Projeto I: Envelope de Matrizes Esparsas

Bryan Alves Do Prado Raimundo - RA 195171
Jean-Yves Roulet - RA 170315
Thiago Felipe Castro Carrenho - RA 224831
Vinicius Jameli Cabrera - RA 225414

22 de Abril de 2020

Universidade Estadual de Campinas
Instituto de Matemática, Estatística e Computação Científica
MS512A Análise Númerica I 1S2020
Prof.: Sandra Augusta Santos

Sumário

1 Introdução

Algumas matrizes possuem uma grande quantidade de zeros, estas são chamadas de matrizes esparsas, pois os valores não nulos estão esparsos, espalhados pela matriz, e ao tratar de problemas reais elas são bem comuns, ou melhor, a distribuição de suas aparições não é nada esparsa, ou seja, matrizes cheias de zeros não podem ser tratadas como zeros à esquerda.

Se essas matrizes aparecem com tanta frequência, e em dimensões altas, achar métodos de computação com essas matrizes que explorem essa alta quantidade de zeros, torna o seu cômputo mais veloz, sem precisar ficar, literalmente, somando zeros, e armazená-las se torna computacionalmente mais leves, não ocupando a memória da máquina com tantos elementos nulos.

Uma forma de armazenamento de matrizes esparsas é a noção de *envelope de matrizes*, que é um armazenamento geral, pois não depende de um tipo de matriz esparsa específica, como matriz de banda, por exemplo. Neste, uma matriz é detalhada em sete vetores, um para a diagonal três para parte triangular superior da matriz, e os outros três para a parte triangular inferior da matriz.

E é exatamente esse tipo de armazenamento que estudaremos nesse projeto, vendo como construí-lo e armazená-lo, como acessá-lo e usá-lo nos cálculos, resolvendo um problema real no final.

Sendo a matriz esparsa separada em triangular superior e inferior no envelopamento, é interessante analisar sua decomposição LU, que torna a resolução de sistemas lineares do tipo $Ax = b$ mais simples, separando em dois casos $Ly = b$ e $Ux = y$, que são bem mais simples de serem computados, pois U e L são triangulares, logo, buscamos formas de cálculos com o envelope e a decomposição LU, que tendem a tornar o cálculo de um sistema linear com uma matriz esparsa menos custoso para a máquina, tanto no seu armazenamento quanto no cálculo do sistema.

Para realizar cálculos computacionais eficientes com matrizes esparsas, são necessários algoritmos, que serão descritos como esquemas modulados em palavras ao longo do texto, para compreensão comum, e implementados na linguagem MATLAB (bem semelhante a OctaveGNU, por isso, referências de ambos foram utilizados) no apêndice.

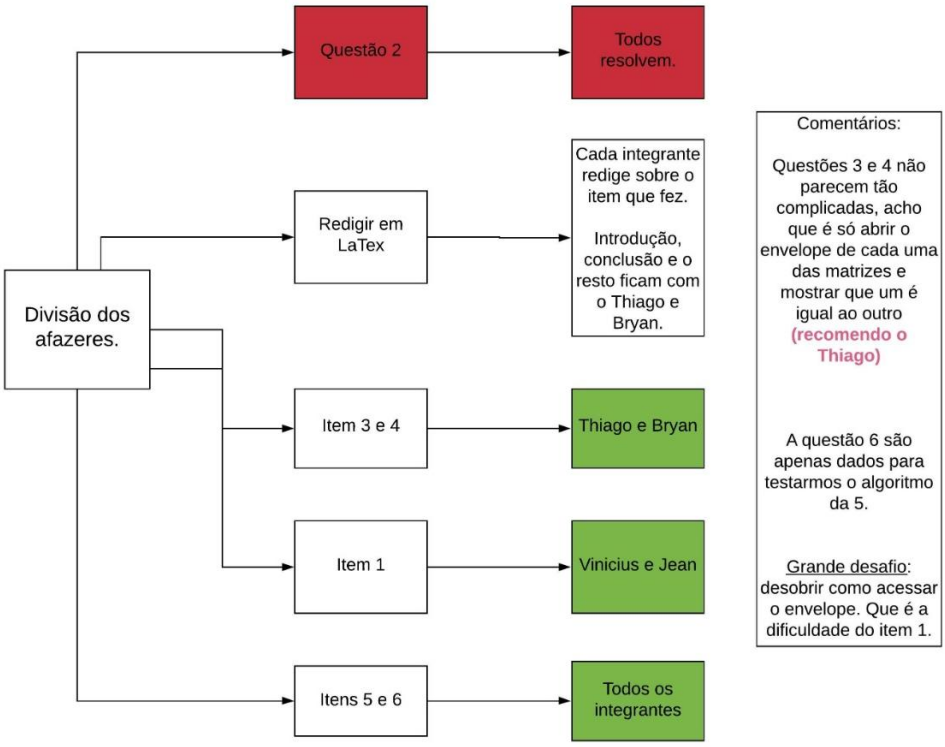
Este projeto é parte da avaliação da disciplina 'Análise Numérica I' (código MS512), da turma A do primeiro semestre de 2020, Universidade Estadual de Campinas. Portanto, utilizamos como referências do projeto alguns livros-texto desta disciplina, como Watkins [2010], e Moler [2004], também utilizados como embasamento para a criação dos itens formulados neste projeto.

2 Desenvolvimento

2.1 Planejamento inicial

Após todos do grupo estudarem o projeto e suas propostas, destacamos inicialmente as tarefas de cada um e os prazos, conforme o fluxograma abaixo.

O item 2, por conta de sua simplicidade, foi decidido que todos resolveriam. Assim, garantimos que todos entenderam o conceito de envelope, a parte de sua escrita e explicação ficou a encargo do Thiago. Bryan e Thiago fizeram juntos os itens 3 e 4, ficando a cargo do Thiago a explicação do 3, e o 4, que é sua continuação, foi descrito pelo Bryan. Vinicius e Jean ficariam com o item 1, que é a chave para conseguir resolvermos os exercícios 5 e 6, portanto, eles também se encarregariam de estar a frente desses dois últimos exercícios, que também seriam resolvidos conjuntamente por todos os integrantes do grupo.



