1991-09

A comparison of three numerical methods for updating regressions.

Raptis, Grigorios J.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A COMPARISON OF THREE NUMERICAL METHODS FOR UPDATING REGRESSIONS

by

Grigorios J. Raptis

September, 1991

Thesis Advisor: Dan C. Boger
Thesis Co-advisor: William B. Gragg

Approved for public release; distribution is unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL *(If applicable)* OR	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS Program Element No / Project No / Task No / Work Unit Accession Number

11. TITLE *(Include Security Classification)*
A COMPARISON OF THREE NUMERICAL METHODS FOR UPDATING REGRESSIONS

12. PERSONAL AUTHOR(S) Grigorios J. Raptis

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*year, month, day*)	15. PAGE COUNT
Master's Thesis	From To	1991, September	95

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	Hyperbolic Rotation, Gram-Schmidt Factorization, Reorthogonalization, Column Permutation, Updating

19. ABSTRACT (*continue on reverse if necessary and identify by block number*)

Three numerical procedures are presented for updating regressions. All three methods are based on QR factorization, but after that they use different philosophies to update the regression coefficients. Elden's algorithm updates using only the triangular matrix R. This procedure does not use orthogonal transformations, but it uses hyperbolic rotations. The modified Gram-Schmidt QR process is used by Gragg-Leveque-Trangenstein's method where the matrix with orthonormal columns is stored and updated. Chan's algorithm computes a column permutation Π and a QR factorization of a matrix A such that a rank deficiency of A will be revealed. Although the three methods are based on different ideas and can be used for different purposes their comparison shows that Chan's algorithm is the only accurate one in the rank deficient case, and that Gragg-Leveque-Trangenstein's method is the cheapest and the most stable.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS REPORT ☐ DTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Dan C. Boger	22b. TELEPHONE (*Include Area code*) (408) 646-2607	22c. OFFICE SYMBOL OR/Bo

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Approved for public release; distribution is unlimited.

A Comparison of Three Numerical Methods for Updating Regressions

by

Grigorios J. Raptis
Lieutenant Commander, Hellenic Navy
B.S., Technical University of Athens
B.S., Hellenic Naval Academy

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the
NAVAL POSTGRADUATE SCHOOL
September 1991

ABSTRACT

Three numerical procedures are presented for updating regressions. All three methods are based on **QR** factorization, but after that they use different philosophies to update the regression coefficients. Elden's algorithm updates using only the triangular matrix **R**. This procedure does not use orthogonal transformations, but it uses hyperbolic rotations. The modified Gram-Schmidt **QR** process is used by Gragg-Leveque-Trangenstein's method where the matrix with orthonormal columns is stored and updated. Chan's algorithm computes a column permutation Π and a **QR** factorization of a matrix **A** such that a rank deficiency of **A** will be revealed. Although the three methods are based on different ideas and can be used for different purposes their comparison shows that Chan's algorithm is the only accurate one in the rank deficient case, and that Gragg-Leveque-Trangenstein's method is the cheapest and the most stable.

TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

Regression analysis provides a variety of modeling techniques that allows the analyst to relate a dependent variable to one or more independent variables. The mathematical complexity of the model and the degree to which it is a realistic model will depend on how much is known about the process being studied and on the purpose of the modeling exercise. In preliminary studies of a process or in cases where prediction is the primary objective, the models will frequently fall in the class of models that are linear in the parameters. That is, the parameters enter the model as simple coefficients of the independent variables or functions of the independent variables. Such models will be referred to loosely as linear models. In many statistical problems, it is useful to express the dependent variable as a linear function of the independent variables. Furthermore, regression analysis can summarize data and quantify the nature and strength of the relationships among variables. Regression analysis can also be used to predict new values of the dependent variables based on observed relationships.

B. ANALYSIS OF REGRESSION

As the reader will see in the following chapter, the main purpose of this thesis is to analyze the efficiency and numerical stability of some procedures for updating regressions. Most regression problems, however, require decisions on which variables to include in the model, the form the variables should take, for example, X, X^2, 1/X, etc., and the functional form of the model. It is assumed that there is a set of t candidate variables, which presumably includes all relevant variables, from which a subset of r variables is to be chosen for the regression equation. The candidate variables may include different forms of the same basic variable, such as X and X^2, and the selection process may include constraints on which variables are to be included. For example, X may be forced into the model if X^2 is in the selected subset. This is a common constraint in building polynomial models.

There are three distinct problem areas related to this general topic:

- The theoretical effects of variable selection on the least squares regression results.
- The computational methods for finding the "best" subset of variables for each subset size.
- The choice of subset size (for the final model), or the "stopping rule".

This thesis mainly discusses the second problem area of finding the more efficient and numerically stable computational methods. Also, we generally discuss the criteria for the "best" subset size choice, in other words the criteria for the "stopping rules".

The simplest linear model involves only one independent variable and states that the true mean of the dependent variable changes at a constant rate as the value of the independent variable increases or decreases. Thus, the functional relationship between the true mean of Y_i, $E(Y_i)$, and X_i is the equation of a straight line,

$$E(Y_i) = \beta_0 + \beta_1 (X_i),$$

where β_0 is the intercept, or the value of $E(Y_i)$ when $X = 0$, and β_1 is the slope of the line, or the rate of change in $E(Y_i)$ per unit change in X. The constants β_0 and β_1 are population parameters which are to be estimated from the sample of observations.

The observations of the dependent variables, Y_i, are assumed to be random observations from populations of random variables with the mean of each population given by $E(Y_i)$. The deviation of an observation Y_i from its population mean $E(Y_i)$ is taken into account by adding a random error, ϵ_i, to give the statistical model

$$Y_i=\beta_0+\beta_1 X_i+\epsilon_i. \qquad (1.1)$$

The subscript i indicates the particular observation unit, i = 1,2,...,n. The X_i are the n observations of the independent variable and are assumed to be measured without error; that is, the observed values of X are assumed to be a set of known constants. The Y_i and X_i are paired observations; both are measured on every observational unit. The random errors ϵ_i are assumed to be normally, independently and identically distributed:[Ref. 1]

$$\epsilon_i \sim NID(0,\sigma^2).$$

1. The Matrix Model

Given the above regression model, we can write it in matrix form as $Y = X\beta + \epsilon$, where Y is the n x 1 column vector of observations of the dependent variable. The n x p matrix X, where $x_{10} = x_{20} = \ldots = x_{n0} = 1$, will be called the regression matrix, and the x_{ij}'s are generally chosen so that the column of X are linearly independent; that is X has rank p. However, in some experimental design situations the elements of X are chosen to be 0 or 1, and the columns of X may be linearly dependent. In this case X is commonly called the design matrix. Also, β is the p x 1 vector of parameters to be estimated; and ϵ is the n x 1 vector of random errors. Writing out the components of $Y = X\beta + \epsilon$ yields

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \cdot \\ \cdot \\ \cdot \\ Y_n \end{pmatrix} = \begin{pmatrix} x_{10} & x_{11} & x_{12} & \cdots & x_{1p} \\ x_{20} & x_{21} & x_{22} & \cdots & x_{2p} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ x_{n0} & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \cdot \\ \cdot \\ \cdot \\ \beta_p \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ \epsilon_n \end{pmatrix}. \qquad (1.2)$$

(nx1) (nxp') (p'x1) (nx1)

The elements of a particular row, r, of X are the coefficients of the corresponding parameters in β which give $E(Y_r)$. The vectors Y and ϵ are random vectors; the elements of these vectors are random variables. The matrix X is considered to be a matrix of known constants. The vector β is a vector of unknown constants and is to be estimated from the given data.

Using the above assumptions on ϵ_i, the random vector ϵ has a multivariate normal distribution with a mean of the zero (0) vector. The variance-covariance matrix for any random vector of n elements is defined as an n x n symmetric matrix with the diagonal elements equal to the variance of ϵ_i and the off-diagonal elements equal to the covariance between ϵ_i and ϵ_j. Let Z be a n x 1 vector of random variables z_1, z_2, z_3, ... ,z_n; the variance-covariance matrix of Z is the following n x n matrix.

$$Var(Z) = \begin{pmatrix} var(z_1) & cov(z_1, z_2) & \dots & cov(z_1, z_n) \\ cov(z_2, z_1) & var(z_2) & \dots & cov(z_2, z_n) \\ \dots & \dots & \dots & \dots \\ cov(z_n, z_1) & cov(z_n, z_2) & \dots & var(z_n) \end{pmatrix} \quad (1.3)$$

The variance-covariance matrix of ϵ is $I\sigma^2$, and since ϵ is independent and identically distributed, the distribution of ϵ is written in shorthand notation as

$$\epsilon \sim N(0, I\sigma^2),$$

where I is the n x n identity matrix and σ^2 is the common variance of all ϵ_i. Furthermore, since the elements of X and β are constants, the $X\beta$ term in the model is a set of constants being added to the vector of random errors, ϵ. Thus, Y is a random vector with mean vector $X\beta$ and variance-covariance matrix $I\sigma^2$:

$$E(Y) = E(X\beta + \varepsilon) = E(X\beta) + E(\varepsilon) = X\beta \quad (1.4)$$

$$Var(Y) = Var(X\beta + \varepsilon) = Var(\varepsilon) = I\sigma^2 \quad (1.5)$$

Var(Y) is the same as Var(ϵ) since adding a constant to a random variable does not change the variance. When ϵ is multivariate normally distributed, Y is also multivariate normally distributed. Thus,

$$Y \sim N(X\beta, I\sigma^2) \tag{1.6}$$

This result is based on the assumption that the linear model being used is the correct model [Ref. 2:.68].

2. The Normal Equations

Before considering the problem of estimating β, we note that we are referring to the model (2), where X_{i0} is not necessarily constrained to be unity. In the case when $X_{i0} \neq 1$ we have to use a notation in which i runs from 0 to p-1 rather than 1 to p. However, since the major application of the theory is to the case $X_{i0} \equiv 1$, it is convenient to separate β_0 from the other β_j's right from the outset.

The oldest method of obtaining an estimate of β is the method of least squares. It dates back at least to Gauss and consists of minimizing $\Sigma_i \epsilon_i^2$ with respect to β (see Fig. 1); that is, setting $q = X\beta$, we minimize $\epsilon^T \epsilon$[1] $= \|Y - q\|^2$ subject to $q \in \Re[X] = W$, where W is the range space of X $\{y: y = Xx$ for some $x\}$. If we let q vary in W, $\|Y - q\|^2$ (the square of the length of $Y - q$) will be a minimum for $q = q^*$ when $(Y - q^*)$ is orthogonal with W (see Fig. 1).

Thus

$$X^T(Y - q^*) = 0 \tag{1.7}$$

or

---

[1] We denote the transpose of a vector y by y^T, and similarly for a matrix transpose.

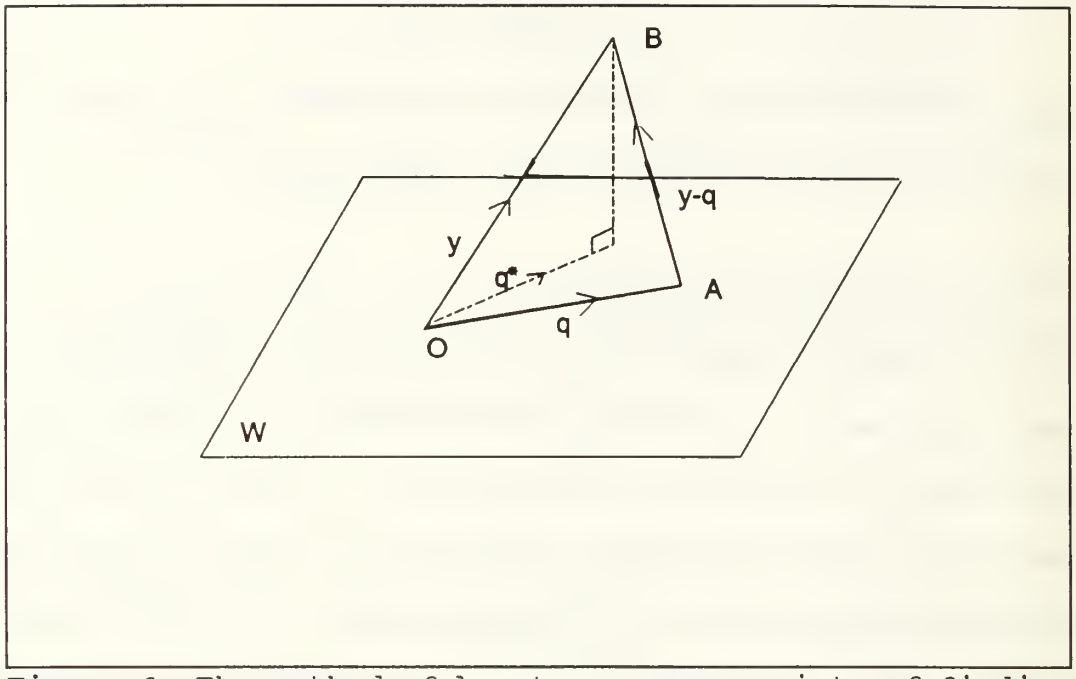

Figure 1. The method of least squares consists of finding A such that AB is a minimum.

$$X^T q^* = X^T Y. \tag{1.8}$$

Here q^* is uniquely determined, being the unique orthogonal projection of Y onto W. Now given that the columns of X are linearly independent, there exists a unique vector ($\hat{\beta}$) such that $q^* = X\hat{\beta}$. Therefore substituting in (1.8) we have

$$X^T X \hat{\beta} = X^T Y \tag{1.9}$$

the so-called normal equation(s). At this point we assume that X has rank p, so $X^T X$ is positive definite and therefore

nonsingular. If so, equation (1.9) has a unique solution, namely,

$$\hat{\beta} = (X^T X)^{-1} X^T Y \tag{1.10}$$

Here $\hat{\beta}$ is called the ordinary least squares estimate of β. Computational methods for calculating $\hat{\beta}$ are given in the following chapters. Notice that $\hat{\beta}$ can also be obtained by writing

$$\varepsilon^T\varepsilon=(Y-X\beta)^T(Y-X\beta) = Y^TY-2\beta^TX^TY+\beta^TX^TX\beta$$

using the fact that $\beta^TX^TY = (\beta^TX^TY)^T = Y^TX\beta$, and differentiating $\epsilon^T\epsilon$ with respect to β. Thus from $\partial\epsilon^T\epsilon/\partial\beta = 0$ we have

$$-2X^TY+2X^TX\beta = 0$$

or

$$X^TX\beta = X^TY.$$

This solution for β gives us a stationary value of $\epsilon^T\epsilon$, and a simple algebraic identity confirms that $\hat{\beta}$ is a minimum [Ref. 2].

The multiplication X^TX generates a p' x p' matrix where the diagonal elements are the sums of squares of each of the independent variables and the off-diagonal elements are the sums of products between independent variables. The general form is

$$X^TX = \begin{pmatrix} n & \sum X_{i1} & \sum X_{i2} & \cdots & \sum X_{ip} \\ \sum X_{i1} & \sum X_{i1}^2 & \sum X_{i1}X_{i2} & \cdots & \sum X_{i1}X_{ip} \\ \sum X_{i2} & \sum X_{i1}X_{i2} & \sum X_{i2}^2 & \cdots & \sum X_{i2}X_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sum X_{ip} & \sum X_{i1}X_{ip} & \sum X_{i2}X_{ip} & \cdots & \sum X_{ip}^2 \end{pmatrix}. \quad (1.11)$$

Summation in all cases is over i = 1 to n, the n observations in the data. When only one independent variable is involved, X^TX consists of only the upper-left 2 x 2 matrix. Inspection of the normal equations will reveal that the elements in this 2 x 2 matrix are the coefficients of β_0 and β_1.

