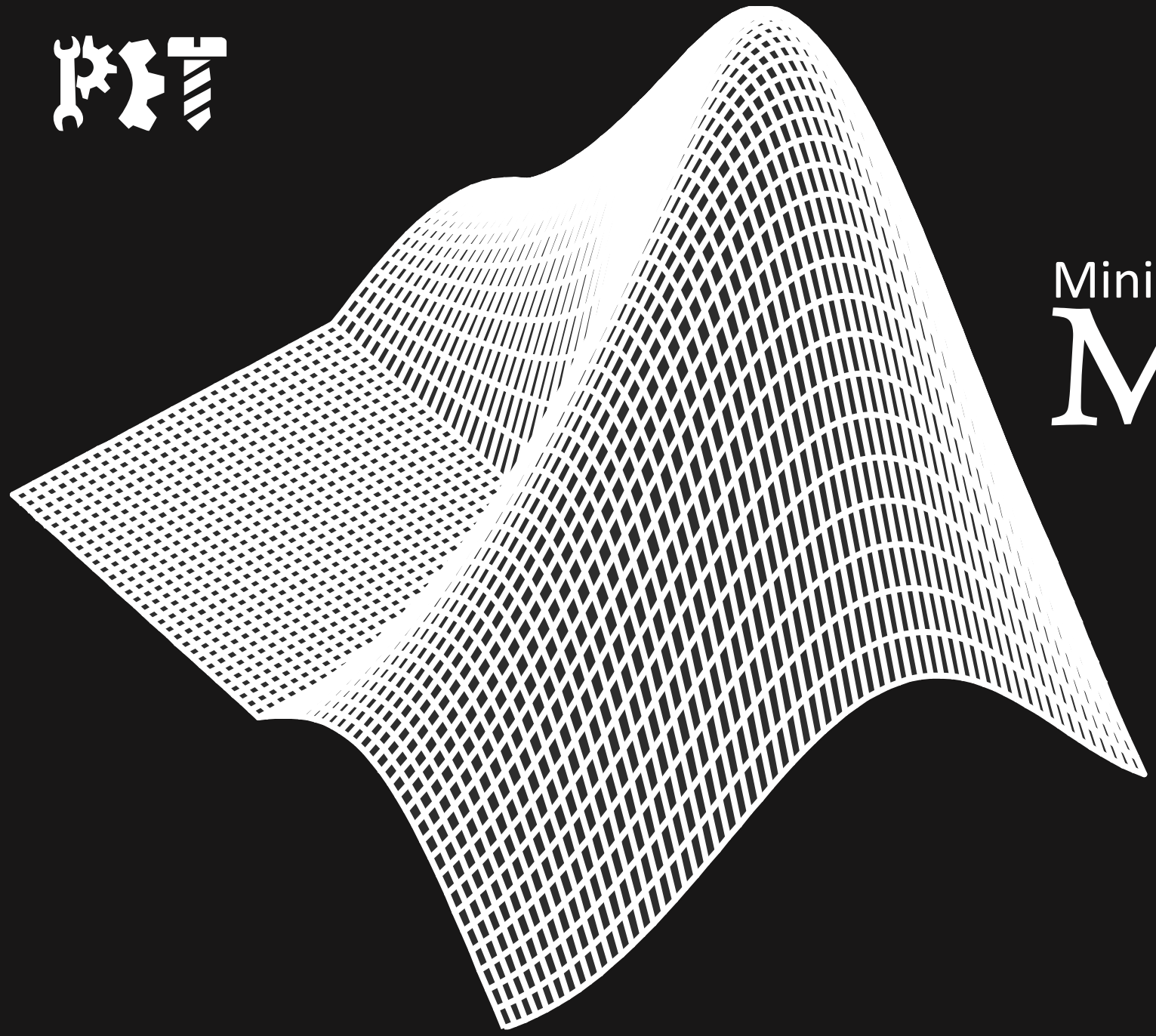


Minicurso

MATLAB

BÁSICO

MÓDULO 1

SEMANA DE ATUALIZACAO E TREINAMENTO DE ENGENHARIA

PET MECANICA

	segunda-feira	terça-feira	quarta-feira	quinta-feira		sexta-feira
8h às 12h	**ARDUINO** (Básico) *LabZorro*	**MATLAB** (Básico) *LabZorro*	**ARDUINO** (Básico) *LabZorro*	**VISITA TÉCNICA** Lingotamento contínuo *Arcelor Mittal*	**MATLAB** (Básico) *LabZorro*	**OFICINA DE JULIA** (Enfoque em EDO) *LabZorro*
12h às 13h	**ALMOÇO**	**ALMOÇO**	**ALMOÇO**	**ALMOÇO**		**ALMOÇO**
13h às 15h	**RODA DE CONVERSA** Tecnologia em materiais e fabricação *Auditório*	**RODA DE CONVERSA** Tecnologia em fluidos e térmica *Auditório*	**VISITA TÉCNICA** Laminação de tiras à quente *Arcelor Mittal*	**RODA DE CONVERSA** Inteligência Artificial e aplicações *Auditório*		**RODA DE CONVERSA** Mercado de trabalho e indústria *Auditório*
15h às 17h	**ANSYS** (Introdução) *LabZorro*	**EXCEL** (Básico + Intermediário) *LabZorro*		**ANSYS** (Introdução) *LabZorro*		**EXCEL** (Básico + Intermediário) *LabZorro*
17h às 19h				**PHOTOSHOP** (Introdução) *Sala do PET*		