E por último, estabelecemos os prazos, conforme mostrado na tabela 1 abaixo:

Data Inicial	Data Final	Tarefa
01/04	08/04	Resolução dos itens 1 a 4
09/04	15/04	Resolução dos itens 5 e 6
16/04	22/04	Redação do Projeto
22/04		Prazo Máximo de Entrega

Tabela 1: Prazos das Tarefas

2.2 Itens do projeto

2.2.1 Item 1

Neste parte do projeto objetivou-se o cômputo por substituição regressiva da solução do sistema linear do tipo $Ux = b$, com $U \in \mathbb{R}^{nxn}$ sendo uma matriz triangular superior e $b \in \mathbb{R}^n$, explorando o uso da estrutura de envelope por colunas da matriz U.

Para tal, foi desenvolvido o algoritmo apresentado abaixo, em palavras (1), com o correspondente implementado em MATLAB (comentado para facilitar a compreensão) no Apêndice 4.1. Este será analisado em detalhes na sequência.

Primeiramente se calcula $x(n)$, dado simplesmente pela fórmula $x(n) = \frac{b(n)}{U_{nn}} = \frac{b(n)}{DIAG(n)}$. Sabe-se que $ENVcol(j+1) - ENVcol(j)$ é o número de elementos da j-ésima coluna de U que pertencem ao envelope desta matriz. Utilizando esta propriedade, são percorridos linha a linha todos estes elementos em ENV associados a cada coluna j de U, através da indexação dada por ENVlin destes mesmos elementos, a fim de construir gradativamente o vetor s. Após se ter percorrido todas as colunas j de U, a partir da última coluna

Algoritmo 1: Cálculo de x em $Ux = b$

Dados: Matriz $U_{n\times n}$ triangular superior armazenada segundo a estrutura de envelope por colunas, nos vetores DIAG, ENV, ENVcol, ENVlin, e o vetor b

Resultado: vetor x tal que $Ux = b$

```
1  TE = Número de elementos de ENV;
2  x = zeros(n,1);
3  s = zeros(n,1);
4  TCaux = 0;
5  para j de n a 1 (passo -1) faça
6  |  x(j) = (b(j) − s(j))/DIAG(j);
7  |  TC = ENVcol(j + 1)-ENVcol(j);
8  |  k=0;
9  |  se TC ≠ 0 então
10 |  |  para i de ENVlin(TE-TC-TCaux) a ENVlin(TE-TCaux-1) faça
11 |  |  |  s(i)=s(i) + ENV(TE-TC-TCaux+k)*x(j);
12 |  |  |  k=k+1
13 |  |  fim
14 |  |  TCaux=TCaux+TC
15 |  fim
16 fim
```

$(j = n)$, o vetor s será dado por $s(i) = \sum_{j=i+1}^{n} a_{ij}x_j$, com $i = (n-1), ..., 1$. Observa-se que é somente preciso ter percorrido a coluna n até a coluna $(i + 1)$ para obter o elemento $s(i)$ por completo.

Graças à utilização da estrutura de envelope por colunas para armazenamento da matriz U evita-se percorrer qualquer elemento nulo a_{ij} que não faça parte do envelope de U.

A partir disso é possível calcular sequencialmente todos os $x(j)$ com $j = (n-1), ..., 1$ por substituição regressiva, conforme as colunas de U vão sendo percorridas, através da fórmula $x(j) = \frac{b(j)-s(j)}{U_{jj}} = \frac{b(j)-s(j)}{DIAG(j)}$, após as respectivas determinações de $s(j)$.

2.2.2 Item 2

A matriz A abaixo, é, conforme o enunciado desse item, $A \in \mathrm{R}^{n\times n}$, com $n = 7$, onde * representa os elementos não nulos da matriz, então, vamos substituir por a_{ij} da respectiva posição:

$$\mathbf{A} = \begin{bmatrix} * & & & & & & \\ & * & & * & & & \\ & & * & & & & \\ & * & & * & & * & \\ & & & * & & & \\ & * & & & & * & * \\ & & & & & & * \end{bmatrix} = \begin{bmatrix} a_{11} & & & & & & \\ & a_{22} & & & a_{25} & & \\ & & a_{33} & & & & \\ & a_{42} & & a_{44} & & a_{46} & \\ & & & & a_{55} & & \\ & a_{62} & & & & a_{66} & a_{67} \\ & & & & & & a_{77} \end{bmatrix}$$

O vetor $DIAG$ armazena todos os valores da diagonal da matriz A, logo:

$DIAG = (a_{11}, a_{22}, a_{33}, a_{44}, a_{55}, a_{66}, a_{77})$;

Construamos primeiro o envelope superior, isto é, os vetores ENV_{sup}, $ENVcol_{sup}$ e $ENVlin_{sup}$. Neste caso, para as colunas de 2 a $n = 7$, os elementos desde o primeiro elemento não nulo de cada coluna até o último elemento antes da diagonal principal entram em ENV_{sup}.

Para as colunas $j = 2$, $j = 3$ e $j = 4$, todos os elementos até a diagonal principal são nulos, logo nenhum destes elementos entra em ENV_{sup}. Para $j = 5$, o primeiro elemento não nulo é a_{25}, logo os elementos a_{25}, a_{35} e a_{45} entram em ENV_{sup}. Para $j = 6$, o primeiro elemento não nulo é a_{46}, e para $j = 7$ ele é a_{67}, logo, entram no vetor também os elementos a_{46}, a_{56} e a_{67}, considerando todas as colunas de A temos o seguinte vetor ENV_{sup}:

$ENV_{sup} = (a_{25}, a_{35}, a_{45}, a_{46}, a_{56}, a_{67}) = (a_{25}, 0, 0, a_{46}, 0, a_{67})$;

Por último, se adiciona um elemento 0, para este ser, realmente o último elemento do vetor (caso o vetor não seja um único elemento 0), logo:

$ENV_{sup} = (a_{25}, a_{35}, a_{45}, a_{46}, a_{56}, a_{67}, 0) = (a_{25}, 0, 0, a_{46}, 0, a_{67}, 0)$;

Percebe-se que mesmo alguns elementos nulos estão no envelope superior, a fim de contemplar a possibilidade de preenchimento dessas posições ao se efetuar operações de redução por linhas sobre a matriz.

