Steepest Edge, Degeneracy and Conditioning in LP

by

R. Fletcher*†

Numerical Analysis Report NA/154, October 1994.

Abstract

Steepest edge criteria have often been shown to work well in Simplex or Active Set methods for Linear Programming (LP). It is therefore important that any technique for resolving degeneracy in LP should conveniently be able to use steepest edge criteria in the selection of the search direction.

A recursive technique for resolving degeneracy [5] is of some interest in that it provides a guarantee of termination, even in the presence of round-off errors. However it works in both primal and dual spaces and would need steepest edge coefficients in both spaces to take full advantage of steepest edge criteria.

A new method of resolving degeneracy is described which provides a similar guarantee of termination and is readily implemented. The method works only in the primal space so that only one set of steepest edge coefficients needs to be maintained.

It is also shown that the steepest edge coefficients provide information from which the expected conditioning of the current LP basis can be calculated cheaply. This provides a more accurate indication of the actual conditioning of a system than is obtained from norm-based condition numbers. A table of condition numbers for the SOL test set is presented and has some interesting implications for the solvability of LP problems when round-off error is present. This idea also enables the conditioning of null space matrices to be estimated.

Keywords. Steepest Edge Method, Degeneracy, Linear Programming, Conditioning, Simplex Method, Active Set Method, Null Space.

* Dept. of Mathematics and Computer Science, Univ. of Dundee, Dundee DD1 4HN.

† Presented at the 15th International Symposium on Mathematical Programming, Ann Arbor, August 15-19, 1994.

1. Introduction

Steepest edge methods were introduced by Goldfarb and Reid [9] as a means of improving the performance of the primal Simplex Method for Linear Programming (LP). Similar ideas are applicable when solving dual LP problems (Forrest and Goldfarb [8]) and to the related situation of solving an LP problem by an Active Set Method (ASM). A steepest edge method incurs an additional cost per iteration to update certain coefficients which are used to normalize edges of the feasible region polytope. Experimental evidence generally indicates that this cost is more than outweighed by an improvement in the number of iterations required.

Degeneracy causes difficulty to Simplex or Active Set Methods when it blocks progress in the current iteration. Often the problem is transformed so that application of a Simplex-like method can be used to remove the blockage. In some cases a considerable number of iterations are required to do this. It is therefore beneficial if advantage can be taken of the steepest edge approach during this procedure.

This author has been interested in a particular primal-dual method [5] for resolving degeneracy. A feature of this method is that it permits the use of heuristics which enable termination to be guaranteed in the presence of arbitrary round-off errors. This property is achieved as a consequence of the recursive solution of *feasibility problems* and the reasons for this are reviewed in Section 2. However as the method works in both primal and dual spaces, it would need steepest edge coefficients in both spaces to take full advantage of the steepest edge approach.

The main aim of this paper is to describe a new method for resolving degeneracy which is also based on the recursive solution of feasibility problems, and so provides a similar guarantee of termination. The method works in the primal space throughout, so that only primal steepest edge coefficients need to be maintained. The method can be extended to solve quadratic programming problems by following the idea described in [6].

Given that steepest edge coefficients are maintained by the algorithm, it is worthwhile to consider if they can be used in any other heuristics in an LP code. Difficulties often arise when the basis matrix in a Simplex-like method becomes nearly singular. It is useful to be able to assess the conditioning of this matrix, so that a *basis repair* technique can be initiated when necessary. It is important that estimates of conditioning are not too pessimistic, for otherwise basis repair would be invoked too frequently. This paper shows that steepest edge coefficients enable a realistic estimate to be made of the conditioning of systems of equations that involve the basis matrix. The amount of additional computation is small, particularly for a sparse LP problem. The idea also permits the conditioning of a null space matrix to be estimated, with application to reduced Hessian methods for quadratic programming and nonlinear programming.

Much of the paper is written from an ASM point of view, but the ideas are also applicable to the Simplex method. The idea for resolving degeneracy is initially presented in Section 3 using a standard Simplex tableau, and an illustrative example is given. Comparisons with Wolfe's method [12] are made, and reasons for preferring the new method are advanced. The new method is described in an ASM context in Section 4 and details of implementation are discussed. The idea of *expected conditioning* is reviewed in Section 5 and it is shown how the availability of steepest edge coefficients makes this idea practicable.

The extension to null space matrices is described in Section 6.

The general approach to solving an LP by an ASM has been outlined for example in [4]. The LP problem is formulated as

$$\begin{array}{lll} \text{minimize} & f\ (= c^T x) & x \in \mathbb{R}^n \\ \text{subject to} & a_i^T x \geq b & i = 1, 2, ..., m \end{array} \tag{1.1}$$

where $m \geq n$ and $a_i \in \mathbb{R}^n$ is the normal vector of the i^{th} constraint. The *active set* denoted by $\mathcal{A}$ is a selection of n constraint indices such that the matrix $A_{\mathcal{A}}$ whose columns are the normal vectors $a_i,\ i \in \mathcal{A}$, is nonsingular. The current point $\hat{x}$ is a vertex of the feasible region, obtained by solving the equations $a_i^T x = b_i,\ i \in \mathcal{A}$.

If $\hat{x}$ is nondegenerate in the sense that $a_i^T \hat{x} > b_i$ for all $i \notin \mathcal{A}$, there are n feasible edges emanating from $\hat{x}$, denoted by $d_i,\ i = 1, 2, ..., n$. It is readily shown that these vectors are the columns of the matrix $A_{\mathcal{A}}^{-T}$. The slope of f along any edge d_i is given by $\lambda_i = c^T d_i$, which can be interpreted as a Lagrange multiplier. The current vertex $\hat{x}$ is optimal if $\lambda_i \geq 0,\ i = 1, 2, ..., n$. Otherwise a decision is made as to which edge d_p to choose as the search direction. In the traditional approach (that adopted by Dantzig [2] in the Simplex method) the choice

$$p = \arg\min_i \lambda_i \tag{1.2}$$

is made, with the hope that the most negative slope will provide a significant reduction in f. However this thinking neglects the fact that the lengths of the d_i may differ. Therefore in the steepest edge method the edges are normalized, giving rise to the choice

$$p = \arg\min_i \frac{\lambda_i}{\|d_i\|_2}. \tag{1.3}$$

This choice requires that the *steepest edge coefficients*

$$\gamma_i = d_i^T d_i \quad i = 1, 2, ..., n \tag{1.4}$$

are available. The main contribution of Goldfarb and Reid [9] lies in pointing out that when column p in the matrix $A_{\mathcal{A}}$ is replaced by another column, a_q say, then it is also possible to update the coefficients γ_i. The appropriate formula is

$$\gamma_i^{\#} = \gamma_i - 2\frac{t_i}{t_p} d_i^T d_p + \left(\frac{t_i}{t_p}\right)^2 \gamma_p \qquad i = 1, 2, ..., n, \quad i \neq p \tag{1.5}$$

where $^{\#}$ denotes successor and $t = A_{\mathcal{A}}^{-1} a_q$. The vector t may be available when updating an invertible representation of $A_{\mathcal{A}}$. Equation (1.5) needs the values $d_i^T d_p$ which can be written as $[A_{\mathcal{A}}^{-1} d_p]_i$ by definition of d_i. These values are not available and require an extra solve with the matrix $A_{\mathcal{A}}$. It is also possible to write down a formula derived from (1.5) in which the reciprocals of the γ_i are updated. I have used this alternative in the hope of avoiding cancellation when $A_{\mathcal{A}}$ is ill-conditioned and the γ_i are large.

