

# **Modelagem Matemático Computacional**

## **Introdução ao Scilab**

**Paulino Ribeiro Villas Boas**

# O que é o Scilab?

---

- Software livre para cálculo numérico e simulação de sistemas físicos
- É como uma linguagem de alto nível (mais fácil de compreender)
- Interpretado
- Matrizes e vetores são dados padrões

## **Usado nas áreas:**

1. Física
2. Processamento de imagens
3. Controle e processamento de sinais
4. Automação industrial
5. Controle de processos
6. Computação gráfica
7. Matemática
8. Modelagem biológica

# Por que aprender Scilab?

---

## 1. Linguagem de alto nível

**Existem 4 níveis de linguagem de programação**

- **Código de máquina**
- **Baixo nível:** Assembly
- **Médio nível, linguagens compiladas:** c, c++, pascal, fortran
- **Alto nível, linguagens interpretadas:** perl, python, matlab, scilab

## 2. Vários tipos de dados: matrizes, vetores, *strings*, listas

## 3. Funções mais elaboradas: inversão de matrizes, autovalores e autovetores

## 4. Várias funções para visualização de dados: plot2d, plot3d

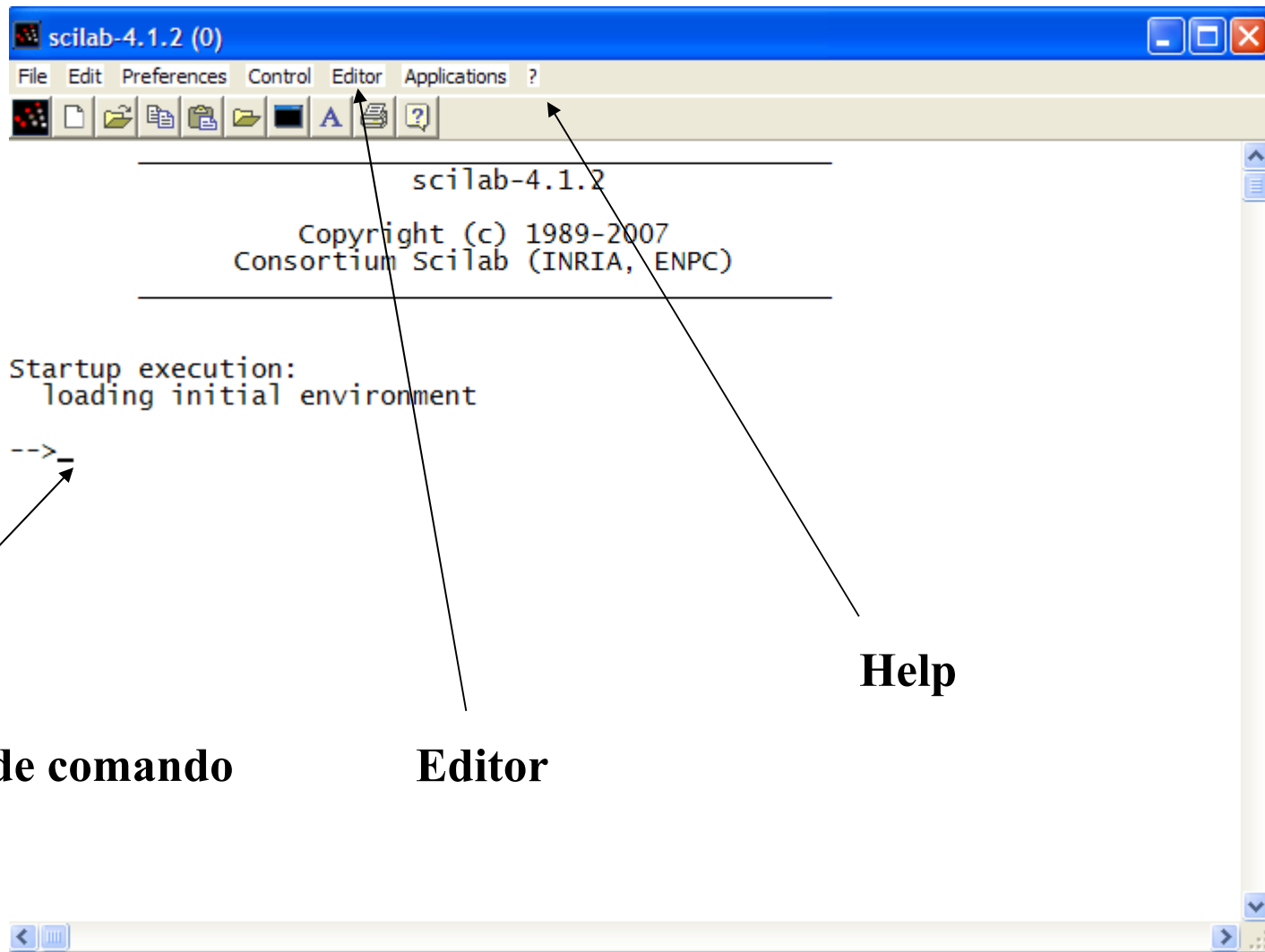
## 5. É um software livre

# Por que o Scilab foi criado?

---

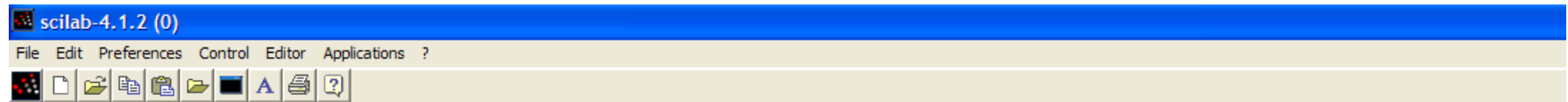
- Criado em 1989 por um grupo de pesquisadores da INRIA e da ENPC com intenção de ser livre
- Disponível como software livre desde 1994 pelo site [\*http://www.scilab.org\*](http://www.scilab.org)
- Consórcio Scilab desde 2003 mantido por diversas empresas.
  - Objetivos do consórcio:
    - organizar cooperação entre os desenvolvedores
    - obter recursos para manutenção da equipe
    - garantir suporte aos usuários
- Portabilidade, vários Sistemas Operacionais:
  - Linux
  - Windows
  - Solaris
  - Unix
  - Mac OS

# Executando o Scilab?



# Variáveis especiais

## Comando who



scilab-4.1.2

Copyright (c) 1989-2007  
Consortium Scilab (INRIA, ENPC)

Startup execution:  
loading initial environment

-->who  
your variables are...

scicos_pal	%scicos_menu	%scicos_short	%scicos_help	%scicos_display_mode	modelica_
libs					
scicos_pal_libs	%scicos_lhb_list	%CmenuTypeOneVector	%helps	WSCI	home
CreateScilabHomeDir	PWD	TMPDIR	guilib	sparselib	SCIHOME
		MSDOS			xdesslib
b		SCI			percentli
polylib	intlilb	elemlib	utililb	statslib	alglilb
armalib				siglib	optlib
tkscilib	tdcslib	s2flib	mtlilb	%F	%T
%gui				%z	%s
%pvm	%tk	\$	%t	%f	%eps
using	31764 elements	out of	5000000.	%io	%i
	and	58 variables	out of	9231	%e

your global variables are...

LANGUAGE	%helps	demolist	%browsehelp	LCC	%toolboxes	%toolboxes_dir
using	1189 elements	out of	11000.			
	and	7 variables	out of	767		

-->\_

## Constantes especiais

---

- **%e**: constante *neperiana*
- **%i**: raiz quadrada de -1, número imaginário
- **%pi**: constante  $\pi$
- **%eps**: máximo valor tal que  $1 + \%eps = 1$
- **%inf**: infinito
- **%nan**: não é um número
- **%t**: verdadeiro
- **%f**: falso

# Operadores de comparação

---

- < menor
- <= menor ou igual
- > maior
- >= maior ou igual
- == igual
- ~= diferente
- <> diferente
- & e
- | ou
- ~ não



## Comandos básicos

---

- **pwd:** Mostra o diretório atual.
- **SCI:** Mostra o diretório onde o Scilab foi instalado.
- **ls:** Lista os arquivos do diretório.
- **chdir(“dir”) ou cd(“dir”):** Muda de diretório.
- **mkdir(“dir”):** Cria um diretório.
- **rmdir(“dir”, ‘s’):** Remove um diretório e todo o seu conteúdo.

## Comandos básicos

---

**exec(“arquivo.sci”):** Executa um *script* Scilab.

**help():** Mostra o help do Scilab.

**disp(var):** Mostra o conteúdo de variáveis.

**save(‘file’, var):** Salva variáveis específicas em um arquivo binário.

**load(‘file’, ‘var’):** recupera os valores salvos em arquivo.

**clear:** Apaga as variáveis não protegidas do ambiente.

## Exercício 1:

- Crie um diretório chamado File.
- Entre nesse diretório.
- Execute os comandos:
  - `A = ones(2,2);`
  - `disp(A);`
- Salve a variável A no arquivo teste.dat
- Apague a variável A
- Carregue o valor de teste.dat na variável A novamente;
- Verifique o valor de A;
- Remova o diretório File.

