000	1200	400	83.38	153.59	37.60	2.21	0.54	4.08

efficient downdating algorithms are intended to be used in regression diagnostics and cross-validation, where repeatedly a number of observations is deleted [3, 27, 42]. It is expected that the improved performance of the proposed downdating methods will facilitate the investigation (evaluation of influential data) of large-scale models. Within this context the downdating of seemingly unrelated regression models is currently considered [10, 12, 27, 34].

Acknowledgments

The authors are grateful to Martin Gutknecht for his valuable comments and suggestions.

Chapter 4

Parallel algorithms for downdating the QR decomposition

Abstract:

A computationally efficient parallel algorithm for downdating the QR decomposition is proposed. The algorithm is a block version of sequential Givens strategies and efficiently exploits the triangular structure of the matrices. An efficient distribution of matrices over the processors is proposed. Furthermore, the algorithm does not require inter-processor communication. The theoretical complexity of the algorithm is derived and experimental results are presented and analyzed. The parallel strategy is scalable and highly efficient for large-scale downdating problems.

4.1 Introduction

The recomputation of the QR decomposition (QRD) of a matrix after rows have been deleted arises often in diverse applications [24]. Consider the QRD of the full rank $A \in$

¹This chapter is a reprint of the paper: P. Yanev and E.J. Kontoghiorghes. Parallel algorithms for downdating the QR decomposition. *Parallel Computing*, 2004 (Submitted).

$\mathbb{R}^{m \times n}$ matrix :

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}, \quad (4.1)$$

with

$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \begin{matrix} d \\ m-d \end{matrix} \quad \text{and} \quad Q^T = \begin{pmatrix} Q_{1,1}^T & Q_{1,2}^T \\ Q_{2,1}^T & Q_{2,2}^T \\ Q_{3,1}^T & Q_{3,2}^T \end{pmatrix} \begin{matrix} n \\ d \\ m-d-n \end{matrix}, \quad (4.2)$$

where $R \in \mathbb{R}^{n \times n}$ is upper triangular and $Q \in \mathbb{R}^{m \times m}$ is orthogonal. The computation of the QRD of A_2 having found the decomposition of A given by (4.1) is known as the downdating QRD problem. Thus, assuming that $m > d+n$ and A_2 has full rank, the downdating QRD is expressed as:

$$Q_2^T A_2 = \begin{pmatrix} R_2 \\ 0 \end{pmatrix} \begin{matrix} n \\ m-d-n \end{matrix}, \quad (4.3)$$

where Q_2 is an orthogonal matrix of order $(m-d)$ and $R_2 \in \mathbb{R}^{n \times n}$ is upper triangular.

The QRD (4.3) is derived in two stages utilizing the computations performed in (4.1) [5, 8, 19, 20, 24]. The first stage computes the factorizations

$$H^T \begin{pmatrix} Q_{2,1}^T \\ Q_{2,2}^T \end{pmatrix} = \begin{pmatrix} Z \\ 0 \end{pmatrix} \begin{matrix} d \\ m-d-n \end{matrix} \quad (4.4)$$

and

$$G^T \begin{pmatrix} Q_{1,1}^T & R \\ Z & 0 \end{pmatrix} = \begin{pmatrix} 0 & B \\ D & E \end{pmatrix} \begin{matrix} n \\ d \end{matrix}, \quad (4.5)$$

where H and G are orthogonal matrices of order $(m-n)$ and $(d+n)$, respectively, $Z \in \mathbb{R}^{d \times d}$ is upper triangular and $|D| = I_d$. The second stage computes the triangular factor R_2 in (4.3) by finding the QRD

$$\tilde{Q}^T B = R_2. \quad (4.6)$$

If Q , in (4.1), is not available, then $Q_{1,1}^T$ and Z in (4.5) can be computed by solving the triangular system $A_1 = Q_{1,1}R$ and the Cholesky factorization $ZZ^T = I_d - Q_{1,1}Q_{1,1}^T$ [24]. Hereafter it is assumed that $Q_{1,1}$ and Z are known and only R_2 , in (4.3), and thus in (4.6), is required. That is, Q_2 is not explicitly computed.

Thus, the downdating problem becomes equivalent to computing (4.5) and (4.6). In the light of this observation sequential and parallel strategies to solve the downdating problem have been designed [4, 8, 19, 20, 24, 28, 32, 33]. A computationally efficient block-downdating algorithm has also been recently proposed [47]. In this paper, a parallel strategy based on this algorithm is investigated. The notation is consistent with that in [47]. The sequential algorithm is briefly presented in the next section. Section 3 considers various parallel strategies and presents the theoretical and computational results. Section 4 offers some conclusions.

4.2 Block downdating of the QRD

Recently an efficient block-generalization of a Givens strategy for single-row downdating of the QRD has been proposed [47]. The algorithm is rich in BLAS-3 operations and takes advantage of the initial triangular structure of the matrices. Let the matrices $Q_{1,1}^T$ and R in (4.5) be partitioned, respectively, as

$$Q_{1,1}^T = \begin{pmatrix} d \\ W^{(1)} \\ W^{(2)} \\ \vdots \\ W^{(g)} \end{pmatrix} \begin{matrix} n_1 \\ n_2 \\ \vdots \\ n_g \end{matrix} \quad \text{and} \quad R = \begin{pmatrix} n_1 & n_2 & \cdots & n_g \\ R_{1,1} & R_{1,2} & \cdots & R_{1,g} \\ & R_{2,2} & \cdots & R_{2,g} \\ & & \ddots & \vdots \\ & & & R_{g,g} \end{pmatrix} \begin{matrix} n_1 \\ n_2 \\ \vdots \\ n_g \end{matrix}, \quad (4.7)$$

where $n = \sum_{i=1}^g n_i$. The sequential algorithm computes (4.5) in g steps. For $Z = Z^{(0)}$ and $E_g^{(0)} = 0$, the i th ($i = 1, \dots, g$) step computes the row-permuted QRD

$$G_i^T \begin{pmatrix} W^{(i)} \\ Z^{(i-1)} \end{pmatrix} = \begin{pmatrix} 0 \\ Z^{(i)} \end{pmatrix} \begin{matrix} n_i \\ d \end{matrix} \quad (4.8a)$$

and the product

$$G_i^T \begin{pmatrix} n_i & \cdots & n_g \\ R_{i,i} & \cdots & R_{i,g} \\ 0 & \cdots & E_g^{(i-1)} \end{pmatrix} = \begin{pmatrix} n_i & \cdots & n_g \\ \widehat{R}_{i,i} & \cdots & \widehat{R}_{i,g} \\ E_i^{(i)} & \cdots & E_g^{(i)} \end{pmatrix}, \quad (4.8b)$$

where G_i is orthogonal and of order $(d + n_i)$, $Z^{(i)}$ is upper triangular and B in (4.5) has the block-triangular structure

$$B = \begin{pmatrix} \widehat{R}_{1,1} & \widehat{R}_{1,2} & \cdots & \widehat{R}_{1,g} \\ & \widehat{R}_{2,2} & \cdots & \widehat{R}_{2,g} \\ & & \ddots & \vdots \\ & & & \widehat{R}_{g,g} \end{pmatrix}. \quad (4.9)$$

The QRD of B in (4.6) is obtained in g steps by computing at the i th step ($i = 1, \dots, g$) the QRDs:

$$Q_i^T \widehat{R}_{i,i} = \widetilde{R}_{i,i} \quad (4.10a)$$

and the products

$$Q_i^T \begin{pmatrix} \widehat{R}_{i,i+1} & \cdots & \widehat{R}_{i,g} \end{pmatrix} = \begin{pmatrix} \widetilde{R}_{i,i+1} & \cdots & \widetilde{R}_{i,g} \end{pmatrix}, \quad (4.10b)$$

where $\widetilde{R}_{i,i}$ is upper triangular and Q_i^T is orthogonal. That is, R_2 in (4.3) is given by:

$$R_2 = \begin{pmatrix} \widetilde{R}_{1,1} & \widetilde{R}_{1,2} & \cdots & \widetilde{R}_{1,g} \\ & \widetilde{R}_{2,2} & \cdots & \widetilde{R}_{2,g} \\ & & \ddots & \vdots \\ & & & \widetilde{R}_{g,g} \end{pmatrix}. \quad (4.11)$$

Algorithm 6 summarizes the steps of this strategy for block downdating the QRD. Householder transformations are used to compute the factorizations (4.5) and (4.6) and the orthogonal matrices G_i and Q_i ($i = 1, \dots, g$) are not explicitly constructed. The theoretical complexity of this algorithm is given by:

$$T_S(g, d) = 2gd^2(2d(3g + 5) + 3g)/3 \equiv O(4g^2d^3). \quad (4.12)$$

Algorithm 6 The sequential block downdating of the QRD.

- 1: Let $Q_{1,1}^T$ and R in (4.5) be partitioned as in (4.7)
 - 2: **for** $i = 1, \dots, g$ **do**
 - 3: Compute the row-permuted QRD $G_i^T \begin{pmatrix} W^{(i)} \\ Z^{(i-1)} \end{pmatrix} = \begin{pmatrix} 0 \\ Z^{(i)} \end{pmatrix}$
 - 4: Compute $G_i^T \begin{pmatrix} R_{i,i} & \cdots & R_{i,g} \\ 0 & \cdots & E_g^{(i-1)} \end{pmatrix} = \begin{pmatrix} \hat{R}_{i,i} & \cdots & \hat{R}_{i,g} \\ E_i^{(i)} & \cdots & E_g^{(i)} \end{pmatrix}$
 - 5: **end for**
 - 6: **for** $i = 1, \dots, g$ **do**
 - 7: Compute the QRD $Q_i^T \hat{R}_{i,i} = \tilde{R}_{i,i}$
 - 8: Compute $Q_i^T \begin{pmatrix} \hat{R}_{i,i+1} & \cdots & \hat{R}_{i,g} \end{pmatrix} = \begin{pmatrix} \tilde{R}_{i,i+1} & \cdots & \tilde{R}_{i,g} \end{pmatrix}$
 - 9: **end for**
-

4.3 Parallel downdating of the QRD

The design of a parallel algorithm requires the efficient distribution of matrices on the processors such that load balancing is achieved together with low inter-processor communication. Let p denotes the number of processors. Assume for simplicity that g in (4.7) is a multiple of p . Consider the block-partitioning of R in (4.7). The cyclic distribution will allocate $R_{:,i}$ ($i = 1, \dots, g$) to the processor P_{λ_i} , where $\lambda_i = (i - 1) \bmod p + 1$. This distribution results in the processor P_j being allocated the $n \times n/p$ matrix

$$R^{(j)} = (R_{:,j} \quad R_{:,j+p} \quad R_{:,j+2p} \quad \cdots \quad R_{:,j+g-p}).$$

The number of non-zero elements of $R^{(j)}$ is given by $g(g-p)/2p^2 + p(p-j)$. Thus, $R^{(p)}$, which is allocated to the last processor, has the maximum density.

In order to reduce the inter-processor communications the factorization (4.8a) is computed by each processor which then, updates its allocated matrix $R^{(j)}$. Furthermore, using a Single Program Multiple Data (SPMD) paradigm the local matrices $R^{(j)}$ ($j = 1, \dots, p$) are assumed to have maximum density and thus share the structure of $R^{(p)}$ with the result that the processors P_1, \dots, P_{p-1} perform computations on zero blocks. Figure 4.1 shows the distribution of the matrices R and E in (4.5), with $p = 4$ and $g = 16$. Note that $Q^* = (Q_{1,1} \quad Z^T)^T$ is duplicated in each processor. The shaded and blank blocks indicate, respectively, the affected and unaffected blocks during the computations. The empty shaded blocks remain zero throughout the factorization process.

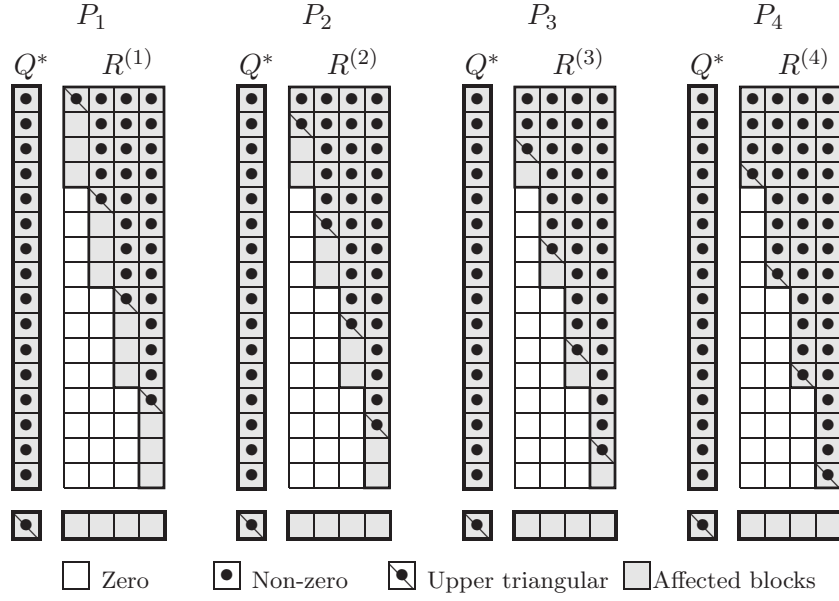


Figure 4.1: The cyclic distribution of the matrices on 4 processors, when $g = 16$.

The parallel downdating of the QRD problem is achieved in three stages. The first stage computes (4.8). Specifically, the factorization (4.8a) is computed by each processor P_j which then updates its local matrix $R^{(j)}$, where $j = 1, \dots, p$. The second stage computes (4.10a), i.e. processor P_j computes the QRDs

$$Q_{i,j}^T \widehat{R}_{i,[i/p]}^{(j)} = \widetilde{R}_{i,[i/p]}^{(j)}, \quad i = j, j+p, \dots, j+g-p. \quad (4.13)$$

The explicitly computed orthogonal matrices $Q_{i,j}^T$ are sent to all processors. That is, P_j sends $Q_{i,j}^T$ to P_r and also receives $Q_{i,r}^T$ in return, where $r = 1, \dots, p$ and $r \neq j$. Finally, in the last stage each processor applies the matrices $Q_{i,j}^T$ ($j = 1, \dots, p$ and $i = j, j+p, \dots, j+g-p$) to update $\widehat{R}^{(j)}$.

Algorithm 7 summarizes the steps of this parallel strategy for block downdating of the QRD. The theoretical computational complexity of this algorithm is given by:

$$T_{P_1}(g, d, p) = 2gd^2(3(g+p-1) + d(6g+11p+1))/3p \equiv O(4g^2d^3/p). \quad (4.14)$$

Recall that d and p denote the number of deleted rows and the number of processors, respectively. The upper-triangular $n \times n$ matrix R is partitioned in $g \times g$ blocks, where $n = gd$.

Algorithm 7 The parallel block downdating of the QRD with cyclic distribution on p processors.

- 1: Let R be block-partitioned as in (4.7), where g is a multiple of p .
 - 2: Allocate $Q_{1,1}$ and Z to all processors.
 - 3: Allocate $R^{(j)} = (R_{:,j} \ R_{:,j+p} \ \cdots \ R_{:,j+g-p})$ to processor P_j ($j = 1, \dots, p$).
 - 4: **each processor** P_j ($j = 1, \dots, p$) **do in parallel:**
 - 5: **for** $i = 1, \dots, g$ **do**
 - 6: Compute the row-permuted QRD $G_i^T \begin{pmatrix} W^{(i)} \\ Z^{(i-1)} \end{pmatrix} = \begin{pmatrix} 0 \\ Z^{(i)} \end{pmatrix}$
 - 7: Compute $G_i^T \begin{pmatrix} R_{i,[i/p]}^{(j)} & \cdots & R_{i,g/p}^{(j)} \\ 0 & \cdots & E_g^{(i-1)} \end{pmatrix} = \begin{pmatrix} \widehat{R}_{i,[i/p]}^{(j)} & \cdots & \widehat{R}_{i,g/p}^{(j)} \\ E_i^{(i)} & \cdots & E_g^{(i)} \end{pmatrix}$
 - 8: **end for**
 - 9: **for** $i = j, j+p, \dots, j+g-p$ **do** Compute the QRD $Q_{i,j}^T \widehat{R}_{i,[i/p]}^{(j)} = \widetilde{R}_{i,[i/p]}^{(j)}$ **end do**
 - 10: Send $Q_{i,j}^T$ to processors P_r ($i = j, j+p, \dots, j+g-p$; $r = 1, \dots, p$ and $r \neq j$).
 - 11: Receive $Q_{k,r}^T$ from processors P_r ($k = r, r+p, \dots, r+g-p$; $r = 1, \dots, p$ and $r \neq j$).
 - 12: **for** $i = 1, \dots, g$ **do**
 - 13: $r := (i-1) \bmod p + 1$
 - 14: Compute $Q_{i,r}^T \begin{pmatrix} \widehat{R}_{i,[(i+1)/p]}^{(j)} & \cdots & \widehat{R}_{i,g/p}^{(j)} \\ \widetilde{R}_{i,[(i+1)/p]}^{(j)} & \cdots & \widetilde{R}_{i,g/p}^{(j)} \end{pmatrix}$
 - 15: **end for**
-

Table 4.1 shows the execution and communication times for Algorithm 7 for some g and d . The execution times of the sequential algorithm and efficiency of the parallel algorithm are also presented. These results show that the efficiency of Algorithm 7 degrades as the number of processors increases because of the high communication costs. The computation time increases more than the communication overheads for increasing g , and thus, the efficiency of the algorithm increases. Thus, Algorithm 7 can achieve high efficiency for relatively large d and exceptionally large g with respect to the number of processors.

