Algoritmos Numéricos DI/CT/UFES

Roteiro para a Aula de Laboratório sobre Resolução de Sistemas Lineares Triangulares e Eliminação progressiva (Triangularização)

1. Sistemas Triangulares e o Algoritmo de Substituições Regressivas.

   Foi visto que para se resolver um sistema linear triangular do tipo:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \cdots + a_{1,n}x_n = b_1 \\ \quad a_{2,2}x_2 + a_{2,3}x_3 + \cdots + a_{2,n}x_n = b_2 \\ \quad\quad \cdots \quad \cdots \quad \vdots \quad \cdots = \cdots \\ \quad\quad a_{i,i}x_i + \cdots + a_{i,n}x_n = b_i \\ \quad\quad\quad \vdots \quad \cdots = \cdots \\ \quad\quad\quad a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1} \\ \quad\quad\quad\quad a_{n,n}x_n = b_n \end{cases}$$

   ou seja, um sistema chamado de triangular superior, pode se empregar algoritmo de substituições regressivas. O algoritmo começa calculando o x_n e, em seguida, obtem o x_{n-1} e assim por diante. A variável x_i é obtida via:

$$x_i = (b_i - \sum_{j=i+1}^{n} (a_{i,j}x_j))/a_{i,i}$$

   Abaixo, está o pseucódigo deste algortimo.

```
ALGORITMO de Substituições Regressivas
INICIO
Ler(A,b,n)
x=b(n)/A(n,n)
Para i de (n-1) até 1, passo(-1)
  s=0
  Para j de (i+1) até n
      s = s + A(i,j)*x(j)
  Fim do j
  x(i) = ( b(i) - s )/A(i,i)
Fim do i
Mostrar(x)
FIM
```

   Por exemplo, usando este algoritmo, é possível obter a solução do seguinte sistema:

$$\begin{cases} 3.0x_1 + 12.5x_2 - 6.2x_3 = 2.4 \\ \quad 20x_2 - 0.1x_3 = 0.0 \\ \quad\quad 5.0x_3 = 35 \end{cases}$$

```
Neste caso, a solução esperada é:   x= 15.120833    0.035000    7.000000
```

   Implementar este algoritmo, isto é, implementar o algoritmo de substituições regressivas. Dica: Se quiser, pode ser usar como ponto de partida, uma versão já iniciada (incompleta) que está disponível em: resolveSistemaTriangularINCOMPLETA.m.

   Depois de implementado, use o seu código (o seu script) para achar a solução do problema acima e mais algum outro, por exemplo do sistema $Ax = b$, com A e b dados por:

```
A =  1     2     3
     0     4     5
     0     0     6

b =  7
    -8
     9
```

Para isso, coloque a matriz A e vetor b do problema na mémoria (digitando a matriz e o vetor b na linha de comando) e resolva o problema triangular $Ax = b$ ativando o código, ou seja, na janela de comandos do octave, seria digitar os seguintes comandos:

```
>>A = [1 2 3; 0 4 5 ; 0 0 6]
>>b = [7; -8;  9]
>>resolveTriangularSuperiorSuaVersao
```

A chamada fica assim se você salvou seu código com o nome resolveTriangularSuperiorSuaVersao.m. Fazer com calma, entendendo o que está implementado e olhando as saídas.

2. Uma outra versão, em um formato distinto, está já implementada e está disponivel no arquivo: resolveTriangularSuperior.m. Esta versão foi implementada no formato de rotina computacional (no octave, chamada de function).
Observar que se trata de um **script especial** do tipo function que tem **DADOS de entrada e DADOS de saída**. Neste tipo de código, no octave, pode haver um ou mais argumentos de entrada e um ou mais argumentos de saída. Observar, também, que a primeira linha do script (do arquivo), deve ter a seguinte sintaxe:

```
function [<lista variaveis de saida>] =  <nomefunçao>(<lista argumentos de entrada>]);
Exemplo:
Uma função para fazer a soma e produto de três valores, poderia ser:
function [soma, prod] = fazSomaProdutodeTresvalores(a, b, c)
   soma    = a+b+c
   produto = a*b*c
endfunction

Sua chamada, na linha de comandos, para os valores 1 2 3, seria:
>> [s, p]=fazSomaProdutodeTresvalores(1,2,3);
```

Para chamá-la (rodar a sequencia de comandos) basta digitar o nome do arquivo na limha de comandos - que deve ser igual ao nome da função- e é preciso colocar os argumentos de entrada entre parênteses- na ordem em que aparecem na function. As variáveis que vão receber os valores de saída são opcionais mas é desejável colocá-las (o(s) nome(s) é escolha do usuário, ao chamar a funcão na linha de comando).

