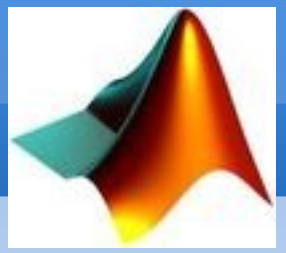
Matlab: Conceitos Básicos

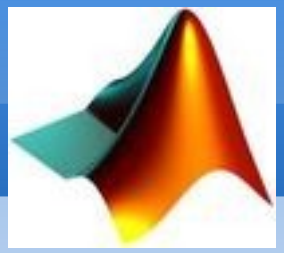


Felix Farias (2006)

http://felixfarias.googlepages.com/

UNICAMP

Introdução

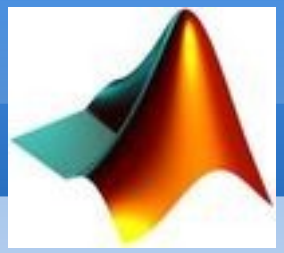


MATLAB é um ambiente de programação e computação numérica proprietário da empresa The MathWorks Incorporation (http://www.mathworks.com/).

Ele permite facilidade na manipulação de matrizes, criação de gráficos de funções e de dados, implementação de algorítmos, interfaces com outros programas e linguagens de programação.

UNICAMP

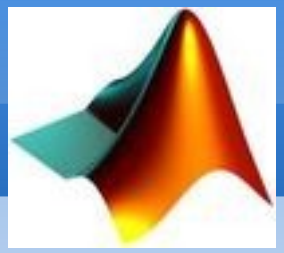
Introdução



O propósito geral do MATLAB é de permitir a realização de cálculos computacionais de forma mais fácil e mais “rápida” que com o uso de linguagens de programação convencionais como C, C++, e Fortran.

UNICAMP

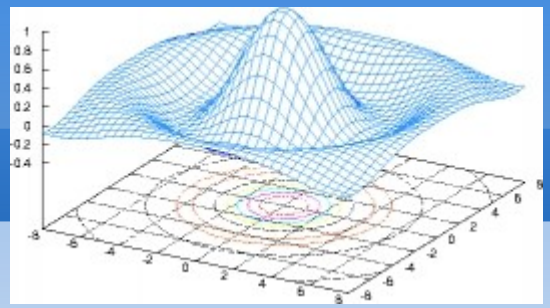
Introdução



Ele é usado por mais de um milhão de pessoas na indústria e no meio acadêmico. Seu preço básico é de em torno de US$2000 para uso comercial e sem ferramentas (toolboxes) adicionais, e possui uma licença de US$100 para uso acadêmico, mas com limitações em suas funcionalidades.

UNICAMP

Alternativas



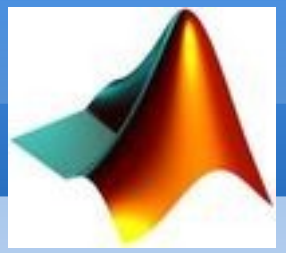
Existem projetos de software livre que desenvolvem programas similiares, com muitas das funcionalidades do matlab e até mesmo compatíveis com sua linguagem de programação.

Dentre eles, destaca-se o GNU Octave que pode ser baixado sem custo do site do projeto:

http://www.octave.org

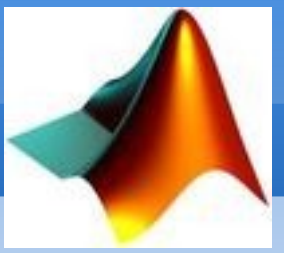
UNICAMP

Conceitos Básicos



UNICAMP

O Matlab como calculadora



```
>> 1+1
ans =
 2
```

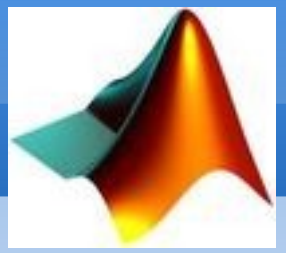
prompt

comandos do usuário

resposta (answer)

