Métodos Numéricos para Engenharia

MÓDULO 8– ÁLGEBRA LINEAR NUMÉRICA - GAUSS JORDAN – LU - TRIDIAGONAL

PROFESSOR LUCIANO NEVES DA FONSECA

Instabilidade na Eliminação de Gauss

O algoritmo da Eliminação Gauss sem pivotamento não apresenta solução, se algum pivô da diagonal principal for nulo!

Exemplo: No Sistema ao lado o primeiro pivô da diagonal principal é nulo.

```
--> A=[0,1,3,5;-1,3,2,7;3,1,-3,2;-4,-2,1,5]
 A  =

   0.   1.   3.   5.
  -1.   3.   2.   7.
   3.   1.  -3.   2.
  -4.  -2.   1.   5.

--> b=[4;3;2;-3];
```

O Sistema tem solução, pois podemos resolvê-lo por Cramer ou mesmo pelo método da inversão.

```
--> x=Cramer(A,b,%f)
 x  =

   1.1975309
   0.2098765
   0.7901235
   0.2839506
```

```
--> x=inv_cofat(A)*b
 x  =

   1.1975309
   0.2098765
   0.7901235
   0.2839506
```

No entanto não há solução por Eliminação de Gauss!

```
--> x=EliminacaoGauss_SemPivot(A,b,%f)
Não há solução única pois matriz A é singular
 x  =

   Inf
   Inf
   Inf
   Inf
```

Instabilidade na Eliminação de Gauss

O algoritmo da Eliminação Gauss sem pivotamento apresenta um erro de arredondamento excessivo, isto é apresenta instabilidade numérica.

Exemplo: No Sistema abaixo demonstra o problema de instabilidade:

```
--> A=[1e-12,1;1000,-1000]
 A  =

   1.000D-12    1.
   1000.       -1000.

--> b=[1000;1];
```

O Sistema tem solução estável por Cramer ou mesmo pelo método da inversão.

```
--> x=Cramer(A,b,%f)
 x  =

   1000.0010
   1000.0000
```

```
--> x=inv_cofat(A)*b
 x  =

   1000.0010
   1000.0000
```

No entanto por Eliminação de Gauss há um considerável erro de arredondamento!

```
--> x=EliminacaoGauss_SemPivot(A,b,%f)
 x  =

   999.98942
   1000.0000
```

Eliminação de Gauss com pivotamento

```
function [C,dist]=PivotarColuna(p,linhas,C,prt)
    [N,M]=size(C)
    [max_lin_p,dist]=max(abs(C(linhas,p))); // pivotamento
    if(dist<>1) then
        C([ p, (dist+p-1)],:) = C([ (dist+p-1) , p ],:) //troca linhas
        if(prt)
            printf("Trocando linhas %d e %d",p,dist+p-1)
            disp(C)
        end
    end
endfunction
```

- O Pivotamento consiste simplesmente em se trocar linhas durante a eliminação
- Antes de se eliminar uma coluna, troca-se a linha do pivô com a linha que tenha o elemento de maior valor absoluto na coluna.

```
function x=EliminacaoGauss(A,b,prt)// com pivotamento
  [N N]=size(A);
  C=[A b];
  if(prt)
      printf("Matriz Aumentada [C=A|b]")
      disp(C)
  end
  for p=1:N-1
      C=PivotarColuna(p,[p:N],C,prt)
      for lin=p+1:N   //eliminação progressiva
          C=EliminarLinha(lin,p,C,prt)
      end
      if (prt)
          printf("Eliminando coluna %d com Pivô %f\n",p,C(p,p))
          disp(C)
      end
   end
   if C(p,p)<>0
      x=SubstituicaoRegressiva(C(:,1:N),C(:,N+1))
   else
      printf("Não há solução única pois matriz A é singular")
      x(1:N)=%inf
   end
endfunction
```

```
function x=EliminacaoGauss_SemPivot(A,b,prt)// sem pivotamento
  [N N]=size(A);
  C=[A b];
  if(prt)
      printf("Matriz Aumentada [C=A|b]")
      disp(C)
  end
  for p=1:N-1
      if C(p,p) == 0 then break; end
      for lin=p+1:N   //eliminação progressiva
          C=EliminarLinha(lin,p,C,prt)
      end
      if (prt)
          printf("\nEliminando coluna %d com Pivô %f\n",p,C(p,p))
          disp(C)
      end
   end
   if C(p,p)<>0
      x=SubstituicaoRegressiva(C(:,1:N),C(:,N+1))
   else
      printf("Não há solução única pois matriz A é singular\n")
      x(1:N)=%inf
   end
endfunction
```

Pivotamento de Gauss

- **O Pivotamento evita elementos nulos na diagonal principal**

  Se um elemento da diagonal principal for nulo, ou se algum elemento da diagonal se anular durante a eliminação, o sistema não terá, a princípio, solução por Eliminação de Gauss.

