ALGORITHMS FOR SOLVING STATISTICAL MODEL SELECTION PROBLEMS

Cristian Gatu

Ph.D. Thesis

Institut d'informatique
Université de Neuchâtel
Switzerland
2004

IMPRIMATUR POUR LA THESE

Algorithms for solving statistical model selection problems

M. Cristian GATU

UNIVERSITE DE NEUCHATEL

FACULTE DES SCIENCES

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

MM. P.-J. Erard (directeur de thèse), E. Kontoghiorghes (co-directeur de thèse), P. Kropf, M. Gilli (Genève) et P. Winker (Erfurt D)

autorise l'impression de la présente thèse.

Neuchâtel, le 22 avril 2004

La doyenne:

Martine Rahier

To my family

Acknowledgments

I am indebted to my supervisor and friend Professor Erricos J. Kontoghiorghes for his support, encouragement and enthusiasm. I am grateful to him for sharing with me four years of professional activities and memorable personal experiences. My thanks are extended to Professor Manfred Gilli (University of Geneva) and Professor Peter Winker (University of Erfurt) for their useful input and kind support during my Ph.D. studies. I am also grateful to the Department of Computer Science, University of Neuchâtel for hosting me and supporting my Ph.D. studies. This thesis was also supported by the Swiss National Science Foundation Grants 1214-056900.99/1 and 101312-100757/1. Many thanks to my colleague Dr. Paolo Foschi for his encouragement, altruism and interesting conversations. I thank also to my dear colleagues and friends Desi Nedyalkova, Petko Yanev and Marc Hofmann for the wonderful time we spent together, long discussions, kind encouragement and friendship. Finally, I thank my parents Constantin and Viorica, my brother Constantin, my sister-in-law Irina and my dearest nephew Cristian for their support and love.

Abstract

Statistical model selection problems arises in diverse areas. Some of the selection methods have exponential complexities and thus, are computationally demanding. The purpose of this thesis is to propose computationally efficient and numerical reliable algorithms used in statistical model selection. Particular emphasis is given to the computationally intensive model selection strategies which evaluate regression trees and have combinatorial solutions. The computational efficiency of the proposed algorithms has been investigated by detailed complexity analysis.

Parallel algorithms to compute all possible subset regression models are designed, implemented and analyzed. A branch-and-bound strategy that computes the best-subset regression models corresponding to each number of variables is proposed. A heuristic version of this strategy is developed. It is based on a tolerance parameter when deciding to cut a subtree. Experimental results which support the theoretical results of the new strategies are shown. The adaptation of the various regression tree strategies to subset Vector Autoregressive model selection problems is pursued. Various special cases for subset selection which exploit the common columns of the data matrices and the Kronecker structure of the variance-covariance matrix are investigated. Within this context, the design of a combinatorial algorithm to compute efficiently the estimators of a seemingly unrelated regressions model with permuted exogenous data matrices is designed.

The algorithms developed in this thesis have as a main computational component the QR decomposition and its modification. Efficient strategies to compute the various matrix factorization problems which arise in the estimation procedures are designed. The non-dense structure of the matrices is exploited, Kronecker products are not explicitely computed and computation of matrix inverses is avoided.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

A common problem in statistics is that of estimating parameters of some assumed relationship between one or more variables. One such relationship is

$$y = f(a_1, \ldots, a_n), \tag{1.1}$$

where y is the output variable and $a_1, \ldots, a_n$ are the input variables. Regression analysis estimates the form of the relationship (1.1) by using the observed values of the variables. This attempt in describing how these variables are related to each other is know as model building. In general, there will be other unmeasured variables that also contribute to y. Thus, the specification of the relationship (1.1) is given by

$$y = f(a_1, \ldots, a_n) + \varepsilon,$$

where ε is the disturbance term of error, whose specific value in any single observation cannot be predicted. The purpose of ε is to characterize the discrepancies that emerge between the actual observed value of y and the values that would be assigned by an exact functional relationship. The difference between the observed and the predicted value of y is called residual.

An important problem in regression analysis is that of deciding which explanatory variables, or regressors, or factors should be included in the model such that it provides better forecast. Given a list of variables which may have an effect on the dependent variable, there are two conflicting criteria for selecting a subset of them. On one hand, the model chosen should include as many variables as possible from the list if reliable prediction are to be obtained from the fitted equations. On the other hand, there are several reasons for which someone would like to reduce the list of variables included in the model. One important reason is the resulting parsimony; it is easier to

work with simpler models. A second reason is that reducing the number of variables often reduces multi-collinearity. Finally, measuring the dependent variable using all factors could be expensive. Thus, it is expected to be able to predict it with sufficient accuracy from a subset of variables which can be measured cheaply. Deleting some independent variables usually biases the estimates of the parameters left in the model, and decreases their dispersions. A trade-off should be found between the two extremes, which is equivalent in finding a balance between the benefits and consequences of reducing the number of variables in the model. This problem is referred to as selecting the best subset or selecting the best regression equation [14, 54, 94, 95, 109, 110].

The objective of this thesis is to design computationally efficient and numerical reliable algorithms for deriving the best subset models. Specifically, an efficient parallel algorithm which generates all possible subset regression models is proposed. A branch-and-bound algorithm which computes the best-subset regression models is described. The problem of subset Vector Autoregressive (VAR) model selection or equivalently, the lag structure identification is also considered. Efficient strategies which exploits the particular structure of the models are presented. They allow the faster investigation of subclasses of VAR models when compared to the standard methods. Finally, an efficient algorithm for generating all the Seemingly Unrelated Regression models with permuted exogenous data matrices is proposed within the context of subset VAR model selection.

1.1 Linear models

A linear model is one relationship in which an endogenous variable y can be expressed as a linear function of independent exogenous variables $a_1, \ldots, a_n$ i.e.

$$y = \beta_1 a_1 + \beta_2 a_2 + \ldots + \beta_n a_n + \varepsilon,$$

where β_i $(i = 1, \ldots, n)$ are unknown constants and ε is the disturbance term or error. If there are m $(m > n)$ sample observations, then this relationship can be written as

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{pmatrix}.$$

In compact form, the latter can be written as

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

where $y, \varepsilon \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ and $\beta \in \mathbb{R}^n$. To complete the description of the linear model (1.2), additional assumptions should be specified. The first assumption is that the expected value of ε is zero, that is, $\mathrm{E}(\varepsilon) = 0$. The second assumption is that A is a non-stochastic matrix which implies $\mathrm{E}(A^T\varepsilon) = 0$. The final assumption is that the various values of the ε are normally distributed, i.e. the variance-covariance matrix of ε is $\sigma^2\Omega$, where Ω is a symmetric non-negative definite matrix and σ is an unknown scalar. In summary, the complete mathematical specification of the (general) Linear Model is given by

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

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 $\sigma^2\Omega$ [102].

1.1.1 The Ordinary Linear Model

Consider the Ordinary Linear Model (OLM)

$$y = A\beta + \varepsilon, \qquad \varepsilon \sim (0, \sigma^2 I_m), \tag{1.3}$$

where $y \in \mathbb{R}^m$ is the response vector, $A \in \mathbb{R}^{m \times n}$ 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 error vector. The OLM assumptions are that each ε_i has the same variance and all disturbances are pairwise uncorrelated. That is, $\mathrm{Var}(\varepsilon_i) = \sigma^2$ and $\mathrm{E}(\varepsilon_i^T\varepsilon_j) = 0$ for $1 \leq i \neq j \leq m$.

The Ordinary Least Squares (OLS) estimator of (1.3) is found by minimizing

$$e^T e = (y - A\beta)^T (y - A\beta).$$

This is equivalent in setting $\partial(e^T e)/\partial\beta = 0$ which gives the least-squares normal equations $A^T A\beta = A^T y$ and thus, the OLS estimator is given by $\hat{\beta} = (A^T A)^{-1} A^T y$. The OLS estimator is the Best Linear Unbiased Estimator (BLUE) for the linear model (1.3) [102]. That is, $\mathrm{E}(\hat{\beta}) = \beta$ and if $\tilde{\beta}$ is another linear unbiased estimator for β, then $\mathrm{E}((\tilde{\beta} - \beta)(\tilde{\beta} - \beta)^T) - \mathrm{E}((\hat{\beta} - \beta)(\hat{\beta} - \beta)^T)$ is non-negative definite.

Alternatively, let the QR Decomposition (QRD) of the explanatory data matrix A be given by

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix}_{m-n}^{n} \qquad \text{and} \qquad Q^T y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}_{m-n}^{n}, \tag{1.4}$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal, satisfying $Q^TQ = QQ^T = I_m$, and $R \in \mathbb{R}^{n\times n}$ is upper triangular. The OLS estimator of (1.3) derives by minimizing $\|e\|^2 = \|y - A\beta\|^2$, that is

$$\begin{aligned}\hat{\beta} &= \underset{\beta}{\operatorname{argmin}} \ \|y - A\beta\|^2 \\ &= \underset{\beta}{\operatorname{argmin}} \ \left\|Q^Ty - Q^TA\beta\right\|^2 \\ &= \underset{\beta}{\operatorname{argmin}} \ (\|y_1 - R\beta\|^2 + \|y_2\|^2) \\ &= R^{-1}y_1.\end{aligned}$$

Notice that the Residual Sum of Squares (RSS) $\|y_2\|^2 = \|y - A\hat{\beta}\|^2$ and σ^2 is estimated by $\hat{\sigma}^2 = \|y_2\|^2/(m-n)$.

1.1.2 The General Linear Model

The difference between the General Linear Model (GLM) and the OLM is that there is a correlation between the disturbances ε_i $(i = 1,\ldots,m)$. The GLM is given by (1.2), i.e.

$$y = A\beta + \varepsilon, \qquad \varepsilon \sim (0, \sigma^2\Omega), \tag{1.5}$$

where Ω is a known, positive definite matrix. The BLUE of β in (1.5) comes from solving the Generalized Least Squares (GLS) problem

$$\underset{\beta}{\operatorname{argmin}} \|y - X\beta\|^2_{\Omega^{-1}},$$

where, $\|\cdot\|_\Omega$ denotes the energy norm, i.e. $\|v\|^2_\Omega = v^T\Omega v$. The latter is equivalent to the normal equations

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

which have solution

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

This solution is computationally expensive and numerical unstable when Ω is ill-conditioned [12, 81]. Furthermore, if Ω is singular, then the numerical solution of the GLM will fail completely and the replacement of Ω^{-1} by the Moore-Penrose generalized inverse would not always give the BLUE of β [77].

To avoid problems associated with the singularity or ill-conditioning of Ω, the GLM (1.5) can be formulated as the Generalized Linear Least Squares Problem (GLLSP)

$$\underset{\upsilon,\beta}{\operatorname{argmin}}\, \upsilon^T \upsilon \qquad \text{subject to} \qquad y = A\beta + C\upsilon, \tag{1.6}$$

where $\Omega \in \mathbb{R}^{m\times m}$ is semi-positive definite with rank g, $\Omega = CC^T$, $C \in \mathbb{R}^{m\times g}$ has full column rank and the random g-element vector υ is defined as $C\upsilon = \varepsilon$. That is, $\upsilon \sim (0, \sigma^2 I_g)$ [77]. Without loss of generality consider the case where Ω is non-singular. For the solution of the GLLSP (1.6), the Generalized QRD (GQRD) can be employed [6, 100]. The GQRD of A and C is given by the QRD (1.4) and the RQD of $Q^T C$, i.e.

$$(Q^T C)P = W \equiv \begin{array}{c} \begin{matrix} n & m-n \end{matrix} \\ \begin{pmatrix} W_{11} & W_{12} \\ 0 & W_{22} \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix} \end{array} \qquad \text{and} \qquad P^T \upsilon = \begin{pmatrix} \upsilon_1 \\ \upsilon_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}, \tag{1.7}$$

where $P \in \mathbb{R}^{m\times m}$ is orthogonal and $W \in \mathbb{R}^{m\times m}$ is upper triangular and non-singular. The GLLSP (1.6) can be equivalently written as

$$\underset{\upsilon,\beta}{\operatorname{argmin}} \left\| P^T \upsilon \right\|^2 \qquad \text{subject to} \qquad Q^T y = Q^T A\beta + Q^T CPP^T \upsilon$$

or

$$\underset{\upsilon_1,\upsilon_2,\beta}{\operatorname{argmin}} \left(\|\upsilon_1\|^2 + \|\upsilon_2\|^2 \right) \quad \text{subject to} \quad \begin{cases} y_1 = R\beta + W_{11}\upsilon_1 + W_{12}\upsilon_2, \\ y_2 = W_{22}\upsilon_2. \end{cases} \tag{1.8}$$

From the second constraint of (1.8) it follows that $\upsilon_2 = W_{22}^{-1} y_2$ and in the first constraint the arbitrary sub-vector υ_1 is set to zero in order to minimize the objective function. Thus, the estimator of β derives from the solution of the upper triangular system $R\beta = y_1 - W_{12}\upsilon_2$. The variance-covariance of the coefficients estimator is given by $\hat{\sigma}^2 R^{-T} W_{11}^T W_{11} R^{-1}$, where $\hat{\sigma}^2 = \|\upsilon_2\|^2 / (m-n)$ is an estimator of σ^2.

1.1.3 The Seemingly Unrelated Regressions Model

The Seemingly Unrelated Regressions (SUR) Model is a special case of GLM and is defined by the set of G regression equations

$$y_i = X_i \beta_i + u_i, \quad i = 1, \ldots, G,$$

where $y_i \in \mathbb{R}^M$ are the response vectors, $X_i \in \mathbb{R}^{M\times k_i}$ are the exogenous matrices with full column rank, $\beta_i \in \mathbb{R}^{k_i}$ are the coefficients and $u_i \in \mathbb{R}^M$ are the disturbances. The expectation of u_i is zero, i.e. $\mathrm{E}(u_i) = 0$ and $\mathrm{E}(u_i u_j^T) = \sigma_{ij} I_M$ $(i, j = 1, \ldots, G)$ [63, 72, 114]. In compact form, the SUR model can be written as

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_G \end{pmatrix} = \begin{pmatrix} X_1 & & & \\ & X_2 & & \\ & & \ddots & \\ & & & X_G \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_G \end{pmatrix} + \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_G \end{pmatrix},$$

or

$$\mathrm{vec}(Y) = (\oplus_{i=1}^G X_i)\,\mathrm{vec}\,(\{\beta_i\}_G) + \mathrm{vec}(U), \tag{1.9}$$

where $Y = (y_1 \ldots y_G)$ and $U = (u_1 \ldots u_G)$. The direct sum of matrices $\oplus_{i=1}^G X_i$ defines a $GM \times K^*$ block-diagonal matrix

$$\oplus_{i=1}^G X_i = X_1 \oplus X_2 \oplus \ldots \oplus X_G = \begin{pmatrix} X_1 & & & \\ & X_2 & & \\ & & \ddots & \\ & & & X_G \end{pmatrix},$$

where $K^* = \sum_{i=1}^G k_i$ [103]. The vec operator stacks the columns of its matrix or set of vectors $\{\beta_i\}_G \equiv \beta_1, \ldots, \beta_G$ argument in a column vector, that is

$$\mathrm{vec}(Y) = \begin{pmatrix} y_1 \\ \vdots \\ y_G \end{pmatrix} \qquad \text{and} \qquad \mathrm{vec}\,(\{\beta_i\}_G) = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_G \end{pmatrix}.$$

For notational convenience the direct sum $\oplus_{i=1}^G$ and the set $\{\cdot\}_G$ are abbreviated to $\oplus_i$ and $\{\cdot\}$, respectively. The disturbance term $\mathrm{vec}(U)$ in (1.9) has zero mean and variance-covariance matrix $\Sigma \otimes I_M$, where $\Sigma = [\sigma_{ij}] \in \mathbb{R}^{G\times G}$ is symmetric and positive definite and $\otimes$ denotes the Kronecker product [58, 72, 74, 105, 106, 113, 114, 126, 127]. That is, $\mathrm{vec}(U) \sim (0, \Sigma \otimes I_M)$ and

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

Notice that $(A \otimes B)(C \otimes D) = AC \otimes BD$ and $\mathrm{vec}(ABC) = (C^T \otimes A)\,\mathrm{vec}(B)$.

1.1.4 The Vector Autoregressive Model

The vector time series $z_t \in \mathbb{R}^G$ is a Vector Autoregressive (VAR) process of order p when its data generating process has the form

$$z_t = \Phi_1 z_{t-1} + \Phi_2 z_{t-2} + \cdots + \Phi_p z_{t-p} + \varepsilon_t, \tag{1.10}$$

where $\Phi_i \in \mathbb{R}^{G\times G}$ are the coefficient matrices and $\varepsilon_t \in \mathbb{R}^G$ is the noise vector. Given a set of realizations of the process in (1.10), $z_1, \ldots, z_M$ and a pre-sample $z_0, \ldots, z_{1-p}$ the parameter matrices are estimated from the linear model

$$\begin{pmatrix} z_1^T \\ z_2^T \\ \vdots \\ z_M^T \end{pmatrix} = \begin{pmatrix} z_0^T & z_{-1}^T & \cdots & z_{1-p}^T \\ z_1^T & z_0^T & \cdots & z_{2-p}^T \\ \vdots & \vdots & \ddots & \vdots \\ z_{M-1}^T & z_{M-2}^T & \cdots & z_{M-p}^T \end{pmatrix} \begin{pmatrix} \Phi_1^T \\ \Phi_2^T \\ \vdots \\ \Phi_p^T \end{pmatrix} + \begin{pmatrix} \varepsilon_1^T \\ \varepsilon_2^T \\ \vdots \\ \varepsilon_M^T \end{pmatrix}. \tag{1.11}$$

In the compact form the model in (1.11) can be written as

$$Y = XB + U, \tag{1.12}$$

where $Y = (y_1 \ldots y_G) \in \mathbb{R}^{M\times G}$ are the response vectors, $X \in \mathbb{R}^{M\times K}$ is the exogenous data matrix having full-column rank and block-Toeplitz structure, $B \in \mathbb{R}^{K\times G}$ is the coefficient matrix, $U = (u_1 \ldots u_G) \in \mathbb{R}^{M\times G}$ are the disturbances and $K = Gp$. A matrix $T \in \mathbb{R}^{n\times n}$ is said to have Toeplipz structure if there exist scalars $r_{-n+1}, \ldots, r_0, \ldots, r_{n-1}$ such that $t_{ij} = r_{j-i}$ for $i, j = 1, \ldots, n$. The expectation of U is zero, i.e. $\mathrm{E}(u_i) = 0$, and $\mathrm{E}(u_i u_j^T) = \sigma_{ij} I_M$ $(i, j = 1, \ldots, G)$ [72, 77, 86, 98, 99, 114].

The VAR model (1.12) can be written as

$$\mathrm{vec}(Y) = (I_G \otimes X)\,\mathrm{vec}(B) + \mathrm{vec}(U), \quad \mathrm{vec}(U) \sim (0, \Sigma \otimes I_M), \tag{1.13}$$

where $\Sigma = [\sigma_{ij}] \in \mathbb{R}^{G\times G}$ has full rank [33, 63]. The OLS and GLS estimators of (1.13) are the same. Let the QRD of $(X \quad Y)$ be given by

$$(X \quad Y) = Q \begin{pmatrix} R \\ 0 \end{pmatrix} = \begin{array}{c} \begin{matrix} Gp & G & M-(p+1)G \end{matrix} \\ \begin{pmatrix} Q_T & Q_Y & Q_N \end{pmatrix} \end{array} \begin{array}{c} \begin{matrix} Gp & G \end{matrix} \\ \begin{pmatrix} R_T & R_{TY} \\ 0 & R_Y \\ 0 & 0 \end{pmatrix} \begin{matrix} Gp \\ G \\ M-(p+1)G \end{matrix} \end{array},$$

where $Q \in \mathbb{R}^{M\times M}$ is orthogonal and $R \in \mathbb{R}^{G(p+1)\times G(p+1)}$ is upper triangular. The OLS estimator of B in (1.13) is computed by $\hat{B} = R_T^{-1} R_{TY}$. The residuals are given by $\hat{U} = Q_Y R_Y$, and the covariance matrix Σ is estimated by $\hat{\Sigma} = \hat{U}^T \hat{U}/\alpha = R_Y^T R_Y/\alpha$, where $\alpha = M$ or $\alpha = M - Gp$.

1.2 The QR Decomposition

The QR Decomposition (QRD) is one of the main computational tools in regression [11, 12, 21, 28, 45, 48, 81, 111]. It is mainly used in the solution of Least-Squares (LS) problems [10, 46, 57, 81]. Different methods have been proposed for forming the QRD (1.4) which is re-written as

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{1.14}$$

where $A \in \mathbb{R}^{m\times n}$, $Q \in \mathbb{R}^{m\times m}$ is orthogonal, $R \in \mathbb{R}^{n\times n}$ is upper-triangular and $m > n$. It is assumed that A has full-column rank. The two main methods for computing the QRD are based on Householder reflectors and Givens rotations [45, 63, 117].

1.2.1 The Householder method

An $m \times m$ Householder transformation (or Householder matrix or Householder reflector) has the form

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

where $h \in \mathbb{R}^m$ satisfies $\|h\|^2 \neq 0$. Householder matrices are symmetric and orthogonal, i.e. $H^T = H$ and $H^2 = I_m$. They can be used to annihilate specified elements of a vector or a matrix [12, 45]. 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 to m by setting $h = x \pm \alpha e_1$, where $\alpha = x^T x$ and e_1 denotes the first column of the $m \times m$ identity matrix I_m.

For the computation of the QRD (1.14) a sequence of n Householder transformations can be applied, i.e. the orthogonal matrix Q is defined as $Q^T = H_n \cdots H_2, H_1$. The ith Householder transformation is of the form

$$H_i = \begin{pmatrix} I_{i-1} & 0 \\ 0 & \tilde{H}_{i-1} \end{pmatrix},$$

where $\tilde{H}_i = I_{m-i+1} - 2(h_i h_i^T)/\|h_i\|^2$ and a zero dimension denotes a null matrix. If $A^{(0)} \equiv A$ and

$$A^{(i)} = H_i A^{(i-1)} \equiv \begin{array}{c} \begin{matrix} i & n-i \end{matrix} \\ \begin{pmatrix} R_{11}^{(i)} & R_{12}^{(i)} \\ 0 & \widetilde{A}^{(i)} \end{pmatrix} \begin{matrix} i \\ m-i \end{matrix} \end{array}, \qquad i = 1, \ldots, n,$$

where $R_{11}^{(i)}$ is upper triangular, then H_{i+1} from the left of $A^{(i)}$ annihilates the last $m-i-1$ elements of the first column of $\widetilde{A}^{(i)}$. The transformation $H_{i+1}A^{(i)}$ affects only $\widetilde{A}^{(i)}$ and it follows that

$$A^{(n)} \equiv \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

1.2.2 The Givens rotation method

An $m \times m$ Givens (plane) rotation is an orthogonal matrix having the structural form

$$G_{i,j}^{(k)} = \begin{array}{c} \begin{matrix} & & \scriptstyle i & & \scriptstyle j & & \end{matrix} \\ \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & & -s & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & s & & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \end{array} \begin{matrix} \\ \\ \leftarrow i \\ \\ \\ \leftarrow j \\ \\ \\ \end{matrix},$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$ for some θ, that is $c^2 + s^2 = 1$ [45]. The rotation $G_{i,j}^{(k)}$ when applied to the left of a matrix annihilates the kth element in the jth row and only the ith and the jth rows are affected.

Specifically, the transformation

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

results in $\tilde{a}_{j,k}$ being zero, where $A, \widetilde{A} \in \mathbb{R}^{m \times n}$ and $1 \leq k \leq n$. It follows that

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

Thus, if the $a_{j,k} \neq 0$ then it can be annihilated using the Givens transformation $G_{i,j}^{(k)}$ by setting $c = a_{i,k}/t$, $s = a_{j,k}/t$ and $t^2 = a_{i,k}^2 + a_{j,k}^2$.

The QRD (1.14) can be computed by applying a sequence of Givens rotations. One of such sequences, referred *column-based*, can be expressed as

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

Here, the rotations $(G^{(i)}_{m-1,m} \cdots G^{(i)}_{i,i+1})$ annihilate the elements $(m,i),\ldots,(i+1,i)$, i.e. the last $m-i$ elements of the ith column, and preserve the previously annihilated elements. While the Householder reflections are useful for introducing zero elements on the grand scale, Givens rotations are important because they can annihilate the elements of a matrix more selectively.

1.3 Statistical Model Selection

A measure of fit for regression is

$$R^2_p = 1 - \mathrm{RSS}_p/(y^T y),$$

where RSS_p denotes the residual sum of squares when fitting the model by including p variables out of n. The value of RSS_p lies between 0 and 1 and the closer it is to 1, the better the fit [52]. Alternatively, the estimate of σ^2,

$$\hat{\sigma}^2_p = \mathrm{RSS}_p/(m-p),$$

can be also used as a measure of fit, where m is the sample size. Small values of $\hat{\sigma}^2_p$ indicate a good fit. The bias of the estimates can be measured by the Mallow's C_p statistic

$$C_p = \mathrm{RSS}_p/\hat{\sigma}^2 - (m-2p),$$

where $\hat{\sigma}^2$ is the estimate of σ^2 when all the n variables are included in the regression. If selecting only the p variables does not lead to much bias in the predicted ones, then $\mathrm{E}(C_p) \approx p$ [87, 88]. Therefore, when considering several candidate models, one can look at the corresponding $\hat{\sigma}^2_p$, R^2_p and C_p values. Other criteria for evaluating the quality of models have been also proposed. A non-exhaustive list is given by adjusted R^2, Mean Square Error of Prediction (MSEP), Akaike Information Criterion (AIC), Information Complexity Criterion (ICOMP) and Bayesian Information Criterion (BIC) [2, 3, 4, 15, 16, 54, 118].

One approach when selecting the best subset models is to search over all possible subsets. This allows the examination of all $2^n - 1$ regression equations constructed out of n variables. As n increases, the number of models to be computed increases exponentially which implies an intensive

computational effort. Thus, this procedure can be improved by forcing a predetermined list of variables to be in and search for the others. Also, "short-cut" methods which reduces the search space can be employed [24, 35, 36, 37, 38, 79, 94, 97, 109, 112].

For large values of n or for multivariate dependent variable the method of generating all subsets becomes infeasible. Thus, polynomial stepwise (greedy) procedures have been proposed. These are based on adding or deleting variables one at a time according to a specific criterion [26, 54, 109, 110]. The Forward Selection procedure starts with no variable in the model and adds one variable at a time until either a stopping criterion is satisfied, or until all variables are selected. At each step, the variable with the largest single degree of freedom F-value, among those eligible is considered for inclusion. That is, variable i is added to a p-factors regression equation if

$$F_i = \max_i \left((\mathrm{RSS}_p - \mathrm{RSS}_{p+i}) / \hat{\sigma}^2_{p+i} \right) > F_{\mathrm{IN}},$$

where $(p+i)$ denotes the quantities computed when variable i is added to the current p-factors equation. The stopping criterion for the procedure is given by the specification of the quantity F_{IN}.

The Backward Elimination procedure starts with the full specified model. At any step, the variable with the smallest F-value is deleted if it does not exceed a specified value. That is, variable i is eliminated from the p-factors equation if

$$F_i = \min_i \left((\mathrm{RSS}_{p-i} - \mathrm{RSS}_p) / \hat{\sigma}^2_p \right) < F_{\mathrm{OUT}}.$$

Similarly, $(p-i)$ denotes the quantity computed when variable i is eliminated from the p-term regression equation. The quantity F_{OUT} gives the stopping criterion of the procedure.

The Stepwise Procedure is a combination of the two procedures just described. It starts with no variable in the model. After a new variable is added following a step like in the forward selection, every factor already selected is examined to check weather it should be deleted, just as in a backward step.

Finding the best subset models can be seen as an optimization problem, i.e. minimize or maximize a selection criterion for the full model subject to the restriction that certain coefficients –the ones deleted– are zero. Within this context, heuristics such as thresholding accepting methods and Genetic algorithms have been proposed [82, 116, 121].

1.4 Parallel processing and statistics

Parallel algorithms have been proposed to solve various problems arising in diverse applications. Emphasis has mainly been given to the solution of large scale industrial and engineering problems

[25, 30, 85, 89, 125]. Although some of these interdisciplinary fields such as signal processing and pattern recognition, contain a strong statistical computing component, overall the use of parallelism in statistics and econometrics can be seen as under developed [1]. This is mainly due to the lack of a strong interface between parallel computing and statistics [62, 68, 75, 76, 93, 124].

Various parallel numerical libraries contain subroutines useful to statistics. For example LAPACK and ScaLAPACK provide routines to solve constrained least squares and matrix problems that are useful in statistical modeling [5, 7, 13]. These routines have been build as a general numerical tool and cannot be used efficiently to solve statistical problems that exhibit special properties and characteristics. The design of purposely build parallel numerical libraries and tools will facilitate the solution of computational intensive statistical problems. Recently advances in parallel software and hardware have made it feasible to develop parallel methods to solve problems arising in statistical and economic modeling. The majority of these advances have been in terms of parallelizing the straightforward parts of existing algorithms or the re-engineering of the main matrix computations [8, 9, 20, 29, 96].

1.5 Overview of the thesis

Each chapter of the thesis is self contained[1]. Chapter 2 presents efficient parallel algorithms for deriving the best subset models. Sequential strategies for computing the residual sum of squares of all possible models of the standard regression using Givens rotations have been previously proposed. Specifically, an algorithm based on columns transpositions which derives all models has been developed [24]. At each step of the algorithm two adjacent columns are transposed and a Givens rotation is applied. Further, a dropping columns algorithm (DCA) which applies the Givens rotations on matrices of smaller size has been introduced [112]. The DCA generates a regression tree. The computations involved in these sequential methods are based on the re-triangularization of a matrix after interchanging or deleting columns. In Chapter 2, an efficient parallelization of the DCA is proposed. Its extension for the General Linear and the Seemingly Unrelated Regressions models and the case where new variables are added to the standard regression model are designed.

Chapter 3 presents a branch-and-bound algorithm (BBA) for finding the best regression equation which avoids the derivation of all subset models provided by the DCA described in Chapter 2. The BBA exploits the properties of the regression tree generated by the DCA and generates the best subset models corresponding to each number of variables. Existing leaps-and-bounds methods are

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

based on Gaussian elimination and inverse matrices [36]. The BBA outperforms by cubic order of complexity the existing leaps-and-bounds method which generates two trees. Strategies which improve the computational performance of the proposed algorithm together with an efficient heuristic version of the BBA which decides to cut sub-trees using a tolerance parameter are also presented.

Chapter 4 considers the regression tree strategy (DCA) of Chapter 2 in order to compute the subset Vector Autoregressive (VAR) models. The VAR model with zero-restriction on the coefficients is formulated as a Seemingly Unrelated Regressions (SUR) model where the exogenous matrices comprise columns of a triangular matrix. The common columns of the exogenous matrices and the Kronecker structure of the variance-covariance of the disturbances are exploited in order to derive efficient estimation algorithms. Within this context o the estimation of the SUR models comprising G equations is described in Chapter 5. An efficient combinatorial algorithm to compute all possible $G!$ SUR models with permuted exogenous data matrices is proposed. Finally, the last Chapter concludes and provides directions for future research.

Chapter 2

Parallel algorithms for computing all possible subset regression models using the QR decomposition

Abstract:

Efficient parallel algorithms for computing all possible subset regression models are proposed. The algorithms are based on the dropping columns method that generates a regression tree. The properties of the tree are exploited in order to provide an efficient load balancing which results in no inter-processor communication. Theoretical measures of complexity suggest linear speed-up. The parallel algorithms are extended to deal with the general linear and seemingly unrelated regression models. The case where new variables are added to the regression model is also considered. Experimental results on a shared memory machine are presented and analyzed.

[1]This chapter is a reprint of the paper: C. Gatu and E.J. Kontoghiorghes. Parallel algorithms for computing all possible subset regression models using the QR decomposition. *Parallel Computing*, 29(4):505–521, 2003.

2.1 Introduction

The problem of computing all possible subset regression models arises in statistical model selection. Most of the criteria used to evaluate the subset models require the residual sum of squares (RSS) [108]. Consider the standard regression model

$$y = A\beta + \varepsilon, \tag{2.1}$$

where $y \in \mathbb{R}^m$ is the dependent variable vector, $A \in \mathbb{R}^{m\times n}$ is the exogenous data matrix of full column rank, $\beta \in \mathbb{R}^n$ is the coefficient vector and $\varepsilon \in \mathbb{R}^m$ is the noise vector. It is assumed that ε has zero mean and variance-covariance matrix $\sigma^2 I_m$. Let the QR decomposition (QRD) of A be given by

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix} \quad \text{and} \quad Q^T y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}, \tag{2.2}$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $R \in \mathbb{R}^{n\times n}$ is upper triangular and non-singular. The least squares (LS) solution and the RSS of (1) are given by $\hat{\beta} = R^{-1}y_1$ and $y_2^T y_2$, respectively [12]. Let $A_{(S)} = AS$ and $\beta_{(S)} = S^T\beta$, where S is an $n \times k$ selection matrix such that AS selects k columns of A. Notice that the columns of S are the columns of the identity matrix I_n. For the LS solution of the modified model

$$y = A_{(S)}\beta_{(S)} + \varepsilon, \tag{2.3}$$

the QRD of $A_{(S)}$ is required. This is equivalent to re-triangularizing R in (2) after deleting columns [61, 63, 69]. That is, computing the factorization

$$Q_{(S)}^T RS = \begin{pmatrix} R_{(S)} \\ 0 \end{pmatrix} \begin{matrix} k \\ n-k \end{matrix} \quad \text{and} \quad Q_{(S)}^T y_1 = \begin{pmatrix} \tilde{y}_1 \\ \hat{y}_1 \end{pmatrix} \begin{matrix} k \\ n-k \end{matrix}. \tag{2.4}$$

The LS estimator for the new model and its corresponding RSS are given by $\hat{\beta}_{(S)} = R_{(S)}^{-1}\tilde{y}_1$ and $\mathrm{RSS}_{(S)} = \mathrm{RSS} + \hat{y}_1^T\hat{y}_1$, respectively. Let e_i denote the ith column of the $n \times n$ identity matrix I_n. Notice that if $S = (e_1, e_2, \ldots, e_k)$, then $Q_{(S)}^T = I_n$ and $R_{(S)}$ is the leading $k \times k$ sub-matrix of RS, where $k = 1, \ldots, n$.

The number of all possible selection matrices S, and thus models, is $2^n - 1$. The leading sub-matrices of R provide n of these models. As n increases, the number of models to be computed increases exponentially. Therefore, efficient algorithms for fitting all possible subset regression

models are required. Sequential strategies for computing the upper-triangular $R_{(S)}$ and the corresponding $\mathrm{RSS}_{(S)}$ for all possible selection matrices S have been previously proposed [24, 112]. Clarke developed an algorithm which derives all models based on a columns transposition strategy [24]. At each step of the algorithm two adjacent columns are transposed and a Givens rotation is applied to re-triangularize the resulted matrix. The order of transposition is important as it applies the minimum $2^n - n - 1$ Givens rotations. Smith and Bremner developed a Dropping Columns Algorithm (DCA) which generates a regression tree [112]. The DCA applies the Givens rotations on matrices of smaller size and overall has less computational complexity. However, unlike Clarke's method, it requires intermediate storage [111]. The computations involved in these sequential methods are based on the re-triangularization of a matrix after interchanging or deleting columns. Givens rotations can be efficiently applied to re-triangularize a matrix after it has been modified by one column or row [43, 45]. Let the $m \times m$ Givens rotation $G_{i,j}^{(k)}$ have the structural form

$$G_{i,j}^{(k)} = \begin{array}{c} \begin{array}{ccccccc} & & i & & j & & \\ & & \downarrow & & \downarrow & & \end{array} \\ \begin{pmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & c & \dots & s & \dots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & -s & \dots & c & \dots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix} \end{array} \begin{array}{l} \\ \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \end{array} , \tag{2.5}$$

where $c^2 + s^2 = 1$. The Givens rotation $G_{i,j}^{(k)}$ is orthogonal and when applied from the left of a matrix, annihilates the kth element on the jth row and only the ith and jth rows are affected. Hereafter the Givens rotation $G_i \equiv G_{i,i-1}^{(i-i)}$. Constructing a Givens rotation requires 6 flops. The time to construct a Givens rotation will be denoted by t. The same time is required to apply the rotation to a 2–element vector [45]. Thus, $tn \equiv 6n$ flops are needed to annihilate an element of a $2 \times n$ non-zero matrix.

