Algorithms for statistical model selection and robust estimation

Thèse présentée à la Faculté des Sciences

Institut d'Informatique

Université de Neuchâtel

Pour l'obtention du grade de docteur ès sciences

Par

Marc Hofmann

Acceptée sur proposition du jury:

Prof. Peter Kropf, directeur de thèse
Université de Neuchâtel, Suisse

Prof. Erricos J. Kontoghiorghes, co-directeur de thèse
University of Cyprus and Queen Mary, University of London, UK

Prof. Manfred Gilli, rapporteur
Université de Genève, Suisse

Prof. Dietmar Maringer, rapporteur
Universität Basel, Schweiz

Dr Irini Moustaki, rapporteur
London School of Economics, UK

Soutenue le 19 mars 2009

Université de Neuchâtel

2009

Université de Neuchâtel

FACULTE DES SCIENCES
Secrétariat-Décanat de la faculté
- Rue Emile-Argand 11
- CP 158
- CH-2009 Neuchâtel

IMPRIMATUR POUR LA THESE

Algorithms for statistical model selection and robust estimation

Marc HOFMANN

UNIVERSITE DE NEUCHATEL

FACULTE DES SCIENCES

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

Mme I .Moustaki (Londres UK),
MM. P. Kropf (directeur de thèse), E. Kontoghiorghes (co-directeur de thèse),
D. Maringer (Bâle) et M. Gilli (Genève)

autorise l'impression de la présente thèse.

Neuchâtel, le 27 août 2009

Le doyen :
F. Kessler

UNIVERSITE DE NEUCHATEL
FACULTE DES SCIENCES
Secrétariat - décanat de la faculté
Rue Emile-Argand 11 - CP 158
CH-2009 Neuchâtel
Félix Kessler

A Roland, Maja, Catia et Corinne.

Acknowledgements

I wish to thank my Ph.D. advisor Erricos John Kontoghiorghes for his precious guidance, Paolo Foschi, Cristian Gatu and Petko Yanev for their help and support. My thanks are extended to Manfred Gilli for his useful input and kind assistance. I am grateful to Maurice Clint, Ahmed Sameh, Peter J. Rousseeuw, Jose Agulló and D. S. G. Pollock for their valuable comments and suggestions. I am endebted to the Computer Science Department of the University of Neuchâtel and its staff, in particular to the head of department Peter Kropf, for hosting me. This work is in part supported by the Swiss National Science Foundation grant 101412-105978.

Abstract

Computationally intensive algorithms for model selection and robust regression are considered. Particular emphasis is put on regression trees. The QR decomposition is the main computational tool to solve the linear models. Givens rotations are employed to compute the orthogonal factorizations. A new pipeline-parallel strategy is proposed for computing the QR decomposition. Algorithms for computing the best subset regression models are investigated. The algorithms extend previously introduced exhaustive and heuristic strategies, which are aimed at solving large-scale model selection problems. An algorithm is proposed to compute the exact least trimmed squares regression. It can efficiently compute the LTS estimators for a range of coverage values. Thus, the coverage parameter h does not need to be known in advance, and the algorithm can be used to examine the degree of contamination of the data. The LTS algorithm is extended to solve the generalized LTS estimation problem of the GLM and SUR model. The singularity problem of the dispersion matrix is avoided by reformulating the estimation problem as a generalized linear least squares problem.

Keywords. Matrix factorizations, parallel algorithms, least squares, model selection, robust regression, regression trees, general linear model, seemingly unrelated regressions.

Contents

1 Introduction **1**

1.1 Linear regression . . . 2
1.2 The ordinary linear model . . . 3
1.3 The general linear model . . . 4
1.4 The seemingly unrelated regressions model . . . 5
1.5 Subset model selection . . . 6
1.6 Least trimmed squares robust regression . . . 6
1.7 The QR decomposition . . . 7
1.7.1 The Householder method . . . 8
1.7.2 The Givens method . . . 8
1.7.3 Givens schemes . . . 9
1.8 Research overview . . . 10

2 Pipeline Givens sequences for computing the QR decomposition on an EREW PRAM **13**

2.1 Introduction . . . 13
2.2 Pipeline-parallel SK sequence . . . 14
2.3 New pipeline-parallel Givens sequences . . . 15
2.4 Conclusions . . . 20

3 Efficient algorithms for computing the best subset regression models for large-scale problems **21**

3.1 Introduction . . . 21
3.2 Subrange model selection . . . 23
3.3 Radius preordering . . . 26
3.4 Heuristic strategies . . . 30
3.5 Conclusions . . . 32
3.A Subrange model selection: complexity analysis . . . 33

4 An exact least trimmed squares algorithm for a range of coverage values **37**

4.1 Introduction . . . 37
4.2 Adding row algorithm . . . 39
4.2.1 Experimental results . . . 41
4.3 Branch and bound algorithm . . . 42

4.4 Conclusions 46
4.A Formal proofs 48

5 Matrix strategies for computing the least trimmed squares estimation of the general linear and SUR models **49**

5.1 Introduction 49
5.2 The adding row algorithm 51
5.3 GLTS estimation of the GLM 52
5.3.1 Experimental results 55
5.4 GLTS estimation of the SUR model 56
5.4.1 Experimental results 60
5.5 Conclusions 61

6 Conclusions **63**

Abbreviations **67**

Bibliography **68**

Chapter 1

Introduction

The general goal of regression analysis is to identify the relationship, the so-called model, between observed variables based on numerical data. In this context, linear least squares (LS) problems arise in various applications such as statistics, econometrics, optimization and signal processing. Research into the development of numerically stable and computationally efficient methods for solving LS problems has been active for more than fifty years (Björck 1996, Golub & Wilkinson 1966, Karasalo 1974, Lawson & Hanson 1974). But, the predictive power of a linear model may lessen if it contains insignificant variables. This is more likely to happen as the number of variables grows. Identifying relevant variables is a fundamental problem in statistical modeling. Some methods for variable selection have exponential complexities and are computationally very demanding. Furthermore, LS estimation is vulnerable to disturbances in the data, which can stem from errors in measuring or recording the data. Therefore, robust regression methods have been designed to circumvent the limitations of traditional linear regression methods in the presence of such outlying data points. Yet, they have not been widely used, in part because they are much more computationally intensive than simple LS estimation.

Likewise, matrix computations play an important role in numerical statistics. The QR decomposition (QRD) method is one of the main computational tools in regression analysis (Björck 1996, Golub & Van Loan 1996, Lawson & Hanson 1974). It is mainly used in the solution of linear equations systems and is often associated with LS problems. The QRD can be conveniently updated when a model needs to be re-estimated consecutive to the addition or deletion of a variable or observation. In order to overcome the computational burden of demanding statistical problems, parallel algorithms have been proposed. The development of parallel processing methods requires strong technical skills in statistics, high performance computing and numerical methods.

The present work is a contribution to the continuing effort toward the development of efficient and reliable algorithms used in computational statistics. The challenge at hand is to tackle the combinatorial complexity of subset model selection and robust regression problems by bringing together the expertise of numerical computing, algorithmic design and statistics.

1.1 Linear regression

The goal of regression analysis is to identify the relationship between observed variables based on numerical data. The regression equation

$$y = f(X, \beta) + \varepsilon \tag{1.1}$$

expresses a relationship between a dependent (endogenous) response variable y and independent (exogenous) predictor variables $X = (x_1, \ldots, x_n)$. The error term ε accounts for the contribution of unmeasured variables. The unobservable parameters $\beta = (\beta_1, \ldots, \beta_n)$ are estimated so as to give the "best fit". The quality of a fit is measured by an objective function. The goal is then to find an estimator $\widehat{\beta}$ which satisfies the following optimization problem:

$$\widehat{\beta} = \underset{\beta}{\operatorname{argmin}}\, Q_f(y, X, \beta),$$

where Q_f is the objective function. Most commonly, the fit is evaluated using the least squares (LS) method, in which case the objective function is given by

$$Q_f(y, X, \beta) = \|y - f(X, \beta)\|^2,$$

where $\| \bullet \|$ denotes the Euclidean norm. The difference between the observed dependent variable y and its estimated value $\widehat{y} = f(X, \widehat{\beta})$ is called the residual and is denoted by $\widehat{\varepsilon}$. The square norm $\|\widehat{\varepsilon}\|^2$ is the residual sum of squares (RSS) of $\widehat{\beta}$. Thus, the aim of the LS method is to minimize the RSS.

Linear regression analysis supposes the relationship f between the variables to be linear. Thus (1.1) may be written

$$y = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon.$$

If there are m $(m > n)$ sample points, then the relationship can be written in matrix form as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \ldots & x_{mn} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{bmatrix},$$

where y_i and ε_i are random variables and x_{ij} is a "known" (i.e. observable and non-random) quantity $(1 \leq i \leq m,\, 1 \leq j \leq n)$. In compact form, the latter can be written

$$y = X\beta + \varepsilon, \tag{1.2}$$

where y, $\varepsilon \in \mathbb{R}^m$, $X \in \mathbb{R}^{m \times n}$ and $\beta \in \mathbb{R}^n$. The first assumption concerning the disturbance ε is that its expected value is zero, that is $\mathrm{E}(\varepsilon) = 0$. The second assumption is that the dispersion matrix of ε is $\sigma^2\Omega$, where Ω is a symmetric non-negative definite matrix and σ is an unknown scalar. The final assumption is that X is a non-stochastic matrix which implies $\mathrm{E}(X^{\mathrm{T}}\varepsilon) = 0$. In summary, the complete mathematical specification of the linear model is given by

$$y = X\beta + \varepsilon, \quad \varepsilon \sim (0, \sigma^2\Omega), \tag{1.3}$$

where the notation $\varepsilon \sim (0, \sigma^2\Omega)$ indicates that the disturbance vector ε comes from a distribution having zero mean and variance-covariance matrix $\sigma^2\Omega$ (Rao & Toutenburg 1995). Having observed the values of X and y, the estimates of β and σ are to be determined.

1.2 The ordinary linear model

Consider the ordinary linear model (OLM)

$$y = X\beta + \varepsilon, \quad \varepsilon \sim (0, \sigma^2 I_m), \tag{1.4}$$

where $y \in \mathbb{R}^m$ is the response vector, $X \in \mathbb{R}^{m\times n}$ is the full rank exogenous matrix, $\beta \in \mathbb{R}^n$ are the coefficients to be estimated and $\varepsilon \in \mathbb{R}^m$ is the disturbance term. The OLM assumptions are that the errors have expectation zero, have equal variances and are uncorrelated. That is, $\mathrm{E}(\varepsilon_i) = 0$, $\mathrm{Var}(\varepsilon_i) = \sigma^2$ and $\mathrm{Cov}(\varepsilon_i, \varepsilon_j) = 0$ if $i \neq j$ $(1 \leq i, j \leq m)$.

The ordinary least squares (OLS) estimator of β in (1.4) is given by

$$(\widehat{\beta}, \widehat{\varepsilon}) = \underset{\beta,\varepsilon}{\operatorname{argmin}} \|\varepsilon\|^2 \quad \text{subject to} \quad \varepsilon = y - X\beta.$$

This minimization problem has a unique solution given by the normal equations

$$X^{\mathrm{T}} X\widehat{\beta} = X^{\mathrm{T}} y.$$

The normal equations are derived from the property that the square norm of the residual vector $\widehat{\varepsilon}$ is minimal when its gradient with respect to each parameter β_i is zero, i.e. $\partial\widehat{\varepsilon}^{\mathrm{T}}\widehat{\varepsilon}/\partial\beta_i = 0$ for all $i = 1, \ldots, n$. The algebraic solution of the normal equations may be written

$$\widehat{\beta} = (X^{\mathrm{T}} X)^{-1} X^{\mathrm{T}} y,$$

assuming that X has full rank.

For the solution of the normal equations, consider the QR decomposition of the explanatory data matrix X:

$$Q^{\mathrm{T}} X = \begin{bmatrix} R \\ 0 \end{bmatrix} \quad \text{and} \quad Q^{\mathrm{T}} y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \tag{1.5}$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal (i.e. $QQ^{\mathrm{T}} = Q^{\mathrm{T}}Q = I_m$) and $R \in \mathbb{R}^{n\times n}$ is upper triangular. The OLS estimator of β in (1.4) is derived by minimizing $\|\varepsilon\|^2 = \|y - X\beta\|^2$, i.e.

$$\begin{aligned}
\widehat{\beta} &= \underset{\beta}{\operatorname{argmin}} \|y - X\beta\|^2 \\
&= \underset{\beta}{\operatorname{argmin}} \|Q^{\mathrm{T}} y - Q^{\mathrm{T}} X\beta\|^2 \\
&= \underset{\beta}{\operatorname{argmin}} (\|y_1 - R\beta\|^2 + \|y_2\|^2) \\
&= R^{-1} y_1.
\end{aligned}$$

Notice that the residual sum of squares of $\widehat{\beta}$ is

$$\mathrm{RSS}(\widehat{\beta}) = \|y - X\widehat{\beta}\|^2 = \|y_2\|^2.$$

Furthermore, the Cholesky decomposition of $X^{\mathrm{T}} X$ is $R^{\mathrm{T}} R$ (Björck 1996, Golub & Van Loan 1996).

The Gauss-Markov theorem states that the OLS estimator is the best linear unbiased estimator (BLUE) of β in (1.4). That is, $\mathrm{E}(\widehat{\beta}) = \beta$ and $\mathrm{Var}(\widetilde{\beta}) - \mathrm{Var}(\widehat{\beta})$ is a positive semidefinite matrix for any other linear unbiased estimator $\widetilde{\beta}$ (Searle 1971).

1.3 The general linear model

The general linear model (GLM) is given in (1.3), i.e.

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

where Ω is a known, positive definite matrix. Contrary to the OLM, the errors are correlated, that is: $\forall(i,j), \mathrm{Cov}(\varepsilon_i, \varepsilon_j) = \sigma^2\Omega_{i,j} \neq 0$. Assuming that Ω is non-singular, the BLUE of β is the solution of the generalized least squares (GLS) problem

$$(\widehat{\beta}, \widehat{\varepsilon}) = \underset{\beta,\varepsilon}{\operatorname{argmin}} \|\varepsilon\|^2_{\Omega^{-1}}, \quad \text{where} \quad \varepsilon = y - X\beta. \tag{1.6}$$

The normal equations are given by

$$X^{\mathrm{T}}\Omega^{-1}X\beta = X^{\mathrm{T}}\Omega^{-1}y$$

and have the algebraic solution

$$\widehat{\beta} = (X^{\mathrm{T}}\Omega^{-1}X)^{-1}X^{\mathrm{T}}\Omega^{-1}y.$$

This solution is computationally expensive and numerically unstable when Ω is ill-conditioned (Björck 1996, Lawson & Hanson 1974). Notice that the GLS estimator does not exist when Ω is singular.

In order to avoid problems associated with a singular or ill conditioned Ω, the GLS (1.6) can be reformulated as a generalized linear least squares problem (GLLSP):

$$(\widehat{\beta}, \widehat{u}) = \underset{\beta,u}{\operatorname{argmin}} \|u\|^2 \quad \text{subject to} \quad y = X\beta + Bu. \tag{1.7}$$

Here, $\Omega \in \mathbb{R}^{m\times m}$ is positive semidefinite with rank k, $B \in \mathbb{R}^{m\times k}$ has full column rank such that $\Omega = BB^{\mathrm{T}}$, and $u \in \mathbb{R}^k$ is defined by $Bu = \varepsilon$. That is, $u \sim (0, \sigma^2 I_k)$ (Kourouklis & Paige 1981). Without loss of generality, consider the case where Ω is non-singular. For the solution of the GLLSP (1.7), the generalized QR decomposition (GQRD) can be employed. The GQRD of X and B is given by the QR decomposition (1.5) and the RQ decomposition of $Q^{\mathrm{T}}B$, i.e.

$$Q^{\mathrm{T}}X = \begin{bmatrix} R \\ 0 \end{bmatrix} \quad \text{and} \quad (Q^{\mathrm{T}}B)P^{\mathrm{T}} = T = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix},$$

where $P \in \mathbb{R}^{m\times m}$ is orthogonal, and $T \in \mathbb{R}^{m\times m}$ is upper triangular and non-singular. The GLLSP (1.7) is equivalent to

$$(\widehat{\beta}, \widehat{u}) = \underset{\beta,u}{\operatorname{argmin}} \|Pu\|^2 \quad \text{subject to} \quad Q^{\mathrm{T}}y = Q^{\mathrm{T}}X\beta + Q^{\mathrm{T}}BP^{\mathrm{T}}Pu$$

and can be written

$$(\widehat{\beta}, \widehat{u}_1, \widehat{u}_2) = \underset{\beta,u_1,u_2}{\operatorname{argmin}}(\|u_1\|^2 + \|u_2\|^2) \quad \text{subject to} \quad \begin{cases} y_1 = R\beta + T_{11}u_1 + T_{12}u_2, \\ y_2 = T_{22}u_2, \end{cases} \tag{1.8}$$

where $u^{\mathrm{T}}P^{\mathrm{T}} = \begin{bmatrix} u_1^{\mathrm{T}} & u_2^{\mathrm{T}} \end{bmatrix}$. It follows from the second constraint of (1.8) that $\widehat{u}_2 = T_{22}^{-1}y_2$. In the first constraint, the arbitrary subvector u_1 may be set to zero to minimize the objective function, i.e. $\widehat{u}_1 = 0$. Thus, the estimator of β derives from the solution of the upper triangular system $R\widehat{\beta} = \widehat{z}$, where $\widehat{z} = y_1 - T_{12}\widehat{u}_2$. The generalized RSS of $\widehat{\beta}$ is given by $\mathrm{GRSS}(\widehat{\beta}) = \|\widehat{u}_2\|^2$. The variance-covariance of the coefficient estimator is given by $\widehat{\sigma}^2 R^{-\mathrm{T}}T_{11}^{\mathrm{T}}T_{11}R^{-1}$, where $\widehat{\sigma}^2 = \|\widehat{u}_2\|^2/(m-n)$ is an estimator of σ^2.

1.4 The seemingly unrelated regressions model

The seemingly unrelated regressions (SUR) model is a special case of the GLM and is defined by the set of G equations

$$y^{(i)} = X^{(i)}\beta^{(i)} + \varepsilon^{(i)}, \quad i = 1, \ldots, G,$$

where $y^{(i)} \in \mathbb{R}^m$ are the response vectors, $X^{(i)} \in \mathbb{R}^{m \times n_i}$ are the exogenous matrices with full column rank, $\beta^{(i)} \in \mathbb{R}^{n_i}$ are the coefficients, and $\varepsilon^{(i)} \in \mathbb{R}^m$ are the errors. Furthermore, $\mathrm{E}(\varepsilon_t^{(i)}) = 0$, $\mathrm{Cov}(\varepsilon_t^{(i)}, \varepsilon_t^{(j)}) = \sigma_{i,j}$ and $\mathrm{Cov}(\varepsilon_s^{(i)}, \varepsilon_t^{(j)}) = 0$ if $s \neq t$ (Kontoghiorghes 2000*a*, Kontoghiorghes & Clarke 1995, Srivastava & Giles 1987, Zellner 1962). In compact form, the SUR model may be written

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(G)} \end{bmatrix} = \begin{bmatrix} X^{(1)} & & & \\ & X^{(2)} & & \\ & & \ddots & \\ & & & X^{(G)} \end{bmatrix} \begin{bmatrix} \beta^{(1)} \\ \beta^{(2)} \\ \vdots \\ \beta^{(G)} \end{bmatrix} + \begin{bmatrix} \varepsilon^{(1)} \\ \varepsilon^{(2)} \\ \vdots \\ \varepsilon^{(G)} \end{bmatrix}$$

or

$$\mathrm{vec}(Y) = \left(\bigoplus_{i=1}^{G} X_i \right) \mathrm{vec}(\{\beta^{(i)}\}_G) + \mathrm{vec}(E), \tag{1.9}$$

where $Y = \left[y^{(1)} \cdots y^{(G)} \right]$, $E = \left[\varepsilon^{(1)} \cdots \varepsilon^{(G)} \right]$, $\{\beta^{(i)}\}_G$ is the ordered set of vectors $\beta^{(1)}$, ..., $\beta^{(G)}$, vec($\bullet$) denotes the vector stack operator and $\bigoplus$ the direct sum of matrices. The vector stack operator stacks a set of vectors or the columns of a matrix, e.g.

$$\mathrm{vec}(\{\beta^{(i)}\}_G) = \begin{bmatrix} \beta^{(1)} \\ \beta^{(2)} \\ \vdots \\ \beta^{(G)} \end{bmatrix} \quad \text{and} \quad \mathrm{vec}(Y) = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(G)} \end{bmatrix}.$$

The direct sum of matrices $\bigoplus_{i=1}^{G} X^{(i)}$ defines the $M \times N$ block-diagonal matrix

$$\begin{aligned} \bigoplus_{i=1}^{G} X^{(i)} &= X^{(1)} \oplus X^{(2)} \oplus \ldots \oplus X^{(G)} \\ &= \begin{bmatrix} X^{(1)} & & & \\ & X^{(2)} & & \\ & & \ddots & \\ & & & X^{(G)} \end{bmatrix}, \end{aligned}$$

where $M = Gm$ and $N = \sum_{i=1}^{G} n_i$ (Regalia & Mitra 1989).

The error term vec(E) in (1.9) has zero mean and variance-covariance matrix $\Sigma \otimes I_m$, where $\Sigma = \left[\sigma_{ij} \right] \in \mathbb{R}^{G \times G}$ is symmetric positive definite, and $\otimes$ denotes the Kronecker product. That is, $\mathrm{vec}(E) \sim (0, \Sigma \otimes I_m)$ and

$$\Sigma \otimes I_m = \begin{bmatrix} \sigma_{11} I_m & \sigma_{12} I_m & \cdots & \sigma_{1G} I_m \\ \sigma_{21} I_m & \sigma_{22} I_m & \cdots & \sigma_{2G} I_m \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{G1} I_m & \sigma_{G2} I_m & \cdots & \sigma_{GG} I_m \end{bmatrix}.$$

Notice that $(A \otimes B)(C \otimes D) = AC \otimes BD$ and $\operatorname{vec}(ABC) = (C^{\mathrm{T}} \otimes A)\operatorname{vec}(B)$ (Kmenta & Gilbert 1968, Kontoghiorghes & Clarke 1993, Kontoghiorghes & Dinenis 1996, Revankar 1974, Schmidt 1978, Srivastava & Dwivedi 1979, Srivastava & Giles 1987, Zellner 1962, Zellner 1963).

1.5 Subset model selection

The problem of computing the best subset regression models arises in statistical model selection. Most of the criteria used to evaluate the subset models rely upon the residual sum of squares (Searle 1971, Sen & Srivastava 1990). Consider the standard regression model (1.4). The columns of the matrix X correspond to the exogenous variables $X = (x_1, \ldots, x_n)$. A submodel S comprises some of the variables in X. There are $\sum_{i=1}^{n} \binom{n}{i} = 2^n - 1$ possible subset models, and their computation is only feasible for small values of n. The dropping column algorithm (DCA) derives all submodels by generating a regression tree (Clarke 1981, Gatu & Kontoghiorghes 2003, Smith & Bremner 1989). The parallelization of the DCA only slightly improves its practical value (Gatu & Kontoghiorghes 2003).

Various procedures such as the forward, backward and stepwise selection try to identify a subset by inspecting very few combinations of variables. However, these methods rarely succeed in finding the best submodel (Hocking 1976, Seber 1977). Other approaches for subset selection include ridge regression, the Non-negative Garrote and the Lasso (Breiman 1995, Fan & Li 2001, Tibshirani 1996). Sequential replacement algorithms are fairly fast and can be used to give some indication of the maximum size of the subsets that are likely to be of interest (Hastie, Tibshirani & Friedman 2001). Branch and bound algorithms for choosing a subset of k features from a given larger set of size n have been investigated within the context of feature selection problems (Narendra & Fukunaga 1977, Roberts 1984, Somol, Pudil & Kittler 2004). These strategies are used when the subset size k of interest is known. Thus, the search space consists of $\binom{n}{k} = n!/(k!(n-k)!)$ subsets.

A computationally efficient branch and bound algorithm (BBA) has been devised (Gatu & Kontoghiorghes 2006, Gatu, Yanev & Kontoghiorghes 2007). The BBA avoids the computation of the whole regression tree. It derives the best subset model for each number of variables; that is, it computes

$$\operatorname*{argmin}_{S} \operatorname{RSS}(S) \quad \text{subject to} \quad |S| = k \quad \text{for all} \quad k = 1, \ldots, n.$$

The BBA is built around the fundamental property

$$\operatorname{RSS}(S_1) \geq \operatorname{RSS}(S_2) \quad \text{if} \quad S_1 \subseteq S_2, \tag{1.10}$$

where S_1 and S_2 are two variable subsets of X (Gatu & Kontoghiorghes 2006). The BBA-1, which is an extension of the BBA, preorders the n variables in the root node according to their strength. The variables i and j are arranged such that $\operatorname{RSS}(X - \{x_i\}) \geq \operatorname{RSS}(X - \{x_j\})$ for each $i \leq j$, where $X - \{x_i\}$ denotes the set (matrix) X from which the ith variable (column) has been deleted. The BBA-1 has been shown to outperform the previously introduced leaps and bounds algorithm (LBA) (Furnival & Wilson 1974).

1.6 Least trimmed squares robust regression

Least squares regression is sensitive to outliers. This has prompted the search for regression estimators which are resistant to data points that deviate from the usual assumptions. The goal of positive-breakdown

estimators is to be robust against the possibility of unannounced outliers (Rousseeuw 1997). The breakdown point provides a crude quantification of the robustness properties of an estimator. Briefly, the breakdown point is the smallest amount of contamination that may cause an estimator to take on arbitrarily large aberrant values (Donoho & Huber 1983).

Consider the standard regression model (1.4). Least squares regression consists in minimizing the residual sum of squares. One outlier may be sufficient to compromise the LS estimator. In other words, the finite-sample breakdown point of the LS estimator is $1/m$ and therefore tends to 0 when m is large (Rousseeuw 1997). Several positive-breakdown methods for robust regression have been proposed, such as the least median of squares (LMS) (Rousseeuw 1984). The LMS is defined by minimizing $\text{med}_i\, \widehat{\varepsilon}_i^2$. The LMS attains the highest possible breakdown value, namely $(\lfloor (m-n)/2 \rfloor + 1)/m$. This means that the LMS fit stays in a bounded region whenever $\lfloor (m-n)/2 \rfloor$ or fewer observations are replaced by arbitrary points (Rousseeuw & Van Driessen 2006).

The least trimmed squares (LTS) estimator possesses better theoretical properties than the LMS (Rousseeuw & Van Driessen 2006, Hössjer 1994). The objective of the LTS estimator is to minimize $\sum_{i=1}^{h} \widehat{\varepsilon}_{[i]}^2$, where $h \in \{1, \ldots, m\}$, and $\widehat{\varepsilon}_{[1]}^2 \leq \cdots \leq \widehat{\varepsilon}_{[m]}^2$ denote the ordered squared residuals. This is equivalent to finding the h-subset of observations with the smallest LS objective function. The LTS regression estimate is then the LS fit to these h points. The breakdown value of LTS with $h = \lfloor (m+n+1)/2 \rfloor$ is equivalent to that of the LMS. In spite of its advantages over the LMS estimator, the LTS estimator has been applied less often because it is computationally demanding. For the multiple linear regression model, the trivial algorithm that explicitly enumerates and computes the RSS for all h-subsets works if the number of observations is relatively small, i.e. less than 30. Otherwise, the computational load is prohibitive. To overcome this drawback, several approximate algorithms have been proposed. These include the Progress algorithm (Rousseeuw & Leroy 1987), the feasible solution algorithm (FSA) (Hawkins 1994, Hawkins & Olive 1999) and the Fast LTS algorithm (Rousseeuw & Van Driessen 2006). However, these algorithms do not inspect all combinations of observations and are not guaranteed to find the optimal solution. An exact algorithm to calculate the LTS estimator has been proposed (Agulló 2001). It is based on a branch and bound procedure that does not require the explicit enumeration of all h-subsets. It therefore reduces the computational load of the trivial algorithm significantly. A tree algorithm to enumerate subsets of observations has been suggested in the context of outlier detection (Belsley, Kuh & Welsch 1980).

1.7 The QR decomposition

The QR decomposition (QRD) is one of the main computational tools in regression (Björck 1984, Björck 1996, Businger & Golub 1965, Fausett, Fulton & Hashish 1997, Golub & Van Loan 1996, Gulliksson & Wedin 1992, Lawson & Hanson 1974, Smith 1991). It is mainly used in solutions of LS problems (Belsley et al. 1980, Golub & Wilkinson 1966, Karasalo 1974, Lawson & Hanson 1974). Different methods have been proposed for forming the QRD (1.5). It is equivalent to

$$Q^{\mathrm{T}} X = \begin{bmatrix} R \\ 0 \end{bmatrix} \quad \text{or} \quad X = Q_1 R, \tag{1.11}$$

where $X \in \mathbb{R}^{m\times n}$, $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \in \mathbb{R}^{m\times m}$ is orthogonal, $Q_1 \in \mathbb{R}^{m\times n}$, $Q_2 \in \mathbb{R}^{m\times(m-n)}$, $R \in \mathbb{R}^{n\times n}$ is upper triangular and $m > n$. It is assumed that X has full column rank. The two main methods for computing the QRD are based on Householder reflectors and Givens rotations (Golub & Van Loan 1996, Kontoghiorghes 2000*a*, Trefethen & Bau 1997).

