Universidade Estadual de Campinas

Instituto de Matemática, Estatística e Computação Científica

(IMECC - Unicamp)

Análise Numérica - Projeto 1:

Envelopes de Matrizes Esparsas

Aluno: Daniel Yan || RA : 214793
Aluno: Matheus Araujo Souza || RA : 184145
Aluno: Daysa De Campos Da Silva || RA : 233469
Aluno: Fábio Henrique Polidoro Paiva || RA : 215570

Abril 2021

Introdução

O objetivo deste projeto é visualizar o comportamento de *matrizes esparsas*, que possuem uma grande quantidade de entradas **nulas**, de forma efetiva. Para isso, examinamos uma forma de guardar as matrizes na memória, de maneira que reduza o espaço utilizado pelo programa, chamada *envelopes*.

Utilizamos o programa **GNU Octave** e a sua linguagem para criar rotinas que resolvam um sistema linear de *matriz esparsa*, dada uma matriz $A \in \mathbb{R}^{n \times n}$, um vetor $b \in \mathbb{R}^n$, e a matriz de permutação do *pivoteamento parcial* P.

Item 1

Algoritmo GNU Octave: Resolução de um sistema linear por substituição regressiva (utilizando o envelope de uma matriz)

Entradas: vetores **DIAG** $\in \mathbb{R}^n$, **ENV** $\in \mathbb{R}^j$, **ENVcol** $\in \mathbb{R}^{n+1}$, **ENVlin** $\in \mathbb{R}^j$ e $\mathbf{b} \in \mathbb{R}^n$

```
 % Algoritmo de substitui  o regressiva (envelope colunas)
% matriz triangular superior (U)

function b = sol_SL_U(DIAG, ENV, ENVcol, ENVlin, b)
    n = size(DIAG, 2);

    for j = n:-1:1
        b(j) = b(j)/DIAG(j);
        for i = ENVcol(j+1)-1:-1:ENVcol(j)
            b(ENVlin(i)) = b(ENVlin(i)) - b(j)*ENV(i);
        end
    end
end
```

Saída: vetor $\mathbf{b} \in \mathbb{R}^n$

Item 2

Para facilitar na visualização, recriou-se a matriz, preenchendo com "0" os espaços vazios e "*" os espaços não-nulos, obtendo-se o seguinte:

$$A = \begin{bmatrix} * & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 & * & 0 & 0 \\ 0 & 0 & * & 0 & 0 & 0 & 0 \\ 0 & * & 0 & * & 0 & * & 0 \\ 0 & 0 & 0 & 0 & * & 0 & 0 \\ 0 & * & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & * \end{bmatrix}$$

Diagonal

O vetor diagonal é dado por:

DIAG(A) =

a_{11}	a_{22}	a_{33}	a_{44}	a_{55}	a_{66}	a_{77}

Envelope Superior

Os vetores Env_{Sup}, EnvCol_{Sup} e EnvLin_{Sup} são dados por:

ENV_{Sup}(A) =

a_{25}	a_{35}	a_{45}	a_{46}	a_{56}	a_{67}	

ENVcol_{Sup}(A) =

1	1	1	1	1	4	6	7

ENVlin_{Sup}(A) =

2	3	4	4	5	6

Envelope Inferior

As estruturas de dados do envelope inferior estão exibidas abaixo:

ENV_{Inf}(A) =

a_{42}	a_{43}	a_{62}	a_{63}	a_{64}	a_{65}	

ENVlin_{Inf}(A) =

1	1	1	1	3	3	7	7

ENVcol_{Inf}(A) =

2	3	2	3	4	5

Item 3

Tal fato[1] será demonstrado de maneira indutiva na dimensão n da matriz.