A construção de $ENVcol_{sup}$ é constituída de, basicamente, dois passos: $ENVcol_{sup}(1) = 1$ e o número de elementos da coluna j no vetor ENV_{sup} é igual a $ENVcol_{sup}(j + 1) - ENVcol_{sup}(j)$, para $j = 2$ até $j = n$. Seja $ENVcol_{sup}$ um vetor de $n + 1$ elementos inicialmente nulos, então defina $ENVcol_{sup}(1) = 1$, e sabendo que não existem elementos não nulos acima da diagonal nas colunas $j = 1$, $j = 2$, $j = 3$ e $j = 4$, temos que:

$ENVcol_{sup}(5) - ENVcol_{sup}(4) = 0$

$ENVcol_{sup}(4) - ENVcol_{sup}(3) = 0$

$ENVcol_{sup}(3) - ENVcol_{sup}(2) = 0$

$ENVcol_{sup}(2) - ENVcol_{sup}(1) = 0$

$\therefore$, $ENVcol_{sup}(5) = ENVcol_{sup}(4) = ENVcol_{sup}(3) = ENVcol_{sup}(2) = ENVcol_{sup}(1) = 1$.

Existem 3 elementos da coluna $j = 5$, 2 da coluna $j = 6$ e 1 da coluna $j = 7$, ou seja:

$ENVcol_{sup}(6) - ENVcol_{sup}(5) = 3$;

$ENVcol_{sup}(7) - ENVcol_{sup}(6) = 2$;

$ENVcol_{sup}(8) - ENVcol_{sup}(7) = 1$;

Portanto, conhecendo o valor de $ENVcol_{sup}(5)$, podemos concluir que o vetor $ENVcol_{sup}$ é:

$ENVcol_{sup} = (1, 1, 1, 1, 1, 4, 6, 7)$;

A construção de $ENVlin_{sup}$ é, simplesmente, armazenar a linha de cada elemento de ENV_{sup}, isto é, $ENV_{sup}(x) = a_{ij} \Rightarrow ENVlin_{sup}(x) = i$, logo:

$ENV_{sup} = (a_{25}, a_{35}, a_{45}, a_{46}, a_{56}, a_{67}, 0) \Longrightarrow ENVlin_{sup} = (2, 3, 4, 4, 5, 6, 0)$.

Passemos agora a analisar a parte triangular inferior da matriz A, isto é, os vetores ENV_{inf}, $ENVcol_{inf}$ e $ENVlin_{inf}$, neste caso, as definições são análogas às realizadas acima, bastando passar de uma visualização por colunas à uma visualização por linhas. As correspondências podem ser representadas da seguinte maneira: $ENV_{inf} \Longrightarrow ENV_{sup}$, $ENVcol_{inf} \Longrightarrow ENVlin_{sup}$ e $ENVlin_{inf} \Longrightarrow ENVcol_{sup}$.

ENV_{inf} contém os elementos de cada linha i de A, desde o primeiro elemento não nulo da linha até o último elemento antes da diagonal principal. Os elementos não nulos da parte inferior são a_{42} e a_{62}, devendo ser inseridos neste vetor; os elementos seguintes a cada um destes elementos, nas respectivas linhas, até os últimos elementos antes da diagonal principal também devem ser inseridos, já que por definição devem pertencer ao envelope por linhas, são estes: a_{43}, a_{63}, a_{64} e a_{65}. Assim, construímos ENV_{inf}:

$ENV_{inf} = (a_{42}, a_{43}, a_{62}, a_{63}, a_{64}, a_{65}) = (a_{42}, 0, a_{62}, 0, 0, 0)$.

Por último, se adiciona um elemento 0, para este ser, realmente o último elemento do vetor (caso o vetor não seja um único elemento 0), logo:

$ENV_{inf} = (a_{42}, a_{43}, a_{62}, a_{63}, a_{64}, a_{65}, 0) = (a_{42}, 0, a_{62}, 0, 0, 0, 0)$.

Percebe-se novamente que alguns elementos nulos estão no envelope superior, também a fim de contemplar a possibilidade de preenchimento dessas posições ao se efetuar operações de redução por linhas sobre a matriz.

$ENVcol_{inf}$ é semelhante a $ENVlin_{sup}$, enquanto este armazena a linha de A à qual cada elemento de ENV_{sup} pertence, $ENVcol_{inf}$ armazena a coluna de A à qual cada elemento de ENV_{inf} pertence, ou seja:

$ENV_{inf}(x) = a_{ij} \Rightarrow ENVcol_{inf}(x) = j$, logo:

$ENV_{inf} = (a_{42}, a_{43}, a_{62}, a_{63}, a_{64}, a_{65}, 0) \Longrightarrow ENVcol_{inf} = (2, 3, 2, 3, 4, 5, 0)$.

Por último, $ENVlin_{inf}$ é construído de forma semelhante a $ENVcol_{sup}$, ambos com $n + 1$ posições inicialmente zero, então o primeiro elemento é 1, mas a diferença do $(j + 1)$-ésimo elemento para o j-ésimo elemento, com $j = 2$ até $j = n$, representa agora o número de elementos da j-ésima linha que pertencem a ENV_{inf}. Assim temos para a parte triangular inferior de A:

$ENVlin_{inf}(1) = 1$;

$ENVlin_{inf}(2) - ENVlin_{inf}(1) = 0$;

$ENVlin_{inf}(3) - ENVlin_{inf}(2) = 0$;

$ENVlin_{inf}(4) - ENVlin_{inf}(3) = 0$;

$ENVlin_{inf}(5) - ENVlin_{inf}(4) = 2$;

$ENVlin_{inf}(6) - ENVlin_{inf}(5) = 0$;

$ENVlin_{inf}(7) - ENVlin_{inf}(6) = 4$;

$ENVlin_{inf}(8) - ENVlin_{inf}(7) = 0$;

Resolvendo essas equações, construímos o vetor $ENVlin_{inf}$, e terminamos a construção do envelope da parte triangular inferior de A:

$ENVlin_{inf} = (1, 1, 1, 1, 3, 3, 7, 7)$.

Finalmente, unindo todos os vetores calculados acima, temos o seguinte envelope resultante para a matriz A:

$DIAG = (a_{11}, a_{22}, a_{33}, a_{44}, a_{55}, a_{66}, a_{77})$;

$ENV_{sup} = (a_{25}, a_{35}, a_{45}, a_{46}, a_{56}, a_{67}, 0) = (a_{25}, 0, 0, a_{46}, 0, a_{67}, 0)$;

$ENVcol_{sup} = (1, 1, 1, 1, 1, 4, 6, 7)$;

$ENVlin_{sup} = (2, 3, 4, 4, 5, 6, 0)$;

$ENV_{inf} = (a_{42}, a_{43}, a_{62}, a_{63}, a_{64}, a_{65}, 0) = (a_{42}, 0, a_{62}, 0, 0, 0, 0)$;

$ENVcol_{inf} = (2, 3, 2, 3, 4, 5, 0)$;

$ENVlin_{inf} = (1, 1, 1, 1, 3, 3, 7, 7)$.