- **O Pivotamento Minimiza problemas de arredondamento.**

  A eliminação de Gauss é muito suscetível a erros de arredondamento, pois para se eliminar uma variável (torna-la nula !!), subtraímos dois números muito próximos, o que é uma fonte primária de erros.

```
--> A=[0,1,3,5;-1,3,2,7;3,1,-3,2;-4,-2,1,5];

--> b=[4;3;2;-3];

--> x=EliminacaoGauss_SemPivot(A,b,%f);
Não há solução única pois matriz A é singular
```

```
--> x=EliminacaoGauss(A,b,%t)
Matriz Aumentada [C=A|b]
   0.   1.   3.   5.   4.
  -1.   3.   2.   7.   3.
   3.   1.  -3.   2.   2.
  -4.  -2.   1.   5.  -3.
Trocando linhas 1 e 4
  -4.  -2.   1.   5.  -3.
  -1.   3.   2.   7.   3.
   3.   1.  -3.   2.   2.
   0.   1.   3.   5.   4.
(L2)=(L2)-(-1.000000)/(-4.000000)*L(1)
(L3)=(L3)-(3.000000)/(-4.000000)*L(1)
(L4)=(L4)-(0.000000)/(-4.000000)*L(1)
Eliminando coluna 1 com Pivô -4.000000

  -4.  -2.    1.     5.    -3.
   0.   3.5   1.75   5.75   3.75
   0.  -0.5  -2.25   5.75  -0.25
   0.   1.    3.     5.     4.
(L3)=(L3)-(-0.500000)/(3.500000)*L(2)
(L4)=(L4)-(1.000000)/(3.500000)*L(2)
Eliminando coluna 2 com Pivô 3.500000

  -4.  -2.    1.     5.          -3.
   0.   3.5   1.75   5.75         3.75
   0.   0.   -2.     6.5714286    0.2857143
   0.   0.    2.5    3.3571429    2.9285714
Trocando linhas 3 e 4
  -4.  -2.    1.     5.          -3.
   0.   3.5   1.75   5.75         3.75
   0.   0.    2.5    3.3571429    2.9285714
   0.   0.   -2.     6.5714286    0.2857143
(L4)=(L4)-(-2.000000)/(2.500000)*L(3)
Eliminando coluna 3 com Pivô 2.500000

  -4.  -2.    1.     5.          -3.
   0.   3.5   1.75   5.75         3.75
   0.   0.    2.5    3.3571429    2.9285714
   0.   0.    0.     9.2571429    2.6285714
Substituição regressiva
x(4)=0.283951
x(3)=0.790123
x(2)=0.209877
x(1)=1.197531
```

```
--> A=[1e-12,1;1000,-1000];

--> b=[1000;1];

--> x=EliminacaoGauss_SemPivot(A,b,%f)
 x  =

   999.98942
   1000.0000
```

```
--> x=EliminacaoGauss(A,b,%t)
Matriz Aumentada [C=A|b]
   1.000D-12    1.       1000.
   1000.       -1000.    1.
Trocando linhas 1 e 2
   1000.       -1000.    1.
   1.000D-12    1.       1000.

Eliminando coluna 1 com Pivô 1000.000000
(L2)=(L2)-(0.000000)/(1000.000000)*L(1)
   1000.   -1000.   1.
   0.       1.      1000.
Substituição regressiva
x(2)=1000.000000
x(1)=1000.001000
```

Cálculo do Determinante pela Eliminação Progressiva de Gauss

```
function d=det_gauss(A,prt)
  [N N]=size(A);
  npivot=0;
  if(prt)
      printf("Matriz A")
      disp(A)
  end
  for p=1:N-1
      [A,dist]=PivotarColuna(p,[p:N],A,prt)
      if A(p,p) == 0 then break; end
      if (prt)
          printf("Eliminando coluna %d com Pivô %f\n",p,A(p,p))
      end
      for lin=p+1:N  //eliminação progressiva
          A=EliminarLinha(lin,p,A,prt)
      end
      if (prt) disp(A) end
      if(dist<>1) npivot=npivot+1 end
   end
   d=prod(diag(A)) * (-1)^npivot
endfunction
```

```
function x=EliminacaoGauss(A,b,prt)// com pivotamento
  [N N]=size(A);
  C=[A b];
  if(prt)
      printf("Matriz Aumentada [C=A|b]")
      disp(C)
  end
  for p=1:N-1
      C=PivotarColuna(p,[p:N],C,prt)
      if C(p,p) == 0 then break; end
      if (prt)
          printf("\nEliminando coluna %d com Pivô %f\n",p,C(p,p))
      end
      for lin=p+1:N  //eliminação progressiva
          C=EliminarLinha(lin,p,C,prt)
      end
      if (prt) disp(C) end
   end
   if C(p,p)<>0
       x=SubstituicaoRegressiva(C(:,1:N),C(:,N+1))
   else
      printf("Não há solução única pois matriz A é singular")
      x(1:N)=%inf
   end
endfunction
```

- A Eliminação Progressiva de Gauss com pivotamento pode ser usada para o cálculo do determinante
- A eliminação Progressiva de Gauss transforma a Matriz A, característica do sistema, em uma matriz triangular superior .
- O determinante de uma matriz triangular superior é simplesmente o produto dos elementos de sua diagonal principal.

Cálculo do Determinante pela Eliminação Progressiva de Gauss

```
--> A=[2,1,3,5;-1,3,2,7;3,1,-3,2;-4,-2,1,5];

--> det_cofat(A)
 ans  =

  -532.
```

```
--> det_gauss(A,%f)
 ans  =

  -532.
```

```
--> det_gauss(A,%t)

   2.   1.   3.   5.
  -1.   3.   2.   7.
   3.   1.  -3.   2.
  -4.  -2.   1.   5.
Trocando linhas 1 e 4
  -4.  -2.   1.   5.
  -1.   3.   2.   7.
   3.   1.  -3.   2.
   2.   1.   3.   5.
Eliminando coluna 1 com Pivô -4.000000
(L2)=(L2)-(-1.000000)/(-4.000000)*L(1)
(L3)=(L3)-(3.000000)/(-4.000000)*L(1)
(L4)=(L4)-(2.000000)/(-4.000000)*L(1)
  -4.  -2.    1.     5.
   0.   3.5   1.75   5.75
   0.  -0.5  -2.25   5.75
   0.   0.    3.5    7.5
Eliminando coluna 2 com Pivô 3.500000
(L3)=(L3)-(-0.500000)/(3.500000)*L(2)
(L4)=(L4)-(0.000000)/(3.500000)*L(2)
  -4.  -2.    1.     5.
   0.   3.5   1.75   5.75
   0.   0.   -2.     6.5714286
   0.   0.    3.5    7.5
Trocando linhas 3 e 4
  -4.  -2.    1.     5.
   0.   3.5   1.75   5.75
   0.   0.    3.5    7.5
   0.   0.   -2.     6.5714286
Eliminando coluna 3 com Pivô 3.500000
(L4)=(L4)-(-2.000000)/(3.500000)*L(3)
  -4.  -2.    1.     5.
   0.   3.5   1.75   5.75
   0.   0.    3.5    7.5
   0.   0.    0.     10.857143

 ans  =

  -532.
```

