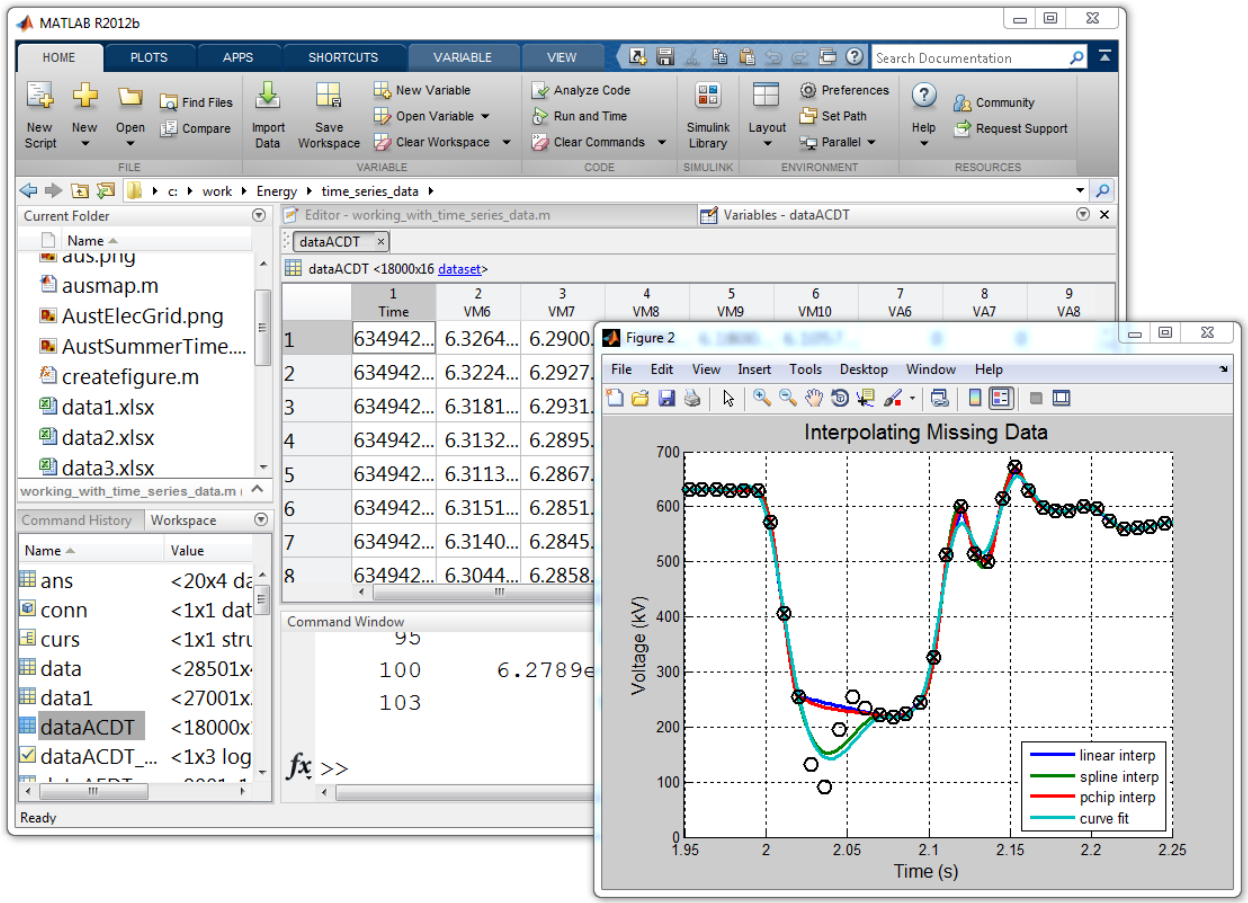
MATLAB Avançado
Aula 1

Melissa Weber Mendonça
melissa.mendonca@ufsc.br

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

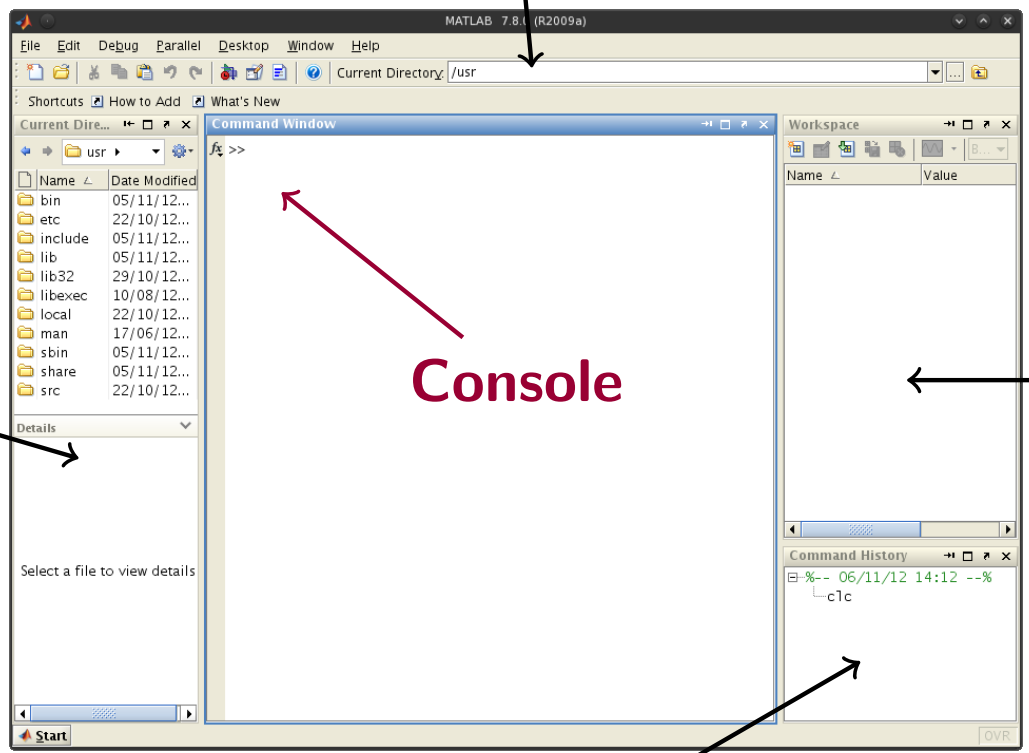
O que é o MATLAB?

Linguagem computacional de alto nível e um ambiente interativo para computação numérica, visualização e programação.



UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Console: Modo Interativo



UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Scripts

Os comandos podem ser entrados diretamente no *console* do MATLAB, ou escritos, em sequência, dentro de um arquivo com extensão `.m` chamado *script*.

Sequências de trabalho possíveis:

1. Escrever os comandos no console em sequência, obtendo as respostas a cada comando.
2. Usando um script:
   - **a)** Escrever os comandos em um arquivo no seu editor de texto preferido (Notepad) e salvar esse arquivo com extensão `.m`;
   - **b)** Ir até a janela do console do MATLAB;
   - **c)** Digitar o nome do arquivo em que você digitou os comandos **sem o** `.m`.

Estruturas de Dados e MATLAB Básico

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Comandos Básicos

- Operações Aritméticas: `+`, `-`, `*`, `/`, `^`

Comandos Básicos

- Operações Aritméticas: `+`, `-`, `*`, `/`, `^`
- **Funções** matemáticas: `sin(pi)`, `abs(-3)`

Comandos Básicos

- Operações Aritméticas: `+`, `-`, `*`, `/`, `^`
- **Funções** matemáticas: `sin(pi)`, `abs(-3)`
- `date`

Comandos Básicos

- Operações Aritméticas: `+`, `-`, `*`, `/`, `^`
- **Funções** matemáticas: `sin(pi)`, `abs(-3)`
- `date`
- `clear` ou `clc`

Comandos Básicos

- Operações Aritméticas: `+`, `-`, `*`, `/`, `^`
- **Funções** matemáticas: `sin(pi)`, `abs(-3)`
- `date`
- `clear` ou `clc`
- `help`

Variáveis

Para atribuir um valor a uma variável no MATLAB, basta digitarmos

```
>>  variavel = valor
```

(não é preciso declarar variáveis no MATLAB).