UNICAMP

O Matlab como calculadora

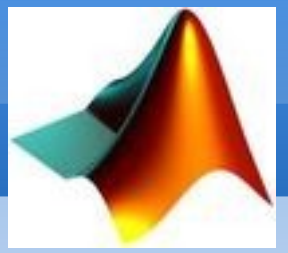


notação científica

```
>> 3*2.0+2^(1E1-9.0E0)
ans =
 8
```

notação científica

O Matlab como calculadora



```
>> 1+2E-1-1
ans =
 0.2000
```

menos subtração

notação científica

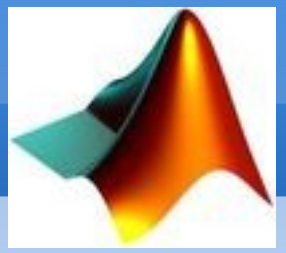
```
>> 3*-1-1
ans =
 -4
```

menos subtração

menos unário

UNICAMP

O Matlab como calculadora



```
>> 3^2 < 2^3
ans =
 0
```

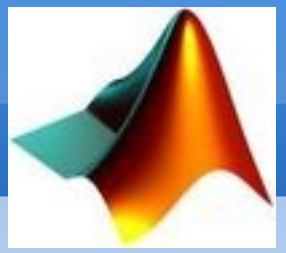
FALSE

```
>> 1+2–3*4/5<6
ans =
 1
```

TRUE

UNICAMP

O Matlab como calculadora

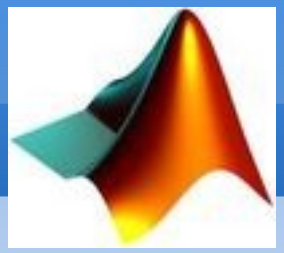


```
>> pi
ans =
 3.1416

>> sin(pi)
ans =
 1.2246e-16
```

UNICAMP

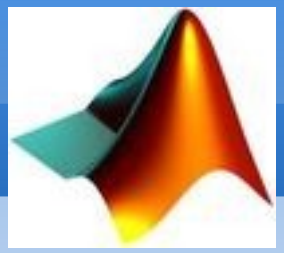
Precedência de Operadores



- Parênteses ()
- Transposta ('), potência (^)
- Mais unário (+), menos unário (-), negação lógica (~)
- Multiplicação (*), divisão (/)
- Adição (+), subtração (-)
- Operador dois pontos (:)

UNICAMP

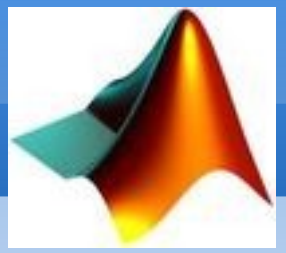
Precedência de Operadores



- Menor que (<), menor ou igual que (<=), maior que (>), maior ou igual que (>=), igual a (==), diferente de (~=)
- AND (&)
  OR (|)

UNICAMP

Precedência de Operadores

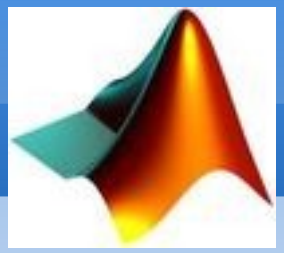


```
>> 3 + 4 * 2^(2 – 1) > 10
```

3 + 4 * 2^(2 – 1) > 10 (parênteses menos)
3 + 4 * 2^1 > 10 (potência)
3 + 4 * 2 > 10 (multiplicação)
3 + 8 > 10 (soma)
11 > 10 (comparação)
1

UNICAMP

Variáveis

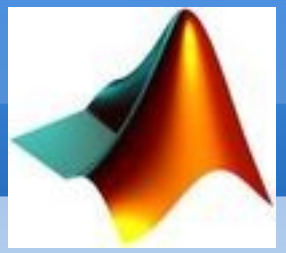


Variáveis podem ser facilmente criadas para armazenar dados e resultados através do operador de atribuição (=).

```
>> x = 2
x =
 2
>> y = x^2
y =
 4
```

UNICAMP

Variáveis

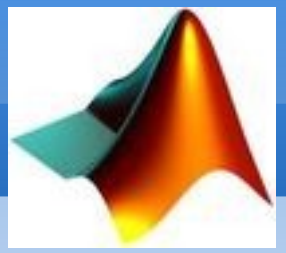


O comando *who* mostra as variáveis que estão atualmente na memória.

```
>> who
Your variables are:

x   y
```

Variáveis



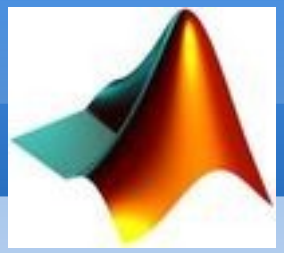
O comando *clear* apaga todas as variáveis da memória.

```
>> x = 1
x =
 1
>> y = 1;
>> clear
>> who
>>
```

ponto e vírgula no final do comando faz com que o resultado da atribuição não seja mostrado na tela