Eliminação de Gauss-Jordan

- Quando uma variável é eliminada em Gauss-Jordan, é eliminada em todas as equações, não somente nas equações subsequentes, que é o caso da Eliminação de Gauss.
- A Eliminação de Gauss consiste na Eliminação Progressiva e a Substituição Regressiva. Só são eliminadas as linhas posteriores ao Pivô que esta sendo utilizado.
- A Eliminação de Progressiva de Gauss-Jordan é idêntica à da de Gauss.
- Também é idêntico o pivotamento
- No entanto, Gauss Jordan usa também uma Eliminação Regressiva, isto é, as linhas anteriores ao Pivô também serão eliminadas.
- A linha do Pivô também é normalizada, de modo que o Pivô tenha valor unitário
- Não é necessário a substituição regressiva, pois o resultado da eliminação é uma matriz diagonal unitária, e o vetor solução x será trivialmente obtido da coluna b após a eliminação.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & | & b_1 \\ a_{21} & a_{22} & a_{23} & | & b_2 \\ a_{31} & a_{32} & a_{33} & | & b_3 \end{bmatrix} \begin{matrix} (I) \\ (II) \\ (III) \end{matrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & | & b_1''' \\ 0 & 1 & 0 & | & b_2''' \\ 0 & 0 & 1 & | & b_3''' \end{bmatrix} \Rightarrow x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1''' \\ b_2''' \\ b_3''' \end{bmatrix}$$

Algoritmo para Eliminação de Gauss-Jordan

```
function x=EliminacaoGaussJordan(A,b,prt)
   [N M]=size(A);
   C=[A b];
   if(prt)
       printf("Matriz Aumentada [C=A|b]")
       disp(C)
   end
   for p=1:N
       C=PivotarColuna(p,[p:N],C,prt)
       if C(p,p) == 0 then break; end
       if (prt)
           printf("(L%d)=(1)/(%f)*(L%d)\n",p,C(p,p),p)
           printf("Eliminando coluna %d com Pivô 1.0\n",p)
       end
       C(p,:)=C(p,:)/C(p,p);  // normalização da linha p
       for lin=[1:p-1,p+1:N] //eliminação regressiva e progressiva
          C=EliminarLinha(lin,p,C,prt)
       end
       if (prt) disp(C) end
   end
   if C(p,p)<>0
       x=C(:,N+1)
   else
       printf("Não há solução única pois matriz A é singular")
       x(1:N)=%inf
   end
endfunction
```

```
function x=EliminacaoGauss(A,b,prt)// com pivotamento
   [N N]=size(A);
   C=[A b];
   if(prt)
       printf("Matriz Aumentada [C=A|b]")
       disp(C)
   end
   for p=1:N-1
       C=PivotarColuna(p,[p:N],C,prt)
       if C(p,p) == 0 then break; end
       if (prt)
           printf("\nEliminando coluna %d com Pivô %f\n",p,C(p,p))
       end
       for lin=p+1:N  //eliminação progressiva
           C=EliminarLinha(lin,p,C,prt)
       end
       if (prt) disp(C) end
   end
   if C(p,p)<>0
       x=SubstituicaoRegressiva(C(:,1:N),C(:,N+1))
   else
       printf("Não há solução única pois matriz A é singular")
       x(1:N)=%inf
   end
endfunction
```

Exemplo Eliminação de Gauss-Jordan

```
--> A=[2,1,3,5;-1,3,2,7;3,1,-3,2,;-4,-2,1,5]
 A  =


   2.    1.    3.    5.
  -1.    3.    2.    7.
   3.    1.   -3.    2.
  -4.   -2.    1.    5.

--> b=[4;3;2;-3]
 b  =


   4.
   3.
   2.
  -3.
```

```
--> EliminacaoGaussJordan(A,b,%t)
Matriz Aumentada [C=A|b]
   2.    1.    3.    5.    4.
  -1.    3.    2.    7.    3.
   3.    1.   -3.    2.    2.
  -4.   -2.    1.    5.   -3.
Trocando linhas 1 e 4
  -4.   -2.    1.    5.   -3.
  -1.    3.    2.    7.    3.
   3.    1.   -3.    2.    2.
   2.    1.    3.    5.    4.
Eliminando coluna 1 com Pivô -4.000000
(L1)=(1)/(-4.000000)*(L1)
(L2)=(L2)-(-1.000000)/(1.000000)*L(1)
(L3)=(L3)-(3.000000)/(1.000000)*L(1)
(L4)=(L4)-(2.000000)/(1.000000)*L(1)
   1.    0.5  -0.25  -1.25   0.75
   0.    3.5   1.75   5.75   3.75
   0.   -0.5  -2.25   5.75  -0.25
   0.    0.    3.5    7.5    2.5
Eliminando coluna 2 com Pivô 3.500000
(L2)=(1)/(3.500000)*(L2)
(L1)=(L1)-(0.500000)/(1.000000)*L(2)
(L3)=(L3)-(-0.500000)/(1.000000)*L(2)
(L4)=(L4)-(0.000000)/(1.000000)*L(2)
   1.    0.   -0.5   -2.0714286   0.2142857
   0.    1.    0.5    1.6428571   1.0714286
   0.    0.   -2.     6.5714286   0.2857143
   0.    0.    3.5    7.5         2.5
```

```
Trocando linhas 3 e 4
   1.    0.   -0.5   -2.0714286   0.2142857
   0.    1.    0.5    1.6428571   1.0714286
   0.    0.    3.5    7.5         2.5
   0.    0.   -2.     6.5714286   0.2857143
Eliminando coluna 3 com Pivô 3.500000
(L3)=(1)/(3.500000)*(L3)
(L1)=(L1)-(-0.500000)/(1.000000)*L(3)
(L2)=(L2)-(0.500000)/(1.000000)*L(3)
(L4)=(L4)-(-2.000000)/(1.000000)*L(3)
   1.    0.    0.   -1.          0.5714286
   0.    1.    0.    0.5714286   0.7142857
   0.    0.    1.    2.1428571   0.7142857
   0.    0.    0.    10.857143   1.7142857
Eliminando coluna 4 com Pivô 10.857143
(L4)=(1)/(10.857143)*(L4)
(L1)=(L1)-(-1.000000)/(1.000000)*L(4)
(L2)=(L2)-(0.571429)/(1.000000)*L(4)
(L3)=(L3)-(2.142857)/(1.000000)*L(4)


   1.    0.    0.    0.    0.7293233
   0.    1.    0.    0.    0.6240602
   0.    0.    1.    0.    0.3759398
   0.    0.    0.    1.    0.1578947
```

Matriz Inversa por Gauss Jordan

- A inversa de uma matriz quadrada A pode ser computada através da eliminação de Gauss Jordan aplicada à matriz aumentada, formada pela matriz A, que se quer inverter, e a matriz identidade, com a mesma ordem da matriz A.
- Para calcular a inversa A^{-1}, devemos então montar a matriz aumentada com A na esquerda e a matriz identidade na direita, e proceder com a eliminação de Gauss Jordan.
- Quando a esquerda contiver a matriz identidade, a direita conterá ter A^{-1}
- O pivotamento não altera o resultado.

