Proceedings of the 5th WSEAS International Conference on Telecommunications and Informatics, Istanbul, Turkey, May 27-29, 2006 (pp110-115)

# Addressing Formulas for Central Triangular Matrices in 1D Arrays

FRANCISCO MORENO[1], SILVIA GUARDATI[2], OSVALDO CAIRÓ[2] & ROBERTO FLÓREZ[3]

(1) School of Systems Engineering
Facultad de Minas, Universidad Nacional de Colombia
Cra 80 65–223 Bloque M8, Oficina 209
Medellín, COLOMBIA

(2) Department of Computer Science
Instituto Tecnológico Autónomo de México
Río Hondo 1 – 01000 México DF, MÉXICO

(3) Department of Systems Engineering
Universidad de Antioquia
Calle 67 #53–108 Bloque 21, Oficina 429
Medellín, COLOMBIA

*Abstract*:– Many of the elements of sparse matrices have zero value. Known examples of these types of matrices are lower and upper triangular, which arise frequently in the solution of linear equation systems. Because many of the elements of these matrices are zero, it is advisable to store them –whether on disk or in memory– in a way that saves space. In this paper sparse matrices called central triangular are analyzed and mechanisms for their efficient storage by means of 1D arrays are proposed.

*Key-Words*: sparse matrices, addressing formulas, efficient storage

## 1 Introduction

Many of the elements of sparse matrices have zero value. Known examples of these types of matrices are lower and upper triangular, which are frequently used for the representation of linear equation systems [4]. Because many of the elements of these matrices are zero, it is advisable to store them –whether on disk or in memory– in a way that saves space.

The analysis of storage by means of 1D arrays for lower and upper triangular matrices and their corresponding addressing formulas has been made in [2, 5]. In [6] some variants of lower and upper triangular matrices are also analyzed. Nevertheless, it is very important to say that other types of sparse matrices including those with symmetry have remained unanalyzed.

In this paper we analyze storage by means of 1D arrays of sparse matrices of the central triangular type, plus one matrix of the rhombus type. In Section 2, the central triangular matrices are defined and analyzed. In Section 3, we analyze the Rhombus Matrix. Section 4 explains a couple of alternative methods to find addressing formulas of sparse matrices, and finally in Section 5 we present conclusions and suggestions for future work.

## 2 Central Triangular Matrices

In this section we analyze both Central Upper Triangular Matrix (CUTM) and Central Semi–Lower Inverse Triangular Matrix (CSLITM).

### 2.1. Central Upper Triangular Matrix

Let us define a *Central Upper Triangular Matrix* of order N, N > 0, and odd. Figure 1 shows an example of this kind of matrix, with N = 7.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **1** | | | | 30 | | | |
| **2** | | | 2 | 8 | 9 | | |
| **3** | | 11 | 3 | 4 | 7 | 8 | |
| **4** | 12 | 30 | 2 | 1 | 20 | 3 | 5 |
| **5** | | | | | | | |
| **6** | | | | | | | |
| **7** | | | | | | | |

Figure 1. Central Upper Triangular Matrix

The elements that are outside of the triangle are zeros and all or most of the elements in the triangle are different from zero. It must be said that this kind of matrix can be applied in different areas with different purposes, for instance:

a) In statistics in the application of the triangular distribution.
b) In microstructure for the recognition of triangular faults –defects– in images of diverse materials as steel.
c) In fractals in order to represent the *Sierpinski Triangle* or *Sierpinski Gasket* [1], which even has applications in electrical circuits and in microscopic patterns, as the case of DNA.
d) In mathematics in order to represent the Pascal Triangle.

### 2.1.1. Representation of a CUTM in a 1D Array

Because in this kind of matrix a lot of elements are zero, the choice is to use a 1D array to store only the values of the triangle in order to save space. These matrices can be represented in a 1D array *by rows*, from left to right. In Figure 2, we represent the matrix shown in Figure 1 using this mechanism.