2. Resolving Degeneracy by Solving Feasibility Problems

Many methods have been suggested for resolving degeneracy in linear programming, but I have been interested in a particular primal-dual method [5] which is also related to a method of Balinski and Gomory [1]. A feature of this method is that it permits the use of heuristics which guarantee that the method terminates with stored values which indicate a solution (or infeasibility/unboundedness) *even when the arithmetic is inexact.* This property is achieved as a consequence of the recursive solution of *feasibility problems* and the reasons for this are reviewed in this section. An alternative way of generating a feasibility problem is then proposed which leads to a new method for resolving degeneracy.

First of all, the problem of solving a system of linear inequalities (Phase I of LP) is considered. In an ASM context this can be expressed as finding a feasible point x of the system

$$r \equiv A^T x - b \geq 0, \tag{2.1}$$

where $x \in \mathbb{R}^n$ and $r \in \mathbb{R}^m$ with $m \geq n$. This can equivalently be expressed as

$$r_i \equiv a_i^T x - b_i \geq 0 \quad i = 1, 2, ..., m \tag{2.2}$$

where a_i denotes a column of A. The notation that $\hat{r}_i$ denotes $a_i^T \hat{x} - b_i$ is used. Various methods are possible for solving (2.1) and my practical experience does not indicate any strong preferences. However in the context of this paper, one method is particularly useful and might be described as the *One-at-a-time method.* If $\hat{x}$ is a vertex (in the sense of the intersection of the n constraints in the active set), and is infeasible, then an index q such that $\hat{r}_q < 0$ is chosen. The ASM is then applied to drive r_q up to zero whilst maintaining feasibility in any constraints that are feasible at the current vertex $\hat{x}$. Thus each step of the method is that obtained by applying the ASM to the LP

$$\begin{array}{lll} \text{maximize} & r_q \quad (\equiv a_q^T x - b_q) & x \in \mathbb{R}^n \\ \text{subject to} & a_i^T x \geq b_i \quad i \in \hat{\mathcal{F}} = \{i : \hat{r}_i \geq 0\}, & \end{array} \tag{2.3}$$

for which $\hat{x}$ is a feasible vertex. If $\hat{r}_q < 0$ and the Lagrange multipliers indicate that $\hat{x}$ is optimal in (2.3), so that no further increase in r_q is possible, then the system (2.1) has no feasible point. Otherwise, if there is no degeneracy, the ASM iteration can increase r_q. Eventually r_q becomes zero and q can be included in $\hat{\mathcal{F}}$. This process is repeated until no infeasible constraints remain.

It may be however that the ASM line search derived from (2.3) is *blocked* due to degeneracy, that is a step of length zero is taken, and no increase in r_q is obtained. Although the blocking constraint can enter the active set, it is possible that the algorithm can cycle infinitely and fail to find a solution. The method [5] avoids this in the following way. If the line search is blocked, all the inactive constraints ($\hat{r}_i > 0$) in (2.3) are ignored and the direction finding problem

$$\begin{array}{lll} \text{maximize} & a_q^T d & d \in \mathbb{R}^n \\ \text{subject to} & a_i^T d \geq 0 \quad i \in \hat{\mathcal{Z}} = \{i : \hat{r}_i = 0\} & \end{array} \tag{2.4}$$

is considered. This is also degenerate, but its dual is the system of equations and inequalities

$$\sum_{i \in \hat{\mathcal{Z}}} a_i \lambda_i + a_q = 0 \tag{2.5a}$$

$$\lambda_i \geq 0 \quad i \in \hat{\mathcal{Z}} \tag{2.5b}$$

in the variables λ_i, $i \in \hat{\mathcal{Z}}$. This system can be solved by the One-at-a-time method. There are three possible outcomes when solving this dual system

- A feasible point of the dual is obtained: in this case the primal LP (2.3) is optimal.
- Infeasibility in the dual is detected: in this case the dual costs provide a feasible descent direction d for (2.3) so that the degeneracy block has been removed (see [7] or [6] for an illustration).
- A degeneracy block occurs in the dual: in this case the dualization process is repeated.

In the first of these cases, when the primal is in Phase I (as for (2.3)) then the indication is that the primal is infeasible. If the primal is in Phase II (so that the objective function in (2.3) is to maximize $-c^T x$ where c is the vector of costs in the LP), then the current primal vertex $\hat{x}$ is an optimal solution of the LP.

The third case corresponds to the aspect of *recursion* that is characteristic of such methods (and others, e.g. Wolfe's method [12]). The base problem (2.3) is thought of as the level 1 problem and the solution of (2.5) is the level 2 problem. If a degeneracy block occurs whilst solving (2.5), a further dual is taken, giving rise to a level 3 problem, and so on. It turns out that problems at all levels share the same LP tableau, so the need for recursion does not detract from the efficiency of the ASM approach nor does it introduce significant complication in the coding, even in a language such as Fortran 77.

The main purpose of this section is to explain how the recursive solution of feasibility problems leads to a guarantee of termination, for both exact and inexact arithmetic. At any level of recursion, the One-at-a-time method exhibits a *monotonic improvement property*, that is either the number of infeasible residuals is reducing or the current objective function is strictly increasing. The occurrence of degeneracy does not contradict this observation because the solution of the higher level problem described above either provides a feasible ascent direction for the current level or indicates that the current level is optimal. Each invocation of recursion removes a constraint (q) from the problem which implies that the number of levels of recursion is finite. For exact arithmetic we also need to argue that there are a finite number of vertices at any level. The above arguments then provide the basis of a proof of termination (see [5], [6]).

For inexact arithmetic a key feature is to ensure that the *stored value* of the cost function r_q (at any level) is strictly improving. If not then the situation is treated as a degeneracy block and if necessary the current stored information is adjusted so as to be consistent with this. It is then possible, with some care, to ensure that the monotonic improvement property holds for inexact arithmetic. Because the range of floating point numbers is finite it can be deduced (without recourse to the finite number of vertices argument, which is not valid for inexact arithmetic) that termination is guaranteed.

In practice there is considerable flexibility in the choice of heuristics that can be applied to adjust the stored information when degeneracy is detected. These ideas have

been explored in [7] and [6]. Of course it is not possible with inexact arithmetic to guarantee that the solution obtained is exact, only that the final stored values are consistent with a solution. Therefore attention to small pivots, conditioning etc. to prevent loss of accuracy is still important, as in other LP codes. This method has been implemented in the LP/QP code *bqpd*, which is available on request from the author. It is felt that the existence of this guarantee of termination for inexact arithmetic, together with the use of other heuristics to prevent loss of accuracy, has contributed strongly to the robustness of the bqpd code.

A disadvantage of this method is that successive levels alternate between the primal and dual spaces of the LP, which does not enable full use of primal steepest edge coefficients to be made. A new primal method for resolving degeneracy has been devised which stays in the primal when recursion takes place and so avoids this disadvantage. The recursion is based on the solution of feasibility problems and so shares the same termination guarantees for inexact arithmetic as bqpd.

