LARGE SPARSE SETS OF LINEAR EQUATIONS

Proceedings of the Oxford conference of the Institute of Mathematics and Its Applications held in April, 1970

Edited by

J. K. REID

A.E.R.E. Harwell

LARGE SPARSE SETS OF LINEAR EQUATIONS

LARGE SPARSE SETS OF LINEAR EQUATIONS

Proceedings of the Oxford conference of the Institute of Mathematics and Its Applications held in April, 1970

Edited by

J. K. REID

A.E.R.E. Harwell

ACADEMIC PRESS · LONDON AND NEW YORK

ACADEMIC PRESS INC. (LONDON) LTD
Berkeley Square House
Berkeley Square,
London, W1X 6BA

U.S. Edition published by
ACADEMIC PRESS INC.
111 Fifth Avenue,
New York, New York 10003

ISBN: 0-12-586150-8

Library of Congress Catalog Card Number: 72-141736

Printed in Great Britain by
ROYSTAN PRINTERS LTD.,
Spencer Court, 7 Chalcot Road,
London, N.W.1.

Contributors

R. J. ALLWOOD, *Department of Civil Engineering, University of Technology, Loughborough, England.*

V. ASHKENAZI, *Department of Civil Engineering, The University of Nottingham, Nottingham, England.*

J. P. BATY, *Department of Civil Engineering and Building Technology, The University of Wales Institute of Science and Technology, Cardiff, Wales.*

R. BAUMANN, *Mathematisches Institut der Technischen Hochschule, München, W. Germany.*

E. M. L. BEALE, *Scientific Control Systems, London, England.*

B. A. CARRÉ, *Department of Electrical Engineering, The University, Southampton, England.*

M. E. CHURCHILL, *Computing Branch, Central Electricity Generating Board, London, England.*

J. DE BUCHET, *S.E.M.A., Paris, France.*

F. HARARY, *Department of Mathematics, University of Michigan, Michigan, U.S.A.*

A. JENNINGS, *Department of Civil Engineering, The Queen's University of Belfast, Belfast, Northern Ireland.*

M. H. E. LARCOMBE, *School of Engineering Science, Structural Computation Unit, University of Warwick, Coventry, England.*

E. C. OGBUOBIRI, *United States Department of the Interior, Bonneville Power Administration, Portland, U.S.A.*

J. K. REID, *Atomic Energy Research Establishment, Harwell, England.*

K. L. STEWART, *Department of Mathematics and Statistics, The Hatfield Polytechnic Hatfield, Herts., England.*

R. P. TEWARSON, *Department of Applied Mathematics, State University of New York, Stony Brook, New York, U.S.A.*

A. D. TUFF, *Department of Civil Engineering, The Queen's University of Belfast, Belfast, Northern Ireland.*

J. WALSH, *Department of Mathematics, University of Manchester, Manchester, England.*

R. A. WILLOUGHBY, *I.B.M., Thomas J. Watson Research Center, New York, U.S.A.*

K. ZOLLENKOPF, *Hamburgische Electricitäts-Werke, Hamburg, W. Germany.*

Preface

This book contains the papers presented at a conference with the same title which was held at St. Catherine's College, Oxford from 5th to 8th April, 1970. This was organised by the Institute of Mathematics and its Applications, following an initial suggestion of W. G. Sherman and H. H. Robertson. The detailed organisation was performed by a committee under the Chairmanship of E. T. Goodwin. The aim of the conference was to bring together original workers in this developing subject and also practitioners in the many different fields in which it is applied. It was hoped that the papers would be biased away from systems arising from the solution of partial differential equations and towards systems from other applications and having a less well-ordered structure in order to counterbalance the bias of most of the published literature.

The papers are included in the order that they were presented at Oxford with the exception of that of E. C. Ogbuobiri, who unfortunately was unable to attend. E. M. L. Beale, R. Allwood and R. Baumann were invited to give introductory talks summarizing the way sparse systems of equations arise in their respective fields; J. Walsh, F. Harary and R. P. Tewarson were invited to speak on three particular aspects of the sparse matrix problem; the remaining papers were selected from those submitted.

About ten minutes were allowed for discussion following each talk. Anyone who contributed to this discussion was invited to hand in a written version for publication and I am very grateful to those who did this. Most of the comments that appear in print, however, are shortened versions constructed by me from notes I took at the time. Typescripts were made available on the day following each talk so that the contributors could correct the comments I had attributed to them. I omitted only those parts of the discussion which I felt brought out no points of interest and I took considerable "poetic licence" in reordering and rewording the contributions.

