

# Matrizes

Tipo de Dados Homogêneo

# Matrizes

- Matrizes são variáveis que contêm uma quantidade potencialmente grande de valores.
- É no tratamento de matrizes que o Scilab mostra grande superioridade sobre linguagens como C, Fortran ou Java.

```
-->A = [1 2 3; 4 5 6]
```

A =

1.	2.	3.
4.	5.	6.

Este comando cria uma matriz 2 x 3, com os valores de cada linha separados por “;”

# Matrizes

- Todas as variáveis Scilab são, a princípio, matrizes.

```
-->x = 7  
x =  
7.
```

```
-->[l,c] = size(x)  
c =  
1.  
l =  
1.
```

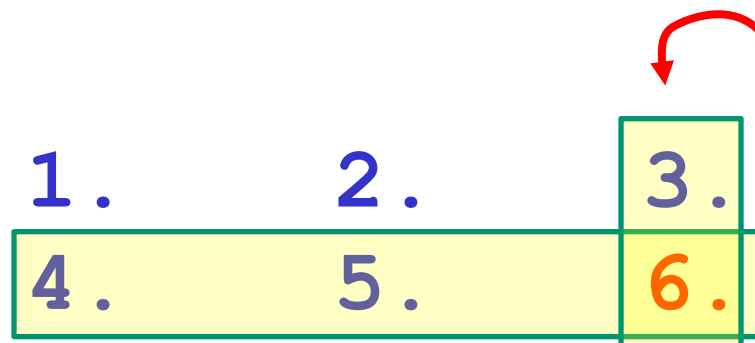
A função "size" retorna o número de linhas e o número de colunas de uma matriz

"x" é uma matriz de uma linha e uma coluna

# Obtendo o valor de um elemento da matriz

-->A = [1 2 3; 4 5 6]

A =



The diagram shows a 2x3 matrix A. The first row contains the values 1., 2., and 3., and the second row contains 4., 5., and 6.. The entire matrix is highlighted in yellow. Red arrows indicate the indexing process: one arrow points from the first row to the second row, and another points from the third column to the value 6. in the second row.

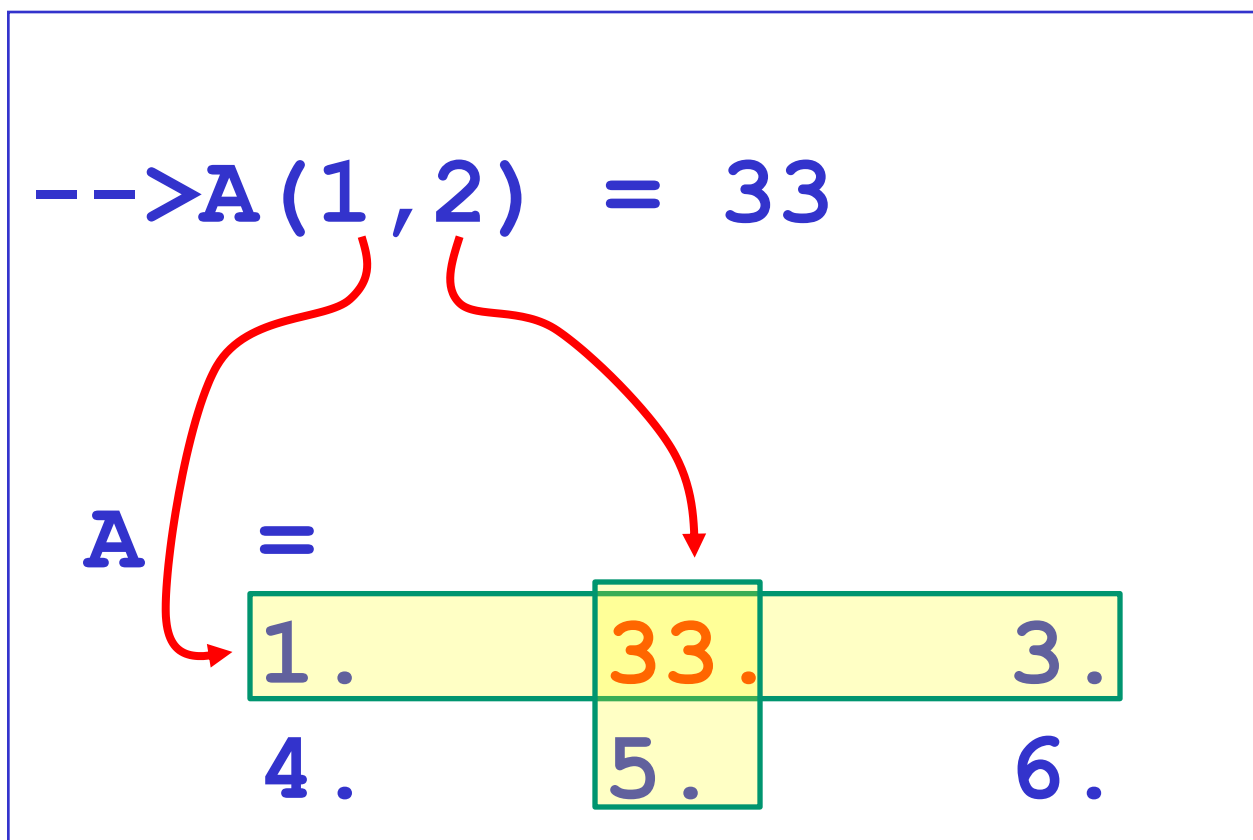
1.	2.	3.
4.	5.	6.

-->e = A(2,3)

e =

6.

# Atribuindo um valor a um elemento da matriz



# Vetores

- Vetores são matrizes de uma única linha ou de uma única coluna.

```
-->v = [10 20 30]
v
    10.    20.    30.
-->u = [10; 20; 30]
u
    10.
    20.
    30.
```

Para acessar o valor 20,  
usa-se "v(1,2)" ou  
simplesmente "v(2)"

Para acessar o valor 20,  
usa-se "v(2,1)" ou  
simplesmente "v(2)"