Caso Base: $n = 2$

Tomemos uma matriz genérica A tal que sua fatoração LU esteja definida. Assim, ela pode ser escrita como:

$$A = LU$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} \end{bmatrix}$$

Realizando a equivalência entre os termos das matrizes, obtemos que:

$$\begin{cases} a_{11} = u_{11} \\ a_{12} = u_{12} \\ a_{21} = l_{21}u_{11} \\ a_{22} = l_{21}u_{12} + u_{22} \end{cases}$$

1) Analisando a diagonal principal

Note que para a fatoração LU estar bem definida, os elementos da diagonal principal de U devem ser não nulos. Ou seja:

$$u_{11} = a_{11} \neq 0$$
$$u_{22} = a_{22} - l_{21}u_{12} \neq 0$$

2) Analisando a porção triangular superior

$$a_{12} = u_{12}$$

Assim, o comportamento da porção superior de U será igual ao comportamento da porção superior de A, ou seja:

i) Se $a_{12} \neq 0$

Neste caso, temos que $u_{12} = a_{12} \neq 0$, logo:

$$\text{ENV}_{sup}(A) = a_{12}$$
$$\text{ENV}_{sup}(U) = u_{12}$$

[1] O fato se refere ao Execício 1.7.38 de Watkins

ii) Se $a_{12} = 0$

Neste caso, temos que $u_{12} = a_{12} = 0$, logo:

$$\text{ENV}_{sup}(A) = \emptyset$$
$$\text{ENV}_{sup}(U) = \emptyset$$

Assim, juntando (i) e (ii), percebemos que a parte superior de U segue o padrão de zeros da parte superior de A, portanto:

$$ENV_{sup}(A) = ENV_{sup}(U)$$

3) Analisando a porção triangular inferior

$$a_{21} = l_{21}u_{11}$$

Como $u_{11} = a_{11} \neq 0$, então o comportamento de l_{21} depende exclusivamente de a_{21}, ou seja:

iii) Se $a_{21} \neq 0$

Neste caso, temos que $l_{21} = \dfrac{a_{21}}{u_{11}} \neq 0$, logo:

$$\text{ENV}_{inf}(A) = a_{21}$$
$$\text{ENV}_{inf}(L) = l_{21}$$

iv) Se $a_{21} = 0$

Neste caso, temos que $l_{21} = \dfrac{a_{21}}{u_{11}} = 0$, logo:

$$\text{ENV}_{sup}(A) = \emptyset$$
$$\text{ENV}_{sup}(L) = \emptyset$$

Assim, juntando (iii) e (iv), percebemos que a parte inferior de L segue o padrão de zeros da parte inferior de A, portanto:

$$ENV_{inf}(A) = ENV_{inf}(L)$$

4) Conclusão para o caso base

Através de (1), (2) e (3), conseguimos perceber que a comportamento dos elementos das matrizes L e U dependem diretamente das partes inferior e superior de A, respectivamente.

Hipótese Indutiva

Dada $M \in \mathbb{R}^{(n-1)x(n-1)}$ tal que a fatoração $M = LU$ esteja bem definida, onde $L \in \mathbb{R}^{(n-1)\times(n-1)}$ é triangular inferior e $U \in \mathbb{R}^{(n-1)\times(n-1)}$ é triangular superior. Neste caso, suponhamos que $Env(L) = Env(M)$ (por colunas), e que $Env(U) = Env(M)$ (por linhas).

Caso Geral

Supondo que a fatoração LU de $A \in \mathbb{R}^{n\times n}$ esteja definida, queremos provar que o $Env(L) = Env(A)$ (por linhas), e que $Env(U) = Env(A)$ (por colunas).

Assim, consideremos as matrizes A, L, U bordeadas da seguinte maneira:

$$A = LU$$

$$\left[\begin{array}{c|c} \hat{A} & c \\ \hline b^t & \alpha \end{array}\right] = \left[\begin{array}{c|c} \hat{L} & 0 \\ \hline s^t & 1 \end{array}\right] \left[\begin{array}{c|c} \hat{U} & v \\ \hline 0^t & \gamma \end{array}\right] = \left[\begin{array}{c|c} \hat{L}\hat{U} & \hat{L}v \\ \hline s^t\hat{U} & s^t v + \gamma \end{array}\right]$$

$$\begin{cases} \hat{A} \in \mathbb{R}^{(n-1)\times(n-1)} \\ \hat{L} \in \mathbb{R}^{(n-1)\times(n-1)}, \text{Triangular Inferior} \\ \hat{U} \in \mathbb{R}^{(n-1)\times(n-1)}, \text{Triangular Superior} \\ c, s, v \in \mathbb{R}^{n-1} \\ \alpha, \gamma \in \mathbb{R}^* \end{cases}$$