# Definição das variáveis

---

- **Sensível a maiúsculas e minúsculas**
- **Palavra única**
- **Até 24 caracteres**
- **Não pode iniciar com número**

## **Exercício 2**

- Verificar se é possível declarar as seguintes variáveis:
- `a = 1;`
- `Var_1 = 2;`
- `2var = 3;`
- `esta variável = 3;`
- `ítems = 2;`
- `b = 2; B = 3;` verifique se `b` e `B` têm o mesmo valor.

## Como salvar os comandos executados

---

- Comando **diary**: Armazena os comandos em um texto
- Exemplo:

```
diary('arquivo.txt');
```

```
a = 100;
```

```
b = 200;
```

```
c = a+b;
```

```
disp(c);
```

```
diary(0);
```

# Calculadora X Ambiente de programação

---

- **Calculadora**

Os comandos são digitados diretamente do *prompt*.

- **Ambiente de programação**

Os comandos são digitados em um arquivo texto.

# **Operações e estruturas básicas**

# Números complexos

- $Z = a + \%i*b$

## Exercício 3:

1. Dados os seguintes números complexos,

$$Z1 = 3 + 5i; \quad Z2 = 7 + 3i$$

execute as seguintes operações:

a)  $Z1 + Z2$ ;

b)  $Z1 * Z2$ ;

c)  $Z1 + \text{sqrt}(-20)$ ;

d) Calcule os módulos de  $Z1$  e  $Z2$  e compare com ***abs(z)***;

Lembre-se

$$|z| = \sqrt{a^2 + b^2}$$



# Vetores

- **Declaração de vetores:**

$X = [x_1 \ x_2 \ x_3 \ \dots]$  vetor linha

$X = [x_1; x_2; x_3; \dots]$  vetor coluna

- **Transposição de vetores:  $X'$**

$$\mathbf{x} = (x_1, x_2, \dots, x_n)$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

## Exercício 4:

1. Verifique a diferença entre:  $x = [1 \ 2 \ 3]$  e  $x = [1; 2; 3]$

2. Dados os vetores:

$$x = [1, 2, 3, 4, 5] \text{ e } y = [2, 4, 6, 8, 10]$$

Calcule:

a)  $z = x + y$ ;

b)  $z = x * y$ ;

c) Formas transpostas de  $x$  e  $y$ ;

d) Dados  $z_1 = x * y'$ ; e  $z_2 = x' * y$ ;

Verifique se  $z_1 = z_2$ .

- **A = Valor\_inicial:incremento:Valor\_final**
- **Exemplos:**
  - **A = 1:10;**
  - **B = 1:2:10;**
  - **C = 1:0.2:10;**
  - **D = 10:-1:1;**
  - **E = 1:%pi:20;**
  - **F = 0:log(%e):20;**
  - **G = 20:-2\*%pi:-10**

# Operações com vetores

---

- **Dimensão:** `length(x)`
- **Número de linhas e colunas:** `[nr,nc] = size(x)`
- **Elementos iguais a 1:** `x = ones(N,1)`
- **Vetores nulos:** `x = zeros(N,1)`
- **Vetores com valores aleatórios:** `x = rand(N,1)`

## Exercício 5:

- Crie:
  - Um vetor unitário com 10 elementos
  - Um vetor nulo com 5 elementos
  - Um vetor com 10 elementos aleatórios
  - Verifique suas dimensões

## Operações com vetores

---

- **Apaga elemento:  $X(i) = []$**
- **Insere elemento  $i$  no final:  $X = [X \ i]$**
- **Acessa último elemento:  $X(\$)$**
- **Acessa elementos entre  $n$  e  $m$ :  $X(n:m)$**
- **Agrupa dois vetores:  $c = [x \ y];$**

## Exercícios 6:

1 - Dado o vetor  $X = [1 \ 2 \ 3 \ 4 \ 5]$ ;

Insira o valor 10 no final

Apague o quinto elemento do vetor

Atribua valor zero aos elementos entre 2 e 4

2 - Dados os vetores

$$X = [\pi \text{ e } \sin(\pi) \log(10)]$$

$$Y = [10,3 \ 1,1 \ -2,2]$$

crie um vetor Z que seja dado pela união de X e Y.

# Matrizes

Uma matriz geral consiste em  $m \times n$  números dispostos em  $m$  linhas e  $n$  colunas:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

# Matrizes

---

**Exemplo**

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

**No Scilab:**

**M = [1 2 3; 4 5 6; 7 8 9]**

# Operações com Matrizes

---

**Matrizes com elementos unitários:  $A = \text{ones}(M,N)$**

**Matrizes com elementos nulos:  $B = \text{zeros}(M,N)$**

**Matriz identidade:  $A = \text{eye}(N,N)$**

## **Exercício 7:**

Dadas as matrizes

$A = [1 \ 2 \ 3; 4 \ 5 \ 6]$ ; e  $B = [7;8;9]$

Determine:

- $A*B$
- $B*A$
- $A*\text{identidade}([A; B'])$
- $A*\text{ones}(A)$
- $A'*\text{ones}(A) + \text{identidade}([A; B'])$



# Operações com Matrizes

---

- **Acesso à linha i:  $A(i,:)$**
- **Acesso à coluna j:  $A(:,j)$**
- **Insere linha no final:  $A = [A; \text{linha}]$**
- **Insere coluna no final:  $A = [A \text{ coluna}]$**
- **Acesso à última linha:  $A(\$,:)$**
- **Acesso à última coluna:  $A(:,\$)$**

## **Exercício 8:**

1. Dada a matriz  $A = [2 \ 4 \ 6; 8 \ 10 \ 12; 1 \ 2 \ 3]$ 
  - Atribua valor zero à linha 3;
  - Multiplique a linha 2 por 10;
  - Remova a última linha
  - Insira o vetor  $B = [1 \ 2 \ 3]$  na última linha de A

# Operações com Matrizes

---

- **Acesso a um conjunto de linhas:**  $A(:, [i:j])$
- **Acesso a um conjunto de colunas:**  $A([i:j], :)$
- **Matriz com número aleatórios:**  $A = \text{rand}(N, M)$

## Exercício 9:

1. Crie uma matriz 5X5 de números aleatórios.
  - Atribua valor 0 à coluna 2.
  - Multiplique os elementos de 2 a 4 da coluna 3 por 10.
  - Divida os elementos de 1 a 3 da coluna 5 por 5.
  - Remova a coluna 3.
  - Remova a linha 2.

# Operações com Matrizes

---

- **Soma:**  $C = A + B$
- **Multiplicação:**  $C = A * B$
- **Multiplicação por um escalar:**  $B = \alpha A$
- **Matriz complexa:**  $C = A + B * \%i$  (A e B reais)
- **Matriz transposta:**  $C = A'$
- **Determinante:**  $d = \det(A)$
- **Diagonal:**  $d = \text{diag}(A)$ .

## Exercícios 10:

Dadas as matrizes ao lado,  
Calcule:

1.  $C = A + B$
2.  $C = A * B$
3.  $C = 10 * A + 5 * B$
4.  $C = A + B \% i$
5.  $C = A' + \text{rand}(B)$
6. Determinante de A
7. Determinante de B
8. Diagonal de A

$$A = \begin{bmatrix} 1 & 3 & 4 & 6 & 8 & 9 \\ 2 & 3 & 4 & 9 & 1 & 3 \\ 3 & 3 & 3 & 6 & 5 & 3 \\ 8 & 8 & 7 & 9 & 9 & 2 \\ 9 & 8 & 2 & 3 & 4 & 1 \\ 1 & 1 & 3 & 8 & 7 & 9 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 2 & 2 & 3 & 4 & 5 \\ 9 & 0 & 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 & 7 & 8 \\ 1 & 9 & 2 & 3 & 5 & 6 \\ 8 & 9 & 0 & 1 & 2 & 3 \\ 4 & 2 & 3 & 4 & 5 & 5 \end{bmatrix}$$

# Polinômios

---

$$P(x) = a_n + a_{n-1}x + \dots + a_2x^{n-2} + a_1x^{n-1} + a_0x^n$$

- Função *poly(a,x, 'flag')*
- *a*: matriz de número reais
- *x*: símbolo da variável
- *flag*: string ("roots", "coeff"), por default seu valor é "roots".

- **Definindo polinômios:**
- $y = \text{poly}([1 \ 2 \ 3], 'x', 'coeff');$   $y = 1 + 2x + 3x^2$
- **ou:**  $x = \text{poly}(0, 'x');$   $y = 1 + 2*x + 3*x^2;$

## Exercício 11:

Dados os seguintes polinômios:

$$y = 1 + 4x + 5x^2 + 6x^3$$

$$z = 3x + 5x^3 + 7x^4$$

Calcule:

a)  $y + z$

e)  $z*y/(z^3)$

b)  $y*z$

c)  $y^2 + 3z$

d)  $z/y$

- **roots(z):** calcula as raízes de um polinômio
- **[r,q] = pdiv(y,z):** efetua a divisão e calcula quociente e resto
- **coeff(y):** retorna os coeficientes do polinômio.