## Expansão de uma matriz

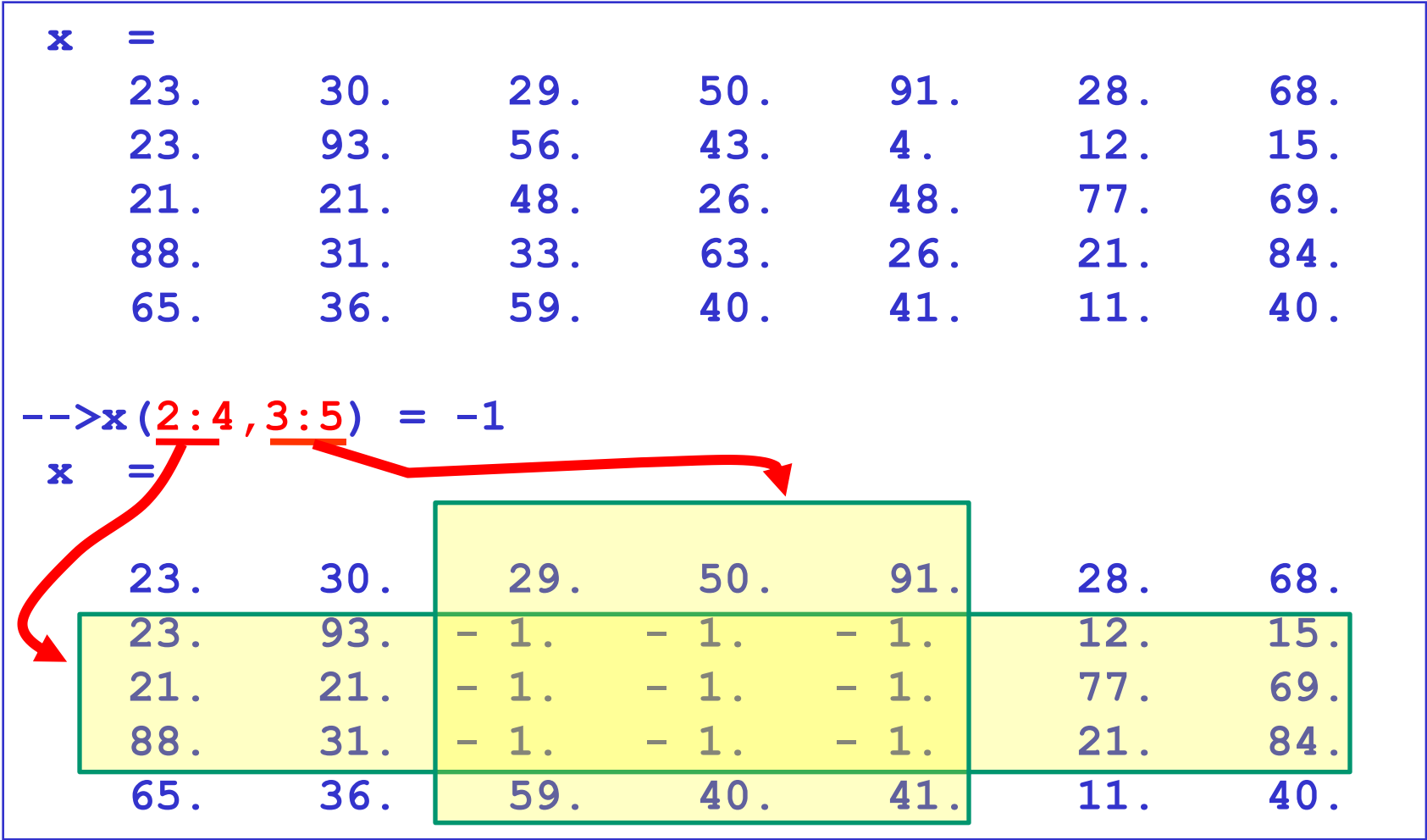
- Uma matriz "cresce" quando se atribui valores a elementos ainda não existentes. No caso, as lacunas geradas são completadas com zeros.

```
-->x = 7; // matriz 1x1
-->x(2,3) = 13
```

**x** =

7.	0.	0.
0.	0.	13.

# Atribuindo um valor a uma parte de uma matriz





# Atribuindo valores a uma parte de uma matriz

**x** =

40.	58.	38.	73.	53.	4.	58.
87.	68.	92.	26.	11.	67.	48.
11.	89.	94.	49.	22.	20.	22.
19.	50.	34.	26.	62.	39.	84.
56.	34.	37.	52.	76.	83.	12.

-->**x** (3:4, 4:5) = [-1 -2; -3 -4]

**x** =

40.	58.	38.	73.	53.	4.	58.
87.	68.	92.	26.	11.	67.	48.
11.	89.	94.	- 1.	- 2.	20.	22.
19.	50.	34.	- 3.	- 4.	39.	84.
56.	34.	37.	52.	76.	83.	12.

# Obtendo os valores de uma linha de uma matriz

x	=	40.	58.	38.	73.	53.	4.	58.
		87.	68.	92.	26.	11.	67.	48.
		11.	89.	94.	49.	22.	20.	22.
		19.	50.	34.	26.	62.	39.	84.
		56.	34.	37.	52.	76.	83.	12.

--> a = x(2, :)

a	=	87.	68.	92.	26.	11.	67.	48.
---	---	-----	-----	-----	-----	-----	-----	-----

":" designa todos os elementos de uma dimensão (no caso, coluna)

# Obtendo os valores de colunas de uma matriz

```
x =
```

91.	28.	68.	40.	58.	38.	73.
4.	12.	15.	87.	68.	92.	26.
48.	77.	69.	11.	89.	94.	49.
26.	21.	84.	19.	50.	34.	26.
41.	11.	40.	56.	34.	37.	52.

```
-->b = x(:,3:5)
```

```
b =
```

68.	40.	58.
15.	87.	68.
69.	11.	89.
84.	19.	50.
40.	56.	34.

# Aritmética matricial

---

- Como todas as variáveis Scilab são matrizes, as operações aritméticas usuais ( $+$ ,  $-$ ,  $*$ ,  $/$ ,  $^$ ) são entendidas pelo Scilab como operações matriciais.
  - Assim,  $a*b$  designa o produto matricial da matriz  $a$  pela matriz  $b$ .
- Operações escalares usam os mesmos símbolos aritméticos, porém precedidos por um "." (ponto) como, por exemplo,  $.*$  e  $.^$ .

# Adição e subtração de matrizes

- Matrizes de mesmas dimensões podem ser somadas ou subtraídas.

```
-->x = [1 2 3; 4 5 6];  
-->y = [10 20 30; 40 50 60];  
-->x + y  
ans =  
    11.    22.    33.  
    44.    55.    66.  
-->x - y  
ans =  
    - 9.    - 18.    - 27.  
    - 36.    - 45.    - 54.
```

# Produto matricial

```
-->x = [1 2 3; 4 5 6]
```

```
x =
```

```
1.    2.    3.
```

```
4.    5.    6.
```

```
-->y = [10 20; 30 40; 50 60]
```

```
y =
```

```
10.    20.
```

```
30.    40.
```

```
50.    60.
```

```
-->x * y
```

```
ans =
```

```
220.    280.
```

```
490.    640.
```

O n° de colunas da 1ª matriz deve ser igual ao n° de linhas da 2ª matriz

$$220 = 1 \times 10 + 2 \times 30 + 3 \times 50$$

# Produto elemento a elemento de matrizes

```
-->x = [1 2; 3 4];
```

```
-->y = [10 20; 30 40];
```

```
-->x * y
```

```
ans =
```

```
70.    100.
```

```
150.    220.
```

Produto  
matricial

```
-->x .* y
```

```
ans =
```

```
10.    40.
```

```
90.    160.
```

Produto elemento  
a elemento

# Multiplicação de matriz por escalar

---

```
-->x = [1 2 3; 4 5 6];
```

```
-->y = 2*x
```

```
y =
```

```
2.
```

```
4.
```

```
6.
```

```
8.
```

```
10.
```

```
12.
```



# Exponenciação em matrizes

```
-->x = [1 2; 3 4];
```

```
-->x^2
```

```
ans =
```

```
7.    10.
```

```
15.   22.
```

```
-->x.^2
```

```
ans =
```

```
1.    4.
```

```
9.   16.
```

Produto matricial

$x * x$

Exponenciação  
elemento a elemento

## Exercício

---

- Usando o comando **for**, faça um programa Scilab que forneça a transposta da matriz

$A = [1 \ 2 \ 3 \ 4; 5 \ 6 \ 7 \ 8; 9 \ 10 \ 11 \ 12].$

# Matriz transposta

$$\mathbf{A} = \begin{bmatrix} 1. & 2. & 3. \\ 4. & 5. & 6. \\ 7. & 33. & 9. \end{bmatrix}$$

$$\longrightarrow \mathbf{B} = \mathbf{A}'$$

$$\mathbf{B} = \begin{bmatrix} 1. & 4. & 7. \\ 2. & 5. & 33. \\ 3. & 6. & 9. \end{bmatrix}$$

$\mathbf{A}'$  é a transposta da matriz  $\mathbf{A}$

# Matriz inversa

**A** =

4.	7.	6.
2.	2.	1.
1.	1.	6.