In order to eliminate the inter-processor communications in the second stage of Algorithm 7 the diagonal block matrices of R , i.e. $R_{1,1}, \dots, R_{g,g}$ in (4.7) are allocated to each of the processors. Thus, the factorizations (4.13) are computed locally by each processor. This has the disadvantage of duplicating data, i.e. the diagonal of R is allocated twice to the processors, but it has the advantage of eliminating communication costs. A drawback of this strategy, as well of Algorithm 7, is that the matrices allocated to the processors have unequal numbers of non-zero blocks. The load increases with increasing the processor

Table 4.1: Execution times, communication times and efficiencies of Algorithm 7.

g	d	2 processors				4 processors			8 processors			16 processors		
		Serial	Time	Comm	Eff	Time	Comm	Eff	Time	Comm	Eff	Time	Comm	Eff
160	10	2.11	1.13	0.01	0.93	0.56	0.01	0.94	1.37	1.11	0.19	1.70	1.55	0.08
240	10	4.91	2.49	0.01	0.99	1.31	0.01	0.94	1.92	1.29	0.32	2.52	2.17	0.12
320	10	8.66	4.37	0.01	0.99	2.24	0.01	0.97	2.41	1.32	0.45	2.78	2.35	0.19
480	10	10.97	5.55	0.01	0.99	2.82	0.01	0.97	2.66	2.12	0.52	4.26	3.52	0.16
640	10	13.78	6.98	0.01	0.99	3.51	0.02	0.98	5.02	3.20	0.34	5.30	4.35	0.16
32	50	5.39	2.73	0.01	0.99	1.65	0.02	0.82	1.82	0.78	0.37	5.26	4.50	0.06
48	50	11.78	5.92	0.01	0.99	3.29	0.04	0.90	3.19	1.18	0.46	6.58	5.37	0.11
80	50	32.86	16.68	0.01	0.99	8.69	0.06	0.95	6.81	1.47	0.60	7.39	5.78	0.28
160	50	153.31	77.41	0.04	0.99	40.21	0.39	0.95	23.60	2.08	0.81	34.23	22.97	0.28
240	50	586.74	301.98	0.09	0.97	155.00	0.48	0.95	81.44	4.31	0.90	72.75	32.14	0.50
320	50	736.11	371.64	0.13	0.99	189.70	0.53	0.97	100.41	5.17	0.92	86.67	36.95	0.53
16	100	9.98	5.24	0.01	0.95	3.37	0.05	0.74	3.27	0.91	0.38	8.39	6.72	0.07
32	100	35.33	18.32	0.01	0.96	10.12	0.06	0.87	7.75	1.07	0.57	12.65	8.13	0.17
64	100	163.74	82.44	0.01	0.99	44.20	0.08	0.93	27.93	3.12	0.73	28.61	14.04	0.36
128	100	642.34	323.41	0.06	0.99	169.73	0.16	0.95	93.35	6.23	0.86	94.71	46.27	0.42
256	100	3082.61	1558.44	0.14	0.99	813.44	1.08	0.95	424.16	9.91	0.91	301.10	86.32	0.64

index, so the execution time is dominated by the last processor which holds the matrices with maximum density. An improved distribution for the matrices can offset this drawback.

An efficient load-balanced distribution allocates the the non-zero blocks of R equally to each processor. The first $g/2$ block columns of R are allocated using the column cyclic distribution scheme of Algorithm 7. The remaining $g/2$ block columns of R are distributed using a reverse (counting backwards) column cyclic allocation scheme. That is, the block column $R_{:,i}$ is allocated to the processor P_{λ_i} , where

$$\lambda_i = \begin{cases} (i-1) \bmod p + 1 & \text{if } i = 1, \dots, g/2, \\ p - (i-1) \bmod p & \text{if } i = g/2 + 1, \dots, g. \end{cases}$$

This distribution results in the processor P_j being allocated the $n \times n/p$ matrix

$$R^{(j)} = (R_{:,j} \quad R_{:,j+p} \quad \cdots \quad R_{:,g+1-j-p} \quad R_{:,g+1-j}).$$

Figure 4.2 shows the distribution of the matrices in (4.5) on the processors, with $p = 4$ and $g = 16$, $Q^* = (Q_{1,1} \quad Z^T)^T$ and $R^* = (R_{1,1} \quad \cdots \quad R_{g,g} \quad 0)$. The shaded blocks are the

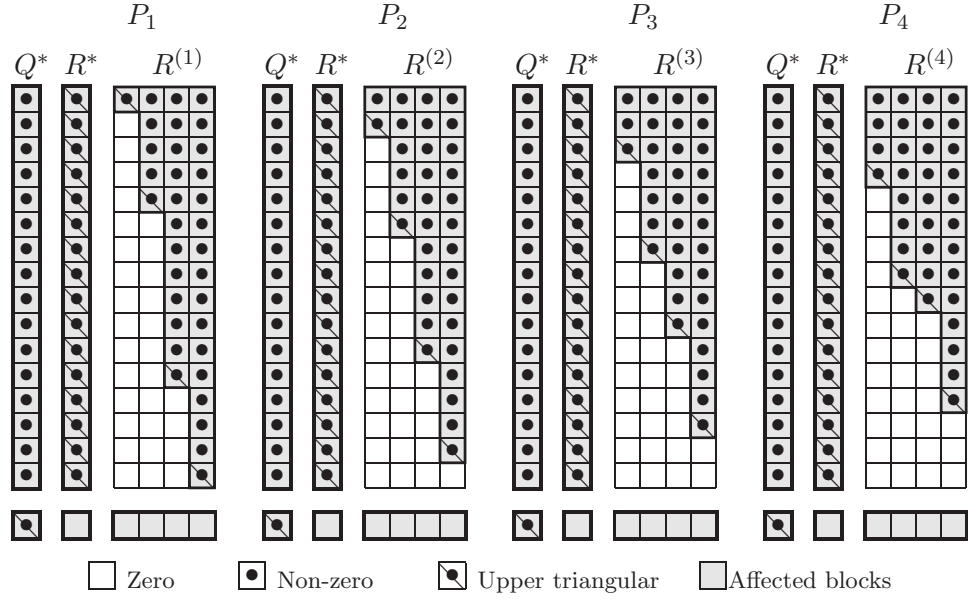


Figure 4.2: The modified cyclic distribution of the matrices on 4 processors, when $g = 16$.

matrices affected during the computation. Note that now each processor operates on the same number of non-zero blocks.

This version of the parallel downdating algorithm comprises of two stages which compute (4.8) and (4.10). In the first stage each processor computes locally the factorization (4.8a) and then, for the computation of (4.8b), it applies G_i^T to its allocated block submatrices of R and E . In the second stage, each processor computes the re-triangularization (4.10a) of the diagonal matrices $\widehat{R}_{i,i}$ ($i = 1, \dots, g$) and then updates locally the allocated matrices $\widehat{R}^{(j)}$ ($j = 1, \dots, p$). Algorithm 8 summarizes the steps of this approach. Note that the main computations are performed by the loop at lines 9-22. The conditional statement at lines 12-20 avoids computations on zero blocks after the factorization (4.8a) at line 11 has been computed. The second conditional statement at lines 15-19 prevents the application of the orthogonal matrix Q_i^T to the matrix $\widehat{R}_{i,i}$ when the processor P_j has been allocated $R_{i,i}$ as part of $R^{(j)}$.

Algorithm 8 does not require inter-processor communication and thus, its theoretical complexity is just the computational complexity. That is,

$$T_{P_2}(g, d, p) = 2gd^2(3(g + p) + d(6g + 16p + 3))/3p \equiv O(4g^2d^3/p). \quad (4.15)$$

Notice that the latter exceeds the complexity of Algorithm 7 in (4.14) by $2gd^2(3 + 5pd +$

Algorithm 8 The modified parallel block downdating of the QRD with cyclic distribution.

- 1: Let R be block-partitioned as in (4.7), where g is a multiple of $2p$.
 - 2: Allocate $Q_{1,1}$, Z and $R_{i,i}$ to all processors ($i = 1, \dots, g$).
 - 3: Allocate $R^{(j)} = (R_{:,j} \ R_{:,j+p} \ \cdots \ R_{:,g+1-j-p} \ R_{:,g+1-j})$ to processor P_j ($j = 1, \dots, p$).
 - 4: **each processor** P_j ($j = 1, \dots, p$) **do in parallel:**
 - 5: **for all** $i = 1, \dots, g/p$ **do**
 - 6: **if** $(i + j - 1 \bmod p = 0)$ **then** $\delta_i^{(j)} := 1$ **else** $\delta_i^{(j)} := p + 1$ **end if**
 - 7: **end for**
 - 8: Let $\lambda_j := 0$ and $\sigma_j := j - 1$
 - 9: **for** $i = g, \dots, 1$ **do**
 - 10: **if** $\sigma_j = 0$ **then** $\lambda_j := \lambda_j + 1$ and $\sigma_j := \delta_{\lambda_j}^{(j)}$ **end if**
 - 11: Compute the row-permuted QRD $G_i^T \begin{pmatrix} W^{(i)} \\ Z^{(i)} \end{pmatrix} = \begin{pmatrix} 0 \\ Z^{(i-1)} \end{pmatrix}$
 - 12: **if** $\lambda_j \neq 0$ **then**
 - 13: Comp. $G_i^T \begin{pmatrix} R_{i,i} & R_{i,g/p+1-\lambda_j}^{(j)} & \cdots & R_{i,g/p}^{(j)} \\ 0 & 0 & \cdots & E_g^{(i)} \end{pmatrix} = \begin{pmatrix} \widehat{R}_{i,i} & \widehat{R}_{i,g/p+1-\lambda_j}^{(j)} & \cdots & \widehat{R}_{i,g/p}^{(j)} \\ T & E_i^{(i-1)} & \cdots & E_g^{(i-1)} \end{pmatrix}$
 - 14: Compute the QRD $Q_i^T \widehat{R}_{i,i} = \widetilde{R}_{i,i}$
 - 15: **if** $\sigma_j = \delta_{\lambda_j}^{(j)}$ **then**
 - 16: Compute $Q_i^T \begin{pmatrix} \widehat{R}_{i,g/p+2-\lambda_j}^{(j)} & \cdots & \widehat{R}_{i,g/p}^{(j)} \end{pmatrix} = \begin{pmatrix} \widetilde{R}_{i,g/p+2-\lambda_j}^{(j)} & \cdots & \widetilde{R}_{i,g/p}^{(j)} \end{pmatrix}$
 - 17: **else**
 - 18: Compute $Q_i^T \begin{pmatrix} \widehat{R}_{i,g/p+1-\lambda_j}^{(j)} & \cdots & \widehat{R}_{i,g/p}^{(j)} \end{pmatrix} = \begin{pmatrix} \widetilde{R}_{i,g/p+1-\lambda_j}^{(j)} & \cdots & \widetilde{R}_{i,g/p}^{(j)} \end{pmatrix}$
 - 19: **end if**
 - 20: **end if**
 - 21: Let $\sigma_j := \sigma_j - 1$
 - 22: **end for**
-

$2d)/3p$ flops. However, Algorithm 8 unlike Algorithm 7, has no inter-processor communication cost. From (4.12) and (4.15) it follows that the efficiency of Algorithm 8 approaches one for very large g , i.e. $\lim_{g \rightarrow \infty} T_S(g, d)/(p \times T_{P_2}(g, d, p)) \approx 1$.

Table 4.2 shows the execution times and actual (and in brackets the theoretical) efficiencies of Algorithm 8 for some g and d . The theoretical complexity is confirmed by the experimental results. Comparing the results of Tables 4.1 and 4.2 it will be observed that Algorithm 8 outperforms Algorithm 7 for larger numbers of processors. Furthermore,

Algorithm 8 is scalable, i.e. the efficiency remains the same when the number of block columns g and the number of processors p are doubled.

Table 4.2: Execution times (sec.) and efficiencies of Algorithm 8.

g	d	2 processors		4 processors		8 processors		16 processors		
		Serial	Time	Eff.	Time	Eff.	Time	Eff.	Time	Eff.
160	10	2.11	1.12	0.94 (0.98)	0.58	0.91 (0.95)	0.32	0.82 (0.89)	0.27	0.49 (0.80)
240	10	4.91	2.56	0.96 (0.98)	1.33	0.92 (0.96)	0.74	0.83 (0.93)	0.48	0.64 (0.86)
320	10	8.66	4.38	0.99 (0.99)	2.32	0.93 (0.97)	1.27	0.85 (0.94)	0.79	0.69 (0.89)
480	10	10.97	5.55	0.99 (0.99)	2.90	0.95 (0.98)	1.59	0.86 (0.96)	0.85	0.81 (0.92)
640	10	13.78	6.94	0.99 (0.99)	3.56	0.97 (0.99)	1.90	0.91 (0.97)	1.01	0.86 (0.94)
32	50	5.39	3.08	0.88 (0.89)	1.85	0.73 (0.78)	1.12	0.60 (0.63)	0.83	0.41 (0.45)
48	50	11.78	6.62	0.89 (0.92)	3.68	0.80 (0.84)	2.22	0.66 (0.71)	1.41	0.52 (0.55)
80	50	32.86	17.41	0.94 (0.95)	9.45	0.87 (0.90)	5.28	0.78 (0.80)	3.19	0.64 (0.66)
160	50	153.31	79.49	0.96 (0.97)	41.71	0.92 (0.94)	22.39	0.86 (0.89)	12.45	0.77 (0.80)
240	50	586.74	303.68	0.97 (0.98)	158.53	0.93 (0.96)	81.46	0.90 (0.92)	43.73	0.84 (0.85)
320	50	736.11	372.96	0.99 (0.99)	191.98	0.96 (0.97)	99.11	0.93 (0.94)	53.06	0.87 (0.89)
16	100	9.98	6.33	0.79 (0.81)	3.95	0.63 (0.65)	2.68	0.47 (0.47)	2.13	0.29 (0.30)
32	100	35.33	19.94	0.89 (0.89)	11.78	0.75 (0.78)	6.96	0.63 (0.63)	5.03	0.44 (0.45)
64	100	163.74	87.87	0.93 (0.94)	47.32	0.87 (0.87)	27.12	0.75 (0.77)	17.78	0.58 (0.61)
128	100	642.34	331.60	0.97 (0.97)	176.68	0.91 (0.93)	93.20	0.86 (0.87)	53.98	0.74 (0.76)
256	100	3082.61	1592.49	0.97 (0.98)	821.98	0.94 (0.96)	428.01	0.90 (0.93)	229.60	0.84 (0.86)

4.4 Conclusions

Two parallel strategies for downdating the QR decomposition have been proposed. These are parallel versions of a recently proposed sequential algorithm which efficiently exploits the triangular structure of the matrices and is rich in BLAS-3 operations [47]. The algorithms have been implemented on a shared memory SUN Enterprise 10 000 (16 CPU UltraSPARC of 400 MHz) using the single-program multiple-data paradigm. Theoretical and experimental results for both strategies have been presented and analyzed. The performance of the first algorithm is degraded by the communication costs which increases significantly with the number of processors. The second strategy has no inter-processor communications, but has some duplicated computations. It is found to outperform the first

parallel strategy (Algorithm 7) when the number of processors is not small. Generally, the second parallel strategy (Algorithm 8) achieves perfect load balancing, scalability and can reach an efficiency close to one when the number of block columns g is reasonably big. In addition, the theoretical complexities have confirmed the computational experiments.