Introdução

Assim, para criar diferentes tipos de variável, usamos os seguintes comandos:

- Números (inteiros ou reais):

```
>>  a = 1
>>  b = 3.14
>>  pi
>>  h = 1e-2
```

Introdução

Assim, para criar diferentes tipos de variável, usamos os seguintes comandos:

- Números (inteiros ou reais):

```
>>  a = 1
>>  b = 3.14
>>  pi
>>  h = 1e-2
```

- Vetores:

```
>>  v = [1,2,3]
>>  v = [1 2 3]
>>  u = [1;2;3]
```

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Introdução

Assim, para criar diferentes tipos de variável, usamos os seguintes comandos:

- Números (inteiros ou reais):

```
>>  a = 1
>>  b = 3.14
>>  pi
>>  h = 1e-2
```

- Vetores:

```
>>  v = [1,2,3]
>>  v = [1 2 3]
>>  u = [1;2;3]
```

- Matrizes:

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

Introdução

Assim, para criar diferentes tipos de variável, usamos os seguintes comandos:

- Números (inteiros ou reais):

```
>>  a = 1
>>  b = 3.14
>>  pi
>>  h = 1e-2
```

- Vetores:

```
>>  v = [1,2,3]
>>  v = [1 2 3]
>>  u = [1;2;3]
```

- Matrizes:

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

- Texto:

```
>>  texto = 'Aqui vai meu texto.'
```

Dicas

Para que o resultado *não* seja mostrado ao final da operação, use `;` ao final do comando.

Exemplo:

```
>>  sin(pi)
>>  sin(pi);
```

Em um script, podemos *comentar* nosso código, usando o símbolo `%`:

```
a = 1;
% Agora, a variável a tem valor 1.
```

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Dicas

Para que o MATLAB imprima o valor de uma variável numérica (escalar, vetor, matriz etc), digite o nome da variável no console e pressione *Enter*.

```
>>  pi
```

Para mostrar um texto, use o comando **disp**.

```
>>  disp('Oi!')
```

Para entrar com comandos longos em várias linhas, use ...

```
>>  soma = 1+2+3+4+5 ...
      +6+7+8+9+10
```

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

MATLAB Básico: Vetores

```
>>  v = [1 3 5]
```

MATLAB Básico: Vetores

```
>>  v = [1 3 5]
>>  w = [7;9;11]
```

MATLAB Básico: Vetores

```
>>  v = [1 3 5]
>>  w = [7;9;11]
>>  v'
```

MATLAB Básico: Vetores

```
>>  v = [1 3 5]
>>  w = [7;9;11]
>>  v'
>>  v(2)
```

MATLAB Básico: Vetores

```
>>  v = [1 3 5]
>>  w = [7;9;11]
>>  v'
>>  v(2)
ans =
    3
```

MATLAB Básico: Vetores

```
>>  v = [1 3 5]
>>  w = [7;9;11]
>>  v'
>>  v(2)
ans =
     3
>>  length(v)
```

MATLAB Básico: Vetores

```
>>  v = [1 3 5]
>>  w = [7;9;11]
>>  v'
>>  v(2)
ans =
     3
>>  length(v)
>>  size(v)
```

MATLAB Básico: Vetores

```
>>  v = [1 3 5]
>>  w = [7;9;11]
>>  v'
>>  v(2)
ans =
     3
>>  length(v)
>>  size(v)
>>  size(v,1)
>>  size(v,2)
```

Operações básicas

Lembre-se de respeitar as dimensões!

```
>>  v+w-z
>>  2*v
>>  a*w
>>  v/w
>>  v^2
>>  v.*w
>>  v./w
>>  v.^2
```

MATLAB Básico: Matrizes

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

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

MATLAB Básico: Matrizes

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

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

MATLAB Básico: Matrizes

```
>>  A = [1 2 3;4 5 6]
A =
     1 2 3
     4 5 6
>>  A(2,1)
ans =
    4
```

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

MATLAB Básico: Matrizes

```
>>  A = [1 2 3;4 5 6]
A =
     1 2 3
     4 5 6
>>  A(2,1)
ans =
    4
>>  A'
```

