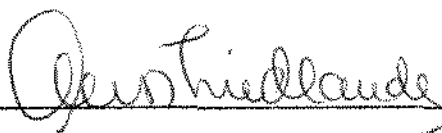


TECNICAS DE RESOLUÇÃO DE SISTEMAS LINEARES  
ESPARSOS E ATUALIZAÇÃO DE MATRIZES ESPARSAS  
PARA USO EM ALGORITMOS DE OTIMIZAÇÃO.

Este exemplar corresponde à redação final da  
tese devidamente corrigida e defendida pela  
Srta. Maria Cristina Wolff e aprovada pela  
Comissão Julgadora.

Campinas, 13 de Novembro de 1989.



Profa. Dra. Ana Friedlander de Martínez Pérez

Dissertação apresentada ao Instituto de  
Matemática, Estatística e Ciência da  
Computação, UNICAMP, como requisito  
parcial para obtenção do Título de  
Mestre em Matemática Aplicada, área:  
Otimização e Pesquisa Operacional.

W832t

11655/BC

UNICAMP  
BIBLIOTECA CENTRAL

**TÉCNICAS DE RESOLUÇÃO DE SISTEMAS LINEARES  
ESPARSOS E ATUALIZAÇÃO DE MATRIZES ESPARSAS  
PARA USO EM ALGORITMOS DE OTIMIZAÇÃO.**

*Maria Cristina Wolff*

A minha família

## Agradecimentos

A minha orientadora, Profa. Dra. Ana Friedlander de Martínez Pérez, pela competente orientação na realização deste trabalho.

Aos amigos do curso de mestrado em Matemática Aplicada (1985) - alguns distantes, outros ausentes - que proporcionaram uma convivência inesquecível.

Aos professores do Departamento de Matemática Aplicada do IMECC que, de algum modo, tenham contribuído na realização deste trabalho.

Aos meus amigos Cristina Lúcia Dias Vaz, Ana Cecilia Durán e Edson Rodrigues Carvalho, pela preciosa amizade, carinho e compreensão ao longo de todos estes anos.

A todos amigos que conheci no decorrer do período do mestrado - ligados ou não a ele - pelo enriquecimento mútuo desfrutado.

Ao CNPq, ao CAPES e à FAPESP, pelo auxílio financeiro durante o curso do mestrado. Em especial, agradeço à FAPESP, pelo acompanhamento criterioso no decorrer do trabalho.

---

## ÍNDICE

---

|                 |   |
|-----------------|---|
| INTRODUÇÃO..... | 1 |
|-----------------|---|

---

### CAPÍTULO I : MÉTODOS NUMÉRICOS PARA RESOLUÇÃO DE SISTEMAS DE EQUAÇÕES LINEARES

---

|   |    |
|---|----|
| 1 . INTRODUÇÃO.....                       | 1  |
| 2 . ELIMINAÇÃO GAUSSIANA.....             | 2  |
| 3 . FATORAÇÃO L U.....                    | 9  |
| 4 . CÁLCULO DA INVERSA DE UMA MATRIZ..... | 13 |
| 5 . CONSIDERAÇÕES NUMÉRICAS.....          | 18 |

---

### CAPÍTULO II : ALGORITMOS PARA OBTER ESTRUTURAS ADEQUADAS PARA DECOMPOSIÇÃO L U E RESOLUÇÃO DE SISTEMAS LINEARES

---

|                               |    |
|-------------------------------|----|
| 1 . INTRODUÇÃO.....           | 26 |
| 2 . PIVÔS.....                | 28 |
| 3 . ESTRUTURA.....            | 31 |
| 4 . MÉTODOS.....              | 36 |
| 4.1. MÉTODO DE MARKOWITZ..... | 37 |

|   |    |
|---|----|
| 4.2. MÉTODO DO PIVÔ PRÉ-DETERMINADO ( $P^3$ ) E MÉTODO PARTICIONADO DO PIVÔ PRÉ-DETERMINADO ( $P^4$ ).....  | 46 |
| 4.3. MÉTODO UTILIZANDO DOIS ESTÁGIOS: ALGORITMO PARA OBTER TRANSVERSAL MÁXIMA + ALGORITMO DE TARJAN. /..... | 78 |

---

## CAPÍTULO III : ESQUEMAS DE ATUALIZAÇÃO DE BASES

---

|   |     |
|---|-----|
| 1 . PROBLEMA DE PROGRAMAÇÃO LINEAR.....     | 97  |
| 1.1. MÉTODO SIMPLEX REVISADO.....           | 98  |
| 2 . ESQUEMAS PARA ATUALIZAÇÃO DE BASES..... | 103 |
| 2.1. MÉTODO DE BARTELS E GOLUB.....         | 103 |
| 2.1.1. IMPLEMENTAÇÃO DE SAUNDERS.....       | 109 |
| 2.1.2. IMPLEMENTAÇÃO DE REID.....           | 116 |
| 2.2. MÉTODO DE FORREST E TOMLIN.....        | 124 |

---

## CAPÍTULO IV : TÉCNICAS DE INVERSÃO E ATUALIZAÇÃO DE BASES EM PACOTES COMPUTACIONAIS

---

|                           |     |
|---------------------------|-----|
| 1 . INTRODUÇÃO.....       | 131 |
| 2 . MPSX.....             | 132 |
| 3 . MINOS.....            | 145 |
| 3.1. MINOS/AUGMENTED..... | 146 |
| 3.2. MINOS 5.0.....       | 149 |

---

## APENDICE

---

IMPLEMENTAÇÃO COMPUTACIONAL.....153

TESTES.....153

---

BIBLIOGRAFIA.....163

---

## INTRODUÇÃO

Um dos objetivos na realização desta tese foi o estudo de métodos para obter uma sequência de pivôs, de modo que fosse conseguida uma boa estrutura para uma matriz esparsa. Esta estrutura seria uma estrutura triangular inferior ou bloco triangular inferior. Considerando um sistema de equações lineares, uma destas estruturas simplificaria sua resolução, pois haveria diminuição nas necessidades de armazenamento e nas operações aritméticas ao realizarmos a eliminação Gaussiana.

Foram analisados os métodos de Markowitz, Hellerman e Rarick ( $P^3$  e  $P^4$ ) e um algoritmo de dois estágios (transversal máxima+Tarjan). Estes métodos são utilizados para ordenar linhas e colunas de matrizes não-simétricas, a fim de preservar a esparsidade das matrizes na fatoração LU. O método de Markowitz difere bastante dos demais mencionados. Pode-se dizer que a escolha de pivô é realizada de forma dinâmica, a cada estágio da eliminação Gaussiana. Nos outros métodos, é escolhida uma sequência de pivôs, à priori.

Em seguida, foram analisados métodos eficientes para atualização da inversa de uma base. Considerando que a base é decomposta em fatores triangulares, os métodos mais conhecidos para a atualização destes fatores são os métodos de Bartels e Golub e o de Forrest e Tomlin. O método de Bartels e Golub é o único esquema estável para atualização de fatores triangulares, embora apresente dificuldades na implementação computacional para problemas de grande porte.



O método de Forrest e Tomlin é atrativo dos pontos de vista da esparsidade e implementação computacional. Porém, é numericamente instável.

Existem implementações estáveis baseadas no método de Bartels e Golub, tais como as implementações de Saunders e de Reid, que também foram estudadas.

O próximo passo foi analisar os itens acima em pacotes computacionais: MINOS e MPSX.

No pacote MPSX/370, a estratégia de escolha de pivôs é semelhante à de Markowitz, porém não tão eficiente. A atualização dos fatores triangulares da base é feita através do método de Forrest e Tomlin.

Com relação ao pacote MINOS, foram estudadas duas versões. A mais antiga é a MINOS/AUGMENTED. Nela, são utilizados dois algoritmos para encontrar uma sequência de pivôs (transversal máxima + Tarjan), de modo que a estrutura final da base é bloco triangular inferior (em alguns casos). A atualização dos fatores triangulares é realizada através do esquema proposto por Saunders. Já o pacote MINOS 5.0, tem certas rotinas de seu antecessor substituídas. A escolha de pivôs e atualização da fatoração LU da base é feita por meio do método descrito e implementado por Reid (baseado no método de Bartels e Golub).

Houve um especial interesse no caso do estudo da estrutura bloco triangular inferior. Além dela ser interessante na resolução de sistemas de equações, (que podem ser resolvidos como uma sequência de os métodos para obtê-la eram usados no pacote MINOS/AUGMENTED.

Assim, os dois estágios para obtenção de tal estrutura (algoritmos de: transversal máxima + Tarjan), foram estudados com detalhes. Pelo interesse que o método despertou, foi realizada uma adaptação da implementação computacional do método e realizados testes com diversos tipos de matrizes (não-simétricas, esparsas, geradas aleatoriamente). Os resultados obtidos com relação à estrutura foram considerados satisfatórios. Pôde-se constatar que o método é extremamente eficiente.

# CAPÍTULO I : MÉTODOS NUMÉRICOS PARA RESOLUÇÃO DE SISTEMAS DE EQUAÇÕES LINEARES

## 1 - INTRODUÇÃO

Consideremos o seguinte sistema de equações lineares :

$$A x = b \quad (1)$$

onde  $A \in \mathbb{R}^{n \times n}$  é uma matriz não-singular,  $x \in \mathbb{R}^{n \times 1}$  e  $b \in \mathbb{R}^{n \times 1}$ .

A eliminação Gaussiana, o algoritmo de Crout, o algoritmo de Doolittle e a fatoração LU são métodos diretos para resolver (1). Todos são algebricamente equivalentes, com pequenas diferenças na sequência dos cálculos.

Basicamente, o objetivo geral de todos os métodos diretos de resolução de (1) é a obtenção de sistemas triangulares de equações que são "mais fáceis" de resolver.

Eles podem ter a forma:

$$U x = c \quad (2)$$

onde  $U$  é uma matriz triangular superior, ou

$$L y = b \quad (3)$$

onde  $L$  é uma matriz triangular inferior.

Ou seja, o objetivo dos métodos para resolver (1) é conseguir obter sistemas do tipo (2) ou (3), que podem ser resolvidos rapidamente.

## 2 - ELIMINAÇÃO GAUSSIANA

O método mais usado para se resolver sistemas de equações lineares e para inverter matrizes é devido a Gauss. Este método consiste em reduzir um sistema de  $n$  equações e  $n$  variáveis :

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{array} \right. \quad (1)$$

a um sistema de  $(n-1)$  equações e  $(n-1)$  variáveis. Utiliza-se a "primeira equação" ( se  $a_{11} \neq 0$  ) para eliminar as  $(n-1)$  primeiras variáveis das  $(n-1)$  equações restantes. Em seguida, a "segunda nova equação" para eliminar as  $(n-2)$  segundas variáveis das  $(n-2)$  equações e assim sucessivamente, até se obter uma equação com uma única variável. Procedendo desta forma, o sistema (1) é reduzido a um sistema triangular superior.

Encontrando o valor da última variável, substitui-se seu valor na equação precedente, determinando o valor de outra variável. Deste modo, os valores de todas variáveis vão sendo obtidos sucessivamente. Este processo é conhecido como retro-substituição (back-substitution).

Portanto, a eliminação Gaussiana consiste na redução de um sistema de equações a um sistema triangular superior, seguido de retro-substituição.

## NOTAÇÃO

Considere  $A \in \mathbb{R}^{n \times n}$ . É conveniente usar a notação:

$$A^{(i)} x = b^{(i)}$$

para o sistema obtido após  $(i-1)$  estágios da Eliminação Gaussiana,  $i=1,2,\dots,n$ , com  $A^{(1)} = A$  e  $b^{(1)} = b$ .

Ao final do processo,  $A^{(n)}$  é triangular superior.

## EXEMPLO

Consideremos :

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$A x = b \quad (1)$$

Com  $A^{(1)} = A$  e  $b^{(1)} = b$ , o sistema (1) pode ser escrito como:

$$A^{(1)} x = b^{(1)}$$

Este é o primeiro estágio da eliminação Gaussiana. Supõe-se que  $a_{11} \neq 0$  (esse elemento é chamado de pivô).

Multiplicando a primeira equação por  $a_{21}/a_{11}$  e subtraindo da segunda equação:

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(1)} \end{pmatrix}$$

onde :

$$a_{22}^{(2)} = a_{22}^{(1)} - (a_{21}^{(1)} / a_{11}^{(1)}) * a_{12}^{(1)}$$

$$a_{23}^{(2)} = a_{23}^{(1)} - (a_{21}^{(1)} / a_{11}^{(1)}) * a_{13}^{(1)}$$

$$b_3^{(1)} = b_3^{(1)} - (a_{31}^{(1)} / a_{11}^{(1)}) * b_1^{(1)}$$

Multiplicando a primeira equação por  $a_{31}/a_{11}$  e subtraindo da terceira equação :

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(2)} \end{pmatrix} \quad (2)$$

O sistema (2), que corresponde ao segundo estágio da eliminação Gaussiana, pode ser escrito como :

$$A^{(2)} x = b^{(2)}$$

No terceiro estágio, supondo  $a_{22}^{(2)} \neq 0$ , de modo análogo, tem-se :

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & 0 & a_{33}^{(3)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(3)} \end{pmatrix}$$

$$A^{(3)} x = b^{(3)}$$

Usando retro-substituição, obtém-se a solução do sistema de equações original.

Considerando  $A \in \mathbb{R}^{n \times n}$ . Para  $k$  estágios,  $k = 1, 2, \dots, n$ , temos :

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \left( a_{ik}^{(k)} / a_{kk}^{(k)} \right) \cdot a_{kj}^{(k)} \quad i, j > k$$

$$b_i^{(k+1)} = b_i^{(k)} - \left( a_{ik}^{(k)} / a_{kk}^{(k)} \right) \cdot b_k^{(k)} \quad i > k$$

onde  $a_{ij}^{(1)} = a_{ij}$ ,  $i, j = 1, 2, \dots, n$  e supõe-se que  $a_{kk}^{(k)} \neq 0$ ,  $k = 1, \dots, n$ .

Os elementos  $a_{kk}^{(k)}$  são chamados de pivôs.

Deve ser observado que não há necessidade de ser escolhida a  $k$ -ésima equação para eliminar  $x_k$  das outras equações de um sistema, no  $k$ -ésimo estágio da eliminação Gaussiana. Pode haver necessidade de mudança de linhas, se o elemento pivô for  $a_{kk}^{(k)} = 0$  neste estágio. Este inconveniente pode ser solucionado através da troca desta equação, por outra qualquer, tal que  $a_{ik}^{(k)} \neq 0$ ,  $i = k+1, \dots, n$ .

Este não é o único caso em que deve haver troca entre linhas. A escolha de pivôs, cujos valores absolutos sejam muito pequenos em relação aos outros elementos da matriz, pode ocasionar problemas de instabilidade numérica no processo de eliminação Gaussiana. Este caso será visto posteriormente.

## DEFINIÇÕES

### ( 1 ) - OPERAÇÕES ELEMENTARES

Na resolução de um sistema de equações, podem ser utilizadas as seguintes operações elementares sobre as linhas de uma matriz.

#### ( 1.a ) permutação das i-ésima e j-ésima linhas.

Consideremos uma matriz  $A \in \mathbb{R}^{n \times n}$ . A permutação entre as linhas i e j da matriz A, pode ser obtida através da multiplicação pela matriz Q, ou seja,  $Q A$ , onde Q é uma matriz de permutação, da forma :

$$Q = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 0 & \dots & 1 \\ & & 1 & \dots & 0 \\ & & & & \ddots & & 1 \end{bmatrix} \begin{matrix} \Rightarrow \text{linha } i \\ \Rightarrow \text{linha } j \end{matrix}$$

coluna i ←
→ coluna j

( 1.b ) multiplicação de uma linha por um escalar não-nulo  $c \in \mathbb{R}$  ( ou  $\mathbb{C}$  ).



A matriz é do tipo ,

$$Q = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & c & \\ & & & \ddots \\ & & & & 1 \end{bmatrix}$$

( 1.c ) substituição da i-ésima linha por :

( i-ésima linha + c \* ( j-ésima linha ) ),  $c \in \mathbb{R}$  ( ou  $\mathbb{C}$  ).

$$Q = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & \dots & c \\ & & 0 & \ddots & 1 \\ & & & & \ddots \\ & & & & & 1 \end{bmatrix}$$

( 2 ) - MATRIZ TRIANGULAR INFERIOR ELEMENTAR

Consideremos  $M \in \mathbb{R}^{n \times n}$  . Uma matriz triangular inferior elementar de ordem  $n$  e índice  $k$  [38] , é uma matriz da forma :

$$M = I - m e_k^T, \quad k=1,2,\dots,n$$

onde :

$e_k$ : indica o vetor coluna canônico de  $\mathbb{R}^n$ , com o valor 1 na posição  $k$ .

$$m = ( 0, 0, \dots, 0, \mu_{k+1}, \dots, \mu_n )^T$$

$I$  : matriz identidade de  $\mathbb{R}^{n \times n}$

$$M = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & -\mu_{k+1} & \ddots \\ & & -\mu_n & & 1 \end{bmatrix}$$

As matrizes triangulares inferior elementares são facilmente invertidas. Suas inversas são dadas por :

$$M^{-1} = I + m e_k^T$$

### ( 3 ) - MATRIZES DE TRANSFORMAÇÕES ELEMENTARES

Dada  $A \in \mathbb{R}^{n \times n}$ . Uma matriz de transformação elementar  $E_k$ , é tal que :

$$E_k = \begin{bmatrix} 1 & \dots & \eta_{1k} & & \\ & \ddots & \vdots & & \\ & & \eta_{kk} & & \\ & & \vdots & \ddots & \\ & & \eta_{nk} & & 1 \end{bmatrix}$$

Em particular, estamos interessados nas matrizes elementares, onde:

$$\begin{cases} \eta_{kk} = 1/a_{kk} \\ \eta_{ik} = -a_{ik}/a_{kk} \quad i=1,2,\dots,n \text{ e } i \neq k \end{cases}$$

Ao multiplicarmos  $E_k A$ , temos :

$$E_k A = [ a_1' \dots a_{k-1}' e_k a_{k+1}' \dots a_n' ]$$

onde  $e_k = ( 0, 0, \dots, 0, 1, 0, \dots, 0 )$   
 $\downarrow$   
 k-ésima posição

$a_j'$ ,  $j = 1, \dots, n$ ,  $j \neq k$  : colunas de  $A$ , modificadas.

### 3 - FATORAÇÃO LU

Consideremos  $A \in \mathbb{R}^{n \times n}$  tendo todos os menores principais não-nulos [19], ou seja :

$$\det |a_{11}| \neq 0, \det \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \neq 0, \dots, \det A \neq 0.$$

A fatoração da matriz  $A$  como o produto  $LU$ , onde

$L$  : matriz triangular inferior

$U$  : matriz triangular superior

é a idéia básica do esquema de eliminação Gaussiana.

O sistema  $Ax = b$  pode ser escrito como :

$$LUx = b$$

podendo ser representado por dois sistemas triangulares:

$$Ly = b$$

$$Ux = y$$

que são resolvidos facilmente.

Na eliminação Gaussiana, temos para  $k$  estágios,  $k=1,2,\dots,n-1$ :

$$\begin{cases} a_{ij}^{(k+1)} = a_{ij}^{(k)} - \left( a_{ik}^{(k)} / a_{kk}^{(k)} \right) * a_{kj}^{(k)} & i, j > k \\ b_i^{(k+1)} = b_i^{(k)} - \left( a_{ik}^{(k)} / a_{kk}^{(k)} \right) * b_k^{(k)} & i > k \end{cases}$$

Supondo que temos todos  $a_{rr}^{(r)} \neq 0$ ,  $r = 1,2,\dots,n$ , e definindo

$m_{ik}$  para  $i > k$  :

$$m_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}$$

temos para  $k = 1,2,\dots,n-1$  :

$$\begin{cases} a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)} & i, j > k \\ b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)} & i > k \end{cases} \quad (1)$$

Considerando a matriz triangular inferior elementar :

$$M_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -m_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & -m_{n,k} & & & 1 \end{bmatrix}$$

A relação (1) pode ser escrita como :

$$A^{(k+1)} = M_k A^{(k)}$$

Usando esta relação para  $k=1, 2, \dots, n-1$ , obtém-se :

$$U = A^{(n)} = M_{n-1} \cdot M_{n-2} \cdot \dots \cdot M_2 \cdot M_1 \cdot A^{(1)}$$

ou

$$U = M_{n-1} \cdot M_{n-2} \cdot \dots \cdot M_2 \cdot M_1 \cdot A \quad (2)$$

A inversa de  $M_k$  é dada por:

$$M_k^{-1} = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & m_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & m_{n,k} & & & 1 \end{bmatrix} \quad (3)$$

Multiplicando (2) por  $M_{n-1}^{-1}, M_{n-2}^{-1}, \dots$ , até  $M_1^{-1}$  :

$$A = M_1^{-1} \cdot M_2^{-1} \cdot \dots \cdot M_{n-1}^{-1} \cdot U \quad (4)$$

O produto das matrizes do tipo de (3) é :

$$M_1^{-1} \cdot M_2^{-1} \dots M_{n-1}^{-1} = \begin{bmatrix} 1 & & & \\ m_{21} & 1 & & \\ m_{31} & m_{32} & 1 & \\ \vdots & \vdots & & \ddots \\ m_{n1} & m_{n2} & m_{n3} & \dots & 1 \end{bmatrix} = L$$

Ou seja :

$$\prod_{i=1}^{n-1} M_i^{-1} = L$$

Portanto, (4) pode ser escrito como a fatora  o triangular:

$$A = L U$$

EXEMPLO

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 1 & -1 \\ 3 & -3 & 2 \end{bmatrix} = A^{(1)}$$

C  culo de  $M_1$

$$M_1 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$

$$A^{(2)} = M_1 \cdot A^{(1)}$$

$$A^{(2)} = \begin{bmatrix} 1 & 4 & 7 \\ 0 & -7 & -15 \\ 0 & -15 & -19 \end{bmatrix}$$

Cálculo de  $M_2$  :

$$M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -15/7 & 1 \end{bmatrix}$$

$$A^{(3)} = M_2 \cdot M_1 \cdot A^{(1)}$$

$$A^{(3)} = \begin{bmatrix} 1 & 4 & 7 \\ 0 & -7 & -15 \\ 0 & 0 & 92/7 \end{bmatrix}$$

Assim, tem-se que :

$$A^{(3)} = M_2 \cdot M_1 \cdot A^{(1)} = U$$

A matriz  $L$  é calculada através de :

$$M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \quad M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 15/7 & 1 \end{bmatrix}$$

Logo:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 15/7 & 1 \end{bmatrix}$$

Assim :

$$L U = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 15/7 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 7 \\ 0 & -7 & -15 \\ 0 & 0 & 92/7 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 1 & -1 \\ 3 & -3 & 2 \end{bmatrix} = A$$

#### 4 - CÁLCULO DA INVERSA DE UMA MATRIZ

Deseja-se resolver o seguinte sistema de equações lineares :

$$A x = b$$

onde  $A \in \mathbb{R}^{n \times n}$ . Obtendo-se  $A^{-1}$ , o sistema acima é resolvido através de:

$$A^{-1}A x = A^{-1}b$$

ou

$$x = A^{-1}b$$

Não é conveniente calcular  $A^{-1}$ , por meio de:

$$A [ y_1 \ y_2 \ \dots \ y_n ] = [ e_1 \ e_2 \ \dots \ e_n ]$$

pois isto equivale a resolver  $n$  conjuntos independentes de equações:

$$A y_j = e_j \quad j=1,2,\dots,n$$

Os métodos mais usados para calcular a inversa de uma matriz são:

- (1) Forma produto da inversa (PFI) (Gauss-Jordan)
- (2) Forma de eliminação da inversa (EFI) (Eliminação Gaussiana)

(1) Método de Gauss-Jordan (PFI)

Consideremos a matriz  $A \in \mathbb{R}^{n \times n}$ , não-singular. Sua inversa pode ser calculada como o produto de matrizes de transformações elementares  $E_k$ ,  $k=1,2,\dots,n$ , tal que:

$$A^{-1} = E_n \cdot E_{n-1} \cdot \dots \cdot E_2 \cdot E_1$$

onde as matrizes  $E_k$  são da forma:

$$E_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & \eta_{kk} & & \\ & & \vdots & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

$$\begin{cases} \eta_{ik} = -a_{ik}^{(k)} / a_{kk}^{(k)} & i=1,2,\dots,n \text{ e } i \neq k \\ \eta_{kk} = 1 / a_{kk}^{(k)} \end{cases}$$

Tal método é chamado de forma produto da inversa (PFI).

EXEMPLO

$$A = \begin{bmatrix} 2 & 1 & 0 \\ -3 & 1 & 2 \\ 4 & 2 & 4 \end{bmatrix} = A^{(0)}$$

Cálculo de  $E_1$ :



$$E_1 = \begin{bmatrix} 1 / a_{11}^{(1)} & 0 & 0 \\ -a_{21}^{(1)} / a_{11}^{(1)} & 1 & 0 \\ -a_{31}^{(1)} / a_{11}^{(1)} & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 0 \\ 3/2 & 1 & 0 \\ -4/2 & 0 & 1 \end{bmatrix}$$

$$E_1 A^{(1)} = \begin{bmatrix} 1 & 1/2 & 0 \\ 0 & 5/2 & 2 \\ 0 & 0 & 4 \end{bmatrix} = A^{(2)}$$

Cálculo de  $E_2$ :

$$E_2 = \begin{bmatrix} 1 & -a_{12}^{(2)} / a_{22}^{(2)} & 0 \\ 0 & 1 / a_{22}^{(2)} & 0 \\ 0 & -a_{32}^{(2)} / a_{22}^{(2)} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1/5 & 0 \\ 0 & -2/5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$E_2 A^{(2)} = \begin{bmatrix} 1 & 0 & -2/5 \\ 0 & 1 & 4/5 \\ 0 & 0 & 4 \end{bmatrix} = A^{(3)}$$

Cálculo de  $E_3$ :

$$E_3 = \begin{bmatrix} 1 & 0 & -a_{13}^{(3)} / a_{33}^{(3)} \\ 0 & 1 & -a_{23}^{(3)} / a_{33}^{(3)} \\ 0 & 0 & 1 / a_{33}^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1/10 \\ 0 & 1 & -1/5 \\ 0 & 0 & 1/4 \end{bmatrix}$$

Tem-se que  $E_3 \cdot E_2 \cdot E_1 \cdot A^{(1)} = I$ . Portanto:

$$A^{-1} = E_3 \cdot E_2 \cdot E_1$$

## (2) Eliminação Gaussiana (EFI)

A inversa também pode ser calculada através da fatoração LU da matriz, ou seja, se temos:

$$A = L U$$

então

$$A^{-1} = U^{-1} L^{-1}$$

A forma de eliminação da inversa (EFI) utiliza a decomposição L U da matriz. As matrizes L e U podem ser escritas como:

$$L = L_1 . L_2 . L_3 . \dots . L_{n-1} . L_n$$

$$U = U_n . U_{n-1} . U_{n-2} . \dots . U_2 . U_1$$

$$L_k = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & l_{kk} & \\ & & l_{nk} & 1 \end{bmatrix} \quad U_k = \begin{bmatrix} 1 & & & u_{1k} \\ & \ddots & & \vdots \\ & & 1 & u_{k-1,k} \\ & & & 1 \end{bmatrix}$$

onde  $k=1,2,\dots,n$ . Desta forma:

$$B = L U = L_1 . L_2 . \dots . L_n . U_n . U_{n-1} . \dots . U_2 . U_1$$

As inversas das matrizes L e U são :

$$U^{-1} = U_1^{-1} . U_2^{-1} . \dots . U_{n-1}^{-1} . U_n^{-1}$$

$$L^{-1} = L_n^{-1} . L_{n-1}^{-1} . \dots . L_2^{-1} . L_1^{-1}$$



$$L_1^{-1} = \begin{bmatrix} 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad L_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \quad L_3^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Logo,  $L^{-1} = L_3^{-1} \cdot L_2^{-1} \cdot L_1^{-1}$

$$L^{-1} = \begin{bmatrix} 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/2 & -1 & 1 \end{bmatrix}$$

Para calcular  $U^{-1}$ :

$$U_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad U_2^{-1} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad U_3^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

Assim,  $U^{-1} = U_3^{-1} \cdot U_2^{-1} \cdot U_1^{-1}$

$$U^{-1} = \begin{bmatrix} 1 & 1 & -3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

Portanto, tem-se que  $A^{-1} = U^{-1}L^{-1}$

## 5 - CONSIDERAÇÕES NUMÉRICAS

### ESCOLHA DE PIVÔS

A escolha de pivô é muito importante quando se trabalha com uma precisão dada. Por exemplo, vamos procurar encontrar a solução do seguinte sistema de equações, considerando-se dois dígitos :

$$\begin{cases} x_1 - x_2 = 0 \\ 0.01 x_1 + x_2 = 1 \end{cases}$$

A solução exata do sistema é  $x_1=x_2= 1/1.01$

Vamos considerar dois casos:

CASO 1 : o pivô é o elemento  $a_{11} = 1$

Então:

$$\begin{cases} x_1 - x_2 = 0 \\ (1+0.01)x_2 = 1.0 \end{cases}$$

Neste caso,  $(1+0.01)$  é calculado como sendo 1.0.

Logo:

$$\begin{cases} x_1 - x_2 = 0 \\ 1.0 x_2 = 1.0 \end{cases}$$

Desta forma,  $x_1=x_2= 1.0$  não é uma aproximação ruim do resultado exato  $x_1=x_2= 1/1.01$ .

CASO 2 : o pivô é o elemento  $a_{21} = 0.01$

Neste caso:

$$\begin{cases} 0.01x_1 + x_2 = 1 \\ x_1 - x_2 = 0 \end{cases}$$

Então:

$$\begin{cases} 0.01x_1 + x_2 = 1 \\ (1+100)x_2 = -100 \end{cases}$$

e  $(1+100)$  é calculado como sendo 100.

Logo:

$$\begin{cases} 0.01x_1 + x_2 = 1 \\ -100x_2 = -100 \end{cases}$$

O resultado obtido é  $x_1=0.0$  e  $x_2=1.0$ , que é um desastre!!

Deste modo, como pôde ser observado, diferentes escolhas de pivôs podem produzir resultados diferentes. O exemplo analisado sugere que a escolha de pivôs pequenos, em relação aos outros elementos da matriz, deve ser evitada.

Considerando  $Ax = b$ ,  $A \in \mathbb{R}^{n \times n}$ . Num dado estágio  $k$ , podemos escolher como pivô o elemento que verifica :

$$|a_{kk}^{(k)}| \geq |a_{ij}^{(k)}| \quad i \geq k \text{ e } j \geq k$$

A estratégia de escolher como pivô, num determinado estágio, o maior elemento em valor absoluto é chamada de estratégia de pivotamento completa [19].

Na prática, esta estratégia consome certo tempo, pois envolve a busca entre  $(n-k+1)(n-k+1)$  elementos, no  $k$ -ésimo estágio. Uma alternativa usada é a busca pelo maior elemento apenas na coluna  $k$  e reordenação das linhas da matriz, de modo que o novo pivô  $a_{kk}^{(k)}$  satisfaça:

$$|a_{kk}^{(k)}| \geq |a_{lk}^{(k)}| \quad l=k+1, \dots, n$$

Esta estratégia é chamada de estratégia de pivotamento parcial. A experiência computacional mostra que na maioria dos casos a estratégia de pivotamento parcial é suficiente na resolução de um sistema de equações [19].

A fim de permitir alguma liberdade na escolha de pivôs, pode-se relaxar a condição exigida nos pivotamentos parcial ou total, considerando um elemento  $a_{kk}^{(k)}$  como pivô, se no estágio  $k$  :

$$|a_{kk}^{(k)}| \geq u |a_{ij}^{(k)}| \quad i,j=k+1, \dots, n \quad (1)$$

ou

$$|a_{kk}^{(k)}| \geq u |a_{ik}^{(k)}| \quad i=k+1, \dots, n \quad (2)$$

onde  $u$  é um parâmetro no intervalo  $0 < u \leq 1$ .

Se  $u = 1$ , em (1) tem-se a estratégia de pivotamento completa e em (2), a estratégia de pivotamento parcial.

Se  $0 < u < 1$ , tem-se em geral um número maior de candidatos a pivô num estágio  $k$ , podendo então ser escolhido o melhor destes.

## NÚMERO DE CONDIÇÃO DE UM SISTEMA LINEAR

Considere o sistema de equações :

$$A x = b \quad (1)$$

onde  $A \in \mathbb{R}^{n \times n}$  e  $A$  é não singular.

Deste modo  $A$  tem uma inversa única. Ou seja, o sistema tem uma única solução  $x$ , que pode ser escrita como :

$$x = A^{-1}b$$

Vamos supor que os dados em  $A$  e  $b$  estão sujeitos as certas perturbações  $\delta A$  e  $\delta b$ , e queremos saber seus efeitos no vetor solução  $x$ .

Quando pequenas mudanças nos dados causam grandes mudanças na solução, dizemos que o problema é mal-condicionado. Caso contrário, ele é bem-condicionado.

Serão usadas normas para medir o tamanho das perturbações  $\delta A$  e  $\delta b$  nos dados.

(a) Supondo que  $A$  é conhecida com precisão, mas o vetor  $b$  está sujeito a perturbações, ou seja,  $(b + \delta b)$ . Logo :

$$A ( x + \delta x ) = ( b + \delta b ) \quad (2)$$

Subtraindo (1) de (2) e multiplicando por  $A^{-1}$ :

$$\delta x = A^{-1} \delta b \quad (3)$$

Tomando as normas  $(\|.\|_1, \|.\|_2$  ou  $\|.\|_\infty)$  em (1) e (3)



$$\|A\| \cdot \|x\| \geq \|b\|$$

e

$$\|\delta x\| \leq \|A^{-1}\| \cdot \|\delta b\|$$

de onde segue a desigualdade :

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|\delta b\|}{\|b\|}$$

onde  $k(A) = \|A\| \cdot \|A^{-1}\|$  é definido como número de condição de A.  $k(A)$  fornece uma medida de limite superior da propagação do erro  $\delta b$  em b, supondo que não há erros no processo de resolução do sistema de equações.

(b) Supondo que b é conhecido com precisão, mas a matriz A está sujeita a perturbações, ou seja,  $(A + \delta A)$ :

$$(A + \delta A) \cdot (x + \delta x) = b \quad (4)$$

Subtraindo (1) de (4) e multiplicando por  $A^{-1}$  :

$$\delta x = -A^{-1} \cdot \delta A \cdot (x + \delta x) \quad (5)$$

Tomando as normas em (1) e (5):

$$\|A\| \cdot \|x\| \geq \|b\|$$

$$\|\delta x\| \leq \|A^{-1}\| \cdot \|\delta A\| \cdot \|x + \delta x\|$$

obtemos a desigualdade :

$$\frac{\|\delta x\|}{\|x + \delta x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|\delta A\|}{\|A\|}$$

onde  $k(A)$  fornece uma indicação da sensibilidade da solução de  $Ax = b$  à variação dos coeficientes de  $A$ .