Comissão do Minicurso

Vitorino Biazi

Robertson Junior

Jhonata Moraes

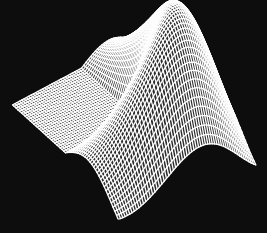
Sávio Alves

Gabriel Nunes

Thiago Bragança

Roteiro do primeiro módulo

- Introdução
- Área de trabalho
- Variáveis
- Operações e funções matemáticas
- Números complexos
- Scripts, Rotinas
- Comandos de fluxo

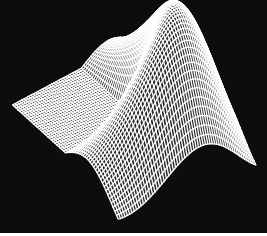


Declaração de Variáveis

→ Nome de variável:

Letras maiúsculas e minúsculas se diferenciam

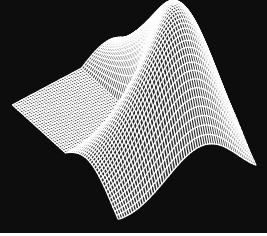
Permitido números, letras e underline.



Declaração de Variáveis

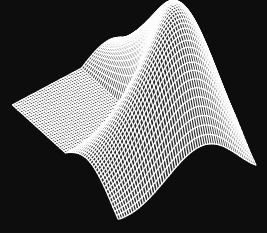
→ Variáveis reservadas

Ans	Variável onde são guardados os resultados de operações não atribuídas a uma variável específica - ans, diminutivo de answer.
pi	Valor de π = 3.1416.
Eps	Unidade de arredondamento da máquina, i.e., o menor valor que adicionado a 1 representa um número maior que 1
Flops	Contador do número de operações efetuadas. Estamos a falar de operações em vírgula flutuante.
Inf	Representa



Operações Aritméticas

Tabela 1.1 - Operações aritméticas entre dois escalares		
Operação	Forma Algébrica	MATLAB
Adição	a + b	a + b
Subtração	a - b	a - b
Multiplicação	a x b	a*b
Divisão pela Direta	a ÷ b	a/b
Divisão pela Esquerda	b ÷ a	a\b
Exponenciação	a^b	a^b
Radiciação	$\sqrt{a}$	sqrt(a)
Fatorial	a!	factorial(a)



Operações Aritméticas

```
>> x = 5

x =

     5
```

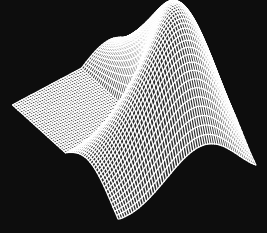
```
>> 2 + 1

ans =

     3
```

```
>> 7 - 9

ans =

    -2
```

```
>> 7*2

ans =

    14
```

```
>> 11/2

ans =

    5.500
```

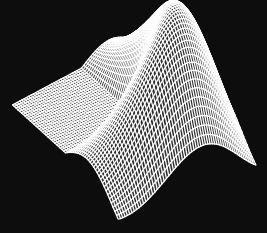
```
>> 3^3

ans =

    27
```



Hierarquia de Operações

→ Assim como na matemática, no MATLAB existe uma ordem de prioridade para execução das operações aritméticas. Abaixo segue uma tabela com essa ordem.

Hierarquia das Operações Aritméticas	
Prioridade	Operação
1ª	Parênteses
2ª	Exponenciação, esquerda à direita
3ª	Multiplicação e Divisão, esquerda à direita
4ª	Adição e Subtração, esquerda à direita

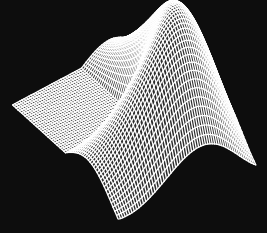


Hierarquia de Operações

→ Um exemplo é a equação de Bháskara

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
>> a = 5;
>> b = 5;
>> c = 5;
>> x = -b+sqrt(b^2-4*4*a*c)/2*a;
ans =
      25.8388
>> x = (-b+sqrt(b^2-4*4*a*c))/(2*a);
ans =
      0.9136
```



Formatos numéricos

→ format short (padrão)

```
0.3333
```

→ format long

```
0.33333333333333333
```

→ format shorte%

```
3.3333e-01
```

→ format longe%

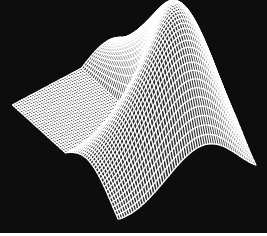
```
3.3333333333333333e-01
```

→ format hex

```
3fd5555555555555
```

→ format bank

```
0.33
```



Comandos básicos – Limpar e ajuda!

→ clc

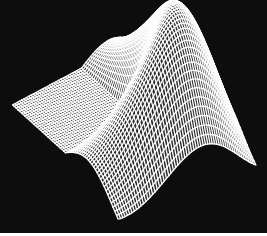
Limpa os comandos do Command Window.

→ clear nome_variável

Apaga a variável “nome_variável”.

→ clear all

Apaga todas as variáveis do Workspace.

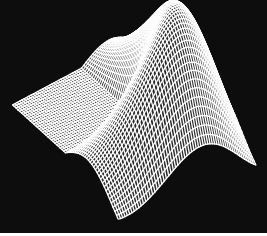


Comandos básicos – Limpar e ajuda!

→ help nome_comando

Exibe uma explicação do funcionamento do comando