The elements of the matrix product X^TY are the sums of products between each independent variable in turn and the dependent variable:

$$X^TY = \begin{pmatrix} \sum Y_i \\ \sum X_{i1}Y_i \\ \sum X_{i2}Y_i \\ \cdots \\ \sum X_{ip}Y_i \end{pmatrix} \quad (1.12)$$

The first element, ΣY_i, is the sum of products between the vectors of ones (the first column of X) and Y. As is mentioned above, if only one independent variable is involved, X^TY consists of only the first two elements.

The fitted regression $\hat{Y} = X\hat{\beta}$, is denoted by $\hat{Y}$ (= $[(\hat{Y}_i)]$), and the elements of

$$e = Y - \hat{Y} = Y - X\hat{\beta} = (I_n - X(X^TX)^{-1}X^T)Y = (I_n - P)Y, \text{ say,}$$

are called the residuals. The minimum value of $\epsilon^T\epsilon$, namely,

$$\begin{aligned} e^Te &= (Y - X\hat{\beta})^T(Y - X\hat{\beta}) \\ &= Y^TY - 2\hat{\beta}^TX^TY + \hat{\beta}^TX^TX\hat{\beta} \\ &= Y^TY - \hat{\beta}^TX^TY + \hat{\beta}^T[X^TX\hat{\beta} - X^TY] \\ &= Y^TY - \hat{\beta}^TX^TY = Y^TY - \hat{\beta}^TX^TX\hat{\beta}, \end{aligned} \tag{1.13}$$

is called the residual sum of squares (RSS). Notice that $\hat{Y}$ and e are uniquely defined by $\hat{\beta}$.

C. VARIABLE SELECTION IN LEAST SQUARES ANALYSIS

The purpose of the least squares analysis will influence the manner in which the model is constructed. Hocking [Ref. 4] relates six potential uses of regression equations given by Mallows [Ref. 3]:

- Providing a good description of the behavior of the response variable
- Prediction of future responses and estimation of mean responses

- Extrapolation, or prediction of responses outside the range of the data
- Estimation of parameters
- Control of a process by varying levels of input
- Developing realistic models of the process.

Assume that the correct model involves $\underline{t}$ independent variables but that a subset of $\underline{p}$ variables, chosen randomly or on the basis of external information, is used in the regression equation. Let X_p and β_p denote submatrices of $\mathbf{X}$ and β that relate to the $\underline{p}$ selected variables. $\hat{\beta}_p$ will denote the least squares estimates of β_p obtained from the p-variate subset model and $MS(Res_p)$ the mean squares residual obtained from the p-variate subset model. After the above the following properties are summarized:

- $MS(Res_p)$ is a positively biased estimate of σ^2 unless the true regression coefficients for all deleted variables are zero.
- $\hat{\beta}_p$ is a biased estimate of β_p and <u>less</u> variable than the corresponding statistics obtained from the t-variate model [Ref. 4].

Stepwise regression methods are variable selection methods which identify good (although not necessarily the best) subset models, with considerably less computing than required for all possible regressions. The subset models are identified sequentially by adding or deleting, depending on the method,

the one variable that has the greatest impact on the residual sum of squares. These stepwise methods are not guaranteed to find the "best" subset for each subset size, and the results produced by different methods may not agree with each other.

D. RESEARCH METHODOLOGY

Given the regression model $Y = X\beta + \epsilon$, where X is n x p, a number of computational techniques have being suggested for solving the following steps:

- Solve the normal equations $X^T X \hat{\beta} = X^T Y$.
- Calculate the residual $e = Y - X\hat{\beta}$.
- Calculate the residual sum of squares $RSS = e^T e$.
- Update the regression model (that is, add or remove a row of X).
- Add or remove a regressor (that is, add or remove a column of X.)
- Calculate an F-statistic for a general linear hypothesis.

Solving the above steps is a common problem in a computer laboratory. These problems arise in a variety of areas and in a variety of contexts. Linear least squares problems are particularly difficult to solve because they frequently involve large quantities of data, and they are frequently ill-

conditioned[2]. The research methodology will be to describe acceptable procedures for updating regressions. The procedures to be chosen should be economically attractive and numerically accurate, in the case that the number of observations in the regression is large compared to the number of variables in the regression model. Each of these procedures (algorithms) has advantages and disadvantages which will be explained below.

E. THESIS ORGANIZATION

1. General

The main purpose of this thesis is to bring to the attention of readers numerically acceptable procedures for adding and deleting rows and columns from a regression model. For the case of a single row to be inserted or deleted, the algorithms use simple techniques: plane or hyperbolic rotations and the Gram-Schmidt process. The algorithms for inserting rows are stable, but the problem of deleting a row may be ill-conditioned and some algorithms for this process may be numerically unstable.

The need for updating regression results arises for various statistical or numerical reasons. When data are

---

[2]A set of linear equations $\mathbf{Bx} = \mathbf{c}$ is said to be ill-conditioned if small errors or variations in the elements of $\mathbf{B}$ and $\mathbf{c}$ have a large effect on the solution $\mathbf{x}$.

arriving sequentially, it may be undesirable or impossible to wait for all the data before obtaining some regression results. In various time-series problems, one is interested in the changing relationships between variables. A regression model with a fixed number of lagged terms, as it moves over a series of data, generates a "window" on the sample, with a new observation added, and an old one deleted[3], as the window moves to the next point in the series [Ref. 5].

2. Basic Schedule

The procedures for updating regressions that we shall discuss here are the following:

- An algorithm based on the normal equations (Efroymson M. A., 1960[Ref. 6], Draper N. and Smith H., 1966[Ref. 7]); this had been the standard introductory approach in regression courses. In this algorithm the regression coefficients are solved using Gauss-Jordan elimination on the normal equations. However, the problem of solving the normal equations is frequently much more ill-conditioned than the original problem of solving the overdetermined linear equations.

- Stepwise regression analysis with orthogonal transformations (Lars Elden 1972)[Ref. 8]. In this

---

[3]But in this case the problem can be ill-conditioned, because the rank can be decreased by this operation.

algorithm a method is presented using QR-decomposition. In the QR-decomposition or factorization, we express a matrix as a product of an orthogonal matrix Q and an upper trapezoidal matrix R (actually R stands for right trapezoidal). Many of the ideas used in this method have been proposed by Golub (1965) [Ref. 9].

■ The modified Gram-Schmidt triangular factorization (W. Gragg, R. Leveque, J. Trangenstein 1978) [Ref. 10]. This approach is **A** = **QR** factorization with **Q** having orthonormal columns and **R** upper triangular. In this algorithm one is able to add or drop regressors (columns of **A**) or observations (rows of **A**) using essentially only the storage needed for **A** and to secure numerical accuracy possibly at the expense of additional computation and storage.

■ Rank revealing QR-factorization (T. Chan, Hansen 1986). In this algorithm a method is presented for computing a column permutation, Π[4], and a QR factorization, $A\Pi = QR$, of an n by m ($n \geq m$) matrix **A** such that a possible rank deficiency[5] of **A** will be revealed in the triangular factor **R** having a small lower right block. Notice that a matrix A is rank deficient with rank deficiency $\underline{d}$ if it has at least $\underline{d}$

---

[4]Π is a permutation matrix where $\Pi \in \mathbb{R}^{nxn}$ and is the product of $P_1P_2...P_{n-1}$. Notice that P_i denotes the matrix representation of the column interchange that precedes step i.

[5]This means that the columns of the observation matrix **A** are not linearly independent.

singular values. For matrices of low rank deficiency, the algorithm can reveal the rank of **A**, and the cost is only slightly more than the cost of one regular QR factorization. A posteriori upper and lower bounds on the singular values[6] of **A** can be used to infer the numerical rank of **A**.

In Chapter II we discuss the general theory about updating regressions. A process for adding and dropping regressors follows. Techniques are presented with stepwise regression in mind, and we discuss how to compute the various quantities of statistical interest using the algorithms.

Chapter III mainly covers computational details of the algorithms, which are used to compute the regression coefficients. In this chapter also a simulation is applied to the basic algorithms for validating the models by creating useful numerical results. In the same chapter we discuss the measures of effectiveness of each model based on the previously generated numerical results. Chapter IV concludes the thesis and offers recommendations.

---

[6]The singular values of a matrix A are the "diagonal elements" of S, one of the three component matrices that A splitted to avoid singularity.

II. BACKGROUND THEORY

A. GENERAL THEORY FOR FITTING A SPECIFIED REGRESSION

Several techniques can be used to solve the problem of updating regressions. Efroymson's algorithm [Ref. 6] is one of the earlier ones used. It uses the method of Gauss-Jordan elimination on the normal equations $A^TAx = A^Tb$. A more efficient algorithm is to use the Cholesky factorization of the Gram matrix A^TA into R^TR where R is upper triangular. The solution to the original system is then found by a two step triangular solve process:

$$R^Ty = A^Tb, \quad Rx = y \quad \Rightarrow \quad A^TAx = R^Ty = A^Tb.$$

Another way to solve the above problem utilizes orthogonal transformations. This approach, called QR factorization, is the basis of the thesis algorithms and may be slightly slower than the normal equation approach, but is more stable numerically. The QR factorization of a matrix can be computed using the following methods:

- Classical Gram-Schmidt (CGM).
- Modified Gram-Schmidt (MGS).
- Givens rotations.
- Householder reflection (Elden's algorithm).

■ QR with Column Pivoting for the Rank Deficiency case (Chan's algorithm).

■ Daniel-Gragg-Kaufman-Stewart method (Gragg-Leveque-Trangenstein's algorithm).

The latter algorithm uses rotations and the Gram-Schmidt process with reorthogonalization.

Applying the QR technique on a square nonsingular system $\mathbf{Ax} = \mathbf{b}$ we have $(QR)\mathbf{x} = \mathbf{b}$ and by the associative law of matrix multiplication $Q(R\mathbf{x}) = \mathbf{b}$. Defining $\mathbf{y} = R\mathbf{x}$, it follows that $Q\mathbf{y} = \mathbf{b}$. Hence, $\mathbf{Ax} = \mathbf{b}$ is solved in two steps:

1. Solve $Q\mathbf{y} = \mathbf{b}$ for the unknown $\mathbf{y}$.
2. Solve $R\mathbf{x} = \mathbf{y}$ for the unknown $\mathbf{x}$.

The second system is solved by back substitution. The first system is easy to solve since Q is orthogonal. Multiplying $Q\mathbf{y} = \mathbf{b}$ by Q^T yields $Q^TQ = Q^T\mathbf{b}$. Since Q is orthogonal, $Q^TQ = I$ and $\mathbf{y} = Q^T\mathbf{b}$. When A has more rows than columns and is of full rank the above process yields the solution of the least squares problem $\|\mathbf{b} - \mathbf{Ax}\|_2^2 = \text{minimum}$. The key to the numerical stability is the orthonormality of the columns of Q. To avoid loss of orthogonality Gragg-Leveque-Trangenstein's algorithm applies reorthogonalization whenever $\|\grave{u}\|/\|u\|$ is small (say, less than $\sqrt{2}/2$), where u and $\grave{u}$ is the vector to be orthogonalized and its orthogonal projection, respectively.

Finally if A is rank deficient then the QR factorization need not give a basis for range(A). This problem can be

corrected, as mentioned below, by computing the QR factorization of the column pivoted version of A, $A\Pi = QR$ where Π is a permutation, and applying Chan's rank revealing scheme.

In the least square problem, statisticians and numerical analysts use different notations for the same entities, i.e., the matrix of the observations of independent variables, the vector of the dependent variables, the vector of random errors, etc. To avoid misunderstandings caused by using both notations in the following chapters, we built the following TABLE I of notational correspondents.

Most numerical analysts use n to represent columns of a matrix and m to represent rows. Others interchanges m and n, and we will use this notation.

TABLE I

NOTATIONS USED IN REGRESSION ANALYSIS

DESCRIPTION	STATISTITIANS	N. ANALYSTS
Independent var. matrix	X	A
Dependent var. col. vector	y	b
Vector of parameters to be estimated	β	x
Rows of indep. var. matrix	n	n
Cols of indep. var. matrix	p	m
Vector of random errors	ϵ	$\underline{r}$
Unique solution to the normal equation	$\hat{\beta}$	$\underline{x}$
The vector of estimated means of the dep. var.	$\hat{Y}$	p[7]
The idempotent $\underline{n \times n}$ matrix	P	QQ^T
The residuals vector	ϵ	r
Residual sum of squares	$SS(Res)$	r^Tr
Signific. level enter/stay	$F^{a/b}$	V

[7]The numerical analysts describe the vector $p = Ax = b-r$, as the "orthoprojector onto $\Re(A)$".

B. METHODS FOR UPDATING REGRESSIONS

Before starting this section, let us introduce some terminology. The upper case letters denote matrices, lower case letters denote vectors, and Greek letters denote scalars. Thus we write

$$x=\beta=\begin{pmatrix}\xi_1\\ \cdot\\ \cdot\\ \cdot\\ \xi_p\end{pmatrix}=\begin{pmatrix}\beta_1\\ \cdot\\ \cdot\\ \cdot\\ \beta_p\end{pmatrix},\quad b=y=\begin{pmatrix}\beta_1\\ \cdot\\ \cdot\\ \cdot\\ \beta_p\end{pmatrix}=\begin{pmatrix}\eta_1\\ \cdot\\ \cdot\\ \cdot\\ \eta_p\end{pmatrix},\quad A=X=\begin{pmatrix}\alpha_{11} & \cdots & \alpha_{1n}\\ \cdot & \cdots & \cdot\\ \cdot & \cdots & \cdot\\ \cdot & \cdots & \cdot\\ \alpha_{n1} & \cdots & \alpha_{nn}\end{pmatrix}=\begin{pmatrix}\xi_{11} & \cdots & \xi_{1n}\\ \cdot & \cdots & \cdot\\ \cdot & \cdots & \cdot\\ \cdot & \cdots & \cdot\\ \xi_{n1} & \cdots & \xi_{nn}\end{pmatrix} \qquad (2.1)$$

following the notational correspondents of TABLE I. For notational convenience we set x = β = b and the normal equations can now be written Xb = y or Ax = b.

1. Lars Elden (1960)

In Elden's algorithm the upper triangular matrix R in the QR-decomposition of A (in the model Ax = b) is determined by successive Householder transformations so that after k steps A has the form

$$P_kP_{k-1}\ \cdots\ P_1X=\begin{pmatrix}R^{(k)} & \vdots & & T^{(k)}\\ 0 & \vdots & a_{k+1}^{(k)} & \vdots & A_0^{(k)}\end{pmatrix}\begin{matrix}\}\ k\\ \}\ n-k\end{matrix}$$
$$\qquad\qquad\quad (k)\qquad\quad (1)\quad (n-k-1)$$

$R^{(k)}$ is upper triangular and $T^{(k)}$ is k x (m - k). Furthermore, $P_{k+1} = I-2w_{k+1}w_{k+1}^T$, where W_{k+1} with $\|w_{k+1}\|_2 = 1$ is now chosen so

that k first rows are left unchanged and

$$P_{k+1}\begin{pmatrix} 0 \\ a_{k+1}^{(k)} \end{pmatrix} = \lambda e_k, \quad \textit{where} \quad |\lambda| = \|a_{k+1}^{(k)}\|_2 .$$

After m steps X will be completely triangularized:

$$P_n P_{n-1} \ldots P_1 A = \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

In stepwise regression analysis, however, the final result may be of the form

$$Q^T A = \begin{pmatrix} R^{(p)} & T^{(p)} \\ 0 & A^{(p)} \end{pmatrix}, \qquad (2.2)$$

where $R^{(p)}$ is an upper triangular p x p matrix. The corresponding right hand side and the residual vector are:

$$Q^T y = \begin{pmatrix} c^{(p)} \\ d^{(p)} \end{pmatrix}, \quad \textit{and} \quad r = \begin{pmatrix} 0 \\ d^{(p)} \end{pmatrix}.$$

The regression coefficients are calculated from the system $R^{(p)}b = c^{(p)}$. In most cases the column vectors of X are not added to the subspace in the natural order, so that only after a permutation of columns is Q^TA of the form shown in equation (2.2).

a. Choice of Vector for Enlarging the Subspace.