Note que $\hat{A} = \hat{L}\hat{U}$ satisfaz a nossa hipótese indutiva. Agora, precisamos saber se:

1) Padrão de b^t = Padrão de s^t

Considere que o primeiro elemento não nulo de s^t esteja na j-ésima posição, então queremos conhecer a x-ésima posição de b^t tal que:

i) $x < j$:

$$(b^t)_x = (s^t\hat{U})_x = \sum_{k=1}^{n-1}(s^t)_k(\hat{U})_{kx} =$$

$$= \sum_{k=1}^{j-1}(s^t)_k(\hat{U})_{kx} + \sum_{k=j}^{n-1}(s^t)_k(\hat{U})_{kx} = 0 + 0 = 0$$

Esse resultado segue de:

$$\begin{cases} (s^t)_k = 0, \text{ para } k < j \text{ e } (s^t)_j \neq 0, \text{ pois } (s^t)_j \text{ é o primeiro elemento não-nulo} \\ (\hat{U})_{kp} = 0 \text{ para } k > p, \text{ pois } \hat{U} \text{ é triangular superior} \end{cases}$$

ii) $x = j$:

$$(b^t)_x = (b^t)_j = (s^t\hat{U})_j = \sum_{k=1}^{n-1} (s^t)_k (\hat{U})_{kj} =$$
$$= (s^t)_j (\hat{U})_{jj} + \sum_{k=1}^{j-1} (s^t)_k (\hat{U})_{kj} + \sum_{k=j+1}^{n-1} (s^t)_k (\hat{U})_{kj} =$$
$$= (s^t)_j (\hat{U})_{jj} \neq 0$$

Esse resultado segue de:

$$\begin{cases} (s^t)_k = 0, \text{ para } k < j \text{ e } (s^t)_j \neq 0, \text{ pois } (s^t)_j \text{ é o primeiro elemento não-nulo} \\ (\hat{U})_{kj} = 0 \text{ para } k > j, \text{ pois } \hat{U} \text{ é triangular superior} \\ (\hat{U})_{jj} \neq 0, \text{ pois a diagonal principal de } \hat{U} \text{ é não-nula} \end{cases}$$

iii) Se $s^t = \vec{0}$:

Nesse caso, é trivial que $b^t = \vec{0}$, pois $b^t = s^t\hat{U} = \vec{0} \cdot \hat{U} = \vec{0}$.

iv) Conclusão:

Da hipótese indutiva, sabemos que o envelope por linhas da parte inferior de $\hat{A}$ é igual ao envelope por linhas de $\hat{L}$. Para verificar se o mesmo se aplica às matrizes A e L, precisamos saber se o padrão de elementos da última linha também são iguais.

Vimos através dos casos (i) e (ii) que, se $s^t \neq \vec{0}$, então a posição do primeiro elemento não nulo de s^t e b^t coincidem. Já no caso (iii) nota-se que, se $s^t = \vec{0}$, então $b^t = \vec{0}$ e o padrão das linhas são iguais e nenhuma das duas possui elemento não-nulo.

Assim, acabamos de provar que o padrão de b^t é o mesmo de s^t e, consequentemente, o envelope por linhas da parte inferior de A é igual ao envelope por linhas de L.

