Capítulo 10

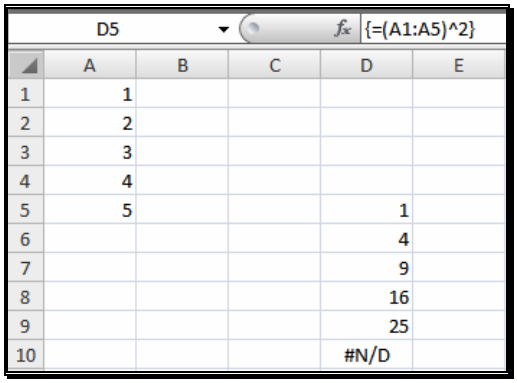
Álgebra de Matrizes (Arranjos)

Já varias vezes tivemos contatos com variáveis indexados em matrizes ou vetores (= matriz especial). Compare o primeiro capitulo, pagina 11, ou no Capítulo 7. Diz-se que o **arranjo** é um *vetor* quando seus itens possuem um único índice e que o arranjo é uma *matriz*, quando seus itens (elementos) possuem dois (ou mais) índices.

No Excel, uma matriz pode ser dada como um intervalo de células, como A1:C3. Mas, também pode ser armazenada na "memória" do Excel e é, em tal caso, chamado de *constante de matriz* (array constant). Para criar uma constante de matriz, utiliza-se a seguinte notação: {1.2.3.4.5}. Podemos usar esta constante como argumento de uma função, por exemplo de SOMA. Escrevemos =SOMA({1.2.3.4.5}), Enter. Resultado: 15. (A fórmula contém uma matriz, mas, a fórmula mesma não é nenhuma *fórmula matricial* (array formula) e não precisa ser entrada utilizando Ctrl+Shift+Enter, como devemos fazer no caso de uma fórmula matricial. (Se bem que, em nosso caso, podemos fazê-lo.) Excel sempre coloca duas chaves { } em torno duma fórmula matricial. Mas, você não deve escrevê-las, isso faz Excel.)

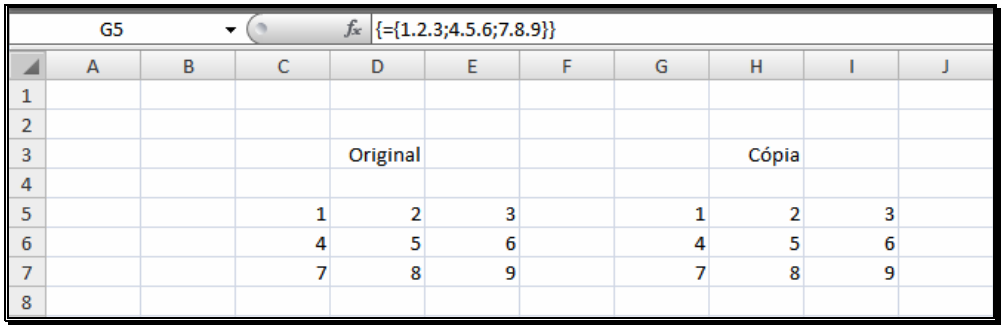
Na edição portuguesa do Excel, devemos escrever os elementos de uma *lista* (vetor) *horizontal*, separados por pontos (não por vírgulas!). Se os elementos ficam numa coluna, eles são separados por ponto e vírgula.

Para mostrar os elementos 1,2,3,4,5 nas células A1:A5, escrevemos ={1;2;3;4;5}, Ctrl+Shift+Enter. Mas, antes devemos selecionar o intervalo A1:A5! Se, em seguida, queremos quadrar estes elementos, então selecionamos outras cinco células verticais, por exemplo D5:D9, escrevemos =(A1:A5)^2 e pressionamos Ctrl+Shift +Enter.



Mas, cometi um equívoco, selecionei D5:D10. Excel escreveu por isso na última célula #N/D (deve ser alemão: *nicht da* –não está-, ou trata-se, porventura, de latim: *non datur*?).

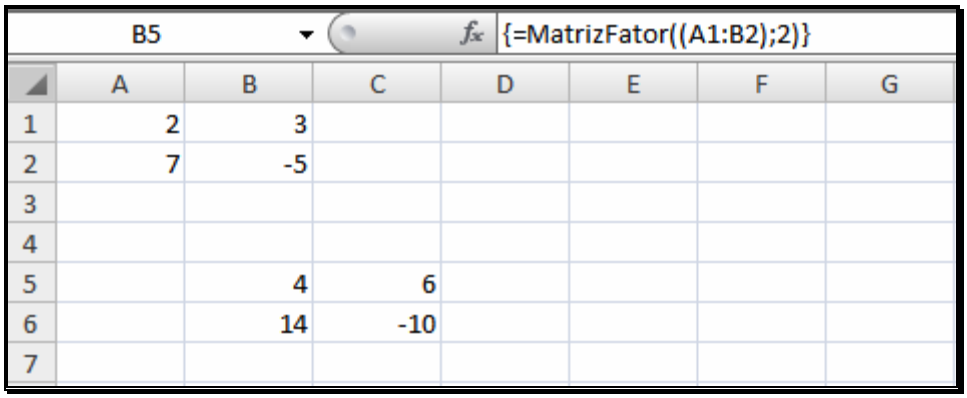
Como podemos *copiar* uma matriz de três linhas e três colunas?



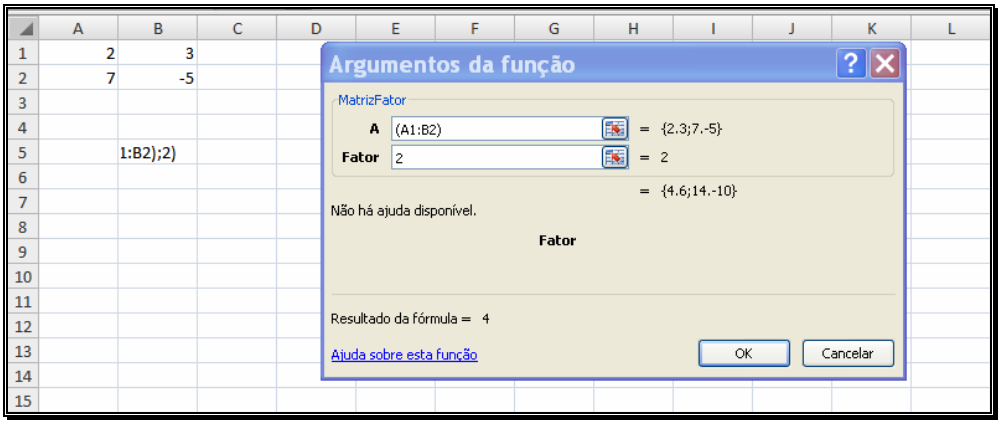
É só selecionar o intervalo, aqui G5:I7, e introduzir a seguinte constante de matriz: ={1.2.3;4.5.6;7.8.9}, pressione Ctrl+Shift+Enter.