Para facilitar os itens seguintes, elaborou-se o algoritmo de construção do envelope de uma matriz A. Este se encontra no Apêndice na sua versão em palavras, algoritmo 3, e na sua versão implementada em MATLAB, Subseção 4.2.

2.2.3 Item 3

O algoritmo 2 é um algoritmo de construção de LU a partir de A, em forma modulada em palavras, com o correspondente em MATLAB na Subseção 4.4 (Apêndice), adaptado de 4.4.1 de Justo u. a. [2019].

O resultado final deste algoritmo é dado pelas matrizes L e U, de forma que a diagonal de L seja composta de n elementos 1, L seja triangular inferior, U triangular superior e $A = L * U$.

Pelo laço 'para' na linha 4 vemos que o algoritmo computa sempre com $i > j$.

Precisamos verificar que $ENV_{inf}(A) = ENV_{inf}(L)$ e $ENV_{sup}(A) = ENV_{sup}(U)$. Pela definição, pertencem ao envelope superior de A os elementos desde o primeiro elemento não nulo de cada coluna até o

Algoritmo 2: Algoritmo de construção de LU a partir de A

Dados: $A_{n\times n}$
Resultado: Decomposição L e U de A

```
1  L = I_n;
2  U = A;
3  para j de 1 a n − 1 faça
4  |  para i de j + 1 a n faça
5  |  |  L(i, j) = U(i, j)/U(j, j);
6  |  |  para k de j + 1 a n faça
7  |  |  |  U(i, k) = U(i, k) − L(i, j) ∗ U(j, k);
8  |  |  fim
9  |  |  U(i, j) = 0;
10 |  fim
11 fim
```

último elemento dessa coluna anterior à diagonal principal, logo o envelope superior de A só é igual ao envelope superior de U se o primeiro elemento não nulo de cada coluna de ambas coincidirem. O mesmo segue analogamente para o envelope inferior, por linhas.

Pelo algoritmo apresentado acima, um elemento $U(a, b)$, com $a < b$ será alterado apenas na linha 8, com j de 1 a $a - 1$, todas as vezes em que $i = a$. Considerando que no começo $U = A$, no final $U(i, j)$ será:

$$U(a,b) = A(a,b) - \sum_{k=1}^{a-1}(L(a,k) * U(k,b)) \tag{1}$$

Agora, se $a = b$, ele será alterado, também apenas na linha 8, com j de 1 a $a - 1 = b - 1$, todas as vezes em que $i = a$, no final $U(i, j)$ será tanto da forma da fórmula acima quanto da fórmula abaixo.

Por último, se $a > b$, $U(a, b)$ será alterado na linha 8, com j de 1 a $b - 1$, todas as vezes em que $i = a$, e então, quando $i, j = a, b$, ele será zerado na linha 9. Porém, antes de ser zerado, ele estará com o seguinte valor:

$$U(a,b) = A(a,b) - \sum_{k=1}^{b-1}(L(a,k) * U(k,b)) \tag{2}$$

Primeiro, verifiquemos que $ENV_{sup}(A) = ENV_{sup}(U)$. Neste caso, trataremos de $a > b$, logo, usaremos (1).

Intuitivamente, como U é a matriz escalonada de A, ela deve manter o envelope, pois os elementos numa coluna em que acima dele há só elementos nulos, não se alterará, pois vai se somar e subtrair zeros a ela.

E é justamente isso que conseguimos a partir de (1), pois, se o elemento é da primeira linha, $a - 1 = 0$, logo a somatória não é lida e $U(1, j) = A(1, j)\ \forall j = 1, ..., n$. Agora, se for um elemento de linhas seguintes, e acima dele temos apenas elementos nulos (na mesma coluna), $U(k, b) = 0\ \forall k = 1, ..., a - 1$ e, portanto $U(a, b) = A(a, b)$, o primeiro elemento não nulo de cada coluna de U está na mesma posição do primeiro elemento não nulo de cada coluna de A, e ambos com mesmo valor.

$\therefore ENV_{sup}(A) = ENV_{sup}(U)$ ■

Agora, verifiquemos que $ENV_{inf}(A) = ENV_{inf}(L)$. Neste caso, trataremos de $a < b$, logo, $L(a, b)$ se calcula quando $U(a, b)$ é da forma (2). Nesta forma, ele é calculado na linha 7 do algoritmo, $L(a, b) = U(a, b)/U(b, b)$, na qual podemos transformar, com $U(b, b) \neq 0$, em:

$$L(a,b) = \frac{U(a,b)}{U(b,b)} = \frac{A(a,b) - \sum_{k=1}^{b-1}(L(a,k) * U(k,b))}{U(b,b)} \tag{3}$$

Na primeira coluna, $b-1=0$, logo o somatório não é lido, portanto, $L(i,1) = \frac{A(i,1)}{U(1,1)} = \frac{A(i,1)}{A(1,1)}$ $\forall i = 2,...,n$, com $A(1,1) \neq 0$. Logo $L(a,1) = 0 \Leftrightarrow A(a,1) = 0$, supondo $L(a,k) = 0$ $\forall k = 1,...,b-1 \Rightarrow L(a,b) = A(a,b)/U(b,b)$ por (3). Assim, por indução, o primeiro elemento não nulo de uma linha de L está na mesma posição, mas não necessariamente com mesmo valor, do primeiro elemento não nulo da mesma linha de A.