1.7.1 The Householder method

An $m \times m$ Householder transformation has the form

$$H = I_m - 2\frac{vv^{\mathrm{T}}}{\|v\|^2},$$

where $v \in \mathbb{R}^m$. Householder matrices are symmetric and orthogonal. They can be used to annihilate specified elements of a vector or a matrix (Björck 1996, Golub & Van Loan 1996). Specifically, let $x \in \mathbb{R}^m$ be non-zero. A Householder matrix H can be chosen such that $y = Hx$ has zero-elements in positions $2, \ldots, m$ by setting $v = x \pm \|x\|^2 e_1$, where e_1 denotes the first column of the identity matrix I_m.

For the computation of the QRD (1.11), a sequence of n Householder transformations is applied. The orthogonal matrix Q is defined as $Q^{\mathrm{T}} = H_n \cdots H_2 H_1$. The ith Householder transformation is of the form

$$H_i = \begin{bmatrix} I_{i-1} & 0 \\ 0 & \widetilde{H}_i \end{bmatrix},$$

where $\widetilde{H}_i = I_{m-i+1} - 2v_i v_i^{\mathrm{T}}/\|v_i\|^2$, and v_i is the ith element of the Householder vector v. Let $X^{(0)} = X$ and

$$X^{(i)} = H_i X^{(i-1)} = \begin{bmatrix} R_{11}^{(i)} & R_{12}^{(i)} \\ 0 & \widetilde{X}^{(i)} \end{bmatrix}, \quad i = 1, \ldots, n,$$

where $R_{11}^{(i)} \in \mathbb{R}^{i\times i}$ is upper triangular, $R_{12}^{(i)} \in \mathbb{R}^{i\times(n-i)}$ and $\widetilde{X}^{(i)} \in \mathbb{R}^{(m-i)\times(n-i)}$. The transformation H_{i+1} is such that it only affects $\widetilde{X}^{(i)}$ and annihilates the last $(m-i+1)$ elements of the first column of $\widetilde{X}^{(i)}$. It follows that

$$X^{(n)} = \begin{bmatrix} R \\ 0 \end{bmatrix}.$$

1.7.2 The Givens method

An $m \times m$ Givens rotation is an orthogonal matrix

$$G_{i,j}^{(\theta)} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \cdots & s & & \\ & & \vdots & \ddots & \vdots & & \\ & & -s & \cdots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix},$$

where $c = \cos\theta$ and $s = \sin\theta$ (Golub & Van Loan 1996). That is, the matrix $G_{i,j}^{(\theta)} = \begin{bmatrix} g_{ij} \end{bmatrix}$ is the identity matrix with the following substitutions:

$$g_{ii} = c, \qquad g_{ij} = s,$$
$$g_{ji} = -s, \qquad g_{jj} = c.$$

Note that $c^2 + s^2 = 1$.

When applied from the left of a matrix X, the rotation $G_{i,j}^{(\theta)}$ only affects the elements on the ith and jth rows. It can be used to annihilate the element $X_{j,k}$, in which case it will be denoted by $G_{i,j}^{(k)}$. Specifically, the transformation

$$\widetilde{X} = G_{i,j}^{(k)} X,$$

where X, $\widetilde{X} \in \mathbb{R}^{m \times n}$ and $G_{i,j}^{(k)} \in \mathbb{R}^{m \times m}$ $(1 \leq k \leq n)$, results in

$$\widetilde{X}_{\ell,:} = \begin{cases} cX_{i,:} + sX_{j,:} & \text{if } \ell = i, \\ cX_{j,:} - sX_{i,:} & \text{if } \ell = j, \\ X_{\ell,:} & \text{otherwise.} \end{cases}$$

Here, $X_{i,:}$ denotes the ith row of the matrix X (Golub & Van Loan 1996). It follows that $\widetilde{X}_{j,k} = 0$ if $c = X_{i,k}/t$ and $s = X_{j,k}/t$, where $t^2 = X_{i,k}^2 + X_{j,k}^2$.

In practice, Givens rotations are not actually applied by constructing the entire Givens matrix. It is much more efficient to employ a Givens procedure which takes advantage of the sparse structure of the transformation matrix. While the Householder method to compute the QRD is more efficient, Givens sequences are particularly useful to selectively annihilate matrix elements, for example to update a precomputed QRD after it has been modified by a row or column (Gill, Golub, Murray & Saunders 1974).

1.7.3 Givens schemes

Consider the QR decomposition in (1.11). The triangular matrix R is derived iteratively from $Q_i^T \widetilde{X}_i = \widetilde{X}_{i+1}$, where $\widetilde{X}_0 = X$ and Q_i is orthogonal. The triangularization process terminates when $\widetilde{X}_\nu$ $(\nu > 0)$ is upper triangular. Here, $Q = Q_0 Q_1 \cdots Q_\nu$ is not computed explicitly. The triangular R can be derived by employing a sequence of Givens rotations. The Givens rotation that annihilates the element $X_{i,j}$ when applied from the left of X has the form

$$G_{i,j} = \text{diag}(I_{i-2}, \widetilde{G}_{i,j}, I_{m-i}), \quad \text{with} \quad \widetilde{G}_{i,j} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix},$$

where $c = X_{i-1,j}/t$, $s = X_{i,j}/t$ and $t^2 = X_{i-1,j}^2 + X_{i,j}^2$ $(\neq 0)$. A Givens rotation only affects the ith and $(i-1)$th rows of X. Thus, $\lfloor m/2 \rfloor$ rotations can be applied simultaneously. A compound disjoint givens rotation (CDGR) comprises rotations that can be applied simultaneously. Parallel algorithms for computing the QR decomposition based on CDGRs have been developed (Cosnard & Daoudi 1994, Cosnard, Muller & Robert 1986, Kontoghiorghes 2002, Luk 1986, Modi 1988, Sameh 1985, Sameh & Brent 1971, Sameh & Kuck 1978). The Greedy sequence employs Givens rotations that do not operate on adjacent planes (Cosnard & Robert 1983, Modi & Clarke 1984). It was found to be optimal with respect to the number of applied CDGRs. However, it is difficult to compute and requires approximately $n/2$ processors, where n is the number of rows. Furthermore, rotations that do not operate on adjacent planes tend to destroy the structure of sparse matrices and make it difficult to develop efficient factorization strategies for structured matrix problems (Kontoghiorghes 2000*b*).

To analyze the complexity of a parallel Givens sequence, an exclusive read, exclusive write (EREW) parallel random access machine (PRAM) computational model can be considered (Fortune & Wyllie 1978).

It is assumed that there are p processors, which can perform p Givens rotations simultaneously. A single time unit is defined as the time required to execute the operation of applying a Givens rotation to two single-element vectors. Thus, the elapsed time necessary to perform a rotation depends on the length of the vectors involved. Computing c and s requires 6 flops. Rotating two elements requires another 6 flops. Hence, annihilating an element and performing the necessary updating of an $m \times n$ matrix requires $6n$ flops. Notice that the Givens rotation is not applied to the first pair of elements, the components of which are set to t and zero, respectively.

1.8 Research overview

The design of tools to tackle computationally intensive problems in model selection and robust regression is considered. Theoretical Givens sequences are designed to compute the QR decomposition efficiently. Algorithms that employ the QR decomposition to perform subset model selection for many variables and least trimmed squares regression are designed. All chapters are self-contained. The first chapter provides a general introduction.

Chapter 2 considers parallel Givens sequences for computing the QR decomposition of an $m \times n$ $(m > n)$ matrix (Hofmann & Kontoghiorghes 2006). The Givens rotations operate on adjacent planes. A pipeline strategy to update the element pairs in the affected matrix rows is employed. This allows a Givens rotation to work on rows that have been partially updated by previous rotations. Two new Givens schemes are developed based on the pipeline approach. They require $n^2/2$ and n processors, respectively. Within this context, a performance analysis in an exclusive read, exclusive write (EREW) parallel random access machine (PRAM) computational model establishes that the proposed schemes are twice as efficient as existing Givens sequences.

In Chapter 3, several new strategies to compute the best subset regression models are proposed (Hofmann, Gatu & Kontoghiorghes 2007). Some of the algorithms are modified versions of existing regression tree methods, while others are new. The first algorithm selects the best models within a given subset size range. It uses a reduced search space and is found to outperform the existing branch and bound algorithm. The properties and computational aspects of the proposed algorithm are discussed in detail. The second new algorithm preorders the variables inside the regression tree. A measure of the distance between a node and the root node is defined. The algorithm applies preordering in all nodes that are within a certain radius from the root node. An efficient method to preorder the variables is employed. Experimental results indicate that the algorithm performs best when the preordering radius is between one quarter and one third of the number of variables. The algorithm has been applied to large-scale subset selection problems that are considered computationally infeasible by conventional exhaustive selection methods. New heuristic strategies are proposed. The most important is one that assigns a different tolerance value to each subset size. It generalizes all exhaustive and heuristic subset selection strategies proposed so far. In addition, it can be used to investigate submodels having non-contiguous sizes, providing a flexible tool for tackling large-scale models.

Chapters 4 and 5 consider new algorithms to compute the robust LTS estimator. Chapter 4 presents a new algorithm to solve LTS regression of the ordinary linear model (Hofmann, Gatu & Kontoghiorghes 2009). The adding row algorithm (ARA) extends existing methods that compute the LTS estimator for a given coverage. It employs a tree based strategy to compute a set of LTS regressors for a range of coverage values.

Thus, prior knowledge of the optimal coverage is not required. New nodes in the regression tree are generated by updating the QR decomposition of the data matrix after adding one observation to the regression model. The ARA is enhanced by employing a branch and bound strategy. The branch and bound algorithm is an exhaustive algorithm that uses a cutting test to prune non-optimal subtrees. It significantly improves over the ARA in computational performance. Observation preordering throughout the traversal of the regression tree is investigated. A computationally efficient and numerically stable calculation of the bounds using Givens rotations is designed around the QR decomposition; the need to explicitly update the triangular factor is avoided. This reduces the overall computational load of the preordering device by approximately half. A solution is proposed to allow preordering when the model is underdetermined. It employs pseudo-orthogonal rotations to downdate the QR decomposition. The strategies are illustrated in practical examples. Experimental results confirm the computational efficiency of the proposed algorithms.

In Chapter 5, the ARA is adapted to the case of the GLM and SUR model with possible singular dispersion matrix (Hofmann & Kontoghiorghes 2009). It searches through a regression tree to find optimal estimates. In each node, an observation is added to a generalized linear least squares problem. The generalized residual sum of squares of the new estimate is obtained by updating a generalized QR decomposition by a single row. Efficient matrix techniques that exploit the sparse structure of the models are exploited. Theoretical measures of computational complexity are provided. Experimental results confirm the ability of the algorithms to identify outlying observations. At the same time, they illustrate the computational intensity of deriving the LTS estimators.

Finally, the last chapter concludes and proposes directions for future research work.

Chapter 2

Pipeline Givens sequences for computing the QR decomposition on an EREW PRAM

Abstract. Parallel Givens sequences for computing the QR decomposition of an $m \times n$ ($m > n$) matrix are considered. The Givens rotations operate on adjacent planes. A pipeline strategy to update the elements in the affected rows is employed. This allows a Givens rotation to work on rows that have been partially updated by previous rotations. Two new Givens schemes are developed based on the pipeline approach. They require $n^2/2$ and n processors, respectively. Within this context, a performance analysis establishes that the proposed schemes are twice as efficient as existing Givens sequences. The employed computational model is an exclusive read/exclusive write (EREW) parallel random access machine (PRAM).

Keywords. Givens rotation, QR decomposition, parallel algorithms, PRAM.

2.1 Introduction

Consider the QR decomposition of the full column rank matrix $A \in \mathbb{R}^{m\times n}$:

$$Q^{\mathrm{T}}A = \begin{bmatrix} R \\ 0 \end{bmatrix} \begin{matrix} n \\ m-n \end{matrix}, \tag{2.1}$$

where $Q \in \mathbb{R}^{m\times m}$ and R is upper triangular of order n. The triangular matrix R in (2.1) is derived iteratively from $Q_i^{\mathrm{T}}\widetilde{A}_i = \widetilde{A}_{i+1}$, where $\widetilde{A}_0 = A$ and Q_i is orthogonal. The triangularization process terminates when $\widetilde{A}_\nu$ ($\nu > 0$) is upper triangular. $Q = Q_0Q_1\cdots Q_\nu$ is not computed explicitly. R can be derived by employing a sequence of Givens rotations. The Givens rotation (GR) that annihilates the element $A_{i,j}$ when applied from the left of A has the form

$$G_{i,j} = \mathrm{diag}(I_{i-2}, \widetilde{G}_{i,j}, I_{m-i}), \quad \text{with} \quad \widetilde{G}_{i,j} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix},$$

where $c = A_{i-1,j}/t$, $s = A_{i,j}/t$ and $t^2 = A_{i-1,j}^2 + A_{i,j}^2$ ($t \neq 0$).

A GR only affects the ith and $(i-1)$th rows of the matrix A. Thus, $\lfloor m/2 \rfloor$ rotations can be applied simultaneously. A compound disjoint givens rotation (CDGR) comprises rotations that can be applied simultaneously. Parallel algorithms for computing the QR decomposition based on CDGRs have been developed (Cosnard et al. 1986, Cosnard & Daoudi 1994, Kontoghiorghes 2002, Luk 1986, Modi 1988, Sameh & Brent 1971, Sameh & Kuck 1978, Sameh 1985). The Greedy sequence (Cosnard & Robert 1983, Modi & Clarke 1984) was found to be optimal; that is, it requires less CDGRs than any other Givens strategy. However, the computation of the indices of the rotations, which do not involve adjacent planes, is non-trivial. The employment of rotations between adjacent planes facilitates the development of efficient factorization strategies for structured matrices (Kontoghiorghes 2000*b*).

An exclusive read/exclusive write (EREW) parallel random access machine (PRAM) computational model is considered (Kontoghiorghes 2000*b*). It is assumed that there are p processors, which can perform p GRs simultaneously. A single time unit is defined as the time required to apply a Givens rotation to two single-element vectors. Thus, the time necessary to perform a rotation depends on the length of the vectors involved. Computing c and s requires 6 flops. Rotating two elements requires another 6 flops. Hence, annihilating an element and performing the necessary updating of an $m \times n$ matrix requires $6n$ flops. Notice that the GR is not applied to the first pair of elements, the components of which are set to t and zero, respectively. To simplify the complexity analysis of the proposed algorithms, it is assumed that m and n are even. Complexities are given in time units.

New parallel Givens sequences to compute the QR decomposition are proposed. Throughout the execution of a Givens sequence, the annihilated elements are preserved. In the next section, a pipelined version of the parallel Sameh and Kuck scheme is presented. Two new pipeline-parallel strategies are discussed in Section 2.3. A theoretical analysis of the complexities is presented. Section 2.4 summarizes what has been achieved.

2.2 Pipeline-parallel SK sequence

The parallel Sameh and Kuck (SK) scheme computes (2.1) by applying up to n GRs simultaneously (Sameh & Kuck 1978). Each GR is performed by one processor. Specifically, the $(2i-1)$th CDGR starts to annihilate the elements in the ith column of A, as illustrated in Figure 2.1a.

The numeral i and the symbol $\bullet$ denote an element annihilated by the ith CDGR and a non-zero element, respectively. The number of CDGRs and time units required by the SK scheme are given by, respectively,

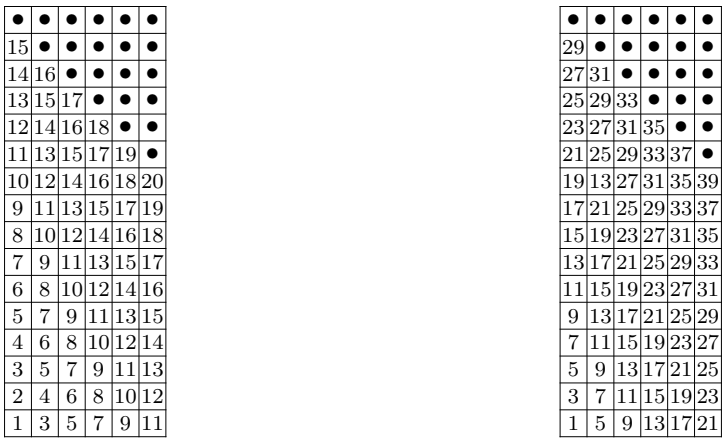
$$C_{\mathrm{SK}}(m,n) = m + n - 2$$

and

$$T_{\mathrm{SK}}(m,n) = (m-1)n + \sum_{j=2}^{n}(n-j+1) = n(2m+n-3)/2.$$

Here it is assumed that $p = n$.

The rotation $G_{i,j}$ operates on two $(n-j+1)$-element subvectors of rows i and $i-1$, respectively. Let $g_{i,j}^{(k)}$ denote the application of $G_{i,j}$ to the kth pair of elements ($k \in \{1,\ldots,n-j+1\}$), which are positioned at $(i, j+k-1)$ and $(i-1, j+k-1)$. The rotation can now be expressed as the sequence of elementary rotations $g_{i,j}^{(1)},\ldots,g_{i,j}^{(n-j+1)}$, which are applied to, respectively, the pairs of elements $\big\{(i,j),(i-1,j)\big\},\ldots,\big\{(i,n),(i-$



(a) SK (b) mSK

Figure 2.1: The SK and modified SK Givens sequences to compute the QR decomposition, where $m = 16$ and $n = 6$.

$1, n)\big\}$. The rotation $G_{i+1,j+1}$ may be initiated before the application of $G_{i,j}$ has been completed. Specifically, $g^{(1)}_{i+1,j+1}$ can be applied once $g^{(2)}_{i,j}$ has been executed. Thus, several GRs may operate concurrently on different elements of a row. The pipelined SK (PipSK) scheme employs this strategy. The first steps of the PipSK scheme are illustrated in Figure 2.2. Like the SK scheme, the PipSK scheme initiates a CDGR every second time unit. Its overall execution time is given by

$$T_{\text{PipSK}}(m, n) = 2C_{\text{SK}}(m, n) - 1 = 2m + 2n - 5.$$

The number of CDGRs performed in parallel is $n/2$, and each CDGR requires $2i$ processors ($i = 1, \ldots, n/2$). The PipSK scheme thus requires $p = \sum_{i=1}^{n/2} 2i \approx n^2/4$ processors. Its annihilation pattern — a modified SK scheme — is shown in Figure 2.1b.

2.3 New pipeline-parallel Givens sequences

Alternative parallel Givens sequences (PGS) that are more suitable for pipelining are shown in Figure 2.3. The number of CDGRs applied by the first PGS (PGS1) (Bojanczyk, Brent & Kung 1984) is given by

$$C_{\text{PGS1}}(m, n) = m + 2n - 3.$$

The pipelined PGS1 (PipPGS1) is illustrated in Figure 2.4. It requires $p \approx n^2/2$ processors. It initiates a CDGR at every time unit, and its time complexity is given by the total number of CDGRs applied. That is,

$$T_{\text{PipPGS1}}(m, n) = C_{\text{PGS1}}(m, n) = m + 2n - 3.$$

The PipPGS1 operates in cycles of $n + 1$ time units. This is shown in Figure 2.5. In each time unit, up to n processors initiate a GR, while the other processors update previously initiated rotations. In one cycle, each processor executes two GRs with complexities T_1 and T_2, respectively, such that $T_1 + T_2 = n + 1$ time units. The PipPGS1 performs better than the SK scheme, utilizing approximately $n/2$ times the number of processors. That is, $T_{\text{SK}}(m, n)/T_{\text{PipPGS1}}(m, n) \approx n$. Hence, the efficiency of the PipPGS1 is twice that of the SK scheme.

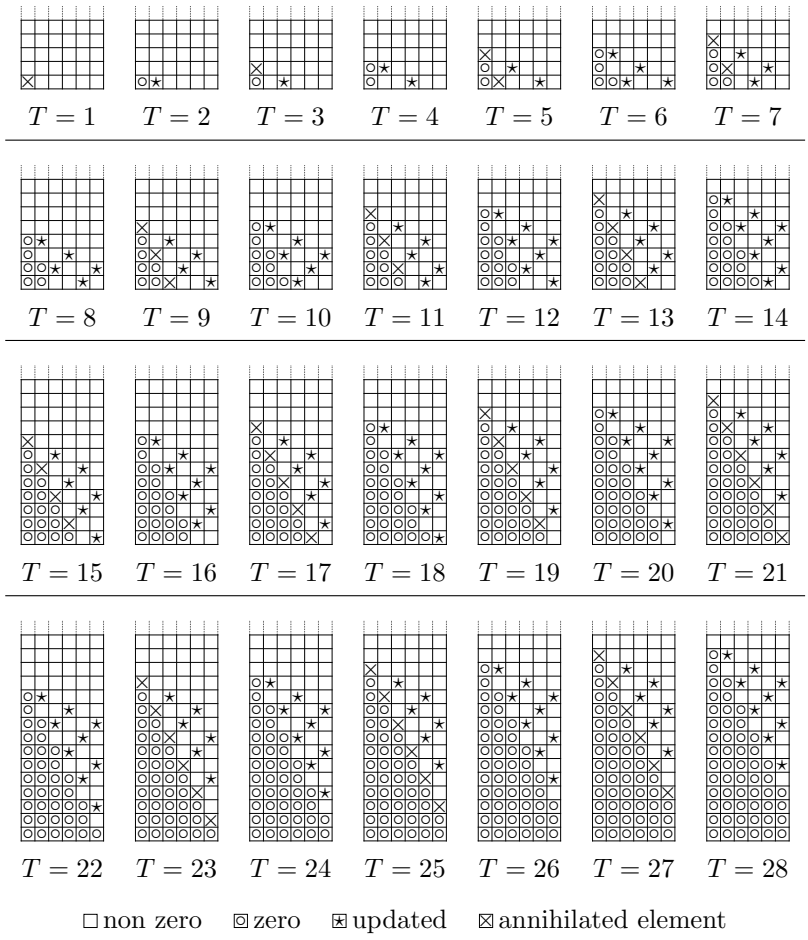


Figure 2.2: Partial annihilation of a matrix by the PipSK scheme when $n = 6$.

•	•	•	•	•	•
15	•	•	•	•	•
14	17	•	•	•	•
13	16	19	•	•	•
12	15	18	21	•	•
11	14	17	20	23	•
10	13	16	19	22	25
9	12	15	18	21	24
8	11	14	17	20	23
7	10	13	16	19	22
6	9	12	15	18	21
5	8	11	14	17	20
4	7	10	13	16	19
3	6	9	12	15	18
2	5	8	11	14	17
1	4	7	10	13	16

(a) PGS1

•	•	•	•	•	•
35	•	•	•	•	•
29	36	•	•	•	•
28	35	41	•	•	•
22	30	37	42	•	•
21	28	35	41	45	•
16	23	31	38	43	46
15	21	28	35	41	45
11	17	24	32	39	44
10	15	21	28	35	41
7	12	18	25	33	40
6	10	15	21	28	35
4	8	13	19	26	34
3	6	10	15	21	28
2	5	9	14	20	27
1	3	6	10	15	21

(b) PGS2

cycle	steps	set
1	1,2	1
2	3,4,5	
3	6,7,8,9	
4	10,11,12,13,14	
5	15,16,17,18,19,20	
6	21,22,23,24,25,26,27	2
7	28,29,30,31,32,33,34	
8	35,36,37,38,39,40	3
9	41,42,43,44	
10	45,46	

(c) PGS2 cycles

Figure 2.3: Parallel Givens sequences for computing the QR decomposition, where $m = 16$ and $n = 6$.

The second PGS (PGS2), illustrated in Figure 2.3b, employs $p = n$ processors. A cycle involves one CDGR and n consecutive GRs. It annihilates up to $2n$ elements in $n+1$ steps. This is illustrated in Figure 2.6. The PGS2 scheme requires more steps than the SK scheme. When m and n are even, the sequence consists of $(m+n-2)/2$ cycles. It can be partitioned in three sets that contain, respectively, the cycles $\{1,\ldots,n-1\}$, $\{n,\ldots,(m-2)/2\}$ and $\{m/2,\ldots,(m+n-2)/2\}$. The sets, cycles and constituent steps are detailed in Figure 2.3c. The ith cycle in the first, second and third set applies $i+1$, $n+1$ and $2(n/2-i+1)$ steps,

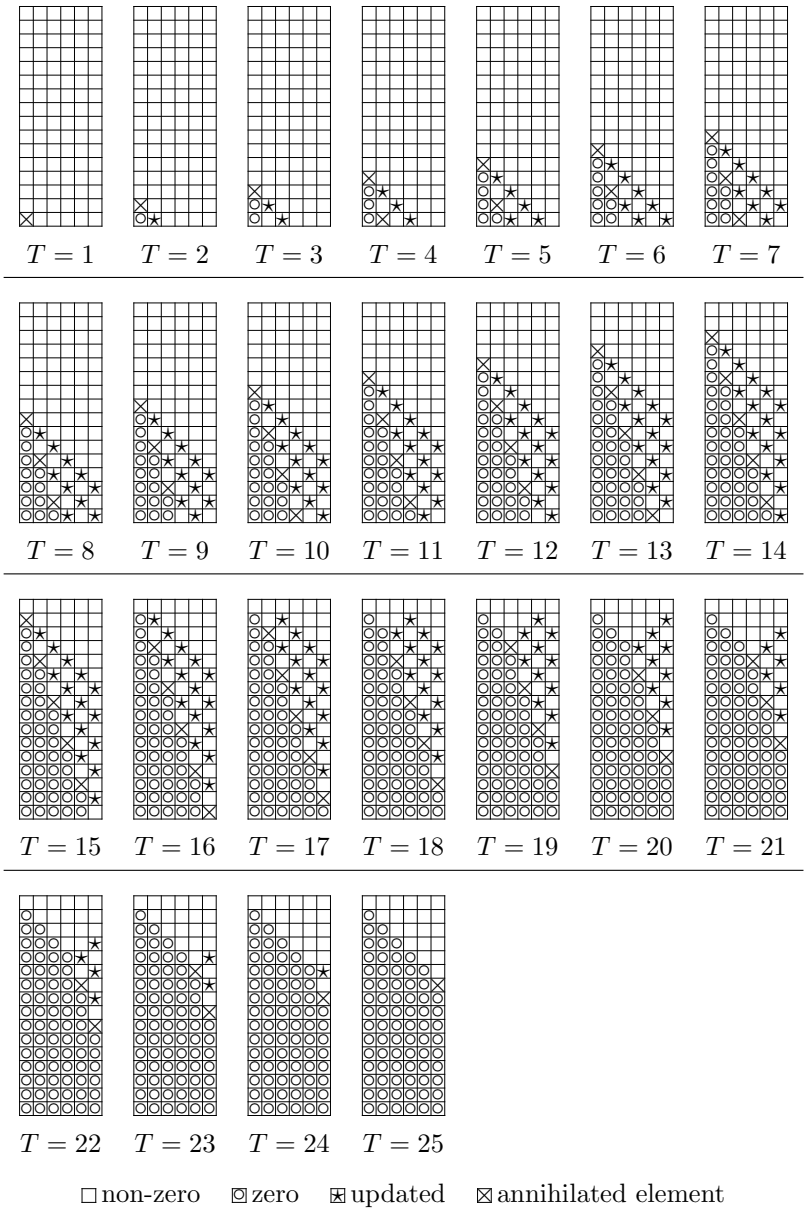


Figure 2.4: Triangularization of a 16×6 matrix by the PipPGS1 algorithm.

respectively. The total number of steps applied by the PGS2 is given by

$$C_{\text{PGS2}}(m,n) = \sum_{i=1}^{n-1}(i+1) + (m-2n)(n+1)/2 + \sum_{i=1}^{n/2} 2i$$
$$\approx (2mn + 2m - n^2)/4.$$

The pipelined PGS2 (PipPGS2) applies $2n$ GRs in a pipeline manner. Figure 2.7 illustrates the annihilation cycle of the PipPGS2, which annihilates $2n$ elements in $n+1$ time units. This is equivalent to two CDGRs of the SK scheme, for which $2n$ time units are required. Figures 2.8 and 2.9 illustrate, respectively, the initial and the final phase of the annihilation process of a 16×6 matrix by the PipPGS2 scheme. An asterisk denotes the start of a new cycle.

A processor initiates a GR at every time unit. When m and n are even, the first $(m-2)/2$ cycles are executed in $n+1$ time units each. The ith cycle in the third set requires $2(n/2 - i + 1) + 1$ time units

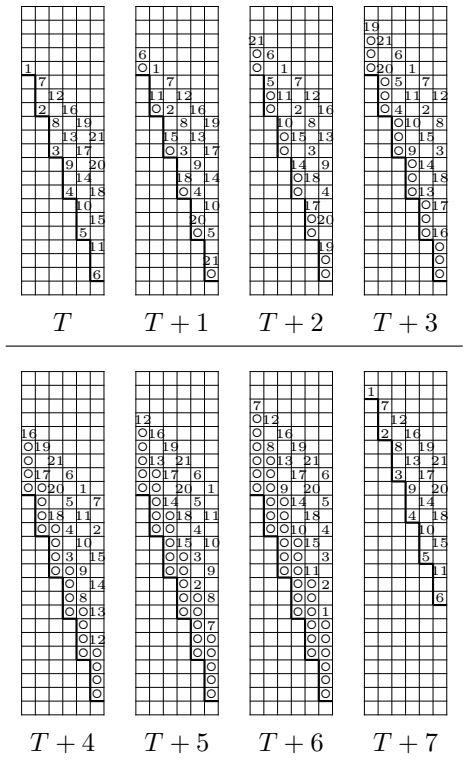


Figure 2.5: Annihilation cycle of the PipPGS1 algorithm when $n = 6$.

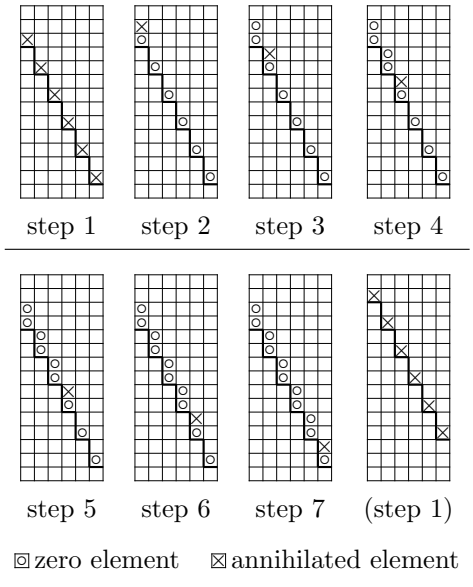


Figure 2.6: Annihilation cycle of the PGS2 scheme when $n = 6$.

