J. Inst. Maths Applics (1974) **13**, 117–119

On the Estimation of Sparse Jacobian Matrices

A. R. CURTIS, M. J. D. POWELL AND J. K. REID
Theoretical Physics Division, U.K.A.E.A. Research Group,
Atomic Energy Research Establishment, Harwell, Berks.

[Received 20 March 1972]

We show how to use known constant elements in a Jacobian matrix to reduce the work required to estimate the remaining elements by finite differences.

1. Introduction

MOST METHODS for solving a system of n non-linear equations

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{1.1}$$

or minimizing a sum of squares

$$\sum_{i=1}^{m} [f_i(\mathbf{x})]^2 \tag{1.2}$$

require the evaluation or estimation of the Jacobian matrix

$$\mathbf{J} = \left\{\frac{\partial f_i}{\partial x_j}\right\}. \tag{1.3}$$

A similar requirement arises in some methods for integrating a stiff system of ordinary differential equations

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}, t). \tag{1.4}$$

We consider here the problem of estimating $\mathbf{J}$ by finite differences using a subroutine that calculates $\mathbf{f}$ given $\mathbf{x}$ or $\mathbf{x}$ and t. We show that considerable savings in the number of calls of this subroutine are possible if $\mathbf{J}$ has a large number of elements that are known constants, for example when it is sparse with a known sparsity pattern.

2. Economizing the Number of Subroutine Calls

We first remark that there is no loss in generality in supposing that all the known elements are zeros for we may express $\mathbf{f}$ in the form

$$\mathbf{f}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x}) + A\mathbf{x} \tag{2.1}$$

where A is the matrix of known derivatives. The Jacobian for $\boldsymbol{\phi}$ has zeros in all the positions corresponding to known derivatives of $\mathbf{f}$.

The simplest way to approximate $\mathbf{J}$ is by the use of the differences

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(\mathbf{x}+h\,\mathbf{e}_j) - \mathbf{f}_i(\mathbf{x})}{h} \tag{2.2}$$

where $\mathbf{e}_j$ is the normalized jth coordinate vector and h is an increment. Of course we wish to avoid calculating (2.2) for known zero derivatives. This could be done by requiring a subroutine that calculates the single function f_i rather than the whole

vector **f**, but this would usually be inefficient in view of the large number of subroutine calls needed and the fact that when the whole vector **f** is required it often happens that complicated expressions can be evaluated once and used a number of times.

If **J** is a band matrix of band-width $m = 2\rho - 1$ then the difference (2.2) is zero if $|i-j| \geqslant \rho$; it follows that we may find simultaneously approximations to columns $j+km$, $k = 0, 1, 2, \ldots$, of **J** from the difference $\mathbf{f}(\mathbf{x}+\sum_k h\,\mathbf{e}_{j+km})-\mathbf{f}(\mathbf{x})$. In this way the total number of subroutine calls needed can be reduced from $n+1$ to $m+1$. This strategy certainly minimizes the total number of function evaluations in view of the number of unknown coefficients in each row of **J**.

In the general case we do not know how to obtain such an optimal strategy, but we have found (see Section 3) that the following procedure usually requires few, if any, more function evaluations than the largest number of non-zeros in a row of **J**. We divide the columns of **J** into groups. To form the first group we inspect the columns in turn and include each that has no unknowns in common with those columns already included. The other groups are formed successively by applying the same procedure to those columns not already included in a group. After the initial calculation of $\mathbf{f}(\mathbf{x})$ a further subroutine call is needed for each group and the point used is obtained from **x** by changing all the variables that correspond to the group columns by h.

It should be noted that the same grouping strategy may be used if higher differences are required in order to obtain more accurate approximations or error estimates. Also we may incorporate individual steps h_j for each x_j to allow for highly non-linear functions f_i. The problem of the choice of these step lengths is discussed by Curtis & Reid (1974).

3. Numerical Tests

To test the effectiveness of our grouping algorithm we used the sparsity patterns of orders 57 and 199 quoted by Willoughby (1971) and the pattern of order 54 used by Curtis & Reid (1971). In each case we used both the matrix and its transpose, and to test whether column ordering is significant we also used five different random column permutations and permutations into ascending and descending order of number non-zeros. The results for the unpermuted matrices are given in Table 1. We found no

TABLE 1

Matrix order (n)	Number non-zeros (n_z)	Whether transposed	Maximal no. of non-zeros in a row	Number of groups (n_g)	$n+n_z$	$n(n_g+1)$
54	291	No	16	16	345	918
		Yes	12	12		702
57	281	No	11	11	338	684
		Yes	11	11		
199	701	No	9	10	900	2189
		Yes	6	9		1990

dependence on permutation, except that (a) on two of the matrices (54 transposed and 57) arranging columns in ascending order of number of non-zeros increased the number of groups by 2, and (b) some permutations on the 199 matrix gave $n_g = 11$ instead of 10, while on its transpose some gave $n_g = 8$ instead of 9.

The grouping is optimal for the two smaller problems. Our experience with permutations shows that it is sub-optimal on matrix 199 transposed and it may be optimal or slightly sub-optimal on 199 itself; note that the matrix with pattern

$$\begin{pmatrix} & X & X \\ X & & X \\ X & X & \end{pmatrix}$$

shows that the optimal number of function calls may exceed the largest number of non-zeros in a row. We must expect it to be sub-optimal in general, but not, our experience suggests, by a large margin.

The last two columns of Table 1 show the total number of evaluations of individual functions $f_i(\mathbf{x})$ necessary to approximate $\mathbf{J}$ in the two cases where (a) a subroutine for evaluating $f_i(\mathbf{x})$ is available and (b) a subroutine for evaluating $\mathbf{f}(\mathbf{x})$ is available and we use the suggested grouping. It will be seen that the latter requires about twice as many evaluations of $f_i(\mathbf{x})$, but we believe that in most cases the gain from the reduced number of subroutine calls and from the single evaluation of expressions required for several functions $f_i(\mathbf{x})$ will usually give ample compensation in computer time, and the required subroutine is likely to be far easier to code.

References

CURTIS, A. R. & REID, J. K. 1971 *J. Inst. Maths Applics* **8**, 344–353.

CURTIS, A. R. & REID, J. K. 1974 *J. Inst. Maths Applics* **13**, 121–126.

WILLOUGHBY, R. A. 1971 Sparse matrix algorithms and their relation to problem classes and computer architecture. In *Large sparse sets of linear equations*. (J. K. Reid, Ed.) London and New York: Academic Press.