A direct method for the solution of large sparse systems of linear equations

M. Shacham* and E. Kehat

Department of Chemical Engineering, Technion—Israel Institute of Technology, Haifa, Israel

A new method for the solution of large sparse systems of linear equations that offers considerable savings of computation time and storage is presented. The proposed method is easily programmable for linear systems with a band type matrix of coefficients and can be used for large sparse systems, where the non-zero elements in the matrix of coefficients are randomly spaced.

The system is restructured by tearing and the whole system is solved as a smaller linear system of the tear variables, using a modification of the Newton method, for which the derivatives are calculated directly from the equations.

(Received March 1974, Revised April 1975)

1. Introduction

Many numerical problems require the solution of a large sparse system of linear equations. Typical examples are the solution of partial differential equations by finite difference approximation, the solution of two point boundary value problems and linear programming. Our objective was to develop an efficient direct method for the solution of large, sparse systems of linear equations. A system of linear equations can be written in the form:

$$\bar{A}\bar{x} = \bar{b} \ . \qquad (1)$$

Direct and iterative methods can be used for the computation of the values of $\bar{x}$.

For direct methods of solution the matrix $\bar{A}$ is brought into the form of an upper triangular matrix by a series of constant multiplications and subtractions of rows. The resultant form of equation (1) with an upper triangular matrix is then solved by back substitution. Gastinel (1970) and Walsh (1971) have reviewed these methods.

The direct methods do not take advantage of the fact that for most of the practical problems the matrix $\bar{A}$ is sparse, and are inefficient for large systems ($n > 100$). Two main drawbacks of the direct methods for the solution of large sparse systems of linear equations are:

1. The matrix $\bar{A}$ and the vector $\bar{b}$ have to be stored. This requires $n(n + 1)$ words of storage for a system of n equations.
2. The number of addition, substraction and multiplication operations is proportional to n^3 (approximately $\frac{1}{3}n^3$ for the Gauss method and $\frac{1}{2}n^3$ for the Jordan method.)

For large systems of linear equations iterative methods are preferred. Their advantage over the direct methods is that it is not necessary to store the whole matrix $\bar{A}$. In many cases the solution may also be attained in smaller numbers of arithmetic operations than is required for direct methods. However, for many cases, the convergence of the iterative methods may be slow or unattainable. The most widely used iterative methods are the Jacobi and Gauss-Seidel methods. The iterative methods for the solution of systems of linear equations were recently reviewed by Young (1971, 1973).

Kron (1963) introduced the concept of decomposition of a large sparse system into a set of smaller subsystems by the use of 'tearing methods'. The current status of decomposition and tearing methods has been reviewed by Harary (1971) and Ledet and Himmelblau (1970).

According to these methods an 'output set' for the system of equations is chosen first. One variable is assigned to each equation as the output variable of this equation. The system of equations is then partitioned into the smallest irreducible subsystem of equations that must be solved simultaneously.

For each subsystem of equations one or more 'tearing' variables are chosen. These 'tearing' variables are the iterates that need be chosen to obtain a solution of the subsystem. The number of tearing variables is usually smaller than the number of equations in the subsystem. An accepted criterion for selecting 'tearing variables' is the minimum number of such variables which will make is possible to solve the whole subsystem. The ordered set of equations that results is then solved by an iterative method.

Descriptions of established methods of partitioning and tearing as well as computer programs for this purpose may be found elsewhere (Ledet and Himmelblau, 1970).

In this paper we propose a method for the direct solution of the sparse system of equations which has been restructured by tearing.

2. The proposed method

Let $\bar{x} = (x_1, x_2, \ldots, x_n)^T$ be the set of variables in (1).

Let us choose $\bar{x}_1 = (x_1, x_2, \ldots, x_m)^T$ as a subset of $\bar{x}$, so that a unique value for the variables $n - m$ may be obtained directly by substitution in equation (2).

$$\begin{aligned} x_{m+1} &= (-a_{11}x_1 - a_{12}x_2 - \ldots - a_{1m}x_m + b_1)/a_{1,m+1} \\ &\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (2) \\ x_n &= (-a_{n-m,1}x_1 - a_{n-m,2}x_2 - \ldots, \\ &\qquad\qquad a_{n-m,n-1}x_{n-1} + b_{n-m})/a_{n-m,n} \end{aligned}$$

where $x_{m+1}, x_{m+2}, \ldots, x_n$ are the output set of variables that can be calculated if $x_1, x_2, \ldots, x_m$ are given.