$(i = 1, \ldots, n/2)$. The overall execution time of the PipPGS2 algorithm is given by

$$T_{\text{PipPGS2}}(m,n) = (m-2)(n+1)/2 + \sum_{i=1}^{n/2}(2i+1)$$
$$\approx (2mn + 2m + n^2)/4.$$

Furthermore, if $m \gg n$ and for the same number of processors,

$$\frac{T_{\text{SK}}(m,n)}{T_{\text{PipPGS2}}(m,n)} \approx 2.$$

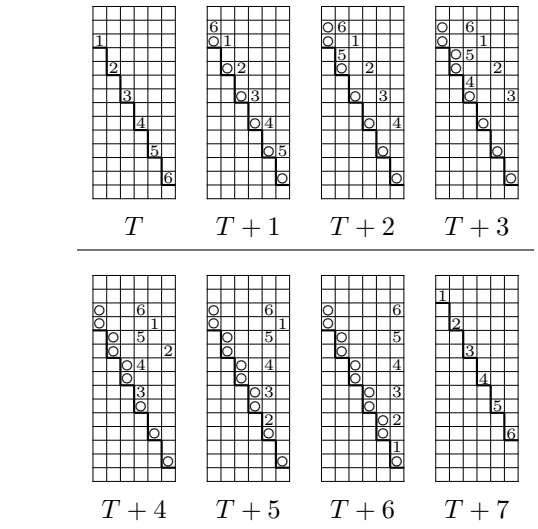


Figure 2.7: Annihilation cycle of the PipPGS2 algorithm when $n = 6$.

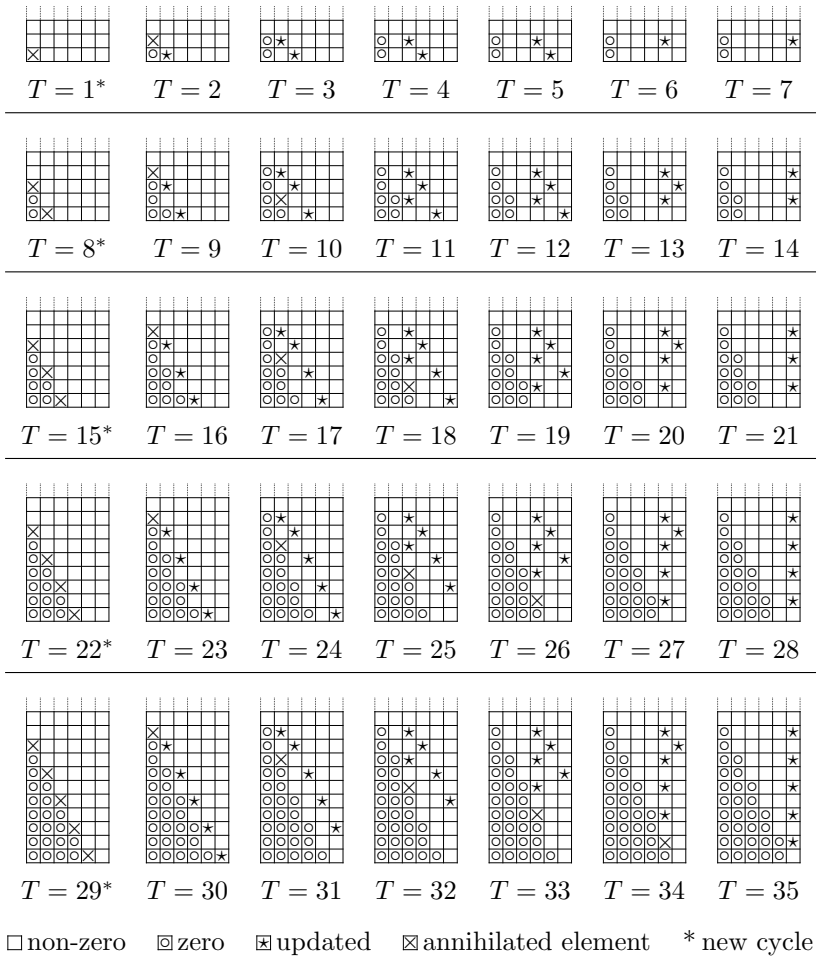


Figure 2.8: Initial annihilation steps of a 16×6 matrix by the PipPGS2 algorithm.

That is, the proposed scheme is twice as fast as the SK scheme.

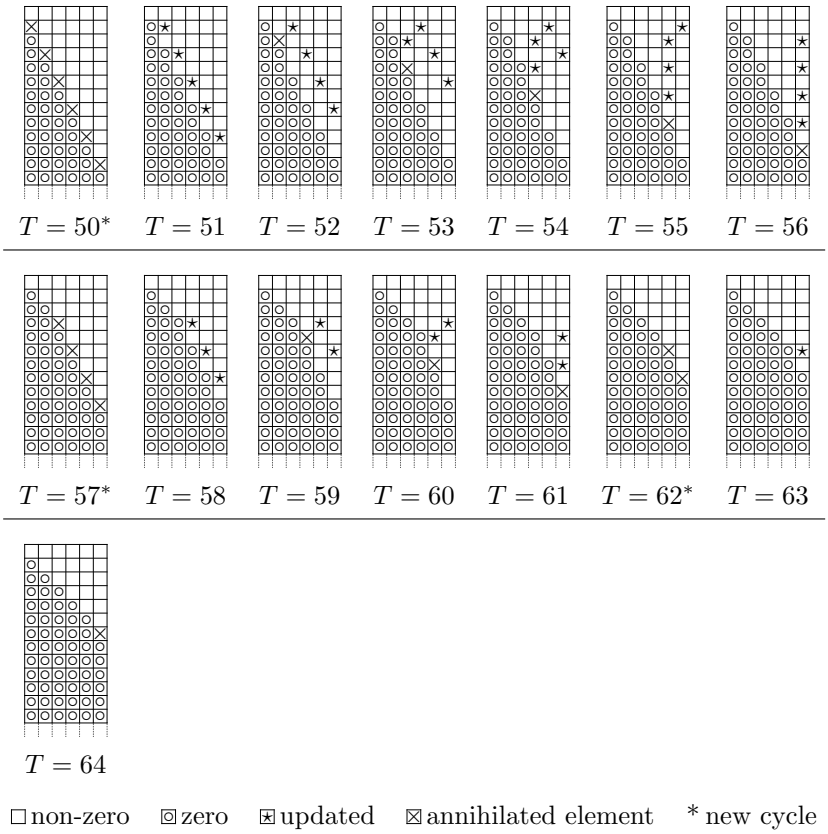


Figure 2.9: Final annihilation steps of a 16×6 matrix by the PipPGS2 algorithm.

2.4 Conclusions

A new pipeline-parallel strategy to compute the QR decomposition is proposed. Its computational complexity is compared to the SK scheme (Sameh & Kuck 1978). The complexity analysis is not based on the unrealistic assumption that all CDGRs, or cycles, have the same execution time. Instead, the number of operations performed by a single Givens rotation is given by the size of the pair of vectors involved in the rotation. It is found that for an equal number of processors, the pipeline-parallel scheme solves the $m \times n$ QR decomposition twice as fast as the SK scheme when $m \gg n$. The complexities are summarized in Table 2.1.

Table 2.1: Summary of the complexities of the SK and pipelined schemes.

Scheme	Processors	Complexity
SK	n	$n(2m + n - 3)/2$
PipSK	$n^2/4$	$2m + 2n - 5$
PipPGS1	$n^2/2$	$m + 2n - 3$
PipPGS2	n	$(2mn + 2m + n^2)/4$

Block versions of the SK scheme to compute the orthogonal factorizations of structured matrices that arise e.g. in econometric estimation problems have previously been designed (Kontoghiorghes 2000*a*, Yanev, Foschi & Kontoghiorghes 2004). Within this context, the Givens rotations are replaced by orthogonal factorizations that employ Householder reflections. Thus, it might be fruitful to investigate the effectiveness of incorporating the pipeline strategy in the design of block algorithms (Kontoghiorghes 2000*c*, Yanev & Kontoghiorghes 2004).

Chapter 3

Efficient algorithms for computing the best subset regression models for large-scale problems

Abstract. Several strategies for computing the best subset regression models are proposed. Some of the algorithms are modified versions of existing regression tree methods, while others are new. The first algorithm selects the best subset models within a given size range. It uses a reduced search space and is found to outperform the existing branch and bound algorithm. The properties and computational aspects of the proposed algorithm are discussed in detail. The second new algorithm preorders the variables inside the regression tree. A measure of the distance (radius) between a node and the root node is defined. The algorithm applies preordering to all nodes which are within a certain radius from the root node. An efficient method to preorder the variables is employed. The experimental results indicate that the algorithm performs best when the preordering radius is between one quarter and one third of the number of variables. The algorithm has been applied to large-scale subset selection problems, which are considered computationally infeasible by conventional exhaustive selection methods. New heuristic strategies are proposed. The most important is one that assigns a different tolerance value to each subset size. It generalizes all exhaustive and heuristic subset selection strategies proposed so far. In addition, it can be used to investigate submodels having non-contiguous sizes. It provides a flexible tool for tackling large-scale models.

Keywords. Best subset regression, regression tree, branch and bound algorithm.

3.1 Introduction

The problem of computing the best subset regression models arises in statistical model selection. Most of the criteria used to evaluate the subset models rely upon the residual sum of squares (RSS) (Searle 1971, Sen & Srivastava 1990). Consider the standard regression model

$$y = A\beta + \varepsilon, \quad \varepsilon \sim (0, \sigma^2 I_m), \tag{3.1}$$

where $y \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ is the exogenous data matrix of full column rank, $\beta \in \mathbb{R}^n$ is the coefficient vector, and $\varepsilon \in \mathbb{R}^n$ is the noise vector. The columns of A correspond to the exogenous variables $V = (v_1, \ldots, v_n)$. A submodel S of (3.1) comprises some of the variables in V. There are $2^n - 1$ possible subset models, and their computation is only feasible for small values of n. The dropping column algorithm (DCA) derives all submodels by generating a regression tree (Clarke 1981, Gatu & Kontoghiorghes 2003, Smith & Bremner 1989). The parallelization of the DCA moderately improves its practical value (Gatu & Kontoghiorghes 2003). Various procedures such as the forward, backward and stepwise selection try to identify a subset by inspecting very few combinations of variables. However, these methods rarely succeed in finding the best submodel (Hocking 1976, Seber 1977). Other approaches for subset selection include ridge regression, the Non-negative Garrote and the Lasso (Breiman 1995, Fan & Li 2001, Tibshirani 1996). Sequential replacement algorithms are fairly fast and can be used to give some indication of the maximum size of the subsets that are likely to be of interest (Hastie et al. 2001). Branch and bound algorithms for choosing a subset of k features from a given, larger set of size n have been investigated within the context of feature selection problems (Narendra & Fukunaga 1977, Roberts 1984, Somol et al. 2004). These strategies are used when the size k of the subset to be selected is known. They search over $n!/(k!(n-k)!)$ subsets.

A computationally efficient branch and bound algorithm (BBA) has been devised (Gatu & Kontoghiorghes 2006, Gatu et al. 2007). The BBA avoids the computation of the whole regression tree, and it derives the best subset model for each number of variables. That is, it computes

$$\underset{S}{\operatorname{argmin}} \operatorname{RSS}(S) \quad \text{subject to} \quad |S| = k \quad \text{for all} \quad k = 1, \ldots, n. \tag{3.2}$$

The BBA was built around the fundamental property

$$\operatorname{RSS}(S_1) \geq \operatorname{RSS}(S_2) \quad \text{if} \quad S_1 \subseteq S_2, \tag{3.3}$$

where S_1 and S_2 are two variable subsets of V (Gatu & Kontoghiorghes 2006). The BBA-1, which is an extension of the BBA, preorders the n variables in the root node according to their strength. The variables i and j are arranged such that $\operatorname{RSS}(V - \{v_i\}) \geq \operatorname{RSS}(V - \{v_j\})$ for each $i \leq j$, where $V - \{v_i\}$ is the set V from which the ith variable has been deleted. The BBA-1 has been shown to outperform the previously introduced leaps and bounds algorithm (LBA) (Furnival & Wilson 1974). Table 3.1 shows the execution times of the BBA and the LBA for datasets with 36 to 48 variables. Note that the BBA outperforms the leaps and bounds with preordering in the root node (LBA-1). A heuristic version of the BBA (HBBA), which uses a tolerance parameter to relax the BBA pruning test, has been discussed. The HBBA might not provide the optimal solution, but the relative residual error (RRE) of the computed solution is smaller than the employed tolerance.

Often, models within a given size range must be investigated. These models, hereafter called subrange subset models, do not require the generation of the whole tree. Thus, the adaptation of the BBA to derive the subrange subset models is expected to have a lower computational cost, and thus, larger-scale models can be tackled. The structural properties of a regression tree strategy which generates the subrange subset models is investigated, and its theoretical complexity derived. A new, non-trivial preordering strategy that outperforms the BBA-1 is designed and analyzed. The new strategy, which can be found to be significantly faster than existing ones, can derive the best subset models from a larger pool of variables. In addition, new heuristic strategies based on the HBBA are developed. The tolerance parameter is either a function of the

Table 3.1: LBA & BBA: Execution time (in seconds) for datasets of different sizes, with and without variable preordering.

# Variables	36	37	38	39	40	41	42	43	44	45	46	47	48
LBA	8	29	44	30	203	57	108	319	135	316	685	2697	6023
BBA	2	5	12	8	35	14	9	55	27	37	97	380	1722
LBA-1	3	16	28	9	82	33	22	203	79	86	306	1326	1910
BBA-1	1	4	13	2	20	11	4	47	18	15	51	216	529

level in the regression tree or of the size of the subset model. The novel strategies decrease execution time while selecting models of similar, or even better, quality.

The proposed strategies, which outperform the existing subset selection BBA-1 and its heuristic version, are aimed at tackling large-scale models. The next section briefly discusses the DCA, and it introduces the all-subset-models regression tree. It generalizes the DCA so as to select only submodels that lie within a given size range. Section 3.3 discusses a novel strategy that preorders the variables of the nodes in various levels of the tree. The significant improvement in computational efficiency over the BBA-1 is illustrated. Section 3.4 presents and compares various new heuristic strategies. Theoretical and experimental results are presented. Conclusions and proposals for future work are discussed in Section 3.5.

The algorithms were implemented in C++ and are available in a package for the R statistical software environment (R Development Core Team 2005). The GNU compiler collection was used to generate the shared libraries. The tests were run on a Pentium-class machine with 512 Mb of RAM in a Linux environment. Real and artificial data have been used in the experiments. A set of artificial variables has been randomly generated. The response variable of the true model is based on a linear combination of a subset of these artificial variables and the addition of some noise. An intercept term is included in the true model.

3.2 Subrange model selection

The DCA employs a straightforward approach to solve the best-subset problem in (3.2). It enumerates and evaluates all possible $2^n - 1$ subsets of V. It generates a regression tree consisting of 2^{n-1} nodes (Gatu & Kontoghiorghes 2003, Smith & Bremner 1989). Each node in the tree corresponds to a subset $S = (s_1, \ldots, s_{|S|})$ of variables and an index k ($k \in \{0, \ldots, |S|-1\}$). The $|S|-k$ subleading models $(s_1, \ldots, s_{k+1})$, $\ldots$, $(s_1, \ldots, s_{|S|})$ are evaluated. A new node is generated by deleting a variable. The descending nodes are given by

$$\big(\mathrm{drop}(S, k+1), k\big), \big(\mathrm{drop}(S, k+2), k+1\big), \ldots, \big(\mathrm{drop}(S, |S|-1), |S|-2\big).$$

Here, the operation $\mathrm{drop}(S, i)$ computes a new subset, which corresponds to the subset S from which the ith variable has been deleted. This is equivalent to downdating the QR decomposition after the ith column has been deleted (Golub & Van Loan 1996, Kontoghiorghes 2000*a*, Smith & Bremner 1989). The DCA employs Givens rotations to move efficiently from one node to another.

The search space of all possible variable subset models can be reduced by imposing bounds on the size of the subset models. The subrange model selection problem is to derive

$$S_j^* = \underset{S}{\operatorname{argmin}} \operatorname{RSS}(S) \quad \text{subject to} \quad |S| = j \quad \text{for all} \quad j = n_{\mathrm{a}}, \ldots, n_{\mathrm{b}}, \tag{3.4}$$

where n_{a} and n_{b} are the subrange bounds ($1 \leq n_{\mathrm{a}} \leq n_{\mathrm{b}} \leq n$). The DCA and Subrange DCA are equivalent when $n_{\mathrm{a}} = 1$ and $n_{\mathrm{b}} = n$. The Subrange DCA generates a subtree of the original regression tree. The nodes (S, k) are not computed when $|S| < n_{\mathrm{a}}$ or $k \geq n_{\mathrm{b}}$. This is illustrated in Figure 3.1. The DCA regression tree with $n = 5$ variables is shown. The blank nodes represent the Subrange DCA subtree for $n_{\mathrm{a}} = n_{\mathrm{b}} = 3$. Portions of the tree that are not computed by the Subrange DCA are shaded. The nodes in the last two levels of the tree produces subsets with one or two variables, i.e. the subsets (4), (5), (4, 5), (3, 5), (2, 5) and (1, 5). These nodes are discarded by the Subrange DCA (case $|S| < n_{\mathrm{a}}$). The rightmost node in the tree produces the subset model $(1, 2, 3, 5)$ of size 4. The Subrange DCA discards this node (case $k \geq n_{\mathrm{b}}$). The Appendix provides a detailed and formal analysis of the Subrange DCA.

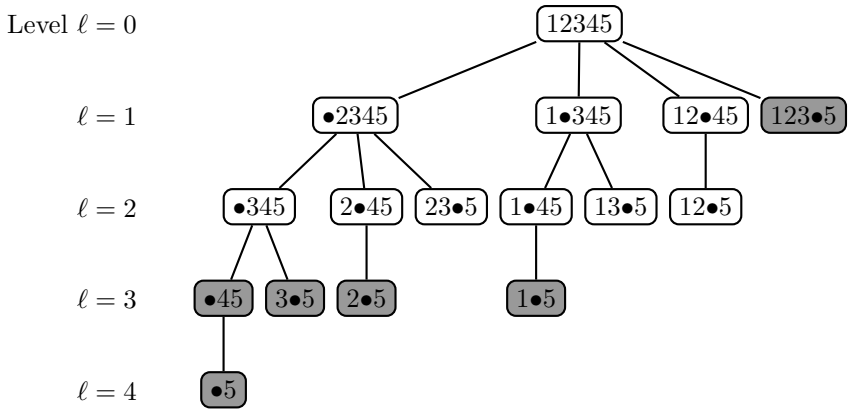


Figure 3.1: The Subrange DCA subtree, where $n = 5$ and $n_{\mathrm{a}} = n_{\mathrm{b}} = 3$.

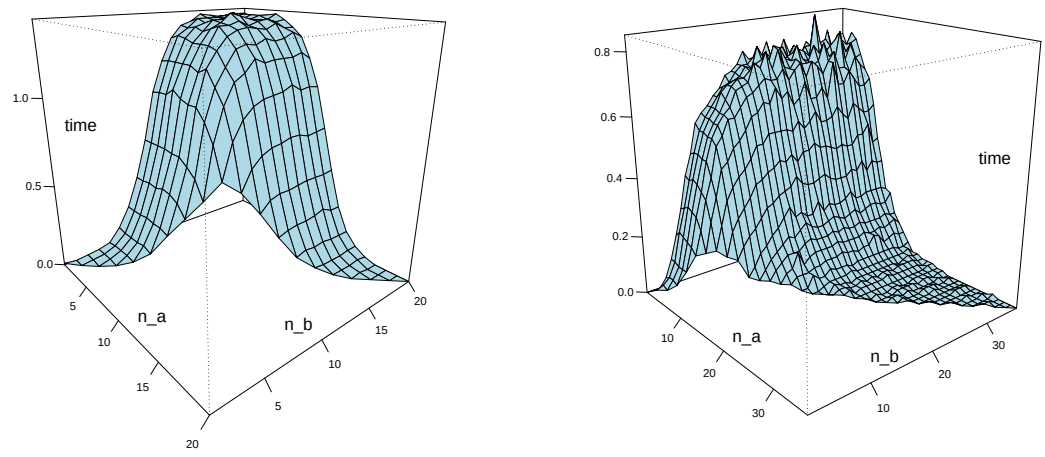
The branch and bound method can be applied in the subtree generated by the Subrange DCA. This strategy is called Subrange BBA and is summarized in Algorithm 1. The Subrange BBA stores the generated nodes in a list. The list is managed according to a last in, first out (LIFO) policy. The RSS of the best subset models are recorded in a table r. The entry r_i holds the RSS of the current best submodel of size i. The initial residuals may be given beforehand based on some earlier results; otherwise, they are set to positive infinity. The entries are sorted in decreasing order. Each iteration removes a node (S, k) from the list. The subleading model $(s_1, \ldots, s_i)$ is evaluated and compared to r_i for all $i = k + 1, \ldots, |S|$. The entry r_i is updated if $\mathrm{RSS}(s_1, \ldots, s_i) < r_i$. If $|S| \leq n_{\mathrm{a}}$, then no child nodes are generated, and the iteration terminates; otherwise, the cutting test $\mathrm{RSS}(S) > r_i$ is computed for all $i = k + 1, \ldots, \min(n_{\mathrm{b}}, |S| - 1)$. If the test fails, the child node $\big(\mathrm{drop}(S, i), i - 1\big)$ is generated and inserted in the node list. Note that if $i < n_{\mathrm{a}}$, then the value $r_{n_{\mathrm{a}}}$ is used in the cutting test. This is illustrated on Line 8 of Algorithm 1. The modified cutting test is more efficient than that of the BBA because $r_{n_{\mathrm{a}}} \leq r_i$ ($i \in \{1, \ldots, n_{\mathrm{a}} - 1\}$). The algorithm terminates when the node list is empty. Notice that the Subrange BBA with preordering (Subrange BBA-1) is obtained by sorting the variables in the initial set V. The Subrange BBA outperforms the standard BBA because it uses a reduced search space and a more efficient cutting test.

The effects of the subrange bounds n_{a} and n_{b} on the computational performance of the Subrange DCA and Subrange BBA-1 have been investigated. Figures 3.2a and 3.2b show the execution times of the Subrange DCA and Subrange BBA-1 for $n = 20$ and 36 variables, respectively. It can be observed that the Subrange DCA is computationally effective in two cases: for narrow (i.e. $n_{\mathrm{b}} - n_{\mathrm{a}} < 2$) and for wide (i.e. $n_{\mathrm{a}} = 1$ and $n_{\mathrm{b}} < n/4$, or $n_{\mathrm{a}} > 3n/4$ and $n_{\mathrm{b}} = n$) size ranges. The Subrange BBA-1, on the other hand, is effective for

Algorithm 1 The Subrange BBA.

```
1:  procedure SubrangeBBA(V, n_a, n_b, r)
2:      Insert (V, 0) in node list
3:      while not empty(node list) do
4:          Extract (S, k) from node list
5:          Update residuals r_{k+1},...,r_{|S|}
6:          if |S| > n_a then
7:              for i = k + 1, ..., min(n_b, |S| − 1) do
8:                  j ← max(i, n_a)
9:                  if RSS(S) > r_j break 7          ▷ exit for-loop on Line 7
10:                 S' ← drop(S, i)
11:                 Insert (S', i − 1) in node list
12:             end for
13:         end if
14:     end while
15: end procedure
```

all ranges such that $n_a > n/2$. This is further confirmed by the results in Table 3.2. The number of nodes generated by the Subrange BBA-1 for the 15 variable Pollute dataset (Miller 2002) is shown. All possible subranges are considered ($1 \leq n_a \leq n_b \leq 15$). For the case $n_a = 1$ and $n_b = 15$, the Subrange BBA-1 generates 381 nodes and is equivalent to the BBA-1 (Gatu & Kontoghiorghes 2006).



(a) Subrange DCA ($n = 20$)

(b) Subrange BBA-1 ($n = 36$)

Figure 3.2: Subrange model selection: execution times in seconds for varying n_a and n_b.

Table 3.2: Number of nodes generated by the Subrange BBA-1 to compute the best subset models of the Pollute dataset for different size ranges.

n_a \ n_b	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	12	37	96	178	276	332	356	373	375	376	377	378	380	381	381
2		31	90	172	270	326	350	367	369	370	371	372	374	375	375
3			75	157	255	311	335	352	354	355	356	357	359	360	360
4				123	221	277	301	318	320	321	322	323	325	326	326
5					173	229	253	270	272	273	274	275	277	278	278
6						103	127	144	146	147	148	149	151	152	152
7							52	69	71	72	73	74	76	77	77
8								38	40	41	42	43	45	46	46
9									11	12	13	14	16	17	17
10										11	12	13	15	16	16
11											12	13	15	16	16
12												13	15	16	16
13													15	16	16
14														15	15
15															1

3.3 Radius preordering

Preordering the variables in the root node significantly increases the computational speed of the BBA. The cost of preordering the variables once is negligible. The aim is to consider a strategy that preorders variable subsets inside the regression tree and is more efficient than the BBA-1. The new strategy is hereafter called BBA with preordering (PBBA). The PBBA sorts the variables according to their strength. The strength of the ith variable is given by its bound $\mathrm{RSS}(S - \{s_i\}) = \mathrm{RSS}(\mathrm{drop}(S, i))$. The main tool for deriving the bound is the downdating of the QR decomposition after the corresponding column of the data matrix has been deleted. This has a cost; therefore, care must be taken to apply preordering in nodes where the expected gain outweighs the cost inherent to the preordering process.

The PBBA preorders the variables in the root nodes of large subtrees. The size of the subtree with root (S, k) is given by 2^{d-1}, where $d = |S| - k$ is the number of active variables. The PBBA defines the node radius $\rho = n - d$, where n is the number of initial variables in the root node $(V, 0)$. The radius of a node is a measure of its distance from the root node. Notice that the root nodes of larger subtrees have a smaller radius, while the root nodes of equally sized subtrees have equal radii. Given a parameter P, variable preordering is only applied in nodes where $\rho < P$ $(0 \leq P \leq n)$. If $P = 0$ or $P = 1$, then the PBBA is equivalent to the BBA or BBA-1, respectively. If $P = n$, then the active variables are preordered in all nodes. Figure 3.3 illustrates the radius of every node in the regression tree for $n = 5$ variables. Shaded nodes are preordered by the PBBA with $P = 3$.

The PBBA is illustrated in Algorithm 2. A node is extracted from the node list at each iteration. If the node radius ρ is less than the given preordering radius P, then the active variables are preordered before updating the residuals table. Nodes which do not improve the current best solution are not generated. The cutting test (Line 9) compares the bound of the current node to the corresponding entry in the residuals table in order to decide whether or not to generate the next child node.

The preordering process sorts the variables in order of decreasing bounds. Given a node (S, k), the bound of the ith active variable $(i \in \{1, \ldots, d\})$ is $\mathrm{RSS}(\mathrm{drop}(S, k + i))$, i.e. the RSS of the model from which the

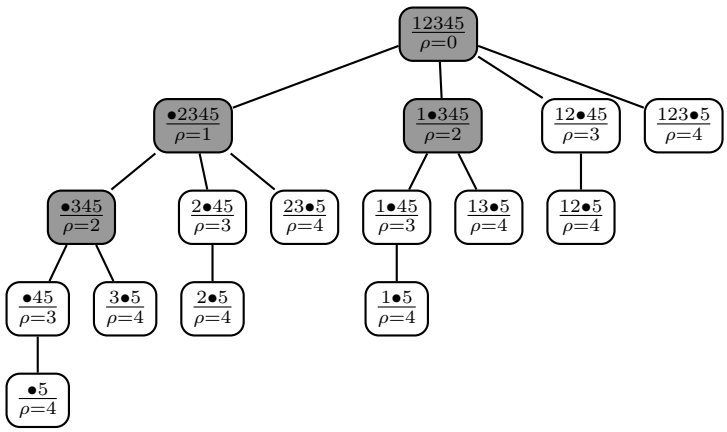


Figure 3.3: Nodes that apply preordering for radius $P = 3$.

Algorithm 2 The BBA with preordering (PBBA).

```
1:  procedure PBBA(V, P, r)
2:      Insert (V, 0) in node list
3:      while not empty(node list) do
4:          Extract (S, k) from node list
5:          ρ ← |V| − |S| + k
6:          if ρ < P then Preorder (s_{k+1}, ..., s_{|S|})
7:          Update residuals r_{k+1}, ..., r_{|S|}
8:          for i = k + 1, ..., |S| − 1 do
9:              if RSS(S) > r_i break 8                ▷ exit for-loop on Line 8
10:             S' ← drop(S, i)
11:             Insert (S', i − 1) in node list
12:         end for
13:     end while
14: end procedure
```

$(k+i)$th variable has been removed. The d active variables of the node (S, k) are represented by the leading $d \times d$ submatrix of the upper triangular $R \in \mathbb{R}^{(d+1)\times(d+1)}$ factor of the QR decomposition. The last column of R corresponds to the response variable y in (3.1). Let $\widetilde{R}$ denote R without its ith column. The drop operation applies $d - i + 1$ biplanar Givens rotations to retriangularize $\widetilde{R}$. That is, it computes:

$$G_d \cdots G_i \widetilde{R} = \begin{bmatrix} \widehat{R} \\ 0 \end{bmatrix}, \tag{3.5}$$

where $\widehat{R} \in \mathbb{R}^{d\times d}$ is upper triangular. The bound of the ith active variable, that is the RSS of the model after deleting the ith variable, is given by $\widehat{R}^2_{d,d}$ — the square of the diagonal element of $\widehat{R}$ in position (d, d).