Currently, within the context of cross-validation, the adaptation of the parallel Algorithm 8 to compute a series of downdating least squares problems is being considered.

Chapter 5

Efficient algorithms for estimating the general linear model

Abstract:

Computationally efficient serial and parallel algorithms for estimating the general linear model are proposed. The sequential block-recursive algorithm is an adaptation of a known Givens strategy that has as a main component the Generalized QR Decomposition. The proposed algorithm is based on orthogonal transformations and exploits the triangular structure of the Cholesky factor of the variance-covariance matrix. Specifically, it computes the estimator of the general linear model by solving recursively a series of smaller and smaller generalized linear least squares problems. The new algorithm is found to outperform significantly the corresponding LAPACK routine. A parallel version of the new sequential algorithm which utilizes an efficient distribution of the matrices over the processors and has low inter-processor communication is developed. The theoretical computational complexity of the parallel algorithms is derived and analyzed. Experimental results are presented which confirm the theoretical analysis. The parallel strategy is found to be scalable and highly efficient for estimating large-scale general linear estimation problems.

¹This chapter is a reprint of the paper: P. Yanev and E.J. Kontoghiorghes. Efficient algorithms for estimating the general linear model. *Parallel Computing*, 2004 (Submitted).

5.1 Introduction

Consider the General Linear Model (GLM)

$$y = X\beta + \varepsilon, \quad \varepsilon \sim (0, \sigma^2\Omega), \quad (5.1)$$

where $y \in \mathbb{R}^m$ is the response vector, $X \in \mathbb{R}^{m \times n}$ is the full rank exogenous data matrix, $\beta \in \mathbb{R}^n$ are the coefficients to be estimated and $\varepsilon \in \mathbb{R}^m$ is the noise with zero mean and variance-covariance matrix $\sigma^2\Omega$. It is assumed that the matrix $\Omega = CC^T$, $C \in \mathbb{R}^{m \times m}$ is known, upper triangular and non-singular, while the scalar σ is unknown [30]. The basic linear unbiased estimator (BLUE) of β is obtained by solving the generalized linear least squares problem (GLLSP)

$$\underset{u, \beta}{\operatorname{argmin}} u^T u \quad \text{subject to} \quad y = X\beta + Cu, \quad (5.2)$$

where u is a random vector defined by $Cu = \varepsilon$, i.e. $u \sim (0, \sigma^2 I_m)$. The GLLSP can be solved using the generalized QR decomposition (GQRD) of $\tilde{X} \equiv (X \ y)$ and C :

$$Q^T \tilde{X} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} \equiv \begin{pmatrix} R & \tilde{y} \\ 0 & \eta \\ 0 & 0 \end{pmatrix} \begin{matrix} n & 1 \\ n \\ 1 \\ m-n-1 \end{matrix} \quad (5.3a)$$

and

$$(Q^T C)\Pi = U^T = \begin{pmatrix} U_{1,1} & r & U_{1,2} \\ 0 & \delta & g \\ 0 & 0 & U_{2,2} \end{pmatrix} \begin{matrix} n & 1 & m-n-1 \\ n \\ 1 \\ m-n-1 \end{matrix}. \quad (5.3b)$$

Here \tilde{R} and U are upper triangular matrices of orders n and m , respectively, and $Q, \Pi \in \mathbb{R}^{m \times m}$ are orthogonal [5, 20]. The GLLSP (5.2) is equivalent to

$$\underset{u, \beta}{\operatorname{argmin}} \|\Pi^T u\|^2 \quad \text{subject to} \quad Q^T y = Q^T X\beta + Q^T C\Pi\Pi^T u,$$

which can be written also as

$$\underset{v_1, v, v_2, \beta}{\operatorname{argmin}} (\|v_1\|^2 + v^2 + \|v_2\|^2) \quad \text{subject to} \quad \begin{cases} \tilde{y} = R\beta + U_{1,1}v_1 + rv + U_{1,2}v_2, \\ \eta = \delta v + gv_2, \\ 0 = U_{2,2}v_2, \end{cases} \quad (5.4)$$

where the vector $u^T\Pi$ is partitioned as $(v_1^T \ v \ v_2^T)$ and $\|\cdot\|$ denotes the Euclidean norm. The values of $v_2 = 0$ and $v = \eta/\delta$ can be derived from the last two constraints in (5.4). Then, setting $v_1 = 0$, the BLUE of β is obtained by solving

$$R\beta = \tilde{y} - \eta r/\delta. \quad (5.5)$$

The triangular structure of C (which is assumed to be available) facilitates the development of efficient algorithms for solving the GLLSP [25, 35, 36, 37]. In this work block recursive sequential and parallel strategies which exploit the structure of the matrices are proposed. The new methods are rich in BLAS-3 operations and solve a series of reduced size GLLSPs.

The algorithms have been implemented on 32 CPUs IBM's p690+ high-end compute node with 27 GB distributed memory. The communications between the processors are realized using the MPI library. The performance of the algorithms has been evaluated experimentally. In addition, the theoretical complexities of the algorithms in number of flops (floating point operations) are also presented. The serial block-Givens strategy is then described and compared with the existing LAPACK routine for estimating the GLM. In section 3 the parallel algorithm is considered and the theoretical and computational results are presented. In section 4 some conclusions are drawn.

5.2 Serial block Givens strategy

An efficient sequential algorithm based on Givens rotations for estimating the GLM has been proposed by Paige [35]. Here a block version based on this sequential strategy is investigated [47]. Consider the GLLSP (5.2), where the matrices $\tilde{X} \equiv (X \ y)$ and C are

partitioned, respectively, as

$$\tilde{X} = \begin{pmatrix} n & 1 \\ X_1 & y_1 \\ X_2 & y_2 \\ \vdots & \vdots \\ X_k & y_k \end{pmatrix} \begin{matrix} n \\ n \\ \vdots \\ n \end{matrix} \quad \text{and} \quad C = \begin{pmatrix} n & n & \cdots & n \\ C_{1,1} & C_{1,2} & \cdots & C_{1,k} \\ & C_{2,2} & \cdots & C_{2,k} \\ & & \ddots & \vdots \\ & & & C_{k,k} \end{pmatrix} \begin{matrix} n \\ n \\ \vdots \\ n \end{matrix}. \quad (5.6)$$

Here $C_{i,i}$ ($i = 1, \dots, k$) are upper triangular where, for simplicity, it is assumed that $m = kn$. The sequential block algorithm computes the solution of the GLLSP (5.2) in k steps. The GQRD (5.3) is computed during the first $(k-1)$ steps. For $j = k-i$ the i th ($i = 1, \dots, k-1$) step computes the smaller GQRD of

$$\begin{pmatrix} X_j & y_j^{(i)} \\ \tilde{X}_{j+1} & y_{j+1}^{(i)} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} C_{j,j} & \tilde{C}_{j,j+1} \\ 0 & \tilde{C}_{j+1,j+1} \end{pmatrix}, \quad (5.7)$$

where $y_{k-1}^{(1)} = y_{k-1}$, $y_k^{(1)} = y_k$, $\tilde{X}_k = X_k$, $\tilde{C}_{k-1,k} = C_{k-1,k}$ and $\tilde{C}_{k,k} = C_{k,k}$. That is, initially the QRD of the first matrix in (5.7) is computed by:

$$Q_i^T \begin{pmatrix} X_j & y_j^{(i)} \\ \tilde{X}_{j+1} & y_{j+1}^{(i)} \end{pmatrix} = \begin{pmatrix} \tilde{X}_j & \tilde{y}_j^{(i)} \\ 0 & \eta_i \\ 0 & 0 \end{pmatrix} \begin{matrix} n \\ 1 \\ n-1 \end{matrix}, \quad (5.8)$$

where $Q_i \in \mathbb{R}^{2n \times 2n}$ is orthogonal and \tilde{X}_j is upper triangular. Then, the orthogonal matrix Q_i^T is applied from the left of the second matrix in (5.7) which is then re-triangularized from the right, i.e.

$$\left(Q_i^T \begin{pmatrix} C_{j,j} & \tilde{C}_{j,j+1} \\ 0 & \tilde{C}_{j+1,j+1} \end{pmatrix} \right) \Pi_i = \begin{pmatrix} n & 1 & n-1 \\ \tilde{C}_{j,j} & r_j^{(i)} & \bar{C}_{j,j+1} \\ 0 & \delta_i & g_i \\ 0 & 0 & \bar{C}_{j+1,j+1} \end{pmatrix} \begin{matrix} n \\ 1 \\ n-1 \end{matrix}. \quad (5.9)$$

Here Π_i is orthogonal and of order $2n$; $\tilde{C}_{j,j}$ and $\bar{C}_{j+1,j+1}$ are upper triangular. Once the i th GQRD of (5.7) has been computed, Π_i is applied from the right of the affected j th and

$(j + 1)$ th block-columns of C , i.e. the product

$$\begin{pmatrix} n & n \\ C_{1,j} & \tilde{C}_{1,j+1} \\ C_{2,j} & \tilde{C}_{2,j+1} \\ \vdots & \vdots \\ C_{j-1,j} & \tilde{C}_{j-1,j+1} \end{pmatrix} \Pi_i = \begin{pmatrix} n & 1 & n-1 \\ \tilde{C}_{1,j} & r_1^{(i)} & \bar{C}_{1,j+1} \\ \tilde{C}_{2,j} & r_2^{(i)} & \bar{C}_{2,j+1} \\ \vdots & \vdots & \vdots \\ \tilde{C}_{j-1,j} & r_{j-1}^{(i)} & \bar{C}_{j-1,j+1} \end{pmatrix} \begin{pmatrix} n \\ n \\ \vdots \\ n \end{pmatrix} \quad (5.10)$$

is computed. Note that this is the most time consuming task in each step of the computation of the GQRD (5.3), especially when k is large, i.e. when $m \gg n$.

Now let $(u_{i-1}^T \tilde{u}_{i-1}^T) \Pi_i$ be partitioned conformably as $(u_i^T \tilde{u}_i^T v_i \bar{u}_i^T)$, where $(u_0^T \tilde{u}_0^T) \equiv u$. The GLLSP (5.2) after the i th step of computing the GQRD (5.3) can be written as

$$\begin{aligned} & \underset{u_i, \tilde{u}_i, v_i, \bar{u}_i, \beta}{\operatorname{argmin}} (\|u_i\|^2 + \|\tilde{u}_i\|^2 + v_i^2 + \|\bar{u}_i\|^2) \quad \text{subject to} \\ & \begin{pmatrix} y_{1:j-1}^{(i)} \\ \tilde{y}_j^{(i)} \\ \eta_i \\ 0 \end{pmatrix} = \begin{pmatrix} X_{1:j-1} \\ \tilde{X}_j \\ 0 \\ 0 \end{pmatrix} \beta + \begin{pmatrix} C_{1:j-1,1:j-1} & \tilde{C}_{1:j-1,j} & r_{1:j-1}^i & \bar{C}_{1:j-1,j+1} \\ 0 & \tilde{C}_{j,j} & r_j^i & \bar{C}_{j,j+1} \\ 0 & 0 & \delta_i & g_i \\ 0 & 0 & 0 & \bar{C}_{j+1,j+1} \end{pmatrix} \begin{pmatrix} u_i \\ \tilde{u}_i \\ v_i \\ \bar{u}_i \end{pmatrix}. \end{aligned}$$

From this it follows that $\bar{C}_{j+1,j+1} \bar{u}_i = 0$, i.e. $\bar{u}_i = 0$ and $\eta_i = \delta_i v_i + g_i \bar{u}_i$, i.e. $v_i = \eta_i / \delta_i$. Thus, the GLLSP (5.2) is equivalent to the reduced GLLSP

$$\underset{u_i, \tilde{u}_i, \beta}{\operatorname{argmin}} (\|u_i\|^2 + \|\tilde{u}_i\|^2) \quad \text{s.t.} \quad \begin{pmatrix} y_{1:j-1}^{(i+1)} \\ y_j^{(i+1)} \end{pmatrix} = \begin{pmatrix} X_{1:j-1} \\ \tilde{X}_j \end{pmatrix} \beta + \begin{pmatrix} C_{1:j-1,1:j-1} & \tilde{C}_{1:j-1,j} \\ 0 & \tilde{C}_{j,j} \end{pmatrix} \begin{pmatrix} u_i \\ \tilde{u}_i \end{pmatrix},$$

where

$$\begin{pmatrix} y_{1:j-1}^{(i+1)} \\ y_j^{(i+1)} \end{pmatrix} = \begin{pmatrix} y_{1:j-1}^{(i)} \\ \tilde{y}_j^{(i)} \end{pmatrix} - v_i \begin{pmatrix} r_{1:j-1}^{(i)} \\ r_j^{(i)} \end{pmatrix}. \quad (5.11)$$

Equation (5.11) shows the size-reduction of the GLLSP after each step of the computation. Following the completion of the $(k - 1)$ th step the final, smallest GLLSP has the form:

$$\underset{\tilde{u}_{k-1}, v_{k-1}, \beta}{\operatorname{argmin}} (\|\tilde{u}_{k-1}\|^2 + \|v_{k-1}\|^2) \quad \text{s.t.} \quad \begin{pmatrix} \tilde{y}^{(k-1)} \\ \eta_{k-1} \end{pmatrix} = \begin{pmatrix} \tilde{X}_1 \\ 0 \end{pmatrix} \beta + \begin{pmatrix} \tilde{C}_{1,1} & r_1^{(k-1)} \\ 0 & \delta_{k-1} \end{pmatrix} \begin{pmatrix} \tilde{u}_{k-1} \\ v_{k-1} \end{pmatrix}$$

or

$$\underset{\tilde{u}_{k-1}, \beta}{\operatorname{argmin}} \|\tilde{u}_{k-1}\|^2 \quad \text{s.t.} \quad y_1^{(k)} = \tilde{X}_1 \beta + \tilde{C}_{1,1} \tilde{u}_{k-1},$$

where \tilde{X}_1 and $\tilde{C}_{1,1}$ are upper triangular and are derived from the QRD (5.8) and RQD (5.9), respectively, and $y_1^{(k)} = \tilde{y}_1^{(k-1)} - \eta_{k-1} r_1^{(k-1)} / \delta_{k-1}$. Thus, the k th step computes the BLUE of β by setting $\tilde{u}_{k-1} = 0$ and solving the upper triangular system $\tilde{X}_1 \beta = y_1^{(k)}$.

Algorithm 9 The sequential block Givens algorithm for solving the GLLSP (5.2).