Exemplos:

Queremos **multiplicar** todos os elementos da matriz (A1:B2) pelo fator 2 e queremos a nova matriz nas células (B5:C6). Olhe a seguinte solução!



Com um click sobre f_x buscamos a função "MatrixFator" (*Definida pelo usuário*) Aparecerá a janela *Argumentos da função*, a qual enchemos como demonstra a figura. Importante: Não clicar sobre OK, deve-se pressionar Ctrl+Shift+Enter.

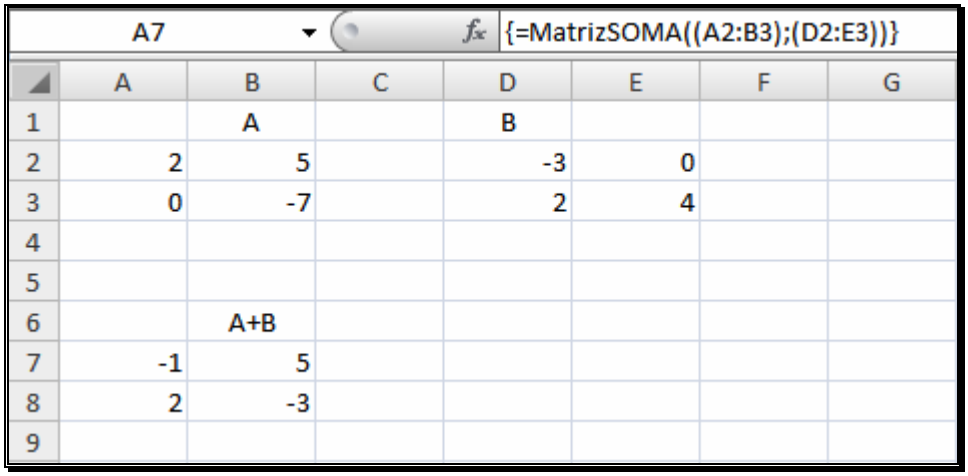


O código da função "MatrizFator", que faz esta maravilha, vem aqui:

```
Option Base 1
Function MatrizFator(A As Variant, fator As Double) As Variant
'Multiplicação de uma matriz por um fator
    Dim n As Integer, i As Integer, j As Integer
    Dim temp As Variant
    m = A.Rows.Count ' número de linhas
    ReDim temp(m, m)
    For i = 1 To m
    For j = 1 To m
    temp(i, j) = fator * A(i, j)
    'MsgBox "temp= " & temp(i, j)
    Next j
    Next i
    MatrizFator = temp

End Function
```

A matriz é chamada de **A**, e *Variant* é o tipo de dados que aceita também matrizes. (Quando você especificar argumentos para um procedimento, eles sempre terão como padrão o tipo *Variant*, que é o tipo de variável padrão no VBA. Uma variável não declarada é automaticamente do tipo *Variant*.) `temp` é uma variável temporária com a qual calculamos os produtos `fator*A(i,j)` da nova matriz. A variável integer `m` contém o número de linhas da matriz **A**. Na linha `ReDim temp(m,m)` dimensionamos `temp` com este número `m`. `Option Base 1` diz à VBA que os índices das matrizes começam todos com 1.

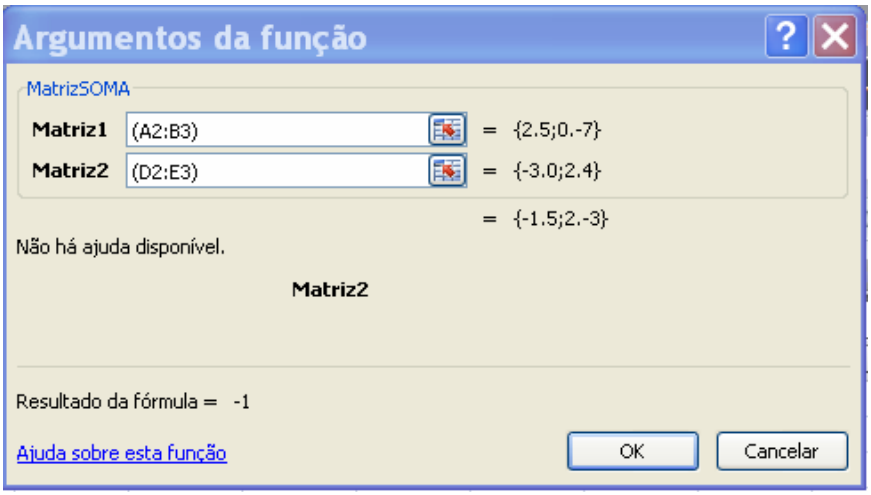
No próximo exemplo vamos **somar** duas matrizes A e B:



	A	B	C	D	E	F	G
1		A		B			
2	2	5		-3	0		
3	0	-7		2	4		
4							
5							
6		A+B					
7	-1	5					
8	2	-3					
9							

Primeiro enchemos as células com os elementos matriciais e depois selecionamos as células (A7:B8), para que elas recebam os elementos da matriz A+B. Podemos escrever a função "MatrizSOMA" na barra de fórmulas como mostra a figura, ou, o que eu prefiro, fazer um clique sobre f_x e escrever os dados na telinha dos *Argumentos da função*. Feche a tela com Ctrl+Shift+Enter, não com OK!

Uma declaração como "DIM R() As Double" define uma *matriz dinâmica* R de números reais que mais à frente deve ser fixada, p.ex. por `ReDIM R(n,m)`.



O código da função "MatrizSOMA" contém somente uma nova variável auxiliar `temp`.