2) Padrão de c = Padrão de v

Considere que o primeiro elemento não nulo de v esteja na i-ésima posição, então queremos conhecer a x-ésima posição de c tal que:

i) $x < i$:

$$\begin{aligned} c_x = (\hat{L}v)_x = \sum_{k=1}^{n-1} (\hat{L})_{xk}(v)_k = \\ = \sum_{k=1}^{i-1} (\hat{L})_{xk}(v)_k + \sum_{k=i}^{n-1} (\hat{L})_{xk}(v)_k = 0 + 0 = 0 \end{aligned}$$

Esse resultado segue de:

$$\begin{cases} (v)_k = 0, \text{ para } k < i \text{ e } (v)_i \neq 0, \text{ pois } (v)_i \text{ é o primeiro elemento não-nulo} \\ (\hat{L})_{pk} = 0 \text{ para } k > p, \text{ pois } \hat{L} \text{ é triangular inferior} \end{cases}$$

ii) $x = i$:

$$\begin{aligned} c_x = c_i = (\hat{L}v)_i = \sum_{k=1}^{n-1} (\hat{L})_{ik}(v)_k = \\ = (\hat{L})_{ii}(v)_i + \sum_{k=1}^{i-1} (\hat{L})_{ik}(v)_k + \sum_{k=i+1}^{n-1} (\hat{L})_{ik}(v)_k = \\ = (\hat{L})_{ii}(v)_i = (v)_i \end{aligned}$$

Esse resultado segue de:

$$\begin{cases} (v)_k = 0, \text{ para } k < i \text{ e } (v)_i \neq 0, \text{ pois } (v)_i \text{ é o primeiro elemento não-nulo} \\ (\hat{L})_{pk} = 0 \text{ para } k > p, \text{ pois } \hat{L} \text{ é triangular inferior} \\ (\hat{L})_{ii} = 1, \text{ pois a diagonal principal de } \hat{L} \text{ é unitária} \end{cases}$$

iii) Se $v = \vec{0}$:

Nesse caso, é trivial que $c = \vec{0}$, pois $c = \hat{L}v = \hat{L} \cdot \vec{0} = \vec{0}$.

iv) Conclusão:

Da hipótese indutiva, sabemos que o envelope por colunas da parte superior de $\hat{A}$ é igual ao envelope por colunas de $\hat{U}$. Para verificar se o mesmo se aplica às matrizes A e U, precisamos saber se o padrão de elementos da última coluna também são iguais.

Vimos através dos casos (i) e (ii) que, se $v \neq \vec{0}$, então a posição do primeiro elemento não nulo de v e c coincidem. Já no caso (iii) nota-se que, se $v = \vec{0}$, então $c = \vec{0}$ e o padrão das linhas são iguais e nenhuma das duas possui elemento não-nulo.

Assim, acabamos de provar que o padrão de c é o mesmo de v e, consequentemente, o envelope por colunas da parte superior de A é igual ao envelope por colunas de U.

Item 4

Supondo que que existe uma matriz de permutação P tal que a decomposição LU de PA esteja bem definida.

Assim, podemos chamar $B = PA = LU$ e, ao aplicar o resultado do Exercício 3 à matriz B, obtemos que:

- Envelope por linhas da parte triangular inferior de $B = PA$ é igual ao envelope por linhas de L
- Envelope por linhas da parte triangular superior de $B = PA$ é igual ao envelope por colunas de U

Item 5

O algoritmo foi feito utilizando uma indução iterativa. Iniciamos a iteração igualando o envelope de L ao envelope inferior de A, e o envelope de U ao envelope superior de A.

Para cada iteração k, procuramos na k-ésima coluna de L e na k-ésima linha de U por termos que estão no envelope, em outras palavras, procuramos nos vetores $ENVcol_{inf}$ e $ENVlin_{sup}$ os elementos que são iguais a k. Para cada um dos elementos encontrados, usamos uma função para encontrar a linha, no caso dos elementos de L, ou a coluna, no caso dos elementos de U.

A função itera pelos elementos de um vetor ($ENVlin_{inf}$ para L e $ENVcol_{sup}$ para U) até chegar num valor maior que um certo x, que é o índice no envelope do elemento que queremos encontrar, e retornamos o índice anterior a esse x.