- 1: Let \tilde{X} and C in (5.3) be partitioned as in (5.6), where $m = kn$
 - 2: Let $y_{k-1}^{(1)} = y_{k-1}$, $y_k^{(1)} = y_k$, $\tilde{X}_k = X_k$, $\tilde{C}_{k-1,k} = C_{k-1,k}$ and $\tilde{C}_{k,k} = C_{k,k}$.
 - 3: **for** $i = 1, \dots, k-1$ **do**
 - 4: Set $j := k - i$
 - 5: Compute the GQRD of $\begin{pmatrix} X_j & y_j^{(i)} \\ \tilde{X}_{j+1} & y_{j+1}^{(i)} \end{pmatrix}$ and $\begin{pmatrix} C_{j,j} & \tilde{C}_{j,j+1} \\ 0 & \tilde{C}_{j+1,j+1} \end{pmatrix}$ as in (5.8) and (5.9)
 - 6: **if** $i \neq k-1$ **then**
 - 7: Compute $\begin{pmatrix} C_{1:j-1,j} & \tilde{C}_{1:j-1,j+1} \end{pmatrix} \Pi_i = \begin{pmatrix} \tilde{C}_{1:j-1,j} & r_{1:j-1}^{(i)} & \bar{C}_{1:j-1,j+1} \end{pmatrix}$
 - 8: **end if**
 - 9: Update the vector y : $\begin{pmatrix} y_{1:j-1}^{(i+1)} \\ y_j^{(i+1)} \end{pmatrix} = \begin{pmatrix} y_{1:j-1}^{(i)} \\ \tilde{y}_j^{(i)} \end{pmatrix} - \frac{\eta_i}{\delta_i} \begin{pmatrix} r_{1:j-1}^{(i)} \\ r_j^{(i)} \end{pmatrix}$
 - 10: **end for**
 - 11: Solve $\tilde{X}_1 \beta = y_1^{(k)}$
-

Algorithm 9 summarizes the steps of this block Givens strategy for estimating the GLM. For the factorizations (5.8) and (5.9) Householder transformations are employed. The orthogonal matrices Q_i and Π_i ($i = 1, \dots, k-1$) are not explicitly constructed. The theoretical complexity of this algorithm is given by:

$$T_{\text{BG}}(m, n) \approx 4m^2n + 14mn^2 - 18n^3. \quad (5.12)$$

Table 5.1 shows the execution times in seconds and the theoretical complexities in number of flops of Block-Givens Algorithm 9 (BG) and the LAPACK (LP) routine DGGGLM which estimates the GLM for some values of n and k , where $m = kn$ [2]. The theoretical complexity of LAPACK is given by:

$$T_{\text{LP}}(m, n) \approx (4m^3 + 12m^2n - 2n^3)/3. \quad (5.13)$$

Note that, theoretically, Algorithm 9 is approximately $m/3n$ times faster than the LAPACK routine, which is confirmed by the experimental results. This improvement is due to the fact that the Algorithm 9 exploits the triangular structure of the large and computationally expensive matrix C in (5.2), while the LAPACK routine assumes that C is dense.

The experimental results (LP/BG) confirm the theoretical ones (T_{LP}/T_{BG}). There is a negligible discrepancy between the two ratios when k is big and n is relatively much smaller. This is due to the increasing overheads which occur from the frequent data exchanges of the submatrices in (5.6).

Table 5.1: Execution times (sec.) and theoretical results of Algorithm 9 and LAPACK.

n	25			50			100		
$k - 2$	128	256	384	64	128	192	32	64	96
LP	25.77	213.35	601.58	26.25	216.62	603.03	27.77	223.35	605.58
BG	0.76	2.88	6.59	1.44	5.32	9.71	2.62	10.56	19.04
LP/BG	33.91	74.08	91.29	18.23	40.72	62.10	10.60	21.15	31.81
T_{LP}/T_{BG}	42.69	85.35	128.01	21.38	42.69	64.02	10.77	21.39	32.04

5.3 Parallel algorithm

The computation of the product (5.10) is the most time consuming task in Algorithm 1. This cost can be reduced by applying the orthogonal matrices Π_i ($i = 1, \dots, k - 1$) to the block columns of C (see line 7 of Algorithm 9) in parallel. An efficient parallel algorithm requires a load-balanced distribution of the matrices over the processors and low inter-processor communication [48]. Let p denotes the number of processors and assume that $(k - 2)$ is a multiple of p , where $m = kn$.

Consider the partitioning of the matrices \tilde{X} and C as in (5.6). To achieve low inter-processor communication, the GQRDs computed in line 5 of Algorithm 9 are executed simultaneously by all processors. That is, the data matrix X , the last two block rows of y and the main, sub- and super-block diagonals of C are duplicated on each processor. The remaining $(k - 2)$ block-rows of C and y are allocated to the processors using a block-row cyclic distribution. Specifically, $C_{i,:}$ ($i = k - 2, \dots, 1$) is allocated to the processor P_{λ_i} , where $\lambda_i = p - (i - 1) \bmod p$. The same distribution scheme is used for the vector y . This distribution will result in the processor P_j ($j = 1, \dots, p$) being allocated the vector

$$\gamma^{(j)} = \left(y_{p+1-j}^T \quad y_{2p+1-j}^T \quad \cdots \quad y_{k-1-j}^T \right)^T \quad (5.14)$$

and the matrix

$$C^{(j)} = \left(C_{p+1-j,:}^T \quad C_{2p+1-j,:}^T \quad \cdots \quad C_{k-1-j,:}^T \right)^T, \quad (5.15)$$

where $\gamma^{(j)} \in \mathbb{R}^{n(k-2)/p \times 1}$ and $C^{(j)} \in \mathbb{R}^{n(k-2)/p \times m}$. Figure 5.1 shows the distribution of the matrices \tilde{X} and C over the processors, with $p = 4$ and $k = 18$. The shaded blocks indicate those copied to all processors. The blank, unshaded, blocks are null and are unaffected during the computation.

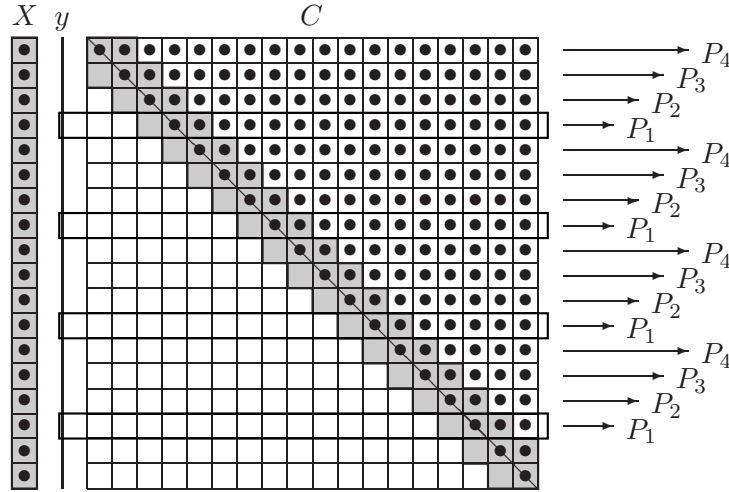


Figure 5.1: The row-block cyclic distribution of the matrices on 4 processors, when $k = 18$.

The parallel algorithm solves the GLLSP (5.2) in k steps. During the first $(k - 1)$ steps, all processors initially compute the same factorizations (5.8) and (5.9). Then each processor updates its allocated submatrix $C^{(j)}$ and subvector $\gamma^{(j)}$. Note that, at the i th step, each processor updates $\lceil q/p \rceil$ blocks, where $q = k - i - 1$. Thus, the processors have equal computational loads. When the local computations have been completed one processor, P_j say, sends one block from $C^{(j)}$ and $\gamma^{(j)}$, which are required for the next step, to the other processors P_r ($r = 1, \dots, p$ and $r \neq j$). That is, at each step only one processor broadcasts an $n \times n$ submatrix and an n -element subvector. This broadcast acts as a barrier-synchronization point for the processors before the next step commences. The parallel strategy is summarized in Algorithm 10. The broadcast performed by processor P_j is shown in lines 13-17 of the parallel algorithm.

The theoretical computational complexity of this algorithm is given by:

$$T_P(m, n, p) \approx (4m^2n - 16mn^2 + 16n^3)/p + 30mn^2 + 34n^3. \quad (5.16)$$

From (5.12) and (5.16) it follows that the computational efficiency of Algorithm 10 approaches one for very large m , i.e. $\lim_{m \rightarrow \infty} T_{BG}(m, n)/(p \times T_P(m, n, p)) \approx 1$. This does

Algorithm 10 The parallel algorithm for solving the GLLSP (5.2) on p processors.

- 1: Let \tilde{X} and C be partitioned as in (5.6), where $m = kn$ and $k - 2$ is a multiple of p .
 - 2: Allocate X , y_{k-1} , y_k , $C_{k,k}$, $C_{i,i}$ and $C_{i,i+1}$ ($i = 1, \dots, k - 1$) to all processors.
 - 3: Allocate $\gamma^{(j)}$ and $C^{(j)}$ as in (5.14) and (5.15), respectively, to processor P_j ($j = 1, \dots, p$).
 - 4: **each processor** P_j ($j = 1, \dots, p$) **do in parallel:**
 - 5: **for** $i = 1, \dots, k - 1$ **do**
 - 6: Set $t := k - i$
 - 7: Compute the GQRD of $\begin{pmatrix} X_t & y_t^{(i)} \\ \tilde{X}_{t+1} & y_{t+1}^{(i)} \end{pmatrix}$ and $\begin{pmatrix} C_{t,t} & \tilde{C}_{t,t+1} \\ 0 & \tilde{C}_{t+1,t+1} \end{pmatrix}$ as in (5.8) and (5.9).
 - 8: Compute $y_{k-i}^{(i+1)} = \tilde{y}_{k-i}^{(i)} - \eta_i / \delta_i r_k^{(i)} - i$.
 - 9: **if** $i \neq k - 1$ **then**
 - 10: Set $q := k - i - 1$
 - 11: Compute $C_{1:n[q/p], nq+1:nq+2n}^{(j)} = C_{1:n[q/p], nq+1:nq+2n}^{(j)} P_i$.
 - 12: Compute $\gamma_{1:n[q/p]}^{(j)} = \gamma_{1:n[q/p]}^{(j)} - v_i C_{1:n[q/p], nq+n+1}^{(j)}$.
 - 13: **if** $j = (i - 1) \bmod p + 1$ **then**
 - 14: Send $y_{t-1}^{(i+1)}$ and $\tilde{C}_{t-1,t}$ to P_r , where $r = 1, \dots, p$ and $r \neq j$.
 - 15: **else**
 - 16: Receive $y_{t-1}^{(i+1)}$ and $\tilde{C}_{t-1,t}$ from P_r , where $r = (i - 1) \bmod p + 1$.
 - 17: **end if**
 - 18: **end if**
 - 19: **end for**
 - 20: Solve $\tilde{X}_1 \beta = y_1^{(k)}$
-

not take into account, however, the inter-processor communication. Table 5.2 shows the execution times and actual (and in brackets the theoretical) efficiency of Algorithm 10 for some μ and n , where $\mu = m/n - 2$. The experimental results confirm the theoretical complexities. Note that, the communication time increases with the number of the processors which affects the efficiency of the algorithm for small size problems. Furthermore, Algorithm 10 is scalable. That is, the efficiency remains constant when the size of the problem m , and consequently μ , is multiplied by $\sqrt{2}$ and the number of the processors p is doubled.

Table 5.2: Execution times (sec.) and efficiency of Algorithm 10.

		Algorithm 10											
		2 processors			4 processors		8 processors		16 processors		32 processors		
μ	n	Serial	Time	Eff.	Time	Eff.	Time	Eff.	Time	Eff.	Time	Eff.	
128	25	0.76	0.41	.93 (.95)	0.24	.79 (.86)	0.16	.59 (.72)	0.13	.37 (.54)	0.19	.13 (.37)	
256	25	2.88	1.54	.94 (.97)	0.82	.88 (.92)	0.50	.72 (.83)	0.36	.50 (.70)	0.38	.24 (.53)	
384	25	6.59	3.47	.95 (.98)	1.75	.94 (.95)	1.08	.76 (.88)	0.63	.65 (.78)	0.72	.29 (.63)	
512	25	11.95	6.09	.98 (.99)	3.17	.94 (.96)	1.81	.83 (.91)	1.12	.67 (.82)	1.01	.37 (.69)	
640	25	18.77	9.51	.99 (.99)	4.82	.97 (.97)	2.69	.87 (.92)	1.50	.78 (.85)	1.28	.46 (.74)	
768	25	26.49	13.54	.98 (.99)	6.82	.97 (.97)	3.75	.88 (.94)	2.02	.82 (.87)	1.61	.51 (.77)	
896	25	36.23	18.51	.98 (.99)	9.35	.97 (.98)	4.96	.91 (.95)	2.65	.85 (.89)	1.94	.58 (.80)	
64	50	1.44	0.79	.91 (.90)	0.49	.73 (.76)	0.36	.50 (.57)	0.31	.29 (.39)	0.34	.13 (.23)	
128	50	5.32	2.83	.94 (.95)	1.58	.84 (.86)	0.97	.69 (.72)	0.77	.43 (.54)	0.86	.19 (.37)	
192	50	9.71	5.18	.94 (.96)	2.76	.88 (.90)	1.88	.65 (.79)	1.26	.48 (.64)	1.27	.24 (.36)	
256	50	17.47	9.09	.96 (.97)	4.80	.91 (.92)	2.82	.77 (.83)	1.97	.55 (.70)	1.70	.32 (.53)	
320	50	27.13	14.14	.96 (.98)	7.45	.91 (.94)	4.22	.80 (.86)	2.76	.61 (.74)	2.12	.40 (.58)	
384	50	39.35	20.07	.98 (.98)	10.76	.91 (.95)	5.95	.83 (.88)	3.78	.65 (.78)	2.61	.47 (.63)	
448	50	53.71	27.38	.98 (.98)	14.34	.94 (.95)	7.83	.86 (.90)	4.62	.73 (.80)	3.04	.55 (.66)	
32	100	2.62	1.61	.81 (.84)	1.11	.59 (.63)	0.88	.37 (.42)	0.81	.20 (.26)	0.83	.10 (.14)	
64	100	10.56	5.94	.89 (.90)	3.61	.73 (.76)	2.47	.53 (.57)	1.98	.33 (.39)	1.78	.19 (.23)	
96	100	19.04	10.39	.92 (.93)	5.92	.80 (.82)	3.81	.62 (.66)	2.76	.43 (.48)	2.56	.23 (.31)	
128	100	26.67	14.27	.93 (.95)	8.06	.83 (.86)	4.89	.68 (.72)	3.45	.48 (.54)	3.15	.26 (.37)	
160	100	41.18	21.78	.95 (.96)	11.91	.86 (.88)	7.08	.73 (.76)	4.86	.53 (.60)	3.63	.35 (.42)	
192	100	58.91	30.93	.95 (.96)	16.76	.88 (.90)	9.83	.75 (.79)	6.47	.57 (.64)	4.75	.39 (.46)	
224	100	79.85	41.65	.96 (.97)	22.40	.89 (.91)	12.86	.78 (.81)	8.02	.62 (.67)	5.68	.44 (.50)	

5.4 Conclusion

Computationally efficient sequential and parallel algorithms for computing the best linear unbiased estimator of the general linear model (5.1) have been proposed. The sequential algorithm is a block version of an efficient serial approach that employs as a main computational component the Generalized QR Decomposition [35]. The new block Givens algorithm exploits the triangular structure of the Cholesky factor C of the dispersion matrix Ω and is rich in BLAS-3 operations. It is found to be $m/3n$ times faster than the

corresponding LAPACK routine DGGGLM for estimating the GLM [2].

The parallel approach is based on the new sequential strategy. The parallel algorithm copies the augmented matrix \tilde{X} and the main, sub- and super-block diagonals of C to all processors. The rest of the matrix C is evenly distributed across the processors. The algorithm duplicates parts of the computation. However, this is compensated for the load balanced distribution of the computationally expensive matrix C resulting in minimal inter-processor communication. The algorithms have been implemented on a parallel computer with distributed memory. The theoretical complexities of both algorithms are stated and experimental results are presented and analyzed. Overall, the parallel algorithm is found to be scalable and capable of solving large scale GLM estimation problems, where $m \gg n$.

Currently, an adaptation of the parallel algorithm to estimate Seemingly Unrelated Regressions -a special class of a GLM which involving Kronecker structures- is being investigated [10, 24, 29, 43, 51].

Chapter 6

Computationally efficient methods for estimating the updated-observations SUR models

Abstract:

Efficient serial and parallel algorithms for estimating the seemingly unrelated regressions model after been updated with new observations are proposed. The sequential block algorithm is based on orthogonal transformations and exploits the sparse structure of the data matrix and the Cholesky factor of the variance-covariance matrix. A parallel version of the new sequential block algorithm is developed. It utilizes an efficient distribution of the matrices over the processors and has low inter-processor communication. Theoretical and experimental results are presented and analyzed. The parallel algorithm is found to be scalable and efficient.

¹This chapter is a reprint of the paper: P. Yanev and E.J. Kontoghiorghes. Computationally efficient methods for estimating the updated-observations SUR models. *Applied Numerical Mathematics*, 2005 (Submitted).