Foram considerados os efeitos de perturbações sobre  $b$  e  $A$  separadamente. Na prática, teríamos que resolver o seguinte problema:

$$(A + \delta A) \cdot (x + \delta x) = (b + \delta b)$$

ou

$$Ax + A\delta x + \delta Ax + \delta A\delta x = b + \delta b \quad (6)$$

Subtraindo (1) de (6) e multiplicando por  $A^{-1}$ :

$$\delta x = -A^{-1} \delta Ax - A^{-1} \delta A \delta x + A^{-1} \delta b \quad (7)$$

Tomando as normas de (7):

$$\|\delta x\| \leq \|A^{-1}\| \cdot (\|\delta b\| + \|\delta A\| \cdot \|x\| + \|\delta A\| \cdot \|\delta x\|)$$

Podemos reescrevê-lo como :

$$\|\delta x\| (1 - \|A^{-1}\| \cdot \|\delta A\|) \leq \|A\| \cdot (\|\delta b\| + \|\delta A\| \cdot \|x\|)$$

Se  $\|A^{-1}\| \cdot \|\delta A\| < 1$  [06] , tem-se :

$$\|\delta x\| \geq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\delta A\|} (\|\delta b\| + \|\delta A\| \cdot \|x\|) \quad (8)$$

para o erro absoluto.

Multiplicando  $\|b\| \leq \|A\| \cdot \|x\|$  por (8), tem-se a relação:

$$\frac{\|\delta x\|}{\|x\|} \leq M k(A) \left[ \frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|} \right]$$

onde

$$M = [1 - k(A) \cdot \|\delta A\| / \|A\|]^{-1}$$

$$k(A) = \|A^{-1}\| \cdot \|A\|$$

Se a perturbação  $\delta A$  em  $A$  é pequena o suficiente, a constante  $M$  é próxima de 1. Logo, a alteração total de  $\|\delta x\|/\|x\|$  nas incógnitas devido a pequenas alterações em  $A$  e  $b$  será pequena se  $k(A)$  não for muito grande. Isto significa que um valor moderado de  $k(A)$  implica que as equações são bem-condicionadas.

## CAPÍTULO II : ALGORITMOS PARA OBTER ESTRUTURAS ADEQUADAS PARA DECOMPOSIÇÃO L U, E RESOLUÇÃO DE SISTEMAS DE EQUAÇÕES LINEARES

### 1 - INTRODUÇÃO

Consideremos o seguinte sistema de equações lineares:

$$A x = b \quad (1)$$

onde  $A \in \mathbb{R}^{n \times n}$ .

De uma maneira informal, podemos dizer que  $A$  é uma matriz esparsa, se vale a pena tirar proveito da quantidade de seus elementos não-nulos na resolução de (1). Numa matriz esparsa é pequena a proporção de elementos não-nulos em relação à quantidade total.

Em particular, numa matriz esparsa, seus elementos podem estar ( salvo permutações de linhas e de colunas ), confinados a certas estruturas interessantes, que simplifiquem os cálculos na eliminação Gaussiana.

Supondo que através de algum método, a estrutura da matriz A seja triangular inferior, ou alguma estrutura bem parecida com esta, certamente, ao resolver o sistema de equações, estarão sendo usadas bem menos operações aritméticas, do que se estivéssemos usando a matriz na forma original.

Ao utilizar a eliminação Gaussiana para resolver um sistema de equações, geralmente a proporção de elementos não-nulos na matriz é alterada, a cada estágio do processo.

Por exemplo, desejamos resolver um sistema de equações, cuja matriz dos coeficientes é representada de forma simbólica, com "x" significando a presença de um elemento não-nulo na posição correspondente da matriz original :

$$A = A^{(1)} = \begin{bmatrix} x & x & x & x & x \\ x & x & & & \\ x & & x & & \\ x & & & x & \\ x & & & & x \end{bmatrix}$$

No 1o. estágio da eliminação, temos:

$$A^{(1)}x = b^{(1)}$$

Considerando  $a_{11}^{(1)} \neq 0$ , temos no 2o. estágio da eliminação :

$$A^{(2)}x = b^{(2)}$$

onde:

$$A^{(2)} = \begin{pmatrix} x & x & x & x & x \\ & o & * & * & * \\ & * & o & * & * \\ & * & * & o & * \\ & * & * & * & o \end{pmatrix}$$

\* : elemento não-nulo, onde antes havia um elemento nulo.

o : elemento não-nulo, com valor modificado.

Como pode ser observado, ocorreram preenchimentos com elementos não-nulos onde antes haviam elementos nulos. Assim, ao resolver um sistema de equações lineares, através da eliminação Gaussiana, seria ideal conseguir minimizar a quantidade de elementos não-nulos a serem criados.

Neste capítulo, serão vistos alguns métodos que buscam atingir, aproximadamente, este objetivo.

## 2 - PIVÔS

Um fator muito importante na resolução de um sistema de equações é a escolha de pivôs. Dependendo da escolha feita, os resultados podem ser péssimos, no que se refere à criação de novos elementos não-nulos (fill-in), bem como na quantidade de operações aritméticas necessárias e, naturalmente, pode ocasionar problemas numéricos.

Considerando, por exemplo, a matriz A :

$$A = \begin{pmatrix} x & x & x & x & x \\ x & x & & & \\ x & & x & & \\ x & & & x & \\ x & & & & x \end{pmatrix}$$

Desejamos resolver o sistema de equações :

$$A x = b$$

usando a eliminação Gaussiana. Faremos a seguir a Eliminação Gaussiana, de forma simbólica.

Escolhendo o elemento  $a_{11}^{(1)} \neq 0$  como pivô, ao eliminarmos os elementos  $a_{21}$ ,  $a_{31}$  e  $a_{51}$ , temos :

$$M_1 A^{(1)} = A^{(2)} = \begin{pmatrix} x & x & x & x & x \\ & o & * & * & * \\ & * & o & * & * \\ & * & * & o & * \\ & * & * & * & o \end{pmatrix}$$

onde  $o$  e  $*$  são como definidos anteriormente e  $M_1$  é uma matriz de transformação elementar.

Neste caso, muitos elementos não-nulos foram criados e a esparsidade da matriz foi prejudicada.

Se o elemento  $a_{21}^{(1)} \neq 0$  fosse escolhido como pivô, no segundo estágio da eliminação, teríamos :

$$M_1 A = A^{(2)} = \begin{pmatrix} & 0 & x & x & x \\ x & x & & & \\ & * & x & & \\ & * & & x & \\ & * & & & x \end{pmatrix}$$

Três elementos não-nulos foram criados e apenas um valor foi modificado. Assim, esta escolha de pivô é bem melhor que a anterior, do ponto de vista da esparsidade da matriz.

Ao invés de manter a estrutura original da matriz, suas linhas e colunas podem ser permutadas, de forma que na estrutura obtida, fique mais evidente a escolha de pivôs convenientes.

Por exemplo, no caso anterior, onde o pivô era o elemento  $a_{21}^{(1)}$ , permutando as linhas 1 e 2, temos :

$$P_1 A = \begin{pmatrix} x & x & & & \\ x & x & x & x & x \\ x & & x & & \\ x & & & x & \\ x & & & & x \end{pmatrix} = \tilde{A}^{(1)}$$

No segundo estágio da eliminação, temos:

$$M_1 P_1 A = \begin{pmatrix} x & x & & & \\ & 0 & x & x & x \\ & * & x & & \\ & * & & x & \\ & * & & & x \end{pmatrix}$$

Como já pôde ser observado, a escolha de pivôs é de extrema importância na resolução de um sistema de equações lineares, tanto do ponto de vista numérico quanto do ponto de vista de esparsidade.



### 3 - ESTRUTURA

Poderíamos perguntar se existe alguma estrutura para a qual a resolução de um sistema de equações seja simples, ao aplicar a eliminação Gaussiana, não provocando preenchimentos ( fill-in ) na matriz. A resposta é : sim . Uma matriz que seja triangular inferior tem essas propriedades.

$$A = \begin{bmatrix} x & & & & \\ x & x & & & \\ x & x & x & & \\ x & & x & x & \\ x & x & & x & x \end{bmatrix}$$

FIGURA 2 : MATRIZ TRIANGULAR INFERIOR

Dado o sistema de equações a ser resolvido :

$$A x = b$$

onde A é a matriz da FIGURA 2. A solução do sistema é obtida facilmente, além de que, não ocorrem preenchimentos na matriz.

Nem sempre é possível conseguir a estrutura triangular para uma matriz, embora ela seja desejável.

Às vezes, uma estrutura bem parecida pode ser encontrada para uma dada matriz ( usando métodos a serem ainda descritos ): uma estrutura bloco triangular inferior ( FIGURA 3 ).

$$B = \begin{bmatrix} B_{11} & & & \\ B_{21} & B_{22} & & \\ B_{31} & B_{32} & B_{33} & \\ B_{41} & B_{42} & B_{43} & B_{44} \end{bmatrix}$$

FIGURA 2

Mas, em alguns casos, nenhuma das duas estruturas, triangular ou bloco triangular inferiores, é conseguida para uma matriz.

Considerando uma matriz  $B \in \mathbb{R}^{n \times n}$ , com estrutura bloco triangular inferior, os blocos  $B_{ii}$ ,  $i=1,2,\dots,N$  ( $N \leq n$ ) podem ser densos. Mesmo assim, resolver um sistema para o qual foi obtida a estrutura bloco triangular inferior, é bem mais simples que resolver o sistema de equações com a matriz na estrutura original. Afinal, o problema pode ser resolvido como uma sequência de sub-problemas.

A seguir, a estrutura bloco triangular inferior será vista com mais detalhes.

## ESTRUTURA BLOCO TRIANGULAR INFERIOR

Consideremos o seguinte sistema de equações lineares :

$$A x = b \quad (1)$$

com  $A \in \mathbb{R}^{n \times n}$  e  $A$  uma matriz não-singular.

Utilizando matrizes de permutação adequadas (quando possível), a estrutura bloco triangular inferior pode ser obtida:

$$B = P A Q = \begin{bmatrix} B_{11} & & & \\ B_{21} & B_{22} & & \\ B_{31} & B_{32} & B_{33} & \\ \vdots & \vdots & \vdots & \ddots \\ B_{N1} & B_{N2} & B_{N3} \dots & B_{NN} \end{bmatrix}$$

FIGURA 4

onde cada bloco  $B_{ii}$ ,  $i=1,2,\dots,N$  é quadrado e irredutível.

Uma matriz é irredutível quando não pode ser colocada na estrutura bloco triangular, através da multiplicação por matrizes de permutação, ou seja, quando  $N=1$  (um único bloco). Caso contrário, ela é dita ser redutível.

Obviamente, estamos considerando a matriz A do exemplo, como uma matriz redutível.

Deve ser observado também que :

$$\sum_{i=1}^N (\dim B_{ii}) = n$$

#### EXEMPLO 1

Consideremos a seguinte matriz:

$$A = \begin{bmatrix} x & x & x \\ & x & x \\ & x & x \end{bmatrix}$$

Sejam as seguintes matrizes de permutação :

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad Q = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Fazendo  $P A Q = B$ , temos:

$$P A Q = B = \left[ \begin{array}{cc|c} x & x & \\ x & x & \\ \hline x & x & x \end{array} \right] = \begin{bmatrix} B_{11} & \\ B_{21} & B_{22} \end{bmatrix}$$

Logo, a matriz  $A$  é redutível.

## EXEMPLO 2

Considerando a matriz:

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | x | x |   | x |   |   |   |   |   |
| 2 | x | x | x |   | x |   |   |   |   |
| 3 |   | x | x |   |   | x |   |   |   |
| 4 | x |   | x | x |   |   | x |   |   |
| 5 |   | x |   | x | x | x |   | x |   |
| 6 |   |   | x |   | x | x |   |   | x |
| 7 |   |   |   | x |   |   | x | x |   |
| 8 |   |   |   |   | x |   | x | x | x |
| 9 |   |   |   |   |   | x |   | x | x |

Não é possível encontrar matrizes de permutação  $P$  e  $Q$ , para que a matriz  $C$  tenha a estrutura bloco triangular inferior, pois  $C$  é irredutível.

A vantagem de conseguir a estrutura bloco triangular para uma matriz é poder resolver um sistema de equações como uma sequência de sub-problemas.

Para  $i=1,2,\dots,N$ ,

$$\begin{cases} B_{ii} y_i = (P b)_i - \sum_{j=1}^{i-1} B_{ij} y_j \\ x = Q y \end{cases}$$

o resultado do sistema (1) pode ser obtido de forma mais simples.

Considerando  $A \in \mathbb{R}^{n \times n}$ . Deve ser observado que uma matriz triangular inferior é um caso particular da estrutura bloco triangular inferior, pois temos :

$$P A Q = \begin{bmatrix} B_{11} & & & \\ B_{21} & B_{22} & & \\ \vdots & \vdots & \ddots & \\ B_{N1} & B_{N2} \dots & & B_{NN} \end{bmatrix}$$

onde  $\dim B_{ii} = 1$ , para  $i=1,2,\dots,N$  e  $N=n$ .

Se houver necessidade de mudanças de linhas e colunas dentro de um bloco diagonal, para manter a estabilidade e esparsidade na resolução do problema, isto não afetará a estrutura bloco triangular da matriz [07].

A escolha conveniente de matrizes de permutação para se conseguir a estrutura bloco triangular inferior de uma matriz está ligada à escolha de uma sequência de pivôs.

A seguir, analisaremos métodos que fornecem sequências de pivôs, de forma a obtermos ou a estrutura triangular inferior ou bloco triangular inferior, quando a matriz dada for redutível.

## 4 - MÉTODOS

Existem vários métodos para obter uma sequência de pivôs, antes de resolver um sistema de equações, de modo a conseguirmos uma boa estrutura para a matriz, facilitando a resolução desse sistema.

Os métodos a serem analisados são:

- (1) Método de Markowitz
- (2) Método do pivô pré-determinado ( $P^3$ )
- (3) Método particionado do pivô pré-determinado ( $P^4$ )
- (4) Método usando 2 estágios : Algoritmo para obter transversal máxima + Algoritmo de Tarjan

O Método de Markowitz difere bastante dos demais mencionados. Podemos dizer que a escolha de pivô é feita de uma forma dinâmica. A cada estágio da eliminação Gaussiana, é escolhido um pivô, que satisfaz certas condições. Se houver uma estrutura triangular "escondida", o método a encontrará.

Já nos outros métodos, podemos dizer que a escolha de pivôs é feita de forma estática, pois a escolha da sequência de pivôs é feita à priori. Havendo uma estrutura triangular, esses métodos a encontrarão. Se não for possível, e a matriz for redutível, a estrutura encontrada será bloco triangular inferior (exceto  $P^3$ ).

Deve ficar bem claro que nem sempre a estrutura bloco triangular pode ser conseguida. Quando a matriz for irredutível, existirá um único bloco, que não poderá ser permutado de forma alguma para a estrutura bloco triangular.

A escolha de uma sequência de pivôs nos métodos mencionados, deve estabelecer um compromisso entre a esparsidade e a estabilidade numérica. Não se pode pensar apenas na esparsidade. Ao escolhermos pivôs cujos valores não sejam adequados, pensando apenas na esparsidade, podemos estar comprometendo a confiabilidade da solução obtida para o sistema de equações.

## 4.1 - MÉTODO DE MARKOWITZ

### ESTRATÉGIA DE MARKOWITZ

A estratégia de Markowitz [27] é um método pioneiro no que se refere à seleção de pivôs.

Consideremos o seguinte sistema de equações lineares a ser resolvido, usando a eliminação Gaussiana:

$$A x = b$$

onde  $A \in \mathbb{R}^{n \times n}$ , e  $A$  é uma matriz não-singular.

Neste método, estaremos escolhendo um pivô à cada estágio da eliminação, segundo um certo critério.

Suponhamos que foram efetuados os  $k-1$  primeiros estágios da eliminação Gaussiana e queremos encontrar o  $k$ -ésimo elemento pivô. No  $k$ -ésimo estágio, temos:

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1k}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2k}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boxed{\begin{matrix} a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & \ddots & \vdots \\ a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{matrix}} & \dots & \vdots \end{bmatrix}$$

A submatriz destacada em  $A^{(k)}$  é conhecida como submatriz ativa  $A_k$ , e  $A_k \in \mathbb{R}^{(n-k+1) \times (n-k+1)}$ , para  $k=1,2,\dots,n$ . É nesta submatriz ativa que escolheremos o elemento pivô no estágio  $k$ .

Sejam os seguintes vetores:

$J(i)^{(k)}$ : quantidade de elementos não-nulos na linha  $i$  da submatriz  $A_k$  em  $A^{(k)}$ .

$IC(j)^{(k)}$ : quantidade de elementos não-nulos na coluna  $j$  da submatriz  $A_k$  em  $A^{(k)}$ .

O custo Markowitz associado ao elemento  $a_{ij}^{(k)}$  é definido como:

$$M_{ijk} = (J(i)^{(k)} - 1) * (IC(j)^{(k)} - 1) \quad i,j=1,2,\dots,n$$

$M_{ijk}$  é a quantidade de elementos que mudarão seus valores de  $A^{(k)}$  para  $A^{(k+1)}$ , se o elemento  $a_{ij}^{(k)}$  for escolhido como pivô. Assim,  $M_{ijk}$  é um limite superior para a quantidade de preenchimentos (fill-in) que pode ocorrer devido a esta escolha de elemento pivô.



Seja:

$$M_k = \min \{ M_{ijk} \mid i,j=k, \dots, n \}$$

no estágio  $k$ . O critério de Markowitz escolhe como pivô, no  $k$ -ésimo estágio, o elemento:

$$a_{ij}^{(k)} \text{ tal que } M_{ijk}=M_k$$

ou seja, o elemento que tem custo Markowitz mínimo, no estágio  $k$ .

Existe um motivo pelo qual a escolha de pivô é realizada a cada estágio da eliminação Gaussiana, obedecendo a este critério. Se fosse escolhida uma sequência fixa de pivôs, usando a matriz original, algum elemento pivô poderia se tornar nulo. Logo, não seria possível seguir a sequência de pivôs até o final.

#### EXEMPLO

$$A = A^{(1)} = \begin{array}{ccccc} & & & & J(i) \\ & & & & \\ \left( \begin{array}{cccc} x & & & x \\ & x & x & \\ & & x & x \\ x & & & x \end{array} \right) & \begin{array}{c} z \\ z \\ z \\ z \end{array} \\ & & & & i=1,2,\dots,n \\ I(i) & \begin{array}{cccc} z & 1 & z & z \end{array} \end{array}$$

No primeiro estágio, temos:

$$M_{111} = (2-1) * (2-1) = 1$$

$$M_{221} = (2-1) * (1-1) = 0$$

$$M_{141} = (2-1) * (3-1) = 2$$

$$M_{231} = (2-1) * (2-1) = 1$$

$$M_{321} = (2-1) * (2-1) = 1$$

$$M_{411} = (2-1) * (2-1) = 1$$

$$M_{341} = (2-1) * (3-1) = 2$$

$$M_{441} = (2-1) * (3-1) = 2$$

$$M_1 = \min \{1, 2, 0, 1, 1, 2, 1, 2\}$$

$$M_{221} = 0$$

Logo, o elemento pivô é  $a_{22}^{(1)}$ . Após o primeiro estágio, temos:

$$A^{(2)} = \begin{bmatrix} x & & x & \\ & x & & x \\ & & x & x \\ x & & & x \\ 2 & 1 & 2 & \end{bmatrix} \begin{matrix} 2 \\ 2 \\ 2 \\ 2 \end{matrix}$$

No segundo estágio, procuramos o pivô em  $A_2$ .

$$M_{112} = (2-1) * (2-1) = 1$$

$$M_{342} = (2-1) * (2-3) = 2$$

$$M_{142} = (2-1) * (3-1) = 2$$

$$M_{412} = (2-1) * (2-1) = 1$$

$$M_{322} = (2-1) * (1-1) = 0$$

$$M_{442} = (2-1) * (3-1) = 2$$

$$M_2 = \min \{1, 2, 0, 2, 1, 2\}$$

$$M_{322} = 0$$

O elemento  $a_{32}^{(2)}$  é o pivô neste estágio.

$$A^{(3)} = \begin{bmatrix} x & x & & \\ & x & & x \\ & & x & x \\ & & x & x \\ 2 & 2 & & \end{bmatrix} \begin{matrix} 2 \\ 2 \\ 2 \\ 2 \end{matrix}$$

No terceiro estágio:

$$M_{112} = (2-1) \times (2-1) = 1$$

$$M_{412} = (2-1) \times (2-1) = 1$$

$$M_{142} = (2-1) \times (2-1) = 1$$

$$M_{442} = (2-1) \times (2-1) = 1$$

$$M_2 = \min \{1, 1, 1, 1\}$$

Neste estágio, qualquer elemento pode ser escolhido como pivô, pois todos têm  $M_{ij2}=1$ . Escolhendo  $a_{14}^{(3)}$  como pivô, após o terceiro estágio da eliminação:

$$A^{(4)} = \begin{pmatrix} \times & \times & & & \\ & \times & \times & & \\ & & \times & \times & \\ & & & * & \end{pmatrix}$$

Apenas o elemento  $a_{44}^{(4)}$  teve seu valor modificado, o que está de acordo com o resultado esperado devido ao critério de Markowitz.

Nem sempre é obtida a melhor escolha de pivô usando a estratégia de Markowitz. Considerando o seguinte exemplo, onde a matriz dada é simétrica e positiva-definida:

|      | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | J(i) |
|------|---|---|---|---|---|---|---|---|---|------|
| 1    | x | x | x | x |   |   |   |   |   | 4    |
| 2    | x | x | x | x |   |   |   |   |   | 4    |
| 3    | x | x | x | x |   |   |   |   |   | 4    |
| 4    | x | x | x | x | x |   |   |   |   | 5    |
| 5    |   |   |   | x | x | x |   |   |   | 3    |
| 6    |   |   |   |   | x | x | x | x | x | 5    |
| 7    |   |   |   |   |   | x | x | x | x | 4    |
| 8    |   |   |   |   |   | x | x | x | x | 4    |
| 9    |   |   |   |   |   |   | x | x | x | 4    |
| I(i) | 4 | 4 | 4 | 5 | 3 | 5 | 4 | 4 | 4 |      |

Escolhendo o elemento (5,5) como pivô, temos:

|      | 5 | 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 | J(i) |
|------|---|---|---|---|---|---|---|---|---|------|
| 5    | x |   |   |   | x | x |   |   |   | 4    |
| 1    |   | x | x | x | x |   |   |   |   | 4    |
| 2    |   | x | x | x | x |   |   |   |   | 4    |
| 3    |   | x | x | x | x |   |   |   |   | 4    |
| 4    |   | x | x | x | x | * |   |   |   | 5    |
| 6    |   |   |   |   | * | x | x | x | x | 5    |
| 7    |   |   |   |   |   | x | x | x | x | 4    |
| 8    |   |   |   |   |   | x | x | x | x | 4    |
| 9    |   |   |   |   |   | x | x | x | x | 4    |
| I(i) | 4 | 4 | 4 | 5 | 5 | 4 | 4 | 4 | 4 |      |

Ocorrem dois preenchimentos com esta escolha do elemento pivô. Já a ordem natural de pivôs :  $a_{11}^{(1)}, a_{22}^{(2)}, \dots, a_{pp}^{(p)}$ , não causa nenhum preenchimento.

Através deste exemplo, podemos observar que usando a estratégia de Markowitz, a quantidade de elementos não-nulos criados num estágio da eliminação Gaussiana, não está sendo necessariamente minimizada.

Em nenhum método de escolha de sequência de pivôs haverá garantia de minimização de preenchimentos (fill-in) [11],[31].

Existem duas desvantagens na estratégia de Markowitz:

(a) a quantidade de elementos não-nulos para os quais precisamos calcular o custo Markowitz  $M_{ij}$ , pode ser muito grande.

(b) podem ser escolhidos elementos muito pequenos como pivô, comprometendo a estabilidade numérica do sistema.

Tendo em vista estes inconvenientes, podemos usar a estratégia de Markowitz generalizada [31], que faz um compromisso entre a estabilidade e a quantidade de preenchimentos.

## ESTRATÉGIA DE MARKOWITZ GENERALIZADA

Com o objetivo de preservar a estabilidade numérica, elementos muito pequenos, ( com relação aos demais da matriz ), não serão aceitos como pivôs.

Procurando não ser tão rígido, é permitida alguma liberdade na escolha do elemento pivô, considerando um parâmetro  $u$  ( $0 < u \leq 1$ ), no  $k$ -ésimo estágio,  $k=1,2,\dots,n-1$

$$|a_{kk}^{(k)}| \geq u \max_k |a_{ik}^{(k)}| \quad i=1,2,\dots,n$$

ou:

$$|a_{kk}^{(k)}| \geq u \max_{i,j} |a_{ij}^{(k)}| \quad i,j=1,2,\dots,n$$

Buscando reduzir a quantidade de elementos  $a_{ij}^{(k)}$  a serem pesquisados na submatriz ativa  $A_k$ , será considerado apenas um certo número de linhas, digamos  $p$  ( $p > 2$ ), de forma que se tenha uma boa chance de manter a quantidade de preenchimentos próxima do mínimo.

As linhas da matriz dada são ordenadas em ordem crescente de quantidade de elementos não-nulos, para que a escolha de pivô no estágio  $k$  seja simplificada.

### EXEMPLO

$$A = \begin{bmatrix} 10^{-5} & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & -3 \\ 0 & 1 & -0.5 & 0 & 0 \\ 1 & 0 & 0 & -4 & 0.5 \\ 0 & 0 & 1.5 & 0 & -1 \end{bmatrix} \quad \begin{matrix} J(i) \\ 2 \\ 3 \\ 2 \\ 3 \\ 2 \end{matrix}$$

$$I(i) \quad \begin{matrix} 2 \\ 3 \\ 3 \\ 1 \\ 3 \end{matrix}$$

Por exemplo, consideremos  $p=3$  e  $u=0.8$ . Após ordenamento de linhas:

$$P_1 A = \begin{bmatrix} 10^{-5} & 1 & 0 & 0 & 0 \\ 0 & 1 & -0.5 & 0 & 0 \\ 0 & 2 & 1 & 0 & -3 \\ 1 & 0 & 0 & -4 & 0.5 \\ 0 & 0 & 1.5 & 0 & -1 \end{bmatrix} \begin{matrix} z \\ z \\ z \\ z \\ z \end{matrix}$$

No primeiro estágio da eliminação:

$$M_{111} = (2-1) \times (2-1) = 1 \qquad M_{121} = (2-1) \times (3-1) = 2$$

$$M_{221} = (2-1) \times (3-1) = 2 \qquad M_{321} = (2-1) \times (3-1) = 2$$

$M_1=1$  e portanto, o elemento pivô é  $a_{11}^{(1)} = 10^{-5}$ .

Verificando se este elemento é um pivô aceitável:

$$|a_{11}^{(1)}| \geq 0.8 \max |a_{i1}^{(1)}| \quad i=2,3,4,5$$

Como:

$$|a_{11}^{(1)}| = 10^{-5} \text{ não é maior ou igual a } 0.8 \times 1$$

o elemento  $a_{11}^{(1)}$  não é aceito como pivô. Escolhemos então,  $a_{12}^{(1)} = 1$ , por exemplo, já que todos os outros elementos têm  $M_{ij1} = 2$ .

Verificando:

$$|a_{12}^{(1)}| \geq 0.8 \max_{i=2,3,4,5} |a_{i2}^{(1)}|$$

$$\text{E, no caso, } |1| \geq 0.8 \times 1$$

Logo, o elemento pivô  $a_{12}^{(1)} = 1$  é o pivô no primeiro estágio da eliminação Gaussiana. Portanto, para o segundo estágio, temos:

$$M_1 P_1 A P_2 = \begin{bmatrix} 1 & 10^{-5} & 0 & 0 & 0 \\ 0 & -10^{-5} & -0.5 & 0 & 0 \\ 0 & -2 \times 10^{-5} & 1 & 0 & -3 \\ 0 & 1 & 0 & -4 & 0.5 \\ 0 & 0 & 1.5 & 0 & -1 \end{bmatrix} \begin{matrix} z \\ z \\ z \\ z \\ z \end{matrix} = A^{(2)}$$

↓  
submatriz ativa

Este processo é realizado até o último estágio da eliminação Gaussiana.

Deve ser observado que se fosse usada a estratégia de Markowitz, o primeiro pivô do exemplo acima seria  $a_{44}^{(1)} = -4$ , pois  $M_{44} = 0$ .

## 4.2 - MÉTODO DO PIVÔ PRÉ-DETERMINADO ( $P^3$ ) E MÉTODO PARTICIONADO DO PIVÔ PRÉ-DETERMINADO ( $P^4$ )

### INTRODUÇÃO

Os algoritmos  $P^3$  [24] e  $P^4$  [25] foram apresentados por Hellerman e Rarick. O objetivo dos algoritmos é encontrar uma sequência de pivôs para uma matriz não-simétrica, de forma que a resolução de um sistema de equações lineares seja simplificado.

Se a matriz dada tiver uma estrutura triangular inferior "escondida", a mesma será encontrada nos passos iniciais nos dois algoritmos.

Serão usados nos algoritmos a serem descritos, termos introduzidos pelos autores em [24], com os quais devemos estar familiarizados. Consideremos a seguinte matriz:

|    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|----|---|---|---|---|---|---|---|---|---|----|----|----|
| 1  | x |   |   |   |   |   |   |   |   |    |    |    |
| 2  |   | x |   |   | x |   |   |   |   |    |    |    |
| 3  | x | x | x | x | x |   |   |   |   |    |    |    |
| 4  |   | x | x | x |   |   |   |   |   |    |    |    |
| 5  | x |   | x | x | x |   |   |   |   |    |    |    |
| 6  |   | x |   | x |   | x |   |   |   |    |    |    |
| 7  | x |   | x |   | x |   | x | x |   |    |    |    |
| 8  |   | x |   | x |   |   | x | x |   |    |    |    |
| 9  | x |   |   |   |   | x |   | x | x |    |    |    |
| 10 |   | x |   | x |   |   |   |   |   | x  | x  | x  |
| 11 |   |   | x |   | x |   |   |   |   | x  | x  | x  |
| 12 |   |   |   | x |   | x |   |   |   | x  | x  | x  |

FIGURA 1



Existem certas colunas na matriz, que têm elementos não-nulos acima da diagonal principal. Tais colunas são chamadas SPIKES. Na matriz B, as colunas 4, 5, 8 e 12 são spikes.

Como podemos observar, tais colunas se encontram dentro de certos blocos. Cada um deles é chamado de BUMP. Na figura 1 existem três bumps.

Os passos gerais dos algoritmos  $P^3$  e  $P^4$  serão, primeiramente, explicados de maneira bastante simples. Antes, alguns vetores precisam ser definidos. Dada uma matriz  $A \in \mathbb{R}^{n \times n}$  :

$IC(i), i=1,2,\dots,n$  : quantidade de elementos não-nulos na coluna  $i$ .

$JC(i), i=1,2,\dots,n$  : quantidade de elementos não-nulos na linha  $i$ .

$CC(i), i=1,2,\dots,n$  : índice da coluna do elemento pivô  $i$ .

$RC(i), i=1,2,\dots,n$  : índice da linha do elemento pivô  $i$ .

Os passos gerais para os dois algoritmos são:

Passo 1: Examinar se existem colunas com um único elemento. Escolher este elemento como pivô e atualizar  $IC(i)$  e  $JC(i)$ . Eliminar a linha e coluna do pivô da lista de índices  $RC(i)$  e  $CC(i)$ . Proceder desta forma até que não haja mais nenhum  $IC(i) = 1$ .

Ao final deste passo, temos:

$$P_1 \ A \ P_2 =$$

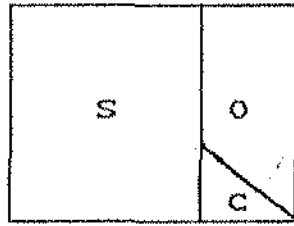


FIGURA 2

As colunas que compõem a seção C são chamadas colunas triangulares finais.

Passo 2: Examinar se existem linhas com um único elemento. Escolher este elemento como pivô e atualizar  $I(i)$  e  $J(i)$ . Eliminar a linha e coluna da lista de índices  $R(i)$  e  $C(i)$ . Proceder desta forma até que não haja mais nenhum  $J(i) = 1$ .

Ao final destes dois passos, temos:

$$P_3 \ P_1 \ A \ P_2 \ P_4 =$$

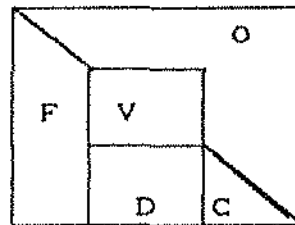


FIGURA 3

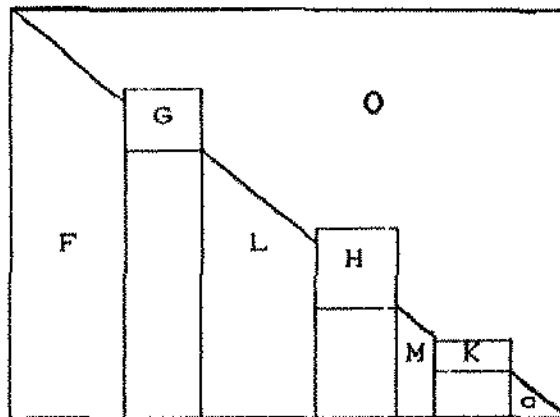
As colunas que compõem a seção F são chamadas colunas triangulares iniciais e a seção V é um bump.

Nas seções C e F, os elementos não-nulos ( que podem existir ou não ) estão abaixo dos pivôs que se encontram na diagonal.

Deve ser observado que se a matriz tiver uma estrutura triangular inferior, por meio destes dois passos, ela será encontrada.

Passo 3: Particionar o bump ( se houver)

O objetivo neste passo é particionar o bump V em bumps menores, chamados bumps externos. Por exemplo, particionando V em:



As seções G, H e K são bumps externos e as seções L e M são compostas de colunas triangulares inferiores.

Os dois algoritmos diferem na forma como os bumps externo são obtidos no passo 3. Além disso, podemos ter bumps externos com dimensões diferentes, usando os dois métodos para a mesma matriz.

No algoritmo  $P^3$ , os bumps externos são obtidos usando um método heurístico. Neste algoritmo, os bumps externos podem ser matrizes reduzíveis.

Já no algoritmo  $P^4$ , a partição do bump em bumps externos é feita segundo um método que consiste de duas etapas. Os bumps externos obtidos no final do processo são matrizes irredutíveis. Depois, é aplicada uma versão simplificada do algoritmo  $P^3$ , a cada bloco irredutível. Desta forma, o algoritmo  $P^4$  é mais eficiente que o  $P^3$ , pois através dele é conseguido, geralmente, uma sequência melhor de pivôs, para ser aplicada à matriz dada.

Usando o algoritmo  $P^4$ , se a matriz for redutível, a estrutura resultante após permutações, será triangular inferior ou bloco triangular inferior. No caso do algoritmo  $P^3$ , a estrutura triangular inferior pode ser encontrada, se houver. Mas, a estrutura bloco triangular inferior, nem sempre é obtida ao final deste algoritmo.