Parallel strategies for computing all possible subset regression models are investigated. An efficient parallelization of the DCA is proposed. Its extension for the general linear and seemingly unrelated regression models, and the case where new variables are added to the standard regression model are considered. Theoretical measures of complexity suggest linear speed-up when $p = 2^\rho$ ($\rho \geq 0$) processors are used. All the parallel algorithms have been implemented using Fortran,

BLAS and MPI on the shared memory SUN Enterprise 10000 (16 CPU UltraSPARC 400 MHz) [34]. The execution times in the experimental results are reported in seconds.

In section 2 a formal description of the regression trees generated by the DCA is presented and their properties are investigated. The parallelization of the DCA together with an updating algorithm for generating all subset regression models are described in section 3. Theoretical measures of complexity are derived. Section 4 considers the extension of the parallel DCA to the general linear and seemingly unrelated regression models. Conclusions and future work are presented and discussed in section 5.

2.2 Regression trees

The DCA has been briefly discussed in [111, 112]. Here a formal and detailed description is given. Let $M^v_{k,\lambda}$ denote the upper triangular factor in the QRD of an exogenous matrix comprising the columns (variables) $v_1,\ldots,v_{k+\lambda}$. Furthermore, the index pair (k,λ) indicates that the columns $k+1,\ldots,k+\lambda-1$ will be deleted one at a time from the triangular (exogenous) matrix in order to obtain new models. Within this context the regression tree $T^v_{k,\lambda}$ defines an $(\lambda-1)$–tree having as root node $M^v_{k,\lambda}$ with the children $T^{v^{(k+i)}}_{k+i-1,\lambda-i}$ for $i=1,\ldots,\lambda-1$. Here, $v^{(k+i)}$ denotes the vector v, without its $(k+i)$th element. Notice that $M^v_{k,\lambda}$ together with the modified response variable Q^Ty can provide the RSS of the sub-models comprising the variables $(v_1),(v_1v_2),\ldots,$ and $(v_1\cdots v_{k+\lambda})$. The models $(v_1),(v_1v_2),\ldots,(v_1\cdots v_k)$ can be extracted from a parent node of the regression tree while the new sub-models provided by $M^v_{k,\lambda}$ are $(v_1\ldots v_{k+1}),\ldots,(v_1\ldots v_{k+\lambda})$. The derivation of a child node from its parent requires a re-triangularization of an upper triangular matrix after deleting a column using Givens rotations. The rotations are also applied on the modified response vector Q^Ty. Emphasis will be given to the retriangularization of the matrices. For simplicity the application of the Givens rotations on the response vector will not be discussed, but it will be taken into consideration for the complexity analysis. Figure 2.1 shows the sequence of Givens rotations for re-triangularizing a 5×5 upper triangular matrix after deleting its 2nd column. Shadowed frames indicate the submatrices affected by the Givens rotations at each stage of the re-triangularization.

The application of the DCA on the regression model (2.1) is equivalent to a leftmost walk on the regression tree $T^v_{0,n}$, where $M^v_{0,n}\equiv R$ in (2.2), $v_i=i$ and $i=1,\ldots,n$. Figure 2.2 shows $T^v_{0,5}$ together with the sub-models which can be extracted from each node. A submodel is denoted by a sequence of numbers which corresponds to the variable indices. The operations *Drop* and *Shift* are used to derive a child node from its parent. Given $M^v_{k,\lambda}$ the Drop operation deletes the $(k+1)$th

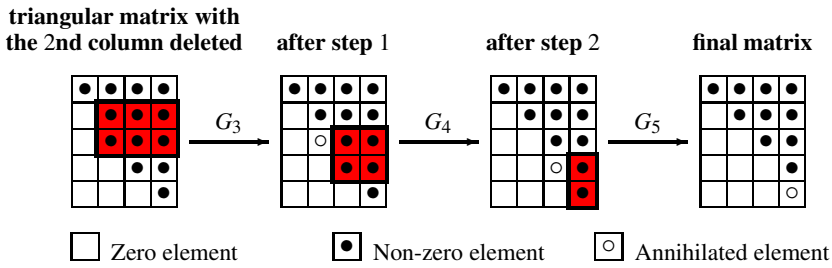


Figure 2.1: Re-triangularization of an $n \times n$ upper triangular matrix after deleting the kth column using Givens rotations, where $n = 5$ and $k = 2$.

column, applies the $\lambda - 1$ Givens rotations $G_{k+2}, \ldots, G_{k+\lambda}$ to re-triangularize the modified matrix and returns $M_{k,\lambda-1}^{v^{(k+1)}}$. Figure 2.1 corresponds to the application of Drop on $M_{1,4}^{v}$ which returns $M_{1,3}^{v^{(2)}}$. Given $M_{k,\lambda}^{v}$ the Shift operation returns $M_{k+1,\lambda-1}^{v}$. That is, it simply modifies the index of the first column to be deleted from $M_{k,\lambda}^{v}$ by incrementing k and decrementing λ. From the parent $M_{k,\lambda}^{v}$ the ith child is obtained by applying $(i-1)$ Shifts followed by a Drop. For example, in Fig. 2.2, $M_{1,2}^{(2,4,5)}$ derives from $M_{0,4}^{(2,3,4,5)}$ after a Shift followed by a Drop. This indicates that the sub-models derived from the subtree $T_{k+i-1,\lambda-i}^{v^{(k+i)}}$ will always comprise the variables $v_1, \ldots, v_{k+i-1}$.

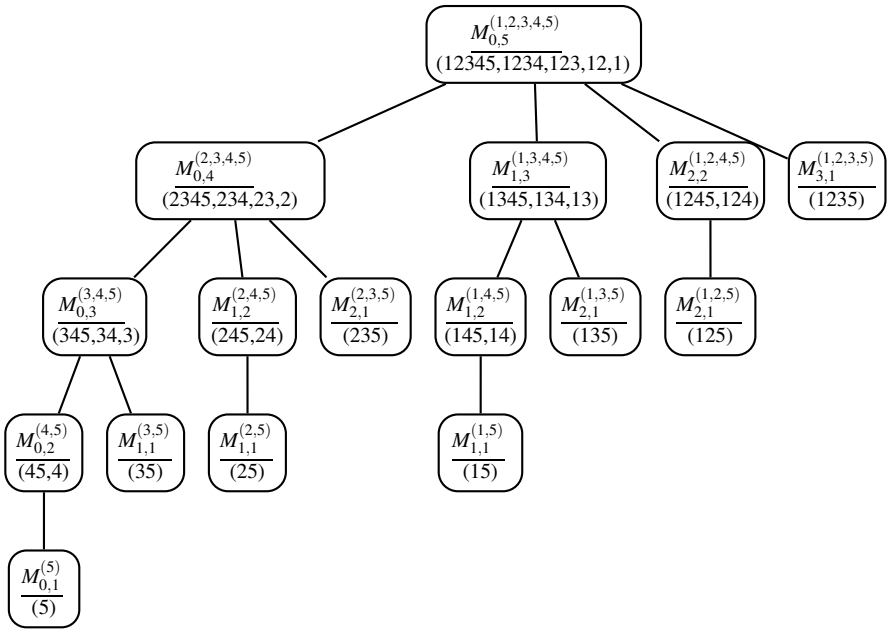


Figure 2.2: The regression tree $T_{0,n}^{v}$, where $n = 5$ and $v = (1, \ldots, n)$.

The *SubTree* procedure shown in Algor. 1 generates the regression tree $T^v_{k,\lambda}$ given as an argument the root node $M^v_{k,\lambda}$. Thus, the DCA for the n–variable model (2.1) is equivalent to SubTree($M^v_{0,n}$), where $v = (1,\ldots,n)$ and in the QRD (2.2) $R \equiv M^v_{0,n}$. The application of Drop on $M^v_{k,\lambda+1}$ depends only on λ and has complexity

$$C_{\text{Ret}}(\lambda) = t\sum_{j=1}^{\lambda}(j+1) = t(\lambda^2+3\lambda)/2. \tag{2.6}$$

Thus, the complexity of generating $T^v_{k,\lambda}$ with root node $M^v_{k,\lambda}$ is given by:

$$\begin{aligned} C(\lambda) &= \sum_{i=1}^{\lambda-1}\big(C_{Ret}(\lambda-i)+C(\lambda-i)\big) \\ &= C_{Ret}(\lambda-1)+2C(\lambda-1) \\ &= 2^{\lambda-1}C(1)+\sum_{i=1}^{\lambda-1}2^{i-1}C_{Ret}(\lambda-i). \end{aligned} \tag{2.7}$$

Now, since $C(1) = 0$ and using (2.6) in (2.7) it follows that

$$C(\lambda) = 3t2^{\lambda} - t(\lambda+2)(\lambda+3)/2. \tag{2.8}$$

Therefore, the complexity of the DCA is $O(2^n)$ and specifically given by:

$$C_{\text{DCA}}(n) = C(n) \approx 3t2^n.$$

Algorithm 1 Generating the regression tree $T^v_{k,\lambda}$ from the root node $M^v_{k,\lambda}$.

1: **procedure** SubTree($M^v_{k,\lambda}$)
2: From $M^v_{k,\lambda}$ obtain the RSS of the sub–models $(v_1\cdots v_{k+1}),\ldots,(v_1\cdots v_{k+\lambda})$
3: **for** $i = 1,\ldots,\lambda-1$ **do**
4: Store $M^v_{k,\lambda}$
5: $M^v_{k+i-1,\lambda-i+1} \longleftarrow$ Apply $i-1$ Shifts on $M^v_{k,\lambda}$
6: $M^{v^{(k+i)}}_{k+i-1,\lambda-i} \longleftarrow$ Apply Drop on $M^v_{k+i-1,\lambda-i+1}$
7: SubTree($M^{v^{(k+i)}}_{k+i-1,\lambda-i}$)
8: **end for**
9: **end procedure**

2.3 The parallel DCA

For the design of an efficient parallel DCA (hereafter PDCA), the properties of the regression trees need to be investigated and exploited. The number of nodes in the $(\lambda-1)-$tree $T^v_{k,\lambda}$ is given by

$$\delta_\lambda = 1 + \sum_{i=1}^{\lambda-1} \delta_i = 2^{\lambda-1},$$

where δ_i $(i = 1,\ldots,\lambda-1)$ is the number of nodes in the subtree $T^{v^{(k+\lambda-i)}}_{k+\lambda-i-1,i}$. Notice that,

$$\delta_{j+1} = 1 + \sum_{i=1}^{j} \delta_i.$$

This indicates that the nodes of $T^v_{k,\lambda}$, excluding the root node, can be divided in two sets of nodes. The first set comprises the $\delta_{\lambda-1}$ nodes of the subtree $T^{v^{(k+1)}}_{k,\lambda-1}$ and the second set consists of the $\delta_{\lambda-1}-1$ nodes of the remaining subtrees. This property applies recursively for each resulted set. Thus, given $p = 2^\rho$ processors $(\rho < n-1)$, half of them are allocated to the $T^{v^{(k+1)}}_{k,\lambda-1}$ and the rest of the processors are allocated to the remaining of the tree, i.e. $T^v_{k+1,\lambda-1}$. This procedure is recursively applied to each $T^{v^{(k+1)}}_{k,\lambda-1}$ and $T^v_{k+1,\lambda-1}$ until each processor is allocated a unique subtree. These subtrees have the same complexity. Figure 2.3 illustrates the case for $n = 5$ and $p = 4$. Dashed boxes indicate nodes derived from a Shift operation which requires no computation. The remaining nodes are obtained using a Drop operation. Notice that the number of Shifts and Drops performed by the processor P_r $(r = 0,\ldots,2^\rho-1)$ is equal to the number of ones and zeros in the binary representation of r, respectively. Thus, P_0 and $P_{2^\rho-1}$ perform only Drops and Shifts, respectively. Shadowed boxes denote the subtrees which have the same complexity.

The PDCA uses a SPMD (Single–Program Multiple–Data) paradigm and is divided to *mapping* and *computation* phases [34]. Initially all the processors are allocated the parent node $M^v_{0,n}$. In the Mapping phase each processor performs a sequence of Drop or Shift operations until it has generated a unique node. In the Computation phase each processor uses this node to generate simultaneously $2^{n-\rho-1}-1$ $(0 \le \rho < n-1)$ nodes. Algorithm 2 summarizes the steps of the PCDA, while Fig. 2.3 illustrates its execution on 4 processors, where $n = 5$. The initial computations in step 1 are not explicitly specified. All the processors can contribute to the computation of the QRD and obtain a copy of the triangular matrix, or one processor computes the QRD and broadcast the triangular factor to the remaining processors. The Mapping phase is shown in lines 4–11 of Algor. 2 which is executed by each processor. Notice that a Drop generates a new node (re-triangularizing a matrix), while a Shift just changes the indices (k,λ) of the node. Furthermore, the time complexity

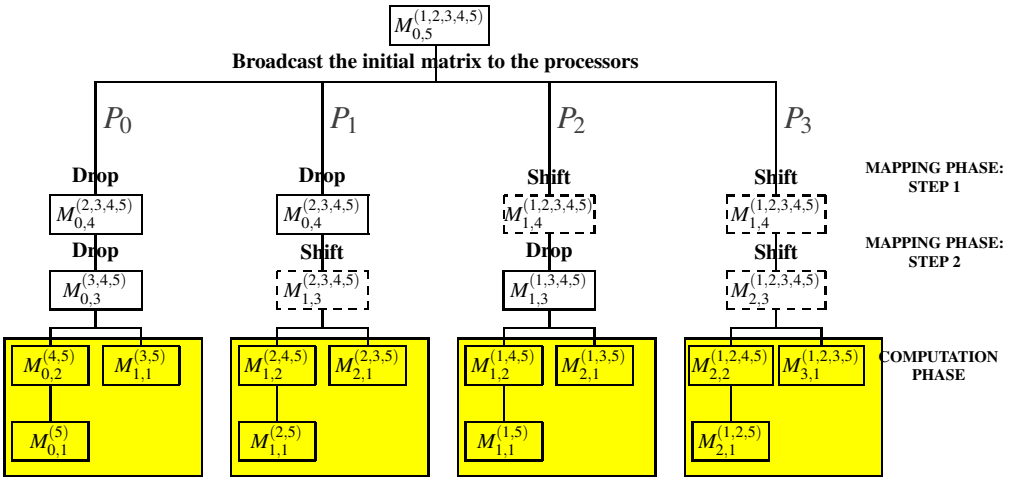


Figure 2.3: The parallel computation of the $T_{0,5}^{v}$ using 4 processors (P_0, P_1, P_2 and P_3) .

of this phase is dominated by the first processor which performs ρ Drop operations and is given by:

$$C_{\text{map}}(n,\rho) = \sum_{j=1}^{\rho} C_{Ret}(n-j) = t\rho\big(3n^2+6n-4-\rho(3n-\rho+3)\big)/6.$$

The Computation phase executes the Subtree routine (see Algor. 1) in line 12 of Algor. 2. This has complexity $C(n-\rho)$ – defined in (2.8) – and thus, the complexity of the PDCA is given by:

$$C_{\text{PDCA}}(n,\rho) = C(n-\rho) + C_{\text{map}}(n,\rho) \approx 3t2^{n-\rho}.$$

Clearly the Computation phase dominates the PDCA which has an exponential complexity. Increasing the number of variables in the model by one it will require the doubling of the processors in order to achieve the same execution time. Even thought, when compared to the serial DCA the PDCA has almost a linear speedup for $\rho < n-1$ and large n, that is,

$$\text{Speedup}(n,2^{\rho}) = C_{\text{DCA}}(n)/C_{\text{PDCA}}(n,\rho) \approx 2^{\rho}.$$

The theoretical measures of complexity do not take into account the overheads occurring during the implementation. These overheads are proportional to the dimension of the root matrix used in the Computation phase by each processor. Thus, if $i < j$, then the overheads of P_i are greater than that of P_j, where $i, j = 0, \ldots, 2^{\rho}-1$. This suggest that the load could be better balanced

Algorithm 2 The PDCA using $p = 2^{\rho}$ processors.

1: **initially do:**

- Compute the QRD of $A \in \mathbb{R}^{m \times n}$:

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix} \quad \text{and} \quad Q^T y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}$$

- Obtain the RSS of the models $(v_1), \ldots, (v_1 \cdots v_n)$
- Let $M^v_{k,\lambda} \equiv R$, where $v_i = i$ $(i = 1, \ldots, n)$, $k = 0$ and $\lambda = n$
- Broadcast $M^v_{k,\lambda}$ to the processors $P_0, \ldots P_{p-1}$

2: $r \longleftarrow$ rank of the processor
3: **each processor do:**
4: **for** $s = 1, \ldots, \rho$ **do**
5: **if** $((r \text{ div } 2^{\rho-s}) \bmod 2) = 0$ **then**
6: $M^v_{k,\lambda} \longleftarrow$ Apply Drop on $M^v_{k,\lambda}$
7: Obtain the RSS of the models $(v_1 \cdots v_{k+1}), \ldots, (v_1 \cdots v_{k+\lambda})$
8: **else**
9: $M^v_{k,\lambda} \longleftarrow$ Apply Shift on $M^v_{k,\lambda}$
10: **end if**
11: **end for**
12: **call** Subtree($M^v_{k,\lambda}$)
13: **end do**

by expanding the Computation phase into a larger number of subtrees of smaller complexity and allocating these efficiently to the processors. Consider the case where each of the 2^{ρ} subtrees obtained after the Mapping phase is divided into 2^{μ} smaller subtrees. Thus, in the Computation phase 2^g subtrees, say $T_0, T_1, \cdots, T_{2^g-1}$, need to be computed, where $g = \mu + \rho$. The *shift cyclic* method can be used to allocate the computations of the subtrees to the processors. This ad-hoc distribution method allocates the subtree T_ζ to the processor P_ξ, where $\zeta = 0, \cdots, 2^g - 1$ and $\xi = (\zeta - (\zeta \div 2^{\rho})) \bmod 2^{\rho}$. Figure 2.4 shows the allocation of the subtrees to the processors, where $g = 4$ and $\rho = 2$.

This method, called PDCA–2, has been implemented with $\mu = \rho$. Table 2.1 shows the execution

T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
P_0	P_1	P_2	P_3	P_3	P_0	P_1	P_2	P_2	P_3	P_0	P_1	P_1	P_2	P_3	P_0

Figure 2.4: The cyclic allocation of 2^g subtrees on 2^ρ processors, where $g = 4$ and $\rho = 2$.

times of each processor when the PDCA and PDCA–2 are used, where $n = 25$ and $\rho = 3$. It can be observed that with PDCA–2 the load is better balanced. Table 2.2 shows the execution time of the serial DCA, $C_{\text{PDCA}}(n,\rho)/t$, the theoretical efficiency, i.e. $\text{Speedup}(n,2^\rho)/2^\rho$, the execution time and the actual efficiency of PDCA and PDCA–2 for various values of n and ρ. The speedup was calculated with respect to the serial time of the DCA. Clearly the PDCA–2 outperforms the PDCA and obtains an efficiency close to the theoretical derived value of 1. Furthermore, Table 2.2 shows clearly the doubling of the execution time when the number of variables is increased by one.

Table 2.1: The execution times of each processor using the PDCA and PDCA–2 , where $n = 25$ and $2^\rho = 8$.

Processor	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
PDCA	89.70	94.56	94.48	99.00	94.46	99.01	98.96	102.89
PDCA–2	97.55	98.44	98.58	98.30	98.55	97.80	97.89	97.72

2.3.1 Variable updating of the regression model

The DCA and PDCA can be extended to solve the variable-updated regression model. Consider adding a new column, say z, to the regression model (2.1) for which the RSS of all subset regression models have already been obtained. In this case the RSS of all 2^n new subset models which comprise the new variable need to be computed. Let the new variable be added at the front of the exogenous matrix. The DCA when applied to the new model will generate the regression tree $T^v_{0,n+1}$ in which the leftmost child $T^{v(1)}_{0,n}$ corresponds to the regression tree derived by the DCA when applied to the original model. This is illustrated in Fig. 2.2, where now, $n = 4$. The root node of the regression tree derives from the QRD

$$\hat{Q}^T \begin{pmatrix} z_1 & R \\ \zeta & 0 \end{pmatrix} = M^v_{0,n+1}, \tag{2.9}$$

where

$$Q^T z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix} \quad \text{and} \quad \tilde{Q}^T z_2 = \begin{pmatrix} \zeta \\ 0 \end{pmatrix} \begin{matrix} 1 \\ m-n-1 \end{matrix} .$$

Table 2.2: Theoretical complexity and execution times of the DCA, PDCA and PDCA–2.

		DCA	Theoretical PDCA		PDCA		PDCA–2	
n	2^ρ	Serial	Complx./t	Efficiency	Time	Efficiency	Time	Efficiency
15	1	0.600	98151	1.00	0.603	0.99	0.610	0.98
15	2		49135	0.99	0.313	0.97	0.310	0.98
15	4		24679	0.99	0.160	0.94	0.157	0.96
15	8		12496	0.98	0.080	0.94	0.080	0.94
19	1	10.49	1572633	1.00	10.60	0.99	10.54	0.99
19	2		786411	0.99	5.41	0.97	5.31	0.99
19	4		393385	0.99	2.80	0.94	2.66	0.98
19	8		196948	0.99	1.40	0.94	1.37	0.96
20	1	21.52	3145475	1.00	21.52	1.00	21.53	0.99
20	2		1572842	0.99	11.03	0.98	10.81	0.99
20	4		786620	0.99	5.63	0.96	5.46	0.98
20	8		393594	0.99	2.88	0.93	2.78	0.97
21	1	43.49	6291180	1.00	43.59	0.99	43.55	0.99
21	2		3145705	0.99	22.54	0.96	22.03	0.98
21	4		1573072	0.99	11.41	0.95	11.16	0.96
21	8		786850	0.99	5.83	0.93	5.47	0.98
25	1	757.28	100662900	1.00	759.89	0.99	773.71	0.98
25	2		50331621	0.99	389.56	0.97	381.54	0.99
25	4		25166122	0.99	201.12	0.94	190.97	0.99
25	8		12583510	0.99	102.89	0.92	98.55	0.97

Here $\tilde{Q}$ and $\hat{Q}$ are orthogonal, $M^v_{0,n+1}$ is upper triangular of order $n+1$ and $\zeta^2 = \|z_2\|^2$. The QRD (2.9) is computed by a sequence of n Givens rotations between adjacent planes that annihilate from bottom to top the elements of $\begin{pmatrix} z_1^T & \zeta^T \end{pmatrix}^T$ except from the first one. Without taking into account these Givens rotations, the complexity of the DCA to solve the updated model is half of that needed to solve the model afresh. The PDCA solves the updated model $M^v_{0,n+1}$ by generating $T^v_{1,n}$. The parallel complexity in this case is given by $C_{PDCA}(n,\rho)$. Notice that if the new transformed variable $Q^T z$ is added at the end of the matrix, then $\hat{Q} = I_{n+1}$, i.e. the QRD (2.9) does not need to be computed. However, this advantage is offset by the non-trivial derivation of the new RSS from the regression tree generated using the DCA.

2.4 The general linear and seemingly unrelated regression models

The DCA can be employed to compute all possible subset models when the dispersion of the the noise vector ε in (2.1) is non-spherical. The General Linear Model (GLM) is the regression model (2.1), where ε has zero mean, variance-covariance matrix $\sigma^2\Omega$, σ is a non-zero scalar and Ω is non-negative definite. Let Ω be non-singular with Cholesky factorization $\Omega = BB^T$, where B is upper triangular. The GLM can be formulated as the Generalized Linear Least Squares Problem (GLLSP):

$$\underset{\beta,\upsilon}{\operatorname{argmin}} \|\upsilon\|^2 \quad \text{subject to} \quad y = A\beta + B\upsilon, \tag{2.10}$$

where $\|\cdot\|$ denotes the Eucledian norm and υ is a random m–element vector with zero mean and variance-covariance matrix $\sigma^2 I_m$. Consider the Generalized QRD (GQRD) of A and B:

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix} \quad \text{and} \quad Q^T B P = W \equiv \begin{pmatrix} W_{11} & W_{12} \\ 0 & W_{22} \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}, \tag{2.11}$$

where W and R are upper triangular and $Q, P \in \mathbb{R}^{m\times m}$ are orthogonal. Let

$$Q^T y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix} \quad \text{and} \quad P^T \upsilon = \begin{pmatrix} \upsilon_1 \\ \upsilon_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}.$$

From this it follows that the GLLSP is reduced to

$$\underset{\beta,\upsilon_1}{\operatorname{argmin}} \|\upsilon_1\|^2 \quad \text{subject to} \quad \tilde{y} = R\beta + W_{11}\upsilon_1, \tag{2.12}$$

where $\upsilon_2 = W_{22}^{-1} y_2$ and $\tilde{y} = y_1 - W_{12}\upsilon_2$. Thus, $\upsilon_1 = 0$, the LS estimator $\hat{\beta} = R^{-1}\tilde{y}$ and the RSS is computed by $\|\upsilon_2\|^2$.

Notice that the modified GLM (2.3) is equivalent to the GLLSP

$$\underset{\beta,\upsilon_1}{\operatorname{argmin}} \|\upsilon_1\|^2 \quad \text{subject to} \quad \tilde{y} = RS\beta_{(S)} + W_{11}\upsilon_1. \tag{2.13}$$

For the solution of (2.13) the GQRD of RS and W_{11} is required. That is, the QRD (2.4) and the RQD $(Q_{(S)}^T W_{11})P_{(S)} = \tilde{W}_{11}$, where $\tilde{W}_{11}$ is upper triangular. Within the context of the DCA the matrix RS is the single column-downdated R and $Q_{(S)}^T$ is a product of Givens rotations. In this case, a Givens rotation, say G_i, when applied from the left of $\begin{pmatrix} RS & W_{11} \end{pmatrix}$ annihilates and fills-in the elements $(i, i-1)$ of RS and W_{11}, respectively. A Givens rotation, say P_i, can be applied from the

right of $G_i W_{11}$ to annihilate the fill-in, that is, $G_i W_{11} P_i$ is upper triangular [64, 73, 98, 99]. Thus, $Q_{(S)}^T$ and $P_{(S)}$ are the products of the left and right Givens rotations, respectively. Now, writing

$$Q_{(S)}^T RS = \begin{pmatrix} R_{(S)} \\ 0 \end{pmatrix} \begin{matrix} n-1 \\ 1 \end{matrix}, \quad Q_{(S)}^T \tilde{y} = \begin{pmatrix} \tilde{y}_{(S)} \\ \eta_{(S)} \end{pmatrix} \begin{matrix} n-1 \\ 1 \end{matrix}, \quad P_{(S)}^T \upsilon_1 = \begin{pmatrix} \upsilon_{(S)} \\ f_{(S)} \end{pmatrix} \begin{matrix} n-1 \\ 1 \end{matrix}$$

and

$$Q_{(S)}^T W_{11} P_{(S)} = W_{(S)} \equiv \begin{array}{c} \begin{matrix} n-1 & \quad 1 \end{matrix} \\ \begin{pmatrix} W_{11}^* & w_{(S)} \\ 0 & \omega_{(S)} \end{pmatrix} \begin{matrix} n-1 \\ 1 \end{matrix} \end{array},$$

it follows that the RSS and solution of the modified GLM is given by $\mathrm{RSS}_{(S)} = \mathrm{RSS} + \hat{f}_{(S)}^2$ and $\hat{\beta}_{(S)} = R_{(S)}^{-1} \tilde{y}_{(S)}^*$, where $\hat{f}_{(S)} = \eta_{(S)} / \omega_{(S)}$ and $\tilde{y}_{(S)}^* = \tilde{y}_{(S)} - w_{(S)} \hat{f}_{(S)}$. Notice that if $S = \begin{pmatrix} I_{\tilde{n}} & 0^T \end{pmatrix}^T$, then $Q_{(S)} = P_{(S)} = I_n$ and $\mathrm{RSS}_{(S)} = \mathrm{RSS} + \|\hat{f}_{(S)}\|^2$, where now $\hat{f}_{(S)} = \omega_{(S)}^{-1} \eta_{(S)}$, $\eta_{(S)} \in \mathbb{R}^{n-\tilde{n}}$ and $\omega_{(S)} \in \mathbb{R}^{(n-\tilde{n}) \times (n-\tilde{n})}$.

Let $M_{k,\lambda+1}^v$ denote the triangular factor $R_{(S)}$ and its corresponding $W_{(S)}$ matrix. The Drop applied to this model derives $M_{k,\lambda}^v$ using λ left and right Givens rotations. The complexity of the jth $(j = 1, \ldots, \lambda)$ left and right rotation are given by $(2j+2)t$ and $(k+\lambda+3-j)t$, respectively. Thus, the complexities of deriving $M_{k,\lambda}^v$ from $M_{k,\lambda+1}^v$, and the regression tree $T_{k,\lambda}^v$ are given, respectively, by:

$$C_{\mathrm{GRet}}(k,\lambda) = t \sum_{j=1}^{\lambda} (j+k+\lambda+5) = t\lambda(2k+3\lambda+11)/2 \tag{2.14}$$

and

$$\begin{aligned} C(k,\lambda) &= \sum_{i=1}^{\lambda-1} (C_{\mathrm{GRet}}(k+i-1, \lambda-i-1) + C(k+i-1, \lambda-i-1)) \\ &= C_{\mathrm{GRet}}(k, \lambda-1) + C(k, \lambda-1) + C(k+1, \lambda-1) \\ &= \sum_{i=1}^{\lambda-1} \Big(\sum_{j=1}^{i} \binom{i-1}{j-1} C_{\mathrm{GRet}}(k+j-1, \lambda-i) \Big) + \sum_{i=1}^{\lambda} \binom{\lambda-1}{i-1} C(k+i-1, 1). \end{aligned}$$

Here $\lambda > 1$, $C(k,1) = 0$ and $\binom{m}{n} = m!/n!(m-n)!$. Using $C(k,1) = 0$ the latter becomes

$$\begin{aligned} C(k,\lambda) &= \sum_{i=1}^{\lambda-1} \Big(\sum_{j=1}^{i} \binom{i-1}{j-1} C_{\mathrm{GRet}}(k+j-1, \lambda-i) \Big) \\ &= t\Big(2^{\lambda}(\lambda+2k+16) - (\lambda+1)(3\lambda+2k+12) - 4 \Big)/2. \end{aligned} \tag{2.15}$$

Thus, the complexity of the DCA when employed to the GLM (denoted by GDCA) is given by $C(0,n)$ which has $O(n2^n)$, that is,

$$C_{\text{GDCA}}(n) = C(0,n) \equiv t\big(2^n(n+16) - (n+1)(3n+12) - 4\big)/2. \tag{2.16}$$

Now, consider the adaptation of the PDCA in the case of the GLM and call this GPDCA. In the Mapping phase of the GPDCA the last processor performs only Shifts (no computations), while the first processor performs only Drops and has the highest complexity which is given by:

$$C_{\text{GPm}}(\rho,n) = \sum_{j=1}^{\rho} C_{\text{GRet}}(0,n-j) = t\rho\big(3n^2+8n-5-\rho(3n-\rho+4)\big)/2.$$

The complexity of each processor during the Computation phase is given by $C(k,n-\rho)$, where k denotes the number of Shifts performed in the Mapping phase. Thus, the last processor $P_{2^\rho-1}$ which performs the maximum of ρ Shifts has in this phase the highest complexity $C(\rho,n-\rho)$. For $n \gg \rho$ the computations during the Mapping phase are negligible when compared to the exponential complexity of the Computation phase. In this case, the complexity of the GPDCA will be given by $C(\rho,n-\rho)$ which is of $O((n+\rho)2^{n-\rho})$. Specifically

$$\begin{aligned} C_{\text{GPDCA}}(n,\rho) &= C(\rho,n-\rho) \\ &\equiv t\big(2^{n-\rho}(n+\rho+16) - 3(n+1)(n+4) + \rho(4n-\rho+13) - 4\big)/2. \end{aligned} \tag{2.17}$$

The speedup of the GPDCA is given by:

$$\text{Speedup}_{\text{G}}(n,2^\rho) = C_{\text{GDCA}}(n)/C_{\text{GPDCA}}(n,\rho) \approx n2^\rho/(n+\rho). \tag{2.18}$$

The GPDCA does not achieve a linear speedup as in the case of the theoretical PDCA. This is due to the different complexities of the subtrees allocated to each processor in the Computation phase. A better load balancing can be achieved using the same approach that has been employed by the PDCA–2. The efficiency of this strategy (GPDCA–2) compared to that of GPDCA is illustrated by the Tables 2.3 and 2.4. Notice that the efficiency obtained by the GPDCA–2 outperforms the theoretical one of the GPDCA.

2.4.1 Seemingly unrelated regression models

A special case of the GLM is the seemingly unrelated regression model. In this model the exogenous matrix A has the block-diagonal structure $\oplus_i^G A^{(i)} = \text{diag}(A^{(1)},\ldots,A^{(G)})$, $A^{(i)} \in \mathbb{R}^{m\times n_i}$ $(i=1,\ldots,G)$, the variance-covariance matrix of the disturbances is given by $\Sigma\otimes I_m$ and $\Sigma \in \mathbb{R}^{G\times G}$

Table 2.3: The execution times of each processor using the GPDCA and GPDCA–2 , where $n = 25$ and $2^{\rho} = 8$.

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
GPDCA	219.00	223.23	223.50	240.23	223.64	240.24	239.85	246.68
GPDCA–2	227.97	229.37	228.45	229.33	229.09	229.76	229.08	230.96

Table 2.4: Theoretical complexity and execution times of the GDCA, GPDCA and GPDCA–2.

		GDCA	Theoretical GPDCA		GPDCA		GPDCA–2	
n	2^{ρ}	Serial	Complx./t	Efficiency	Time	Efficiency	Time	Efficiency
15	1	1.430	507446	1.00	1.448	0.99	1.450	0.99
15	2		261722	0.96	0.748	0.96	0.730	0.98
15	4		134781	0.94	0.392	0.91	0.370	0.97
15	8		69279	0.91	0.224	0.80	0.184	0.97
19	1	25.02	9174348	1.00	25.14	0.99	25.20	0.99
19	2		4717944	0.97	13.04	0.96	12.75	0.98
19	4		2424227	0.95	6.98	0.90	6.44	0.97
19	8		1244621	0.92	3.57	0.88	3.16	0.98
20	1	50.51	18873610	1.00	50.62	0.99	50.66	0.99
20	2		9698616	0.97	26.11	0.97	25.68	0.98
20	4		4980069	0.94	13.81	0.91	12.94	0.98
20	8		2555281	0.92	6.90	0.91	6.44	0.98
21	1	103.19	38796485	1.00	104.22	0.99	103.50	0.99
21	2		19922165	0.97	54.46	0.95	52.02	0.98
21	4		10222884	0.95	27.75	0.93	26.74	0.96
21	8		5242194	0.93	14.57	0.88	13.18	0.98
25	1	1802.23	687864700	1.00	1815.98	0.99	1809.64	0.99
25	2		352320400	0.98	932.33	0.97	909.20	0.99
25	4		180354000	0.95	481.17	0.94	459.32	0.98
25	8		92273720	0.93	246.68	0.92	230.96	0.98

is positive-definite [63, 113, 114, 126]. Here $\otimes$ and $\oplus$ are the Kronecker and direct sum of matrices operators [103]. Thus, in (2.12), and consequently (2.13), the upper triangular matrix $R = \text{diag}(R^{(1)}, \ldots, R^{(G)})$, $R^{(i)} \in \mathbb{R}^{n_i \times n_i}$, $W_{11} \in \mathbb{R}^{n^{(G)} \times n^{(G)}}$ and $n^{(i)} = \sum_{j=1}^{i} n_j$, $(i = 1, \ldots, G)$. Let $RS_{i,j}$ denote the matrix R after deleting the jth column from $R^{(i)}$, where $i = 1, \ldots, G$ and $j = 1, \ldots, n_i$.