6.1 Introduction

Consider the seemingly unrelated regressions (SUR) model, defined by the set of G regressions

$$y_i = X_i\beta_i + \varepsilon_i, \quad i = 1, \dots, G, \quad (6.1)$$

where $y_i \in \mathbb{R}^T$ are the response vectors, $X_i \in \mathbb{R}^{T \times k_i}$ are full column rank data matrices, $\beta_i \in \mathbb{R}^{k_i}$ are the coefficients to be estimated and $\varepsilon_i \in \mathbb{R}^T$ are the disturbance vectors, which have zero mean and variance-covariance matrix $\sigma_{i,i}I_T$. Note that, the disturbances in the SUR model are contemporaneously correlated across the regression equations, i.e. $E(\varepsilon_i\varepsilon_j^T) = \sigma_{i,j}I_T$ ($i, j = 1, \dots, G$) [24, 43, 44].

The compact form of the SUR model is given by

$$\begin{pmatrix} y_1 \\ \vdots \\ y_G \end{pmatrix} = \begin{pmatrix} X_1 & & \\ & \ddots & \\ & & X_G \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_G \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_G \end{pmatrix}, \quad (6.2)$$

which can be equivalently written as

$$\text{vec}(Y) = (\oplus_{i=1}^G X_i)\text{vec}(\{\beta_i\}_G) + \text{vec}(E), \quad (6.3)$$

where $Y = (y_1 \cdots y_G)$, $E = (\varepsilon_1 \cdots \varepsilon_G)$, $\oplus_{i=1}^G X_i = \text{diag}(X_1, \dots, X_G)$, $\{\beta_i\}_G$ denotes a set of G vectors and $\text{vec}(\cdot)$ is the vector operator that stacks the columns of a matrix or set of vectors. The disturbances, $\text{vec}(E)$ in (6.3), have zero mean and variance-covariance matrix $\Sigma \otimes I_T$, i.e. $\text{vec}(E) \sim (0, \Sigma \otimes I_T)$, where $\Sigma = [\sigma_{i,j}] \in \mathbb{R}^{G \times G}$ is symmetric positive semidefinite matrix [24, 43, 44, 45, 51]. For simplicity, the data matrix $\oplus_{i=1}^G X_i$ is abbreviated to $\oplus_i X_i$ and the coefficients $\{\beta_i\}_G$ to $\{\beta_i\}$. The notation is consistent with that in [27].

The best linear unbiased estimator (BLUE) of $\{\beta_i\}$ can be obtained by solving the generalized linear least squares problem (GLLSP)

$$\underset{U, \{\beta_i\}}{\text{argmin}} \|U\|_F^2 \quad \text{subject to} \quad \text{vec}(Y) = (\oplus_i X_i)\text{vec}(\{\beta_i\}) + \text{vec}(UC^T), \quad (6.4)$$

where $\Sigma = CC^T$, the matrix $U \in \mathbb{R}^{T \times G}$ is such that $UC^T = E$ and the $\|\cdot\|_F$ is the Frobenius norm, defined as $\|U\|_F^2 = \sum_{i=1}^T \sum_{j=1}^G U_{i,j}^2$ [10]. It will be assumed that the matrix $C \in \mathbb{R}^{G \times G}$ is the upper-triangular Cholesky factor of Σ and is part of the original data. From the properties of the Kronecker product and the $\text{vec}(\cdot)$ operator follows that $\text{vec}(E) \equiv \text{vec}(UC^T) = (C \otimes I_T)\text{vec}(U)$, and thus, $\text{vec}(U) \sim (0, I_{GT})$ [39, 41].

The GLLSP (6.4) can be solved using the generalized QR decomposition (GQRD) of $\oplus_i X_i$ and $(C \otimes I_T)$, which first computes the QR decomposition (QRD)

$$Q^T(\oplus_i X_i) = \begin{pmatrix} \oplus_i R_i & \\ & 0 \end{pmatrix} \begin{matrix} K \\ GT - K \end{matrix} \quad (6.5a)$$

and then the RQ decomposition (RQD)

$$(Q^T(C \otimes I_T))\Pi = W \equiv \begin{pmatrix} K & GT - K \\ W_{1,1} & W_{1,2} \\ 0 & W_{2,2} \end{pmatrix} \begin{matrix} K \\ GT - K \end{matrix}, \quad (6.5b)$$

where $K = \sum_{i=1}^G k_i$, R_i and W are upper triangular matrices of order k_i and GT , respectively, and $Q, \Pi \in \mathbb{R}^{GT \times GT}$ are orthogonal [29, 37, 38].

The GLLSP (6.4) is equivalent to

$$\operatorname{argmin}_{U, \{\beta_i\}} \|\Pi^T \operatorname{vec}(U)\|^2 \text{ s.t. } Q^T \operatorname{vec}(Y) = Q^T(\oplus_i X_i) \operatorname{vec}(\{\beta_i\}) + Q^T(C \otimes I_T) \Pi \Pi^T \operatorname{vec}(U),$$

which after the computation of the GQRD (6.5) can be written as

$$\operatorname{argmin}_{\tilde{u}_i, \hat{u}_i, \{\beta_i\}} \sum_{i=1}^G (\|\tilde{u}_i\|^2 + \|\hat{u}_i\|^2) \text{ subject to} \\ \begin{pmatrix} \operatorname{vec}(\{\tilde{y}_i\}) \\ \operatorname{vec}(\{\hat{y}_i\}) \end{pmatrix} = \begin{pmatrix} \oplus_i R_i & \\ & 0 \end{pmatrix} \operatorname{vec}(\{\beta_i\}) + \begin{pmatrix} W_{1,1} & W_{1,2} \\ 0 & W_{2,2} \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\{\tilde{u}_i\}) \\ \operatorname{vec}(\{\hat{u}_i\}) \end{pmatrix}, \quad (6.6)$$

where $\|\cdot\|$ denotes the Euclidean norm and the vectors $Q^T \operatorname{vec}(Y)$ and $\Pi^T \operatorname{vec}(U)$ are partitioned, respectively, as

$$Q^T \operatorname{vec}(Y) = \begin{pmatrix} \operatorname{vec}(\{\tilde{y}_i\}) \\ \operatorname{vec}(\{\hat{y}_i\}) \end{pmatrix} \begin{matrix} K \\ T - K \end{matrix} \quad \text{and} \quad \Pi^T \operatorname{vec}(U) = \begin{pmatrix} \operatorname{vec}(\{\tilde{u}_i\}) \\ \operatorname{vec}(\{\hat{u}_i\}) \end{pmatrix} \begin{matrix} K \\ T - K \end{matrix}.$$

From the constrains in (6.6) it follows that $\operatorname{vec}(\{\hat{y}_i\}) = W_{2,2} \operatorname{vec}(\{\hat{u}_i\})$, i.e. $\operatorname{vec}(\{\hat{u}_i\})$ can be computed. Then, setting $\operatorname{vec}(\{\tilde{u}_i\}) = 0$, the BLUE of the SUR model is given by

$$\oplus_i R_i \operatorname{vec}(\{\hat{\beta}_i\}) = \operatorname{vec}(\{\tilde{\hat{y}}_i\}), \quad (6.7)$$

where $\operatorname{vec}(\{\tilde{\hat{y}}_i\}) = \operatorname{vec}(\{\tilde{y}_i\}) - W_{1,2} \operatorname{vec}(\{\hat{u}_i\})$.

Often the SUR model is updated by new observations [6, 7, 24]. The estimation of the updated SUR model has been discussed in [27]. The purpose of this work is to design efficient strategies to compute the updated estimators. The next section reviews the updated SUR model estimation procedure. Section 3 considers sequential strategies for computing the main matrix factorizations arising in the estimation procedure. Parallel algorithms are considered in Section 4. The algorithm is implemented on a virtual shared memory parallel machine SUN Enterprise 10 000 (16 CPU UltraSPARC of 400 MHz) using a single-program multiple-data (SPMD) programming paradigm. Finally, Section 5 concludes.

6.2 Updating the SUR model with new observations

The updated-observation SUR (UO-SUR) model is defined as the original SUR model (6.1) with an equal number of new observations added to each regression equation. Let the new observations be denoted by

$$y_i^{(s)} = X_i^{(s)}\beta_i + \varepsilon_i^{(s)}, \quad i = 1, \dots, G, \quad (6.8)$$

which can be written equivalently as

$$\text{vec}(Y^{(s)}) = (\oplus_i X_i^{(s)})\text{vec}(\{\beta_i\}) + \text{vec}(E^{(s)}), \quad (6.9)$$

where $\text{vec}(E^{(s)}) \sim (0, \Sigma^{(s)} \otimes I_{T^{(s)}})$, $\Sigma^{(s)} \in \mathbb{R}^{(G \times G)}$ is positive definite and non-singular and $T^{(s)}$ is the number of observations added to each equation. The problem of estimating the UO-SUR model is the solution of the SUR model

$$\begin{pmatrix} y_i \\ y_i^{(s)} \end{pmatrix} = \begin{pmatrix} X_i \\ X_i^{(s)} \end{pmatrix} \beta_i + \begin{pmatrix} \varepsilon_i \\ \varepsilon_i^{(s)} \end{pmatrix}, \quad i = 1, \dots, G, \quad (6.10)$$

after (6.1) has been solved. The UO-SUR model can be equivalently written as

$$\begin{pmatrix} \text{vec}(Y) \\ \text{vec}(Y^{(s)}) \end{pmatrix} = \begin{pmatrix} \oplus_i X_i \\ \oplus_i X_i^{(s)} \end{pmatrix} \text{vec}(\{\beta_i\}) + \begin{pmatrix} \text{vec}(E) \\ \text{vec}(E^{(s)}) \end{pmatrix}. \quad (6.11)$$

Note that, the disturbances $\text{vec}(E^{(s)})$ and $\text{vec}(E)$ are not correlated, i.e.

$$\begin{pmatrix} \text{vec}(E) \\ \text{vec}(E^{(s)}) \end{pmatrix} \sim \left(0, \begin{pmatrix} \Sigma \otimes I_T & 0 \\ 0 & \Sigma^{(s)} \otimes I_{T^{(s)}} \end{pmatrix} \right).$$

The BLUE of the UO-SUR model is obtained by solving a similar to (6.4) GLLSP (UO-GLLSP), which is given by

$$\begin{aligned} & \underset{U, U^{(s)}, \{\beta_i\}}{\operatorname{argmin}} \|U\|_F^2 + \|U^{(s)}\|_F^2 \quad \text{subject to} \\ & \begin{pmatrix} \operatorname{vec}(Y) \\ \operatorname{vec}(Y^{(s)}) \end{pmatrix} = \begin{pmatrix} \oplus_i X_i \\ \oplus_i X_i^{(s)} \end{pmatrix} \operatorname{vec}(\{\beta_i\}) + \begin{pmatrix} C \otimes I_T & 0 \\ 0 & C^{(s)} \otimes I_{T^{(s)}} \end{pmatrix} \begin{pmatrix} \operatorname{vec}(U) \\ \operatorname{vec}(U^{(s)}) \end{pmatrix} \end{pmatrix} \quad (6.12) \end{aligned}$$

where $C^{(s)} \in \mathbb{R}^{G \times G}$ is the upper-triangular Cholesky factor of $\Sigma^{(s)}$ and $U^{(s)}(C^{(s)})^T = E^{(s)}$. Using the GQRD (6.5), the constraints (6.12) becomes equivalent to

$$\begin{pmatrix} \operatorname{vec}(\tilde{y}_i) \\ \operatorname{vec}(\hat{y}_i) \\ \operatorname{vec}(Y^{(s)}) \end{pmatrix} = \begin{pmatrix} \oplus_i R_i \\ 0 \\ \oplus_i X_i^{(s)} \end{pmatrix} \operatorname{vec}(\{\beta_i\}) + \begin{pmatrix} W_{1,1} & W_{1,2} & 0 \\ 0 & W_{2,2} & 0 \\ 0 & 0 & C^{(s)} \otimes I_{T^{(s)}} \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\tilde{u}_i) \\ \operatorname{vec}(\hat{u}_i) \\ \operatorname{vec}(U^{(s)}) \end{pmatrix}$$

and from the solution of the GLLSP (6.6), it follows that the UO-GLLSP can be reduced to

$$\begin{aligned} & \underset{\tilde{u}_i, U^{(s)}, \{\beta_i\}}{\operatorname{argmin}} \sum_{i=1}^G \|\tilde{u}_i\|^2 + \|U^{(s)}\|_F^2 \quad \text{subject to} \\ & \begin{pmatrix} \operatorname{vec}(\{\tilde{y}_i\}) \\ \operatorname{vec}(Y^{(s)}) \end{pmatrix} = \begin{pmatrix} \oplus_i R_i \\ \oplus_i X_i^{(s)} \end{pmatrix} \operatorname{vec}(\{\beta_i\}) + \begin{pmatrix} W_{1,1} & 0 \\ 0 & C^{(s)} \otimes I_{T^{(s)}} \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\{\tilde{u}_i\}) \\ \operatorname{vec}(U^{(s)}) \end{pmatrix}, \end{pmatrix} \quad (6.13) \end{aligned}$$

where $\operatorname{vec}(\{\tilde{y}_i\})$ is computed as in (6.7). The reduced in size UO-GLLSP can be solved using the updated GQRD (UGQRD)

$$(Q^{(s)})^T \begin{pmatrix} \oplus_i R_i \\ \oplus_i X_i^{(s)} \end{pmatrix} = \begin{pmatrix} \oplus_i R_i^{(s)} \\ 0 \end{pmatrix} \begin{pmatrix} K \\ GT^{(s)} \end{pmatrix} \quad (6.14a)$$

and

$$\left((Q^{(s)})^T \begin{pmatrix} W_{1,1} & 0 \\ 0 & C^{(s)} \otimes I_{T^{(s)}} \end{pmatrix} \right) \Pi^{(s)} = W^{(s)} = \begin{pmatrix} K & GT^{(s)} \\ W_{1,1}^{(s)} & W_{1,2}^{(s)} \\ 0 & W_{2,2}^{(s)} \end{pmatrix} \begin{pmatrix} K \\ GT^{(s)} \end{pmatrix}, \quad (6.14b)$$

where $Q^{(s)}, \Pi^{(s)} \in \mathbb{R}^{K+GT^{(s)} \times K+GT^{(s)}}$ are orthogonal and $\oplus_i R_i^{(s)}$ and $W^{(s)}$ are upper triangular of order K and $K + GT^{(s)}$, respectively. Note that after the UGQRD (6.14) the UO-GLLSP becomes similar to the GLLSP (6.6) and can be solved equivalently.