MATLAB Básico: Matrizes

```
>>  A = [1 2 3;4 5 6]
A =
     1 2 3
     4 5 6
>>  A(2,1)
ans =
     4
>>  A'
>>  size(A)
```

MATLAB Básico: Matrizes

```
>>  A = [1 2 3;4 5 6]
A =
     1 2 3
     4 5 6
>>  A(2,1)
ans =
     4
>>  A'
>>  size(A)
>>  size(A,1)
>>  size(A,2)
```

Operações com Matrizes

```
>> A = [1 2;3 4]
>> B = [2 1;0 3]
>> A+B
>> A-B
>> A*B
>> 2*A
>> B/3
>> A'
>> A.*B
>> A./B
>> A/B
```

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Funções básicas

```
>>  eye(n)
>>  zeros(m,n)
>>  ones(m,n)
>>  rand(m,n)
>>  size(A)
>>  inv(A)
>>  reshape(A,m,n)
```

Matrizes como vetores

O MATLAB permite que se acesse os elementos de uma matriz usando um índice único; nesse caso, os elementos são acessados da seguinte maneira:

$$A(i + m(j - 1)) = A(i, j),$$

com $1 \leq i \leq m$, $1 \leq j \leq n$, $A \in \mathbb{R}^{m \times n}$.

```
>> A(3)
>> length(A)
```

Um texto é um vetor!

Um texto funciona como uma lista (vetor):

```
>>  texto = 'Palavra'
>>  texto(1) = 'P'
>>  texto(2) = 'a'
>>  texto(1:2) = 'Pa'
>>  length(texto)
>>  size(texto)
>>  texto'
```

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Matrizes

No MATLAB, tudo é matriz!

Slicing

O MATLAB oferece uma maneira fácil de se acessar subelementos de matrizes, chamada *slicing*. Nesta operação, usamos a sintaxe

```
A(linhainicial:linhafinal, colunainicial:colunafinal)
```

para acessar a submatriz determinada entre as linhas `linhainicial` e `linhafinal`, e entre as colunas `colunainicial` e `colunafinal`. Aqui, é preciso tomar cuidado para que as dimensões da matriz resultante sejam consistentes.

Slicing

```
>>  A(i,j)
>>  A(i,:)
>>  A(:,j)
>>  A(:,:)
>>  A(1:2,:)
>>  A(1,2:3)
>>  A(:)
```

Apagando/adicionando elementos

Podemos apagar elementos de matrizes (ou linhas/colunas inteiras) usando a seguinte sintaxe:

```
>>  A(i,:) = []
>>  A(:,j) = []
```

Podemos também acrescentar elementos a qualquer momento:

```
>>  lista = [1,3,4,5]
>>  lista = [lista 2]
>>  lista
```

Dica

As operações acima também se aplicam a texto!

```
>>  frase = 'Oi, como vai?'
>>  frase = [frase(1:length(frase)-1) ' você?']
>>  frase(1:2)
>>  frase = strcat('Oi,',' como',' vai',' você?')
```

Atenção: `strcat` não preserva os espaços em branco.

Laços de repetição

Quando é necessário repetir certo comando de código várias vezes, usamos a estrutura `for`:

```
for i = 1:3
    i
end
```

Se quisermos usar um passo diferente de 1, podemos acrescentar um terceiro argumento:

```
for i = 3:-1:1
    i
end
```

Laços de repetição (2)

Quando é necessário repetir certo comando de código várias vezes **até que** uma certa condição seja satisfeita, usamos a estrutura `while`:

```
i = 1;
while i < 3
   disp('Mais um.')
   i = i + 1;
end
```

if - else - end

O `if` ("se") representa uma sentença lógica condicional:

```
if (sentença lógica)
   faça (1)
else
   faça (2)
end
```

Em Matlab, uma sentença lógica pode ter dois valores:

0 (Falso) ou 1 (Verdadeiro)

Importante

Atenção: ao compararmos números reais, devemos tomar cuidado com erro de arredondamento e a representação por ponto flutuante.