## Exercício 12:

Dados os seguintes polinômios:

$$y = 5 + 3x + 10x^2 + 8x^3 + 10x^4 + 6x^5$$

$$z = 2x + 3x^3 + 4x^4 + 5x^5$$

Calcule:

- a) suas raízes
- b) os coeficientes
- c) o resto e o quociente das divisões:  
 $y/z$  e  $z/y$

# Matrizes de polinômios

---

- Os elementos da matriz podem ser polinômios:
- Exemplo:
- $s = \text{poly}(0, 's');$
- $A = [1-2*s+s^3 \quad 3*s+4*s^2; \quad s \quad 2*s]$

## Exercício 13:

Dadas as matrizes de polinômios:

$$A = [2*x^2 + 3*x \quad x; 1 \quad x^3+2];$$

$$B = [3*x^4 + x^2 \quad x^5; 8*x + 1 \quad 5];$$

Calcule:

$$A*B$$

$$A/B$$

Determinantes de A e B



# Matrizes de polinômios

---

- Se  $A$  é uma matriz de polinômios:
- $A = A(\text{'num'})$ : retorna apenas os numeradores
- $A = A(\text{'den'})$ : retorna apenas os denominadores
- Exemplo:
- $s = \text{poly}(0, \text{'s'})$ ;
- $A = [(1+2*s+3*s^3)/(s+2) \quad 3*s+1/(2*s+1); s^4/(s^2+2) \quad 3*s^2+4*s^3]$
- $N = A(\text{'num'})$ ;
- $D = A(\text{'den'})$ ;

# Matrizes simbólicas

---

- Uma matriz simbólica pode ser construída com elementos do tipo `string`:
- `M=['a' 'b';'c' 'd'] ;`
- Se atribuirmos valores às variáveis podemos visualizar a forma numérica da matriz com a função `evstr()`:
- Exemplo:
- `a = 1;`
- `b = 4;`
- `c = 3;`
- `d = 5;`
- `evstr(M);`

# Matrizes: operadores especiais

- **Operador \: divisão à esquerda.**
- Seja  $\mathbf{Ax}=\mathbf{b}$  um sistema de equações lineares escrito na forma matricial, sendo  $\mathbf{A}$  a matriz de coeficientes,  $\mathbf{x}$  o vetor da incógnitas e  $\mathbf{b}$  o vetor dos termos independentes:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}_{n \times n}$$
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$
$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_n \end{bmatrix}$$

## Matrizes: operadores especiais

A resolução deste sistema é  $\mathbf{x}=\mathbf{A}^{-1}\mathbf{b}$ , ou seja, basta obter a matriz inversa de  $\mathbf{A}$  e multiplicá-la pelo vetor  $\mathbf{b}$ . No Scilab isto pode ser feito desta forma:

```
A=[1 3;3 4];b=[5;2];  
x=inv(A)*b
```

$$\begin{cases} 1x + 3y = 5 \\ 3x + 4y = 2 \end{cases}$$

Esta solução pode ser obtida com o operador “divisão à esquerda” cujo símbolo é \:

```
x=A\b
```

### Exercício 14:

1. Resolva o sistema linear. Substitua as soluções na equação para confirmar a solução.

$$\begin{cases} 2x + 3y + 3z = 2 \\ 4x + 3y + 2z = 1 \\ 3x + 7y + 9z = 5 \end{cases}$$

# Matrizes: operadores especiais

- **Operador . (ponto)**

Este operador é usado com outros operadores para realizar operações elemento a elemento.

## Exemplo:

**A = [1 2 3; 3 4 6; 7 8 9];**

**B = [2 4 6; 8 10 12; 14 16 18];**

**-->A.\*B**

**ans =**

<b>2.</b>	<b>8.</b>	<b>18.</b>
<b>24.</b>	<b>40.</b>	<b>72.</b>
<b>98.</b>	<b>128.</b>	<b>162.</b>

**-->A./B**

**ans =**

<b>0.5</b>	<b>0.5</b>	<b>0.5</b>
<b>0.375</b>	<b>0.4</b>	<b>0.5</b>
<b>0.5</b>	<b>0.5</b>	<b>0.5</b>

# Matrizes esparsas

---

- Uma matriz é dita esparsa quando possui uma grande quantidade de elementos iguais a zero.
- A matriz esparsa é implementada através de um conjunto de listas ligadas que apontam para elementos não zero. De forma que os elementos que possuem valor zero não são armazenados.

## Em Scilab:

```
A = [0 0 1; 2 0 0; 0 3 0]
```

```
-->sparse(A)
```

```
ans =
```

```
( 3, 3) sparse matrix
```

```
( 1, 3)    1.
```

```
( 2, 1)    2.
```

```
( 3, 2)    3.
```

# Matrizes esparsas

---

- **Exemplo:**
  - $A = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix}$  ;
  - $A = \text{sparse}(A)$ ;
  - $B = \begin{bmatrix} 0 & 1 & 0 \\ 2 & 0 & 2 \\ 3 & 0 & 0 \end{bmatrix}$ ;
  - $B = \text{sparse}(B)$ ;
  - $C = A * B$ ;
  - Para obter a matriz C na forma completa:
  - **$B = \text{full}(B)$ ;**

# Matrizes esparsas

---

- **Funções que criam matrizes esparsas:**
- **`sprand(n,m,fill)`: Matriz esparsa aleatória com  $n*m*fill$  elementos não nulos.**
- **`speye(n,n)`: Matriz identidade esparsa**
- **`spzeros(n,m)`: Matriz esparsa nula de dimensões  $n \times m$**
- **`spones(A)`: Coloca valor 1 onde  $A_{ij}$  é diferente de zero**

## Exercício 15:

Verifique as formas completas das matrizes abaixo com comando `full`.

- `A = sprand(2,2,0.1);`
- `B = speye(2,2);`
- `C = spzeros(2,2);`
- `D = spones(A);`



# Listas

- Uma lista é um agrupamento de objetos não necessariamente do mesmo tipo.
- Uma lista simples é definida no Scilab pelo comando *list*, que possui esta forma geral:

$$L = \text{list}(a_1, a_2, a_3 \dots a_N)$$

onde  $a_1, a_2, a_3 \dots a_N$  são os elementos da lista

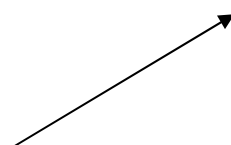
## Exemplo:

```
L = list(23, 1+2*%i, 'palavra', eye(2,2))
```

```
-->L
```

```
L =  
  L(1)  
  23.  
  L(2)  
  1. + 2.i  
  L(3)  
  palavra  
  L(4)  
  1.  0.  
  0.  1.
```

$L = [23, 1 + 2i, 'palavra', \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}]$



## Exercício 16:

Verifique os valores de:

$L(1)$ ,  $L(2)$ ,  $L(4)$  e  $L(4)$

# Listas

---

- Podemos criar sublistas, ou seja, listas dentro de listas.

- **Exemplo:**

```
L = list(23,1+2*%i,'palavra',eye(2,2))
```

```
L(4) = list('outra palavra',ones(2,2))
```

- Acessando elementos dentro da lista da lista:
  - `L(4)(1)`
  - `L(4)(2)`
- Agrupando duas listas:
  - `L1 = list(5,%pi, 'velocidade', rand(2,2));`
  - `L2 = list(1+2*%i,ones(3,3), 'aceleração');`
  - `L = list(L1,L2);`

## Funções elementares

---

**imag(x):** Mostra a parte imaginária de um complexo

**real(x):** Mostra a parte real de um complexo

**log(x), log10(x), log2(x):** Logaritmos natural, base 10 e base 2

**modulo(x,y):** Mostra o resto da divisão de x por y

**abs(x):** Retorna o valor absoluto (se x é real) e o módulo (se x é complexo)

## Funções elementares

---

**round(x):** Arredonda o valor de  $x$  para o inteiro mais próximo

**floor(x):** Arredonda para o menor inteiro

**ceil(x):** Arredonda para o maior inteiro

**sqrt(x):** Calcula a raiz quadrada de  $x$

# Funções trigonométricas

---

**$\cos(x)$ ,  $\sin(x)$ ,  $\tan(x)$ ,  $\cotg(x)$ : Retorna cosseno, seno, tangente ou cotangente de  $x$  ( $x$  deve estar em radianos)**

**$\acos(x)$ ,  $asin(x)$ ,  $atan(x)$ : Retorna o ângulo (em radianos)**

## **Exercício 17:**

Calcule:

$$a = \sin(\pi/2);$$

$$b = \tan(\pi);$$

$$c = \cotg(\pi/3);$$

$$d = \cos(\pi/4) + \sin(\pi/4);$$

## Exercícios 18:

Dados:

$x = [0.5 \ 3.4 \ 4 \ 2.8 \ 1.5];$

$y = [0.9 \ 2.2 \ 5 \ 1.1 \ 1.7];$

**Calcule:**

a)  $\text{seno}(x)$ ,  $\text{cosseno}(x.*y)$ ,  $\text{tangente}(y)$

b)  $\log(x)$ ,  $\log_{10}(x.*y')$ ,  $\log_2(x'.*y)$

c)  $\text{ceil}(x)$

d)  $\text{floor}(y)$

e)  $\text{round}(x.*y)$

f)  $\text{sqrt}(x) + \text{floor}(y.*y)$

g) Verifique se  $\text{abs}(2+2*%i) = \text{sqrt}(8)$

# **Programando com o Scilab**

## Características da linguagem Scilab

---

- O Scilab é um interpretador de comandos e por isso o código gerado não precisa ser compilado.
- Facilidade e simplicidade da linguagem estruturada.
- Não há necessidade de declaração prévia das variáveis.