### METODO DO PIVÔ PRE-DETERMINADO ( $P^3$ )

As definições abaixo são válidas para os algoritmos  $P^3$  e  $P^4$ . Consideremos  $A \in \mathbb{R}^{n \times n}$ .

$L$  : índice de spikes num bump externo

$CC(i)$ ,  $i=1,2,\dots,n$ : índice da coluna do elemento pivô

$RC(i)$ ,  $i=1,2,\dots,n$ : índice da linha do elemento pivô

$IC(i)$ ,  $i=1,2,\dots,n$ : quantidade de elementos não-nulos na coluna  $i$

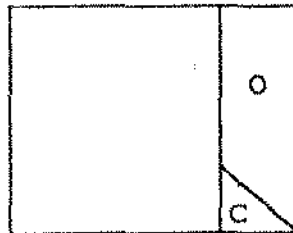
$JC(i)$ ,  $i=1,2,\dots,n$ : quantidade de elementos não-nulos na linha  $i$

Considerando a seguinte matriz para mostrar o algoritmo:

|      | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | J(i) |
|------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|------|
| 1    |   |   |   | x | x |   |   | x |   |    |    |    |    |    |    |    | 3    |
| 2    | x |   |   |   | x |   | x | x |   |    | x  |    |    |    |    | x  | 6    |
| 3    |   |   |   |   |   |   | x |   |   |    |    |    |    |    |    |    | 1    |
| 4    |   | x |   |   | x | x | x |   | x |    |    |    |    |    | x  | x  | 7    |
| 5    |   | x |   | x | x |   |   |   |   |    |    |    |    |    |    |    | 3    |
| 6    |   |   |   |   | x |   |   |   |   | x  |    |    | x  |    | x  |    | 4    |
| 7    |   |   |   | x |   |   |   |   |   | x  |    |    |    |    |    |    | 2    |
| 8    |   |   |   | x |   |   |   |   |   | x  |    |    |    |    |    |    | 2    |
| 9    |   |   | x |   | x | x |   | x |   |    |    | x  |    |    |    |    | 5    |
| 10   | x | x |   |   | x |   |   |   |   |    |    |    |    |    |    | x  | 4    |
| 11   | x |   | x |   |   |   |   |   |   |    |    | x  |    |    |    |    | 3    |
| 12   |   |   | x |   |   | x |   |   |   |    |    | x  |    |    |    |    | 4    |
| 13   | x |   |   |   |   |   |   |   |   |    |    |    |    |    | x  |    | 2    |
| 14   |   |   |   |   |   |   | x |   |   | x  |    |    | x  |    |    |    | 3    |
| 15   | x |   |   |   |   |   |   |   | x | x  |    |    | x  | x  | x  |    | 6    |
| 16   | x | x |   |   |   |   |   | x |   |    | x  |    |    |    |    |    | 4    |
| I(i) | 6 | 4 | 3 | 4 | 7 | 3 | 4 | 4 | 2 | 5  | 2  | 3  | 3  | 1  | 4  | 4  |      |

FIGURA 4

Primeiramente, verificamos se existem colunas triangulares finais, para termos a estrutura da figura abaixo:



Procuramos por algum contador  $I(i) = 1$ , e o encontramos na linha 15. Escolhemos o elemento (15,14) como o último pivô (16o pivô), atualizamos  $I(i)$  e  $J(i)$ , e "eliminamos" seus índices dos vetores  $C(i)$  e  $R(i)$ .

|      | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 15 | 16 | 14 | J(i) |
|------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|------|
| 1    |   |   |   | x | x |   |   | x |   |    |    |    |    |    |    |    | 3    |
| 2    | x |   |   |   | x |   | x | x |   |    | x  |    |    |    |    | x  | 6    |
| 3    |   |   |   |   |   |   | x |   |   |    |    |    |    |    |    |    | 1    |
| 4    |   | x |   |   | x | x | x |   | x |    |    |    |    | x  | x  |    | 7    |
| 5    |   | x |   | x | x |   |   |   |   |    |    |    |    |    |    |    | 3    |
| 6    |   |   |   | x | x |   |   |   |   | x  |    |    | x  | x  |    |    | 4    |
| 7    |   |   |   | x |   |   |   |   |   | x  |    |    |    |    |    |    | 2    |
| 8    |   |   |   | x |   |   |   |   | x |    |    |    |    |    |    |    | 2    |
| 9    |   |   | x |   | x | x |   | x |   |    |    | x  |    |    |    |    | 5    |
| 10   | x | x |   |   | x |   |   |   |   |    |    |    |    |    |    | x  | 4    |
| 11   | x |   | x |   |   |   |   |   |   |    |    | x  |    |    |    |    | 3    |
| 12   |   |   | x |   |   | x |   |   |   |    |    | x  |    | x  |    |    | 4    |
| 13   | x |   |   |   |   |   |   |   |   |    |    |    |    |    |    | x  | 2    |
| 14   |   |   |   |   |   |   | x |   |   | x  |    |    | x  |    |    |    | 3    |
| 15   | x | x |   |   |   |   |   | x |   | x  |    |    |    |    |    |    | 4    |
| 16   | x |   |   |   |   |   |   |   | x | x  |    |    | x  | x  |    | ⊙  | *    |
| I(i) | 5 | 4 | 3 | 4 | 7 | 3 | 4 | 4 | 1 | 4  | 2  | 3  | 2  | 3  | 4  | *  |      |

Ainda existe uma coluna com um único elemento: a coluna 9. Portanto, o elemento (4,9) é o 15o. pivô. Atualizamos I(i), J(i), R(i) e C(i). Verificamos, então, que não existe mais nenhuma coluna com um único elemento. Deste modo, chegamos à estrutura como da FIGURA 2:

|      | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 10 | 11 | 12 | 13 | 15 | 16 | 9 | 14 | J(i) |
|------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|---|----|------|
| 1    |   |   |   | x | x |   |   | x |    |    |    |    |    |    |   |    | 3    |
| 2    | x |   |   |   | x |   | x | x |    | x  |    |    |    | x  |   |    | 6    |
| 3    |   |   |   |   |   |   | x |   |    |    |    |    |    |    |   |    | 1    |
| 5    |   | x |   | x | x |   |   |   |    |    |    |    |    |    |   |    | 3    |
| 6    |   |   |   | x | x |   |   |   | x  |    |    | x  | x  |    |   |    | 4    |
| 7    |   |   |   | x |   |   |   |   | x  |    |    |    |    |    |   |    | 2    |
| 8    |   |   |   | x |   |   |   |   | x  |    |    |    |    |    |   |    | 2    |
| 9    |   |   | x |   | x | x |   | x |    |    | x  |    |    |    |   |    | 5    |
| 10   | x | x |   |   | x |   |   |   |    |    |    |    |    | x  |   |    | 4    |
| 11   | x |   | x |   |   |   |   |   |    |    | x  |    |    |    |   |    | 3    |
| 12   |   |   | x |   |   | x |   |   |    |    | x  |    | x  |    |   |    | 4    |
| 13   | x |   |   |   |   |   |   |   |    |    |    |    |    |    |   | x  | 2    |
| 14   |   |   |   |   |   |   | x |   | x  |    |    | x  |    |    |   |    | 3    |
| 15   | x | x |   |   |   |   |   | x |    | x  |    |    |    |    |   |    | 4    |
| 4    |   | x |   |   | x | x | x |   |    |    |    |    | x  | x  | ⊙ |    | *    |
| 16   | x |   |   |   |   |   |   |   |    | x  |    | x  | x  |    | x | ⊙  | *    |
| I(i) | 5 | 3 | 3 | 4 | 5 | 2 | 3 | 4 | 4  | 2  | 3  | 2  | 2  | 3  | * | *  |      |

Em seguida, verificamos se existem colunas triangulares iniciais ( estrutura da FIGURA 2 ).

Procuramos por alguma linha com um único elemento e, no caso, temos a linha 3 com  $J(3) = 1$ . Então,  $(3,7)$  é escolhido como primeiro elemento pivô. Atualizamos  $J(i)$ ,  $I(i)$ ,  $CC(i)$  e  $RC(i)$ . Constatamos que não existe mais nenhuma linha com único elemento. Deste modo, temos a estrutura da matriz da FIGURA 3 para a matriz dada.

|        | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 8 | 10 | 11 | 12 | 13 | 15 | 16 | 9 | 14 | $J(i)$ |
|--------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|---|----|--------|
| 3      | ⊗ |   |   |   |   |   |   |   |    |    |    |    |    |    |   |    | 3      |
| 1      |   |   |   |   | x | x |   | x |    |    |    |    |    |    |   |    | 5      |
| 2      | x | x |   |   |   | x |   | x |    | x  |    |    |    |    |   | x  | 3      |
| 5      |   |   | x |   | x | x |   |   |    |    |    |    |    |    |   |    | 4      |
| 6      |   |   |   |   |   | x |   |   | x  |    |    | x  | x  |    |   |    | 2      |
| 7      |   |   |   |   | x |   |   |   | x  |    |    |    |    |    |   |    | 2      |
| 8      |   |   |   |   | x |   |   |   | x  |    |    |    |    |    |   |    | 5      |
| 9      |   |   |   | x |   | x | x | x |    |    | x  |    |    |    |   |    | 4      |
| 10     |   | x | x |   |   | x |   |   |    |    |    |    |    |    |   | x  | 3      |
| 11     |   | x |   | x |   |   |   |   |    |    | x  |    |    |    |   |    | 4      |
| 12     |   |   |   | x |   |   | x |   |    |    | x  |    | x  |    |   |    | 2      |
| 13     |   | x |   |   |   |   |   |   |    |    |    |    |    |    | x |    | 2      |
| 14     | x |   |   |   |   |   |   |   | x  |    |    | x  |    |    |   |    | 2      |
| 16     |   | x | x |   |   |   |   | x |    | x  |    |    |    |    |   |    | 4      |
| 4      | x |   | x |   |   | x | x |   |    |    |    |    | x  | x  | ⊗ |    |        |
| 15     |   | x | x |   |   |   |   | x |    | x  |    |    |    |    | x | ⊗  |        |
| $I(i)$ |   | 5 | 3 | 3 | 4 | 5 | 2 | 4 | 4  | 2  | 3  | 2  | 2  | 3  |   |    |        |

FIGURA 3

Tentaremos particionar o bump da matriz acima, em bumps externos. Neste passo, como não existem mais linhas com um único elemento, vamos escolher linhas com a menor quantidade possível de elementos não-nulos. A partir de agora, estaremos "construindo" um bump externo.

No exemplo, temos  $\text{MIN}(J(i)) = 2$ , e existem várias linhas com este contador. Como proceder? Escolheremos uma coluna que será um spike, através de uma função chamada tally function. Esta função fornece a quantidade de elementos não-nulos que existem numa coluna  $m$ , considerando as linhas com contadores  $J(i)$  menores ou iguais a um certo valor mínimo, digamos  $k$ . Ou seja:

$tk(m)$  = quantidade de elementos não-nulos que a coluna  $m$  possui, considerando as linhas cujos contadores são  $J(i) \leq k$ .

Escolhemos como spike, uma coluna que tem a maior quantidade possível de elementos não-nulos, segundo esta tally function. Desta forma, estaremos reduzindo os contadores  $J(i)$ . O objetivo é tentar conseguir, pelo menos, um contador  $J(i) = 1$ . Enquanto isto não ocorre, existe um contador  $L$  que indica quantas colunas spikes (e quem são elas) existem dentro do bump externo que está sendo construído.

No bump da FIGURA 5, encontramos  $\text{MIN}(J(i))=2$  para as linhas 7, 8, 13 e 14. Para escolher uma coluna spike, usamos tally function:

|         | 1 | 2 | 3 | 4 | 5 | 6 | 8 | 10 | 11 | 12 | 13 | 15 | 16 | $J(i)$ |
|---------|---|---|---|---|---|---|---|----|----|----|----|----|----|--------|
| 1       |   |   |   | x | x |   | x |    |    |    |    |    |    | 3      |
| 2       | x |   |   |   | x |   | x |    | x  |    |    |    | x  | 5      |
| 5       |   | x |   | x | x |   |   |    |    |    |    |    |    | 3      |
| 6       |   |   |   |   | x |   |   | x  |    |    | x  | x  |    | 4      |
| 7       |   |   |   | x |   |   |   | x  |    |    |    |    |    | 2      |
| 8       |   |   |   | x |   |   |   | x  |    |    |    |    |    | 2      |
| 9       |   |   | x |   | x | x | x |    |    | x  |    |    |    | 5      |
| 10      | x | x |   |   | x |   |   |    |    |    |    |    |    | 4      |
| 11      | x |   | x |   |   |   |   |    |    | x  |    |    |    | 3      |
| 12      |   |   | x |   |   | x |   |    |    | x  |    | x  |    | 4      |
| 13      | x |   |   |   |   |   |   |    |    |    |    |    | x  | 2      |
| 14      |   |   |   |   |   |   |   | x  |    |    | x  |    |    | 2      |
| 16      | x | x |   |   |   |   | x |    | x  |    |    |    |    | 4      |
| $tz(i)$ | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 3  | 0  | 0  | 1  | 0  | 1  |        |



No caso, a coluna 10 é escolhida como spike, pois  $\max(tk(m)) > 1$  é único. Se tivéssemos  $\max(tk(m)) = 1$  e não fosse único, aumentaríamos o valor de  $k$  de seu valor prévio e calcularíamos tally function, apenas para as colunas que tivessem empate em  $tk(m)$ . Faríamos isto até que tivéssemos  $\max(tk(m)) > 1$ .

Quando  $\max(tk(m)) > 1$ , é único, esta coluna é um spike. Se não é única, a coluna com maior contador  $I(i)$  é escolhida para ser o spike. Assim,  $J(i)$  será reduzido na atualização.

No exemplo, feita a escolha da coluna 10 como spike e atualização de  $J(i)$ , temos algumas possibilidades:

(i) Se  $\min(J(i)) > 1$ , há outro spike no bump que está sendo construído.

(ii) Se  $\min(J(i)) = 1$  e é único, esta coluna é triangular e, portanto, escolhida como pivô.

(iii) Se  $\min(J(i)) = 1$ , mas não é único, precisamos calcular tally function até conseguirmos que  $tk(m) > 1$ , para alguma coluna.

No exemplo, após atualização de  $J(i)$ , temos:

|         | 1 | 2 | 3 | 4 | 5 | 6 | 8 | 11 | 12 | 13 | 15 | 16 | 10 | $J(i)$ |
|---------|---|---|---|---|---|---|---|----|----|----|----|----|----|--------|
| 1       |   |   |   | x | x |   | x |    |    |    |    |    |    | 3      |
| 2       | x |   |   |   | x |   | x | x  |    |    |    | x  |    | 5      |
| 3       |   | x |   | x | x |   |   |    |    |    |    |    |    | 3      |
| 6       |   |   |   |   | x |   |   |    |    | x  | x  |    | x  | 3      |
| 7       |   |   |   | x |   |   |   |    |    |    |    |    | x  | 1      |
| 8       |   |   |   | x |   |   |   |    |    |    |    |    | x  | 1      |
| 9       |   |   | x |   | x | x | x |    | x  |    |    |    |    | 5      |
| 10      | x | x |   |   | x |   |   |    |    |    |    | x  |    | 4      |
| 11      | x |   | x |   |   |   |   |    | x  |    |    |    |    | 3      |
| 12      |   |   | x |   |   | x |   |    | x  |    | x  |    |    | 4      |
| 13      | x |   |   |   |   |   |   |    |    |    |    | x  |    | 2      |
| 14      |   |   |   |   |   |   |   |    |    | x  |    |    |    | 1      |
| 16      | x | x |   |   |   |   | x | x  |    |    |    |    | x  | 4      |
| $tk(i)$ | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 1  |    |        |

Como temos vários contadores  $J(i) = 1$ , precisamos calcular tally function. Através dela, chegamos à conclusão que a melhor escolha possível é a coluna 4 e a linha pode ser a 7 ou 8. O elemento pivô escolhido é (7,4). Atualizamos  $J(i)$ , e temos agora:

|        | 4 | 1 | 2 | 3 | 5 | 6 | 8 | 11 | 12 | 13 | 15 | 16 | 10 | $J(i)$ |
|--------|---|---|---|---|---|---|---|----|----|----|----|----|----|--------|
| 7      | ⊗ |   |   |   |   |   |   |    |    |    |    |    | x  | *      |
| 1      | x |   |   |   | x |   | x |    |    |    |    |    |    | 2      |
| 2      |   | x |   |   | x |   | x | x  |    |    |    | x  |    | 5      |
| 3      | x |   | x |   | x |   |   |    |    |    |    |    |    | 2      |
| 4      |   |   |   |   | x |   |   |    |    | x  | x  |    | x  | 3      |
| 8      | x |   |   |   |   |   |   |    |    |    |    |    | x  | 0      |
| 9      |   |   |   | x | x | x | x |    | x  |    |    |    |    | 5      |
| 10     |   | x | x |   | x |   |   |    |    |    |    | x  |    | 4      |
| 11     |   | x |   | x |   |   |   |    | x  |    |    |    |    | 3      |
| 12     |   |   |   | x |   | x |   |    | x  |    | x  |    |    | 4      |
| 13     |   | x |   |   |   |   |   |    |    |    |    | x  |    | 2      |
| 14     |   |   |   |   |   |   |   |    |    | x  |    |    | x  | 1      |
| 16     |   | x | x |   |   |   | x | x  |    |    |    |    |    | 4      |
| $I(i)$ |   | 5 | 3 | 3 | 6 | 2 | 4 | 2  | 3  | 2  | 2  | 3  |    |        |

Podemos verificar que a linha 8 não tem mais nenhum elemento. Porém, a coluna 10 (spike), tem um elemento nesta posição. Portanto, o elemento (8,10) é o pivô. Atualizamos  $J(i)$  e temos um bump externo construído.

|    | 4 | 10 | 1 | 2 | 3 | 5 | 6 | 8 | 11 | 12 | 13 | 15 | 16 | $J(i)$ |
|----|---|----|---|---|---|---|---|---|----|----|----|----|----|--------|
| 7  | ⊗ | x  |   |   |   |   |   |   |    |    |    |    |    |        |
| 8  | x | ⊗  |   |   |   |   |   |   |    |    |    |    |    |        |
| 1  | x |    |   |   |   | x |   | x |    |    |    |    | x  | 2      |
| 2  |   |    | x |   |   | x |   | x | x  |    |    |    |    | 5      |
| 3  | x |    |   | x |   | x |   |   |    |    |    |    |    | 2      |
| 4  |   | x  |   |   |   | x |   |   |    |    | x  | x  |    | 3      |
| 9  |   |    |   |   | x | x | x | x |    | x  |    |    |    | 5      |
| 10 |   |    | x | x |   | x |   |   |    |    |    |    | x  | 4      |
| 11 |   |    | x |   | x |   |   |   |    | x  |    |    |    | 3      |
| 12 |   |    |   |   | x |   | x |   |    | x  |    | x  |    | 4      |
| 13 |   |    | x |   |   |   |   |   |    |    |    |    | x  | 2      |
| 14 |   | x  |   |   |   |   |   |   |    |    | x  |    |    | 1      |
| 16 |   |    | x | x |   |   |   | x | x  |    |    |    |    | 4      |

Os contadores  $J(i)$  são verificados e temos que, na linha 14 existe um único elemento. Portanto, o elemento (14,13) é um novo pivô. Atualizamos  $J(i)$  :

|    | 4 | 10 | 13 | 1 | 2 | 3 | 5 | 6 | 8 | 11 | 12 | 15 | 16 | $J(i)$ |
|----|---|----|----|---|---|---|---|---|---|----|----|----|----|--------|
| 7  | ⊗ | x  |    |   |   |   |   |   |   |    |    |    |    |        |
| 8  | x | ⊗  |    |   |   |   |   |   |   |    |    |    |    |        |
| 14 |   | x  | ⊗  |   |   |   |   |   |   |    |    |    |    |        |
| 1  | x |    |    |   |   |   | x |   | x |    |    |    |    | 2      |
| 2  |   |    |    | x |   |   | x |   | x | x  |    |    | x  | 5      |
| 3  |   |    |    |   | x |   | x |   |   |    |    |    |    | 2      |
| 5  | x |    |    |   |   |   | x |   |   |    |    |    |    | 2      |
| 6  |   | x  | x  |   |   |   | x |   |   |    |    | x  |    | 5      |
| 9  |   |    |    |   |   | x | x | x | x |    | x  |    |    | 4      |
| 10 |   |    |    | x | x |   | x |   |   |    |    |    | x  | 3      |
| 11 |   |    |    | x |   | x |   |   |   |    | x  |    |    | 4      |
| 12 |   |    |    |   |   | x |   | x |   |    | x  | x  |    | 2      |
| 13 |   |    |    | x |   |   |   |   |   |    |    |    | x  | 4      |
| 14 |   |    |    | x | x |   |   |   | x | x  |    |    |    |        |

Como podemos observar, existem dois casos distintos de contador  $J(i)=1$

Quando temos  $\min(J(i))=1$ , único, esta coluna é triangular. Isto pode ocorrer dentro de um bump externo ou não. Caso  $\min(J(i))=1$  e não é único, estamos num bump externo e necessitamos escolher uma coluna triangular e prosseguir com a construção do bump externo.

Por exemplo, na FIGURA 1, as colunas 2, 3, 7, 10 e 11 são colunas triangulares dentro de bumps. Já as colunas 1, 6 e 9 são colunas triangulares fora de bumps. As colunas 6 e 9 foram obtidas depois que os bumps 4x4 e 2x2 foram construídos, respectivamente.

Voltando ao exemplo, verificamos que não existem  $J(i)=1$ . Portanto, iniciamos a construção de um novo bump externo. Através de tally function, verificamos que a coluna 5 é um spike. Atualizamos  $J(i)$ :

|    | 1 | 2 | 3 | 6 | 8 | 11 | 12 | 13 | 16 | 5 | J(i) |
|----|---|---|---|---|---|----|----|----|----|---|------|
| 1  |   |   |   |   | x |    |    |    |    | x | 1    |
| 2  | x |   |   |   | x | x  |    |    | x  | x | 4    |
| 3  |   | x |   |   |   |    |    |    |    | x | 1    |
| 6  |   |   |   |   |   |    |    | x  |    | x | 1    |
| 9  |   |   | x | x | x |    | x  |    |    | x | 4    |
| 10 | x | x |   |   |   |    |    |    | x  | x | 3    |
| 11 | x |   | x |   |   |    | x  |    |    |   | 3    |
| 12 |   |   | x | x |   |    | x  | x  |    |   | 4    |
| 13 | x |   |   |   |   |    |    |    | x  |   | 2    |
| 16 | x | x |   |   | x | x  |    |    |    |   | 4    |

Usando tally function, obtemos as seguinte colunas triangulares:

|    | 2 | 13 | 8 | 1 | 3 | 6 | 11 | 12 | 16 | 5 | J(i) |
|----|---|----|---|---|---|---|----|----|----|---|------|
| 5  | ⊗ |    |   |   |   |   |    |    |    | x |      |
| 6  |   | ⊗  |   |   |   |   |    |    |    | x |      |
| 1  |   |    | ⊗ |   |   |   |    |    |    | x |      |
| 2  |   |    | x |   |   |   | x  |    | x  | x | 3    |
| 9  |   |    | x |   |   | x | x  |    | x  | x | 3    |
| 10 | x |    |   |   |   | x |    |    | x  | x | 2    |
| 11 |   |    |   |   | x | x |    |    | x  |   | 3    |
| 12 |   | x  |   |   |   | x | x  |    | x  |   | 3    |
| 13 |   |    |   |   | x |   |    |    | x  |   | 2    |
| 16 | x |    | x |   | x |   |    | x  |    |   | 2    |

Agora, temos que  $\min(J(i))=2$ . Portanto, existe outro spike neste bump. Através de tally function, temos que a coluna 1 é um spike. Então:

|    | 2 | 13 | 8 | 3 | 6 | 11 | 12 | 16 | 1 | 5 | J(i) |
|----|---|----|---|---|---|----|----|----|---|---|------|
| 5  | ⊗ |    |   |   |   |    |    |    |   | x |      |
| 6  |   | ⊗  |   |   |   |    |    |    |   | x |      |
| 1  |   |    | ⊗ |   |   |    |    |    |   | x |      |
| 2  |   |    | x |   |   | x  |    | x  | x | x | 2    |
| 9  |   |    | x |   |   | x  | x  |    | x | x | 3    |
| 10 | x |    |   |   |   |    |    | x  | x | x | 1    |
| 11 |   |    |   |   | x | x  |    | x  | x |   | 2    |
| 12 |   | x  |   |   | x | x  |    | x  |   |   | 3    |
| 13 |   |    |   |   |   |    |    | x  | x |   | 1    |
| 16 | x |    | x |   |   | x  |    |    | x |   | 1    |

Usando tally function, temos que  $t_1(16)=2$  e é único. O pivô neste caso é o elemento (10,16). Atualizamos  $J(i)$ . Como  $J(13)=0$ , não temos mais nenhum elemento nesta linha:

|    | 2 | 15 | 8 | 16 | 3 | 6 | 11 | 12 | 1 | 5 | $J(i)$ |
|----|---|----|---|----|---|---|----|----|---|---|--------|
| 5  | ⊗ |    |   |    |   |   |    |    | x |   |        |
| 6  |   | ⊗  |   |    |   |   |    |    | x |   |        |
| 1  |   |    | ⊗ |    |   |   |    |    | x |   |        |
| 10 | x |    |   | ⊗  |   |   |    |    | x | x |        |
| 2  |   |    | x |    |   |   | x  |    | x | x | 1      |
| 9  |   |    | x |    | x | x |    | x  | x | x | 3      |
| 11 |   |    |   |    | x |   |    | x  | x |   | 2      |
| 12 |   | x  |   |    | x | x |    | x  |   |   | 3      |
| 13 |   |    |   | x  |   |   |    |    | x |   | 0      |
| 16 | x |    |   |    |   |   | x  |    | x |   | 1      |

Verificamos que na coluna 1 (spike), existe um elemento na linha 13. Logo, o elemento (13,1) é pivô. Assim, ordenamos um dos spikes no bump que está sendo construído. Atualizamos  $J(i)$ :

|    | 2 | 15 | 8 | 16 | 1 | 3 | 6 | 11 | 12 | 5 | $J(i)$ |
|----|---|----|---|----|---|---|---|----|----|---|--------|
| 5  | ⊗ |    |   |    |   |   |   |    |    | x |        |
| 6  |   | ⊗  |   |    |   |   |   |    |    | x |        |
| 1  |   |    | ⊗ |    |   |   |   |    |    | x |        |
| 10 | x |    |   | ⊗  | x |   |   |    |    | x |        |
| 13 |   |    |   | x  | ⊗ |   |   |    |    |   |        |
| 2  |   |    | x |    |   |   |   | x  |    | x | 1      |
| 9  |   |    | x |    |   | x | x |    | x  | x | 3      |
| 11 |   |    |   |    |   | x |   |    | x  |   | 2      |
| 12 |   | x  |   |    |   | x | x |    | x  |   | 3      |
| 16 | x |    |   |    | x |   |   | x  |    |   | 1      |

Para obtermos uma nova coluna triangular, usamos novamente tally function e o elemento (16,11) é escolhido como pivô. Em seguida,  $J(i)$  é atualizado.

|    | 2 | 15 | 8 | 10 | 1 | 11 | 3 | 6 | 12 | 5 | J(i) |
|----|---|----|---|----|---|----|---|---|----|---|------|
| 5  | ⊗ |    |   |    |   |    |   |   |    | x |      |
| 6  |   | ⊗  |   |    |   |    |   |   |    | x |      |
| 1  |   |    | ⊗ |    |   |    |   |   |    | x |      |
| 10 | x |    |   | ⊗  | x |    |   |   |    | x |      |
| 13 |   |    |   | x  | ⊗ |    |   |   |    |   |      |
| 16 | x |    |   |    | x | ⊗  |   |   |    |   |      |
| 2  |   |    | x | x  |   | x  |   |   |    | x | 0    |
| 9  |   |    | x |    |   |    | x | x | x  | x | 3    |
| 11 |   |    |   |    |   |    | x |   | x  |   | 2    |
| 12 |   | x  |   |    |   |    | x | x | x  |   | 3    |

Novamente, temos uma linha sem elementos. Mas, existe um elemento nesta linha, que pertence ao spike (coluna 5). Neste caso, o pivô é (2,5).

|    | 2 | 15 | 8 | 10 | 1 | 11 | 5 | 3 | 6 | 12 | J(i) |
|----|---|----|---|----|---|----|---|---|---|----|------|
| 5  | ⊗ |    |   |    |   |    | x |   |   |    |      |
| 6  |   | ⊗  |   |    |   |    | x |   |   |    |      |
| 1  |   |    | ⊗ |    |   |    | x |   |   |    |      |
| 10 | x |    |   | ⊗  | x |    | x |   |   |    |      |
| 13 |   |    |   | x  | ⊗ |    |   |   |   |    |      |
| 16 | x |    | x |    | x | ⊗  |   |   |   |    |      |
| 2  |   |    | x | x  | x | x  | ⊗ |   |   |    |      |
| 9  |   |    | x |    |   | x  |   | x | x | x  | 3    |
| 11 |   |    |   |    | x |    |   | x |   | x  | 2    |
| 12 |   | x  |   |    |   |    |   | x | x | x  | 3    |

Assim, os spikes que tínhamos foram ordenados. Em seguida, verificamos através dos  $J(i)$  atualizados, se existe alguma coluna triangular, ou se iniciamos um novo bump. Como  $\min(J(i))=2$ , tem início a construção de outro bump externo.

Usando tally function verificamos que as colunas 3 ou 12 podem ser spikes. Fazemos uma escolha arbitrária e a coluna 3 é o spike. Atualizamos  $J(i)$ :

|    | 6 | 12 | 3 | J(i) |
|----|---|----|---|------|
| 9  | x | x  | x | 2    |
| 11 |   | x  | x | 1    |
| 12 | x | x  | x | 2    |

O elemento pivô é (11,12). Atualizamos JC(i):

|    | 12 | 6 | 8 | JC(i) |
|----|----|---|---|-------|
| 11 | ⊗  |   | x |       |
| 9  | x  | x | x | 1     |
| 12 | x  | x | x | 1     |

Em seguida, temos como pivô o elemento (9,6). Finalmente, a coluna spike é ordenada e temos o pivô (12,3). Assim:

|    | 12 | 6 | 3 |
|----|----|---|---|
| 11 | ⊗  |   | x |
| 9  | x  | ⊗ | x |
| 12 | x  | x | ⊗ |

Portanto, ao final do algoritmo  $P^3$ , temos a seguinte sequência de pivôs e estrutura final para a matriz dada:

|    | 7 | 4 | 10 | 13 | 2 | 15 | 8 | 16 | 1 | 11 | 5 | 12 | 6 | 3 | 9 | 14 | C  | R  |
|----|---|---|----|----|---|----|---|----|---|----|---|----|---|---|---|----|----|----|
| 2  | ⊗ |   |    |    |   |    |   |    |   |    |   |    |   |   |   |    | 7  | 3  |
| 7  |   | ⊗ | x  |    |   |    |   |    |   |    |   |    |   |   |   |    | 4  | 7  |
| 8  |   | x | ⊗  |    |   |    |   |    |   |    |   |    |   |   |   |    | 10 | 8  |
| 14 | x |   | x  | ⊗  |   |    |   |    |   |    |   |    |   |   |   |    | 13 | 14 |
| 5  |   | x |    |    | ⊗ |    |   |    |   | x  |   |    |   |   |   |    | 2  | 5  |
| 6  |   |   | x  | x  |   | ⊗  |   |    |   |    | x |    |   |   |   |    | 15 | 6  |
| 1  |   | x |    |    |   |    | ⊗ |    |   |    | x |    |   |   |   |    | 8  | 1  |
| 10 |   |   |    |    | x |    |   | ⊗  | x |    |   | x  |   |   |   |    | 16 | 10 |
| 13 |   |   |    |    |   | x  |   |    | ⊗ |    |   |    |   |   |   |    | 1  | 13 |
| 16 |   |   |    |    | x |    | x |    | x | ⊗  |   |    |   |   |   |    | 11 | 16 |
| 2  | x |   |    |    |   | x  | x | x  | x |    | ⊗ |    |   |   |   |    | 5  | 2  |
| 11 |   |   |    |    |   |    |   | x  |   |    |   | ⊗  |   | x |   |    | 12 | 11 |
| 9  |   |   |    |    |   | x  |   |    |   | x  |   | x  | ⊗ | x |   |    | 6  | 9  |
| 12 |   |   |    |    | x |    |   |    |   |    |   | x  | x | ⊗ |   |    | 3  | 12 |
| 4  | x |   |    |    | x | x  |   | x  |   | x  |   | x  |   |   | ⊗ |    | 9  | 4  |
| 15 |   |   | x  | x  |   | x  |   | x  |   |    |   |    |   | x | ⊗ |    | 14 | 15 |

Maiores detalhes a respeito do algoritmo são encontrados em [24].

## MÉTODO PARTICIONADO DO PIVO PRE-DETERMINADO ( $P^4$ )

Como no algoritmo  $P^3$ , procuramos encontrar certas estruturas, como da FIGURA 2. Porém, o particionamento do bump  $V$ , em bumps externos é feito de modo diferente.

Utilizando um processo dividido em dois estágios, obtemos bumps externos que são matrizes irredutíveis. Em seguida, os elementos destes bumps são ordenados usando uma versão simplificada do algoritmo  $P^3$ . Desta forma, o algoritmo  $P^4$  busca encontrar a estrutura bloco triangular inferior para uma matriz dada.

Os passos básicos do algoritmo são:

- (1) obtenção de colunas triangulares finais
- (2) obtenção de colunas triangulares iniciais
- (3) particionamento do bump:

(a) Tentativa de assinalamento através do método de assinalamento maximal de Ford-Fulkerson

Neste passo, buscamos encontrar uma transversal máxima no bump, ou seja, procuramos um ordenamento das colunas do bump, de modo que existam elementos não-nulos na diagonal principal do bump

Obtendo uma transversal máxima, podemos associar um grafo à matriz, que será utilizado no próximo passo do algoritmo.

(b) Partições através de relações predecessor-sucessor  
( $P - S$ ).



É definida uma relação de ordem entre as colunas de uma matriz que constitui uma ordenação fraca no grafo associado. Utilizando esta relação de ordem, a matriz é particionada de modo que ficam definidos bumps externos, que são matrizes irredutíveis.

(c) Escolha de spikes e assinalamentos finais de pivôs.

Através de uma versão simplificada do algoritmo  $P^3$ , temos uma sequência de pivôs dentro de cada bump externo, de modo que a quantidade de elementos não-nulos acima da diagonal é a menor possível.

Para explicar o método, será utilizada a mesma matriz do algoritmo anterior (FIGURA 4).