We shall add to the subspace the vector that will make the norm of the residual vector decrease as much as

possible. Since only the m - k last rows of the matrix are changed in the kth step we can without loss of generality restrict ourselves to considering the first step of the analysis. We have the linear system of equations $Ax = b$, with A and b as in (2.1). Let $a_j = (\alpha_{1j}, \alpha_{2j}, \ldots, \alpha_{nj})^T$ denote the jth column vector of A. We shall multiply $(A : b)$ by an orthogonal matrix $P = I - 2ww^T$, where $\|w\|_2^8 = 1$, so that for some j we have $Pa_j = \alpha e_1$, where $|\lambda| = \|a_j\|_2$. After this transformation the residual vector is given by the last n - 1 components of Pb and since we were to decrease the norm of the residual vector as much as possible, j shall be chosen so that the first component $(Pb)_1$ of Pb will be as large as possible. Now we have $(Pb)_1 = (Pb)^T e_1 = (Pb)^T \lambda e_1/\lambda = b^T P^T P a_j/\lambda = b^T a_j/\lambda$. Thus we shall select j so that the quantity

$$v_j = \frac{(b^T a_j)^2}{|a_j|_2^2}$$

will be as large as possible [Ref. 9].

b. Choice of Vector for Diminishing the Subspace.

Suppose that after a number of steps $(A : b)$ has the form

---

[8]The Euclidean length of a vector is often denoted $\|x\|_2$, also called the 2-norm, since the components of x are raised to the second power.

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}.$$

$$\begin{pmatrix} R^{(k)} & D^{(k)} & c^{(k)} \\ 0 & A^{(k)} & d^{(k)} \end{pmatrix} \qquad (2.3)$$

where $R^{(k)}$ is an upper triangular (k x k) matrix. Let $r_p = (\rho_{1p}, \rho_{2p}, \ldots, \rho_{pp}, 0, \ldots, 0)^T$ denote the pth column vector of R_k. Now if the jth column vector of A is removed we can by successive rotations in the (j, j+1), (j+1, j+2), ..., (k-1, k) planes transform R_k to triangular form apart from the jth column. That is, we compute the coordinates of the remaining column vectors of A in the orthonormal basis where the jth base vector has been excluded.

Let $Q_{j,j+1}^{(\theta_j)}$ denote the orthogonal rotation matrix in the (j, j+1) plane which deviates from the unit matrix only in the elements $q_{j,j} = q_{j+1,j+1} = \cos\theta_j$ and $q_{j,j+1} = -q_{j+1,j} = \sin\theta_j$. If θ_j is chosen so that

$$\cos\theta_j = \frac{\rho_{j,j+1}}{\sqrt{\rho_{j,j+1}^2 + \rho_{j+1,j+1}^2}} \quad , \quad \sin\theta_j = \frac{\rho_{j+1,j+1}}{\sqrt{\rho_{j,j+1}^2 + \rho_{j+1,j+1}^2}}$$

we get, $Q_{j,j+1}^{(\theta_j)} r_{j+1} = Q_{j,j+1}^{(\theta_j)} (\rho_{1,j+1}, \ldots \rho_{j,j+1}, \rho_{j+1,j+1}, 0, \ldots, 0)^T =$

$= (\rho_{1,j+1}, \ldots, \rho_{j-1,j+1}, \rho'_{j,j+1}, 0, 0, \ldots, 0)^T$, where

$\rho'_{j,j+1} = \sqrt{\rho_{j,j+1}^2 + \rho_{j+1,j+1}^2}$.

Further we have

$$Q_{j,j+1}^{(\theta_j)} r_j = (\rho_{1,j}, \ldots, \rho_{j-1,j}, \rho'_{j,j}, \rho'_{j+1,j}, 0, \ldots, 0)^T$$

where $\rho'_{j,j} = \cos\theta_j \, \rho_{j,j}$ and $\rho'_{j+1,j} = -\sin\theta_j \, \rho_{j,j}$. After $\underline{k - j - 1}$ rotation we have transformed R_k to the form

$$R_0^{(k)} = Q_k R^{(k)} = \begin{pmatrix} \rho_{11} & \cdots & \rho_{1,j-1} & \rho_{1,j} & \rho_{1,j+1} & \cdots & \rho_{1,k} \\ & \ddots & \vdots & \vdots & \vdots & & \vdots \\ & & \rho_{j-1,j-1} & \rho_{j-1,j} & \rho_{j-1,j+1} & \cdots & \rho_{j-1,k} \\ & & & \rho'_{j,j} & \rho'_{j,j+1} & \cdots & \rho'_{j,k} \\ & & & 0 & \ddots & & \vdots \\ & & & \vdots & \ddots & \ddots & \vdots \\ & & & \vdots & \vdots & \ddots & \vdots \\ & & & \vdots & \vdots & & \rho'_{k-1,k} \\ & & & \rho'_{k,j} & 0 & \cdots & 0 \end{pmatrix}$$

where $Q_k = Q_{k-1,k}^{(\theta_{k-1})}, \ldots, Q_{j,j+1}^{(\theta_j)}$.

The preceding can be illustrated by the following schematic example (k = 5, j = 3)

$$R = \begin{pmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{pmatrix} \underset{\text{on } (3,4)}{\overset{\text{rotation}}{\Rightarrow}} \begin{pmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & + & \square & x \\ & & & & x \end{pmatrix} \underset{\text{on } (4,5)}{\overset{\text{rotation}}{\Rightarrow}} \begin{pmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & x & & x \\ & & + & & \square \end{pmatrix}$$

where x = any nonzero element, + = nonzero element being introduced, □ = element being annihilated.

To update the transposed inverse $R^{-T} = X^T$ we multiply by the same rotation matrices from the left as in the following schematic example.

$$X^T = \begin{pmatrix} x & & & & \\ x & x & & & \\ x & x & x & & \\ x & x & x & x & \\ x & x & x & x & x \end{pmatrix} \overset{\text{rotation on } (3,4)}{\Rightarrow} \begin{pmatrix} x & & & & \\ x & x & & & \\ x & x & \square & + & \\ x & x & x & x & \\ x & x & x & x & x \end{pmatrix} \overset{\text{rotation on } (4,5)}{\Rightarrow} \begin{pmatrix} x & & & & \\ x & x & & & \\ x & x & & x & \\ x & x & \square & x & + \\ x & x & x & x & x \end{pmatrix}$$

Since $R_0^{(k)}$ can be transformed to triangular form by permuting its columns, it is easy to see that $(R_0^{(k)})^{-1}$ will be triangular if the same permutation is performed on its rows. Therefore the jth column of $Q_k(R^{(k)})^{-T} = (R_0^{(k)})^{-T}$ has only one nonzero element. Let $x_j = (0, \ldots, 0, \xi_{jj}, \ldots, \xi_{jk})^T$ denote the jth column vector of $(R^{(k)})^{-T}$. Then $Q_k x_j = \lambda e_k$. Since $\|x_j\|_2 = \|Q_k x_j\|$ we have $|\lambda| = \|x_j\|_2$. The right hand side of (2.3) is multiplied by Q_k and as the dimension of the subspace is decreased the increase of the norm of the residual vector will come from the last component $(Q_k c^{(k)})_k$ of $Q_k c^{(k)}$. But

$$(Q_k c^{(k)})_k = e_k^T Q_k c^{(k)} = \frac{\lambda e_k^T Q_k c^{(k)}}{\lambda} = \frac{x_j^T Q_k^T Q_k c^{(k)}}{\lambda} = \frac{x_j^T c^{(k)}}{\lambda}.$$

Thus in each step we have to find for which value of j

$$v_j = \frac{(x_j^T c^{(k)})^2}{\|x_j\|_2^2}$$

is minimal, and if for this j the increase of the norm of the residual vector is not significant the jth column vector of X is removed from the subspace (the variable $\underline{x}_j$ is deleted from the regression).

c. Inserting a Row (Observation).

In some applications it may be desirable to insert a row to the matrix of observations after the analysis is finished and study what influence that row can have on the regression coefficients. This can be done without having to start from the beginning again using the QR-factorization that has already been done. The procedure can be illustrated by the following somewhat simplified example. Suppose we have the system of equations Rx = c, where

$$R = \begin{pmatrix} \rho_{11} & \rho_{12} & \cdot & \cdot & \cdot & \rho_{1n} \\ & \rho_{22} & \cdot & \cdot & \cdot & \rho_{2n} \\ & & \cdot & & & \cdot \\ & & & \cdot & & \cdot \\ & & & & \cdot & \cdot \\ & & & & & \rho_{nn} \end{pmatrix}, \quad c = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \cdot \\ \cdot \\ \cdot \\ \gamma_n \end{pmatrix}$$

The system is augmented by a row (an observation):

$$R_0 = \begin{pmatrix} \rho_{11} & \rho_{12} & \cdot & \cdot & \cdot & \rho_{1n} \\ & \rho_{22} & \cdot & \cdot & \cdot & \rho_{2n} \\ & & \cdot & & & \cdot \\ & & & \cdot & & \cdot \\ & & & & \cdot & \cdot \\ & & & & & \rho_{nn} \\ \alpha_{n+1,1} & \alpha_{n+1,2} & \cdot & \cdot & \cdot & \alpha_{n+1,n} \end{pmatrix}, \quad c_0 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \cdot \\ \cdot \\ \cdot \\ \gamma_n \\ \beta_{n+1} \end{pmatrix}$$

If we multiply $[R_0 \vdots c_0]$ by a rotation matrix $Q_{1n+1}(\theta_1)$ in the (1, n+1) plane where θ_1 is chosen so that

$$\cos \Theta_1 = \rho_{11} / \sqrt{\rho_{11}^2 + a_{n+1,1}} \quad , \quad \sin \Theta_1 = a_{n+1,1} / \sqrt{\rho_{11}^2 + a_{n+1,1}^2}$$

we obtain

$$R_0^{(1)} = Q_{1,n+1}^{(\Theta_1)} R_0 = \begin{pmatrix} \rho_{11}^{(1)} & \rho_{12}^{(1)} & \cdots & \rho_{1n}^{(1)} \\ & \rho_{22} & \cdots & \rho_{2n} \\ & & \ddots & \vdots \\ & & & \rho_{nn} \\ 0 & a_{n+1,2}^{(1)} & \cdots & a_{n+1,n}^{(1)} \end{pmatrix}$$

where

$$\rho_{1,j}^{(1)} = \cos \Theta_1 \, \rho_{1,j} + \sin \Theta_1 \, a_{n+1,j} \qquad j=1,\ldots,n$$

$$a_{n+1,j}^{(1)} = -\sin \Theta_1 \, \rho_{1,j} + \cos \Theta_1 \, a_{n+1,j} \qquad j = 2,\ldots,n.$$

After n successive rotations in the (1, n+1), (2, n+1), ..., (n, n+1) planes

$$R_0^{(n)} = Q_{n,n+1}(\Theta_n) \ldots Q_{1,n+1}(\Theta_1) R_0 = \begin{pmatrix} \rho_{11}^{(1)} & \rho_{12}^{(1)} & \cdots & \rho_{1n}^{(1)} \\ & \rho_{22}^{(2)} & \cdots & \rho_{2n}^{(2)} \\ & & \ddots & \vdots \\ & & & \rho_{nn}^{(n)} \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

c_0 has been transformed:

$$c_0^{(n)} = Q_{n,n+1}(\Theta_n), \ldots, Q_{1,n+1}(\Theta_1) c_0 = (c_1^{(1)}, c_2^{(2)}, \ldots, c_n^{(n)}, b_{n+1}^{(n)})^T$$

Now we have the system $R_0^{(n)} x = (c_1^{(2)}, c_2^{(2)}, \ldots, c_n^{(n)})^T$

(the component $b_{n+1}^{(n)}$ belongs to the residual vector).

d. Removing a Row (Observation).

Sometimes it may be desirable to remove a row of the matrix of observations after the solution has been obtained. This can be done in a very simple manner without the analysis having to be done from the beginning again. Suppose that the matrix **A** of observation has been decomposed:

$$Q_A^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \quad , \quad Q_A^T b = \begin{pmatrix} c \\ d \end{pmatrix} \tag{2.4}$$

Let a_ν ($\nu = 1,\ldots,m$) denote the row vectors of **X**. Then the normal equations of the system **Ax** = **b** can be written in the following form:

$$\sum_{\nu=1}^{m} a_\nu^T a_\nu = \sum_{\nu=1}^{m} a_\nu^T b_\nu .$$

If a_p is the row of A that is going to be removed, let

$$S = \begin{pmatrix} R \\ i a_p \end{pmatrix} \qquad \textit{where } i^2 = -1 .$$

Then $S^T S = R^T R - a_p^T a_p = R^T Q_A^T Q_A R - a_p^T a_p =$

$$= A^T A - a_p^T a_p =$$

$$= \sum_{\nu=1,\nu\neq p}^{m} a_\nu^T a_\nu .$$

It is easy to see that if the vector c in (2.4) is augmented by a component ib^p, the corresponding right hand side of the normal equation will be

$$\sum_{\nu=1,\ \nu\neq p}^{m} a_\nu^T b_\nu .$$

S can be brought to triangular form by successive multiplications by matrices $T_{1,n+1}$, $T_{2,n+1}$, ..., $T_{n,n+1}$ where $T_{\nu,n+1}$ is not unitary but complex orthogonal ($T_{\nu,n+1}^2 = I$). Consider the first step:

Let

$$T_{1,n+1} = \frac{1}{\sqrt{\rho_{11}^2-\alpha_{p1}^2}} \begin{pmatrix} \rho_{11} & & & i\alpha_{p1} \\ & 1 & & 0 \\ & & \ddots & \\ & 0 & & 1 \\ i\alpha_{p1} & & & -\rho_{11} \end{pmatrix}. \qquad (2.5)$$

Then

$$T_{1,n+1}\, S = \begin{pmatrix} \rho'_{11} & \rho'_{12} & \cdots & \rho'_{1n} \\ & \rho_{22} & \cdots & \rho_{2n} \\ & & \ddots & \vdots \\ & & & \rho_{nn} \\ 0 & i\alpha'_{p2} & \cdots & i\alpha'_{pn} \end{pmatrix}$$

where

$$\rho'_{1j} = \frac{(\rho_{11}\rho_{1j} - \alpha_{p1}\alpha_{pj})}{\sqrt{\rho_{11}^2 - \alpha_{p1}^2}} \qquad j = 1, \ldots, n$$

$$\alpha'_{pj} = \frac{1(\alpha_{p1}\rho_{1j} - \rho_{11}x_{pj})}{\sqrt{\rho_{11}^2 - \alpha_{p1}^2}} \qquad j = 2, \ldots, n.$$

After n transformations S has been brought to the form

$$\begin{pmatrix} R'' \\ 0 \end{pmatrix},$$

and the right hand side to the form

$$\begin{pmatrix} c'' \\ 1b''_p \end{pmatrix};$$

where we denote with double prime (") the final components of the normal equations. Then we have the system R"z = c" to solve. Notice that the component $1b''_p$ belongs to the residual vector. The hyperbolic rotations that are used to update the regression coefficient by working only on the matrix R are not unitary. Non-unitary transformations can destroy numerical stability.