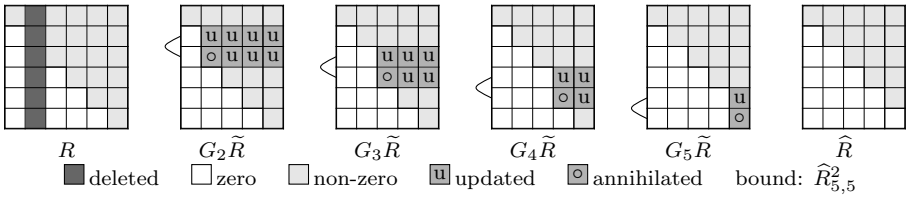
The rotation G_j reduces the element $\widetilde{R}_{j+1,j}$ to zero ($j \in \{i, \ldots, d\}$). Note that only the last $d - j + 1$ elements in rows j and $j + 1$ of $\widetilde{R}$ are affected. The application of a rotation to two biplanar elements x and

y can be written

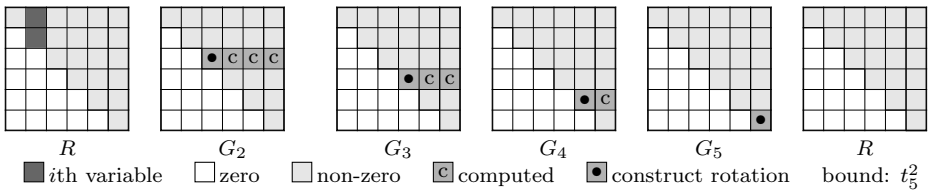
$$\begin{bmatrix} \widetilde{x} \\ \widetilde{y} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

If c and s are chosen such that $c = x/t$ and $s = y/t$, then $\widetilde{x} = t$ and $\widetilde{y} = 0$, where $t^2 = x^2 + y^2$ $(t \neq 0)$. The number of nodes in which the variables are preordered increases exponentially with the preordering radius P. This computational overhead will have a significant impact on the overall performance of the PBBA. Figure 3.4a shows the retriangularization of a 6×6 triangular matrix after deleting the second column using Givens rotations.

The complete and explicit triangularization of $\widetilde{R}$ is not necessary in determining $\widehat{R}^2_{d,d}$, the bound of the ith variable. It can be avoided by only computing the elements of $\widetilde{R}$ that play a role in deriving $\widehat{R}_{d,d}$. Thus, the Givens rotation G_j $(j \in \{i, \ldots, d\})$ will only update the last $d-j$ elements of the $(j+1)$th row of $\widetilde{R}$ explicitly, which are required by the subsequent rotation. The jth row of $\widetilde{R}$ is not modified, and the subdiagonal element $\widetilde{R}_{j+1,j}$ is not annihilated. The cost of this strategy to derive the bound of the ith variable is approximately half. The bound is given by t_d^2, the square of the value t determined by the dth Givens rotation. In order to optimize the implementation, the application of the Givens sequence is simulated without modifying the original triangular R, and the bounds are computed without copying the matrix to temporary store. The strategy is illustrated in Figure 3.4b. Its implementation is shown in Algorithm 3.



(a) Downdating the QR decomposition after deleting the corresponding column.



(b) Simulating the Givens sequence.

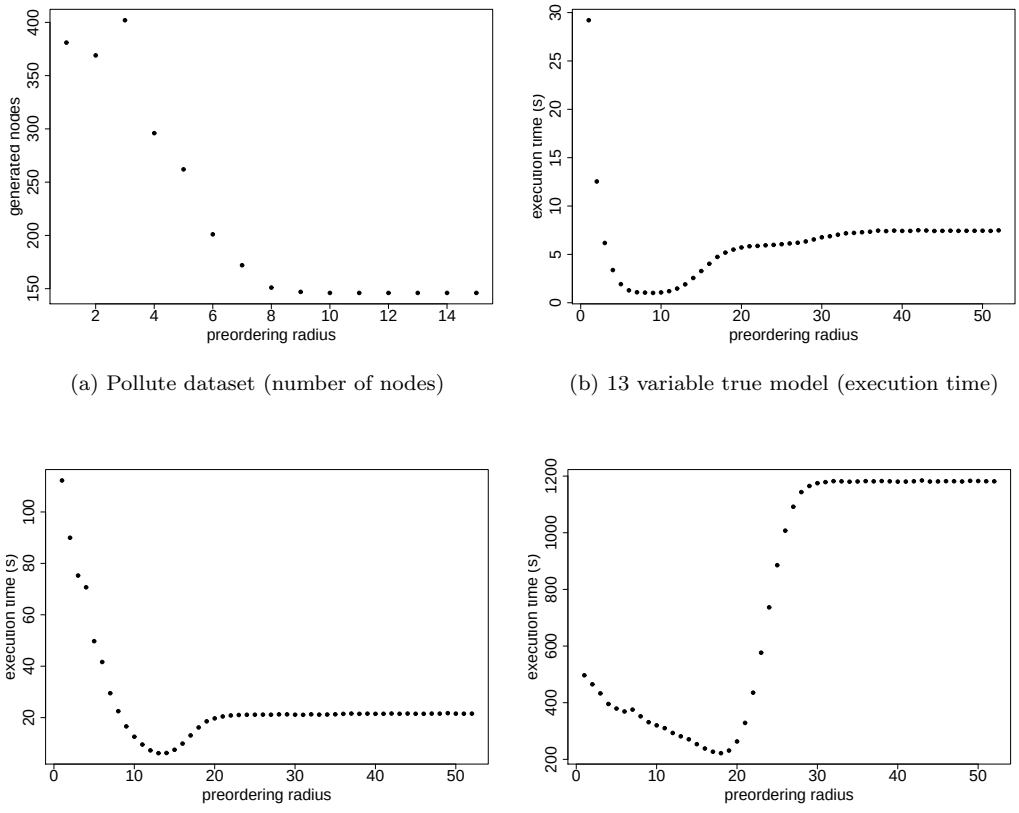
Figure 3.4: Exploiting the QR decomposition to compute the bound of the ith active variable ($i = 2$, $d = 5$).

Figure 3.5 illustrates the effects of the preordering radius P on the number of generated nodes and the execution time of the PBBA. Figure 3.5a illustrates the number of nodes generated for the Pollute dataset (15 variables) for all preordering radii $P = 1, \ldots, 15$. The number of nodes generated by the PBBA decreases steadily for $P \leq 8$, where a minimum of 146 nodes is reached. The BBA-1 generates 381 nodes. The other three figures illustrate the execution times of the PBBA for artificial datasets with 52 variables. The true models comprise 13, 26 and 39 variables, respectively. In all three cases, the PBBA represents a significant improvement over the BBA-1. For the small true model (13 variables), the BBA-1 and the PBBA require 30 seconds and 1 second ($P = 9$), respectively. For the medium true model (26 variables), the execution times are, respectively, 112 and 6 ($P = 13$) seconds. That is, the PBBA with radius 13 is almost 20 times faster than the BBA-1. Finally, for the big true model (39 variables), the PBBA with $P = 18$ is over 2 times faster

Algorithm 3 Computing the bound of the ith active variable.

procedure BOUND(R, i)
  $x_j \leftarrow R_{i,j}$ **for** $j = i+1, \ldots, d+1$
  for $j = i+1, \ldots, d+1$ **do**
    $y_k \leftarrow R_{j,k}$ **for** $k = j, \ldots, d+1$
    $t \leftarrow \sqrt{x_j^2 + y_j^2}$; $c \leftarrow x_j/t$; $s \leftarrow y_j/t$
    $x_k \leftarrow -sx_k + cy_k$ **for** $k = j+1, \ldots, d+1$
  end for
  return t^2
end procedure

than the BBA-1, which requires 500 seconds. These tests show empirically that values between $n/4$ and $n/3$ are a good choice for the preordering radius P.



(a) Pollute dataset (number of nodes)

(b) 13 variable true model (execution time)

(c) 26 variable true model (execution time)

(d) 39 variable true model (execution time)

Figure 3.5: Computational cost of the PBBA for the Pollute dataset and artificial data with 52 variables.

Table 3.3 shows the execution time of the PBBA for two datasets with $n = 64$ and $n = 80$ variables, respectively. The employed preordering radius is $P = \lfloor n/3 \rfloor$. The number of variables in the true model is given by n_{true}. Different ranges n_{a} and n_{b} have been used. When $n_{\text{a}} = 1$ and $n_{\text{b}} = n$, the PBBA computes all best-subset models. Here, the BBA-1 is computationally impracticable. It can be observed that the execution time is significantly better for narrow size ranges.

Table 3.3: Execution time (in seconds) of the Subrange PBBA for datasets with $n = 64$ and $n = 80$ variables. The preordering radius is $P = \lfloor n/3 \rfloor$, and the number of true variables is given by n_{true}.

$n = 64$ ($P = 21$)				$n = 80$ ($P = 26$)			
n_{true}	n_{a}	n_{b}	Time	n_{true}	n_{a}	n_{b}	Time
16	1	64	119	20	1	80	4205 (70 min)
	8	24	50		10	30	2309 (38 min)
32	1	64	3415	40	1	80	177383 (2 days)
	24	40	4		20	60	25732 (8 hours)
48	1	64	3531	60	1	80	1293648 (15 days)
	36	60	1		40	80	178 (3 min)

3.4 Heuristic strategies

In order to gain in computational efficiency, the heuristic BBA (HBBA) relaxes the objective of finding an optimal solution. That is, the HBBA is able to tackle large-scale models when the exhaustive BBA is found to be computationally infeasible. The heuristic algorithm ensures that

$$\text{RRE}(\widetilde{S}_i) < \tau \quad \text{for} \quad i = 1, \ldots, n,$$

where $\widetilde{S}_i$ is the (heuristic) solution subset of size i, and τ is a tolerance parameter ($\tau > 0$). Generally, the RRE of a subset S_i is given by

$$\text{RRE}(S_i) = \frac{|\text{RSS}(S_i) - \text{RSS}(S_i^*)|}{\text{RSS}(S_i^*)},$$

where S_i^* is the optimal subset of size i, which is reported by the BBA. The space of all possible submodels is not searched exhaustively. The HBBA aims to find an acceptable compromise between the brevity of the search ($\tau \to \infty$) and the quality of the solutions computed ($\tau \to 0$). The modified cutting test (Gatu & Kontoghiorghes 2006) is given by

$$(1 + \tau) \cdot \text{RSS}(S) > r_{j+1}. \tag{3.6}$$

Note that the HBBA is equivalent to the BBA for $\tau = 0$. Furthermore, (3.6) is equivalent to $0 > r_{j+1}$ if $\tau = -1$, which implies that the cutting test is never true and all nodes are generated. Hence, the HBBA with $\tau = -1$ is equivalent to the DCA.

In order to increase the capability of the heuristic strategy to tackle large subset selection problems, a new heuristic algorithm is proposed. The Level HBBA employs different tolerance values on different levels of the regression tree. It uses higher values close to the root node to encourage the cutting of large subtrees. Lower tolerance values are employed on lower levels of the tree in order to select good quality subset models. The tolerance function employed by the Level HBBA is formally defined by

$$\lambda(\ell) = 2\tau(n - \ell - 1)/(n - 1), \quad \ell \in \{0, \ldots, n - 1\},$$

where ℓ denotes the level and τ the average tolerance. The graph of the function λ is shown in Figure 3.6. The indices of the tree levels are shown in Figure 3.1.

A test is conducted to compare the Level HBBA with the HBBA. The test employs simulated datasets with $n = 36$ variables. The true model contains n_{true} true variables. Three types of datasets are simulated

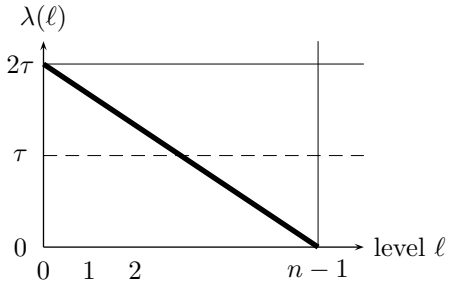


Figure 3.6: The level tolerance function λ.

with, respectively, 9, 18 and 27 true variables. The (mean) tolerance parameter is set to $\tau = 0.2$ for both algorithms. For each dataset type, 32 sample datasets are generated. The results are summarized in Table 3.4. It shows the mean number of nodes and the mean RRE. Note that the Level HBBA generates slightly fewer nodes, but the derived solution subsets are of lower quality than those computed by the HBBA. In both cases, the mean RRE is significantly lower than the value of the tolerance parameter.

Table 3.4: Mean number of nodes and RRE produced by the HBBA and Level HBBA.

n_{true}	9		18		27	
Algorithm	nodes	RRE	nodes	RRE	nodes	RRE
HBBA	14278	6e-4	47688	3e-4	35062	9e-4
Level HBBA	13129	8e-4	34427	5e-4	21455	3e-3

The Size HBBA can assign a different tolerance value to each subset size. Thus, the degree of importance of each subset size can be expressed. Subset sizes of greater importance are assigned a lower tolerance value. Less relevant subset sizes are given higher tolerance values. By setting a very high tolerance value, subset model sizes can effectively be excluded from the search. In this manner, the Size HBBA can be employed to investigate non-contiguous size ranges, unlike the Subrange BBA. The Size HBBA satisfies

$$\text{RRE}(\widetilde{S}_i) \leq \sigma_i \quad \text{for all} \quad i = 1, \ldots n,$$

where i denotes the size of the subset model and σ_i the corresponding tolerance value. It uses a variant of the cutting test (3.6). Given a node (S, k), the child node $(\text{drop}(S, i), i-1)$ is cut if

$$(1 + \sigma_j) \cdot \text{RSS}(S) > r_j \quad \text{for all} \quad j = i, \ldots, |S| - 1$$

($i \in \{k+1, \ldots, |S|-1\}$). The Size HBBA is equivalent to the previous algorithms, i.e.

$$\text{Size HBBA} \equiv \begin{cases} \text{DCA} & \text{if } \sigma_i = -1; \\ \text{Subrange DCA} & \text{if } \sigma_i = -1 \text{ for } n_{\text{a}} \leq i \leq n_{\text{b}} \text{ and } \sigma_i \gg 0 \text{ otherwise;} \\ \text{BBA} & \text{if } \sigma_i = 0; \\ \text{Subrange BBA} & \text{if } \sigma_i = 0 \text{ for } n_{\text{a}} \leq i \leq n_{\text{b}} \text{ and } \sigma_i \gg 0 \text{ otherwise;} \\ \text{HBBA} & \text{if } \sigma_i = \tau. \end{cases}$$

The Size HBBA extends all previously proposed algorithms with the exception of the Level HBBA. Thus, it can be seen as more than a mere heuristic algorithm, as it allows a very flexible investigation of all subset models.

The Size HBBA is particularly efficient when $\sigma_i = \tau > 0$ $(i \leq n/2)$ and $\sigma_j = 0$ $(j > n/2)$. Under these conditions, the Size HBBA finds the best solution for all submodels with more than $n/2$ variables. The following test shows that the Size HBBA produces solution subsets of better quality than the HBBA at a comparable cost. This is illustrated in Table 3.5. Datasets with $n = 36$ variables are simulated for the test. The HBBA is run with $\tau = 0.2$; the Size HBBA is run with $\sigma_i = 0.2$ $(i \leq 18)$ and $\sigma_j = 0$ $(j > 18)$. Furthermore, the results are consistent with the observed behavior of the Subrange BBA. For larger subsets, wider subranges can be included in the search at a reasonable cost. In case of the Size HBBA, constraints on larger submodels can be stricter (i.e. a lower tolerance) without impeding the computational efficiency. This may be due to the asymmetric structure of the tree (see Figure 3.1). A low tolerance for large subsets does not prevent the algorithm from cutting big subtrees on the left side of the regression tree.

Table 3.5: Mean number of nodes and RRE produced by the HBBA and Size HBBA ($n = 36$ variables).

n_{true}	9		18		27	
Algorithm	nodes	RRE	nodes	RRE	nodes	RRE
HBBA	12781	8e-4	38716	4e-4	39907	1e-3
Size HBBA	15079	2e-4	39457	2e-4	40250	3e-4

3.5 Conclusions

Various algorithms for computing the best subset regression models are discussed. They improve and extend exhaustive and heuristic strategies aiming at solving large-scale model selection problems. The new algorithms are based on a dropping column algorithm (DCA), which derives all possible subset models by generating a regression tree (Gatu & Kontoghiorghes 2003, Gatu & Kontoghiorghes 2006, Smith & Bremner 1989).

An algorithm (Subrange DCA) that computes all subset models within a given range of model sizes is proposed. The Subrange DCA is a generalization of the DCA. It generates a subtree of the all-subsets tree derived by the DCA. Theoretical measures of complexity are derived (see Appendix). The theoretical complexities are confirmed by experiment. A branch and bound strategy (Gatu & Kontoghiorghes 2006) is applied in the tree generated by the Subrange DCA.

The preordering of the initial set of variables improves the computational performance of the branch and bound algorithm (BBA) significantly. However, the BBA with preordering (BBA-1) might fail to detect significant combinations of variables in the root node. Hence, a more robust preordering strategy is considered. Subsets of variables are sorted inside the regression tree after some variables have already been deleted. Thus, important combinations of variables are more likely to be identified and exploited by the algorithm.

A preordering BBA (PBBA) that generalizes the BBA-1 is investigated. Unlike the BBA-1, the PBBA applies variable preordering in nodes of arbitrary radii. The node radius provides a measure of the distance between a node and the root node. Experiments have shown that the number of nodes generated by the PBBA decreases as the preordering radius increases. However, variable preordering requires the retriangularization

of an upper triangular matrix after deleting a column, and it incurs a considerable computational overhead. A computationally efficient strategy to compute the strength of a variable is designed, which avoids the explicit retriangularization and reduces the computational load. The best performance is achieved when the variables are preordered in nodes with a radius between one quarter and one third of the total number of variables. The PBBA significantly reduces the computational time required to derive the best submodels when compared to the existing BBA-1. This allows the PBBA to tackle subset selection problems that have previously been considered computationally infeasible.

A second class of algorithms is presented. They improve upon the heuristic BBA (HBBA) (Gatu & Kontoghiorghes 2006). The so-called Level HBBA applies different tolerances on different levels of the regression tree. When the algorithms are employed with equal mean tolerance, the Level HBBA generates fewer nodes than the HBBA. Although the submodels computed by the Level HBBA are of lower quality than those derived by the HBBA, the relative residual errors remain far below the mean tolerance. On the other hand, the size heuristic BBA (Size HBBA) assigns different tolerance values to variable subsets of different sizes. The subset models derived by the Size HBBA are of higher quality than those produced by the HBBA. For a comparable computational effort, the submodels obtained from the Size HBBA are closer the optimal solution. The Size HBBA generalizes the DCA, Subrange DCA, BBA, Subrange BBA and HBBA. This makes the Size HBBA a powerful and flexible tool for computing subset models. Within this context, it extends the Subrange BBA by allowing the investigation of submodels in non-contiguous size ranges.

Computationally less expensive criteria to be used by the PBBA to preorder the variables should be investigated. Parallel strategies to compute the bound of a model after deleting a variable need consideration (Hofmann & Kontoghiorghes 2006). It might be fruitful to explore the possibility of designing a dynamic heuristic BBA, which would automatically determine the tolerance value in a given node based on a learning strategy. The BBA could be parallelized by employing a task farming strategy on heterogeneous parallel systems. The adaptation of the strategies to the vector autoregressive model is currently under investigation (Gatu & Kontoghiorghes 2005, Gatu & Kontoghiorghes 2006).

3.A Subrange model selection: complexity analysis

Let the pair (S, k) denote a node of the regression tree, where S is a set of n variables and k the number of passive variables ($0 \leq k < n$). A formal representation of the DCA regression tree is given by

$$\Delta(S,k) = \begin{cases} (S,k) & \text{if } k = n-1, \\ \big((S,k), \Delta(\text{drop}(S,k+1),k), \ldots, \Delta(\text{drop}(S,n-1),n-2)\big) & \text{if } k < n-1. \end{cases}$$

The operation $\text{drop}(S,i)$ deletes the ith variable in $S = (s_1, \ldots, s_n)$. The QR decomposition is downdated after the corresponding column of the data matrix has been deleted. Orthogonal Givens rotations are employed to reconstruct the upper triangular factor. An elementary operation is defined as the rotation of two vector elements. The cost of one elementary operation is 6 flops. The number of elementary operations required by the drop operation is

$$T_{\text{drop}}(S,i) = (n-i+1)(n-i+2)/2.$$

The passive variables $s_1, \ldots, s_k$ are not dropped, i.e. they are inherited by all child nodes. All active variables $s_{k+1}, \ldots, s_n$, except the last one, are dropped in turn to generate new nodes. The structure of the tree can

be expressed in terms of the number of active variables $d = n - k$. The simplified representation $\Delta(d)$ of the regression tree $\Delta(S, k)$ is given by

$$\Delta(d) = \begin{cases} (d) & \text{if } d = 1, \\ \big((d), \Delta(d-1), \ldots, \Delta(1)\big) & \text{if } d > 1, \end{cases}$$

where (d) is a node with d active variables. The number of nodes and elementary operations are calculated, respectively, by

$$N(d) = 1 + \sum_{i=1}^{d-1} N(d-i) = 2^{d-1}$$

and

$$T(d) = \sum_{i=1}^{d-1} (T_{\text{drop}}(d, i) + T(d-i)) = 7 \cdot 2^{d-1} - (d^2 + 5d + 8)/2.$$

Here, $T_{\text{drop}}(d, i)$ is the complexity of dropping the ith of d active variables ($i \in \{1, \ldots, d\}$).

Let n_{a} designate a model size ($1 \leq n_{\text{a}} \leq n$). Then, $\Delta_{n_{\text{a}}}(S, k)$ denotes the subtree of $\Delta(S, k)$ that consists of all nodes that produce exactly one model of size n_{a} ($0 \leq k < n_{\text{a}}$). It is equivalent to

$$\Delta_a(d) = \begin{cases} (d) & \text{if } d = a, \\ \big((d), \Delta_a(d-1), \ldots, \Delta_1(d-a)\big) & \text{if } d > a, \end{cases}$$

where $a = n_{\text{a}} - k$. The number of nodes is calculated by

$$\begin{aligned} N_a(d) &= \begin{cases} 1 & \text{if } d = a, \\ 1 + \sum_{i=1}^{a} N_{a-i+1}(d-i) & \text{if } d > a \end{cases} \\ &= \binom{d}{a} = \frac{d!}{a!(d-a)!}. \end{aligned}$$

Similarly, the number of elementary operations required to construct $\Delta_a(d)$ is calculated by

$$\begin{aligned} T_a(d) &= \begin{cases} 0 & \text{if } d = a, \\ \sum_{i=1}^{a} (T_{\text{drop}}(d, i) + T_{a-i+1}(d-i)) & \text{if } d > a \end{cases} \\ &= T_{\text{drop}}(d, 1) + T_{a-1}(d-1) + T_a(d-1). \end{aligned}$$

The closed form

$$T_a(d) = \sum_{i=0}^{a-1} \sum_{j=i}^{d-a+i-1} \binom{j}{i} T_{\text{drop}}(d-j, 1)$$

is obtained by the generating function

$$G(x, y) = (1 - y(1+x))^{-1} \sum_{0<i<j} T_{\text{drop}}(j, i) x^i y^j$$

of $T_a(d)$. For $k = 0$ (i.e. $a = n_a$ and $d = n$), this corresponds to the number of elementary operations necessary to compute all subset models comprising n_{a} out of n variables.

Now, let $\Delta_{n_{\text{a}}, n_{\text{b}}}(S, k)$ denote the tree that produces all subset models with i variables, $i = n_{\text{a}}, \ldots, n_{\text{b}}$ ($1 \leq n_{\text{a}} \leq n_{\text{b}} \leq n$, $0 \leq k < n_{\text{a}}$). It is equivalent to

$$\Delta_{a,b}(d) = \begin{cases} (d) & \text{if } d = a, \\ \Delta_{a,b-1}(d) & \text{if } d = b, \\ \big((d), \Delta_{a,b}(d-1), \ldots, \Delta_{1,b-a+1}(d-a), \ldots, \Delta_{1,1}(d-b)\big) & \text{if } d > b, \end{cases}$$

where $a = n_\mathrm{a} - k$ and $b = n_\mathrm{b} - k$. This tree can be seen as the union of all trees $\Delta_c(d)$, $c = a, \ldots, b$. Hence, the number of nodes and operations can be calculated, respectively, by

$$N_{a,b}(d) = \sum_{c=a}^{b} N_c(d) - \sum_{c=a}^{b-1} N'_c(d)$$

and

$$T_{a,b}(d) = \sum_{c=a}^{b} T_c(d) - \sum_{c=a}^{b-1} T'_c(d).$$

Here,

$$N'_c(d) = \binom{d-1}{c}$$

and

$$T'_c(d) = \sum_{i=0}^{c-1} \sum_{j=i}^{d-c+i-2} \binom{j}{i} T_{\mathrm{drop}}(d-j, 1)$$

are the number of nodes and operations that have been counted twice. Specifically, these are given by the subtree $\Delta'_{n_\mathrm{c}}(S,k)$, which represents the intersection of the two trees $\Delta_{n_\mathrm{c}}(S,k)$ and $\Delta_{n_\mathrm{c}+1}(S,k)$ $(1 \leq n_\mathrm{c} < n)$. Its structure is given by

$$\Delta'_c(d) = \begin{cases} (d) & \text{if } d = c+1, \\ \big((d), \Delta'_c(d-1), \ldots, \Delta'_1(d-a)\big) & \text{if } d < c+1, \end{cases}$$

where $c = n_\mathrm{c} - k$.

Chapter 4

An exact least trimmed squares algorithm for a range of coverage values

Abstract. A new algorithm for solving exact least trimmed squares (LTS) regression is presented. The adding row algorithm (ARA) extends existing methods that compute the LTS estimator for a given coverage. It employs a tree strategy to compute a set of LTS regressors for a range of coverage values. Thus, prior knowledge of the optimal coverage is not required. New nodes in the regression tree are generated by updating the QR decomposition of the data matrix after adding one observation to the regression model. The ARA is enhanced by employing a branch and bound strategy. The branch and bound algorithm is an exhaustive algorithm that uses a cutting test to prune non-optimal subtrees. It significantly improves the computational performance of the ARA. Observation preordering throughout the traversal of the regression tree is investigated. A computationally efficient and numerically stable calculation of the bounds using Givens rotations is designed around the QR decomposition, avoiding the need to explicitly update the triangular factor when an observation is added. This reduces the overall computational load of the preordering device by approximately half. A solution is proposed to allow preordering when the model is underdetermined. It employs pseudo-orthogonal rotations to downdate the QR decomposition. The strategies are illustrated in practical examples. Experimental results confirm the computational efficiency of the proposed algorithms.

Keywords. Least trimmed squares, outliers, regression tree algorithms, QR factorization.

4.1 Introduction

Least squares regression is sensitive to outliers. This has prompted the search for regression estimators which are resistant to data points that deviate from the usual assumptions. The goal of positive-breakdown estimators is to be robust against possible outliers (Rousseeuw 1997). The breakdown point provides a crude quantification of the robustness properties of an estimator. Briefly, the breakdown point is the smallest amount of contamination that may cause an estimator to take on arbitrarily large aberrant values (Donoho

& Huber 1983).

Consider the standard regression model

$$y = X\beta + \varepsilon, \tag{4.1}$$

where $y \in \mathbb{R}^n$ is the dependent-variable vector, $X \in \mathbb{R}^{n\times p}$ is the exogenous data matrix of full column rank, $\beta \in \mathbb{R}^p$ is the coefficient vector, and $\varepsilon \in \mathbb{R}^n$ is the noise vector. It is usually assumed that ε is normally distributed with zero mean and variance-covariance matrix $\sigma^2 I_n$. Least squares (LS) regression consists in minimizing the residual sum of squares (RSS). One outlier may be sufficient to compromise the LS estimator. In other words, the finite-sample breakdown point of the LS estimator is $1/n$ and therefore tends to 0 when n is large (Rousseeuw 1997). Several positive-breakdown methods for robust regression have been proposed, such as the least median of squares (LMS) (Rousseeuw 1984). The LMS is defined by minimizing $\text{med}_i\, \widehat{\varepsilon}_i^2$. The LMS attains the highest possible breakdown value, namely $(\lfloor (n-p)/2 \rfloor + 1)/n$. This means that the LMS fit stays in a bounded region whenever $\lfloor (n-p)/2 \rfloor$ or fewer observations are replaced by arbitrary points (Rousseeuw & Van Driessen 2006).

The least trimmed squares (LTS) estimator possesses better theoretical properties than the LMS (Rousseeuw & Van Driessen 2006, Hössjer 1994). The objective of the LTS estimator is to minimize

$$\sum_{i=1}^{h} \widehat{\varepsilon}_{[i]}^2,$$

where $h \in \{1, \ldots, n\}$, and $\widehat{\varepsilon}_{[1]}, \ldots, \widehat{\varepsilon}_{[n]}$ denote the residuals sorted in increasing order. The parameter h is called coverage. This is equivalent to finding the h-subset of observations with the smallest LS objective function. The LTS regression estimate is then the LS fit to these h points. The breakdown value of LTS with $h = \lfloor (n+p+1)/2 \rfloor$ is equivalent to that of the LMS. In spite of its advantages over the LMS estimator, the LTS estimator has been applied less often because it is computationally demanding. For the multiple linear regression model, the trivial algorithm that explicitly enumerates and computes the RSS for all h-subsets works if the number of observations is relatively small, i.e. less than 30; otherwise, the computational load is prohibitive. To overcome this drawback, several approximate algorithms have been proposed. These include the Progress algorithm (Rousseeuw & Leroy 1987), the feasible solution algorithm (FSA) (Hawkins 1994, Hawkins & Olive 1999) and the Fast LTS algorithm (Rousseeuw & Van Driessen 2006). However, these algorithms do not inspect all combinations of observations and are not guaranteed to find the optimal solution. An exact algorithm to calculate the LTS estimator has been proposed (Agulló 2001). It is based on a branch and bound procedure that does not require the explicit enumeration of all h-subsets. It therefore reduces the computational load of the trivial algorithm significantly. A tree algorithm to enumerate subsets of observations has been suggested in the context of outlier detection (Belsley et al. 1980).