Os dois passos iniciais são exatamente os mesmos de  $P^3$ . Portanto, após obtenção das colunas triangulares finais e iniciais, temos a estrutura da FIGURA 5.

|      | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 8 | 10 | 11 | 12 | 13 | 15 | 16 | 9 | 14 | J(i) |
|------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|---|----|------|
| 2    | ⊗ |   |   |   |   |   |   |   |    |    |    |    |    |    |   |    |      |
| 1    |   |   |   |   | x | x |   | x |    |    |    |    |    |    |   |    | 3    |
| 2    | x | x |   |   |   | x |   | x |    | x  |    |    |    |    |   | x  | 5    |
| 5    |   |   | x |   | x | x |   |   |    |    |    |    |    |    |   |    | 3    |
| 6    |   |   |   |   |   | x |   |   | x  |    |    | x  | x  |    |   |    | 4    |
| 7    |   |   |   |   | x |   |   |   | x  |    |    |    |    |    |   |    | 2    |
| 8    |   |   |   |   | x |   |   |   | x  |    |    |    |    |    |   |    | 2    |
| 9    |   |   |   | x |   | x | x | x |    |    | x  |    |    |    |   |    | 5    |
| 10   |   | x | x |   |   | x |   |   |    |    |    |    |    |    |   | x  | 4    |
| 11   |   | x |   | x |   |   |   |   |    |    | x  |    |    |    |   |    | 3    |
| 12   |   |   |   | x |   |   | x |   |    |    | x  |    |    | x  |   |    | 4    |
| 13   |   | x |   |   |   |   |   |   |    |    |    |    |    |    |   | x  | 2    |
| 14   | x |   |   |   |   |   |   |   | x  |    |    | x  |    |    |   |    | 2    |
| 16   |   | x | x |   |   |   |   | x |    | x  |    |    |    |    |   |    | 4    |
| 4    | x |   | x |   |   | x | x |   |    |    |    |    | x  | x  | ⊗ |    |      |
| 15   |   | x |   |   |   |   |   |   | x  |    |    | x  | x  |    | x | ⊗  |      |
| I(i) |   | 5 | 3 | 3 | 4 | 6 | 2 | 4 | 4  | 2  | 3  | 2  | 2  | 3  |   |    |      |

No passo (3), existem 3 etapas a serem realizadas:

(a) Tentativa de assinalamento através do método de assinalamento maximal de Ford-Fulkerson

Aqui, procura-se ordenar as colunas do bump, de tal forma que se tenham elementos não-nulos na diagonal principal, para obter uma transversal máxima. Assim, um grafo pode ser associado à matriz resultante deste processo.

O processo consiste em examinar sequencialmente cada coluna do bump e assinalar como pivô o primeiro elemento não-nulo possível, assinalando sua linha e coluna como TA (assinalado por tentativa).

Considerando o bump abaixo, temos:

|    | 1  | 2  | 3  | 4  | 5  | 6  | 8  | 10 | 11 | 12 | 13 | 15 | 16 |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1  |    |    |    | ⊗  | x  |    | x  |    |    |    |    |    |    | TA |
| 2  | ⊗  |    |    |    | x  |    | x  |    | x  |    |    |    | x  | TA |
| 3  |    | ⊗  |    | x  | x  |    |    |    |    |    |    |    |    | TA |
| 4  |    |    |    | ⊗  |    |    |    | x  |    |    | x  | x  |    | TA |
| 7  |    |    |    | x  |    |    |    | ⊗  |    |    |    |    |    | TA |
| 8  |    |    |    | x  |    |    |    | x  |    |    |    |    |    |    |
| 9  |    |    | ⊗  |    | x  | x  | x  |    |    | x  |    |    |    | TA |
| 10 | x  | x  |    |    | x  |    |    |    |    |    |    |    | ⊗  | TA |
| 11 | x  |    | x  |    |    |    |    |    |    | ⊗  |    |    |    | TA |
| 12 |    |    | x  |    |    | ⊗  |    |    |    | x  |    | x  |    | TA |
| 13 | x  |    |    |    |    |    |    |    |    |    |    |    | x  |    |
| 14 |    |    |    |    |    |    |    | x  |    |    | ⊗  |    |    | TA |
| 16 | x  | x  |    |    |    |    | ⊗  |    | x  |    |    |    |    | TA |
|    | TA | TA | TA | TA | TA | TA | TA | TA |    | TA | TA |    | TA |    |

FIGURA 6

Quando se encontra uma coluna na qual não se consegue assinalar nenhum elemento, é necessário encontrar uma cadeia de elementos não-nulos. O procedimento é o seguinte:

Iniciamos a cadeia num elemento não-nulo numa coluna não-assinalada e tentamos chegar até um elemento não-nulo numa linha que não tenha sido assinalada e que esteja dentro de uma coluna TA. A técnica é baseada no método de rotulação de Ford-Fulkerson.

Vejamos o método através da matriz da FIGURA 6.

As colunas 11 e 15, bem como as linhas 8 e 13 não foram assinaladas e serão consideradas não-TA. Assim, precisamos encontrar cadeias para que as colunas 11 e 15 se tornem TA.

Iniciando do elemento (2,11), i.é, numa coluna não-TA, achamos uma cadeia que termina no elemento (13,1), numa linha não-TA.

Cadeia: (2,11), (2,1), (13,1)

|    | 1  | 2  | 3  | 4  | 5  | 6  | 8  | 10 | 11 | 12 | 13 | 15 | 16 |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1  |    |    |    | ⊗  | x  |    | x  |    |    |    |    |    |    | TA |
| 2  | ⊗  |    |    |    | x  |    | x  |    | ⊗  |    |    |    | x  | TA |
| 3  |    | x  |    | x  | x  |    |    |    |    |    |    |    |    | TA |
| 6  |    |    |    | ⊗  |    |    |    | x  |    |    | x  | x  |    | TA |
| 7  |    |    |    | x  |    |    |    | ⊗  |    |    |    |    |    | TA |
| 8  |    |    |    | x  |    |    |    | x  |    |    |    |    |    | TA |
| 9  |    |    | ⊗  |    | x  | x  | x  |    |    | x  |    |    |    | TA |
| 10 | x  | x  |    |    | x  |    |    |    |    |    |    |    | ⊗  | TA |
| 11 | x  |    | x  |    |    |    |    |    | ⊗  |    |    |    |    | TA |
| 12 |    |    | x  |    |    | ⊗  |    |    |    | x  |    | x  |    | TA |
| 13 | x  |    |    |    |    |    |    |    |    |    |    |    | x  | TA |
| 14 |    |    |    |    |    |    |    | x  |    |    | ⊗  |    |    | TA |
| 16 | x  | x  |    |    |    |    | ⊗  |    | x  |    |    |    |    | TA |
|    | TA | TA | TA | TA | TA | TA | TA | TA |    | TA | TA |    | TA |    |

A linha 13 é não-TA e o elemento (13,1) está numa coluna TA.

Agora, a coluna 1 é considerada não-TA e a coluna 11 como TA. Logo:

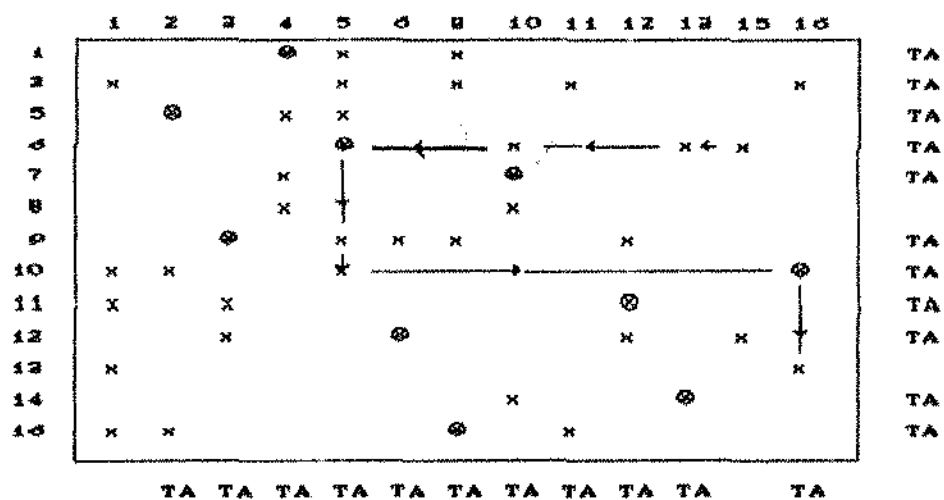
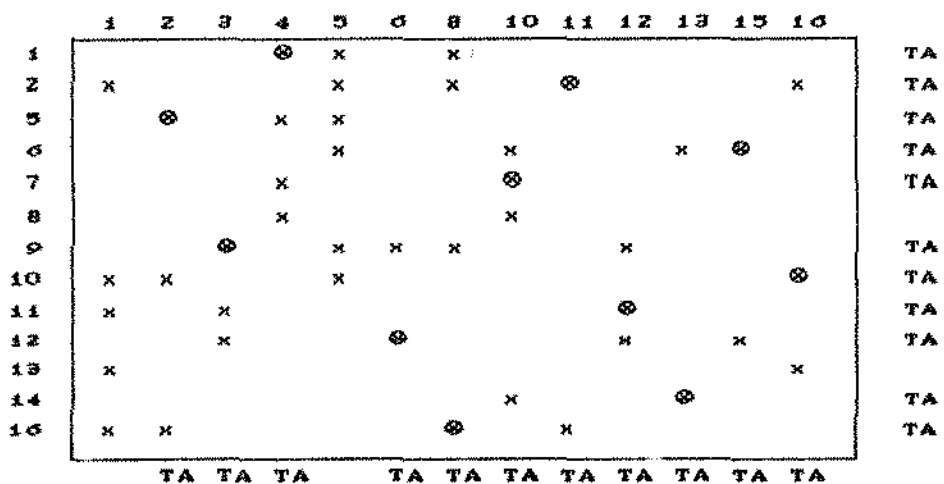


FIGURA 7

Na FIGURA 7 , vamos agora considerar a coluna 15. Temos a seguinte cadeia:

Cadeia: (6,15), (6,9), (10,6), (10,16), (13,16)

A coluna 5 deixa de ser IA. Portanto, as colunas 1 e 5 são não-IA.



Considerando agora a coluna 1, temos diretamente o assinalamento do elemento (13,1) como pivô. Portanto, a coluna 1 e linha 13 são consideradas TA.

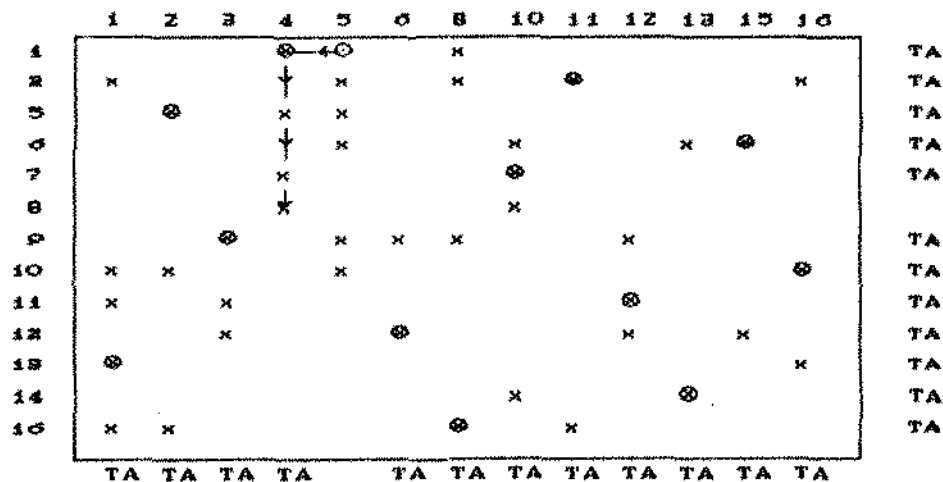
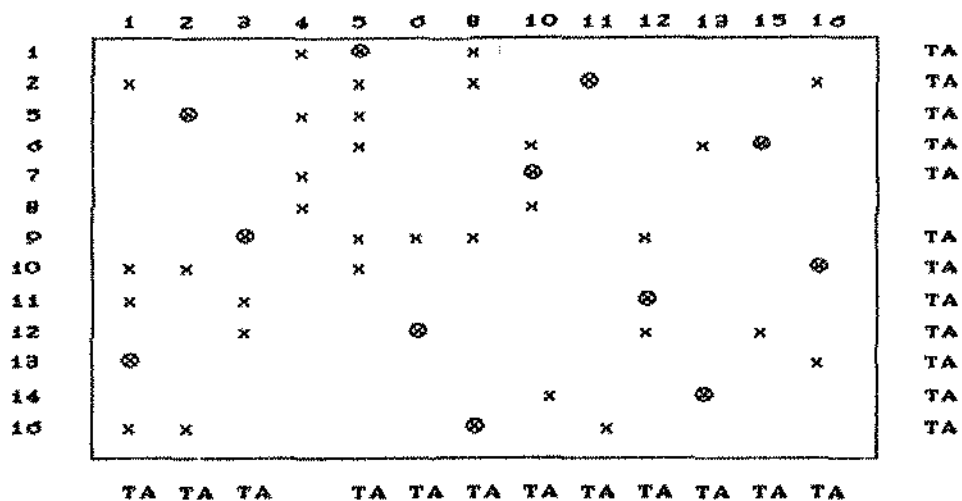


FIGURA 8

Agora, falta apenas assinalar a coluna 5. Usando a FIGURA 8, temos a seguinte cadeia:

Cadeia: (1,5), (1,4), (8,4)

a coluna 5 passa a ser TA e a coluna 4, não TA.



Temos diretamente que, na coluna 4, o elemento (8,4) é um pivô. Considerando a linha 8 e coluna 4 como TA, temos todos os pivôs assinalados.

|    | 1  | 2  | 3  | 4  | 5  | 6  | 8  | 10 | 11 | 12 | 13 | 15 | 16 |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1  |    |    |    | x  | ⊗  |    | x  |    |    |    |    |    |    | TA |
| 2  | x  |    |    |    | x  |    | x  |    | ⊗  |    |    |    | x  | TA |
| 3  |    | ⊗  |    | x  | x  |    |    |    |    |    |    |    |    | TA |
| 4  |    |    |    |    | x  |    |    | x  |    |    | x  | ⊗  |    | TA |
| 7  |    |    |    | x  |    |    |    | ⊗  |    |    |    |    |    | TA |
| 8  |    |    |    | ⊗  |    |    |    | x  |    |    |    |    |    | TA |
| 9  |    |    | ⊗  |    | x  | x  | x  |    |    | x  |    |    |    | TA |
| 10 | x  | x  |    |    | x  |    |    |    |    |    |    |    | ⊗  | TA |
| 11 | x  |    | x  |    |    |    |    |    |    | ⊗  |    |    |    | TA |
| 12 |    |    | x  |    |    | ⊗  |    |    |    | x  |    | x  |    | TA |
| 13 | ⊗  |    |    |    |    |    |    |    |    |    |    |    | x  | TA |
| 14 |    |    |    |    |    |    | x  |    |    |    | ⊗  |    |    | TA |
| 16 | x  | x  |    |    |    | ⊗  |    | x  |    |    |    |    |    | TA |
|    | TA | TA | TA | TA | TA | TA | TA | TA | TA | TA | TA | TA | TA |    |

Após ordenação de colunas, obtemos o bump permutado e chegamos ao final do método.

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

(b) Partição do bump através de relações predecessor-sucessor (P - S).

Neste passo, procuramos particionar o bump em bumps externos, onde cada um deles será uma matriz irredutível, ao final de certo processo. As seguintes definições são necessárias:

(i) a coluna X "precede imediatamente" a coluna Y ( $X \leq Y$ ) se ela tem um elemento não-nulo na linha TA de Y.

(ii) a coluna X "precede" a coluna Y, se existe uma sequência de colunas  $Z_0, \dots, Z_k$  onde  $Z_0 = X$ ,  $Z_k = Y$  e  $Z_{i-1}$  precede imediatamente  $Z_i$ ,  $i=1, 2, \dots, k$ .

(iii) a coluna X "sucede" a coluna Y se Y precede X.

As relações descritas definem uma ordenação fraca num grafo. Através destas relações, serão construídos certos conjuntos que formarão bumps externos. Temos que :

(a) Se  $X \leq Y$  e  $Y \leq X$ , então  $X=Y$ , ou seja, X e Y são equivalentes. Logo, X e Y pertencem ao conjunto E.

(b) Se  $X \leq Y$ , mas não temos  $X=Y$  para  $Y \in E$ , então  $X \in P$  (conjunto dos predecessores de Y)

(c) Se  $X \geq Y$ , mas não temos  $X=Y$  para  $Y \in E$ , então  $X \in S$  (conjunto dos sucessores de Y)

(d) Se X não está em nenhum dos conjuntos E, P ou S, então  $X \in \bar{B}$ .

Para obter tais conjuntos, procedemos como segue.

Selecionar arbitrariamente uma coluna do bump, digamos k. Por exemplo, supor  $k=5$ .

Vejamos quem são as colunas que precedem  $k$ , usando as definições. Obtemos as seguintes relações.

Examinando a linha 1A da coluna 5:

$$5 \geq 5$$

$$5 \geq 4 \quad (1)$$

$$5 \geq 8$$

Examinando a linha 8, associada à coluna 4:

$$10 \geq 4$$

Examinando linha 16, associada à coluna 8:

$$1 \geq 8$$

$$2 \geq 8$$

$$11 \geq 8$$

Examinando linha 2, associada à coluna 11:

$$5 \geq 11$$

$$5 \geq 1$$

$$5 \geq 8$$

$$5 \geq 16$$

Examinando linha 5, associada à coluna 2:

$$5 \geq 2$$

$$5 \geq 4$$

Examinando linha 10, associada à coluna 1:

$$5 \geq 16$$

$$5 \geq 1$$

$$5 \geq 2$$

Examinando linha 13, associada à coluna 16

$$16 \geq 1$$



Temos então, as seguintes relações:

$$\begin{array}{lll}
 5 \geq 5 & 11 \geq 8 & 5 \geq 2 \geq 8 \\
 5 \geq 4 & 5 \geq 11 & 5 \geq 11 \geq 8 \\
 5 \geq 8 & 5 \geq 16 & 5 \geq 16 \geq 1 \geq 8 \\
 10 \geq 4 & 5 \geq 1 & 1 \geq 8 \\
 5 \geq 2 & 2 \geq 8 & 16 \geq 1
 \end{array}$$

Relações obtidas ao procurar colunas que sucedem  $k=5$ .

Examinando a coluna 5:

$$\begin{array}{lll}
 5 \leq 5 & 5 \leq 11 & 5 \leq 2 \\
 5 \leq 3 & 5 \leq 1 & 5 \leq 16
 \end{array}$$

Examinando coluna 11:

$$11 \leq 8 \quad 5 \leq 8$$

Examinando coluna 2:

$$2 \leq 1 \quad 2 \leq 8$$

Examinando coluna 16

$$16 \leq 5$$

Examinando coluna 3:

$$3 \leq 12 \quad 3 \leq 6$$

Examinando coluna 1:

$$1 \leq 12 \quad 1 \leq 16$$

$$1 \leq 11 \quad 1 \leq 8$$

As relações aqui são:

$$\begin{array}{lll}
 5 \leq 5 & 5 \leq 11 & 5 \leq 2 \\
 5 \leq 8 & 2 \leq 8 & 16 \leq 6 \\
 1 \leq 8 & 5 \leq 16 & 11 \leq 8
 \end{array}$$

$$5 \leq 3 \leq 12 \Rightarrow 5 \leq 12$$

$$5 \leq 1 \leq 16 \Rightarrow 5 \leq 16$$

$$5 \leq 3 \leq 6 \Rightarrow 5 \leq 6$$

Além destas, temos que:

$$10 \leq 15$$

$$10 \leq 4 \quad (\text{II})$$

$$10 \leq 13$$

### CONSTRUÇÃO DOS CONJUNTOS

#### CONJUNTO E ( COLUNAS EQUIVALENTES )

$$5 \geq 5 \quad \text{e} \quad 5 \leq 5 \Rightarrow 5 = 5$$

$$5 \geq 8 \quad \text{e} \quad 5 \leq 8 \Rightarrow 5 = 8$$

$$1 \geq 8 \quad \text{e} \quad 1 \leq 8 \Rightarrow 1 = 8$$

$$2 \geq 8 \quad \text{e} \quad 2 \leq 8 \Rightarrow 2 = 8$$

$$5 \geq 11 \quad \text{e} \quad 5 \leq 11 \Rightarrow 5 = 11$$

$$5 \geq 16 \quad \text{e} \quad 5 \leq 16 \Rightarrow 5 = 16$$

$$11 \geq 8 \quad \text{e} \quad 11 \leq 8 \Rightarrow 11 = 8$$

Portanto:

$$E = \{ 5, 1, 8, 2, 16, 11 \}$$

#### CONJUNTO P ( COLUNAS PREDECESSORAS )

$$X \leq Y, \text{ mas não } X = Y \text{ e } Y \in E \Rightarrow X \in P$$

$$\text{De (I): } 4 \leq 5 \quad \text{e} \quad Y = 5 \in E \Rightarrow 4 \in P$$

$$\text{De (I) e (II): } 10 \leq 4 \leq 5 \Rightarrow 10 \leq 5 \Rightarrow 10 \in P$$

$$\text{Portanto, } P = \{ 4, 10 \}$$

### CONJUNTO S ( COLUNAS SUCESSORAS )

$X \geq Y$ , mas não  $X = Y$  e  $Y \in E \Rightarrow X \in S$

$3 \geq 5 \Rightarrow 3 \in S$

$12 \geq 5 \Rightarrow 12 \in S$

$6 \geq 5 \Rightarrow 6 \in S$

$15 \geq 5 \Rightarrow 15 \in S$

Logo,  $S = \{ 3, 12, 6, 15 \}$

A coluna 13 não pertence a nenhum dos conjuntos. Portanto:

$\bar{B} = \{ 13 \}$

Nesta construção, P precede E e S sucede E.  $\bar{B}$  pode ser colocado entre P e E ou E e S. Ao final do processo, o conjunto E é uma matriz irredutível. Porém, P, S e  $\bar{B}$  podem não ser. Assim, deve ser verificado se existem colunas iniciais e finais nestes conjuntos. Se, após isto, as matrizes finais ainda são maiores que  $3 \times 3$ , o método predecessor-sucessor deve ser aplicado novamente aos conjuntos. O processo é realizado até que as matrizes finais sejam irredutíveis.

No exemplo dado:

|    | 10 | 4 | 13 | 5 | 11 | 2 | 1 | 16 | 8 | 15 | 3 | 12 | 6 |
|----|----|---|----|---|----|---|---|----|---|----|---|----|---|
| 7  | ⊗  | x |    |   |    |   |   |    |   |    |   |    |   |
| 8  | x  | ⊗ |    |   |    |   |   |    |   |    |   |    |   |
| 14 | x  |   | ⊗  |   |    |   |   |    |   |    |   |    |   |
| 1  |    | x |    | ⊗ |    |   |   |    |   |    |   |    | x |
| 2  |    |   | x  | ⊗ |    |   | x | x  | x |    |   |    | x |
| 5  |    | x |    |   | ⊗  |   |   |    |   |    |   |    |   |
| 10 |    |   | x  |   |    | x | ⊗ | x  |   |    |   |    |   |
| 13 |    |   |    |   |    |   | x | ⊗  |   |    |   |    |   |
| 16 |    |   | x  | x | x  |   |   |    | ⊗ |    |   |    |   |
| 6  | x  |   | x  | x |    |   |   |    |   | ⊗  |   |    |   |
| P  |    |   |    | x |    |   |   |    | x |    | ⊗ | x  | x |
| 11 |    |   |    |   |    | x |   |    |   |    | x | ⊗  |   |
| 12 |    |   |    |   |    |   |   |    |   | x  | x | x  | ⊗ |

O conjunto  $P$  é uma matriz  $2 \times 2$  densa. O conjunto  $\bar{B}$  consiste de um único elemento e o conjunto  $E$  é uma matriz irredutível. Portanto, precisamos verificar apenas se existem colunas triangulares iniciais para a matriz relacionada ao conjunto  $S$ . Temos então que:

|    |    |   |    |   |
|----|----|---|----|---|
|    | 15 | 9 | 12 | 6 |
| 6  | ⊗  |   |    |   |
| 9  |    | ⊗ | x  | x |
| 11 |    | x | ⊗  |   |
| 12 |    | x | x  | ⊗ |

Assim, o particionamento do bump em bumps externos ( que agora são matrizes irredutíveis ) é o seguinte:

|    |    |   |    |   |    |   |   |    |   |    |   |    |   |
|----|----|---|----|---|----|---|---|----|---|----|---|----|---|
|    | 10 | 4 | 13 | 5 | 11 | 2 | 1 | 10 | 8 | 13 | 3 | 12 | 6 |
| 7  | ⊗  | x |    |   |    |   |   |    |   |    |   |    |   |
| 8  | x  | ⊗ |    |   |    |   |   |    |   |    |   |    |   |
| 14 | x  |   | ⊗  |   |    |   |   |    |   |    |   |    |   |
| 1  |    | x |    | x |    |   |   | x  |   |    |   |    |   |
| 2  |    |   |    | x | x  |   | x | x  | x |    |   |    |   |
| 5  |    | x |    | x |    | x |   |    |   |    |   |    |   |
| 10 |    |   |    | x |    | x | x | x  |   |    |   |    |   |
| 13 |    |   |    |   |    | x | x |    |   |    |   |    |   |
| 16 |    |   |    | x | x  | x |   | x  |   |    |   |    |   |
| 6  | x  |   | x  | x |    |   |   |    | ⊗ |    |   |    |   |
| 9  |    |   |    | x |    |   |   | x  |   | ⊗  | x | x  |   |
| 11 |    |   |    |   |    |   | x |    |   | x  | ⊗ |    |   |
| 12 |    |   |    |   |    |   |   |    |   | x  | x | ⊗  |   |

FIGURA 9

(c) Escolha de spikes e assinalamentos finais de pivôs.

Após particionar o bump original em bumps externos, é necessário encontrar uma ordenação das linhas e colunas dentro de cada bump, de modo que tenhamos uma pequena quantidade de elementos não-nulos acima da diagonal principal.

Para isto, utiliza-se tally function, já definida em  $P^2$ , em cada bump externo. Considerando o bump externo da figura 9:

|      | 5 | 11 | 2 | 1 | 10 | 8 | J(i) |
|------|---|----|---|---|----|---|------|
| 1    | x |    |   |   |    | x | 2    |
| 2    | x | x  |   | x | x  | x | 5    |
| 5    | x |    | x |   |    |   | 2    |
| 10   | x |    | x | x | x  |   | 4    |
| 12   |   |    |   | x | x  |   | 2    |
| 15   |   | x  | x | x |    | x | 4    |
| I(i) | 4 | 2  | 3 | 4 | 3  | 3 |      |

Uma coluna que tem a maior quantidade de elementos não-nulos é escolhida como spike, usando tally function. Assim, tentamos reduzir os contadores  $J(i)$  e tentar obter ao menos um contador  $J(i)=1$ .

No exemplo,  $\min(J(i))=2$  e  $tz(5)=2$  é único. A coluna 5 é um spike. Atualiza-se  $J(i)$ .

|    | 11 | 2 | 1 | 10 | 8 | 5 | J(i) |
|----|----|---|---|----|---|---|------|
| 1  |    |   |   |    | x | x | 1    |
| 2  | x  |   | x | x  |   | x | 3    |
| 5  |    | x |   |    |   | x | 1    |
| 10 |    | x | x | x  |   | x | 3    |
| 12 |    |   | x | x  |   |   | 2    |
| 15 | x  | x | x |    | x |   | 4    |

Como existem vários contadores  $J(i)=1$ , através de tally function é escolhida a melhor coluna triangular. a melhor escolha possível de pivô é (5,2). Em seguida, atualizamos  $J(i)$ .

|    | 2 | 11 | 1 | 10 | 8 | 5 | J(i) |
|----|---|----|---|----|---|---|------|
| 5  | ⊗ |    |   |    |   | x |      |
| 1  |   |    |   |    | x | x | 1    |
| 2  |   | x  | x | x  | x | x | 4    |
| 10 | x |    | x | x  |   | x | 2    |
| 12 |   |    | x | x  |   |   | 2    |
| 15 | x | x  | x |    | x |   | 3    |

Verificamos que existe um único  $J(i)=1$ . Logo, encontramos uma coluna triangular e o elemento (1,8) é pivô. Atualizando  $J(i)$ , temos:

|    | 2 | 9 | 11 | 1 | 10 | 5 | J(i) |
|----|---|---|----|---|----|---|------|
| 5  | ⊗ |   |    |   |    | x |      |
| 1  |   | ⊗ |    |   |    | x |      |
| 2  |   | x | x  | x | x  | x | 3    |
| 10 | x |   |    | x | x  | x | 2    |
| 13 |   |   |    | x | x  |   | 2    |
| 16 | x | x | x  | x |    |   | 3    |

Como agora  $\min(j(i))=2$ , sabemos que existe outro spike neste bump. Então, usamos tally function para a escolha deste outro spike. A melhor escolha é a coluna 1. Atualizando  $J(i)$ , obtemos:

|    | 2 | 9 | 11 | 10 | 1 | 5 | J(i) |
|----|---|---|----|----|---|---|------|
| 5  | ⊗ |   |    |    |   | x |      |
| 1  |   | ⊗ |    |    |   | x |      |
| 2  |   | x | x  | x  | x | x | 2    |
| 10 | x |   |    | x  | x | x | 1    |
| 13 |   |   |    | x  | x |   | 1    |
| 16 | x | x | x  | x  | x |   | 1    |

Temos vários  $J(i)=1$ . Através de tally function, é escolhida uma coluna triangular. No caso, o elemento (10,16) é o pivô. Logo:

|    | 2 | 9 | 10 | 11 | 1 | 5 | J(i) |
|----|---|---|----|----|---|---|------|
| 5  | ⊗ |   |    |    |   | x |      |
| 1  |   | ⊗ |    |    |   | x |      |
| 10 | x |   | ⊗  |    | x | x |      |
| 2  |   | x | x  | x  | x | x | 1    |
| 13 |   |   | x  |    | x |   | 0    |
| 16 | x | x | x  | x  | x |   | 1    |

Como podemos verificar, não existe mais nenhum elemento na linha 13. Porém, a coluna 1, que é um spike, possui um elemento nesta linha. Assim, vamos escolher o elemento (13,1) como pivô e assinalamos este spike. Assim:

|    | 2 | 9 | 10 | 1 | 11 | 5 | J(i) |
|----|---|---|----|---|----|---|------|
| 5  | ⊗ |   |    |   |    | x |      |
| 1  |   | ⊗ |    |   |    | x |      |
| 10 | x |   | ⊗  | x |    | x |      |
| 13 |   |   | x  | ⊗ |    |   |      |
| 2  |   | x | x  | x | x  | x | 1    |
| 16 | x | x |    | x | x  |   | 1    |

Novamente, temos vários  $J(i)=1$ . Após o processo já conhecido, concluímos que o elemento pivô é  $(16,11)$ . Atualizamos  $J(i)$ :

|    | 2 | 8 | 16 | 1 | 11 | 5 | $J(i)$ |
|----|---|---|----|---|----|---|--------|
| 5  | ⊗ |   |    |   |    |   | x      |
| 1  |   | ⊗ |    |   |    |   | x      |
| 10 | x |   | ⊗  | x |    |   | x      |
| 13 |   |   | x  | ⊗ |    |   |        |
| 16 | x | x |    | x | ⊗  |   |        |
| 2  |   | x | x  | x | x  |   | x      |

0

Bem, agora resta assinalar como pivô o elemento  $(2,5)$ , que pertence à coluna spike. Desta forma, temos o bump ordenado:

|    | 2 | 8 | 16 | 1 | 11 | 5 |
|----|---|---|----|---|----|---|
| 5  | ⊗ |   |    |   |    | x |
| 1  |   | ⊗ |    |   |    | x |
| 10 | x |   | ⊗  | x |    | x |
| 13 |   |   | x  | ⊗ |    |   |
| 16 | x | x |    | x | ⊗  |   |
| 2  |   | x | x  | x | x  | ⊗ |

e chegamos ao final do algoritmo.

Estrutura final

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

## 4.3 - MÉTODO UTILIZANDO DOIS ESTÁGIOS : ALGORITMO PARA OBTER TRANSVERSAL MÁXIMA + ALGORITMO DE TARJAN

### INTRODUÇÃO

Consideremos uma matriz  $A \in \mathbb{R}^{n \times n}$  esparsa. A estrutura bloco triangular para a matriz pode ser conseguida (se ela for redutível), utilizando um método de dois estágios [11]:

(a) permutar linhas da matriz, de modo que a diagonal principal seja composta por elementos não-nulos ( encontrar uma transversal ) e

(b) utilizar permutações simétricas para obter a estrutura bloco triangular.

O conjunto de elementos não-nulos na diagonal de uma matriz é chamado de transversal. No primeiro estágio do método, procuramos encontrar uma transversal máxima (n elementos não-nulos na diagonal).

No segundo estágio, temos dois algoritmos que podem ser usados para obter a permutação simétrica:

(i) Algoritmo de Sargent e Westerberg e

(ii) Algoritmo de Tarjan

Cada um deles será visto com detalhes.



## METODO

### (a) ALGORITMO PARA ENCONTRAR UMA TRANSVERSAL MÁXIMA

Neste estágio, buscamos encontrar uma transversal máxima na matriz, ou seja, procuramos uma ordenação para suas linhas, de modo que existam elementos não-nulos na diagonal principal da matriz.

Se é possível obter uma transversal máxima através de permutações de linhas ( ou de colunas ), um grafo pode ser associado à matriz resultante. Os elementos da diagonal principal ( ou,  $i=1,2,\dots,n$  ) serão os nós e os elementos  $a_{ij}$  (  $i \neq j$  ), arcos direcionados. Este grafo será utilizado nos algoritmos do estágio seguinte.

A seguir, será visto um algoritmo para encontrar uma transversal máxima para uma matriz dada.

O algoritmo a ser descrito segue o trabalho realizado por Duff [108], [109] e Gustavson [123].

As colunas da matriz dada são examinadas uma a uma e são efetuadas permutações entre linhas, quando for necessário colocar um elemento não-nulo na diagonal. Vejamos como.

Supondo que já foram encontradas permutações entre linhas, de tal modo que existam elementos não-nulos nas  $k-1$  primeiras posições diagonais. Então, a coluna  $k$  é examinada e procuramos por uma permutação entre linhas, tal que:

(i) sejam preservadas as  $k-1$  primeiras posições diagonais dos elementos não-nulos.

(ii) seja colocado um elemento não-nulo na posição  $k$

Executando o algoritmo deste modo, temos um elemento não-nulo sendo adicionado à transversal, a cada estágio.

O processo de obter a adição de um elemento não-nulo na transversal, num estágio, é chamado de assinalamento.

Em certos casos, adicionar um elemento à transversal é conseguido de forma trivial. Se a coluna  $k$  tem um elemento na linha  $k$  ou abaixo dela ( por exemplo, linha  $i$ ,  $k < i$  ), fazemos uma simples permutação entre estas linhas, obtendo um elemento na posição  $(k,k)$ . Isto é chamado de assinalamento barato ( cheap assignment ) [11].

#### EXEMPLO

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | x |   |   | x |   |   |   |
| 2 |   | x |   |   |   | x |   |
| 3 |   |   | x |   |   |   | x |
| 4 | x |   |   | x |   |   |   |
| 5 |   | x |   |   | x |   |   |
| 6 |   |   |   |   |   |   | x |
| 7 |   |   |   |   |   | x |   |

Analisando a coluna 6, através da permutação das linhas 6 e 7, conseguimos um elemento não-nulo na posição  $(6,6)$ .