Exemplo:

```
>>    3-1.1-0.9
ans   =
         1.0000
>>    1 == (3-1.1-0.9)
ans   =
         0     (falso!)
```

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Estruturas de dados Heterogêneas

Muitas vezes, gostaríamos de armazenar dados da seguinte forma:

Título	Núm. Páginas	Datas de Empréstimo e Devolução
"Álgebra Linear"	205	12/08, 15/08
"Cálculo"	346	10/09, 12/09
"Geometria"	123	04/08, 05/09
"Topologia"	253	01/08, 04/09

Porém, estes dados são de natureza *heterogênea*: misturamos texto (string), números e intervalos. Como armazenar isso em uma só tabela no MATLAB?

Estrutura *Cell*

No MATLAB, podemos fazer o seguinte:

```
>> tabela = { 'Algebra Linear', 205, [1208, 1508];
>>            'Calculo', 346, [1009,1209];
>>            'Geometria', 123, [0408,0509];
>>            'Topologia', 253, [0108,0409] }
```

A célula funciona como uma matriz, mas aqui os índices são dados sempre entre chaves: `{}`.

Comandos

Para ver o que está armazenado na variável `tabela`, basta usarmos o comando

```
>> celldisp(tabela)
```

Para verificar o tamanho de uma célula, usamos o comando

```
>> size(tabela)
```

Para criar uma célula vazia com *m* por *n* elementos, usamos o comando

```
>> tabela = cell(m,n)
```

Podemos também calcular a *transposta* de uma célula:

```
>> tabela'
>> transpose(tabela)
```

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Acessando dados dentro de uma célula

Existem duas maneiras de acessar elementos dentro de uma célula:

- Se usamos índices entre parênteses, estamos acessando um subconjunto da célula original.
- Se usamos chaves ({}), estamos acessando os valores no interior de cada elemento da célula.

Exemplo:

```
>>  sub = tabela(1:2,1:2)
>>  tabela(2,:) = {'MATLAB', 300, [1201, 1401]};
>>  tabela
```

Conversão de tipos

Note que mesmo as células que contêm valores numéricos não estão armazenadas como números. Repare nos colchetes:

```
>>  vetor = tabela(:,2)
vetor =
          [205]
          [300]
          [123]
          [253]
>>  3*vetor
    Undefined function 'mtimes' for input
     arguments of type 'cell'.
```

Podemos facilmente converter esses dados para uma variável numérica usando o comano `cell2mat`:

```
>>  vetor = cell2mat(tabela(:,2))
>>  3*vetor
```

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Acessando valores

Para acessarmos o conteúdo de uma célula individual, usamos as chaves. Por exemplo, na nossa tabela,

```
>>  tabela{1,1}
```

é um texto com valor `'Algebra Linear'`, enquanto que

```
>>  tabela{1,2}
```

é um número de valor 205.

Acessando valores - slicing

O resultado de um acesso simultâneo a várias células, por exemplo usando slicing, é uma *lista* de valores: Note que se fizermos

```
>>  teste = tabela{1:3,2}
```

a variável `teste` conterá apenas o primeiro resultado da operação! Para armazenarmos todo o resultado do acesso a estes valores da célula, podemos associar o resultado a uma lista com o mesmo número de elementos que o número de resultados do acesso:

```
>>  [a,b,c] = tabela{1:3,2}
```

Acessando valores - slicing

Se todos os elementos selecionados da célula possuirem o mesmo tipo de dados, podemos atribuir esses elementos selecionados a uma só variável. Por exemplo, neste caso todos os dados selecionados são números, e assim:

```
>>  numpaginas = [tabela{1:3,2}]
```

No nosso exemplo, também temos variáveis de texto dentro da primeira coluna da célula. Podemos acessar um subconjunto do texto contido em uma das células associando os respectivos índices. Por exemplo:

```
>>  tabela{1,1}(1:3)
```

Outras possibilidades...

Célula de células:

```
>> v = { {1, 'teste', [1;2]};
         {[0,3], 12, 'nome', rand(4,4)} }
```

Neste caso, os elementos devem ser referenciados da seguinte forma:

```
>> v{2}{1}
```

Mais comandos

Podemos, analogamente ao que fizemos com vetores, concatenar células:

```
>> C1 = {'Joao', 16}
>> C2 = {'Maria', 18; 'Ricardo', 13}
>> cola = {C1 C2}
>> uniao = [C1; C2]
```

cuidado!