In the system of equations (2) there are only $(n - m)$ equations from the original n equations of the system of equation (1). The m equations left may be rewritten in the form:

$$\begin{aligned} f_1(\bar{x}_1) &= a_{n-m+1,1}x_1 + a_{n-m+1,2}x_2 + \ldots, \\ &\vdots \qquad\qquad\qquad a_{n-m+1,n}x_n - b_{n-m+1} = 0 \\ f_m(\bar{x}_1) &= a_{n1}x_1 + a_{n2}x_2 + \ldots, a_{nn}x_n - b_n = 0 \qquad (3) \end{aligned}$$

$\bar{f} = [f_1(\bar{x}_1), f_2(\bar{x}_1), \ldots, f_m(\bar{x}_1)]^T = 0$ is a system of m linear equations, where the principal variables are contained in vector $\bar{x}_1$ and the other variables may be calculated by substitution from system (2).

Since the system $\bar{f}_1 = 0$ is linear, it may be solved directly by means of Newton Raphson method in *one iteration*.

Furthermore, the partial derivatives $\dfrac{\partial f_i}{\partial x_j}$ which are required for the Newton–Raphson method, may be calculated from equation (4).

*Present Address: Department of Chemical Engineering, Ben Gurion University of the Negev, Beer-Sheva 84-120, Israel.

$$\frac{\partial f_i}{\partial x_j} = \frac{f_i}{x_j} \tag{4}$$

where f_i is calculated from the systems of equation (2) and (3) by setting $\bar{b} = 0$ and setting all the elements of $\bar{x}_1$ except x_j to zero.

The problems of step size or curve fitting usually associated with the numerical computation of derivatives, are thus sidestepped.

Proof

Let us reorder and renumber the equations and variables of (1) as required by equations (2) and (3).

Now, let us break up the reordered form of $\bar{A}$ as shown:

$$\bar{A} = \begin{array}{c|c} & \begin{array}{ccc} 1 & m & n \end{array} \\ \begin{array}{c} 1 \\ \\ n-m \\ \\ n \end{array} & \left[\begin{array}{c:c} \bar{T} & \bar{S} \\ \hdashline \bar{F}_1 & \bar{F}_2 \end{array}\right] \end{array} \tag{5}$$

where $\bar{T}$ is an $m \times (n-m)$ matrix:

$$\bar{T} = \begin{bmatrix} a_{11} & a_{12} & \dots a_{1m} \\ \vdots & & \\ a_{n-m,1} & a_{n-m,2} & \dots a_{n-m,m} \end{bmatrix} \tag{6}$$

$\bar{S}$ is a lower triangular square matrix with dimension $(n-m) \times (n-m)$

$$\bar{S} = \begin{bmatrix} a_{1,m+1} & & \\ a_{2,m+1} & a_{2,m+2} & \\ \vdots & & \\ a_{n-m,m+1} & a_{n-m,m+2} & \dots a_{n-m,n} \end{bmatrix} \tag{7}$$

and the dimensions of $\bar{F}_1$ and $\bar{F}_2$ are $m \times m$ and $(n-m) \times m$ respectively.

$$\bar{F}_1 = \begin{bmatrix} a_{n-m+1,1} & a_{n-m+1,2} & \dots a_{n-m+1,m} \\ \vdots & & \\ a_{n1} & a_{n2} & \dots a_{nm} \end{bmatrix} \tag{8}$$

$$\bar{F}_2 = \begin{bmatrix} a_{n-m+1,m+1} & a_{n-m+1,m+2} & \dots a_{n-m+1,n} \\ \vdots & & \\ a_{n,m+1} & a_{n,m+2} & \dots a_{nn} \end{bmatrix}$$

Further $\bar{S}$ may be rewritten as $\bar{S} = \bar{L} + \bar{D}$ where $\bar{L}$ is a lower triangular matrix with zeros on the diagonal and $\bar{D}$ is a diagonal matrix.