```
>> help sin
sin - Sine of argument in radians
This MATLAB function returns the sine of the
elements of X.
Y=sin(X)
Reference page for sin
See also asin, asind, sind, sinh.
Other uses of sin
fixedpoint/sin, symbolic/sin
```



Comandos básicos – Exibir, salvar e carregar

→ who

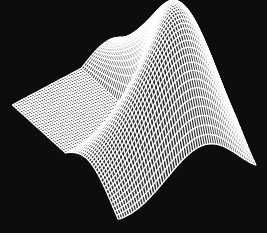
Exibe quem são as variáveis atualmente no Workspace.

```
>> who
your variables are:
a b
```

→ whos

Além de exibir as variáveis mostra também a dimensão, número de bytes e a classe da variável.

```
>> whos
Name   Size   Bytes   Class
a      1x1        8   double
b      2x3       48   double
```



Comandos básicos – Exibir, salvar e carregar

→ save nome_ficheiro

Salva as variáveis do Workspace em formato binário ou formato ascii.

→ load nome_ficheiro

Abri no Workspace variáveis salvas em um arquivo.

Funções f(x)

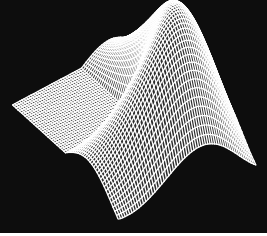
→ trigonométricas

asin(x) acos(x) atan(x)

acsc(x) asec(x) acot(x)

asinh(x) acosh(x) atanh(x)

acsch(x) asech(x) acoth(x)



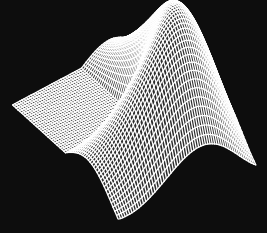
Funções f(x)

→ Exemplos

```
>> x = pi/3;
>> sin(x)
ans =
      0.8660
```

```
>> x = pi/3;
>> sinh(x)
ans =
      1.2494
```

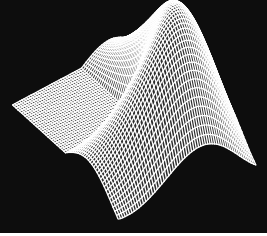
```
>> x = 0.8860;
>> asin(x)
ans =
      1.0471
```



Funções f(x)

→ conversão (ângulo)

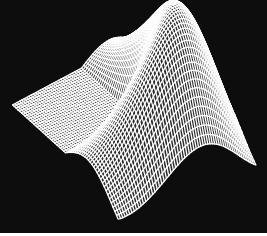
```
degtorad(180)
radtodeg(pi/3)
```



Funções f(x)

→ Trigonométricas (grau em degree)

```
sind(x)  cosd(x)  tand(x)
```

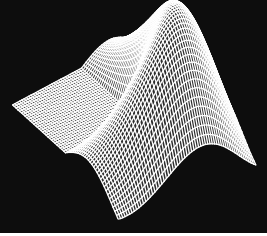


Funções f(x)

→ conversão (ângulo)

```
>> degtorad(180)

ans =

       3.1416
```

```
>> radtodeg(pi/3)

ans =

      60.000
```



Funções f(x)

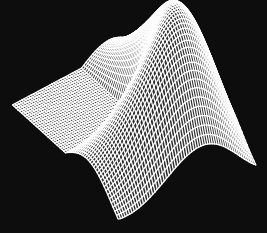
→ Exponencial e Logarítmica

`exp(x)`
Potenciação com o número de Euler como base

`sqrt(x)`
Raiz quadrada de x

`log(x)`
Logaritmo neperiano

`log10(x)`
Logaritmo de x na base 10



Funções f(x)

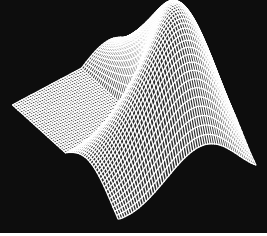
→ Exponencial e Logarítmica

```
>> exp(2)

ans =

       7.3891
```

```
>> log(exp(2))

ans =

       2
```

```
>> log10(1000)

ans =

       3
```

```
>> sqrt(64)