$$\begin{array}{c} \qquad [A] \qquad\qquad\qquad [I] \\ \left[\begin{array}{ccc|ccc} a_{11} & a_{12} & a_{13} & 1 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 1 & 0 \\ a_{31} & a_{32} & a_{33} & 0 & 0 & 1 \end{array}\right] \begin{array}{c} (I) \\ (II) \\ (III) \end{array} \end{array} \Rightarrow \begin{array}{c} [I] \qquad\qquad\qquad [A^{-1}] \\ \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & z_{11} & z_{12} & z_{13} \\ 0 & 1 & 0 & z_{21} & z_{22} & z_{23} \\ 0 & 0 & 1 & z_{31} & z_{32} & z_{33} \end{array}\right] \end{array}$$

Inversa por Gauss Jordan

```
function Ai=inv_gauss(A,prt)
  [N N]=size(A);
  C=[A diag(ones(1:N))];
  if(prt)
      printf("Matriz Aumentada [C=A|Ai]")
      disp(C)
  end
  for p=1:N
      C=PivotarColuna(p,[p:N],C,prt)
      if C(p,p) == 0 then break; end
      if (prt)
          printf("Eliminando coluna %d com Pivô %f\n",p,C(p,p))
          printf("(L%d)=(1)/(%f)*(L%d)\n",p,C(p,p),p)
      end
      C(p,:)=C(p,:)/C(p,p);  // normalização da linha p
      for lin=[1:p-1,p+1:N] //eliminação regressiva e progressiva
          C=EliminarLinha(lin,p,C,prt)
      end
      if (prt) disp(C)  end
   end
   if C(p,p)<>0
      Ai=C(1:N,N+1:2*N)
   else
      printf("Não há solução única pois matriz A é singular")
      Ai=[]
   end
endfunction
```

```
function x=EliminacaoGaussJordan(A,b,prt)
  [N M]=size(A);
  C=[A b];
  if(prt)
      printf("Matriz Aumentada [C=A|b]")
      disp(C)
  end
  for p=1:N
      C=PivotarColuna(p,[p:N],C,prt)
      if C(p,p) == 0 then break; end
      if (prt)
          printf("(L%d)=(1)/(%f)*(L%d)\n",p,C(p,p),p)
          printf("Eliminando coluna %d com Pivô 1.0\n",p)
      end
      C(p,:)=C(p,:)/C(p,p);  // normalização da linha p
      for lin=[1:p-1,p+1:N] //eliminação regressiva e progressiva
          C=EliminarLinha(lin,p,C,prt)
      end
      if (prt) disp(C) end
   end
   if C(p,p)<>0
      x=C(:,N+1)
   else
      printf("Não há solução única pois matriz A é singular")
      x(1:N)=%inf
   end
endfunction
```

Inversa por Gauss Jordan

```
--> A=[2,1,3,5;-1,3,2,7;3,1,-3,2,;-4,-2,1,5]
 A  =


  2.   1.   3.   5.
 -1.   3.   2.   7.
  3.   1.  -3.   2.
 -4.  -2.   1.   5.
```

```
--> Ai=inv_cofat(A)
 Ai  =

   0.1954887  -0.1296992   0.0921053  -0.0507519
  -0.1729323   0.2781955  -0.0526316  -0.1954887
   0.1729323  -0.0281955  -0.1973684  -0.0545113
   0.0526316   0.0131579   0.0921053   0.0921053
```

```
--> Ai=inv_gauss(A,%f)
 Ai  =

   0.1954887  -0.1296992   0.0921053  -0.0507519
  -0.1729323   0.2781955  -0.0526316  -0.1954887
   0.1729323  -0.0281955  -0.1973684  -0.0545113
   0.0526316   0.0131579   0.0921053   0.0921053
```

```
--> Ai=inv_gauss(A,%t)
Matriz Aumentada [C=A|Ai]
   2.   1.   3.   5.   1.   0.   0.   0.
  -1.   3.   2.   7.   0.   1.   0.   0.
   3.   1.  -3.   2.   0.   0.   1.   0.
  -4.  -2.   1.   5.   0.   0.   0.   1.
Trocando linhas 1 e 4
  -4.  -2.   1.   5.   0.   0.   0.   1.
  -1.   3.   2.   7.   0.   1.   0.   0.
   3.   1.  -3.   2.   0.   0.   1.   0.
   2.   1.   3.   5.   1.   0.   0.   0.
Eliminando coluna 1 com Pivô 1.000000
(L1)=(1)/(1.000000)*(L1)
(L2)=(L2)-(-1.000000)/(1.000000)*L(1)
(L3)=(L3)-(3.000000)/(1.000000)*L(1)
(L4)=(L4)-(2.000000)/(1.000000)*L(1)
   1.   0.5  -0.25  -1.25   0.   0.   0.  -0.25
   0.   3.5   1.75   5.75   0.   1.   0.  -0.25
   0.  -0.5  -2.25   5.75   0.   0.   1.   0.75
   0.   0.    3.5    7.5    1.   0.   0.   0.5
Eliminando coluna 2 com Pivô 1.000000
(L2)=(1)/(1.000000)*(L2)
(L1)=(L1)-(0.500000)/(1.000000)*L(2)
(L3)=(L3)-(-0.500000)/(1.000000)*L(2)
(L4)=(L4)-(0.000000)/(1.000000)*L(2)
   1.   0.  -0.5  -2.071   0.  -0.143   0.  -0.214
   0.   1.   0.5   1.643   0.   0.286   0.  -0.071
   0.   0.  -2.    6.571   0.   0.143   1.   0.714
   0.   0.   3.5   7.5     1.   0.      0.   0.5
Trocando linhas 3 e 4
   1.   0.  -0.5  -2.071   0.  -0.143   0.  -0.214
   0.   1.   0.5   1.643   0.   0.286   0.  -0.071
   0.   0.   3.5   7.5     1.   0.      0.   0.5
   0.   0.  -2.    6.571   0.   0.143   1.   0.714
```

```
Eliminando coluna 3 com Pivô 1.000000
(L3)=(1)/(1.000000)*(L3)
(L1)=(L1)-(-0.500000)/(1.000000)*L(3)
(L2)=(L2)-(0.500000)/(1.000000)*L(3)
(L4)=(L4)-(-2.000000)/(1.000000)*L(3)
   1.   0.   0.  -1.       0.143  -0.143   0.  -0.143
   0.   1.   0.   0.571  -0.143   0.286   0.  -0.143
   0.   0.   1.   2.143   0.286   0.      0.   0.143
   0.   0.   0.   10.86   0.571   0.143   1.   1.
Eliminando coluna 4 com Pivô 1.000000
(L4)=(1)/(1.000000)*(L4)
(L1)=(L1)-(-1.000000)/(1.000000)*L(4)
(L2)=(L2)-(0.571429)/(1.000000)*L(4)
(L3)=(L3)-(2.142857)/(1.000000)*L(4)

   1.   0.   0.   0.   0.195  -0.13    0.092  -0.051
   0.   1.   0.   0.  -0.173   0.278  -0.053  -0.195
   0.   0.   1.   0.   0.173  -0.028  -0.197  -0.055
   0.   0.   0.   1.   0.053   0.013   0.092   0.092
 Ai  =

   0.195  -0.13    0.092  -0.051
  -0.173   0.278  -0.053  -0.195
   0.173  -0.028  -0.197  -0.055
   0.053   0.013   0.092   0.092
```