2. Gragg-Leveque-Trangenstein (1978)

These algorithms are based on the use of (orthogonal) plane rotations and the modified Gram-Schmidt process with

reorthogonalization. We compute the following quantities of interest in a regression analysis, with stepwise regression:

- The least squares coefficients b,
- The residuals r = y - Xb,
- The ANOVA table[9],
- The covariance matrix for the regression coefficients, and
- Partial F-tests for the significance of a given variable in the regression.

a. Stepwise Regression: Adding and Deleting Regressors

Let us have entered $k - 1$ columns of X in the model $y = Xb$, and we wish to add the next column. This means that we want to find a $k \times k$ matrix Q_k whose columns form an orthonormal basis for the span of the k columns of X, assuming that Q_{k-1} is known. The remaining columns of X are regressed upon the first k columns. Let R_k be a $k \times k$ triangular matrix which maps the orthonormal basis into the original basis. Also r_k is the orthogonal projection of the vector y of dependent variable onto the space orthogonal to the space of the first k columns. In other words, r_k is the residual vector. Mathematically, with $k - 1$ columns entered, we have factored:

---

[9]Here, the residual sum of squares is computed directly, and not by subtracting the sum of squares due to regression from the total sum of squares.

$$\begin{pmatrix} X & y \end{pmatrix} = \begin{pmatrix} Q_{k-1} & A_{k-1} & r_{k-1} \end{pmatrix} \begin{pmatrix} R_{k-1} & T_{k-1} & c_{k-1} \\ 0 & I & 0 \\ 0^T & 0^T & 1 \end{pmatrix} \tag{2.6}$$

where Q_{k-1} is n x (k-1) and

$$Q_{k-1}{}^T Q_{k-1} = I; \tag{2.7}$$

also the columns of A_{k-1} and r_{k-1} are orthogonal to the columns of Q_{k-1}. This means that

$$Q_{k-1}{}^T A_{k-1} = 0, \quad Q_{k-1}{}^T r_{k-1} = 0; \tag{2.8}$$

Also R_{k-1} is (k-1) x (k-1) upper triangular; and

$$c_{k-1} = Q_{k-1}{}^T y. \tag{2.9}$$

The above procedure was a step of the Gram-Schmidt process that can be viewed as a partial QR factorization. Although there are some similarities between the Gram-Schmidt factorization and Householder's factorization, there are also some important differences. We are going to discuss that in the next chapter during the comparison process.

To update the above factorization (i.e., entering the kth column or, in other words, to add the kth variable), we perform the following steps:

- Repartition:

$$\begin{pmatrix} X & y \end{pmatrix} = \begin{pmatrix} Q_{k-1} & a_k & \tilde{A}_k & r_{k-1} \end{pmatrix} \begin{pmatrix} R_{k-1} & s_k & \tilde{T}_k & c_{k-1} \\ 0^T & 1 & 0^T & 0 \\ 0 & 0 & I & 0 \\ 0^T & 0 & 0^T & 1 \end{pmatrix} \qquad (2.10)$$

■ Normalization, and the equation (2.10) becomes:

$$\begin{pmatrix} Q_{k-1} & \frac{1}{\|a_k\|} a_k & \tilde{A}_k & r_{k-1} \end{pmatrix} \begin{pmatrix} R_{k-1} & s_k & \tilde{T}_k & c_{k-1} \\ 0^T & \|a_k\| & 0^T & 0 \\ 0 & 0 & I & 0 \\ 0^T & 0 & 0^T & 1 \end{pmatrix}$$

■ Orthogonalization, with $(q_k \equiv a_k/\|a_k\|)$, to obtain:

$$\begin{pmatrix} Q_{k-1} & q_k & \tilde{A}_k - q_k(q_k^T \tilde{A}_k) & r_{k-1} - q_k(q_k^T r_{k-1}) \end{pmatrix} \begin{pmatrix} R_{k-1} & s_k & \tilde{T}_k & c_{k-1} \\ 0^T & \|a_k\| & q_k^T \tilde{A}_k & q_k^T r_{k-1} \\ 0 & 0 & I & 0 \\ 0^T & 0 & 0^T & 1 \end{pmatrix}$$

■ Relabel, so that the equation (2.10) can be written as:

$$\begin{pmatrix} Q_k & A_k & r_k \end{pmatrix} \begin{pmatrix} R_k & T_k & c_k \\ 0 & I & 0 \\ 0^T & 0^T & 1 \end{pmatrix}$$

The above mathematical statements complete the insertion of the kth regressor. Furthermore the work required is essentially $2n(p - k)$ multiplications and additions. The storage space required and the part of storage affected by the algorithm is illustrated by these partitioned matrices.

Suppose now that we have to enter the kth variable (column) into the model. In this case the appropriate column to be added is the one which minimizes the norm of the residuals. We have that $q_k^T r_k = 0$ and from the orthogonalization and relabel steps, that

$$r_k = r_{k-1} - q_k(q_k^T r_{k-1}) \quad \text{or} \quad r_{k-1} = r_k + (q_k^T r_{k-1})\, q_k .$$

By the Pythagorean theorem,

$$\|r_{k-1}\|^2 - \|r_k\|^2 = (q_k^T r_{k-1})^2 \qquad (2.11)$$

so when we add the kth variable to the model the change in the residual is $(q_k^T r_{k-1})^2$. Recalling that $q_{kT} \equiv a_k^T/\|a_k\|$, the change can be expressed as $(q_k^T r_{k-1})^2 = (a_k^T r_{k-1}/\|a_k\|)^2$. So the chosen column to add at the kth stage is a column a_j, $k \le j \le p$, for which $|a_j^T r_{k-1}|/\|a_j\|$ is maximal. A test for significance should be satisfied for the above column to be entered into the regression. The work for choosing the column to add is on the order of $2n(p - k)$ multiplications and additions.

To compute the regression coefficients we have to follow the next steps. A $n \times k$ matrix X_k is formed by the first k columns of X. It has been factored $X_k = Q_k R_k$, where Q_k is $n \times k$ with orthonormal columns, and R_k is upper triangular. The projection of y onto the range of X can be written as:

$$y - r_k = Q_k c_k = X b_k = Q_k R_k b_k .$$

So the regression coefficient b_k can be computed easily by solving the following triangular system

$$R_k b_k = c_k,$$

reguiring an order of $0.5k^2$ multiplications and additions.

The next step is the computation of the total sum of squares $\|y\|^2$. A smart mathematical computation is taking place here. After computing the sum of squares due to the regression at the kth stage of the regression, as:

$$\|X_k b_k\|^2 = \|Q_k R_k b_k\|^2 = \|R_k b_k\|^2 = \|c_k\|^2,$$

since c_k is a k-vector, the residual sum of squares is computed directly as $\|r_k\|^2$, rather than by subtraction which would mean a loss of accuracy due to cancellation. This is recommended, especially if the mean residual sum of squares is to be used in sensitive tests for significance levels. The common criterion for terminating the selection process is the ratio of the reduction in residual sum of squares caused by the next candidate variable to be considered to the residual mean square from the model including that variable. This criterion can be expressed in terms of a critical "F-to-enter" or in terms of a critical "significance level to enter" (SLE), where F is the "F-test" of the partial sum of squares of the variable being considered. We can compute the "F-to-enter", Fe, for the kth variable by using the residual sum of squares before and after the insertion as in (2.6). Let $RSS_{k-1} = (r_{k-1})^T(r_{k-1})$ and $RSS_k = (r_k)^T(r_k)$ be respectively the residual sum of squares after each variable addition. Also let (n - m)

be the degrees of freedom for the last model, where n and m are the numbers of rows and the columns of the current matrix X of the independent variables. After the above, the "F-to-enter" is

$$Fe = \frac{RSS_{k-1} - RSS_k}{RSS_k/(n-m)}.$$

To compute the covariance matrix of the regression coefficients b_k using the normal equations, $(X_k^T X_k)^{-1}$ usually is computed. For avoiding this numerically unstable process, a new procedure is followed in this algorithm. Using the above orthogonal decomposition, it is found that:

$$(X_k^T X_k)^{-1} = (R_k^{-T})^T (R_k^{-T}),$$

where $R_{k-T} = (R_k^{-1})^T = (R_{kT})^{-1}$. The approach of the algorithm is to solve $R_k^T V_k = I$, and compute $(X_k^T X_k)^{-1}$ as $V_k^T V_k$. Hence, V_k is updated as follows:

$$\begin{pmatrix} I & 0 \\ 0^T & 1 \end{pmatrix} = \begin{pmatrix} R_{k-1}^T & 0 \\ r_{k-1}^T & \|a_k\| \end{pmatrix} \begin{pmatrix} V_{k-1} & 0 \\ -\frac{1}{\|a_k\|} r_k^T V_{k-1} & \frac{1}{\|a_k\|} \end{pmatrix} \equiv R_k^T V_k.$$

Thus, V_k differs from V_{k-1} only in the bordering of a new row and column. The above updating of V_k is mathematically equivalent to forwardsolving $R_k^T V_k = I$ directly. Notice that the computation of variances (i.e., the diagonal of the

covariance matrix) of the b_k requires only on the order of k^2 multiplications and additions. The variances of b_k are needed for significance tests (i.e., t-test and F-test).

Having entered k variables (columns) into the regression, we have the following factorization:

$$\begin{pmatrix} X & y \end{pmatrix} = \begin{pmatrix} Q_k & A_k & r_k \end{pmatrix} \begin{pmatrix} R_k & T_k & c_k \\ 0 & I & 0 \\ 0^T & 0^T & 1 \end{pmatrix}$$

$$(k) \quad (p-k) \quad (1)$$

where Q_k is $n \times k$ with orthonormal columns, R_k is $k \times k$ upper triangular, and

$$Q_k^T r_k = 0, \qquad Q_k^T A_k = 0, \qquad c_k = Q_k^T y. \tag{2.12}$$

Now, to delete the jth regressor, where $1 \le j \le k$, the next steps are followed. The columns of X are permuted, so that for some permutation matrix P,

$$\begin{pmatrix} X & y \end{pmatrix} P = \begin{pmatrix} Q_k & A_k & r_k \end{pmatrix} \begin{pmatrix} R_{k,(1,1)} & R_{k,(1,2)} & s_{j,1} & T_{k,1} & c_{k,1} \\ 0 & R_{k,(2,2)} & s_{j,2} & T_{k,2} & c_{k,2} \\ 0 & 0 & 0 & I & 0 \\ 0^T & 0^T & 0 & 0 & 1 \end{pmatrix} \tag{2.13}$$

$$(j-1) \quad (k-j) \quad (1) \quad (p-k) \quad (1)$$

For $k = 6$ and $j = 3$, the right-hand factor looks like the following:

$$\begin{pmatrix} R_{k,(1,1)} & R_{k,(1,2)} & s_{j,1} \\ 0 & R_{k,(2,2)} & s_{j,2} \end{pmatrix} = \begin{pmatrix} x & x & x & x & x & x \\ & x & x & x & x & x \\ & & x & x & x & x \\ & & x & x & x & \\ & & & x & x & \\ & & & & x & \end{pmatrix},$$

where x's denote possible nonzeros, and zeros are left blank. After that, by using a **Givens** transformation or (rotation), the subdiagonal of this matrix is annihilated. A **Givens** rotation is any orthogonal matrix of the form

$$\begin{pmatrix} I & & & \\ & G_{jk} & & \\ & & I & \\ & & & 1 \end{pmatrix} = G_{k-1,k}, \ldots, G_{j,j+1}$$

(j-1) (k-j+1) (n-k) (1)

where G_{jk} is of the form

$$\begin{pmatrix} \gamma & 0 & . & . & . & 0 & \sigma \\ 0 & 1 & & & & & 0 \\ . & & . & & & & . \\ . & & & . & & & . \\ . & & & & . & & . \\ 0 & & & & & 1 & 0 \\ \sigma & 0 & . & . & . & 0 & -\gamma \end{pmatrix}$$

The j and k subscript in G_{jk} correspond to the row numbers associated with the γ's: The first γ is in row k and the second γ is in row j. Next the equation (2.13) is partitioned and after using the orthogonality of **Givens** rotations and

performing multiplications, the following equation is obtained:

$$\begin{pmatrix} Q_{k,1} & Q_{k,2}G^T & A_k & r_k \end{pmatrix} \begin{pmatrix} R_{k,(1,1)} & R_{k,(1,2)} & s_{j,1} & T_{k,1} & c_{k,1} \\ 0 & GR_{k,(2,2)} & Gs_{j,2} & GT_{k,2} & Gc_{k,2} \\ 0 & 0 & 0 & I & 0 \\ 0^T & 0^T & 0 & 0^T & 1 \end{pmatrix} \tag{2.14}$$

Furthermore (2.14) is repartitioned and relabeled to get its last shape:

$$\begin{pmatrix} Q_{k-1} & A_{k-1} & r_{k-1} \end{pmatrix} \begin{pmatrix} R_{k-1} & T_{k-1} & c_{k-1} \\ 0 & I & 0 \\ 0^T & 0^T & 1 \end{pmatrix}$$

$(k-1)\,(p-k+1)\,(1) \quad (k-1)\,(p-k+1)\,(1)$

Note that (2.14) holds with k replaced by k - 1 and that the deleted column is ready to reenter the regression at any time. The above algorithm uses approximately $n(p+2k-3j)$ multiplications and additions.