One of the difficulties in the past has been that expertise has been developed for particular applications and has been left buried in computer programs. I think that this conference has gone some way towards correcting this. We heard about current practice in the fields of linear programming, structural analysis, surveying, power system analysis and network flow. We also heard from several numerical analysts working on the sparse matrix problem with-

out having any particular field in mind. Workers in each of these categories gained by learning of the practice of others, and it is to be hoped that readers of this book will find it similarly useful and will not restrict their attention to those chapters which obviously discuss their own particular interest.

The really large problem places particular demands on both the hardware and software of a computer. Readers who are interested in this aspect should pay particular attention to the paper of Willoughby. It is also considered by Allwood and some interesting comments appear in the discussion following de Buchet's talk.

I should like to thank the secretarial staff of the Institute of Mathematics and its Applications for their work during the evenings of the conference in typing drafts of the discussions, and Mrs. M. Johnson for her help with the numerous typing jobs that were necessary during the preparation of the book.

J. K. REID

A.E.R.E., Harwell
January 1971

Contents

Sparseness in Linear Programming

E. M. L. BEALE

(Scientific Control Systems Ltd.)

Summary

This expository paper reviews some aspects of linear programming from the point of view of manipulating sparse matrices. Some typical linear programming formulations are described in algebraic terms, to indicate how and why large sparse matrices arise in practical problems. The standard simplex method used for solving linear programming problems is then outlined, both in its original form and using the product form of the inverse matrix method. The last section of the paper indicates the main ideas used to find an accurate and compact product form representation of the inverse of an arbitrary sparse matrix. In particular it shows how the ideas of triangular decomposition can improve this process.

1. Introduction

Linear programming is widely used for economic planning in industry, particularly the process industries. It is also being used increasingly in other contexts, such as agriculture and defence, to throw light on the best allocation of scarce resources.

This paper consists of an exposition of some aspects of linear programming from the point of view of manipulating sparse matrices. Section 2 indicates how these matrices arise in typical practical problems. The mechanics of the standard method of solving linear programming problems, Dantzig's Simplex Method, are described in section 3. This section also introduces the Product Form of the Inverse Matrix Method, which enables sparseness to be exploited much more fully than when using the original simplex method. Section 4 is devoted to methods of finding an accurate and compact product form representation of the inverse of an arbitrary sparse matrix. This topic is of great importance in linear programming ,and also in other contexts. It will be discussed more fully by De Buchet [2].

In order to avoid attempting to cover too wide a field within a single paper, extensions of linear programming are not discussed here in any detail. But it is perhaps worth pointing out that one reason for the intense research and

development activity in methods for exploiting the sparseness of linear programming matrices over the past year or two has been the growing realisation that many large integer programming problems can be solved effectively by branch and bound methods, but that the solution of one integer programming problem in this way requires the solution of a (possibly long) sequence of linear programming problems. Actually these particular linear programming problems are best solved by a variant of the simplex method known as parametric programming, but the same sparse matrix techniques are used.

2. Linear Programming Formulations

The basic problem of linear programming is to choose *nonnegative* values of variables x_j $(j = 1, \ldots n)$ to minimize some linear *objective function* $\Sigma_j c_j x_j$ subject to m linear equality constraints

$$\Sigma_j a_{ij} x_j = b_i \qquad (i = 1, \ldots m).$$

Note that this formulation allows inequality constraints, since we can turn the inequality

$$\Sigma_j a_{ij} x_j \leqslant b_i$$

into an equation by adding the nonnegative *slack variable* s_i and writing

$$\Sigma_j a_{ij} x_j + s_i = b_i.$$

We can also maximize the objective function by changing signs. In practice the problem is usually input to the computer as if it were written

$$\left.\begin{aligned} x_0 + \Sigma_j a_{0j} x_j &= b_0 \\ \Sigma_j a_{ij} x_j &= b_i \qquad (i = 1, \ldots m) \end{aligned}\right\} \tag{2.1}$$

where the objective is to maximize the dummy variable x_0. Positive coefficients a_{0j} then represent costs and negative a_{0j} represent revenues.