Given an element (i, j) belonging to the triangle, we are interested in determining which position corresponds to this element in the 1D array.

The position of element (i, j) is determined in the following way.

***a*** = Total of elements belonging to rows that precede in the triangle to element (i, j).
***b*** = Column that occupies the element (i, j) in row i of the triangle –taking nonempty positions into account–.

Therefore the position of element (i, j) is given by: ***a + b***. For instance, if we take element (3,3) then ***a* = 4** and ***b* = 2** –position 6 in the 1D array of Figure 2– since the element (3,3) is in the second occupied column of row 3 of the triangle. Note that for an element located in row i, the value ***a*** corresponds to the sum of the first (i - 1) odd numbers.

On the other hand, value ***b*** is obtained by:

a) For the element (1,4) of row 1 that is located in the central column of the matrix ($\lceil N/2 \rceil$ = 4, the operation $\lceil x \rceil$ rounds up to the next integer greater than x), a total of 3 units should be subtracted from its column to get a corresponding value ***b***.
b) For the elements of row 2: (2,3), (2,4), and (2,5); 2 units from their respective columns should be subtracted to get a corresponding value ***b***.
c) For the elements of row 3: (3,2), (3,3), (3,4), (3,5), and (3,6); one unit of their respective columns should be subtracted to get a corresponding value ***b***.

Finally for the elements of row 4, subtraction is not necessary. This process can be seen in Table 1.

Table 1. Summary of the process to get value ***b*** in a CUTM

| | Values to substract from j to get b |
|---|---|
| Elements row i = 1, column j | $3 = 4 - 1 = \lceil N/2 \rceil - i$ |
| Elements row i = 2, column j | $2 = 4 - 2 = \lceil N/2 \rceil - i$ |
| Elements row i = 3, column j | $1 = 4 - 3 = \lceil N/2 \rceil - i$ |
| Elements row i = 4, column j | $0 = 4 - 4 = \lceil N/2 \rceil - i$ |

Therefore:

$$a = \sum_{h=1}^{i-1} (2h - 1) = i^2 - 2i + 1 = (i - 1)^2$$

$$b = j - (\lceil N/2 \rceil - i)$$

The position for an element (i, j) is given by:

$$Pos(i, j) = i^2 - i + j - \lceil N/2 \rceil + 1 \qquad (1)$$

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 30 | 2 | 8 | 9 | 11 | 3 | 4 | 7 | 8 | 12 | 30 | 2 | 1 | 20 | 3 | 5 |

Figure 2. Representation by rows of a CUTM

For the element (3,3) belonging to the matrix shown in Figure 1, formula (1) is applied:

$$\boldsymbol{Pos(i, j)} = 3^2 - 3 + 3 - \lceil 7/2 \rceil + 1 = 6$$

### 2.2. Central Semi–Lower Inverse Triangular Matrix

Let us now define a *Central Semi–Lower Inverse Triangular Matrix* of order N, N > 1, and odd. This type of matrix is used in applications of microstructure [3]. An example of this type of matrix (with N = 7) is shown in Figure 3.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | 34 | 20 | 43 | 3 | 6 | |
| 6 | | | 23 | 10 | 5 | | |
| 7 | | | | 1 | | | |

Figure 3. Central Semi–Lower Inverse Triangular Matrix

#### 2.2.1. Representation of a CSLITM in a 1D Array

The matrix of Figure 3 can be represented in a 1D array by rows as is shown in Figure 4.

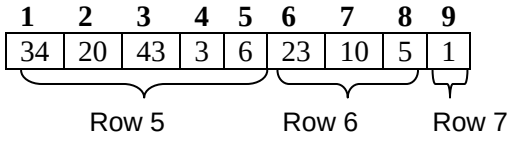


Figure 4. Representation by rows of a CSLITM

Using a method similar to that presented before, we can determine a formula for this kind of matrix. Nevertheless, the process to get ***a*** and ***b*** is a little more complicated. Let us have an element (i, j), then ***a*** represents the total of elements belonging to rows that precede in the triangle of the CSLITM to the element (i, j).