However, the chosen column to drop is that one which yields the smallest increase in the residuals. This increase for the jth independent variable is $|\beta_j|^2 / \|v_j\|^2$, where β_j is the jth regression coefficient, and $\|v_j\|^2$ is the jth diagonal element of $(X_k^T X_k)^{-1}$. Thus the column to be dropped, if a test for lack of significance is satisfied, is the column j for which $|\beta_j| / \|v_j\|$ is smallest.

b. Adding and Deleting Observations

Now n rows of X have entered into the regression, and an additional row x^T and observation n of y is going to be inserted. In this situation, a factorization has taken place and we have:

$$\begin{pmatrix} X_{n+1} & y_{n+1} \end{pmatrix} \equiv \begin{pmatrix} X_n & y_n \\ x^T & n \end{pmatrix} = \begin{pmatrix} Q_n & 0 & r_n \\ 0^T & 1 & 0 \end{pmatrix} \begin{pmatrix} R_n & c_n \\ x^T & n \\ 0^T & 1 \end{pmatrix},$$

where Q_n is n x p with orthonormal columns, R_n is p x p upper triangular, also

$$Q_n^T Q_n = I \text{ and } p \times p,\ Q_n^T r_n = 0, \text{ and } c_k = Q_{nT} y_n. \qquad (2.15)$$

Furthermore the Givens rotations are applied and after relabeling, repartitioning and performing some elementary operations we get the following:

$$\begin{pmatrix} & \\ Q_{n+1} & \tilde{r}+\gamma q \end{pmatrix} \begin{pmatrix} R_{n+1} & c_{n+1} \\ 0^T & 1 \end{pmatrix}.$$

Notice that (2.15) holds with n replaced by n + 1 and that the above algorithm requires n + 1 additional storage locations for q. However, we can easily update $V = R^{-T}$, since

$$I = R_n^T V_n = (R_n^T \ x) \begin{pmatrix} V_n \\ 0 \end{pmatrix},$$

and we apply the Givens rotations to get:

$$I = (R_n^T \; x)\, G^T G \begin{pmatrix} V_n \\ 0 \end{pmatrix} \equiv (R_{n+1}^T \quad 0) \begin{pmatrix} V_{n+1} \\ v^T \end{pmatrix} = R_{n+1}^T V_{n+1}.$$

Now we want to delete the jth observation. Then if x^T is the jth row of X, n is the jth entry of y, q^T is the jth row of Q, and ρ is the jth entry of r, a permutation matrix P can be obtained such that:

$$P \begin{pmatrix} X_n & y_n \end{pmatrix} \equiv \begin{pmatrix} X_{n-1} & y_{n-1} \\ x^T & \eta \end{pmatrix} = \left(P \begin{pmatrix} Q_n \\ \end{pmatrix} \right) \begin{pmatrix} R_n & c_n \\ 0^T & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \tilde{Q} & \tilde{r} \\ q^T & \rho \end{pmatrix} \begin{pmatrix} R_n & c_n \\ 0^T & 1 \end{pmatrix} = \begin{pmatrix} \tilde{Q} & 0 & \tilde{r} \\ q^T & 1 & \rho \end{pmatrix} \begin{pmatrix} R_n & c_n \\ 0^T & 0 \\ 0^T & 1 \end{pmatrix}.$$

Next, we apply the Gram-Schmidt process. After this we have:

$$\omega = \sigma = (1 - q^T q)^{\frac{1}{2}}, \quad \sigma \tilde{q} = -\tilde{Q} q, \quad \tilde{\gamma} = \tilde{q}^T \tilde{r}, \quad r_{n-1} = \tilde{r} - \tilde{\gamma} \tilde{q};$$

when $\sigma = 0$, $(\tilde{q}^T, \omega)$ is chosen by a special feature of the orthogonalization code. Now if we use this orthogonalization, we get:

$$\begin{pmatrix} X_{n-1} & y_{n-1} \\ x^T & n \end{pmatrix} = \begin{pmatrix} \tilde{Q} & \tilde{q} & r_{n-1} \\ q^T & \omega & \tilde{\rho} \end{pmatrix} \begin{pmatrix} I & q & 0 \\ 0^T & \sigma & \tilde{\gamma} \\ 0^T & 0 & 1 \end{pmatrix} \begin{pmatrix} R_n & c_n \\ 0^T & 0 \\ 0^T & 1 \end{pmatrix} = \begin{pmatrix} \tilde{Q} & \tilde{q} & r_{n-1} \\ q^T & \omega & \tilde{\rho} \end{pmatrix} \begin{pmatrix} R_n & c_n \\ 0^T & \tilde{\gamma} \\ 0^T & 1 \end{pmatrix}. \quad (2.16)$$

Next we choose Givens transformations such that, $(q^T, \omega) G^T \equiv (q^T, \omega) G_{p,p+1} \ldots G_{1,p+1} = (0^T, \tau)$. Since $\| (q^T, \omega) \| = 1$, it follows that $\tau = \pm 1$. So the equation (2.16) can be written as:

$$\begin{pmatrix} X_{n-1} & y_{n-1} \\ x^T & \eta \end{pmatrix} = \begin{pmatrix} \tilde{Q} & \tilde{q} & r_{n-1} \\ q^T & \omega & \tilde{\rho} \end{pmatrix} \begin{pmatrix} G^T & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} G & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} R_n & c_n \\ 0^T & \tilde{\gamma} \\ 0^T & 1 \end{pmatrix}$$

$$= \begin{pmatrix} Q_{n-1} & 0 & r_{n-1} \\ 0^T & \tau & \tilde{\rho} \end{pmatrix} \begin{pmatrix} R_{n-1} & c_{n-1} \\ s^T & \gamma \\ 0^T & 1 \end{pmatrix}. \qquad (2.17)$$

because the matrix

$$\begin{pmatrix} \tilde{Q} & \tilde{q} \\ q^T & \omega \end{pmatrix} G^T \equiv \begin{pmatrix} Q_{n-1} & q^* \\ 0^T & \tau \end{pmatrix}$$

has orthonormal columns it follows $q^* = 0$ and because we have:

$$G \begin{pmatrix} R_n \\ 0^T \end{pmatrix} \equiv G_{1,p+1} \cdots G_{p,p+1} \begin{pmatrix} R_n \\ 0^T \end{pmatrix} = \begin{pmatrix} R_{n-1} \\ s^T \end{pmatrix},$$

where R_{n-1} is an upper triangular matrix. However from the equation (2.17) above, we obtain the desired factorization which is:

$$\begin{pmatrix} X_{n-1} & y_{n-1} \end{pmatrix} = \begin{pmatrix} Q_{n-1} & r_{n-1} \end{pmatrix} \begin{pmatrix} R_{n-1} & c_{n-1} \\ 0^T & 1 \end{pmatrix},$$

and we can recover the dropped row by:

$$x^T = \tau s^T, \qquad \eta = \tau\gamma + \tilde{\rho}.$$

Finally, we have to update $V = R^{-T}$. Now,

$$\begin{pmatrix} I & 0 \\ 0^T & 1 \end{pmatrix} = \begin{pmatrix} R_n^T & 0 \\ q^T & \omega \end{pmatrix} \begin{pmatrix} V_n & 0 \\ -q^T \dfrac{V_n}{\omega} & \dfrac{1}{\omega} \end{pmatrix} = \begin{pmatrix} R_n^T & 0 \\ q^T & \omega \end{pmatrix} G^T G \begin{pmatrix} V_n & 0 \\ -q^T \dfrac{V_n}{\omega} & \dfrac{1}{\omega} \end{pmatrix}$$

$$= \begin{pmatrix} R_{n-1}^T & s \\ 0 & \tau \end{pmatrix} \begin{pmatrix} V_{n-1} & v \\ v^{*T} & \nu \end{pmatrix} = \begin{pmatrix} R_{n-1}^T V_{n-1} + s v^{*T} & R_{n-1}^T v + s \\ \tau v^{*T} & \tau \nu \end{pmatrix}.$$

Since $\tau = \pm 1$, it follows that $v^* = 0$. Hence $R_{n-1}^T V_{n-1} = I$, and V_{n-1} is the desired update [Ref. 10].

3. Tony F. Chan (1986)

An algorithm is presented for computing a column permutation Π and a QR factorization $A\Pi = QR$ of an $\underline{n}$ by $\underline{m}$ ($n \geq m$) matrix A such that a possible rank deficiency of A will be revealed in the triangular factor R having a small lower right block. For matrices of low rank deficiency, the algorithm reveals the rank of A, and the cost is only slightly more than the cost of one regular QR factorization. An upper and lower bound on the singular values of A are stated. These can be used to infer the numerical rank of A.

A very useful factorization of an $\underline{n}$ by $\underline{m}$ ($n \geq m$) matrix is the QR factorization, given by $A\Pi = QR$, where $\Pi \in R^{nxn}$ is a permutation matrix, $Q \in R^{mxn}$ has orthogonal columns and satisfies $Q^TQ = I_n$, and $R \in R^{nxn}$ is upper triangular.

If A has full rank, then R is nonsingular. In many applications in which A is nearly rank deficient, it is desirable to select the permutation so that the rank

deficiency is exhibited in R having a small lower right block. For if R is partioned as

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

where R_{22} is d by d, then it is easy to show that $\sigma_{n-r+1}(A) \leq \|R_{22}\|_2$, where we have used the notation σ_i to denote the ith singular value of A, with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$. Therefore, if $\|R_{22}\|_2$ is small, then A has at least d small singular values and thus is close to being rank d deficient. The converse, unfortunately, is not true. In other words, if A has d small singular values, then it is not guaranteed that a given QR factorization of A has a small $\|R_{22}\|_2$. Let $A_n \in \mathbb{R}^{n \times n}$ be the matrix of order n illustrated below:

$$A_n = diag(1, s, s^2, \ldots, s^{n-1}) \begin{pmatrix} 1 & -c & -c & \cdots & -c \\ 0 & 1 & -c & \cdots & -c \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & -c \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix},$$

where s and c satisfy $s^2 + c^2 = 1$. For n = 50, c = 0.2, we have $\sigma_n(A_n) \approx 10^{-4}$. On the other hand, A_n is its own QR factorization and obviously has no small R_{22} block for any value of d.

Besides being able to reveal rank deficiency of A, a QR factorization with a small R_{22} block is very useful in many applications, such as in the rank deficient least squares

problem and in the subset selection problem. Therefore a variety of techniques have been proposed to compute it. Since QR factorization is essentially unique once the permutation Π is fixed, these techniques all amount to finding an appropriate column permutation of A. Perhaps the best known of these is the column pivoting strategy. Although this strategy is usually very effective in producing a triangular factor R with small $\|R_{22}\|$, very little is known in theory about its behavior, and it can fail on some matrices. Chan's algorithm does not require computing the SVD of A and is most closely related to one recently developed by Foster [Ref. 11].

a. Revealing Rank One Deficiency

Assume that A is nearly rank one deficient. We would like to find a column permutation of A such that the resulting QR factorization has a small pivotal element r_{nn}. It turns out that this permutation can be found by inspecting the size of the elements of the singular vector of A corresponding to the smallest singular value σ_n. This procedure was first pointed out in 1976 [Ref. 12].

Assume that there is a vector $x \in R^n$ with $\|x\|_2 = 1$ such that $\|Ax\|_2 = \epsilon$, and let Π be a permutation such that if $\Pi^T x = y$, then $|\eta_n| = \|y\|_\infty$ and $\|y\|_2 = \|x\|_2 = 1$. Such a x can be

obtained by the LINPACK[10] condition estimator. Now, because of these we have $|\eta_n| \geq \sqrt{1/n}$ and furthermore

$$Q^T A x = Q^T A \Pi \Pi^T x = Ry = \begin{pmatrix} \cdot \\ \cdot \\ \cdot \\ \rho_{nn}\eta_n \end{pmatrix}.$$

Therefore,

$$\varepsilon = |Ax|_2 = |Q^T Ax|_2 = |R\,y|_2 \geq |\rho_{nn}\eta_n|,$$

from which we have the result, $|\rho_{nn}| \leq \sqrt{n}\,\varepsilon$. Now let $v \in R^n$ with $|v_n| = 1$ be the right singular vector of A corresponding to the smallest singular value σ_n. Then we have

$$|Av|_2 = \sigma_n.$$

Therefore, by the above, if we define the permutation Π by

$$|(\Pi^T v)|_n = |v|_\infty,$$

then $A\Pi$ has a QR factorization with a pivot ρ_{nn} at least as small as $\sqrt{n}\sigma_n$ in absolute value.

Since only the permutation Π is needed, it is not necessary to compute the SVD of A in order to find v exactly. In practice, one can use a few steps of inverse iteration to compute an approximation to v from which the permutation Π can be determined. In the more interesting case where $\sigma_n < \sigma_{n-1}$, the

---

[10]Together, LINPACK and EISPACK represent the state of the art in software for matrix computation.

inverse iteration should converge rapidly. This suggests a two pass algorithm in which one first computes any QR factorization of A, then performs inverse iteration with R to find an approximate v, then determines Π, and then computes the QR factorization of $A\Pi$.

b. Revealing Higher Dimensional Rank Deficiency.

In this section, we consider the case where A is nearly d deficient, with d > 1. Our goal is to find a permutation Π such that if

$$A\Pi = QR \equiv Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

is the QR factorization of $A\Pi$, with $R_{22} \in R^{dxd}$, then $\|R_{22}\|$ is small in some norm.

A natural way to extend the one dimensional result of subsection a is to repeatedly apply the one dimensional algorithm, for d = 1,2,...,R_{11}, the leading principal triangular part of R. Suppose that we have already isolated a small d x d block R_{22}. To isolate a small (d+1) x (d+1) block, we compute, using the one dimensional algorithm given in subsection a, a permutation P such that $R_{11}P = Q_1\tilde{R}_{11}$ is the QR factorization of $R_{11}P$ and where the (n-d, n-d)th element of $\tilde{R}_{11}$ is small. Then with

$$\tilde{\Pi} \equiv \Pi \begin{pmatrix} P & 0 \\ 0 & I \end{pmatrix},$$

where, P is a permutation matrix such that $P \in R^{ixi}$, $|(P^T v)_1| = \|P^T v\|_\infty$ and the singular vector $v \in R^i$ corresponding to $\sigma_{min}(R_{11})$ with $\|v\|_2 = 1$, and $\delta_i = \sigma_{min}(R_{11})$, also

$$\tilde{Q} \equiv Q \begin{pmatrix} Q_1 & 0 \\ 0 & I \end{pmatrix}.$$

After the above, it can be easily verified that

$$A\tilde{\Pi} \equiv \tilde{Q} \begin{pmatrix} \tilde{R}_{11} & Q_1^T R_{12} \\ 0 & R_{22} \end{pmatrix}$$

is the QR factorization of $A\tilde{\Pi}$.

To make the above procedure more understandable, the updating process is illustrated for n = 5 and i = 2. The permutation is defined by moving the second column of R to the last column. Thus

$$R\,\Pi_1 = \begin{pmatrix} x & x & x & x & x \\ & x & x & x & x \\ & \square_1 & x & x & + \\ & & \square_2 & x & + \\ & & & \square_3 & + \end{pmatrix},$$

where again, x = any nonzero element, + = nonzero element being introduced and □ = element being annihilated. So working from the left in planes (i,i+1), (i+1,i+2), ... , (n-1,n) we

annihilate the subdiagonal of $R\Pi_i$ by rotations. Simultaneously we compute $\tilde{Q} = QQ_1$. The computation $\tilde{\Pi} = \Pi P_1$ is trivial, like that of $R\Pi_i$. If $\sigma_{min}(R_{11})$ is "tiny" then we have for n = 5,

$$A = \tilde{Q}\tilde{R}\,\tilde{\Pi}^H = \tilde{Q}\begin{pmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ - & - & - & - & - \\ & & & & \epsilon \end{pmatrix}\tilde{\Pi}^H$$

with ϵ also tiny. If ϵ is negligible we may drop the last row of $\tilde{R}$ and the last column of $\tilde{Q}$ to get

$$A \equiv \tilde{Q}\begin{pmatrix} x & x & x & x & \square_4 \\ & x & x & x & \square_3 \\ & & x & x & \square_2 \\ & & & x & \square_1 \end{pmatrix}\tilde{\Pi}^H$$

with $\tilde{Q}$ having orthonormal columns. Now if we annihilate the last column of the upper trapezoidal matrix, as before, from the bottom to top, and then drop the last columns of the Π and R matrices, we obtain $A \equiv \tilde{Q}\tilde{R}\,\tilde{\Pi}^H$.

The above algorithm, to produce the desired QR factorization, is based on the following two assertions for i = n, n-1, ..., n-d+1:

- R_{11} has a small singular value so that the (i,i)th element of $\tilde{R}_{11}$ is guaranteed to be small.

- The last [i.e., the (n-i)th] row of $Q_1^T R_{12}$ is small.

If these two assertions are true, then the lower (n-i+1) x (n-i+1) block of R is small and we have the desired QR factorization. But these two assertions are true because of the following lemmas proved in [Ref. 13].

