PUC

Series: Monographs in Computer Science
and Computer Applications

Nº 6/70

THE SPARSE SYSTEM

by

Cesar Simões Salim
and
Helene K. Salim

Computer Science Department - Rio Datacenter

CENTRO TÉCNICO CIENTÍFICO
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 209 — ZC-20
Rio de Janeiro — Brasil

1 - The sparse system is a set of subroutines that manipulates both sparse matrices and polynomials which are stored in the memory of the computer as linked lists. The system can be split conveniently into two distinct parts; the first concerns sparse matrices and the second concerns sparse polynomials; so these topics will be presented separately.

2 - The structure of sparse matrices in storage.
We shall say that a matrix is sparse when at most 1/3 of its elements are non-zero. The sparse system is applicable to many special types of matrices including band-type (tridiagonal) matrices, some triangular matrices, PERT - generated matrices, and so on.

Each element a (i, j) of a (n x m) matrix A will be stored in a three computer word cell, as shown in figure 1.

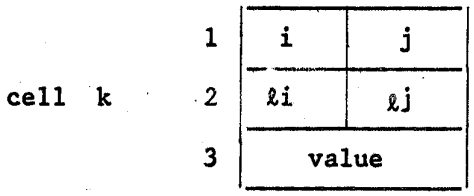


Figure 1

In word 1 we store both i and j, respectively the row and the column of a(i, j). In the second we have the two links and the third keeps the value associated to a(i, j).

The structure is a circularly linked list for each row and column, with a header cell for each row and column.

It can be easily set for both fixed-word length and variable length word computers. In the first case we have to write a simple function to pack and unpack the first two words and in the latter case, it's just a question of definition of size for integer and real variables.

The programs shown in this paper have been written for the B-3500, with 5 digits for integer constants and 12 for real ones, therefore no packing and unpacking has been necessary.

A header cell has the same structure as the others, but has a "flag" so it can be distinguished from common cells. As an example, the header cell for row n could be represented as follows:

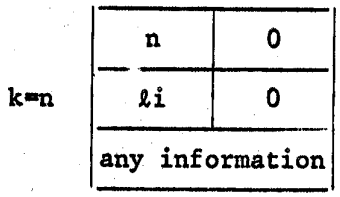


Figure 2

Note that the column index is zero. Conversely, the header cell for column n would have n for j-field and 0 for i-field.

In the programs of the system, the fields have been specified as: KEEP(K, J), and for cell K, we have for