UNICAMP

Vetores

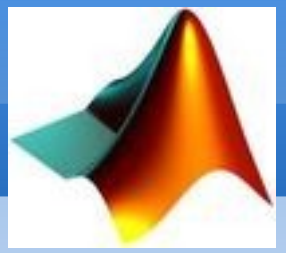


O vetor é definido através de colchetes e seus elementos devem ser separados por espaço ou vírgula para vetor linha ou ponto e vírgula ou enter para vetor coluna.

```
>> a = [1 2 3 4 5 6 9 8 7]
a =
 1  2  3  4  5  6  9  8  7
```

UNICAMP

Vetores

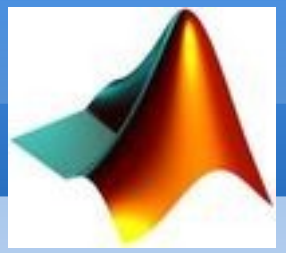


```
>> a = [1 2 3]
a =
 1 2 3

>> a = [1;2;3]
a =
 1
 2
 3
```

UNICAMP

Vetores

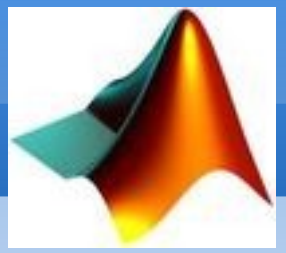


Criação de um vetor com elementos igualmente separados:

```
>> t = 0:2:10
t =
 0 2 4 6 8 10
```

UNICAMP

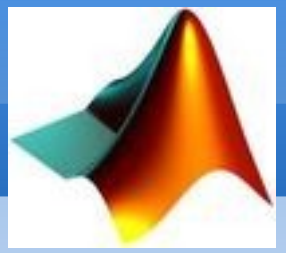
Vetores



Vetores podem ser manipulados e operados facilmente como variáveis comuns:

```
>> a = [1 2 3];
>> b = [ 4 5 6];
>> c = 1 + a
c =
 2 3 4
>> d = a + b
d =
 5 7 9
```

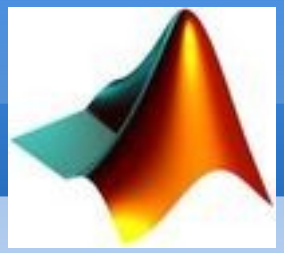
Vetores



O operador dois pontos (:) pode ser utilizado pra escolher uma determinada faixa de dados do vetor:

```
>> a = [11 12 13 14 15 16 17 18 19 20];
>> b = a(3:6)
b =
 13 14 15 16
>> c = b - a(1:4)
c =
 2 2 2 2
```

Vetores

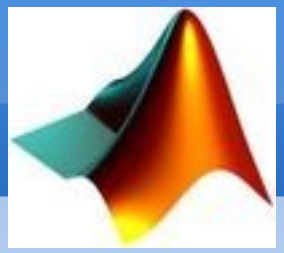


As funções do matlab podem ser aplicadas a elementos individuais de um vetor também.

```
>> x = 0:0.1:2*pi;
>> y = sin(x)
y =
 0.00000  0.09983  0.19867  0.29552
 0.38942  0.47943  0.56464  0.64422 ...
```