Let $B \in R^{nxk}$ be a matrix containing any subset of k columns of A. Then

$$\sigma_{min}(B) \equiv \sigma_k(B) \leq \sigma_k(A).$$

Also if the matrix $W \equiv [w_{n-r+1},\ldots,w_n] \in R^{nxr}$ has been computed by the above algorithm then it should satisfy the following properties:

- $\|w_i\|_2 = 1$,
- $(w_i)_j = 0$ for $j > i$,
- $|(w_i)_i| = \|w_i\|_\infty >= 1/\sqrt{i}$,
- $\|A\Pi w_i\|_2 = \delta_i \leq \sigma_i(A)$.

Finally, Chan's algorithm [Ref. 13] computes a permutation Π and a QR factorization of A given by $A\Pi = QR$ where the elements of the lower d x d upper triangular block of R satisfies

$$|r_{jj}| \leq \sigma_j\sqrt{j} + \sum_{k=1}^{j-1} 2^{(j-1-k)}\sigma_k\sqrt{k} \leq 2^{(j-1)}\sigma_j\sqrt{n}$$

for $n-d < i \leq j \leq n$.

Very often, it is desirable to be able to infer the rank of A from its QR factorization by estimating the small singular values of A from the triangular factor R [Ref. 14]. For this, we state bounds on the singular values of A in terms of the matrices R and W, given by the following inequality, for $1 \leq j \leq d$,

$$\frac{\sigma_{n-j+1}}{\sqrt{j}\,\|(W_2^{j-1})\|_2} \leq \delta_{n-j+1} \leq \sigma_{n-j+1} \leq \|R_{22}^{j}\|_2 \leq \sigma_{n-j+1}\sqrt{j}\,\|(W_2^{j})^{-1}\|_2,$$

where, $W \in R^{nxn}$, $R \in R^{nxn}$ and computed by the rank revealing algorithm, R_{22}^{j} and W_2^{j} denote the lower right j by j upper triangular blocks of R and W respectively. These bounds are proved in [Ref. 13].

Now using the obtained $A\Pi = QR$ factorization we can solve the least square problem $Ax = b$ as follows. Let r be the rank of the n x m matrix A, then the regression coefficients are given by:

$$x = \Pi_{n,i}((R_{i,i})^{-1}Q_{n,i}b)$$

where i = 1, 2, 3, ... , r.[Ref. 15]

The work for the $A\Pi = QR$ factorization is given by: $W(r) = m^2(n - m/3) + Im^2r + 2m^2r$, where I is the number of inverse iterations used at each step. Usually I = 2 is sufficient in practice. So

$$W(r) = m^2(n - m/3) + 4m^2r. \qquad (2.18)$$

C. PROCESS

In the problem of finding a vector $x \in R^n$ such that $Ax = b$ where the data matrix $A \in R^{mxn}$ and the observation vector $b \in R^m$ are given and $m \geq n$ when there are more equations than unknowns, we say that the system $Ax = b$ is overdetermined. Usually an overdetermined system has no exact solution since b must be an element of range (A), a proper subspace of R^m.

This suggests that we strive to minimize $\|Ax - b\|_p$ for suitable choice of p. Different norms render different optimum solutions. However, much progress has been made in this area, and there are several good techniques available for 1-norm and ∞-norm minimization. The oldest method for solving the full rank least squares problem is the method of normal equations. The accuracy of the computed normal equations solution depends on the square of the condition number, and the algorithm is not always accurate.

For the above reason some techniques are established based on the QR factorization method. The Householder and Gram-Schmidt QR approaches to the least squares problem are more stable than the normal equation method. Furthermore techniques for rank revealing QR-factorization (i.e., Chan's algorithm discussed in subsection 3) can give a solution to the subset selection problem even if A is nearly rank deficient.

In the following chapters we will try to compare the results of the above algorithms computationally to verify the effectiveness of each on the least squares problem.

III. RESULTS AND COMPARISONS

A. PREVIOUS COMPARISONS

There have been a number of studies comparing Householder and Modified Gram-Schmidt QR techniques to form an orthonormal basis of $\mathfrak{R}(A)$ [Ref. 16]. Moreover, in Elden's paper there is a comparison between his and Efroymson's algorithm for updating regressions. There are comparisons in work and accuracy between the Classical and Modified Gram-Schmidt methods in the Gragg-Leveque-Trangenstein paper. Tony Chan also compares his column permutation QR factorization algorithm with the regular QR algorithm, in case of required work [Ref. 17]. Chan's comparison is applied only on the QR factorization part of the regression updating procedure.

Furthermore, an algorithm for solution of the subset selection problem is presented in a technical support package of NASA Tech Briefs [Ref. 18] that can be mentioned as a modified Chan's algorithm. So a comparison can be done between these algorithms. Finally a justification of the use of the reorthogonalization in Gram-Schmidt QR factorization is presented in the Daniel-Gragg-Kaufman-Stewart paper [Ref. 19].

B. MODEL AND DATA

The data in TABLE II-1 will be used to illustrate the model selection methods in the full rank case. In the table below there are two matrices, the 13 x 5 matrix **X** consisting of a column of ones, followed by the 4 column vectors of the observations on the independent variables, and the 13 x 1 column vector **Y** of observations of the dependent variable. In order to conform with previous results, the data utilized in this experiment is identical to that used in Elden's algorithm [Ref. 8]. We began the stepwise regression analysis by inserting and deleting columns of the observation matrix **X** and we continued by inserting and deleting rows of the model.

The data in TABLE II-2 and TABLE II-3, have one and two dependent columns respectively. We obtained the first of them by subtracting columns 2 and 3 of the data in TABLE II-1 and the second by repeating the first column of the same data.

TABLE II-1

CASE STUDY 1 (INPUT DATA)

MATRIX OF OBSERVATIONS

MATRIX OF OBSERV. ON THE INDEPENDENT VARIABLES					COLUMN VECTOR OF OBSER. ON DEPENDENT VARIABLES
1.00	7.00	26.00	6.00	44.00	78.50
1.00	1.00	29.00	15.00	52.00	74.30
1.00	11.00	56.00	8.00	29.00	104.30
1.00	11.00	31.00	8.00	17.00	87.60
1.00	7.00	52.00	6.00	33.00	95.90
1.00	11.00	55.00	9.00	22.00	109.20
1.00	3.00	71.00	17.00	6.00	102.70
1.00	1.00	31.00	22.00	44.00	72.50
1.00	2.00	54.00	18.00	22.00	93.10
1.00	21.00	47.00	4.00	26.00	115.90
1.00	1.00	40.00	23.00	10.00	83.80
1.00	11.00	66.00	9.00	12.00	113.30
1.00	10.00	68.00	8.00	12.00	109.40

To obtain the data in the Table II-2, (a non-full rank matrix), we replaced the column 4 of the Table II-1 with a new column, column 6. This new column was obtained by subtracting columns 2 and 3 of the matrix of observations in Table II-1. Now the new matrix of observations has a linearly dependent column and so it is rank one deficient.

TABLE II-2

CASE STUDY 2a (INPUT DATA)

MATRIX OF OBSERVATIONS

MATRIX OF OBSERV. ON THE INDEPENDENT VARIABLES					COLUMN VECTOR OF OBSER. ON DEPENDENT VARIABLES
1.00	60.00	31.00	29.00	-45.00	78.50
1.00	52.00	1.00	29.00	-28.00	74.30
1.00	20.00	11.00	56.00	-45.00	104.30
1.00	47.00	11.00	31.00	-20.00	87.60
1.00	33.00	7.00	52.00	-45.00	95.90
1.00	22.00	11.00	55.00	-30.00	109.20
1.00	6.00	3.00	71.00	-68.00	102.70
1.00	60.00	1.00	31.00	-30.00	78.50
1.00	22.00	2.00	54.00	-52.00	93.10
1.00	26.00	21.00	47.00	-26.00	115.90
1.00	34.00	1.00	47.00	-39.00	83.80
1.00	12.00	11.00	66.00	-55.00	113.30
1.00	12.00	10.00	68.00	-58.00	109.40

Furthermore in Table II-3 we inserted once more the column 1 of the matrix in Table II-2 as a new column 7. This column took the third place in the matrix and reduced its rank to rank two deficient.

TABLE II-3

CASE STUDY 2b (INPUT DATA)

MATRIX OF OBSERVATIONS

MATRIX OF OBSERV. ON THE INDEPENDENT VARIABLES						COLUMN VECTOR OF OBSER. ON DEPENDENT VARIABLES
1.00	60.00	1.00	7.00	26.00	-19.00	78.50
1.00	52.00	1.00	1.00	29.00	-26.00	74.30
1.00	26.00	1.00	11.00	56.00	-45.00	104.30
1.00	47.00	1.00	11.00	31.00	-26.00	87.60
1.00	33.00	1.00	7.00	52.00	-45.00	95.90
1.00	22.00	1.00	11.00	55.00	-44.00	109.20
1.00	6.00	1.00	3.00	71.00	-68.00	102.70
1.00	34.00	1.00	1.00	31.00	-30.00	72.50
1.00	22.00	1.00	2.00	54.00	-52.00	93.10
1.00	26.00	1.00	21.00	47.00	-26.00	115.90
1.00	34.00	1.00	1.00	40.00	-39.00	93.10
1.00	12.00	1.00	11.00	66.00	-55.00	113.30
1.00	12.00	1.00	10.00	68.00	-58.00	109.40

C. SIMULATION

As mentioned, the three compared algorithms use a QR factorization technique to solve the regression problem. But the three techniques used have different philosophies and different advantages and disadvantages. However, to obtain the numerical results needed for the comparison we used three

analogous sets of MATLAB codes. It is important that Elden's algorithm is not a "full" updating regression algorithm because there is no way to update regressions using Householder QR factorization. Elden adds or deletes columns or rows at the beginning or the end of the observations matrix and does not explicitly update the Householder QR factorization. Because of this we did not implement Elden's algorithm. Instead, we used the basic QR factorization algorithm to solve each individual problem. For column insertions and deletions on the right of the matrix our algorithms are computationally equivalent with Elden's algorithms.

D. VALIDATION

A numerical example was needed to illustrate the theory developed in Chapter II, so for each of the three algorithms we ran the corresponding MATLAB codes on the data of the case study. All computations are done on a 286 PC in double precision, with a relative machine precision of about 10^{-16}. Notice that during the stepwise procedure on case study 1, there was no rank deficient case, so there was no illustration of the effectiveness of Chan's algorithm to reveal the numerical rank of a given matrix.

E. MEASURES OF EFFECTIVENESS

1. Case Study 1 (Full Rank Case)

In order to compare the performances of the algorithms, the STSC'S Statistics/Graphics package STATGRAPHICS 4.0 was used on the same data. The results of the algorithms and STATGRAPHICS stepwise variable selection procedure are presented in the following tables.

STEPWISE REGRESSION ANALYSIS WITH ORTHOGONAL TRANSFORMATIONS

NUMBER OF OBSERVATIONS 13

STEP № 1

TABLE III-1. Summary Statistics for Stepwise Selection of Variables for the Data in TABLE II after step 1.

VARIABLE ENTERED AFTER THE COLUMN OF ONES: Column 5				
	(1) ELDEN	(2) GRAGG-LEVEQUE-TRANGENST.	(3) CHAN	(4) S.G.S. STATGRARH. STSC
CONST.	117.5679312	117.5679312	117.5679312	117.567931
VARIABLE COEFFICIENTS				
ALPHA5	-0.7381618	-0.7381618	-0.7381618	-0.738162
RES. SUM SQUAR.	883.8669169	883.8669169	883.8669169	881.89616
Fe LEVEL	22.7985202	22.7985202	22.7985202	22.7985
F OF GENER. MODEL	22.7995203	22.7995203	22.7995203	22.80

STEP Nº 2

TABLE III-2. Summary Statistics for Stepwise Selection of Variables for the Data in TABLE II after step 2.

VARIABLE ENTERED: Column 2				
	(1)	(2)	(3)	(4)
CONST.	103.0973816	103.0973816	103.0973816	103.097382
VARIABLE COEFFICIENTS				
ALPHA5	-0.6139536	-0.6139536	-0.6139536	-0.613954
ALPHA2	1.4399583	1.4399583	1.4399583	1.439958
RSS	74.7621122	74.7621122	74.7621122	74.7621
Fe	108.2239093	108.2239093	108.2239093	108.2239
F GEN. MODEL	176.6269531	176.6269531	176.6269531	176.6270

STEP Nº 3

TABLE III-3. Summary Statistics for Stepwise Selection of Variables for the Data in TABLE II after step 3.

VARIABLE ENTERED : Column 3				
	(1)	(2)	(3)	(4)
CONST.	71.6483069	71.6483069	71.6483069	71.648307
VARIABLE COEFFICIENTS				
ALPHA5	-0.2365402	-0.2365402	-0.2365402	-0.236540
ALPHA2	1.4519379	1.4519379	1.4519379	1.451938
ALPHA3	0.4161098	0.4161098	0.4161098	0.41611
RSS	47.9727294	47.9727294	47.9727294	47.9727
Fe	5.0258647	5.0258647	5.028647	5.0259
F GEN. MODEL	166.8316801	166.8316801	166.8316801	166.8317

STEP № 4

TABLE III-4. Summary Statistics for Stepwise Selection of Variables for the Data in TABLE II after step 4.

VARIABLE REMOVED: Column 5				
	(1)	(2)	(3)	(4)
CONST.	52.5773489	52.5773489	52.5773489	52.577349
VARIABLE COEFFICIENTS				
ALPHA2	1.4683057	1.4683057	1.4683057	1.468306
ALPHA3	0.6622505	0.6622505	0.6622505	0.662250
RSS	57.9044832	57.9044832	57.9044832	57.9045
Fe	1.8632873	1.8632873	1.8632873	1.8633
F GEN. MODEL	229.5036971	229.5036971	229.5036971	229.5040

STEP № 5

TABLE III-5. Summary Statistics for Stepwise Selection of Variables for the Data in TABLE II after step 5.

OBSERVATION ENTERED: Row 3, (a second time)				
	(1)	(2)	(3)	(4)
CONST.	52.6817201	52.6817201	52.6817201	52.681720
VARIABLE COEFFICIENTS				
ALPHA2	1.4584656	1.4584656	1.4584656	1.458466
ALPHA3	0.6594452	0.6594452	0.6594452	0.659445
RSS	59.9550974	59.9550974	59.9550974	59.9551
Fe	-----	-----	-----	-----
F GEN. MODEL	250.3437770	250.3437770	250.3437770	250.3438

STEP № 6

TABLE III-6. Summary Statistics for Stepwise Selection of Variables for the Data in TABLE II after step 6.

OBSERVATION ENTERED: Row 2, (a second time)				
	(1)	(2)	(3)	(4)
CONST.	53.0380112	53.0380112	53.0380112	53.038011
VARIABLE COEFFICIENTS				
ALPHA2	1.4484905	1.4484905	1.4484905	1.448490
ALPHA3	0.6549147	0.6549147	0.6549147	0.654915
RSS	60.8055442	60.8055442	60.8055442	60.8055
Fe	-----	-----	-----	-----
F GEN. MODEL	312.7948771	312.7948771	312.7948771	312.7950

STEP № 7

TABLE III-7. Summary Statistics for Stepwise Selection of Variables for the Data in TABLE II after step 7.