$\therefore ENV_{inf}(A) = ENV_{inf}(L)$ ■

Como exemplo disso, usemos a seguinte matriz A, sua decomposição em L e U, e então o cálculo de $ENV_{sup}(A)$, $ENV_{inf}(A)$, $ENV_{sup}(U)$ e $ENV_{inf}(L)$:

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

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & -\frac{1}{3} & \frac{1}{3} & 1 \end{bmatrix}; \mathbf{U} = \begin{bmatrix} 1 & 2 & 0 & 4 \\ 0 & 3 & 0 & 3 \\ 0 & 0 & 3 & 8 \\ 1 & 1 & 1 & -\frac{14}{3} \end{bmatrix}$$

$ENV_{sup}(A) = (a_{12}, a_{14}, a_{24}, a_{34})$ e $ENV_{inf}(A) = (a_{21}, a_{41}, a_{42}, a_{43})$;

$ENV_{sup}(U) = (u_{12}, u_{14}, u_{24}, u_{34})$ e $ENV_{inf}(L) = (l_{21}, l_{41}, l_{42}, l_{43})$;

Que, como queríamos mostrar, apesar dos elementos terem valores diferentes, o envelope é o mesmo, pois considera as mesmas posições. Podemos perceber, também, nesse exemplo, as igualdades mostradas na resolução do item, como $L(i,1) = A(i,1)/A(1,1)$ $\forall i = 2,...,n$, nesse caso com $n=4$ e $A(1,1)=1$, assim como o primeiro elemento não nulo de cada coluna de A ser igual ao primeiro elemento não nulo de cada coluna de U, ou seja $A(1,1) = U(1,1) = 1$, $A(1,2) = U(1,2) = 2$, $A(3,3) = U(3,3) = 3$ e $A(1,4) = U(1,4) = 4$.

2.2.4 Item 4

O processo de fatoração LU com uso de pivoteamento parcial é importante por inúmeros fatores, entre eles podemos citar: A resolução do problema da presença de zeros na diagonal (algo que faz com que matrizes de simples resolução tornem-se problemáticas), um menor custo computacional, um melhor aproveitamento da estrutura de matrizes esparsas e uma forma de se tentar aumentar a estabilidade numérica de uma matriz. A base do pivoteamento parcial consiste em utilizar de uma matriz P, que pode já ser dada ou obtida via algoritmo, que realiza a troca de linhas, permutação, como forma de escolher o maior elemento em módulo presente nos pivôs e utilizá-lo como base para o cálculo dos multiplicadores.

Considerando P conhecida, podemos denotar uma matriz $B = PA$, e dessa forma, aplicarmos os mesmos procedimentos do item 3. Ao realizarmos tais procedimentos, fica evidente que a constância da igualdade $ENV_{sup}(PA)$=$ENV_{sup}(U)$ e $ENV_{inf}(PA)$=$ENV_{inf}(L)$ continua, porém, como anteriormente já constatado, não com os mesmos elementos necessariamente. Tal fato já era esperado, visto que, a matriz P é uma matriz quadrada binária composta apenas de 0 e 1, o que faz com que a matriz PA não sofre nenhuma mudança estrutural em relação a suas dimensões. Para isto ficar mais visível, utilizaremos a mesma matriz A utilizada

no item 3:

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

Para tal matriz, encontramos $P = I$:

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

E tivemos como resultado para L e para U:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}; \mathbf{U} = \begin{bmatrix} 1 & 2 & 0 & 4 \\ 0 & 3 & 0 & 3 \\ 0 & 0 & 3 & 8 \\ 0 & -1 & 1 & -3 \end{bmatrix}$$

Com isso, fica visível que $ENV_{sup}(PA)$=$ENV_{sup}(U)$ e $ENV_{inf}(PA)$=$ENV_{inf}(L)$.

Importante ressaltarmos a possibilidade de encontrarmos a matriz P via algoritmo (Um algoritmo desse tipo encontra-se no apêndice) sendo assim, ela ser conhecida de antemão não é algo estritamente necessário.

2.2.5 Item 5

Tópicos que serão trabalhados neste item levando em consideração o que foi proposto no enunciado (hipóteses):

- $PAx = b$ com b conhecida
- PA=LU
- Computar LU com o algoritmo de eliminação gaussiana
- Utilizar envelopes por linhas e colunas, respectivamente, para o cálculo de L e U, utilizando se dessa estrutura para a obtenção do resultado de um dado vetor x

O objetivo principal deste item é a resolução do sistema $(PA)x = PB$, onde a matriz P é a matriz de permutação e B é conhecida de antemão, para tal, considerando o gasto computacional e a efetividade na obtenção da solução, dividimos PA em L e U, sendo a primeira uma matriz triangular inferior com diagonal um e a segunda uma matriz triangular superior, sendo que, tal divisão foi feita se utilizando da lógica estrutural dos envelopes.

Para um pleno entendimento e uma compreensão facilitada, dividimos o algoritmo em múltiplas funções, algo que facilita o acompanhamento passo a passo do código e nos permite visualizar com plenitude a lógica implantada no mesmo. O fluxo principal do código se encontra no arquivo "$PROGRAMA_PRINCIPAL$".

A notação utilizada para designar cada variável em cada função está explicitada em comentários nos códigos presentes no apêndice, no entanto,devemos ressaltar que tratamos PA e Pb simplesmente como A e b, visto que, P é uma matriz permutação conhecida de antemão.

Como do item 2 já tínhamos o algoritmo necessário para o envelopamento de PA, trabalhamos com os

envelopes obtidos de antemão, assim, para a obtenção de L e U via envelopes, nos utilizamos dos mesmos já citados e do vetor b. A lógica para a obtenção de L e U via envelopes foi desenvolvida através de pesquisa e entendimento pleno do funcionamento de envelopes, associado a sugestão do PED Giovanni. A base do algoritmo é a obtenção de L e U calculados iterativamente através das submatrizes da matriz A.

$$\left[\begin{array}{c|c} \overline{A} & b \\ \hline c^T & a_{nn} \end{array}\right] = \left[\begin{array}{c|c} \overline{L} & \overrightarrow{0} \\ \hline l^T & 1 \end{array}\right] \cdot \left[\begin{array}{c|c} \overline{U} & u \\ \hline \overrightarrow{0}^T & u_{nn} \end{array}\right]$$

Fazendo as operações necessárias e reorganizando de forma a facilitar a compreensão do problema:

$$\begin{cases} \overline{L}u = b \\ \overline{U}^T l = c \\ u_{nn} = a_{nn} - l^T u \end{cases}$$

Com isso, progredimos o algoritmo aumentando em um a dimensão de A a cada iteração:

$$A = \left[\begin{array}{cc|c} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \hline a_{31} & a_{32} & a_{33} \end{array}\right]$$

Para que possamos construir L e U a cada iteração, obtemos o envelope de A^k através do envelope de A.

Portanto, utilizamos as iterações de forma regressiva para que se calcule u, l e $u_{nn}eu$, ou seja, os novos elementos a serem adicionados em L e U. Importante ressaltarmos que as atualizações de L e U são armazenadas no próprio envelope de A, isso otimiza os cálculos, evitando a criação de vetores auxiliares.