In general terms, the variables x_j represent *levels of activities*, for example the rate of production for some product, and the constraints represent resources; for example b_i might be the total amount of available labour and a_{ij} amount of labour required per unit of activity x_j. But in order to understand how and why real linear programming problems produce large sparse matrices we must look at formulations in more detail. It is also appropriate to use somewhat different notation, since one common feature of most linear programming models is that a natural description requires many different subscripts for both the variables and constants. It is therefore convenient to start by defining the subscripts, then the constants and variables, and finally

the constraints and objective function. It is obviously very important to distinguish clearly between variables whose optimum values are to be determined by the model and constants whose values are assumed. A useful trick is therefore to use capital letters for constants and small letters for variables.

A typical production planning model may involve three subscripts

i for materials

j for resources (machine capacities etc.)

and k for production activities.

The constants may then be

A_{ik} the amount of material i produced per unit level of activity k, where A_{ik} is negative if the material is an input used in the activity.

B_{jk} the amount of resource j required per unit level of activity k

C_k the operating cost per unit level of activity k

P_{Bi} the cost of material i, if bought from outside

P_{Si} the revenue obtained per unit of material i sold

Q_i the stock of material i available for purchase

R_j the available amount of resource j

$S_{Li}\, S_{Ui}$ the lower and upper limits on the amount of material i that can be sold.

The variables may then be

b_i the amount of material i bought (defined only for those i for which P_{Bi} is defined)

s_i the amount of material i sold (defined only for those i for which P_{Si} is defined)

x_k the level of production activity k.

The problem is then to maximize

$$\Sigma_i P_{Si} s_i - \Sigma_i P_{Bi} b_i - \Sigma_k C_k x_k$$

subject to the material balance constraints

$$\Sigma_k A_{ik} x_k + b_i - s_i = 0 \text{ for all } i,$$

the capacity constraints

$$\Sigma_k B_{jk} x_k \leqslant R_j \text{ for all } j,$$

the availability constraints

$$b_i \leqslant Q_i \text{ for all } i \text{ where } 0 \leqslant Q_i < \infty$$

and demand constraints of the form

$$S_{Li} \leqslant s_i \leqslant S_{Ui} \text{ for all } i,$$

together with the usual nonnegativity constraints

$$x_k \geqslant 0 \text{ for all } k,$$

and

$$b_i \geqslant 0 \text{ for all } i.$$

A model of this kind is often relatively small. It may contain up to about 100 constraints and about 2 or 3 times as many variables, including slack variables. It may not be very sparse, particularly if it is small; but on the other hand the variables b_i and s_i have few nonzero coefficients and many of the A_{ik} and B_{jk} may also vanish.

The model becomes both larger and sparser if one studies serveral different plants in the same model. This may be appropriate if one is concerned with deciding which products to make where, to produce the best balance between production and distribution costs, or which is the best way to allocate raw materials available in limited quantities between plants. The model may then have two additional subscripts

l for location of plants

and m for markets.

We may then need to consider extra constants

$C_{Tii_1i_2}$, the cost of transporting one unit of material i from location l_1 to location l_2, and there will be corresponding variables.

$y_{ii_1i_2}$, the amount of material i transported from location l_1 to location l_2.

The problem is then to maximize

$$\Sigma_i \Sigma_l \Sigma_m P_{Siim} s_{iim} - \Sigma_i \Sigma_l P_{Bil} b_{il} - \Sigma_k \Sigma_l C_{kl} x_{kl} - \Sigma_i \Sigma_{l_1} \Sigma_{l_2} C_{Tii_1i_2} y_{ii_1i_2},$$

subject to the material balance constraints

$$\Sigma_k A_{ikl} x_{kl} + b_{il} - \Sigma_m s_{iim} + \Sigma_{l_1} y_{ii_1l} - \Sigma_{l_2} y_{iii_2} = 0, \text{ for all } i \text{ and } l,$$

the capacity constraints

$$\Sigma_k B_{jkl} x_{kl} \leqslant R_{jl}, \text{ for all } j \text{ and } l,$$

the availability constraints

$$\Sigma_l b_{il} \leqslant Q_i, \text{ for all } i$$

and demand constraints of the form

$$S_{Lim} \leqslant \Sigma_l s_{ilm} \leqslant S_{Uim} \text{ for all } i \text{ and } m$$

together with the usual nonnegativity constraints

$$x_{kl} \geqslant 0,\ b_{il} \geqslant 0,\quad s_{ilm} \geqslant 0,\quad y_{il_1l_2} \geqslant 0.$$

Such a model may contain a few hundred constraints and a larger number of variables, but the average number of nonzero coefficients per column (i.e. per variable) may still be not more than about 6.