The idea is very simple and is one which has been used in other contexts. Consider the direction finding problem (2.4). This can be reposed as the problem of finding a feasible point d of the system

$$\begin{aligned} a_q^T d &= 1 \\ a_i^T d &\geq 0 \qquad i \in \hat{\mathcal{Z}}. \end{aligned} \tag{2.6}$$

The constraint $a_q^T d = 1$ requires that a strict ascent direction for r_q in (2.3) is obtained, and the remaining constraints ensure feasibility. Roughly speaking, to solve (2.6), an initial d is chosen which makes the constraint $a_q^T d = 1$ active. This introduces infeasibility into the constraints $a_i^T d \geq 0$, $i \in \hat{\mathcal{Z}}$ which is subsequently removed by the One-at-a-time method. There are three possible outcomes when solving this system

- A feasible point of the system is obtained: in this case a feasible ascent direction for (2.3) is obtained and the degeneracy block has been removed.
- It is found that (2.6) has no feasible solution: in this case it follows that $\hat{x}$ is optimal in (2.3).
- A degeneracy block occurs whilst solving (2.6): in this case the same technique can be used recursively to resolve the situation.

Each invocation of recursion brings an equality constraint $a_q^T d = 1$ into the active set and so reduces the number of free variables by one. Thus there is an upper bound on the number of levels of recursion. This and the monotonic improvement property of the One-at-a-time method enable termination to be guaranteed, for both exact and inexact arithmetic.

3. A New Primal Method for Resolving Degeneracy

In this section the idea implicit in (2.6) is applied to give a new method for resolving degeneracy in the standard LP tableau formulation of the Simplex method (e.g. Dantzig [2] or Fletcher [4], Chapter 8.2). Some properties of the new method are outlined and an illustrative example is given. The section concludes by comparing the method with Wolfe's method [12].

The Simplex formulation of LP considers a system of constraints of the form

$$Ax = b \qquad x \geq 0 \tag{3.1}$$

where $A \in \mathbb{R}^{m \times n}$ (this usage of A and b differs from that of the previous sections). The variables x are partitioned into basic variables x_B and nonbasic variables x_N, indexed by the sets B and N respectively. Similarly A is partitioned as $A = [A_B \,|\, A_N]$ where A_B is an $m \times m$ nonsingular matrix. Then (3.1) is equivalent to the system

$$x_B + \hat{A} x_N = \hat{x}_B, \qquad x_B \geq 0, \quad x_N \geq 0 \tag{3.2}$$

where $\hat{A} = A_B^{-1} A_N$. The current value of x is denoted by $\hat{x}$, with $\hat{x}_N = 0$ and $\hat{x}_B = A_B^{-1} b$. By convention, this system is represented by the *tableau*

$$\begin{array}{c} \\ x_B \end{array} \begin{array}{c} \begin{array}{c} x_N \end{array} \\ \left[\begin{array}{c|c} \hat{A} & \hat{x}_B \end{array}\right] \end{array}, \tag{3.3}$$

and it is convenient to refer to rows and columns of the tableau by the variables to which they correspond. The coefficients in any column (q say) of $\hat{A}$ express the rate of change of $-x_B$ with respect to changes in x_q (this follows from (3.2)). If a variable $q \in B$ is interchanged with a varible $p \in N$, there are standard rules for updating the entries in the tableau. This is referred to as *pivoting* on $\hat{a}_{qp}$. In practice the tableau is not stored explicitly, but relevant rows and columns are generated as necessary. However the concept of a tableau is often invaluable for explaining how a method works.

Clearly $\hat{x}$ is feasible in (3.2) (and hence (3.1)) if $\hat{x}_B \geq 0$. Otherwise (Phase I) feasibility can be restored by the One-at-a-time method as follows. Let $\hat{x}_q < 0$, $q \in B$. If the entries in row q of $\hat{A}$ are nonnegative then no increase in x_q is possible and (3.1) has no solution. Otherwise some $p \in N$ is chosen such that $\hat{a}_{qp} < 0$. Then (in absence of degeneracy) x_q can be increased by increasing x_p. A ratio test determines the amount of increase that is possible. If $x_q \nearrow 0$ without introducing any new infeasibility in $\hat{x}$, then a pivot can be taken on $\hat{a}_{qp}$ and the infeasibility in x_q is removed. If the ratio test indicates that $x_{q'} \searrow 0$ for some $q' \in B$, then a pivot on $\hat{a}_{q'p}$ is made, and the attempt to increase x_q is repeated. Eventually feasibility is restored to x_q. The procedure is repeated until either no infeasibility remains or it is found that no solution exists.

Once a feasible point is located, the same method is used to minimize the objective function $f = c^T x$ (Phase II). Setting $x_0 = -f$, the equation

$$x_0 + \hat{c}^T x_N = -\hat{f} \tag{3.4}$$

is obtained by eliminating x_B from f, where $\hat{c} = c_N - A_N^T A_B^{-T} c_B$ and $\hat{f} = c_B^T \hat{x}_B$, A row corresponding to (3.4) is adjoined to the tableau and the target variable to increase is x_0. The method of the previous paragraph is applied (with $q = 0$ and allowing an arbitrary increase in x_0) either until optimality is detected or until an unbounded increase in x_0 occurs. In fact it is more satisfactory for guaranteeing termination for inexact arithmetic if the user supplies a lower limit $f_{\min}$ on f, and the code terminates when x_0 would increase to $-f_{\min}$.

If degeneracy arises (either in Phase I or Phase II) then it is convenient to denote $d_q = x_q - \hat{x}_q$ and consider a sub-tableau formed by row q of (3.3) and the degenerate rows, that is

$$\begin{array}{c} \\ d_q \\ x_D \end{array} \begin{array}{c} x_N \\ \left[\begin{array}{c|c} \hat{a}_q & 0 \\ \hline \hat{A}_D & 0 \end{array}\right] \end{array} \tag{3.5}$$

where $D = \{i : \hat{x}_i = 0\}$. If a degeneracy block has occurred then column p of $\hat{A}_D$ contains some positive entries (and $\hat{a}_{qp} < 0$ as above).

The first step in the procedure for resolving degeneracy in the sub-tableau is to pivot on the $\hat{a}_{qp}$ element. A key observation for the resulting sub-tableau is that we can increase d_q and keep x_B feasible if the d_q column in the sub-tableau is non-positive. We achieve this by solving a *level 2 feasibility problem*. Thus we temporarily overwrite the zero r.h.s. of this sub-tableau by the negative of the d_q column, and this becomes the *level 2 right hand side*. Because the d_q column contains some positive entries, the level 2 right hand side contains some negative entries which are regarded as being infeasible. The next step in the process is to try to restore feasibility to the level 2 right hand side by the One-at-a-time method. Once this level 2 problem is solved, it is possible to pivot on a negative entry in the d_q column (in row p' say), restore the r.h.s. to zero, and then replace d_q by $x_q - \hat{x}_q$ with $\hat{x}_q$ on the r.h.s.. This gives a level 1 tableau in which there is no degeneracy block and $x_{p'}$ can be increased (see the example below).

The level 2 pivots are also made in the level 1 tableau. Since the pivots are chosen from rows in the subtableau (3.5), which have zero level 1 right hand sides, these pivots do not change any of the level 1 right hand sides. Also, because the d_q column is the negative of the level 2 r.h.s., no level 2 pivots can be chosen from the d_q column (since the target variable in the One-at-a-time method is negative).

It may be that optimality is detected whilst solving the level 2 problem. In this case a pivot can be taken on the d_q element in the target row, and a level 1 problem is obtained which is also optimal.

If a degeneracy block is detected whilst the level 2 problem is being solved, then this is resolved in the same way by reference to a level 3 sub-sub-tableau, and so on. The set-up is best thought of as being derived from a single tableau, with different right hand sides being stored for the sub-tableau at each level of recursion. Pivots at level L are chosen from the level L sub-tableau but the pivots are made throughout the level 1 tableau. In general, pivots are not made in columns corresponding to variables d_q that have become nonbasic at lower levels of recursion.