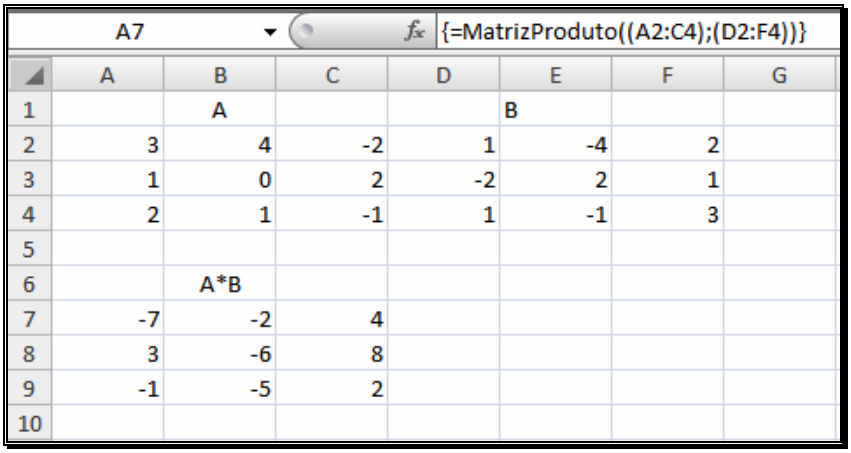
```
Option Base 1
Function MatrizSOMA(matriz1 As Variant, matriz2 As Variant) As Variant
'Soma de duas matrizes
    Dim m As Integer, i As Integer, j As Integer
    Dim temp As Variant
    m = matriz1.Rows.Count

    ReDim temp(m, m)
    For i = 1 To m
    For j = 1 To m
    temp(i, j) = matriz1(i, j) + matriz2(i, j)
    'MsgBox "temp= " & temp(i, j)
    Next j
    Next i
    MatrizSOMA = temp

End Function
```

Você sabe que o **produto de duas matrizes A** = $[a_{ij}]$ e **B** = $[b_{ij}]$ é calculado pela fórmula

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{ip}b_{pj} = \sum_{k=1}^{p} a_{ik}b_{kj} \quad (1)$$

onde **A** é uma matriz m x p (m linhas e p colunas) e B é de p x n (p linhas e n colunas) A matriz **C** = $[_{ij}]$ é m x n, ou seja, ela tem m linhas e n colunas.

Isso significa, que o produto de **A** por **B** está apenas definido quando o número de linhas de **B** é exatamente igual ao número de colunas de **A**. No exemplo seguinte, temos duas matrizes quadradas, ambas de 3 linhas e 3 colunas. **A** matriz produto **C** aparecerá no intervalo (A7:C9) –por que é aqui, onde queremos que fique o resultado. Por isso selecionamos as células (A7:C9), antes de pressionar as teclas Ctrl+Shift+Enter.



A7 | {=MatrizProduto((A2:C4);(D2:F4))}

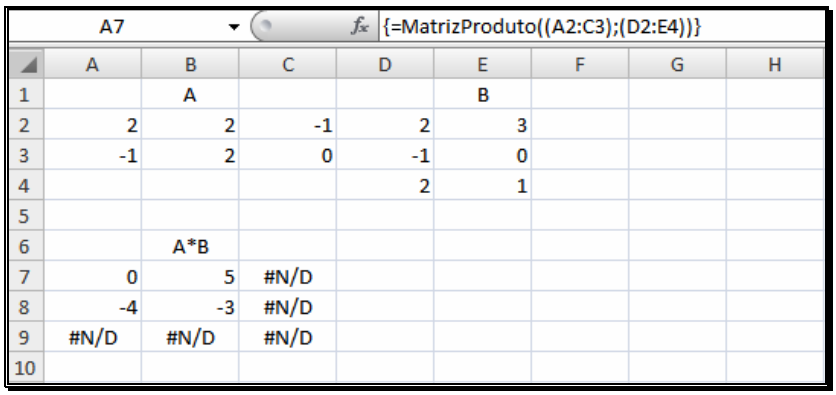
	A	B	C	D	E	F	G
1		A			B		
2	3	4	-2	1	-4	2	
3	1	0	2	-2	2	1	
4	2	1	-1	1	-1	3	
5							
6		A*B					
7	-7	-2	4				
8	3	-6	8				
9	-1	-5	2				
10							

O código foi ampliado pelo Loop `For k =1 To p` que executa a soma em (1) Note a introdução do contador, n, das colunas da matriz **B**. A matriz produto **A**·**B** é de m linhas e n colunas, m x n.

```
Option Base 1
Function MatrizProduto(A As Variant, B As Variant) As Variant
'Produto de duas matrizes
    Dim n As Integer, p As Integer
    Dim i As Integer, j As Integer, k As Integer, m As Integer
    Dim temp As Variant, prod As Variant
    m = A.Rows.Count
    n = B.Columns.Count
    p = B.Rows.Count
    'MsgBox "n= " & n & "m= " & m & "p= " & p

    ReDim temp(m, n)

    For i = 1 To m
    For j = 1 To n
     prod = 0
     For k = 1 To p
       prod = prod + A(i, k) * B(k, j)
     Next k
    temp(i, j) = prod
    Next j
    Next i
    MatrizProduto = temp

End Function
```

Seguem agora mais alguns exemplos sobre o **produto** de matrizes e vetores. (Um vetor é uma matriz de uma única coluna ou linha.)

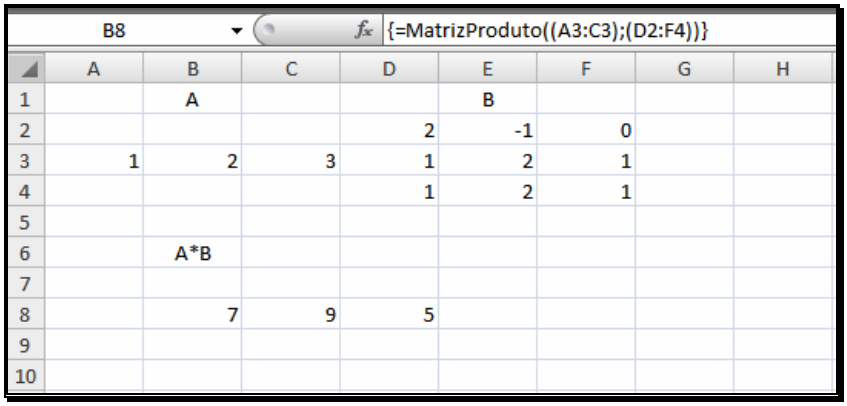
Aqui multiplicamos uma matriz **A** = ("2x3") por uma **B** = ("3x2"). A matriz produto **C** = **A**·**B** vai ter 2 linhas e 2 colunas. Eu havia selecionado 3 linhas e 3 colunas, e Excel encheu as células sem elementos com #N/D.