6.3 Sequential strategies for computing the UGQRD (6.14)

The serial algorithms proposed here take advantage of the sparse structure of the matrices in (6.14) and are rich in BLAS-3 operations [47, 50]. Sequentially, the computation of the UGQRD is performed in two stages. Let the upper-triangular Cholesky matrices $W_{1,1}$ and $C^{(s)} \otimes I_{T^{(s)}}$ be partitioned, respectively, as

$$W_{1,1} = \begin{pmatrix} k_1 & \cdots & k_G \\ A_{1,1} & \cdots & A_{1,G} \\ & \ddots & \vdots \\ & & A_{G,G} \end{pmatrix} \begin{matrix} k_1 \\ \vdots \\ k_G \end{matrix} \text{ and } C^{(s)} \otimes I_{T^{(s)}} = \begin{pmatrix} T^{(s)} & \cdots & T^{(s)} \\ C_{1,1} & \cdots & C_{1,G} \\ & \ddots & \vdots \\ & & C_{G,G} \end{pmatrix} \begin{matrix} T^{(s)} \\ \vdots \\ T^{(s)} \end{matrix}, \quad (6.15)$$

where $A_{i,i}$ and $C_{i,i}$ ($i = 1, \dots, G$) are upper triangular. The first stage computes the UQRD (6.14a) in G steps. The i th ($i = 1, \dots, G$) step derives the QRD

$$Q_i^T \begin{pmatrix} R_i \\ X_i^{(s)} \end{pmatrix} = \begin{pmatrix} R_i^{(s)} \\ 0 \end{pmatrix} \begin{matrix} k_i \\ T^{(s)} \end{matrix} \quad (6.16a)$$

and computes the product

$$Q_i^T \begin{pmatrix} k_i & \cdots & k_G & T^{(s)} & \cdots & T^{(s)} \\ A_{i,i} & \cdots & A_{i,G} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & C_{i,i} & \cdots & C_{i,G} \end{pmatrix} = \begin{pmatrix} \hat{A}_{i,i} & \cdots & \hat{A}_{i,G} & B_{i,i} & \cdots & B_{i,G} \\ D_{i,i} & \cdots & D_{i,G} & \hat{C}_{i,i} & \cdots & \hat{C}_{i,G} \end{pmatrix} \begin{matrix} k_i \\ T^{(s)} \end{matrix}, \quad (6.16b)$$

where $R_i^{(s)} \in \mathbb{R}^{k_i \times k_i}$ is upper triangular and Q_i^T is orthogonal and of order $(k_i + T^{(s)})$. Note that, after the UQRD (6.14a) is completed, the orthogonal matrix $Q^{(s)}$ is given by

$$Q^{(s)} = \begin{pmatrix} K & GT^{(s)} \\ \oplus_i Q_i^{(1,1)} & \oplus_i Q_i^{(1,2)} \\ \oplus_i Q_i^{(2,1)} & \oplus_i Q_i^{(2,2)} \end{pmatrix} \begin{matrix} K \\ GT^{(s)} \end{matrix}, \quad \text{where } Q_i = \begin{pmatrix} k_i & T^{(s)} \\ Q_i^{(1,1)} & Q_i^{(1,2)} \\ Q_i^{(2,1)} & Q_i^{(2,2)} \end{pmatrix} \begin{matrix} k_i \\ T^{(s)} \end{matrix}$$

and the modified Cholesky matrix in (6.14b) has the form

$$(Q^{(s)})^T \begin{pmatrix} W_{1,1} & 0 \\ 0 & C^{(s)} \otimes I_{T^{(s)}} \end{pmatrix} = \begin{pmatrix} k_1 & \cdots & k_G & T^{(s)} & \cdots & T^{(s)} \\ \widehat{A}_{1,1} & \cdots & \widehat{A}_{1,G} & B_{1,1} & \cdots & B_{1,G} \\ & \ddots & \vdots & & \ddots & \vdots \\ & & \widehat{A}_{G,G} & & & B_{G,G} \\ D_{1,1} & \cdots & D_{1,G} & \widehat{C}_{1,1} & \cdots & \widehat{C}_{1,G} \\ & \ddots & \vdots & & \ddots & \vdots \\ & & D_{G,G} & & & \widehat{C}_{G,G} \end{pmatrix} \begin{pmatrix} k_1 \\ \vdots \\ k_G \\ T^{(s)} \\ \vdots \\ T^{(s)} \end{pmatrix}.$$

The second stage computes the URQD (6.14b), i.e. it annihilates the matrices $D_{i,j}$ ($i, j = 1, \dots, G, i \leq j$) from the right in G steps. The i th ($i = 1, \dots, G$) step annihilates the matrices $D_{t,t+j-1}$ by computing (for $t = G - i + 1$ and $j = 1, \dots, i$) the RQD

$$\begin{pmatrix} D_{t,t+j-1}^{(i-1)} & \widehat{C}_{t,t}^{(j-1)} \end{pmatrix} \Pi_{i,j} = \begin{pmatrix} 0 & \widehat{C}_{t,t}^{(j)} \end{pmatrix} \quad (6.17a)$$

and the product

$$\begin{pmatrix} k_{t+j-1} & T^{(s)} \\ \widehat{A}_{1,t+j-1}^{(i-1)} & B_{1,t}^{(j-1)} \\ \vdots & \vdots \\ \widehat{A}_{t+j-1,t+j-1}^{(i-1)} & B_{t+j-1,t}^{(j-1)} \\ D_{1,t+j-1}^{(i-1)} & \widehat{C}_{1,t}^{(j-1)} \\ \vdots & \vdots \\ D_{t-1,t+j-1}^{(i-1)} & \widehat{C}_{t-1,t}^{(j-1)} \end{pmatrix} \Pi_{i,j} = \begin{pmatrix} k_{t+j-1} & T^{(s)} \\ \widehat{A}_{1,t+j-1}^{(i)} & B_{1,t}^{(j)} \\ \vdots & \vdots \\ \widehat{A}_{t+j-1,t+j-1}^{(i)} & B_{t+j-1,t}^{(j)} \\ D_{1,t+j-1}^{(i)} & \widehat{C}_{1,t}^{(j)} \\ \vdots & \vdots \\ D_{t-1,t+j-1}^{(i)} & \widehat{C}_{t-1,t}^{(j)} \end{pmatrix} \begin{pmatrix} k_1 \\ \vdots \\ k_{t+j-1} \\ T^{(s)} \\ \vdots \\ T^{(s)} \end{pmatrix}, \quad (6.17b)$$

where $\widehat{C}_{t,t}^{(j)} \in \mathbb{R}^{T^{(s)} \times T^{(s)}}$ is upper triangular, $\widehat{A}_{m,n}^{(0)} = \widehat{A}_{m,n}$, $B_{m,n}^{(0)} = B_{m,n}$, $D_{m,n}^{(0)} = D_{m,n}$, $\widehat{C}_{m,n}^{(0)} = \widehat{C}_{m,n}$ ($m, n = 1, \dots, G, m \leq n$) and $B_{m,n}^{(i-1)} = 0$ when $i > 1$ and $m \geq n + i - 1$. After the computation of the URQD (6.14b), the matrices $W_{1,1}^{(s)}$, $W_{1,2}^{(s)}$ and $W_{2,2}^{(s)}$ are given, respectively, by

$$W_{1,1}^{(s)} = \begin{pmatrix} \widehat{A}_{1,1}^{(G)} & \cdots & \widehat{A}_{1,G}^{(G)} \\ & \ddots & \vdots \\ & & \widehat{A}_{G,G}^{(G)} \end{pmatrix}, W_{1,2}^{(s)} = \begin{pmatrix} B_{1,1}^{(G)} & \cdots & B_{1,G}^{(G)} \\ \vdots & & \vdots \\ B_{G,1}^{(G)} & \cdots & B_{G,G}^{(G)} \end{pmatrix} \text{ and } W_{2,2}^{(s)} = \begin{pmatrix} \widehat{C}_{1,1}^{(G)} & \cdots & \widehat{C}_{1,G}^{(G)} \\ & \ddots & \vdots \\ & & \widehat{C}_{G,G}^{(G)} \end{pmatrix}.$$

Figure 6.1 illustrates the process of computing the UGQRD (6.14). An arc between two blocks denotes an UQRD or an URQD done in stage 1 and stage 2, respectively. The orthogonal matrices $Q_i^{(s)}$ and $\Pi_{i,j}^{(s)}$ are not explicitly computed ($i, j = 1, \dots, G, i \leq j$).

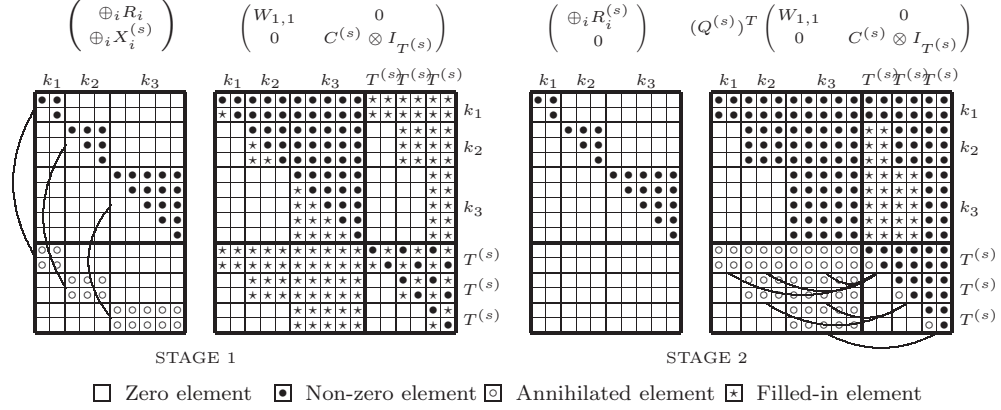


Figure 6.1: Computing the UGQRD (6.14), where $G = 3$ and $T^{(s)} = 2$.

Algorithm 11 Sequential algorithm for solving the UGQRD (6.14).

- 1: Let $W_{1,1}$ and $C^{(s)} \times I_{T^{(s)}}$ in (6.13) be partitioned as in (6.15).
 - 2: Let $\forall i D_{t,t}^{(i)} = D_{t,t}$ and $\widehat{C}_{t,t}^{(0)} = \widehat{C}_{t,t}$ ($t = 1, \dots, G$).
 - 3: **for** $i = 1, \dots, G$ **do**
 - 4: Compute the UQRD (6.16a) and the product (6.16b).
 - 5: **end for**
 - 6: **for** $i = 1, \dots, G$ **do**
 - 7: Let $t := G - i + 1$
 - 8: **for** $j = 1, \dots, i$ **do**
 - 9: Compute the URQD of $\begin{pmatrix} D_{t,t}^{(i-1)} & \widehat{C}_{t,t}^{(j-1)} \end{pmatrix}$ as in (6.17a) and the product (6.17b).
 - 10: **end for**
 - 11: **end for**
-

Algorithm 11 summarizes the steps of the sequential strategy for solving the UGQRD (6.14). The main computational tools utilized throughout the computations is the UQRD (at line 4) and the equivalent URQD (at line 9). Hereafter only the UQRD will be considered. Two different strategies for computing this factorization are proposed. The first approach is to compute the updating of the QRD using the standard LAPACK QRD routine. Note that this strategy does not take into account the upper triangular structure of

the submatrix used in the updating. The theoretical complexity of this approach is given by

$$T_{\text{LAPACK}}(n, s, m) = 2n^2(s + 2n/3) + 2mn(n + 2s + 1). \quad (6.18)$$

Here the updated upper-triangular matrix is of dimension $(n \times n)$, s is the number of new rows added to this matrix and m is the number of columns of the matrix multiplied with the orthogonal factor after the updating.

The second approach to compute the URQD is a block updating strategy and takes advantage of the initial upper-triangular structure of the matrix, which has to be updated. It partitions the matrices in blocks of size $(v \times v)$, where n , m and s are multiples of v . The updating of the QRD is computed in n/v steps, where in each step a QRD of a $(v + s) \times v$ matrix is derived and then the orthogonal matrix is multiplied with a matrix of dimensions $(v + s) \times m$. The theoretical complexity of this approach is given by

$$T_{\text{BLOCK}}(n, s, m, v) = 2nv(s + 2v/3) + n(2m + n - v)(v + 2s + 1). \quad (6.19)$$

Note that $T_{\text{LAPACK}}(n, s, m) \geq T_{\text{BLOCK}}(n, s, m, v)$, for $n \geq v \geq 1$. The equality holds if and only if $v = n$. The theoretical complexity of Algorithm 11 can be calculated using each of the two aforementioned strategies. Specifically, in line 4, the UQRD of a matrix with dimensions $(k_i + T^{(s)}) \times k_i$ is computed and the orthogonal matrix is then multiplied with a $(k_i + T^{(s)}) \times (\sum_{j=i}^G k_j + T^{(s)}(G - i + 1))$ matrix, for $i = 1, \dots, G$. Finally, in line 9, the URQD of a $T^{(s)} \times (T^{(s)} + k_{G+j-i})$ matrix is computed and the orthogonal factor is applied to a matrix with dimensions $(\sum_{t=1}^{G+j-i} k_t + T^{(s)}(G - i)) \times (T^{(s)} + k_{G+j-1})$, for $i = 1, \dots, G$ and $j = 1, \dots, i$. The overall complexity of the Algorithm 11, is given by:

$$T_S = \sum_{i=1}^G T_{\text{L/B}}(k_i, T^{(s)}, \sum_{j=i}^G k_j + T^{(s)}(G - i + 1)) + \sum_{i=1}^G \sum_{j=1}^i T_{\text{L/B}}(T^{(s)}, k_r, \sum_{t=1}^r k_t + T^{(s)}(G - i)),$$

where $r = G + j - i$ and $T_{\text{L/B}}$ is either the T_{LAPACK} or T_{BLOCK} complexity (the block-size parameter v of T_{BLOCK} has been omitted).

The BLOCK algorithm for the two important cases of the SUR model is investigated. In the first case, all equations have the same number of variables, i.e. $\forall i k_i = k$. The second case, the i th equation has $k_i = ik$ variables, where $k = k_1$ and $i = 1, \dots, G$. The theoretical order of complexities of these two cases are given, respectively, by:

$$T_{\text{BS}_1}(G, k, T^{(s)}, v) \approx G^2(2GkT^{(s)}(2k + T^{(s)}) + v(3k^2 + 2GkT^{(s)} + G(T^{(s)})^2))/3 \quad (6.20)$$

and

$$T_{BS_2}(G, k, T^{(s)}, v) \approx G^4 k (5v(k + T^{(s)}) + 2T^{(s)}(4Gk + T^{(s)})) / 20. \quad (6.21)$$

Table 6.1 and 6.2 show the execution times and the theoretical complexities of Algorithm 11 for different values of G and k_i ($i = 1, \dots, G$). The values of the theoretical complexity in hundred millions of floating point operations (flops) of Algorithm 11 are presented in brackets. Table 6.1 considers the SUR model, where $k_i = k$ ($i = 1, \dots, G$) and G varies from 5 to 40. Table 6.2 presents the results for the SUR model comprising 5 equations, i.e. $G = 5$. The number of variables k_i is the same for all equations in the first two columns, i.e. $k = k_1 = \dots = k_G$, while in the last two columns $k_i = ik$ ($i = 1, \dots, G$). The best block size for the BLOCK approach is found to be $v = 20$. The experimental results show that the BLOCK algorithm outperforms the LAPACK algorithm. For larger dimensions the BLOCK strategy is found to be up to twice faster than the LAPACK strategy. The obtained results are confirmed by the theoretical complexities.

Table 6.1: Execution times and theoretical complexities of Algorithm 11, $k_i = k$, $i = 1, \dots, G$ and $v = 20$.

$T^{(s)}$	G	$k = 5$		$k = 20$		$k = 40$	
		BLOCK	LAPACK	BLOCK	LAPACK	BLOCK	LAPACK
25	5	0.02 (0.02)	0.02 (0.02)	0.09 (0.07)	0.09 (0.08)	0.23 (0.20)	0.23 (0.23)
25	10	0.14 (0.11)	0.15 (0.13)	0.53 (0.43)	0.51 (0.49)	1.41 (1.20)	1.38 (1.35)
25	15	0.45 (0.33)	0.48 (0.42)	1.61 (1.33)	1.60 (1.49)	4.22 (3.65)	4.11 (4.05)
25	20	1.02 (0.75)	1.18 (0.95)	3.61 (2.99)	3.56 (3.36)	10.10 (8.19)	9.67 (9.02)
25	40	7.95 (5.59)	8.34 (7.15)	27.34 (21.95)	28.32 (24.84)	75.67 (60.06)	72.50 (65.58)
50	5	0.08 (0.06)	0.11 (0.11)	0.22 (0.20)	0.27 (0.28)	0.54 (0.50)	0.60 (0.64)
50	10	0.48 (0.37)	0.76 (0.76)	1.38 (1.22)	1.80 (1.79)	3.22 (2.98)	3.65 (3.88)
50	15	1.51 (1.16)	2.37 (2.38)	4.25 (3.72)	5.38 (5.54)	10.04 (9.06)	11.44 (11.86)
50	20	3.49 (2.64)	5.45 (5.45)	9.71 (8.36)	12.51 (12.55)	23.88 (20.35)	25.99 (26.70)
50	40	26.83 (19.81)	42.33 (41.31)	73.53 (61.65)	93.76 (93.78)	176.98 (149.76)	197.49 (197.26)

6.4 Parallel algorithm for computing the UGQRD

The computational steps of the UQRD (6.16) can be performed independently and thus, in parallel (see the first stage of Figure 6.1). In order to construct an efficient parallel

Table 6.2: Execution times and theoretical complexities of Algorithm 11, $v = 20$ and $G = 5$.