ans =

       8
```



Funções f(x)

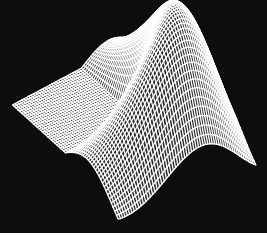
→ Arredondamento

round(x)
Arredonda um número decimal para o inteiro mais próximo.

floor(x)
Arredonda um número decimal para o inteiro anterior.

ceil(x)
Arredonda um número decimal para o inteiro posterior.

rem(x,y)
Obtém-se o valor do resto da divisão de x por y.



Funções f(x)

→ Arredondamento

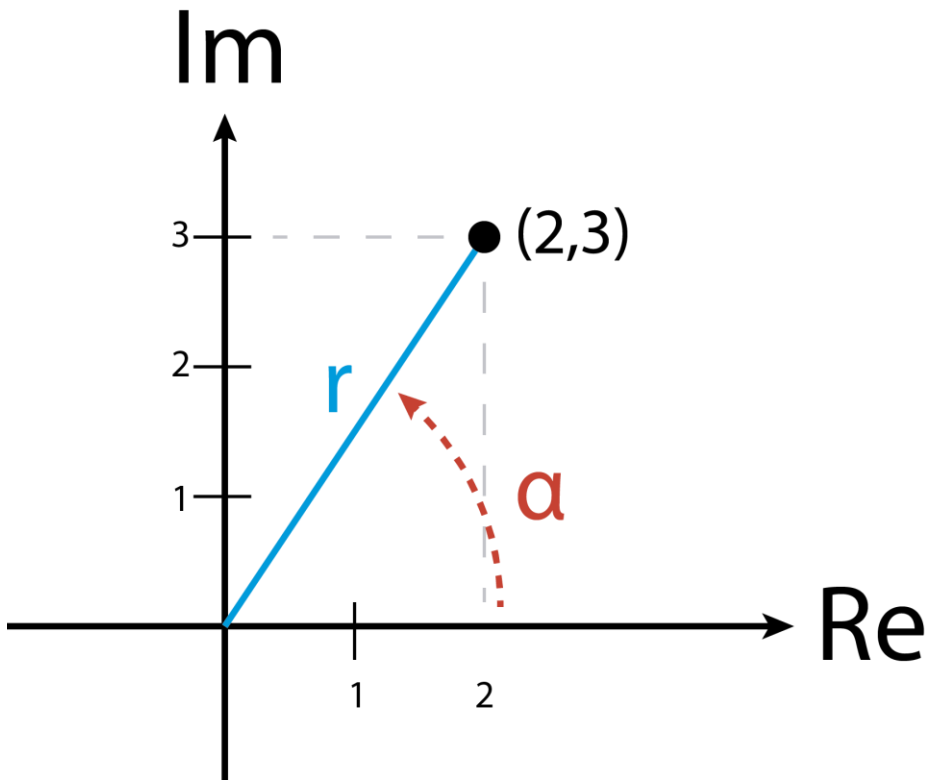
```
>> round(1.43)

ans =

     1
```

```
>> floor(1.5)

ans =

     1
```

```
>> ceil(1.5)

ans =

     2
```

```
>> rem(8,5)

ans =

     3
```

→ Números Complexos




```
x=2+3i
```

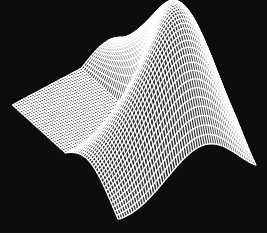
→ Números Complexos

Coordenadas retangulares para polares:

$$r = \sqrt{x^2 + y^2} \qquad \alpha = tan^{-1}\left(\frac{y}{x}\right)$$

Coordenadas polares para retangulares:

$$x = r \cdot \cos(\alpha) \qquad y = r \cdot sen(\alpha)$$

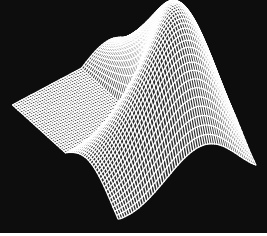


Números Complexos

→ Números Complexos

```
>> x = 3+4i

X =

       3.0000 + 4.0000i

>> y = 5+3j

y =

       5.0000 + 3.0000i
```

```
>> z = 3+sqrt(-1)

z =

       3.0000 + 1.0000i
```

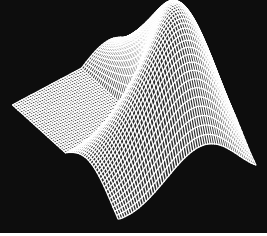


Números Complexos

→ Operação Aritmética

```
>> c1 = 3+2i

c1 =

        3.0000 + 2.0000i

>> c2 = 5-i

c2 =

        5.0000 - 1.0000i

>> c1+c2

ans =

        8.0000 + 1.0000i
```

```
>> c3 = 4

c3 =

        4

>> c1-c2/c3

ans =

        1.7500 + 2.2500i

>> i^2

ans =

        -1
```

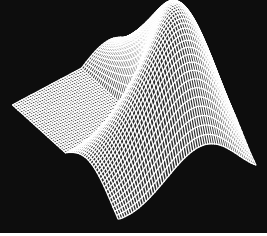


Números Complexos

→ abs(x)

Calcula o valor absoluto

```
>> x = [1+2i 2+3i]

x =

      1.0000 + 2.0000i    2.0000+3.0000i

>> abs(x)

ans =

      2.2361    3.6056
```

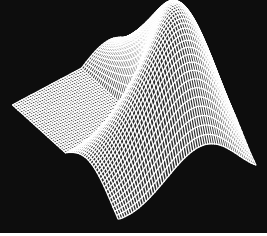


Números Complexos

→ angle(x)

Retorna o valor do ângulo de fase (em radianos) de um número complexo.

```
>> x = [3 2i]

x =

       3.0000 + 2.0000i

>> angle(x)

ans =

       0.5880
```

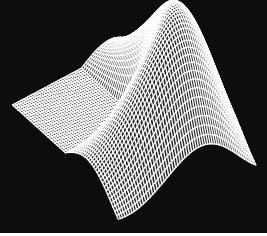


Números Complexos

→ conj(x)

Retorna o valor do complexo conjugado de um dado número complexo x.