--> **IA** = **inv**(A)

**IA** =

- 0.33333333	1.0909091	0.1515152
0.33333333	- 0.5454545	- 0.2424242
0.	- 0.0909091	0.1818182

A função "inv" retorna a matriz inversa de uma matriz

# Matriz inversa

```
-->A * IA
```

```
ans =
```

1.	0.	- 4.441D-16
1.110D-16	1.	- 1.110D-16
5.551D-17	0.	1.

```
-->IA * A
```

```
ans =
```

1.	8.327D-17	0.
0.	1.	0.
0.	0.	1.

Erro de  
aproximação

## Programa: sistemas de equações lineares

---

- Um sistema de equações lineares  $\mathbf{ax} = \mathbf{b}$  pode ser resolvido pela inversa de uma matriz.
  - Multiplicando os dois lados do sistema por  $\mathbf{a}^{-1}$ , tem-se:

$$\mathbf{a}^{-1}\mathbf{ax} = \mathbf{x} = \mathbf{a}^{-1}\mathbf{b}$$

- Resolva um sistema de equações lineares, por meio do Scilab, considerando:

$$\mathbf{a} = \begin{bmatrix} -2 & -1 & 3 \\ 2 & 1 & 1 \\ -4 & 1 & 3 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix}$$

# Programa: sistemas de equações lineares

---

```
-->a = [-2 -1 3; 2 1 1;-4 1 3];  
-->b = [4 0 1]';  
-->x = inv(a)*b  
  
x =  
    0.1666667  
   -1.3333333  
    1.
```

# Programa: sistemas de equações lineares

- A precisão do resultado calculado pode ser avaliada calculando  $\mathbf{ax} - \mathbf{b}$ , que deve ser "zero".

```
-->residuo = a*x - b  
residuo =  
    0.  
    1.110D-16  
    0.
```

Erro de  
aproximação



## Construção de vetores regulares

- Vetores com valores regularmente espaçados podem ser construídos de forma similar à utilizada no comando **for**

```
-->x = 10:13
```

```
x =
```

```
10.    11.    12.    13.
```

```
-->x = 12:-0.5:10
```

```
x =
```

```
12.    11.5    11.    10.5    10.
```

# Função linspace

- É utilizada para criar um vetor regular especificando seus limites e o número de pontos desejados.

Limite  
inferior

Limite  
superior

Número  
de pontos

```
-->x = linspace(0,10,3)
```

```
x =  
    0.    5.   10.
```

```
-->x = linspace(0,10,6)
```

```
x =  
    0.    2.    4.    6.    8.   10.
```

## Funções zeros e ones

- São utilizadas para criar matrizes com apenas elementos zeros e uns respectivamente. Para tanto, deve-se passar a dimensão desejada da matriz.

```
-->x = zeros (2, 3)
```

```
x =
```

```
0.    0.    0.
```

```
0.    0.    0.
```

```
-->y = ones (2, 3)
```

```
y =
```

```
1.    1.    1.
```

```
1.    1.    1.
```

## Função eye

- É utilizada para criar uma matriz identidade. Para tanto, deve-se passar a dimensão desejada da matriz.

```
-->I = eye(4, 4)
```

```
I =
```

1.	0.	0.	0.
0.	1.	0.	0.
0.	0.	1.	0.
0.	0.	0.	1.

## Função rand

- É utilizada para criar uma matriz com elementos aleatórios. Para tanto, deve-se passar a dimensão desejada da matriz.
  - Gera números aleatórios entre 0 e 1.
  - A cada chamada, gera novos números.

```
-->m = rand(2,3)
```

```
m =
```

```
    0.2113249    0.0002211    0.6653811
```

```
    0.7560439    0.3303271    0.6283918
```

```
-->n = rand(2,3)
```

```
n =
```

```
    0.8497452    0.8782165    0.5608486
```

```
    0.6857310    0.0683740    0.6623569
```

# Função rand

Pega a parte inteira  
de um número

Fator de escala

```
-->m = int(rand(2,3)*100)
```

```
m =
```

```
21.
```

```
0.
```

```
66.
```

```
75.
```

```
33.
```

```
62.
```

# Construindo matrizes a partir de matrizes

---

```
-->x = [1 2; 3 4];
```

```
-->y = [10 20; 30 40];
```

```
-->z = [x y]
```

```
z =
```

1.	2.	10.	20.
3.	4.	30.	40.

```
-->z = [x ; y]
```

```
z =
```

1.	2.
3.	4.
10.	20.
30.	40.

# Funções Scilab são matriciais

- Se uma determinada função for ativada com um argumento matricial, seu resultado será uma matriz.

```
-->x = 0:0.8:%pi
```

```
x =
```

```
0.      0.8      1.6      2.4
```

```
-->y = sin(x)
```

```
y =
```

```
0.      0.7173561      0.9995736      0.6754632
```



## Exercício

---

- Faça um programa em Scilab que faça a leitura dos valores de duas matrizes (A e B) de mesma dimensão. A seguir o programa deve imprimir a soma dessas duas matrizes.
- Faça um programa em Scilab que leia os dados de duas matrizes e mostre o resultado do produto dessas duas matrizes sem usar a multiplicação de matrizes do Scilab.
- Faça um programa que leia os valores de uma matriz e calcule a soma dos valores das colunas dessa matriz, armazenando cada soma em uma posição de um vetor, ou seja, o somatório da coluna 1 deve ser armazenado na posição 1 do vetor, o somatório da coluna 2 deve ser armazenado na posição 2 do vetor e assim por diante.

