STABLE MODIFICATION OF EXPLICIT LU FACTORS FOR SIMPLEX UPDATES

R. FLETCHER and S.P.J. MATTHEWS
Department of Mathematical Sciences, University of Dundee, Dundee, DD1 4HN, Scotland, UK

Received 16 August 1983
Revised manuscript received 15 May 1984

This paper presents a method for modifying an LU factorization of a matrix A when a Simplex update is performed on A. At each stage of the algorithm, two options are allowed, and in one of these a row permutation of L is made so that the LU structure is always maintained. The expanding file of operators used in other methods is not required. The result of an error analysis is presented in which the error is bounded by the growth in the partial sums of LU. Various pivotal strategies which minimize a growth bound are discussed and an example is given in which the worst case for growth occurs repeatedly. Numerical results from experiments on ill-conditioned problems show no tendency for error growth.

Key words: Simplex Update, LU Factors, Explicit Modification, Stable.

1. Introduction

In the Simplex method for linear programming a representation of a nonsingular matrix $A \in \mathbb{R}^{n \times n}$ is stored which enables inverse operations (such as the solution of a system of equations $Ax = b$) to be carried out effectively. On each iteration of the method one column of $A$ is changed and the representation is updated. Many different representations have been suggested; they can be compared on the grounds of both efficiency and numerical stability. The efficiency criterion also depends on whether the matrix $A$ is sparse (many zero elements) or dense.

In the Revised Simplex method, the explicit inverse $A^{-1}$ is updated using an expression that can be derived from the Sherman–Morrison formula (for example, Fletcher [5]). When $A$ is dense, this and other early methods are reasonably efficient ($O(n^2)$ operations per iteration) but they do not adequately take advantage of sparsity in $A$, and are numerically unstable when $A$ is ill-conditioned. Currently the most effective approach is thought to be a product form method [9, 1, 2, 7, 10, 13] in which the representation of $A$ involves a product of elementary triangular and permutation matrices. In a sparse matrix environment the product can be stored in a file in condensed form and so the method is effective in taking advantage of sparsity. Each update requires further elementary matrices to be included in the
representation, and so the file expands as more and more iterations are performed. When the file becomes sufficiently large it is necessary to reinvert, that is to calculate a new representation directly from the current $A$ matrix so as to keep the total computation time per iteration within bounds. The product form method of Forrest and Tomlin [7] is often very efficient in practice, but there are stability problems due to the possibility of large growth in the elements of the representation, and this must be taken into account when implementing the method. Bartels and Golub [1, 2] first recognised the importance of maintaining numerical stability and developed a product form method with this in mind, which was developed for the sparse case by Reid [10] and later by Saunders [13]. Finally an $LQ$ representation of $A$ has also been suggested in which $Q$ is an orthogonal matrix which is not stored in the method (Gill and Murray [8], Saunders [11]). This method has been developed by Saunders [12] in product form.

In this paper the well known representation of $A$ in terms of $LU$ factors is considered. We show that it is possible to update these factors when they are represented explicitly, without the need for an expanding file of operators. Hence for dense matrices no reinversion is necessary. In our method two situations are identified: one in which the standard update is used (essentially that suggested by Bennett [3] for a rank one update and also used in [1] and [2]), the other in which a different update is used and an interchange is made to restore the standard form. The algorithm cannot fail directly, and by choosing between these two possibilities to minimize a bound on error growth, a stable method is obtained. This is true even if any of the $A$ matrices is rank deficient. The new factors can be obtained in $O(n^2)$ operations. In Section 3 we present an error analysis which gives a bound for the round-off errors. We also give examples in which the worst case for growth in the factors occurs on every step of the method. As with similar examples constructed in relation to the partial pivoting strategy for Gaussian elimination (Wilkinson [15]) the factors grow exponentially. However in practice this situation is extremely unlikely and we regard the method as being acceptably stable, a view taken by other authors in similar circumstances (e.g. [4]). In Section 4 we present the results of numerical experiments designed to test the stability of the method as the matrix $A$ changes from being well-conditioned to ill-conditioned over a number of cycles. No significant tendency for the errors in $A$ to grow is observed. Some experiments with the ‘worst case’ matrices are also described.

In Section 5 we discuss the practical importance of the method. We feel that the combination of simplicity and stability make the method particularly suitable for small and medium scale $LP$, and the method might also be useful in certain types of sparse applications. The ideas are also applicable to updating $LU$ factors of rectangular matrices and to the problem of removing a row from a matrix. We are actively pursuing an application of the method to Quadratic Programming. The method can also be used in more general circumstances related to decomposition in the form considered by Todd [14].