A7 | {=MatrizProduto((A2:C3);(D2:E4))}

	A	B	C	D	E	F	G	H
1		A			B			
2	2	2	-1	2	3			
3	-1	2	0	-1	0			
4				2	1			
5								
6		A*B						
7	0	5	#N/D					
8	-4	-3	#N/D					
9	#N/D	#N/D	#N/D					
10								

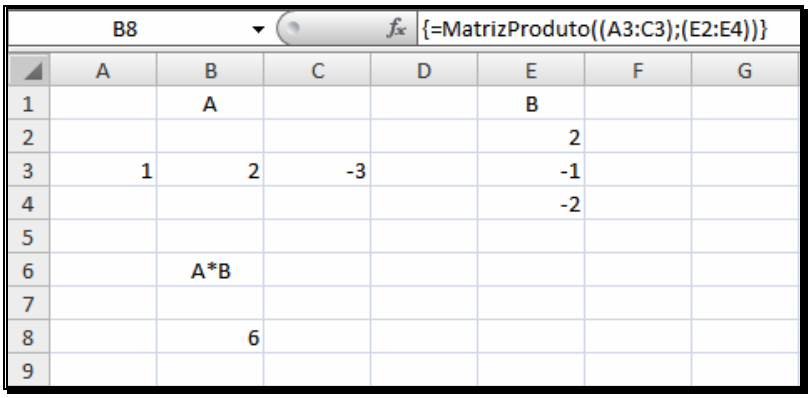
No exemplo a seguir, foi determinado de antemão o tamanho da matriz produto. Deveria ser 1x3, ou seja uma linha e três colunas. Por isso foram elegidas três células em linha horizontal: B8,C8,D8.



B8 | {=MatrizProduto((A3:C3);(D2:F4))}

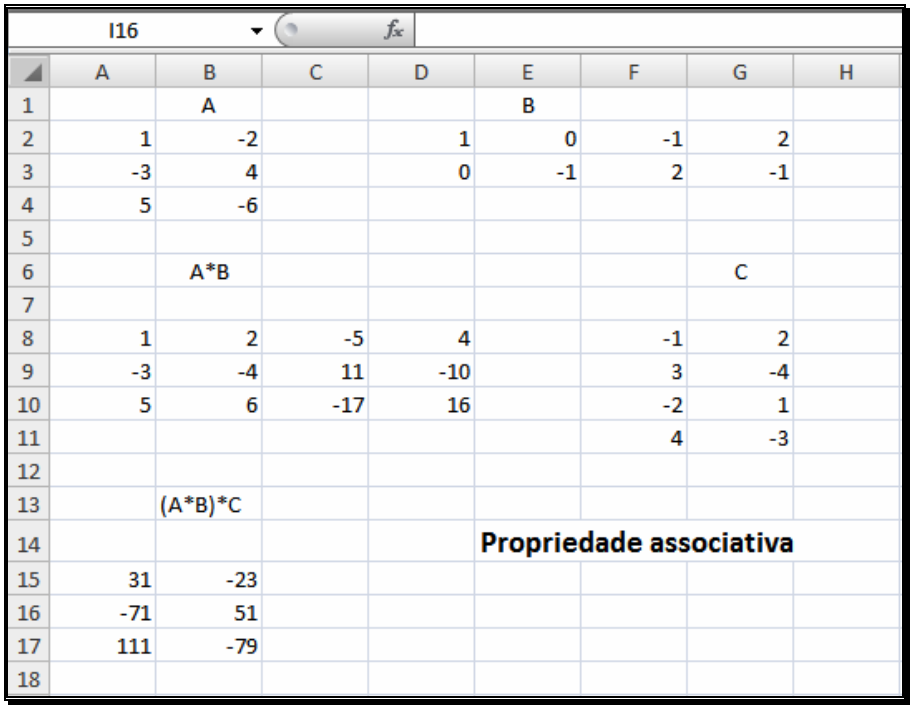
	A	B	C	D	E	F	G	H
1		A			B			
2				2	-1	0		
3	1	2	3	1	2	1		
4				1	2	1		
5								
6		A*B						
7								
8		7	9	5				
9								
10								

Para determinar o **Produto escalar** de dois vetores, multiplicamos um vetor horizontal com um vetor vertical, o que produz um mero número, no caso 6 (=Produto escalar dos vetores)



B8 | {=MatrizProduto((A3:C3);(E2:E4))}

	A	B	C	D	E	F	G
1		A			B		
2					2		
3	1	2	-3		-1		
4					-2		
5							
6		A*B					
7							
8		6					
9							



I16

	A	B	C	D	E	F	G	H
1		A			B			
2	1	-2		1	0	-1	2	
3	-3	4		0	-1	2	-1	
4	5	-6						
5								
6		A*B					C	
7								
8	1	2	-5	4		-1	2	
9	-3	-4	11	-10		3	-4	
10	5	6	-17	16		-2	1	
11						4	-3	
12								
13		(A*B)*C						
14					**Propriedade associativa**			
15	31	-23						
16	-71	51						
17	111	-79						
18								