No sistema linear acima, um detalhe importante é que lidamos com U^T, mesmo que U seja envelopado por colunas, quando U^T, apenas trocamos o papel de $ENVcol$ e $ENVlin$ e utilizamos os algoritmos de triangular inferior.

Analisando as funções utilizadas ao longo do Atualizamos constantemente L e U a cada iteração dentro do próprio envelope de A. Nos utilizamos do algoritmo desenvolvido no item 1 para a resolução de um sistema triangular superior, além de, termos desenvolvido um algoritmo semelhante que resolve sistemas triangulares inferiores; ambos os algoritmos resolvem sistemas com matrizes envelopadas através de substituição regressiva.

Ao final de todo esse processo, temos L e U devidamente calculadas e envelopadas, com isso, resolvemos os últimos dois sistemas através do algoritmo de resolução de sistemas triangulares inferiores, e com isso, computamos o vetor x que é solução do sistema.

2.2.6 Item 6

Considerando o problema proposto, é necessário implementarmos o algoritmo desenvolvido no item 5 para a resolução do sistema $(PA)f = B$, sendo P uma matriz de permutação definida pelo vetor de índices (2,3,1,4,5,7,6,9,10,11,8,12,13). Com isso, podemos definir A, B, PB e PA como:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha & 0 & 0 & -1 & -\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha & 1 & 0 & 0 & -\alpha & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 & -\alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 1 & \alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 1 \end{pmatrix}; B = \begin{pmatrix} 0 \\ 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 15 \\ 0 \\ 20 \\ 0 \\ 0 \\ 0 \end{pmatrix};$$

$$PA = \begin{pmatrix} \alpha & 0 & 0 & -1 & -\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha & 1 & 0 & 0 & -\alpha & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 & -\alpha & 0 \\ 0 & 0 & 0 & 0 & \alpha & 0 & 1 & 0 & \alpha & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 1 & \alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & 1 \end{pmatrix}.; PB = \begin{pmatrix} 0 \\ 0 \\ 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 15 \\ 0 \\ 20 \\ 0 \\ 0 \end{pmatrix};$$

Como forma de explicitar ainda mais a esparsidade de PA, é interessante ocultarmos os termos nulos na matriz, obtendo assim:

$$PA = \begin{pmatrix} \alpha & & & -1 & -\alpha & & & & & & & & \\ & 1 & & & & -1 & & & & & & & \\ & & 1 & & & & & & & & & & \\ \alpha & & 1 & & \alpha & & & & & & & & \\ & & & 1 & & & & -1 & & & & & \\ & & & & \alpha & 1 & & & -\alpha & -1 & & & \\ & & & & & & 1 & & & & & & \\ & & & & & & & 1 & \alpha & & & -\alpha & \\ & & & & \alpha & & 1 & & \alpha & 1 & & & -1 \\ & & & & & & & & & 1 & & & -1 \\ & & & & & & & & & & 1 & & \\ & & & & & & & & \alpha & & 1 & \alpha & \\ & & & & & & & & & & & -\alpha & 1 \end{pmatrix}.$$

Com isso, ao aplicarmos o algoritmo obtido no item 5, temos como resultado o vetor x que resolve o sistema linear, no problema original, esse vetor é f, que por questão de estética, escrevemos a transposta f^T:

$$f^T = \begin{pmatrix} -28,28 & -30 & 10 & -30 & 14,14 & -30 & 0 & -30 & 7,07 & -25 & 20 & -35,35 & -25 \end{pmatrix};$$

Sendo assim, resolvemos o problema proposto, onde a i-ésima componente de f, isto é, $f(i)$, representa a força resultante aplicada na barra i da treliça, de forma que o equilíbrio de forças seja obtido, sob as restrições do problema: nó 1 rígido (não se move nem na horizontal nem na vertical), nó 8 se movendo só na vertical e ângulo $\alpha = 1/\sqrt{2}$.

3 Conclusão

Desenvolvido os itens acima, em especial com a resposta obtida no item 6 e com o algoritmo construído no item 5, concluímos que o envelopamento de matrizes esparsas realmente torna a resolução de sistemas lineares menos custoso para a máquina, sendo necessárias menos operações. Concluímos também que a decomposição LU auxilia ainda mais, pois o envelope explora a divisão da matriz em uma superior e uma inferior, e ao juntarmos a mesma com o pivoteamento parcial, temos uma maior eficiência na computação dos problemas, evitando erros simples, como a presença de zeros em posição de pivô, e a manutenção da estabilidade numérica de uma dada matriz.

Logo, dado uma matriz esparsa sem formato definido, ao invés de simplesmente calcular um sistema linear na maneira comum, é preferível (isto é, menos custoso para a máquina), decompor a matriz em LU e trabalhar com os envelopes.

Envelopar a matriz não foi difícil de se fazer, e a decomposição LU já era conhecida, mas o grupo encontrou dificuldades para unir as duas coisas, isto é, dado os envelopes orientados por linhas e por colunas de uma matriz esparsa, achar os envelopes por linha de L e por coluna de U, no entanto, após um longo período de pesquisa e desenvolvimento, conseguimos compreender a lógica por trás disso, e dessa forma, construímos o código proposto no item 5

Outra questão que nos trouxe certa dificuldade no projeto foi a escrita em LaTeX, em especial passar os algoritmos (tanto em formato de palavras quanto em MATLAB) e a parte de referência bibliográfica.

Há de se ressaltar o desenvolvimento em múltiplas áreas que tivemos dado a diversidade encontrada no projeto, apesar do foco proposto inicialmente para a resolução dos itens por cada integrante do grupo, foco esse que foi respeitado, tivemos também o auxílio e uma participação geral do grupo em cada item proposto, mesmo que de forma breve, o que nos propiciou o trabalho por uma vertente mais teórica/escrita e por uma vertente de programação/computacional.

Temos de indicar também que durante nossas pesquisas, encontramos inúmeros problemas reais e teóricos que se utilizam e tratam dessa vertente de estudo, incluindo o problema do item 6, isso nos ajuda ainda mais a compreender a importância de tais áreas do conhecimento e o quanto o seu desenvolvimento é benéfico.

Com isso, concluímos que tal projeto auxiliou muito em nosso aprendizado sobre matrizes esparsas e envelopes, nos fazendo compreender o funcionamento e aplicabilidade de ambos, além de ter nos propiciado uma interessante forma de aprimorar habilidades gerais, como a programação em MATLAB/Octave e o desenvolvimento de provas matemáticas.