Now let us break up $\bar{x}$ and $\bar{b}$:

$$\bar{x} = \begin{bmatrix} \bar{x}_1 \\ - \\ x_2 \end{bmatrix} \quad \bar{b} = \begin{bmatrix} \bar{b}_1 \\ - \\ b_2 \end{bmatrix}$$

where $\quad \bar{x}_1$ has already defined and

$$\bar{x}_2 = (x_{m+1}, x_{m+2}, \dots x_n)^T$$
$$\bar{b}_1 = (b_1, b_2 \dots b_{n-m})^T$$

and $\quad \bar{b}_2 = (b_{n-m+1}, b_{n-m+2}, \dots b_n)^T$

The systems (2) and (3) may be rewritten respectively as:

$$\bar{x}_2 = \bar{D}^{-1}(-\bar{T}\bar{x}_1 - \bar{L}\bar{x}_2 + \bar{b}_1) \tag{10}$$

and

$$\bar{f}(\bar{x}_1) = \bar{F}_1\bar{x}_1 + \bar{F}_2\bar{x}_2 - \bar{b}_2 \,. \tag{11}$$

Note that equation (10) may be solved for $\bar{x}_2$ row by row by substitution only.

For the sake of the proof we shall solve equation (10) explicitly for $\bar{x}_2$ by:

$$\bar{x}_2 = (\bar{I} + \bar{D}^{-1}\bar{L})^{-1}\,\bar{D}^{-1}(-\bar{T}\bar{x}_1 + \bar{b}_1) \tag{12}$$

Substituting (12) into (11) obtains:

$$\bar{f}(\bar{x}_1) = \bar{F}_1\bar{x}_1 + \bar{F}_2(\bar{I} + \bar{D}^{-1}\bar{L})^{-1}\,\bar{D}^{-1}(-\bar{T}\bar{x}_1 + \bar{b}_1) - \bar{b}_2$$

After rearrangement:

$$\bar{f}(\bar{x}_1) = [\bar{F}_1 - \bar{F}_2(\bar{I} + \bar{D}^{-1}\bar{L})^{-1}\bar{T}D^{-1}]\bar{x}_1 + \bar{F}_2(\bar{I} + \bar{D}^{-1}\bar{L})\,\bar{D}^{-1}\bar{b}_1 - \bar{b}_2 \tag{14}$$

From (14) it is obvious that $\bar{f}(\bar{x}_1)$ is a linear function of $\bar{x}_1$ *only* and it therefore can be solved for $\bar{x}_1$ by means of the Newton–Raphson method.

For the proof of equation (4) let $\alpha_{11}, \alpha_{12}, \dots, \alpha_{1m}, \alpha_{21}, \dots, \alpha_{mm}$ the constant multipliers of $\bar{x}_1$ from (14). Thus:

$$\bar{\alpha} = \bar{F}_1 - \bar{F}_2(\bar{I} + \bar{D}^{-1}\bar{L})^{-1}\,\bar{D}^{-1}\bar{T}\,. \tag{15}$$

Then the partial derivative is:

$$\frac{\partial f_i}{\partial x_j} = \alpha_{ij}$$

and by calculating the function values from (14), setting $\bar{b} = 0, x_1, x_2, \dots, x_{j-1}, x_{j+1}, \dots, x_m = 0$ and $x_j \neq 0$ one gets

$$\begin{aligned} f_1(\bar{x}) &= \alpha_{1j}x_j \\ f_2(\bar{x}) &= \alpha_{2j}x_j \\ &\vdots \\ f_m(\bar{x}) &= \alpha_{mj}x_j \end{aligned} \tag{16}$$

Since all the additional linear terms vanish, the partial derivatives may be calculated as shown in Equation (4) and the proof is completed.

The algorithm for the solution of a set of linear equations by the proposed method is:

1. Transform the equations into the form of Equations (2) and (3). Arrange the equations so that the minimum number of tear variables is required. (When it is more convenient Equation (5) may be used instead of (2) and (3)).
2. Assume initial values for the tear variables ($\bar{x}_1 \neq 0$) and compute $\bar{f}(x_1)$ by Equations (2) and (3). (It is convenient to choose $\bar{x}_1 = (1, 1, \dots, 1)^T$.
3. Compute the partial derivatives by Equations (2), (3) and (4).
4. Compute $\bar{x}_1^C$ the exact solution for the tear set by one iteration of the Newton–Raphson method.
5. Compute the rest of the values of x_i from equation (2).

This algorithm (excluding the selection of the tear variables and the ordering of the equations) can be easily programmed. A subroutine for solving linear systems by this method is shown in Appendix 1.

For simple problems, or symmetrical systems, the minimum set of tear variables is easily formed by inspection. For complex problems it is recommended to use methods that determine the tear variables and the optimal order of computation (Harary, 1971; Ledet and Himmelblau, 1970).