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | x |   |   | x |   |   |   |
| 2 |   | x |   |   |   | x |   |
| 3 |   |   | x |   |   |   | x |
| 4 | x |   |   | x |   |   |   |
| 5 |   | x |   |   | x |   |   |
| 6 |   |   |   |   |   | x |   |
| 7 |   |   |   |   |   |   | x |

Nem sempre é possível obter um assinalamento barato. Neste caso, procede-se da seguinte maneira.

Supondo que temos  $k-1$  elementos não-nulos na diagonal, procuramos por uma sequência de colunas  $c_1, c_2, \dots, c_j$ , com  $i_1=k$ , tendo elementos não-nulos nas linhas  $r_1, r_2, \dots, r_j$ , com  $r_i=c_{i+1}$ ,  $i=1, 2, \dots, j-1$  e  $r_j \geq k$ . Então, a sequência de mudança de linhas  $(r_1, r_2), (r_2, r_3), \dots, (r_{j-1}, r_j)$  fornece o resultado desejado.

#### EXEMPLO

|   | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | x |   |   |   |   |   |
| 2 |   | x |   |   |   | x |
| 3 |   |   | x |   |   |   |
| 4 |   | x |   | x |   |   |
| 5 |   |   |   |   | x |   |
| 6 |   |   |   | x |   |   |

$c_1=6$

$r_1=2 \Rightarrow c_2=2$

$r_2=4 \Rightarrow c_3=4$

$r_3=6 \Rightarrow c_4=6$

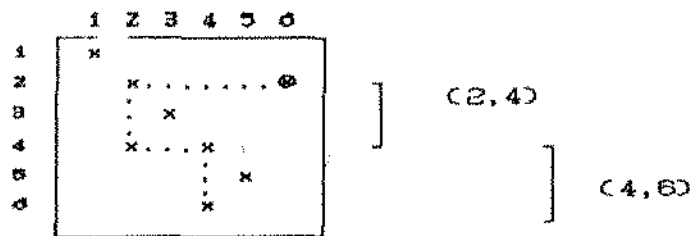
sequência:  $(r_1, r_2), (r_2, r_3) : (2, 4), (4, 6)$

|   | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | x |   |   |   |   |   |
| 4 |   | x |   | x |   |   |
| 2 |   |   | x |   |   |   |
| 3 |   | x |   |   |   | x |
| 5 |   |   |   |   | x |   |
| 6 |   |   |   | x |   |   |

$(r_1, r_2)=(2, 4)$

|   | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | x |   |   |   |   |   |
| 4 |   | x |   | x |   |   |
| 2 |   |   | x |   |   |   |
| 3 |   |   |   | x |   |   |
| 5 |   |   |   |   | x |   |
| 6 |   | x |   |   |   | x |

$(r_2, r_3)=(4, 6)$



Deve ser observado que nem sempre é possível conseguir uma transversal máxima, devido à matriz ser singular. Tal matriz é dita ser simbolicamente ou estruturalmente singular. (FIGURA 1)

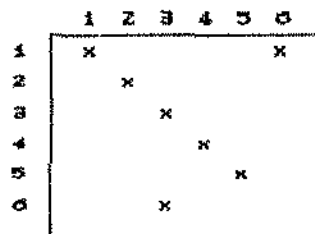


FIGURA 1: MATRIZ ESTRUTURALMENTE SINGULAR

Supondo que não conseguimos obter um elemento não-nulo na posição  $(k,k)$ , continuamos com o algoritmo, a partir da coluna  $k+1$ , tentando colocar outros elementos na diagonal. Mas, um elemento nulo deve ser posto na posição  $(k,k)$ .

Se ao final do algoritmo dado, tivermos uma transversal máxima, iniciamos então, o segundo estágio do método, no qual serão utilizadas permutações simétricas para termos a estrutura bloco triangular para a matriz.

No final deste estágio, os elementos pivôs estão na diagonal principal.

(b) PERMUTAÇÕES SIMÉTRICAS PARA OBTER A ESTRUTURA BLOCO TRIANGULAR

Para este estágio, serão vistos dois algoritmos:

- (i) Algoritmo de Sargent e Westerberg e
- (ii) Algoritmo de Tarjan.

Dada  $A \in \mathbb{R}^{n \times n}$ . Através do primeiro estágio, temos permutações entre linhas da matriz  $A$ , de modo a conseguir uma transversal máxima (a menos que a matriz seja estruturalmente singular).

Assim, após o primeiro estágio, temos:

$$P_1 A \quad (1)$$

com elementos não-nulos na diagonal. Em seguida, desejamos realizar permutações simétricas (i.é, linhas e colunas permutadas identicamente) para que a matriz de (1) seja bloco triangular inferior (supondo  $A$  redutível). Ou seja, queremos encontrar uma matriz de permutação  $Q$ , tal que:

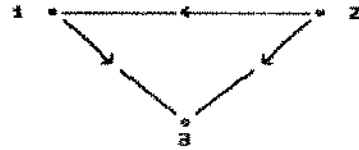
$$Q^T (P_1 A) Q = \begin{pmatrix} B_{11} & & & \\ B_{21} & B_{22} & & \\ \vdots & \vdots & \ddots & \\ B_{N1} & B_{N2} & \dots & B_{NN} \end{pmatrix}$$

Os algoritmos serão descritos com a ajuda de grafos direcionados, associados às matrizes.

# EXEMPLO

$$A = \begin{bmatrix} a_{11} & 0 & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix}$$

MATRIZ ORIGINAL



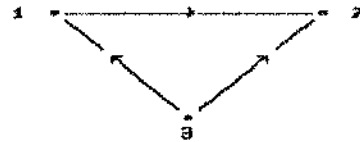
GRAFO ASSOCIADO À MATRIZ

Os elementos da diagonal são considerados nós do grafo e cada elemento não-nulo  $a_{ij}$  fora da diagonal, um arco direcionado  $(i,j)$ .

Ao aplicarmos permutações simétricas a uma matriz, seu grafo associado não é alterado. Apenas os nós recebem nova rotulação.

# EXEMPLO

$$A_1 = \begin{bmatrix} a_{11} & a_{12} & 0 \\ 0 & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$



Considerando as seguintes matrizes de permutação:

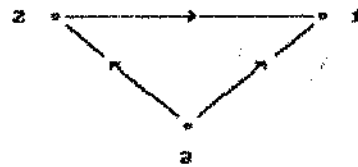
$$Q^T = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Temos que:

$$Q^T A_1 Q = \begin{bmatrix} a_{33} & a_{31} & a_{32} \\ 0 & a_{11} & a_{12} \\ 0 & 0 & a_{22} \end{bmatrix}$$

O grafo associado à  $Q^T A_1 Q$  é:



Um caminho do nó  $v_i$  para o nó  $v_k$  num grafo, é uma sequência de arcos  $(v_1, v_2), (v_2, v_3), \dots, (v_{k-1}, v_k)$ . Ele é um ciclo se  $v_1 = v_k$ .

Um subgrafo consiste de um subconjunto de nós e de todos arcos que são pares dos nós pertencentes a este subconjunto.

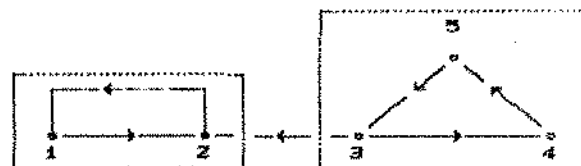
Um subgrafo é fortemente conectado se há um caminho de qualquer um de seus nós para outro qualquer.

Ele é uma componente forte se é fortemente conectado e não pode ser aumentado, através da adição de nós extras e arcos associados, para outro grafo fortemente conectado.

#### EXEMPLO

|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | x | x |   |   |   |
| 2 | x | x |   |   |   |
| 3 |   | x | x | x |   |
| 4 |   |   |   | x | x |
| 5 |   | x | x |   | x |

2 componentes fortes



Os conjuntos de nós que compõem as componentes fortes correspondem aos blocos diagonais da matriz. Cada nó pode pertencer a apenas uma componente forte (que consiste de um único nó). Desta forma, as componentes fortes definem uma partição do grafo.

Duff provou que a estrutura bloco triangular é essencialmente única, e que podem ocorrer permutações entre linhas e colunas dentro de um bloco diagonal, bem como entre os blocos, em certos casos [07].

## ALGORITMO DE SARGENT E WESTERBERG

Podem ser construídos algoritmos para encontrar a estrutura triangular de uma matriz, a partir da observação que se  $A$  é uma permutação simétrica de uma matriz  $C$ , i.é,

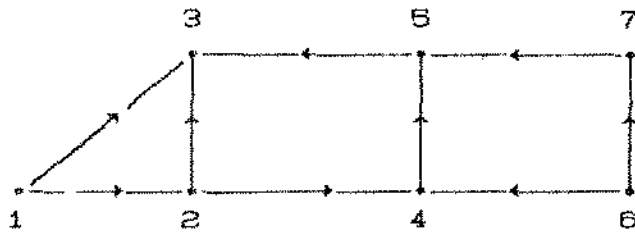
$$A = P^T C P$$

e  $A$  é triangular, deve haver um nó no grafo associado a  $C$ , do qual não parte nenhum arco.

Claramente, este nó deve ser rotulado como primeiro nó, no novo grafo rotulado. Eliminamos então este nó, bem como os arcos que apontam para ele. Novamente, haverá no grafo restante, um nó do qual não parte nenhum arco. Rotulamos este nó como o segundo e o eliminamos, juntamente com os arcos que apontam para ele. Prosseguindo desta forma, a matriz obtida, associada a esta nova rotulação do grafo, será triangular inferior.

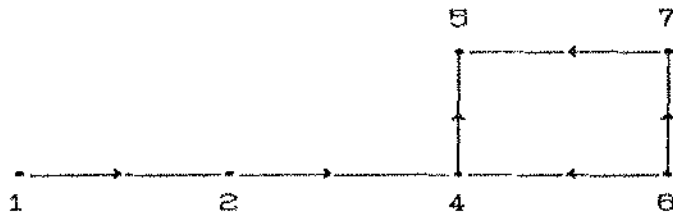


# EXEMPLO



Caminho:  $1 \rightarrow 2 \rightarrow 3$

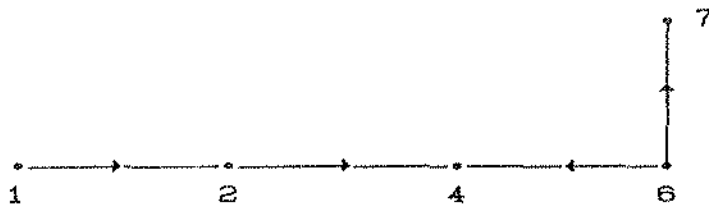
Ao chegar ao nó 3, não existe nenhum arco partindo dele. Logo, o nó 3 é rotulado como primeiro e "apagado" do grafo, bem como seus arcos.



Novamente, seguindo o caminho, temos:

Caminho:  $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$

Como no caso anterior, não parte nenhum arco do nó 5. Assim, ele é rotulado como segundo na nova rotulação, apagado do grafo, juntamente com os arcos que chegam nele.



Em seguida, temos o caminho  $1 \rightarrow 2 \rightarrow 4$ . Nenhum nó parte do nó 4. Ele é rotulado como terceiro nó na nova rotulação, e temos:



No caminho que está sendo seguido, não parte nenhum arco do nó 2, que passa a ser o quarto nó e o nó 1, o quinto na nova rotulação. Acabou-se o caminho seguido, e agora temos:



Prosseguindo de maneira análoga, o nó 7 fica sendo o sexto nó da nova rotulação e o nó 6, o sétimo. Assim, a sequência de caminhos, usada para chegar a esta rotulação é:

|         |   |   |   |   |   |   |   |   |   |    |    |
|---------|---|---|---|---|---|---|---|---|---|----|----|
| Caminho | { |   |   |   |   |   |   |   |   |    |    |
|         |   |   | 3 | 4 | 5 | 4 |   |   |   |    |    |
| passo   | 1 | 2 | 2 | 2 | 2 | 2 | 2 |   |   | 7  |    |
|         | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 6  | 6  |
|         | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | x | x | x |   |   |   |   |
| 2 |   |   | x | x | x |   |   |
| 3 |   |   |   | x |   |   |   |
| 4 |   |   |   |   | x | x |   |
| 5 |   |   |   | x |   | x |   |
| 6 |   |   |   |   | x |   | x |
| 7 |   |   |   |   |   | x | x |

MATRIZ ORIGINAL

|   | 3 | 5 | 4 | 2 | 1 | 7 | 6 |
|---|---|---|---|---|---|---|---|
| 3 | x |   |   |   |   |   |   |
| 5 | x | x |   |   |   |   |   |
| 4 |   |   | x | x |   |   |   |
| 2 | x |   |   | x | x |   |   |
| 1 | x |   |   |   | x | x |   |
| 7 |   | x |   |   |   |   | x |
| 6 |   |   | x |   |   | x | x |

MATRIZ APÓS NOVA ROTULAÇÃO  
DOS NÓS

Deve ser observado que no passo 9, iniciou-se um novo caminho, pois o que estava sendo seguido, terminou.

Sargent e Westerberg generalizaram esta idéia para o caso bloco triangular. Este algoritmo utiliza o fato que todos os nós em qualquer ciclo pertencem à mesma componente forte. Começando de qualquer nó do grafo, um caminho é seguido até que se encontre:

- (i) um ciclo ( que identificamos ao encontrar o mesmo nó duas vezes ), ou
- (ii) um nó do qual não parte nenhum arco.

Quando um ciclo é encontrado, os nós que o compõem são condensados num único nó: um nó composto.

Se for encontrado um nó (ou nó composto), sem que nenhum arco esteja deixando-o, este nó corresponde a uma componente forte. Este nó e seus arcos são "apagados" do grafo, obtendo assim, um subgrafo.

Continuamos com o algoritmo:

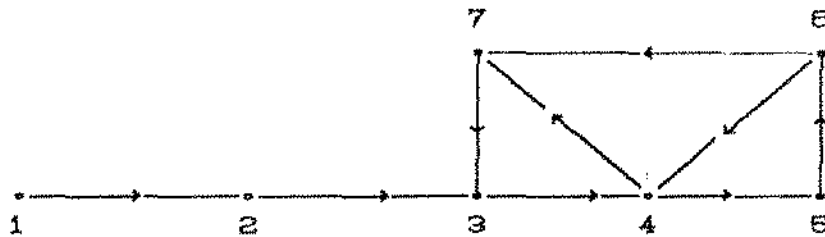
- (a) a partir do último nó no caminho que está sendo seguido,

ou

(b) iniciamos outro caminho , a partir de outro nó, se o caminho que estava sendo seguido, acabou.

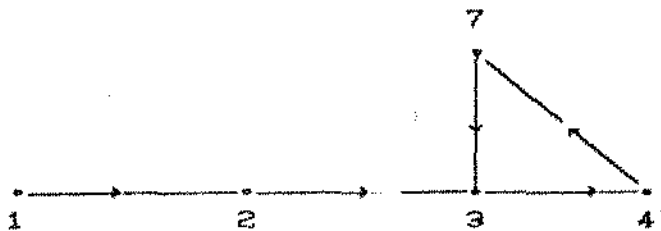
Deste modo, os blocs da estrutura desejada são obtidos sucessivamente.

#### EXEMPLO



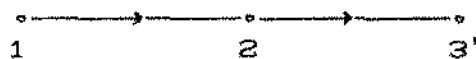
Começamos o caminho a partir do nó 1.

Caminho:  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 4$ . Foi encontrado um ciclo:  $4 \rightarrow 5 \rightarrow 6$ . Fazemos dele um nó composto:  $4'$



Seguindo o caminho, temos:

Caminho:  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4' \rightarrow 7 \rightarrow 3$ . Outro ciclo é encontrado:  $3 \rightarrow 4' \rightarrow 7$ , que é condensado num nó composto:  $3'$



Não existem arcos deixando o nó 3'. Assim, este é o primeiro nó a ser rotulado.

Rotulando 3':

3': ( 3    4'    7 )

3': ( 3    ( 4    5    6 )    7 )  
       1o. 2o. 3o. 4o. 5o.

Eliminando o nó 3' e arco, temos o subgrafo:

1.----->2

Assim, rotulando o nó 2 como sexto nó na nova rotulação, e o nó 1 como sétimo, temos:

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | x | x |   |   |   |   |   |
| 2 |   |   | x | x |   |   |   |
| 3 |   |   |   | x | x |   |   |
| 4 |   |   |   |   | x | x | x |
| 5 |   |   |   |   |   | x | x |
| 6 |   |   |   |   | x |   | x |
| 7 |   |   | x |   |   |   | x |

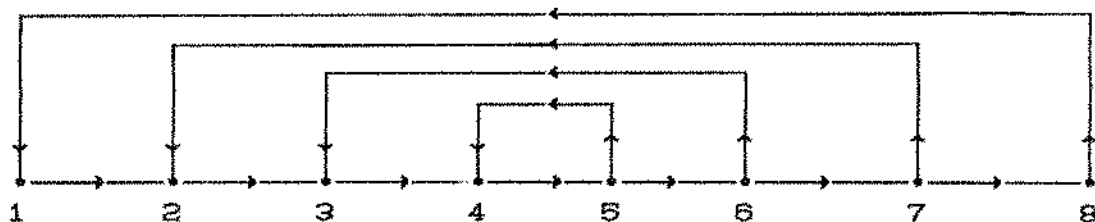
MATRIZ ORIGINAL

|   | 3 | 4 | 5 | 6 | 7 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 3 | x | x |   |   |   |   |   |
| 4 |   | x | x |   | x |   |   |
| 5 |   |   |   | x | x |   |   |
| 6 |   |   | x |   | x | x |   |
| 7 |   | x |   |   |   | x |   |
| 2 | x |   |   |   |   |   | x |
| 1 |   |   |   |   |   |   | x |

APÓS NOVA ROTULAÇÃO

Neste método, existe uma dificuldade associada ao novo rotulamento, no passo dos nós compostos.

## EXEMPLO



Sucessivos nós compostos:

1o. nó : ( 4 5 )

2o. nó : ( 3 ( 4 5 ) 6 )

3o. nó : ( 2 ( 3 ( 4 5 ) 6 ) 7 )

4o. nó : ( 1 ( 2 ( 3 ( 4 5 ) 6 ) 7 ) 8 )

Quantidade de novas rotulações:  $2+4+6+8=20$

## ALGORITMO DE TARJAN

O algoritmo de Tarjan segue a idéia básica do algoritmo de Sargent e Westerberg, traçando caminhos e identificando componentes fortes. Mas, ele tem a vantagem de evitar o passo de condensação de nós [16].

Novamente, seguimos caminhos através do grafo associado a uma matriz, mas agora, os nós são mantidos numa pilha. Esta pilha mantém:

(a) os nós que estão no mesmo caminho corrente, ou

(b) os nós para os quais ocorreu um "backtrack" ( este "backtrack" significa que ocorreu um ciclo ao seguirmos um caminho ).

Para cada nó na pilha é utilizado um apontador ( lowlink ) para indicar qual o nó que está mais abaixo na pilha, num caminho seguido. Todos "lowlinks" são inicializados na pilha, apontando para o próprio nó.

O algoritmo de Tarjan consiste numa certa quantidade de passos principais, cada um consistindo de uma sequência de passos secundários.

Passo principal: começa pela colocação - no caminho e na pilha - de qualquer nó que não esteja na pilha, num passo anterior.

Sequência de passos secundários: o caminho é aumentado de mais um nó a cada passo, ou reduzido (se houver um "backtrack").

O passo principal termina quando acabam a pilha e o caminho que está sendo seguido. O algoritmo termina quando todos os nós foram ordenados tornando impossível iniciar outro passo principal.

#### EXEMPLO

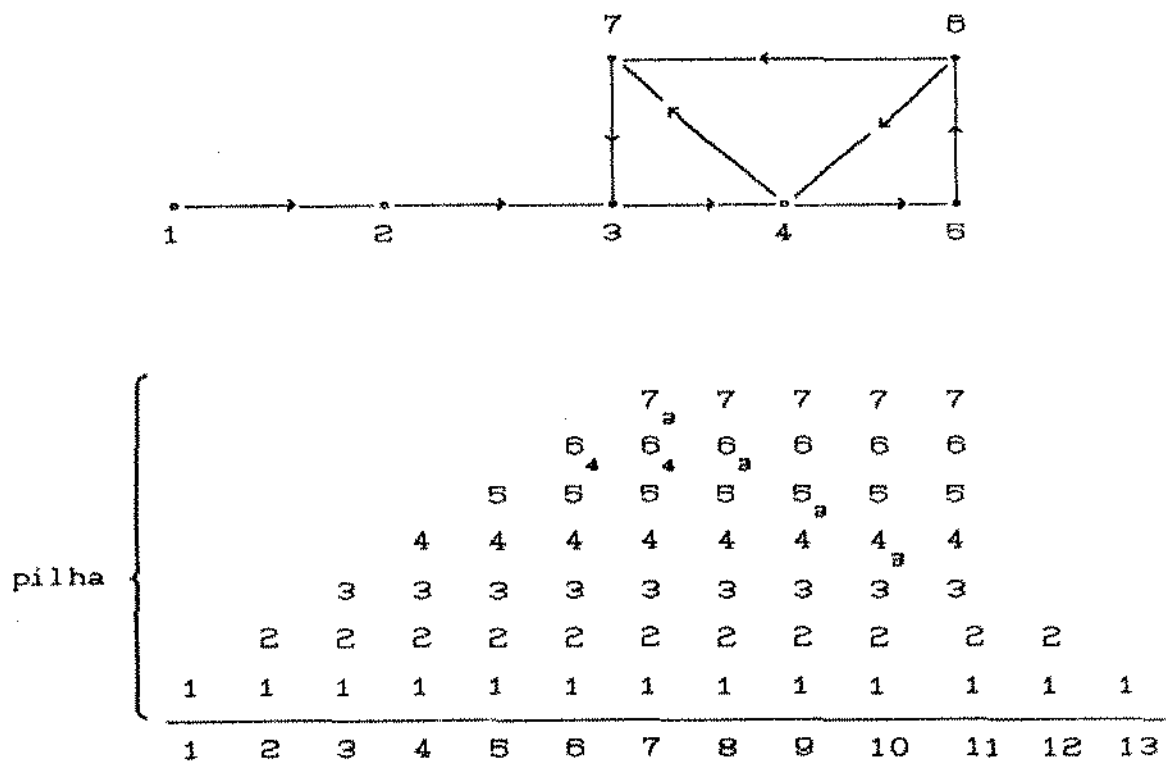


FIGURA 1

Estamos seguindo o caminho  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$ . No passo 6, encontramos o caminho fechado  $(4 \ 5 \ 6)$  e este fato é indicado por  $6_4$ . Aqui, o índice 4 é lowlink. Isto significa que há um ciclo e que o nó 6 está "preso" ao nó 4. No passo 7, o caminho fechado  $(3 \ 4 \ 5 \ 6 \ 7)$  é encontrado e indicado por  $7_3$ . O lowlink aponta para o nó onde inicia o novo ciclo. No passo 8, não há mais arcos deixando o nó 7. Este nó é removido do caminho ( ele é rotulado como primeiro nó ), mas não da pilha, pois ele é parte do caminho fechado  $3 - 7$ . O lowlink de 7 é removido e passado para 6 ( $6_3$ ). O nó 7 é o primeiro nó na nova rotulação. De forma análoga, seguimos até o passo 10.

No passo 11, os nós 3, 4, 5, 6 e 7 estão todos no mesmo ciclo e são retirados da pilha. Eles pertencem à mesma componente forte. Em seguida, temos as componentes fortes 2 e 1. Logo:

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | x | x |   |   |   |   |   |
| 2 |   |   | x | x |   |   |   |
| 3 |   |   |   | x | x |   |   |
| 4 |   |   |   |   | x | x | x |
| 5 |   |   |   |   |   | x | x |
| 6 |   |   |   |   |   |   | x |
| 7 |   |   |   | x |   |   | x |

MATRIZ ORIGINAL

|   | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 7 | x |   |   | x |   |   |   |
| 6 | x | x | x |   |   |   |   |
| 5 |   | x | x |   |   |   |   |
| 4 | x |   | x | x |   |   |   |
| 3 |   |   |   | x | x |   |   |
| 2 |   |   |   |   | x | x |   |
| 1 |   |   |   |   |   | x | x |

APÓS NOVA ROTULAÇÃO

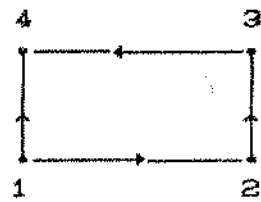
O resultado final é semelhante em ambos os algoritmos, apenas diferindo na ordem dos elementos num bloco. Porém, foi evitado o passo de rotular nós compostos, como no caso do algoritmo anterior.

Num passo típico do algoritmo, existem os seguintes casos:

(1) Encontramos um nó que não está na pilha. Neste caso, é só adicioná-lo à pilha e aumentar o caminho com este nó.



# EXEMPLO



Estamos seguindo um caminho iniciando a partir do nó 1. No passo 2, temos:

|       |   |   |     |
|-------|---|---|-----|
|       |   | 2 |     |
|       | 1 | 1 | ... |
| passo | 1 | 2 | ... |

Como no passo anterior ( passo 1) não havia o nó 2, ele é adicionado à pilha e ao caminho.

(2) São examinados todos os arcos a partir do nó corrente e seu lowlink aponta para um nó que está abaixo dele. Neste caso, este nó é retirado do caminho, mas não da pilha e seu lowlink é transferido para o nó que está imediatamente abaixo.

# EXEMPLO



|   |   |   |                |                |                |   |   |
|---|---|---|----------------|----------------|----------------|---|---|
|   |   |   | 4              | 4              | 4              | 4 |   |
|   |   |   | 4 <sub>1</sub> |                |                |   |   |
|   |   | 3 | 3 <sub>2</sub> | 3 <sub>1</sub> | 3              | 3 |   |
|   | 2 | 2 | 2              | 2              | 2 <sub>1</sub> | 2 |   |
| 1 | 1 | 1 | 1              | 1              | 1              | 1 | 5 |
| 1 | 2 | 3 | 4              | 5              | 6              | 7 | 8 |

No passo 5, o nó 4 é retirado do caminho, mas não da pilha (ver passo 6) e seu lowlink, 1, é transferido para o nó que está imediatamente abaixo, ou seja, temos 3<sub>1</sub>.

(3) Todos os arcos do nó corrente são examinados, e seu lowlink não aponta para um nó abaixo dele. Neste caso, o nó corrente e todos acima dele na pilha constituem uma componente forte e são removidos da pilha.

#### EXEMPLO

No exemplo anterior, no passo 7, temos esta situação. Os nós 1, 2, 3 e 4 constituem uma componente forte e são retirados da pilha.

## CAPÍTULO III : ESQUEMAS DE ATUALIZAÇÃO DE BASES

### 1 - PROBLEMA DE PROGRAMAÇÃO LINEAR

Um problema de programação linear consiste em maximizar ou minimizar uma função linear ( chamada função objetivo ), cujas variáveis estão sujeitas a um sistema de equações lineares de igualdades e/ou desigualdades ( chamadas restrições do problema ). A matriz associada a este sistema é chamada matriz tecnológica [02].

O problema pode ser escrito como :

$$\begin{aligned} \text{MIN } Z &= c_1 x_1 + c_2 x_2 + \dots + c_n x_n \\ \text{s.a } &\begin{cases} a_{11} x_1 + a_{12} x_2 + \dots + a_{1n} x_n = b_1 \\ a_{21} x_1 + a_{22} x_2 + \dots + a_{2n} x_n = b_2 \\ \vdots \\ a_{m1} x_1 + a_{m2} x_2 + \dots + a_{mn} x_n = b_m \end{cases} \end{aligned}$$

onde :

$Z$  : função objetivo

$c_j$  ,  $j = 1, 2, \dots, n$  : custo unitário da variável  $x_j$

$x_j$  ,  $j = 1, 2, \dots, n$  : variável de decisão

$a_{ij}$  ,  $i = 1, 2, \dots, m$  ;  $j = 1, 2, \dots, n$  : coeficientes tecnológicos

$b_i$  ,  $i = 1, 2, \dots, m$  : disponibilidade do recurso  $i$

Na forma matricial, temos :

$$\begin{aligned} \text{MIN } Z &= c^T x \\ \text{s.a } &\begin{cases} A x = b \\ x \geq 0 \end{cases} \quad (1) \end{aligned}$$

com  $A \in \mathbb{R}^{n \times m}$ .

Consideremos o problema (1). Supondo que temos uma base inicial  $B \in \mathbb{R}^{n \times n}$  e uma solução básica factível inicial. A matriz  $A$ , bem como os vetores  $x$  e  $c$ , podem ser particionados, de modo que:

$$A = [ B : N ]$$

$$x = [ x_B : x_N ]$$

$$c = [ c_B : c_N ]$$

Assim, o problema (1) pode ser escrito como:

$$\begin{aligned} \text{MIN } Z &= c_B^T x_B + c_N^T x_N \\ B x_B + N x_N &= b \\ \begin{bmatrix} x_B \\ \dots \\ x_N \end{bmatrix} &\geq 0 \end{aligned}$$

onde  $x_B$  e  $x_N$  são as variáveis chamadas básicas e não-básicas, respectivamente. O problema acima pode ser resolvido através do Método Simplex Revisado.

## 1.1 - MÉTODO SIMPLEX REVISADO

Escrevendo a inversa da base como um produto de matrizes de transformações elementares, temos

$$B^{-1} = E_n \cdot E_{n-1} \dots E_1$$

O método simplex revisado pode ser resumido como:

PASSO 0 :  $x_B = B^{-1}b$

$$x_N = 0$$

$$Z = c_B B^{-1}b$$

PASSO 1 : Cálculo do vetor  $\Pi$  ( multiplicadores ) :

$$\Pi^T = c_B^T B^{-1}$$

$$= c_B^T E_n, E_{n-1}, \dots, E_1$$

Este passo é conhecido como "backward transformation" (ou BTRAN ), pois as matrizes de transformações elementares são multiplicadas na ordem reversa (  $n, n-1, \dots, 1$  ).

PASSO 2 : Encontrar o custo reduzido de cada coluna  $a_j \in A$

$$z_j - c_j = c_B^T B^{-1}a_j - c_j$$

$$= \Pi^T a_j - c_j$$

$$\text{Considerar } \max_j z_j - c_j = z_q - c_q$$

Se  $z_q - c_q \leq 0$  → PARAR : a solução corrente é ótima. Caso contrário, ir para o Passo 3.

PASSO 3 : Se  $z_q - c_q > 0$ , calcular :

$$\alpha_q = B^{-1}a_q = E_n, E_{n-1}, \dots, E_1, a_q$$

Este passo é conhecido como "Forward transformation" ( FTRAN ), pois as matrizes de transformações elementares são multiplicadas na ordem direta (  $1, 2, \dots, n$  ).

PASSO 4 : Se  $\alpha_q \leq 0$ , PARAR : solução ilimitada. Caso  $\alpha_q > 0$ , determinar a variável básica que vai deixar a base. Fazendo  $\beta = B^{-1}b$ , escolhemos a linha pivô p, através do teste da razão :

$$\frac{\beta_p}{\alpha_p} = \min_i \left\{ \frac{\beta_i}{\alpha_i} \quad , \quad \alpha_i > 0 \right\}$$

PASSO 5 : atualizar  $B^{-1}$  e  $\beta$ , através da multiplicação por:

$$E_{n+1} = \begin{bmatrix} 1 & \dots & \eta_{1,n+1} & & \\ & \ddots & \eta_{p,n+1} & & \\ & & \eta_{n,n+1} & \dots & 1 \end{bmatrix}$$

onde :

$$\begin{cases} \eta_{i,n+1} = \frac{-\alpha_i}{\alpha_p} & (i \neq p) \\ \eta_{p,n+1} = \frac{1}{\alpha_p} \end{cases}$$

As matrizes  $E_k$ ,  $k = 1, 2, \dots, n$  são chamadas de "vetores ETA", pois cada uma delas é completamente definida através dos elementos  $\eta_{ik}$  e pelo índice da linha pivô. Estes vetores são armazenados sequencialmente num arquivo [18].

A nova inversa da base é dada por :

$$\begin{aligned} (\bar{B})^{-1} &= E_{n+1} \cdot B^{-1} \\ &= E_{n+1} \cdot E_n \dots E_1 \end{aligned}$$

E também :

$$\begin{aligned} \bar{\beta} &= E_{n+1} \cdot \beta \\ &= E_{n+1} \cdot E_n \dots E_1 \cdot b \end{aligned}$$

A atualização da inversa é feita através da forma produto da inversa (PFI) .

Existem basicamente dois métodos que são usados para inversão e atualização de bases em P.L. : PFI e EFI .

Durante muitos anos, a técnica padrão para atualização de bases, usada em pacotes, foi a forma produto da inversa (PFI).

Porém, este esquema não é tão eficiente quanto a forma de eliminação da inversa (EFI). É amplamente reconhecida a eficiência do esquema EFI sobre o PFI, em termos de rapidez e precisão [18].

A forma de eliminação da inversa (EFI) utiliza a fatoração triangular da base ( decomposição LU ) e tem sido implementada num grande número de pacotes comerciais de PL [18].

Na EFI, os fatores triangulares são usados na atualização da inversa da base, bem como nas iterações posteriores à reinversão.

Deste modo, o passo 5 ( de atualização de bases ), será alterado para um procedimento que utiliza fatores triangulares L e U.

Os métodos mais conhecidos para atualizar fatores triangulares de uma base são:

- (1) Método de Bartels e Golub
- (2) Método de Forrest e Tomlin

O método de Bartels e Golub é o único esquema estável para atualizar fatores LU de uma base [36], mas apresenta dificuldades na implementação computacional para problemas de grande porte, devido à perda de esparsidade nos fatores triangulares.

Já o método de Forrest e Tomlin é de fácil implementação computacional e muito atrativo do ponto de vista da esparsidade. Porém, é numericamente instável [36].

Existem implementações baseadas no método de Bartels e Golub, tais como as implementações de Saunders e de Reid, ambas estáveis e eficientes, que mantêm a esparsidade de bases.

Na implementação descrita por Saunders, são utilizadas as estruturas de bumps e spikes, obtidas com o algoritmo  $P^4$ . A base é decomposta em fatores triangulares. Através duma permutação simétrica, a matriz triangular superior passa a ter uma estrutura especial, e é armazenada explicitamente. A base inicial é modificada pela troca de uma de suas colunas. Através de permutação simétrica, obtém-se uma matriz que só não é triangular superior devido à sua última linha, composta por elementos não-nulos. Esta matriz é triangularizada por meio da eliminação Gaussiana.

A implementação deste método de atualização é usada no pacote MINOS/AUGMENTED.

No método descrito por Reid, a base original é decomposta em fatores triangulares, através do método de Markowitz. Após troca de uma coluna, a matriz triangular superior passa a apresentar um bump. Então, por meio de permutações de linhas e colunas, é conseguida uma estrutura que é, pelo menos, bloco triangular superior. Em seguida, a matriz é reduzida à estrutura triangular superior, através da eliminação Gaussiana.