The model becomes even larger, but often more valuable, when time is considered explicitly. In an industry where seasonal factors are important one must compromise between minimizing storage costs, by making the production pattern follow the forecast demand as closely as possible, and minimizing production costs, by producing at an even rate throughout the year or possibly by producing most when raw materials are cheapest.

The formulation of multi-time period models involves an extra subscript t for the time period, which is added to the variables and constraints, thus increasing the size of the problem very considerably. One also adds storage activities, which are essentially the same as the transport activities $y_{il_1l_2}$ except that storage transports material from one time period to the next instead of from one location to another in the same time period.

Other multi-time period models arise in long-term studies of capital investments. The time periods are then usually years.

A multi-time period model may contain well over 1000 constraints. Some contain over 2000 constraints, but the density of these large problems is almost always well under 1%.

3. The Simplex Method

In principle, the simplex method involves taking the equations (2.1) and solving for x_0 and m of the other variables, which we call basic variables and denote $X_1 \ldots X_m$. The resulting equations can then be written in the form

$$\left.\begin{aligned} x_0 &= \bar{a}_{00} + \Sigma_j \bar{a}_{0j}(-x_j) \\ X_i &= \bar{a}_{i0} + \Sigma_j \bar{a}_{ij}(-x_j), \qquad i = 1, 2, \ldots m \end{aligned}\right\} \tag{3.1}$$

where the summation on the right-hand side extends only over the *nonbasic variables*, i.e. those that do not occur on the left-hand side. Corresponding to this formulation we have a trial solution to the problem, obtained by setting all the nonbasic variables equal to zero, so that $x_0 = \bar{a}_{00}$ and the basic variables X_i take the values $\bar{a}_{i0}$. The array of coefficients in (3.1) is known as a *tableau.*

If all $\bar{a}_{i0}$ are non-negative then the trial solution is *feasible*, since no variable takes an impossible value. If all the $\bar{a}_{0j}$ are also nonnegative, then x_0 cannot be made larger than $\bar{a}_{00}$ without making some nonbasic variable x_j negative, so the trial solution is optimal.

If the trial solution is feasible but not optimal, then we may find some negative coefficient $\bar{a}_{0q}$. This indicates that x_0 can be increased by increasing the variable x_q. If x_q can be increased indefinitely without driving any of the basic variables X_i negative, i.e. if all the coefficients $\bar{a}_{iq} \leqslant 0$, then the problem has an unbounded solution. But otherwise we must stop increasing x_q when some basic variable, say X_p drops to zero. The formula for finding X_p is

$$\left(\frac{\bar{a}_{p0}}{\bar{a}_{pq}}\right) = \min\left(\frac{\bar{a}_{i0}}{\bar{a}_{iq}}\right),$$

where the minimum is taken over those i for which $\bar{a}_{iq} > 0$. Having found such a variable, we can use the equation for X_p to solve for x_q in terms of the other variables, and can then substitute for x_q throughout (3.1). We will then have a new tableau of the form (3.1) but with an improved trial solution. The whole process can then be repeated until an optimal solution is found.

The operation of solving for x_q and substituting for it in terms of the other variables is known as a *pivoting operation*. We say that we are pivoting in the variable x_q instead of X_p.

If the trial solution is not feasible, then one can find a new variable x_q to be pivoted in to reduce the extent of the infeasibility, i.e. the sum of the magnitudes of the values of all variables that are negative. The details can be found in textbooks on linear programming, such as Beale [1].

This in outline is the original simplex method. In practice, the logic of the simplex method is carried out without computing the full tableau (3.1), since one can take better advantage of the sparseness of the original set of equations (2.1). We may write (2.1) as a matrix equation

$$\mathbf{Ax} = \mathbf{b}. \tag{3.2}$$

We then define $\mathbf{B}$ as the square submatrix of $\mathbf{A}$ formed by taking the columns of coefficients of the basic variables $x_0, X_1, \ldots X_m$. If we now premultiply both sides of (3.2) by $\mathbf{B}^{-1}$, we obtain the equation

$$\mathbf{B}^{-1}\mathbf{Ax} = \boldsymbol{\beta}, \tag{3.3}$$

where

$$\boldsymbol{\beta} = \mathbf{B}^{-1}\mathbf{b}. \tag{3.4}$$

Now (3.3) in effect expresses the basic variables $x_0, X_1, \ldots X_m$ as linear functions of the nonbasic variables. It must therefore be equivalent to (3.1) and

we see that the components of $\boldsymbol{\beta}$ are the $(\bar{a}_{i0})$ of (3.1), while the remaining coefficients in the tableau (3.1) are the elements of the product $\mathbf{B}^{-1}\mathbf{A}$.