# Comandos para iteração: for

- O laço **for**

```
for variavel = inicio:incremento:fim
    instrucao_1
    Instrucao_2
    ...
    instrucao_n
end
```

- **Exemplo:**

```
a = 0;
for i=1:1:3
    a = a+1;
end
```

```
L = list(2,[1 2; 3 4],'elemento');
for k=L
    disp(k);
end
```

# Comandos para iteração: while

- O laço **while**

```
while condicao
    instrucao_1
    instrucao_2
    ... ..
    instrucao_n
end
```

- O laço *while* repete uma seqüência de instruções enquanto uma condição for satisfeita.
- Útil quando não se sabe o número de iterações.

# Comandos para iteração: while

---

- **O laço `while`**

Operadores permitidos:

- `==` (igual a)
- `<` (menor que)
- `>` (maior que)
- `<=` (menor ou igual)
- `>=` (maior ou igual)
- `<>` ou `~=` (diferente)

- **Exemplo:**

```
x = 1; v = [];  
while x <= 16;  
    v = [v x]; x = x*2;  
end
```

## Comandos condicionais: if – then – else

---

- **Comandos condicionais são usados para executar seqüências de instruções a partir da avaliação de condições booleanas.**
- *if – then – else*

```
if condicao_1 then  
    sequencia_1  
else  
    sequencia_2  
end
```

**Avalia a condicao\_1 se ela for verdadeira (T, true) executa a sequencia\_1, caso contrário executa a sequencia\_2.**

# Comandos condicionais: if – then – elseif – else – end

## Forma geral:

```
if condicao_1 then
    sequencia_1
elseif condicao_2
    sequencia_2
...
else
    sequencia_n
end
```

- Se a *condicao\_1* for verdadeira executa a *sequencia\_1*.
- Se a *condicao\_1* for falsa avalia a *condicao\_2* e assim sucessivamente
- Se todas as condições são falsas executa a *sequencia\_n*.

## Comandos condicionais: if – then – else

---

### Exemplo:

```
x = -1;  
if x < 0 then  
    y = 2*x;  
else  
    y = x;  
end  
disp(y);
```

```
x = 10;  
if x < 0 then  
    y = -x;  
elseif x == 1  
    y = x;  
elseif x == 2  
    y = 2*x;  
else  
    y = 5*x;  
end  
disp(y);
```

## Comandos condicionais: select – case

### Forma geral:

```
select var
  case expressao_1
    sequencia_1
  case expressao_2
    Sequencia_2
  ...
  else
    sequencia_n
end
```

- O valor da variável *var* é comparado às expressões.
- Se os valores são iguais, a seqüência correspondente é executada.

## Comandos condicionais: select – case

---

**Exemplo:**

```
M = ['a','c','d'];  
x = ['a' 'b'];  
select M(1,1)  
    case 'a'  
        y = 'Letra a encontrada';  
    case 'b'  
        y = 'Letra b encontrada';  
end  
disp(y);
```



- Os *scripts* são arquivos de texto que contém comandos para serem usados em um *prompt* do Scilab.
- Por convenção estes arquivos possuem extensão *.sce*
- Os arquivos são criados no editor de texto do Scilab, o Scipad (ou em qualquer outro editor de texto).
- Os arquivos são executados no Scilab:
  - com o comando *exec*: *exec('nome\_do\_arquivo.sce');*
  - ou com o menu *File > File Operations* selecionando o arquivo e clicando no botão *exec*

### **Exercício 20:**

- Dado o vetor posição de um objeto que está se deslocando a uma aceleração constante  $a = 10\text{m/s}^2$ , com velocidade inicial  $v = 5\text{m/s}$  e posição  $S_0 = 1\text{m}$ , calcule a sua posição e velocidade entre  $t=1$  e  $10\text{s}$ . Armazene os dados em vetores.
- *Atenção: O script deve ser digitado em um editor de textos e salvo com a extensão .sce, por exemplo exercicio20.sce.*
- **Para executar no Scilab, digite: `exec('exercicio20.sce')`.**
- **Note que você deve estar no diretório em que o arquivo exercicio20.sce se encontra.**

# Funções

---

- **Devem ser usadas para operar sobre um conjunto de dados, realizar algumas instruções e retornar o resultado. Ex.: encontrar o valor máximo de um vetor**
- **Melhor estratégia em programação é quebrar um programa em várias partes menores (funções)**
- **Regra de escopo: variáveis definidas dentro de um função (conhecidas como variáveis locais) não permanecem no ambiente após a execução da função.**

# Funções

---

- **Por convenção, arquivos de funções recebem extensão .sci.**  
**Por exemplo: “funcao.sci”**
- **Uma função pode ser carregada de três formas:**
  - **no ambiente Scilab através do comando: `getf('funcao.sci')`**
  - **usando o comando `deff` ou**
  - **digitando o texto no Scipad e clicando no menu Execute, opção load into Scilab**

- **Definição:**

```
function [y1,...,yn]= nome_da_funcao(x1,...,xm)  
    instrucao_1  
    instrucao_2  
        ...  
    instrucao_p  
endfunction
```

**onde:**

- **x1,...,xm** são os argumentos de entrada;
- **y1,...,yn** são argumentos de saída e
- **instrução\_1,...,instrucao\_p** são as instruções executadas pela função.

# Recomendação para construir programas e funções em Scilab

---

1. Antes de começar a programar, recomenda-se entender bem o problema e em seguida planejar a melhor estratégia para resolvê-lo.
2. Divida em partes o problema – cada uma será uma função diferente.
3. Para começar, crie um arquivo para ser o programa principal. Digamos: “*program.sce*”. Lembre-se de salvar as suas modificações regularmente.
4. A entrada e saída de arquivos devem ser realizadas dentro do programa principal.
5. Visualização de resultados, como por exemplo gráficos, também deve ser realizado dentro do programa principal.
6. Deixe para funções apenas partes específicas do programa.
7. Dê nomes sugestivos para as funções, por exemplo: “*max\_element.sci*” para uma função que retorne o maior elemento de um vetor.
8. Use sempre o **help** do Scilab. Ele pode resolver inúmeros problemas rapidamente.

## Exemplo:

1. Definir uma função que converte um número complexo da forma cartesiana para a polar.

No arquivo: `cart_to_polar.sci`, digite:

```
function [mod,ang] = cart_to_polar(re,im)  
    mod = sqrt(re^2 + im^2);  
    ang = atan(im/re) * 180/%pi;  
endfunction
```

No arquivo `program.sce` (arquivo de execução) digite:

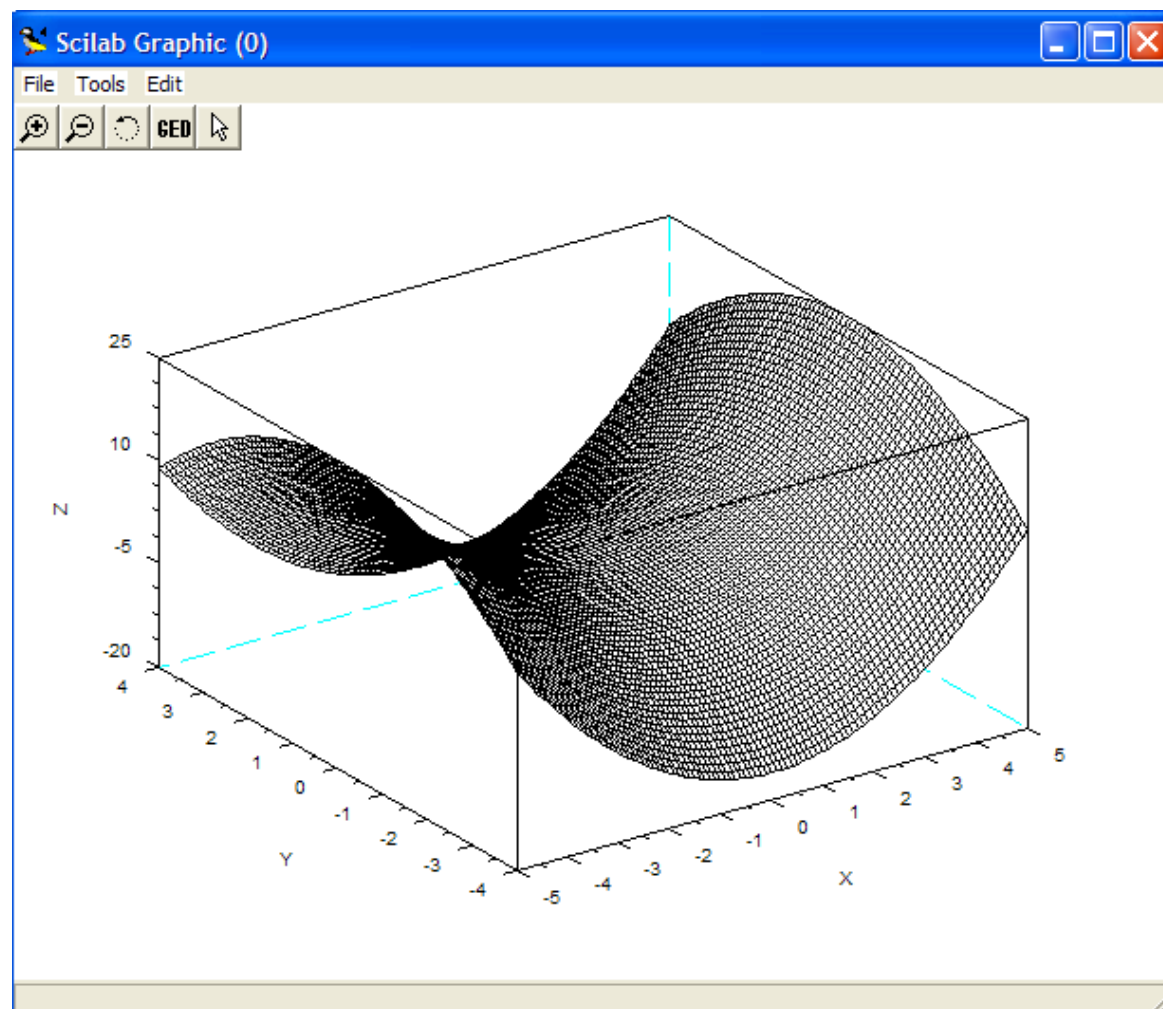
```
exec(' cart_to_polar.sci');  
z = 2 + 2*%i;  
[mod,ang] = cart_to_polar(real(z),imag(z));  
disp(mod);  
disp(ang);
```

## **Exercício 21:**

- 1 – Desenvolva uma função que calcule as raízes de uma equação do segundo grau usando a fórmula de Báskara. A função deve receber um polinômio e retornar um vetor com as soluções. Deve-se criar um arquivo texto para a função e um para a execução do programa.
- 2 – Crie uma função que calcule o fatorial de um número usando o comando de iteração for. Faça o mesmo usando o comando while.



# Gráficos no Scilab



- **Para gerar gráficos bidimensionais:**

- **plot2d(x,y,style)**

- Onde x e y são vetores.

- **Exemplo:**

- $x = [-2*\%pi:0.1:2*\%pi];$

- $y = \sin(x);$

- **plot2d(x,y);**

**Style: tipo de linha do gráfico. Valores inteiros positivos definem linhas contínuas, valores negativos definem pontos.**

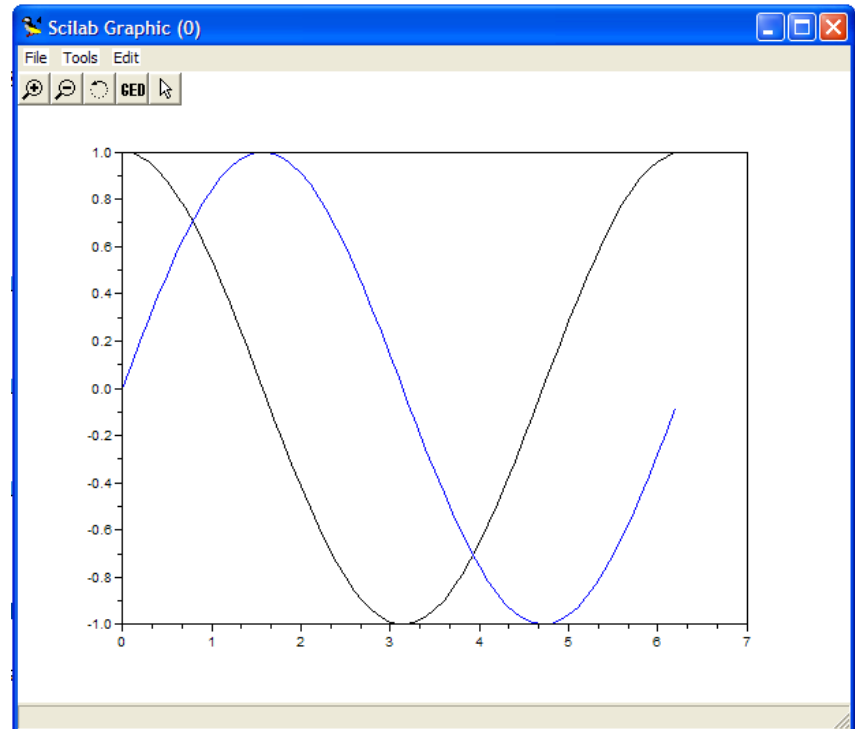
- **plot2d(x,y,-1);**

- **plot2d(x,y,2);**

- **y** pode ser uma matriz, sendo que o número de linhas de **y** deve ser igual ao número de elementos de **x**

- **Exemplo:**

```
x = [0:0.1:2*%pi];  
y = [sin(x)' cos(x)'];  
plot2d(x,y);
```



# Gráficos no Scilab

- **x e y** podem ser matrizes de mesma dimensão

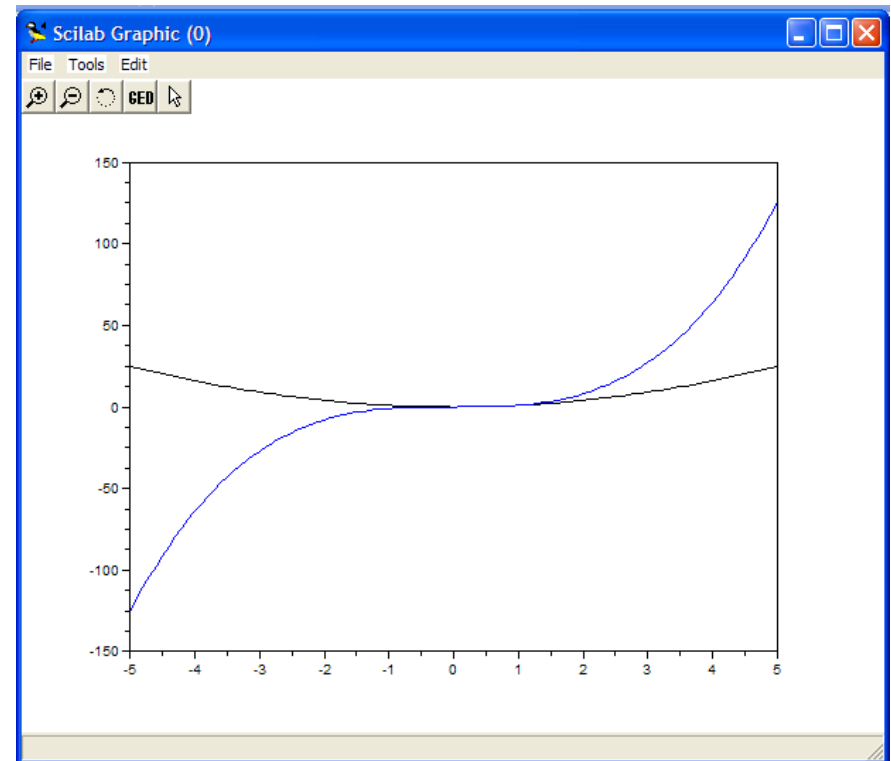
- **Exemplo:**

```
t = [-5:0.1:5];
```

```
x = [t' t'];
```