UNICAMP

Gráficos



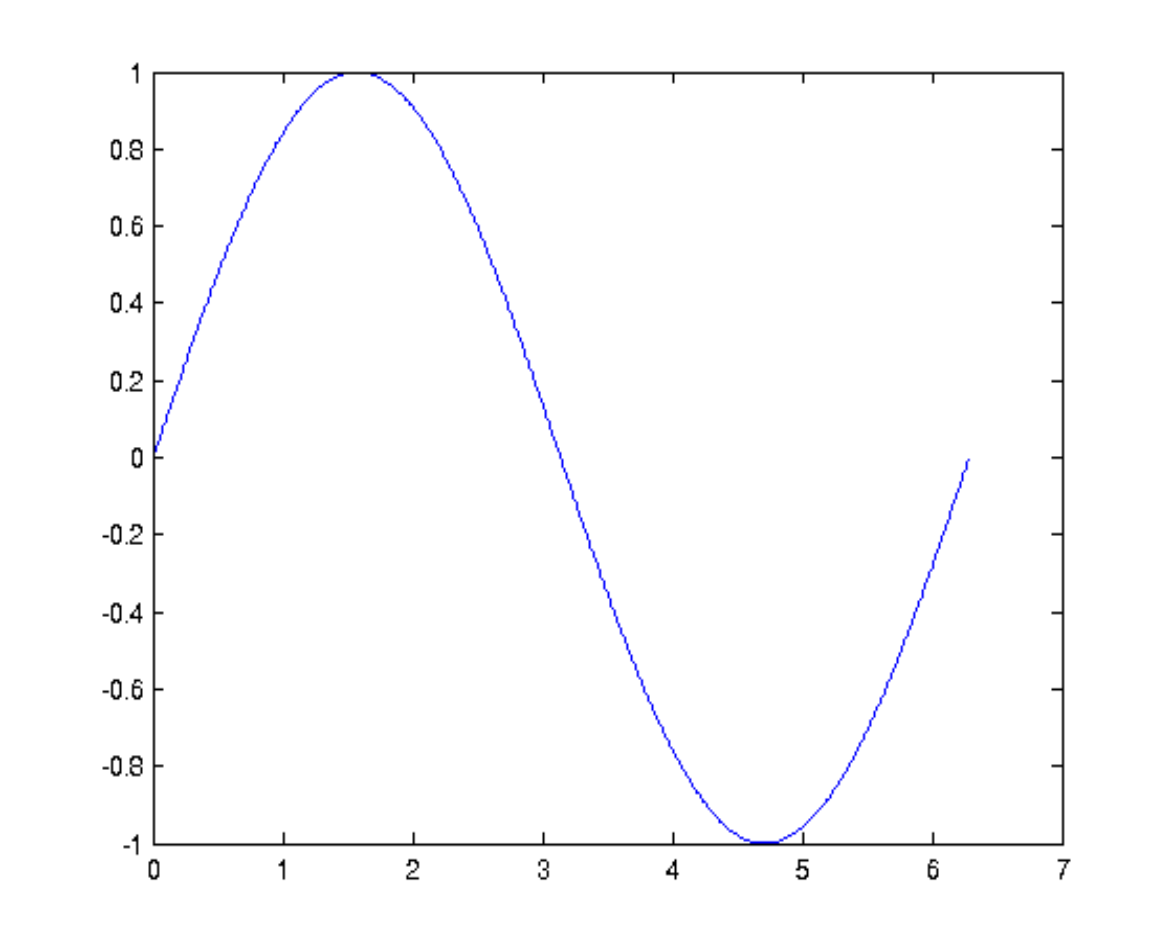
É fácil criar gráficos no Matlab.

Para plotar a função seno do exemplo anterior simplesmente utiliza-se o comando:

```
>> plot(x,y)
```

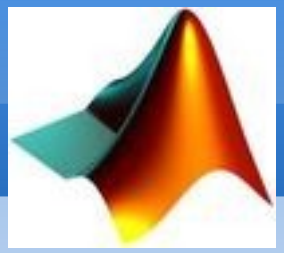
UNICAMP

Gráficos



UNICAMP

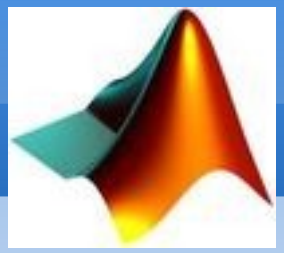
Funções e constantes especiais



O Matlab possui um conjunto de funções padrões como sin, cos, log, exp, sqrt e constantes especiais como pi, i e j para raíz de -1 (complexos).

```
>> sqrt(-1)
ans =
0 + 1.0000i
```

Funções e constantes especiais

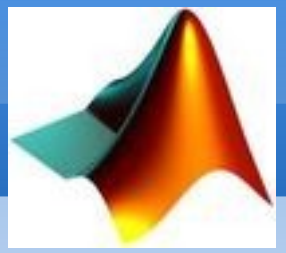


O Matlab possui um conjunto de funções padrões como sin, cos, log, exp, sqrt e constantes especiais como pi, i e j para raíz de -1 (complexos).

```
>> 5j^2
ans =
-25
```