In fact there is a more efficient way of implementing the process. Once the level 2 problem has been solved, a return to level 1 can be made whilst d_q is nonbasic. Then d_q becomes the variable to increase at level 1. If d_q is driven up to $-\hat{x}_q$ then d_q can be replaced by x_q as the nonbasic variable, and the level 1 r.h.s. is updated, but no pivot in the tableau is required. In this case two pivot steps are saved. If a pivot in a nondegenerate level 1 row is made (row p' say), then a pivot on $\hat{a}_{p'q}$ is made, d_q becomes basic and the level 1 r.h.s. is updated. Also d_q is replaced by x_q and the old value of $\hat{x}_q$ is added to the r.h.s.. In this case one pivot is saved.

There is also another simplification that can be made. It may be whilst solving the level L problem ($L > 1$) that optimality in some row (p'' say) is detected. If this happens a pivot can be made on element $\hat{a}_{p''q}$, where column q corresponds to the nonbasic variable d_q derived from the target variable at level $L-1$. The outcome is that optimality of the target variable at level $L-1$ is detected. By induction it follows that level 1 is optimal. In fact it is better simply to pivot on the element in row p'' and column corresponding to the target variable from level 1. This returns to the level 1 problem with just one pivot.

It is also possible to give an upper bound on the number of levels of recursion. When a degeneracy block occurs at level L then the target variable d_{q_L} becomes nonbasic but does not participate in the optimality test at level $L+1$ or above. Also there is at least one row with a positive r.h.s. in the level L subtableau (for $L > 1$) which cannot participate in further recursion. Consequently it can be shown that the maximum number of levels is equal to the minimum dimension of the level 1 tableau. The same arguments that are outlined in Section 2 thus lead to guaranteed termination both for exact and inexact arithmetic.

An example is now given that illustrates most aspects of the method. Consider the LP

$$\begin{array}{ll} \text{minimize} & -\frac{3}{4}x_1 + 20x_2 - \frac{1}{2}x_3 + 6x_4 \\ \text{subject to} & \frac{1}{4}x_1 - 8x_2 - x_3 + 9x_4 \le 0 \\ & \frac{1}{2}x_1 - 12x_2 - \frac{1}{2}x_3 + 3x_4 \le 0 \\ & x_3 \le 1 \qquad\qquad x_i \ge 0 \quad i = 1,2,3,4 \end{array} \tag{3.6}$$

which was devised by Beale to illustrate cycling in the Simplex method. Slack variables x_5, x_6, x_7 are added to the first three constraints and x_0 as defined in (3.4) is the target variable for the objective function. The initial tableau is

$$\begin{array}{c} \\ x_0 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & \end{array} \\ \left[\begin{array}{ccccc} \boxed{-\frac{3}{4}} & 20 & -\frac{1}{2} & 6 & 0 \\ \hline \frac{1}{4} & -8 & -1 & 9 & 0 \\ \frac{1}{2} & -12 & -\frac{1}{2} & 3 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{array}\right] \end{array} \tag{3.7}$$

and is feasible so that no Phase I is required. The optimality test on the x_0 row indicates that the nonbasic variable x_1 is to be increased, but the resulting line search is blocked by both variables x_5 and x_6. The sub-tableau for resolving degeneracy is therefore set up, by ignoring any nondegenerate rows (here the x_7 row), and introducing $d_0 = x_0 - \hat{x}_0 = x_0$.

The first step of the process is to make d_0 nonbasic which is achieved by pivoting on

the boxed element in (3.7), giving rise to the tableau

$$\begin{array}{c} \\ x_1 \\ x_5 \\ x_6 \\ \hline x_7 \end{array} \begin{array}{c} \begin{array}{cccc} d_0 & x_2 & x_3 & x_4 \end{array} \\ \left[\begin{array}{cccc|c} -\frac{4}{3} & -\frac{80}{3} & \frac{2}{3} & -8 & 0 \\ \frac{1}{3} & -\frac{4}{3} & -\frac{7}{6} & 11 & 0 \\ \frac{2}{3} & \frac{4}{3} & \boxed{-\frac{5}{6}} & 7 & 0 \\ \hline 0 & 0 & 1 & 0 & 1 \end{array}\right] \end{array} \begin{array}{l} \\ \leftarrow \frac{4}{3} \\ \leftarrow -\frac{1}{3} \\ \leftarrow -\frac{2}{3} \\ \\ . \end{array} \tag{3.8}$$

To define the level 2 right hand sides, the negative of the d_0 column overwrites the zero right hand sides at level 1, as indicated by the left-pointing arrows, and it can be seen that infeasibility is introduced into the x_5 and x_6 rows. Note that x_1, which has become basic, is included among the level 2 rows and is not ignored.

The next step is to solve the level 2 problem by using the One-at-a-time method to restore feasibility to the level 2 rows in (3.8). The largest infeasibility at level 2 is $\hat{x}_6 = -\frac{2}{3}$, which determines x_6 as the target residual, and the negative boxed element indicates that x_6 can be increased by increasing x_3. A ratio test between the r.h.s. and the x_3 column indicates that x_6 reaches zero before x_1, so that a pivot can be taken on the boxed element. This gives rise to the tableau

$$\begin{array}{c} \\ x_1 \\ x_5 \\ x_3 \\ \hline x_7 \end{array} \begin{array}{c} \begin{array}{cccc} d_0 & x_2 & x_6 & x_4 \end{array} \\ \left[\begin{array}{cccc|c} \boxed{-\frac{4}{5}} & -\frac{128}{5} & \frac{4}{5} & -\frac{12}{5} & 0 \\ -\frac{3}{5} & -\frac{16}{5} & -\frac{7}{5} & \frac{6}{5} & 0 \\ -\frac{4}{5} & -\frac{8}{5} & -\frac{6}{5} & -\frac{42}{5} & 0 \\ \hline \frac{4}{5} & \frac{8}{5} & \frac{6}{5} & \frac{42}{5} & 1 \end{array}\right] \end{array} \begin{array}{l} \\ \leftarrow \frac{4}{5} \\ \leftarrow \frac{3}{5} \\ \leftarrow \frac{4}{5} \\ \\ . \end{array} \tag{3.9}$$

It is seen that the level 2 right hand sides are feasible. Correspondingly the elements in the level 2 rows of the d_0 column are non-positive, so that it is possible to increase d_0 without creating infeasibility in x_B. Thus a feasible solution to the level 1 direction finding problem has been obtained. It is now possible to pivot on any negative element in the d_0 column (e.g. the x_1 row), and replace d_0 by x_0, giving the tableau

$$\begin{array}{c} \\ x_0 \\ \hline x_5 \\ x_3 \\ x_7 \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & x_6 & x_4 \end{array} \\ \left[\begin{array}{cccc|c} -\frac{5}{4} & 32 & -1 & 3 & 0 \\ \hline -\frac{3}{4} & 16 & -2 & 3 & 0 \\ -1 & 24 & -2 & -6 & 0 \\ 1 & -24 & 2 & 6 & 1 \end{array}\right] \end{array} \tag{3.10}$$

in which there is no degeneracy block and x_1 can be increased.

As mentioned above, it is more efficient to omit the final pivot in which the target variable d_0 becomes basic, and instead carry out a ratio test on the level 1 rows of (3.9), using the d_0 column. This saves an unnecessary pivot operation. The outcome here is a pivot on the $(x_7,\ d_0)$ element in (3.9). After replacing d_0 by x_0, the tableau

$$\begin{array}{c} \\ x_1 \\ x_5 \\ x_3 \\ \\ x_0 \end{array} \left[\begin{array}{cccc|c} x_7 & x_2 & x_6 & x_4 & \\ 1 & -24 & 2 & 6 & 1 \\ \frac{3}{4} & -2 & -\frac{1}{2} & \frac{15}{2} & \frac{3}{4} \\ 1 & 0 & 0 & 0 & 1 \\ \hline \frac{5}{4} & 2 & \frac{3}{2} & \frac{21}{2} & \frac{5}{4} \end{array}\right] \tag{3.11}$$

is obtained, which is optimal.