Furthermore, let $\tilde{R}_j^{(i)}$ denote $R^{(i)}$ without its jth column and W_{11} be partitioned as

$$W_{11} = \begin{array}{c} \begin{matrix} n_1 & n_2 & \cdots & n_G \end{matrix} \\ \begin{pmatrix} \widetilde{W}_{1,1} & \widetilde{W}_{1,2} & \cdots & \widetilde{W}_{1,G} \\ \widetilde{W}_{2,1} & \widetilde{W}_{2,2} & \cdots & \widetilde{W}_{2,G} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{W}_{G,1} & \widetilde{W}_{G,2} & \cdots & \widetilde{W}_{G,G} \end{pmatrix} \end{array} \begin{matrix} n_1 \\ n_2 \\ \vdots \\ n_G \end{matrix}, \tag{2.19}$$

where $i, j = 1, \ldots, G$. The re-triangularization of $RS_{i,j}$ is obtained in two stages. In the first stage, as in the case of the GLM, $n_i - j$ rotations between adjacent planes are applied from the left and right of $\begin{pmatrix} \tilde{R}_j^{(i)} & \widetilde{W}_{i,i} & \ldots & \widetilde{W}_{i,G} \end{pmatrix}$ and $\begin{pmatrix} \widetilde{W}_{i,1}^T & \ldots \widetilde{W}_{i,i}^T \end{pmatrix}^T$, respectively, in order to re-triangularize $\tilde{R}_j^{(i)}$ and $\widetilde{W}_{i,1}$. Let $\widehat{Q}_{(S)}^T$ and $\widehat{P}_{(S)}$ denote the products of the left and right Givens rotations, respectively, and Π^T be the permutation matrix

$$\Pi^T = \begin{pmatrix} I_{n^{(i)}-1} & 0 & 0 \\ 0 & 0 & I_{n^{(G,i)}-1} \\ 0 & 1 & 0 \end{pmatrix},$$

where $n^{(G,i)} = n^{(G)} - n^{(i)}$. Thus,

$$\Pi^T \widehat{Q}_{(S)}^T RS_{i,j} = \begin{pmatrix} R_{(S)} \\ 0 \end{pmatrix} \begin{matrix} n^{(G)} - 1 \\ 1 \end{matrix},$$

where $R_{(S)} = \text{diag}(R_{(S)}^{(1)}, \ldots, R_{(S)}^{(G)})$, $R_{(S)}^{(i)}$ is upper triangular and $R_{(S)}^{(q)} \equiv R^{(q)}$ for $q = 1, \cdots, G$ and $q \neq i$.

The second stage computes the RQD $W_{(S)} = \widehat{W}\tilde{P}_{(S)}$, where $\widehat{W} = \Pi^T \widehat{Q}_{(S)}^T W_{11} \widehat{P}_{(S)}$, $W_{(S)}$ and $\widehat{Q}_{(S)}^T W_{11} \widehat{P}_{(S)}$ are upper triangular and $\tilde{P}_{(S)}$ is the product of $n^{(G,i)}$ Givens rotations. The μth ($\mu = 1, \ldots, n^{(G,i)}$) rotation, say $\tilde{P}_\mu$, annihilates the $(n^{(i)} + \mu - 1)$th element of the last row of $\widehat{W}$ by rotating adjacent planes. This is illustrated in Fig. 2.5, where $G = 3$, $n_1 = 4$, $n_2 = 6$, $n_3 = 3$, $i = 2$ and $j = 3$. An arc denotes the affected columns during the rotation.

The complexity of retriangularizing $RS_{i,j}$ and that of generating all 2^{n_i} possible models by deleting one or more column from the ith block of the SUR model are given respectively, by:

$$C_{\text{SRet}}(i, j) = t\big(j^2 + j(2n^{(G)} + 7) + n^{(G,i)}(n^{(G)} + n^{(i)} - 5) - 2n^{(i)} - 8\big)/2$$

and

$$C_{\text{SGenB}}(i) \approx t2^{n_i}\big((n^{(G,i)} + 2)(n^{(G)} + n^{(i-1)} + 2) + 28\big).$$

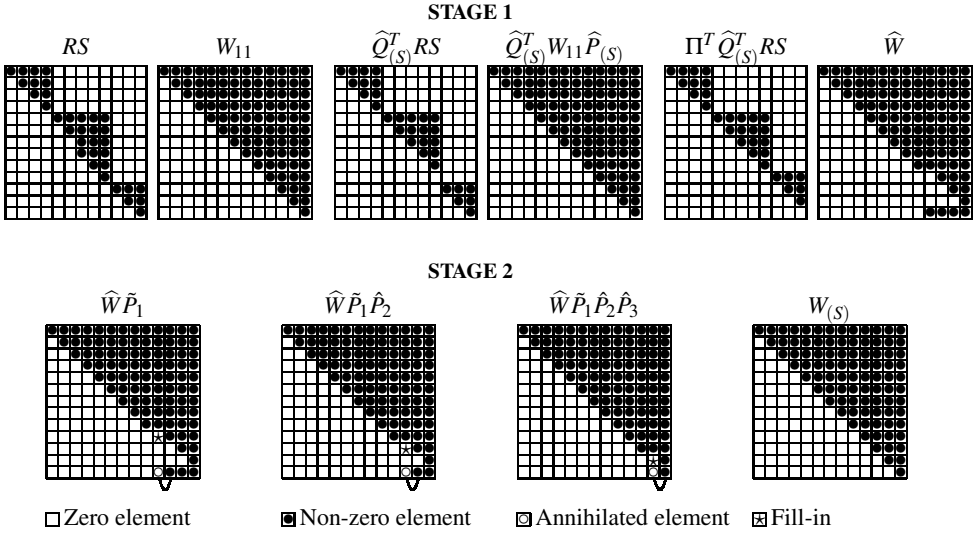


Figure 2.5: The two stages of solving a seemingly unrelated regression model after deleting a variable.

Thus, the complexity of the DCA applied to seemingly unrelated regression models (SDCA) is given by

$$\begin{aligned} C_{\text{SDCA}}(G,N) &= \sum_{i=1}^{G-1}\left(\sum_{j=0}^{n^{(i-1)}}\binom{n^{(i-1)}}{j}C_{\text{SGenB}}(i+1)\right)+\sum_{j=0}^{n^{(G-1)}}\binom{n^{(G-1)}}{j}C_{\text{SGenB}}(G) \\ &\approx t\sum_{i=1}^{G-1}\left(2^{n^{(i)}+1}((n^{(G,i)}+1)(n^{(G)}+2)+28)\right)+t2^{n^{(G)}+1}(n^{(G)}+11). \end{aligned}$$

2.5 Conclusions

A parallel algorithm has been developed to compute the RSS of all possible subset models of the standard regression model. The algorithm (PDCA) is a parallelization of the DCA proposed in [111, 112]. The properties of the regression tree generated by the DCA have been studied in order to derive an efficient load-balancing strategy. The PDCA uses a single–program multiple–data paradigm and requires no inter-processor communication. The theoretical measures of complexity had showed that the PDCA has a linear speedup. Experimental results on a shared memory machine indicated that overheads cause a non-perfect load balancing among the processors. This resulted in the efficiency of the PDCA to divert from the theoretical one. A second algorithm (PDCA–2) which obtains a better load balancing and an efficiency close to the maximum theoretical value of

one has been designed.

The DCA and PDCA have been extended to GDCA and GPDCA, respectively, in order to compute the RSS of all possible subset models of the general linear model. In this case the theoretical complexity has shown that the speedup obtained by the GPDCA is lower than that obtained by the PDCA. The main reason for this is that unlike the PDCA, the GPDCA allocates to the processors subtrees of different complexity. However, GPDCA–2, which is an extension of PDCA–2, has achieved an efficiency closed to one.

The adaptation of the serial GDCA to solve the seemingly unrelated regression model has also been developed. The employment of this algorithm and its parallelization for estimating all subsets of a seemingly unrelated regression model arising in some vector autoregressive processes are currently considered. In this case $R^{(1)} = \cdots = R^{(G)}$ have dimension $n \times n$ and in (2.19) $W_{11} = B \otimes I_n$, where $B \in \mathbb{R}^{G \times G}$.

The proposed algorithms will be inefficient for heterogeneous parallel systems. In such platforms a dynamic distribution, such as that obtained by task-farming, can yield a better performance. Furthermore, it will be computationally not feasible to consider all models when the number of variables, i.e. n, is very big. In such cases a parallel procedure which computes the best subset without examining all the possible subsets needs to be developed. A possibility is to use a branch and bound algorithm based on some criteria (statistics), or some other heuristic approaches [36, 38, 53, 54, 116]. Currently these non-trivial parallel strategies, the extension of the existing algorithms to other linear models (e.g. mixed and simultaneous equation models) and the adaptation of the PDCA to multiple-row diagnostics are investigated [10].

Chapter 3

Branch-and-bound algorithms for computing the best-subset regression models

Abstract:

An efficient branch-and-bound algorithm for computing the best-subset regression models is proposed. The algorithm avoids the computation of the whole regression tree that generates all possible subset models. Specifically, it is formally shown that if the branch-and-bound test holds, then the current subtree together with its right-hand side subtrees are cut. This reduces significantly the computational burden of the proposed algorithm when compared to an existing leaps-and-bounds method. The criteria used in identifying the best subsets are based on the residual sum of squares (RSS). Various statistics associated with model selection are function of RSS. The algorithm is based on orthogonal transformations and outperforms by $O(n^3)$ the existing leaps-and-bounds strategy which generates two trees. Strategies and heuristics which improve the computational performance of the proposed algorithm are investigated. A computational efficient heuristic version of the branch-and-bound algorithm which decides to cut subtrees using a tolerance parameter is proposed. The heuristic algorithm derives models closed to the best ones. However, it is shown analytically that the relative error of the RSS of the computed subsets is smaller than the value of

[1]This chapter is a reprint of the paper: C. Gatu and E.J. Kontoghiorghes. Branch-and-bound algorithms for computing the best-subset regression models. *Journal of Computational and Graphical Statistics*, 2003. (Submitted).

the tolerance parameter which lies between zero and one. Computational results and experiments on random and real data are presented and analyzed.

3.1 Introduction

A common problem in statistical model selection is the computation of the best-subset regression models [27, 36, 55, 56, 95]. Consider the standard regression model

$$y = A\beta + \varepsilon, \tag{3.1}$$

where $y \in \mathbb{R}^m$ is the dependent variable vector, $A \in \mathbb{R}^{m\times n}$ is the exogenous data matrix of full column rank, $\beta \in \mathbb{R}^n$ is the coefficient vector and $\varepsilon \in \mathbb{R}^m$ is the noise vector. It is assumed that ε has zero mean and variance-covariance matrix $\sigma^2 I_m$.

One approach to search for the best models is the straight-forward method of generating all $2^n - 1$ possible sub-models [54, 79, 94]. As n increases the number of models to be computed increases exponentially. Efficient strategies for moving from one model to another with minimum computational cost have been previously proposed [24, 35, 112]. Further improvements are possible by exploiting the structural properties of the problem and using parallel strategies [42]. Still, for a large number of variables the consideration of all models will be computational infeasible. In such cases procedures that compute the best sub-models without investigating all possible subsets need to be considered. Existing methods based on a leaps-and-bounds strategy derive the best models for each number of variables by generating two trees [36]. The first one, the bounds tree, provides half of the models (2^{n-1}) that does not include the last variable. For obtaining a child from its parent node a Gaussian elimination step is used [47]. The construction of this tree aims to provide good bounds in early stages of the execution of the algorithm. The second tree, called the regressions tree, provides the other half of the models $(2^{n-1} - 1)$ that includes the last variable. The models are generated by moving from one node to another. A matrix inverse is computed each time a model is derived. The problem of finding the best-subset model of given size d, where $1 \leq d \leq n$, has been previously considered. The use of the QR decomposition (QRD) instead of inverse matrices in this context has also been discussed [95].

A branch-and-bound algorithm (hereafter abbreviated to BBA) that generates the best regression models corresponding to each number of variables is proposed [80, 83]. The algorithm avoids computing all-possible-subset models by exploiting the properties of a regression tree that consists

of 2^{n-1} nodes [42]. Each of these nodes provides a number of regression models. The computational tool used by the BBA is the QRD. Specifically, it employs Givens rotations to compute the QRD of a triangular matrix after deleting a column [45, 63]. Theoretical measures of complexity are derived in order to compare the proposed BBA against existing methods. Several strategies and heuristics that improve the computational performance of the BBA are presented. The algorithms are implemented using FORTRAN, BLAS and LAPACK on a Sun-Fire-280R of two SPARC-v9 processors with a clock speed of 750MHz and 4Gb of RAM.

The next section briefly discusses the use of the QRD to compute the least-squares and residual sum of squares of the regression model. The regression tree that generates all possible models is introduced. Section 3 presents the BBA and various strategies for improving the efficiency and the applicability of the algorithm. Various heuristics that allow the investigation of large-scale models are also considered. Experimental results are analyzed in section 4. Conclusions and notions for future work are presented and discussed in the final section.

3.2 Computing all possible sub-models

The QRD of the exogenous matrix A in (3.1) is given by

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}, \tag{3.2}$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $R \in \mathbb{R}^{n\times n}$ is upper triangular and non-singular. Let

$$Q^T y = \tilde{y} = \begin{pmatrix} \tilde{y}_1 \\ \tilde{y}_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}. \tag{3.3}$$

The least-squares estimator of β is given by

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \|y - A\beta\|^2 = \underset{\beta}{\operatorname{argmin}} \|Q^T(y - A\beta)\|^2 = R^{-1}\tilde{y}_1,$$

where $\|\cdot\|$ denotes the Euclidian norm. The residual sum of squares (RSS) is computed by

$$\mathrm{RSS} = \|y - A\hat{\beta}\|^2 = \|Q^T(y - A\hat{\beta})\|^2 = \|\tilde{y}_2\|^2.$$

Let S denote a selection matrix, which comprises d columns of the $n \times n$ identity matrix I_n. Consider the modified regression model

$$y = A_{(S)}\beta_{(S)} + \varepsilon, \tag{3.4}$$

where $A_{(S)} = AS \in \mathbb{R}^{m\times d}$ and $\beta_{(S)} = S^T\beta \in \mathbb{R}^d$. The least-squares estimator of $\beta_{(S)}$, i.e., $\hat{\beta}_{(S)}$, is obtained from the solution of

$$\underset{\beta_{(S)}}{\operatorname{argmin}}\|y - A_{(S)}\beta_{(S)}\|^2 = \underset{\beta_{(S)}}{\operatorname{argmin}}\|Q^T(y - AS\beta_{(S)})\|^2 = \underset{\beta_{(S)}}{\operatorname{argmin}}\|\tilde{y}_1 - RS\beta_{(S)}\|^2.$$

Now, computing the QRD of RS

$$Q_{(S)}^T RS = \begin{pmatrix} R_{(S)} \\ 0 \end{pmatrix} \begin{matrix} d \\ n-d \end{matrix} \quad \text{and} \quad Q_{(S)}^T \tilde{y}_1 = \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \end{pmatrix} \begin{matrix} d \\ n-d \end{matrix}, \tag{3.5}$$

the least-squares estimator and the residual sum of squares of the modified model (3.4) are given by $\hat{\beta}_{(S)} = R_{(S)}^{-1}\hat{y}_1$ and $\mathrm{RSS}_{(S)} = \mathrm{RSS} + \hat{y}_2^T\hat{y}_2$, respectively. The factorization (3.5) is equivalent to the re-triangularization of R in (3.2) after deleting columns [60, 61, 63, 65, 69, 71]. Notice that if S consists of the first d $(d = 1,\ldots,n)$ columns of I_n, then $Q_{(S)}^T = I_n$ and $R_{(S)}$ is the leading $d \times d$ sub-matrix of RS.

Now, let $V = [v_1, v_2, \ldots, v_d]$ denote the $d = |V|$ indices of the selected columns (variables) included in the sub-matrix $R_{(S)}$ (model). The sub-models corresponding to the sub-sets $[v_1]$, $[v_1, v_2]$, $\cdots$, $[v_1, v_2, \ldots, v_d]$ are immediately available. The problem is equivalent to computing all $2^n - 1$ subsets of $V = [v_1, v_2, \ldots, v_n]$, where $v_i = i$ $(i = 1,\ldots,n)$. In solving this problem, a function called Drop will be used. This function re-triangularizes a triangular matrix after deleting a column. That is,

$$\mathrm{Drop}(V, i) = [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n], \quad \text{where } i = 1,\ldots,n-1.$$

The re-triangularization performed by Drop is obtained by applying a sequence of Givens rotations using $(n-i)(n-i+3)/2$ floating point operations (flops) [45, 63]. The application of the Givens rotations on the modified response vector y is not discussed but is included in the complexity analysis. The rotations are between adjacent planes. Figure 3.1 shows the re-triangularization of a 6×6 matrix after deleting the 3rd column. A rotation G_j $(j = 4, 5, 6)$ annihilates the element at position $(j, j-1)$. The rows affected by the rotations are indicated by an arc.

A formal and detailed description of the regression tree (hereafter abbreviated to RT) that generates all subset models has been given in [42]. Here the basic concepts using a more convenient notation are introduced. A node of the RT is a tuple (V, k), where V is the set of indices and k $(k = 0,\ldots,|V|-1)$ indicates that the children of this node will include the first k variables. If $V = [v_1, v_2, \ldots, v_n]$, then the RT $T(V, k)$ is an $(n-1)$–tree having as root the node (V, k), where $k = 0,\ldots,n-1$. The children are defined by the tuples $(\mathrm{Drop}(V, i), i-1)$ for $i = k+1,\ldots,n-1$.

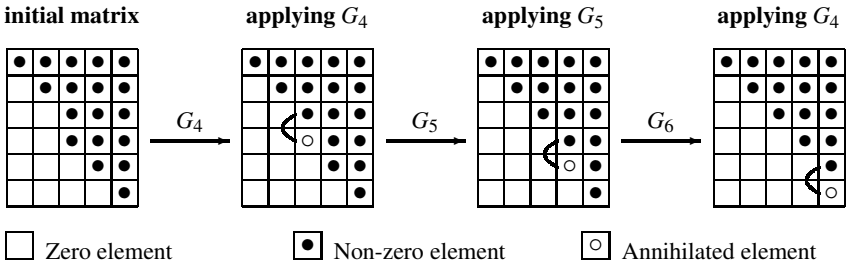


Figure 3.1: Re-triangularization of an $n \times n$ upper triangular matrix after deleting the ith column using Givens rotations, where $n = 6$ and $i = 3$.

Formally this can be expressed recursively as

$$T(V,k) = \begin{cases} (V,k) & \text{if} \quad k = n-1, \\ \big((V,k), T(\text{Drop}(V,k+1),k), \cdots, T(\text{Drop}(V,n-1),n-2)\big) & \text{if} \quad k < n-2. \end{cases}$$

The number of nodes in the sub-tree $T(\text{Drop}(V,i),i-1)$ is given by $\delta_i = 2^{n-i-1}$ and $\delta_i = 2\delta_{i+1}$, where $i = 1,\ldots,n-1$ [42].

Computing all possible subset regressions of a model having n variables is equivalent to generating $T(V,0)$, where $V = [1,2,\ldots,n]$. Generally, the complexity –in terms of flops– of generating $T(V,k)$ is given by

$$18 \cdot 2^{|V|-k} - 3(|V|-k+2)(|V|-k+3). \tag{3.6}$$

Figure 3.2 shows $T(V,k)$ together with the sub-models generated from each node, where $V = [1,2,3,4,5]$ and $k = 0$. A sub-model is denoted by a sequence of numerals that correspond to the indices of variables.

3.3 The Branch-and-Bound Algorithm (BBA)

Various statistics, such as R^2 and C_p, are associated with model selection [53, 87, 88, 94, 108]. Many of these criteria are monotone functions of the RSS for subsets with the same number of variables. That is, if $\text{RSS}(V_1) \geq \text{RSS}(V_2)$, then $f(\text{RSS}(V_1)) \geq f(\text{RSS}(V_2))$, where V_1 and V_2 are sets of independent variables, $|V_1| = |V_2|$, $\text{RSS}(V_i)$ is the RSS of the model comprising the variables in V_i $(i = 1,2)$ and f is the statistic R^2 or C_p. In this context, the problem of finding the best sub-models is reduced to one of finding the sub-models of size d with the minimum RSS, where

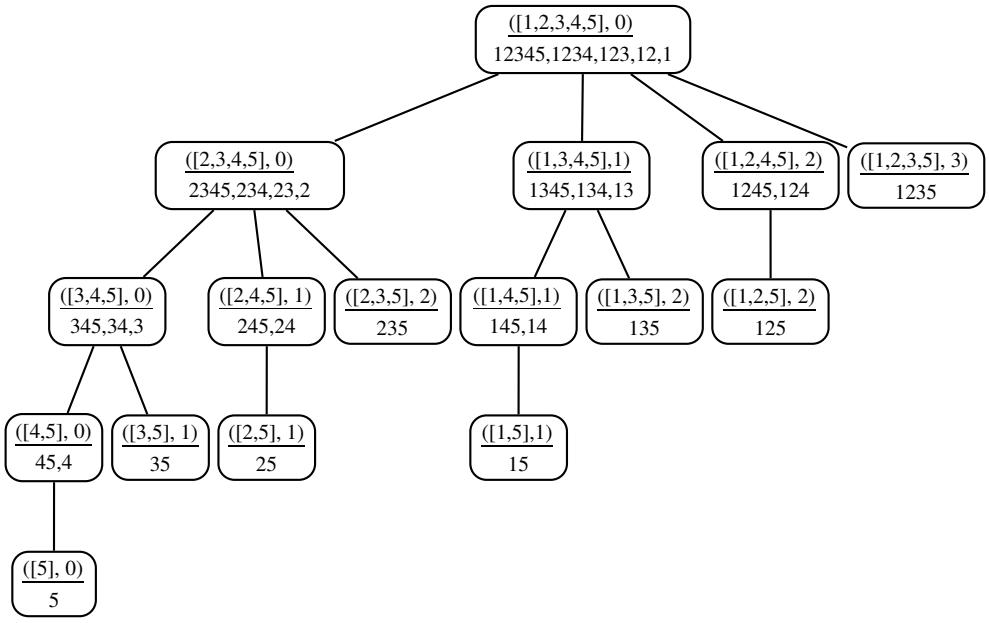


Figure 3.2: The regression tree $T(V,k)$, where $V = [1,2,3,4,5]$ and $k = 0$.

$d = 1,\ldots,n$. All methods which find the best-subset models without computing all regressions are based on the fundamental property

$$\mathrm{RSS}(V_1) \geq \mathrm{RSS}(V_2), \quad \text{where } V_1 \subseteq V_2. \tag{3.7}$$

That is, deleting variables from a regression increases the RSS of the modified model.

The number of evaluated sub-sets when searching for the best models can be restricted by using (3.7). Let $V = [1,2,\ldots,n]$ and $r_j^{(g)}$ ($j = 1,\ldots,n$ and $g = 1,\ldots,2^{n-1}$) denote the current minimum value of the RSS for the models with j variables after g nodes of the RT $T(V,0)$ have been derived. Notice that the generating order of the children nodes is not important. The root node $(V,0)$ provides the values $r_1^{(1)}, r_2^{(1)}, \ldots, r_n^{(1)}$ from the RSS of the available sub-sets $[1],[1,2],\cdots,[1,2,\ldots,n]$, respectively. Once the gth node $([v_1,\ldots,v_{i-1},v_{i+1},\ldots,v_d],i-1)$ has been derived, the sub-sets $[v_1,\ldots,v_{i-1},v_{i+1}]$, $\cdots$, $[v_1,\ldots,v_{i-1},v_{i+1},\ldots,v_d]$ become available and the $r_j^{(g)}$ $(j = i,\ldots,d-1)$ are updated. For $j = 1,\ldots,i-1$ and $j = d,\ldots,n$, the minimum values of the RSS are not modified and $r_j^{(g)} = r_j^{(g-1)}$. After the whole RT $T(V,0)$ has been generated, the minimum RSSs corresponding to the best regression models for each number of variables are given by $r_1^{(2^{n-1})},\ldots,r_n^{(2^{n-1})}$.

Lemma 1 *$r_j^{(g)} \geq r_{j+1}^{(g)}$, where $g = 1,\ldots,2^{n-1}$ and $j = 1,\ldots,n-1$.*

PROOF. The proof is by induction. For $g = 1$ only the root node $([1,2,\ldots,n],0)$ is derived. The initial values are $r_j^{(1)} = \mathrm{RSS}([1,2,\ldots,j])$. From (3.7) and $[1,\ldots,j] \subset [1,2,\ldots,j,j+1]$ it follows that $r_j^{(1)} \geq r_{j+1}^{(1)}$ for $j = 1,\ldots,n-1$. The inductive hypothesis is that Lemma 1 holds for $1 \leq g < 2^{n-1}$. It is required to show that $r_j^{(g+1)} \geq r_{j+1}^{(g+1)}$ $(j = 1,\ldots n-1)$ after the $(g+1)$th node of the RT $T([1,2,\ldots,n],0)$ has been generated.

Consider the $(g+1)$th node $(\mathrm{Drop}(V,i),i-1)$ which has been derived from (V,k), where $0 \leq k < |V|$ and $k < i < |V|$. The $(g+1)$th node provides the sub-sets

$$W_j = [v_1,\ldots,v_{i-1},v_{i+1},\ldots,v_{j+1}], \quad \text{where } j = i,\ldots,|V|-1.$$

The affected values are

$$r_j^{(g+1)} = \min(r_j^{(g)},\mathrm{RSS}(W_j)) \quad \text{for} \quad j = i,\ldots,|V|-1.$$

The $r_j^{(g+1)}$, when modified, becomes

$$r_j^{(g+1)} = \begin{cases} \mathrm{RSS}(W_j) \geq \mathrm{RSS}(W_{j+1}) \geq \min(r_{j+1}^{(g)},\mathrm{RSS}(W_{j+1})) = r_{j+1}^{(g+1)} & \text{if } i \leq j < |V|-1, \\ \mathrm{RSS}(W_{|V|-1}) \geq \mathrm{RSS}(V) \geq r_{j+1}^{(g)} = r_{j+1}^{(g+1)} & \text{if } j = |V|-1. \end{cases}$$

Otherwise $r_j^{(g+1)} = r_j^{(g)}$ and from the inductive hypothesis it follows that $r_j^{(g+1)} = r_j^{(g)} \geq r_{j+1}^{(g)} \geq r_{j+1}^{(g+1)}$. This completes the proof. ■

Lemma 2 *$r_{|W|}^{(g)} \leq RSS(W)$, where W is any set obtained from a node of $T(V,i-1)$, $r_i^{(g)} \leq RSS(V)$ and $1 \leq i < |V|$.*

PROOF. Let $W = [w_1,\ldots,w_{|W|}]$ denote any set obtained from a node of $T(V,i-1)$. From the definition of $T(V,i-1)$ it follows that $W \subseteq V$ and contains the first $i-1$ variables of V. Thus, W can be written as $[v_1,\ldots,v_{i-1},w_i,\ldots,w_{|W|}]$, where $i \leq |W|$. From Lemma 1 and (3.7) it follows, respectively, that $r_{|W|}^{(g)} \leq r_i^{(g)}$ and $\mathrm{RSS}(V) \leq \mathrm{RSS}(W)$. Since $r_i^{(g)} \leq \mathrm{RSS}(V)$ the latter implies $r_{|W|}^{(g)} \leq \mathrm{RSS}(W)$. This completes the proof. ■

Corollary 1 *$(1-\tau)r_{|W|}^{(g)} \leq RSS(W)$, where W is any set obtained from a node of $T(V,i-1)$, $0 \leq \tau < 1$, $(1-\tau)r_i^{(g)} \leq RSS(V)$ and $1 \leq i < |V|$.*

PROOF. From the proof of Lemma 2 it results that $r_{|W|}^{(g)} \leq r_i^{(g)}$ and since $(1-\tau)r_i^{(g)} \leq \mathrm{RSS}(V)$ it follows that $(1-\tau)r_{|W|}^{(g)} \leq \mathrm{RSS}(V) \leq \mathrm{RSS}(W)$ which completes the proof. ■

Now, let (V,k) be one of the g generated nodes of the RT $T([1,2,\ldots,n],0)$, where $0 \leq k < |V| \leq n$ and $1 \leq g \leq 2^{n-1}$. Consider also that the children nodes $(\text{Drop}(V,k+1),k)$, ..., $(\text{Drop}(V,i-1),i-2)$ have been generated, where $k+1 \leq i < |V|$. The remaining sub-trees $T(\text{Drop}(V,\rho),\rho-1)$ need to be derived for $\rho = i,\ldots,|V|-1$. The bound of the node (V,k) is denoted by $b_V \equiv \text{RSS}(V)$. If $r_i^{(g)} \leq b_V$, then, from Lemma 2 it follows that $r_{|W|}^{(g)} \leq \text{RSS}(W)$, where W is any sub-set obtained from a node of $T(V,i-1)$. The sub-trees of $T(V,i-1)$ are given by $T(\text{Drop}(V,\rho),\rho-1)$, where $\rho = i,\ldots,|V|-1$. This implies that the remaining sub-trees of (V,k) cannot improve the values of $r_j^{(g)}$, where $j = 1,\ldots,n$. This suggests that prior to the computation of $(\text{Drop}(V,i),i-1)$ the $r_i^{(g)}$ is compared with b_V, where $i = k+1,\ldots,|V|-1$. Specifically, if $r_i^{(g)} \leq b_V$, then all the sub-trees $T(\text{Drop}(V,\rho),\rho-1)$ are eliminated for $\rho = i,\ldots,|V|-1$. Otherwise, $(\text{Drop}(V,i),i-1)$ is computed and $r_j^{(g+1)}$ $(j = i,\ldots,|V|-1)$ updated. This procedure is repeated for the next child, i.e., $(\text{Drop}(V,i+1),i)$. Figure 3.3 illustrates this method, where $V = [3,4,5]$, $k = 0$, $i = 1$, $g = 13$, $(r_1^{(g)},\ldots,r_5^{(g)}) \equiv (668,615,597,592,548)$ and $b_V = 720$. The RSS of each model is shown in brackets. Since $r_1^{(g)} \leq b_V$, it follows from Lemma 2 that the node comprising only the last variable, i.e., $([5],0)$, will have bigger RSS than $r_1^{(g)}$. This implies that all the children of $([3,4,5],0)$, i.e., $([4,5],0)$, $([3,5],1)$ and $([5],0)$ need not to be computed.

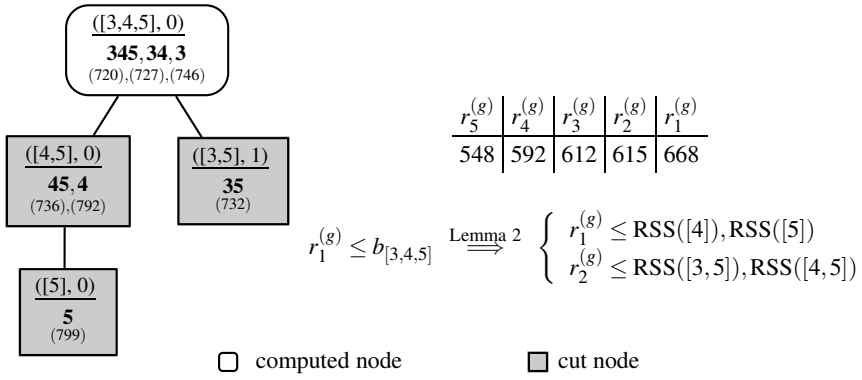


Figure 3.3: The cutting step in the sub-tree $T(V,k)$, where $V = [3,4,5]$ and $k = 0$.

The processing of the nodes from $(\text{Drop}(V,k+1),k)$ to $(\text{Drop}(V,|V|-1),|V|-2)$ is the most efficient. If the sub-tree $T(\text{Drop}(V,i),i-1)$ is cut, then all the remaining sub-trees, i.e., $T(\text{Drop}(V,\rho),\rho-1)$ for $\rho = i,\ldots,|V|-1$ could also be cut. However, other processing orders will derive sub-trees that become obsolete at a latter stage. The branch-and-bound procedure that generates the best-subset regression models is summarized by Algorithm 3. Notice that the algorithm uses the recursive procedure ProcessSubtree, which processes the sub-tree with the root node (V,k). Fig. 3.4 and

Table 3.1 show, respectively, the RT and execution steps of Algorithm 3 for a 5–variables model. The shadowed nodes (entries in cases of Table 3.1) are not computed, i.e., cut.

Algorithm 3 Branch-and-bound procedure for finding the best-subset regression models.

1: Compute the QRD of $A \in \mathbb{R}^{m\times n}$: $Q^T A = \binom{R}{0}_{m-n}^{n}$ and $Q^T y = \tilde{y}$
2: Let $V = [1,2,\ldots,n]$, $k = 0$ and $r_j = \text{RSS}([1,2,\ldots,j]) \equiv \sum_{i=j+1}^{m} \tilde{y}_i^2$ where $j = 1,\ldots,n$
3: **call** ProcessSubtree(V,k)

4: **def** ProcessSubtree(V,k) = **do**
5: **for** $i = k+1,\ldots,|V|-1$ **do**
6: **if** ($r_i > \text{RSS}(V)$) **then**
7: $V^{(i)} \leftarrow \text{Drop}(V,i)$
8: $r_j = \min(r_j, \text{RSS}([v_1^{(i)}, v_2^{(i)}, \ldots v_j^{(i)}]))$, where $j = i,\ldots,|V|-1$
9: **else**
10: Cut the remaining sub-trees and go to step 13
11: **end if**
12: **end for**
13: **call** ProcessSubtree($V^{(j)}, j-1$), where $j = k+1,\ldots,\min(i,|V|-2)$
14: **end def**

The computational efficiency of the BBA improves when more nodes are cut. That is, if bigger sub-trees are bounded with bigger values [36]. This can be achieved by pre-ordering the variables in the initial set so that the node $([1,2,\ldots,n],0)$ satisfies

$$\text{RSS}(Drop(V,1)) \geq \text{RSS}(Drop(V,2)) \geq \cdots \geq \text{RSS}(Drop(V,n-1)) \geq \text{RSS}(V). \tag{3.8}$$

The BBA that uses pre-ordering (3.8) is denoted by BBA–1. Another approach, BBA–2, which might cut bigger sub-trees at early stages, processes first the node with the minimum bound. Thus, if $(V_1,k_1),\ldots,(V_d,k_d)$ denote the d $(0 < d \leq 2^{n-1})$ possible nodes that can be processed at some stage of the BBA, then the node that corresponds to $\min(\text{RSS}(V_1),\ldots,\text{RSS}(V_d))$ is processed first.

For models with a large number of variables, the BBA might become computationally infeasible. In such cases various heuristics that provide solutions close to the optimum one should be applied [78, 91, 119, 120, 122]. One possibility is to cut sub-trees (step 6 of Algorithm 3) using a tolerance parameter. That is, when $(1-\tau)r_i^{(g)} \leq \text{RSS}(V)$, where τ is the tolerance with $0 \leq \tau < 1$. Clearly this heuristic algorithm, HBBA, is equivalent to the BBA when $\tau = 0$. Generally, the closer τ is to zero, the greater the chance of obtaining a solution close to the optimum. Specifically, let

Table 3.1: Execution steps of the BBA for a 5–variables model.