UNICAMP

Ajuda

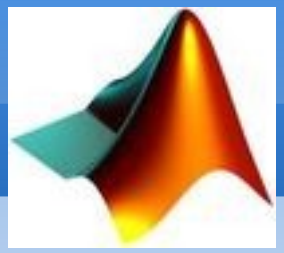


Um texto rápido de ajuda do Matlab sobre qualquer de suas funções pode ser solicitado com o comando *help* e uma janela de ajuda mais completa com o comando *doc*.

```
>> help sqrt
 SQRT   Square root.
    SQRT(X) is the square root of the elements of X. Complex results are produced if X is not positive.
```

UNICAMP

Ajuda

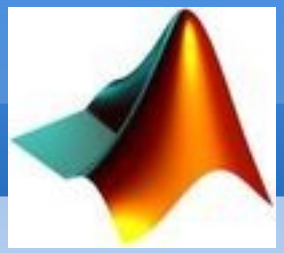


Um texto rápido de ajuda do Matlab sobre qualquer de suas funções pode ser solicitado com o comando *help* e uma janela de ajuda mais completa com o comando *doc*.

```
>> doc sqrt
```

UNICAMP

Polinômios



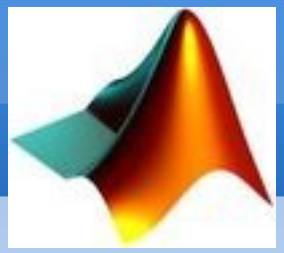
Polinômios são representados no Matlab através de vetores, onde cada elemento do vetor corresponde a um dos coeficientes do polinômio.

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

$$p=[a_n\ a_{n-1}\ \ldots\ a_1\ a_0]$$

UNICAMP

Polinômios



A ordem dos coeficientes é decrescente, como por exemplo:

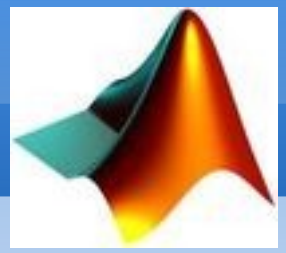
x = [2 5 -10]

corresponde ao polinômio:

$$2x^2+5x-10$$

UNICAMP

Polinômios



Se o polinômio não possuir um coeficiente intermediário o elemento equivalente no vetor deve ser preenchido com zero:

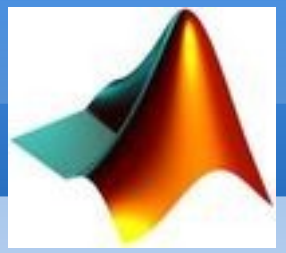
x = [1 4 0 -5]

corresponde ao polinômio:

$$x^3+4x^2-5$$

UNICAMP

Polinômios

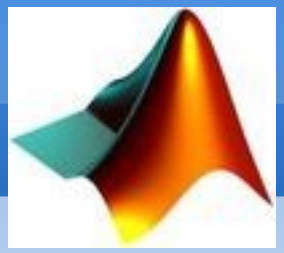


O valor de um polinômio para um dado ponto “x” pode ser avaliado pela função *polyval*.

```
>> z = polyval([1 4 0 -5],0)
z =
-5
```

UNICAMP

Polinômios

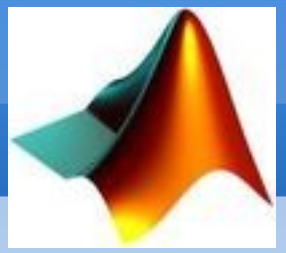


Os valores em um conjunto de pontos também podem ser calculados com esta função através de um vetor.

```
>> z = polyval([1 4 0 -5],0:0.1:1)
z =
-5.00000  -4.95900  -4.83200  -4.61300
-4.29600  -3.87500  -3.34400  -2.69700
-1.92800  -1.03100   0.00000
```

UNICAMP

Polinômios

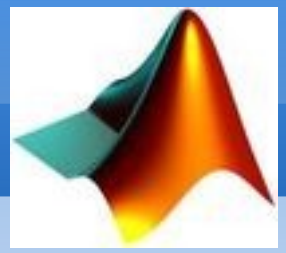


As raízes de um polinômio podem ser encontradas facilmente com o comando *roots*.