To illustrate a different outcome of the technique, let the $(x_6,\ x_3)$ entry in (3.7) be changed to $\frac{1}{2}$ so that the corresponding entry in (3.8) changes to $\frac{1}{6}$. Then there are no negative elements in the x_6 row in (3.8), indicating that the level 2 problem is optimal, so that the direction finding problem at level 1 is infeasible. Hence $\hat{x} = 0$ is optimal in the parent problem. An optimal tableau for the latter in conventional form is then obtained by pivoting on the (x_6, d_0) entry, and replacing d_0 by x_0.

Another technique for resolving degeneracy is that of Wolfe [12]. As in our new method, the recursively derived problems are in the primal LP variables, so that the technique is also able to make full use of primal steepest edge coefficients. It is therefore important to explain why we do not prefer Wolfe's technique. The technique is based on perturbing degenerate constraints to be strictly feasible when a degeneracy block occurs. For example in (3.7) the right hand sides $\hat{x}_5$ and $\hat{x}_6$ could be perturbed to have the value 1, and these would be the level 2 right hand sides. The level 2 tableau is thus not degenerate and a pivot on the $(x_6,\ x_1)$ entry gives rise to the tableau

$$\begin{array}{c} \\ x_0 \\ x_5 \\ x_1 \\ x_7 \end{array} \left[\begin{array}{cccc|c} x_6 & x_2 & x_3 & x_4 & \\ \frac{3}{2} & 2 & -\frac{5}{4} & \frac{21}{5} & 0 \\ \hline -\frac{1}{2} & -2 & -\frac{3}{4} & \frac{3}{4} & 0 \\ 2 & -24 & -1 & 6 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{array}\right] \begin{array}{l} \\ \\ \leftarrow \frac{1}{2} \\ \leftarrow 2 \\ \\ \end{array} \tag{3.12}$$

Now the optimality test indicates that x_3 should be increased. As all the entries in the level 2 part of this column are nonpositive, the level 2 tableau is seen to be unbounded. This signifies that the degeneracy block at level 1 has been removed.

This example illustrates that in Wolfe's technique the level L problem ($L > 1$) is terminated by recognising *unboundedness* in the level L sub-tableau. Heuristics could be used to ensure monotonic improvement of the objective function during the solution of the level L problem, but these would *not* imply a guarantee of termination for inexact arithmetic. Indeed if one were allowed to introduce arbitrary errors into a column at any

stage, it would be possible to ensure that some positive entries are always present in the level L rows of a column with a negative cost. This would ensure that the level L problem never terminates. Although unlikely, it is not impossible to imagine this situation occurring in practice, particularly for highly degenerate large problems for which insufficiently large tolerances have been set.

4. Implementation as an ASM

The general approach to solving an LP by an ASM has been outlined for example in [4] and the relation to a tableau representation (as a special case of a linear complementarity problem (LCP)) is given in [6]. The new method is readily applied in an ASM context by formulating an ASM for (1.1) in terms of a standard Simplex method tableau.

For an ASM, the constraints in (1.1) are expressed in terms of a vector of *residuals*

$$r = A^T x - b \geq 0. \tag{4.1}$$

A permutation matrix P is introduced so that

$$Pr = \begin{bmatrix} r_{\mathcal{A}} \\ r_{\mathcal{I}} \end{bmatrix} = \begin{bmatrix} A_{\mathcal{A}}^T \\ A_{\mathcal{I}}^T \end{bmatrix} x - \begin{pmatrix} b_{\mathcal{A}} \\ b_{\mathcal{I}} \end{pmatrix} \tag{4.2}$$

where the set $\mathcal{I}$ is the complement of $\mathcal{A}$ and contains the indices of inactive constraints (and possibly some degenerate constraints). The matrix $A_{\mathcal{I}}$ is the matrix whose columns are the normal vectors $a_i,\ i \in \mathcal{I}$, and vectors such as $r_{\mathcal{A}}$ collect the elements $r_i,\ i \in \mathcal{A}$. The current value of the active constraints is $\hat{r}_{\mathcal{A}} = 0$ so (4.2) defines the current vertex to be

$$\hat{x} = A_{\mathcal{A}}^{-T} b_{\mathcal{A}}. \tag{4.3}$$

The active constraints are used to eliminate x from the problem using

$$x = A_{\mathcal{A}}^{-T}(b_{\mathcal{A}} + r_{\mathcal{A}}) = \hat{x} + A_{\mathcal{A}}^{-T} r_{\mathcal{A}}. \tag{4.4}$$

Thus the residuals of constraints in $\mathcal{I}$ can be written from (4.2) and (4.4) as

$$\begin{aligned} r_{\mathcal{I}} &= A_{\mathcal{I}}^T(\hat{x} + A_{\mathcal{A}}^{-T} r_{\mathcal{A}}) - b_{\mathcal{I}} \\ &= \hat{r}_{\mathcal{I}} + \hat{A}^T r_{\mathcal{A}} \end{aligned} \tag{4.5}$$

where $\hat{r}_{\mathcal{I}} = A_{\mathcal{I}}^T \hat{x} - b_{\mathcal{I}}$ denotes the current value of residuals in $\mathcal{I}$, and

$$\hat{A} = A_{\mathcal{A}}^{-1} A_{\mathcal{I}} \tag{4.6}$$

is a matrix that arises in the LCP tableau for an LP (see [6]). (The use of $\hat{A}$ here differs from its usage in Section 3.) Likewise the objective function can be expressed as

$$f = c^T x = c^T(\hat{x} + A_{\mathcal{A}}^{-T} r_{\mathcal{A}}) = \hat{f} + \hat{c}^T r_{\mathcal{A}} \tag{4.7}$$

test problems	numbers of variables n	general constraints	nonzero elements in A
adlittle	97	56	465
share2b	79	96	730
share1b	225	117	1182
beaconfd	262	173	3476
israel	142	174	2358
brandy	249	220	2150
e226	282	223	2767
capri	353	271	1786
bandm	472	305	2659
stair	467	356	3857
etamacro	688	400	2489

Table 1. Characteristics of SOL test problems

where $\hat{f}$ is the current function value and $\hat{c} = A_{\mathcal{A}}^{-1}c$ is the vector of Lagrange multipliers. Equations (4.5) and (4.7) can be written in tableau form as

$$\begin{array}{c} \\ r_0 \\ r_{\mathcal{I}} \end{array} \begin{array}{c} \begin{array}{cc} r_{\mathcal{A}} & \end{array} \\ \left[\begin{array}{c|c} \hat{c}^T & -\hat{f} \\ \hline -\hat{A}^T & \hat{r}_{\mathcal{I}} \end{array}\right] \end{array} \tag{4.8}$$

where $r_0 = -f$, which is equivalent to the standard Simplex tableau.