Therefore for the elements of row 5: (5,2), (5,3), (5,4), (5,5), and (5,6); they have ***a*** = 0 because there are no rows in the triangle which precede them. For the elements of row 6: (6,3), (6,4), and (6,5); ***a*** = 5, and for the element of row 7: (7,4); ***a*** = 5 + 3 = 8. This process can be seen in Table 2.

Table 2. Summary of the process to get value ***a*** in a CSLITM

| Row i ε CSLITM | ω = Total rows which precede to i in the CSLITM | Values to add in order to get *a* |
|---|---|---|
| 5 | 0 | 0 |
| 6 | 1 | 5 |
| 7 | 2 | 5 + 3 |

Now let us define ***t*** as the total of rows that separate row ***i*** of the central row $\lceil N/2 \rceil = 4$. For example, for ***i*** = 5, ***t*** = 1; for ***i*** = 6, ***t*** = 2, and for ***i*** = 7, ***t*** = 3. So we get:

$$\boldsymbol{t = i - \lceil N/2 \rceil}$$

Therefore ***w*** = ***t*** - 1 (see second column of Table 2), that is:

$$\boldsymbol{w = i - \lceil N/2 \rceil - 1}$$

Note that when ***w*** = 0, we should add 0 elements. When ***w*** = 1 we should add 5 elements, and when ***w*** = 2 we should add 5 + 3 elements (see third column of Table 2). Therefore:

$$\boldsymbol{a} = \sum_{h=1}^{w} c$$

Now the nature of component ***c*** should be determined. For the particular case of a matrix of order N = 7, sum begins with elements of row 5 (5 elements), afterward with the elements of the row 6 (3 elements), etc. Therefore the total elements of the first ***w*** rows of a CSLITM is given by the sum going back from odd numbers starting in N - 2. That is:

$$\boldsymbol{a} = \sum_{h=1}^{w} (N - (2h - 1) - 1) = \boldsymbol{w}(\mathrm{N} - \boldsymbol{w} - 1)$$

On the other hand, value ***b*** is obtained:

a) For the elements of row 5: (5,2), (5,3), (5,4), (5,5), and (5,6); one unit of their respective columns should be subtracted to get a corresponding value ***b***.
b) For the elements of row 6: (6,3), (6,4), and (6,5); 2 units of their respective columns should be subtracted to get a corresponding value ***b***.
c) Finally, for the element of the row 7: (7,4); 3 units of their respective columns should be subtracted to get a corresponding value ***b***.

This process can be seen in Table 3.

Table 3. Summary of the process to get value ***b*** in a CSLITM

| | **Value to subtract from j to get *b*** |
|---|---|
| Elements row i = 5, column j | 1 = 5 - 4 = i - ⌈N/2⌉ |
| Elements row i = 6, column j | 2 = 6 - 4 = i - ⌈N/2⌉ |
| Elements row i = 7, column j | 3 = 7 - 4 = i - ⌈N/2⌉ |

Therefore, $\boldsymbol{b}$ = j - (i - ⌈N/2⌉), then the position of an element (i, j) will be given by $\boldsymbol{a} + \boldsymbol{b}$:

$$\boldsymbol{Pos(i, j)} = \boldsymbol{w}(\mathrm{N} - \boldsymbol{w} - 1) + \mathrm{j} - \mathrm{i} + \lceil \mathrm{N}/2 \rceil \quad (2)$$

# 3 Rhombus Matrix

Now let us define a *Rhombus Matrix* of order N, N > 0, and odd. An example of this type of matrix (with N = 7) is shown in Figure 5. A Rhombus Matrix can be seen as a composition of a CUTM and a CSLITM.