Sistemas Tridiagonais

- Sistemas tridiagonais $Ay = r$ são sistemas lineares esparsos que surgem na solução de muitos problemas de engenharia.
- Em métodos numéricos aparecem particularmente na resolução de equações diferenciais pelos métodos de diferenças finitas e também na resolução de splines cúbicas.
- Em um sistema tridiagonal, os únicos elementos não nulos na matriz característica A são os elementos da diagonal principal (dp), da diagonal acima da diagonal principal (du)e na diagonal abaixo da diagonal principal (dl).
- Deste modo a Eliminação de Gauss só precisa ser feita na linha abaixo do pivô.
- Como a matriz A é esparsa, podemos armazenar somente os três vetores diagonais dp, dl e du, evitando o armazenamento desnecessário dos elemento nulos.

dp_1	du_2	0	0	0	0	0	0	0	0	0		y_1		r_1
dl_2	dp_2	du_3	0	0	0	0	0	0	0	0		y_2		r_2
0	dl_3	dp_3	du_4	0	0	0	0	0	0	0		y_3		r_3
0	0	dl_4	dp_4	du_5	0	0	0	0	0	0		y_4		r_4
0	0	0	dl_5	dp_5	du_6	0	0	0	0	0		y_5		r_5
0	0	0	0	dl_6	dp_6	du_7	0	0	0	0	X	y_6	=	r_6
0	0	0	0	0	dl_7	dp_7	du_8	0	0	0		y_7		r_7
0	0	0	0	0	0	dl_8	dp_8	...	0	0		y_8		r_8
0	0	0	0	0	0	0	...	...	du_{n-1}	0		...		...
0	0	0	0	0	0	0	0	dl_{n-1}	dp_{n-1}	du_n		y_{n-1}		r_{n-1}
0	0	0	0	0	0	0	0	0	dl_n	dp_n		y_n		r_n

Resolução de sistemas tridiagonais

- Sistemas tridiagonais podem ser resolvidos diretamente pelo algoritmo da Eliminação de Gauss, quando serão eliminadas as linhas abaixo do pivô (as outras já são nulas!!).
- Por este motivo, estes sistemas tridiagonais podem ser resolvidos de uma maneira mais eficientemente e mais estável numericamente através do Método de Thomas, que nada mais é que um Eliminação de Gauss Simplificada
- Neste método, não precisamos montar a matriz característica A, precisamos apenas os 3 vetores 'dp(1:n)', 'dl(2:n)' e 'du(2:n)' o velor 'r(1:n)' com os termos independentes.
- Então fazemos um eliminação progressiva em uma dimensão no vetor dp, seguida de uma substituição regressiva em uma dimensão para encontramos o vetor solução y(1:n).

eliminação progressiva

$$
\begin{aligned}
& for\ k = 2{:}N \\
& \qquad m = \frac{dl_{k-1}}{dp_{k-1}} \\
& \qquad dp_k = dp_k - m\, du_{k-1} \\
& \qquad r_k = r_k - m\, r_{k-1} \\
& end
\end{aligned}
$$

substituição regressiva

$$
\begin{aligned}
& y_n = \frac{r_n}{dp_n} \\
& for\ k = N-1{:}-1{:}1 \\
& \qquad y_k = \frac{r_k - du_k * y_{k-1}}{dp_k} \\
& end
\end{aligned}
$$

Algoritmo para Resolução de Sistemas Tridiagonais

```
function y=tridiagonal(dl,dp,du,r)
    N=length(r);
    for k=2:N  //eliminação progressiva
        m=dl(k-1)/dp(k-1);
        dp(k)=dp(k)-m*du(k-1);
        r(k)=r(k)-m*r(k-1);
    end
    y(1,N)=r(N)/dp(N); //substituição regressiva
    for k= N-1:-1:1
        y(1,k)=(r(k)-du(k)*y(k+1))/dp(k);
    end
endfunction
```