```
>> conj(1+4i)

ans =

     1.0000 - 4.0000i
```

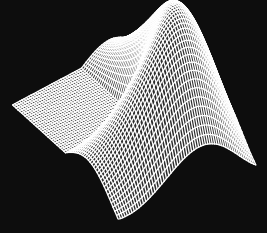


Números Complexos

→ imag(x)

Retorna o valor da parte imaginária de um dado número complexo x.

```
>> imag(4+5i)

ans =

     5
```

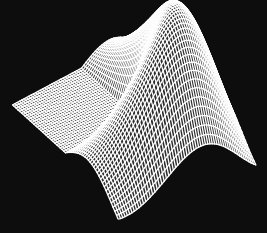


Números Complexos

→ real(x)

Retorna o valor da parte real de um dado número complexo x.

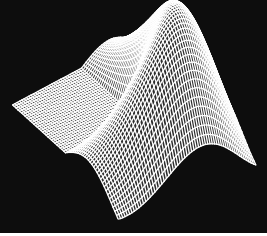
```
>> real(4+5i)

ans =

     4
```



Rotinas ou Arquivos M-Files - Scripts

Conhecidos como:

- → M-Files
- → Rotinas
- → Scripts Files



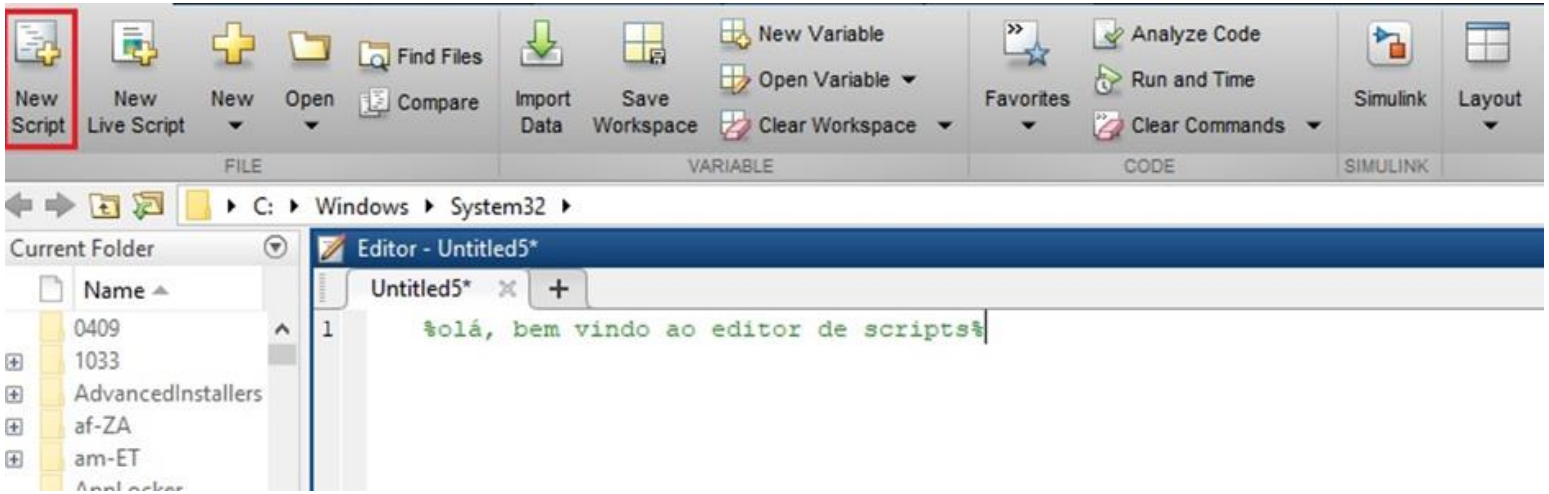
Rotinas ou Arquivos M-Files - Scripts

Características dos Scripts:

- → Os comandos são executados na ordem em que eles foram escritos.
- → Em scripts que possuem comandos de saída, essa saída é mostrada no Command Window
- → Podem ser editados e também serem executados várias vezes
- → Devem ser salvos antes de serem executados

Rotinas ou Arquivos M-Files - Scripts

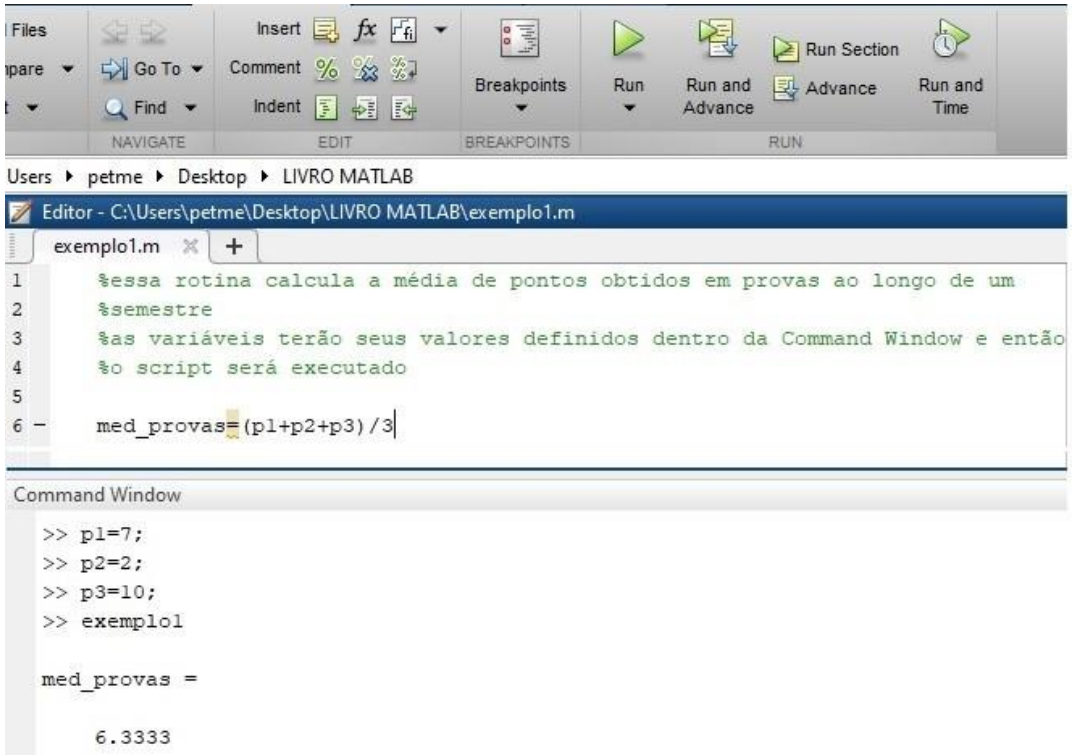
Criando e salvando um script:



Rotinas ou Arquivos M-Files - Scripts

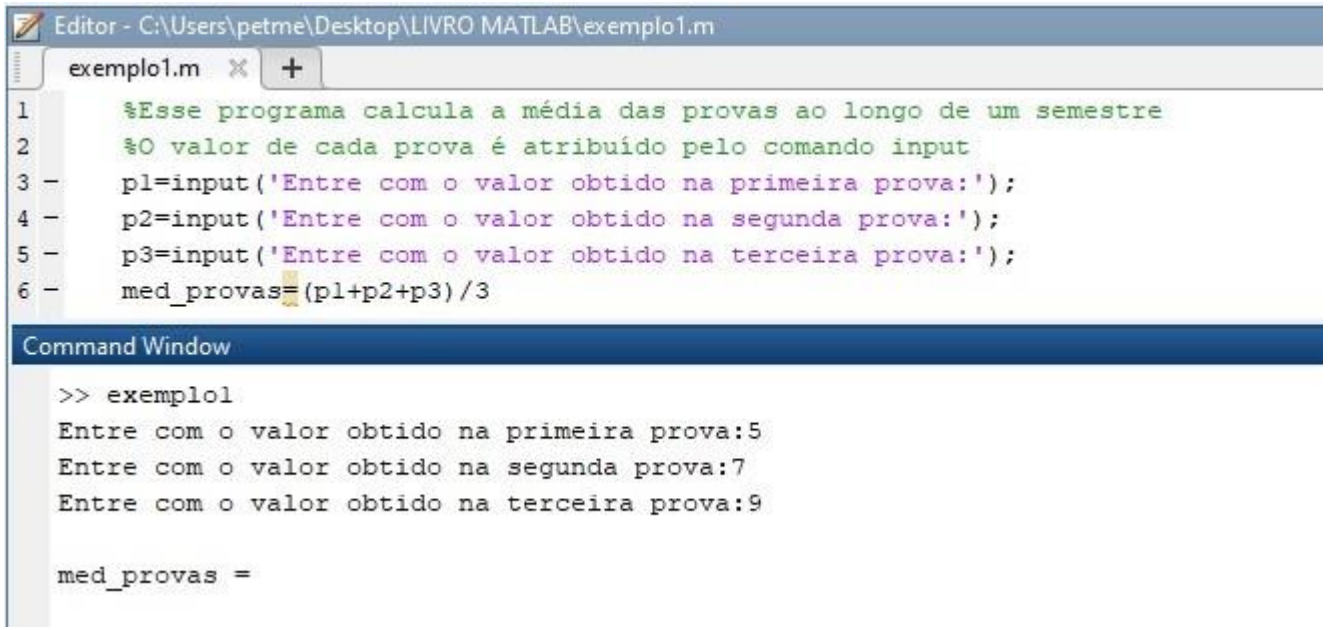
Definindo variáveis

Podemos usar variáveis que foram definidas no Command Window



Rotinas ou Arquivos M-Files - Scripts

As variáveis podem também serem declaradas no script e terem os seus valores atribuídos no Command Window.