Os métodos de Forrest-Tomlin e Bartels-Golub, bem como suas modificações, serão analisados a seguir.



## 2 - ESQUEMAS PARA ATUALIZAÇÃO DE BASES

### 2.1 - MÉTODO DE BARTELS-GOLUB

Devido à instabilidade numérica de PFI, Bartels e Golub [01] propuseram como alternativa, uma fatoração LU da base, que pode ser atualizada de maneira estável.

Considerando o problema :

$$\text{MIN } Z = c^T x$$

$$\text{s.t. } Ax = b$$

$$x \geq 0$$

onde  $A \in \mathbb{R}^{n \times m}$ , Ou então:

$$\text{MIN } Z = c_B^T x_B + c_N^T x_N$$

$$B x_B + N x_N = b$$

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} \geq 0$$

onde  $B \in \mathbb{R}^{n \times n}$  e  $N \in \mathbb{R}^{n \times (m-n)}$ .

Temos a fatoração da base inicial :

$$B = L U$$

$$\text{ou } L^{-1} B = U \quad (1)$$

A equação (1) pode ser escrita como :

$$L^{-1} (b_1 \dots b_p \dots b_n) = (u_1 \dots u_p \dots u_n)$$

onde  $b_i$ ,  $u_i$ ,  $i = 1, 2, \dots, n$  são as colunas das matrizes  $B$  e  $U$ , respectivamente.

Feita a decomposição  $LU$  da base inicial, as atualizações para  $L$  são representadas na forma produto, enquanto que a matriz  $U$  é armazenada e atualizada explicitamente [20].

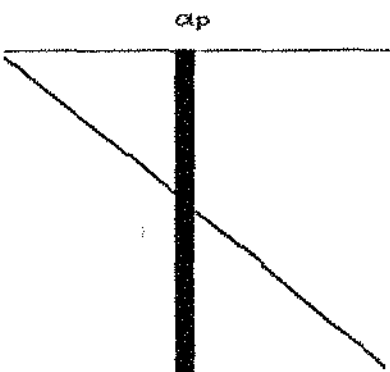
Supondo que numa iteração do método simplex, a coluna  $b_p$  seja trocada pela coluna  $a_q \in A$ . Então, a nova base é :

$$L^{-1} \tilde{B} = \tilde{U}$$

Considerando  $\alpha_p = L^{-1} a_q$  :

$$L^{-1} (b_1 \dots a_q \dots b_n) = (u_1 \dots \alpha_p \dots u_n)$$

Ou seja :

$$L^{-1} \tilde{B} =$$


Em geral,  $L^{-1} \tilde{B}$  não é mais triangular superior, diferindo da matriz  $U$ , apenas na coluna  $p$ .

Usando uma matriz de permutação  $Q$ , a coluna  $\alpha_p$  pode ser colocada no final da matriz  $L^{-1} \tilde{B}$ , obtendo então, uma matriz Hessenberg superior :

$$L^{-1} \tilde{B} Q = \tilde{U} Q = H =$$

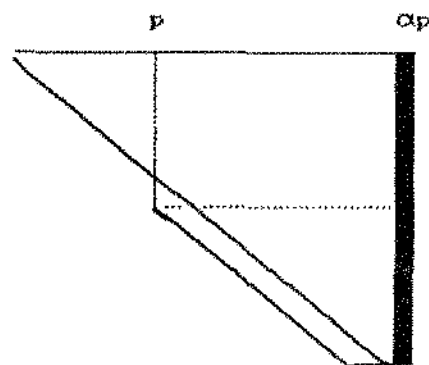


FIGURA 1

A matriz de permutação faz com que a coluna  $a_p$  seja colocada no final da matriz  $L^{-1} \tilde{B}$ , movendo as colunas intermediárias, de  $p+1$  até  $n$ , uma posição à esquerda.

Deste modo, a estrutura da FIGURA 1 pode ser facilmente reduzida à forma triangular superior, através da eliminação dos elementos da subdiagonal.

Ao serem eliminados os elementos não-nulos da subdiagonal, considerando como pivôs os elementos da diagonal principal, este método não será numericamente estável [41].

Assim, para cada coluna  $j = p, \dots, n-1$ , o pivô escolhido entre o elemento da diagonal ou da subdiagonal é aquele que tem maior valor absoluto. Portanto, deve ser permitida a permutação entre linhas, se necessário, na escolha dos pivôs [10],[26].

Então, através de uma sequência de matrizes de permutação (quando houver necessidade de troca de linhas) e de matrizes triangulares inferiores elementares (para eliminar elementos da subdiagonal), a estrutura triangular superior é obtida novamente.

Considerando a matriz  $G$  como resultante da sequência de matrizes necessárias para que a forma triangular superior seja obtida, tem-se :

$$G L^{-1} \tilde{B} Q = G \tilde{U} Q = G H = \bar{U}$$

ou

$$G L^{-1} \tilde{B} Q = \bar{U}$$

onde  $\bar{U}$  é uma matriz triangular superior. Tem-se então que :

$$\tilde{B} = L G^{-1} \bar{U} Q^T$$

A inversa da base  $\tilde{B}$  é dada por :

$$\boxed{(\tilde{B})^{-1} = Q (\bar{U})^{-1} G L^{-1}}$$

Embora o método seja altamente preciso, a cada eliminação de um elemento da subdiagonal, os elementos das linhas de  $p$  até  $n-1$  da matriz da FIGURA 1, vão sendo alterados, podendo desta forma, causar preenchimentos na matriz triangular superior. Assim, para atualizar a matriz  $U$ , é preciso uma estrutura de dados complexa, para que não haja perda de eficiência do método [20].

#### IMPLEMENTAÇÃO COMPUTACIONAL

A base  $B \in \mathbb{R}^{n \times n}$  é decomposta em fatores triangulares e :

$B = L_1 L_2 \dots L_n U_n U_{n-1} \dots U_1$ , onde:

$$L_k = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & l_{kk} & \\ & & | & \\ & & l_{nk} & \\ & & & 1 \end{bmatrix} \quad \text{e} \quad U_k = \begin{bmatrix} 1 & u_{1k} & & \\ & u_{k-1,k} & & \\ & & 1 & \\ & & & \ddots & 1 \end{bmatrix}$$

para  $k=1,2,\dots,n$ .

Os elementos de  $L_k^{-1}$  e  $U_k^{-1}$  têm apenas sinais opostos a dos obtidos em  $L_k$  e  $U_k$ , respectivamente.

A inversa é :

$$B^{-1} = U_1^{-1} \cdot U_2^{-1} \cdot \dots \cdot U_n^{-1} \cdot L_n^{-1} \cdot L_{n-1}^{-1} \cdot \dots \cdot L_1^{-1}$$

São utilizados dois arquivos para armazenar estas matrizes :  
 "backward ETA file " ( para  $U^{-1}$  ) e " forward ETA file " ( para  $L^{-1}$  ).  
 Cada uma das matrizes  $U_i^{-1}$  e  $L_i^{-1}$ ,  $i=1,2,\dots,n$  é armazenada como um vetor ETA [21],[39].

Supondo que a coluna  $b_p$  da base original foi modificada para  $a_q$ , tem-se que :

$$L^{-1} \tilde{B} = \tilde{U}$$

$$L_n^{-1} \cdot L_{n-1}^{-1} \cdot \dots \cdot L_1^{-1} \cdot [ b_1 \dots a_q \dots b_n ] = [ u_1 \dots a_p \dots u_n ]$$

Através de permutação de colunas, é obtida uma matriz Hessenberg superior, que será triangularizada por meio de eliminação Gaussiana.

$$L_n^{-1} \cdot L_{n-1}^{-1} \cdot \dots \cdot L_1^{-1} \tilde{B} Q = \tilde{U} Q = H$$

As matrizes  $U_i$ ,  $i=1,\dots,p-1$  permanecem inalteradas.  $U_p$  é apagada e as demais matrizes  $U_k$ ,  $k=p+1,\dots,n$  são modificadas para  $\tilde{U}_k$ .

A atualização das matrizes  $U_k$ , é realizada do seguinte modo. As matrizes  $G_s^{-1}$  ( $s=p, \dots, n-1$ ) são matrizes de transformações elementares, incluindo possível permutação entre elementos não-nulos da diagonal e subdiagonal de  $H$ . Então, para  $k=p+1, \dots, n$ , tem-se :

Passo 1 : Cálculo de  $G^{-1}$

$$G = \prod_{s=p}^{n-1} G_s^{-1} \quad . \quad \text{Obtenho } G^{-1}.$$

Passo 2 : Atualização

$$\tilde{U}_{k\_ETA} = G^{-1} U_{k\_ETA}$$

onde  $U_{k\_ETA}$  é a coluna não-unitária da matriz  $U_k$ . A partir de  $\tilde{U}_{k\_ETA}$ , a matriz  $\tilde{U}_k$  é construída, para  $k=p+1, \dots, n$ .

O último vetor ETA, relacionado à coluna  $\alpha_p$ , é atualizado por:

$$\begin{aligned} \alpha_p' &= G_{n-1}^{-1} . G_{n-2}^{-1} . \dots . G_p^{-1} . \alpha_p \\ &= G_{n-1}^{-1} . \dots . G_p^{-1} . L_n^{-1} . \dots . L_1^{-1} . \alpha_q \end{aligned}$$

Chamando  $\tilde{U}_{n+1}^{-1} = \alpha_p'$ , tem-se :

$$(\bar{U})^{-1} = U_1^{-1} . U_2^{-1} . \dots . U_{p-1}^{-1} . \tilde{U}_{p+1}^{-1} . \dots . \tilde{U}_n^{-1} . \tilde{U}_{n+1}^{-1}$$

O arquivo " backward ETA file " é escrito como :

$$U_1^{-1} . U_2^{-1} . \dots . \tilde{U}_{p-1}^{-1} . \dots . \tilde{U}_n^{-1} . \tilde{U}_{n+1}^{-1}$$

e o arquivo " forward ETA file", como

$$G_{n-1}^{-1} . G_{n-2}^{-1} . \dots . G_p^{-1} . L_n^{-1} . \dots . L_1^{-1}$$

## 2.1.1 - IMPLEMENTAÇÃO DE SAUNDERS

A implementação da atualização de Bartels-Golub feita por Saunders [36], tira proveito das estruturas de bump e spikes, obtidas através do algoritmo  $P^4$  [17], de Hellerman e Rarick. A estrutura conseguida para a matriz, ao final do algoritmo é triangular inferior ou bloco triangular inferior, quando a matriz for redutível [05].

Consideremos uma base  $B$ , após  $P^4$  :

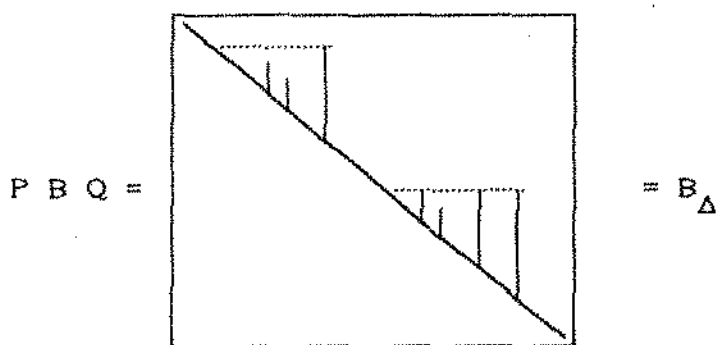
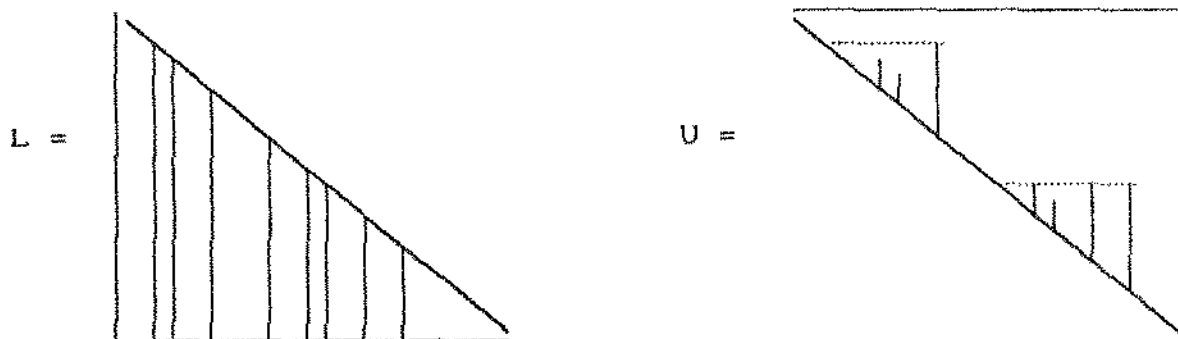


FIGURA 2

A fatoração  $L U$  da matriz da FIGURA 2 é a seguinte :



Os elementos da diagonal da matriz  $U$  são "1". Os preenchimentos (fill-in) que podem ocorrer eventualmente na decomposição  $LU$ , estão restritos às colunas spikes.

Nesta implementação, a matriz  $U$  é armazenada explicitamente. Portanto, será visto como obter uma melhor estrutura para  $U$ , de modo que o armazenamento e cálculo relacionados com seus elementos sejam realizados de forma eficiente.

### PARTICIONAMENTO DA MATRIZ $U$

Consideremos  $P$  uma matriz de permutação que mova todos os spikes para o final da matriz  $U$ , mantendo a ordem das colunas. Através de permutação simétrica, ou seja,  $P^T U P$ , a matriz passa a ter a seguinte estrutura :

$$\tilde{U} = \left[ \begin{array}{c|c} I & R \\ \hline 0 & F \end{array} \right]$$

FIGURA 3

onde  $F$  é triangular superior.

Será visto o que acontece à base, no caso de atualização, num passo do método simplex.

Consideremos :

$$\text{MIN } Z = c^T x$$

$$\text{s. a. } \begin{cases} Ax = b \\ x \geq 0 \end{cases} \quad , \text{ com } A \in \mathbb{R}^{n \times m}$$



Na forma matricial:

$$\text{MIN } Z = c_B^T x_B + c_N^T x_N$$

$$B x_B + N x_N = b$$

$$\begin{bmatrix} x_B \\ \dots \\ x_N \end{bmatrix} \geq 0, \text{ com } B \in \mathbb{R}^{m \times n}.$$

Utilizando o algoritmo  $P^4$ , se  $B$  for uma matriz redutível, temos :

$$P_1 B P_2 = B_{\Delta}$$

Fazendo a decomposição  $\tilde{L} U$  da base :

$$P_1 B P_2 = B_{\Delta} = \tilde{L} U$$

ou  $L^{-1} P_1 B P_2 = L^{-1} B_{\Delta} = U$

O próximo passo é particionar  $U$  para obter a estrutura da

FIGURA 2 .

$$P^T L^{-1} P_1 B P_2 P = P^T L^{-1} B_{\Delta} P = U'$$

Portanto, tem-se que :

$$P^T L^{-1} B_{\Delta} P = U' \quad (1)$$

A equação (1) pode ser escrita como :

$$P^T L^{-1} P_1 (b_1 \dots b_p \dots b_n) P_2 P = (u_1' \dots u_p' \dots u_n')$$

Supondo que a coluna  $b_p$  seja trocada pela coluna  $a_q \in A$ , numa iteração. Então, fazendo  $a_p = L^{-1} P_1 a_q$ , tem-se:

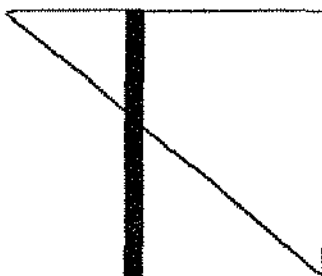
$$P^T L^{-1} P_1 (b_1 \dots a_q \dots b_n) P_2 P = (u_1' \dots \alpha_p \dots u_n')$$

ou  $P^T L^{-1} P_1 B^* P_2 P = \tilde{U}$

ou , ainda :

$$P^T L^{-1} \tilde{B} P = \tilde{U}$$

A estrutura obtida é :

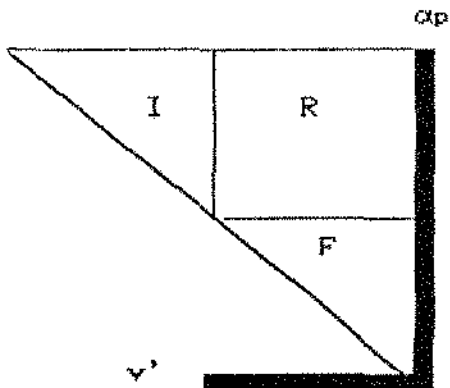
$$\tilde{U} =$$


The diagram shows a square matrix structure. A thick vertical line, representing a 'spike', runs through the matrix. A diagonal line runs from the top-left corner to the bottom-right corner. The matrix is partitioned into four quadrants by these lines: a triangle in the top-left, a rectangle in the top-right, a rectangle in the bottom-left, and a triangle in the bottom-right.

Permutando o spike para a última posição da matriz  $\tilde{U}$ , é obtida uma matriz Hessenberg superior :

$$P^T L^{-1} \tilde{B} P Q = \tilde{U} Q = H \quad (2)$$

Pré-multiplicando (1) por  $Q^T$ , ou seja, realizando permutação simétrica  $(Q^T \tilde{U} Q)$ , temos :

$$Q^T \tilde{U} Q =$$


The diagram shows a square matrix structure. The top-left quadrant is a triangle labeled 'I'. The top-right quadrant is a rectangle labeled 'R'. The bottom-left quadrant is a rectangle labeled 'F'. The bottom-right quadrant is a triangle. A thick vertical line runs along the right edge of the matrix, and a thick horizontal line runs along the bottom edge. The labels 'Qp' and 'v'' are placed near the top-right and bottom-left corners respectively.

FIGURA 4

Através da eliminação Gaussiana, a matriz da FIGURA 4 é reduzida a uma matriz triangular superior. Apenas o vetor linha  $v'$  e  $F$  estão envolvidos na eliminação.

Considerando  $G_1$  como a matriz resultante da sequência de matrizes necessárias na obtenção da matriz triangular superior, temos:

$$G_1 Q^T P^T L^{-1} \tilde{B} P Q = G_1 Q^T \tilde{U} Q = G_1 Q^T H = \bar{U}$$

A inversa da base  $\tilde{B}$  é dada por :

$$(\tilde{B})^{-1} = P Q (\bar{U})^{-1} G_1 Q^T P^T L^{-1}$$

#### DETALHES DE IMPLEMENTAÇÃO COMPUTACIONAL DO MÉTODO

(1) A matriz de restrições,  $A$ , é armazenada por colunas através de uma lista de índices de linhas e de apontadores para os valores numéricos dos coeficientes.

(2) A matriz  $U$  é particionada em  $R$  e  $F$ . Estas duas matrizes são armazenadas por colunas, mas separadamente.

A matriz  $R$  é armazenada como uma matriz esparsa e  $F$ , como uma matriz densa.

(3) A refatoração da base é feita após um certo número de iterações ( por exemplo,  $k$  ), ou sempre que não houver memória suficiente para atualizar  $L$ ,  $R$  ou  $F$ .

Se forem encontradas na base  $\underline{s}$  spikes (através de  $P^4$ ), serão reservadas  $t(t+1)/2$  posições de memória para  $F$ , onde  $t=s+k$ . Este valor  $\underline{t}$  é um limitante superior para a quantidade de spikes antes de uma refatoração da base. Temos à priori, que a base será refatorada a cada  $\underline{k}$  iterações. Desta forma, se houver a criação de um spike a cada iteração, poderão ser criados no máximo  $\underline{k}$  spikes, antes de cada refatoração. Assim, o número máximo de spikes entre a iteração em que a base possui  $\underline{s}$  spikes e a próxima refatoração é  $t=s+k$ . Das  $t(t+1)/2$  posições disponíveis de memória, primeiramente é armazenado  $F$  e após, no restante disponível, as matrizes  $R$  e  $L$ .

#### ESTABILIDADE

A sequência dos elementos pivôs obtida usando o algoritmo  $P^4$ , pode necessitar de reordenação para garantir estabilidade numérica da base inicial. Utilizando um critério de tolerância para estabilidade, com parâmetro  $(0 < u \leq 1)$ , pode ser decidido se um determinado pivô é aceito na ordenação estabelecida por  $P^4$ .

Porém, devido às estruturas de bumps e spikes, se um determinado pivô não for aceito, uma permutação entre linhas pode aumentar consideravelmente a quantidade total de spikes, além da perda da estrutura obtida usando o algoritmo. Portanto, devem haver permutações apenas entre colunas de um mesmo bump, para que não haja alteração da estrutura da matriz.

Consideremos a matriz  $A \in \mathbb{R}^{n \times n}$ . Após o algoritmo  $P^4$ , temos :

$$B = P A Q = L U \quad \text{e} \quad B = \langle a_{ij} \rangle, \quad i, j = 1, \dots, n.$$

Supondo que o elemento  $a_{kk} \neq 0$  não foi aceito como pivô, deve ser procurado nas outras colunas do bump, um elemento  $a_{kl}$  ( $l \neq k$ ), que satisfaça :

$$|a_{kl}| \geq u |a_{kk}| \quad l \neq k$$

Vejamos como realizar esta tarefa, uma vez que as matrizes são armazenadas por colunas. Fazendo  $L^T p = e_k$ , temos:

$$B = L U$$

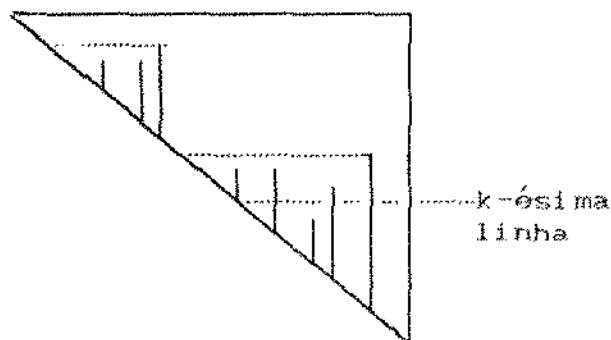
$$B^T = U^T L^T$$

$$B^T p = U^T L^T p$$

$$B^T p = U^T e_k$$

ou  $p^T B = e_k^T U$

$$p^T B = (0, \dots, 1, \dots, 0)$$



Os elementos de  $p^T B$  estão na k-ésima linha do bump. É escolhida como coluna para o novo elemento pivô :

$$|p^T b_l| = \max |p^T b_j|$$

onde  $j$  é índice de colunas spikes no bump e  $B = [b_1 \dots b_e \dots b_n]$ .

Se  $|a_{kl}| \geq u |a_{kj}|$ , o elemento  $a_{kl} \neq 0$  é o novo pivô.

## 2 . 1 . 2 - IMPLEMENTAÇÃO DE REID

Considerando a base inicial  $B$ , é usada a estratégia de Markowitz de escolha de pivôs para realizar a fatoraço~o LU e obter:

$$L^{-1} B = U$$

Modificando a base em uma coluna, temos:

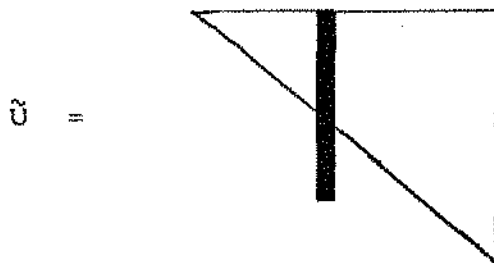
$$L^{-1} \tilde{B} = \tilde{U}$$

onde  $\tilde{U}$  é como a matriz da FIGURA 4.

A matriz  $\tilde{U}$  difere de  $U$ , apenas na p-ésima coluna.

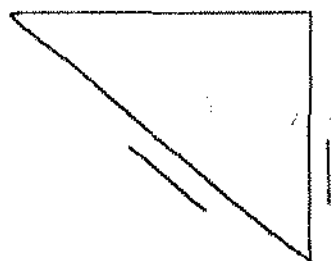
Foi sugerido por Saunders que poderia ser tirado vantagem do fato que, no caso em que as matrizes fossem esparsas, seria improvável que a coluna spike ( FIGURA 4 ) atingisse a última linha da matriz  $\tilde{U}$ .

Ou seja :



Assim, ao permutar a coluna spike para o final da matriz  $\tilde{U}$ , seria obtida uma matriz Hessenberg superior, com uma quantidade menor de elementos não-nulos na subdiagonal.

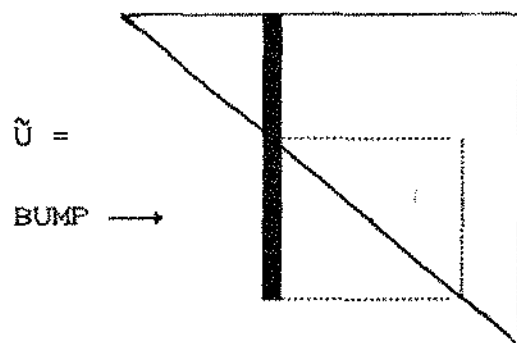
$$H = \tilde{U} Q :$$



Através da eliminação Gaussiana, a estrutura triangular superior seria conseguida novamente.

Foi proposto por Reid, antes de permutar o spike para o final da matriz  $\tilde{U}$ , tentar conseguir uma estrutura melhor, de modo que, ao realizar a eliminação Gaussiana, houvesse diminuição na quantidade de operações aritméticas, bem como na de fill-in.

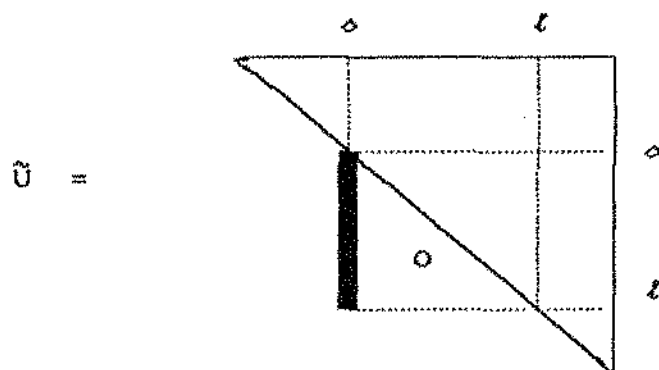
Supondo que o spike não atinge até a última linha de  $\tilde{U}$  :



O algoritmo de Tarjan poderia ser usado para encontrar uma estrutura triangular ou bloco triangular superior para o bump [106]. Porém, será visto um método descrito por Reid, que busca, através de permutações de linhas e colunas do bump, encontrar tais estruturas.

## MÉTODO

Consideremos  $\tilde{U} \in \mathbb{R}^{n \times n}$ ,



O spike está na coluna 0 e vai até a linha  $l$ . É tentado obter uma estrutura triangular superior para o bump. O método será descrito em termos gerais.

Temos, então :

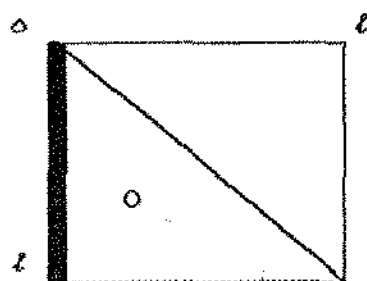


FIGURA 5

Com possível exceção do elemento  $(0,0)$ , que pertence ao spike, todos os outros elementos da diagonal principal são não-nulos, pois são originários da matriz  $U$ .

A partir do spike, são procuradas colunas que possuam um único elemento. Se a coluna  $k$  é a primeira que tem um único elemento ( FIGURA 6.a ), teremos então o elemento  $(k,k)$  na  $s$ -ésima posição diagonal, através de permutação simétrica.



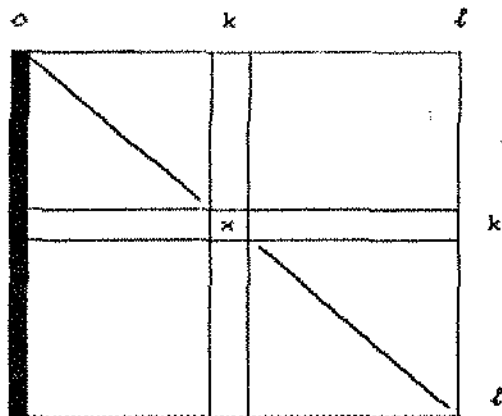


FIGURA 6. a

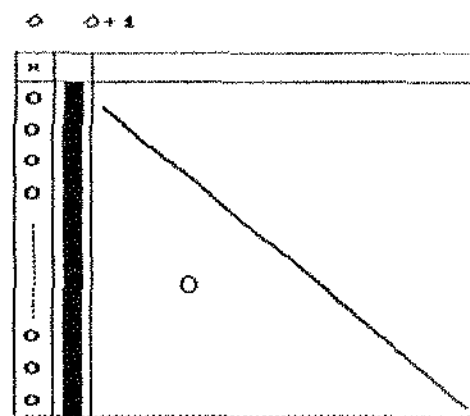
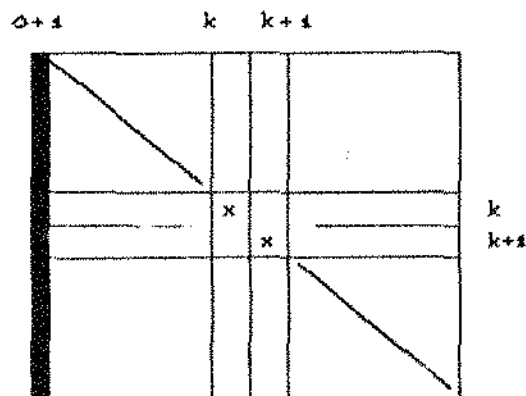


FIGURA 6. b

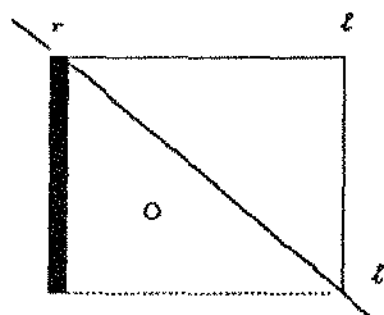
Os antigos elementos diagonais  $\delta+1, \delta+2, \dots, k-1$ , (agora  $\delta+2, \delta+3, \dots, k$ ), são deslocados uma posição à direita. Além disso, o spike é reduzido em uma posição, ocupando agora a coluna  $\delta+1$  (FIGURA 6. b).

Nenhuma das colunas, de  $\delta+2$  até  $k$  (anteriormente,  $\delta+1$  até  $k-1$ ) podem ter um único elemento. Portanto, o processo continua a partir da nova  $k$ -ésima coluna da submatriz obtida.

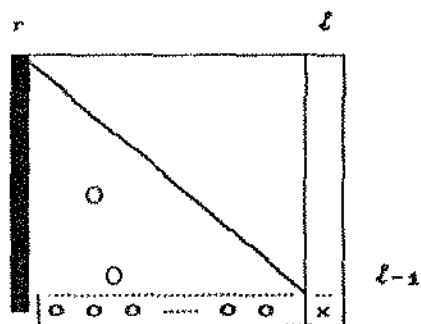


Este passo termina quando não se tem mais nenhuma coluna com um único elemento, chegando à coluna 1.

Em seguida é realizado um processo semelhante, procurando agora por linhas com um único elemento. Provavelmente, o bump original seja menor neste passo. Portanto, supõe-se que  $r \leq 0$ .



As linhas de  $m-1$  até  $r$  são examinadas, e é procurada uma que possua um único elemento. Se a próxima linha satisfaz esta condição, seu elemento é permutado para a  $m$ -ésima posição diagonal.



O motivo pelo qual se inicia a busca a partir da linha  $l-1$  é devido ao fato que existem sempre dois elementos nesta linha : o elemento que determina o comprimento do spike e o elemento diagonal, proveniente da matriz triangular superior.

O processo continua, a partir da linha  $p$ , até que não haja mais nenhuma linha com um único elemento.

Deve ser observado que as permutações realizadas neste passo são necessariamente simétricas.

O próximo passo é permutar o spike para o final do bump reordenado, para se obter uma matriz Hessenberg superior. Devido aos passos anteriores, a única coluna na qual é possível encontrar um único elemento é a coluna spike. Se isto ocorrer, os passos anteriores são repetidos até que:

- (1) não haja mais bump ( i.é, a matriz foi triangularizada )
- (2) a última coluna não possua um único elemento.

Para o caso (1),  $P S Q = U$  (o bump é triangular superior)

$$(\tilde{B})^{-1} = Q (U)^{-1} P L^{-1}$$

Para o caso (2),  $P S Q = H$  ( bump Hessenberg superior)

$$(\tilde{B})^{-1} = Q (U) G P L^{-1}$$

Cada pivô estará numa coluna que tem uma quantidade mínima de elementos não-nulos e dentro desta coluna, ele estará numa linha com quantidade mínima de elementos não-nulos.

### ARMAZENAMENTO DE MATRIZES

A matriz que mais apresenta dificuldades de armazenagem é a matriz triangular superior. Isto, devido à ocorrência de fill-in, durante a redução da matriz Hessenberg superior para a forma triangular superior, através da eliminação Gaussiana [34].

Um esquema adequado para armazenar matrizes esparsas é o esquema de lista-ligada ( linked list ). À cada elemento não-nulo da matriz, estão associados dois números inteiros : um indica em que coluna o elemento não-nulo se encontra ( JCOL ) e o outro, onde é encontrado o próximo elemento da mesma linha ( LINK ).

#### EXEMPLO

$$A = \begin{bmatrix} 7.3 & 2.7 & & 6.2 \\ & 0.9 & & \\ & & 6.1 & 2.1 \\ & & & 3.7 \end{bmatrix}$$

|        | 1   | 2   | 3   | 4   | 5   | 6   | 7 | 8   |
|--------|-----|-----|-----|-----|-----|-----|---|-----|
| ISTART | 6   | 8   | 3   | 5   |     |     |   |     |
| VALUE  | 2.7 | 6.2 | 6.1 | 2.1 | 3.7 | 7.3 | - | 0.9 |
| JCOL   | 2   | 4   | 3   | 4   | 4   | 1   | - | 2   |
| LINK   | 2   | 0   | 4   | 0   | 0   | 1   | - | 0   |

Como se lê :

LINHA 1 - ISTART(1) = 6

VALUE = 7.3      JCOL = 1      LINK = 1

VALUE = 2.7      JCOL = 2      LINK = 2

VALUE = 6.2      JCOL = 4      LINK = 0 → não há mais

elementos não-nulos nesta linha. Ir para a próxima.

LINHA 2 - ISTART(2) = 8

VALUE = 0.9      JCOL = 2      LINK = 0 → não há mais

elementos não-nulos nesta linha. Ir para a linha 3.

LINHA 3 - ISTART(3) = 3

VALUE = 6.1      JCOL = 3      LINK = 4

VALUE = 2.1      JCOL = 4      LINK = 0 → ir para a linha 4

LINHA 4 - ISTART(4) = 5

VALUE = 3.7 JCOL = 4 LINK = 0. Final da matriz.

Outra possibilidade é usar o esquema sugerido por Gustavson [22], onde são necessários dois arquivos. Num deles, são armazenados, por linhas, os valores dos elementos não-nulos, juntamente com os índices de suas respectivas colunas. O outro arquivo contém os índices das linhas dos elementos não-nulos que existem em cada coluna da matriz.