This kind of matrix can be used to represent a fractal called *Peano Curve* [7]. Although based on abstract mathematics, fractals have practical applications in graphical computation, digitalization of images, and in the modeling of complex natural structures.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| **1** | | | | 30 | | | |
| **2** | | | 2 | 8 | 9 | | |
| **3** | | 11 | 3 | 4 | 7 | 8 | |
| **4** | 12 | 30 | 2 | 1 | 20 | 3 | 5 |
| **5** | | 34 | 20 | 43 | 3 | 6 | |
| **6** | | | 23 | 10 | 5 | | |
| **7** | | | | 1 | | | |

Figure 5. Rhombus Matrix

### 3.1. Representation of a Rhombus Matrix in a 1D Array

The matrix of Figure 6 can be represented in a 1D array by rows as is shown in Figure 7.

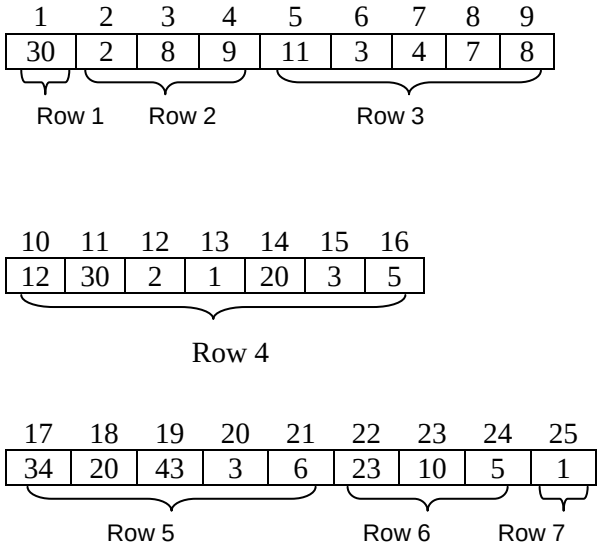


Figure 6. Representation by rows of a Rhombus Matrix

In this case, to find an addressing formula of an element (i, j) belonging to the Rhombus Matrix, we will use a method called *Composition*, described in the following section.

# 4 Alternative Methods for Obtaining Addressing Formulas

In this section we will analyze two methods: composition methods, and coordinate transformation method.

### 4.1. Composition Method

In order to get an addressing formula for a Rhombus Matrix we can use the formulas

corresponding to a CUTM and a CSLITM, since a Rhombus Matrix can be seen as a composition of these two types of matrices. Consider the element (i, j) ∈ Rhombus Matrix. We can perform this algorithm:

***Pos (i, j)***:
**if** (i ∈ CUTM)
  **then**
    To apply formula (1)
  **else**
    To apply formula (2) + Total of elements of a CUTM
**end–if**

The total of elements of a CUTM is:

$$\sum_{h=1}^{\lceil N/2 \rceil}(2h-1) = \lceil N/2 \rceil^2$$

which is indeed the sum of first ⌈N/2⌉ odd numbers. Therefore the formula is:

***Pos(i, j)***: (3)
**if** (i ∈ CUTM)
  **then**
    Pos = $i^2$ - i + j - ⌈N/2⌉ + 1
  **else**
    z = ⌈N/2⌉
    w = i - z - 1
    Pos = w(N - w -1) + j - i + z + $z^2$
**end–if**

It is possible to get a straight formula for the Rhombus Matrix without using such composition technique. This formula is:

***Pos(i, j)*** = $i^2$ - i + j - ⌈N/2⌉ + 1 - ⌊2i/(N + 3)⌋ * $(2i - N - 1)^2/2$ (4)

where ⌊x⌋ truncates the decimal part of x.

Due to its complexity we will not show the process to get formula (4). In this case, although *the direct* formula (4) is better in the sense that it does not require a decision structure (**if–else**), its disadvantage is its complex meaning and the difficult process needed to get it.