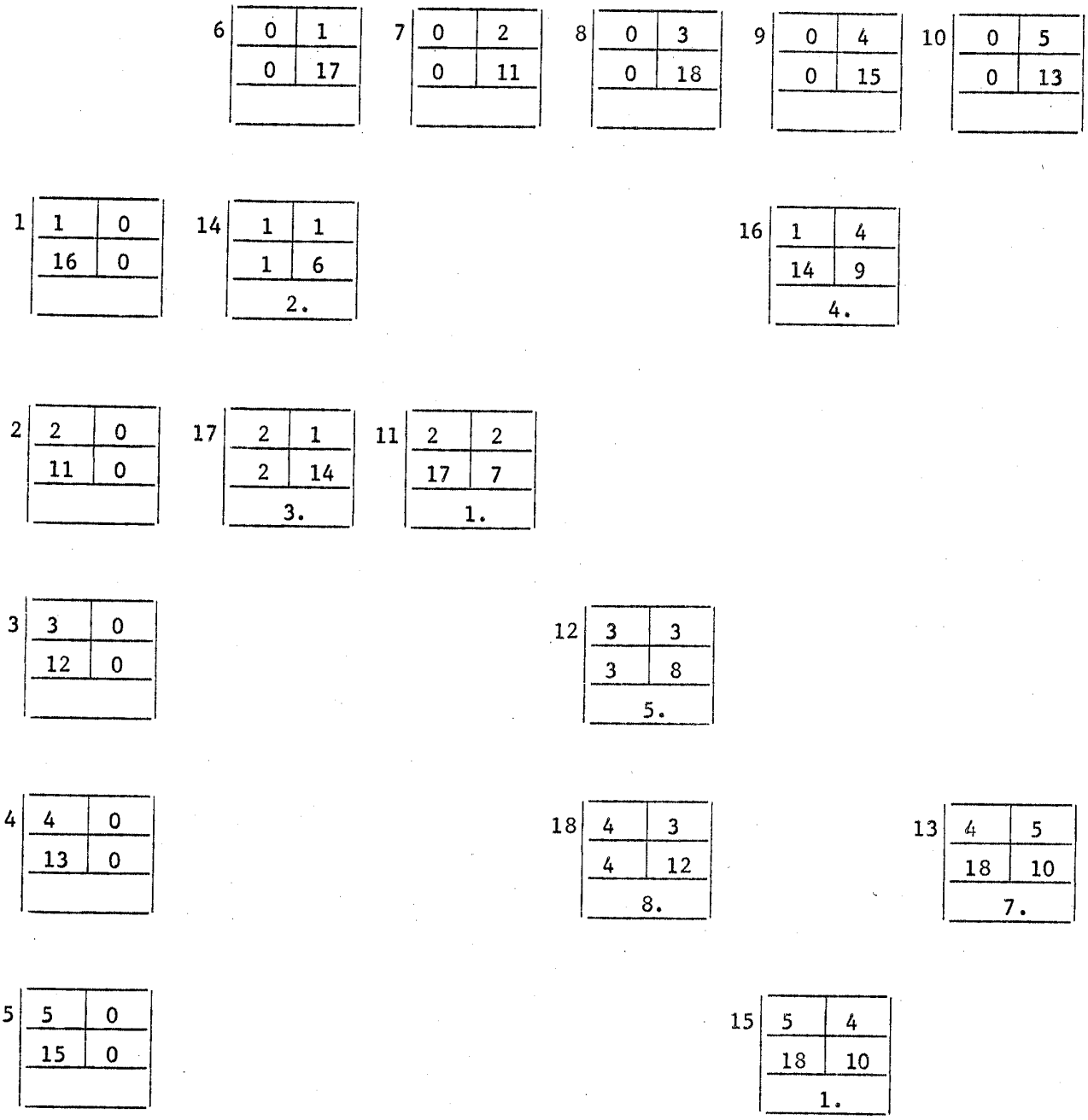
J = 1	→	i	word 1
J = 2	→	j	
J = 3	→	ℓi	word 2
J = 4	→	ℓj	
VALOR(K)	→	value	word 3

For didatical purposes we'll consider a cell for the header of row i and another one for column i. They can be put into one cell and the "flag" of the header cell would then be $K \leq n$. If this system is to be installed in a computer, this way of storing headers should be used instead of the more explicit form presented.

Example. Let A be the sparse matrix

$$A = \begin{vmatrix} 2 & 0 & 0 & 4 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 8 & 0 & 7 \\ 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

The internal structure will be:



Note also that each row link points to a cell whose "j" is smaller, except in the case of the header. Similarly for the column link, the "i"s are smaller, except when the link belongs to the header.

3. Parameters for the subroutines.

There are some parameters most commonly used in the system that will be outlined separately , in order to avoid repetition later.

The system affords working with 2 matrices simultaneously.

MATX - matrix that contains the structure of the sparse matrix given. Its dimension is (NTOT, 4), where

NTOT - dimension of the structure, serves both MATX and VALOR. Its value must obey the relation NTOT > m+n+e, where e is the number of non-zero elements of the sparse matrix.

VETOR - vector of values of corresponding elements in the sparse matrix.

M - number of rows of sparse matrix

N - number of columns of sparse matrix

IRG - parameter thar differentiates two matrices, when it is the case. Then the first one will correspond to irg=1 and the second to irg= -1.

ARG - multiplicative constant. Its use will be determined for each subroutine specifically.

IDISP - pointer to next available index in the structure for the first matrix.

IDIBM - same as idisp for the second matrix, when irg= -1.

IH - index of header of a row or column. If ih> m, then it is a column header or if ih≤m, a row header.

Note - Both matrices must be of the same dimension (NTOT, 4)

4. Description of the subroutine.

4.1 - SUBROUTINE HEAD (MATX, NTOT, M, N)

Creates the header cells for the structure. Must be the first subroutine called when the system is to be used for matrices.

4.2 - SUBROUTINE ENTRA (I, J, MATX, NTOT, M, IDISP, IRG, KA)

Inserts element a(i, j) in structure MATX (defines the links).

KA - index of the cell in the structure where element a(i, j) has been inserted. If KA=0, then the element is already present in the structure and hasn't been changed.

4.3 - SUBROUTINE LEMAT(MATX, VALOR, NTOT, N, M, IDISP, IRG)

Reads elements s(i, j) from punched cards, in the following format.