Quantities used in the ASM can be obtained from selected rows and columns of the tableau, so that the tableau is only represented implicitly. In Phase I, if r_q is the target residual to be increased, multipliers for the optimality test are obtained from row q of the tableau and are given by the row vector $-a_q^T A_{\mathcal{A}}^{-T}$. This requires a solve with $A_{\mathcal{A}}$ and is the standard 'ftran' operation of LP. In Phase II, $a_0 = -c$ replaces a_q. Denominators for the ratio test are obtained from column p of the tableau where p is the index chosen in the optimality test. This column is the vector $A_{\mathcal{I}}^T A_{\mathcal{A}}^{-T} e_p$ and requires a solve with $A_{\mathcal{A}}^T$ on the unit vector e_p, followed by inner products with the columns of $A_{\mathcal{I}}$. This solve operation is the 'btran' operation of LP. Moreover, if steepest edge coefficients are updated, an additional solve with $A_{\mathcal{A}}$ is required.

A pivot operation in the tableau causes one column of $A_{\mathcal{A}}$ to be interchanged with a column of $A_{\mathcal{I}}$. It is important if large problems are to be solved efficiently that a sparse invertible representation of $A_{\mathcal{A}}$ is available which can be updated efficiently when columns are interchanged. A good reference to recent developments in this respect is given by Suhl [10].

A preliminary LP code has been written and shows considerable promise. Results are given for a variety of small to medium sized LPs in the SOL test set, the characteristics of which are shown in Table 1. The numbers of pivots required by the new code is shown

test problems	new method				bqpd (not steepest edge)			
	with s.edge		without s.e.		dense		sparse	
adlittle	78	(18)*	154	(20)	119	(11)	89	(9)
share2b	87	(51)	125	(88)	110	(66)	90	(55)
share1b	237	(153)	374	(243)	387	(225)	262	(127)
beaconfd	37	(0)	36	(0)	31	(0)	28	(0)
israel	177	(10)	377	(117)	399	(8)	394	(8)
brandy	209	(103)	317	(225)	381	(319)	274	(226)
e226	367	(72)	703	(127)	663	(144)	574	(88)
capri	313	(239)	479	(334)	480	(367)	484	(405)
bandm	305	(160)	588	(272)	522	(282)	446	(211)
stair	341	(207)	515	(318)	636	(342)	592	(337)
etamacro	557	(308)	873	(452)	1293	(829)	947	(530)

* bracketed figure is number of pivots in Phase I

Table 2. Comparison of total numbers of pivots required

test problems	new method (dense factors)		bqpd (not steepest edge)	
	with s.edge	without s.e.	dense	sparse
adlittle	1.5	2.4	7.7	1.6
share2b	1.8	2.0	6.5	1.6
share1b	21.6	27.6	90.4	12.5
beaconfd	11.1	8.9	25.9	3.1
israel	6.7	16.1	58.0	14.6
brandy	42.5	45.8	123.6	20.0
e226	42.4	69.1	279.5	33.7
capri	74.4	97.2	306.2	27.8
bandm	223.1	247.1	504.8	46.9
stair	192.1	232.8	639.5	163.2
etamacro	379.9	406.9	2461.1	70.8

Table 3. Comparison of computing time (seconds on a SUN ELC)

in Table 2. The method is initialized by taking as many equality constraints as possible into the active set, and completing the set with simple bounds chosen to keep $A_{\mathcal{A}}$ well-conditioned. The effect of using the steepest edge criterion (1.3) as against the Dantzig criterion (1.2) is shown, and an improvement mostly in the range of 30$\sim$50% is observed. A comparison against the bqpd code which also uses the Dantzig criterion is given. This is not directly comparable since a different method is used in Phase I, but nonetheless a significant improvement is also apparent. The differing results for the dense and sparse matrix algebra packages for bqpd are caused by the difference between partial pivoting and Markowitz pivoting in the crash start, and the random effect of differing round-off

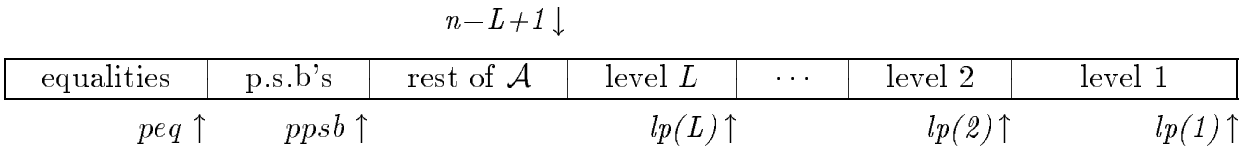
propagation.

However the improvement obtained by steepest edge is won at the expense of an extra solve with $A_{\mathcal{A}}$ on each iteration, so a more important statistic is the total computing time required to solve each problem. This is shown in Table 3. In this case the overall improvement is around 20% although there is considerable variation from problem to problem. Nonetheless this improvement is worth having, particularly as there are other benefits from having steepest edge coefficients available (see the next section). However it is likely that to maintain steepest edge coefficients in both primal and dual spaces, which requires two extra solves, would not be cost effective.

The new code has been implemented for a more general form of an LP problem given by

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} && l \leq \begin{pmatrix} x \\ A^T x \end{pmatrix} \leq u \end{aligned} \tag{4.9}$$

where $A \in \mathbb{R}^{n \times m}$. Many of the features of bqpd (see [7], [6]) have been incorporated into the code and similar data structures are used. Special attention is given to equality constraints ($l_i = u_i$) and to pseudo-bounds (p.s.b's) ($l_i < \hat{x}_i < u_i$) that arise in $\mathcal{A}$. At level 1 the indexing vector ls (see [6]) contains the indices of $\mathcal{A}$ in the first n positions followed by those of $\mathcal{I}$. Pointers peq and $ppsb$ to the end of the groups of equality and psb constraints in $\mathcal{A}$ are also kept. The length of ls is $n+m$ or $n+m+1$ according to whether the algorithm is in Phase I or II. A sign is associated with each element of ls which indicates whether the lower $(+)$ or upper $(-)$ bound is currently being considered. At level L the indices in $\mathcal{I}$ are grouped according to level and there is a list of pointers lp whose elements point to the last entry in each group of indices. The general structure of ls may be pictured as



The index of the target variable r_q at level j is stored in $ls(lp(j))$. In general the elements of $\mathcal{A}$ are the first $n-L+1$ elements in ls and $ls(lp(j))$ for $j = 2, ..., L$.

The opportunity has also been taken to improve the efficiency of the dense matrix package used in the new code. This gives considerably improved performance over the dense code used by bqpd, as Table 3 indicates. However the new code is not finalized and decisions have yet to be taken about how best to truncate near-zero quantities that arise due to rounding errors. There are also some options in regard to how Phase I is carried out. At levels $L > 1$ the One-at-a-time method makes it easy to stack up the cost functions at different levels of recursion. However the code at level 1 is not used at higher levels since it caters for various special features such as two-sided bounds, equalities, psb's, etc. Therefore it is straightforward to use a different Phase I method at level 1: for example by minimizing an l_1 sum of constraint violations in some way.

5. Steepest Edge and Conditioning

In this and the next section it is shown how the availability of steepest edge coefficients provides useful spin-off in that it enables realistic estimates of the actual conditioning of the systems that arise in an ASM to be assessed. The estimates arise from the idea of *expected conditioning* introduced in [3]. This idea is summarised below and the arguments that support its effectiveness are reviewed. Application of the idea in an ASM context is described and it is shown that availability of steepest edge coefficients makes the estimates readily available. A table of condition estimates for SOL problems is given and provokes some interesting questions.