Exemplo: Centro de Massa

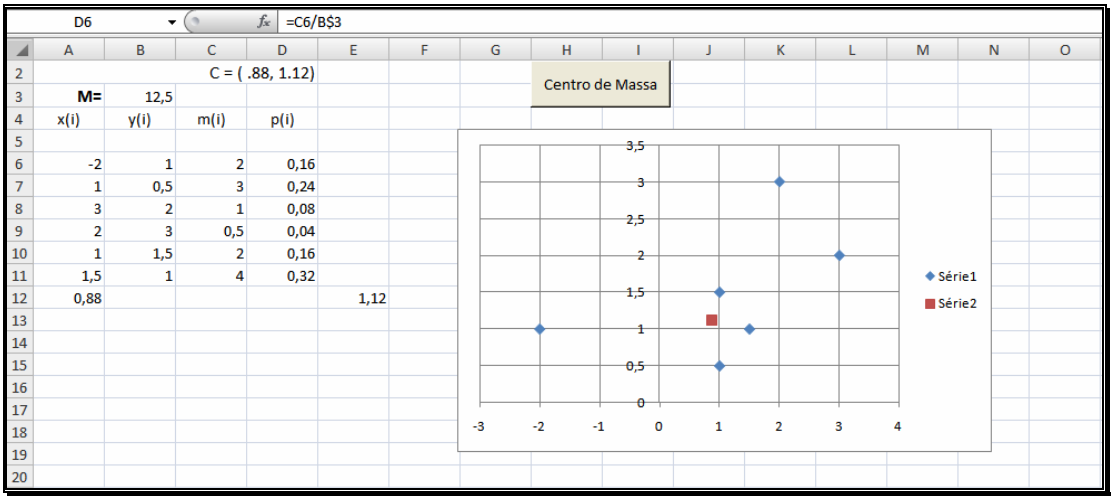
Vamos resumir o que aprendemos nas paginas anteriores com um exemplo sobre o *centro de massa* (de gravidade) de um sistema discreto de n partículas. A abscissa xc do centro de massa, C(xc,yc), obtém-se pela fórmula

$$x_c = \frac{\sum_{i=1}^{n} m_i x_i}{\sum_{i=1}^{n} m_i} \qquad (1)$$

No ponto C pode-se considerar concentrada toda a massa $M = \sum_{i=1}^{n} m_i$.

Podemos reescrever a primeira fórmula introduzindo os *pesos* $p_i = m_i/M$:

$$x_c = \sum_{i=1}^{n} p_i x_i \quad \text{e} \quad y_c = \sum_{i=1}^{n} p_i y_i \quad \text{com} \sum_{i=1}^{n} p_i = 1 \quad (2)$$



Na linha 12 temos as coordenadas do centro de massa xc = 0,88 e yc = 1,12. Para desenhar a Série1, elegemos A6:B12. O cursor fica no princípio em A6, depois pressione F8, F5: A12, OK, Shift F8. Com F5 saltamos ao começo da Série2 em E6. Depois F8, F5: E12, OK, Shift F8. *Inserir>* etc. A Série2 consta só do ponto C. O botão "Centro de Massa" pertence ao programa a seguir. Veja outra vez o capítulo *Gráficos 1* para recapitular o uso das teclas F8 F5.

Claro que não se precisa nenhuma macro para calcular C, será fácil fazê-lo na planilha mesma, somando os produtos x(i)*p(i) e y(i)*p(i). Mas, a macro "Pontos()" com o botão "Centro de Massa" pode ser usada para aplicações mais complicadas, por exemplo no caso das fórmulas de interpolação do seguinte capítulo. A macro baseia-se num programa do site www.cidse.itcr.ac.cr/cursos-linha/NUMERICO cujos autores se dedicam à produção de cursos grátis na internet para estudiosos das ciências matemáticas e outras.

De valor especial é o enfoque pedagógico que se nota ao estudar o código. Ele mostra, outra vez, como se lê dados da planilha usando o objeto Range. A instrução Set configura R para apontar para as células selecionadas, veja nosso capítulo sobre o triângulo de Pascal. A linha com `IF n>1 And m = 4 Then` não contém código, e VBA o aceita assim, sem protesta, e segue com a próxima linha. (Vamos pensar que o Range selecionado é uma Matriz de dimensão n x m. A posição (i,j) da matriz corresponde à linha i e à coluna j. Para contar linhas e colunas, VBA precisa saber onde começa e onde termina o Range.)

A linha `Cells(2,1) = "C = (" + Str(xc) + ", " + Str(yc)+ ")"` coloca as coordenadas do centro de massa na célula A1 (que foi ampliada) como String (uma String é uma série de caracteres), pois a função Str converte um número em uma string. Isso não é sempre muito desejado, pois o resultado é exibido com todas as casas decimais possíveis.

Se mudar a massa m(6) de 4 por 3 terá o resultado C = (.826086956521739, 1.1304347826087). O que acha? Seguramente poderíamos escrever, como fizemos na sub-rotina "SegGrau()",

```
Cells(2, 1) = "xc=": Cells(2, 2) = xc
Cells(2, 3) = "yc=": Cells(2, 4) = xc
```

em vez de `Cells(2,1) = "C = ("+ Str(xc)+ ", "+ Str(yc)+ ")"`. No caso dos números complexos vimos o problema inverso, quando utilizamos a função Val para mudar uma string para um número real; veja também nossa sub-rotina "complexos" no Capítulo 7.

(Geral) — Pontos

```
Private Sub Pontos()

    Dim R As Range '= objeto (aqui o conjunto de células selecionadas)
    Dim n As Integer, m As Integer, i As Integer
    Dim x() As Double   ' matriz dinámica
    Dim y() As Double
    Dim p() As Double
    Dim Somapesos As Double, cx As Double, cy As Double

    Set R = Selection
    n = R.Rows.Count     ' Número de linhas
    m = R.Columns.Count  ' Número de colunas
    MsgBox "n= " & n & ", m= " & m

    If n > 1 And m = 4 Then 'tudo em orden, pois foram selecionadas as células
            Else
            MsgBox ("Deve selecionar os dados!")
         Exit Sub
    End If

    ReDim x(n)
    ReDim y(n)
    ReDim p(n)

    Somapesos = 0
    xc = 0
    yc = 0
        If n > 1 Then

        For i = 1 To n
            x(i) = R(i, 1) 'leemos os dados das suas células
            y(i) = R(i, 2)
            p(i) = R(i, 4)
                           'cálculo do centro de massa
            Somapesos = Somapesos + p(i)
            xc = xc + x(i) * p(i)
            yc = yc + y(i) * p(i)
        Next i
        If Abs(Somapesos - 1) < 0.001 Then  'a soma dos pesos deve ser 1
            xc = xc / Somapesos
            yc = yc / Somapesos
                               'escreve C na célula  A2
```

```
                                'escreve C na célula  A2
            Cells(2, 1) = "C = (" + Str(xc) + "," + Str(yc) + ")"
            Cells(1, 1) = ""  'limpa A1 de mensagens prévios
        Else
                                'mensagem de erro
            Cells(1, 1) = "Erro, os pesos  soman " + Str(Somapesos)
            Cells(2, 1) = ""  'limpa A2 de valores prévios
        End If
    Else
        Cells(2, 1) = ""
                                'janela de advertência: selecione dados!
        MsgBox ("Deve selecionar os dados")
        Exit Sub                ' termina a sub-rotina se não tem dados
    End If
End Sub
```

E muito importante familiarizar-se com o desenho de mensagens de erro, utilizando, por exemplo, as mensagens contidas no programa anterior.