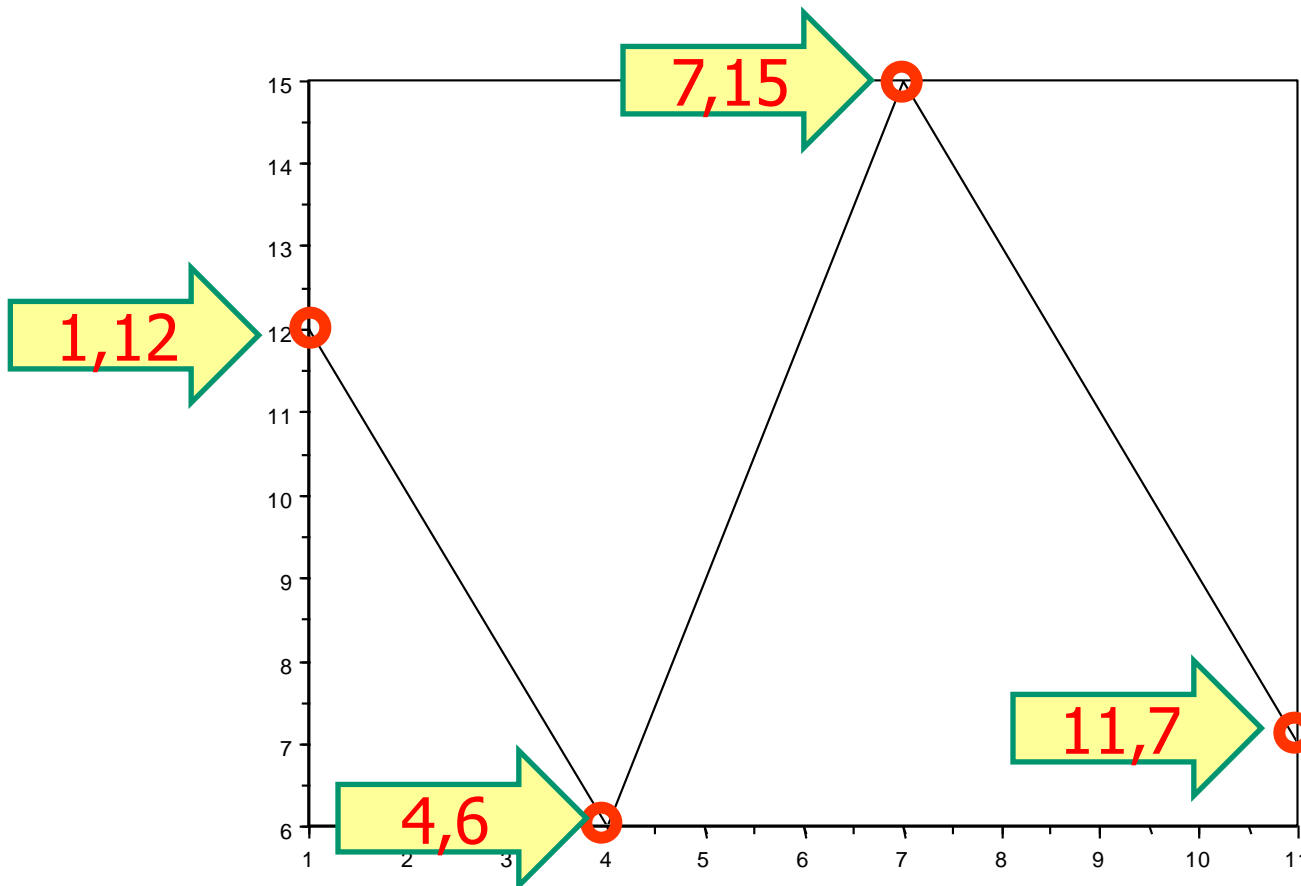
## Vetores e gráficos

---

- Vetores são muito úteis para a construção de gráficos.
- O comando mais simples é `plot2d(x,y)`, onde `x` e `y` são vetores com as mesmas dimensões.
  - Tal comando constrói um gráfico unindo, por retas, os pontos com coordenadas : `(x(1), y(1))`, `(x(2), y(2))`, `(x(3), y(3))`, ...

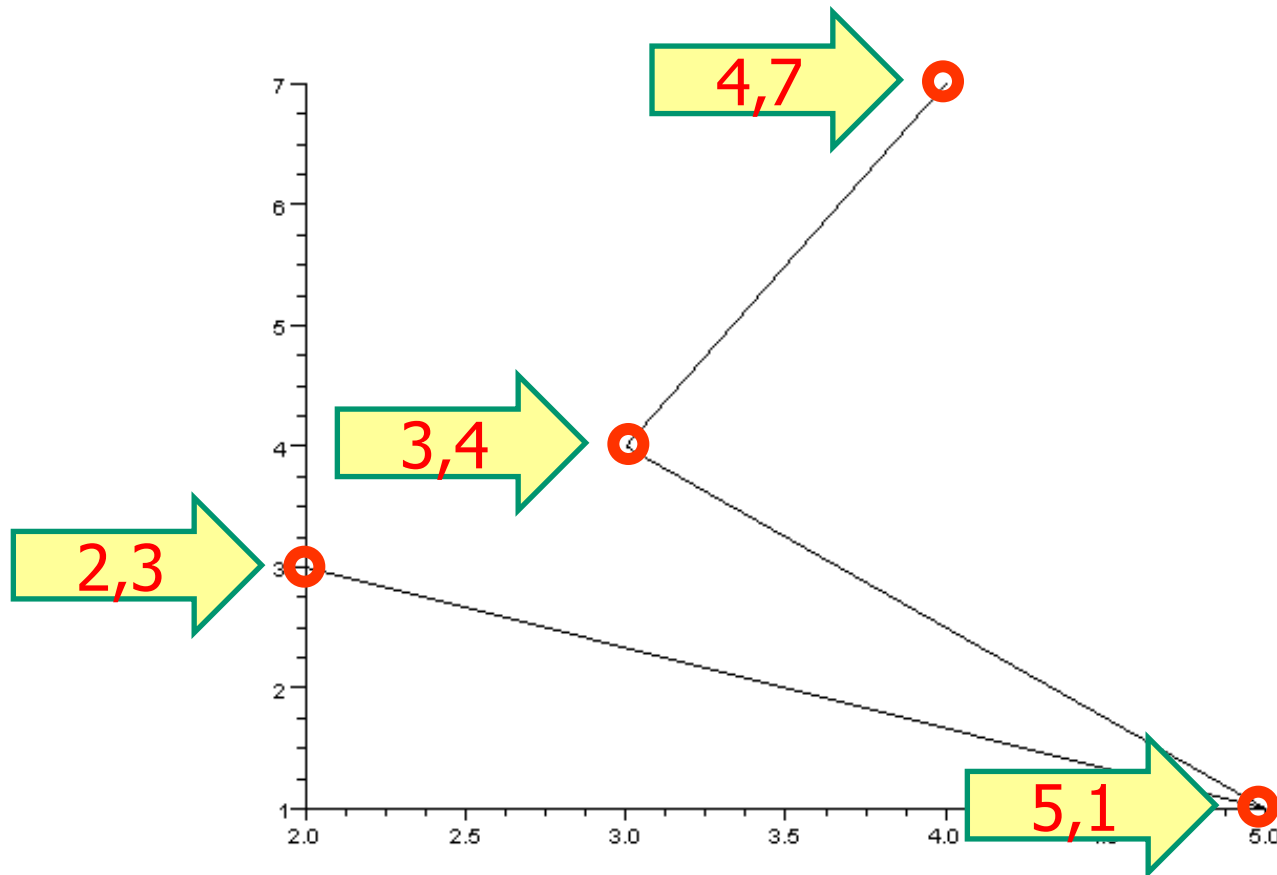
# Vetores e gráficos

```
-->x = [1 4 7 11]; y = [12 6 15 7];  
-->plot2d(x,y)
```