No caso dos elementos U_{kj}, consideramos que já calculamos os termos da somatória $\sum_{x=1}^{k-1} L_{kx}U_{xj}$ nas iterações anteriores, o termo $L_{kx}U_{xj}$, por exemplo, foi calculado na iteração x. Agora basta subtrair A_{kj} pela somatória, e sabendo que $L_{kk} = 1$ por definição, obtemos o valor de U_{kj}, além disso, sabemos também o valor de U_{kk}, calculando da mesma forma.

Já os elementos L_{ik}, eles precisam do valor de U_{kk}, que já calculamos anteriormente, então conseguimos calcular o seu valor facilmente também.

Por fim, precisamos calcular para as iterações seguintes. Para cada par de elementos $L_{ik}U_{kj}$, procuramos o termo L_{ij} ou U_{ij}, dependendo de qual índice é maior. No caso do L_{ij}, por exemplo, procuramos os elementos que estão na coluna j e no envelope, da mesma forma que fizemos anteriormente, e para cada elemento, vemos se a linha daquele elemento é igual a i.

Algoritmos GNU Octave: Sub função e algoritmo principal na qual foi empregada a construção dos fatores L e U, trabalhando com os envelopes das porções triangular inferior e superior de PA.

Entradas: vetores **DIAG** $\in \mathbb{R}^n$, **ENV Sup** $\in \mathbb{R}^j$, **ENV Inf** $\in \mathbb{R}^j$, **ENVcol sup** $\in \mathbb{R}^{n+1}$, **ENVcol inf** $\in \mathbb{R}^{n+1}$, **ENVlin sup** $\in \mathbb{R}^j$, **ENVlin inf** $\in \mathbb{R}^j$ e $\mathbf{b} \in \mathbb{R}^n$

Sub Função: Achar índices: Utilizada para encontrar os elemento que estão na linha k de U ou a coluna k de L.

```

function [top, indices] = acha_indices(value, array, n)
    indices = zeros(1,n);
    top = 1;
    for i = 1:length(array)
        if array(i) == value
            indices(top) = i;
            top=top+1;
        end
    end
end
```

Sub Função: Aonde está: Utilizada para encontrar a posição do ENv lin para a matriz L e Env col para a matriz U.

```
  function    pos = onde_esta(i,ini, ENVpos, n)
    pos = 0;
    for j = ini:n
        if (ENVpos(j)) <= i && ENVpos(j+1) > i
            pos = j;
            break
        end
    end
end
```

Função principal: Fatora Env Algoritmo para a fatoração L e U com envelopes.

```
%function [DIAG, ENV_inf, ENVlin_inf, ENVcol_inf, ENV_sup, ENVcol_sup, ENVlin_sup] = fatora_env_daniel(DIAG, ENV_inf, ENVlin_inf, ENVcol_inf, ENV_sup, ENVcol_sup, ENVlin_sup)
    n = max(size(DIAG));
    for k = 1:n-1
        % aux junta os indices dos elementos que estao no envelope na linha k de U ou coluna k de L

        [size_u, aux_u] = acha_indices(k, ENVlin_sup, n);
        [size_l, aux_l] = acha_indices(k, ENVcol_inf, n);
        for i = 1:size_l-1
            ENV_inf(aux_l(i)) = ENV_inf(aux_l(i))/DIAG(k);
            lin = onde_esta(aux_l(i), 1, ENVlin_inf, n);
            for j = 1:size_u-1
                col = onde_esta(aux_u(j), 1, ENVcol_sup, n);
                sub = ENV_inf(aux_l(i))*ENV_sup(aux_u(j));

                if lin > col
                    % procura o elemento no L.
                    [top, aux] = acha_indices(col, ENVcol_inf, n);
                    for x = 1:top-1
                        if (onde_esta(aux(x), 1, ENVlin_inf, n) == lin)
                            ENV_inf(aux(x)) = ENV_inf(aux(x)) - sub;
                        end
                    end
                elseif lin < col
                    % procura o elemento no U
                    [top, aux] = acha_indices(lin, ENVlin_sup, n);
                    for x = 1:top-1
                        if (onde_esta(aux(x), 1, ENVcol_sup, n) == col)
                            ENV_sup(aux(x)) = ENV_sup(aux(x)) - sub;
                        end
```

```
                    end
                else
                    DIAG(lin) = DIAG(lin)- sub;
                end

            end
        end
    end
%clear col, i, k, x, aux, sub;
%endfunction
```