Step	Node	Models (RSS)					Bound	min RSS for # of variables				
								5	4	3	2	1
0.	([1,2,3,4,5], 0)	12345 (548)	1234 (592)	123 (612)	12 (615)	1 (668)	–	548	592	612	615	668
1.	([2,3,4,5], 0)		2345 (660)	234 (664)	23 (673)	2 (702)	548	548	592	612	615	668
2.	([1,3,4,5], 1)		1345 (605)	134 (612)	13 (641)		548	548	592	612	615	668
3.	([1,2,4,5], 2)		1245 (596)	124 (615)			548	548	592	612	615	668
4.	([1,2,3,5], 3)		1235 (592)				548	548	592	612	615	668
5.	([1,2,5], 2)			125 (597)			596	548	592	**597**	615	668
6.	([1,4,5], 1)			145 (618)	14 (648)		605	548	592	597	615	668
7.	([1, 3, 5], 2)			135 (618)			605	548	592	597	615	668
8.	([1,5], 1)				15 (618)		618	548	592	597	615	668
9.	([3,4,5], 0)			345 (720)	34 (727)	3 (746)	660	548	592	597	615	668
10.	([2,4,5], 1)			245 (667)	24 (685)		660	548	592	597	615	668
11.	([2,3,5], 2)			235 (666)			660	548	592	597	615	668
12.	([2,5], 1)				25 (675)		667	548	592	597	615	668
13.	([4,5], 0)				45 (736)	4 (792)	720	548	592	597	615	668
14.	([3,5], 1)				35 (732)		720	548	592	597	615	668
15.	([5], 0)					5 (799)	736	548	592	597	615	668

W^* be a set obtained from the eliminated sub-trees of $T(V,i-1)$ so that $\mathrm{RSS}(W^*)$ is the optimal value of RSS for models with $|W^*|$ variables, that is,

$$\mathrm{RSS}(W^*) = \min\{\mathrm{RSS}(W); \quad W \subseteq V \text{ and } |W| = |W^*|\}.$$

The relative error of the RSS corresponding to models with $|W^*|$ variables provided by HBBA is defined by

$$\zeta = (r_{|W^*|}^{(2^{n-1})} - \mathrm{RSS}(W^*))/\mathrm{RSS}(W^*)). \tag{3.9}$$

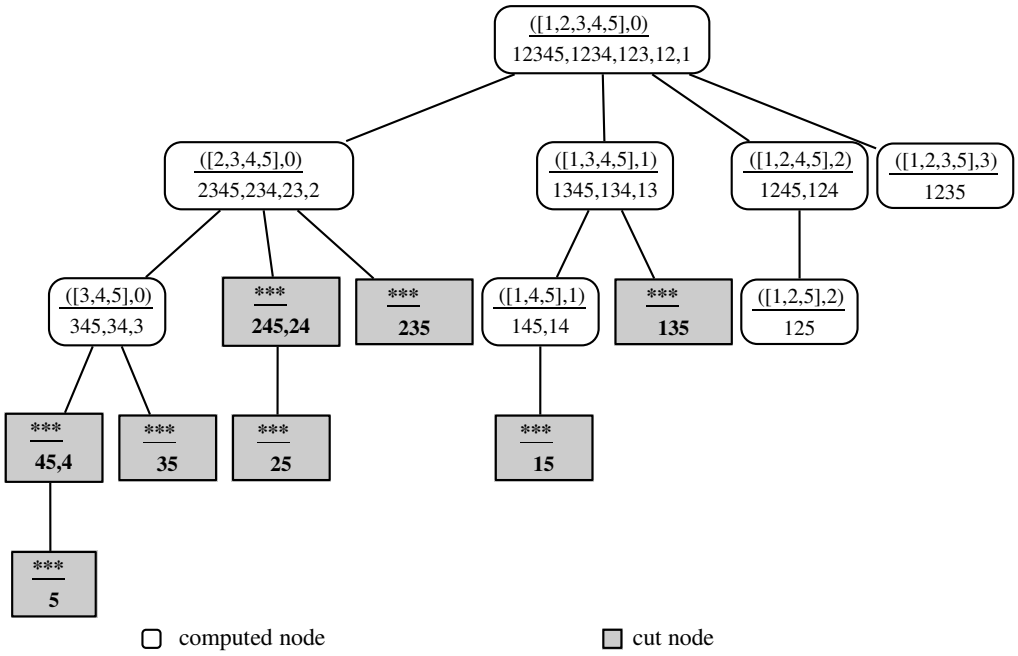


Figure 3.4: The regression tree $T(V,k)$ generated by the BBA, where $V = [1,\ldots,n]$ and $k = 0$.

From Corollary 1 it follows that $r^{(g)}_{|W^*|} - \mathrm{RSS}(W^*) \leq \tau r^{(g)}_{|W^*|}$, and since $r^{(2^{n-1})}_{|W^*|} \leq r^{(g)}_{|W^*|}$ it follows that

$$\zeta \leq \tau r^{(g)}_{|W^*|} / \mathrm{RSS}(W^*) \leq \tau. \tag{3.10}$$

That is, the relative error ζ of the solution provided by the heuristic HBBA does not exceed the value of the tolerance parameter τ.

3.4 Computational results

The number of nodes generated by the BBA depends on the model. If all the nodes of the RT need to be computed, then the BBA has the same complexity as that of the dropping-columns algorithm that generates all sub-models. That is, an upper bound for the number of flops required by the BBA is given by (3.6) for $|V| = n$ and $k = 0$. This will be denoted by $C_{\mathrm{BBA}}(n)$. The Leaps-and-Bounds Algorithm (hereafter LBA) proposed by Furnival and Wilson in [36] generates two trees each having 2^{n-1} nodes. At each step of the LBA an inverse matrix is computed on the regressions tree and a pivoting (Gaussian elimination) is applied on the bounds tree. Both operations generate a new node. The flops required to generate a new node of size n in the regressions tree and to perform

the pivoting on the bounds tree are given by $(20n^3+3n^2+n)/6$ and $2n^2-n$, respectively. Thus, the complexity of deriving a new node of size n, both on the regressions and the bounds trees, is given by

$$C_{\text{StepLBA}}(n) = 5(4n^3-3n^2-n)/6. \tag{3.11}$$

The regressions tree generated by the LBA for an n–variables initial model has n levels. The nodes at level i $(i = 0,1,\ldots n-1)$ comprise $n-i$ variables and are obtained after deleting a combination of i variables from the original model. The number of possibilities of deleting these i variables from the model is given by $\mathrm{C}_{n-1}^{i} = (n-1)!/(i!(n-1-i)!)$. Notice that the last variable is never deleted. Thus, the complexity of the LBA when generating all nodes is given by

$$\begin{aligned} C_{\text{LBA}}(n) &= \sum_{i=1}^{n-1} (\mathrm{C}_{n-1}^{i} C_{\text{StepLBA}}(n-i)) \\ &= 5(n-1)(2^n(2n^2+11n+6)-24n^2-8n)/48. \end{aligned} \tag{3.12}$$

The ratio between the upper bound complexities of the LBA and BBA is

$$C_{\text{LBA}}(n)/C_{\text{BBA}}(n) \approx 0.0058(2n^3+9n^2-5n-6) \equiv O(n^3).$$

Table 3.2 shows the execution times (seconds) of the LBA and standard BBA used on models with different number of variables. The data were randomly generated from a uniform distribution. These data are used to evaluate the computational performance of the different procedures. Table 3.2 also shows the time required to compute all the nodes of the RT. In this case, the BBA is equivalent to the dropping-columns algorithm in [42]. The models generated by the two algorithms are the same. The number of nodes processed by the algorithms are also shown. It can be observed that the BBA outperforms the LBA in terms of computational performance. The two algorithms have also been compared on different 20–variables initial models using random data. The BBA derived the best models 4 to 30 times faster than the LBA.

Table 3.3 shows the significant improvement in speed that can be achieved using various versions of the BBA. Recall that BBA–1 and BBA–2 are the standard BBA when pre-ordering of variables and minimum bounds have been used, respectively. This improvement in speed is also shown for the case of the heuristic HBBA when a pre-ordering of the variables is performed. This method is denoted by HBBA–1. The tolerances $\tau = 0.1$ and $\tau = 0.25$ are used in the experiments. Clearly, the BBA–1 is the most computational efficient when an exhaustive search is performed to obtain the best models. The HBBA–1 brings a significant improvement in terms of computational performance when heuristics are used.

Table 3.2: Execution times in seconds of the LBA and BBA.

	Generating all models			Cutting sub-trees			
	Time			LBA		BBA	
# of Var	LBA	BBA	LBA / BBA	Time	Nodes	Time	Nodes
15	2.17	0.15	14	0.17	1958	0.02	969
16	4.55	0.28	16	0.15	1556	0.03	760
17	9.49	0.54	18	0.58	5932	0.08	2966
18	19.93	1.07	19	0.94	8870	0.13	4383
19	43.64	2.17	20	2.55	23316	0.29	11655
20	88.73	4.48	20	11.35	108806	1.07	54403
21	184.51	8.65	21	7.77	64348	0.80	32174
22	387.50	17.02	23	7.13	49994	0.72	24988
23	811.99	34.16	24	8.18	57060	0.81	28511
24	1739.70	68.20	26	62.76	454332	5.74	227159
25	3617.78	136.00	27	53.66	358008	5.60	178997
26	7610.29	270.20	28	73.93	443718	6.61	221854
27	15793.16	541.40	29	53.13	325488	4.60	162669
28	32381.89	1079.74	30	451.00	2649534	42.18	1324643
29	68327.62	2151.78	32	244.12	1264478	27.21	632231
30	142094.05	4317.85	33	221.25	1018050	22.30	509014

Tables 3.4 and 3.5 present the performance of the heuristics HBBA and HBBA–1, respectively, on experimental and random data for different values of tolerance τ. The experimental data sets Ozone [19, 22, 51, 84] and Pollute [44, 92, 95] are used. The first data set consists of daily measurements of ozone concentration and eight meteorological quantities for 330 days in 1976 in the Los Angeles basin. The latter data set consists of 15 variables and is an age-adjusted mortality rate per $100,000$ population. Notice that for $\tau = 0.0$ the HBBA and HBBA–1 perform an exhaustive search and are equivalent to the standard BBA and BBA–1, respectively. Figures 3.5 and 3.6 show the RSS and relative errors (3.9) of the models obtained by HBBA and HBBA–1 for the Pollute data set. From the experimental results the following observations can be made:

- The BBA–1 is faster than the HBBA when the tolerance used is not very large.
- The HBBA–1 is significantly faster than the HBBA. This allows the HBBA–1 to use smaller tolerances and thus, derive models close to the optimum ones.
- The initial pre-ordering of variables results in the investigation of fewer nodes and consequently in a significant reduction of computational time.

Table 3.3: Execution times in seconds for various versions of the BBA.

	Exhaustive methods			Heuristics with $\tau = 0.1$		Heuristics with $\tau = 0.25$	
# of Var	BBA	BBA–1	BBA–2	HBBA	HBBA–1	HBBA	HBBA–1
15	0.02	0.01	0.02	0.01	0.01	^{B}0.009	0.003
20	1.14	0.05	0.78	0.48	0.02	0.16	0.009
25	5.60	0.32	3.79	1.78	0.05	^{B}0.20	0.010
30	22.54	1.09	18.23	3.46	0.04	1.46	0.005
35	171.64	3.01	117.12	42.47	0.32	4.77	0.040
40	10049.32	45.09	6644.69	168.17	1.31	^{B}12.27	0.030
41	3197.91	63.22	3163.20	80.94	1.12	^{B}0.89	0.070
42	28176.72	76.09	20128.03	4949.46	3.15	255.20	0.090
43	31567.22	289.52	21293.12	1353.42	4.50	^{B}115.70	0.093
44	3806.57	89.07	2443.04	266.99	2.78	^{B}11.93	0.086
45	47342.35	149.80	29566.28	2105.87	1.74	^{B}17.25	0.042

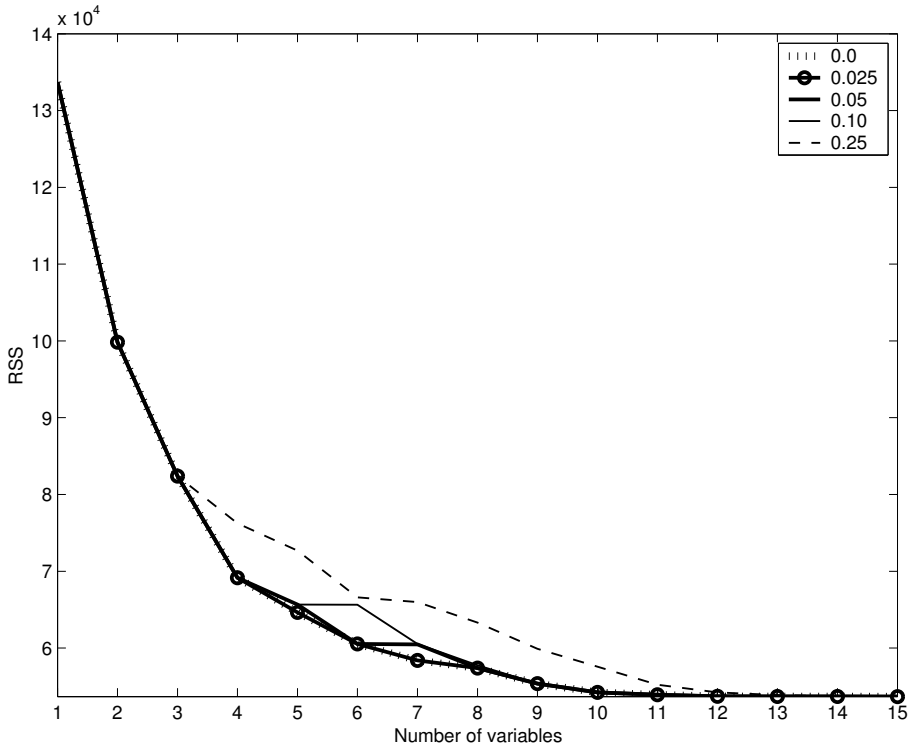
^{B} The execution time of HBBA is better than the one of the exhaustive BBA–1.

- As the number of variables in the initial model increases, the tolerance of the heuristic algorithm needs to be reduced in order to obtain correct models.
- The relative error (3.9) and (3.10) can be used effectively when considering the tolerance of the heuristic algorithms.
- The HBBA–1 found models closer to the optimal one compared to the HBBA, when the same tolerance is used.

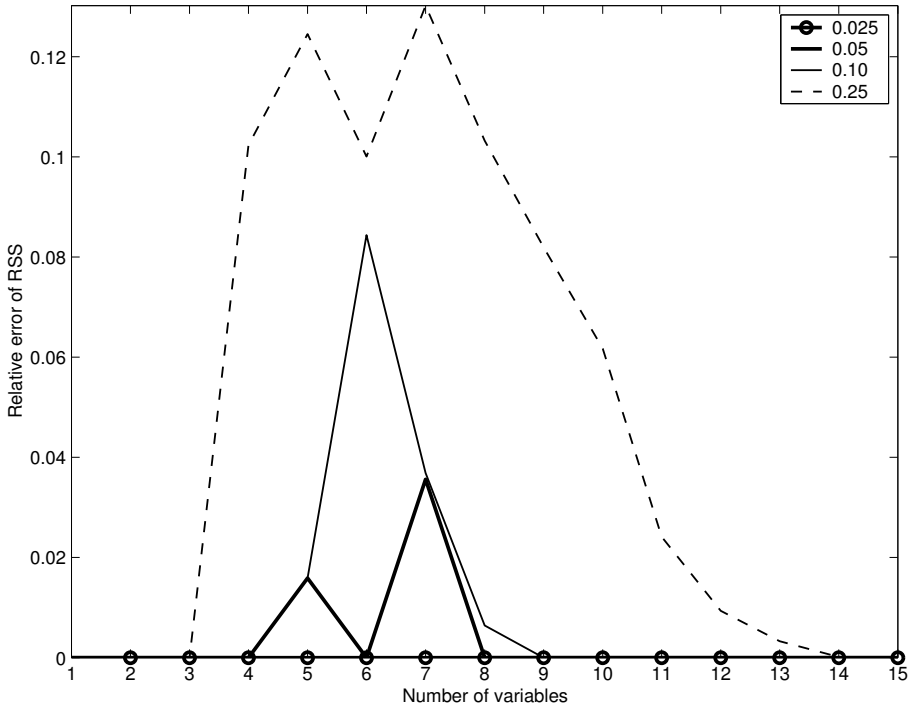
Table 3.4: Execution times in milliseconds of the HBBA.

	Experimental data				Random data			
	OZONE		POLLUTE		15 variables		30 variables	
τ	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
0.0	$^{(1)}$2.39	*143	$^{(1)}$20.00	*710	$^{(1)}$24.55	*969	$^{(1)}$22590	*509014
0.01	2.28	*130	19.99	*608	22.94	*788	17540	371049
0.025	2.03	105	16.38	*523	21.28	645	12300	240875
0.05	1.85	90	14.99	438	17.14	430	6860	119364
0.1	1.38	60	11.83	335	14.37	267	3490	53234
0.25	0.69	21	6.80	134	8.10	99	1450	18993

$^{(1)}$ Equivalent to the execution time of the standard BBA. * The optimal models are found.

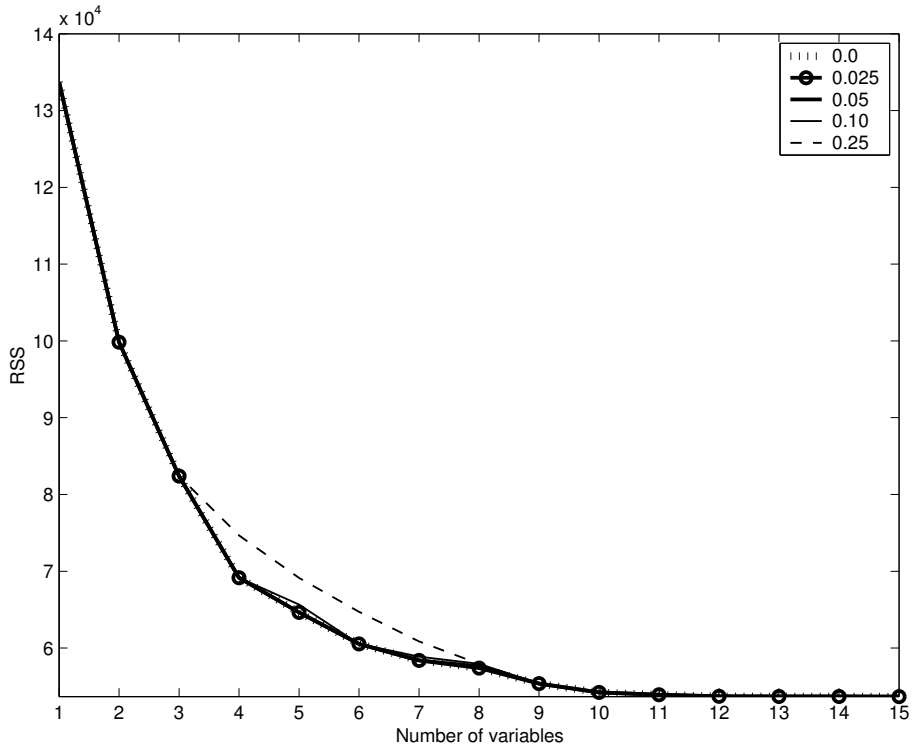


(a) The absolute values of the RSS.

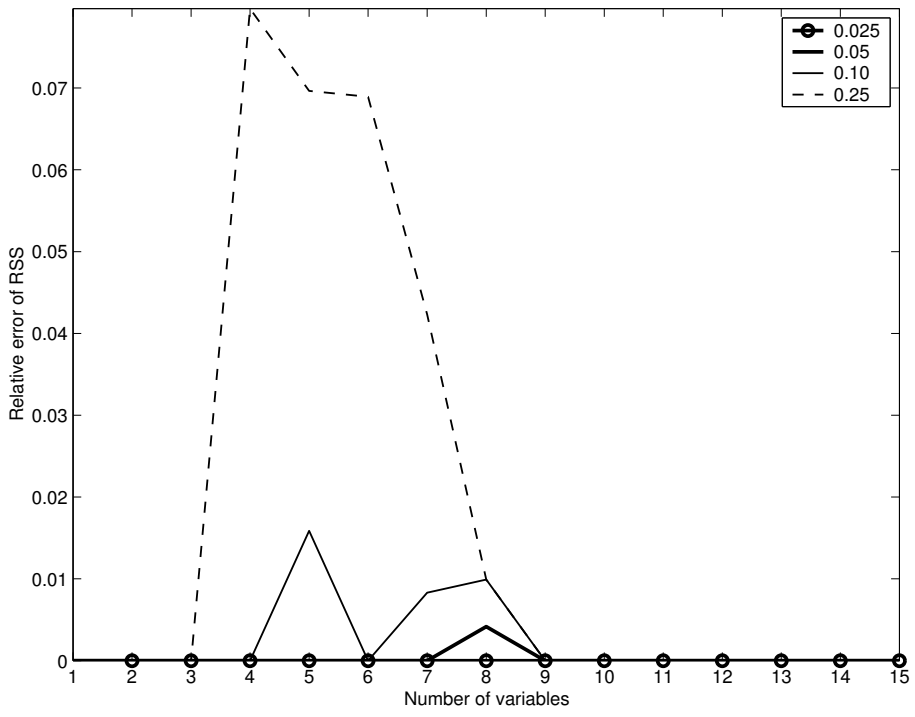


(b) The relative errors of the RSS.

Figure 3.5: The results of the HBBA on the POLLUTE data set with tolerance $\tau = 0.0$, 0.025, 0.05, 0.1 and 0.25.



(a) The absolute values of the RSS.



(b) The relative errors of the RSS.

Figure 3.6: The results of the HBBA–1 on the POLLUTE data set with tolerance $\tau = 0.0$, 0.025, 0.05, 0.1 and 0.25.

Table 3.5: Execution times in milliseconds of the HBBA–1.

	Experimental data				Random data			
	OZONE		POLLUTE		15 variables		30 variables	
τ	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
0.0	[(1)]0.39	*13	[(1)]12.95	*381	[(1)]12.85	*237	[(1)]1068.81	*17229
0.01	0.26	*6	10.71	*302	12.22	205	704.22	10074
0.025	0.24	*5	9.42	*248	11.10	176	392.68	4794
0.05	0.21	*4	8.48	197	8.94	117	154.76	1430
0.1	0.17	*3	5.55	111	6.56	62	35.67	204
0.25	0.06	*2	2.68	36	2.48	16	4.80	18

(1) Equivalent to the execution time of the BBA–1. * The optimal models are found.

3.5 Conclusions

A branch-and-bound algorithm (BBA) has been developed to compute the best-subset regression models corresponding to each number of variables. The algorithm exploits the properties of the regression tree that generates all sub-models and avoids its computation. Specifically, it has been proven (Lemmas 1 and 2) that an entire subtree can be cut when the branch-and-bound test holds. The main computational tool is the QR decomposition and the residual sum of squares is used for selecting the models. The ratio between the upper bound complexities of the existing Leaps-and-Bounds Algorithm (LBA) and the BBA is shown to be $O(n^3)$. Computational experiments confirm the theoretical results.

To increase the performance of the BBA, i.e., to cut more nodes, two strategies are proposed. The first, BBA–1, pre-orders the variables prior to the execution of the algorithm so that bigger sub-trees are bounded with bigger values. The second, BBA–2, selects the node at each step with the smallest bound from all possible nodes that can be processed. The experimental results show that the BBA–1 performs best when an exhaustive search is performed. The BBA can be modified to compute a set of the best sub-models for each number of variables.

A heuristic version of the BBA (HBBA) that uses a tolerance parameter when deciding to cut a subtree is proposed. In this case, a subtree is also cut when the branch-and-bound test fails, but the relative improvement obtained by computing the subtree is smaller than the tolerance parameter. This HBBA significantly improves the computational performance of the BBA, especially when the variables are pre-ordered. The HBBA might not provide the optimal solution. It is shown in (3.10) that the relative error of the computed solution is smaller than the tolerance parameter used.

The performance of the algorithms on various data sets is investigated. For smaller tolerance parameters, the heuristic algorithm found the best models and reduced significantly the computation cost of the BBA performing an exhaustive search.

The Branch-and-Bound algorithms and their Heuristic versions can be extended and adapted to other subset model-selection problems. The algorithms need to be modified so that a restriction on the size of the selected models can be imposed. The dynamic calculation of the tolerance within the execution process of the HBBA, an extensive study of these methods to various models and their comparison with other (e.g., Forward, Backward and Greedy) selection methods should be investigated. Currently, the parallelization of the BBA on a cluster of workstations and the adaptation of the BBA to subset selection strategies for VAR models are being considered.

Chapter 4

Efficient strategies for deriving the subset VAR models

Abstract:

Algorithms for computing the subset Vector Autoregressive (VAR) models are proposed. These algorithms can be used to choose a subset of the most statistically-significant variables of a VAR model. In such cases, the selection criteria are based on the residual sum of squares. The VAR model with zero coefficient restrictions is formulated as a Seemingly Unrelated Regressions (SUR) model. Furthermore, the SUR model is transformed into one of smaller size, where the exogenous matrices comprise columns of a triangular matrix. Efficient algorithms which exploit the common columns of the exogenous matrices, sparse structure of the variance-covariance of the disturbances and special properties of the SUR models are investigated. The main computational tool of the selection strategies is the generalized QR decomposition and its modification.

[1]This chapter is a reprint of the paper: C. Gatu and E.J. Kontoghiorghes. Efficient strategies for deriving the subset VAR models. *Computational Management Science*, 2003. (Submitted).

4.1 Introduction

A common problem in the Vector Autoregressive (VAR) process modeling is the lag structure identification or, equivalently, the specification of the subset VAR models [23, 39, 42, 86, 101, 120]. The vector time series $z_t \in \mathbb{R}^G$ is a VAR process of order p when its data generating process has the form

$$z_t = \Phi_1 z_{t-1} + \Phi_2 z_{t-2} + \cdots + \Phi_p z_{t-p} + \varepsilon_t, \tag{4.1}$$

where $\Phi_i \in \mathbb{R}^{G\times G}$ are the coefficient matrices and $\varepsilon_t \in \mathbb{R}^G$ is the noise vector. Given a set of realizations of the process in (4.1), $z_1, \ldots, z_M$ and a pre-sample $z_0, \ldots, z_{1-p}$ the parameter matrices are estimated from the linear model

$$\begin{pmatrix} z_1^T \\ z_2^T \\ \vdots \\ z_M^T \end{pmatrix} = \begin{pmatrix} z_0^T & z_{-1}^T & \cdots & z_{1-p}^T \\ z_1^T & z_0^T & \cdots & z_{2-p}^T \\ \vdots & \vdots & \ddots & \vdots \\ z_{M-1}^T & z_{M-2}^T & \cdots & z_{M-p}^T \end{pmatrix} \begin{pmatrix} \Phi_1^T \\ \Phi_2^T \\ \vdots \\ \Phi_p^T \end{pmatrix} + \begin{pmatrix} \varepsilon_1^T \\ \varepsilon_2^T \\ \vdots \\ \varepsilon_M^T \end{pmatrix}. \tag{4.2}$$

In the compact form the model in (4.2) can be written as

$$Y = XB + U, \tag{4.3}$$

where $Y = (y_1 \ldots y_G) \in \mathbb{R}^{M\times G}$ are the response vectors, $X \in \mathbb{R}^{M\times K}$ is the exogenous data matrix having full-column rank and block-Toeplitz structure, $B \in \mathbb{R}^{K\times G}$ is the coefficient matrix, $U = (u_1 \ldots u_G) \in \mathbb{R}^{M\times G}$ are the disturbances and $K = Gp$. The expectation of U is zero, i.e. $\mathrm{E}(u_i) = 0$, and $\mathrm{E}(u_i u_j^T) = \sigma_{ij} I_M$ $(i, j = 1, \ldots, G)$ [72, 77, 86, 98, 99, 114]. The VAR model (4.3) can be written as

$$\mathrm{vec}(Y) = (I_G \otimes X)\,\mathrm{vec}(B) + \mathrm{vec}(U), \quad \mathrm{vec}(U) \sim (0, \Sigma \otimes I_M), \tag{4.4}$$

where vec is the vector operator and $\Sigma = [\sigma_{ij}] \in \mathbb{R}^{G\times G}$ has full rank [33, 63]. The Ordinary and Generalized Least Squares estimators of (4.4) are the same and given by

$$\hat{B} = (X^T X)^{-1} X^T Y.$$

Often zero-coefficient constraints are imposed on the VAR models. This might be due to the fact that the data generating process in (4.1) contains only a few non-zero coefficients. Also, over-fitting the model might yeld in loss of efficiency when it is used for further testing, such as

forecasting [86, 121]. A zero-restricted VAR model (ZR–VAR) is called subset VAR model. When prior knowledge about zero-coefficient constraints are not available, several subset VAR models have to be compared with respect to some specified criterion. If the purpose is the identification of a model as close as possible to the data generating process, then the use of an information criterion for evaluating the subset models is appropriate. The selection criteria such as Akaike Information Criterion (AIC), Hannan-Quinn (HQ) and Schwarz Criterion (SC) are based on the residual sum of squares or the estimated residual covariance matrix [2, 3, 50, 107]. There is a trade-off between a good fit, i.e. small value of the residual sum of squares, and the number of non-zero coefficients. That is, there is a penalty related to the number of included non-zero coefficients.

Finding good models can be seen as an optimization problem, i.e. minimize or maximize a selection criterion over a set of sub-models derived from a finite realization of the process in (4.1) by applying a selection rule [121]. Let $B = (b_1 \dots b_G)$ and $S_i \in \mathbb{R}^{K \times k_i}$ $(i = 1, \dots, G)$ denote a selection matrix such that $\beta_i = S_i^T b_i$ corresponds to the non-zero coefficients of b_i – the ith column of B. Furthermore, let $X_i = XS_i$ which are the columns of X that correspond to the non-zero coefficients of b_i. Thus, the ZR–VAR model is equivalent to the Seemingly Unrelated Regressions (SUR) model

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_G \end{pmatrix} = \begin{pmatrix} XS_1 & & & \\ & XS_2 & & \\ & & \ddots & \\ & & & XS_G \end{pmatrix} \begin{pmatrix} S_1^T b_1 \\ S_2^T b_2 \\ \vdots \\ S_G^T b_G \end{pmatrix} + \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_G \end{pmatrix},$$

or

$$\operatorname{vec}(Y) = (\oplus_{i=1}^{G} X_i) \operatorname{vec}(\{\beta_i\}_G) + \operatorname{vec}(U), \quad \operatorname{vec}(U) \sim (0, \Sigma \otimes I_M), \tag{4.5}$$

where $\oplus_{i=1}^{G} X_i = \operatorname{diag}(X_1, \dots, X_G)$, $\{\beta_i\}_G$ denotes the set $\{\beta_1, \dots, \beta_G\}$ and $\operatorname{vec}(\{\beta_i\}_G) = (\beta_1^T \dots \beta_G^T)^T$. For notational convenience the direct sum $\oplus_{i=1}^{G}$ and the set $\{\cdot\}_G$ are abbreviated to $\oplus_i$ and $\{\cdot\}$, respectively.

One possible approach to search for the optimal models is to enumerate all $2^{pG^2} - 1$ possible subset VAR models. However this approach is infeasible even for modest values of G and p. Thus, existing methods search in a smaller given subspace. One selection method is to enforce a whole coefficient matrix $\Phi_i (1 \leq i \leq p)$, or a combination of coefficient matrices to be zero. In this case, the number of the subset VAR models to be evaluated is $2^p - 1$. Polynomial top-down and bottom-up strategies based on the deletion and, respectively, inclusion of the coefficients in each

equation separately have been also previously proposed [86]. Alternative methods use optimization heuristics such as Threshold Accepting [121].

Several algorithms for computing the subset VAR models are presented. The ZR–VAR model which is formulated as a SUR model, is transformed into one of smaller size, where the exogenous matrices comprise columns of a triangular matrix [33]. The common columns of the exogenous matrices and the Kronecker structure of the variance-covariance of the disturbances are exploited in order to derive efficient estimation algorithms.

In the next section the numerical solution of the ZR–VAR model is given. Section 3 presents an efficient variable-downdating strategy of the subset VAR model. Section 4 describes an algorithm for deriving all subset VAR models by moving efficiently from one model to another. Special cases which take advantage of the special Toeplitz structure of the data matrix and the Kronecker structure of the variance-covariance matrix are described in Section 5. Conclusion and future work are presented and discussed in Section 6.

4.2 Numerical solution of the ZR–VAR model

Orthogonal transformations can employed to reduce to zero $M-K$ observations of the VAR model. This results in an equivalent transformed model with less observations, and thus, to a smaller-size estimation problem. Consider the QR decomposition (QRD) of the exogenous matrix X

$$\bar{Q}^T X = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} K \\ M-K \end{matrix}, \quad \text{with } \bar{Q} = \begin{pmatrix} \overset{K}{\bar{Q}_A} & \overset{M-K}{\bar{Q}_B} \end{pmatrix}, \tag{4.6}$$

where $\bar{Q} \in \mathbb{R}^{M \times M}$ is orthogonal and $R \in \mathbb{R}^{K \times K}$ is upper-triangular. Let

$$\bar{Q}^T Y = \begin{pmatrix} \overset{G}{\tilde{Y}} \\ \hat{Y} \end{pmatrix} \begin{matrix} K \\ M-K \end{matrix} \text{ and } \quad \bar{Q}^T U = \begin{pmatrix} \overset{G}{\tilde{U}} \\ \hat{U} \end{pmatrix} \begin{matrix} K \\ M-K \end{matrix}, \tag{4.7}$$

where

$$\begin{pmatrix} \mathrm{vec}(\tilde{U}) \\ \mathrm{vec}(\hat{U}) \end{pmatrix} \sim \begin{pmatrix} \Sigma \otimes I_K & 0 \\ 0 & \Sigma \otimes I_{M-K} \end{pmatrix}.$$

Premultiplying (4.4) by $(I_G \otimes \bar{Q}_A \quad I_G \otimes \bar{Q}_B)^T$ gives

$$\begin{pmatrix} \mathrm{vec}(\bar{Q}_A^T Y) \\ \mathrm{vec}(\bar{Q}_B^T Y) \end{pmatrix} = \begin{pmatrix} I_G \otimes \bar{Q}_A^T X \\ I_G \otimes \bar{Q}_B^T X \end{pmatrix} \mathrm{vec}(B) + \begin{pmatrix} \mathrm{vec}(\bar{Q}_A^T U) \\ \mathrm{vec}(\bar{Q}_B^T U) \end{pmatrix}.$$

From (4.6) and (4.7) it follows that the latter can be written as

$$\begin{pmatrix} \mathrm{vec}(\tilde{Y}) \\ \mathrm{vec}(\hat{Y}) \end{pmatrix} = \begin{pmatrix} I_G \otimes R \\ 0 \end{pmatrix} \mathrm{vec}(B) + \begin{pmatrix} \mathrm{vec}(\tilde{U}) \\ \mathrm{vec}(\hat{U}) \end{pmatrix}$$

which is equivalent to the reduced-size model

$$\mathrm{vec}(\tilde{Y}) = (I_G \otimes R)\,\mathrm{vec}(B) + \mathrm{vec}(\tilde{U}), \quad \mathrm{vec}(\tilde{U}) \sim (0, \Sigma \otimes I_K). \tag{4.8}$$

From the latter, it follows that (4.5) is equivalent to the smaller in size SUR model

$$\mathrm{vec}(\tilde{Y}) = (\oplus_i R^{(i)})\,\mathrm{vec}\,(\{\beta_i\}) + \mathrm{vec}(\tilde{U}), \quad \mathrm{vec}(\tilde{U}) \sim (0, \Sigma \otimes I_K), \tag{4.9}$$

where $R^{(i)} = RS_i \in \mathbb{R}^{K \times k_i}$ [32, 63, 66].