High-breakdown estimators may pick up local linear trends with different slopes compared to the global linear trend. This means that high-breakdown estimators may have arbitrarily low efficiency (Stefanski 1991, Morgenthaler 1991). For higher values of the breakdown point the possibility for this to happen is bigger. Thus, for small n/p it is preferable to use a method with lower breakdown value, such as the LTS with larger h (Rousseeuw 1997). Here, an adding row algorithm (ARA) is proposed, which computes the exact LTS estimates for a range $\{h_{\min}, \ldots, h_{\max}\}$ of coverages. It renders possible the efficient computation and investigation of a set of exact LTS estimators for distinct coverage values. A branch and bound algorithm (BBA) improves upon the ARA. Its computational efficiency is further improved by preordering the observations.

The ARA, and its application to LTS regression, is discussed in Section 4.2. The BBA is introduced in Section 4.3, observation preordering is investigated and experimental results are presented. Conclusions are presented in Section 4.4. Simulations are conducted in the R statistical software environment, on a Pentium-class machine with 2Gb of RAM.

4.2 Adding row algorithm

As noted by Belsley et al. (1980), there is a strong correspondence between row selection techniques and procedures for computing the all-possible-column-subsets regression. Within the context of variable subset selection, a dropping column algorithm (DCA) has been discussed (Gatu & Kontoghiorghes 2003, Gatu & Kontoghiorghes 2006, Gatu et al. 2007, Smith & Bremner 1989). The adding row algorithm (ARA) presented here computes the all-observation-subsets regression. The organization of the algorithm is similar to that of the DCA and is determined by the all-subsets tree (Furnival & Wilson 1974, Gatu & Kontoghiorghes 2003, Smith & Bremner 1989). The tree is illustrated in Figure 4.1, where the number of observations in the model is $n = 4$.

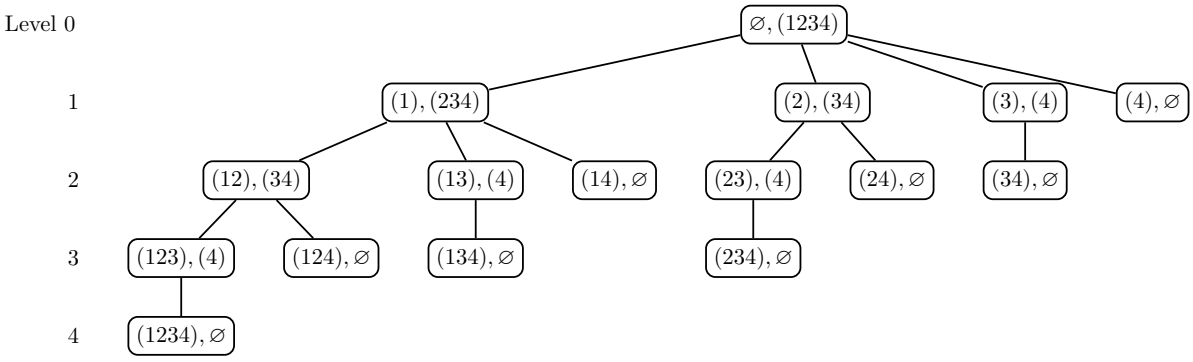


Figure 4.1: The ARA regression tree where $J = (1234)$.

The observations or points in model (4.1) are designated by their indices $J = (1, \ldots, n)$. A node is denoted by (S, A), where S and A are two disjoint, ordered subsets of observations ($S \subseteq J$, $A \subseteq J$, $S \cap A = \varnothing$). The indices in S and A designate the observations that are selected and available, respectively. Observations in A can be selected to form new nodes. The observation subset model that contains the observations in S is defined by $\begin{bmatrix} X_S & y_S \end{bmatrix}$ and is assumed to be of full rank. Let $\widehat{\beta}_S$ denote the LS estimator of the points in S. The regression model is represented by means of the numerically stable QR decomposition (QRD)

$$Q_S^{\mathrm{T}} \begin{bmatrix} X_S & y_S \end{bmatrix} = \begin{bmatrix} R_S & z_S \\ 0 & w \end{bmatrix} \begin{matrix} p \\ |S|-p \end{matrix}, \tag{4.2}$$

where Q_S is orthogonal, and R_S is square upper triangular and non-singular. RSS(S) denotes the RSS of $\widehat{\beta}_S$ and is given by $w^{\mathrm{T}}w$. Note that for underdetermined models ($|S| < p$), the LS estimator is undefined and the RSS is 0. The orthogonal factor Q_S^{T} is typically a product of Givens rotations or Householder reflectors (Golub & Van Loan 1996).

Given any node (S, A), its child nodes are

$$\big(\mathrm{add}(S, a_1), A_{2:}\big), \big(\mathrm{add}(S, a_2), A_{3:}\big), \ldots, \big(\mathrm{add}(S, a_{|A|}), \varnothing\big),$$

where $A = (a_1, \ldots, a_{|A|})$, and $A_{i:}$ denotes the subset of A that contains all but the $i-1$ first observations. The operation $\text{add}(S, a_i)$ constructs the new linear model $S \cup \{a_i\}$ by updating the linear model S with observation a_i ($i \in \{1, \ldots, |A|\}$). Effective algorithms to update the quantities R_S, z_S and $\text{RSS}(S)$ after adding an observation exist (Gill et al. 1974). A Cholesky updating routine based on orthogonal Givens rotations can be found in the Linpack numerical library (Dongarra, Bunch, Moler & Stewart 1979). It requires approximately $3p^2$ flops (Björck, Park & Eldén 1994).

A straight forward brute force (BF) method to compute the exact LTS estimator $\widehat{\beta}_h$ for a given coverage h consists in enumerating all possible h-subsets, solving the LS problem for each subset. This implies computing $\binom{n}{h} = n!/(h!(n-h)!)$ QRDs. Thus, the computational cost amounts to approximately $T_{\text{BF}}(h) = \binom{n}{h} \cdot 3hp^2$ flops, where $3hp^2$ is the approximate number of flops necessary to compute one QRD. On the other hand, the specialized algorithm ARA_h lists the observation subsets in an order predetermined by the all-subsets tree. Starting at the root node, it traverses the ancestor nodes of all nodes on level h. The nodes on level h are included in the traversal. Although it enumerates more subsets than the BF algorithm, the computational load is lower. By exploiting the information gathered in intermediate nodes, the QRDs are obtained cheaply. That is, the QRD in a node on level ℓ is partially available as the QRD in the parent node on level $\ell - 1$. The new subset model is obtained by selecting an available observation. Numerically, this implies updating the parent QRD by one observation.

Let $\Delta(S, A)$ denote the all-observation-subsets tree with root node (S, A), and let $\Delta_h(S, A)$ denote the tree generated by the ARA_h. It is equivalent to the tree employed in feature subset regression by Narendra & Fukunaga (1977) and by Agulló's (2001) exact LTS algorithm (hereafter denoted by AGLA). Formally,

$$\Delta_h(S, A) = \begin{cases} \varnothing & \text{if} \quad |S| + |A| < h \quad \text{or} \quad |S| > h, \\ \big((S, A), \Delta_h(\text{add}(S, a_1), A_{2:}), \ldots, \Delta_h(\text{add}(S, a_{|A|}), \varnothing)\big), & \text{otherwise.} \end{cases}$$

The tree $\Delta_h(\varnothing, J)$ consists of $\binom{n+1}{h}$ nodes, where $n = |J|$. Thus, the computational cost of the ARA is approximately $T_{\text{ARA}}(h) = \binom{n+1}{h} \cdot 3p^2$ flops. In other words,

$$T_{\text{BF}}(h)/T_{\text{ARA}}(h) \approx \alpha(1-\alpha)n + \alpha = O(n),$$

where $n \gg p$ and $h = \alpha n$ ($\alpha < 1$). The ARA_h is $O(n)$ times faster than the brute force method.

The optimal value of h will both resist outliers in the data and give the highest efficiency, i.e. will accurately reveal the global linear trend. In other words, if the data stem from a normal distribution with q outliers, then the optimal h will be $n - q$, hopefully neglecting the q outliers and using the information within the other $n-q$ observations to form an accurate estimate of β. In practice however, this value is never known before using LTS regression (Atkinson & Cheng 1999). The ARA_h must be executed repetitively to compute the LTS estimators for all coverages h from $h_{\min}$ to $h_{\max}$, once for each h. This implies generating $\sum_{h=h_{\min}}^{h_{\max}} \binom{n+1}{h}$ nodes. Notice that some nodes will be computed several times over. It is thus inefficient to analyze the same data several times for different values of h.

Now, let $\Delta_{h_{\min}}^{h_{\max}}(S, A)$ denote the subtree of $\Delta(S, A)$ that consists of the nodes that are either on level ℓ or are an ancestor of a node on level ℓ, for all $\ell = h_{\min}, \ldots, h_{\max}$. Formally,

$$\Delta_{h_{\min}}^{h_{\max}}(S, A) = \begin{cases} \varnothing & \text{if} \quad |S| + |A| < h_{\min} \quad \text{or} \quad |S| > h_{\max}, \\ \big((S, A), \Delta_{h_{\min}}^{h_{\max}}(\text{add}(S, a_1), A_{2:}), \ldots, \Delta_{h_{\min}}^{h_{\max}}(\text{add}(S, a_{|A|}), \varnothing)\big), & \text{otherwise.} \end{cases}$$

The $\mathrm{ARA}_{h_{\min}}^{h_{\max}}$ generates the tree $\Delta_{h_{\min}}^{h_{\max}}(\varnothing, J)$ and returns the set of LTS estimators $\widehat{\beta}_h$ for all $h = h_{\min}, \ldots, h_{\max}$. It is optimal in the sense that it does not generate any unnecessary nodes and does not generate any node more than once. The number of nodes that it computes is given by

$$N_{\mathrm{ARA}}(h_{\min}, h_{\max}) = \sum_{h=h_{\min}}^{h_{\max}} \binom{n+1}{h} - \sum_{h=h_{\min}}^{h_{\max}-1} \binom{n}{h},$$

thus improving over the previous approach. The complete procedure for generating $\Delta_{h_{\min}}^{h_{\max}}(\varnothing, J)$ is given in Algorithm 4. Nodes which await processing are held in a node list. The list is managed according to a last in, first out (LIFO) strategy (Burks, Warren & Wright 1954, Newell & Shaw 1957). The output of the algorithm is $r_{h_{\min}:h_{\max}}$, where r_h is the RSS of the LTS estimate $\widehat{\beta}_h$. A detailed analysis of the tree structure and of the associated complexities has been given in the context of variable subset selection (Hofmann et al. 2007).

Algorithm 4 The adding row algorithm (ARA).

```
1:  procedure ARA(J, h_min, h_max, r)
2:      r_h ← +∞ for h = h_min, ..., h_max
3:      Insert node (∅, J) in node list
4:      while not empty(node list) do
5:          Remove node (S, A) from node list
6:          ρ ← RSS(S)
7:          if ρ < r_|S| then r_|S| ← ρ
8:          if |S| + |A| ≥ h_min and |S| < h_max then
9:              for i = 1, ..., |S| + |A| − h_min + 1 do
10:                 (S', A') ← (add(S, a_i), A_{i+1:})
11:                 Insert (S', A') in node list
12:             end for
13:         end if
14:     end while
15: end procedure
```

The ARA computes all LTS estimators simultaneously. It traverses the tree $\Delta_{h_{\min}}^{h_{\max}}$ only once and avoids needless computations. In contrast, the AGLA can determine only one LTS regressor at a time. Thus, in order to obtain the same list of LTS estimators, the AGLA must be executed $h_{\max} - h_{\min} + 1$ times. This implies several independent traversals of the subsets tree and the computation of redundant nodes.

4.2.1 Experimental results

To see the effect of the coverage h on the LTS estimator, the ratio suggested by Atkinson & Cheng (1999) is considered:

$$R_h = \frac{\widehat{\sigma}_h^2}{\widehat{\sigma}_{\mathrm{LS}}^2},$$

where $\widehat{\sigma}_h^2 = r_h/(h-p)$ and r_h is the RSS of the LTS estimator with coverage h. The variance estimate associated with the LS estimator is given by $\widehat{\sigma}_{\mathrm{LS}}^2 = \widehat{\sigma}_n^2$. The ratio R_h is examined with different numbers of data fitted in the model. Two data models are used in the experiment. Atkinson & Cheng (1999) use the

model

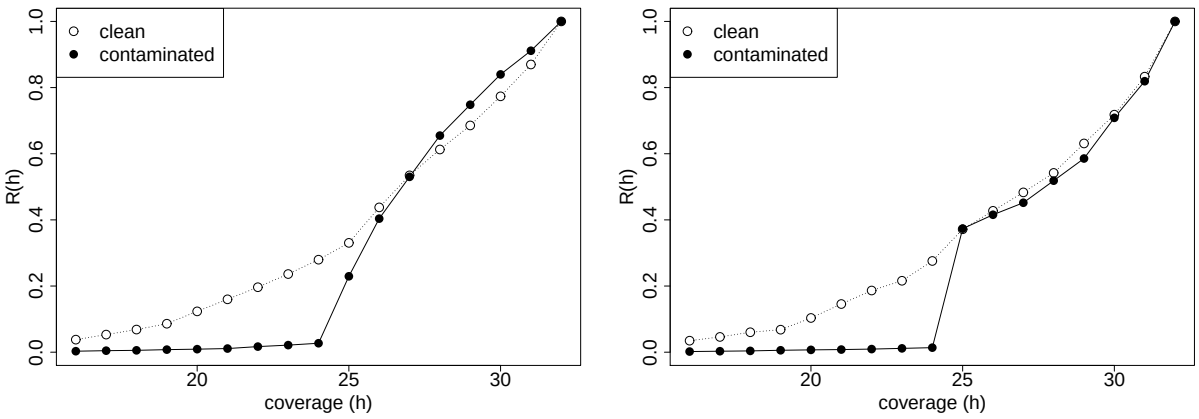
$$y_i = \beta_0 + \sum_{j=1}^{p-1} \beta_j x_{i,j} + \varepsilon_i, \quad i = 1, \ldots, n,$$

where $\varepsilon_i \sim N(0,1)$ for good data, whereas bad data are simulated from $N(12,1)$. Rousseeuw & Van Driessen (2006) use the model

$$y_i = x_{i,1} + x_{i,2} + \ldots + x_{i,p-1} + 1 + \varepsilon_i, \quad i = 1, \ldots, n,$$

where $\varepsilon_i \sim N(0,1)$ and $x_{i,j} \sim N(0,100)$. Outliers are introduced by replacing some of the $x_{i,1}$ by values that are normally distributed with mean 100 and variance 100.

For each of the two models, 100 datasets are generated with $n = 32$ and $p = 5$ and contaminated with $q = 8$ outliers. The ARA is executed with $h_{\min} = 16$ and $h_{\max} = 32$. The ARA correctly discriminates the contaminated observations in all simulated datasets. This means that in all test cases, the LTS estimator $\widehat{\beta}_{n-q=24}$ does not include any of the contaminated data points. Figure 4.2 illustrates the ratio R_h for two datasets. Each plot shows the ratio R_h for one dataset with and without contamination.



(a) Data: Atkinson & Cheng (1999).

(b) Data: Rousseeuw & Van Driessen (2006).

Figure 4.2: The ratio R_h for two types of data.

4.3 Branch and bound algorithm

The ARA is computationally prohibitive even for a moderate number of observations. The ARA can be optimized to avoid the explicit enumeration of all observation subsets. Given two sets of observations S_1 and S_2,

$$\text{if} \quad S_1 \subset S_2, \quad \text{then} \quad \text{RSS}(S_1) \leq \text{RSS}(S_2),$$

where $\text{RSS}(S)$ is the RSS of $\widehat{\beta}_S$, the LS estimator of the observations in S. That is, adding observations cannot cause the RSS of the LS estimator to decrease. This property can be used to restrict the number of evaluated subsets while searching for the best observation subset models.

Let $r_j^{(g)}$ denote the minimal RSS of all models with j observations after g nodes have been generated ($g \in \{0, \ldots, 2^n\}$). For any g, the following relationship is satisfied:

$$r_1^{(g)} \leq r_2^{(g)} \leq \ldots \leq r_n^{(g)}.$$

See Lemma 1 in Appendix 4.A for a formal proof. After the whole regression tree $\Delta(\varnothing, J)$ has been generated ($g = 2^n$), the entry r_h contains the RSS of the LTS estimator with coverage h. Consider the gth node (S, A), and let its bound be $\mathrm{RSS}(S)$. A cutting test is devised. Specifically,

$$\text{if} \quad \mathrm{RSS}(S) \geq r^{(g-1)}_{|S|+|A|-i+1}, \quad \text{then} \quad \mathrm{RSS}(V) \geq r^{(g)}_{|V|}, \tag{4.3}$$

where V is any model obtained from $\Delta(S, A_{i:})$ ($i \in \{1, \ldots, |A|\}$). See Lemma 2 for a formal proof. This implies that the subtrees $\Delta(\mathrm{add}(S, a_i), A_{i+1:})$ cannot improve the current values in $r_{1:n}$. A procedure to compute the regression tree follows. Given a node (S, A), its child nodes $(\mathrm{add}(S, a_i), A_{i+1:})$ are computed from left to right, i.e. for increasing i. For each i, if $\mathrm{RSS}(S) > r_{|S|+|A|-i+1}$, the corresponding child and its younger siblings (i.e. child nodes on the right) are discarded; otherwise, the child node is computed and inserted in a node list. The procedure is repeated for the next node in the list. This is illustrated in Algorithm 5. The break statement on Line 9 exits the for-loop on Line 8. Note that the cutting test is not effective in the first p levels of the tree, where the RSS of the submodels S is 0.

Algorithm 5 The branch and bound algorithm (BBA).

```
1:  procedure BBA(J, r)
2:      r_i ← +∞ for i = 1, ..., |J|
3:      Insert node (∅, J) in node list
4:      while not empty(node list) do
5:          Remove node (S, A) from node list
6:          ρ ← RSS(β̂_S)
7:          if ρ < r_|S| then r_|S| ← ρ
8:          for i = 1, ..., |A| do
9:              if ρ ≥ r_{|S|+|A|−i+1} break 8
10:             (S', A') ← (add(S, a_i), A_{i+1:})
11:             Insert (S', A') in node list
12:         end for
13:     end while
14: end procedure
```

The computational efficiency of the BBA improves when more nodes are cut. That is, if bigger subtrees have bigger bounds. This can be achieved by preordering the observations in each node (S, A). The BBA with preordering (PBBA) constructs nodes using "stronger" observations first by sorting the observations in A according to their strength (Agulló 2001). The exact bound $\mathrm{RSS}(S \cup \{a_i\})$ can be computed to determine the strength of the ith observation in A. This approach involves $|A|$ rank-1 Cholesky updates (Gill et al. 1974, Golub & Van Loan 1996). Note that the observations in A are sorted in decreasing order of the RSS. An alternative measure of the strength is given by the absolute residuals (Resid). The residual vector is computed with respect to $\widehat{\beta}_S$, the LS estimator of the subset S of observations. This merely involves solving an upper triangular system and is computationally non-expensive. The observations in A are sorted in decreasing order of the absolute residuals.

An efficient procedure to update the RSS after adding an observation is needed. Let $\widetilde{R}$ denote the right-hand side of (4.2) and $\widetilde{x}^{\mathrm{T}}$ the row vector corresponding to the added observation. The updating process is

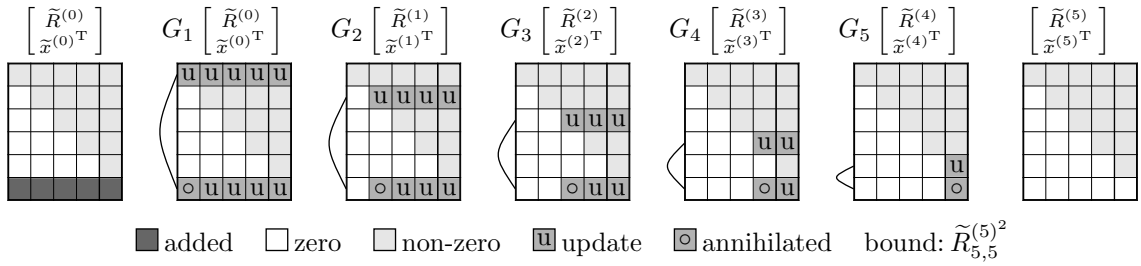
illustrated in Figure 4.3a. It involves the application of $p+1$ Givens rotations, where p is the number of independent variables. The ith Givens rotation G_i can be written

$$G_i = \begin{bmatrix} c_i & s_i \\ -s_i & c_i \end{bmatrix},$$

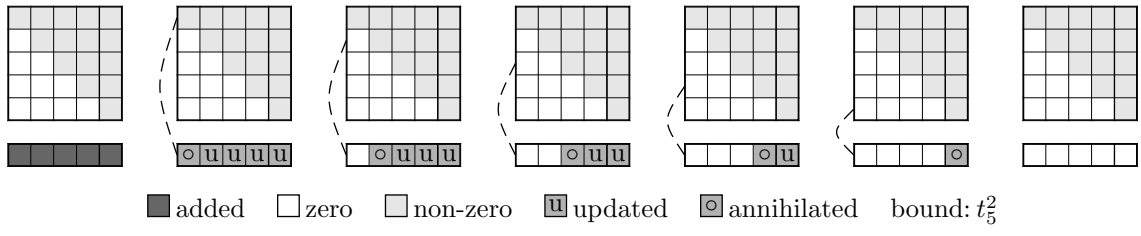
where $c_i = \widetilde{R}_{i,i}^{(i-1)}/t_i$, $s_i = \widetilde{x}_i^{(i-1)}/t_i$ and $t_i^2 = \widetilde{R}_{i,i}^{(i-1)^2} + \widetilde{x}_i^{(i-1)^2}$. The application of G_i is given by

$$G_i \begin{bmatrix} \widetilde{R}_{i,i:p+1}^{(i-1)} \\ \widetilde{x}_{i,i:p+1}^{(i-1)^{\mathrm{T}}} \end{bmatrix} = \begin{bmatrix} \widetilde{R}_{i,i:p+1}^{(i)} \\ \widetilde{x}_{i,i:p+1}^{(i)^{\mathrm{T}}} \end{bmatrix},$$

where $\widetilde{R}^{(i)}$ and $\widetilde{x}^{(i)}$ respectively denote $\widetilde{R}$ and $\widetilde{x}$ modified by the first i Givens rotations, $\widetilde{R}^{(0)} \equiv \widetilde{R}$, $\widetilde{x}^{(0)} \equiv \widetilde{x}$ and $\widetilde{x}^{(p+1)} \equiv 0$. The standard colon notation is used in order to denote submatrices and subvectors (Golub & Van Loan 1996). The bound, i.e. the RSS, is given by the square of element $(p+1, p+1)$ in $\widetilde{R}^{(p+1)} = G_{p+1} \cdots G_1 \widetilde{R}$. The procedure is computationally expensive. Note that the construction of a Givens rotation does not involve previously modified elements of $\widetilde{R}$. Thus, the sequence of Givens rotations can be carried out without explicitly modifying the upper triangular factor, as illustrated in Figure 4.3b. The bound is then given by $t_{p+1}^2 = \widetilde{R}_{p+1,p+1}^2 + \widetilde{x}_{p+1}^{(p)^2}$, the square of the quantity t determined in computing the $(p+1)$th Givens rotation. Thus, the computational complexity is roughly divided by two. The new procedure to derive the bound is illustrated by Algorithm 6. Note that it can easily be adapted to downdate the RSS with one observation by substituting hyperbolic plane rotations for Givens rotations (Alexander, Pan & Plemmons 1988, Golub & Van Loan 1996).



(a) Updating the QR decomposition after adding a row.



(b) Simulating the Givens sequence.

Figure 4.3: Exploiting the QR decomposition to compute the bound of an observation.

On the first p levels of the regression tree (i.e. $\ell < p$), the subset models S are underdetermined, i.e. $|S| < p$. The estimator $\widehat{\beta}_S$ and associated quantities are undefined. Let (S, A) denote the current node, and $U = S \cup A$. The estimator $\widehat{\beta}_U$ can be used in place of $\widehat{\beta}_S$ to evaluate the strength of the observations. It is either available in the parent node or is obtained from the left sibling node by a rank-1 Cholesky downdate. The observations a_i in A are sorted in order of increasing $\mathrm{RSS}(U - \{a_i\})$, where $\mathrm{RSS}(U - \{a_i\})$ denotes the quantity $\mathrm{RSS}(U)$ downdated with observation a_i. Alternatively, the observations can be sorted in increasing

Algorithm 6 Computing the exact bound of an observation x.

procedure BOUND(R, x)
  for $j = 1, \ldots, p+1$ **do**
    $t \leftarrow \sqrt{R_{j,j}^2 + x_j^2}$; $c \leftarrow R_{j,j}/t$; $s \leftarrow x_j/t$
    $x_j \leftarrow -sx_j + cR_{i,j}$ **for** $j = i+1, \ldots, p+1$
  end for
  return t^2
end procedure

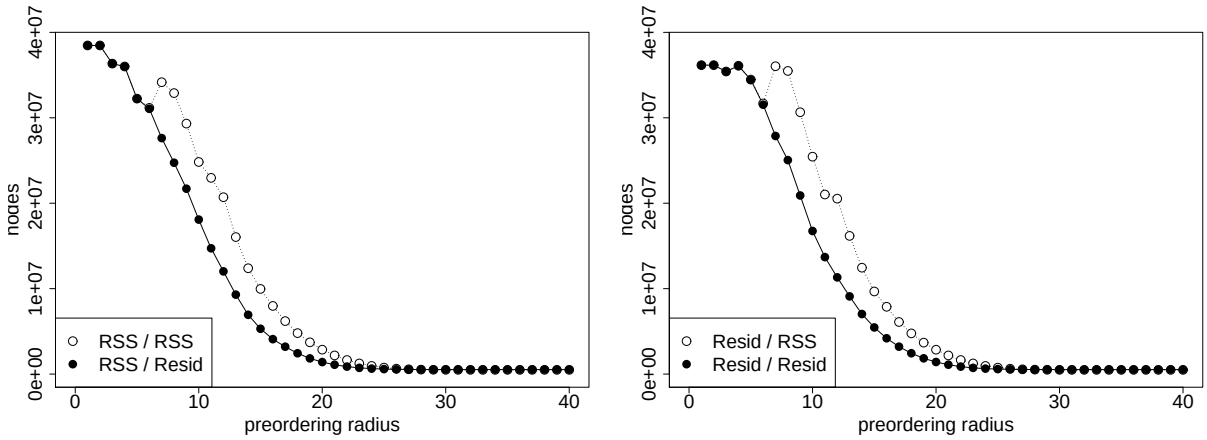
order of the absolute residuals with respect to $\widehat{\beta}_U$. For notational convenience, Resid/RSS will denote a preordering strategy that uses the absolute residuals and the RSS to define the weights when, respectively, $\ell < p$ and $\ell \geq p$. Other preordering strategies are Resid/Resid, RSS/RSS and RSS/Resid.

Preordering remains an expensive procedure. The algorithm should preorder the available observations in nodes which are at the root of big subtrees, i.e. where a potentially large number of nodes are cut. A preordering radius can be used to restrict the use of preordering (Hofmann et al. 2007). Specifically, observations are preordered in nodes whose distance from the root node is smaller than a given preordering radius. In the present context, this is equivalent to preordering the observations in all nodes such that $|A| > n - \pi$, where π denotes the preordering radius. Notice that if $\pi = 0$, no preordering occurs. If $\pi = 1$, the observations are preordered in the root node; if $\pi = n$, the observations are preordered in all nodes of the tree. For more control, two preordering radii can be defined: π_1 for $\ell < p$ and π_2 for $\ell \geq p$.

Figures 4.4 and 4.5 illustrate the computational cost of the PBBA for various preordering radii. The employed dataset contains $n = 40$ observations (8 of which are outliers), $p = 4$ variables and is generated according to Rousseeuw & Van Driessen (2006). All LTS estimates for coverage values $h_{\min} = 20$ to $h_{\max} = 40$ are computed. Four different preordering strategies are illustrated. It can be seen from Figure 4.4 that all preordering strategies generate about the same number of nodes, which decreases steadily for increasing values of π, $\pi < n/2$. The number of nodes remains constant when π is greater than approximately $n/2$. Thus, the chosen preordering radius should be close to $n/2$, and Resid should be used to preorder the observations, as it is cheaper to compute than the RSS. These findings are confirmed by Figure 4.5, which shows that a preordering radius greater than $n/2$ does not significantly improve the execution time of the algorithm. Moreover, employing Resid when $\ell \geq p$ leads to execution times that are up to 3 times lower for $\pi \geq n/2$.

The PBBA is equivalent to the AGLA under the conditions that follow. The AGLA computes the LTS estimate for one coverage value only, i.e. $h_{\min} = h_{\max} = h$. The observations are preordered in the root node and in nodes on level $\ell \geq p$. Specifically, the AGLA is equivalent to the PBBA with $\text{Resid}_{(\pi_1=1)}/\text{RSS}_{(\pi_2=n)}$. For given $h_{\min}$ and $h_{\max}$, the PBBA performs twice faster than the AGLA if it employs the same preordering strategy. Contrary to the AGLA, the PBBA does not need to step through the regression tree more than once and hence does not generate any redundant nodes.