A second technique to get addressing formulas of some sparse matrices is shown in Section 4.2.

## 4.2. Coordinate Transformation Method

Let us now define a *Central Lower Triangular Matrix* (CLTM) of order N, N > 0, and odd. An example of this kind of matrix (with N = 7) is shown in Figure 7.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | 1 | | | |
| 5 | | | 55 | 43 | 66 | | |
| 6 | | 56 | 23 | 10 | 5 | 69 | |
| 7 | 23 | 34 | 11 | 19 | 2 | 65 | 11 |

Figure 7. Central Lower Triangular Matrix

We can represent the matrix of Figure 7 in a 1D array by rows as is shown in Figure 8.

Although a formula for a CLTM can be found using a similar method as the one presented before, we can indeed get an addressing formula for a CLTM making a coordinate transformation with the CUTM. Table 4 shows this mapping.

From Table 4 the following mapping can be set:

a) Column ∈ CUTM = Column ∈ CLTM
b) Row ∈ CUTM = Row ∈ CLTM - ⌊N/2⌋

Table 4. Mapping between some elements CUTM and a CLTM

| **Element ∈ CLTM** | **Element corresponding in a CUTM** | **Position in the 1D array** |
|---|---|---|
| (4,4) | (1,4) | 1 |
| (5,3) | (2,3) | 2 |
| (5,4) | (2,4) | 3 |
| (5,5) | (2,5) | 4 |
| (6,2) | (3,2) | 5 |

Therefore the addressing formula of a CLTM is:

***Pos(i, j)*** = $(i')^2$ - i' + j - ⌈N/2⌉ + 1 (5)

with $i'$ = i - ⌊N/2⌋.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 55 | 43 | 66 | 56 | 23 | 10 | 5 | 69 | 23 | 34 | 11 | 19 | 2 | 65 | 11 |

Figure 8. Representation by rows of a CLTM

# 5 Conclusions

Although memory and disk space is cheaper everyday, it is important to develop methods that using these resources in an efficient way. In this paper we presented different types of sparse matrices with both their areas of application and their representation in a 1D array in order to save space.

This saving of space is possible because we store only those elements different from zero and because the matrices analyzed have some kind of symmetry. The corresponding addressing formulas were also deduced. They allow access to the stored information. We also presented a method to find addressing formulas for sparse matrices that are a combination of other types of matrices, for example a Rhombus Matrix can be seen as a combination of two central triangular matrices. Finally, a method of coordinate transformation was analyzed. It allows addressing formulas of sparse matrices that have some kind of displacement (whether in x−axis, y−axis, or even in both) with others matrices already analyzed, to be obtained.

In future work, we hope to analyze other types of sparse matrices that have some type of symmetry. We also want to develop, alternative methods, which help obtain addressing formulas of sparse matrices.

*References*

[1] Barnsley, M. *Fractals Everywhere*, Academic Press Professional, 1993.

[2] Cairó O. & Guardati, S. *Data Structures*, McGraw−Hill, 2005.

[3] Göken M. *Studies of Metallic Surfaces and Microstructures with Atomic Force Microscopy*, Digital Instruments Santa Barbara, Santa Barbara, California, USA, 1998. http://www.veeco.com/appnotes/AN28_Metallic_082004_RevA1.pdf

[4] Grossman S. *Álgebra Lineal*, McGraw Hill, 1996.

[5] Horowitz E. & Sahni S. *Fundamentals of data Structures in Pascal*, Computer Science Press, 1990.

[6] Moreno F. & Flórez R. *Fórmulas de Direccionamiento en Matrices Triangulares*, Journal Facultad de Ingeniería Universidad de Antioquia. Medellín, 2001.

[7] Ting C. & Liming H. *World of Fractal*, Technical Report, National University of Singapur, 2005. www.math.nus.edu.sg/aslaksen/gem-projects/maa/WorldofFractal.pdf