Consiste simplesmente de uma eliminação progressiva em uma dimensão no vetor dp, seguida de uma substituição regressiva em uma dimensão para encontramos o vetor solução y

```
--> dl=[7,9,4,1,2,34,32,65,23,12,12,5,9,3];

--> dp=[1,3,5,3,4,6,7,5,3,5,6,8,10,7,14];

--> du=[3,7,3,2,1,5,3,9,8,24,12,6,7,8];

--> A=diag(du,+1)+diag(dp)+diag(dl,-1)
 A  =

   1.   3.   0.   0.   0.   0.    0.    0.    0.    0.    0.    0.    0.    0.   0.
   7.   3.   7.   0.   0.   0.    0.    0.    0.    0.    0.    0.    0.    0.   0.
   0.   9.   5.   3.   0.   0.    0.    0.    0.    0.    0.    0.    0.    0.   0.
   0.   0.   4.   3.   2.   0.    0.    0.    0.    0.    0.    0.    0.    0.   0.
   0.   0.   0.   1.   4.   1.    0.    0.    0.    0.    0.    0.    0.    0.   0.
   0.   0.   0.   0.   2.   6.    5.    0.    0.    0.    0.    0.    0.    0.   0.
   0.   0.   0.   0.   0.   34.   7.    3.    0.    0.    0.    0.    0.    0.   0.
   0.   0.   0.   0.   0.   0.    32.   5.    9.    0.    0.    0.    0.    0.   0.
   0.   0.   0.   0.   0.   0.    0.    65.   3.    8.    0.    0.    0.    0.   0.
   0.   0.   0.   0.   0.   0.    0.    0.    23.   5.    24.   0.    0.    0.   0.
   0.   0.   0.   0.   0.   0.    0.    0.    0.    12.   6.    12.   0.    0.   0.
   0.   0.   0.   0.   0.   0.    0.    0.    0.    0.    12.   8.    6.    0.   0.
   0.   0.   0.   0.   0.   0.    0.    0.    0.    0.    0.    5.    10.   7.   0.
   0.   0.   0.   0.   0.   0.    0.    0.    0.    0.    0.    0.    9.    7.   8.
   0.   0.   0.   0.   0.   0.    0.    0.    0.    0.    0.    0.    0.    3.   14.

--> r=[1;1.98;4.99;2.98;6.97;4.01;4.98;2.25;3.37;6.94;4.92;3.04;5.97;7.91;1.89];
```

```
--> x=EliminacaoGauss(A,r)
 x  =

  -0.770
   0.590
   0.8
  -1.44
   2.05
   0.210
  -0.270
  -0.090
   1.260
   0.680
  -1.060
   0.260
   2.280
  -2.590
   0.690
```

```
--> x=tridiagonal(dl,dp,du,r)'
 x  =

  -0.770
   0.590
   0.8
  -1.44
   2.05
   0.210
  -0.270
  -0.090
   1.260
   0.680
  -1.060
   0.260
   2.280
  -2.590
   0.690
```

Conteúdo Opcional

Fatoração LU (opcional)

- A Fatoração LU transforma a matriz A em duas matrizes L e U, através de uma eliminação de Gauss
- L é triangular inferior com diagonal unitária
- U é uma matriz triangular superior
- U é obtida diretamente do resultado final de uma Eliminação de Gauss
- L é obtida através dos fatores utilizados na Eliminação Progressiva das colunas, adicionados de uma diagonal principal unitária.
- Durante a eliminação, as trocas de linhas do pivotamento devem ser guardadas para posterior permutação dos elemento do vetor b (na resolução de um sistema Ax =b)

$$\underset{A}{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}} \Rightarrow \underset{(L)}{\begin{bmatrix} 1 & 0 & 0 \\ f_{21} & 1 & 0 \\ f_{31} & f_{32} & 1 \end{bmatrix}} \underset{(U)}{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}' & a_{23}' \\ 0 & 0 & a_{33}'' \end{bmatrix}}$$

Resolução de Sistemas através dos Fatores LU da matriz característica A

$$\underset{A}{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}} \Rightarrow \underset{(L)}{\begin{bmatrix} 1 & 0 & 0 \\ f_{21} & 1 & 0 \\ f_{31} & f_{32} & 1 \end{bmatrix}} \underset{(U)}{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}' & a_{23}' \\ 0 & 0 & a_{33}'' \end{bmatrix}}$$

- Uma vez fatorada a matriz A em duas matrizes L e U, o sistema $Ax = b$ pode ser resolvido rapidamente através de duas substituições, uma progressiva $Ly = b$ e outra regressiva $Ux = y$:

$$1)\ Ly = b \quad (entramos\ com\ b\ obtemos\ y)$$

$$2)\ Ux = y \quad (entramos\ com\ y\ obtemos\ x)$$

$$\boxed{Ax = b \quad (LU)x = b \quad \mathrm{L}(Ux) = b}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow \underset{Ly = b}{\left[\begin{array}{ccc|c} 1 & 0 & 0 & b_1 \\ f_{21} & 1 & 0 & b_2 \\ f_{31} & f_{32} & 1 & b_3 \end{array}\right]} \Rightarrow \underset{Ux = y}{\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & y_1 \\ 0 & a_{22}' & a_{23}' & y_2 \\ 0 & 0 & a_{33}'' & y_3 \end{array}\right]} \Rightarrow x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Exemplo de Fatoração LU

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

$$U = \begin{bmatrix} 2 & 1 & -3 \\ -1 & 3 & 2 \\ 3 & 1 & -3 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Pivotando linha 1 e 3

$$U = \begin{bmatrix} \mathbf{3} & 1 & -3 \\ -1 & 3 & 2 \\ 2 & 1 & -3 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$L_2 = L_2 - \left(\frac{-1}{3}\right) L_1$$

$$L_3 = L_3 - (2/3)L_1$$

$$U = \begin{bmatrix} 3 & 1 & -3 \\ 0 & \mathbf{10/3} & 1 \\ 0 & 1/3 & -1 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ 2/3 & 0 & 1 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$L_3 = L_3 - (1/3)/(10/3)\, L_2$$

$$U = \begin{bmatrix} 3 & 1 & -3 \\ 0 & 10/3 & 1 \\ 0 & 0 & -1.1 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ 2/3 & 0.1 & 1 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ 2/3 & 1/10 & 1 \end{bmatrix} \quad U = \begin{bmatrix} 3 & 1 & -3 \\ 0 & 10/3 & 1 \\ 0 & 0 & -1.1 \end{bmatrix}$$

$$U = \begin{bmatrix} 3 & 1 & -3 \\ 0 & 10/3 & 1 \\ 0 & 0 & -1.1 \end{bmatrix}$$

U é obtida diretamente do Resultado final Eliminação de Gauss.

L é obtida através dos fatores utilizados na Eliminação de Gauss, acrescido de uma diagonal principal unitária.

$$A_r = LU = \begin{bmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ 2/3 & 1/10 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & -3 \\ 0 & 10/3 & 1 \\ 0 & 0 & -1.1 \end{bmatrix} = \begin{bmatrix} 3 & 1 & -3 \\ -1 & 3 & 2 \\ 2 & 1 & -3 \end{bmatrix}$$

Notar que as mesmas permutações de linhas (pivotamento) feitas na matriz U, devem também ser feitas na matriz L. No entanto, somente as colunas anteriores à diagonal principal devem ser pivotadas