OBSERVATION REMOVED: Row 1				
	(1)	(2)	(3)	(4)
CONST.	53.0380116	53.0380116	53.0380116	53.038012
VARIABLE COEFFICIENTS				
ALPHA2	1.4484905	1.4484905	1.4484905	1.448490
ALPHA3	0.6549147	0.6549147	0.6549147	0.654915
RSS	60.8055442	60.8055442	60.8055442	60.8055
Fe	-----	-----	-----	-----
F GEN. MODEL	312.7948771	312.7948771	312.7948771	312.7949

2. Case Study 2 (Rank Deficient Case)

Here we want to test the thesis algorithms on a rank deficient case. So we used the singular matrices in Tables II-2 and II-3. The first matrix has one linearly dependent column (column 5) and the second two linearly dependent columns (column 3 and 6). We applied the corresponding algorithms' MATLAB codes to those matrices. The results of the algorithms' effectiveness are shown in the following tables.

TABLE IV-1. Summary statistics for the rank deficient case, (case study 2a).

RESULTS OF CASE STUDY 2a REGRESSION				
	(1) ELDEN	(2) GRAGG-LEVEQUE-TRANGENST.	(3) CHAN	(4) S.G.S. STATGRARH. STSC
CONST.	$3.2155*10^{18}$	-146.000000	71.6483069	-----
VARIABLE COEFFICIENTS				
ALPHA5	$-0.0353*10^{18}$	2.000000	-0.2365402	-----
ALPHA2	$-0.0194*10^{18}$	$2.5676*10^{14}$	1.4519379	-----
ALPHA3	$-0.0173*10^{18}$	$-2.5676*10^{14}$	0.4161098	-----
ALPHA6	$-0.0132*10^{18}$	$-2.5676*10^{14}$	0.0000000	-----
RES. SUM SQUAR.	$5.7494*10^{36}$	45.8653934	47.9727294	-----
F OF GENER. MODEL	$9.2803*10^{-34}$	116.4231889	166.8316801	-----

TABLE IV-2. Summary statistics for the rank deficient case, (case study 2b).

RESULTS OF CASE STUDY 2b REGRESSION				
	(1)	(2)	(3)	(4)
CONST.	$5.786*10^{14}$	$-7.6870*10^{45}$	71.6483097	-----
VARIABLE COEFFICIENTS				
ALPHA5	-0.007242	$3.8663*10^{12}$	-0.2365402	-----
ALPHA7	$-5.786*10^{14}$	$7.6870*10^{45}$	0.0000000	-----
ALPHA2	$0.708*10^{14}$	$-3.8663*10^{12}$	1.4519379	-----
ALPHA3	$-0.708*10^{14}$	$3.8663*10^{12}$	0.4161098	-----
ALPHA6	$-0.708*10^{14}$	$3.8663*10^{12}$	0.0000000	-----
RSS	59.17750	$1.4738*10^{28}$	47.9727294	-----
F GEN. MODEL	212.9952033	$3.953*10^{-31}$	166.8316801	-----

A discussion of the above results is given in the following chapter.

IV. CONCLUSIONS AND RECOMMENDATIONS

A. GENERAL

The major emphasis of this thesis was to examine the performance of three algorithms for updating regressions. As mentioned, the algorithms examined were Elden (Householder), Gragg-Leveque-Trangenstein and Chan. After running the corresponding MATLAB codes we obtain the results shown in Tables III and IV. The discussion of these results will be divided into two categories: the accuracy and stability, and the number of computations.

1. Accuracy and Stability

a. Full Rank Case

The results of the case study 1, in Table III, show us that no one algorithm uniformly outperforms any other with regard to accuracy in the full rank case. All algorithms give exactly the same results in all steps.

b. Deficient Rank Case

Comparing the results of the case study 2 (TABLES IV-1 and IV-2) with the results of the case study 1 step 3 (TABLE III-3) we can see that they are identical except for the coefficients of variables ALPHA6 and ALPHA7. These

variables are used only in the case study 2 and they correspond to the linearly dependent columns of the matrix of observations (see TABLES II-1 and II-2). The values of these variables' coefficients in both rank deficient cases are zero. Because of the above we can say that the results of Chan's algorithm in the rank deficient case without the linearly dependent columns are the same as the other algorithms' results in the full rank case. However in case study 2 (rank one and rank two deficient cases) only Chan's algorithm gives reasonable results. The results of the other two algorithms are totally wrong. Now it is easy to understand that the philosophy of Chan's algorithm is to drop the linearly dependent columns of the matrix of observations and work with the rest of them to obtain the regression coefficients.

2. The Number of Computations

To compare the volume of computation of the present algorithms we consider that all m column vectors of an n x m matrix A are added to the subspace.

a. Computing Regression Coefficients at Each Step

In the case where the regression coefficients are computed at each step the work of algorithms is as follows.

The number of flops[11] of Elden's method is about $\underline{2nm^2}$, $(2(n - m)m^2 + 4m^3/3$ for producing the QR factorization, $m^3/3$ for updating the inverse of the triangular matrix and $m^3/3$ for computing the regression coefficients). Elden's algorithm, as mentioned in Chapter II subsection B1, is updating regression without using at each step the matrix Q but working only on the triangular matrix R. This procedure avoids computations on matrix Q and so is cheap.

If the data matrix is not very rank degenerate Chan uses same work as Elden on the updating regression procedure, (Householder without updating Q). Using the equation (2.18) for this case with r = m we have a total work for the rank revealing QR factorization of $W(m) = nm^2 + 11m^3/3$. Also the amount of flops for computing the regression coefficients is $m^3/3$. In other words the work for Chan's QR factorization of $R\Pi$ is about $\underline{nm^2 + 4m^3}$ flops.

Gragg-Leveque-Trangenstein's algorithm requires 8nm flops for column insertion, 0 flops for column deletion, 3m(m + 2n) for row insertion, m(3m + 14n) flops for row deletion. In this case the regression coefficients are computed at each step we have to add the work of solving the triangular system $R_m b_m = c_m$. That requires a total of $m^3/2$ flops. This amount of flops is because, as mentioned in Chapter II subsection B2,

---

[11]Flops number means the whole number of multiplications and additions.

this algorithm updates the regression using the matrix Q for keeping the numerical stability. That means that in the worst case the algorithm requires a total work of about $6m^2 + 28nm + m^3/2$ flops.

After the above we can see that Elden's and Gragg-Leveque-Trangenstein's algorithms outperforms Chan's algorithm. As mentioned, in this thesis we always examine the case in which the observation matrix has more rows than columns (i.e., $n > m$). Keeping a ratio $n/m = 2$ Gragg-Leveque-Trangenstein's algorithm is the cheapest for n bigger than 18. This means that for big applications this algorithm is the cheapest.

b. Computing Regression Coefficients Once

In the case where the regression coefficients are computed once we have a lower order work for this computation which can be ignored. So the total work for the algorithms is: Elden's $2nm^2 - m^3/3$, Gragg-Leveque-Trangenstein's $6m^2 + 24nm$ and Chan's $nm^2 + 11m^3/3$. Keeping the same ratio as above Gragg-Leveque-Trangenstein's algorithm is still the cheapest for n bigger than 15.

B. RECOMMENDATIONS

In this thesis we examined three procedures of QR factorization to update the variable selection problem. The

above factorization can be also used on several other mathematical and statistical procedures. Important research can be done on using the QR updating algorithms to solve the linear programming problem using the simplex method and Karmarkar's algorithm.

As mentioned in Chapter III section D, we used Householder codes to simulate the procedure of Elden's algorithm. A closer comparison can be done by a full simulation of this algorithm. Finally, it should be interesting to extend this thesis in case where we have an observation matrix with fewer rows than columns (i.e., $n < m$).

APPENDIX A. MATLAB CODES FOR HOUSEHOLDER'S (ELDEN) ALGORITHM

The codes qrhf.m, qrhup.m, qrucd.m, qruri.m, qrurd.m qrorth.m and rot.m were reproduced here with Professor's Gragg permission.

```
  function W = qrhf(A)

% W = qrhf(A):
%
% W is a pack formed array which contains the HOUSEHOLDER QR
% FACTORIZATION of A. We have  A = QR  with  Q unitary and R
% upper trapezoidal with nonnegative diagonal elements.  The
% commands R = W, [Q R] = qrhup(R), executed by qrh, unpack.
% However this is not necessary, and in fact it's inefficient,
% for most applications.

% Copyright  (c)  16 February 1991 by Bill Gragg.  All rights
% reserved.

% qrhf calls sgn.

% begin qrhf
     [n m] = size(A);
     for k = 1:min(m,n)
        q = k:n;   u = A(q,k);   r = norm(u);   t = u(1);
        if r > 0
           t = u(1);        s = - sgn(t);     u(1) = t - r*s;
           t = abs(t);      t = 1 + t/r;      d = sqrt(2*t);
           A(q,k) = u/d;    d = r*sqrt(t);    u = u/d;
           if k < m
              p = k+1:m;    W = A(q,p);       W = W - u*(u'*W);
              t = W(1,:);    W(1,:) = s'*t;    A(q,p) = W;
           end
        end
     end
     W = A;
% end qrhf


  function [Q,R] = qrhup(R)

% [Q R] = qrhup(R):
```

```
% Computes the  ELEMENTWISE  QR factorization of A given the
% output R. Thus [Q R]=qrh(A):=qrhup(qrhf(A)) is essentially
% equivalent with matlab's function qr, except that we execute
% an inexpensive unitary diagonal scaling to make the diagonal
% elements of  R  nonnegative. It is NOT USEFUL to have Q in
% elementwise form to solve the  LS  problem, norm(b - Ax) =
% minimum. The purpose of matlab having Q in such form may be
% to avoid having to explain how the Householder least squares
% algorithm really works.
%
% Copyright  16  February  1991 by  Bill  Gragg.  All rights
% reserved.

% qrhup calls sgn.

% begin qrhup
     [n m] = size(R);  s = - sgn(diag(R));  e = ones(n-m,1);
     Q = diag([s;e]);  q = m+1:n;           z = zeros(n-m,1);
     root2 = sqrt(2);
     for k = min(m,n):-1:1
        q = [k q];   u = R(q,k);   r = norm(u);
        if r > 0
          u = u/(r/root2); T = Q(q,q);  Q(q,q) = T - u*(u'*T);
        end
        R(k,k) = r;
        if k < n
           R(k+1:n,k) = z;
        end
        z = [z;0];
     end
% end qrhup



  function W = sgn(Z)

% w = sgn(z) or W = sgn(Z):
%
% For z a complex number w is  z/abs(z) if z̃ = 0 and + 1 if
% z = 0. Thus sgn is the same as matlab's sign function EXCEPT
% when z = 0. We always have abs(w) = 1. W is the elementwise
% (Schur) sgn function of the complex matrix Z.

% Copyright (c) 19 January  1991  by  Bill  Gragg. All rights
% reserved.

% sgn calls no extrinsic functions.

% begin sgn
```

```
    W = sign(Z);   p = find(Z == 0);   n = length(p);
        W(p) = ones(n,1);
% end sgn

% Total flops (scalar case, see csgn):
% Real case:  0 flops.  Complex  case:  1 sqrt + 4 mults + 1
% add.



  function [x,r] = qrhls1(W,b)

% [x r] = qrhls1(W,b):
%
% Given that the HOUSEHOLDER QR FACTORIZATION of A, is stored
% in packed form in the array W, qhrs SOLVES the least squares
% problem  norm(b - Ax) = minimum  for x. It also efficiently
% computes the residual r := (b - Ax). We assume that rank(A)
% = m <= n, where A has m columns and n rows.
%
% qhrs calls no extrinsic functions.
%
% begin qrhs
    [n m] = size(W);   root2 = sqrt(2);
% Forward solving:  computing Q'b = H(m)H(m-1)...H(1)b.
    for k = 1:m
       q = k:n;   u = W(q,k);   r1 = norm(u);   d(k) = r1;
       if r1 > 0
          u = u/(r1/root2); v = b(q); b(q) = v - u*(u'*v);
       end
    end
% Backsolving:  solving Rx = b1 := b(1:m) for x.
    s = - sgn(diag(W));   x = zeros(m,1);
    x(m) = s(m)'*b(m)/d(m);
    for k = m-1:-1:1
       p = [k+1 p];  x(k) = (s(k)'*b(k) - W(k,p)*x(p))/d(k);
    end
% Computing the residual r.
    if m < n  r = b - W*x; else r = 0;
    end
% end qrhls1



  function [Eb,RSS,F,Fe] = regres2(Q,R,P,RSSp,b,r)

% [Eb RSS F Fe]= regres2(Q,R,P,RSSp,b,r):
%
```

```
% Eb = E(b), is the expectation of  beta of the linear least
% squares problem, norm(b - Ax) = minimum, RSS is the residual
% sum of squares and  F  the statistic  for a general linear
% hypothesis. Also Fe is the significance level to enter, SLE,
% "F-to-enter", for the new variable, on where A = QRP' is the
% QR  general  FACTORIZATION of A. p = A*x is the orthogonal
% projection of b onto R(A) and the residual vector r = b - pr
% = b - Ax, is the orthogonal projection of b onto N(A'), the
% orthogonal complement of R(A).  It is assumed that A is not
% a zero matrix.
%
% regres2 calls rows, cols, ones and norm.
%
% begin regres2

   Eb = r + Q*R*P'*x;
   A = Q*R*P';          n = rows(A);          m = cols(A);
   e = ones(n,1);       RSS = r'*r;           v = Q'*b;
   f = (norm(v)-abs(e'*b)/sqrt(n))*
       (norm(v)+abs(e'*b)/sqrt(n));
   df = (n-m)/(m-1);  F = (df*f)/RSS;
   Fe = (abs(RSSp-RSS)*(n-m))/RSS;

% end regres2
```