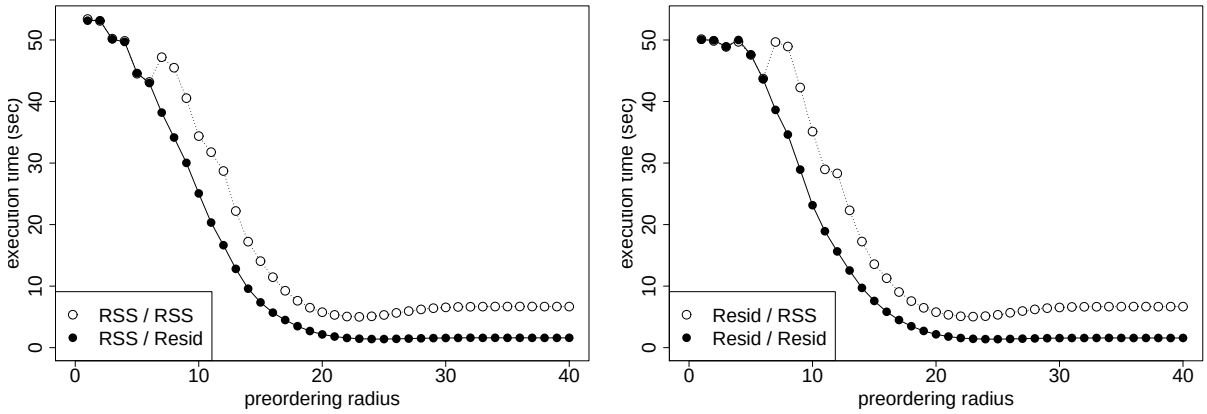
A simulation is conducted to compare the PBBA and the AGLA. Datasets with $n = 32, 36, 40, 44, 48$ observations and $n/4$ outliers are used in the experiment. The PBBA employs the absolute residuals (Resid) to preorder the observations. The preordering radius is $\pi = n/2$. The LTS estimates are computed for three different ranges $\{h_{\min}, \ldots, h_{\max}\}$: $\{n/2, \ldots, n\}$, $\{n/2, \ldots, 3n/4\}$ and $\{3n/4, \ldots, n\}$. For each n, 100 datasets are generated. The mean execution time and the mean number of nodes are reported. The results show that



(a) RSS/Resid and RSS/RSS

(b) Resid/Resid and Resid/RSS

Figure 4.4: Number of nodes produced by the PBBA for different preordering criteria (data: $n = 40$, $p = 4$).



(a) RSS/Resid and RSS/RSS

(b) Resid/Resid and Resid/RSS

Figure 4.5: Execution time (in seconds) of the PBBA for different preordering criteria (data: $n = 40$, $p = 4$).

the PBBA is 6 to 10 times faster than the AGLA. In some instances, the number of nodes generated by the AGLA is less than the number of nodes generated by the PBBA. The AGLA employs the RSS to sort the observations and preorders the observations in more nodes than the PBBA ($\pi_2 = n$). However, the RSS is more expensive to compute, and a preordering radius which is too large adds to the computational load.

4.4 Conclusions

Various strategies to compute the exact least trimmed squares (LTS) regressors are proposed. The adding row algorithm (ARA) is based on a regression tree. The computational tool is a rank-1 Cholesky updating procedure. The ARA computes a set of LTS estimates for a given coverage range. Thus, the coverage h does not need to be known in advance, and the algorithm can be used to examine the degree of contamination of the data. The branch and bound algorithm (BBA) avoids the explicit enumeration of all observation subsets. The BBA with observation preordering (PBBA) sorts the observations in each node to further

Table 4.1: Time (in seconds) required by the PBBA and AGLA to compute a range of LTS regressors for three different coverage ranges $\{h_{\min}, \ldots, h_{\max}\}$.

			AGLA			**PBBA**			
n	p	q	mean	min	max	mean	min	max	AGLA/PBBA
			Time in seconds						
32	3	8	1.16	0.60	2.71	0.18	0.092	0.40	6.44
36	3	9	3.51	1.30	11.08	0.54	0.16	1.52	6.50
40	4	10	17.51	8.17	52.44	2.34	0.91	6.74	7.48
44	4	11	49.68	19.92	123.88	6.75	2.56	16.36	7.36
48	5	12	270.73	132.55	1'164.31	31.58	12.16	122.47	8.57
			Number of nodes						
32	3	8	99'434	50'322	238'444	97'138	44'742	215'763	1.02
36	3	9	268'115	91'892	881'282	275'415	72'665	821'995	0.97
40	4	10	974'200	462'648	2'964'662	1'061'924	424'824	3'168'084	0.92
44	4	11	2'506'074	987'432	6'292'358	2'966'393	1'021'971	7'517'632	0.84
48	5	12	10'477'284	5'146'928	44'844'933	12'276'869	4'724'249	46'391'496	0.85

(a) $h_{\min} = n/2$, $h_{\max} = n$

			AGLA			**PBBA**			
n	p	q	mean	min	max	mean	min	max	AGLA/PBBA
			Time in seconds						
32	3	8	0.87	0.47	1.88	0.15	0.068	0.30	5.80
36	3	9	2.47	0.92	6.99	0.42	0.12	1.12	5.88
40	4	10	13.43	7.52	37.05	1.95	0.84	5.42	6.88
44	4	11	36.03	17.16	87.42	5.47	2.08	13.85	6.59
48	5	12	205.12	114.84	752.72	26.28	10.59	80.96	7.81
			Number of nodes						
32	3	8	77'835	40'851	169'718	84'428	37'008	180'373	0.92
36	3	9	198'201	72'552	579'627	233'379	63'821	641'923	0.85
40	4	10	776'800	411'723	2'222'448	946'947	393'425	2'670'110	0.82
44	4	11	1'906'482	878'052	4'907'138	2'613'071	826'507	7'087'033	0.73
48	5	12	8'231'868	4'612'913	30'672'512	10'971'795	3'868'554	32'010'797	0.75

(b) $h_{\min} = n/2$, $h_{\max} = 3n/4$

			AGLA			**PBBA**			
n	p	q	mean	min	max	mean	min	max	AGLA/PBBA
			Time in seconds						
32	3	8	0.40	0.10	1.44	0.061	0.012	0.22	6.67
36	3	9	1.42	0.16	5.83	0.21	0.02	0.76	6.76
40	4	10	5.51	1.00	25.71	0.67	0.11	2.99	8.22
44	4	11	18.18	2.19	60.85	2.13	0.24	6.88	8.54
48	5	12	87.88	11.99	753.21	8.79	1.18	64.96	10.00
			Number of nodes						
32	3	8	29'901	6'560	114'792	21'804	3'964	100'873	1.37
36	3	9	96'467	9'662	432'003	72'002	5'609	336'567	1.34
40	4	10	271'261	47'830	1'353'596	194'495	28'420	1'093'940	1.40
44	4	11	811'691	88'112	2'857'928	580'998	53'587	2'248'152	1.40
48	5	12	3'044'579	362'517	27'933'149	2'139'972	209'355	20'633'518	1.42

(c) $h_{\min} = 3n/4$, $h_{\max} = n$

increase the computational efficiency. In this context, a fast procedure to compute the bounds has been devised. Experiments confirm the computational efficiency of the PBBA.

A heuristic strategy which provides a solution reasonably close to the optimum can lead to smaller execution times. The heuristic BBA (HBBA) uses a tolerance parameter τ to cut subtrees (Gatu & Kontoghiorghes 2006). Although very efficient for variable subset regression, it does not entail significantly lower execution times when employed in the context of observation subset selection. The PBBA traverses the regression tree starting at the root node, which contains the empty subset model. Thus, submodels derived by the PBBA in nodes close to the root node have small bounds, and the tolerance parameter τ has little effect on the cutting test.

The PBBA is compared to the exact algorithm (AGLA) presented by Agulló (2001). The AGLA computes the LTS regressor for a given coverage h, usually $n/2$. It will discard relevant data if the degree of contamination is less than 50%. Thus, it might fail to reveal the global linear trend of the data. The PBBA can be seen as a generalization of the AGLA. Given a coverage range $\{h_{\min}, \ldots, h_{\max}\}$, the PBBA uses a more general pattern to traverse the all-subsets tree. It can thus extract the set of LTS regressors during a single execution. Furthermore, the PBBA employs a more efficient preordering strategy, resulting in a smaller computational load. Experiments show that the PBBA is 6 to 10 times faster than the AGLA to compute a range of LTS regressors. It is an efficient tool to examine the degree of contamination of the data, revealing the exact LTS estimator which is both robust and accurate.

4.A Formal proofs

The proofs of the Lemmas 1 and 2 are given. These closely follow the proofs given in (Gatu & Kontoghiorghes 2006).

Lemma 1 *$r_j^{(g)} \leq r_{j+1}^{(g)}$ for all $j = 1, \ldots, n-1$.*

Proof. 1 *The proof is by induction on the number of generated nodes. Initially, $r_j^{(0)} = +\infty$ and the proposition holds. By inductive hypothesis, the proposition holds if g nodes have been generated. It must be shown that the proposition holds after the $(g+1)$th node has been computed.*

Consider the $(g+1)$th node (S, A), with $|S| = j$. It selects the observations in S and affects r_j which, when modified, becomes $r_j^{(g+1)} = RSS(S) \leq r_j^{(g)}$. Thus, by inductive hypothesis, $r_j^{(g+1)} \leq r_{j+1}^{(g)} = r_{j+1}^{(g+1)}$. The model S was derived from its parent model S_{par} by adding an observation. Hence, $RSS(S) \geq RSS(S_{par})$ and $r_{j-1}^{(g+1)} \leq RSS(S_{par}) \leq r_j^{(g+1)}$. This completes the proof.

Lemma 2 *Given the node (S, A) and a constant $\alpha > 0$,*

$$\text{if} \quad r_{|S|+|A|}^{(g)} \leq \alpha RSS(S), \quad \text{then} \quad r_{|V|}^{(g)} \leq \alpha RSS(V),$$

where V is any observation subset obtained from $\Delta(S, A)$.

Proof. 2 *Any observation subset V of size j ($j \in [|S|+1, |S|+|A|]$) was obtained by adding one or more observations to S. Thus, $RSS(V) \geq RSS(S)$. From Lemma 1 it follows that $r_j^{(g)} \leq r_{|S|+|A|}^{(g)}$. Hence, if $r_{|S|+|A|}^{(g)} \leq \alpha RSS(S)$, then $r_j^{(g)} \leq \alpha RSS(V)$. This completes the proof.*

Chapter 5

Matrix strategies for computing the least trimmed squares estimation of the general linear and SUR models

Abstract. An algorithm for computing the exact least trimmed squares (LTS) estimator of the standard regression model has recently been proposed. The LTS algorithm is adapted to the general linear and seemingly unrelated regressions models with possible singular dispersion matrices. It searches through a regression tree to find the optimal estimates and has combinatorial complexity. The model is formulated as a generalized linear least squares problem. Efficient matrix techniques are employed to update the generalized residual sum of squares of a subset model. Specifically, the new algorithm utilizes previous computations to update a generalized QR decomposition by a single row. The sparse structure of the model is exploited. Theoretical measures of computational complexity are provided. Experimental results confirm the ability of the new algorithms to identify outlying observations.

Keywords. Least trimmed squares, general linear model, seemingly unrelated regressions, generalized linear least squares.

5.1 Introduction

Algorithms for least trimmed squares (LTS) regression of the ordinary linear model have been proposed (Agulló 2001, Rousseeuw & Leroy 1987, Rousseeuw & Van Driessen 2006). Robust multivariate methods, and multivariate LTS in particular, have also been investigated (Agulló, Croux & Van Aelst 2008, Hubert, Rousseeuw & Van Aelst 2008, Rousseeuw, Van Aelst, Van Driessen & Agulló 2004). Recently, a fast branch and bound strategy for computing the LTS estimator has been designed (Hofmann et al. 2009). Here, new numerical strategies are derived to solve the LTS regression problem for the general linear model (GLM) and seemingly unrelated regressions (SUR) model (Srivastava & Dwivedi 1979, Srivastava & Giles 1987, Zellner 1962). The new strategies exploit the matrix properties of the linear models.

Let the GLM be given by

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

where $y \in \mathbb{R}^m$, $X \in \mathbb{R}^{m\times n}$ and $\beta \in \mathbb{R}^n$ $(m \geq n)$. Without loss of generality, it will be assumed that $\Omega \in \mathbb{R}^{m\times m}$ is symmetric positive definite. The objective of the generalized LTS (GLTS) estimator is to minimize $\sum_{i=1}^{h} u_{[i]}^2$, where $u_{[1]}^2$, ..., $u_{[m]}^2$ are the generalized squared residuals sorted in increasing order, and h is the coverage parameter $(h \geq \lfloor m+n+1 \rfloor/2)$. This is equivalent to finding the h-subset of observations with the smallest generalized least squares (GLS) objective function.

The SUR model is a special case of the GLM and is written

$$y^{(i)} = X^{(i)}\beta^{(i)} + \varepsilon^{(i)}, \quad i = 1, \ldots, G, \tag{5.2}$$

where $y^{(i)} \in \mathbb{R}^m$, $X^{(i)} \in \mathbb{R}^{m\times n_i}$ $(m \geq n_i)$, $\beta^{(i)} \in \mathbb{R}^{n_i}$ and $\varepsilon^{(i)} \in \mathbb{R}^m$. Furthermore, $\mathrm{E}\big(\varepsilon_k^{(i)}\big) = 0$ and contemporaneous disturbances are correlated, i.e. $\mathrm{Var}\big(\varepsilon_t^{(i)}, \varepsilon_t^{(j)}\big) = \sigma_{ij}$ and $\mathrm{Var}\big(\varepsilon_s^{(i)}, \varepsilon_t^{(j)}\big) = 0$ if $s \neq t$. In compact form, the model may be written

$$\mathrm{vec}(Y) = \bigoplus_{i=1}^{G} X^{(i)} \,\mathrm{vec}(\{\beta^{(i)}\}_G) + \mathrm{vec}(E), \quad \mathrm{vec}(E) \sim (0, \Sigma \otimes I_m), \tag{5.3}$$

where $Y = \begin{bmatrix} y^{(1)} & \cdots & y^{(G)} \end{bmatrix} \in \mathbb{R}^{m\times G}$, $E = \begin{bmatrix} \varepsilon^{(1)} & \cdots & \varepsilon^{(G)} \end{bmatrix} \in \mathbb{R}^{m\times G}$, and $\Sigma = \begin{bmatrix} \sigma_{ij} \end{bmatrix} \in \mathbb{R}^{G\times G}$. The set of vectors $\{\beta^{(1)}, \ldots, \beta^{(G)}\}$ is denoted by $\{\beta^{(i)}\}_G$. The direct sum of matrices $\mathrm{diag}\big(X^{(1)}, \ldots, X^{(G)}\big)$ is denoted by $\bigoplus_{i=1}^{G} X^{(i)}$, and vec(•) is the vector operator, which stacks a set of column vectors.

Here, the GLM and SUR model are reformulated as a generalized linear least squares problem (GLLSP) (Foschi & Kontoghiorghes 2002, Kontoghiorghes & Clarke 1995, Paige 1978, Paige 1979*b*, Paige 1979*a*). The solution of the GLLSP can be obtained by orthogonal factorization methods and is numerically stable. Furthermore, the GLLSP can be updated efficiently after one observation has been added to the regression model (Kontoghiorghes 2004, Yanev & Kontoghiorghes 2007). The best linear unbiased estimator (BLUE) of β in (5.1) is the solution of the GLLSP

$$(\widehat{\beta}, \widehat{u}) = \underset{\beta,u}{\mathrm{argmin}} \|u\|^2 \quad \text{subject to} \quad y = X\beta + Cu,$$

where $C \in \mathbb{R}^{m\times m}$ is the lower triangular Cholesky factor of Ω, i.e. $\Omega = CC^{\mathrm{T}}$. The generalized residual sum of squares of $\widehat{\beta}$ is $\mathrm{RSS}(\widehat{\beta}) = \|\widehat{u}\|^2$.

Similarly, the BLUE of $\{\beta^{(i)}\}_G$ of the SUR model in (5.3) is obtained by solving the GLLSP

$$\big(\{\widehat{\beta}^{(i)}\}_G, \widehat{U}\big) = \underset{\{\beta^{(i)}\},U}{\mathrm{argmin}} \|U\|_{\mathrm{F}} \quad \text{subject to} \quad \mathrm{vec}(Y) = \left(\bigoplus_{i=1}^{G} X^{(i)}\right) \mathrm{vec}(\{\beta^{(i)}\}_G) + (C \otimes I_m)\,\mathrm{vec}(U),$$

where $\|\bullet\|_{\mathrm{F}}$ denotes the Frobenius norm, $C \in \mathbb{R}^{G\times G}$ is upper triangular such that $CC^{\mathrm{T}} = \Sigma$, and $U \in \mathbb{R}^{m\times G}$. The generalized residual sum of squares is given by $\mathrm{RSS}(\{\widehat{\beta}^{(i)}\}_G) = \|\widehat{U}\|_{\mathrm{F}}$.

Section 5.2 provides a brief description of the adding row algorithm, which computes the GLTS estimators for a range of coverage values. Sections 5.3 and 5.4 adapt the LTS algorithm to the GLM and SUR model. Within this context, emphasis is given to the development of efficient numerical strategies to update the GLLSP. The sparse matrix structures are exploited to minimize the cost of expensive matrix operations. Experimental results are presented. Section 5.5 concludes.

5.2 The adding row algorithm

The adding row algorithm (ARA) computes the LTS estimates for a coverage range $\{h_{\max}, \ldots, h_{\min}\}$ by generating all possible observation subsets (Hofmann et al. 2009). The observation subsets are organized by the all-subsets regression tree (Belsley et al. 1980, Gatu & Kontoghiorghes 2003). The tree is illustrated in Figure 5.1 for $m = 4$ observations. A node (S, A) corresponds to a set of selected observations S and a set of available observations A. The observations in A are used to produce new nodes. For example, in node $\big((12), (34)\big)$ on level 2, the selected observations are 1 and 2. The two child nodes are obtained by adding observations 3 and 4, respectively. If observation 4 is selected, then observation 3 is discarded to prevent duplicate subsets. Thus, it is ensured that all submodels are generated exactly once. A branch and bound strategy is employed to reduce computational time (Furnival & Wilson 1974, Gatu & Kontoghiorghes 2006, Hofmann et al. 2009).

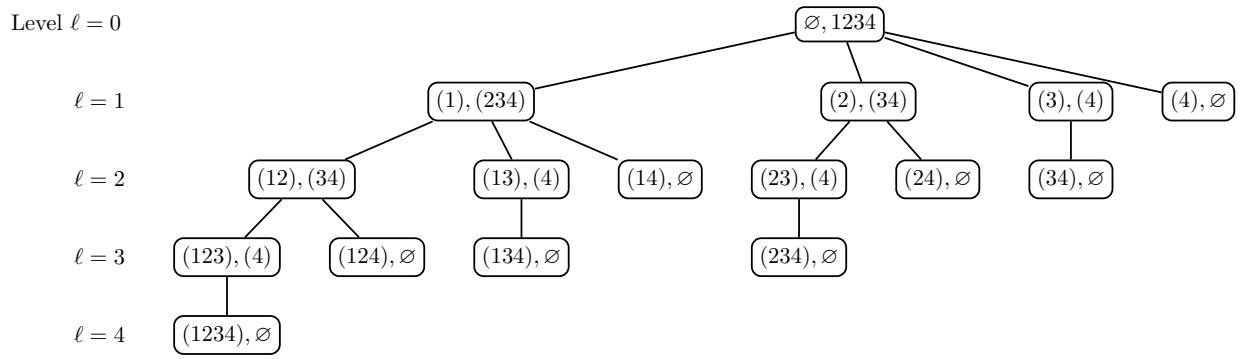


Figure 5.1: The ARA regression tree for $m = 4$ observations.

Here, the algorithm is extended to deal with the subrange LTS problem of the GLM. The generalized ARA derives the solution of a subset GLLSP equivalent to

$$(\widehat{\beta}_S, \widehat{u}_S) = \underset{\beta, u}{\operatorname{argmin}} \|u\|^2 \quad \text{subject to} \quad y_S = X_S\beta + B_S u \tag{5.4}$$

in each node. The quantities $y_S \in \mathbb{R}^{|S|}$, $X_S \in \mathbb{R}^{|S|\times n}$ and $B_S \in \mathbb{R}^{|S|\times m}$ are the subvector and submatrices of, respectively, y, X and B that correspond to the $|S|$ observations (rows) in S. The matrix B_S is such that $B_S B_S^{\mathrm{T}} = \Omega_S$, the dispersion matrix of the observation subset model S. The RSS is given by $\|\widehat{u}_S\|^2$. Furthermore, each node will store the quantities y_A, X_A and B_A, which correspond to the $|A|$ observations (rows) in A. The best RSS for every subset size is stored in a list that is updated each time a better solution is found. This is illustrated in Algorithm 7. The input argument J is the set of all observations. The output argument r contains the RSS of each computed estimator. The operation add(S, A, i) updates the subset model S with the ith observation in A ($i \in \{1, \ldots, |A|\}$).

The algorithm employs the generalized QR decomposition (GQRD) to compress the information. When a new child node is computed, an observation is selected from A and added to S. Computationally, the critical operation consists in updating the GQRD by one row. This is discussed in the next section.

Algorithm 7 Generalized adding row algorithm (GARA).

1: **procedure** ARA(J, $h_{\min}$, $h_{\max}$, r)
2: $r_i \leftarrow +\infty$ **for** $i = h_{\min}, \ldots, h_{\max}$
3: Insert $(\varnothing, J)$ in node list
4: **while not** empty(node list) **do**
5: Remove (S, A) from node list
6: **if** $\mathrm{RSS}(S) < r_{|S|}$ **then** $r_{|S|} \leftarrow \mathrm{RSS}(S)$
7: **if** $|S| < h_{\max}$ **then**
8: **for** $i = 1, \ldots, |S| + |A| - h_{\min} + 1$ **do**
9: **if** $\mathrm{RSS}(S) \geq r_{|S|+|A|-i+1}$ **break** 8 ▷ exit for-loop on Line 8
10: $(\widetilde{S}, \widetilde{A}) \leftarrow \mathrm{add}(S, A, i)$
11: Insert $(\widetilde{S}, \widetilde{A})$ in node list
12: **end for**
13: **end if**
14: **end while**
15: **end procedure**

5.3 GLTS estimation of the GLM

The solution of (5.4) is obtained from the generalized QR decomposition of X_S and B_S:

$$Q_S^{\mathrm{T}} X_S = \begin{bmatrix} R_S \\ 0 \end{bmatrix} \quad \text{and} \quad Q_S^{\mathrm{T}} B_S P_S^{\mathrm{T}} = \begin{bmatrix} T_S & T_{12} \\ 0 & T_{22} \end{bmatrix}. \tag{5.5}$$

Here, $Q_S \in \mathbb{R}^{|S|\times|S|}$ and $P_S \in \mathbb{R}^{m\times m}$ are orthogonal, $T_S \in \mathbb{R}^{n\times(m-|S|+n)}$ is upper trapezoidal, $T_{12} \in \mathbb{R}^{n\times(|S|-n)}$, and $T_{22} \in \mathbb{R}^{(|S|-n)\times(|S|-n)}$ is upper triangular. The GLLSP is then equivalent to

$$(\widehat{\beta}_S, \widehat{u}_1, \widehat{u}_2) = \underset{\beta,u_1,u_2}{\operatorname{argmin}}(\|u_1\|^2 + \|u_2\|^2) \quad \text{subject to} \quad \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} R_S \\ 0 \end{bmatrix} \beta + \begin{bmatrix} T_S & T_{12} \\ 0 & T_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix},$$

where $y_S^{\mathrm{T}} Q_S = \begin{bmatrix} y_1^{\mathrm{T}} & y_2^{\mathrm{T}} \end{bmatrix}$. Thus, $\widehat{u}_2 = T_{22}^{-1} y_2$, and the reduced-size GLLSP is given by

$$(\widehat{\beta}_S, \widehat{u}_1) = \underset{\beta,u_1}{\operatorname{argmin}} \|u_1\|^2 \quad \text{subject to} \quad z_S = R_S\beta + T_S u_1, \tag{5.6}$$

where $z_S = y_1 - T_{12}\widehat{u}_2$. It is satisfied by $\widehat{u}_1 = 0$ and $\widehat{\beta}_S = R_S^{-1} z_S$. The value of the objective function is $\mathrm{RSS}(\widehat{\beta}_S) = \|\widehat{u}_2\|^2$.

When a row $\begin{bmatrix} \omega & x^{\mathrm{T}} & b^{\mathrm{T}} \end{bmatrix}$ is added to the GLLSP in (5.4) to form the GLLSP of observation subset $\widetilde{S}$, the new problem is stated as follows:

$$(\widehat{\beta}_{\widetilde{S}}, \widehat{u}_{\widetilde{S}}) = \underset{\beta,u}{\operatorname{argmin}} \|u\|^2 \quad \text{subject to} \quad \begin{bmatrix} \omega \\ y_S \end{bmatrix} = \begin{bmatrix} x^{\mathrm{T}} \\ X_S \end{bmatrix} \beta + \begin{bmatrix} b^{\mathrm{T}} \\ B_S \end{bmatrix} u.$$

The solution is found by updating the reduced GLLSP in (5.6):

$$(\widehat{\beta}_{\widetilde{S}}, \widetilde{u}_1) = \underset{\beta,u_1}{\operatorname{argmin}} \|u_1\|^2 \quad \text{subject to} \quad \begin{bmatrix} \zeta \\ z_S \end{bmatrix} = \begin{bmatrix} x^{\mathrm{T}} \\ R_S \end{bmatrix} \beta + \begin{bmatrix} b_1^{\mathrm{T}} \\ T_S \end{bmatrix} u_1, \tag{5.7}$$

where $b^{\mathrm{T}}P_S^{\mathrm{T}} = \begin{bmatrix} b_1^{\mathrm{T}} & b_2^{\mathrm{T}} \end{bmatrix}$ and $\zeta = \omega - b_2^{\mathrm{T}}\widehat{u}_2$. It is important to note that the row b^{T} is transformed by the orthogonal P_S^{T} before it is added to the linear system. Now, consider the updated GQRD

$$\widetilde{Q}^{\mathrm{T}} \begin{bmatrix} x^{\mathrm{T}} \\ R_S \end{bmatrix} = \begin{bmatrix} R_{\widetilde{S}} \\ 0 \end{bmatrix} \quad \text{and} \quad \widetilde{Q}^{\mathrm{T}} \begin{bmatrix} b_1^{\mathrm{T}} \\ T_S \end{bmatrix} \widetilde{P}^{\mathrm{T}} = \begin{bmatrix} T_{\widetilde{S}} & t \\ 0 & \tau \end{bmatrix}.$$

The GLLSP is equivalent to

$$(\widehat{\beta}_{\widetilde{S}}, \widetilde{v}_1, \widetilde{\eta}) = \underset{\beta, v_1, \eta}{\operatorname{argmin}} (\|v_1\|^2 + \eta^2) \quad \text{subject to} \quad \begin{bmatrix} \widetilde{y}_1 \\ \xi \end{bmatrix} = \begin{bmatrix} R_{\widetilde{S}} \\ 0 \end{bmatrix} \beta + \begin{bmatrix} T_{\widetilde{S}} & t \\ 0 & \tau \end{bmatrix} \begin{bmatrix} v_1 \\ \eta \end{bmatrix}, \tag{5.8}$$

where $\begin{bmatrix} \widetilde{y}_1^{\mathrm{T}} & \xi \end{bmatrix} = \begin{bmatrix} \zeta & z_S^{\mathrm{T}} \end{bmatrix} \widetilde{Q}$. Hence, $\widetilde{\eta} = \xi/\tau$ and

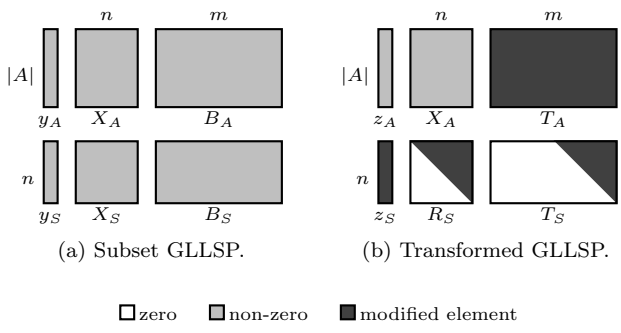
$$(\widehat{\beta}_{\widetilde{S}}, \widetilde{v}_1) = \underset{\beta, v_1}{\operatorname{argmin}} \|v_1\|^2 \quad \text{subject to} \quad z_{\widetilde{S}} = R_{\widetilde{S}}\beta + T_{\widetilde{S}} v_1 \tag{5.9}$$

is the reduced GLLSP, where $z_{\widetilde{S}} = \widetilde{y}_1 - t\widetilde{\eta}$. It is satisfied by $\widetilde{v}_1 = 0$ and $\widehat{\beta}_{\widetilde{S}} = R_{\widetilde{S}}^{-1} z_{\widetilde{S}}$. The updated value of the objective function is

$$\mathrm{RSS}(\widehat{\beta}_{\widetilde{S}}) = \mathrm{RSS}(\widehat{\beta}_S) + \widetilde{\eta}^2. \tag{5.10}$$

As expected, the RSS is non-decreasing when an observation is added to the GLM. Thus, employing the RSS as the value for the bound of a node, the ARA can implement a branch and bound strategy to reduce the number of visited nodes. Note that every time the GLLSP is updated by an observation, the GQRD is reduced by one column on the right.