Algoritmo para Fatoração LU - Idêntico ao da Eliminação de Gauss com Pivotamento

```
function [L,U,P]= FatoracaoLU(A,prt)
   [N M]=size(A);
   L=diag(ones(1:N))
   P=L //Matriz de permutação de linhas
   U=A
   if (prt) disp([L,U])  end
   for p=1:N-1
     [U,dist]=PivotarColuna(p, [p:N],U,%f)
     if(dist<>1) // se houve pivotamento
       P([ p, (dist+p-1)],:) = P([ (dist+p-1) , p ],:) //troca linhas
       for (col=1:p) // troca linhas abaixo da diagonal
         if (col<p) L([p, (dist+p-1)],col) = L([(dist+p-1),p],col) end
       end
       if (prt)
         printf("Trocando linhas %d e %d de U\n",p,p+dist-1)
         printf("Trocando linhas %d e %d de L abaixo da diagonal",p,p+dist-1)
         disp([L,U])
       end
     end
     if U(p,p) == 0 then break; end
     if (prt)
        printf("Eliminando coluna %d com Pivô %f\n",p,U(p,p))
     end
     for lin=p+1:N   //eliminação progressiva
        [U,m]=EliminarLinha(lin,p,U,prt)
        L(lin,p) = m;
     end
     if (prt) disp([L,U])  end
   end
endfunction
```

- Notar que na fatoração, só entramos com a Matriz característica A.
- Não há vetor b, pois não estamos resolvendo o sistema Ax=b, mas simplesmente fatorando a matriz A.
- A saída constituirá da Matriz L, da Matriz U e da matriz P de permutações de linhas
- O algoritmo consiste somente de uma eliminação progressiva, idêntica à de Gauss.
- A matriz U é inicializada com a matriz A, as matrizes L e P com a matriz identidade.
- A matriz U de saída é o resultado da eliminação progressiva Gauss.
- A matriz L de saída é formada pelos fatores m de eliminação com um a diagonal principal unitária.
- As mesmas permutações de linhas (pivotamento) feitas na matriz U, devem também ser feitas na matriz P. No entanto, na matriz L só nas linhas abaixo da diagonal principal devem ser permutadas.

Algoritmo para Fatoração LU - Idêntico ao da Eliminação de Gauss com Pivotamento

```
function [U,L,P]= FatoracaoLU(U,prt)
  [N M]=size(U);
  L=diag(ones(1:N))
  P=diag(ones(1:N))
  if (prt)
      printf("Matrizes [U : L : P]")
      disp([U,L,P])
  end
  for p=1:N-1
    [U,L,P,dist]=PivotarColuna_LU(p,[p:N],U,L,P,prt)
    if U(p,p) == 0 then break; end
    if (prt)
        printf("Eliminando coluna %d com Pivô %f em U\n",p,U(p,p))
        printf("Escrevendo fatores  m na coluna %d de L\n",p)
    end
    for lin=p+1:N   //eliminação progressiva
       [U,m]=EliminarLinha(lin,p,U,prt)
       L(lin,p) = m; //Salvar fatores m em L
    end
    if (prt) disp([U,L,P])  end
  end
endfunction
```

- Notar que na fatoração, só entramos com a Matriz característica A.
- Não há vetor b, pois não estamos resolvendo o sistema Ax=b, mas simplesmente fatorando a matriz A.
- A saída constituirá da Matriz L, da Matriz U e da matriz P de permutações de linhas
- O algoritmo consiste somente de uma eliminação progressiva, idêntica à de Gauss.
- A matriz U é inicializada com a matriz A, as matrizes L e P com a matriz identidade.
- A matriz U de saída é o resultado da eliminação progressiva Gauss.
- A matriz L de saída é formada pelos fatores m de eliminação com um a diagonal principal unitária.

```
function [U,L,P,dist]=PivotarColuna_LU(p,linhas,U,L,P,prt)
    [N,M]=size(U)
    [max_lin_p,dist]=max(abs(U(linhas,p))); // pivotamento
    if(dist<>1) then
        U([ p, (dist+p-1)],:) = U([ (dist+p-1) , p ],:)
        P([ p, (dist+p-1)],:) = P([ (dist+p-1) , p ],:)
        for (col=1:p) // troca linhas de L abaixo da diagonal
         if (col<p) L([p, (dist+p-1)],col) = L([(dist+p-1),p],col) end
        end
        if(prt)
            printf("Trocando linhas %d e %d de U e L, e de P se col<p ",p,dist+p-1)
            disp([U,L,P])
        end
    end
endfunction
```

- Precisamos de um novo algoritmo de pivotamento, pois as mesmas permutações de linhas (pivotamento) feitas na matriz U, devem também ser feitas na matriz P. No entanto, na matriz L só nas linhas abaixo da diagonal principal devem ser pivotadas.

Exemplo de Fatoração LU

```
--> A=[2,1,-3;-1,3,2;3,1,-3];

--> [L,U,P]= FatoracaoLU(A,%t)

   1.   0.   0.   2.   1.  -3.
   0.   1.   0.  -1.   3.   2.
   0.   0.   1.   3.   1.  -3.
Trocando linhas 1 e 3 de U
Trocando linhas 1 e 3 de L abaixo da diagonal
   1.   0.   0.   3.   1.  -3.
   0.   1.   0.  -1.   3.   2.
   0.   0.   1.   2.   1.  -3.
Eliminando coluna 1 com Pivô 3.000000
(L2)=(L2)-(-1.000000)/(3.000000)*L(1)
(L3)=(L3)-(2.000000)/(3.000000)*L(1)
   1.      0.   0.   3.   1.      -3.
  -0.333   1.   0.   0.   3.333    1.
   0.667   0.   1.   0.   0.333   -1.
Eliminando coluna 2 com Pivô 3.333333
(L3)=(L3)-(0.333333)/(3.333333)*L(2)
   1.      0.    0.   3.   1.      -3.
  -0.333   1.    0.   0.   3.333    1.
   0.667   0.1   1.   0.   0.      -1.1
 L  =

   1.      0.    0.
  -0.333   1.    0.
   0.667   0.1   1.
 U  =

   3.   1.      -3.
   0.   3.333    1.
   0.   0.      -1.1
 P  =

   0.   0.   1.
   0.   1.   0.
   1.   0.   0.
```

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

$$U = \begin{bmatrix} 3 & 1 & -3 \\ 0 & 10/3 & 1 \\ 0 & 0 & -1.1 \end{bmatrix}$$

U é obtida diretamente do Resultado final Eliminação de Gauss.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ 2/3 & 1/10 & 1 \end{bmatrix}$$

L é obtida através dos fatores utilizados na Eliminação de Gauss, acrescido de uma diagonal principal unitária.