#### EXEMPLO

|        | 1 | 2   | 3   | 4   | 5 | 6   | 7   | 8   | 9 | 10  |
|--------|---|-----|-----|-----|---|-----|-----|-----|---|-----|
| ISTART | 3 | 8   | 6   | 10  |   |     |     |     |   |     |
| I      | 3 | 1   | 2   | 1   |   |     |     |     |   |     |
| VALUE  | - | 7.3 | 2.7 | 6.2 | - | 6.1 | 2.1 | 0.9 | - | 3.7 |
| JCOL   | - | 1   | 2   | 4   | - | 3   | 4   | 2   | - | 4   |

onde  $I(i)$  : quantidade de elementos não-nulos na linha  $i$ ,  $i=1,\dots,4$

|        | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------|---|---|---|---|---|---|---|---|---|----|
| JSTART | 3 | 1 | 6 | 8 |   |   |   |   |   |    |
| J      | 1 | 2 | 1 | 3 |   |   |   |   |   |    |
| ICOL   | 1 | 2 | 1 | - | - | 3 | - | 1 | 3 | 4  |

onde  $J(i)$  : quantidade de elementos não-nulos na coluna  $i$ ,  $i=1,\dots,4$

Durante o processo de eliminação, ao ser escolhido o elemento pivô na matriz  $H$ , a linha pivô passa a ocupar novas posições nos arquivos, e suas antigas são apagadas.

Se ocorrer fill-in durante a eliminação Gaussiana, ambos arquivos devem ser atualizados e, havendo necessidade, devem ser aumentados para que seja possível incluir o novo elemento.

## 2.2 - MÉTODO DE FORREST E TOMLIN

Forrest e Tomlin [18], [39] propuseram uma variação do método de atualização de fatores triangulares de Bartels e Golub.

Embora o método seja atrativo por explorar a esparsidade, não permitindo fill-in nos fatores triangulares da base, o método não é numericamente estável [36].

Considerando a fatora<sup>ção</sup> da base B:

$$B = L U$$

pode ser escrita como :

$$L^{-1} ( b_1 \dots b_p \dots b_n ) = ( u_1 \dots u_p \dots u_n )$$

Supondo que numa iteração, p é a linha pivô e que a coluna da base,  $b_p$ , é trocada pela coluna  $a_q \in A$ , temos que :

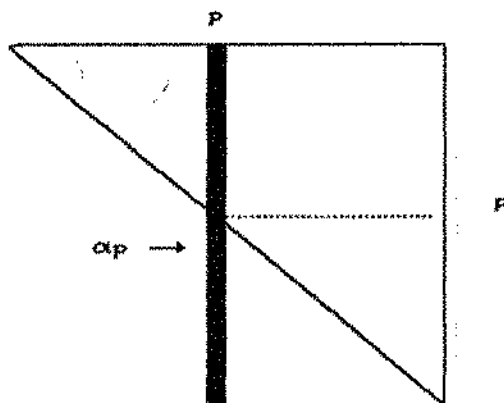
$$L^{-1} \tilde{B} = \tilde{U}$$

onde, provavelmente,  $\tilde{U}$  não é mais triangular superior, diferindo de U apenas na coluna p. Considerando  $\alpha_p = L^{-1} a_q$ , obtemos :

$$L^{-1} ( b_1 \dots a_q \dots b_n ) = ( u_1 \dots \alpha_p \dots u_n )$$

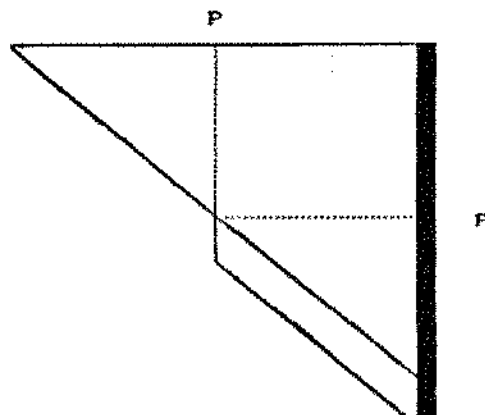
Ou, ainda,

$$L^{-1} \tilde{B} = \tilde{U} =$$



Através de uma matriz de permutação  $Q$ , a coluna  $\alpha_p$  pode ser colocada no final da matriz  $L^{-1} \tilde{B}$ :

$$L^{-1} \tilde{B} Q = H =$$



A matriz  $H$  é Hessenberg superior e pode facilmente ser reduzida à estrutura triangular superior.

É neste passo que o método de Forrest - Tomlin difere do de Bartels - Golub.

A estratégia usada para esta tarefa é: eliminar os elementos da linha  $p$ , nas colunas de  $p$  até  $n-1$  da matriz  $H$ , ou seja, eliminar os elementos  $h_{pp}, \dots, h_{p,n-1}$ .

Utilizando uma matriz elementar linha  $R^{-1}$ , onde :

$$R^{-1} = I - e_p r^T$$

$$r^T = (0, 0, \dots, 0, r_{p+1}, \dots, r_n)$$

temos :

$$R^{-1} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & r_{p+1} & \dots & r_n \\ & & 1 & & \\ & & & 1 & \end{bmatrix}$$

Como as colunas de  $p$  até  $n-1$  da matriz  $H$  são idênticas às colunas de  $p+1$  até  $n$  da matriz original  $U$ , a mesma matriz  $R^{-1}$  eliminará os elementos  $h_{pp}, \dots, h_{p,n-1}$  de  $H$ , ou  $u_{p,p+1}, \dots, u_{pn}$  de  $U$ .

Este fato pode ser utilizado para simplificar o cálculo de  $r^T$ . Deseja-se obter  $r^T$  de modo que os elementos  $u_{p,p+1}, \dots, u_{pn}$  sejam nulos. Para isto, é preciso resolver o seguinte sistema :

$$r^T = \bar{u}^T U^{-1} \quad (1)$$

onde  $\bar{u}^T = (0, 0, \dots, u_{pp}, u_{p,p+1}, \dots, u_{pn})$

A equação (1) pode ser escrita como :

$$U^T r = \bar{u}$$

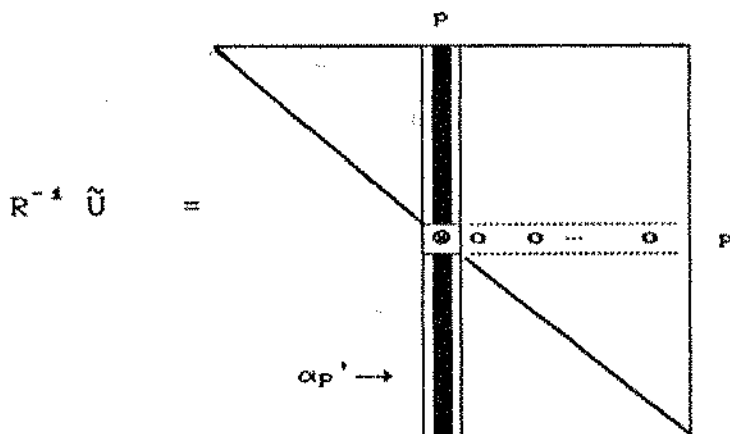
Resolvendo o sistema, temos  $r^T$  e a matriz  $R^{-1}$  pode ser obtida facilmente, pois :

$$R^{-1} = I - e_p r^T$$



$$\begin{aligned} R^{-1} U &= (I - \epsilon_p r^T) U \\ &= U - \epsilon_p r^T U \\ &= U - \epsilon_p \bar{U}^T U^{-1} U \\ &= U - \epsilon_p \bar{U}^T \end{aligned}$$

Usando a mesma idéia para  $\tilde{U}$ , temos então que:



A coluna  $\alpha_p$  agora é  $\alpha_p' = R^{-1} L^{-1} a_q$ .

Multiplicando a matriz da FIGURA 7, por uma matriz de permutação  $Q$ , de tal forma que a coluna  $\alpha_p$  seja colocada no final da matriz, temos :

$$R^{-1} \tilde{U} Q =$$

Pós-multiplicando  $R^{-1}H$  por  $Q^T$ , isto é, obtendo uma permutação simétrica,  $(Q^T R^{-1} \tilde{U} Q)$ , teremos :

$$Q^T R^{-1} \tilde{U} Q =$$

Os elementos que faziam parte da subdiagonal após esta permutação, são os elementos da diagonal principal. As linhas de  $p+1$  a  $n$  da matriz  $R^{-1} \tilde{U} Q$  são agora as linhas de  $p$  a  $n-1$  da matriz  $Q^T R^{-1} \tilde{U} Q = \bar{U}$ .

Portanto, temos que :

$$\bar{U} = Q^T R^{-1} \tilde{U} Q$$

onde  $\bar{U}$  é triangular superior.

Mas  $\tilde{U} = L^{-1} \tilde{B}$ . Deste modo:

$$\bar{U} = Q^T R^{-1} L^{-1} \tilde{B} Q$$

ou

$$\tilde{B} = L R Q \bar{U} Q^T$$

A inversa da nova base  $\tilde{B}$  é dada por :

$$(\tilde{B})^{-1} = Q (U)^{-1} Q^T R^{-1} L^{-1}$$

#### IMPLEMENTAÇÃO COMPUTACIONAL

Como no método de Bartels e Golub, supondo que houve modificação na base original, devido à alteração de sua p-ésima coluna pela coluna  $a_q$ , a atualização de  $U$  processa-se do seguinte modo:

As matrizes  $U_j$ ,  $j = 1, 2, \dots, p-1$  permanecem inalteradas.  $U_p$  é apagada e  $U_k$ ,  $k = p+1, \dots, n$  são modificadas para  $\tilde{U}_k$ .

É aqui que os dois métodos diferem. No caso do método de Forrest e Tomlin, os novos  $\tilde{U}_k$  ( $k = p+1, \dots, n$ ) diferem de  $U_k$  apenas por terem os elementos  $u_{pk}$  iguais a zero.

O vetor  $r^T$ , necessário para a construção de  $R^{-1}$ , é calculado através de :

Passo 1 : criar um vetor linha  $r^T = (0,0,\dots,0)$ . Deixar as matrizes  $U_i^{-1}$ ,  $i=1,\dots,p-1$  inalteradas. Apagar a matriz  $U_p^{-1}$ . Fazer  $k=p+1$ .

Passo 2 : Para  $U_k^{-1}$ , se  $u_{pk} \neq 0$ , colocar este valor na  $k$ -ésima posição de  $r^T$  e fazer o elemento  $u_{pk}=0$  em  $U_k^{-1}$ , obtendo assim,  $\tilde{U}_k^{-1}$ . Fazer  $r^T \tilde{U}_k^{-1}$  para conseguir novo  $r^T$ . Repetir o mesmo procedimento para  $k=p+1,\dots,n$ .

A dificuldade relacionada ao preenchimento ( fill-in ) é eliminada, mas o preço é a perda da estabilidade numérica do Método de Forrest e Tomlin.

## CAPÍTULO IV : TÉCNICAS DE INVERSÃO E ATUALIZAÇÃO DE BASES EM PACOTES COMPUTACIONAIS

### 1 - INTRODUÇÃO

MPSX (Mathematical Programming System Extended), na linguagem PL/I e MINOS (Modular in-core Nonlinear Optimization System), em FORTRAN são dois pacotes computacionais, de ampla utilização na resolução (em particular), de problemas de programação linear de médio e grande porte.

Nosso objetivo neste capítulo é analisar nestes pacotes, os procedimentos para obter a inversa de uma base e sua posterior atualização, baseado nos estudos realizados nos capítulos 2 e 3.

No MPSX/370, o procedimento de inversão difere um pouco dos métodos estudados no capítulo 2. É um método semelhante ao de Markowitz, porém não tão eficiente no que se refere à preenchimentos no cálculo da inversa e portanto, será visto com mais detalhes neste capítulo. A atualização da inversa da base é feita através do método de Forrest e Tomlin.

Existem duas versões do pacote MINOS. Na mais antiga (1977), MINOS/AUGMENTED, primeiramente é escolhida uma sequência de pivôs, para permutar a base e obter uma estrutura mais adequada para realizar a decomposição LU e posterior inversão.

Esta sequência de pivôs é realizada através de um processo de dois estágios: encontrar uma transversal máxima e obter permutações simétricas (algoritmo de Tarjan). A cada bloco irredutível, conseguido ao final deste processo, é aplicada uma versão simplificada do algoritmo  $P^3$ , para que haja uma melhor ordenação de linhas e colunas dentro de cada bump externo. A atualização da base é realizada da forma descrita por Saunders (capítulo 3) [36].

Já o pacote MINOS 5.0 (1983), é semelhante ao seu antecessor. Certas rotinas do MINOS/AUGMENTED foram substituídas por um pacote de rotinas para calcular e atualizar uma fatoração LU, que é o trabalho descrito e implementado por Reid (capítulo 3) [34].

## 2 - MPSX

O pacote MPSX/370 foi criado com o objetivo de resolver ampla classe de problemas de programação linear de médio e grande porte [03].

Serão vistas as técnicas usadas para cálculo da inversa de uma base e atualização, além de como é feito o controle de estabilidade numérica.

### CÁLCULO DA INVERSA DE UMA BASE

Neste pacote, a forma de eliminação da inversa (EFI) foi escolhida, por ser um esquema mais eficiente que o PFI.

A base  $B \in \mathbb{R}^{n \times n}$  é decomposta em fatores triangulares:

$$B = L U$$

e cada uma das matrizes  $L$  e  $U$  é facilmente decomposta num produto de matrizes elementares  $L_i$  e  $U_i$ ,  $i=1,2,\dots,n$ , tal que:

$$B = L_1.L_2.\dots.L_n.U_n.U_{n-1}.\dots.U_1$$

onde

$$L_k = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & l_{kk} & \\ & & \vdots & \ddots \\ & & l_{nk} & \dots & 1 \end{bmatrix} \quad U_k = \begin{bmatrix} 1 & \dots & u_{1k} & \\ & \ddots & \vdots & \\ & & u_{k-1,k} & \\ & & 1 & \ddots \\ & & & & 1 \end{bmatrix}$$

para  $k=1,2,\dots,n$ .

A estrutura de dados utilizada no pacote MPSX/370 foi baseada no trabalho desenvolvido por Gustavson [22].

O cálculo da inversa é realizado em duas fases:

(I) Fase Booleana e

(II) Fase Numérica

## (I) FASE BOOLEANA

Nesta fase, não se trabalha com a base original, mas com uma matriz booleana associada, ou seja, onde existir elemento não-nulo, colocaremos o valor "1" na posição correspondente e onde houver elemento nulo, um "0".

# EXEMPLO

|     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|
| 0.5 |     | 0.4 |     | 1.0 |     |
|     |     | 1.0 |     | 4.0 |     |
| 8.0 |     | 7.9 | 3.9 |     | 7.6 |
|     | 6.4 |     | 2.1 |     | 2.9 |
| 4.0 |     | 2.0 |     | 3.0 | 8.0 |
|     |     |     | 5.0 | 6.0 |     |

MATRIZ ORIGINAL

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 |

MATRIZ BOOLEANA

O objetivo nesta fase é encontrar uma sequência de pivôs, de modo que haja pequena quantidade de preenchimentos no cálculo da inversa da matriz.

Dada uma matriz  $A \in \mathbb{R}^{n \times n}$  (não-singular), são definidos alguns vetores para  $i=1,2,\dots,n$  e  $j=1,2,\dots,n_1$  ( $n_1 \leq n \times n$ ).

$I(i)$  : quantidade de elementos não-nulos na coluna  $i$

$J(i)$  : quantidade de elementos não-nulos na linha  $i$

$INCC(j)$  : índice das colunas dos elementos não-nulos, por ordem crescente de linhas.

$INRC(j)$  : índice das linhas dos elementos não-nulos, por ordem crescente de colunas.

No caso do exemplo dado, temos:

|       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ICD   | 3 | 1 | 4 | 3 | 4 | 3 |   |   |   |   |   |   |   |   |   |   |   |   |
| JCD   | 3 | 2 | 4 | 3 | 4 | 2 |   |   |   |   |   |   |   |   |   |   |   |   |
| INCCD | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 6 | 6 |
| INRCD | 1 | 3 | 5 | 3 | 5 | 1 | 3 | 4 | 6 | 2 | 4 | 6 | 1 | 3 | 5 | 6 | 4 | 5 |



A fase booleana é composta por três passos:

**PASSO 1** - Selecionar colunas que tenham uma linha com único elemento. Escolhido o pivô, atualizar  $IC(i)$  e  $J(i)$ . Eliminar linha e coluna do pivô, dos vetores  $INRC(i)$  e  $INCC(i)$ .

EXEMPLO : pivô 1 -  $a_{42}$

|       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ICD   | 3 | * | 4 | 2 | 4 | 2 |   |   |   |   |   |   |   |   |   |   |   |   |
| JCD   | 3 | 2 | 4 | * | 4 | 2 |   |   |   |   |   |   |   |   |   |   |   |   |
| INCCD | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 6 | 6 |
| INRCD | 1 | 3 | 5 | 3 | 5 | 1 | 3 | 4 | 6 | 2 | 4 | 6 | 1 | 3 | 5 | 6 | 4 | 5 |

**PASSO 2** - Selecionar linhas que tenham uma coluna com único elemento. Proceder de maneira análoga ao passo anterior.

Com a sequência de pivôs obtida nos dois passos, tem-se a seguinte estrutura para a matriz:

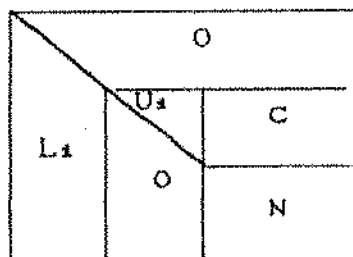


FIGURA 1

As estruturas  $L_1$  e  $U_1$  são obtidas usando os passos 1 e 2, respectivamente.

A submatriz  $N$ , que pode não ter sido triangularizada após os dois passos é chamada de NUCLEUS.

**PASSO 3** - A eliminação Gaussiana é aplicada à base de forma simbólica, a fim de decompô-la num produto de matrizes triangular inferior e superior. Os pivôs são escolhidos no NUCLEUS através de uma regra heurística, que procura minimizar os preenchimentos. O método é descrito abaixo:

(A) Escolher coluna com mínimo  $I(i)$ , i.é., a coluna com a menor quantidade de elementos não-nulos. Em caso de empate, escolhe-se a coluna que tem  $J(i)$  mínimo,  $i=1,2,\dots,n$ .

#### EXEMPLO

|        | 1 | 2 | 3 | 4 | 5 | 6 | $J(i)$ |
|--------|---|---|---|---|---|---|--------|
| 1      | x | x | x | x | x | x | 6      |
| 2      | x | x |   | x | x | x | 5      |
| 3      |   |   | x |   | x | x | 3      |
| 4      | x |   | x |   |   |   | 2      |
| 5      |   | x | x | x | x |   | 4      |
| 6      |   | x | x | x | x |   | 4      |
| $I(i)$ | 3 | 4 | 5 | 4 | 5 | 3 |        |

No exemplo acima, embora haja empate entre  $I(1)$  e  $I(6)$ , a coluna 1 deve ser escolhida, pois  $\min(J(i))=2$ , enquanto que, para a coluna 6,  $\min(J(i))=3$ .

(B) Escolhida a coluna pivô, escolher a linha pivô que tenha  $J(i)$  mínimo.

Sempre que um pivô é escolhido, a eliminação Gaussiana é aplicada e os vetores de linhas e colunas são atualizados de modo simbólico. Além disso,  $I(i)$  e  $J(i)$ , também são atualizados.

Nem sempre esta regra heurística fornece o melhor resultado. Consideremos o exemplo abaixo: supondo que nos passos (1) e (2) seja obtida a seguinte submatriz  $N$ :

|        |   |   |   |   |   |   |   |        |
|--------|---|---|---|---|---|---|---|--------|
| $N =$  | x | x | x | x | x | x | x | $J(i)$ |
|        |   | x | x |   |   |   |   | 7      |
|        | x | x | x | x |   | x | x | 2      |
|        |   | x |   |   | x | x | x | 7      |
|        | x |   |   |   | x | x | x | 5      |
|        |   | x |   | x | x | x |   | 4      |
|        |   | x |   |   |   |   | x | 3      |
|        | x |   |   |   |   | x |   | 2      |
| $I(i)$ | 2 | 8 | 3 | 3 | 3 | 5 | 4 | 5      |
|        |   |   |   |   |   |   | x | 3      |

- Escolha de pivô através da regra heurística

(A) coluna com  $I(i)$  mínimo: coluna 1

(B) linha com  $J(i)$  mínimo: linha 1 ( ou 3 )

Elemento pivô:  $n_{11}$  (poderia ser  $n_{31}$ )

Após eliminação Gaussiana simbólica:

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| ⊗ | x | x | x | x | x | x | x |
|   | x | x |   |   |   |   |   |
| ⊗ | ⊗ | ⊗ | * | ⊗ | x | ⊗ |   |
|   | x |   |   | x | x | x | x |
|   | x |   | x | x | x |   |   |
|   | x |   |   |   |   | x | x |
|   | x |   |   |   | x |   |   |
|   | x |   |   |   |   | x | x |

onde:

⊗ : pivô

⊗ : elemento com valor alterado

\*

- Escolha de pivô através do Método de Markowitz

Custo Markowitz associado ao elemento  $n_{ij}^{(k)}$ :

$$M_{ijk} = (r_i^{(k)} - 1) * (c_j^{(k)} - 1) \quad i, j = k, \dots, n.$$

No caso,  $k=1$ :

$$M_k = \min \{ M_{ijk} \mid i, j = k, \dots, n \}, \text{ no estágio } k=1.$$

$n_{ij}^{(k)}$  é escolhido, tal que  $M_{ijk} = M_k$ .

Logo,  $n_{22}$  é o elemento pivô neste estágio.

Após eliminação Gaussiana simbólica:

|   |   |   |   |   |   |   |     |
|---|---|---|---|---|---|---|-----|
| x | ⊖ |   | x | x | x |   | x   |
|   | x | ⊗ |   |   |   |   |     |
| x | ⊖ |   | x |   | x | x | x   |
|   | x |   |   | x | x | x | x   |
|   |   |   | x | x | x |   |     |
|   | x |   |   |   |   |   | x x |
|   |   |   |   |   | x |   |     |
|   |   |   |   |   |   |   | x x |

onde:

⊗ : pivô

⊖ : elemento com valor alterado

x : elemento criado

Podemos observar que a escolha do pivô usando a regra heurística proposta não é tão eficiente, comparada com o Método de Markowitz.

Os dois primeiros passos do esquema proposto são idênticos ao do Método de Markowitz, pois são escolhidos os elementos que têm custo Markowitz  $M_{ijk}=0$ . Nos passos (1) e (2), são procuradas linhas e colunas com único elemento, respectivamente. Porém, no passo (3), a regra heurística mostra-se não tão eficiente quanto o Método de Markowitz.

O elemento escolhido como pivô através desta regra, pode não ter custo Markowitz mínimo. Assim um elemento não muito conveniente pode ser escolhido como pivô, num determinado estágio da eliminação Gaussiana simbólica.

Deste modo, o esquema proposto para escolha de sequência de pivôs no pacote MPSX/370, não é tão eficiente quanto ao de Markowitz, embora seja muito parecido.

Deve ser observado que na fase booleana, a escolha da sequência de pivôs visa apenas a preservação da esparsidade, independente de quaisquer considerações numéricas.

## (II) - FASE NUMÉRICA

Nesta fase, a inversa da base é calculada, utilizando a sequência de pivôs encontrada na fase booleana. A eliminação Gaussiana é aplicada ao NUCLEUS e:

$$N = L_2 U_2$$

ou

$$B = \begin{array}{|c|c|c|} \hline & & 0 \\ \hline L_1 & U_1 & C \\ \hline & 0 & U_2 \\ & & L_2 \\ \hline \end{array}$$

Juntando as submatrizes C e  $U_2$ , temos:

$$B = \begin{array}{|c|c|c|} \hline & & 0 \\ \hline L_1 & U_1 & \bar{U}_2 \\ \hline & 0 & L_2 \\ \hline \end{array}$$

ou

$$B = \begin{bmatrix} L_1 & L_2 & \bar{U}_2 & U_1 \end{bmatrix}$$

e

$$\bar{U}_2 = \begin{bmatrix} C \\ \hline L_2^{-1} N \end{bmatrix}$$

Deve ser observado que a matriz C não é atualizada.

Os pivôs nesta fase são testados de acordo com considerações numéricas, pois um pivô com valor muito pequeno com relação aos demais elementos da coluna, pode resultar em valores muito grandes para estes elementos, prejudicando a estabilidade numérica.

## ESQUEMA DE ATUALIZAÇÃO DA INVERSA

No pacote MPSX/370, a atualização da inversa da base utiliza o esquema de Forrest e Tomlin (capítulo 3) [18], [39], [40].

Considerando uma base  $B \in \mathbb{R}^{n \times n}$ , onde  $B = [b_1 \dots b_p \dots b_n]$ .

A decomposição LU desta base fornece:

$$B = L U$$

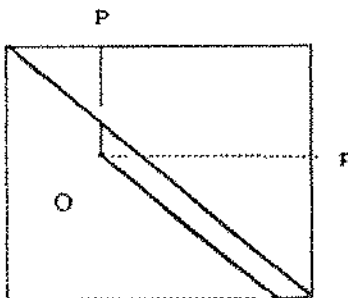
$$[b_1 \dots b_p \dots b_n] = L U \quad (1)$$

Supondo que a coluna  $b_p$  da base em (1) foi modificada para  $a_q$ . Fazendo  $\alpha_p = L^{-1}a_q$ , temos:

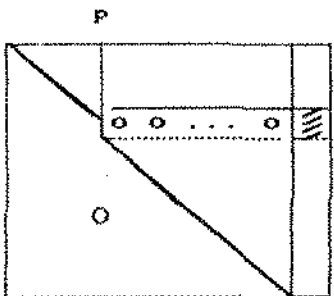
$$L^{-1} [b_1 \dots a_q \dots b_n] = [u_1 \dots \alpha_p \dots u_n]$$

$$L^{-1} \tilde{B} = \tilde{U}$$

Geralmente, a matriz  $\tilde{U}$  não é mais triangular superior. Permutando a coluna  $\alpha_p$  para o final da matriz:

$$L^{-1} \tilde{B} Q = \tilde{U} Q = H =$$


onde  $H$  é uma matriz Hessenberg superior. Os elementos  $h_{pp}, \dots, h_{p,n-1}$  de  $H$  devem ser eliminados, através de  $R^{-1}$ , ou seja:

$$R^{-1} L^{-1} \tilde{B} Q = R^{-1} H =$$

(2)

A coluna  $\alpha_p$ , atualizada, é:

$$\alpha_p = R^{-1} L^{-1} a_q$$

Pós-multiplicando (2) por  $Q^T$ , para que a p-ésima linha passe a ser a n-ésima, obtém-se:

$$Q^T R^{-1} \tilde{U} Q = \bar{U}$$

onde  $\bar{U}$  é triangular superior.

#### Como se procede computacionalmente

Inicialmente:

$$B = L_1 L_2 \dots L_n U_n U_{n-1} \dots U_1$$

onde  $U_i$  e  $L_i$  são matrizes de transformações elementares.

A inversa da base é:

$$B^{-1} = U_1^{-1} U_2^{-1} \dots U_n^{-1} L_n^{-1} L_{n-1}^{-1} \dots L_1^{-1}$$

As matrizes  $U^{-1}$  e  $L^{-1}$  são armazenadas em dois arquivos distintos, como vetores ETA.

Supondo que a coluna  $b_p$  foi trocada pela coluna  $a_q$  na base, a atualização desta nova base se processa como segue:

(1) Calcular

$$\alpha_p = L^{-1} a_q = L_n^{-1} L_{n-1}^{-1} \dots L_1^{-1} a_q$$



(2) Calcular  $r^T$

Criar  $r^T = (0, 0, \dots, 0)$

As matrizes  $U_j^{-1}$ ,  $j=1, 2, \dots, p-1$  permanecem inalteradas.  $U_p^{-1}$  é apagada. Repetir para  $k=p+1, \dots, n$ :

Para  $U_k^{-1}$ , se  $u_{pk} \neq 0$ , colocar este valor na  $k$ -ésima posição do vetor  $r^T$  e fazer o elemento  $u_{pk} = 0$  em  $U_k^{-1}$ , obtendo  $\tilde{U}_k^{-1}$ . Atualizar  $r^T$  através de:  $r^T \tilde{U}_k^{-1}$ . Repetir este mesmo processo para o próximo  $k$ , até que  $k=n$ .

Em seguida, obtido  $r^T$ , construir  $R^{-1}$  e fazer:

$$R^{-1} L^{-1} \tilde{B} = R^{-1} \tilde{U} Q = R^{-1} U_1^{-1} U_2^{-1} \dots U_{p-1}^{-1} \tilde{U}_{p+1}^{-1} \dots \tilde{U}_n^{-1} \tilde{U}_{n+1}^{-1}$$

onde:

$$\tilde{U}_{n+1}^{-1} = R^{-1} L_n^{-1} \dots L_1^{-1} a_q$$

Permutando a  $p$ -ésima linha para a última posição de  $H$ , obtém-se:

$$\bar{U} = Q^T R^{-1} U_1^{-1} \dots U_{p-1}^{-1} \tilde{U}_{p+1}^{-1} \dots \tilde{U}_n^{-1} \tilde{U}_{n+1}^{-1} Q$$

e  $\bar{U}$  é triangular superior.

## CONTROLE DA ESTABILIDADE NUMÉRICA

No decorrer dos cálculos na fase numérica, é extremamente importante a verificação dos valores numéricos que estão sendo obtidos. Deve ser possível decidir se determinado elemento calculado pode ser considerado zero ou não [03].

Esta decisão é de grande importância, pois este elemento pode ser escolhido como pivô em cálculos posteriores.

Consideremos a matriz  $A \in \mathbb{R}^{n \times m}$  e a base  $B \in \mathbb{R}^{n \times n}$ . Por exemplo, calculando:

$$\bar{a} = B^{-1} a$$

onde  $a \in A$  é um vetor coluna original de  $A$ ,  $\bar{a}$  o vetor atualizado e  $B^{-1}$ , a inversa da base.

Iniciando com  $a^{(0)} = a$ , calculamos:

$$a^{(k+1)} = \left( \prod_{i=1}^{k+1} U_i^{-1} \right) \left( \prod_{i=k+1}^1 L_i^{-1} \right) \quad k=1,2,\dots,n-1$$

e no final,  $a^{(n)} = \bar{a}$ .

O vetor coluna  $a$  tem  $n$  posições e será indicado por  $a(i)$ ,  $i=1,2,\dots,n$ . Para cada um dos vetores  $a(i)^{(k)}$ ,  $k=1,2,\dots,n$ , temos:

$$\mu(\bar{a}) = \max_{i,k} |a(i)^{(k)}|$$

Ao final do cálculo, um elemento  $\bar{a}(i)$  é considerado zero se:

$$|\bar{a}(i)| < \mu(\bar{a}) \cdot \tau$$

onde  $\tau = 10^{-13}$  é uma tolerância fixa.

### 3 - MINOS

MINOS é um pacote computacional utilizado para solucionar problemas de grande porte de programação linear e não-linear, sujeitos a restrições lineares [29].

O problema é expresso na seguinte forma:

$$\text{MIN } F(x) = f(x^N) + c^T x$$

$$\text{s.a.} \quad A x = b$$

$$l \leq x \leq u$$

onde  $A \in \mathbb{R}^{m \times n}$  ( $m \leq n$ ). O vetor  $x$  é particionado em variáveis lineares ( $x^L$ ) e não-lineares ( $x^N$ ):

$$x = \begin{bmatrix} x^N \\ x^L \end{bmatrix}$$

Se  $f(x^N) = 0$ , temos um problema de programação linear.

Nosso objetivo é estudar, apenas no caso de programação linear, as técnicas usadas no cálculo da inversa da base, bem como sua atualização.

Como já foi mencionado, existem duas versões do pacote MINOS: MINOS/AUGMENTED e MINOS 5.0. Veremos a seguir, para cada um deles, as técnicas de cálculo da inversa de uma base e como é realizada sua atualização.

### 3.1 - MINOS/AUGMENTED

#### (1) - Cálculo da inversa de uma base

Neste pacote, a inversa da base  $B \in \mathbb{R}^{n \times n}$  é calculada através da subrotina INVERT [28], [37].

INVERT chama, entre outras subrotinas, P4, que será vista com mais detalhes.

#### SUBROTINA P4

O objetivo desta subrotina é permutar linhas e colunas da base B, de modo a obter a estrutura bloco triangular inferior. Isto é conseguido através das seguintes subrotinas:

- (i) TRNSVL : para obter transversal máxima
- (ii) BUMPS : através do algoritmo de Tarjan, obtém-se permutações simétricas para conseguir a estrutura bloco triangular inferior para a base B.
- (iii) P3 : à cada bump externo conseguido após TRNSVL e BUMPS, é aplicada uma versão simplificada do algoritmo  $P^3$  de Hellerman e Rarick. Assim, linhas e colunas dentro de cada bump são permutadas, permitindo que a estrutura seja a mais próxima possível da triangular inferior, com exceção de poucas colunas (spikes). Dentro dos spikes, a quantidade de elementos não-nulos é a menor possível.

Ao final da subrotina P4, temos que  $B_{\Delta}$  é bloco triangular inferior:

$$P_1 B P_2 = B_{\Delta}$$

e em seguida, é calculada sua decomposição em fatores triangulares, através da subrotina FACTOR, também chamada por INVERT.

#### SUBROTINA FACTOR

Verificamos se a sequência obtida em P4 é aceita. Pode ser necessário trocar um elemento pivô num bump, para preservar a estabilidade numérica. As mudanças estão restritas às colunas de um mesmo bump, pois caso contrário, a estrutura conseguida em P4 pode ser totalmente perdida.

Por exemplo, num bump  $n \times n$ , se temos para o elemento pivô  $akk$ :

$$|akk| < TOL * \max_i |aik| \quad i=k, \dots, n$$

e  $TOL=0.001$ , este pivô é rejeitado.

Selecionamos entre os spikes deste bump, um novo elemento pivô. É considerada a linha  $k$  e entre todos os elementos não-nulos dos spikes que se encontram nesta linha, encontramos o que tem maior valor absoluto:

$$\alpha_{max} = \max_j |akj|$$

onde  $j$  é o índice de colunas spikes.

A nova coluna  $\phi$ , que trocará de posição com a coluna  $k$ , é escolhida de modo que :

$$|a_{k\phi}| > \text{TOLR} * \alpha_{\max}$$

onde  $\text{TOLR}=0.1$ .

O processo descrito equivale à eliminação Gaussiana com permutação de colunas.

Após INVERT, temos que:

$$B_{\Delta} = P_1 B P_2 = L U$$

## 2 - Esquema de atualização da inversa

A atualização da fatoração LU da base é realizada usando a subrotina MODLU.