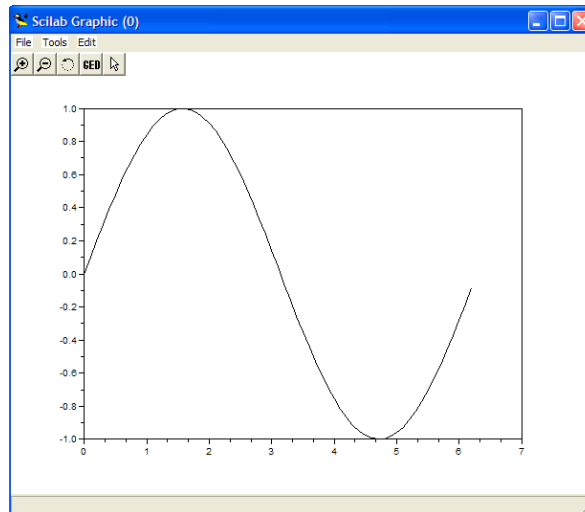
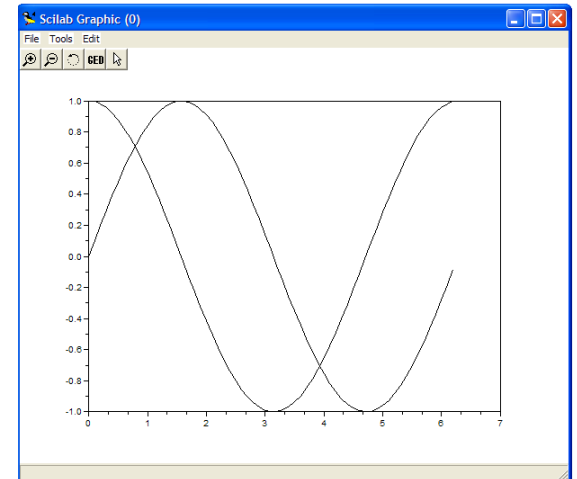
```
y = [(t^2)' (t^3)'];
```

```
plot2d(x,y);
```



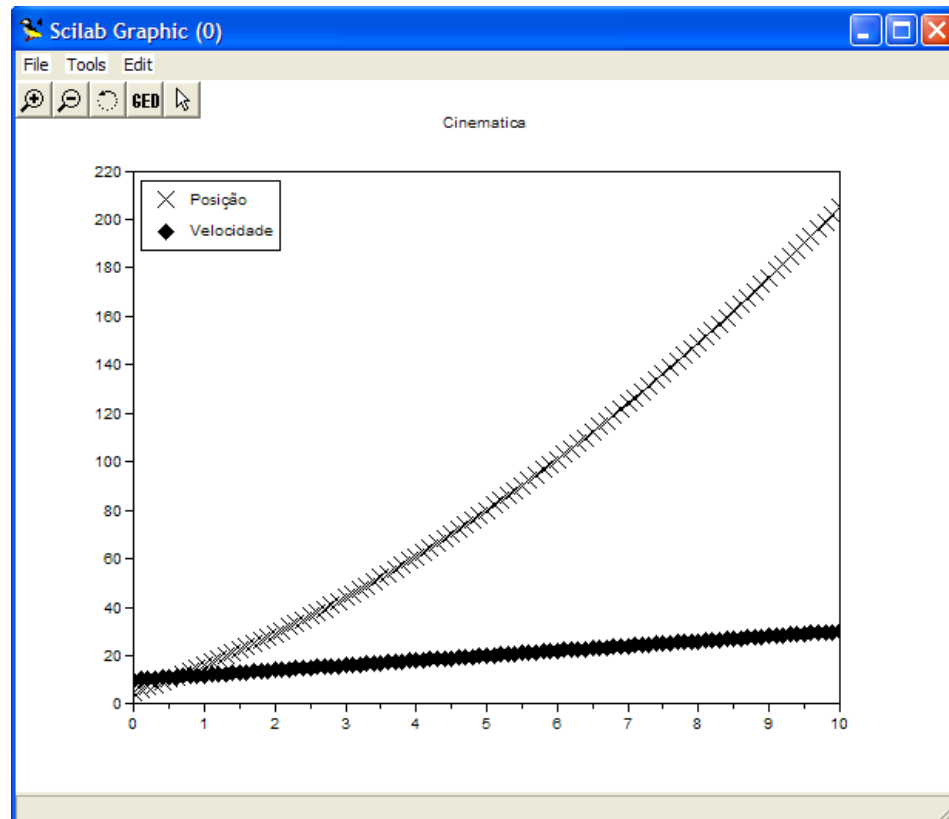
# Gráficos no Scilab

- Comandos básicos:
- `clf`: limpa a tela, evitando que o próximo gráfico se sobreponha ao anterior:
- Exemplo:
- $y = \sin(x)$ ;
- `plot2d(x,y)`;
- $z = \cos(x)$ ;
- `plot2d(x,z)`;
- Mas:
- `clf;plot2d(x,y)`;



# Gráficos no Scilab

- Comandos básicos:
- `xtitle('titulo')`: apresenta o título do gráfico
- `legend('legenda1', 'legenda2',...)`
- Exemplo:
- `t = 0:0.1:10;`
- `S = 5 + 10*t + 0.5*2*t.*t;`
- `V = 10 + 2*t;`
- `plot2d(t,S,-2);`
- `plot2d(t,V,-4);`
- `xtitle('Cinematica');`
- `legend('Posição', 'Velocidade');`



# Gráficos no Scilab

- **Comandos básicos:**
- **subplot:** divide um janela de um gráfico em sub-graficos
- **Exemplo:**

```
subplot(221)
```

```
plot2d(x,sin(x))
```

```
subplot(222)
```

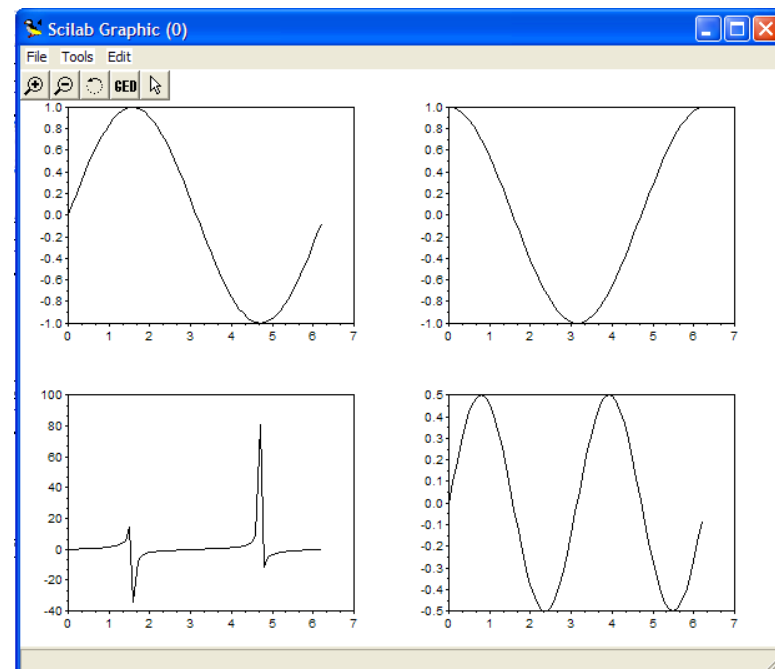
```
plot2d(x,cos(x))
```

```
subplot(223)
```

```
plot2d(x,tan(x))
```

```
subplot(224)
```

```
plot2d(x,sin(x).*cos(x))
```



- **Comandos básicos:**
- **logflag: define escala linear ou logarítmica**
  - “nn” – linear x linear
  - “nl” – linear x logarítmica
  - “ll” – logarítmica x logarítmica

## **Exemplo:**

```
x = 1:100;  
subplot(1,2,1);  
plot2d(x,y, logflag='nn');  
xtitle('Escala linear');  
subplot(1,2,2);  
plot2d(x,y, logflag='ll');  
xtitle('Escala log-log');
```