# Vetores e gráficos

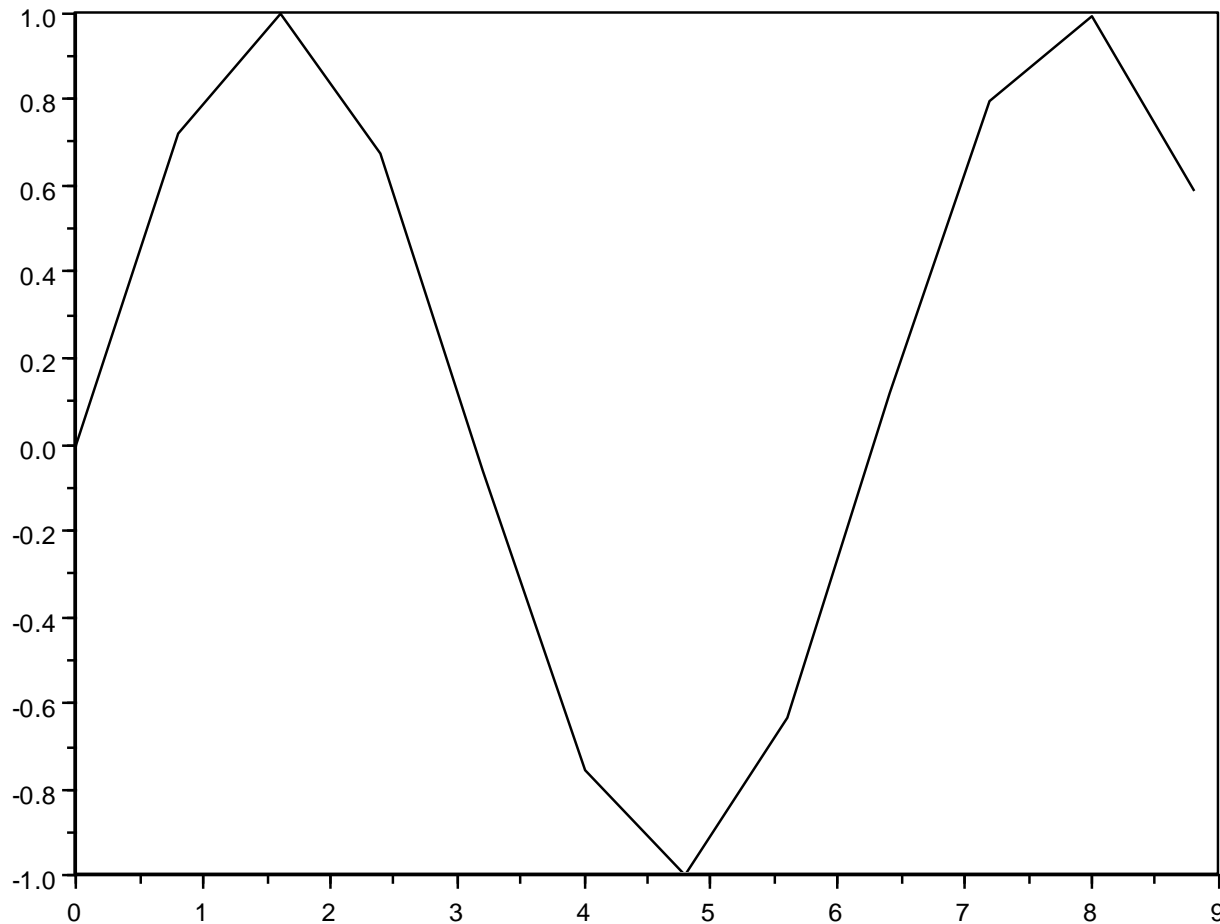
```
-->x = [2 5 3 4]; y = [3 1 4 7];  
-->plot2d(x,y)
```