I, J, VALUE → 2I5, E14.8 Last card must be a zero in the 5^{th} column. An echo of input data is produced.

Subroutine ENTRA is required.

4.4 - SUBROUTINE SAIMAT (MATX, VALOR, NTOT, N, M, WORD)

Prints matrix stored in MATX, element by element, in the format WORD-- (I, J)= VALUE

WORD - name of the matrix to be printed. If may have at most 6 digits.

4.5 - SUBROUTINE DELETA (MATX, K, NTOT, IRG)

Deletes element of index K from the structure MATX. Storing K in a stack of available indexes (LUGAR).

4.6 - SUBROUTINE DEBUGA (MATX, VALOR, NTOT, IDISP, IQUI)

Prints contents of matrix structure MATX, as it is in storage.

IQUI - number that indicates the location of the call statement for the subroutine.

The following subroutines perform operations on the matrices.

4.7 - SUBROUTINE AMSBMA (MATX1, VALOR1, MATX2, VALOR2, ARG, NTOT, N, M, IDISP)

Calculates the sum of matrices of the form
MATX1 ←MATX1 + ARG * MATX2

Subroutines ENTRA and DELETA are required and IRG = 1.

4.8 - SUBROUTINE ADSUB (P, TEMP, X, M, ARG)

Calculates the vector.

P = TEMP + ARG * X

where P, TEMP, X are vectors of dimension M that have to be defined in the calling subroutine.

4.9 - SUBROUTINE CONSMT (MATX, NTOT, ARG, VALOR, IH, M)

Calculates the product of row or column ih by constant arg.

4.10- FUNCTION ESCAL (C, X, N)

Computes the dot product of vectors C and X where C and X are vectors of dimension N that have to be defined in the calling subroutine.

4.11- SUBROUTINE FILFIL (MATX, NTOT, VALOR, I, J, M, IRG, IDISP, C, TEMP)

Exchanges rows or columns i and j.
C and TEMP are auxiliary vectors of dimension M that have to

be defined in the calling subroutine.

Subroutines ENTRA and DELETA are required.

4.12- SUBROUTINE LINCOL (MATX, I, M, VALOR, NTOT, TEMP, C, IDISP, IRG)

a)- Exchanges row i and column i if $i \leq 0$.

b)- Exchanges elements from row and column i, such that $a(i, k) \rightarrow a(k, i)$ if $k > i$.

C and TEMP are auxiliary vectors of dimension M that have to be defined in the calling subroutine.

Subroutines ENTRA and DELETA are required.

4.13- SUBROUTINE MATMAT (MATX1, VALOR1, MATX2, VALOR2, NTOT, N, M, NIBM, IDIBM, C)

Computes the product of the matrices

MATX2 (M, NIBM) ← MATX1 (M, N) * MATX2 (N, NIBM)

where NIBM— number of columns of MATX2.

C is a vector of dimension NIBM that has to be defined in the calling subroutine.

Subroutines ENTRA and DELETA are required, for IRG= -1.

4.14- SUBROUTINE MATVET (MATX, VALOR, V, TEMP, NTRA, NTOT, N, M)

Sets vector TEMP to the product of the matrix stored in MATX and vector V —— TEMP = MATX * V

if NTRA ≠ 0, it will take the transpose of the stored matrix. V (Dimension N) and TEMP (Dimension M) have to be defined in the calling subroutine.

4.15- SUBROUTINE MAXFIL (MATX, VALOR, IH, KMAX, M, NTOT)

Sets VMAX to the element of maximum absolute value row or column ih. KMAX will be the index of the cell in the structure where VMAX has occured.

4.16- SUBROUTINE MAXMAT (MATX, VALOR, KMAX, VMAX, M, NTOT)

Sets VMAX to the element of maximum absolute value of the matrix MATX. KMAX will be the index of the cell in the structure where VMAX has occured.

Subroutine MAXFIL is required.

4.17- SUBROUTINE SISTEM(MATX, NTOT, VALOR, V, X, N, P, R, TEMP,TOL)

Sets X to the solution of the linear system

MATX * X = V

by the method of conjugate gradients.
P, TEMP and R are auxiliary vectors of dimension N that have to be defined in the calling subroutine.

X - initial guess to the solution, dimension N
V - independent vector for the system, dimension N

TOL - bound for the modulus of residual vector R = AX - V
subroutines MATVET, ADSUB and function ESCAL are required.