APPENDIX B. MATLAB CODES FOR GRAGG-LEVEQUE-TRANGENSTEIN'S ALGORITHM

```
  function [Q,R] = qruci(Q,R,a,i)

% [Q R] = qruci(Q,R,a,i):
%
% UPDATES the QR factorization  A = QR  when a is INSERTED as
% COLUMN i of A. Q  has orthonormal columns and  R  is upper
% trapezoidal with at least as many columns as rows.

% Copyright  (c)  20  July 1991 by  Bill  Gragg.  All rights
% reserved,

% qruci calls qrorth and rot.

% begin qruci
    [r m] = size(R);          n = length(a);   [q s t] =
    qrorth(Q,a);
    R(:,i+1:m+1) = R(:,i:m);  R(:,i) = s;      m = m + 1;
    if r < n
       Q = [Q q];   R = [R;zeros(1,m)];
       r = r + 1;   R(r,i) = t;
    end
    for k = r-1:-1:i
       p = k+1:m;            q = k:k+1;          [c s t] =
       rot(R(q,i));
       R(q,i) = [t;0];       G = [c - s';s c'];  Q(:,q) =
       Q(:,q)*G;
       R(q,p) = G'*R(q,p);
    end
    if i < r
       q = i+1:r;            d = diag(R);        d =
       sgn(d(q));
       D = diag(d);          Q(:,q) = Q(:,q)*D;  R(q,:) =
       D'*R(q,:);
    end
% end qruci
```

```
  function [Q,R,a] = qrucd(Q,R,i)

% [Q R a] = qrucd(Q,R,i):
```

```
%
% UPDATES the QR factorization A = QR when COLUMN i is DELETED
% from A. Q has orthonormal columns and R is upper trapezoidal
% with at least as many columns as rows. The deleted column is
% called a [Ref 19].
% Copyright (c) 20 July 1991 by Bill Gragg. All rights
% reserved,

% qrucd calls rot.

% begin qrucd
    if nargout > 2     a = Q*R(:,i);  end,
    [r m] = size(R);   R(:,i) = [];   m = m - 1;
    for k = i:r-1
       p = k+1:m;      q = k:k+1;     [c s t] = rot(R(q,k));
       R(q,k) = [t;0];                G = [c -s';s c'];
       Q(:,q) = Q(:,q)*G;
       if k < m           R(q,p) = G'*R(q,p);   end
    end
    if r > m
       r = r - 1;         q = 1:r;
       Q = Q(:,q);        R = R(q,:);
    else
       s = sgn(R(r,r));                Q(:,r) = Q(:,r)*s;
       R(r,:) = s'*R(r,:);
    end
% end qrucd



  function [Q,R] = qruri(Q,R,a,j)

% [Q R] = qruri(Q,R,a,j):
%
% UPDATES the factorization A = QR when a' is INSERTED as ROW
% j of A. Q has orthonormal columns and R is upper trapezoidal
% with at least as many columns as rows.

% Copyright (c) 20 July 1991 by Bill Gragg. All rights
% reserved,
% qruri calls rot.
% begin qruri
    m = length(a);  [n r] = size(Q);
    n = n + 1;      Q(j:n,:) = [zeros(1,r);   Q(j:n-1,:)];
    r = r + 1;      Q(:,r) = zeros(n,1);      Q(j,r) = 1;
    R = [R;a'];
    for k = 1:r-1
      p = k+1:m;                q = [k r];
      [c s t] = rot(R(q,k));
      R(q,k) = [t;0];           G = [c -s';s c'];
      Q(:,q) = Q(:,q)*G;
```

```
      if k < m                    R(q,p) = G'*R(q,p);    end
   end
   if r > m
      r = r - 1;                  q = 1:r;
      Q = Q(:,q);                 R = R(q,:);
   else
      s = sgn(R(r,r));            Q(:,r) = Q(:,r)*s;
      R(r,:) = s'*R(r,:);
   end
% end qruri



  function [Q,R,a] = qrurd(Q,R,j)

% [Q R a] = qrurd(Q,R,j):
%
% UPDATES the QR factorization  A = QR  when ROW j is DELETED
% from A. Q has orthonormal columns and R is upper trapezoidal
% with at least as many columns as rows. a' = A(j,:) = Q(j,:)R
% is the deleted row.


% qrurd calls qrorth and rot.
% Copyright  (c)  20  July 1991 by  Bill  Gragg.  All rights
% reserved,

% begin qrurd
   [n r] = size(Q);   [r m] = size(R);   t = [1:j-1 j+1:n];
   if r < n
      q = zeros(n,1);          q(j) = 1;          r = r + 1;
      Q(:,r) = qrorth(Q,q);    R(r,:) = zeros(1,m);
   end
   for k = r-1:-1:1
      p = k:m ;                q = [k r];
      [c s u] = rot(Q(j,q));
      Q(j,r) = u;              G = [-s c';c s'];
      Q(t,q) = Q(t,q)*G;
      R(q,p) = G'*R(q,p);
   end
   r = r - 1;       q = 1:r;       Q = Q(t,q);
   a = R(r+1,:)';   R = R(q,:);
   d = diag(R);     d = sgn(d);    D = diag(d);    Q = Q*D;
   R = D'*R;
% end qrurd



  function [q,r,s,k] = qrorth(Q,a)

% [q r s k] = qrorth(Q,a) or q = qrorth(Q):
```

```
%
% ORTHONORMALIZES  column a against the columns of  Q  using
% REORTHOGONALIZATION. It is  assumed that  Q  has  (nearly)
% orthonormal columns. Let [n m] = size(Q). An error message
% occurs if m > n.  For m <= m we have
%
%                [Q a] = [Q q][I r]
%                                [  s]  with  Q'q = 0,  s >= 0
%
% and, if m < n,
%
%                                    q'q = 1.
%
% If m = n we take  q := 0 and  s := 0. If a is not input we
% take a := 0. For m < n and a = 0 we get a unit vector q in
% the orthogonal complement of the range of Q.


% qrorth calls no extrinsic functions.
% Copyright  (c)  20  July 1991 by  Bill  Gragg.  All rights
% reserved,

% begin qrorth
     [n m] = size(Q);
     if nargin < 2  a = zeros(n,1);  end
     if m > n  error('Q has too many columns.'),  end
     if m == n  q = zeros(n,1);  r = Q'*a;  s = 0;  k = 0;
  return,  end
     norma = norm(a);    t = norma/2;
     r = Q'*a;    b = a - Q*r;    s = norm(b);    k = 1;
     if norma == 0  zflag = 1;  k = 0;  end
     while 0 == 0
        if s > t  q = b/s;  break,  end
        if k > 4  error('Process did not terminate in 5
  iterations.'),  end
        if s <= t*eps/10
           [u j] = min(norms(Q'));
           if s == 0
              s = eps/4;    b(j) = s;
           else
              b(j) = b(j) + s*eps/4;
           end
           norma = s;    k = k + 1/2;
        end
        t = s/2;    b = b - Q*(Q'*b);  s = norm(b);  k = k + 1;
        end
     end
     if zflag  r = zeros(m,1);    s = 0;  end
% end qrorth
```

```
  function [c,s,r] = rot(x,y)

% [c s r] = rot(x,y) or [c s r] = rot(z):
%
% Carefully computes the Gram-Schmidt QR factorization
%
%                              z := [x] = [c] r
%                                   [y]   [s]   .
%
% Note that
%
%                            [x] = [c -s'] [r] =: QR
%                            [y]   [s  c'] [0]
%
% is a full QR factorization. This is a "tool of the trade" in
% computational  linear  algebra. Note that  Q  and  Q'  are
% ROTATIONS and that
%
%                              [ c' s'] [x] = [r]
%                              [-s  c ] [y]   [0] .

% Copyright (c) 28 October 1990  by  Bill  Gragg.  All  rights
% reserved.

% rot calls no extrinsic functions.

% begin rot
     if nargin < 2  y = x(2);  x = x(1);  end
     c = sign(x);   s = sign(y);   x = abs(x);   y = abs(y);
     if y > 0
        if x > y
          t = y/x;   u = sqrt(1 + t*t);   r = x*u;    v = t/u;
          u = 1/u;    c = u*c;              s = v*s;
        else
          t = x/y;   u = sqrt(1 + t*t);   r = y*u;    v = 1/u;
          u = t/u;    c = u*c;              s = v*s;

        end
     else
        r = x;   if r == 0  c = 1;  end
     end
% end rot

% Total flops:
%    Real case:     1 sqrt +  5 mults + 1 add.
%    Complex case:  1 sqrt + 11 mults + 3 adds.
```

```
  function [x,r] = qrmgsls(Q,R,b)

% [x r] = qrmgsls(Q,R,b):

% Solves the  norm(b - Ax) = minimum  for  x  given the  QR
% factorization A = QR  and computes the minimal norm rnorm
% using the MODIFIED GRAM-SCHMIDT process (mgs).
%
% qrmgss calls gfsb.

% begin qrmgsls
% Compute the "Fourier coefficients" c = Q'b and the residual
% vector r = b - Ax = b - QRx = b - Qc = (I - QQ')b, WITHOUT
% COMPUTING x.

     r = b;
     for i = 1:cols(Q)
        q = Q(:,i);   t = q'*r;   c = [c;t];
     end
%    Compute r and backsolve for x.
     x = gfsb(R,c);   r = b - Q*c;
% end qrmgsls
```

APPENDIX C. MATLAB CODES FOR CHAN'S ALGORITHM

```
    function [Q,R,Pi,rank,W,delta] = rrqr(A,tol)
% R = rrqr(A,tol)
% [Q,R,Pi,rank,W,delta] = rrqr(A,tol)
%
% Compute a rank revealing QR factorization of A:
%    A*Pi = Q*R = Q [ R_11 R_12 ] ,
%                 [  0   R_22 ]
% such that Q is m-by-n, R is triangular n-by-n, and
%  1)  R_11 is rank-by-rank and well-conditioned,
%  2)  rank = numerical rank of A with respect to tol,
%      defined as the number of singular values greater than
%      tol,
%  3)  norm(R_22) is of the same order of magnitude as the
%      (rank+1)'th singular value.
%
% Also, return a matrix  W  whose columns are orthonormal and
% span an aproximation to the null "N" space of A, and the δ
% delta(rank:n1:2) containing lower and upper bounds for the
% last n-rank+1 singular values of A (the first rank-1 rows of
% delta are zero).
%
% If no tol is specified, sqrt(n)*norm(A,1)*eps is default.
%
% This program is an implementation of the algorithm described
% in the paper: T. Chan, "Rank Revealing QR factorizations",
% Lin. Alg. Appl. 88/89 (1987), 67-82.
% The use of the factor  c_max_ratio was suggested in: C. H.
% Bischof, &  P. C. Hansen,  "Structure preserving and  rank
% revealing  QR-factorizations",  SISSC, to  appear
  [m,n] = size(A); delta = zeros(n,2); c_max_ratio = 10;
  if (nargin==1), tol = sqrt(n)*norm(A,1)*eps; end
  if (nargout>4), W = []; end

% Compute an initial QR factorization A*Pi = Q*R.
  [Q,R,Pi] = qr(A); Q = Q(:,1:n); R = R(1:n,:);

% Prepare  for the iterations.  Estimate  smallest  singular
% value of R.
  nu = 0; i = n;
  [sest,v] = ccvl(R); if (nargout>5), delta(n,1) = sest; end

% Loop  until a  singular value  estimate larger than tol is
% found.
  while sest < tol, nu = nu+1;

% Update the matrix W.
```

```
  if (nargout>4), W = [[v;zeros(nu-1,1)],W]; end

% Find the element in v with greatest index, numerically
% within a factor  c_max_ratio  of the  numerically  largest
% element.
  vmax = norm(v,inf);
  for k=i:-1:1
    if (vmax <= abs(v(k))*c_max_ratio), break, end
  end

% If necessary, generate the permutation that brings the pivot
% element of v to the last position, apply the permutation to
% W, Pi, and R, compute a new QR factorization of R(1:i,1:i),
% and update Q and R.
  if (k<i)
    p = [k+1:i,k]; Pi(:,k:i) = Pi(:,p); R(:,k:i) = R(:,p);
    if (nargout>4), W(k:i,:) = W(p,:); end
    for j=k:i-1
      [c,s] = gen_g(R(j,j:n),R(j+1,j:n));
      [R(j,j:n),R(j+1,j:n)] =
      app_g_left(c,s,R(j,j:n),R(j+1,j:n));
      R(j+1,j) = 0;
      [Q(:,j),Q(:,j+1)] = app_g_right(c,s,Q(:,j),Q(:,j+1));
    end
  end

% Provide an upper bound for the i'th singular value.
  if (nargout>5), delta(i,2) = norm(R(i:n,i:n)); end

% Estimate the smallest singular value of R(1:i-1,1:i-1),
% which is a lower bound for the (i-1)'th singular value.
  i = i-1; [sest,v] = ccvl(R(1:i,1:i));
  if (nargout>5), delta(i,1) = sest; end

  end

% Finish the computation.  If nargout < 2, return R.
  if (nargout>5), delta(i,2) = norm(R(rank:n,rank:n)); end
  rank = n - nu; if (nargout>4), W = Pi*W; end
  if (nargout < 2), Q = R; end
% This algorithm is described by T. Chan & P. Hansen [Ref 15].



  function [x,r] = basic(Q,R,Pi,b,rank)
%
% [x r] = basic(Q,R,Pi,b,rank)
%
% Compute the basic solution.  If the RRQR of A is
%    A*Pi = Q*R = Q [ R_11 R_12 ] ,
%                   [  0   R_22 ]
```

```
% where A_11 is rank-by-rank, then the basic solution is
%    x = Pi*[ inv(R_11) 0 ]*Q'*b .
%           [     0     0 ]

% Ref.: G. H. Golub & C. F. Van Loan, "Matrix Computations",
% Johns Hopkins, 1989.  Subsection 5.5.6.

% Per Christian Hansen, UNI-C, 07/11/90.

  x = Pi(:,1:rank)*(R(1:rank,1:rank)\(Q(:,1:rank)'*b));
  r = b - Q*R*Pi*x;
```

LIST OF REFERENCES

1. Rawlings, J. O., Applied Regression Analysis: A Research Tool, Wiley & Sons, 1988.

2. Seber, G. A., Linear Regression Analysis, John Wiley & Sons, 1977.

3. Mallows, C.L., Data Analysis In a Regression Context, Proceedings of University of Kentucky Conference on Regression With a Large number of Predictor Variables, Department of Statistics, University of Kentucky.

4. Hocking, R. R., "The analysis and selection of variables in linear regression", Biometrics, vol. 32, pp 17-22, 1973.

5. Chambers J. M., "Regression Updating", Journal of the American Statistical Association, vol. 66, pp 744-748, 1971.

6. Efroymson, M. A., Multiple Regression Analysis, Mathematical Methods for Digital Computers, vol. I, edit. Anthony Ralston and Herbert S. Wilf, New York: John Wiley & Sons, pp 191-203, 1960.

7. Draper, N. R., and Smith, H., Applied Regression Analysis, New York: John Wiley & Sons, 1966.

8. Elden, L., Stepwise Regression Analysis with Orthogonal Transformations, Linkoping University Report MAT-R-1972-2, Linkoping, Sweden, 1972.

9. Golub, G., "Numerical Methods for Solving Linear Least Squares Problems", Numerische Mathematik, vol. 7, pp 206-216, 1965.

10. Gragg, W. B., Leveque, R. J., Trangenstein, J. A., Numerically Stable Methods for Updating Regressions, Journal of the American Statistical Association, vol 74, pp 161-168, March 1979.

11. Foster, V. L., "Rank and Null Space Calculations Using Matrix Decomposition Without Column Interchanges", Linear Algebra Applications, vol 74, pp 47-71, 1986.

12. Golub, H. G., Klema, V., Stewart, W. G., Rank Degeneracy And Least Squares Problems, Technical Report STAN-CS-76-559, Computer Science Department, Stanford University, 1976.

13. Chan, F. T., "Rank Revealing QR Factorizations", Linear Algebra and its Applications, vol 88/89, pp 67-82, 1987.

14. Karasalo, I., "A Criterion For Truncation of the QR Decomposition Algorithm for the Singular Linear Least Squares Problem", BIT, vol 14, pp 156-166, 1974.

15. Chan, T. F., Hansen, P. C., "Some Applications of the Rank Revealing QR Factorization", CAM Report 90-09, Department of Mathematics, UCLA.

16. Hager, W. W., Applied Numerical Linear Algebra, Prentice Hall, 1988.

17. Chan, T. F., "On the Existence and Computation of LU-Factorizations with Small Pivots", Mathematics of Computation, vol. 42, pp 738-754, August 1984.

18. Technical Support Package, Algorithm for Solution of Subset-Regression Problem, NASA Tech Briefs, ARC-12145.

19. Daniel, W. J., Gragg, W. B., Kaufman, L., Stewart, G. W., "Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization", Mathematics of Computations, vol. 30, pp 772-795, October 1976.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Dr. Dan C. Boger, Code OR/Bo Department of Operations Research Naval Postgraduate School Monterey, CA 93943-5002	2
4.	Dr. William B. Gragg, Code MA/Gr Department of Mathematics Naval Postgraduate School Monterey, CA 93943-5002	1
5.	Dr. David A. Schrady, Code OR/So Department of Operations Research Naval Postgraduate School Monterey, CA 93943-5002	1
6.	Embassy of Greece Naval Attache 2228 Massachusetts Ave., N.W. Washington, D.C. 20008	5
7.	Grigorios J. Raptis Ploutarhou 4 Salamis 18900 Pireaus, Greece	2

Thesis
R2199 Raptis
c.1 A comparison of three
numerical methods for
updating regressions.