```
a = input('Mensagem');
fprintf('Mensagem');
disp('Mensagem');
```

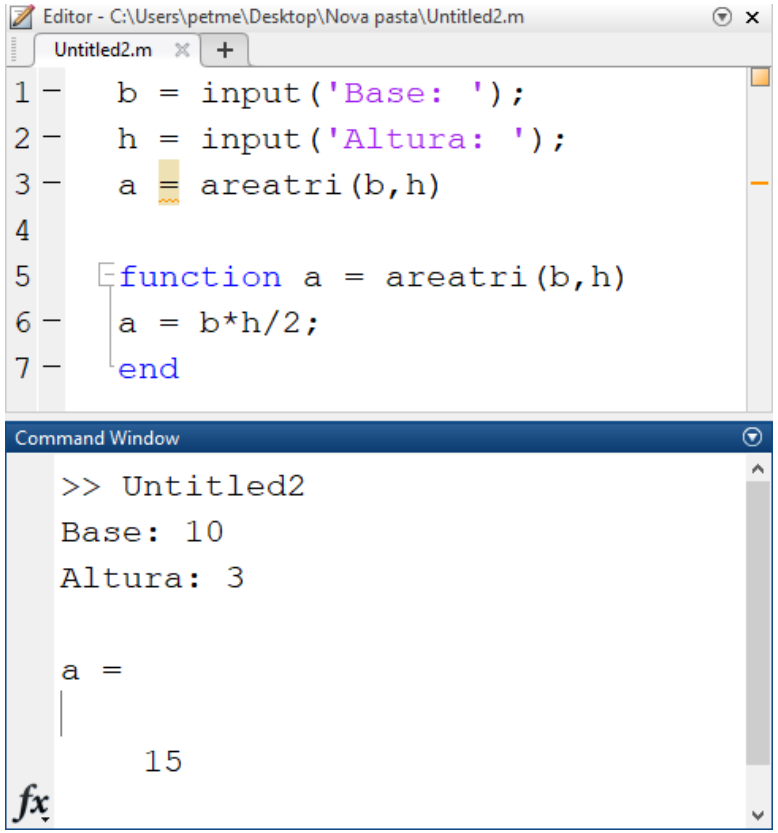
Rotinas ou Arquivos M-Files - Scripts

Funções

Assim como em outras linguagens de programação, o MATLAB permite a criação de funções. Podem ser declaradas no fim do script ou em um script à parte.