This method requires $m + 2$ computations of the system of equations and the solution of a set of linear equations of dimension m. Therefore, the method is more efficient if a minimum set of tear variables is used. Essentially, this method reduces the dimension of the problem from n to m.

This method has the advantage of iteration methods of small storage requirements and the advantage of direct methods of obtaining the solution in a fixed number of arithmetic operations.

The proposed method can also be used for the inversion of sparse matrices and for linear systems with more than one right hand side vector of constants. (The inversion of a matrix is equivalent to the problem of the solution of n systems of n linear equations, which vary only in the right hand vector of constants). If there are k right hand side vectors, the solution algorithm must be repeated k times. However, since the equations have to be rearranged only once and the derivatives are the same for the k systems, the amount of computation is considerably reduced. For a system with k right hand vectors (for the inversion of a $n \times n$ matrix $k = n$) the computation of equations (2) and (3) $m + 2k$ times and the solution of a reduced system of k sets of m linear equations, are required.

As for simple direct methods, the proposed method may be sensitive to rounding errors. This problem has not yet been completely solved.

3. Examples

3.1. *A system with a non-symmetric band of non-zero elements in the matrix of coefficients*, taken from Carnahan *et al.* (1969) The matrix formulation of the problem is:

$$\begin{bmatrix} s & 0 & -s & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -c & 0 & -c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & s & 0 & -s & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c & 0 & c & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & s & 0 & -s & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -c & 0 & -c & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & s & 0 & -s & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c & 0 & c & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & s & 0 & -s \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c & 0 & -c \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & s \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \end{bmatrix} = \begin{bmatrix} -5 \\ 10 \\ 0 \\ 10 \\ 0 \\ 10 \\ 0 \\ 10 \\ 5 \\ 10 \\ 0 \end{bmatrix} \tag{17}$$

where $s = 0{\cdot}5$ and $c = 0{\cdot}86603$.

If x_{11} is taken as the tear variable, the system is easily transformed into:

$$\begin{array}{ll} x_{10} = -sx_{11} & x_9 = -10/c - x_{11} \\ x_8 = 5 + s(x_{11} - x_9) & x_7 = 10/c - x_9 \\ x_6 = x_{10} + s(x_9 - x_7) & x_5 = -10/c - x_7 \\ x_4 = x_8 + s(x_7 - x_5) & x_3 = 10/c - x_5 \\ x_2 = x_6 + s(x_5 - x_3) & x_1 = -10/c - x_3 \\ f(x_{11}) = s(x_1 - x_3) - x_4 + 5\ . & \end{array} \tag{18}$$

An exact solution of this problem was obtained in five minutes with a desk calculator, using the proposed algorithm. Fourteen multiplications and divisions were needed. The solution of this problem by Carnahan *et al.* (1969) by the Gauss–Jordan method required about 600 multiplications and divisions and the use of a computer.

3.2. *A linearised material balance of a chemical plant* leads to the following linear system with randomly spaced non-zero elements (taken from Dorn *et al.* (1972) **(See Fig. 1.)**
(The zero elements of this coefficient matrix are not shown for the sake of clarity). By tearing variables x_4, x_5, x_6 the coefficient matrix can be rearranged into the form of equation (5). **(Fig. 2)**
The algorithm of the proposed method requires calculation of the whole system five times and a solution of a 3×3 linear system of equations. An exact solution of this system was obtained by the proposed method in one iteration. By the Gauss–Seidel method 22 iterations (thus 22 calculations of the whole system) were required for solution where the absolute error of every one of the variables is less than 10^{-7}. (The starting vector was $\bar{x}^0 = (1, 1, \ldots, 1)^T$). The advantage of the proposed method is obvious. For this case it is extremely difficult to use direct methods efficiently because of the unordered spacing of the non-zero elements.