# Gráfico seno(x) – 1ª Versão

```
-->x = 0:0.8:3*%pi;  
-->y = sin(x);  
-->plot2d(x,y)
```

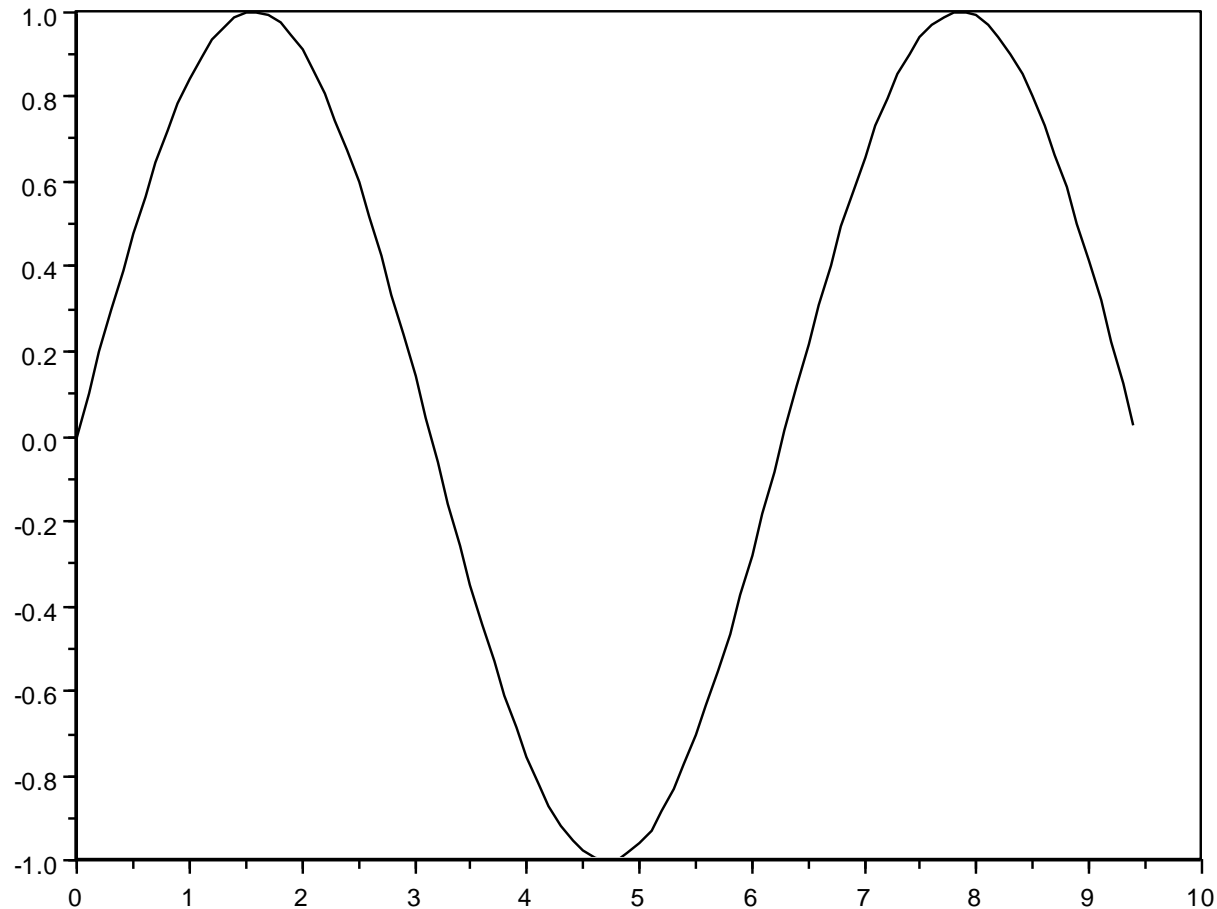
O espaçamento de  
0.8 está grande



## Gráfico seno(x) – 2ª Versão

```
-->x = 0:0.1:3*%pi;  
-->y = sin(x);  
-->plot2d(x,y)
```

O espaçamento de  
0.1 está bem melhor



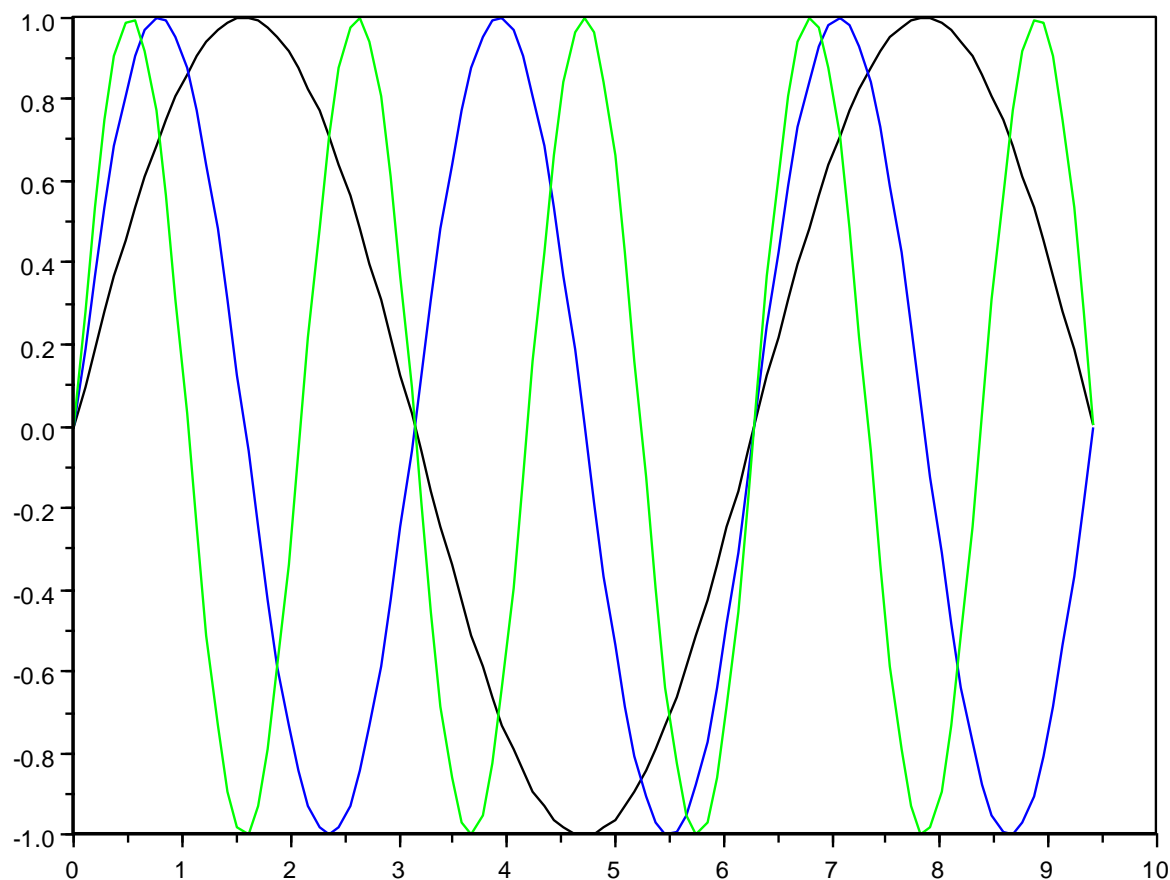
## Gráfico com várias curvas

---

- A função **plot2d** pode ser usada para traçar várias curvas em um único gráfico.
- No caso, **plot2d(x,M)**, onde
  - **x** é um vetor coluna, e
  - **M** é uma matriz com o mesmo número de linhas de **x**, gera um gráfico de **x** versus cada coluna de **M**.

# Gráfico com várias curvas

```
-->x = linspace(0, 3*%pi, 101)';  
-->plot2d(x,[sin(x) sin(2*x) sin(3*x)])
```



$x$  é um vetor coluna (e  $\sin(x)$ ,  $\sin(2*x)$  e  $\sin(3*x)$  também são)



# Matrizes de strings

---

```
-->a = ["s1" "s2"]
```

```
a =
```

```
!s1  s2  !
```

```
-->b = ["s1" ; "s2"]
```

```
b =
```

```
!s1  !
```

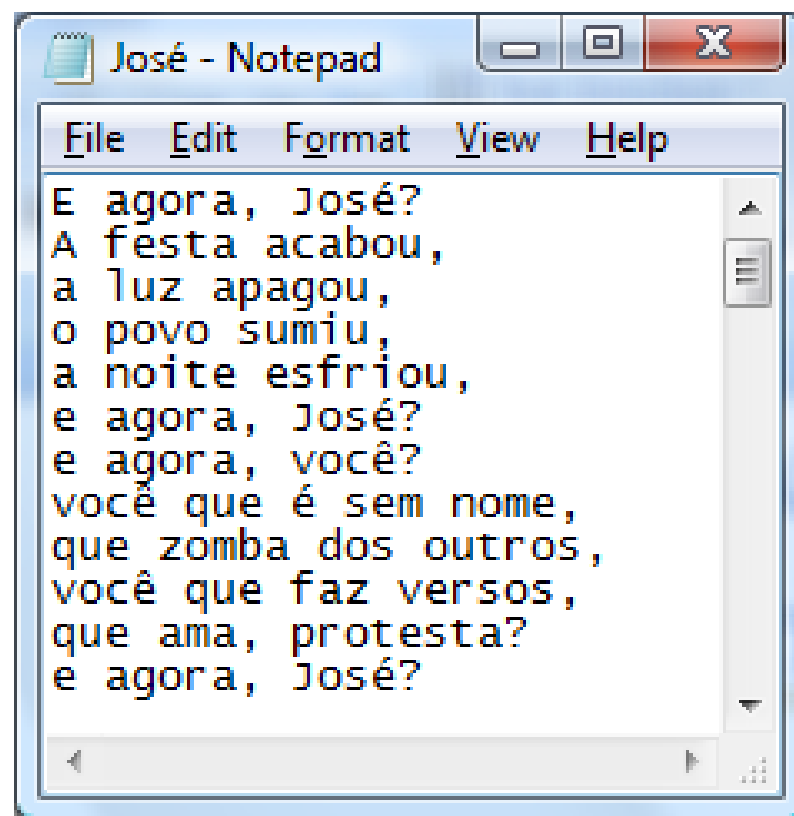
```
!    !
```

```
!s2  !
```

# Leitura de arquivos como matrizes de strings

- O comando `s = mgetl(arq)`, onde `arq` é o apontador de um arquivo já aberto, lê todas as linhas do arquivo referenciado por `arq` e coloca cada uma delas como um elemento do vetor coluna de strings `s`.
- Exemplificação de uso:

```
fpath = uigetfile();  
arq = mopen(fpath, "r");  
linhas = mgetl(arq);  
mclose(arq);
```



# Leitura de arquivos como matrizes de strings

---

- Este programa usado com o arquivo "José" produz:

```
-->linhas
linhas =
!E agora, José?           !
!A festa acabou,         !
!a luz apagou,           !
!o povo sumiu,           !
!a noite esfriou,        !
!e agora, José?          !
!e agora, você?          !
!você que é sem nome,    !
!que zomba dos outros,   !
!você que faz versos,    !
!que ama, protesta?      !
!e agora, José?          !
```

# Matrizes numéricas e arquivos no Scilab

---

- Os comandos já vistos de leitura e gravação de arquivos podem ser usados para a leitura de matrizes, mas o Scilab oferece mecanismos mais simples através dos comandos **fscanfMat** e **fprintfMat**.
- Estes comandos leem ou gravam arquivos que contêm somente números em formato tabular, à exceção das primeiras linhas que podem conter textos.
- A abertura e o fechamento dos arquivos são feitas automaticamente.