4.18- SUBROUTINE SUMFIL (MATX, NTOT, VALOR, M, ARG, IN, JH, IRG, IDISP)

Sets row or column ih to the sum of row or column ih and row or column jh multiplied by constant arg.

Subroutine ENTRA and DELETA are required.

4.19- SUBROUTINE TRANSP (MATX, M, VALOR, NTOT, TEMP, C, IDISP, IRG)

Substitutes matrix MATX by its transpose.
C and TEMP are auxiliary vectors of dimension M that have to be defined in the calling subroutine.
Subroutine LINCOL is required.

5. Presentation of the subroutines and examples. *

There will be a sample test for the routines and the listing of the program, except for the basic routines that are used by some others. (they are inplicitly tested).

As the reader will see in the test of subroutines MATMAT and AMSMBA, the result of the operations on sparse matrices may not be sparse; even so the system handles them, although there is a waste of core storage.

---

* Listings are available upon request from
Pontifícia Universidade Católica - Rio Datacentro.

6 - Structure of Sparse Polynomials in storage.

We shall say that a polynomial is sparse when it has at most 1/3 of mon-zero coefficients. For instance,p(x) is a sparse polynomial: $p(x) = 20x^{15} + 14x^{3} + 8^{x} - 1$

It is important to store the coefficient and the exponent of each element and we shall use a three computer word cell, as shown in figure 3.

COEFFICIENT	word 1
EXPONENT	word 2
LINK	word 3

Figure 3

In word 1 we store the coefficient, in word 2 the exponent and in the third one we are pointing to the next element of the polynomial. For instance, in p(x) we shall have:

1	20.	4	14.	7	8.	10	-1.
2	15	5	3	8	1	11	0
3	4	6	7	9	10	12	-13

Figure 4

We can denote the last element of the polynomial by the negative link, pointing to the next free element of available space.

As we use several polynomials in the same program, we can create a stack of available space. Also, corresponding to each stored polynomial, two items of information are kept:

1st - the index of first element of polynomial

2nd - the last element of polynomial points to the next free position in available space or to the next first element of a new polynomial.

The linkage of this available space is made by a special subroutine named LINKAR that must be called in the begining of the program.

We must have a pointer variable to the first free position in the stack of available space. In cases of insertion or deletion of an element in polynomial this variable must be changed to its new value.

In our programs we create three separate matrices for the available space: one containing the values of coefficients, other containing the exponents and the last one the links. It is very practial because the coefficients are floating point numbers and the exponents and links are integer numbers.

Therefore, in each program we have to dimension three vectors, so that all polynomials to be treated and the resulting polynomials can be stored in those vectors. At the same time we create the pointer variable to the first position (1) of the stack of available space and we call SUBROUTINE LINKAR at the beginning of the program.

7 - Description of the Subroutines.

We have written two kinds of subroutines: the first are essential or fundamental subroutines and the second are application sub - routines.

The fundamental subroutines are:

7.1 - LINKAR - it links the stack of available space.

7.2 - COPOLI and COPOLT - are two subroutines that copy a polynomial to another area of storage. COPOLI copies the polynomial beginning at the link of a given element and COPOLT copies the polynomial beginning at the element itself.

7.3 - INSERT - it serves to insert an element into or to delete one from a polynomial.

7.4 - ADDSUB - it adds or subtracts two polynomials.

7.5 - VALNUM - it determines the numerical value of a given polynomial p(x) for a given x.

7.6 - CTXPOL - it multiplies a constant by a polynomial.

7.7 - MULTIP - it multiplies one polynomial by another.

7.8 - DIVISU - it divides one polynomial into another.

7.9 - DERPOL - it finds the derivative of a polynomial.

7.10- INTPOL - it finds the integral of a polynomial.

7.11- READER - it reads a polynomial from cards.

7.12- WRITER - it prints the polynomial.

The set of application subroutines is:

7.13- POTPOL - it computes C-power of a given polynomial where C is an integer constant.

7.14- MDCPOL - it calculates the greater common divisor between two given polynomials.

7.15- ROOTS - it finds the roots of a given polynomial by Newton-Raphson method.

8 - Bibliographical References.

Donald E. Knuth - The Art of Computer Programming
vol. 1 - Addison - Wesley - 1968.

A. Ralston and Willf - Mathematical Methods for Digital Computers.
John Wiley - 1960.

Scientific Subroutine Package - IBM/360