$T^{(s)}$	k	$k_i = k$		$k_i = ik$	
		BLOCK	LAPACK	BLOCK	LAPACK
25	5	0.02 (0.02)	0.02 (0.02)	0.06 (0.05)	0.06 (0.06)
25	10	0.04 (0.03)	0.04 (0.04)	0.16 (0.14)	0.16 (0.16)
25	20	0.09 (0.07)	0.09 (0.08)	0.50 (0.45)	0.54 (0.55)
25	40	0.23 (0.20)	0.23 (0.23)	1.76 (1.56)	2.23 (2.32)
50	5	0.08 (0.06)	0.11 (0.11)	0.18 (0.16)	0.22 (0.23)
50	10	0.11 (0.10)	0.16 (0.16)	0.39 (0.37)	0.44 (0.47)
50	20	0.22 (0.20)	0.27 (0.28)	1.10 (1.03)	1.18 (1.27)
50	40	0.54 (0.50)	0.60 (0.64)	3.56 (3.33)	4.07 (4.35)
100	5	0.27 (0.22)	0.74 (0.75)	0.57 (0.52)	1.06 (1.12)
100	10	0.39 (0.34)	0.87 (0.92)	1.16 (1.09)	1.69 (1.82)
100	20	0.67 (0.63)	1.22 (1.30)	2.77 (2.70)	3.39 (3.76)
100	40	1.44 (1.39)	2.14 (2.26)	8.72 (7.81)	9.32 (10.13)
200	5	1.12 (0.84)	4.98 (5.39)	2.11 (1.87)	6.37 (6.72)
200	10	1.49 (1.26)	5.48 (6.00)	3.97 (3.59)	8.17 (8.97)
200	20	2.41 (2.18)	6.67 (7.32)	8.71 (7.97)	13.24 (14.51)
200	40	4.86 (4.36)	9.66 (10.35)	22.96 (20.45)	27.83 (30.05)

algorithm, a load-balanced distribution of the matrices over the processors should be employed, which also provides low inter-processor communication [50, 48]. Let the number of processors is denoted by p and assume for simplicity that G is multiple of $2p$.

The distribution of the matrices depends mainly on the number of variables in each regression equation k_i ($i = 1, \dots, G$). As in the sequential algorithm, the two cases of $k_i = k$ and $k_i = ik$ ($i = 1, \dots, G$) are considered. In order to choose a load-balanced distribution, the theoretical complexities of the different steps during the computation of the UQRD (6.16) and the URQD (6.17) are investigated. The number of flops performed at the i th step of the UQRD (6.16) using the faster BLOCK approach, is given by:

$$T_{\text{BLOCK}_i} = 2k_i v(2v + 3T^{(s)}) + 3k_i \left(2 \sum_{j=i}^G k_j + 2T^{(s)}(G - i + 1) + k_i - v \right) (v + 2T^{(s)} + 1).$$

Here k_i is a multiple of the block size v . Figure 6.2 shows how the complexity change in each step of the computation of the UQRD (6.16). Note that, the shape of the curves will

remain the same for different values of G , k , v or $T^{(s)}$.

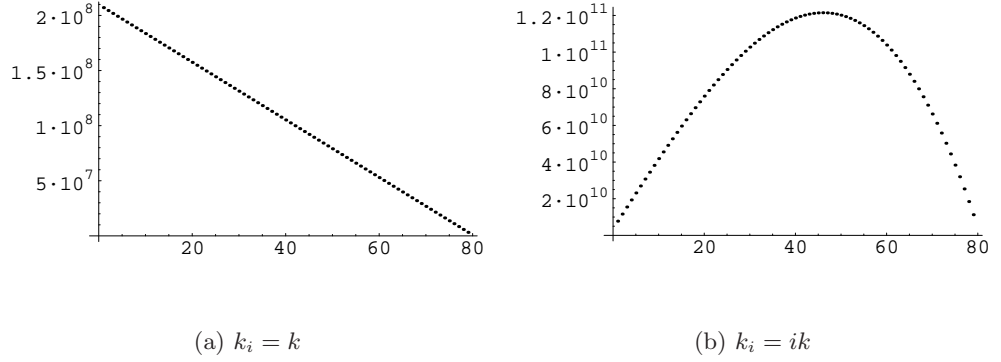


Figure 6.2: Theoretical complexity of the i th step ($i = 1, \dots, G$) of the UQRD (6.16), where $G = 80$, $k = 40$, $v = 20$ and $T^{(s)} = 50$.

An efficient load-balanced distribution allocates the G equations to the processors, in such a way, that the overall computational complexity assigned to each processor is the same. For the case shown in Figure 6.2(a), i.e. where $k_i = k$ ($i = 1, \dots, G$), a completely load-balanced distribution can be achieved when G is a multiple of $2p$. Specifically, the matrices computed during the first $G/2$ steps of the UQRD (6.16) are allocated to the processors using the cyclic distribution scheme. The matrices computed during the remaining $G/2$ steps are distributed using a reverse (counting backwards) cyclic allocation scheme. That is, the matrices affected in the i th step

$$\begin{pmatrix} k_i \\ R_i \\ X_i^{(s)} \end{pmatrix} k_i \quad \text{and} \quad \begin{pmatrix} k_i & \cdots & k_G & T^{(s)} & \cdots & T^{(s)} \\ A_{i,i} & \cdots & A_{i,G} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & C_{i,i} & \cdots & C_{i,G} \end{pmatrix} k_i \quad (6.22)$$

are allocated to the processor P_{γ_i} , where

$$\gamma_i = \begin{cases} (i-1) \bmod p + 1 & \text{if } i = 1, \dots, G/2, \\ p - (i-1) \bmod p & \text{if } i = G/2 + 1, \dots, G. \end{cases} \quad (6.23)$$

Figure 3 shows the distribution of the matrices over the processors, with $p = 4$, $G = 8$ and $k_i = k$ ($i = 1, \dots, G$).

The second case of the SUR model shown in Figure 6.2(b), i.e. where $k_i = ik$ ($i = 1, \dots, G$) and $k = k_1$, utilizes the same distribution scheme (6.23), as that illustrated in

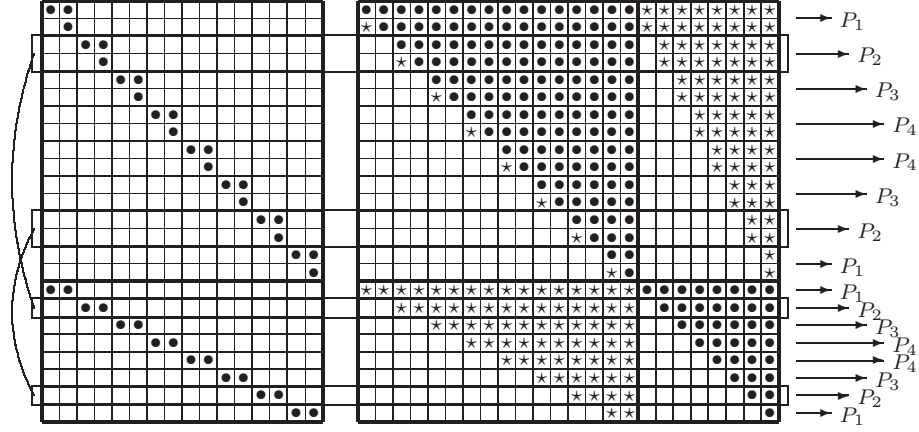


Figure 6.3: Distribution scheme for 4 processors, where $k_i = k$ ($i = 1, \dots, G$) and $G = 8$.

Figure 3. Note that the distribution does not achieve a perfect load balance among the processors. As shown in Figure 6.2(b), during the computation of the UQRD (6.16) the execution time will be dominated by the last processors. However, the distribution achieves load balanced computations during the solution of the URQD (6.17), which is the most significant time consuming operation of the GQRD (6.5). Note that, for $k_i = ik$ ($i = 1, \dots, G$) each processor has been allocated exactly $kG(G+1)/2p + T^{(s)}G/p$ rows.

The parallel algorithm computes the UGQRD (6.14) in two stages. During the first stage, the UQRD (6.16) is computed in G/p steps. Each processor derives the UQRD (6.16a) of its allocated matrices and then computes the product (6.16b). The second stage computes the URQD (6.17) in G steps. At the i th step ($i = 1, \dots, G$), the processor, P_γ say, which has locally the matrices $(D_{t,t}^{(i-1)} \cdots D_{t,G}^{(i-1)})$ and $\widehat{C}_{t,t}^{(i-1)}$ as in (6.17) (for $t = G - i + 1$) sends them to the other processors P_r ($r = 1, \dots, p$ and $r \neq \gamma_j$). Then, similarly to (6.17), each processor computes the URQD (6.17a) and updates locally the corresponding allocated block-columns. The parallel strategy is summarized in Algorithm 12.

The theoretical order of computational complexities of Algorithm 12, when $k_i = k$ and $k_i = ik$ ($i = 1, \dots, G$) is given, respectively, by

$$T_{BP_1}(G, k, T^{(s)}, v, p) \approx G^2(4GkT^{(s)}(2k + T^{(s)}) + 3v(2k^2 + 2kT^{(s)} + p(T^{(s)})^2))/6p \quad (6.24)$$

and

$$T_{BP_2}(G, k, T^{(s)}, v, p) \approx G^3(24G^2k^2T^{(s)} + 5v(3Gk(k + T^{(s)}) + 4(T^{(s)})^2))/60p. \quad (6.25)$$

Algorithm 12 The parallel algorithm for solving the UGQRD (6.14) on p processors.

- 1: Let $W_{1,1}$ and $C^{(s)} \times I_{T^{(s)}}$ in (6.13) be partitioned as in (6.15).
 - 2: Let $\forall i D_{t,t}^{(i)} = D_{t,t}$ and $\widehat{C}_{t,t}^{(0)} = \widehat{C}_{t,t}$ ($t = 1, \dots, G$).
 - 3: Allocate the matrices in (6.14) to the processors with respect to the values of k_i .
 - 4: **each processor** P_γ ($\gamma = 1, \dots, p$) **do in parallel:**
 - 5: **for** $j = 1, \dots, G/p$ **do**
 - 6: **if** $j > G/2p$ **and** $k_i = ik$ ($i = 1, \dots, G$) **then**
 - 7: Compute the UQRD (6.16a) and the product (6.16b) for $i = jp + 1 - \gamma$.
 - 8: **else**
 - 9: Compute the UQRD (6.16a) and the product (6.16b) for $i = (j - 1)p + \gamma$.
 - 10: **end if**
 - 11: **end for**
 - 12: **for** $i = 1, \dots, G$ **do**
 - 13: Let $t := G - i + 1$
 - 14: **if** P_γ has the matrix $\widehat{C}_{t,t}$ **then**
 - 15: Send $(D_{t,t}^{(i-1)} \dots D_{t,G}^{(i-1)} \widehat{C}_{t,t})$ to P_r , where $r = 1, \dots, p$ and $r \neq \gamma$.
 - 16: **else**
 - 17: Receive $(D_{t,t}^{(i-1)} \dots D_{t,G}^{(i-1)} \widehat{C}_{t,t})$ from the broadcasting processor.
 - 18: **end if**
 - 19: **for** $j = 1, \dots, i$ **do**
 - 20: Compute the factorization of $(D_{t,t+j-1}^{(i-1)} \widehat{C}_{t,t}^{(j-1)})$ as in (6.17a).
 - 21: Update locally the corresponding block-columns similarly to (6.17b).
 - 22: **end for**
 - 23: **end for**
-

From (6.20) and (6.24) it follows that the computational efficiency approaches one for very large G , i.e. $\lim_{G \rightarrow \infty} T_{BS_1}(G, k, T^{(s)}, v) / (p \times T_{BP_1}(G, k, T^{(s)}, v, p)) \approx 1$, which is also true in the case for (6.21) and (6.25), i.e. $\lim_{G \rightarrow \infty} T_{BS_2}(G, k, T^{(s)}, v) / (p \times T_{BP_2}(G, k, T^{(s)}, v, p)) \approx 1$. Note that, these theoretical complexities do not take into account the inter-processor communications. Table 6.3, 6.4 and 6.5 shows the execution times and actual (and in brackets the theoretical) efficiencies of Algorithm 12 for some $T^{(s)}$ and k . In Table 6.3 $k_i = k$ ($i = 1, \dots, G$) and $G = 32$. Table 6.4 shows the scalability of the algorithms for larger G and fixed value of k . That is, the efficiency remains constant when the size of the

number of equations in the SUR model, i.e. G and the number of processors are doubled. In Table 6.5 $k_i = ik$ ($i = 1, \dots, G$) for $k = 5$ and $k = 10$. The theoretical complexity is confirmed by the experimental results.

Table 6.3: Execution times (sec.) and efficiencies of Algorithm 12 for $k_i = k$ ($i = 1, \dots, G$) and $G = 32$.

$T^{(s)}$	k	2 processors			4 processors		8 processors		16 processors	
		Serial	Time	Eff.	Time	Eff.	Time	Eff.	Time	Eff.
25	5	4.35	2.39	.91 (.93)	1.35	.81 (.82)	0.81	.67 (.67)	0.61	.45 (.47)
25	10	7.05	3.84	.92 (.94)	2.21	.80 (.85)	1.28	.69 (.70)	0.86	.51 (.52)
25	20	14.27	7.63	.94 (.95)	4.32	.83 (.88)	2.4	.74 (.75)	1.65	.54 (.58)
25	40	37.65	20.33	.93 (.96)	11	.86 (.90)	6.03	.78 (.80)	3.96	.59 (.65)
50	5	14.79	8.24	.90 (.93)	4.58	.81 (.81)	2.92	.63 (.64)	2.04	.45 (.45)
50	10	21.01	11.43	.92 (.94)	6.66	.79 (.83)	3.97	.66 (.67)	2.7	.49 (.49)
50	20	38.21	20.83	.92 (.95)	11.82	.81 (.85)	6.86	.70 (.71)	4.71	.51 (.53)
50	40	88.99	47.55	.94 (.96)	26.14	.85 (.88)	14.77	.75 (.76)	10.11	.55 (.59)
100	5	55.11	30.77	.90 (.93)	18.35	.75 (.81)	11.27	.61 (.64)	8.12	.42 (.44)
100	10	77.78	43.19	.90 (.93)	23.58	.82 (.82)	14.94	.65 (.65)	11.05	.44 (.46)
100	20	121.89	66.64	.91 (.94)	38.19	.80 (.83)	22.68	.67 (.68)	16.12	.47 (.49)
100	40	251.53	137.09	.92 (.95)	75.75	.83 (.86)	44.78	.70 (.72)	31.33	.50 (.54)
200	5	220.99	123.51	.89 (.92)	69.67	.79 (.80)	45.07	.61 (.63)	32.52	.42 (.43)
200	10	288.5	161.75	.89 (.93)	92.21	.78 (.81)	58.46	.62 (.64)	43.91	.41 (.45)
200	20	455.97	253.19	.90 (.93)	143.76	.79 (.82)	88.47	.64 (.66)	63.12	.45 (.47)
200	40	875.92	488.03	.90 (.94)	268.43	.82 (.84)	162.8	.67 (.68)	115.07	.48 (.49)

6.5 Conclusions

The computational aspects of estimating the SUR model after it has been updated by new observations have been considered. Sequential and parallel algorithms to compute the main tool of the estimation procedure -the generalized QR decomposition of the data matrices- have been proposed. The algorithms are rich in BLAS-3 operations and exploit efficiently the triangular structure of the matrices. For reasonably big matrices the BLOCK sequential strategy is found to outperform the LAPACK sequential strategy. The parallel algorithm is based on a SPMD programming paradigm and employs different partitioning techniques

Table 6.4: Execution times (sec.) and efficiencies of Algorithm 12 for $k_i = k$ ($i = 1, \dots, G$) and $T^{(s)} = 50$.

k	G	2 processors			4 processors		8 processors		16 processors	
		Serial	Time	Eff.	Time	Eff.	Time	Eff.	Time	Eff.
5	32	14.79	8.24	.90 (.93)	4.58	.81 (.81)	2.92	.63 (.64)	2.04	.45 (.45)
5	64	105.62	57.3	.92 (.96)	30.96	.85 (.89)	17.67	.75 (.78)	11.05	.60 (.61)
5	96	363.43	189.32	.96 (.97)	99.79	.91 (.92)	55.76	.81 (.84)	33.32	.68 (.70)
5	128	866.26	445.09	.97 (.98)	235.1	.92 (.94)	130.9	.83 (.87)	71.86	.75 (.76)
10	32	21.01	11.43	.92 (.94)	6.66	.79 (.83)	3.97	.66 (.67)	2.7	.49 (.49)
10	64	161.56	84.78	.95 (.97)	45.42	.89 (.90)	26.22	.77 (.80)	15.68	.64 (.64)
10	96	540.02	280.07	.96 (.98)	148.55	.91 (.93)	81.56	.83 (.85)	47.06	.72 (.73)
10	128	1348.72	685.66	.98 (.98)	358.55	.94 (.95)	195.7	.86 (.88)	111.13	.76 (.78)

Table 6.5: Execution times (sec.) and efficiencies of Algorithm 12 for $k_i = ik$ ($i = 1, \dots, G$).