# Comando `fprintfMat`

Nome físico  
do arquivo

Matriz  
numérica

Formato dos  
números

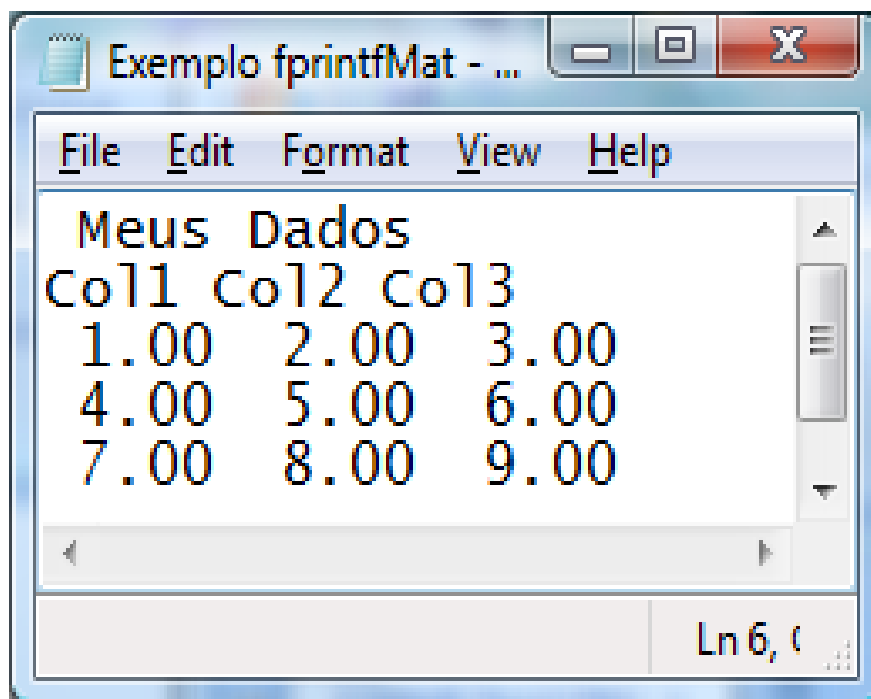
```
fprintfMat(arq, m, '%5.2f', Cabecalho);
```

Grava o conteúdo da matriz  
"m" no arquivo "arq"

É um vetor coluna de  
strings que são gravadas nas  
primeiras linhas e,  
normalmente, contém uma  
explicação sobre os campos  
presentes no arquivo

# Comando fprintfMat

```
a = [1 2 3; 4 5 6; 7 8 9];  
arq = uigetfile();  
Cabecalho = [" Meus Dados "; "Col1 Col2 Col3"];  
fprintfMat(arq, a, "%5.2f", Cabecalho);
```



# Comando `fscanfMat`

---

Matriz  
numérica

Nome físico  
do arquivo

```
m = fscanfMat(arq) ;
```

Lê uma matriz do arquivo  
"arq", armazenando-a em "m".  
Linhas com texto no início do  
arquivo são ignoradas

# Comando fscanfMat

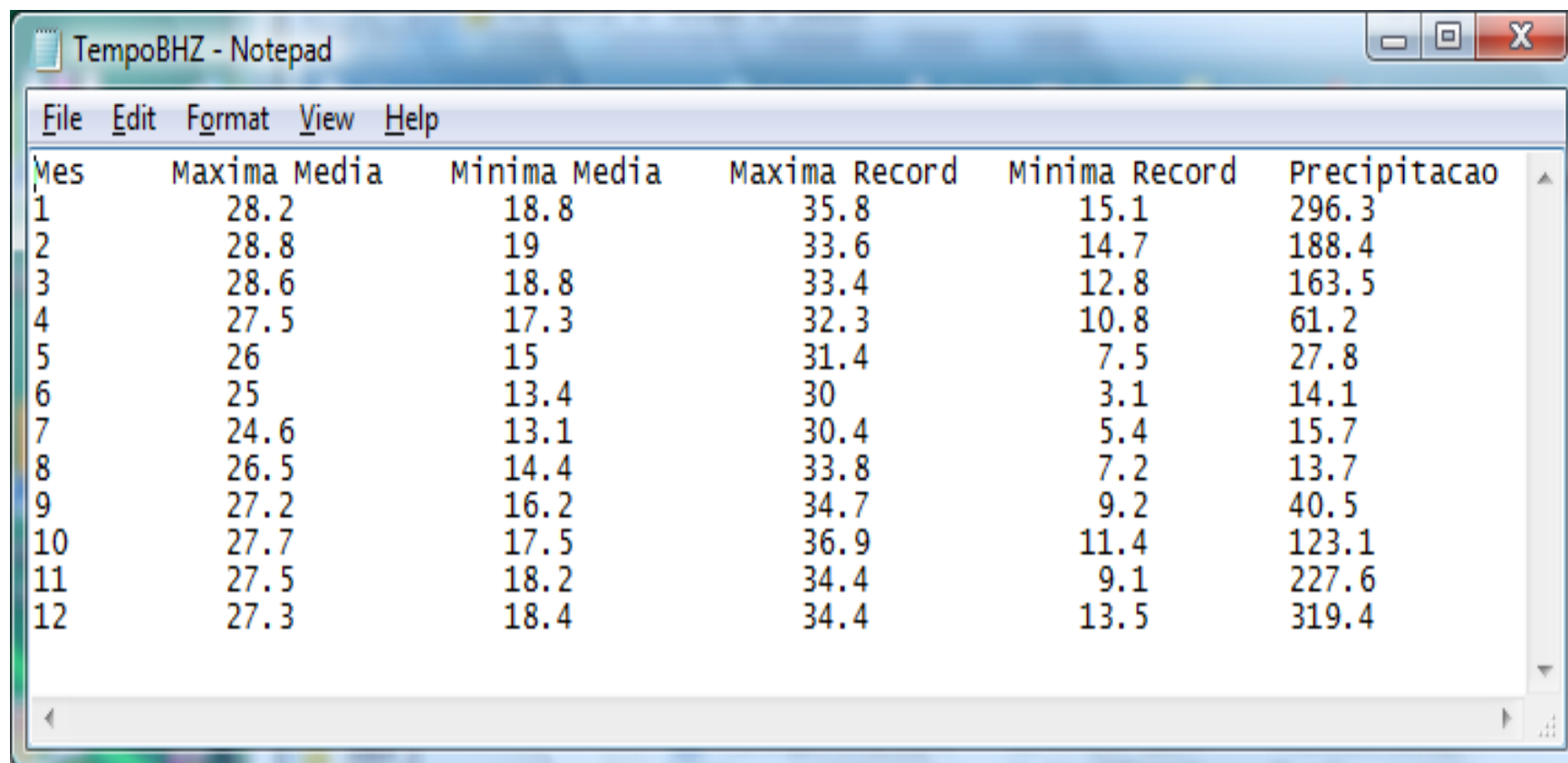
---

```
arquivo = uigetfile();  
m = fscanfMat(arquivo);
```

```
m =  
  
    1.    2.    3.  
    4.    5.    6.  
    7.    8.    9.
```



# Programa: Clima em Belo Horizonte



TempoBHZ - Notepad

Mes	Maxima Media	Minima Media	Maxima Record	Minima Record	Precipitacao
1	28.2	18.8	35.8	15.1	296.3
2	28.8	19	33.6	14.7	188.4
3	28.6	18.8	33.4	12.8	163.5
4	27.5	17.3	32.3	10.8	61.2
5	26	15	31.4	7.5	27.8
6	25	13.4	30	3.1	14.1
7	24.6	13.1	30.4	5.4	15.7
8	26.5	14.4	33.8	7.2	13.7
9	27.2	16.2	34.7	9.2	40.5
10	27.7	17.5	36.9	11.4	123.1
11	27.5	18.2	34.4	9.1	227.6
12	27.3	18.4	34.4	13.5	319.4