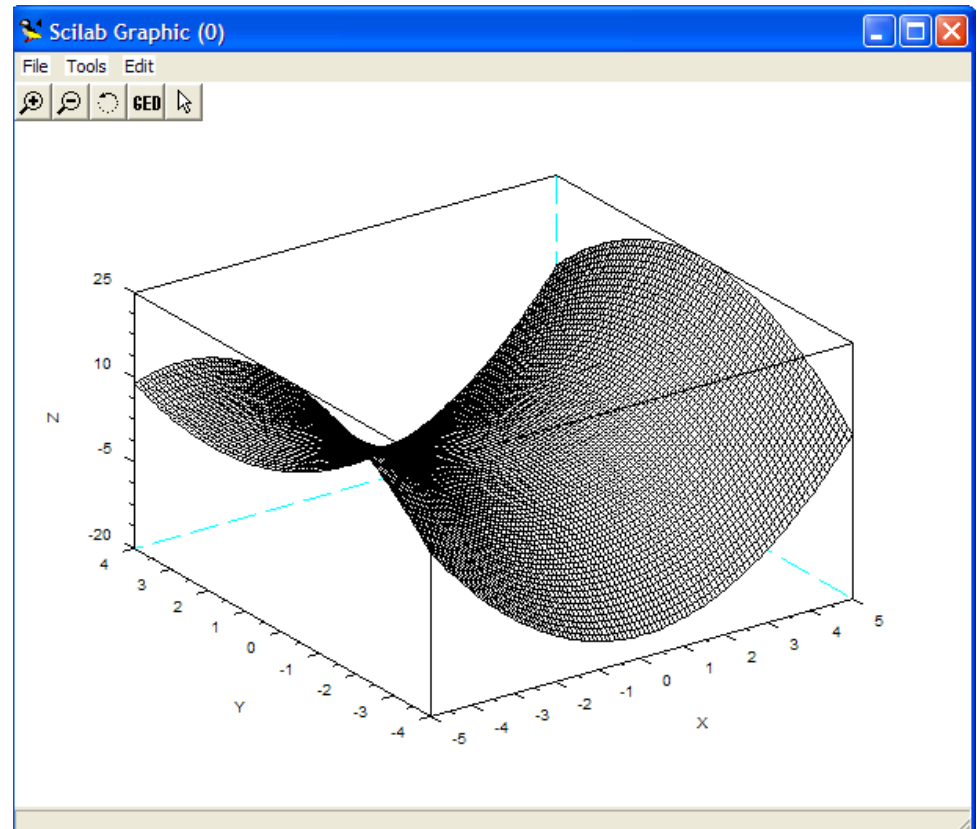
- Gráficos tridimensionais
- **mesh**: gera gráficos em 3D
- **Exemplo:**

```
[X,Y]=meshgrid(-5:0.1:5,-4:0.1:4);
```

```
Z=X.^2-Y.^2;
```

```
xtitle('z=x2-y ^2');
```

```
mesh(X,Y,Z);
```



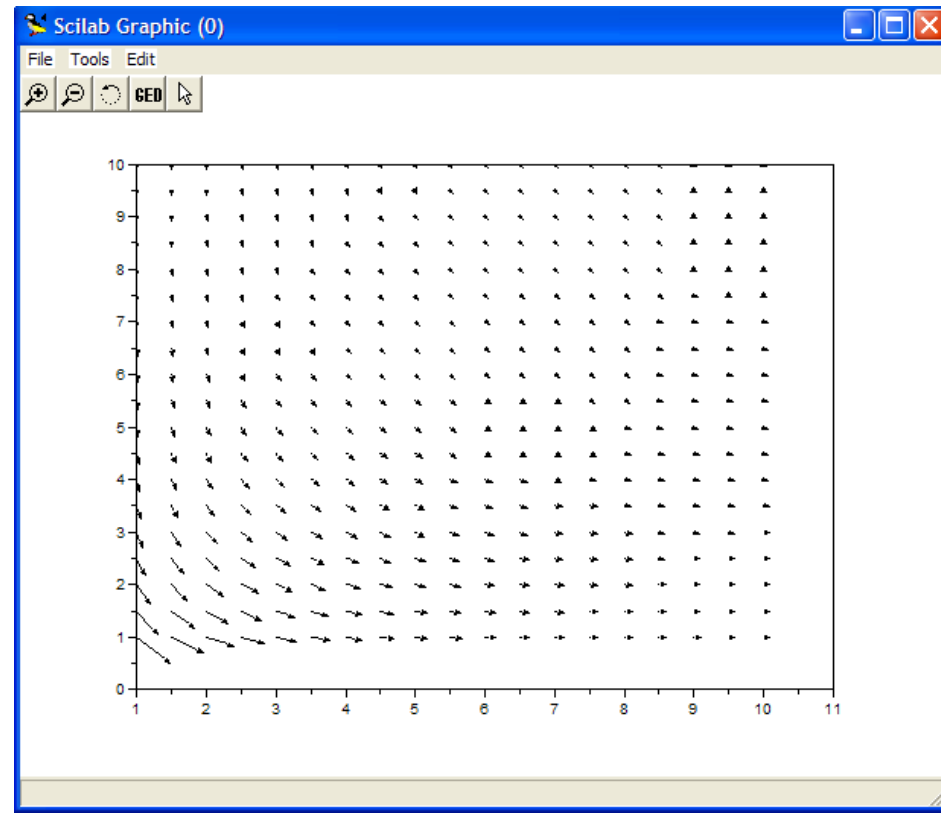
# Gráficos no Scilab

- **Campo vetorial**
- **champ** – mostra campos vetoriais
- **Exemplo:**

Velocidade da água em movimento circular

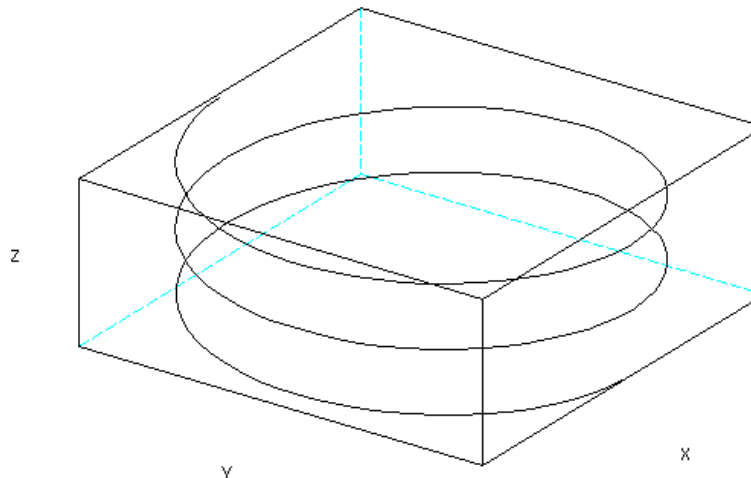
$$\mathbf{V}(x,y) = (y / x^2 + y^2) \mathbf{i} - (x / x^2 + y^2) \mathbf{j}$$

```
[x,y] = meshgrid(1:0.5:10,1:0.5:10);  
vx = y./(x.*x + y.*y);  
vy = -x./(x.*x + y.*y);  
champ(x(1,:),y(:,1),vx,vy);
```



- **Curvas paramétricas**
- **param3d: Gera uma curva paramétrica em 3D**
- **Exemplo:**

```
t=0:0.1:5*%pi;  
param3d(sin(t),cos(t),t/10,35,45,"X@Y@Z",[2,3])
```



- **Matplot: Mostra matrizes em 2D usando cores.**
- **Exemplo:**

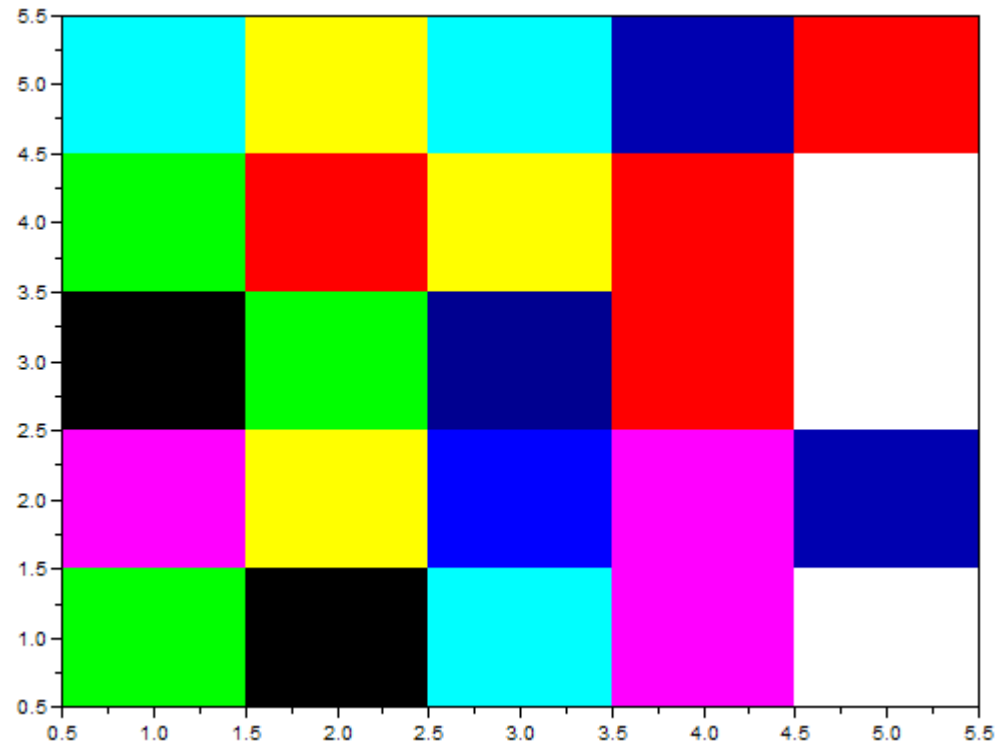
```
-->Matplot([1 2 3;4 5 6])
```

```
-->A = round(rand(5,5)*10)
```

A =

4.	7.	4.	10.	5.
3.	5.	7.	5.	8.
1.	3.	9.	5.	8.
6.	7.	2.	6.	10.
3.	1.	4.	6.	8.

```
--> Matplot(A)
```