Sistemas de equações lineares

Matrizes e vetores são igualmente usados quando se trata de resolver um sistema de m equações com n incógnitas (simplesmente: *sistema linear*). Um caso especial é ele onde m = n. Isto é, a situação em que o número de equações é igual ao número de incógnitas e a matriz dos coeficientes é quadrada. Na verdade, há inúmeras aplicações em que m ≠ n.

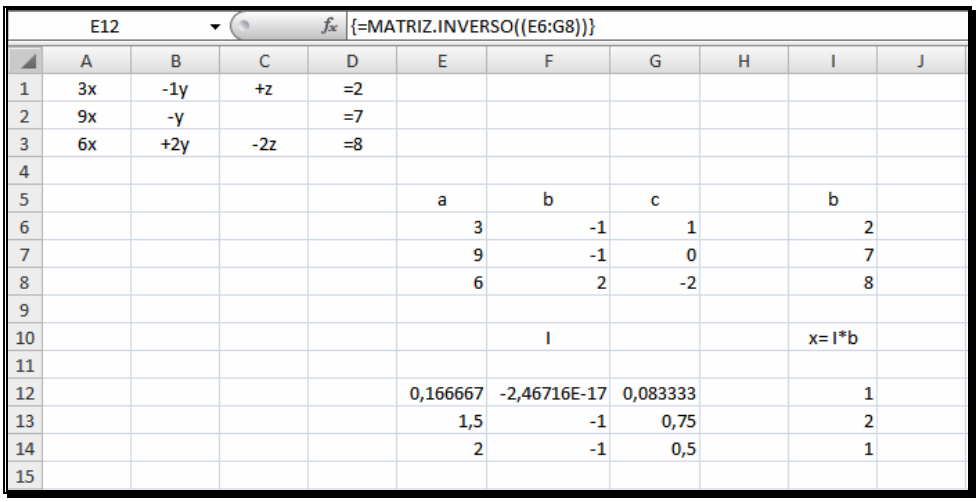
Seja $\begin{vmatrix} 3x & -y & +z & =2 \\ 9x & -y & & =7 \\ 6x & +2y & -2z & =8 \end{vmatrix}$ um sistema linear com A = $\begin{pmatrix} 3 & -1 & 1 \\ 9 & -1 & 0 \\ 6 & 2 & -2 \end{pmatrix}$

Numa forma concisa escreve-se isso como $\mathbf{A \cdot x = b}$ com a solução $\mathbf{x = A^{-1}b}$. **x** e **b** são vetores e $\mathbf{A^{-1}}$ é a matriz inversa de **A**. No velho BASIC havia um função "Mat" e INV com as quais foi fácil resolver o sistema:

```
5     OPTION BASE 1
10    DIM A[3,3], I[3,3],X[3,1],B[3,1]
20    MAT   READ A,B
30    MAT   I=INV(A)
40    MAT   X=I*B
50    MAT   PRINT  X
100   DATA  3,-1,1
110   DATA  9,-1,0
120   DATA  6,2,-2
130   DATA  2,7,8
200   END
```

A solução é o vetor 1.
2.
1.

Como fazemos isso no Excel? Veja:



E12 {=MATRIZ.INVERSO((E6:G8))}

	A	B	C	D	E	F	G	H	I	J
1	3x	-1y	+z	=2						
2	9x	-y		=7						
3	6x	+2y	-2z	=8						
4										
5					a	b	c		b	
6					3	-1	1		2	
7					9	-1	0		7	
8					6	2	-2		8	
9										
10						I			x= I*b	
11										
12					0,166667	-2,46716E-17	0,083333		1	
13					1,5	-1	0,75		2	
14					2	-1	0,5		1	
15										

Com "MATRIZ.INVERSO" calculamos a inversa **I** nas células (E12:G14), ou no intervalo que você escolher. Pressione Ctrl+Shift+Enter.

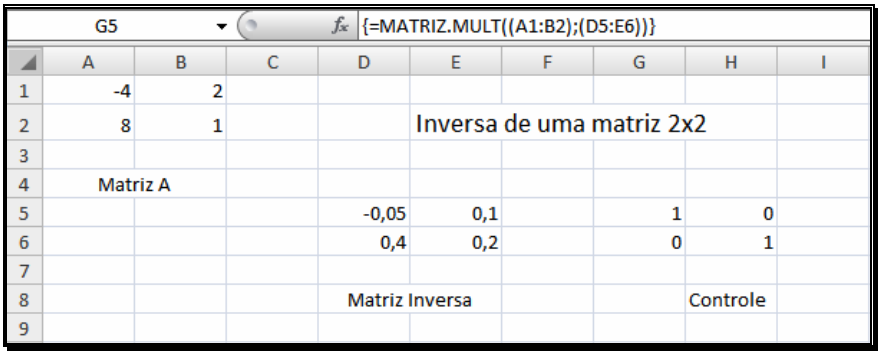
Escolhi (I12:I14) para o vetor solução **x**, que é o produto de **I** por **b**. Pronto. A função MATRIZ.INVERSO determina a inversa com uma exatidão de 15 casas decimais.

No caso de uma matriz 2x2 existem fórmulas simples para o cálculo dos coeficientes da matriz inversa. O seguinte procedimento contém essas fórmulas. Trata-se de uma variação da função "Segundograu" do Capítulo 7. Escrevemos em 4 células os elementos da matriz **A**, por exemplo em (A1:B2). Depois selecionamos 4 células para a **matriz inversa** e, em seguida, clicamos sobre f_x para chamar nossa função "inversa2". Na janela dos argumentos da função escrevemos ao lado de "MatrizA" o intervalo (A1:B2). Não clicamos sobre OK, mas sim pressionamos as teclas Ctrl+Shift+Enter.