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0$$

UNICAMP

Polinômios

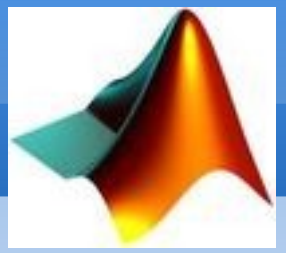


As raízes de um polinômio podem ser encontradas facilmente com o comando *roots*.

$$2x^3+x^2-2x+1=0$$

```
>> roots([2 1 -2 1])
ans =
-1.43756 + 0.00000i
0.46878 + 0.35785i
0.46878 - 0.35785i
```

Polinômios



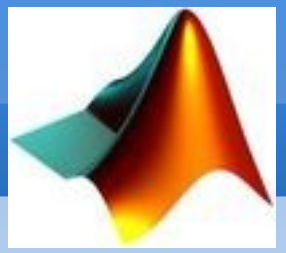
A multiplicação de polinômios é feita através da convolução de seus termos com a função conv:

```
>> x = [1 1];
>> y = [1 -1];
>> z = conv(x,y)
z =
1 0 -1
```

$$(x+1)(x-1)=x^2-1$$

UNICAMP

Polinômios



E de maneira similar a divisão é feita com o comando *deconv*, que retorna *duas* saídas, o resultado e o resto da divisão.

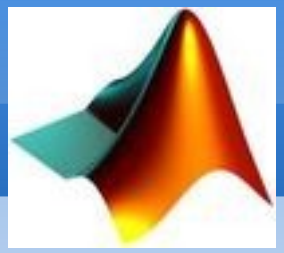
```
>> [x2, r] = deconv(z,y)
x2 =
 1 1
R =
 0 0 0
```

comando retorna duas saídas

$$\frac{(x^2-1)}{(x-1)} = (x+1)$$

UNICAMP

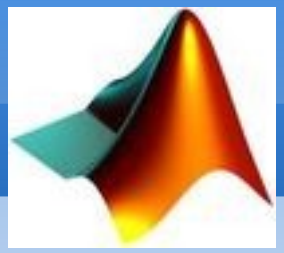
Polinômios



As operações de adição e subtração de polinômios de mesmo grau é simples, basta operar os vetores correspondentes:

```
>> x=[1 0 0];
>> y=[1 -1 5];
>> z=x+y
z =
 [2 -1 5]
```

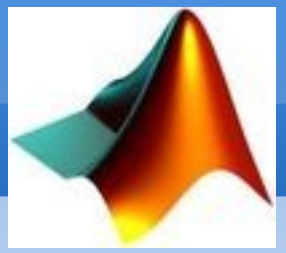
Polinômios



Quando os polinômios diferem em grau é necessário “completar” os termos do que tem menor grau para operar os vetores com soma e subtração:

```
>> x=[1 0 0 0];
>> y=[1 -1 5];
>> z=x+[0 y]
z =
 [1 1 -1 5]
```

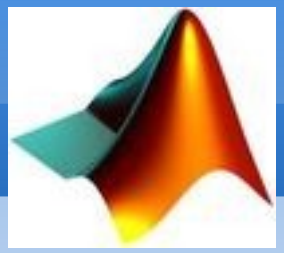
Matrizes



A criação de matrizes é equivalente a de vetores, utiliza-se os separadores espaço ou vírgula entre os elementos de uma mesma linha e ponto e vírgula ou enter para separar as linhas.

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

Matrizes

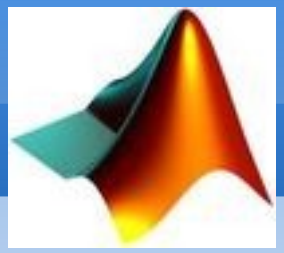


A transposta de uma matriz pode ser obtida com o operador apóstrofo (').

```
>> A = [1 2 3;4 5 6;7 8 9];
>> A'
ans =
 1 4 7
 2 5 8
 3 6 9
```

UNICAMP

Matrizes

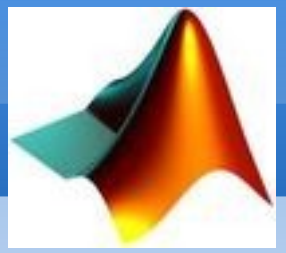


No caso de números complexos o operador apóstrofo retorna a transposta conjugada:

```
>> A = [(1+i) (2+2i);(3+3i) (4+4i)];
>> A'
ans =
 1-1i  3-3i
 2-2i  4-4i
```