This leads to the following procedure to compute the subrange GLTS. The quantities $(y_A, X_A, B_A) = (y, X, B)$, which correspond to the set of all available observations, are stored in the root node of the regression tree. The quantities (y_S, X_S, B_S) are empty in the root node. Child nodes are derived by removing an observation from A and adding it to S. This corresponds to removing a row from (y_A, X_A, B_A) and appending it to (y_S, X_S, B_S). Upon reaching a node on level n, the fully determined subset GLLSP that corresponds to the observation subset S is solved ($|S| = n$). The GQRD of X_S and B_S is computed according to (5.5), and the quantities (z_S, R_S, T_S) in (5.6) are determined. The quantities (y_S, X_S, B_S) and (y_A, X_A, B_A) become, respectively, (z_S, R_S, T_S) and (z_A, X_A, T_A), where $z_A = y_A$ and $T_A = B_A P_S^{\mathrm{T}}$. This is illustrated in Figure 5.2. The RSS is determined and stored in the current node. Note that the observations in A are transformed by P_S^{T}. In this way, rows can be selected from (z_A, X_A, T_A) and added directly to (z_S, R_S, T_S) to produce new child nodes, without requiring further transformation (see (5.7)).

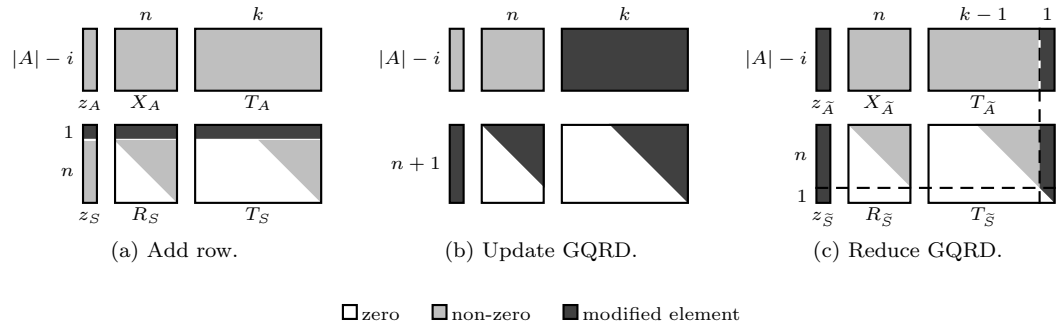


(a) Subset GLLSP. (b) Transformed GLLSP.

Figure 5.2: Solving the subset GLLSP S in a node (S, A) on level n.

In a node on level $\ell \geq n$, the ith child node is derived by removing the ith row from (z_A, X_A, T_A) and appending it to (z_S, R_S, T_S) as in (5.7). The $i-1$ first rows of (z_A, X_A, T_A) are discarded. This is illustrated

in Figure 5.3a, where $k \leq m$ is the number of columns in T_A and T_S. The GQRD is updated and the new GLLSP in (5.8) is formed (see Figure 5.3b). Note that the orthogonal transformation $\widetilde{P}^{\mathrm{T}}$ is applied to the remaining rows of T_A. Finally, the linear system is reduced to $(z_{\widetilde{S}}, R_{\widetilde{S}}, T_{\widetilde{S}})$ in (5.9). The rows in (z_A, X_A, T_A) are reduced to $(z_{\widetilde{A}}, X_{\widetilde{A}}, T_{\widetilde{A}})$ in the same manner. That is, the last column of T_A is removed and multiplied by $\widetilde{\eta}$ before being subtracted from z_A. This is illustrated in Figure 5.3c. The new RSS is computed from the RSS in the parent node according to (5.10).



(a) Add row. (b) Update GQRD. (c) Reduce GQRD.

Figure 5.3: Updating the subset GLLSP S with the ith observation of A.

The GQRD is updated by orthogonal Givens rotations (Golub & Van Loan 1996, Paige 1978). The Givens sequence is illustrated in Figure 5.4, where $p = |A|$ is the number of available observations. A row is added to (z_S, R_S, T_S) in Figure 5.4a (see also Figure 5.3a). First, $(k - n - 1)$ Givens rotations are applied to the right of T_S and T_A, as shown in Figure 5.4b, to retriangularize T_S. Next, four pairs of Givens rotations are applied from the left (Figures 5.4c–5.4f) to retriangularize R_S (see also Figure 5.3b). A rotation $\widetilde{Q}_i$ from the left annihilates one subdiagonal element in R_S, causing the fillup of a subdiagonal element in T_S. The fillup is annihilated by a Givens rotation $\widetilde{P}_j^{\mathrm{T}}$ applied from the right. Finally, in Figure 5.4g, the GLLSP is reduced (see also Figure 5.3c).

Now, consider a node with p available observations. Let $T(n, k, p)$ denote the cost of computing all possible subset models, where k is the number of columns in the right part of the GQRD ($n > 0$, $p > 0$, $k \geq n + p$). It can be expressed as

$$\begin{aligned} T(n,k,p) &= \sum_{i=1}^{p} \big(T_{\mathrm{u}}(n,k,p-i) + T(n,k-1,p-i) \big) \\ &= T_{\mathrm{u}}(n,k,p-1) + T(n,k,p-1) + T(n,k-1,p-1), \end{aligned}$$

where T_{u} is the cost of one update operation. The closed form is given by

$$T(n,k,p) = \sum_{j=1}^{p} \sum_{i=k-p+j}^{k} \binom{p-j}{k-i} T_{\mathrm{u}}(n,i,j-1).$$

The cost, in elementary rotations, of the update operation is given by

$$T_{\mathrm{u}}(n,k,p) = 3n^2/2 + k(p+1) - p/2 + 4n - 1/2.$$

Here, an elementary rotation corresponds to the application of a Givens transformation to two single-element vectors. In deriving the complexity T_{u}, it is assumed that the cost of adding a multiple of a vector to another vector of the same length is half that of rotating the same vectors (see Figure 5.4g). Thus, $T_{\mathrm{u}}(n,k,p) \in O(n^2 + kp)$, and a conservative estimate of the overall cost of the ARA can be given by $T_{\mathrm{ARA}}(m,n) \in O\big(2^m(m^2+n^2)\big)$.

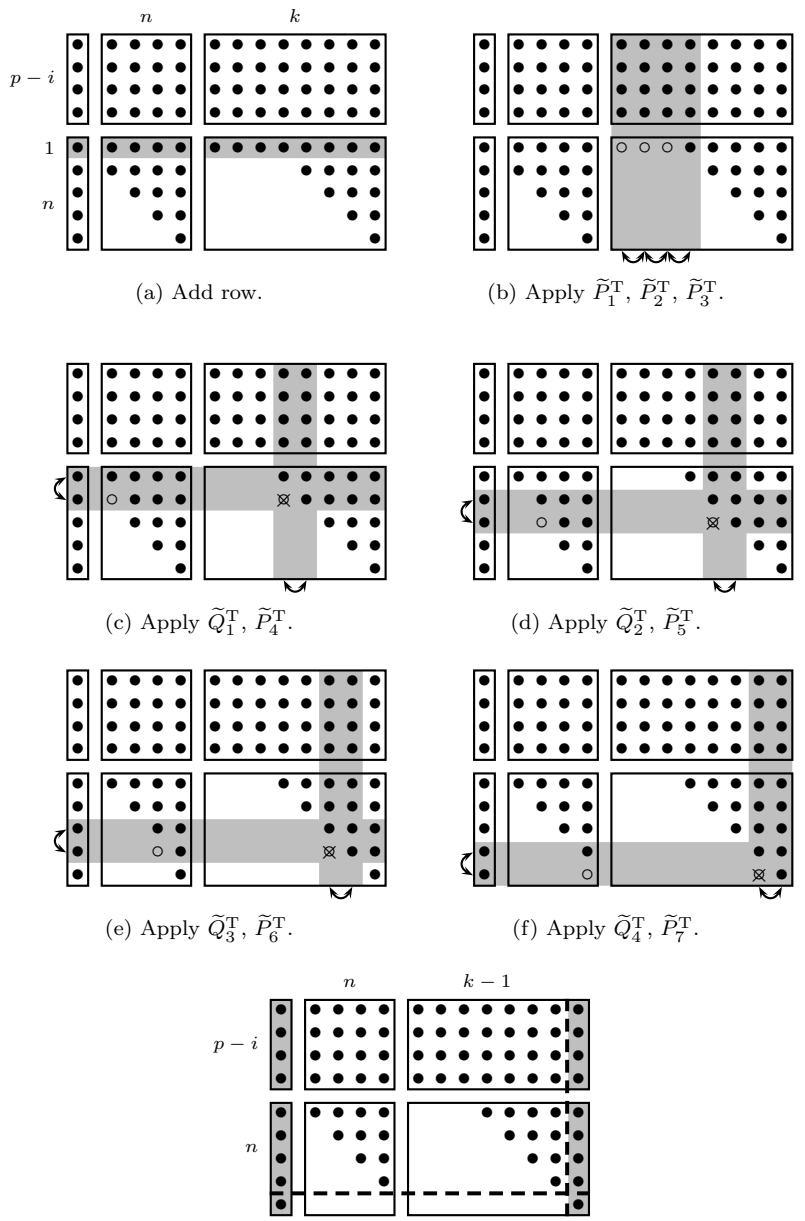


Figure 5.4: Givens sequence for updating the subset GLLSP S with the ith observation of A ($|A| = p$).

5.3.1 Experimental results

The GLM is simulated for different m and n, where m is the number of observations and n the number of variables. The variance-covariance matrix $\Omega \in \mathbb{R}^{m \times m}$ used in the experiment is given by

$$\Omega_{i,j} = \frac{\sigma^2}{1 - \varphi^2} \varphi^{|i-j|}, \tag{5.11}$$

where $\sigma = 0.25$ and $\varphi = 0.9$. For each m and n, five datasets are generated according to

$$y = X\beta + Cu,$$

where $u \sim (0, 1)$, $X \sim (0, 10)$, and C is the upper triangular Cholesky factor of Ω. Outliers are introduced by replacing some values of y by values that are normally distributed with mean 100 and standard deviation

10. The mean execution time of the ARA is shown in Table 5.1, where m_{out} indicates the number of outliers. The experiment reveals that the computational cost remains high. Thus, the algorithm cannot be used to solve large-scale problems. However, the ARA identified the outlying observations in all sample datasets. Figure 5.5 illustrates the RSS for a simulated GLM with $m = 24$ observations and $m_{\text{out}} = 6$ outliers. It can be seen that the RSS remains low for subsets with less than $h^* = m - m_{\text{out}} = 18$ observations. The RSS increases dramatically for subsets with more than 18 observations. This is an indication that the latter contain contaminated points. Thus, the plot correctly reveals the contamination rate (25%). In this case, the optimal LTS estimator is given by the observation subset of size h^* derived by the ARA, which contains all clean data points and no outliers.

Table 5.1: Mean execution time (in seconds) of the ARA for simulated GLM with m observations, n variables and m_{out} outliers.

m	n	m_{out}	time (s)
16	4	4	4
20	4	5	14
24	4	6	62
28	8	7	4'404
32	8	8	18'038
36	8	9	62'551

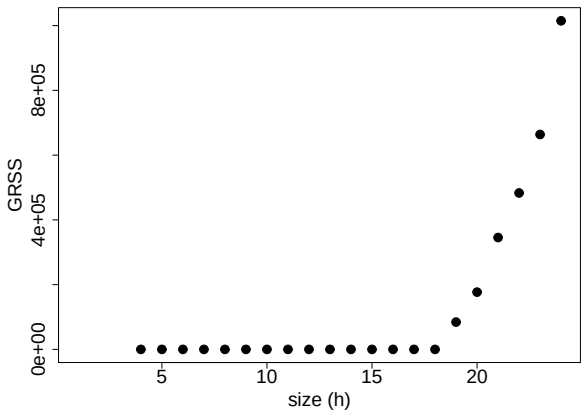


Figure 5.5: The RSS computed by the ARA for a simulated GLM with $m = 24$ observations and $m_{\text{out}} = 6$ outliers.

5.4 GLTS estimation of the SUR model

The ARA can be adapted to the balanced SUR model. In this context, S and A designate sets of *compound* observations. A compound observation consists of G observations. That is, the full SUR model consists of m compound observations, where the jth observation (row) of the ith compound observation corresponds to the ith observation (row) of the jth regression equation.

The organization of the ARA remains the same. Given a node (S, A), a new node $(\widetilde{S}, \widetilde{A})$ is derived by removing a compound observation from A and adding it to S. This corresponds to updating the subset SUR-GLLSP S with a compound observation to form the new subset SUR-GLLSP $\widetilde{S}$. The ARA can update the SUR-GLLSP once it is fully determined, i.e. when each regression equation is fully determined. If the G regression equations are sorted such that $n_1 \leq n_2 \leq \ldots \leq n_G$, then the subset SUR-GLLSP contained in a node on level $\ell = n_G$ is fully determined and can be solved. From there on, new nodes are derived by updating the subset SUR-GLLSP.

The subset model S of the SUR model given in (5.2) is

$$y_S^{(i)} = X_S^{(i)}\beta^{(i)} + \varepsilon_S^{(i)}, \quad i = 1, \ldots, G. \tag{5.12}$$

In other words, the model is balanced and contains $|S|$ compound observations. In compact form, it may be written

$$\operatorname{vec}(Y_S) = \bigoplus_{i=1}^{G} X_S^{(i)} \operatorname{vec}(\{\beta^{(i)}\}_G) + \operatorname{vec}(E_S), \quad \operatorname{vec}(E_S) \sim (0, \Sigma \otimes I_k),$$

where $Y_S = \begin{bmatrix} y_S^{(1)} & \cdots & y_S^{(G)} \end{bmatrix} \in \mathbb{R}^{|S| \times G}$ and $E_S = \begin{bmatrix} \varepsilon_S^{(1)} & \cdots & \varepsilon_S^{(G)} \end{bmatrix} \in \mathbb{R}^{|S| \times G}$. The corresponding subset SUR-GLLSP is

$$(\{\widehat{\beta}_S^{(i)}\}_G, \widehat{U}_S) = \underset{\{\beta^{(i)}\}, U}{\operatorname{argmin}} \|U\|_{\mathrm{F}} \quad \text{subject to} \quad \operatorname{vec}(Y_S) = \left(\bigoplus_{i=1}^{G} X_S^{(i)} \right) \operatorname{vec}(\{\beta^{(i)}\}_G) + (C \otimes I_{|S|}) \operatorname{vec}(U).$$

For brevity, it may be written

$$(\widehat{\beta}_S, \widehat{u}_S) = \underset{\beta, u}{\operatorname{argmin}} \|u\|^2 \quad \text{subject to} \quad y_S = \left(\bigoplus_{i=1}^{G} X_S^{(i)} \right) \beta + (C \otimes I_{|S|})u, \tag{5.13}$$

where $y_S = \operatorname{vec}(Y_S)$, $\beta \in \mathbb{R}^N$ and $u \in \mathbb{R}^{M_S}$, $M_S = G|S|$ and $N = \sum_i n_i$.

Consider the GQRD of $\bigoplus X_S^{(i)}$ and $C \otimes I_{|S|}$:

$$Q_S^{\mathrm{T}} \bigoplus_{i=1}^{G} X_S^{(i)} = \begin{bmatrix} \bigoplus R_S^{(i)} \\ 0 \end{bmatrix} \quad \text{and} \quad Q_S^{\mathrm{T}} (C \otimes I_{|S|}) P_S^{\mathrm{T}} = \begin{bmatrix} T_S & T_{12} \\ 0 & T_{22} \end{bmatrix},$$

where $R_S^{(i)} \in \mathbb{R}^{n_i \times n_i}$ and $T_{22} \in \mathbb{R}^{(M_S - N) \times (M_S - N)}$ are upper triangular, and $T_S \in \mathbb{R}^{N \times N}$ is block upper triangular. The transformations $Q_S \in \mathbb{R}^{M_S \times M_S}$ and $P_S \in \mathbb{R}^{M_S \times M_S}$ are orthogonal. Here, $Q_S = \begin{bmatrix} \bigoplus Q_1^{(i)} & \bigoplus Q_2^{(i)} \end{bmatrix}$, where

$$Q_S^{(i)^{\mathrm{T}}} X_S^{(i)} = \begin{bmatrix} R_S^{(i)} \\ 0 \end{bmatrix} \tag{5.14}$$

is the QRD of $X_S^{(i)}$ with $Q_S^{(i)} = \begin{bmatrix} Q_1^{(i)} & Q_2^{(i)} \end{bmatrix}$ such that

$$Q_1^{(i)^{\mathrm{T}}} X_S^{(i)} = R_S^{(i)} \quad \text{and} \quad Q_2^{(i)^{\mathrm{T}}} X_S^{(i)} = 0.$$

The subset SUR-GLLSP in (5.13) is equivalent to

$$(\widehat{\beta}_S, \widehat{u}_1, \widehat{u}_2) = \underset{\beta, u_1, u_2}{\operatorname{argmin}} (\|u_1\|^2 + \|u_2\|^2) \quad \text{subject to} \quad \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \bigoplus R_S^{(i)} \\ 0 \end{bmatrix} \beta + \begin{bmatrix} T_S & T_{12} \\ 0 & T_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{5.15}$$

where

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \bigoplus Q_1^{(i)^{\mathrm{T}}} \\ \bigoplus Q_2^{(i)^{\mathrm{T}}} \end{bmatrix} y_S.$$

Thus, $\widehat{u}_2 = T_{22}^{-1} y_2$, and the reduced SUR-GLLSP is

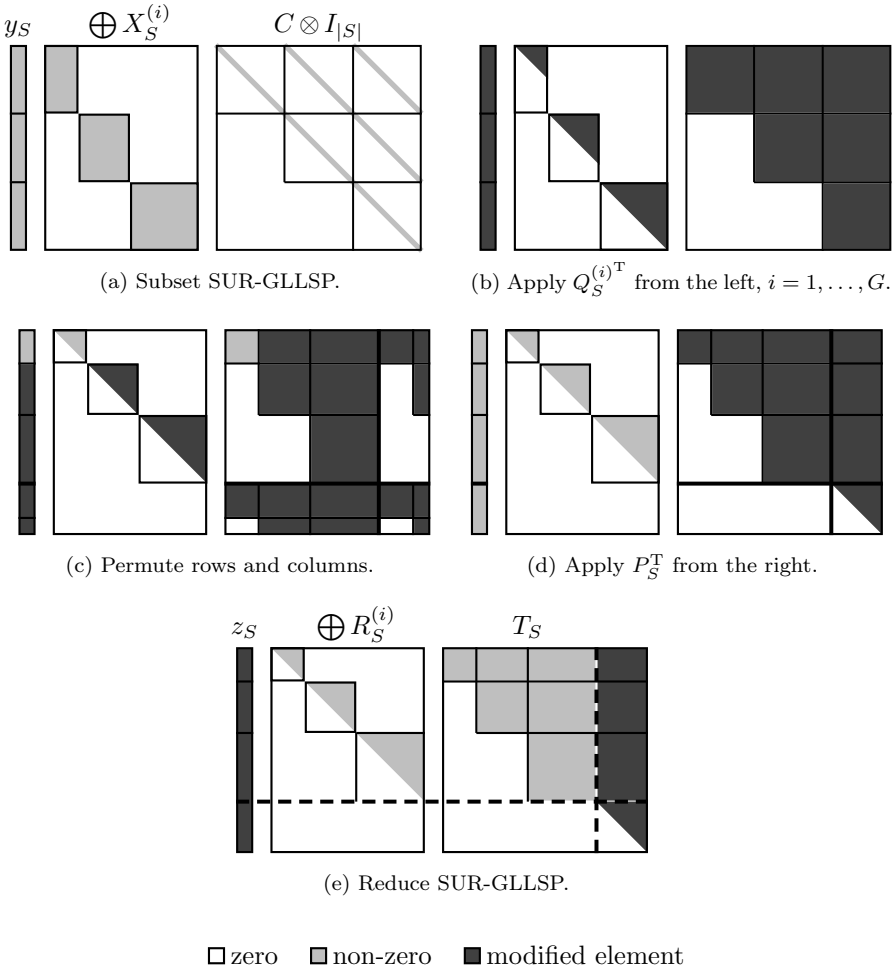
$$(\widehat{\beta}_S, \widehat{u}_1) = \underset{\beta, u_1}{\operatorname{argmin}} \|u_1\|^2 \quad \text{subject to} \quad z_S = \left(\bigoplus_{i=1}^{G} R_S^{(i)} \right) \beta + T_S u_1, \tag{5.16}$$

where $z_S = y_1 - T_{12}\widehat{u}_2$. It follows that $\widehat{u}_1 = 0$, and the estimator of the SUR model is given by

$$\left(\bigoplus_{i=1}^{G} R_S^{(i)} \right) \widehat{\beta}_S = z_S,$$

which is equivalent to solving the G triangular systems $R_S^{(i)} \widehat{\beta}_S^{(i)} = z_S^{(i)}$, where $\widehat{\beta}_S = \operatorname{vec}(\{\widehat{\beta}_S^{(i)}\})$ and $z_S = \operatorname{vec}(\{z_S^{(i)}\})$ are suitably partitioned. The value of the objective function is $\operatorname{RSS}(\widehat{\beta}_S) = \|\widehat{u}_2\|^2$.

The procedure to solve the subset SUR-GLLSP when $|S| = n_G$ is illustrated in Figure 5.6. First, the G orthogonal transformations ${Q_S^{(i)}}^{\mathrm{T}}$ defined in (5.14) are applied from the left to form the upper triangular $R_S^{(i)}$ (see Figure 5.6b). Next, the rows are permuted to form $\bigoplus R_S^{(i)}$. The same permutation is applied from the right to the columns of the transformed $C \otimes I_{|S|}$ (see Figure 5.6c). Then, the orthogonal transformation P_S^{T} is applied from the right to obtain the block upper triangular T_S in (5.15) (see Figure 5.6d). Finally, the GLLSP is reduced by a block column according to (5.16) (see Figure 5.6e). Arranging the G regression equations in order of increasing number of variables facilitates the permutation of the rows and columns, which can be performed in place. A remarkable difference with the GLM is that the observations that are available for selection — i.e. the compound observations in A — are not affected by the orthogonal transformation P_S^{T}.



(a) Subset SUR-GLLSP.

(b) Apply ${Q_S^{(i)}}^{\mathrm{T}}$ from the left, $i = 1, \ldots, G$.

(c) Permute rows and columns.

(d) Apply P_S^{T} from the right.

(e) Reduce SUR-GLLSP.

Figure 5.6: Solving the subset SUR-GLLSP ($|S| = n_G$, $G = 3$).

Updating the subset SUR model S given in (5.12) with a compound observation is equivalent to adding an observation to every regression equation. The new model is written

$$\begin{bmatrix} y_S^{(i)} \\ \omega^{(i)} \end{bmatrix} = \begin{bmatrix} X_S^{(i)} \\ x^{(i)^{\mathrm{T}}} \end{bmatrix} \beta^{(i)} + \begin{bmatrix} \varepsilon_S^{(i)} \\ \varphi^{(i)} \end{bmatrix}, \quad i = 1, \ldots, G.$$

The G observations are taken from the original model in (5.3); thus, $\mathrm{vec}(\{\varphi^{(i)}\}) \sim (0, \Sigma)$. The solution is found by updating the reduced SUR-GLLSP in (5.16):

$$(\widehat{\beta}_{\widetilde{S}}, \widetilde{u}_1, \widetilde{u}_2) = \underset{\beta, u_1, u_2}{\operatorname{argmin}} (\|u_1\|^2 + \|u_2\|^2) \quad \text{subject to} \quad \begin{bmatrix} z_S \\ w \end{bmatrix} = \begin{bmatrix} \bigoplus R_S^{(i)} \\ \bigoplus x^{(i)^{\mathrm{T}}} \end{bmatrix} \beta + \begin{bmatrix} T_S & 0 \\ 0 & C \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{5.17}$$

where $w = \mathrm{vec}(\{\omega^{(i)}\})$. The GQRD of $\bigoplus R_S^{(i)}$ and T_S is updated by a sequence of Givens rotations such that

$$\widetilde{Q}^{\mathrm{T}} \begin{bmatrix} \bigoplus R_S^{(i)} \\ \bigoplus x^{(i)^{\mathrm{T}}} \end{bmatrix} = \begin{bmatrix} \bigoplus R_{\widetilde{S}}^{(i)} \\ 0 \end{bmatrix} \quad \text{and} \quad \widetilde{Q}^{\mathrm{T}} \begin{bmatrix} T_S & 0 \\ 0 & C \end{bmatrix} \widetilde{P}^{\mathrm{T}} = \begin{bmatrix} T_{\widetilde{S}} & \widetilde{T}_{12} \\ 0 & \widetilde{T}_{22} \end{bmatrix}, \tag{5.18}$$

and the SUR-GLLSP in (5.17) is equivalent to

$$(\widehat{\beta}_{\widetilde{S}}, \widetilde{v}_1, \widetilde{v}_2) = \underset{\beta, v_1, v_2}{\operatorname{argmin}} (\|v_1\|^2 + \|v_2\|^2) \quad \text{subject to} \quad \begin{bmatrix} \widetilde{y}_1 \\ \widetilde{w} \end{bmatrix} = \begin{bmatrix} \bigoplus R_{\widetilde{S}}^{(i)} \\ 0 \end{bmatrix} \beta + \begin{bmatrix} T_{\widetilde{S}} & \widetilde{T}_{12} \\ 0 & \widetilde{T}_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \tag{5.19}$$

where

$$\begin{bmatrix} \widetilde{y}_1 \\ \widetilde{w} \end{bmatrix} = \widetilde{Q}^{\mathrm{T}} \begin{bmatrix} z_S \\ w \end{bmatrix}.$$

Thus, $\widetilde{v}_2 = \widetilde{T}_{22}^{-1} \widetilde{w}$, and the reduced SUR-GLLSP is given by

$$(\widehat{\beta}_{\widetilde{S}}, \widetilde{v}_1) = \underset{\beta, v_1}{\operatorname{argmin}} \|v_1\|^2 \quad \text{subject to} \quad z_{\widetilde{S}} = \bigoplus_{i=1}^{G} R_{\widetilde{S}}^{(i)} \beta + T_{\widetilde{S}} v_1, \tag{5.20}$$

where $z_{\widetilde{S}} = \widetilde{y}_1 - \widetilde{T}_{12} \widetilde{v}_2$. It is satisfied by $\widetilde{v}_1 = 0$ and $\widehat{\beta}_{\widetilde{S}} = \left(\bigoplus R_{\widetilde{S}}^{(i)} \right)^{-1} z_{\widetilde{S}}$, the updated value of the objective function being $\mathrm{RSS}(\widehat{\beta}_{\widetilde{S}}) = \mathrm{RSS}(\widehat{\beta}_S) + \|\widetilde{v}_2\|^2$.

The various steps to update the subset SUR-GLLSP are illustrated schematically in Figure 5.7. First, the new data is appended according to (5.17) (see Figure 5.7a). Next, the G equations are updated by applying the orthogonal transformation $\widetilde{Q}^{\mathrm{T}}$ defined in (5.18) from the left (see Figure 5.7b). Then, the orthogonal $\widetilde{P}^{\mathrm{T}}$ is applied from the right to restore the block upper triangular structure of $T_{\widetilde{S}}$ in (5.19) (see Figure 5.7c). Finally, the model is reduced according to (5.20) (see Figure 5.7d). Givens sequences are employed to update the matrices (Foschi, Belsley & Kontoghiorghes 2003, Kontoghiorghes 2004). As before, the available observations A are not affected by the orthogonal transformation $\widetilde{P}^{\mathrm{T}}$.

Let $T(G, N, p)$ denote the estimated cost of deriving all nodes from a given node (S, A), where $N = \sum_i n_i$, and $p = |A|$ is the number of available compound observations. It is given in terms of elementary rotations and can be written

$$\begin{aligned} T(G, N, p) &= \sum_{i=1}^{p} \big(T_{\mathrm{u}}(G, N) + T(G, N, p - i) \big) \\ &= (2^p - 1)\, T_{\mathrm{u}}(G, N), \end{aligned}$$

where $T_{\mathrm{u}}(G, N) \in O(GN^2)$ is the estimated cost of updating the subset SUR-GLLSP. A rough, conservative estimate of the complexity of the ARA to compute all LTS estimators of the balanced SUR model is $T(G, N, m) \in O(2^m GN^2)$.

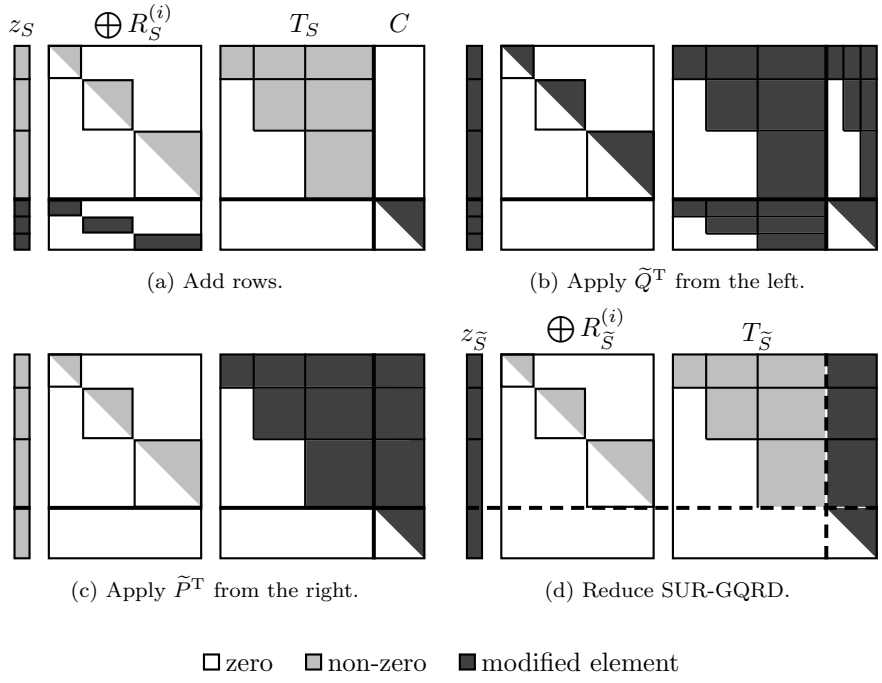


Figure 5.7: Updating the SUR-GLLSP with a compound observation ($G = 3$).