3.3. *A system of linear submatrices obtained by the usual five point difference approximation for the Laplace equation in a rectangle* (Reid; 1971)
For a $k \times l$ grid, the matrix A is of the order of $n = k \times l$ and has an $l \times l$ block form:

$$A = \begin{bmatrix} \bar{\bar{T}} & -\bar{I} & & & \\ -\bar{I} & \bar{\bar{T}} & -\bar{I} & & \\ & -\bar{I} & \bar{\bar{T}} & -\bar{I} & \\ & & \cdots & & \\ & & & \cdots & \\ & & & -\bar{I} & \bar{\bar{T}} \end{bmatrix} \tag{19}$$

where $\bar{I}$ is the $k \times k$ identity matrix and $\bar{\bar{T}}$ is the $k \times k$ tridiagonal matrix:

$$\bar{\bar{T}} = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot & \cdot \\ & & & & \cdot & \cdot & \cdot \\ & & & & & -1 & 4 & -1 \\ & & & & & & -1 & 4 \end{bmatrix} \tag{20}$$

Obviously, the minimum number of tear streams is the smaller value of either k or l. If k has the smaller value, the first k variables can be used as the tear variables and the system can be transformed into:

$$\begin{array}{ll} x_{i+k} = c_1 + c_2 + c_3 + 4x_i - b_i & i = 1, 2, \ldots, n - k \\ f_j \phantom{_{+k}} = c_1 + c_2 + c_3 + 4x_i - b_i & i = n - k + 1, \ldots, n, j = i + k - n \end{array}$$

where:

$$c_1 = \begin{cases} 0 & \text{for } i \leqslant k \\ -x_{i-k} & \text{for } i > k \end{cases}$$

$$c_2 = \begin{cases} 0 & \text{for the first row of each block} \\ -x_{i-1} & \text{elsewhere} \end{cases}$$

$$c_3 = \begin{cases} 0 & \text{for the last row of each block} \\ -x_{i+1} & \text{elsewhere} \end{cases}$$

$$\bar{b} = 1$$

The problem was solved for $k = 5$ and $l = 2, 6, 10$ by the proposed method by the Gauss elimination method with pivoting on the largest element of each column and by the

Table 1 Comparison of the CPU Times for Example 3.3

$n = k \times l$	*CPU TIME (msec)*		
	Proposed method	*Gauss elimination*	*Gauss–Seidel* double sweep*
10	10	20	20
30	30	460	180
50	40	2040	390

*The criteria for convergence for this method was that the Euclidean norm of the errors was less than 10^{-7}.

$$\begin{array}{r|cccccccccccccccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \\
\hline
1 & 1 &&&&&&&&&&&&&&&&&&&&& -1 && \\
2 && 1 &&&&&&&&&&&&&&&&&&&&& -1 & \\
3 &&& 1 &&&&&&&&&&&&&&&&&&&&& -1 \\
4 & -\frac{55}{90} &&& 1 &&&&&&&&&&&&&&& -\frac{55}{90} &&&&& \\
5 && -\frac{1}{2} &&& 1 &&&&&&&&&&&&&&& -\frac{1}{2} &&&& \\
6 & -\frac{35}{90} & -\frac{1}{2} & -1 &&& 1 &&&&&&&&&&&&& -\frac{35}{90} & -\frac{1}{2} & -1 &&& \\
7 &&&& -\frac{10}{11} &&& 1 &&&&&&&&&&&&&&&&& \\
8 &&&&& -\frac{6}{7} &&& 1 &&&&&&&&&&&&&&&& \\
9 &&&&&& -\frac{5}{11} &&& 1 &&&&&&&&&&&&&&& \\
10 &&&& -\frac{1}{11} &&&&&& 1 &&&&&&&&&&&&&& \\
11 &&&&& -\frac{1}{7} &&&&&& 1 &&&&&&&&&&&&& \\
12 &&&&&& -\frac{6}{11} &&&&&& 1 &&&&&&&&&&&& \\
13 &&&&&&& -\frac{3}{5} &&&&&& 1 &&&&&&&&&&& \\
14 &&&&&&&& -\frac{2}{3} &&&&&& 1 &&&&&&&&&& \\
15 &&&&&&&&& -\frac{4}{5} &&&&&& 1 &&&&&&&&& \\
16 &&&&&&&&&& -\frac{1}{7} &&& -\frac{1}{7} &&& 1 &&&&&&&& \\
17 &&&&&&&&&&& -\frac{1}{5} &&& -\frac{1}{5} &&& 1 &&&&&&& \\
18 &&&&&&&&&&&& -\frac{9}{10} &&& -\frac{9}{10} &&& 1 &&&&&& \\
19 &&&&&&& -\frac{2}{5} &&&&&&&&&&&& 1 &&&&& \\
20 &&&&&&&& -\frac{1}{3} &&&&&&&&&&&& 1 &&&& \\
21 &&&&&&&&& -\frac{1}{5} &&&&&&&&&&&& 1 &&& \\
22 &&&&&&&&&& -\frac{6}{7} &&& -\frac{6}{7} &&&&&&&&& 1 && \\
23 &&&&&&&&&&& -\frac{4}{5} &&& -\frac{4}{5} &&&&&&&&& 1 & \\
24 &&&&&&&&&&&& -\frac{1}{10} &&& -\frac{1}{10} &&&&&&&&& 1
\end{array}$$