Item 6

Escrevendo o sistema de forças da treliça no formato matricial de um sistema linear $Af = b$, temos que:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a & 0 & 0 & -1 & -a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a & 0 & 1 & 0 & a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a & 1 & 0 & 0 & -a & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a & 0 & 1 & 0 & a & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & a & 0 & 0 & -a & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & 0 & 1 & a & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -a & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \\ f_{11} \\ f_{12} \\ f_{13} \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 15 \\ 0 \\ 20 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Assim, como conhecemos a matriz de permutação P descrita pelo vetor k de índice das linhas

$$k = (3, 1, 2, 4, 5, 7, 6, 11, 8, 9, 10, 12, 13)$$

tal que a fatoração $PA = LU$ está bem definida, então podemos aplicar o algoritmo construído no *Exercício 5* para obter a os vetores da representação em envelope de L e U.

Pré-multiplicando o sistema linear original pela matriz P, obtemos que:

$$Af = b \implies P(Af) = Pb \implies (PA)f = Pb \implies (LU)f = Pb \implies L(Uf) = Pb$$

Ou seja, podemos calcular o vetor f das forças aplicadas na treliça através da resolução de dois sistemas lineares:

1.) $Lx = Pb$, onde L é triangular inferior;
2.) $Uf = x$, onde U é triangular superior

Note que, como conhecemos a representação em envelope de L e U, podemos aplicar o algoritmo desenvolvido no *Exercício 1* para obter a solução do segundo sistema linear, que tira proveito da representação de U por estrutura de envelope em colunas.

No entanto, para o primeiro sistema é necessário uma adaptação no algoritmo, de modo que possamos utilizar a representação de L por estrutura de envelope em linhas a nosso favor. A seguir, temos justamente essa adaptação.

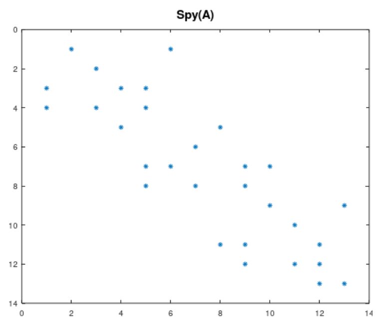
```
% Algoritmo de substituicao progressiva (envelope linhas)
% matriz triangular inferior (L)

function b = sol_SL_L(ENV, ENVlin, ENVcol, b)
    n = size(ENVlin, 2) -1;

    for i = 1:n
        for j = ENVlin(i):ENVlin(i+1)-1
            b(i) = b(i) - b(ENVcol(j)) *ENV(j);
        end
        %b(i) = b(i)/DIAG(i); essa linha nao e necessario pois a DIAG(i) = 1
    end
end
```

Assim, a realização do primeiro sistema linear ($Lx = Pb$) nos forneceu o seguinte vetor x abaixo. A partir dele, podemos partir para a realização do segundo sistema linear ($Uf = x$), o qual nos fornece, finalmente, o vetor f das forças aplicadas na treliça, dado abaixo.

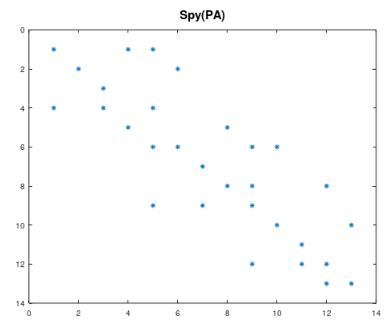
$$Lx = Pb \implies x = \begin{bmatrix} 0.00000 \\ 0.00000 \\ 10.00000 \\ -10.00000 \\ 10.00000 \\ 5.00000 \\ 0.00000 \\ 0.00000 \\ 20.00000 \\ 0.00000 \\ 20.00000 \\ -33.33333 \\ -25.00000 \end{bmatrix} \implies Uf = x \implies f = \begin{bmatrix} -28.28427 \\ -30.00000 \\ 10.00000 \\ -30.00000 \\ 14.14214 \\ -30.00000 \\ 0.00000 \\ -30.00000 \\ 7.07107 \\ -25.00000 \\ 20.00000 \\ -35.35534 \\ -25.00000 \end{bmatrix}$$

