1.2 Representação de dados

Prof. Dr. Sidney Bruce Shiki
e-mail: bruce@ufscar.br
Prof. Dr. Vitor Ramos Franco
e-mail: vrfranco@ufscar.br

UFSCar – Universidade Federal de São Carlos
DEMec - Departamento de Engenharia Mecânica

Conteúdo

- Introdução
- Variáveis numéricas simples
- Vetores e matrizes
- Char e string
- Células
- Estruturas
- Exercícios

Introdução

- O passo primordial para qualquer código é a representação das informações;
- Esta aula tratará diferentes formas de se representarem dados (numéricos, textuais, ...);
- A forma a ser utilizada depende muito da aplicação.

Introdução

- Fato interessante: diferente de outras linguagens (como C, Fortran,...), em MATLAB na maioria das vezes não é necessário declarar as variáveis;
- Exemplo:

C

```
void main
{
   int a; // declara "a"
   a = 3 + 2;  // soma 3 com 2
}
```

MATLAB

```
a=3+2;
```

Variáveis numéricas simples

- Como mostrado no exemplo anterior, podemos atribuir variáveis simples simplesmente fazendo:

```
a=5.1414;
```

- Ao criar uma variável desse tipo, você notará que a mesma é da classe *double* ou *double precision*.

Variáveis numéricas simples

- Números complexos: você pode utilizar a constante especial “j” ou “i” para indicar a constante imaginária “j=sqrt(-1)”;
- As partes reais e imaginária desse número podem ser acessadas por `real()` e `imag()`

```
c=2.1j+3;
Parte_real = real(c);
Parte_imag = imag(c);
```

Variáveis numéricas simples

- Variáveis especiais:
  - π (3,141593...)

```
pi
```

  - "ε" da máquina (menor intervalo numérico)

```
eps
```

  - ∞ (infinito):

```
Inf
```

  - Not A Number

```
NaN
```

Vetores e matrizes

- A representação numérica que será mais frequentemente utilizada é de vetores e matrizes;
- O início e fim desse array é dado pelos sinais "[" e "]" respectivamente, as linhas da matriz são separadas pelo símbolo ";"
- Exemplo de criação:

```
Vetor1 = [12 pi 0 9];
Vetor2 = [2;2;2;0];
Matriz = [1 2 3;4 5 6];
```

Vetores e matrizes

- Operações:
  - Soma

```
Vetor1 = [12 pi 0 9];Vetor2 = [2 2 2 0];
Soma = Vetor1 + Vetor2;
```

  - Multiplicação e “divisão” matricial:

```
A=[1 2;3 3];B=[3 2;2 0];
C = A*B;
D = A/B;
```

  - Multiplicação/divisão ponto a ponto: é comum queremos operar cada termo de uma matriz ou vetor por outro de mesma ordem

```
A=[1 2;3 3];B=[3 2;2 0];
C = A.*B;D = A./B;
```

Vetores e matrizes

- Operações:
  - Matriz transposta:

```
A = [12 pi 0 9];
B = A';
```

  - Matriz inversa:

```
A = [12 pi;0 0];
B = inv(A);
```

Vetores e matrizes

- Matrizes especiais
  - Matriz de zeros:

```
A = zeros(3,3);
```

  - Matriz identidade:

```
A = eye(4);
```

  - Matriz de 1's:

```
A = ones(4,2);
```

  - Matriz de números aleatórios:

```
A = rand(4,4);    % entre 0 e 1
B = randn(10,2);  % distribuição normal
```

Vetores e matrizes

- Acessando números específicos na matriz:
  - Acessar um valor apenas:

```
A = [3 2 pi;0 90 4];
b = A(1,3);
c = A(4,4);
```

  - Acessar a coluna inteira:

```
D = A(:,2);  % 2ª coluna
```

  - Acessar linha inteira:

```
E = A(1,:);  % 1ª linha
```

  - Acessar faixas da matriz:

```
F = A(1:2,2:3);   %linha 1 até 2, coluna 2 até 3
```

Char e string

- Vetor de caracteres pode ser declarado com aspas simples:

```
Texto = 'Eu sou um texto';
TextoParte = Texto(1:9);% Podemos acessar
                        %partes do vetor de char
```

- Uma string única pode ser declarada com a função `string()`:

```
Texto = string('Eu sou uma string');
```

Células

- Muitas vezes queremos agrupar dados de diferentes formatos em uma variável;
- Não conseguimos fazer isso com as variáveis apresentadas anteriormente (sem “gambiarras”);
- Células são utilizadas para armazenar dados heterogêneos (similar a uma planilha);
- Imagine exemplos onde isso seria necessário.

Células

- Para criar células você pode preencher a mesma com “{“ e “}” preenchendo a “matriz” por completo com qualquer tipo de variáveis;

```
Dados = {'Nome','CPF','Idade';
         'João',510111023,23;
         'Maria',123123123,49};
```

- Ou usando o comando `cell()` e preenchendo itens um a um:

```
Dados = cell(3,3); %aloca tamanho da célula
Dados{1,1}='Nome';Dados{1,2}='CPF';
Dados{1,3}='Idade';Dados{2,1}='João';
Dados{2,2}=510111023;Dados{2,3}=23;
Dados{3,1}='Maria;Dados{3,2}=123123123;
Dados{3,3}=49;
```

Estruturas

- Outra forma mais elegante de se representar dados heterogêneos é usando estruturas (`structs('Campo',Valor,'Campo',Valor,...)`);
- Nesta forma você nomeia campos para se inserir dados:

```
Dados_Joao = struct('Nome','João','Idade',
23,'CPF',510111023);
Dados_Maria = struct('Nome','Maria','Idade',
49,'CPF',123123123);
```

Estruturas

- Se achar mais fácil, você pode declarar uma struct() vazia e ir criando os campos de pouco em pouco:

```
Dados_Joao = struct();
Dados_Joao.Nome='João';
Dados_Joao.Idade=23;
Dados_Joao.CPF=510111023;

Dados_Maria = struct();
Dados_Maria.Nome='Maria';
Dados_Maria.Idade=49;
Dados_Maria.CPF=123123123;
```

Exercícios

- Exercício 01 – Faça um script que receberá Nome completo, CPF e Idade de um pai e uma mãe (pode inventar os valores e nomes), use struct() para armazenar os dados. Crie também uma lista com 10 nomes de bebês. Seu código deverá sortear um entre os 10 nomes e completar os dados do bebê (Nome completo, CPF e Idade) em uma nova struct(). O CPF do bebê será a soma do CPF da mãe e do pai. (Obs.: use os comandos que aprendeu até o momento no curso, se precisar de algum outro comando use o `help` do MATLAB para achar).

Perguntas ?