4 Apêndice

4.1 Algoritmo em MATLAB do item 1

Algoritmo 1 implementado em MATLAB, com comentários para facilitar a compreensão.

```
%Entradas(matriz 'U' armazenada segundo a estrutura de envelope por colunas e 'b')
prompt1 = 'Insira o vetor DIAG = '; %vetor (1xn)
prompt2 = 'Insira o vetor ENV = '; %ultimo elemento eh zero (primeira posicao livre)
prompt3 = 'Insira o vetor ENVcol = '; %vetor (1x(n+1))
prompt4 = 'Insira o vetor ENVlin = '; %mesma dimensao de ENV, ultimo elemento eh zero (livre)
prompt5 = 'Insira o vetor b = ';
DIAG =input(prompt1);
ENV = input(prompt2);
ENVcol =input(prompt3);
ENVlin =input(prompt4);
b = input(prompt5);

%Definicoes
[¬,n] = size(DIAG);
[¬,TE] = size(ENV);

%Declaracoes
x = zeros(n,1); %vetor de saida
s = zeros(n,1); %vetor auxiliar ('s(n)' eh zero a fim de realizar o calculo de 'x(n)')
TC_aux = 0; %variavel auxiliar

for j = n:-1:1

    x(j)= (b(j)-s(j))/DIAG(j); %calculo do j-esimo elemento de 'x'
    TC = ENVcol(j+1)-ENVcol(j); %numero de elementos da coluna 'j'que pertencem ao envelope
    k=0; %acumulador para percorrer os elementos de ENV cada coluna de 'U'

    if TC ≠ 0 %se TC == 0 nao ha elemento nao-nulo na coluna 'j' de 'U'

        for i = ENVlin(TE - TC - TC_aux):ENVlin(TE - TC_aux -1)

            s(i) = s(i) + ENV(TE - TC - TC_aux + k)*x(j); %atualizacao de 's' por linha
            k = k +1;

        end

        TC_aux = TC_aux + TC; %contabiliza as colunas ja percorridas

    end

end

x %exibe o resultado obtido
```

4.2 Algoritmo em MATLAB do item 2

Algoritmo 3 implementado em MATLAB.

```
function [DIAG, ENVS, ENVcolS, ENVlinS, ENVI, ENVcolI, ENVlinI]=envelope(A)
    n=size(A,1);
    DIAG=zeros(1,n); ENVcolS=zeros(1,n+1); ENVlinI=zeros(1,n+1);
    ENVS=zeros(1,1); ENVI=zeros(1,1);
    ENVlinS=zeros(1,1); ENVcolI=zeros(1,1);
    cs=1; ci=1;
    ks=0; ki=0;
    ENVcolS(1)=1; ENVcolS(2)=1;
    ENVlinI(1)=1; ENVlinI(2)=1;
    for j=1:n
        DIAG(j)=A(j,j);
        ss=0;
        si=0;
        for i=1:(j-1)
            if(A(i,j)~=0)
                ss=1;
            end
            if(ss==1)
                cs=cs+1;
                ks=ks+1;
                ENVS(ks)=A(i,j);
                ENVlinS(ks)=i;
            end
            ENVcolS(j+1)=cs;
            if(A(j,i)~=0)
                si=1;
            end
            if(si==1)
                ci=ci+1;
                ki=ki+1;
                ENVI(ki)=A(j,i);
                ENVcolI(ki)=i;
            end
            ENVlinI(j+1)=ci;
        end
    end
    ENVS = [ENVS 0];
    ENVlinS = [ENVlinS 0];

    ENVI = [ENVI 0];
    ENVcolI = [ENVcolI 0];

    if (size(ENVS,2) == 2)
        if (ENVS == [0 0])
            ENVS = [0];
            ENVlinS = [0];
        end
    end

    if (size(ENVI,2) == 2)
```

```
51          if (ENVI == [0 0])
52               ENVI = [0];
53              ENVcolI = [0];
54          end
55      end
56
57  end
```

4.3 Algoritmo em palavras do item 2

Algoritmo modulado em palavras, para facilitar o entendimento dos passos feitos no item 2, e auxiliar na resolução dos itens seguintes.

Algoritmo 3: Cálculo do envelope orientado por colunas da porção triangular superior de A e do envelope orientado por linhas da porção triangular inferior de A

Dados: Matriz $A_{n \times n}$
Resultado: Vetores $DIAG$, ENV_{sup}, $ENVcol_{sup}$, $ENVlin_{sup}$, ENV_{inf}, $ENVcol_{inf}$ e $ENVlin_{inf}$

```
cs = 1;
ci = 1;
ks = 0;
ki = 0;
ENVcol_sup(1) = 1;
ENVcol_sup(2) = 1;
ENVlin_inf(1) = 1;
ENVlin_inf(2) = 1;
para j de 1 a n faça
    DIAG(j) = a_jj;
    ss = 0;
    si = 0;
    para i de 1 a j − 1 faça
        se a_ij ≠ 0 então
            ss = 1
        fim
        se ss = 1 então
            cs = cs + 1;
            ks = ks + 1;
            ENV_sup(ks) = a_ij;
            ENVlin_sup(ks) = i
        fim
        ENVcol_sup(j + 1) = cs;
        se a_ji ≠ 0 então
            si = 1
        fim
        se si = 1 então
            ci = ci + 1;
            ki = ki + 1;
            ENV_inf(ki) = a_ji;
            ENVcol_inf(ki) = i
        fim
        ENVlin_inf(j + 1) = ci
    fim
fim
se ENV_sup ≠ [0] então
    Adicione elemento 0 aos vetores ENV_sup e ENVlin_sup
fim
se ENV_inf ≠ [0] então
    Adicione elemento 0 aos vetores ENV_inf e ENVcol_inf
fim
```

4.4 Algoritmo em MATLAB do item 3

Algoritmo em MATLAB da função que calcula LU de dada matriz $A \in \mathbb{R}^{n \times n}$ relacionado a 2.

```
function [L,U]=fatoraLU(A)
    n=size(A,1);
    L=eye(n,n);
    U=A;
    for j=1:n-1
        for i=j+1:n
            L(i,j) = U(i,j)/U(j,j);
            U(i,j+1:n) = U(i,j+1:n) - L(i,j)*U(j,j+1:n);
            U(i,j)=0;
        end
    end
end
```

4.5 Algoritmo em MATLAB do item 4

Tal algoritmo encontra a melhor matriz P e executa o pivoteamento parcial.

```
function[L, U, P]=pivotLU(A)
    [m, n] = size(A);
    L=eye(n);
    P=eye(n);
    U=A;
    for z=1:m-1
        pivot=max(abs(U(z:m,z)))
        for j=z:m
            if(abs(U(j,z))==pivot)
                ind=j
                break;
            end
        end
        U([z,ind],z:m)=U([ind,z],z:m)
        L([z,ind],1:z-1)=L([ind,z],1:z-1)
        P([z,ind],:)=P([ind,z],:)
        for j=z+1:m
            L(j,z)=U(j,z)/U(z,z)
            U(j,z:m)=U(j,z:m)-L(j,z)*U(z,z:m)
        end
        pause;
    end
end
```

Uma versão em palavras pode ser encontrada em Ruggiero, tal algoritmo foi a base para a construção do código em MATLAB.