$G = 32$		2 processors			4 processors		8 processors		16 processors	
k	$T^{(s)}$	Serial	Time	Eff.	Time	Eff.	Time	Eff.	Time	Eff.
5	25	158.05	84.9	.93 (.95)	46.61	.85 (.87)	26.22	.75 (.76)	16.32	.61 (.63)
5	50	345.72	183.23	.94 (.95)	102.02	.85 (.86)	59.27	.73 (.74)	37.26	.58 (.60)
10	25	640.32	345.41	.93 (.94)	185.69	.86 (.87)	104.8	.76 (.77)	61.29	.65 (.66)
10	50	1206.51	653.09	.92 (.95)	351.46	.86 (.87)	202.7	.74 (.76)	121.45	.62 (.63)
$G = 64$		2 processors			4 processors		8 processors		16 processors	
5	25	4929.9	2529	.97 (.98)	1341	.92 (.93)	736.4	.84 (.86)	432.75	.71 (.75)
10	25	18646	9645	.97 (.98)	5139	.91 (.93)	2765	.84 (.86)	1603	.73 (.76)

with respect to the values of k_i ($i = 1, \dots, G$), i.e. the number of variables in the regressions of the SUR model, in order to achieve load balancing. The two main cases $k_i = k$ and $k_i = ik$ ($i = 1, \dots, G$) have been considered. Some small parts of the computations are duplicated. However, this is compensated by the minimal interprocessor communication. The execution times and the theoretical complexities of the sequential and the parallel algorithms are presented and analyzed. The theoretical results confirm the numerical experiments, which have illustrated the efficiency and scalability of the proposed parallel algorithm.

Chapter 7

Conclusions and future research

Computationally efficient algorithms for computing the estimation of Least Squares (LS) problems have been proposed and analyzed. The algorithms have the QR decomposition (QRD), or the Generalized QRD (GQRD), as the main computational tool. This allowed the design of computationally efficient and numerically stable implementations. Specifically, the block generalization of the Givens sequence have been employed for computing the QRD and the GQRD. In each problem which has been investigated the structure and the properties of the matrices involved in the estimation procedures have been exploited. The computational details of the implemented algorithms have been discussed. In order to assess the computational efficiency and scalability of the proposed algorithms the theoretical complexities and numerical results have been derived and analyzed.

Various strategies for computing the QRD of the set of matrices A_1, \dots, A_G which have common columns have been investigated in Chapter 2. The novel strategies use a weighted directed graph to express the common-columns relationship among the matrices. The nodes of the graph represent the triangular factors R_1, \dots, R_G derived from the QRDs of A_1, \dots, A_G , respectively. An edge between two nodes exists if and only if the columns of one of the matrices is a subset of the columns of the other. The weight of an edge is the computational complexity of deriving the triangular factor of the subset matrix given the QRD of the larger matrix. The problem is equivalent to construct a graph and to find the minimum cost for traversing all the nodes. The proposed algorithm computes the QRD of the matrices A_1, \dots, A_G by deriving the minimum spanning tree of the graph. The theoretical complexity of this strategy is found to be of double exponential order. Two alternative heuristic strategies have also been considered. The heuristic algorithms for

deriving the minimum spanning tree of the graph include some artificial nodes which are not present in the initial set of matrices. The artificial nodes are introduced in order to reduce the total computational cost. The first strategy had an exponential order of complexity which made it computational infeasible. The second heuristic approach had a linear order of complexity. Experimental results found that this heuristic strategy outperformed the straight forward approach of re-triangularizing the matrices one at a time.

In Chapter 3, five computational strategies for block downdating of the LS solutions have been proposed. The new strategies, which are block version of Givens rotations, are rich in BLAS-3 operations and exploit the sparsity of the orthogonal matrix Q . The numerically reliable method require that columns of the orthogonal matrix must be available [8]. The Givens strategy uses plane rotations to delete the new observations one at a time. The Givens method outperforms the downdating LAPACK routine when the number of deleted observations d is small compared to the number of variables n . It also outperforms the new block-downdating algorithm for very small $d \ll n$. For not very small d the proposed block-downdating algorithm is computationally most efficient.

The parallelization of the new block-downdating algorithm using various distributions of the computed matrices over the processors has been discussed in Chapter 4. Two parallel strategies for downdating the QRD have been proposed. The algorithms have been implemented using the single-program multiple-data (SPMD) paradigm. The performance of the first strategy is degraded by the communication cost which increases exponentially with the number of processors. The second strategy has no inter-processor communication, but has some duplicated computations which makes it suitable for large-scale problems on bigger number of processors. Generally, the second algorithm achieves perfect load balancing, scalability and efficiency close to one for large-scale problems.

In Chapter 5, computationally efficient serial and parallel algorithms for estimating the general linear model (GLM) have been proposed. The serial block Givens algorithm is an adaptation of a known recursive Givens strategy. It presents the GLM as a generalized linear least squares problem (GLLSP). The recursive algorithm is based on orthogonal factorization and uses the GQRD as a main computational tool. The triangular structure of the Cholesky factor of the variance-covariance matrix is efficiently exploited. The estimation of the GLM is derived recursively by solving a series of smaller and smaller GLLSP problems. This reduces significantly the computational burden of the standard estimation procedure. The new algorithm is found to be $m/3n$ times faster than the cor-

responding LAPACK routine for estimating the GLM, where m and n denote the number of observations and variables, respectively. A parallel approach based on the new sequential strategy has been also designed. It copies the main, sub- and super-block diagonals of the augmented Cholesky factor to all processors and distributes the remaining matrix evenly. The algorithm duplicates parts of the computations, but it gains from the minimal inter-processor communication. Overall, the parallel algorithm is found to be scalable and efficient for large-scale least-squares problems.

An adaptation of these parallel algorithms to estimate the updated-observation seemingly unrelated regressions model (SUR) have been proposed in Chapter 6. An efficient serial algorithm has been also developed. The best linear unbiased estimator of the SUR model has been computed after formulating the SUR as a GLLSP. The parallel algorithm exploits the block-sparse structure of the computed matrices and computational experiments indicated its scalability and efficiency.

7.1 Future research

Parallel algorithms to compute all possible subset models of the ordinary, general and SUR model have been considered. Furthermore, a branch and bound algorithm for computing the best subset regression models have been developed [15, 16, 17, 18]. The minimum spanning tree approach and the heuristic strategies that have been proposed in [46] can be adapted for the computation of all possible subset regression models. This promising approach merits investigation.

Computationally efficient methods for updating the SUR models with new observations have been proposed. The problems of adding and deleting exogenous variables from the SUR models have also been addressed [27]. The non straightforward problem of deleting observations from the SUR model (downdating) needs to be considered [10, 12, 27]. Special attention should be given to the numerical stability of the downdating algorithms [30]. Within the framework of investigating influential data, various methods to solve the SUR model after unequal number of observations have been added or deleted from some of the regressions should be investigated.

Furthermore, the application of the various techniques for up-downdating the LS problem and QRD should be investigated in the context of regression diagnostics and cross-validation, where repeatedly a number of observations is added and/or deleted [3, 27].

The estimation of the SUR model subject to linear constraints needs to be pursued. The estimation of these constrained models implies the solution of a particular quadratic programming problem. Existing procedures should be adapted to exploit the structure of the model. The special case of separable inequality constraints should be considered. Within this context the design of efficient sequential and parallel techniques for solving augmented systems which exploit the structure of the matrices need to be developed. Block-recursive algorithms based on the GQRD are currently under investigation. The algorithms should be adapted to handle the special structures of the matrices which are found in various estimation problems.

The various parallel algorithms need to be considered when the models are sparse and which is very often the case. Direct factorization strategies such as multifrontal QRD and iterative methods need to be considered. The problem of estimating ill-conditioned SUR models need to be addressed as well.

Bibliography

- [1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Englewood Cliffs NJ, 1993.
- [2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, 1992.
- [3] D. A. Belsley, E. Kuh, and R. E. Welsch. *Regression Diagnostics: Identifying Influential Observations and Sources of Collinearity*. John Wiley and Sons, 1980.
- [4] C. Bendtsen, C. Hansen, K. Madsen, H. B. Nielsen, and M. Pinar. Implementation of QR up- and downdating on a massively parallel computer. *Parallel Computing*, 21:49–61, 1995.
- [5] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.
- [6] W. M. Bolstad. An estimation of seemingly unrelated regression model with contemporaneous covariances based on an efficient recursive algorithm. *Comm. Statist. Simulation Comput.*, 16(3):689–698, 1987.
- [7] J.-P. Chavas. Recursive estimation of simultaneous equation models. *Journal of Econometrics*, 18:207–217, 1982.
- [8] L. Eldén and H. Park. Block downdating of least squares solutions. *SIAM Journal on Matrix Analysis and Applications*, 15(3):1018–1034, 1994.
- [9] L. Eldén and H. Park. Perturbation and error analyses for block downdating of a Cholesky decomposition. *BIT Numerical Mathematics*, 36(2):247–263, June 1996.

- [10] P. Foschi, D. Belsley, and E.J. Kontoghiorghes. A comparative study of algorithms for solving seemingly unrelated regressions models. *Computational Statistic & Data Analysis*, 44(1-2):3–35, 2003.
- [11] P. Foschi, L. Garin, and E. J. Kontoghiorghes. Numerical and computational methods for solving sur models. In E.J. Kontoghiorghes, B. Rustem, and S.Siokos, editors, *Computational Methods in Decision-Making, Economics and Finance*, volume 74 of *Applied Optimization*, pages 405–427. Kluwer Academic Publishers, 2002.
- [12] P. Foschi and E. J. Kontoghiorghes. Solution of seemingly unrelated regression models with unequal size of observations. *Computational Statistics and Data Analysis*, 41(1):211–229, 2002.
- [13] P. Foschi and E. J. Kontoghiorghes. Estimation of VAR models: computational aspects. *Computational Economics*, 21(1-2):3–22, 2003.
- [14] P. Foschi and E.J. Kontoghiorghes. Estimating seemingly unrelated regression models with vector autoregressive disturbances. *Journal of Economic Dynamics and Control*, 28(1):27–44, 2003.
- [15] C. Gatu and E. J. Kontoghiorghes. Parallel algorithms for computing all possible subset regression models using the QR decomposition. *Parallel Computing*, (29):505–521, 2003.
- [16] C. Gatu and E. J. Kontoghiorghes. Branch-and-bound algorithms for computing the best-subset regression models. *Journal of Computational and Graphical Statistics*, 2005. (Forethcoming).
- [17] C. Gatu and E. J. Kontoghiorghes. Efficient strategies for deriving the subset VAR models. *Computational Management Science*, 2005. (In press).
- [18] C. Gatu and E. J. Kontoghiorghes. Estimating all possible SUR models with permuted exogenous data matrices derived from a VAR process. *Journal of Economic Dynamics and Control*, 2005. (In press).
- [19] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28(126):505–535, 1974.

- [20] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, Maryland, 3ed edition, 1996.
- [21] G. H. Golub and J. H. Wilkinson. Note on the iterative refinement of least squares solution. *Numerische Mathematik*, 9:139–148, 1966.
- [22] I. Karasalo. A criterion for truncation of the QR decomposition algorithm for the singular linear least squares problem. *BIT*, 14:156–166, 1974.
- [23] E. J. Kontoghiorghes. Parallel strategies for computing the orthogonal factorizations used in the estimation of econometric models. *Algorithmica*, 25:58–74, 1999.
- [24] E. J. Kontoghiorghes. *Parallel Algorithms for Linear Models: Numerical Methods and Estimation Problems*, volume 15 of *Advances in Computational Economics*. Kluwer Academic Publishers, Boston, MA, 2000.
- [25] E. J. Kontoghiorghes. Parallel Givens sequences for solving the general linear model on a EREW PRAM. *Parallel Algorithms and Applications*, 15(1-2):57–75, 2000.
- [26] E. J. Kontoghiorghes. Parallel strategies for rank- k updating of the QR decomposition. *SIAM Journal on Matrix Analysis and Applications*, 22(3):714–725, 2000.
- [27] E. J. Kontoghiorghes. Computational methods for modifying seemingly unrelated regressions models. *Journal of Computational and Applied Mathematics*, 162(1):247–261, 2004.
- [28] E. J. Kontoghiorghes and M. R. B. Clarke. Solving the updated and downdated ordinary linear model on massively parallel SIMD systems. *Parallel Algorithms and Applications*, 1(2):243–252, 1993.
- [29] E. J. Kontoghiorghes and M. R. B. Clarke. An alternative approach for the numerical solution of seemingly unrelated regression equations models. *Computational Statistics & Data Analysis*, 19(4):369–377, 1995.
- [30] E. J. Kontoghiorghes and E. Dinenis. Computing 3SLS solutions of simultaneous equation models with a possible singular variance-covariance matrix. *Computational Economics*, 10:231–250, 1997.

- [31] J. B. Kruskal. On the shortest spanning tree of graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.
- [32] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice–Hall Englewood Cliffs, 1974.
- [33] S. J. Olszanskyj, J. M. Lebak, and A. W. Bojanczyk. Rank- k modification methods for recursive least squares problems. *Numerical Algorithms*, 7:325–354, 1994.
- [34] S. Orbe, E. Ferreira, and J. Rodriguez-Poo. An algorithm to estimate time varying parameter SUR models under different type of restrictions. *Computational Statistics & Data Analysis*, 42(3):363–383, 2003.
- [35] C. C. Paige. Numerically stable computations for general univariate linear models. *Communications on Statistical and Simulation Computation*, 7(5):437–453, 1978.
- [36] C. C. Paige. Computer solution and perturbation analysis of generalized linear least squares problems. *Mathematics of Computation*, 33(145):171–183, 1979.
- [37] C. C. Paige. Fast numerically stable computations for generalized linear least squares problems. *SIAM Journal on Numerical Analysis*, 16(1):165–171, 1979.
- [38] C. C. Paige. Some aspects of generalized QR factorizations. In M. G. Cox and S. J. Hammarling, editors, *Reliable Numerical Computation*, pages 71–91. Clarendon Press, Oxford, UK, 1990.
- [39] D. S. G. Pollock. *The Algebra of Econometrics (Wiley series in Probability and Mathematical Statistics)*. John Wiley and Sons, 1979.
- [40] R. C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technical Journal*, 36:1389–1401, 1957.
- [41] P. A. Regalia and S. K. Mitra. Kronecker products, unitary matrices and signal processing applications. *SIAM Review*, 31(4):586–613, 1989.
- [42] J. Shao. Linear model selection by cross-validation. *Journal of the American Statistical Association*, 88:486–494, 1993.
- [43] V. K. Srivastava and T. D. Dwivedi. Estimation of seemingly unrelated regression equations Models: a brief survey. *Journal of Econometrics*, 10:15–32, 1979.

- [44] V. K. Srivastava and D. E. A. Giles. *Seemingly Unrelated Regression Equations Models: Estimation and Inference (Statistics: Textbooks and Monographs)*, volume 80. Marcel Dekker, Inc., 1987.
- [45] L. G. Telser. Iterative estimation of a set of linear regression equations. *Journal of the American Statistical Association*, 59:845–862, 1964.
- [46] P. Yanev, P. Foschi, and E. J. Kontoghiorghes. Algorithms for computing the QR decomposition of a set of matrices with common columns. *Algorithmica*, 39:83–93, 2004.
- [47] P. Yanev and E. J. Kontoghiorghes. Efficient algorithms for block downdating of least squares solutions. *Applied Numerical Mathematics*, 49(1):3–15, 2004.
- [48] P. Yanev and E. J. Kontoghiorghes. Parallel algorithms for downdating the QR decomposition. *Parallel Computing*, 2004. (Submitted).
- [49] P. Yanev and E. J. Kontoghiorghes. Computationally efficient methods for estimating the updated-observations SUR models. *Applied Numerical Mathematics*, 2005. (Submitted).
- [50] P. Yanev and E. J. Kontoghiorghes. Efficient algorithms for estimating the general linear model. *Parallel Computing*, 2005. (Forethcoming).
- [51] A. Zellner. An efficient method of estimating seemingly unrelated regression equations and tests for aggregation bias. *Journal of the American Statistical Association*, 57:348–368, 1962.