$$A_r = LU = \begin{bmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ 2/3 & 1/10 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & -3 \\ 0 & 10/3 & 1 \\ 0 & 0 & -1.1 \end{bmatrix} = \begin{bmatrix} 3 & 1 & -3 \\ -1 & 3 & 2 \\ 2 & 1 & -3 \end{bmatrix}$$

$$A = PLU = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ 2/3 & 1/10 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & -3 \\ 0 & 10/3 & 1 \\ 0 & 0 & -1.1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & -3 \\ -1 & 3 & 2 \\ 3 & 1 & -3 \end{bmatrix}$$

Notar que as mesmas permutações de linhas (pivotamento) feitas na matriz U, devem também ser feitas na matriz L. No entanto, somente as colunas anteriores à diagonal principal devem ser pivotadas

Resolução de um sistema $Ax = b$ através dos fatores LU

Uma vez feita a fatoração, podemos introduzir o vetor de entrada b e calcular a saída x.
Primeiro permutamos o vetor b
Então fazemos uma substituição progressiva em $Ly = b$ para encontramos y.
Depois faremos um substituição regressiva em $Ux = y$ para encontrarmos x

$$b = \begin{bmatrix} -1 \\ 12 \\ 0 \end{bmatrix} \qquad P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad b_p = Pb = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 12 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 12 \\ -1 \end{bmatrix}$$

substituição progressiva

$L\,y = b_p$

$$\begin{bmatrix} 1 & 0 & 0 & | & 0 \\ -1/3 & 1 & 0 & | & 12 \\ 2/3 & 1/10 & 1 & | & -1 \end{bmatrix} \begin{matrix} (I) \\ (II) \\ (III) \end{matrix}$$

$(1)y_1 = 0 \quad \rightarrow y_1 = 0$

$\left(-\frac{1}{3}\right) y_1 + (1)y_2 = 12 \quad \rightarrow \quad y_2 = 12$

$\left(\frac{2}{3}\right) y_1 + \left(\frac{1}{10}\right) y_2 + (1)y_3 = -1 \rightarrow \quad y_3 = -2.2$

substituição regressiva

$U\,x = y$

$$\begin{bmatrix} 3 & 1 & -3 & | & 0 \\ 0 & 10/3 & 1 & | & 12 \\ 0 & 0 & -1.1 & | & -2.2 \end{bmatrix} \begin{matrix} (I) \\ (II) \\ (III) \end{matrix}$$

$(-1.1)x_3 = -2.2 \quad \rightarrow x_3 = 2$

$\left(\frac{10}{3}\right) x_2 + (1)x_3 = 12 \quad \rightarrow x_2 = 3$

$(3)x_1 + (1)x_2 + (-3)x_3 = 0 \quad \rightarrow x_1 = 1$

$$x = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

Algoritmo Resolução de um sistema $Ax = b$ através dos fatores LU

- O Algoritmo recebe os resultados da fatoração LU e o vetor b, para resolver o sistema Ax=b através duas substituições (Progressiva e Regressiva).

```
function x= SolucaoLU(U,L,P,b,prt)
    bp=P*b
    y=SubstituicaoProgressiva(L,bp,%f)
    x=SubstituicaoRegressiva(U,y,%f)
    if (prt)
        printf("[b bp]")
        disp([b bp])
        printf("Sistema L y = bp")
        disp(y)
        printf("Sistema U x = y")
        disp(x)
    end
endfunction
```

```
function y=SubstituicaoProgressiva(A,b,prt)
  [N,M]=size(A)
  y(1)=b(1);
  if (prt)  printf("Substituição Progressiva\ny(%d)=%f\n",1,y(1)); end
  for lin=2:N
        y(lin)= b(lin)-A(lin,1:lin-1)*y(1:lin-1);
         if (prt) printf("y(%d)=%f\n",lin,y(lin)) end
  end
endfunction
```

- Na sequência completa, começamos coma Matriz A, que é fatorada em L, U e matriz de permutações P.
- Entramos então com o vetor de entrada b e permutamos suas linhas $b_p = Pb$
- Fazemos então uma substituição progressiva em $Ly = b_p$ para encontra y.
- Depois fazemos um substituição regressiva em $Ux = y$ para encontrarmos x
- Este procedimento se justifica ser tivermos vários vetores b de entrada, para o mesmo sistema A

```
--> A=[2,1,-3;-1,3,2;3,1,-3];

--> [U,L,P]= FatoracaoLU(A,%f);

--> b=[-1;12;0];

--> x=SolucaoLU(U,L,P,b,%t);
[b bp]
  -1.    0.
   12.   12.
   0.   -1.
Sistema L y = bp
   0.
   12.
  -2.2
Sistema U x = y
   1.
   3.
   2.
```

Exemplo: Resolver os sistemas $Ax = b_1$, $Ax = b_2$ e $Ax = b_3$

Mesma matriz característica A, porém com três vetores distintos de entrada b_1, b_2 e b_3

Entrada | Sistema | Saída

$$b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow \boxed{A = L\,U} \Rightarrow x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

```
--> A=[2,1,-3;-1,3,2;3,1,-3];

--> [U,L,P]= FatoracaoLU(A,%f);
```

Primeiro Fatoramos a Matriz Característica A em L e U

```
--> b=[-1;12;0];

--> x=SolucaoLU(U,L,P,b,%t);
[b bp]
  -1.     0.
   12.    12.
   0.    -1.
Sistema L y = bp
   0.
   12.
  -2.2
Sistema U x = y
   1.
   3.
   2.
```

```
--> b=[0;4;1];

--> x=SolucaoLU(U,L,P,b,%t);
[b bp]
   0.    1.
   4.    4.
   1.    0.
Sistema L y = bp
   1.
   4.3333333
  -1.1
Sistema U x = y
   1.
   1.0000000
   1.
```

```
--> b=[-9;117;5];

--> x=SolucaoLU(U,L,P,b,%t);
[b bp]
  -9.      5.
   117.    117.
   5.     -9.
Sistema L y = bp
   5.
   118.66667
  -24.2
Sistema U x = y
   14.000000
   29.
   22.000000
```

```
--> b=[215;-93;267];

--> x=SolucaoLU(U,L,P,b,%t);
[b bp]
   215.    267.
  -93.    -93.
   267.    215.
Sistema L y = bp
   267.
  -4.
   37.400000
Sistema U x = y
   52.
   9.
  -34.
```