```
function saida = nome (parametros)
<comandos>
end
```

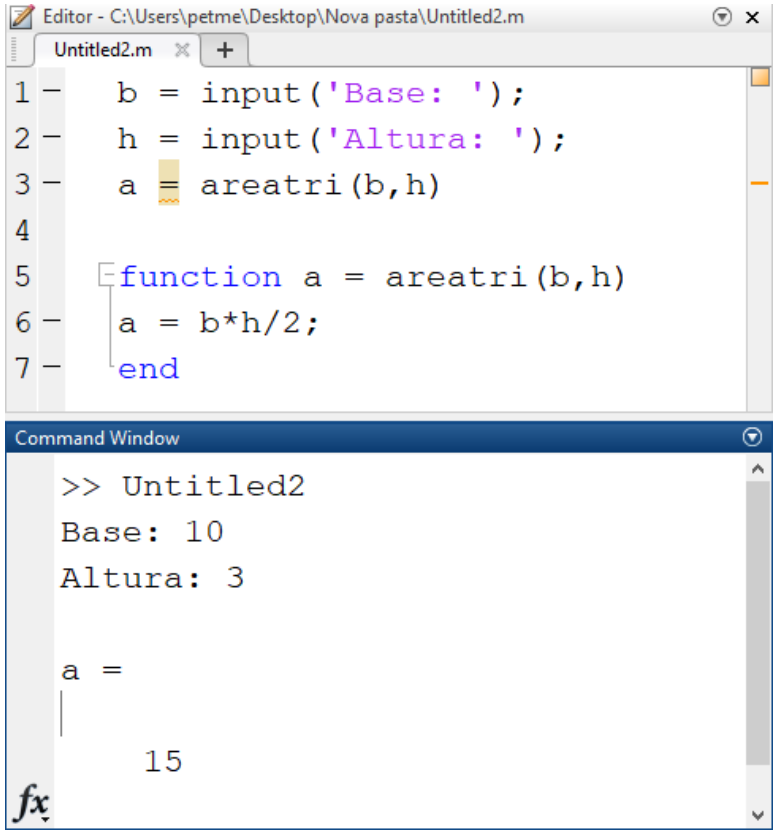
Para executar a função, chame o nome da função e os parâmetros.

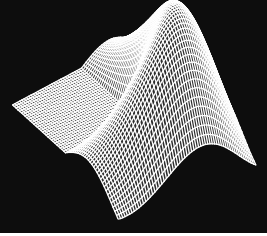


Rotinas ou Arquivos M-Files - Scripts

Funções

Para salvar em um arquivo externo, salve o arquivo com o mesmo nome da função.

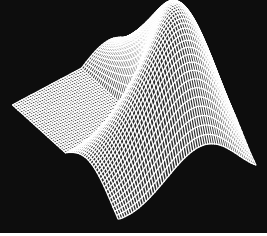




Operadores lógicos e Relacionais – Operadores relacionais

Os operadores relacionais servem para fazer a comparação entre dois dados. No caso do MATLAB, é possível realizar a comparação de duas matrizes de mesmo número de linhas e colunas ou para comparar uma matriz e um escalar.

Operador	Significado
<	menor que
<=	menor ou igual
>	maior que
>=	maior ou igual
==	igual
~=	diferente (não igual)



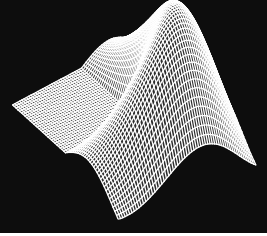
Operadores lógicos e Relacionais – Operadores relacionais

→ Exemplo

Se a comparação for verdadeira o Matlab irá retornar 1. Caso não seja, o programa irá retornar 0.

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

A =

     1     2     3
     4     5     6

>> B = [0 6 3; 7 6 6]

B =

     0     6     3
     7     6     6

>> c = 4

c =

     4
```

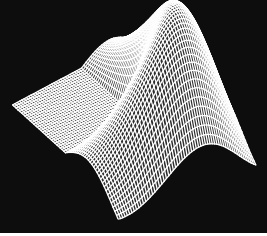
```
>> A < B

ans =

     0     1     0
     1     1     0

>> A >= c

ans =

     0     0     0
     1     1     1
```