The simplex method does not require the computation of the entire matrix $\mathbf{B}^{-1}\mathbf{A}$. We need to be able to pick out the top row, the coefficients $\bar{a}_{0j}$, but these can be found by forming the inner products of the top row of $\mathbf{B}^{-1}$ with the columns of $\mathbf{A}$. And having selected a column $\mathbf{a}_q$ of $\mathbf{A}$ representing the coefficients of x_q, the variable to be pivoted in, we need to form the $(\bar{a}_{iq})$ as the components of $\mathbf{B}^{-1}\mathbf{a}_q$.

Since the original matrix $\mathbf{A}$ is generally much sparser than $\mathbf{B}^{-1}\mathbf{A}$, it is advantageous to work with some representation of $\mathbf{B}^{-1}$ and the original matrix $\mathbf{A}$ rather than with the tableau. This requires that

(a) we have a compact representation of $\mathbf{B}^{-1}$, and

(b) we can update $\mathbf{B}^{-1}$ from one iteration to the next without doing a complete matrix inversion.

Let us first consider point (b). After each iteration the new matrix $\mathbf{B}$ differs from the previous matrix only in a single column. We therefore consider the effect of this on the inverse. To see that the inverses must be simply related, consider the algebraic equations

$$\left.\begin{aligned} x_1 &= b_{11}y_1 + b_{12}y_2 + \ldots + b_{1m}y_m + b_{1m+1}y_{m+1} \\ x_2 &= b_{21}y_1 + b_{22}y_2 + \ldots + b_{2m}y_m + b_{2m+1}y_{m+1} \\ &\cdots \\ x_m &= b_{m1}y_1 + b_{m2}y_2 + \ldots + b_{mm}y_m + b_{mm+1}y_{m+1}. \end{aligned}\right\} \tag{3.5}$$

To form the inverse of the matrix

$$\mathbf{B} = \begin{pmatrix} b_{11} \ldots b_{1m} \\ \vdots \quad\; \vdots \\ b_{m1} \ldots b_{mm} \end{pmatrix},$$

we can perform m pivot steps on the system (3.5) in turn, in each case solving for y_i in terms of x_i and the remaining variables and then substituting for y_i in the other equations. We will then have transformed (3.5) into a system of the form

$$\left.\begin{aligned} y_1 &= \bar{b}_{11}x_1 + \bar{b}_{12}x_2 + \ldots + \bar{b}_{1m}x_m + \bar{b}_{1,m+1}y_{m+1} \\ y_2 &= \bar{b}_{21}x_1 + \bar{b}_{22}x_2 + \ldots + \bar{b}_{2m}x_m + \bar{b}_{2,m+1}y_{m+1} \\ &\cdots \\ y_m &= \bar{b}_{m1}x_1 + \bar{b}_{m2}x_2 + \ldots + \bar{b}_{mm}x_m + \bar{b}_{m,m+1}y_{m+1}. \end{aligned}\right\} \tag{3.6}$$

The matrix

$$\mathbf{B}^{-1} \text{ is then given by } \begin{pmatrix} \bar{b}_{11} \dots \bar{b}_{1m} \\ \vdots \quad\quad \vdots \\ \bar{b}_{m1} \dots \bar{b}_{mm} \end{pmatrix},$$

as can be seen by putting $y_{m+1} = 0$ in (3.5) and (3.6), since (3.5) is then the matrix equation

$$\mathbf{x} = \mathbf{B}\mathbf{y}$$

and (3.6) becomes

$$\mathbf{y} = \mathbf{B}^{-1}\mathbf{x}.$$

But now suppose we want to replace some column, say the second, of $\mathbf{B}$ by the coefficients of y_{m+1} in (3.5). Then we must express $y_1, y_{m+1}, y_3, \dots y_m$ in terms of $x_1 \dots x_m$ with $y_2 = 0$. But, given (3.6), this can be achieved very simply by solving the second equation for y_{m+1} in terms of $y_2, x_1, x_2, \dots x_m$ and then substituting for y_{m+1} throughout (3.6).