The best linear unbiased estimator (BLUE) of the SUR model in (4.9) comes from the solution of the Generalized Linear Least Squares Problem (GLLSP)

$$\underset{V,\{\beta_i\}}{\mathrm{argmin}} \|V\|_F^2 \quad \text{subject to } \mathrm{vec}(\tilde{Y}) = (\oplus_i R^{(i)})\,\mathrm{vec}\,(\{\beta_i\}) + \mathrm{vec}(VC^T). \tag{4.10}$$

Here $\Sigma = CC^T$, the random $V \in \mathbb{R}^{K \times G}$ is defined as $VC^T = \tilde{U}$ which implies $\mathrm{vec}(V) \sim (0,\, I_{GK})$, and $\|\cdot\|_F$ denotes Frobenius norm i.e. $\|V\|_F^2 = \sum_{i=1}^{K} \sum_{j=1}^{G} V_{i,j}^2$ [72, 77, 98, 99]. The upper-triangular $C \in \mathbb{R}^{G \times G}$ is the Cholesky factor of Σ. For the solution of (4.10) consider the Generalized QR Decomposition (GQRD) of the matrices $\oplus_i R^{(i)}$ and $C \otimes I_K$:

$$Q^T(\oplus_i R^{(i)}) = \begin{pmatrix} \oplus_i R_i \\ 0 \end{pmatrix} \begin{matrix} K^* \\ GK-K^* \end{matrix} \tag{4.11a}$$

and

$$\begin{aligned} Q^T(C \otimes I_K)\Pi P &= \begin{pmatrix} W^{(0,1)} & W^{(0,2)} \\ W^{(0,3)} & W^{(0,4)} \end{pmatrix} P \\ &= W \equiv \begin{array}{c} \begin{matrix} K^* & GK-K^* \end{matrix} \\ \begin{pmatrix} W_{11} & W_{12} \\ 0 & W_{22} \end{pmatrix} \begin{matrix} K^* \\ GK-K^* \end{matrix} \end{array}, \end{aligned} \tag{4.11b}$$

where $K^* = \sum_{i=1}^{G} k_i$, $\oplus R_i$ and W are upper triangular of order K^* and GK, respectively, and Π is a $GK \times GK$ permutation matrix defined as $\Pi = (\oplus_i (I_{k_i}\ 0)^T \oplus_i (0\ I_{K-k_i})^T)$. Furthermore, $W^{(0,j)}$ $(j = 1,2,3,4)$ are block-triangular and $R_i \in \mathbb{R}^{k_i \times k_i}$ is the upper-triangular factor in the QRD of $R^{(i)}$. That is,

$$Q_i^T R^{(i)} = \begin{pmatrix} R_i \\ 0 \end{pmatrix} \begin{matrix} k_i \\ K-k_i \end{matrix}, \quad \text{with} \quad Q_i^T = \begin{pmatrix} Q_{Ai}^T \\ Q_{Bi}^T \end{pmatrix} \begin{matrix} k_i \\ K-k_i \end{matrix} \quad (i = 1,\ldots,G), \tag{4.12}$$

where $Q_i \in \mathbb{R}^{K \times K}$ is orthogonal and Q in (4.11a) is defined by

$$Q = (\oplus_i Q_{Ai} \quad \oplus_i Q_{Bi}) = \begin{pmatrix} Q_{A1} & & & Q_{B1} & & \\ & \ddots & & & \ddots & \\ & & Q_{AG} & & & Q_{BG} \end{pmatrix}.$$

Now, since $\|V\|_F^2 = \|P^T \Pi^T \operatorname{vec}(V)\|^2$, the GLLSP (4.10) is equivalent to

$$\begin{aligned} &\underset{V,\{\beta_i\}}{\operatorname{argmin}} \|P^T \Pi^T \operatorname{vec}(V)\|^2 \quad \text{subject to} \\ &\qquad Q^T \operatorname{vec}(\tilde{Y}) = Q^T (\oplus_i R^{(i)}) \operatorname{vec}(\{\beta_i\}) + Q^T (C \otimes I_K) \Pi P P^T \Pi^T \operatorname{vec}(V), \end{aligned} \tag{4.13}$$

where $\|\cdot\|$ denotes the Euclidian norm. Using (4.11a) and (4.11b) the latter can be re-written as

$$\begin{aligned} &\underset{\{\tilde{v}_{Ai}\},\{\tilde{v}_{Bi}\},\{\beta_i\}}{\operatorname{argmin}} \sum_{i=1}^{G} (\|\tilde{v}_{Ai}\|^2 + \|\tilde{v}_{Bi}\|^2) \quad \text{subject to} \\ &\qquad \begin{pmatrix} \operatorname{vec}(\{\tilde{y}_{Ai}\}) \\ \operatorname{vec}(\{\tilde{y}_{Bi}\}) \end{pmatrix} = \begin{pmatrix} \oplus_i R_i \\ 0 \end{pmatrix} \operatorname{vec}(\{\beta_i\}) + \begin{pmatrix} W_{11} & W_{12} \\ 0 & W_{22} \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\{\tilde{v}_{Ai}\}) \\ \operatorname{vec}(\{\tilde{v}_{Bi}\}) \end{pmatrix}, \end{aligned} \tag{4.14}$$

where $\tilde{y}_{Ai}, \tilde{v}_{Ai} \in \mathbb{R}^{k_i}$, $\tilde{y}_{Bi}, \tilde{v}_{Bi} \in \mathbb{R}^{K-k_i}$,

$$\tilde{y}_{Ai} = Q_{Ai}^T \tilde{y}_i, \quad \tilde{y}_{Bi} = Q_{Bi}^T \tilde{y}_i \tag{4.15a}$$

and

$$P^T \Pi^T \operatorname{vec}(V) = \begin{pmatrix} \operatorname{vec}(\{\tilde{v}_{Ai}\}) \\ \operatorname{vec}(\{\tilde{v}_{Bi}\}) \end{pmatrix} \begin{matrix} K^* \\ GK-K^* \end{matrix}. \tag{4.15b}$$

From the constraint in (4.14) it follows that

$$\operatorname{vec}(\{\tilde{v}_{Bi}\}) = W_{22}^{-1} \operatorname{vec}(\{\tilde{y}_{Bi}\}), \quad i = 1,\ldots,G \tag{4.16}$$

and the GLLSP is reduced to

$$\underset{\{\tilde{v}_{Ai}\},\{\beta_i\}}{\operatorname{argmin}} \sum_{i=1}^{G} \|\tilde{v}_{Ai}\|^2 \quad \text{subject to } \operatorname{vec}(\{\tilde{\tilde{y}}_i\}) = (\oplus_i R_i)\operatorname{vec}(\{\beta_i\}) + W_{11}\operatorname{vec}(\{\tilde{v}_{Ai}\}), \tag{4.17}$$

where

$$\operatorname{vec}(\{\tilde{\tilde{y}}_i\}) = \operatorname{vec}(\{\tilde{y}_{Ai}\}) - W_{12}\operatorname{vec}(\{\tilde{v}_{Bi}\}). \tag{4.18}$$

The solution of (4.17), and thus, the BLUE of (4.9), is obtained by setting $\tilde{v}_{Ai} = 0$ $(i = 1,\ldots,G)$ and solving the linear system

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

or, equivalently, by solving the set of triangular systems

$$R_i\hat{\beta}_i = \tilde{\tilde{y}}_i, \quad i = 1,\ldots,G.$$

4.3 Variable-downdating of the ZR–VAR model

Consider the re-estimation of the SUR model (4.9) when new zero constraints are imposed to the coefficients β_i $(i = 1,\ldots,G)$. That is, after estimating (4.9) the new SUR model to be estimated is given by

$$\operatorname{vec}(\tilde{Y}) = (\oplus_i \tilde{R}^{(i)})\operatorname{vec}(\{\tilde{\beta}_i\}) + \operatorname{vec}(\tilde{U}), \quad \operatorname{vec}(\tilde{U}) \sim (0, \Sigma \otimes I_K), \tag{4.19}$$

where $\tilde{R}^{(i)} = R^{(i)}\tilde{S}_i$, $\tilde{\beta}_i = \tilde{S}_i^T\beta_i$ and $\tilde{S}_i \in \mathbb{R}^{k_i \times \tilde{k}_i}$ is a selection matrix $(0 \le \tilde{k}_i \le k_i)$. This is equivalent to solving the GLLSP

$$\underset{V,\{\tilde{\beta}_i\}}{\operatorname{argmin}} \|V\|_F^2 \quad \text{subject to } \operatorname{vec}(\tilde{Y}) = (\oplus_i \tilde{R}^{(i)})\operatorname{vec}(\{\tilde{\beta}_i\}) + \operatorname{vec}(VC^T). \tag{4.20}$$

From (4.11) and (4.15) it follows that (4.20) can be written as

$$\underset{\{\tilde{v}_{Ai}\},\{\tilde{v}_{Bi}\},\{\tilde{\beta}_i\}}{\operatorname{argmin}} \sum_{i=1}^{G} (\|\tilde{v}_{Ai}\|^2 + \|\tilde{v}_{Bi}\|^2) \quad \text{subject to}$$

$$\begin{pmatrix} \operatorname{vec}(\{\tilde{y}_{Ai}\}) \\ \operatorname{vec}(\{\tilde{y}_{Bi}\}) \end{pmatrix} = \begin{pmatrix} \oplus_i R_i\tilde{S}_i \\ 0 \end{pmatrix} \operatorname{vec}(\{\tilde{\beta}_i\}) + \begin{pmatrix} W_{11} & W_{12} \\ 0 & W_{22} \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\{\tilde{v}_{Ai}\}) \\ \operatorname{vec}(\{\tilde{v}_{Bi}\}) \end{pmatrix}.$$

Using (4.16) and (4.18) the latter becomes

$$\underset{\{\tilde{v}_{Ai}\},\{\tilde{\beta}_i\}}{\operatorname{argmin}} \|\operatorname{vec}(\{\tilde{v}_{Ai}\})\|^2 \quad \text{subject to } \operatorname{vec}(\{\tilde{\tilde{y}}_i\}) = (\oplus_i R_i \tilde{S}_i)\operatorname{vec}(\{\tilde{\beta}_i\}) + W_{11}\operatorname{vec}(\{\tilde{v}_{Ai}\}). \tag{4.21}$$

Now, consider the GQRD of the matrices $\oplus_i R_i \tilde{S}_i$ and W_{11}, that is

$$\tilde{Q}^T(\oplus_i R_i \tilde{S}_i) = \begin{pmatrix} \oplus_i \tilde{R}_i \\ 0 \end{pmatrix} \begin{matrix} \tilde{K}^* \\ K^* - \tilde{K}^* \end{matrix} \tag{4.22a}$$

and

$$\tilde{Q}^T W_{11} \tilde{\Pi} \tilde{P} = \tilde{W} = \begin{matrix} \tilde{K}^* & K^* - \tilde{K}^* \end{matrix} \begin{pmatrix} \tilde{W}_{11} & \tilde{W}_{12} \\ 0 & \tilde{W}_{22} \end{pmatrix} \begin{matrix} \tilde{K}^* \\ K^* - \tilde{K}^* \end{matrix}, \tag{4.22b}$$

where $\tilde{K}^* = \sum_{i=1}^{G} \tilde{k}_i$, $\oplus_i \tilde{R}_i$ and $\tilde{W}$ are upper triangular of order $\tilde{K}^*$ and K^*, respectively, and $\tilde{\Pi} = (\oplus_i (I_{\tilde{k}_i}\ 0)^T \oplus_i (0\ I_{k_i - \tilde{k}_i})^T)$. Notice that, the upper-triangular $\tilde{R}_i \in \mathbb{R}^{\tilde{k}_i \times \tilde{k}_i}$ comes from the QRD

$$\tilde{Q}_i^T R_i \tilde{S}_i = \begin{pmatrix} \tilde{R}_i \\ 0 \end{pmatrix} \begin{matrix} \tilde{k}_i \\ k_i - \tilde{k}_i \end{matrix}, \quad \text{with} \quad \tilde{Q}_i^T = \begin{pmatrix} \tilde{Q}_{Ai}^T \\ \tilde{Q}_{Bi}^T \end{pmatrix} \begin{matrix} \tilde{k}_i \\ k_i - \tilde{k}_i \end{matrix} \quad (i = 1, \ldots, G), \tag{4.23}$$

where $\tilde{Q}_i \in \mathbb{R}^{k_i \times k_i}$ is orthogonal. The latter factorization is a re-triangularization of a triangular factor after deleting columns [61, 63, 70, 123]. Furthermore, $\tilde{Q}$ in (4.22) is defined by $\tilde{Q} = (\oplus_i \tilde{Q}_{Ai} \quad \oplus_i \tilde{Q}_{Bi})$. If $\tilde{y}_{Ai}^* = \tilde{Q}_{Ai}^T \tilde{\tilde{y}}_i$, $\tilde{y}_{Bi}^* = \tilde{Q}_{Bi}^T \tilde{\tilde{y}}_i$,

$$\tilde{P}^T \tilde{\Pi}^T \operatorname{vec}(\{\tilde{v}_{Ai}\}) = \begin{pmatrix} \operatorname{vec}(\{\tilde{v}_{Ai}^*\}) \\ \operatorname{vec}(\{\tilde{v}_{Bi}^*\}) \end{pmatrix} \begin{matrix} \tilde{K}^* \\ K^* - \tilde{K}^* \end{matrix}$$

and

$$\operatorname{vec}(\{\tilde{y}_i^*\}) = \operatorname{vec}(\{\tilde{y}_{Ai}^*\}) - \tilde{W}_{12}\operatorname{vec}(\{\tilde{v}_{Bi}^*\}),$$

then the solution of the GLLSP (4.21) is obtained by solving

$$(\oplus_i \tilde{R}_i)\operatorname{vec}(\{\hat{\tilde{\beta}}_i\}) = \operatorname{vec}(\{\tilde{y}_i^*\}),$$

or $\tilde{R}_i \hat{\tilde{\beta}}_i = \tilde{y}_i^*$ $(i = 1, \ldots, G)$. Notice that, the GQRD (4.22) is the most expensive computation required for deriving the BLUE of the SUR model (4.19) after the factorization (4.11) has been computed.

An efficient strategy for computing the orthogonal factorization (4.22b) has been proposed within the context of updating SUR models [67]. Notice that $\tilde{Q}^T W_{11}\tilde{\Pi}$ in (4.22b) can be written as

$$\tilde{Q}^T W_{11}\tilde{\Pi} = W^{(0)} = \begin{array}{c} \begin{array}{cc} \tilde{K}^* & K^*-\tilde{K}^* \end{array} \\ \begin{pmatrix} W^{(0,1)} & W^{(0,2)} \\ W^{(0,3)} & W^{(0,4)} \end{pmatrix} \begin{array}{l} \tilde{K}^* \\ K^*-\tilde{K}^* \end{array} \end{array} , \tag{4.24}$$

where $W^{(0,i)}$ $(i = 1,\ldots,4)$ has a block-triangular structure. That is, $W^{(0)}$ has the structural form

$$W^{(0)} = \left(\begin{array}{cccc|cccc} \tilde{k}_1 & \tilde{k}_2 & \ldots & \tilde{k}_G & k_1-\tilde{k}_1 & k_2-\tilde{k}_2 & \ldots & k_G-\tilde{k}_G \\ \hline W_{11}^{(0,1)} & W_{12}^{(0,1)} & \ldots & W_{1G}^{(0,1)} & W_{11}^{(0,2)} & W_{12}^{(0,2)} & \ldots & W_{1G}^{(0,2)} \\ 0 & W_{22}^{(0,1)} & \ldots & W_{2G}^{(0,1)} & 0 & W_{22}^{(0,2)} & \ldots & W_{2G}^{(0,2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & W_{GG}^{(0,1)} & 0 & 0 & \ldots & W_{GG}^{(0,2)} \\ \hline W_{11}^{(0,3)} & W_{12}^{(0,3)} & \ldots & W_{1G}^{(0,3)} & W_{11}^{(0,4)} & W_{12}^{(0,4)} & \ldots & W_{1G}^{(0,4)} \\ 0 & W_{22}^{(0,3)} & \ldots & W_{1G}^{(0,3)} & 0 & W_{22}^{(0,4)} & \ldots & W_{1G}^{(0,4)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & W_{GG}^{(0,3)} & 0 & 0 & \ldots & W_{GG}^{(0,4)} \end{array}\right) \begin{array}{l} \tilde{k}_1 \\ \tilde{k}_2 \\ \vdots \\ \tilde{k}_G \\ k_1-\tilde{k}_1 \\ k_2-\tilde{k}_2 \\ \vdots \\ k_G-\tilde{k}_G \end{array} . \tag{4.25}$$

The orthogonal matrix $\tilde{P}$ in (4.22b) computes the RQ decomposition of (4.25) using a sequence of $(G+1)$ orthogonal factorizations. That is, $\tilde{P} = \tilde{P}^{(0)}\tilde{P}^{(1)}\ldots\tilde{P}^{(G)}$, where $\tilde{P}^{(i)} \in \mathbb{R}^{K^*\times K^*}$ $(i = 0,\ldots,G)$ is orthogonal. Initially, $\tilde{P}^{(0)}$ triangularizes the blocks of the main block-diagonal of $W^{(0)}$. I.e. $\tilde{P}^{(0)} = \text{diag}(\tilde{P}_{11}^{(0)},\ldots,\tilde{P}_{1G}^{(0)},\tilde{P}_{41}^{(0)},\ldots,\tilde{P}_{4G}^{(0)})$, where the RQ decomposition of $W_{jj}^{(0,i)}$ is given by $W_{jj}^{(1,i)} = W_{jj}^{(0,i)}\tilde{P}_{ij}^{(0)}$ $(i = 1,4$ and $j = 1,\ldots,G)$. Here, $W_{jj}^{(1,i)}$ is triangular. The matrix

$$W^{(0)}\tilde{P}^{(0)} = W^{(1)} \equiv \begin{pmatrix} W^{(1,1)} & W^{(1,2)} \\ W^{(1,3)} & W^{(1,4)} \end{pmatrix}$$

has the same structure as in (4.25), but with $W^{(1,1)}$ and $W^{(1,4)}$ being triangular. The transformation $W^{(i+1)} = W^{(i)}\tilde{P}^{(i)}$ annihilates the ith super block-diagonal of $W^{(i,3)}$, i.e. the block $W_{j,j+i-1}^{(i,3)}$ $(j = 1,\ldots,G-i+1)$, and preserves the triangular structure of $W^{(1,1)}$ and $W^{(1,4)}$. Specifically, $\tilde{P}^{(i)}$ is

defined by

$$\tilde{P}^{(i)} = \left(\begin{array}{cccc|cccc} I_{J_i} & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & \tilde{P}_1^{(i,1)} & \cdots & 0 & \tilde{P}_1^{(i,2)} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \tilde{P}_{G-i+1}^{(i,1)} & 0 & \cdots & \tilde{P}_{G-i+1}^{(i,2)} & 0 \\ \hline 0 & \tilde{P}_1^{(i,3)} & \cdots & 0 & \tilde{P}_1^{(i,4)} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \tilde{P}_{G-i+1}^{(i,3)} & 0 & \cdots & \tilde{P}_{G-i+1}^{(i,4)} & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & I_{\rho_i} \end{array}\right),$$

where $J_i = \sum_{j=1}^{i-1} \tilde{k}_i$ and $\rho_i = \sum_{j=1}^{i-1} (k_i - \tilde{k}_i)$. Figure 4.1 shows the process of re-triangularizing $W^{(0)}$, where $G = 4$.

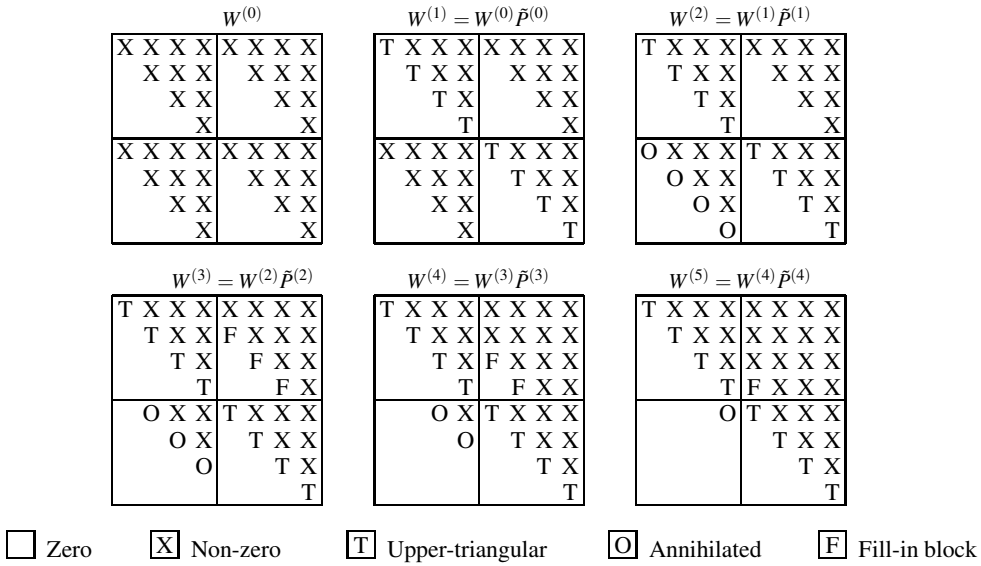


Figure 4.1: Re-triangularization of $W^{(0)}$ in (4.25).

4.4 Deriving the subset VAR models

All possible subset VAR models can be generated by moving from one model to another. Consider deleting only the μth variable from jth block of the reduced ZR–VAR model (4.9). This is equivalent to the SUR model in (4.19), where $\tilde{S}_j = (e_1 \dots e_{\mu-1}\ e_{\mu+1} \dots e_{k_j})$, e_l is the lth column of the

identity matrix I_{k_j}, $\tilde{S}_i = I_{k_i}$, $\tilde{\beta}_i = \beta_i$, $\tilde{k}_i = k_i$, for $i = 1,\ldots,G$ and $i \neq j$. Thus, the ZR–VAR model to be estimated is equivalent to the SUR model

$$\begin{pmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_j \\ \vdots \\ \tilde{y}_G \end{pmatrix} = \begin{pmatrix} R^{(1)} & & & & \\ & \ddots & & & \\ & & R^{(j)}\tilde{S}_j & & \\ & & & \ddots & \\ & & & & R^{(G)} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \tilde{S}_j^T\beta_j \\ \vdots \\ \beta_G \end{pmatrix} + \begin{pmatrix} \tilde{u}_1 \\ \vdots \\ \tilde{u}_j \\ \vdots \\ \tilde{u}_G \end{pmatrix}. \tag{4.26}$$

The BLUE of (4.26) comes from the solution of the GLLSP in (4.21). Now, let W_{11} be partitioned as

$$W_{11} = \begin{array}{c} \begin{matrix} k_1 & k_2 & \cdots & k_j & \cdots & k_G \end{matrix} \\ \begin{pmatrix} \Gamma_{1,1} & \Gamma_{1,2} & \cdots & \Gamma_{1,j} & \cdots & \Gamma_{1,G} \\ 0 & \Gamma_{2,2} & \cdots & \Gamma_{2,j} & \cdots & \Gamma_{2,G} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Gamma_{j,j} & \cdots & \Gamma_{j,G} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & \Gamma_{G,G} \end{pmatrix} \end{array} \begin{matrix} k_1 \\ k_2 \\ \vdots \\ k_j \\ \vdots \\ k_G \end{matrix}.$$

The computation of the GQRD (4.22) can be efficiently derived in two stages. The first stage, initially computes the QRD

$$\check{Q}^T\tilde{R}^{(j)} = \begin{pmatrix} \tilde{R}_j \\ 0 \end{pmatrix}$$

and the product $\check{Q}^T(\Gamma_{j,j}\cdots\Gamma_{j,G}) = (\Gamma^*_{j,j}\cdots\Gamma^*_{j,G})$. Then, it computes the RQD $\Gamma^*_{j,j}\check{P} = \tilde{\Gamma}^*_{j,j}$ and the product $(\Gamma^T_{1,j}\cdots\Gamma^T_{j-1,j})^T\check{P} = (\tilde{\Gamma}^T_{1,j}\cdots\tilde{\Gamma}^T_{j-1,j})^T$. Here, the orthogonal $\check{Q}^T$ and $\check{P}$ are the products of $k_i - \mu$ left and right Givens rotations which re-triangularize $\tilde{R}^{(j)}$ and $\Gamma^*_{j,j}$, respectively. Furthermore, let

$$\check{\Pi}^T = \begin{pmatrix} I_{K^*_j-1} & 0 & 0 \\ 0 & 0 & I_{K^*_G-K^*_j} \\ 0 & 1 & 0 \end{pmatrix}, \quad \text{with} \quad K^*_j = \sum_{i=1}^{j} k_i.$$

Thus, in (4.22a) $\tilde{Q}^T = \check{\Pi}^T\check{Q}^T_*$ and

$$\tilde{Q}^T \oplus_i (R_i\tilde{S}_i) = \begin{pmatrix} \oplus_i\tilde{R}_i \\ 0 \end{pmatrix}^{\tilde{K}^*}_{1},$$

where $\check{Q}_*^T = \text{diag}(I_{K_{j-1}^*}, \check{Q}^T, I_{K^*-K_j^*})$ and $K^* \equiv K_G^*$.

The second stage computes the RQD $\tilde{W} = (\check{\Pi}^T \check{W})\hat{P}$, where $\tilde{W}$ and $\check{W}$ are upper-triangular, $\check{W} = \check{Q}_*^T W_{11} \check{P}_*$, $\check{P}_* = \text{diag}(I_{K_{j-1}^*}, \check{P}, I_{K^*-K_j^*})$ and $\hat{P}$ is the product of $K^* - K_j^*$ Givens rotations. The ρth rotation ($\rho = 1, \ldots, K^* - K_j^*$), say $\hat{P}_\rho$, annihilates the $(K_j^* + \rho - 1)$th element of the last row of $\check{\Pi}^T \check{W}$ by rotating adjacent planes. Figure 4.2 illustrates the computation of the two stages, where $G = 3$, $k_1 = 4$, $k_2 = 5$, $k_3 = 3$, $j = 2$ and $\mu = 2$. Notice that in (4.22b), $\tilde{\Pi}$ is the identity matrix and $\tilde{P} \equiv \check{P}^*$.

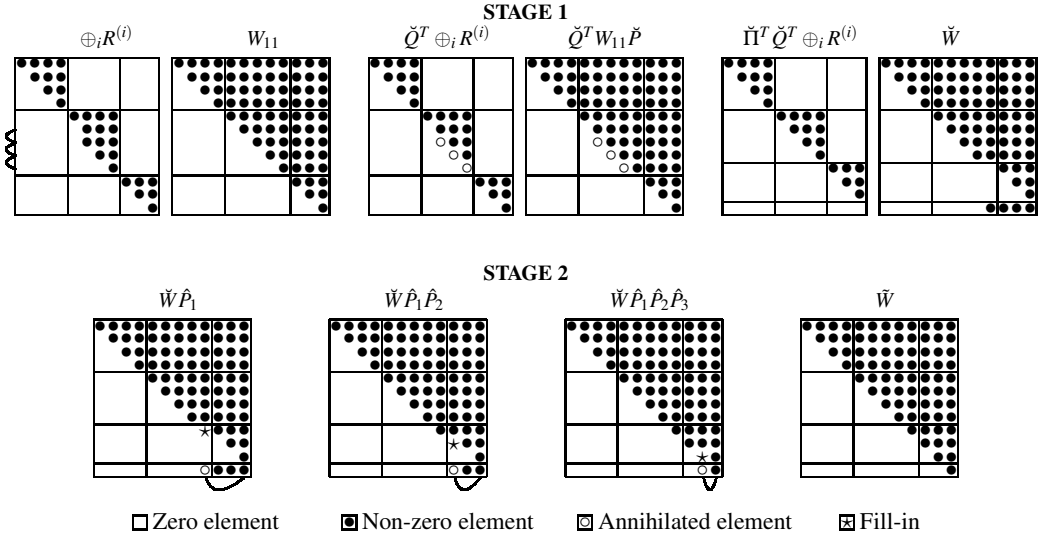


Figure 4.2: The two stages of estimating a ZR–VAR model after deleting one variable.

Now, let $V = [v_1, v_2, \ldots, v_n]$ denote the set of $n = |V|$ indices of the selected columns (variables) included in the sub-matrices $R^{(i)}$ $(i = 1, \ldots, G)$. The sub-models corresponding to the sub-sets $[v_1]$, $[v_1, v_2]$, $\cdots$, $[v_1, v_2, \ldots, v_n]$ are immediately available. A function Drop will be used to derive the remaining sub-models [42]. This function downdates the ZR–VAR by one variable. That is,

$$\text{Drop}(V, i) = [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n], \quad \text{where } i = 1, \ldots, n-1.$$

An efficient algorithm, called Dropping Columns Algorithm (DCA) has been previously introduced within the context of generating all subset models of the ordinary and general linear models [24, 42, 112]. The DCA generates a regression tree. It moves from one node to another by applying a Drop operation, that is, by deleting a single variable. A formal and detailed description of the regression tree which generates all subset models can be found in [42]. Here the basic concepts using a more convenient notation are introduced.

Let V denote the set of indices and $0 \leq \gamma < |V|$. A node of the regression tree is a tuple (V,γ), where γ indicates that the children of this node will include the first γ variables. If $V = [v_1, v_2, \ldots, v_n]$, then the regression tree $T(V,\gamma)$ is a $(n-1)$–tree having as root the node (V,γ), where $\gamma = 0,\ldots,n-1$. The children are defined by the tuples $(\text{Drop}(V,i), i-1)$ for $i = \gamma+1,\ldots,n-1$. Formally this can be expressed recursively as

$$T(V,\gamma) = \begin{cases} (V,\gamma) & \text{if} \quad \gamma = n-1, \\ \big((V,\gamma), T(\text{Drop}(V,\gamma+1),\gamma), \cdots, T(\text{Drop}(V,n-1),n-2)\big) & \text{if} \quad \gamma < n-2. \end{cases}$$

The number of nodes in the sub-tree $T(\text{Drop}(V,i), i-1)$ is given by $\delta_i = 2^{n-i-1}$ and $\delta_i = 2\delta_{i+1}$, where $i = 1,\ldots,n-1$ [42].

Computing all possible subset regressions of a model having n variables is equivalent to generating $T(V,0)$, where $V = [1,2,\ldots,n]$. Generally, the complexity –in terms of flops– of generating $T(V,\gamma)$ in the General Linear model case is of $O((|V|+\gamma)2^{|V|-\gamma})$. Figure 4.3 shows $T(V,\gamma)$ together with the sub-models generated from each node, where $V = [1,2,3,4,5]$ and $\gamma = 0$. A sub-model is denoted by a sequence of numerals which correspond to the indices of variables.

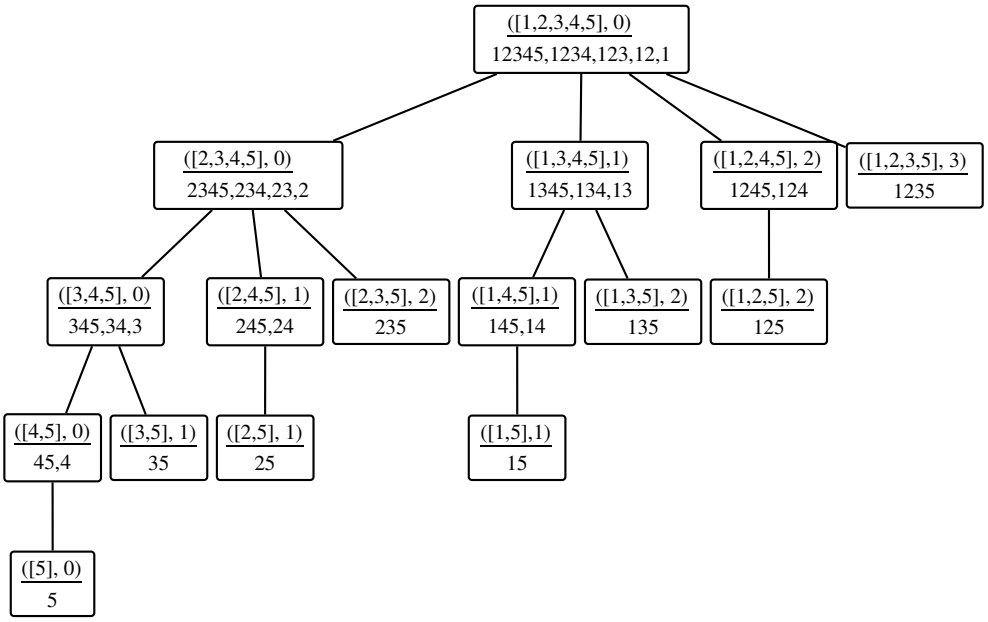


Figure 4.3: The regression tree $T(V,\gamma)$, where $V = [1,2,3,4,5]$ and $\gamma = 0$.

The DCA will generate all the subset VAR models by deleting one variable from the upper-triangular regressors of the reduced-size model (4.8). It avoids estimating each ZR–VAR model

afresh, i.e. it derives efficiently the estimation of one ZR–VAR model from another after deleting a single variable. Algorithm 4 summarizes this procedure.

Algorithm 4 Generating the regression tree $T(V,\gamma)$ given the root node (V,γ).

```
1: procedure SubTree(V, γ)
2:    From (V, γ) obtain the the sub–models (v_1 ··· v_{γ+1}), ..., (v_1 ··· v_{|V|})
3:    for i = γ + 1, ..., |V| − 1 do
4:       V^(i) ← Drop(V, i)
5:       SubTree(V^(i), i − 1)
6:    end for
7: end procedure
```

4.5 Special cases

The method of generating all subset VAR models becomes rapidly infeasible when the dimensions of the generating process (4.1), i.e. G and p increase. Thus, two approaches can be envisaged. The first is to compare models from a smaller given search space. The second is the use of heuristic optimization techniques [121]. Here, the former approach is considered.

A simplified approach is to consider a block-version of Algorithm 4, i.e. a ZR–VAR model is derived by deleting a block rather than a single variable. Within this context, in Figure 4.3 the numerals will represent indices of blocks of variables. This approach will generate $2^G - 1$ subset VAR models and can be implemented using fast block-downdating algorithms [63]. Notice that the deletion of the entire jth block is equivalent in deleting the jth row from all $\Phi_1, \ldots, \Phi_p$. This is different than the method in [86] where a whole coefficient matrix Φ_i $(i = 1, \ldots, p)$ is deleted at one time.

4.5.1 Deleting identical variables

Deleting the same variables from all the G blocks of the ZR–VAR model corresponds to deletion of whole columns from some of the coefficient matrices $\Phi_1, \ldots, \Phi_p$. This is equivalent to the SUR

model in (4.9), where $S_i \equiv S \in \mathbb{R}^{K\times\tilde{k}}$ for $i = 1,\ldots,G$ and $0 \leq \tilde{k} < K$. Thus, (4.9) can be written as

$$\begin{pmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_G \end{pmatrix} = \begin{pmatrix} RS & & \\ & \ddots & \\ & & RS \end{pmatrix} \begin{pmatrix} S^T b_1 \\ \vdots \\ S^T b_G \end{pmatrix} + \begin{pmatrix} \tilde{u}_1 \\ \vdots \\ \tilde{u}_G \end{pmatrix}. \tag{4.27}$$

The estimation of (4.27) comes from the solution of GLLSP (4.10), where now, $R^{(i)} = RS$, $\beta_i = S^T b_i$ and $k_i = \tilde{k}$ for $i = 1,\ldots,G$. The orthogonal matrices in (4.12) are identical, i.e. $Q_i^T = \check{Q}^T$ for $i = 1,\ldots,G$ and have the structure

$$\check{Q}^T = \begin{pmatrix} \check{Q}_A^T \\ \check{Q}_B^T \end{pmatrix} \begin{matrix} \tilde{k} \\ K-\tilde{k} \end{matrix}.$$

Multiplying respectively, Q^T and Q from the left and right of $(C \otimes I_K)$ it results

$$Q^T (C \otimes I_K) Q = \begin{pmatrix} C \otimes W_A \\ C \otimes W_B \end{pmatrix} (\oplus_i \check{Q}_A \quad \oplus_i \check{Q}_B) = \begin{pmatrix} C \otimes I_{\tilde{k}} & 0 \\ 0 & C \otimes I_{K-\tilde{k}} \end{pmatrix}.$$

The latter is upper-triangular. Figure 4.4 shows the computation of $Q^T (C \otimes I_K) Q$, where $G = 3$, $K = 5$ and $\tilde{k} = K - 1$. In this case, the permutation matrix Π in (4.11) is not required , i.e. $\Pi = I_{KG}$ and the matrix $P \equiv Q$. Notice that for the construction of Q in (4.11) only a $K \times K$ orthogonal matrix $\check{Q}$ needs to be computed rather than the $K \times K$ matrices $Q_1,\ldots,Q_G$.