No código resolveTriangularSuperior.m fornecido (que resolve um sistema linear triangular Superior) os dados de entrada são a matriz A e o vetor b, só há um valor de saída, que é o vetor x (a solução). Estes dados devem ter dimensões compatíveis. Observar que na versão fornecida, a dimensão é resgatada dos dados da matriz A via o comando: [n,n]=size(A).

Usando esta implementação, obter a solução do sistema linear dado abaixo, com os dados :

```
A =  1    2   3
     0    4   5
     0    0   6

b =  7
    -8
     9

>>A = [1  2 3; 0 4 5 ; 0 0 6]
>>b = [7; -8;  9]
>>xsol = resolveTriangularSuperior(A,b)

A solução esperada é: xsol =  10.2500   -3.8750    1.5000
```

3. Implementar o algoritmo para resolver um sistema linear triangular inferior. Os dados de entrada devem ser a matriz A e o vetor b, só há um valor de saída, que é o vetor x (a solução). Usando sua esta implementação, obter a solução do sistema linear, com A e b dados abaixo:

```
A =  1    0   0
     2    3   0
     4    5   6

b =  7
    -8
     9

>>A = [1 0 0; 2 3 0 ; 4 5 6]
>>b = [7; -8;  9]
>>xsol = resolveTriangularinferior(A,b)

A solução esperada é: xsol =  7.0000  -7.3333   2.9444
```

4. Implementar o algoritmo de Triangularização de uma matriz (também chamada da fase de Eliminação progressiva da Eliminação de Gauss) sem a estratégia de pivoteamento. Dica: ver o pseudo-código nos slides.

5. Usando o método de Eliminação de Gauss SEM pivoteamento (Triangularização SEM pivoteamento + Substituições Regressivas), resolva os sistemas lineares $Ax = b$, com as matrizes A e vetores b, dados abaixo:

```
exemplo (1)
A1=[10 -2 1 ; 5 2 5 ; -1 -1 0];
b1=[0; 4; 1 ];
ou seja, faça:
>>A1=[10 -2 1 ; 5 2 5 ; -1 -1 0];
>>b1=[0; 4; 1 ];
>>x1=elimGaussSemPivot(A1,b1)
%OBS: A solução esperada é: x1 = -0.28070 -0.71930 1.36842
exemplo (2):
A2=[ -3 6 9 3; 2 -4 -5 -1; -3 8 8 1; 1 2 -6 4];
b2 =[12; -3; 8 ;3];
ou seja, faça:
>>A2=[ -3 6 9 3; 2 -4 -5 -1; -3 8 8 1; 1 2 -6 4];
>>b2 =[12; -3; 8 ;3];
>>x2=elimGaussSemPivot(A2,b2)
%OBS: A solução esperada é: x2 = 10.0000 1.5000 3.0000 2.0000
```

Observe que apesar do sistema $A2x = b2$ ser não singular, não vai ser possível obter a solução do sistema com a eliminação de Gauss ingênua. Entenda o motivo.
(Obs.: os exemplos acima já estão digitados no arquivo exemplosConjA.m) então pode copiar de lá ou carregar os exemplos digitando o nome do arquivo na linha de comando).

6. Rode o código implementado elimGaussSemPivot.m para obter a solução do sistema $Ax = b$ com matriz A e vetor b, dados abaixo:

```
A3=[ -3 8 -2 3; 0.47 -2 6 2; -2 3 1 6; 70 -1 2 3];
b3=[ 6; 6.47; 8; 74];
```