### SUBROTINA MODLU

Temos que:

$$L^{-1} B_{\Delta} = U$$

Utilizando uma matriz de permutação  $P$ , os spikes são movidos para o final da matriz  $U$ . Através de  $P^T U P$ , a matriz passa a ter a seguinte estrutura:

$$U' = P^T U P = \left[ \begin{array}{c|c} I & R \\ \hline O & F \end{array} \right]$$

$$\text{Ou : } P^T L^{-1} B_{\Delta} P = U'$$

$$P^T L^{-1} P_1 B P_2 P = U'$$

Supondo que a coluna  $p$  da base é trocada por  $a_q \in A$  e considerando  $\alpha_p = L^{-1} P_1 a_q$ :

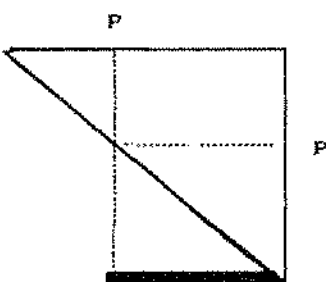
$$P^T L^{-1} \tilde{B} P = \tilde{U}$$

e  $\tilde{U}$  geralmente não é mais triangular superior.

Permutando o spike  $\alpha_p$  para o final de  $\tilde{U}$ , é obtida uma matriz Hessenberg superior:

$$P^T L^{-1} \tilde{B} P Q = \tilde{U} Q = H$$

Por meio da permutação simétrica:

$$Q^T U Q =$$


Através da eliminação Gaussiana com permutação de linhas, a estrutura triangular superior é restituída:

$$G Q^T \tilde{U} Q = G Q^T H = \bar{U}$$

e  $\bar{U}$  é triangular superior.

### 3.2 - MINOS 5.0

Nesta versão mais recente do MINOS/AUGMENTED [30], as subrotinas P4 (para fatoração LU da base) e MODLU (atualização da base) são substituídas pelo método descrito e implementado por Reid [34].

Dada a base  $B \in \mathbb{R}^{n \times n}$ . A estratégia de Markowitz é usada para escolha dos pivôs e para realizar fatoração LU da base. O esquema de atualização da base é o de Bartels e Golub.

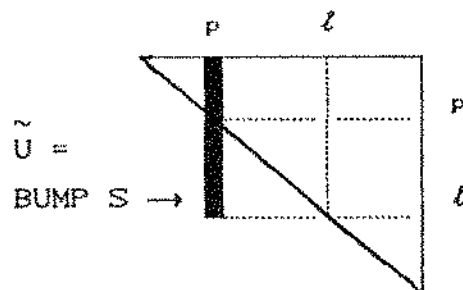
Considerando a base fatorada:

$$L B^{-1} = U$$

Com a alteração de uma das colunas da base original, digamos  $b_p$ , pela coluna  $a_q \in A$ , temos:

$$L \tilde{B}^{-1} = U$$

No caso das matrizes serem esparsas, é pouco provável que o spike em  $\tilde{U}$  atinja a última linha da matriz. Portanto, temos o seguinte bump:

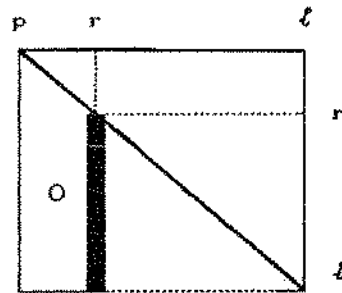


Se o spike fosse permutado para o final de  $\tilde{U}$ , teríamos uma matriz Hessenberg superior e seria aplicada a eliminação Gaussiana para restituir a estrutura triangular superior.

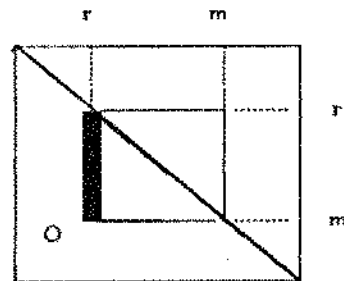
Porém, Reid descreveu um método muito semelhante ao de Markowitz, com o objetivo de diminuir os cálculos necessários para obter novamente a estrutura triangular superior da matriz  $\tilde{U}$ .



PASSO 1: busca-se encontrar no BUMP  $S \in \mathbb{R}^{(p-l) \times (p-l)}$ , colunas com um único elemento, ou seja, escolhemos como pivô o elemento  $a_{ij}^{(k)}$ , com custo Markowitz  $M_{ijk} = (J(i)^{(k)} - 1) * (I(j)^{(k)} - 1) = 0$ ,  $i, j = p \dots l$ . As colunas são examinadas uma a uma, a partir da esquerda.



PASSO 2: são procuradas linhas com um único elemento. Ao final deste passo, temos:



Em seguida, o spike é permutado para o final do bump reordenado, com o objetivo de obter uma matriz Hessenberg superior.

Os passos (1) e (2) são repetidos e ocorre uma das duas possibilidades:

- (i) não há mais bump ( a matriz é triangular superior ), ou
- (ii) o bump é uma matriz Hessenberg superior.

No caso (i), apenas com permutações de linhas e colunas no bump, a estrutura triangular superior para  $\tilde{U}$  foi obtida, sem necessidade de cálculos numéricos. Ou seja:

$$P \tilde{U} Q = \bar{U}$$

onde  $\bar{U}$  é triangular superior.

No caso (ii), a eliminação Gaussiana é aplicada, porém a quantidade de cálculos para obter novamente a estrutura triangular superior é menor que se fosse aplicada à matriz  $\tilde{U}$ , antes do método. Portanto, após o método descrito por Reid, temos que:

$$P \tilde{U} Q = H$$

e através da eliminação:

$$G P \tilde{U} Q = G H = \bar{U}$$

onde  $\bar{U}$  é triangular superior.

---

## APENDICE

---

---

### IMPLEMENTAÇÃO COMPUTACIONAL

---

Foi realizada uma adaptação da implementação computacional do método usando dois estágios, descrito em [11]. Dada uma matriz esparsa, o objetivo do método é encontrar uma sequência de pivôs, de modo que a matriz permutada tenha a estrutura bloco triangular, quando possível. O motivo do interesse especial pelos dois estágios é devido ao fato dos mesmos serem utilizados em subrotinas do pacote MINOS/AUGMENTED.

A adaptação do algoritmo para obter transversal máxima foi baseada em [08] e [09].

A escolha do algoritmo de Tarjan para permutar a matriz para a estrutura bloco triangular é devido a este ser um algoritmo mais eficiente que o de Sargent e Westerberg [11], [33]. Para a adaptação da implementação do algoritmo de Tarjan foram utilizados [07] e [15].

O método foi implementado no computador VAX11/785, com sistema operacional VMS V4.3, utilizando a linguagem FORTRAN 77.

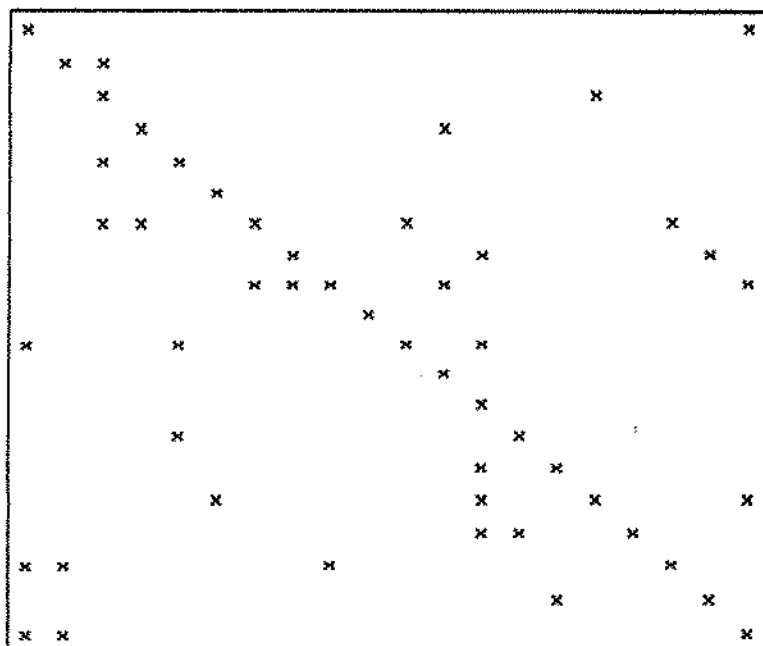
---

### TESTES

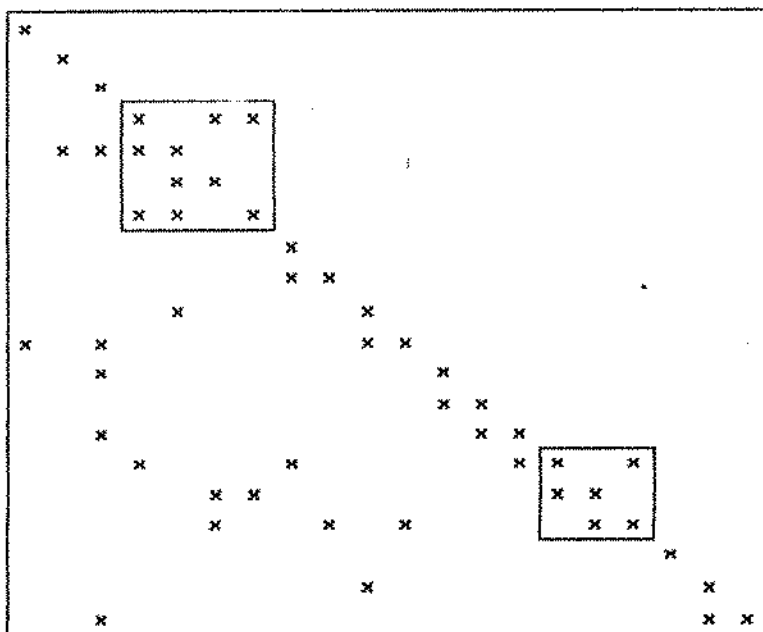
---

## TESTE 1

Matriz Original

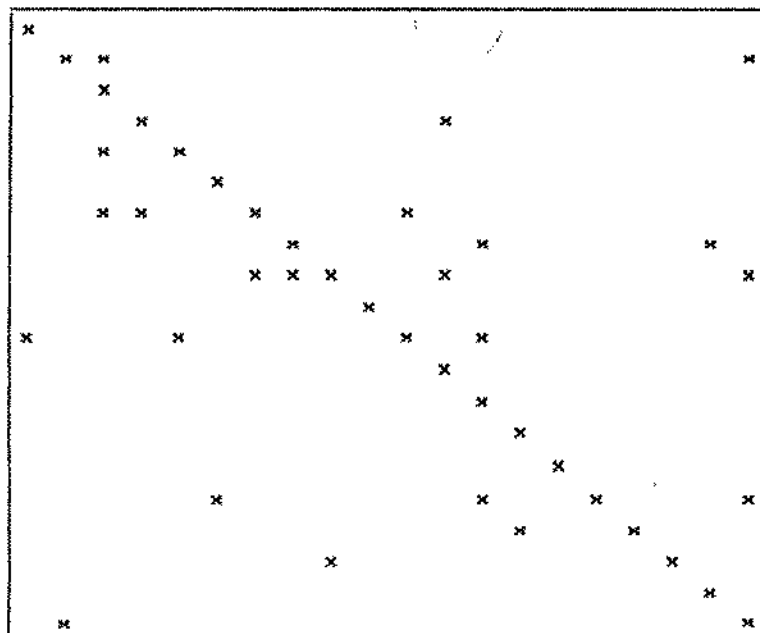


Estrutura final

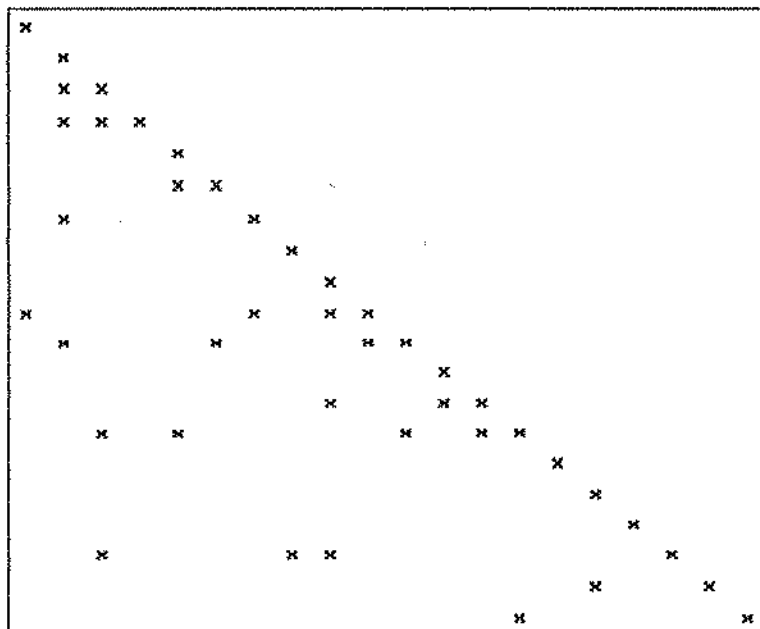


## TESTE 2

Matriz Original

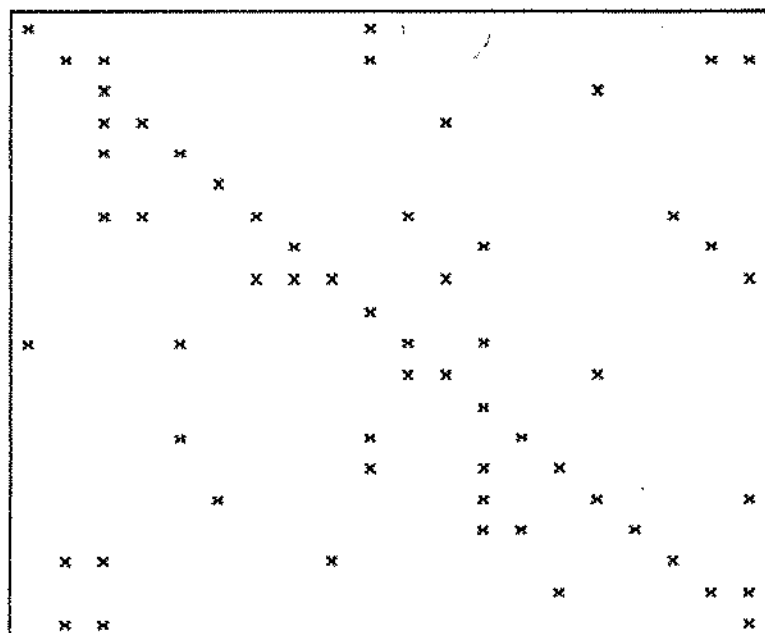


Estrutura Final

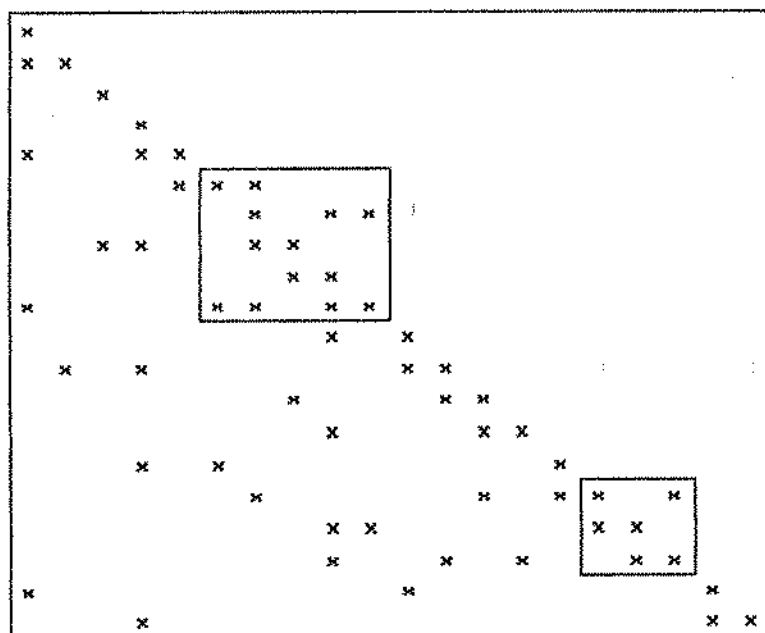


# TESTE 3

Matriz original

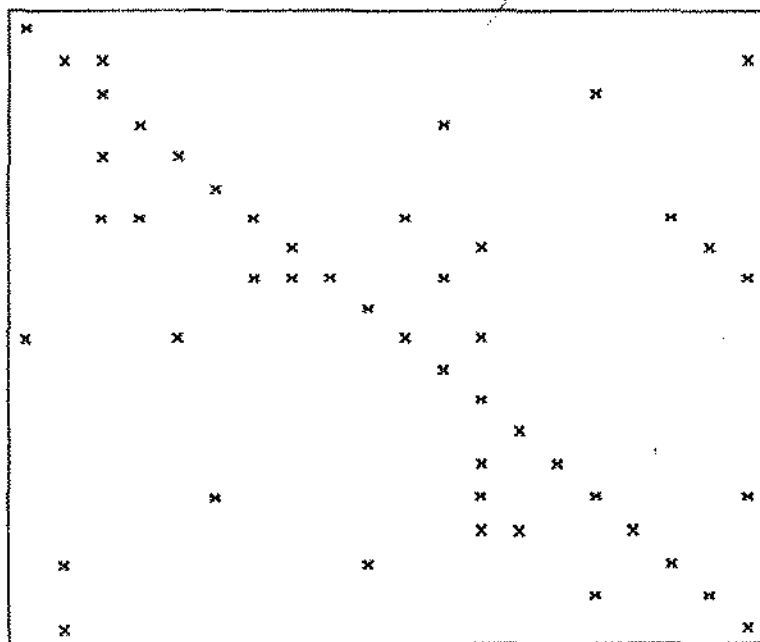


Estrutura Final

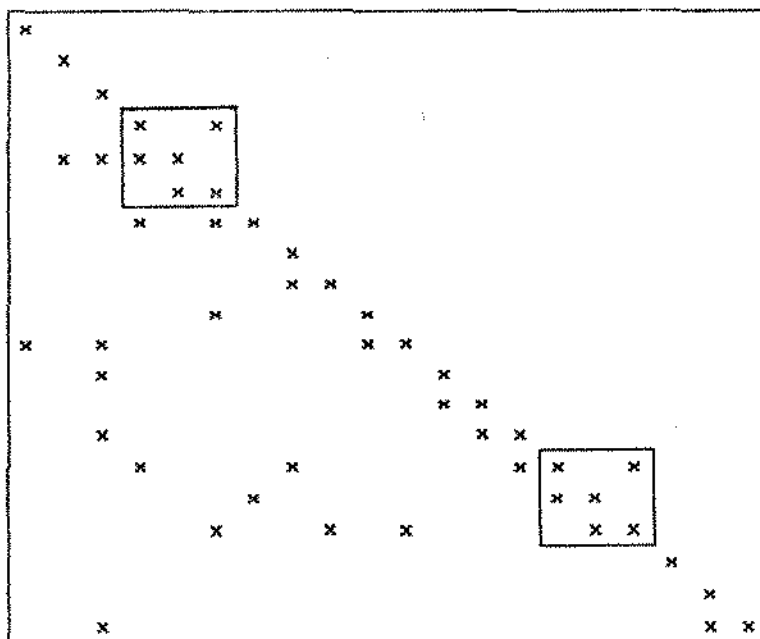


# TESTE 4

Matriz Original

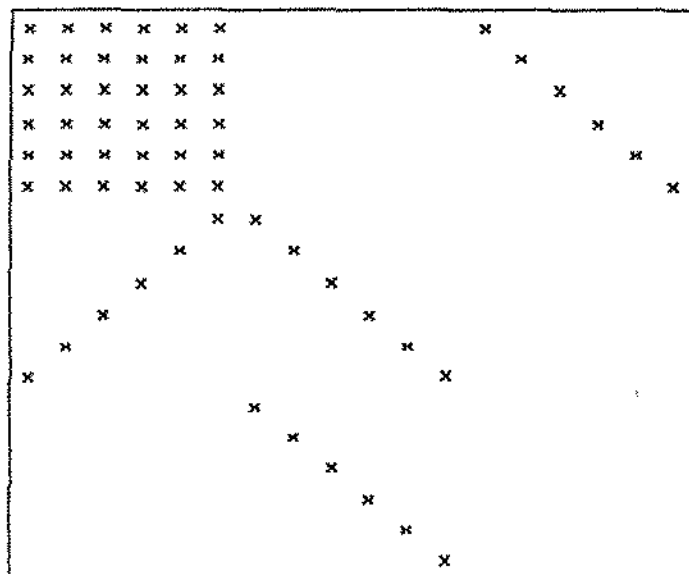


Estrutura Final

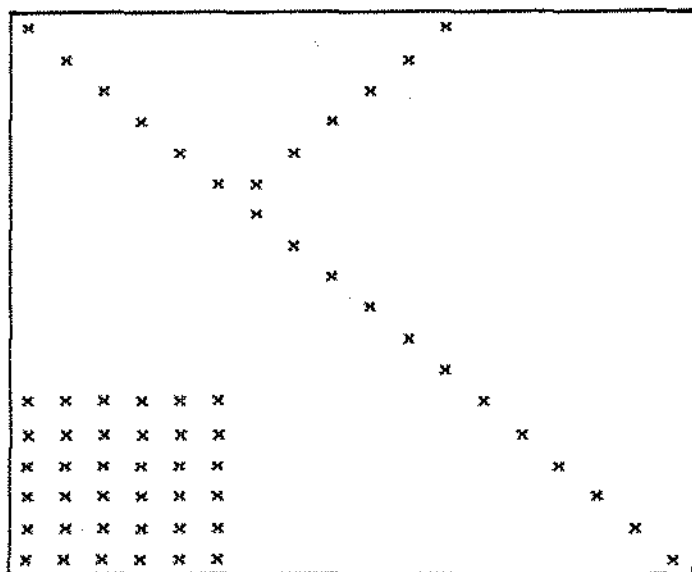


# TESTE 5

Matriz Original

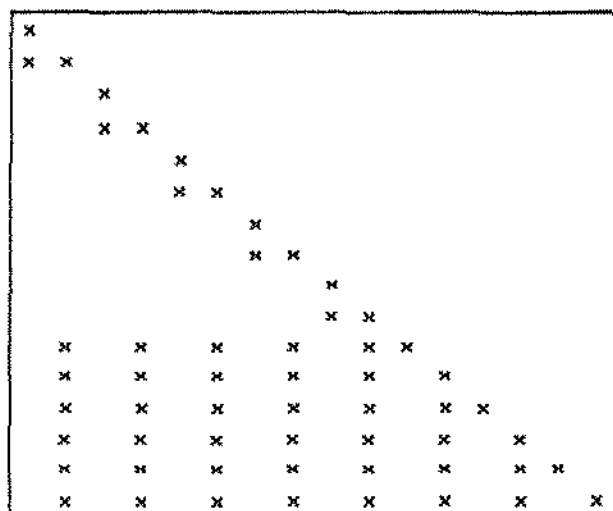


Matriz após permutação de linhas



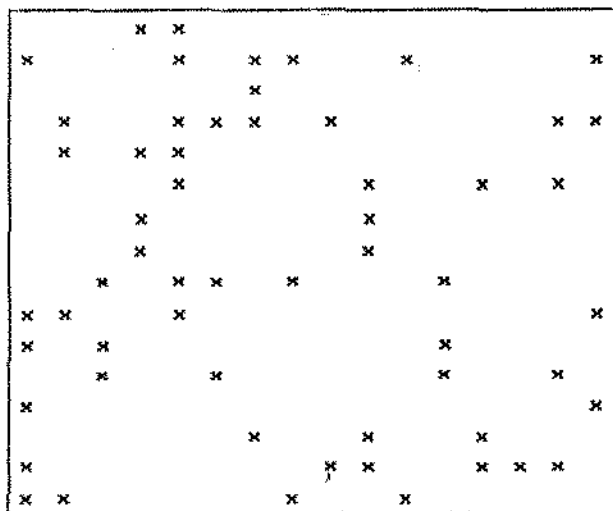


## Estrutura final

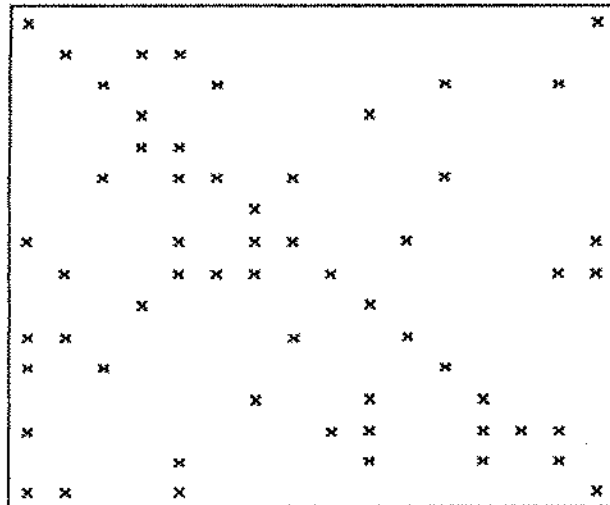


## TESTE 6

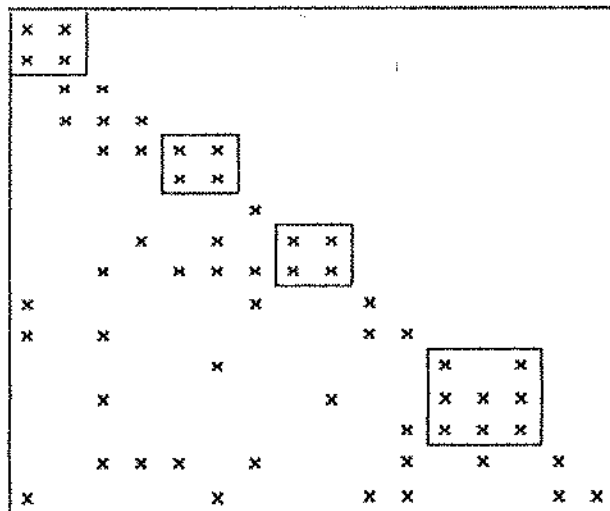
### Matriz original



Matriz após permutação de linhas



Estrutura final



## TESTE 7

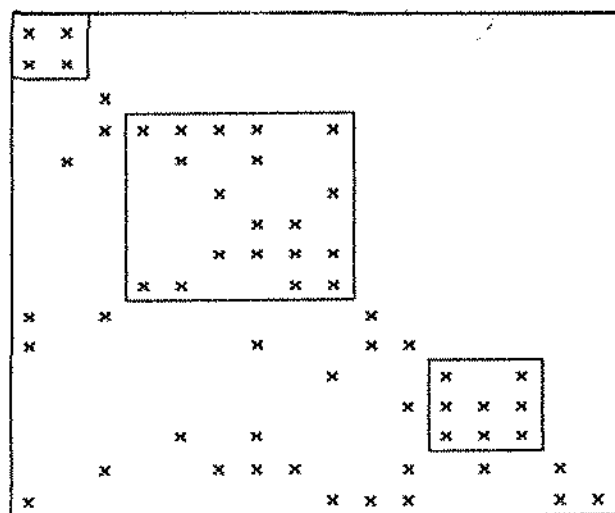
Matriz original (usada nos algoritmos  $P^3$  e  $P^4$ )

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
|   |   | x | x |   | x |   |   |   |   |
| x |   |   | x |   | x | x |   | x |   |
|   |   |   |   |   | x |   |   |   |   |
|   | x |   | x | x | x |   | x |   | x |
| x |   | x | x |   |   |   |   |   |   |
|   |   |   | x |   |   |   | x | x | x |
|   |   | x |   |   |   |   | x |   |   |
|   |   |   | x |   |   |   | x |   |   |
|   | x |   | x | x |   | x |   | x |   |
| x | x |   | x |   |   |   |   |   | x |
| x |   | x |   |   |   |   | x |   |   |
|   |   | x |   | x |   |   | x |   | x |
| x |   |   |   |   |   |   |   |   | x |
|   |   |   |   | x |   | x |   | x |   |
| x |   |   |   |   | x | x |   | x | x |
| x | x |   |   | x |   | x |   |   |   |

Matriz após permutação de linhas

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| x | x |   |   |   | x |   | x |   |   |
| x | x |   |   | x |   |   |   |   | x |
|   |   | x |   | x | x |   | x |   |   |
|   |   |   | x |   |   |   | x |   |   |
|   | x |   | x | x |   |   |   |   |   |
|   |   | x |   |   |   |   | x |   | x |
|   |   |   |   | x |   |   |   |   |   |
|   |   | x | x |   | x |   |   |   |   |
|   | x |   | x | x | x |   | x |   | x |
|   |   |   | x |   |   |   | x |   |   |
| x |   |   | x |   | x | x |   | x |   |
| x |   | x |   |   |   |   |   | x |   |
|   |   |   |   | x |   | x |   | x |   |
| x |   |   |   |   | x | x |   | x | x |
|   |   |   | x |   |   | x |   | x | x |
| x |   |   |   |   |   |   |   |   | x |

# Estrutura final



---

## BIBLIOGRAFIA

---

- [01] Bartels, R.H. e Golub, G.H. (1969). The simplex method of linear programming using LU decomposition. Comm ACM 12, 266-268.
- [02] Bazaraa, M.S. e Jarvis, J.J. (1977). Linear Programming and Network Flows. John Wiley and Sons, New York.
- [03] Benichou, M., Gauthier, J.M., Hentges, G. e Ribière, G. (1977). The efficient solution of large-scale programming problems - some techniques and computational results. Mathematical Programming 13, 280-322.
- [04] Bunch, J.R. e Rose, D.J. (eds.) (1976). Sparse Matrix Computations. Academic Press, New York and London.
- [05] Coleman, T. F. (1984). Large Sparse Numerical Optimization. Springer-Verlag, Berlin, Heidelberg, New York and Tokyo.
- [06] Daniel, J.W. e Noble, B. (1969). Applied Linear Algebra. Prentice-Hall, New Jersey.
- [07] Duff, I.S. (1977). On permutations to block triangular form. J.Inst. Maths. Applies 19, 339-342.
- [08] Duff, I.S. (1981a). On algorithms for obtaining a maximum transversal. ACM Trans. Softw. 7, 315-330.
- [09] Duff, I.S. (1981b). Algorithm 575. Permutations for a zero-free diagonal. ACM Trans. Softw. 7, 387-390.
- [10] Duff, I.S. (1981c). Direct methods for solving sparse systems of linear equations. SIAM J. Sci. Stat. Comput. 5, 605-619.

- [11] Duff, I.S., Erisman, A.M. & Reid, J.K. (eds.) (1986). Direct Methods for Sparse Matrices. Clarendon Press, Oxford.
- [12] Duff, I.S. & Reid, J.K. (1974). A comparison of sparsity orderings for obtaining a pivotal sequence in Gaussian elimination. J. Ins. Maths. Applics. 14, 281-291.
- [13] Duff, I.S. & Reid, J.K. (1976). A comparison of some methods for the solution of sparse overdetermined systems of linear equations. J. Ins. Maths. Applics. 17, 267-280.
- [14] Duff, I.S. & Reid, J.K. (1978a). An implementation of Tarjan's algorithm for a block triangularization of a matrix. ACM Trans. Math. Softw. 4, 137-147.
- [15] Duff, I.S. & Reid, J.K. (1978b). Algorithm 529. Permutations to block triangular form. ACM Trans. Math. Softw. 4, 189-192.
- [16] Duff, I.S. & Reid, J.K. (1979). Some design features of a sparse matrix code. ACM Trans. Math. Softw. 5, 18-35.
- [17] Erisman, A.M., Grimes, R.G., Lewis, J.G. & Poole, W.G. Jr. (1985). A structurally stable modification of Hellerman-Rarick's  $P^4$  algorithm for reordering unsymmetric sparse matrices. SIAM J. Num. Anal. 22, 369-385.
- [18] Forrest, J.J.H. & Tomlin, J.A. (1972). Updated triangular factors of the basis to maintain sparsity in the product form simplex method. Mathematical Programming 2, 263-278.
- [19] Forsythe, G. & Moler, C.B. (1967). Computer Solution of Linear Algebraic Equations. Prentice-Hall, New Jersey.

[20] Gill, P.E., Murray, W., Saunders, M.A. & Wright, M.H. (1984). Sparse matrix methods in optimization. SIAM J. Sci. Stat. Comput. 5, 562-589.

[21] Goldfarb, D. (1977). On the Bartels-Golub decomposition for linear programming bases. Mathematical Programming 13, 274-279.

[22] Gustavson, F.G. (1976a). Some basic techniques for solving sparse systems of linear equations (in Rose, D.J. & Willoughby, R.A. eds), 41-52.

[23] Gustavson, F.G. (1976b). Finding the block triangular form of a sparse matrix. (in Bunch, J.R. & Rose, D.J. eds.), 275-289.

[24] Hellerman, E. & Rarick, D.C. (1971). Reinversion with the preassigned pivot procedure. Mathematical Programming 1, 195-216.

[25] Hellerman, E. & Rarick, D.C. (1972). The partitioned preassigned pivot procedure (P\*) (in Rose, D.J. & Willoughby, R.A., eds), 67-76.

[26] Magnanti, T.L. (1976). Optimization for sparse systems. (in Bunch, J.R. & Rose, D.J. eds), 147-176.

[27] Markowitz, H.M. (1957). The elimination form of the inverse and its applications to linear programming. Management Science 3, 255-269.

[28] Murtagh, B.A. & Saunders, M.A. (1977). "MINOS user's guide", Report SOL 77-9, Department of Operations Research, Stanford University, CA.

[29] Murtagh, B.A. & Saunders, M.A. (1978). Large-scale linearly constrained optimization. Mathematical Programming 14, 41-72.

[30] Murtagh, B.A. & Saunders, M.A. (1983). "MINOS 5.0 user's guide", Report SOL83-20, Department of Operations Research, Stanford University, CA.

[31] Østerby, O. & Zlatev, Z. (1983). Direct Methods for Sparse Matrices. Lecture Notes in Computer Science 157, Springer-Verlag, Berlin, Heidelberg, New York and Tokyo.

[32] Perold, A.F. (1980). A degeneracy exploiting LU factorization for simplex method. Mathematical Programming 19, 239-254

[33] Reid, J.K. (1977). Solution of linear systems of equations: direct methods (general). "Sparse Matrix Techniques" (Barker, V.A. ed), Lecture Notes in Mathematics 572, Springer, Berlin, 102-129.

[34] Reid, J.K (1982). A sparsity-exploiting variant of the Bartels-Golub decomposition to linear bases. Mathematical Programming 24, 55-69.

[35] Rose, D.J. & Willoughby, R.A (eds.) (1972). Sparse Matrices and their applications. Plenum Press, New York.

[36] Saunders, M.A. (1976). A fast, stable implementation of the simplex method using Bartels-Golub updating. (in Bunch, J.R. & Rose, D.J. eds), 213-226.

[37] Saunders, M.A. (1977). "MINOS system manual", Report SOL 77-31, Department of Operations Research, Stanford University, CA.

[38] Stewart, G.W. (1973). Introduction to Matrix Computations. Academic Press, New York and London.



[39] Tomlin, J.A. (1972). Modifying triangular factors of the basis in the simplex method. (in Rose, D.J. & Willoughby, R.A., eds), 77-85.

[40] Tomlin, J.A. (1974). On pricing and backward transformation in linear programming. Mathematical Programming 8, 42-47.

[41] Wilkinson, J.H. (1961). Error analysis of direct methods of matrix inversion. J. ACM 8, 281-330.