Operadores lógicos e Relacionais – Operadores lógicos

É possível combinar duas ou mais comparações (operações relacionais) utilizando os operadores lógicos

Operador	Significado
&	E
|	OU
~	NÃO

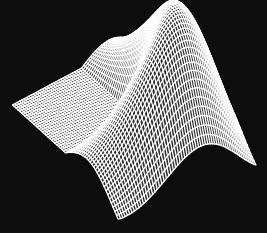


Operadores lógicos e Relacionais – Operadores lógicos

→ E

Quando duas expressões forem combinadas com E (&) seu resultado só será 1 (*verdadeiro*) se o resultado das duas expressões individualmente for 1 (verdadeiro). Caso uma delas seja 0, o resultado da operação & é 0 (*falso*).

Combinação com E	Falso	Verdadeiro
Falso	0	0
Verdadeiro	0	1

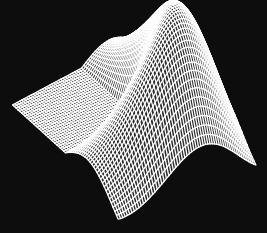


Operadores lógicos e Relacionais – Operadores lógicos

→ OU

O resultado combinação de duas expressões utilizando com OU (|) será 1 se pelo menos o resultado individual de uma das expressões for 1. OU é 0 apenas se as duas expressões combinadas forem 0.

Combinação com OU	Falso	Verdadeiro
Falso	0	1
Verdadeiro	1	1

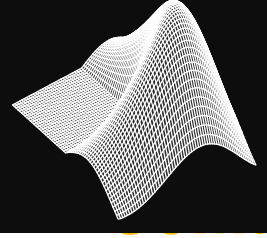


Operadores lógicos e Relacionais – Operadores lógicos

→ NÃO

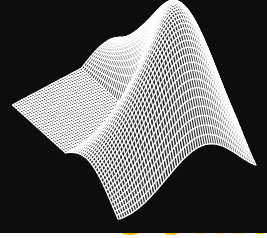
O papel da operação NÃO (~) é inverter o resultado de toda a expressão. Ou seja, se o resultado da expressão for 1, utilizar a operação NÃO a transforma para 0, e vice-versa.

```
>> A = 1;
>> B = 2;
>> C = 4;

>> A > B & C > B

ans =

     0

>> A > B | C > B

ans =

     1
```



Comandos de fluxo

Os comandos de fluxo são usados para desviar o fluxo natural do código, ou seja, executar ou não uma ação dependendo de uma condição ou executar uma mesma ação repetidas vezes.



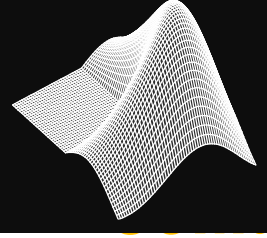
Comandos de fluxo – For

→ For

O for é um comando de loop é utilizado quando se quer realizar uma série de comandos por um número de vezes fixo e predefinido.

Estrutura básica do comando

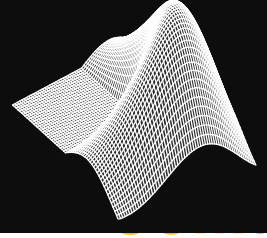
```
for i = 1:10

<comandos>

end
```



Comandos de fluxo – For

→ Exemplo

```
for i=1:5
    for j = 1:5
        A(i,j) = i + j;
    end
end

A =

     2     3     4     5     6
     3     4     5     6     7
     4     5     6     7     8
     5     6     7     8     9
     6     7     8     9    10
```



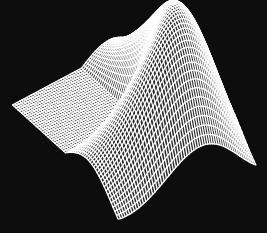
Comandos de fluxo – While

→ While

O loop while é executado enquanto o resultado de condição predeterminada for verdadeira.

Estrutura básica do comando

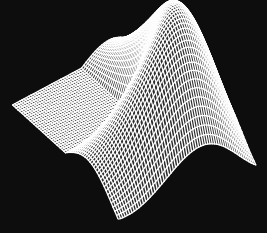
```
while <condição>

<comandos>

end
```



Comandos de fluxo – While

→ Exemplo

```
a = 1;
b = 10;
i = 1;
while b >= a
A(i,i) = i^2;
b = b-i;
i = i+1;
end

A =

     1     0     0     0
     0     4     0     0
     0     0     9     0
     0     0     0    16
```



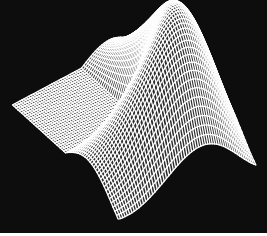
Comandos de fluxo – If-Else

→ If-Else

O operador if-else serve para executar um bloco de comandos se uma condição for verdadeira. Senão, outro bloco de comandos será executado.

Estrutura básica do comando

```
if <condição>
<comandos1>

else
<comandos2>
end
```



Comandos de fluxo – If-Else

→ Exemplo

```
a = rand(1,1)
if a < 0.5
    A = linspace(0,70,8)
else
    A = linspace(1,50,8)
end

a =

    0.8491


A =

     1     8    15    22    29    36    43    50
```