Temos um método simples para o controle do resultado, pois o produto **I·A** deve sempre produzir a **matriz identidade** de ordem 2 (no caso de uma matriz quadrada de ordem n resulta uma matriz identidade de ordem n). Usando a função "MATRIZ.MULT" -ou nossa função "MatrizProduto"-, este controle será rapidamente feito.

```
Function inversa2(matrizA As Variant) As Variant
    Dim a As Double, b As Double, c As Double, d As Double
    Dim e As Double
    Dim inv(1 To 2, 1 To 2) As Variant
    'escrever a,b,c,d num bloco de 4 células
      a = matrizA(1)
      b = matrizA(2)
      c = matrizA(3)
      d = matrizA(4)
      e = a * d - b * c

      inv(1, 1) = d / e
      inv(1, 2) = -b / e
      inv(2, 1) = -c / e
      inv(2, 2) = a / e

      inversa2 = inv

End Function
```

Exemplo:



	A	B	C	D	E	F	G	H	I
1	-4	2							
2	8	1							
3									
4									
5				-0,05	0,1		1	0	
6				0,4	0,2		0	1	
7									
8									
9									

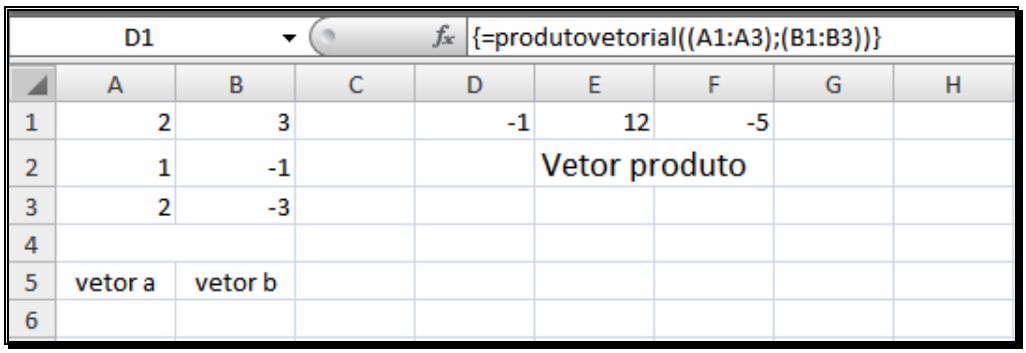
A fórmula em D5 diz =inversa2((A1:B2))

Produto vetorial em R^3

Aproveitamos a oportunidade para transformar o program anterior num programa para calcular o produto vetorial:

```
Function produtovetorial(vetorA As Variant, vetorB As Variant) As Variant
    Dim a1 As Double, a2 As Double, a3 As Double
    Dim b1 As Double, b2 As Double, b3 As Double
    Dim c1 As Double, c2 As Double, c3 As Double
    Dim prod(1 To 3) As Variant

      a1 = vetorA(1): b1 = vetorB(1)
      a2 = vetorA(2): b2 = vetorB(2)
      a3 = vetorA(3): b3 = vetorB(3)

      'c1 = a2 * b3 - a3 * b2: c2 = a3 * b1 - a1 * b3: c3 = a1 * b2 - a2 * b1

      prod(1) = a2 * b3 - a3 * b2
      prod(2) = a3 * b1 - a1 * b3
      prod(3) = a1 * b2 - a2 * b1


      produtovetorial = prod

End Function
```

A figura mostra os vetores coluna **a** = (2,1,2) e **b**= (3,-1,-3) e o produto vetorial **c** = (-1,12,-5), ou seja **c** = -**i** +12**j** -5**k**



Se não queremos introduzir explicitamente os coeficientes a1,...,b3, temos a opção de ler os conteúdos das células com a propriedade Cells(i,j), como já fizemos no programa "SegGrau" no Capítulo 7. O programa a seguir utiliza também o tipo de dada genérico "Object", que recebe todo tipo de objetos definidos. Este tipo genérico é muito menos eficiente do que uma declaração específica de objeto, tal como Dim a As Workbook, Dim b As Chart etc. Mas, é necessário haver visto uma vez este tipo de dado. (Também houvéssemos podido usar As Variant em vez de As Object.)

```
Option Base 1
Function prodvet(A As Object, B As Object) As Variant
    Dim temp(3, 1)
      temp(1, 1) = A.Cells(2, 1).Value * B.Cells(3, 1).Value - _
                   A.Cells(3, 1).Value * B.Cells(2, 1).Value
      temp(2, 1) = A.Cells(3, 1).Value * B.Cells(1, 1).Value - _
                   A.Cells(1, 1).Value * B.Cells(3, 1).Value
      temp(3, 1) = A.Cells(1, 1).Value * B.Cells(2, 1).Value - _
                   A.Cells(2, 1).Value * B.Cells(1, 1).Value

      prodvet = temp
End Function
```

O produto vetorial aparecerá como vetor coluna.

O método de Gauss

Embora o Excel ofereça, como vimos, excelentes métodos para resolver sistemas de equações algébricas, parece-nos indispensável pôr o clássico método de Gauss num ambiente VBA. Para o ensino nas aulas de Álgebra Linear ou nos cursos de Cálculo Numérico, sempre foi fundamental uma discussão do método de Gauss.

Este método consiste, essencialmente, em transformar por etapas sucessivas a matriz original do sistema numa matriz triangular superior. Após obtenção da matriz transformada (matriz triangular superior), o sistema pode ser resolvido por substituição "ascendente" ("retroativa"). Para melhor explicar este algoritmo de eliminação, resolveremos o seguinte sistema

$$2x_1 + 3x_2 - x_3 = 5 \quad (1)$$
$$4x_1 + 4x_2 - 3x_3 = 3 \quad (2)$$
$$2x_1 - 3x_2 + x_3 = -1 \quad (3)$$

Os passos a executar são:

1. Formar a matriz ampliada do sistema

$$B_0 = \left(\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ 2 & -3 & 1 & -1 \end{array}\right)$$

2. Consideraremos como pivô a primeira linha, L_1^0 = (2, 3, -1 , 5), para o processo de eliminação.

3. Nossa meta é zerar todos os coeficientes abaixo da diagonal principal; para isso vamos

4. calcular o fator $m_{21}^0 = -(a_{21}^0 / a_{11}^0) = -4/2 = -2$ (passo 1)

5. Vamos somar à 2ª equação a 1ª, multiplicada pelo coeficiente $m_{21}^0 = -2$, e colocar o resultado na 2ª linha, isto não altera a solução do sistema. Resulta uma nova segunda linha: L_2^1 = (0, -2, -1, -7) com $a_{21}^1 = 0$. A primeira linha não sofre mudança.