The DCA can be modified to generate the subset VAR models derived by deleting identical variables from each block. Given a node (V,γ), the set V denotes the indices of the non-deleted (selected) variables. The parameter γ has the same definition as in section 4. The model (4.27) is estimated. This provides $G|V| - \gamma$ sub-leading VAR models. Then, one variable is deleted, specifically $V^{(i)} \equiv [v_1,\ldots,v_{i-1},v_{i+1},\ldots,v_{|V|}]$ for $i = \gamma+1,\ldots,|V|$. The procedure is recursively repeated for $V^{(\gamma+1)},\ldots,V^{(|V|)}$.

This method is summarized by Algorithm 5 which generates a regression tree of $2^K - 1$ nodes. Each node corresponds to one of the possible combination of selecting variables out of K. In general, the regression tree with the root node (V,γ) has $2^{|V|-\gamma} - 1$ nodes and provides $2^{|V|-\gamma-1}(|V| + \gamma + 2) - 1$ subset VAR models. Figure 4.5 shows the regression tree for the case $K = 4$ and $G = 2$. Each node shows (V,γ) and the indices of the corresponding subset VAR model in (4.27) together with its sub-leading models.

Notice that Algorithm 5 generates all the subset VAR models by deleting the same variables from each block when initially $V \equiv [1,\ldots,K]$ and $\gamma = 0$. Compared to the standard variable-deleting strategy of Algorithm 4, it requires $O(G)$ less computational complexity in order to generate $(K + 2)2^{K-1} - 1$ out of the $2^{KG} - 1$ possible subset VAR models.

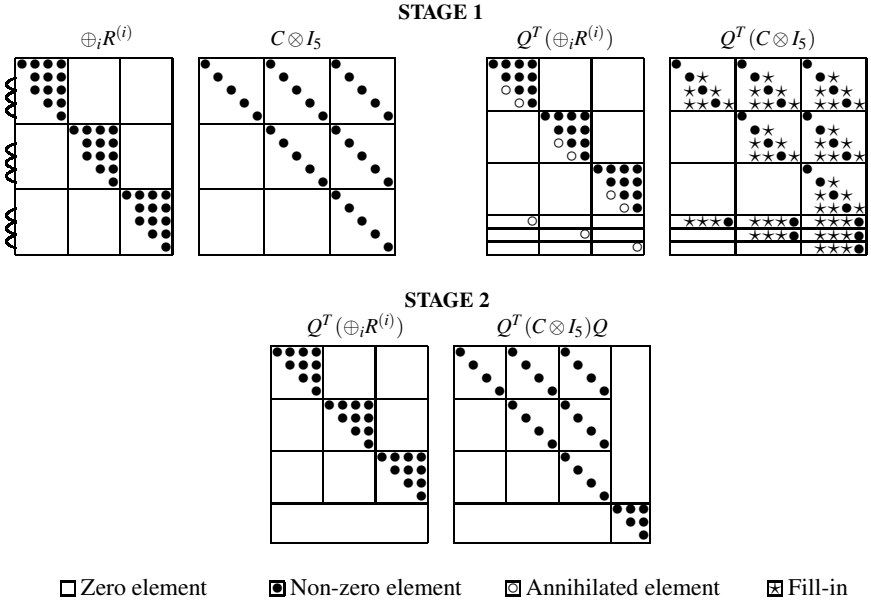


Figure 4.4: The two stages of estimating a SUR model after deleting the same variable from each block.

Algorithm 5 Generating the subset VAR models by Deleting Identical Variables (DIV).

1: **procedure** SubTree_DIV(V, γ)
2: Let the selection matrix $S \equiv [e_{v_1} \cdots e_{v_{|V|}}]$
3: Estimate the subset VAR model $\text{vec}(\tilde{Y}) = (I_K \otimes RS)\,\text{vec}(\{S^T b_i\}) + \text{vec}(\tilde{U})$
4: **for** $i = \gamma + 1, \ldots, |V|$ **do**
5: $V^{(i)} \equiv [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{|V|}]$
6: **if** ($|V^{(i)}| > 0$) **then** SubTree_DIV($V^{(i)}, i-1$) **end if**
7: **end for**
8: **end procedure**

4.5.2 Deleting subsets of variables

The computational burden is reduced when subsets of variables are deleted from the blocks of the ZR–VAR model (4.9). Consider the case of proper subsets and specifically when

$$S_i = \begin{pmatrix} k_{i+1} & k_i - k_{i+1} \\ S_{i+1} & S_i^* \end{pmatrix}, \quad i = 1, \ldots, G-1. \tag{4.28}$$

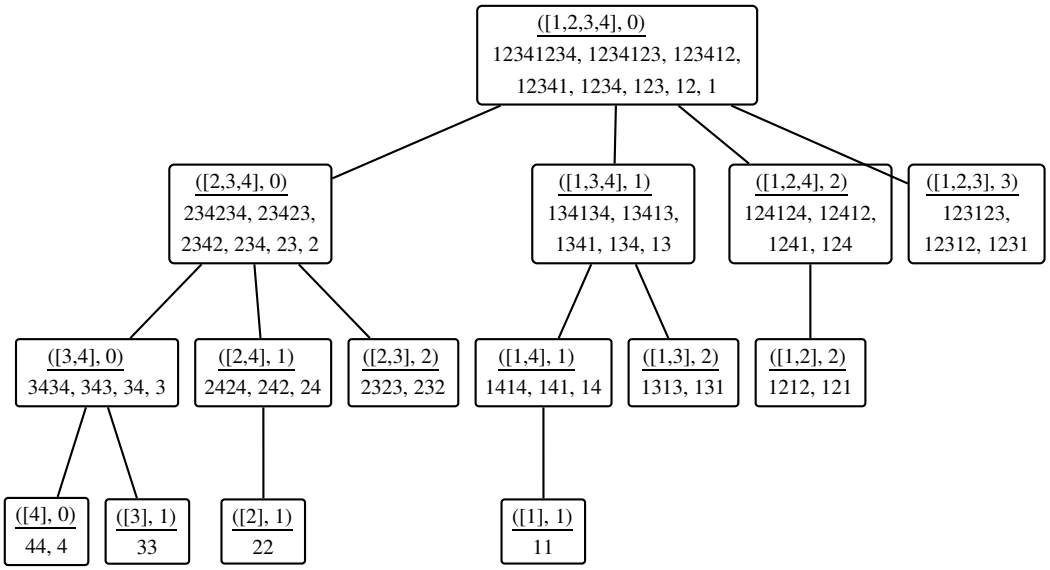


Figure 4.5: The regression tree $T(V,\gamma)$, where $V=[1,2,3,4]$, $K=2$, $G=2$ and $\gamma=0$.

From the QRD of $R^{(1)}$

$$Q_1^T R^{(1)} = \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \begin{matrix} k_1 \\ K-k_1 \end{matrix}, \tag{4.29}$$

it follows that the QRD of $R^{(i)}$ can be written as

$$Q_1^T R^{(i)} = \begin{pmatrix} R_i \\ 0 \end{pmatrix} \quad \text{with} \quad Q_1^T = \begin{pmatrix} Q_{Ai}^T \\ Q_{Bi}^T \end{pmatrix} \begin{matrix} k_i \\ K-k_i \end{matrix}, \quad i=1,\ldots,G, \tag{4.30}$$

where R_i is the leading triangular $k_i \times k_i$ sub-matrix of R_1. Now, the GLLSP (4.10) can be written as

$$\underset{V,\{\beta_i\}}{\operatorname{argmin}} \|Q^T \operatorname{vec}(V)\| \quad \text{subject to } Q^T \operatorname{vec}(Y) = Q^T \oplus_i (R^{(i)}\beta_i) + Q^T (C \otimes I_K) Q Q^T \operatorname{vec}(V), \tag{4.31}$$

where $Q^T = (\oplus_i Q_{Ai}^T \quad \oplus_i Q_{Bi}^T)$. The latter is equivalent to (4.14), but with $\tilde{v}_{Ai} = Q_{Ai}^T v_i$, $\tilde{v}_{Bi} = Q_{Bi}^T v_i$. Furthermore, the triangular W_{pq} $(p,q = 1,2)$ can be partitioned as

$$W_{11} = \left(\begin{array}{c|c|c|c} c_{11}I_{k_1} & \begin{matrix} c_{12}I_{k_2} \\ 0 \end{matrix} & \cdots & \begin{matrix} c_{1G}I_{k_G} \\ 0 \end{matrix} \\ \hline 0 & c_{22}I_{k_2} & \cdots & \begin{matrix} c_{2G}I_{k_G} \\ 0 \end{matrix} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & c_{GG}I_{k_G} \end{array}\right), \quad W_{12} = \left(\begin{array}{c|c|c|c} 0 & \begin{matrix} 0 & 0 \\ c_{12}I_{k_1-k_2} & 0 \end{matrix} & \cdots & \begin{matrix} 0 & 0 \\ c_{1G}I_{k_1-k_G} & 0 \end{matrix} \\ \hline 0 & 0 & \cdots & \begin{matrix} 0 & 0 \\ c_{2G}I_{k_2-k_G} & 0 \end{matrix} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & 0 \end{array}\right),$$

$$W_{21} = 0, \qquad W_{22} = \left(\begin{array}{c|c|c|c} c_{11}I_{K-k_1} & \begin{matrix} 0 & c_{12}I_{K-k_1} \end{matrix} & \cdots & \begin{matrix} 0 & c_{1G}I_{K-k_1} \end{matrix} \\ \hline 0 & c_{22}I_{K-k_2} & \cdots & \begin{matrix} 0 & c_{2G}I_{K-k_2} \end{matrix} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & c_{GG}I_{K-k_G} \end{array}\right).$$

This simplifies the computation of the estimation of the ZR–VAR model (4.9) [72, 74]. The (4.16) becomes

$$\tilde{v}_{Bi} = (\tilde{y}_{Bi} - \sum_{j=i+1}^{G} c_{ij}(0 \quad I_{K-k_i})\tilde{v}_{Bj})/c_{ii}, \quad (i = G,\ldots,1) \tag{4.32a}$$

and the estimation $\hat{\beta}_i$ $(i = 1,\ldots,G)$ is computed by solving the triangular system

$$R_i\hat{\beta}_i = \big(\tilde{y}_{Ai} - \sum_{j=i+1}^{G} c_{ij}\begin{pmatrix} 0 & 0 \\ I_{k_i-k_j} & 0 \end{pmatrix}\tilde{v}_{Bj}\big). \tag{4.32b}$$

The proper subsets VAR models are generated by enumerating all the possible selection matrices in (4.28) and estimating the corresponding models using (4.32). This enumeration can be obtained by considering all the possibilities of deleting variables on the first block, i.e. generating S_1 and then constructing the remaining selection matrices $S_2,\ldots,S_G$ conformly with (4.28).

This method is summarized by Algorithm 6 and consists of two procedures. The first, SubTreeM, is the modified SubTree procedure of Algorithm 4. It generates the regression tree as shown in Figure 4.3. In addition, for each node (V,γ), the ProperSubsets procedure is executed. The latter performs no factorization, but computes the estimated coefficients using (4.32). Specifically, it derives all possible proper subsets $(S_1,\ldots,S_G)$ in (4.28), for $S_1 = [e_{v_1},\ldots,e_{v_{\gamma+1}}],\ldots,$ $[e_{v_1},\ldots,e_{v_{\gamma+1}},\ldots,e_{v_{|V|}}]$. The ProperSubsets procedure is based on a backtracking scheme. That is, given $S_1,\ldots,S_{i-1}$ $(i = 1,\ldots,G)$, it generates a new S_i and increments i. If this is not possible, then it performs a backtracking step, i.e. it decrements i and repeats the procedure. As shown in

the Appendix, the number of proper subsets VAR models generated by Algorithm 6 is given by

$$f(K,G) = \begin{cases} 2^K - 1 & \text{if} \quad G = 1; \\ \sum_{i=1}^{\min(K,G-1)} C_{G-2}^{i-1} C_K^i 2^{K-i} & \text{if} \quad G \geq 2, \end{cases}$$

where $C_n^k = n!/k!(n-k)!$.

Algorithm 6 Generating the subset VAR models by deleting proper subsets of variables.

procedure SubTreeM(V, γ)
  Compute the QRD (4.29) for $S_1 = [e_{v_1}, \ldots, e_{v_{|V|}}]$
  ProperSubsets(V, γ)
  for $i = \gamma + 1, \ldots, |V| - 1$ **do**
    $V^{(i)} \leftarrow \text{Drop}(V, i)$
    SubTreeM($V^{(i)}, i-1$)
  end for
end procedure

procedure ProperSubsets(V, γ)
  Let $S_1 \leftarrow [e_{v_1}, \ldots, e_{v_\gamma}]$; $k_1 \leftarrow \gamma$; $i \leftarrow 1$
  while ($i \geq 1$) **do**
    if ($i = 1$ **and** $k_1 < |V|$) **or** ($i > 1$ **and** $k_i < k_{i-1}$) **then**
      $k_i \leftarrow k_i + 1$; $S_i(k_i) \leftarrow e_{v_{k_i}}$
      if ($i = G$) **then**
        Extract $R_1, \ldots R_G$ in (4.30) from R_1 in (4.29) corresponding to $(S_1, \ldots S_G)$
        Solve the GLLSP (4.31) using (4.32)
      else
        $i \leftarrow i + 1$; $k_i \leftarrow 0$
      end if
    else
      $i \leftarrow i - 1$
    end if
  end while
end procedure

The order of generating the models is illustrated in Figure 4.6 for the case $K = 4$ and $G = 3$. The highlighted models are the common models which are generated when the proper subsets are

in increasing order, i.e.

$$S_{i+1} = \overset{\begin{matrix} k_i & k_{i+1}-k_i \end{matrix}}{\begin{pmatrix} S_i & S^*_{i+1} \end{pmatrix}}, \quad i = 2, \ldots, G.$$

In this case the ZR–VAR model is permuted so that (4.28) holds, and thus, Algorithm 6 can be employed.

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

Figure 4.6: The sequence of proper subset models generated by Algorithm 6, for $G = 3$ and $K = 4$.

The computational burden of the ProperSubsets procedure can be further reduced by utilizing previous computations. Assume the proper subsets VAR model corresponding to $(S_1, \ldots, S_{G-1}, S_G)$ has been estimated. Consider now the estimation of the proper subsets VAR model corresponding to $(S_1, \ldots, S_{G-1}, \tilde{S}_G)$, where $\tilde{S}_G = (S_G \quad e_{v_{k_G+1}})$. For example, in Figure 4.6, this is the case when moving from step 15 to step 16. Let $\tilde{y}_{BG} = (\psi_{BG} \quad \tilde{\tilde{y}}_{BG})^T$ and $\tilde{v}_{BG} = (\upsilon_{BG} \quad \tilde{v}^*_{BG})^T$, where $\tilde{\tilde{y}}_{BG}, \tilde{v}^*_{BG} \in \mathbb{R}^{K-\tilde{k}_G}$ and $\tilde{k}_G = k_G + 1$. That is ψ_{BG} and υ_{BG} are the first elements of $\tilde{y}_{BG}$ and $\tilde{v}_{BG}$, respectively. Notice that from $\tilde{k}_G = k_G + 1$, it implies that $K - k_i < K - k_G$ for $i = 1, \ldots, G-1$. Thus, in (4.32a),

υ_{BG} corresponds to a zero entry and therefore,

$$\tilde{v}_{Bi} = (\tilde{y}_{Bi} - \sum_{j=i+1}^{G-1} c_{ij}(0 \quad I_{K-k_i})\tilde{v}_{Bj} - c_{iG}(0 \quad I_{K-k_i})\tilde{v}^*_{BG})/c_{ii} \quad \text{for} \quad i = G-1,\ldots,1.$$

The recursive updating formulae (4.32a) become

$$\tilde{\tilde{v}}_{BG} = \tilde{\tilde{y}}_{BG}/c_{GG} \equiv \tilde{v}^*_{BG} \quad \text{and} \quad \tilde{\tilde{v}}_{Bi} = \tilde{v}_{Bi} \quad \text{for} \quad i = G-1,\ldots,1.$$

Now,

$$\begin{aligned}\tilde{\tilde{y}}_i &= \tilde{y}_{Ai} - \sum_{j=i+1}^{G-1} c_{ij} \begin{pmatrix} 0 & 0 \\ I_{k_i-k_j} & 0 \end{pmatrix} \tilde{v}_{Bj} - c_{iG} \begin{pmatrix} 0 & 0 \\ I_{k_i-k_G-1} & 0 \end{pmatrix} \tilde{v}^*_{BG} \\ &= \tilde{y}_i + (c_{iG}\upsilon_{BG})e_{k_G+1},\end{aligned}$$

where $\tilde{y}_i = R_i\hat{\beta}_i$, i.e. the righthand-side of (4.32b). The estimation of the proper subsets VAR model comes from the solution of the triangular systems

$$\tilde{R}_G\hat{\beta}^*_G = \begin{pmatrix} \tilde{y}_{AG} \\ \psi_{BG} \end{pmatrix} \quad \text{and} \quad R_i\hat{\beta}^*_i = \tilde{\tilde{y}}_i, \quad \text{for} \quad i = G-1,\ldots,1.$$

The computational cost of the QRDs (4.12) is also reduced in the general subsets case where $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_G$. The QRD of $R^{(i)} = RS_i$ is equivalent to re-triangularizing the smaller in size matrix R_{i+1} after deleting columns. Notice that $RS_i = RS_{i+1}S^T_{i+1}S_i$ and

$$\begin{aligned}Q^T_{i+1}RS_i &= Q^T_{i+1}RS_{i+1}S^T_{i+1}S_i \\ &= \begin{pmatrix} R_{i+1} \\ 0 \end{pmatrix} S^T_{i+1}S_i \\ &= \begin{pmatrix} R_{i+1}S^*_i \\ 0 \end{pmatrix}.\end{aligned}$$

Here $S^*_i = S^T_{i+1}S_i$ is of order $k_{i+1} \times k_i$ and selects the subset of S_{i+1} and in turn the selected columns from R_{i+1}. Now, computing the QRD

$$\hat{Q}^T_i(R_{i+1}S^*_i) = \begin{pmatrix} R_i \\ 0 \end{pmatrix} \begin{matrix} k_i \\ k_{i+1}-k_i \end{matrix} \tag{4.33}$$

it follows that the orthogonal Q^T_i of the QRD (4.12) is given by $Q^T_i = \check{Q}^T_i Q^T_{i+1}$, where

$$\check{Q}^T_i = \begin{pmatrix} \hat{Q}^T_i & 0 \\ 0 & I_{K-k_{i+1}} \end{pmatrix}.$$

Thus, following the initial QRD of $R^{(G)} = RS_G$, the remaining QRDs of $R^{(i)}$ are computed by (4.33) for $i = G-1,\ldots,1$.

Consider the case where $S_G \subseteq \cdots \subseteq S_2 \subseteq S_1$. Computations are simplified if the GLLSP (4.10) is expressed as

$$\underset{V,\{\beta_i\}}{\operatorname{argmin}} \|V\|_F^2 \quad \text{subject to } \operatorname{vec}(\tilde{Y}) = (\oplus_i L^{(i)}) \operatorname{vec}(\{\beta_i\}) + \operatorname{vec}(VC^T), \tag{4.34}$$

where $L^{(i)} = LS_i$ and now, L and C are lower triangular [72]. Thus, instead of (4.6), the QL decomposition of X needs to be computed:

$$\bar{Q}^T X = \begin{pmatrix} 0 \\ L \end{pmatrix} \begin{matrix} M-K \\ K \end{matrix}, \quad \text{with} \quad \bar{Q}^T (Y \quad U) = \begin{pmatrix} \hat{Y} & \hat{U} \\ \tilde{Y} & \tilde{U} \end{pmatrix} \begin{matrix} M-K \\ K \end{matrix}.$$

Furthermore, the QL decomposition

$$Q_i^T L^{(i+1)} = \begin{pmatrix} 0 \\ L_{i+1} \end{pmatrix} \begin{matrix} K-k_i \\ k_i \end{matrix}$$

can be seen as the re-triangularization of L_i after deleting columns [70]. If S_{i+1} is a proper subset of S_i, i.e. $S_i = (S_i^* \quad S_{i+1})$, then L_{i+1} is the trailing lower $k_{i+1} \times k_{i+1}$ sub-matrix of L_i [72]. Notice that if (4.9) rather than (4.34) is used, then R_{i+1} derives from the more computational expensive (updating) QRD

$$Q_{i+1}^T \begin{pmatrix} R_{i1} \\ R_{i2} \end{pmatrix} = \begin{pmatrix} R_{i+1} \\ 0 \end{pmatrix}, \text{ where } \quad R_i = \begin{matrix} \\ \\ \end{matrix} \begin{matrix} k_i - k_{i+1} & k_{i+1} \end{matrix} \begin{pmatrix} R_i^* & R_{i1} \\ 0 & R_{i2} \end{pmatrix} \begin{matrix} k_i - k_{i+1} \\ k_{i+1} \end{matrix}.$$

4.6 Conclusion and future work

Efficient numerical and computational strategies for deriving the subset VAR models have been proposed. The VAR model with zero-coefficient restriction, i.e. ZR–VAR model, has been formulated as a SUR model. Initially, the QR decomposition is employed to reduce to zero $M-K$ observations of the VAR, and consequently, ZR–VAR model. This results in an equivalent and smaller-size estimation problem. The numerical estimation of the ZR–VAR model has been derived. Within this context an efficient variable-downdating strategy has been presented. The main computational tool of the estimation procedures is the Generalized QR decomposition.

An algorithm which generates all subset VAR models by efficiently moving from one model to another has been described. The algorithm generates a regression tree and avoids estimating each ZR–VAR model afresh. However, this strategy is computational infeasible even for modest size VAR models due to the exponential number $(2^{pG}-1)$ of sub-models that derives. An alternative block-version of the algorithm generates 2^G sub-models. At each step of the block-strategy a whole block of observations is deleted from the VAR model. The deletion of the ith block is equivalent in deleting the ith row from each coefficient matrix $\Phi_1,\ldots,\Phi_p$ in (4.1).

Two special cases of subset VAR models which are derived by taking advantage of the Toeplitz structure of the data matrix and the Kronecker structure of the variance-covariance matrix have been presented. Both of them require $O(G)$ less computational complexity than generating the models afresh. The first special case derives $(pG+2)2^{(pG-1)}-1$ subset VAR models by deleting the same variable from each block of the reduced ZR–VAR model. The second case is based on deleting subsets of variables from each block of the regressors in the ZR–VAR model (4.10). An algorithm that derives all proper subsets models given the initial VAR model has been designed. This algorithm generates $\sum_{i=1}^{\min(K,G-1)} C_{G-2}^{i-1} C_K^i 2^{K-i}$ models, when G is greater than one. In both cases the computational burden of deriving the generalized QR decomposition (4.11), and thus, estimating the submodels, is significantly reduced. In the former case only a single column-downdating of a triangular matrix is required. This is done efficiently using Givens rotations. The second case performs no factorizations, but efficiently computes the coefficients using (4.32).

The new algorithms allow the investigation of more subset VAR models when trying to identify the lag structure of the process in (4.1). The implementation and application of the proposed algorithms need to be pursued. These methods are based on a regression tree structure. This suggest that a branch and bound strategy which finds the best models without generating the whole regression tree should be considered [39]. Within this context the use of parallel computing to allow the tackling of large scale models merits investigation.

The permutations of the exogenous matrices $X_1,\ldots,X_G$ in the SUR model (4.5) can provide $G!$ new subset VAR models. If the ZR–VAR model (4.9) has been already estimated, then the computational cost of estimating these new subset models will be significantly lower since the exogenous matrices $X_1,\ldots,X_G$ have been already factorized ($i=1,\ldots,G$). Furthermore, the efficient computation of the RQ factorization in (4.11b) should be investigated for some permutations, e.g. when two adjacent exogenous matrices are permuted. Strategies that generate efficiently all $G!$ sub-models and the best ones using the branch and bound method are currently under investigation.

4.A Appendix

Lemma 3 *The recurrence*

$$f(K,G) = \begin{cases} 0 & \text{if} \quad K = 0 \quad \text{and} \quad G \geq 1, \\ 1 & \text{if} \quad K \geq 0 \quad \text{and} \quad G = 0, \\ \sum_{i=0}^{K-1} \sum_{j=0}^{G} f(i,j) & \text{if} \quad K \geq 1 \quad \text{and} \quad G \geq 1, \end{cases} \tag{4.35}$$

denotes the number of proper subsets models defined by (4.28) *with maximum K variables,* $(v_1, \ldots, v_K)$ *and G blocks.*

PROOF. The proof is by double induction. For $K = 1$ and $G \geq 1$,

$$f(1,G) = \sum_{j=0}^{G} f(0,j) = f(0,0) = 1.$$

This is the case where there is only one possible model, i.e. $S_i = [v_1]$, for $i = 1, \ldots, G$. The inductive hypothesis is that Lemma 1 is true for some $K, G \geq 1$. It has to be proven that Lemma 1 is also true for $K+1$ and $G \geq 1$.

Let v_{K+1} be the new variable. First, there is a new model defined by $S_i = [v_{K+1}]$, for $i = 1, \ldots, G$. Furthermore, from the inductive hypothesis there are $f(K,G)$ models which do not include v_{K+1}. Consider now all the possibilities for which S_j includes v_{K+1}, when $j = 1, \ldots, G$. From the proper subsets property, it follows that $S_1, \ldots, S_{j-1}$ include also v_{K+1}. Furthermore, if two of these sets are different, then, before adding v_{K+1}, there are i $(2 \leq i \leq j)$ and $\alpha \geq 1$, so that, $S_{i-1} = [v_1, \ldots, v_\rho, v_{\rho+1}, \ldots, v_{\rho+\alpha}]$ and $S_i = [v_1, \ldots, v_\rho]$. Now, if v_{K+1} is included, then it follows $S_{i-1} = [v_1, \ldots, v_\rho, v_{\rho+1}, \ldots, v_{\rho+\alpha}, v_{K+1}]$ and $S_i = [v_1, \ldots, v_\rho, v_{K+1}]$. However, this contradicts the definition in (4.28) and therefore $S_1 \equiv \cdots \equiv S_{j-1} \equiv S_j$. Thus, the number of possibilities for which S_j includes v_{K+1} is the number of models with maximum K variables and $G - j + 1$ blocks. From the inductive hypothesis this number is given by $f(K, G - j + 1)$.

Hence, the number of proper subsets models defined by (4.28) with maximum $K+1$ variables and G blocks is given by

$$\begin{aligned} 1 + f(K,G) + \sum_{j=1}^{G} f(K, G-j+1) &= \sum_{i=0}^{K-1} \sum_{j=0}^{G} f(i,j) + \sum_{j=1}^{G} f(K, G-j+1) + f(K,0) \\ &= \sum_{i=0}^{K} \sum_{j=0}^{G} f(i,j) = f(K+1, G), \end{aligned}$$

which completes the proof. ∎

Lemma 4 *The recurrence* (4.35) *simplifies to*

$$f(K,G)=\begin{cases}1 & \text{if} \quad G=0,\\ 2^K-1 & \text{if} \quad G=1,\\ \sum_{i=1}^{min(K,G-1)} C_{G-2}^{i-1}C_K^i 2^{K-i} & \text{if} \quad G\geq 2.\end{cases}$$

PROOF. Consider the recurrence (4.35). If $K\geq 1$, then $f(K,G)=\sum_{j=0}^{G-1} f(K-1,j)+2f(K-1,G)$. Thus, (4.35) can be written as $F_K^T=F_{K-1}^T\Lambda=F_0^T\Lambda^K$, where $F_K\in\mathbb{R}^{G+1}$, $F_0=(1,0,\ldots,0)^T$ and $\Lambda\in\mathbb{R}^{(G+1)\times(G+1)}$ is given by

$$\Lambda=\begin{pmatrix}1 & 1 & 1 & \cdots & 1 & 1\\ & 2 & 1 & \cdots & 1 & 1\\ & & 2 & \cdots & 1 & 1\\ & & & \ddots & \vdots & \vdots\\ & & & & 2 & 1\\ & & & & & 2\end{pmatrix}.$$

Now, consider the computation of Λ^K which requires the Jordan form of Λ [115, pp. 335–341]. That is, $\Lambda=\Theta\bar{D}\Theta^{-1}$, where $\bar{D}=D+N$, $D\in\mathbb{R}^{(G+1)\times(G+1)}$ is diagonal, $N\in\mathbb{R}^{(G+1)\times(G+1)}$, $N^G=0$ and $DN=ND$. Specifically

$$D=\begin{pmatrix}1 & & & & & \\ & 2 & & & & \\ & & 2 & & & \\ & & & \ddots & & \\ & & & & 2 & \\ & & & & & 2\end{pmatrix} \quad \text{and} \quad N=\begin{pmatrix}0 & 0 & & & & \\ & 0 & 1 & & & \\ & & 0 & 1 & & \\ & & & \ddots & \ddots & \\ & & & & 0 & 1\\ & & & & & 0\end{pmatrix}.$$

The upper-triangular matrix Λ has the two eigenvalues $\lambda=1$ and $\mu=2$ with the multiplicities 1 and G, respectively. The eigenvectors $v_1=(1,0,0,\ldots,0)^T$ and $v_2^{(1)}=(1,1,0,\ldots,0)^T$ corresponds to the eigenvalues λ and μ, respectively. The remaining eigenvectors $v_2^{(i)}$ $(i=2,\ldots,G)$, corresponding to the multiple eigenvalue μ, are recursively computed by deriving the solution w of $(\Lambda-2I_{G+1})w=v_2^{(i-1)}$. Thus, $\Theta=[v_1,v_2^{(1)},\ldots,v_2^{(G)}]$ and is given by

$$\theta_{i,j}=\begin{cases}1, & \text{if} \quad i=0, \quad j=0,1 \quad \text{or} \quad i=1, \quad j=0,\\ 0, & \text{if} \quad i=0,1, \quad j\geq 2 \quad \text{or} \quad i=1, \quad j=0 \quad \text{or} \quad i\geq 2, \quad j<i,\\ (-1)^{i+j}C_{j-2}^{i-2}, & \text{if} \quad i\geq 2, \quad j\geq i,\end{cases}$$

where $C_n^k = n!/k!(n-k)!$. Furthermore Θ^{-1} is given by

$$\theta_{i,j}^{-1} = \begin{cases} 1 & \text{if} \quad i=j=0,1, \\ -1 & \text{if} \quad i=0, \quad j=1, \\ 0 & \text{if} \quad i=0,1, \quad j\geq 2 \quad \text{or} \quad i=1, \quad j=0 \quad \text{or} \quad i\geq 2, \quad j<i, \\ C_{j-2}^{i-2} & \text{if} \quad i\geq 2, \quad j\geq i. \end{cases}$$

That is

$$\Theta^{-1} \equiv \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & C_0^0 & C1^0 & C_2^0 & \dots & C_{G-3}^0 & C_{G-2}^0 \\ 0 & 0 & 0 & C_1^1 & C_2^1 & \dots & C_{G-3}^1 & C_{G-2}^1 \\ 0 & 0 & 0 & 0 & C_2^2 & \dots & C_{G-3}^2 & C_{G-2}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & C_{G-3}^{G-3} & C_{G-2}^{G-3} \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & C_{G-2}^{G-2} \\ & & & & & & & \end{pmatrix}.$$

Now, $\Lambda^K = (\Theta\bar{D}\Theta^{-1})^K = \Theta(D+N)^K\Theta^{-1}$ and

$$(D+N)^K = \begin{cases} C_K^0 D^K + C_K^1 D^{K-1}N + \cdots + C_K^K N^K, & \text{if} \quad K < G-1, \\ C_K^0 D^K + C_K^1 D^{K-1}N + \cdots + C_K^{K-G+1} D^{K-G+1}N^{G-1}, & \text{if} \quad K \geq G-1. \end{cases}$$

For the case $K \geq G-1$, the latter can be written as

$$(D+N)^K = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 2^K C_K^0 & 2^{K-1}C_K^1 & 2^{K-2}C_K^2 & \dots & 2^{K-G+1}C_K^{G-1} \\ 0 & 0 & 2^K C_K^0 & 2^{K-1}C_K^1 & \dots & 2^{K-G+2}C_K^{G-2} \\ 0 & 0 & 0 & 2^K C_K^0 & \dots & 2^{K-G+3}C_K^{G-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 2^K C_K^0 \end{pmatrix}. \tag{4.36}$$

In the case $K < G-1$, (4.36) has band-diagonal structure with band-width $K+1$.

Now $F_0 = (1,0,0,\dots,0)^T$ and $F_K = F_0\Lambda^K$. Thus, only the first row of the matrix Λ^K needs to be computed. Furthermore, the first row of Θ is $(1,1,0,\dots,0)$ which implies that the first row of the product $\Theta(D+N)^K$, say r, is given by

$$r = \begin{cases} (1, 2^K C_K^0, 2^{K-1}C_K^1, \dots, 2^0 C_K^K, 0, \dots, 0), & \text{if} \quad k < G-1 \\ (1, 2^K C_K^0, 2^{K-1}C_K^1, \dots, 2^{K-G+1}C_K^{K-G+1}), & \text{if} \quad K \geq G-1. \end{cases} \tag{4.37}$$

From $r\Theta^{-1} = F_K$ it follows that

$$f(K,G) = \begin{cases} 1, & \text{if} \quad G = 0, \\ 2^K - 1, & \text{if} \quad G = 1, \\ \sum_{i=1}^{\min(K,G-1)} C_{G-2}^{i-1} C_K^i 2^{K-i}, & \text{if} \quad G \geq 2, \end{cases}$$

which completes the proof. ■

Chapter 5

Estimating all possible SUR models with permuted exogenous data matrices

Abstract:

The Vector Autoregressive (VAR) process with zero coefficient constraints can be formulated as a Seemingly Unrelated Regressions (SUR) model. Within the context of subset VAR model selection a computationally efficient strategy to generate and estimate all $G!$ SUR models when permuting the exogenous data matrices is proposed, where G is the number of the regression equations. The combinatorial algorithm is based on orthogonal transformations, exploits the particular structure of the modified models and avoids the estimation of these models afresh by utilizing previous computation. Theoretical measured of complexity are derived to prove the efficiency of the proposed algorithm.

[1]This chapter is a reprint of the paper: C. Gatu and E.J. Kontoghiorghes. Estimating all possible SUR models with permuted exogenous data matrices: computational aspects. *Computational Management Science*, Spring 2004. (To be submitted).