First the idea of expected conditioning is reviewed. Consider the nonsingular system

$$Ax = b \tag{5.1}$$

and let a perturbation E in A induce a perturbation h in x, so that

$$(A + E)(x + h) = b. \tag{5.2}$$

It is assumed that $A + E$ is also nonsingular so that

$$h = (A + E)^{-1}Ex \tag{5.3}$$

relates h to E. In practice E is not known, but some estimate of its magnitude is often available. Condition estimates represent an attempt to express the magnitude of h in terms of the magnitude of E, usually in a relative sense.

A common technique is to take a norm inequality in (5.3) and rearrange, giving

$$\frac{\|h\|}{\|x\|} \leq \kappa_{\text{norm}} \frac{\|E\|}{\|A\|} \tag{5.4}$$

where

$$\kappa_{\text{norm}} = \|(A + E)^{-1}\|\|A\| \tag{5.5}$$

is the *norm condition number*. (Various other ways of presenting this result are possible.) The ratio $\|E\|/\|A\|$ is regarded as a measure of the relative error of the matrix A, and (5.4) indicates that the relative error in the solution is at most κ_{norm} times the relative error in the matrix. Often κ_{norm} is used as an estimate of the actual conditioning of (5.1) in some sense. However it is argued below that this can be unreliable.

In the expected conditioning approach, a particular assumption is made about how the perturbation E arises, which enables a statistical estimate of its effect to be made. It is most convenient to assume that E is defined by

$$e_{ij} = \varepsilon_{ij} a_{ij} \qquad \forall i, j \tag{5.6}$$

where the ε_{ij} are sampled from a uniform distribution in $[-\varepsilon, \varepsilon]$ for some ε. Thus ε measures the componentwise relative error in A. The ε_{ij} have mean and variance given by

$$\mathcal{E}(\varepsilon_{ij}) = 0 \qquad \mathcal{E}(\varepsilon_{ij}^2) = \tfrac{1}{3}\varepsilon^2, \tag{5.7}$$

where $\mathcal{E}(.)$ denotes expected value, and are assumed to be statistically independent, that is

$$\mathcal{E}(\varepsilon_{ij}\varepsilon_{kl}) = 0 \qquad \forall\, i,j \neq k,l. \tag{5.8}$$

It is also assumed that the effect of variations in E on the value of $(A+E)^{-1}$ can be ignored. It is convenient to introduce the notation that a vector or matrix in *square* brackets denotes the corresponding vector or matrix in which all the elements have been *squared*. It then follows from (5.3) (5.6) (5.7) and (5.8) that

$$\mathcal{E}\left([h]\right) = \tfrac{1}{3}\varepsilon^2[(A+E)^{-1}][A][x]. \tag{5.9}$$

Since $\|h\|^2 = e^T[h]$ (where $e^T = (1,1,...,1)$ and using the 2-norm), it follows that

$$\begin{aligned}\mathcal{E}\left(\frac{\|h\|^2}{\varepsilon^2\|x\|^2}\right) &= \tfrac{1}{3}e^T[(A+E)^{-1}][A][x]/e^T[x] \\ &= \tfrac{1}{3}\theta^T[x]/e^T[x]\end{aligned} \tag{5.10}$$

where $\theta^T = e^T[(A+E)^{-1}][A]$. This gives the expected value of the square of the actual conditioning of the problem for a particular x. It is argued in [3] that the estimate

$$\mathcal{E}\left(\frac{\|h\|}{\varepsilon\|x\|}\right) \simeq 0.8\sqrt{\mathcal{E}\left(\frac{\|h\|^2}{\varepsilon^2\|x\|^2}\right)} \simeq 0.462\sqrt{(\theta^T[x]/e^T[x])} \tag{5.11}$$

is adequate for practical purposes. This provides an estimate

$$\kappa_x = 0.462\sqrt{(\theta^T[x]/e^T[x])} \tag{5.12}$$

of the actual conditioning of a particular system $Ax = b$. An attainable upper bound over all $x \neq 0$ is readily deduced as

$$\kappa_A = \max_{x\neq 0}\mathcal{E}\left(\frac{\|h\|}{\varepsilon\|x\|}\right) = 0.462\sqrt{\|\theta\|_\infty} \tag{5.13}$$

which may be regarded as defining a condition number κ_A that is relevant to all systems involving A. It is important to realise that (5.13) has been obtained by *averaging* over E and *bounding* over x, in contrast to (5.5). It is also possible [3] to obtain lower bounds on expected conditioning in a similar way.

Practical experience [3] indicates that (5.12) and (5.13) provide much more accurate estimates of actual conditioning than does (5.5). An important property of (5.12) and (5.13) is they are invariant under row scaling of the equations (5.1). Moreover κ_A and κ_{A^T} differ and so can reveal differences in conditioning between $Ax = b$ and $A^Ty = c$. The matrix

$$L = \begin{bmatrix} 0.9525 & & & & \\ 0.6350 & 0.2245 & & & \\ 0.4762 & 0.2694 & 0.05499 & & \\ 0.3810 & 0.2694 & 0.09427 & 0.01361 & \\ 0.3175 & 0.2566 & 0.11780 & 0.03024 & 0.003381 \end{bmatrix} \tag{5.14}$$

due to Wilkinson [11] provides an example of this for which $\kappa_L = 3.12$ and $\kappa_{L^T} = 174$ (taking $(L+E)^{-1}$ as the computed inverse of L). Taking the right hand side as e, and solving $Lx = e$ and $L^T y = e$ gives $\kappa_x = 62.2$ and $\kappa_y = 1.87$. In practice we would like to regard the true value of actual conditioning as being $\|h\|/(u\|x\|)$ where u is the unit round-off. Solving the systems in single precision and estimating h using iterative refinement gives the actual conditioning of $Lx = e$ as 12.9 and that of $L^T x = e$ as 0.191. Thus there is a considerable difference in the conditioning of L and L^T as the expected condition numbers predict. The actual condition estimates κ_x and κ_y are within a factor of ten. Other experiments reported in [3] show even closer agreement. In contrast the norm condition number for both L and L^T is 1239 for the 2-norm and larger for other common norms. This gives a very pessimistic estimate of actual conditioning and does not predict the difference in conditioning between L and L^T. It is also of interest that the LINPACK condition estimator in matlab gives condition estimates of 1554 for $Lx = b$ and 1303 for $L^T x = b$ so again does not provide adequate estimates of actual conditioning.

The main purpose of this section is to relate these ideas to the steepest edge coefficients γ defined in (1.4). In an ASM the current vertex is defined by the system

$$A_{\mathcal{A}}^T \hat{x} = b_{\mathcal{A}}. \tag{5.15}$$

The appropriate value of θ for calculating $\kappa_{\hat{x}}$ and $\kappa_{A_{\mathcal{A}}^T}$ (see (5.12) and (5.13)) is

$$\theta^T = e^T [A_{\mathcal{A}}^{-T}][A_{\mathcal{A}}^T] \tag{5.16}$$

(taking $A_{\mathcal{A}}^{-T}$ as the computed value of the inverse). Since d_i is a column of $A_{\mathcal{A}}^{-T}$ it follows that $[d_i] = [A_{\mathcal{A}}^{-T}]e_i$ and hence

$$\gamma_i = d_i^T d_i = e^T [d_i] = e^T [A_{\mathcal{A}}^{-T}] e_i. \tag{5.17}$$

Thus the coefficients θ required in (5.12) and (5.13) are given by

$$\theta = [A_{\mathcal{A}}]\gamma. \tag{5.18}$$

The availability of steepest edge coefficients γ thus enables the vector θ, and hence the condition estimates (5.12) and (5.13), to be readily computed, particularly when $A_{\mathcal{A}}$ is a sparse matrix. There are various ways in which these estimates might be used, related to the detection of ill-conditioning and the setting of suitable tolerances. It is advantageous that (5.18) is invariant under scaling of the constraints, so that (5.12) and (5.13) provide a more reliable indication of conditioning than might be obtained say by examining $\|\gamma\|$. However, because the 2-norm is used to measure x in (5.10) and (5.11), it is still important to scale the components of x to have similar magnitude if at all possible.