5.4.1 Experimental results

SUR models with $G = 3$ regression equations and m compound observations are simulated according to

$$\mathrm{vec}(\{y^{(i)}\}_3) = \left(\bigoplus_{i=1}^{3} X^{(i)}\right)\beta + (C \otimes I_m)u,$$

where $X^{(i)} \sim (0, 10)$ and $u \sim (0, 1)$. C is the upper triangular Cholesky factor of Σ, which is generated like Ω in (5.11). Outlying observations are injected by replacing the y-values of some compound observations by values that are normally distributed with mean 100 and standard deviation 10. Five sample datasets are generated for each problem size m. The mean execution time is reported in Table 5.2. The number of (compound) outliers is given by m_{out}. The algorithm is computationally infeasible for large-scale SUR models. However, the ARA correctly revealed the outlying data points in all sample datasets. One example is illustrated in Figure 5.8.

Table 5.2: Mean execution time (in seconds) for the simulated SUR model.

m	n_i ($G = 3$)	m_{out}	time (s)
16	4,6,8	4	12
20	4,6,8	5	96
24	4,6,8	6	537
28	6,8,10	7	10561
32	6,8,10	8	46120

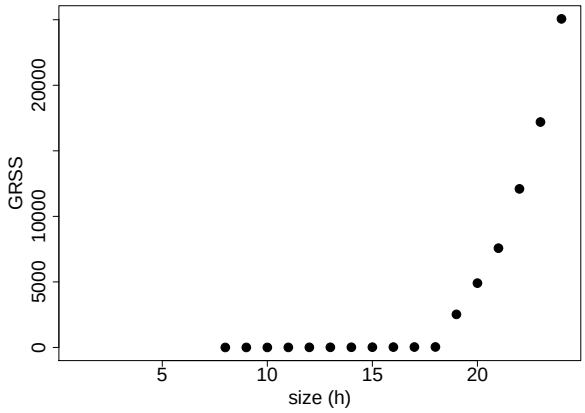


Figure 5.8: The RSS computed by the ARA for a simulated SUR model with $m = 24$ observations and $m_{\text{out}} = 6$ outliers.

5.5 Conclusions

New strategies have been designed to compute the generalized LTS (GLTS) estimators of the GLM and SUR model. The adding row algorithm (ARA) is based on an algorithm for solving the LTS problem of the standard regression model. It computes all observation subsets to find the best GLTS estimators for a range of coverage values. The observation subsets are organized by the all-subsets tree. The singularity problem of the dispersion matrix is avoided by reformulating the estimation problem as a generalized linear least squares problem (GLLSP). The main computational tool is the generalized QR decomposition (GQRD). Efficient strategies that apply Givens rotations to update the GQRD are employed. The orthogonal transformations are applied efficiently by exploiting the sparse matrix structure. Thus, the GLTS estimators are obtained cheaply and without having to solve a GLLSP from scratch. Furthermore, the generalized residual sum of squares (RSS) is non-decreasing when an observation is added to a subset model. Thus, the ARA can employ a branch and bound strategy, where the bound of each node in the regression tree is given by the RSS.

Although the ARA cannot compute the GLTS estimates for larger problem sizes, simulations show that it succeeds in detecting outlying observations. Future work can address the GLTS problem of the unbalanced SUR model. The G regression equations of the unbalanced SUR model can consist of unequal numbers of observations (Foschi & Kontoghiorghes 2002). Thus, the combinatorial complexity of the ARA rises from 2^m for the SUR model to 2^{Gm} for the unbalanced SUR model. Therefore, feasability studies of adapting the Fast LTS algorithm (Rousseeuw & Van Driessen 2006) to the GLM, with further research aiming at the use of parallel computing to solve problems of larger scale, could show very promising results.

Chapter 6

Conclusions

Given the importance of the QR decomposition (QRD) in the field of least squares (LS) regression, the design of parallel algorithms as a computationally efficient method to solve the QRD is investigated in Chapter 2. Special interest is paid to the solution of the QRD on an exclusive read, exclusive write (EREW) parallel random access machine (PRAM). In this context, a new pipeline-parallel strategy that employs orthogonal Givens rotations to compute the QRD is proposed (Hofmann & Kontoghiorghes 2006). A theoretical analysis of the number of required floating point operations shows that the new method is twice as efficient as the well-known SK scheme (Sameh & Kuck 1978).

It is well-known that the problem of deriving all variable subset models is computationally very demanding. Therefore, new variable selection algorithms are investigated in Chapter 3 to tackle problem sizes hitherto out of reach (Hofmann et al. 2007). The new algorithms are based on the already existing dropping column algorithm (DCA) (Gatu & Kontoghiorghes 2003, Smith & Bremner 1989). These employ the QRD for computing the LS estimators corresponding to all variable subsets in an efficient manner. Regression trees are employed to organize the search space consisting of all possible variable subsets. The complexity – i.e. the number of nodes – of tree based algorithms is combinatorial. In an effort to reduce the number of nodes generated by the ordinary DCA, a new algorithm that takes advantage of the structural properties of the regression tree is developed: the Subrange DCA presented here derives the subset models for a specified range of subset sizes and does not generate the entire regression tree. Thus, the Subrange DCA is more efficient than the ordinary DCA in cases where not all model sizes are of interest. By inspecting a limited number of subset sizes, experiments confirm that the Subrange DCA can tackle problem sizes that the ordinary DCA was not able to handle.

The efficiency of tree based methods for subset model selection is enhanced by employing a branch and bound strategy to avoid the explicit enumeration of all variable subsets (Gatu & Kontoghiorghes 2006). The number of subsets evaluated by the branch and bound algorithm (BBA) decreases by preordering the variables. The variables are sorted according to their strength, which is determined by downdating a QR decomposition after deletion of a column. Doing so incurs a significant computational overhead, and variable preordering cannot be applied in all nodes of the regression tree. Preordering of the variables must be applied selectively in nodes where the potential number of nodes that will be discarded by the BBA is large. To this end, a node radius is defined in Chapter 3, which can be seen as a measure of the distance between a node and the root node of the regression tree. The BBA with preordering (PBBA) preorders the variables

in all nodes that lie within a given node radius, the so-called preordering radius. Furthermore, it employs an efficient procedure to compute the strength of the variables, which avoids the explicit retriangularization of the QRD. Test runs have revealed that the PBBA is most efficient when the preordering radius is between one fourth and one third of the total number of variables. By combining the PBBA with the Subrange DCA, models with up to 80 variables can be tackled in a reasonable amount of time.

To achieve greater execution speed, former research work (Gatu & Kontoghiorghes 2006) has introduced the heuristic BBA (HBBA), somewhat sacrificing solution quality. There, the loss in solution quality is determined by a tolerance parameter. In order to improve this heuristic approach, two extensions of the HBBA are presented in Chapter 3: The Level HBBA uses different values for the tolerance parameter on different levels of the regression tree. It generates fewer nodes than the ordinary HBBA when used with the same mean tolerance. Remarkably, although the subset models computed by the Level HBBA are of lower quality than those derived by the ordinary HBBA, it has been observed that the relative residual error is far below the mean tolerance. On the other hand, the second, more important extension – the Size HBBA – assigns different values to the tolerance parameter for different subset sizes. The Size HBBA improves the quality of the models computed by the ordinary HBBA. This means that for the same computational effort, the Size HBBA produces submodels closer to the best possible solution than the ordinary HBBA. In fact, the Size HBBA represents a generalization of the DCA, Subrange DCA, BBA and HBBA, and can be combined with the PBBA. This makes the Size HBBA a powerful and flexible tool for computing subset models. Most notably, it generalizes the Subrange DCA by rendering possible the investigation of non-contiguous submodel sizes.

The regression tree methods proposed above can be adapted to solve the least trimmed squares (LTS) regression problem (Hofmann et al. 2009). The adding row algorithm (ARA) developed for this purpose in Chapter 4 is an observation subset selection algorithm; in each node of the regression tree, a linear estimator is derived by updating a QR decomposition with a row (observation). The structure of the all-subsets tree is exploited to determine the LTS regressors for a range of coverage values. Given the fact that the exact value of the coverage parameter h does not need to be known in advance, the ARA can be used to efficiently examine the degree of data contamination. Thereupon, a branch and bound algorithm (LTS BBA) that avoids the explicit enumeration of all observation subsets is designed, considerably reducing the execution time. In this context, the effect of observation preordering on the execution time of the LTS BBA is investigated, and an efficient procedure to compute the numerical bounds is designed. Experimental trials show that the LTS BBA with observation preordering (LTS PBBA) is significantly faster than the LTS BBA.

Typically, state of the art methods to compute the exact LTS regression derive one estimator at a time for a given value of the coverage parameter. Therefore, knowledge about the coverage value is required beforehand. If no such knowledge is available, one usually chooses the value $h = m/2$ for the coverage parameter, where m is the number of observations. In most cases, the degree of data contamination is less than 0.5. As a consequence, conventional procedures might discard relevant data and might fail to reveal the global linear trend. In comparison, methods based on the LTS ARA derive LTS estimators for a range of coverage values in a more efficient manner, and therefore do not require prior knowledge of the coverage value. In practice, the LTS PBBA developed in Chapter 4 is 6 to 10 times faster for computing a range of LTS estimators than conventional methods. That makes the LTS PBBA an efficient tool to examine the degree of data contamination and to determine the coverage value that is both robust and statistically efficient.

The promising LTS BBA is extended in Chapter 5 to compute the generalized LTS (GLTS) estimators of the general linear model (GLM) (Hofmann & Kontoghiorghes 2009). Here, the seemingly unrelated regressions (SUR) model stands as a special case of the GLM. The singularity problem of the dispersion matrix is avoided by reformulating the GLM as a generalized linear least squares problem (GLLSP). The generalized QR decomposition serves as the main computational tool to solve the GLLSP. Efficient Givens strategies, which take advantage of the sparse matrix structure, are employed to update the orthogonal factorizations. Thus, the GLTS estimators can be obtained at a reasonable cost and without having to solve a GLLSP from scratch. The GLTS BBA is applicable for models with up to about 30 observations. Experimental simulations have confirmed the ability of this algorithm to detect outlying observations.

Research outlook

Future research can address the design ob block versions of the SK scheme to compute the orthogonal factorizations of structured matrices (Kontoghiorghes 2000*a*, Yanev et al. 2004). Within this context, the Givens rotations are replaced by orthogonal factorizations which employ Householder reflections. Thus, it might be fruitful to investigate the effectiveness of incorporating the pipeline strategy in the design of block algorithms (Kontoghiorghes 2000*c*, Yanev & Kontoghiorghes 2004).

Furthermore, the employment by the PBBA of computationally less expensive criteria to evaluate the variable strength could be considered, as well as parallel strategies to downdate the QR decomposition after deletion of a variable. One could also explore the possibility of designing a dynamic HBBA, which would employ a learning strategy to automatically determine the value of the tolerance parameter in any given node. A parallelization of the BBA, employing a task farming strategy on heterogeneous parallel systems, could be looked into.

The LTS problem of the SUR model where the regressions are modified independently merits special attention. That is, observations are deleted from one regression at a time, yielding an unbalanced SUR model in each node of the ARA tree (Foschi & Kontoghiorghes 2002). The supposed feasibility of adapting the Fast LTS algorithm (Rousseeuw & Van Driessen 2006) to the GLM, together with the use of parallel computing in order to solve larger-scale problems, could open promising opportunities. The possibility of adapting the Fast LTS algorithm to subset model selection can be considered. Specifically, a new C-step (concentration step) needs to be defined. Given a subset model S_1, the C-step would compute a subset model S_2 of the same size in constant time such that $\text{RSS}(S_2) \leq \text{RSS}(S_1)$. For a given subset size j, the algorithm would construct a converging sequence of subsets $\text{RSS}(S_1) \geq \text{RSS}(S_2) \geq \ldots \geq \text{RSS}(S_k) = \text{RSS}(S_{k+1})$ by repeating C-steps, where k is a finite number and $|S_i| = j$ $(1 \leq i \leq k+1)$. It is hoped that by choosing a sufficiently large number of initial subsets S_1 and keeping the subsets with the lowest RSS a good, if not optimal, solution would be found.

Finally, a parallelization of the regression tree in the context of LTS regression constitutes another attractive problem. An LTS algorithm that proceeds by downdating the estimator could be an interesting alternative to the existing LTS ARA presented here. The design of numerical methods to downdate structured matrix problems, such as the GLM and SUR model, with an observation remains an open challenge. The effects on the regression coefficients of deleting an observation need further thorough investigation to support the detection of influential data.

Abbreviations

AGLA Agulló's (2001) algorithm

ARA adding row algorithm

BBA branch and bound algorithm

BBA-1 branch and bound algorithm with preordering in the root node

BF brute force

BLUE best linear unbiased estimator

CDGR compound disjoint Givens rotation

DCA dropping column algorithm

EREW exclusive read/exclusive write

FSA feasible solution algorithm

GLM general linear model

GLLSP generalized linear least squares problem

GLS generalized least squares

GLTS generalized least trimmed squares

GQRD generalized QR decomposition

GR Givens rotation

HBBA heuristic branch and bound algorithm

LBA leaps and bounds algorithm

LBA-1 leaps and bounds algorithm with preordering in the root node

LIFO last in, first out

LMS least median of squares

LS least squares

LTS least trimmed squares

OLM ordinary linear model

OLS ordinary least squares

PBBA preordering branch and bound algorithm

PGS parallel Givens sequence

PipPGS pipeline-parallel Givens sequence

PipSK pipeline-parallel Sameh and Kuck

PRAM parallel random access machine

QRD QR decomposition

Resid absolute residuals

RRE relative residual error

RSS residual sum of squares

SK Sameh and Kuck

SUR seemingly unrelated regressions

Bibliography

Agulló, J. (2001), 'New algorithms for computing the least trimmed squares regression estimator', *Computational Statistics and Data Analysis* **36**, 425–439.

Agulló, J., Croux, C. & Van Aelst, S. (2008), 'The multivariate least-trimmed squares estimator', *Journal of Multivariate Analysis* **99**(3), 311–338.

Alexander, S. T., Pan, C.-T. & Plemmons, R. J. (1988), 'Analysis of a recursive least squares hyperbolic rotation algorithm for signal processing', *Linear Algebra and its Applications* **98**, 3–40.

Atkinson, A. C. & Cheng, T.-C. (1999), 'Computing least trimmed squares regression with the forward search', *Statistics and Computing* **9**, 251–263.

Belsley, D. A., Kuh, A. E. & Welsch, R. E. (1980), *Regression diagnostics: identifying influential data and sources of collinearity*, John Wiley and Sons, New York.

Björck, Å. (1984), *A general updating algorithm for constrained linear least squares problems*, Vol. 5, SIAM Journal on Scientific and Statistical Computing.

Björck, Å. (1996), *Numerical methods for least squares problems*, Society for Industrial and Applied Mathematics.

Björck, Å., Park, H. & Eldén, L. (1994), 'Accurate downdating of least squares solutions', *SIAM Journal on Matrix Analysis and Applications* **15**(2), 549–568.

Bojanczyk, A., Brent, R. & Kung, H. (1984), 'Numerically stable solution of dense systems of linear equations using mesh-connected processors', *SIAM Journal on Scientific and Statistical Computing* **5**, 95–104.

Breiman, L. (1995), 'Better subset regression using the nonnegative Garrote', *Technometrics* **37**(4), 373–384.

Burks, A. W., Warren, D. W. & Wright, J. B. (1954), 'An analysis of a logical machine using parenthesis-free notation', *Mathematical Tables and Other Aids to Computation* **8**(46), 53–57.

Businger, P. & Golub, G. H. (1965), 'Linear least squares solutions by Householder transformations', *Numerische Mathematik* **7**, 269–276.

Clarke, M. R. B. (1981), 'Statistical algorithms: algorithm AS 163: a Givens algorithm for moving from one linear model to another without going back to the data', *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **30**(2), 198–203.

Cosnard, M. & Daoudi, M. (1994), 'Optimal algorithms for parallel Givens factorization on a coarse–grained PRAM', *Journal of the ACM* **41**(2), 399–421.

Cosnard, M., Muller, J.-M. & Robert, Y. (1986), 'Parallel QR decomposition of a rectangular matrix', *Numerische Mathematik* **48**, 239–249.

Cosnard, M. & Robert, Y. (1983), Complexité de la factorisation QR en parallèle, *in* 'Comptes rendus des sances de l'Acadmie des sciences', Vol. 297 of *Série 1, mathématiques*, Gauthier-Villars, Paris, France, pp. 137–139.

Dongarra, J. J., Bunch, J. R., Moler, C. B. & Stewart, G. W. (1979), *LINPACK users' guide*, SIAM, Philadelphia.

Donoho, D. L. & Huber, P. J. (1983), The notion of breakdown point, *in* P. J. Bickel, K. A. Doksum, E. L. Lehman & J. L. Hodges, eds, 'A Festschrift for Erich L. Lehmann in honor of his sixty-fifth birthday', CRC Press.

Fan, J. & Li, R. (2001), 'Variable selection via nonconcave penalized likelihood and its oracle properties', *Journal of the American Statistical Association* **96**(456), 1348–1360.

Fausett, D. W., Fulton, C. T. & Hashish, H. (1997), 'Improved parallel QR method for large least squares problems involving Kronecker products', *Journal of Computational and Applied Mathematics* **78**, 63–78.

Fortune, S. & Wyllie, J. (1978), Parallelism in random access machines, *in* 'Proceedings of the 10th annual ACM symposium on theory of computing', pp. 114–118.

Foschi, P., Belsley, D. A. & Kontoghiorghes, E. J. (2003), 'A comparative study of algorithms for solving seemingly unrelated regressions models', *Computational Statistics and Data Analysis* **44**(1-2), 3–35.

Foschi, P. & Kontoghiorghes, E. J. (2002), 'Seemingly unrelated regression model with unequal size observations: computational aspects', *Computational Statistics and Data Analysis* **41**(1), 211–229.

Furnival, G. & Wilson, R. (1974), 'Regression by leaps and bounds', *Technometrics* **16**, 499–511.

Gatu, C. & Kontoghiorghes, E. J. (2003), 'Parallel algorithms for computing all possible subset regression models using the QR decomposition', *Parallel Computing* **29**(4), 505–521.

Gatu, C. & Kontoghiorghes, E. J. (2005), 'Efficient strategies for deriving the subset VAR models', *Computational Management Science* **2**, 253–278.

Gatu, C. & Kontoghiorghes, E. J. (2006), 'Branch-and-bound algorithms for computing the best subset regression models', *Journal of Computational and Graphical Statistics* **15**, 139–156.

Gatu, C., Yanev, P. & Kontoghiorghes, E. J. (2007), 'A graph approach to generate all possible regression submodels', *Computational Statistics and Data Analysis* . Forthcoming.

Gill, P. E., Golub, G. H., Murray, W. & Saunders, M. A. (1974), 'Methods for modifying matrix factorizations', *Mathematics of Computations* **28**, 505–535.

Golub, G. H. & Van Loan, C. F. (1996), *Matrix computations*, 3rd edn, Johns Hopkins University Press, Baltimore, Maryland.

Golub, G. H. & Wilkinson, J. H. (1966), 'Note on the iterative refinement of least squares solutions', *Numerische Mathematik* **9**, 139–148.

Gulliksson, M. & Wedin, P.-Å. (1992), 'Modifying the QR decomposition to constrained and weighted linear least squares', *SIAM Journal on Matrix Analysis and Applications* **13**(4), 1298–1313.

Hastie, T., Tibshirani, R. & Friedman, J. (2001), *The elements of statistical learning*, Springer series in statistics, Springer-Verlag, New York.

Hawkins, D. M. (1994), 'The feasible solution algorithm for least trimmed squares regression', *Computational Statistics and Data Analysis* **17**, 185–196.

Hawkins, D. M. & Olive, D. J. (1999), 'Improved feasible solution algorithms for high breakdown estimation', *Computational Statistics and Data Analysis* **30**, 1–11.

Hocking, R. R. (1976), 'The analysis and selection of variables in linear regression', *Biometrics* **32**, 1–49.

Hofmann, M., Gatu, C. & Kontoghiorghes, E. J. (2007), 'Efficient algorithms for computing the best-subset regression models for large-scale problems', *Computational Statistics and Data Analysis* **52**, 16–29.

Hofmann, M., Gatu, C. & Kontoghiorghes, E. J. (2009), 'An exact least-trimmed-squares algorithm for a range of coverage values', *Journal of Computational and Graphical Statistics* . In press.

Hofmann, M. & Kontoghiorghes, E. J. (2006), 'Pipeline Givens sequences for computing the QR decomposition on a EREW PRAM', *Parallel Computing* **32**(3).

Hofmann, M. & Kontoghiorghes, E. J. (2009), 'Matrix strategies for computing the least trimmed squares estimation of the general linear and SUR models', *Computational Statistics and Data Analysis* . Under revision.

Hössjer, O. (1994), 'Rank-based estimates in the linear model with high breakdown point', *Journal of the American Statistical Association* **89**, 149–158.

Hubert, M., Rousseeuw, P. J. & Van Aelst, S. (2008), 'High-breakdown robust multivariate methods', *Statistical Science* **23**(1), 92–119.

Karasalo, I. (1974), 'A criterion for truncation of the QR decomposition algorithm for the singular linear least squares problem', *Bit* **14**, 156–166.

Kmenta, J. & Gilbert, R. F. (1968), 'Small sample properties of alternative estimators of seemingly unrelated regressions', *Journal of the American Statistical Association* **63**, 1180–1200.

Kontoghiorghes, E. J. (2000*a*), *Parallel algorithms for linear models: numerical methods and estimation problems*, Vol. 15 of *Advances in computational economics*, Kluwer Academic Publishers, Boston.

Kontoghiorghes, E. J. (2000*b*), 'Parallel givens sequences for solving the general linear model on a EREW PRAM', *Parallel Algorithms and Applications* **15**(1–2), 57–75.

Kontoghiorghes, E. J. (2000*c*), 'Parallel strategies for rank-k updating of the QR decomposition', *SIAM Journal on Matrix Analysis and Applications* **22**(3), 714–725.

Kontoghiorghes, E. J. (2002), 'Greedy Givens algorithms for computing the rank-k updating of the QR decomposition', *Parallel Computing* **28**(9), 1257–1273.

Kontoghiorghes, E. J. (2004), 'Computational methods for modifying seemingly unrelated regressions models', *Journal of Computational and Applied Mathematics* **162**(1), 247–261.

Kontoghiorghes, E. J. & Clarke, M. R. B. (1993), 'Solving the updated and downdated ordinary linear model on massively parallel SIMD systems', *Parallel Algorithms and Applications* **1**(2), 369–377.

Kontoghiorghes, E. J. & Clarke, M. R. B. (1995), 'An alternative approach for the numerical solution of seemingly unrelated regression equation models', *Computational Statistics and Data Analysis* **19**(4), 369–377.

Kontoghiorghes, E. J. & Dinenis, E. (1996), Solving triangular seemingly unrelated regression equation models on massively parallel systems, *in* M. Gilli, ed., 'Computational economic systems: models, methods and econometrics', Vol. 5 of *Advances in computational economics*, Kluwer academic publishers.

Kourouklis, S. & Paige, C. C. (1981), 'A constrained least squares approach to the general Gauss-Markov linear model', *Journal of the American Statistical Association* **76**, 620–625.

Lawson, C. L. & Hanson, R. J. (1974), *Solving least squares problems*, Prentice-Hall.

Luk, F. T. (1986), 'A rotation method for computing the QR decomposition', *SIAM Journal on Scientific and Statistical Computing* **7**(2), 452–459.

Miller, A. J. (2002), *Subset selection in regression*, Vol. 95 of *Monographs on Statistics and Applied Probability*, 2nd edn, Chapman and Hall.

Modi, J. J. (1988), *Parallel algorithms and matrix computation*, Oxford Applied Mathematics and Computing Science Series, Oxford University Press.

Modi, J. J. & Clarke, M. R. B. (1984), 'An alternative Givens ordering', *Numerische Mathematik* **43**, 83–90.

Morgenthaler, S. (1991), 'A note on efficient regression estimators with positive breakdown point', *Statistics and Probability Letters* **11**, 469–472.

Narendra, P. M. & Fukunaga, K. (1977), 'A branch and bound algorithm for feature subset selection', *IEEE Transactions on Computers* **26**(9), 917–922.

Newell, A. & Shaw, J. C. (1957), Programming the logic theory machine, *in* 'Proceedings of the Western Joint Computer Conference', Institute of Radio Engineers, pp. 230–240.

Paige, C. C. (1978), 'Numerically stable computations for general univariate linear models', *Communications in Statistics Part B — Simulation and Computation* **7**(5), 437–453.

Paige, C. C. (1979*a*), 'Computer solution and perturbation analysis of generalized linear least squares problems', *Mathematics of Computation* **33**(145), 171–183.

Paige, C. C. (1979*b*), 'Fast numerically stable computations for generalized linear least squares problems', *SIAM Journal on Numerical Analysis* **16**(1), 165–171.

R Development Core Team (2005), *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *http://www.r-project.org*

Rao, C. R. & Toutenburg, H. (1995), *Linear models: least squares and alternatives*, Springer series in statistics, Springer.

Regalia, P. A. & Mitra, S. K. (1989), 'Kronecker products, unitary matrices and signal processing applications', *SIAM Review* **31**(4), 586–613.

Revankar, N. S. (1974), 'Some finite samples results in the context of two seemingly unrelated regression equations', *Journal of the American Statistical Association* **69**, 187–190.

Roberts, S. J. (1984), 'Statistical algorithms: algorithm AS 199: A branch and bound algorithm for determining the optimal feature subset of given size', *Applied Statistics* **33**(2), 236–241.

Rousseeuw, P. J. (1984), 'Least median of squares regression', *Journal of the American Statistical Association* **79**, 871–880.

Rousseeuw, P. J. (1997), Introduction to positive-breakdown methods, *in* G. S. Maddala & C. R. Rao, eds, 'Handbook of statistics', Vol. 15: robust inference, Elsevier, pp. 101–121.

Rousseeuw, P. J. & Leroy, A. M. (1987), *Robust regression and outlier detection*, John Wiley and Sons, New York.

Rousseeuw, P. J., Van Aelst, S., Van Driessen, K. & Agulló, J. (2004), 'Robust multivariate regression', *Technometrics* **46**, 293–305.

Rousseeuw, P. J. & Van Driessen, K. (2006), 'Computing LTS regression for large data sets', *Data Mining and Knowledge Discovery* **12**, 29–45.

Sameh, A. H. (1985), Solving the linear least squares problem on a linear array of processors, *in* 'Algorithmically Specialized Parallel Computers', Academic Press, Inc., pp. 191–200.

Sameh, A. H. & Brent, R. P. (1971), 'On Jacobi and Jacobi like algorithms for a parallel computer', *Mathematics of Computation* **25**(115), 579–590.

Sameh, A. H. & Kuck, D. J. (1978), 'On stable parallel linear system solvers', *Journal of the ACM* **25**(1), 81–91.

Schmidt, P. (1978), 'A note on the estimation of seemingly unrelated regression systems', *Journal of Econometrics* **7**, 259–261.

Searle, S. R. (1971), *Linear models*, John Wiley, New York.

Seber, G. A. F. (1977), *Linear regression analysis*, John Wiley, New York.

Sen, A. & Srivastava, M. (1990), *Regression analysis. Theory, methods and applications*, Springer.

Smith, D. M. (1991), Regression using QR decomposition methods, PhD thesis, University of Kent, UK.

Smith, D. M. & Bremner, J. M. (1989), 'All possible subset regressions using the QR decomposition', *Computational Statistics and Data Analysis* **7**(3), 217–235.

Somol, P., Pudil, P. & Kittler, J. (2004), 'Fast branch and bound algorithms for optimal feature selection', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(7), 900–912.

Srivastava, V. K. & Dwivedi, T. D. (1979), 'Estimation of seemingly unrelated regression equations models: a brief survey', *Journal of Econometrics* **10**, 15–32.

Srivastava, V. K. & Giles, D. E. A. (1987), *Seemingly unrelated regression equations models: estimation and inference*, Vol. 80 of *Statistics: textbooks and monographs*, Marcel Dekker, New York.

Stefanski, L. A. (1991), 'A note on high-breakdown estimators', *Statistics and Probability Letters* **11**, 353–358.

Tibshirani, R. J. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* **58**(1), 267–288.

Trefethen, L. N. & Bau, D. (1997), *Numerical linear algebra*, SIAM.

Yanev, P., Foschi, P. & Kontoghiorghes, E. J. (2004), 'Algorithms for computing the QR decomposition of a set of matrices with common columns', *Algorithmica* **39**, 83–93.

Yanev, P. & Kontoghiorghes, E. J. (2004), 'Efficient algorithms for block downdating of least squares solutions', *Applied Numerical Mathematics* **49**, 3–15.

Yanev, P. & Kontoghiorghes, E. J. (2007), 'Computationally efficient methods for estimating the updated-observations SUR models', *Applied Numerical Mathematics* **57**(11-12), 1245–1258.

Zellner, A. (1962), 'An efficient method of estimating seemingly unrelated regression equations and tests for aggregation bias', *Journal of the American Statistical Association* **57**, 348–368.

Zellner, A. (1963), 'Estimators for seemingly unrelated regression equations: some exact finite sample results', *Journal of the American Statistical Association* **58**.