5.1 Introduction

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

$$y_i = X_i\beta_i + u_i, \quad i = 1,\dots,G, \tag{5.1}$$

where $y_i \in \mathbb{R}^M$, $X_i \in \mathbb{R}^{M\times k_i}$, $\beta_i \in \mathbb{R}^{k_i}$ and $u_i \in \mathbb{R}^M$. The expectation of u_i is zero, i.e. $\mathrm{E}(u_i) = 0$ and $\mathrm{E}(u_i u_j^T) = \sigma_{ij} I_M$ $(i,j = 1,\dots,G)$ [63, 72, 114]. In compact form, the SUR model can be written as

$$\mathrm{vec}(Y) = (\oplus_{i=1}^G X_i)\,\mathrm{vec}\,(\{\beta_i\}_G) + \mathrm{vec}(U), \quad \mathrm{vec}(U) \sim (0, \Sigma\otimes I_M), \tag{5.2}$$

where vec is the vector operator, $Y = (y_1 \dots y_G)$, $\oplus_{i=1}^G X_i = \mathrm{diag}(X_1,\dots,X_G)$, $\{\beta_i\}_G$ denotes the set $\{\beta_1,\dots,\beta_G\}$, $U = (u_1 \dots u_G)$ and $\Sigma = [\sigma_{ij}] \in \mathbb{R}^{G\times G}$ has full rank [33, 63]. For notational convenience the direct sum $\oplus_{i=1}^G$ and the set $\{\cdot\}_G$ are abbreviated to $\oplus_i$ and $\{\cdot\}$, respectively. The best linear unbiased estimator (BLUE) of the SUR model (5.2) comes from the solution of the Generalized Linear Least Squares Problem (GLLSP)

$$\underset{V,\{\beta_i\}}{\mathrm{argmin}} \|V\|_F^2 \quad \text{subject to } \mathrm{vec}(Y) = (\oplus_i X_i)\,\mathrm{vec}\,(\{\beta_i\}) + \mathrm{vec}(VC^T). \tag{5.3}$$

Here $\Sigma = CC^T$, the random $V \in \mathbb{R}^{M\times G}$ is defined as $VC^T = U$ which implies $\mathrm{vec}(V) \sim (0, \quad I_{GM})$, and $\|\cdot\|_F$ denotes Frobenius norm i.e. $\|V\|_F^2 = \sum_{i=1}^M \sum_{j=1}^G V_{i,j}^2$ [72, 77, 98, 99]. The upper-triangular $C \in \mathbb{R}^{G\times G}$ is the Cholesky factor of Σ. For the solution of (5.3) consider the Generalized QR Decomposition (GQRD) of the matrices $\oplus_i X_i$ and $C\otimes I_M$:

$$Q^T(\oplus_i X_i) = \begin{pmatrix} \oplus_i R_i \\ 0 \end{pmatrix} \begin{matrix} K^* \\ GM-K^* \end{matrix} \tag{5.4a}$$

and

$$Q^T(C\otimes I_M)P = W \equiv \begin{matrix} \\ K^* \\ GM-K^* \end{matrix} \overset{\begin{matrix} K^* & GM-K^* \end{matrix}}{\begin{pmatrix} W_{11} & W_{12} \\ 0 & W_{22} \end{pmatrix}}, \tag{5.4b}$$

where $K^* = \sum_{i=1}^G k_i$, Q and P are $GM\times GM$ orthogonal matrices and $\oplus R_i$ and W are upper triangular of order K^* and GM, respectively. Now, since $\|V\|_F^2 = \|P^T\,\mathrm{vec}(V)\|^2$ and writting

$$Q^T\,\mathrm{vec}(Y) = \begin{pmatrix} \mathrm{vec}\,(\{y_{Ai}\}) \\ \mathrm{vec}\,(\{y_{Bi}\}) \end{pmatrix} \begin{matrix} K^* \\ GM-K^* \end{matrix} \quad \text{and} \quad P^T\,\mathrm{vec}(V) = \begin{pmatrix} \mathrm{vec}\,(\{v_{Ai}\}) \\ \mathrm{vec}\,(\{v_{Bi}\}) \end{pmatrix} \begin{matrix} K^* \\ GM-K^* \end{matrix},$$

the GLLSP (5.3) is equivalent to

$$\underset{\{v_{Ai}\},\{v_{Bi}\},\{\beta_i\}}{\operatorname{argmin}} \sum_{i=1}^{G} (\|v_{Ai}\|^2 + \|v_{Bi}\|^2) \quad \text{subject to}$$
$$\begin{pmatrix} \operatorname{vec}(\{y_{Ai}\}) \\ \operatorname{vec}(\{y_{Bi}\}) \end{pmatrix} = \begin{pmatrix} \oplus_i R_i \\ 0 \end{pmatrix} \operatorname{vec}(\{\beta_i\}) + \begin{pmatrix} W_{11} & W_{12} \\ 0 & W_{22} \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\{v_{Ai}\}) \\ \operatorname{vec}(\{v_{Bi}\}) \end{pmatrix}, \tag{5.5}$$

where $\|\cdot\|$ denotes the Euclidian norm [67]. From the constraint in (5.5) it follows that $\operatorname{vec}(\{v_{Bi}\})$ and BLUE of $\operatorname{vec}(\{\beta_i\})$ are computed from the solution of the triangular system

$$\begin{pmatrix} \oplus_i R_i & W_{12} \\ 0 & W_{22} \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\{\hat{\beta}_i\}) \\ \operatorname{vec}(\{v_{Bi}\}) \end{pmatrix} = \begin{pmatrix} \operatorname{vec}(\{y_{Ai}\}) \\ \operatorname{vec}(\{y_{Bi}\}) \end{pmatrix}. \tag{5.6}$$

The Vector Autoregressive (VAR) model with zero-restriction (ZR–VAR) on the $K \times G$ coefficients can be formulated as the SUR model (5.2). Ideally, the enumeration and estimation of all possible $2^{KG} - 1$ ZR–VAR models is required. This is computationally infeasible even for modest values of K and G. Therefore, subclasses of these models are considered [41]. Within this context, $G!$ subset VAR models can be generated by permuting the exogenous matrices of the ZR–VAR model.

In the next section, the numerical solution of the permuted SUR model is presented. Special cases together with their numerical solutions taking advantage of the structure of the models are also described. An efficient algorithm for estimating all $G!$ permuted SUR models is given in Section 5.3. In Section 5.4, theoretical measures of complexity of the combinatorial algorithm are derived to prove the efficiency of the proposed algorithm. Finally, Section 5.5 concludes.

5.2 Numerical estimation of the permuted SUR model

The GQRD (5.4) is the main computational component for obtaining the BLUE of (5.2). In (5.4a), $R_i \in \mathbb{R}^{k_i \times k_i}$ is the upper-triangular factor in the QRD of X_i. That is,

$$Q_i^T X_i = \begin{pmatrix} R_i \\ 0 \end{pmatrix} \begin{matrix} k_i \\ M-k_i \end{matrix}, \quad \text{with} \quad Q_i^T = \begin{pmatrix} Q_{Ai}^T \\ Q_{Bi}^T \end{pmatrix} \begin{matrix} k_i \\ M-k_i \end{matrix} \quad (i = 1, \ldots, G), \tag{5.7}$$

where $Q_i \in \mathbb{R}^{M \times M}$ is orthogonal and Q in (5.4a) is defined by

$$Q = (\oplus_i Q_{Ai} \quad \oplus_i Q_{Bi}) = \begin{pmatrix} Q_{A1} & & & Q_{B1} & & \\ & \ddots & & & \ddots & \\ & & Q_{AG} & & & Q_{BG} \end{pmatrix}. \tag{5.8}$$

For the RQD (5.4b), initially the matrix Q is multiplied from the right of $Q^T(C\otimes I_M)$ to give

$$Q^T(C\otimes I_M)Q=\begin{pmatrix} W^{(1,0)} & W^{(2,0)} \\ W^{(3,0)} & W^{(4,0)} \end{pmatrix} \equiv W_0$$

$$= \left(\begin{array}{cccc|cccc} \overset{k_1}{W^{(1,0)}_{1,1}} & \overset{k_2}{W^{(1,0)}_{1,2}} & \overset{...}{\dots} & \overset{k_G}{W^{(1,0)}_{1,G}} & \overset{M-k_1}{0} & \overset{M-k_2}{W^{(2,0)}_{1,2}} & \overset{...}{\dots} & \overset{M-k_G}{W^{(2,0)}_{1,G}} \\ 0 & W^{(1,0)}_{2,2} & \dots & W^{(1,0)}_{2,G} & 0 & 0 & \dots & W^{(2,0)}_{2,G} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & W^{(1,0)}_{G,G} & 0 & 0 & \dots & 0 \\ \hline 0 & W^{(3,0)}_{1,2} & \dots & W^{(3,0)}_{1,G} & W^{(4,0)}_{1,1} & W^{(4,0)}_{1,2} & \dots & W^{(4,0)}_{1,G} \\ 0 & 0 & \dots & W^{(3,0)}_{1,G} & 0 & W^{(4,0)}_{2,2} & \dots & W^{(4,0)}_{1,G} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & W^{(4,0)}_{G,G} \end{array}\right) \begin{matrix} k_1 \\ k_2 \\ \vdots \\ k_G \\ M-k_1 \\ M-k_2 \\ \vdots \\ M-k_G \end{matrix} . \tag{5.9}$$

Here

$$\begin{pmatrix} W^{(1,0)}_{i,i} & W^{(2,0)}_{i,i} \\ W^{(3,0)}_{i,i} & W^{(4,0)}_{i,i} \end{pmatrix} = c_{ii}I_M$$

and

$$\begin{pmatrix} W^{(1,0)}_{i,j} & W^{(2,0)}_{i,j} \\ W^{(3,0)}_{i,j} & W^{(4,0)}_{i,j} \end{pmatrix} = c_{ij}Q_i^TQ_j,$$

for $i,j=1,\dots,G, i<j$. Finally, the RQD of (5.9) is computed in $G-1$ steps. The ith ($i=1,\dots,G-1$) step computes the factorization

$$\left(\begin{array}{c|cccc} W^{(1,i-1)}_{1,i+1} & W^{(2,i-1)}_{1,1} & W^{(2,i-1)}_{1,2} & \dots & W^{(2,i-1)}_{1,i} \\ W^{(1,i-1)}_{2,i+1} & W^{(2,i-1)}_{2,1} & W^{(2,i-1)}_{2,2} & \dots & W^{(2,i-1)}_{2,i} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W^{(1,i-1)}_{i+1,i+1} & W^{(2,i-1)}_{i+1,1} & W^{(2,i-1)}_{i+1,2} & \dots & W^{(2,i-1)}_{i+1,i} \\ \hline W^{(3,i-1)}_{1,i+1} & W^{(4,i-1)}_{1,1} & W^{(4,i-1)}_{1,2} & \dots & W^{(4,i-1)}_{1,i} \\ W^{(3,i-1)}_{2,i+1} & 0 & W^{(4,i-1)}_{2,2} & \dots & W^{(4,i-1)}_{2,i} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W^{(3,i-1)}_{i,i+1} & 0 & 0 & \dots & W^{(4,i-1)}_{i,i} \end{array}\right) P_i = \left(\begin{array}{c|cccc} W^{(1,i)}_{1,i+1} & W^{(2,i)}_{1,1} & W^{(2,i)}_{1,2} & \dots & W^{(2,i)}_{1,i} \\ W^{(1,i)}_{2,i+1} & W^{(2,i)}_{2,1} & W^{(2,i)}_{2,2} & \dots & W^{(2,i)}_{2,i} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W^{(1,i)}_{i+1,i+1} & W^{(2,i)}_{i+1,1} & W^{(2,i)}_{i+1,2} & \dots & W^{(2,i)}_{i+1,i} \\ \hline 0 & W^{(4,i)}_{1,1} & W^{(4,i)}_{1,2} & \dots & W^{(4,i)}_{1,i} \\ 0 & 0 & W^{(4,i)}_{2,2} & \dots & W^{(4,i)}_{2,i} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & W^{(4,i)}_{i,i} \end{array}\right) . \tag{5.10}$$

The matrix P_i is the product of i orthogonal matrices. That is, $P_i = P^*_{i,1} \cdots P^*_{i,i}$. Here,

$$P^*_{i,j} = \begin{array}{c} \begin{array}{cccc} {\scriptstyle k_{i+1}} & {\scriptstyle \lambda_{i,j}} & {\scriptstyle M-k_{i-j+1}} & {\scriptstyle \mu_{i,j}} \end{array} \\ \begin{pmatrix} P^{(A)}_{i,j} & 0 & P^{(B)}_{i,j} & 0 \\ 0 & I_{\lambda_{i,j}} & 0 & 0 \\ P^{(C)}_{i,j} & 0 & P^{(D)}_{i,j} & 0 \\ 0 & 0 & 0 & I_{\mu_{i,j}} \end{pmatrix} \end{array} \begin{array}{l} {\scriptstyle k_{i+1}} \\ {\scriptstyle \lambda_{i,j}} \\ {\scriptstyle M-k_{i-j+1}} \\ {\scriptstyle \mu_{i,j}} \end{array},$$

where $\lambda_{i,j} = \sum_{l=1}^{i-j}(M-k_l)$, $\mu_{i,j} = \lambda_{i,0} - \lambda_{i,j-1}$ and $j = 1, \ldots, i$. Furthermore,

$$\begin{array}{c} \begin{array}{cc} {\scriptstyle k_{i+1}} & {\scriptstyle M-k_{i-j+1}} \end{array} \\ \begin{pmatrix} W^{(3,i-1)}_{i-j+1,i+1} & W^{(4,i-1)}_{i-j+1,i-j+1} \end{pmatrix} \end{array} P_{i,j} = \begin{array}{c} \begin{array}{cc} {\scriptstyle k_{i+1}} & {\scriptstyle M-k_{i-j+1}} \end{array} \\ \begin{pmatrix} 0 & W^{(4,i,j)}_{i-j+1,i-j+1} \end{pmatrix} \end{array},$$

$W^{(4,i,j)}_{i-j+1,i-j+1}$ is upper-triangular and

$$P_{i,j} = \begin{pmatrix} P^{(A)}_{i,j} & P^{(B)}_{i,j} \\ P^{(C)}_{i,j} & P^{(D)}_{i,j} \end{pmatrix}.$$

Figure 5.1 illustrates this strategy to compute (5.10), where $G = 5$. An arc indicates the columns affected from the application of $P^*_{i,j}$ ($i = 1, \ldots, G-1$ and $j = 1, \ldots, i$). Notice that $P^*_{i,j}$ can be considered as block-version of a Givens rotation [31, 33, 63]. Thus, W in (5.4b) is given by

$$W = \left(\begin{array}{cccc|cccc} W^{(1,G-1)}_{1,1} & W^{(1,G-1)}_{1,2} & \ldots & W^{(1,G-1)}_{1,G} & W^{(2,G-1)}_{1,1} & W^{(2,G-1)}_{1,2} & \ldots & W^{(2,G-1)}_{1,G} \\ 0 & W^{(1,G-1)}_{2,2} & \ldots & W^{(1,G-1)}_{2,G} & W^{(2,G-1)}_{2,1} & W^{(2,G-1)}_{2,2} & \ldots & W^{(2,G-1)}_{2,G} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & W^{(1,G-1)}_{G,G} & W^{(2,G-1)}_{G,1} & W^{(2,G-1)}_{G,2} & \ldots & W^{(2,G-1)}_{G,G} \\ \hline 0 & 0 & \ldots & 0 & W^{(4,G-1)}_{1,1} & W^{(4,G-1)}_{1,2} & \ldots & W^{(4,G-1)}_{1,G} \\ 0 & 0 & \ldots & 0 & 0 & W^{(4,G-1)}_{2,2} & \ldots & W^{(4,G-1)}_{2,G} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & W^{(4,G-1)}_{G,G} \end{array}\right).$$

The matrix P is not constructed explicitely.

An equivalent strategy for computing the RQD of (5.9) annihilates the blocks of form $W^{(3,i)}$ row-by-row and from bottom-to-top. This is performed in $G-1$ steps. The ith step computes the

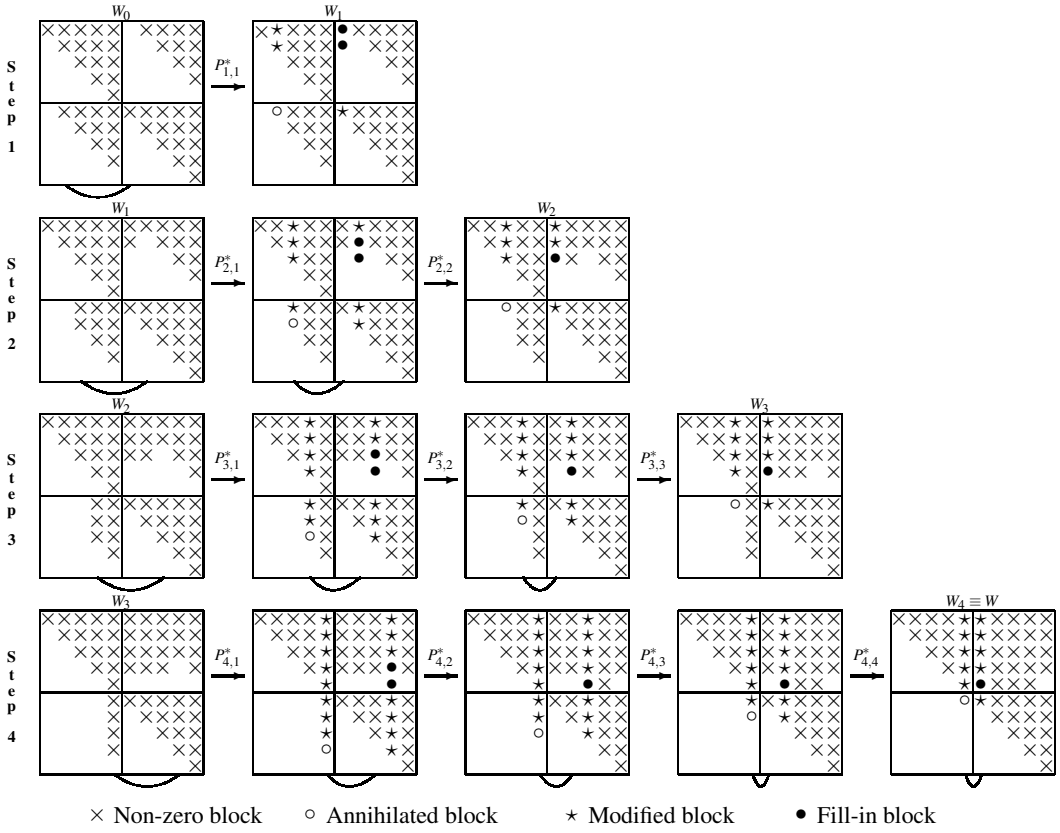


Figure 5.1: The $G-1$ steps for re-triangularizing (5.9), where $G=5$.

factorization

$$\left(\begin{array}{ccc|c} W^{(1,i-1)}_{G-i,G-i+1} & \cdots & W^{(1,i-1)}_{G-i,G} & 0 \\ W^{(1,i-1)}_{G-i+1,G-i+1} & \cdots & W^{(1,i-1)}_{G-i+1,G} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & W^{(1,i-1)}_{G,G} & 0 \\ \hline W^{(3,i-1)}_{G-i,G-i+1} & \cdots & W^{(3,i-1)}_{G-i,G} & W^{(4,i-1)}_{G-i,G-i} \end{array}\right) \tilde{P}_i = \left(\begin{array}{ccc|c} W^{(1,i)}_{G-i,G-i+1} & \cdots & W^{(1,i)}_{G-i,G} & W^{(2,i)}_{G-i,G-i} \\ W^{(1,i)}_{G-i+1,G-i+1} & \cdots & W^{(1,i)}_{G-i+1,G} & W^{(2,i)}_{G-i+1,G-i} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & W^{(1,i)}_{G,G} & W^{(2,i)}_{G,G-i} \\ \hline 0 & \cdots & 0 & W^{(4,i)}_{G-i,G-i} \end{array}\right) \tag{5.11a}$$

and the modification

$$\left(\begin{array}{ccc|c} W_{1,G-i+1}^{(r,i-1)} & \cdots & W_{1,G}^{(r,i-1)} & W_{1,G-i}^{(r+1,i-1)} \\ \vdots & \ddots & \vdots & \vdots \\ W_{G-i-1,G-i+1}^{(r,i-1)} & \cdots & W_{G-i-1,G}^{(r,i-1)} & W_{G-i-1,G-i}^{(r+1,i-1)} \end{array}\right)\tilde{P}_i = \left(\begin{array}{ccc|c} W_{1,G-i+1}^{(r,i)} & \cdots & W_{1,G}^{(r,i)} & W_{1,G-i}^{(r+1,i)} \\ \vdots & \ddots & \vdots & \vdots \\ W_{G-i-1,G-i+1}^{(r,i)} & \cdots & W_{G-i-1,G}^{(r,i)} & W_{G-i-1,G-i}^{(r+1,i)} \end{array}\right), \tag{5.11b}$$

where $r = 1, 3$. The matrix $\tilde{P}_i$ is the product of i orthogonal matrices. That is, $\tilde{P}_i = \tilde{P}_{i,1}^* \cdots \tilde{P}_{i,i}^*$. Here,

$$\tilde{P}_{i,j}^* = \begin{array}{c} \begin{array}{cccc} \tilde{\lambda}_{i,j} & k_{G-i+j} & \tilde{\mu}_{i,j} & M-k_{G-i} \end{array} \\ \left(\begin{array}{cccc} I_{\tilde{\lambda}_{i,j-1}} & 0 & 0 & 0 \\ 0 & \tilde{P}_{i,j}^{(A)} & 0 & \tilde{P}_{i,j}^{(B)} \\ 0 & 0 & I_{\tilde{\mu}_{i,j}} & 0 \\ 0 & \tilde{P}_{i,j}^{(C)} & 0 & \tilde{P}_{i,j}^{(D)} \end{array}\right) \begin{array}{c} \tilde{\lambda}_{i,j} \\ k_{G-i+j} \\ \tilde{\mu}_{i,j} \\ M-k_{G-i} \end{array} \end{array},$$

where now, $\tilde{\lambda}_{i,j} = \sum_{l=1}^{j} k_{G-i+l}$, $\tilde{\mu}_{i,j} = \tilde{\lambda}_{i,i} - \tilde{\lambda}_{i,j}$, and $j = 1, \ldots, i$. Furthermore,

$$\begin{array}{cc} k_{G-i+j} & M-k_{G-i} \end{array} \\ \left(\begin{array}{cc} W_{G-i,G-i+j}^{(3,i-1)} & W_{G-i,G-i}^{(4,i-1)} \end{array}\right)\tilde{P}_{i,j} = \begin{array}{cc} k_{G-i+j} & M-k_{G-i} \end{array} \\ \left(\begin{array}{cc} 0 & W_{G-i,G-i}^{(4,i,j)} \end{array}\right),$$

$W_{G-i,G-i}^{(4,i,j)}$ is upper-triangular and

$$\tilde{P}_{i,j} = \begin{pmatrix} \tilde{P}_{i,j}^{(A)} & \tilde{P}_{i,j}^{(B)} \\ \tilde{P}_{i,j}^{(C)} & \tilde{P}_{i,j}^{(D)} \end{pmatrix}.$$

Figure 5.2 shows the stages of this annihilation strategy, where $G = 5$ and the ith stage ($i = 1, \ldots, G-1$) performs i annihilation. An arc is drawn between the columns of the blocks to be annihilated and the pivot column.

Let $\xi = (\xi_1, \ldots, \xi_G)$ denote a permutation of the indices $(1, \ldots, G)$. Consider the estimation of the permuted ZR–VAR model

$$\text{vec}(Y) = (\oplus_i X_{\xi_i})\,\text{vec}\,(\{\beta_{\xi_i}\}) + \text{vec}(U), \quad \text{vec}(U) \sim (0, \Sigma \otimes I_M), \tag{5.12}$$

after (5.2) has been estimated. As in (5.3) the BLUE of (5.12) comes from the solution of the GLLSP

$$\underset{V,\{\beta_{\xi_i}\}}{\text{argmin}} \|V\|_F^2 \quad \text{subject to } \text{vec}(Y) = (\oplus_i X_{\xi_i})\,\text{vec}\,(\{\beta_{\xi_i}\}) + \text{vec}(VC^T) \tag{5.13}$$

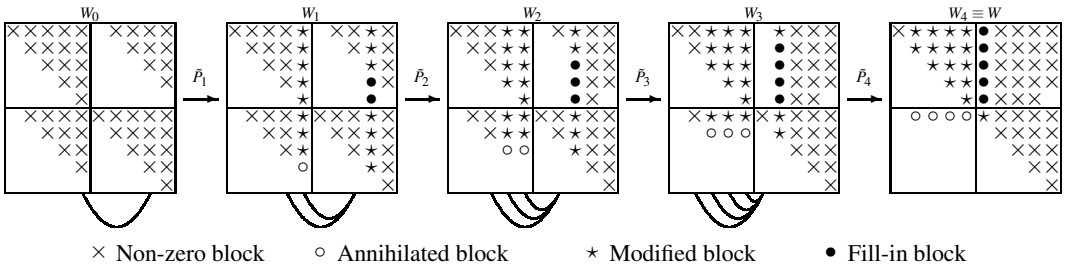


Figure 5.2: Alternative $G-1$ steps for re-triangularizing (5.9), where $G=5$.

which requires the computation of the GQRD of the matrices $\oplus_i X_{\xi_i}$ and $(C \otimes I_M)$. From (5.4a), (5.7) and (5.8) it follows that the QRD of $\oplus_i X_{\xi_i}$ is given by

$$Q_\xi^T(\oplus_i X_{\xi_i}) = \begin{pmatrix} R_{\xi_i} \\ 0 \end{pmatrix}, \tag{5.14}$$

where $Q_\xi = (\oplus_i Q_{A\xi_i} \quad \oplus_i Q_{B\xi_i})$. The RQD of $Q_\xi^T(C \otimes I_M)$ needs to be computed afresh.

The computational burden of computing the RQD of (5.9) can be reduced significantly in some cases. Consider the permutation ξ of the indices $1,\ldots,G$ such as $\xi_i = i$ for $i = 1,\ldots,t$ and $1 \le t \le G-2$. In this case, the matrix $Q_\xi^T(C \otimes I_M)Q_\xi$ has the structure

$$\widehat{W}_0 = \begin{pmatrix} \widehat{W}^{(1,0)} & \widehat{W}^{(2,0)} \\ \widehat{W}^{(3,0)} & \widehat{W}^{(4,0)} \end{pmatrix},$$

where

$$\begin{aligned} \begin{pmatrix} \widehat{W}_{i,j}^{(1,0)} & \widehat{W}_{i,j}^{(2,0)} \\ \widehat{W}_{i,j}^{(3,0)} & \widehat{W}_{i,j}^{(4,0)} \end{pmatrix} &= c_{ij} Q_{\xi_i}^T Q_{\xi_j} \\ &\equiv \begin{pmatrix} W_{i,j}^{(1,0)} & W_{i,j}^{(2,0)} \\ W_{i,j}^{(3,0)} & W_{i,j}^{(4,0)} \end{pmatrix}, \quad \text{for } i,j = 1,\ldots,t \text{ and } i \le j. \end{aligned} \tag{5.15}$$

Given the RQDs (5.10) for $i = 1,\ldots,t-1$, it follows that

$$\widehat{W}_{t-1} \equiv \begin{pmatrix} \widehat{W}_{i,j}^{(1,t-1)} & \widehat{W}_{i,j}^{(2,t-1)} \\ \widehat{W}_{i,j}^{(3,t-1)} & \widehat{W}_{i,j}^{(4,t-1)} \end{pmatrix} = \begin{cases} \begin{pmatrix} W_{i,j}^{(1,t-1)} & W_{i,j}^{(2,t-1)} \\ 0 & W_{i,j}^{(4,t-1)} \end{pmatrix} & \text{if} \quad 1 \le i \le j \le t, \\ \begin{pmatrix} \widehat{W}_{i,j}^{(1,0)} & \widehat{W}_{i,j}^{(2,0)} \\ \widehat{W}_{i,j}^{(3,0)} & \widehat{W}_{i,j}^{(4,0)} \end{pmatrix} & \text{otherwise.} \end{cases} \tag{5.16}$$

Thus, for obtaining the RQD of $Q_\xi^T(C \otimes I_M)$ only the RQDs (5.10) for $i = t,\ldots,G-1$ need to be computed.

Consider now the case where $\xi_i = i$ for $i = G-t+1,\ldots,G$ and $1 \le t \le G-2$. The matrix $Q_\xi^T(C \otimes I_M)Q_\xi$ has the structure

$$\widehat{W}_0 = \begin{pmatrix} \widehat{W}^{(1,0)} & \widehat{W}^{(2,0)} \\ \widehat{W}^{(3,0)} & \widehat{W}^{(4,0)} \end{pmatrix},$$

where

$$\begin{aligned} \begin{pmatrix} \widehat{W}_{i,j}^{(1,0)} & \widehat{W}_{i,j}^{(2,0)} \\ \widehat{W}_{i,j}^{(3,0)} & \widehat{W}_{i,j}^{(4,0)} \end{pmatrix} &= c_{ij} Q_{\xi_i}^T Q_{\xi_j} \\ &\equiv \begin{pmatrix} W_{i,j}^{(1,0)} & W_{i,j}^{(2,0)} \\ W_{i,j}^{(3,0)} & W_{i,j}^{(4,0)} \end{pmatrix}, \quad \text{for } i,j = G-t+1,\ldots,G \text{ and } i \le j. \end{aligned} \tag{5.17}$$

The computation (5.11) can be utilized to derive

$$\widehat{W}_{i,j}^{(t-1)} = \begin{cases} \begin{pmatrix} \widehat{W}_{i,j}^{(1,0)} & \widehat{W}_{i,j}^{(2,0)} \\ \widehat{W}_{i,j}^{(3,0)} & \widehat{W}_{i,j}^{(4,0)} \end{pmatrix} & \text{if} \quad 1 \le i \le G \text{ and } 1 \le j \le G-t, \\ \begin{pmatrix} W_{i,j}^{(1,t-1)} & W_{i,j}^{(2,t-1)} \\ 0 & W_{i,j}^{(4,t-1)} \end{pmatrix} & \text{if} \quad G-t+1 \le i \le j \le G, \\ \begin{pmatrix} \widetilde{W}_{i,j}^{(1,t-1)} & \widetilde{W}_{i,j}^{(2,t-1)} \\ \widetilde{W}_{i,j}^{(3,t-1)} & \widetilde{W}_{i,j}^{(4,t-1)} \end{pmatrix} & \text{if} \quad 1 \le i \le G-t \text{ and } G-t+1 \le j \le G. \end{cases} \tag{5.18}$$

Here, the blocks $\widetilde{W}_{i,j}^{(l,t-1)}(l = 1,2,3,4)$ are obtained in $t-1$ steps by setting $\widetilde{W}_{i,j}^{(l,0)} = \widehat{W}_{i,j}^{(l,0)}$ and performing the modifications

$$\left(\begin{array}{ccc|c} \widetilde{W}_{1,G-i+1}^{(r,i-1)} & \cdots & \widetilde{W}_{1,G}^{(r,i-1)} & \widetilde{W}_{1,G-i}^{(r+1,i-1)} \\ \vdots & \ddots & \vdots & \vdots \\ \widetilde{W}_{G-i-1,G-i+1}^{(r,i-1)} & \cdots & \widetilde{W}_{G-i-1,G}^{(r,i-1)} & \widetilde{W}_{G-i-1,G-i}^{(r+1,i-1)} \end{array}\right) \tilde{P}_i = \left(\begin{array}{ccc|c} \widetilde{W}_{1,G-i+1}^{(r,i)} & \cdots & \widetilde{W}_{1,G}^{(r,i)} & \widetilde{W}_{1,G-i}^{(r+1,i)} \\ \vdots & \ddots & \vdots & \vdots \\ \widetilde{W}_{G-i-1,G-i+1}^{(r,i)} & \cdots & \widetilde{W}_{G-i-1,G}^{(r,i)} & \widetilde{W} x_{G-i-1,G-i}^{(r+1,i)} \end{array}\right), \tag{5.19}$$

where $r = 1,3$ and $i = 1,\ldots,t-1$. After $\widehat{W}_{t-1}$ in (5.18) has been constructed the RQD of $Q_\xi^T(C \otimes I_M)$ is obtained by computing (5.11) for $i = t,\ldots,G-1$.

5.3 Generating all the permuted ZR–VAR models

There are $G!$ possible ways of permuting the matrices $X_1,\ldots,X_G$ in the ZR–VAR model (5.2). The estimation of the resulting models derives by enumerating the $G!$ permutations $\xi = (\xi_1,\ldots,\xi_G)$

and solving the corresponding GLLSP (5.13). For simplicity let denote the model in (5.12) corresponding to a permutation ξ as the model ξ. Recall that for the solution of (5.13) only the RQD of $Q_{\xi}^{T}(C \otimes I_M)$ is required since the QRD of $\oplus_i X_{\xi_i}$ is available.

Algorithms which generate the sequence of all permutations have been extensively discussed [59, 104]. A convenient method to generate all permutations is to derive one permutation from another by applying an adjacent transposition [104, pp. 168–171]. This approach yields an efficient algorithm for generating all the permuted ZR–VAR models. The order of permutations is illustrated in Figure 5.3 for the case $G = 4$. The adjacent transposed indices are shadowed.

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

Figure 5.3: The sequence of permutations generated by applying adjacent transpositions, where $G = 4$.

Let $\tau^{(i)}$ $(i = 1,\ldots,(G-1)!)$ denote the permutations of the indices $(1,2,\ldots,G-1)$ such that they derive from one to another by applying an adjacent transposition. The superscript indicates the generation order. The permutations ξ of order G are obtained by inserting the new element G in all possible positions of each τ. That is, $\xi^{((j-1)G+1)} = (\tau_1^{(j)},\ldots,\tau_{G-1}^{(j)},G)$, $\xi^{((j-1)G+2)} = (\tau_1^{(j)},\ldots,G,\tau_{G-1}^{(j)})$, $\ldots$, $\xi^{(jG)} = (G,\tau_1^{(j)},\ldots,\tau_{G-1}^{(j)})$. In order to keep the property of adjacent transpositions in $\tau^{(j+1)}$ the insertion is done in reverse order, i.e. $\xi^{(jG+1)} = (G,\tau_1^{(j+1)},\ldots,\tau_{G-1}^{(j+1)})$, $\xi^{(jG+2)} = (\tau_1^{(j+1)},G,\ldots,\tau_{G-1}^{(j+1)})$, $\ldots$, $\xi^{((j+1)G)} = (\tau_1^{(j+1)},\ldots,\tau_{G-1}^{(j+1)},G)$, for $j = 1,\ldots,(G-1)!$ and $(j \bmod 2) = 1$. This method is summarized by Algorithm 7 which, mainly, consists of the recursive procedure, "Permute". The new permutations ξ are stored in a $G! \times G$ matrix Υ (line 8 of Algorithm 7).

The sequence of $G!$ permutations is divided in $(G-1)!/2$ sub-sequences of length $2G$. They have the form $\xi^{(jG-G+1)},\xi^{(jG-G+2)},\ldots,\xi^{(jG)},\xi^{(jG+1)},\xi^{(jG+2)},\ldots,\xi^{(jG+G)}$ where $j = 1,\ldots,(G-1)!$ and $(j \bmod 2) = 1$. For the estimation of the model $\xi^{(jG-G+1)}$ the RQD of W_0 in (5.9) is computed by using (5.10) for $i = 1,\ldots,G-1$. The next $G-2$ models, $\xi^{(jG-G+2)},\ldots,\xi^{(jG-1)}$ are derived by applying the transpositions $(t+1,t+2)$, where $t = G-2,\ldots,1$. It follows that $\xi_i^{(jG-t)} = \xi_i^{(jG-G+1)}$

Algorithm 7 The Generate procedure which constructs the $G!$ permutations of $(1,2,\ldots,G)$ by adjacent transpositions.

procedure Generate(G)
  $j \leftarrow 0$
  $\xi_i \leftarrow i$; $\theta_i \leftarrow i$; $\delta_i \leftarrow -1$; where $i = 1,\ldots,G$
  Permute(1)
end procedure

procedure Permute(i)
  if $(i > G)$ **then**
    $j \leftarrow j+1$; $\Upsilon_{j,:} \leftarrow \xi^T$
  else
    Permute($i+1$)
    for $l = 1,\ldots,i-1$ **do**
      Move(i, δ_i)
      Permute($i+1$)
    end for
    $\delta_i \leftarrow -\delta_i$
  end if
end procedure

procedure Move(i,d)
  aux $\leftarrow \xi_{\theta_i+d}$; $\xi_{\theta_i} \leftarrow$ aux; $\xi_{\theta_i+d} \leftarrow i$;
  $\theta_{\text{aux}} \leftarrow \xi_i$; $\xi_i \leftarrow \xi_i + d$
end procedure

for $i = 1,\ldots,t$. Thus, for the estimation of the model $\xi^{(jG-t)}$ the construction of $\widehat{W}_{t-1}$ in (5.16) is followed by the computation (5.10) for $i = t,\ldots,G-1$ and $t = G-2,\ldots,1$. Finally, for the estimation of the model $\xi^{(jG)}$ there is no computation which can be utilized and, thus, the RQD of W_0 is computed afresh.

Similarly, for the estimation of the model $\xi^{(jG+1)}$ the RQD of W_0 in (5.9) is computed using (5.11a) and (5.11b) for $i = 1,\ldots,G-1$. The models $\xi^{(jG+2)},\ldots,\xi^{(jG+G-1)}$ derive by applying the transpositions $(G-t-1,G-t)$, where $t = G-2,\ldots,1$. Thus, $\xi_i^{(jG+G-t)} = \xi_i^{(jG+1)}$ for $i = G-t+1,\ldots,G$. Furthermore, for the estimation of the model $\xi^{(jG+G-t)}$ the computation of $\widehat{W}_{t-1}$

in (5.18) and that of (5.11) are required for $i = t, \ldots, G-1$ and $t = G-2, \ldots, 1$.