Now in practice one might not have the coefficients $(\bar{b}_{i,m+1})$ explicitly available. But we see that these are defined by

$$\mathbf{B}^{-1}\,\mathbf{b}_{m+1},$$

where $\mathbf{B}^{-1}$ denotes the current inverse $\begin{pmatrix} \bar{b}_{11} \dots \bar{b}_{1m} \\ \vdots \quad\quad \vdots \\ \bar{b}_{m1} \dots \bar{b}_{mm} \end{pmatrix}$

and $\mathbf{b}_{m+1}$ denotes the column of coefficients of y_{m+1} in (3.5). So the whole process of revising the inverse consists of one matrix times vector multiplication to find the $\bar{b}_{i,m+1}$ followed by a single pivoting operation. In the linear programming application, the matrix times vector multiplication must be done anyway, to determine the coefficients $(\bar{a}_{iq})$ before selecting the variable X_p, so we only need consider the pivoting operation.

Now each pivoting operation can be represented as premultiplying B^{-1} by an *elementary transformation matrix* $\mathbf{T}$ which is a unit matrix except that its pth column (if the columns are numbered $0, 1, \dots m$) is replaced by $-\bar{a}_{iq}/\bar{a}_{pq}$ in its off-diagonal elements and $1/\bar{a}_{pq}$ in its diagonal element.

This leads us to consider the *product form representation* of the inverse of a sparse matrix, due to Dantzig and Orchard-Hays [4]. This implies that we start with a unit matrix and write down a sequence of elementary transformation matrices representing the effects on $\mathbf{B}^{-1}$ of replacing each column of the original unit matrix by a column of the actual required $\mathbf{B}$ in turn. So $\mathbf{B}^{-1}$ is then written in the form

$$\mathbf{B}^{-1} = \mathbf{T}_r \mathbf{T}_{r-1} \dots \mathbf{T}_3 \mathbf{T}_2 \mathbf{T}_1, \tag{3.7}$$

where r may equal m, or it may be smaller if there are some unit vectors in the final matrix **B**. At first sight, this may seem to be a very expanded way in which to express $\mathbf{B}^{-1}$, but this is not so. Each elementary transformation matrix can be defined in the computer by one index defining the non-unit column and one row-index with an associated numerical value defining the nonzero elements in this column. The other columns play such a shadowy existence that these matrices are often referred to as vectors, specifically *η-vectors.*

To form any row, or any linear combination of rows, of $\mathbf{B}^{-1}$ we define a row-vector **c** giving the weights to be attached to the rows, and form the product

$$\mathbf{c}\,\mathbf{T}_r\,\mathbf{T}_{r-1} \ldots \mathbf{T}_3\,\mathbf{T}_2\,\mathbf{T}_1$$

starting from the left. We therefore have a sequence of r vector by matrix multiplications, but because of the nature of the **T** matrices each multiplication never alters more than one element in the row vector. This process is known as a *Backward Transformation* because the elementary transformations are used in the opposite order to that in which they were generated.

To form any column of the current tableau $\mathbf{B}^{-1}\mathbf{A}$ we form the product

$$\mathbf{T}_r\,\mathbf{T}_{r-1} \ldots \mathbf{T}_3\,\mathbf{T}_2\,\mathbf{T}_1\,\mathbf{a}$$

starting from the right. We therefore have a sequence of r simple matrix by vector multiplications. This process is known as a *Forward Transformation* because the elementary transformations are used in the same order as that in which they were generated.

After each pivot operation, the representation of $\mathbf{B}^{-1}$ is updated by adding another elementary transformation to the end of the list. This is a relatively quick and painless operation, but it makes subsequent backward and forward transformation a little slower and possibly less accurate. So after a while one stops iterating and throws away the list of elementary transformations, retaining only the sequence numbers (or names) of the variables whose columns form the current basis **B**. We then form a new list of elementary transformations, representing the effects of replacing the columns of an initial unit matrix by the columns of the current **B** in some order. Techniques for choosing the sequence of pivot operations during this inversion process are vital to a good linear programming code, and have been developed extensively during the last few years. Some of the main features are described in the next section, and in other papers presented at this conference.

Incidentally, the need to invert from time to time to speed up the solution process provides a good opportunity to control the build-up of rounding error in the current values of the basic variables, defined by the vector $\boldsymbol{\beta}$ in (3.3) and (3.4). At each interation the new $\boldsymbol{\beta}$ is found by premultiplying the previous