Por último, podemos visualizar como são as estruturas das matrizes L e U computadas através do algoritmo do *Item 5*.

Comparando as figuras (1) e (2) com as figuras (3) e (4) fica nítido a importância do resultado do *Item 4*. Isto pois a estrutura de envelope de A definitivamente não é parecida com as de L e de U, enquanto a da matriz PA é a mesma, nos ajudando a fixar a ideia de que a comparação entre os envelopes tem de ser feita no caso correto.

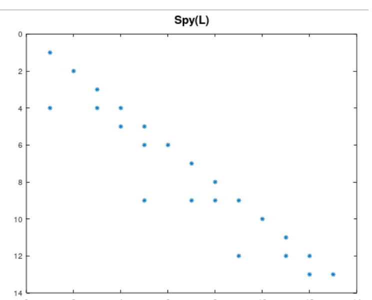


1) Padrão de Elementos da Matriz A

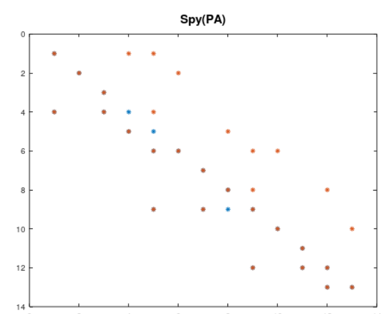


2) Padrão de Elementos da Matriz A permutada

Nas figuras (3) e (4), podemos observar que a única posição em L que foi preenchida é a L_{98}, ou seja, ela contém um elemento não-nulos em L e um elementos nulo em PA, e está indicada com um ponto azul na figura (4).

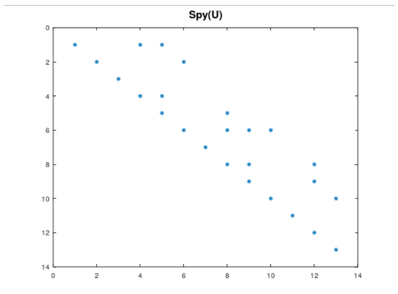


3) Padrão de Elementos da Matriz L

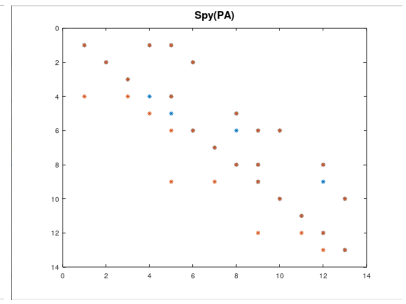


4) Elementos preenchidos na matriz L

Igualmente, nas figuras (5) e (6), podemos observar dois preenchimentos em U, nas posições $U_{6,8}$ e $U_{9,12}$.



5) Padrão de Elementos da Matriz U



6) Elementos preenchidos na matriz U

Conclusão

Ao fazer a média de 25 amostras de tempo de execução do nosso algoritmo do *Exercício 5* em uma máquina com 6gb de memória RAM, obteve-se que o tempo médio é dado por (0,06$\pm$0,02) s. Este método aparenta ser mais eficiente se bem implementado, uma vez que tende a realizar menos operações, principalmente quando tratando de matrizes esparsas com envelope s pequenos. No entanto, a visualização do *envelope* é mais complexo, dificultando a implementação dos algoritmos, assim como a prova demonstrada nos itens 3 e 4.

Referências

- **D. S. Watkins**, *Fundamentals of Matrix Computations*, New Jersey: John Wiley & Sons, 2 ed., 2002