In practice the method of solving (5.15) or related systems takes account of unit columns in $\mathcal{A}$, in such a way that no rounding error is attributed to these columns. Thus it is appropriate to modify the assumption about E so that no perturbation of the unit columns is made. The appropriate change to (5.18) is to replace any unit columns of $[A_{\mathcal{A}}]$ by zero columns. This modified form has been used in all calculations.

test problems	final active set		max over all iterations	
	$\kappa_{\hat{x}}$	$\kappa_{A_{\mathcal{A}}^T}$	κ_{d_p}	$\kappa_{A_{\mathcal{A}}^T}$
adlittle	5.08	37.9	16.9	432
share2b	34.4	168	463	64108
share1b	18.5	3391	172	77829
beaconfd	0.541	3.82	657	26534
israel	1.65	653	163	7558
brandy	1.26	148	3522	23342
e226	2.03	1604	86.6	13536
capri	2.05	686	110	21945
bandm	0.953	92.7	121	17258
stair	2.16	236	14.7	51592
etamacro	0.944	222	146	741215

Table 4. Expected condition estimates for SOL test problems

If steepest edge coefficients for the dual system $A_{\mathcal{A}}\lambda = c$ are available then these can similarly be used to determine condition numbers κ_λ and $\kappa_{A_{\mathcal{A}}}$ which predict the conditioning of the dual system. However with the new method these estimates are not readily available. Some earlier calculations for the SOL test set, in which both primal and dual steepest edge coefficients were available, indicated that in most cases the condition numbers κ_x and κ_λ were of similar magnitude.

Expected condition estimates obtained whilst solving the SOL test problems are given in Table 4. For the final active set the numbers $\kappa_{\hat{x}}$ and $\kappa_{A_{\mathcal{A}}^T}$ are given. In fact, the current vertex $\hat{x}$ is not obtained by solving (5.15) but by updating the previous estimate with a multiple of the vector d_p, which is determined by solving the system $A_{\mathcal{A}}^T d_p = e_p$. Thus the value of κ_{d_p} is more relevant to the calculation that is actually made on an intermediate iteration. Therefore the maximum values of κ_{d_p} and $\kappa_{A_{\mathcal{A}}^T}$ over all iterations are also tabulated. As expected the numbers $\kappa_{\hat{x}}$ and κ_{d_p} are seen to provide much sharper estimates of actual conditioning than $\kappa_{A_{\mathcal{A}}^T}$. However it may be a useful rule of thumb that the latter is about a factor of 100 larger.

These experiments provoke interesting questions regarding how much precision is needed to solve LP problems by Simplex-like methods. For the final active set, the systems here are remarkably well conditioned. Even the maximum value of κ_{d_p} over all iterations is quite small, the worst case being the brandy problem. With this possible exception, one might therefore conclude that 4 decimal digit floating point arithmetic should be sufficient to provide at least one or two digits of accuracy in the final solution. In fact most LP codes use double or higher precision and I suspect that they might have the occasional difficulty with the above set of problems in single precision, and even more so in 4 digit arithmetic. I feel that we still have a lot to learn about the interaction of precision and conditioning in Simplex and ASM solvers, which could have important implications for the reliability of such codes.

6. Conditioning of the Null Space Matrix

In quadratic programming and nonlinear programming by active set methods, an important concept is that of the null space matrix Z whose columns provide a basis for directions within the active constraint manifold. A general formulation (e.g. [4]) is to write

$$B = [\, A_{\mathcal{A}} \quad V \,] \qquad B^{-1} = \begin{bmatrix} Y^T \\ Z^T \end{bmatrix} \tag{6.1}$$

where $A_{\mathcal{A}} \in \mathbb{R}^{n \times m}$ $(m \leq n)$ is the matrix whose columns are the normal vectors of the constraints in the active set, and V is some convenient matrix that is adjoined to make B nonsingular. Null space methods require the ability to calculate vectors of the form Zu and $Z^T u$. This is done implicitly, for example to calculate the vector $x = Zu$, the system

$$B^T x = \begin{pmatrix} 0 \\ u \end{pmatrix} \tag{6.2}$$

is solved.

If primal steepest edge coefficients are available, these measure the lengths (squared) of the columns of Y and Z. However it is the case, as in Section 5, that the conditioning of Zu and $Z^T u$ is in general different. Primal steepest edge coefficients only enable us to say something about Zu, for which κ_{B^T} applies, by virtue of (6.2) and (5.15) ff. in the previous section.

A particularly important case is when V is a selection of columns from the unit matrix. In this case we can assume that these columns do not contribute to the perturbation matrix E. Then (5.13) and (5.18) simplify to give

$$\theta = [A_{\mathcal{A}}]\gamma_{\mathcal{A}} \qquad \kappa_{B^T} = 0.462\sqrt{\|\theta\|_\infty} \tag{6.3}$$

where $\gamma_{\mathcal{A}}$ collects the elements of γ that correspond to indices in $\mathcal{A}$. Of course (6.3) may be further simplified by zeroing any unit columns in $[A_{\mathcal{A}}]$ when computing θ, for the same reasons. Again (6.3) is readily computed and provides a measure of the ill-conditioning in B^T which could for example indicate that a change in the definition of V is needed. Apropos of this situation, it may become necessary to make pivoting tests based on the magnitude of $z_i^T u$ for some u, where z_i denotes a column of Z. The availability of steepest edge coefficients suggests that such tests should be normalized by comparing the magnitudes of $(z_i^T u)^2/\gamma_i$.

7. References

[1] Balinski M.L. and Gomory R.E. (1963). A mutual primal-dual simplex method, in *Recent Advances in Mathematical Programming*, R.L.Graves and P.Wolfe (Eds.), McGraw-Hill, New York.

[2] Dantzig G.B. (1963). *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J..

[3] Fletcher R. (1985). Expected Conditioning, *IMA J. Num. Anal.*, **5**, 247-273.

[4] Fletcher R. (1987). *Practical Methods of Optimization, 2nd.Edition*, John Wiley, Chichester.
[5] Fletcher R. (1988). Degeneracy in the presence of roundoff errors, *Linear Algebra Appl.*, **106**, 149-183.
[6] Fletcher R. (1993). Resolving degeneracy in quadratic programming, *Annals of O.R.*, **47**, 307-334.
[7] Fletcher R. and Hall J.A.J. (1990). Towards reliable linear programming, in *Numerical Analysis 1989*, D.F.Griffiths and G.A.Watson (Eds.), Pitman Research Notes in Mathematics 228, Longman, Harlow.
[8] Forrest J.J. and Goldfarb D. (1992). Steepest-edge simplex algorithms for linear programming, *Math. Programming*, **57**, 341-374.
[9] Goldfarb D. and Reid J.K. (1977). A practical steepest-edge simplex algorithm, *Math. Programming*, **12**, 361-371.
[10] Suhl U.H. (1994). MOPS – Mathematical OPtimization System, *European J. Operational Research*, **72**, 312-322.
[11] Wilkinson J.H. (1965). *The Algebraic Eigenvalue Problem*, O.U.P., Oxford.
[12] Wolfe P. (1963). A technique for resolving degeneracy in linear programming, *SIAM J. Applied Math.*, **11**, 205-211.