Fig. 1

Gauss–Seidel double sweep method. (PL/I programming with double precision on an IBM 370/168 computer). The Gauss–Seidel double sweep method has been tested since Brandon (1974) suggested that this is the best method for the solution of this type of problem.

Comparison of the CPU time required is shown in **Table 1.** For $n = 50$ the proposed method is 50 times faster than another direct method and 10 times faster than an efficient iterative method.

A computer program for the solution of this example is given in Appendix 2.

4. Conclusions

The new method combines advantages of direct and indirect methods and results in considerable savings of computing time and storage for large sparse sets of linear systems and can be useful for the solution of partial differential equations and for linear programming. Essentially, the dimension of the system is reduced from n to m. In addition it is shown how the derivatives of the Newton method can be calculated directly from the functions.

The restructuring of the system of equations is not a deterring requirement. For diagonal band type matrices the variables of any one row (less one) can be taken as the tear variables in a general program of the proposed method.

We have also found that this method is efficient for the inversion of matrices.

$$
\begin{array}{r|ccc|ccccccccccccccccccccc}
 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 1 & 2 & 3 \\
\hline
7 & -\frac{10}{11} & & & 1 \\
8 & & -\frac{6}{7} & & & 1 \\
9 & & & -\frac{5}{11} & & & 1 \\
10 & -\frac{1}{11} & & & & & & 1 \\
11 & & -\frac{1}{7} & & & & & & 1 \\
12 & & & -\frac{6}{11} & & & & & & 1 \\
13 & & & -\frac{3}{5} & & & & & & & 1 \\
14 & & & & & -\frac{2}{3} & & & & & & 1 \\
15 & & & & & & -\frac{4}{5} & & & & & & 1 \\
16 & & & & & & & -\frac{1}{7} & & & -\frac{1}{7} & & & 1 \\
17 & & & & & & & & -\frac{1}{5} & & & -\frac{1}{5} & & & 1 \\
18 & & & & & & & & & -\frac{9}{10} & & & -\frac{9}{10} & & & 1 \\
19 & & & -\frac{2}{5} & & & & & & & & & & & & & 1 \\
20 & & & & & -\frac{1}{3} & & & & & & & & & & & & 1 \\
21 & & & & & & -\frac{1}{5} & & & & & & & & & & & & 1 \\
22 & & & & & & & -\frac{6}{7} & & & -\frac{6}{7} & & & & & & & & & 1 \\
23 & & & & & & & & -\frac{4}{5} & & & -\frac{4}{5} & & & & & & & & & 1 \\
24 & & & & & & & & & -\frac{1}{10} & & & -\frac{1}{10} & & & & & & & & & 1 \\
1 & & & & & & & & & & & & & & & & & & -1 & & & & 1 \\
2 & & & & & & & & & & & & & & & & & & & & -1 & & & 1 \\
3 & & & & & & & & & & & & & & & & & & & & & -1 & & & 1 \\
\hline
4 & 1 & & & & & & & & & & & & & & & -\frac{55}{90} & & & & & & -\frac{55}{90} \\
5 & & 1 & & & & & & & & & & & & & & & -\frac{1}{2} & & & & & & -\frac{1}{2} \\
6 & & & 1 & & & & & & & & & & & & & -\frac{35}{90} & -\frac{1}{2} & -1 & & & & -\frac{35}{90} & -\frac{1}{2} & -1
\end{array}
$$

Fig. 2

Appendix 1 PL/1 subroutine for the solution of a sparse system of linear equations