4.6 Algoritmos em MATLAB do item 5

4.6.1 Função para a resolução de uma triangular inferior

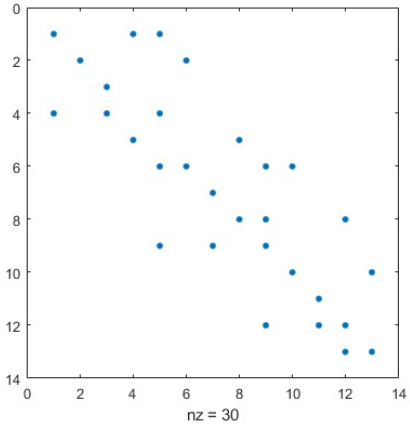
```
function solutionVector = resolverTriangularInferior(DIAG, ENV_inf, ENVlin_inf, ...
    ENVcol_inf, b)

    s = 0;
    TL_aux = 0;
    [¬,n] = size(DIAG);
    x = zeros(n,1);
    x(1)= (b(1)-s)/DIAG(1);

    for i = 2:n

        s=0;
        TL = ENVlin_inf(i+1)-ENVlin_inf(i);
        k=0;

        if TL ≠ 0

            for j = ENVcol_inf(TL_aux + 1):ENVcol_inf(TL_aux +TL)

                s = s + ENV_inf(TL_aux +1 + k)*x(j);
                k = k +1;

            end

            TL_aux = TL_aux + TL;

        end

        x(i)= (b(i)-s)/DIAG(i);

    end

    solutionVector = x;

end
```

4.6.2 Função para a resolução de uma triangular superior

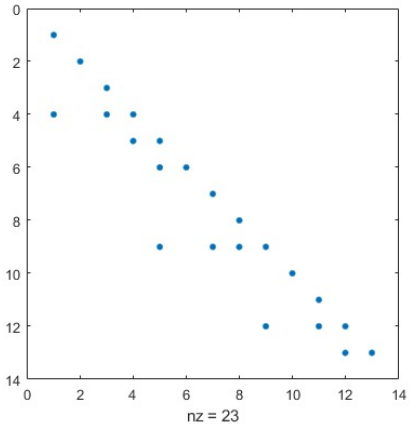
```
function solutionVector =  resolverTriangularSuperior(DIAG, ENV_sup, ENVcol_sup, ...
    ENVlin_sup, b)
    %DIAG ten dimens o (1xn)
    %ENV_sup tem dimens o TE -> ltimo elemento  a primeira posi o vazia, precisa ...
        existir e ser preechida com zero -> S  PODE EXISTIR UMA POSI O LIVRE (A PRIMEIRA)
    %ENVcol_sup tem dimens o (1x(n+1)) -> primeiro elemento  1 e ltimo elemento  o ...
         ndice da posi o livre em ENV_sup
    %ENVlin_sup tem mesma dimens o de ENV_sup, O LTIMO ELEMENTO  ZERO E REPRESENTA A ...
        POSI O LIVRE DE ENV -> PRECISA EXISTIR

    TC_aux = 0;
    [¬,n] = size(DIAG);
    [¬,TE] = size(ENV_sup);

    s = zeros(n,1);
    x = zeros(n,1);

    for j = n:-1:1

        x(j)= (b(j)-s(j))/DIAG(j);

        TC = ENVcol_sup(j+1)-ENVcol_sup(j);

        k=0;

        if TC ≠ 0

            for i = ENVlin_sup(TE - TC - TC_aux):ENVlin_sup(TE - TC_aux -1)

                s(i) = s(i) + ENV_sup(TE - TC - TC_aux + k)*x(j);
                k = k +1;

            end

            TC_aux = TC_aux + TC;

        end
    end

    solutionVector = x;

end
```

4.7 Spys obtidos durante a resolução do item 6

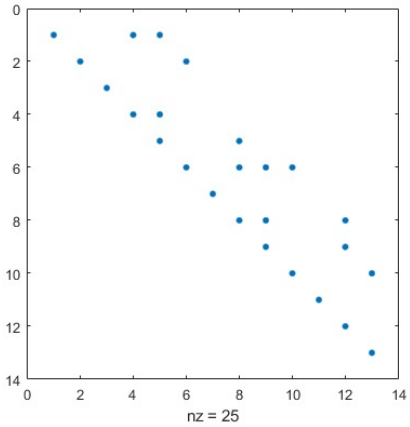
PA:



L:



U:



Referências Bibliográficas

[Justo u. a. 2019] Justo, Dagoberto Adriano R. ; Sauter, Esequia ; Azevedo, Fabio S. de ; Guidi, Leonardo F. ; Almeida Konzen, Pedro H. de: *Cálculo Numérico - Um Livro Colaborativo - Versão Octave*. 2019. – URL https://www.ufrgs.br/reamat/CalculoNumerico/livro-oct/main.html

[Moler 2004] Moler, Cleve B.: *Numerical Computing with MATLAB, Philadelphia*. Philadelphia : SIAM, 2004

[Ruggiero] Ruggiero, Marcia Aparecida G.: *Algoritmo (Resolução do sistema linear $Ax = b$ através da fatoração LU com pivoteamento parcial)*. – URL https://www.ime.unicamp.br/~marcia/AlgebraLinear/Arquivos%20PDF/algo_lu.pdf

[Watkins 2010] Watkins, David S.: *Fundamentals of Matrix Computations*. 3. ed. New Jersey : John Wiley & Sons, 2010. – ISBN 978-0-470-52833-4