# Programa: Clima em Belo Horizonte

---

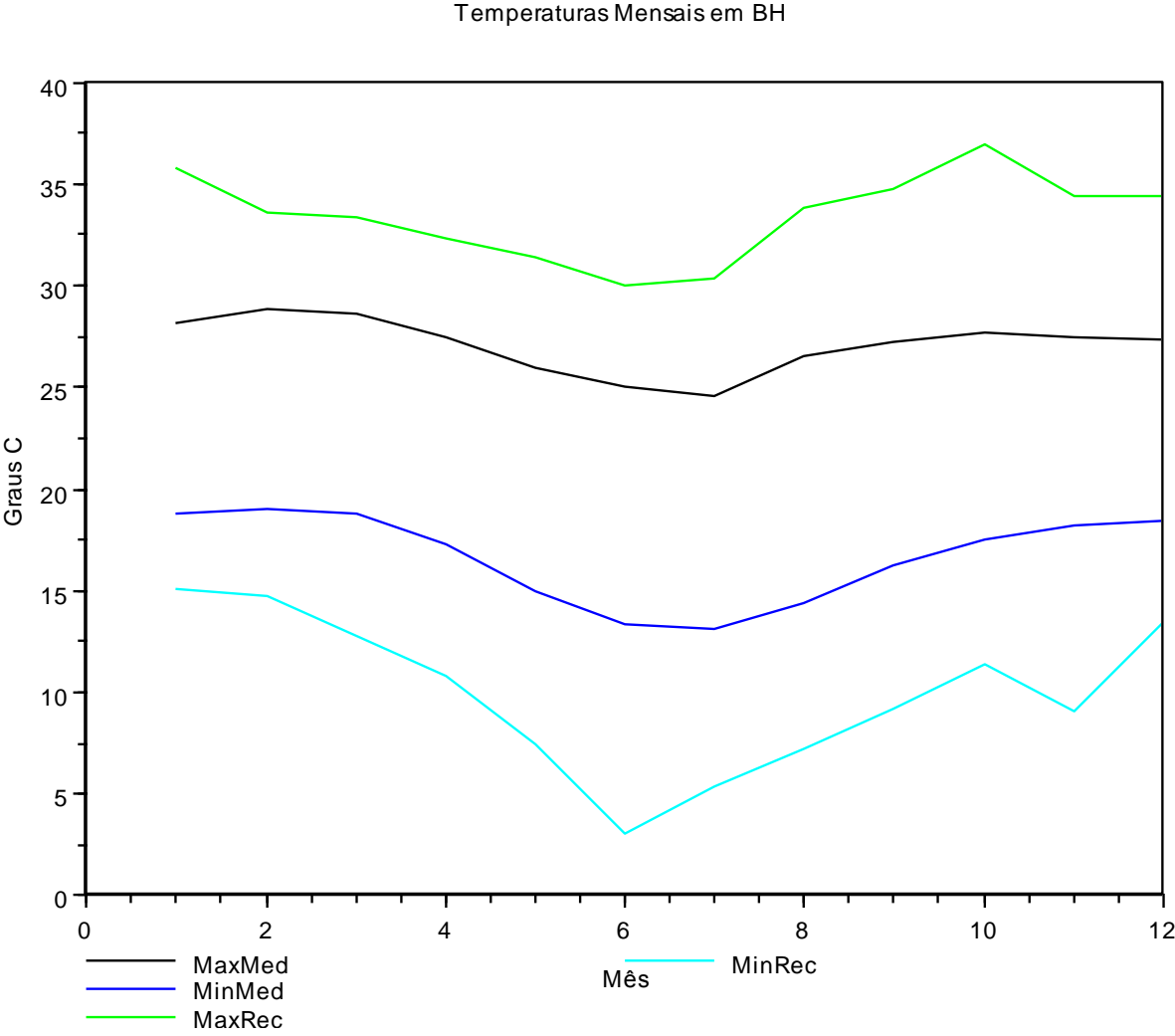
- Faça um programa que:
  - Leia o arquivo apresentado para uma matriz **ClimaBH**, usando a função **fscanfMat**, que ignora linhas de cabeçalho em um arquivo.
  - Da matriz **ClimaBH**, extraia os vetores **MaxMed**, **MinMed**, **MaxRec**, **MinRec** e **Precip**, com significados óbvios.
  - Gere um gráfico que tenha simultaneamente os valores de **MaxMed**, **MinMed**, **MaxRec** e **MinRec**.

# Programa: Clima em Belo Horizonte

---

```
arqClima = uigetfile();  
ClimaBH = fscanfMat(arqClima);  
  
MaxMed = ClimaBH(:,2); // MaxMed = 2a coluna  
MinMed = ClimaBH(:,3); // MinMed = 3a coluna  
MaxRec = ClimaBH(:,4); // MaxRec = 4a coluna  
MinRec = ClimaBH(:,5); // MinRec = 5a coluna  
Precip = ClimaBH(:,6); // Precip = 6a coluna  
  
plot2d([1:12],[MaxMed MinMed MaxRec MinRec],...  
leg="MaxMed@MinMed@MaxRec@MinRec");  
  
xtitle("Temperaturas Mensais em BH","Mês","Graus C");
```

# Programa: Clima em Belo Horizonte



# Laboratório

---

- Faça um programa em Scilab que:
  - Usando o comando `mopen`, `mclose`, `mgetl` e `mfscanf`, faça um programa que leia os dados do arquivo 'TempoBHZ.txt', salvando estes dados em uma matriz T. Imprima a matriz.
  - Usando o comando `fscanfMat`, faça:
    - Leia os dados do arquivo 'TempoBHZ.txt', transferindo esses dados para a matriz M.
    - Calcule a média mensais das médias máximas e mínimas, das máximas e mínimas registradas e das precipitações.
    - Faça um gráfico mostrando os dados da matriz com as legendas `MedMax`, `MedMin`, `MaxReg`, `MinReg` e precipitação.
    - Acrescente ao gráfico o título 'Temperaturas Mensais em BH', ao eixo x 'Mês' e y 'Graus C'

## Exercício

---

- Faça um programa para manter dados sobre cidades do Brasil.
  - Este programa deve manter os seguintes dados: nome da cidade, estado, população, taxa de desemprego e taxa de analfabetismo.
  - Para isso o programa deve ter um MENU que permita, cadastrar uma nova cidade, remover uma cidade, alterar os dados de uma cidade (o nome e o estado não podem ser alterados), mostrar a cidade com a maior população, mostrar a cidade com a menor taxa de analfabetismo, mostrar a cidade com a maior e a menor taxa de desemprego.
  - Os dados das cidades devem ser salvos no arquivo 'cidades.txt' antes do termino da execução do programa e devem ser recuperados ao iniciar a execução do programa.

## Exercício – Continuação

---

- Altere o procedimento de inserção para que ele permita inserir uma nova cidade apenas se não existir outra cidade com o mesmo nome e estado.
- Adicione ao menu a opção de remoção de uma cidade e a implemente. Para remover uma cidade, o programa deve receber o nome da cidade e o estado.
- Implemente uma opção para mostrar a população média das cidades.

# Matrizes e expressões lógicas

- O resultado de uma expressão lógica envolvendo matrizes é uma matriz de valores lógicos.

<pre>--&gt;a = [3 7; 8 2] a =     3.    7.     8.    2. --&gt;a &gt; 5 ans =     F    T     T    F</pre>	<pre>--&gt;a = [3 7; 8 2] ; --&gt;b = [5 6; 7 8] ; --&gt;a &gt; b ans =     F    T     T    F</pre>
--	---



# Matrizes e expressões lógicas

```
-->a = [3 9; 12 1]
-->x = 0; y = 0;
-->if a > 5 then; x = 10000; end;
-->if a > 0 then; y = 10000; end;
-->[x y]
ans =
    0.    10000.
-->a(a>5) = -1
a =
    3.    - 1.
    - 1.    1.
```

# Notas

---

- Arquivos permitem o tratamento de grandes volumes de dados
- Matrizes são absolutamente essenciais para programas que tratam com problemas algébricos ou com tabelas
- A manipulação de matrizes é muito flexível no Scilab
- Matrizes são também essenciais para o desenho de gráficos no Scilab