LINEQ.

```
/*****************************************************************************************/
/*                                                                                       */
/*      PURPOSE                                                                          */
/*            OBTAIN A SOLUTION OF A SPARSE SET OF LINEAR EQUATIONS                      */
/*      METHOD                                                                           */
/*            USING A SPECIALLY ORDERED SYSTEM WHICH IS A FUNCTION OF THE TEAR VARIABLES ONLY,  */
/*            A SMALLER SET OF LINEAR EQUATIONS IS OBTAINED. THIS SMALLER SET IS THEN SOLVED BY A  */
/*            STANDARD GAUSS ELIMINATION METHOD. (SEE FOR EXAMPLE: SUBROUTINE SIMQ FROM IBM,  */
/*            SYSTEM/360 SCIENTIFIC SUBROUTINE PACKAGE, FORM NO. GH20-0205-4, 1970)      */
/*      DESCRIPTION OF THE PARAMETERS                                                    */
/*            K—FIXED—NUMBER OF THE TEAR VARIABLES                                       */
/*            XX—(K)BIN(53)—RESULTANT SOLUTION OF THE TEAR VARIABLES.                    */
/*            ERR—BIN(53)—AN INDICATOR FOR THE ACCURACY OF THE SOLUTION. FOR A NORMAL EXIT ERR  */
/*                  CONTAINS THE EUCL. NORM OF THE FUNCTION VALUES AT THE SOLUTION FOR A */
/*                  SINGULAR SYSTEM ERR IS SET TO 1(ONE).                                */
/*            FUN—FUNCTION—CALCULATES THE FUNCTION VALUES FOR GIVEN TEAR VARIABLES       */
/*                  A N *N SYSTEM OF LINEAR EQUATIONS IS RECONSTRUCTED IN THE FOLLOWING FORM:  */
/*                  X(K + 1) = -(A(1, 1) *X(1) + A(1, 2) *X(2) + . . . + A(1, K) *X(K) - B(1))/A(1, K + 1)  */
/*                  X(K + 2) = -(A(2, 1) *X(1) + A(2, 2) *X(2) + . . . + A(2, K + 1) *X(K) - B(2))/A(2, K + 2)  */
/*                      :                                                                */
/*                  X(N) = -(A(N - K, 1) *X(1) + A(N - K, 2) *X(2) + . . .               */
/*                        . . + A(N - K, N - 1) *X(N - 1) - B(N - K))/A(N - K, N)        */
/*                  Y(1) = A(N - K + 1, 1) *X(1) + A(N - K, 2) *X(2) + . . .             */
/*                        . . + A(N - K + 1, N) *X(N) - B(N - K - 1)                     */
/*                        :                                                              */
/*                  Y(K) = A(N, 1) *X(1) + A(N, 2) *X(2) + . . . + A(N, N) *X(N) - B(N)  */
/*                  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  */
/*                  (X(1), X(2), . . ., X(K) ARE THE TEAR VARIABLES)                     */
/*                                                                                       */
/*                  USAGE                                                                */
/*                        CALL FUN(SG, XX, Y)                                            */
/*                        SG—BIT(1)—INDICATES THE VALUE THAT THE B VECTOR RECEIVES. FOR SG = '1'  */
/*                          B RECEIVES ITS REAL VALUE, ELSE IT IS SET AT ZERO.           */
/*                        XX—(K)BIN(53)—CURRENT VALUE OF THE TEAR VARIABLES.             */
/*                        Y—(K)BIN(53)—RESULTANT FUNCTION VALUE                          */
/*                                                                                       */
/*****************************************************************************************/

        PROC(K, XX, FUN, ERR);
DCL     (XX( *), P(K), DY(K, K), Y(K), ERR) BIN(53), K FIXED, ST BIT(1), FUN ENTRY(BIT(1), ( *)BIN(53), ( *)BIN(53)),
        DMLSQ ENTRY(( *,  *)BIN(53), ( *)BIN(53), FIXED, BIT(1));
/*      DMLSQ = STANDARD IBM METHOD, USING GAUSS ELIMINATION FOR THE SOLUTION OF A SYSTEM OF  */
/*      LINEAR EQUATIONS                                                                 */
/*      SET INITIAL VALUE OF XX TO 1                                                     */
XX = 1;
/*      CALCULATE THE FUNCTION VALUE                                                     */
CALL FUN('1'B, XX, Y);
/*      CALCULATE THE PARTIAL DERIVATIVES                                                */
DO I = 1 TO K; P = O; P(I) = XX(I); CALL FUN('O'B, P, DY( *, I)); DY( *, I) = DY( *, I)/XX(I);
END;
/*      CALCULATE THE EXACT SOLUTION                                                     */
IF K = 1 THEN XX(1) = XX(1) - Y(1)/DY(1, 1);
      ELSE DO; P = -Y; CALL DMLSQ(DY, P, K, ST);
           IF ST THEN ERR = 1; /*      SINGULAR SYSTEM                                   */
           ELSE DO; XX( *) = XX( *) + P( *); CALL FUN('1'B, XX, Y);
                ERR = SQRT(SUM(Y *Y));
END; END;
END LINEQ;
```