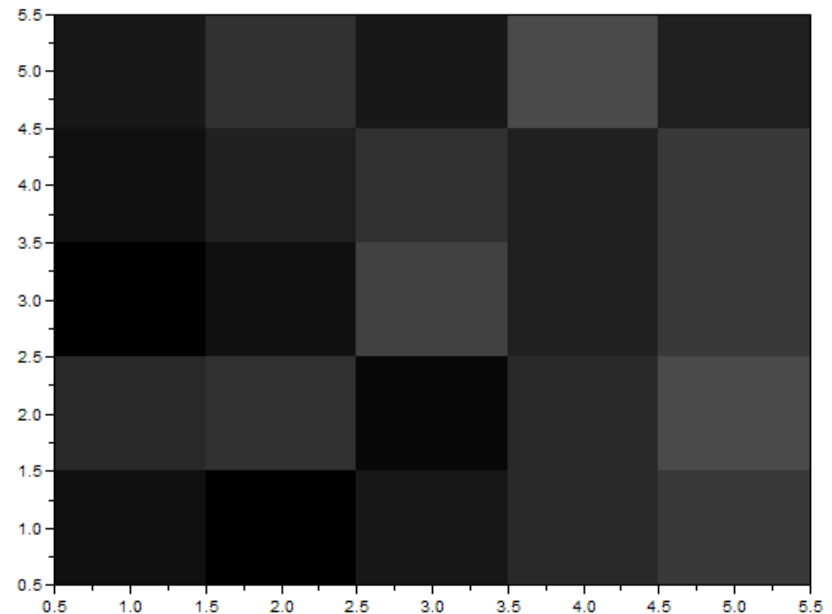
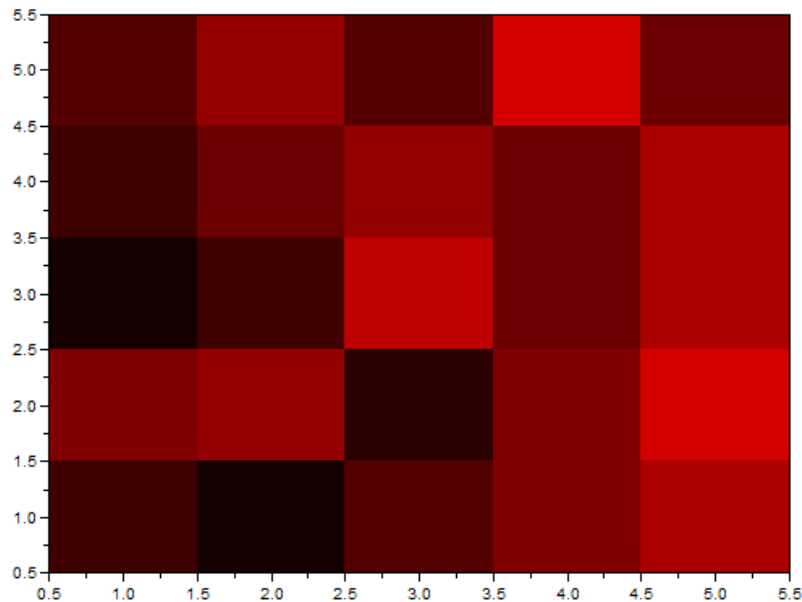
- **Colormap: Define o mapa de cores**
- **Exemplo:**

```
--> xset("colormap",graycolormap(32))
```

```
--> Matplot(A)
```

```
--> xset("colormap",hotcolormap(32))
```

```
--> Matplot(A)
```



- **Bar:** mostra graficamente um histograma
- **Exemplo**
- `h = [1 4 5 10 8 7 4 3 1];`
- `bar(h);`
- `f = [0.3 0.1 0.005 0.01 0.2 0.12 0.43 0.5 0.32 0.12 0.15];`
- `x = 0:0.1:1;`
- `scf();`
- `bar(x,f,'r');`

# **Comandos de entrada e saída**



## Comandos de entrada e saída

---

- **A = fscanfMat(“file.txt”):** Lê uma matriz de um arquivo
- **fprintfMat(“file.txt”,A):** Escreve uma matriz em um arquivo

### **Exemplo 1:**

```
A = rand(10,2);  
fprintfMat('Teste.txt',A);
```

### **Exemplo2:**

```
clear  
B = fscanfMat('Teste.txt');  
C = rand(10,2)*5;  
A = B + C;  
fprintfMat('Teste.txt',A);
```

## Comandos de entrada e saída

---

- **mopen: [fd,err]=mopen(“file”, “mode”):** abre um arquivo
- **mode:**
  - r:** abre apenas para leitura
  - w:** cria um novo arquivo para escrita
  - a:** abre um arquivo para adição:
- **[num\_read, num] = mfscanf(u, “%f”):** lê um elemento do arquivo
- **str=mgetstr(n [,fd] ):** Lê um caracter.
- **fprintf(file,format,value\_1,..,value\_n):** escreve em um arquivo
- **err=feof(fd):** Verifica se o final do arquivo foi encontrado.

# Comandos de entrada e saída

---

## **Exemplo:**

```
fd = mopen("teste_io.txt",'w');  
a = "Valor de teste";  
b = 1.1;  
fprintf(fd,'%s %1.2f',a,b);  
mclose(fd);
```

# Comandos de entrada e saída

## Exemplo:

Crie um arquivo e escreva dentro dele: “A B C”  
Feche o arquivo  
Abra o arquivo considerando *append* e escreva: “V X Z”  
Feche o arquivo  
Abra o arquivo e leia seu conteúdo.  
Verifique o conteúdo

## Solução:

```
fd = mopen(“novo_teste.txt”, 'w');  
a = “A”; b = “B”; c = “C”;  
fprintf(fd, ‘A:%t %t %t’, a, b, c);  
fclose(fd);  
//Abrindo com append  
fd = mopen(“novo_teste.txt”, 'a');  
a = “V”, b = “X”, c = “Z”;  
fprintf(fd, ‘B:%t %t %t’, a, b, c);  
fclose(fd);  
//Le o arquivo  
fd = mopen(“novo_teste.txt”, 'r');  
frase = [];  
while(meof(fd) == 0)  
    str = mgetstr(1, fd);  
    frase = [frase str];  
end
```

# Funções importantes

---

- **find:** Acha os índices dos elementos de um vetor que contém os elementos procurados:

## Exemplo:

```
A = [ 1 2 3 4 5 6]; B = find(A < 3); disp(A(B));
```

```
C = ['r', 's','r','t','a']; find(C == 'r');
```

- **sort:** organiza em ordem decrescente

```
Exemplo: A = [1 2 3;5 6 7; 2 3 1]; v = sort(A,'r'); s = sort(A,'c');
```

- **unique:** extrai os componentes de um único vetor.

```
Exemplo: A = [1 3 4 2 2 2 4 5 6 6 6 3 3]; B = unique(A);
```

- **union:** extraí os elementos da união de 2 vetores:

```
Exemplo: A = [ 1 1 1 2 2 2 3 4 4]; B = [1 2 5 6 7]; C= union(A,B);
```

## Funções importantes

---

**Vectorfind:** Acha em uma matriz um dado vetor (linhas ou colunas da matriz).

Exemplo:

$A = [1 \ 2 \ 3; 2 \ 3 \ 4; 4 \ 5 \ 6]$

$B = [1 \ 2 \ 3]$

`vectorfind(A,B,'r')`

## Exercício 22:

---

- Gere uma matriz aleatória A (5X5) com valores entre 0 e 10.
- Arredonde seus valores.
- Ache os diferentes valores da matriz e coloque no vetor B.
- Ordene as linhas da matriz em ordem decrescente.
- Coloque a união entre A e B em C.
- Ache as posições dos valores de  $A < 5$  na matriz
- Escreva essa matriz no arquivo “Mat.txt”

# Histograma

Histograma é uma forma de visualizar as frequências dos dados de um vetor em forma de barra. Para encontrar as frequências é necessário conhecer os limites  $xmin$  e  $xmax$  do vetor  $v$  e o número de caixas  $N$  em que se queira distribuir os dados. A idéia é colocar na mesma caixa o número de elementos que está entre o valor mínimo e o valor máximo da caixa.

## Exercício 23:

Crie um vetor aleatório de 10000 posições com o comando `rand(1,10000,'norm')`, encontre o histograma desse vetor e faça o gráfico correspondente. Qual é a forma da curva encontrada?

Dicas:

1. Crie um arquivo para ser o programa principal: “*programa.sce*”. Crie o vetor nesse arquivo, chame a função abaixo e faça o gráfico do histograma usando `plot2d`
2. Crie uma função com o nome “*histograma.sci*” que receba os parametros:  $v$  e  $N$ , onde  $v$  é o vetor e  $N$  é o número de caixas, e retorne  $[x,hx]$ , onde  $x$  é o valor da coordenada  $x$  e  $hx$  é a frequência.



**FIM**