6. Falta, agora anular o elemento $a_{31}^0 = 2$ em B_0. Para isso, repetimos o procedimento (passo 2): $m_{31}^0 = -(a_{31}^0/a_{11}^0) = -2/2 = -1$. Somamos, agora, à 3ª equação a 1ª, multiplicada pelo coeficiente $m_{31}^0 = -1$. A nova terceira linha será L_3^1 = (0, **-6**, 2, -6). E a matriz B_1 tem a forma

$$B_1 = \left(\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & -6 & 2 & -6 \end{array}\right)$$

Falta anular a_{32}^{1} = -**6**.

6. Determinamos $m_{32}^{1} = -(a_{32}^{1}/a_{22}^{1}) = -(-6/-2) = -3$ (passo 3)

7. Chegamos, assim, à nova matriz ampliada B_1

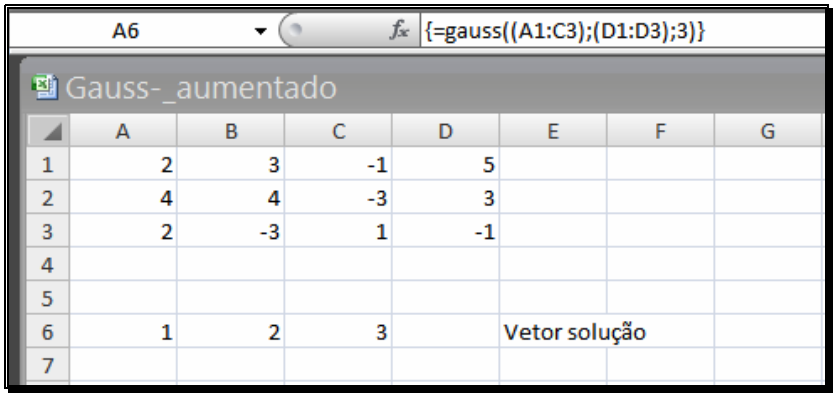
$$B_2 = \left(\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 0 & 5 & 15 \end{array}\right)$$

Vemos que o sistema original foi reduzido a um sistema triangular equivalente ao sistema original, e cuja matriz é uma matriz em escada.

$$\begin{aligned} 2x_1 + 3x_2 - x_3 &= 5 \quad (4) \\ -2x_2 - x_3 &= -7 \quad (5) \\ 5x_3 &= 15 \quad (6) \end{aligned}$$

Com a obtenção de uma matriz em escada termina a parte descendente do método de eliminação de Gauss. Neste momento, verifica-se se o sistema obtido é possível, isto é , se não há equações com o primeiro membro nulo e o segundo não nulo. Se o sistema for possível, resolve-se ele de "baixo para cima" (parte "ascendente" ou "retroativa" do algoritmo). Obteremos em nosso caso o seguinte vetor solução: x_3= 3; x_2 = 2; x_1 = 1.

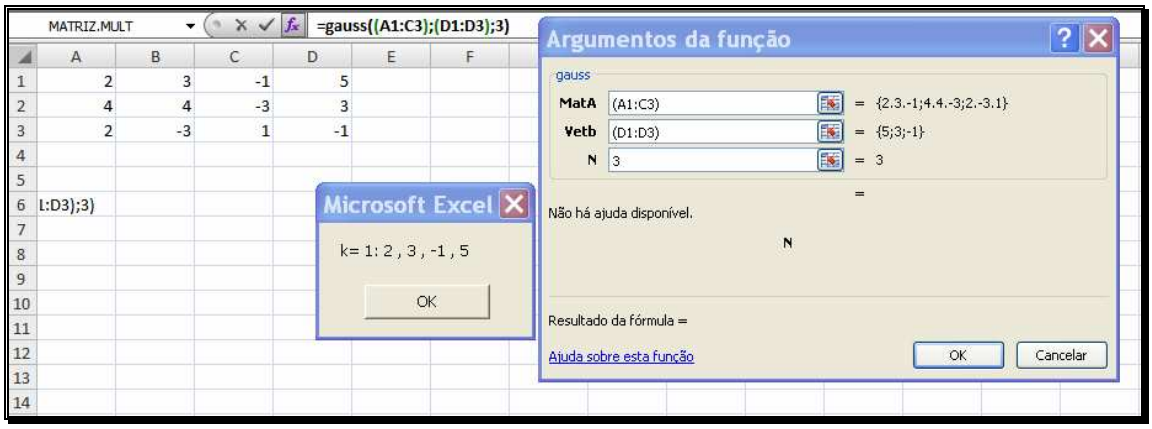
Temos aqui o nosso exemplo produzido com a função "gauss" que se orienta estritamente no processo descrito.



	A	B	C	D	E	F	G
1	2	3	-1	5			
2	4	4	-3	3			
3	2	-3	1	-1			
4							
5							
6	1	2	3		Vetor solução		
7							

```
Function gauss(matA, vetb, n)
Dim B(100, 100), x()
Dim i As Integer, j As Integer, k As Integer, p As Integer
Dim m As Integer, q As Integer

'Matriz aumentada
For p = 1 To n Step 1
  For q = 1 To n Step 1
     B(p, q) = matA(p, q)
  Next q
     B(p, n + 1) = vetb(p)
Next p

'produção da forma triangular
For k = 1 To n - 1 Step 1
    For i = k + 1 To n + 1 Step 1
        m = B(i, k) / B(k, k)
        For j = k To n + 1 Step 1
            B(i, j) = B(i, j) - m * B(k, j)
        Next j
     Next i
 Next k
        For k = 1 To n

        MsgBox "k= " & k & ": " & B(k, 1) & " , " & B(k, 2) & " , " & B(k, 3) & " , " & B(k, 4)

        Next k
'Substituição ascendente
ReDim x(1 To n)
x(n) = B(n, n + 1) / B(n, n)
For i = n - 1 To 1 Step -1
    Sum = 0
    For j = i + 1 To n Step 1
        Sum = Sum + B(i, j) * x(j)
    Next j
    x(i) = (B(i, n + 1) - Sum) / B(i, i)
Next i
gauss = x
'O vetor solução x() é um vetor linha 1xn
    MsgBox "fim"
End Function
```

No principio serão passados os parâmetros para a função "gauss". A função lê os valores das matrizes matA e vetb da planilha:



e ela cria a matriz aumentada B. Com a MsgBox podemos observar a formação das novas linhas para cada valor de k (isso é só uma ajuda pedagógica que se pode transformar num comentário).