This procedure is summarized by Algorithm 8. Notice that this algorithm has the matrix Υ as input. This may results into storage problems. However, the Algorithms 7 and 8 can be combined so that this problem is avoided. The corresponding permuted model is estimated once a new permutation ξ is derived. The place of the permutation ξ in a $2G$–length sequence is given by $(j \bmod 2G) + 1$, where $j = 1, \ldots, G!$.

Algorithm 8 Generating all $G!$ permuted ZR–VAR models.

Compute the QRDs (5.7)
$j \leftarrow 0; \quad d \leftarrow -1$
while $(j < G!)$ **do**
  $j \leftarrow j+1; \quad \xi \leftarrow \Upsilon_{j,:}$
  Construct $Q_\xi \equiv (\oplus_i Q_{A\xi_i} \quad \oplus_i Q_{B\xi_i})$
  Compute $Q_\xi^T (C \otimes I_M) Q_\xi$
  if $(d < 0)$ **then**
    Compute the RQDs (5.10), $\quad i = 1, \ldots, G-1$
    for $t = G-2, \ldots, 0$ **do**
      $j \leftarrow j+1; \quad \xi \leftarrow \Upsilon_{j,:}$
      Construct $\widehat{W}^{(t-1)}$ in (5.16)
      Compute the RQDs (5.10), $\quad i = t, \ldots, G-1$
    end for
  else
    Compute the RQDs (5.11a) and perform (5.11b), $\quad i = 1, \ldots, G-1$
    for $t = 0, \ldots, G-2$ **do**
      $j \leftarrow j+1; \quad \xi \leftarrow \Upsilon_{j,:}$
      Construct $\widehat{W}^{(t-1)}$ in (5.18)
      Compute the RQDs (5.11a) and perform (5.11b), $\quad i = t, \ldots, G-1$
    end for
  end if
  $d \leftarrow -d$
end while

5.4 Complexity considerations

For the estimation of the $G!$ permuted ZR–VAR models the most demanding computation is the GQRD (5.4). The QRD (5.4a) is available throughout the generating process. Thus, for the complexity analysis only the RQD (5.4b) will be taken into account. Furthermore, for the derivation of the theoretical measures of complexity it will be assumed that $k_1 = k_2 = \cdots = k_G \equiv M/2$. All quantities denote floating point operations (flops) [45].

The complexity of computing the RQD of W_0, is given by

$$\mathrm{C}_W = 4G^3M^3/3.$$

In section 5.2 two strategies for re-triangularizing W_0 in (5.9) have been presented. Each of these strategies consist of $G-1$ steps which corresponds to the computations (5.10) and (5.11). The number of flops required by these computations are given, respectively, by

$$\mathrm{St}_1(i) = iM^2\big(9i(3M+2)+19M+6\big)/24$$

and

$$\mathrm{St}_2(i) = iM^2\big(6G(3M+2)-3i(2M+3)-4M-6\big)/12,$$

where $i = 1,\ldots,G-1$. Thus, the complexities of computing the RQD of W_0 using the column-wise and row-wise block-annihilation strategies are given, respectively, by

$$\begin{aligned}\mathrm{C}_1 &= \sum_{i=1}^{G-1}\mathrm{St}_1(i) = (G-1)GM^2(3G(3M+2)+5M)/24\\ &\approx 9G^3M^3/24\end{aligned}$$

and

$$\begin{aligned}\mathrm{C}_2 &= \sum_{i=1}^{G-1}\mathrm{St}_2(i) = (G-1)GM^2(4G(3M+2)-M-4)/24\\ &\approx G^3M^3/2.\end{aligned}$$

As shown in section 5.2, the estimation of the sub-sequences of G models $\xi^{(jG-G+1)},\ldots,\xi^{(jG)}$ ($j = 1,\ldots,(G-1)!$ and $(j \bmod 2) = 1$) can be obtained by employing (5.10) and utilizing previous

	Computing afresh		Using (5.10)	Using (5.11)	Previous computations	
	only (5.4b)	(5.4)	$G\,\mathrm{C}_1$	$G\,\mathrm{C}_2$	Cp_1	Cp_2
Complexity	$\frac{4}{3}G^4M^3$	$\frac{7}{4}G^4M^3$	$\frac{3}{8}G^4M^3$	$\frac{1}{2}G^4M^3$	$\frac{9}{32}G^4M^3$	$\frac{21}{46}G^4M^3$
Ratio vs. (5.4b)	–	–	0.28	0.38	0.21	0.34
Ratio vs. (5.4)	–	–	0.21	0.29	0.16	0.26

Table 5.1: The complexities of various strategies for estimating a sub-sequence of G models.

computations. This requires

$$\begin{aligned}\mathrm{Cp}_1(G) &= 2\mathrm{C}_1 + \sum_{t=1}^{G-2}\sum_{i=t}^{G-1}\mathrm{St}_1(i) = (G-1)M^2\big(27G^3(3M+2)+G^2(211M+144)\\ &\qquad -2G(121M+114)+48(2M+3)\big)/288\\ &\approx 9G^4M^3/32.\end{aligned}$$

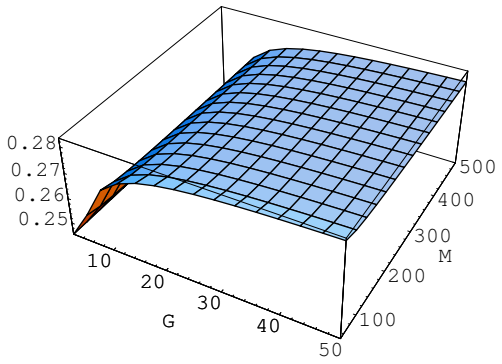
For the estimation of the remaining G models, i.e. $\xi^{(jG+1)},\ldots,\xi^{(jG+G)}$, the computations (5.11) are employed in order to utilize previous computation. However the extra computation (5.19) is also required. This takes

$$\mathrm{Ext}(i) = iM^2(G-i-1)(3M+2)/2$$

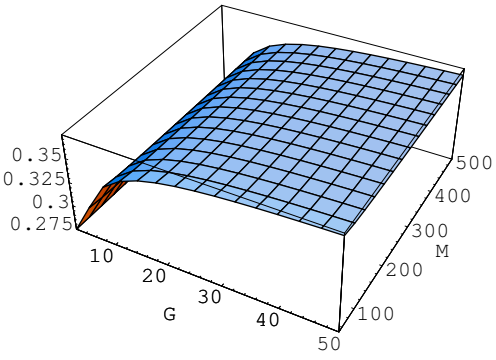
flops and thus, the complexity of the estimation procedure of the G models in this case is given by

$$\begin{aligned}\mathrm{Cp}_2(G) &= 2\mathrm{C}_2 + \sum_{t=1}^{G-2}\big(\mathrm{Ext}(i)+\sum_{i=t}^{G-1}\mathrm{St}_1(i)\big)\\ &= (G-1)M^2\big(25G^3(3M+2)+G^2(29M+G)-4G(M+9)-60M\big)/144\\ &\approx 21G^4M^3/46.\end{aligned}$$

The orders of complexities of the various methods for computing the RQD of W_0 in (5.9) are shown in Table 5.1. Figure 5.4 plots the ratios between the complexities of computing the RQD of W_0 using the column- and row-wise strategies and the one of computing the RQD of W_0 afresh for $G = 1,\ldots,50$ and $M = 50,\ldots,500$. Figure 5.5 shows the ratios between the complexities of generating the two sub-sequences of G models each when employing the strategies which utilize previous computation and the one which computes the RQDs afresh for $G = 1,\ldots,50$ and $M = 50,\ldots,500$. Notice that these plots confirm the theoretical complexities.

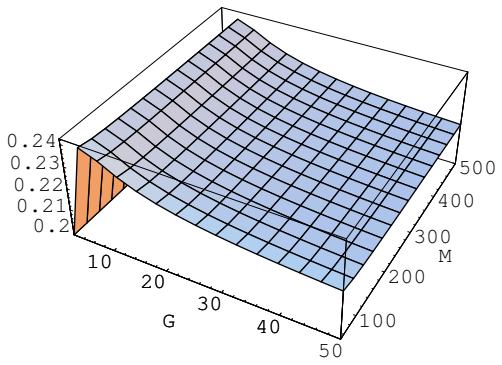


(a) C_1/C_W

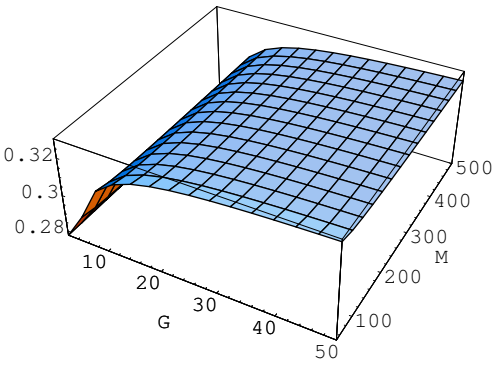


(b) C_2/C_W

Figure 5.4: Efficiency of the column- and row-wise strategies for computing the RQD of (5.9).



(a) $Cp_1(G)/(G\,C_W)$



(b) $Cp_2(G)/(G\,C_W)$

Figure 5.5: Efficiency obtained by utilizing previous computation when generating G models.

Summarizing, the complexity of generating the $G!$ ZR–VAR permuted models, i.e. computing the RQD (5.4b) by using Algorithm 8, is given by

$$C_{\text{Tot}} = (G-1)!(Cp_1(G) + Cp_2(G))/2.$$

Thus, the efficiency obtained by employing the Algorithm 8 rather than estimating the models

afresh is given by the ratio

$$\mathrm{C}_{\mathrm{Tot}}/(G!\,\mathrm{C}_W) \approx 0.28.$$

Now, taking into account the QRD of $\oplus_i X_i$ in the complexity of estimating the models afresh gives an improvement of 80%. Notice that this improvement in speed is for deriving only $2G$ models. The gain becomes more significant when deriving all the $G!$ models, i.e. when repeated $(G-1)!/2$ times.

5.5 Conclusions

A computational efficient algorithm for estimating the SUR models with permuted exogenous data matrices has been proposed. The main computation required for the estimation of the models is the generalized QR decomposition (5.4). The particular properties of the models are investigated and used to derive efficiently the estimation of the models. Specifically, the QR factorization (5.4a) of the exogenous data matrices is performed only once, for the original model, and it is available for the estimation of the remaining $G!-1$ models. Furthermore, two efficient strategies for computing the RQD (5.4b) have been presented. These strategies are adapted to the special cases of the permuted models so that the previous computation is re-used. These special cases arise when the $G!$ models are derived such that the permutations of the block matrices derive one from another by applying an adjacent transposition. The ratio between the complexity - in term of flops - of the proposed strategy and the complexity of deriving the estimators afresh is approximatively $1/4$. This ratio is obtained without taking into consideration that the RQD (5.4a) is available throughout the generating process. Otherwise, the efficiency of the proposed algorithm would have been even greater. The implementation and the application of the proposed algorithm need to be pursued.

The problem of estimating the SUR models with permuted exogenous data matrices arise in subset Vector Autoregressive (VAR) modeling [17, 41, 86, 90, 121]. The estimation of all subset VAR models is infeasible even for modest dimensions of the original model. Thus, alternative approaches should be envisaged. The first is the use of heuristics and genetic algorithms [17, 120]. A second alternative is the search in a sub-class of models. Computationally efficient methods which generates subset VAR models by deleting the same variables, or proper subsets of variables, from each block of equations in (5.1) has been previously proposed. A third special case is the permuted ZR–VAR models.

In all these three case, the computational burden of deriving the generalized QR decomposition (5.4) and, thus, estimating the models, is significantly reduced. In order to make the applicability

of these methods feasible a common framework (i.e. tree-structure) should be designed. This will allow a full investigation of the subset VAR models. The development of such framework is currently under investigation.

Chapter 6

Conclusions and future research

An efficient parallel algorithm has been developed to derive all subset models of the standard regression model. The algorithm (PDCA) is a parallelization of the dropping columns algorithm (DCA) proposed in [111, 112]. The properties of the regression tree generated by the DCA have been studied in order to derive an efficient load-balancing strategy. The PDCA uses a single–program multiple–data paradigm and does not require inter-processor communication. Experimental results have confirmed the linear speed-up suggested by theoretical measures of complexity. The PDCA has the restriction that the number of processors is less than 2^{n-1}, which is even the case for common small-size problems ($n > 10$). As expected, the efficiency of the PDCA will decrease when the number of processors increases. This is due to the overheads and redundant computations during the initial stage of the algorithm, i.e. the mapping phase. The extension of the serial DCA and PDCA to deal with the General Linear (GLM) and Seemingly Unrelated Regression (SUR) models has been also discussed.

The PDCA proposed in Chapter 2 will be inefficient for heterogeneous parallel systems, e.g. clusters of workstations. In such platforms a dynamic distribution, such as that obtained by task-farming, can yield a better load balancing and thus, more efficient algorithms [63]. These non-trivial dynamic distribution strategies need to be investigated. One approach is to consider the $\rho^{2^{n-1}}$ possibilities of assigning the 2^{n-1} nodes of the regression tree in Figure 2.2 to the ρ processors. In this case each edge has a weight which corresponds to the cost of deriving a child node from its parent and the interprocessor communication cost. A genetic algorithm can be employed to find the tree which generates all nodes with minimum computational and communication costs.

For a large number of variables the consideration of all models will be computationally infeasible. In Chapter 3 a branch-and-bound algorithm (BBA) has been developed to compute the best

regression equation corresponding to each number of variables. The algorithm avoids the computation of the whole regression tree that generates all subset models and outperforms an existing leaps-and-bounds method by cubic order of complexity. The main computational tool is the QR decomposition and the residual sum of squares (RSS) is used for selecting the models. Any other criterion which is a monotone function of the RSS for subsets with the same size can be used. That is, if $\mathrm{RSS}(V_1) \geq \mathrm{RSS}(V_2)$, then $f(\mathrm{RSS}(V_1)) \geq f(\mathrm{RSS}(V_2))$, where V_1 and V_2 are sets of independent variables, $|V_1| = |V_2|$, $\mathrm{RSS}(V_i)$ is the RSS when fitting the model comprising the variables in V_i $(i = 1,2)$ and f is a statistic like R^2 or C_p.

The BBA described in Chapter 3 can be modified to compute the best subset regressions when restricting the size of the selected models within a given range. In this case the regression tree is different than the one investigated in [40, 42]. Computational strategies which exploit the characteristics of the new derived tree are currently under investigation. Furthermore, the regression tree generated by the BBA does not exhibit the characteristics of the one generated by the PDCA [42]. Thus, for the parallelization of the BBA a new strategy needs to be developed.

A heuristic approach which significantly improves the computational performance of the BBA has been presented in Chapter 3. The heuristic algorithm uses a tolerance parameter when deciding to cut a subtree. The value of the tolerance is given priori. The performance of the heuristic strategy on data sets has been investigated for various tolerance values. The heuristic BBA might not provide the optimal solution. However, it was formally shown that the relative error of the computed solution is smaller than the tolerance parameter used. The heuristic BBA can be further developed: an adaptive algorithm should be designed which calculates dynamically the value of the tolerance within the execution process of the BBA. Also an extensive study of these methods to various models and their comparison with other selection methods (e.g. Forward selection, Backward elimination, Stepwise regression) should be investigated.

In Chapter 4, the DCA is adapted to derive all the subset Vector Autoregressive (VAR) models by efficiently deleting one variable at a time from the model (4.4). A subset is formulated as a VAR model with zero-restrictions on the coefficients (ZR–VAR). However, the method of investigating all subsets becomes rapidly infeasible when the dimensions of the generating process (4.1), i.e. G and p, increase. Thus, alternative approaches should be considered. Within this context, several special cases of subset VAR models which are derived by taking advantage of the common columns of the data matrices and the Kronecker structure of the variance-covariance matrix have been proposed in Chapters 4 and 5.

The first special case derives $(pG+2)2^{(pG-1)} - 1$ subset VAR models by deleting the same

variable from each block of the model in (4.4) which corresponds to deletion of whole columns from some of the coefficient matrices $\Phi_1,\ldots,\Phi_p$ in (4.1). The second case is based on deleting proper subsets of variables from each block of the regressors in the ZR–VAR model (4.9). An algorithm that derives all proper subsets models given the initial VAR model has been designed. This algorithm generates $\sum_{i=1}^{\min(K,G-1)} C_{G-2}^{i-1} C_K^i 2^{K-i}$ models, where $C_i^j = i!/(j!(i-j)!)$ and $K = Gp$.

Furthermore, given the estimation of the ZR–VAR model in (5.2), there are $G!$ ways of permuting the matrices $X_1,\ldots,X_G$ and thus, subset VAR models. Their estimation derive by enumerating the $G!$ permutations $\xi = (\xi_1,\ldots,\xi_G)$ and solving the corresponding Generalized Linear Least Squares Problem (5.13). An efficient algorithm for deriving all permuted ZR–VAR models has been presented in Chapter 5. This algorithm generates the permutations by applying adjacent transpositions and re-utilize previous computation.

In all cases the computational burden of deriving the generalized QR decomposition (4.11), and thus, estimating the sub-models, is significantly reduced. The first two cases require $O(G)$ less computational complexity than generating the models afresh. The third strategy requires 70% less computation for the QR decomposition (4.11b), while the QR decomposition (4.11a) is available through the generating process.

A common framework should be defined that can be used to provide a full investigation of all subset VAR models rather than a specific class of sub-models investigated by the proposed three strategies. The incorporation and functioning of these strategies in a common framework of investigating all subsets will require the modification, adaptation and derivation of new computational strategies.

The tree-structure generated by DCA suggests that a branch-and-bound strategy should be designed. Thus, the best-subset VAR models can be derived without computing the whole regression tree derived by DCA [40, 49, 80, 83]. The cutting test, in this case, should be based on statistical criteria such as the Information Complexity criterion (ICOMP) [17, 18]. Heuristics and Genetic Algorithms can be employed to facilitate the use of the Branch-and-Bound method to the investigation of VAR models which are computationally intensive [17, 121].

For even modest-size VAR models their complete investigation might be computationally infeasible, even if a branch-and-bound strategy is employed. For this, heuristics and Genetic Algorithms should be considered in order to facilitate the investigation of these models [17, 121]. Within this context the use of high performance computing (including parallel computing) needs to be pursued. Specifically the parallel algorithm (PDCA) for computing all subset regression models in [42] will be inefficient for conventional clusters of workstations. In such cases alternative parallel strategies

need to be designed [63].

Bibliography

[1] P. Zinterhof A. Uhl. Special issue on Parallel Computing in Image and Video Processing. *Parallel Computing*, 28(7-8):941–1219, 2002.

[2] H. Akaike. Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mathematics*, 21:243–247, 1969.

[3] H. Akaike. A new look at statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[4] D. M. Allen. Mean square error of prediction as a criterion for selectiong variables. *Technometrics*, 13(3):469–475, 1971.

[5] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK users' guide*. SIAM, Philadelphia, 1992.

[6] E. Anderson, Z. Bai, and J. J. Dongarra. Generalized QR factorization and its applications. *Linear Algebra and its Applications*, 162:243–271, 1992.

[7] P. Arbenz, E. Gallopoulos, B. Phillippe, and Y. Saad. Special issue on Parallel Matrix Algorithms and Applications (PMAA'02). *Parallel Computing*, 29(9):1117–1273, 2003.

[8] J. F. Ball, R. E. Dorsey, and J. D. Johnson. Non-linear optimization on a parallel intel i860 risc based architecture. *Computational Economics*, 10:279–294, 1997.

[9] P. M. Beaumont and P. T. Bradshaw. A distributed parallel genetic algorithm for solving optimal growth models. *Computational Economics*, 8:159–179, 1995.

[10] D. A. Belsley, E. Kuh, and R. E. Welsch. *Regression Diagnostics: Identifying Influential Observations and Sources of Collinearity*. John Wiley and Sons, 1980.

[11] Å. Björck. A general updating algorithm for constrained linear least squares problems. *Siam Journal on Scientific and Statistical Computing*, 5(2):394–402, 1984.

[12] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.

[13] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaley. *ScaLAPACK Users' Guide*. SIAM, Philadelphia, 1997.

[14] G. E. P. Box. Use and abuse of regression. *Technometrics*, 8(4):625–629, 1966.

[15] H. Bozdogan. ICOMP: A new model-selection criterion. In H. H. Bock, editor, *Classification and related methods of data analysis*. North-Holland Elesvier Science, 1987.

[16] H. Bozdogan. Akaike's information criterion and recent developments in information complexity. *Journal of Mathematical Psychology*, 44:62–91, 2000. (Special Issue on Model Selection).

[17] H. Bozdogan and P. Bearse. Informational complexity criteria for detecting influential observations in dynamiy multivariate linear models using the genitic algorithm. *Journal of Statistical Planning and Inference*, 114:31–44, 2003.

[18] H. Bozdogan and D. M. A. Haughton. Informational complexity criteria for regression models. *Computational Statistics & Data Analysis*, 28:51–76, 1998.

[19] L. Breiman and J. H. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80(391):580–619, 1985.

[20] J. M. Bull, G. D. Riley, J. Rasbash, and H. Goldstein. Parallel implementation of a multilevel modelling package. *Computational Statistics & Data Analysis*, 31(4):457–474, 1999.

[21] P. Businger and G. H. Golub. Linear least squares solutions by Householder transformations. *Numerische Mathematik*, 7:269–276, 1965.

[22] G. Casella and E. Moreno. Objective bayesian variable selection. Technical Report 2002-023, Department of Statistics, University of Florida, USA, 2002.

[23] J. C. Chao and P. C. B. Phillips. Model selection in partially nonstationary vector autoregressive processes with reduced rank structure. *Journal of Econometrics*, 91(2):227–271, 1999.

[24] M. R. B. Clarke. Algorithm AS163. A Givens algorithm for moving from one linear model to another without going back to the data. *Applied Statistics*, 30(2):198–203, 1981.

[25] A. Clemetis, M. Mineter, and R. Marciano. Special issue on High Performance Computing with Geographical Data. *Parallel Computing*, 29(10):1275–1504, 2003.

[26] N. R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley & Sons, Inc., 1966.

[27] David Edwards and Tomáš Havránek. A fast model selection procedure for large families of models. *Journal of the American Statistical Association*, 82(397):205–213, 1987.

[28] D. W. Fausett, C. T. Fulton, and H. Hashish. Improved parallel QR method for large least squares problems involving Kronecker products. *Journal of Computational and Applied Mathematics*, 78:63–78, 1997.

[29] J. Faust and R. Tryon. A distributed block approach to solving near-block-diagonal systems with an application to a large macroeconomic model. *Computational Economics*, 8:303–316, 1995.

[30] M. Florian and M. Grendreau. Special issue on Applications of Parallel Computing in Transportation. *Parallel Computing*, 27(12):1521–1653, 2001.

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

[32] P. Foschi and E. J. Kontoghiorghes. Estimation of seemingly unrelated regression models with unequal size of observations: computational aspects. *Computational Statistics and Data Analysis*, 41(1):211–229, 2002.

[33] P. Foschi and E. J. Kontoghiorghes. Estimation of VAR models: computational aspects. *Computational Economics*, 21(1-2):3–22, 2003.

[34] I. Foster. *Designing and Building Parallel Programs*. Addison-Wesley, 1995.

[35] G. M. Furnival. All possible regressions with less computation. *Technometrics*, 13(2):403–408, 1971.

[36] G. M. Furnival and Jr. R. W. Wilson. Regression by leaps and bounds. *Technometrics*, 16(4):499–511, 1974. Replublished in Technometrics, 42 (2000), pp. 69–79.

[37] M. J. Garside. The best sub-set in multiple regression analysis. *Applied Statistics*, 14(2/3):196–200, 1965.

[38] M. J. Garside. Some computational procedures for the best subset problem. *Applied Statistics*, 20:8–15, 1971.

[39] C. Gatu and E. J. Kontoghiorghes. A branch and bound algorithm for computing the best subset regression models using the QR decomposition. Technical Report RT-2002/08-1, Institut d'informatique, Université de Neuchâtel, Switzerland, 2002.

[40] C. Gatu and E. J. Kontoghiorghes. Branch-and-bound algorithms for computing the best-subset regression models. *Journal of Computational and Graphical Statistics*, 2003. (Submitted).

[41] C. Gatu and E. J. Kontoghiorghes. Efficient strategies for deriving the subset VAR models. *Computational Management Science*, 2003. (Submitted).

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

[43] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28(126):505–535, 1974.

[44] R. F. Gnust and R. L. Mason. *Regression analysis and its applications: a data-oriented approach*, volume 34 of *Statistics: textbooks and monographs*. Marcel Dekker, New York, 1980.

[45] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, Maryland, 3ed edition, 1996.

[46] G. H. Golub and J. H. Wilkinson. Note on the iterative refinement of least squares solution. *Numerische Mathematik*, 9:139–148, 1966.

[47] J. H. Goodnight. A tutorial on the SWEEP operator. *The American Statistician*, 33(3):116–135, 1979.

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

[49] D. J. Hand. Branch and bound in statistical data analysis. *Statistician*, 30(1):1–13, 1981.

[50] E. J. Hannan and B. G. Quinn. The determination of the order of an autoregression. *Journal of the Royal Statistical Society, Series B*, 41(2):190–195, 1979.

[51] T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*, volume 43 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, 1990.

[52] I. S. Helland. On the interpretation an use of r^2 in regression analysis. *Biometrics*, 43:61–69, 1987.

[53] R. R. Hocking. Criteria for selection of a subset regression: which one should be used? *Technometrics*, 14(4):967–970, 1972.

[54] R. R. Hocking. The analysis and selection of variables in linear regression. *Biometrics*, 32:1–49, 1976.

[55] R. R. Hocking. Developments in linear regression methodology: 1959-1982. *Technometrics*, 25(3):219–230, 1983.

[56] R. R. Hocking and R. N. Leslie. Selection of the best subset in regression analysis. *Technometrics*, 9(4):531–540, 1967.

[57] I. Karasalo. A criterion for truncation of the QR decomposition algorithm for the singular linear least squares problem. *BIT*, 14:156–166, 1974.

[58] J. Kmenta and R. F. Gilbert. Small sample properties of alternative estimators of seemingly unrelated regressions. *Journal of the American Statistical Association*, 63:1180–1200, 1968.

[59] D. E. Knuth. *The art of computer programming, Sorting and Searching*, volume 3. Addison–Wesley, 1973.

[60] E. J. Kontoghiorghes. New parallel strategies for block updating the QR decomposition. *Parallel Algorithms and Applications*, 5(1+2):229–239, 1995.

[61] E. J. Kontoghiorghes. Parallel strategies for computing the orthogonal factorizations used in the estimation of econometric models. *Algorithmica*, 25:58–74, 1999.

[62] E. J. Kontoghiorghes. Special issue on Parallel processing and statistics. *Computational Statistics & Data Analysis*, 31(4):373–516, 1999.

[63] E. J. Kontoghiorghes. *Parallel Algorithms for Linear Models: Numerical Methods and Estimation Problems*, volume 15 of *Advances in Computational Economics*. Kluwer Academic Publishers, Boston, MA, 2000.

[64] E. J. Kontoghiorghes. Parallel Givens sequences for solving the general linear model on a EREW PRAM. *Parallel Algorithms and Applications*, 15(1-2):57–75, 2000.

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

[66] E. J. Kontoghiorghes. Parallel strategies for solving SURE models with variance inequalities and positivity of correlations constraints. *Computational Economics*, 15(1+2):89–106, 2000.

[67] E. J. Kontoghiorghes. Computational methods for modifying seemingly unrelated regressions models. *Journal of Computational and Applied Mathematics*, 162:247–261, 2004.

[68] E. J. Kontoghiorghes, editor. *Handbook of Parallel Computing and Statistics*. Statistics: Textbooks and Monograph Series. Marcel Dekker, Inc., 2004. (In press).

[69] E. J. Kontoghiorghes and M. R. B. Clarke. Parallel reorthogonalization of the QR decomposition after deleting columns. *Parallel Computing*, 19(6):703–707, 1993.

[70] E. J. Kontoghiorghes and M. R. B. Clarke. Parallel reorthogonalization of the QR decomposition after deleting columns. *Parallel Computing*, 19(6):703–707, 1993.

[71] E. J. Kontoghiorghes and M. R. B. Clarke. Solving the updated and downdated ordinary linear model on massively parallel SIMD systems. *Parallel Algorithms and Applications*, 1(2):243–252, 1993.

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

[73] E. J. Kontoghiorghes and M. R. B. Clarke. Solving the general linear model on a SIMD array processor. *Computers and Artificial Intelligence*, 14(4):353–370, 1995.

[74] E. J. Kontoghiorghes and E. Dinenis. Solving triangular seemingly unrelated regression equations models on massively parallel systems. In M. Gilli, editor, *Computational Economic Systems: Models, Methods & Econometrics*, volume 5 of *Advances in Computational Economics*, pages 191–201. Kluwer Academic Publishers, 1996.

[75] E. J. Kontoghiorghes, H.-H. Nägeli, and P. Pardalos. Special issue on Economics, Statistics, Finance and Optimization. *Parallel Algorithms and Applications*, 15(1-2):1–130, 2000.

[76] E. J. Kontoghiorghes, D. Trystram, and A. Sameh. Special issue on Parallel Matrix Algorithms and Applications. *Parallel Computing*, 28(2):151–368, 2002.

[77] S. Kourouklis and C. C. Paige. A constrained least squares approach to the general Gauss–Markov linear model. *Journal of the American Statistical Association*, 76(375):620–625, 1981.

[78] H. M. Krolzig and D. F. Hendry. Computer automation of general-to-specific model selection procedures. *Journal of Economic Dynamics & Control*, 25(6-7):831–866, 2001.

[79] L. R. LaMotte and R. R. Hocking. Computational efficiency in the selection of regression variables. *Technometrics*, 12(1):83–93, 1970.

[80] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[81] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice–Hall Englewood Cliffs, 1974.

[82] R. Leardi, R. Boggia, and M. Terrile. Genetic algorithms as a strategy for feature selection. *Journal of Chemometrics*, 6:267–281, 1992.

[83] E. K. Lee. Branch-and-bound methods. In P. M. Pardalos and M. G. C. Resende, editors, *Handbook of Applied Optimazion*, pages 53–65. Oxford University Press, 2002.

[84] H. K. H. Lee. A framework for nonparametric regression using neural networks. Technical Report 00-32, Institute of Statistics and Decision Sciences, Duke University, USA, 2000.

[85] J. Leszczynski. Special issue on Computational Chemistry. *Parallel Computing*, 26(7-8):817–1060, 2000.

[86] H. Lütkepohl. *Introduction to Multiple Time Series Analysis*. Springer–Verlag, second edition, 1993.

[87] C. L. Mallows. Some comments on C_P. *Technometrics*, 15:661–675, 1973.

[88] C. L. Mallows. More comments on C_P. *Technometrics*, 37(4):362–372, 1995.

[89] J.W. Manke. Special issue on Parallel Computing in Aerospace. *Parallel Computing*, 27(4):329–536, 2001.

[90] D. G. Maringer. Finding the relevant risk factors for asset pricing. *Computational Statistics & Data Analysis*, 2004. (In press).

[91] J. McClave. Subset autoregression. *Technometrics*, 17(2):213+, 1975.

[92] G. C. McDonald and R. C. Schwing. Instabilities of regression estimates relating air pollution to mortality. *Technometrics*, 15:463–482, 1973.

[93] E. J. Kontoghiorghes M.H. Gutknech and V. Simoncini. Special issue on numerical algorithms, parallelism and applications. *Applied Numerical Mathematics*, 49(1):1–333, 2004.

[94] A. J. Miller. Selection of subsets of regression variables. *J.R. Statist. Soc.*, 147:389–425, 1984.

[95] A. J. Miller. *Subset selection in regression*, volume 95 of *Monographs on statistics and applied probability*. Chapman and Hall, 2nd edition, 2002. (Related software can be found at URL: http://users.bigpond.net.au/amiller/).

[96] K. Murphy, M. Clint, and R.H. Perrott. Re-engineering statistical software for efficient parallel execution. *Computational Statistics & Data Analysis*, 31(4):441–456, 1999. (Special issue on parallel processing and statistics).

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

[98] C. C. Paige. Numerically stable computations for general univariate linear models. *Communications in Statistics*, B7:437–453, 1978.

[99] C. C. Paige. Fast numerically stable computations for generalized least squares problems. *SIAM J. on Numerical Analysis*, 16:165–171, 1979.

[100] C. C. Paige. Some aspects of generalized QR factorizations. In M. G. Cox and S. J. Hammarling, editors, *Reliable Numerical Computation*, pages 71–91. Clarendon Press, Oxford, UK, 1990.

[101] A. Pitard and J. F. Viel. A model selection tool in multi-pollutant time series: The Granger-causality diagnosis. *Environmetrics*, 10(1):53–65, 1999.

[102] C. R. Rao and H. Toutenburg. *Linear Models: Least Squares and Alternatives*. Springer series in Statistics. Springer, 1995.

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

[104] E. M. Reingold, J. Nievergelt, and N. Deo. *Combinatorial Algorithms: Theory and Practice*. Prentice–Hall, 1977.

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

[106] P. Schmidt. A note on the estimation of seemingly unrelated regression systems. *Journal of Econometrics*, 7:259–261, 1978.

[107] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[108] S. R. Searle. *Linear Models*. John Wiley and Sons, Inc., 1971.

[109] G. A. F. Seber. *Linear regression analysis*. John Wiley & Sons, 1977.

[110] A. Sen and M. Srivastava. *Regression Analysis. Theory, Methods, and Applications*, volume 38 of *Springer Texts in Statistics*. Springer-Verlag, New York Inc, 1990.

[111] D. M. Smith. *Regression using QR decomposition methods*. PhD Thesis, University of Kent, UK, 1991.

[112] D. M. Smith and J. M. Bremner. All possible subset regressions using the QR decomposition. *Computational Statistics and Data Analysis*, 7:217–235, 1989.

[113] V. K. Srivastava and T. D. Dwivedi. Estimation of seemingly unrelated regression equations Models: a brief survey. *Journal of Econometrics*, 10:15–32, 1979.

[114] V. K. Srivastava and D. E. A. Giles. *Seemingly Unrelated Regression Equations Models: Estimation and Inference (Statistics: Textbooks and Monographs)*, volume 80. Marcel Dekker, Inc., 1987.

[115] G. Strang. *Linear Algebra and Its Applications*. Academic Press, 1976.

[116] A. Sudjianto, G. S. Wasserman, and H. Sudarbo. Genetic subset regression. *Computers & Industrial Engineering*, 30(4):839–849, 1996.

[117] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.

[118] L. Wasserman. Bayesian model selection and model averaging. *Journal of Mathematical Psychology*, 44, 2000. (Special Issue on Model Selection).

[119] P. Winker. Identification of multivariate AR-models by threshold accepting. *Computational Statistics & Data Aanlysis*, 20(3):295–307, 1995.

[120] P. Winker. Optimized multivariate lag structure selection. *Computational Economics*, 16(1–2):87–103, 2000.

[121] P. Winker. *Optimization Heuristics in Econometrics: Applications of Threshold Accepting*. Wiley Series in Probability and Statistics. John Wiley &Sons, Ltd, 2001.

[122] P. Winker. *Optimization Heuristics in Econometrics: Applications of Threshold Accepting*, chapter 12. Wiley Series in Probability and Statistics. John Wiley &Sons, Ltd, 2001.

[123] P. Yanev, P. Foschi, and E. J. Kontoghiorghes. Algorithms for computing the QR decomposition of a set of matrices with common columns. *Algorithmica*, 39(1):83–93, 2004.

[124] P. Yanev and E. J. Kontoghiorghes. Efficient algorithms for block downdating of least squares solution. *Applied Numerical Mathematics*, 49(1):3–15, 2004.

[125] L T. Yang, Y. Pan, and M. Guo. Special issue on Parallel and Distributed Scientific and Engineering Computing. *Parallel Computing*, 29(11-12):1505–1817, 2003.

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

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