Appendix 2: Main program and function for the solution of example 3.3

```
LIN: PROCEDURE OPTIONS(MAIN:
/*      THIS MAIN PROGRAM SOLVES EXAMPLE 3.3 (FOR N = 50) BY THE PROPOSED METHOD.        */
DCL     FUN ENTRY(BIT(1), ( *)BIN(53), ( *)BIN(53)), ((XX, YY)(5), ERR) BIN(53), I FIXED,
        (X(50) BIN(53), (N, K) FIXED) EXTERNAL;
K = 5; N = 50; CALL LINEQ(K, XX, FUN, ERR);
CALL FUN('1'B, XX, YY);
              PUT EDIT(' * * * THE SOLUTION FOR N = ',N,' * * *')
                    (PAGE, A, F(2), A)(('X(',I, ') =' X(I) DO I = 1 TO N))
                    (SKIP, A, F(2), A, E(23, 15));
IF ERR > 1E-5 THEN PUT EDIT(' * * * THE ERROR IS ', ERR, ' * * *')
        (SKIP(5), A, E(12, 5), A);
END LIN;
```

```
FUN:   PROC(SG, XX, Y);
DCL    SG BIT(1), ((XX, Y)(*), B(50), C1, C2, C3) BIN(53),
       (I, KK) FIXED, (X(50) BIN(53), (K, N) FIXED) EXTERNAL;
IF SG THEN B = 1; ELSE B = 0;
DO I = 1 TO K; X(I) = XX(I); END; KK = 0;
DO I = 1 TO N; KK = KK + 1; C1, C2, C3 = 0;
       IF I > K THEN C1 = -X(I - K); IF KK > 1 THEN C2 = -X(I - 1);
       IF KK < K THEN C3 = -X(I + 1);
       IF I < = N - K THEN X(I + K) = C1 + C2 + 4*X(I) + C3 - B(I);
             ELSE Y(KK) = C1 + C2 + 4*X(I) + C3 - B(I);
       IF KK = K THEN KK = 0;
END; DCL YY(5) BIN(53); YY = Y; PUT DATA (X, YY);
END FUN;
```

References

BRANDON, D. M. Jr. (1974). The implementation and Use of Sparse matrix techniques in general simulation programs, *The Computer Journal*, Vol. 17, pp. 165-171.

CARNAHAN, B., LUTHER, H. A., and WILKES, J. O. (1969). *Applied Numerical Methods*, New York, Wiley.

DORN, W. S., and MCCRACKEN, D. D. (1972). *Numerical Methods with Fortran IV Case Studies*, New York: Wiley.

GASTINEL, N. (1970). *Linear Numerical Analysis*, Paris: Herman.

HARARY, F. (1971). Sparse Matrices and Graph Theory in Reid, J. K. (ed.), *Proc. Oxford Conf. of the Inst. of Mathematics and its Applications*, New York: Academic Press, pp. 139-150.

KRON, G. (1963). *Diacoptics, Piecewise Solution of Large Scale Systems*, New York: McDonald.

LEDET, W. P., and HIMMELBLAU, D. M. (1970). Decomposition Procedures for Solving of Large Scale Systems, *Advances in Chemical Engineering*, No. 8, New York: Academic Press, pp. 186-254.

REID, J. K. (1971). Large Sparse Sets of Linear Equations, in Reid, J. K. (ed.), *Proc. Conf. of the Inst. of Mathematics and its Applications*, New York: Academic Press.

WALSH, J. (1971). Direct and Indirect Methods, in Reid, J. K. (ed.), *Proc. Conf. of the Inst. of Mathematics and its Applications*, New York: Academic Press, pp. 41-74.

YOUNG, D. M. (1971). *Iterative Solutions of Large Linear Systems*, New York: Academic Press.

YOUNG, D. M. (1973). Iterative Solution of Large Systems of Linear Algebraic Equations with Sparse, Positive Definite Matrices, in Byron, G. D., and Hall, C. A. (eds.), *Numerical Solution of Systems of Non-Linear Algebraic Equations*, New York: Academic Press, pp. 101-156.