UNICAMP

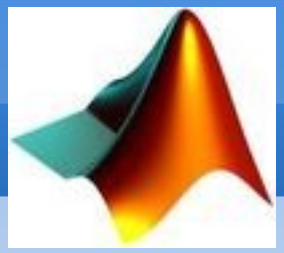
Matrizes



Neste caso, para apenas uma transposição dos termos utiliza-se ponto e apóstrofo (.'):

```
>> A = [(1+i) (2+2i);(3+3i) (4+4i)];
>> A.'
ans =
 1+1i  3+3i
 2+2i  4+4i
```

Matrizes



A simples multiplicação de uma matriz por uma constante pode ser feita com o uso do operador '*'.

```
>> A = [1 2 3; 0 1 1;2 1 0];
>> 2*A
 2 4 6
 0 2 2
 4 2 0
```

UNICAMP

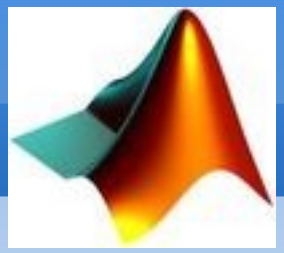
Matrizes

Multiplicação entre matrizes ou matrizes e vetores também é feita de forma similar, mas é necessário lembrar de verificar a ordem e os números de linhas e colunas.

```
>> A = [1 2 3; 0 1 1;2 1 0];
>> x = [2;4;5];
>> b = A*x
b =
 25
 9
 8
```

UNICAMP

Matrizes



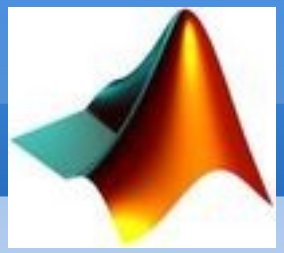
Divisão de matrizes pode ser feita através da multiplicação pela matriz inversa ou pelos operadores “/” e “\”.

```
>> x = b * inv(A);
>> x = b / A;
>> x = A \ b;
```

O operador “/” é equivalente a função inv. Já o operador “\” utiliza algorítmos mais eficientes que o de inversão de matrizes para solução do sistema linear Ax=b.

UNICAMP

Matrizes



Para multiplicar apenas os elementos de uma matriz utiliza-se o operador ponto asterísco ".*". Observação, as matrizes devem possuir o mesmo tamanho.

```
>> A = [1 2 3; 0 1 1;2 1 0];
>> A.*A
ans =
 1 4 9
 0 1 1
 4 1 0
```

UNICAMP

Potenciação

O mesmo se aplica para potenciação. Uma matriz quadrada pode ser elevada a um expoente para se multiplicar por ela mesma como multiplicação de matriz (*) ou como multiplicação elemento-a-elemento (.*).

```
>> A = [1 2 3; 0 1 1;2 1 0];
>> A^2
ans =
 7 7 5
 2 2 1
 2 5 7
```

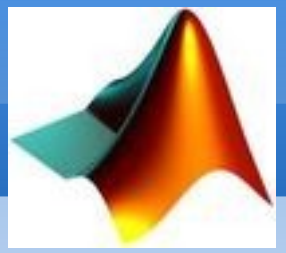
Potenciação

O mesmo se aplica para potenciação. Uma matriz quadrada pode ser elevada a um expoente para se multiplicar por ela mesma como multiplicação de matriz (*) ou como multiplicação elemento-a-elemento (.*).

```
>> A = [1 2 3; 0 1 1;2 1 0];
>> A.^2
ans =
 1 4 9
 0 1 1
 4 1 0
```

UNICAMP

Matrizes



Algumas funções importantes:

Matriz Inversa

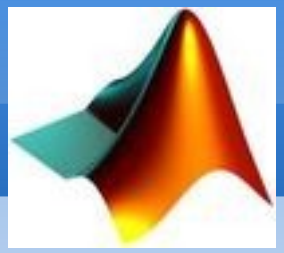
```
>> I = inv(A)
```

Autovalores

```
>> v = eig(A)
```

UNICAMP

Matrizes



Algumas funções importantes:

Polinômio Característico

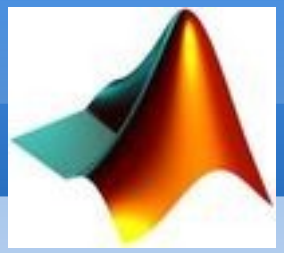
```
>> p = poly(A)
```

Raízes do Polinômio Característico

```
>> roots(p)
```

UNICAMP

Funções Personalizadas

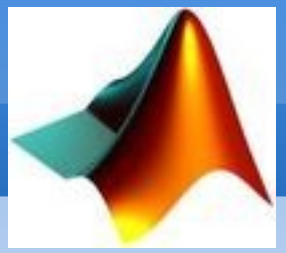


Formato geral:

```
function [r1,r2,...]=nome_funcao(p1,p2,...)
código que usa p1, p2,...
guarda resultados nas variáveis de saída
p1 = ...
p2 = ...
...
end
```

UNICAMP

Funções Personalizadas



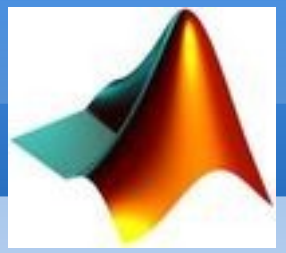
Exemplo: uma função fatorial simples

```
function ans=fatorial(x)
f=1;
if (x>1)
 f=x*fatorial(x-1);
end
ans=f;
end
```

UNICAMP

Funções Personalizadas

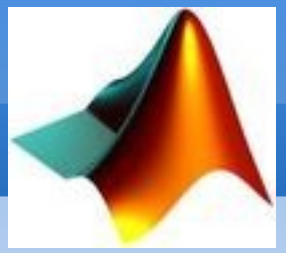
Exemplo: fatorial de uma matriz



```
function ans=mfatorial(x)
if (min(min(x))<0)
 error('A matriz nao pode conter elementos
negativos');
end
[a,b]=size(x);
f=max(x,ones(a,b));
if(sum(sum(f-1))>0)
 f=f.*mfatorial(f-1);
end;
ans=f;
end
```

UNICAMP

Controle de Fluxo

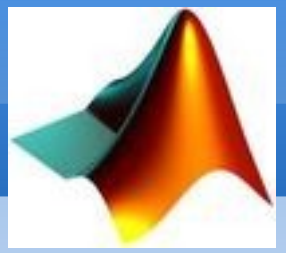


O comando de decisão IF:

```
if (condição)
   bloco de comandos
else
   bloco de comandos
end
```

UNICAMP

Controle de Fluxo

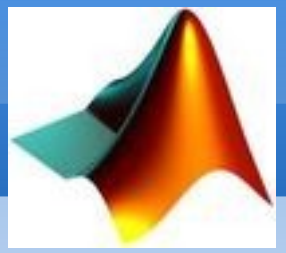


O comando de decisão IF:

```
if (x>0)
   y=1;
else
   y=0;
end
```

UNICAMP

Controle de Fluxo

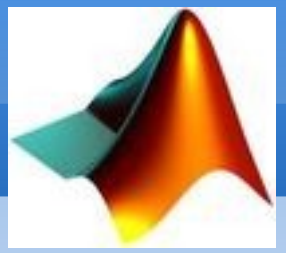


O comando de loop WHILE:

```
while (condição)
   bloco de comandos
end
```

UNICAMP

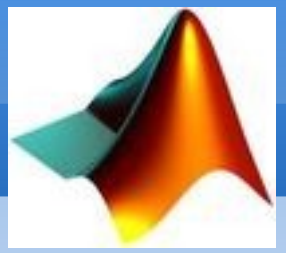
Controle de Fluxo



O comando de loop WHILE:

```
t=1.1;
while (t<10)
   t=t*t;
end
```

Controle de Fluxo

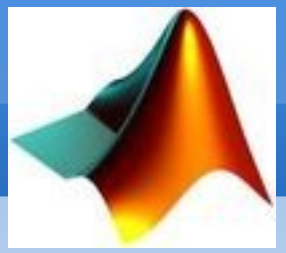


O comando de loop FOR:

```
for  variável = expressão
   bloco de comandos
end
```

UNICAMP

Controle de Fluxo

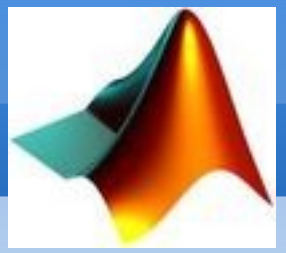


O comando de loop FOR:

```
for  t = 1:1:10
   y(t)=0.2*t^2;
end
```

UNICAMP

Referências



http://www.mathworks.com/access/helpdesk

http://www.octave.org/

http://www.engin.umich.edu/group/ctm/

UNICAMP