Aula básica de OCTAVE/MATLAB

Lucas Nunes Sequeira

28 de março de 2019

OCTAVE - linguagem open source desenvolvida para computação matemática, possui uma boa interface gráfica e uma grande quantidade de funções built-in com uma ótima referência tanto pelo software como online.

Variáveis, expressões e declarações

Octave não é uma linguagem tipada, assim não é preciso escrever em seu código o tipo de variável a ser declarada (mas tome CUIDADO!)

```
>> x = 5;
>> y = 'Bananas Assassinas';
>> z = [1, 1, 2, 3, 5, 8];
```

Sempre que você precisar de uma ajuda sobre alguma função 'built-in', basta no terminal do OCTAVE digitar: **help *função***

```
>> help print
'print' is a function from the file /usr/share/octave/4.2.2/m/plot/util/print.m

-- print ()
-- print (OPTIONS)
-- print (FILENAME, OPTIONS...
```

Os operadores matemáticos são bastante intuitivos, como `+`, `-`, `*`, `/` etc; então isso não será um problema. No entanto uma coisa que pode vir a ser útil é o operador **exponencial**.

```
>> x = 5;
>> y = x**2.25 % sem ';' -> printa valor da variável a seguir
y = 37.384
```

Você também pode pedir ao usuário que digite uma entrada pelo terminal utilizando o comando **input()** de fácil implementação.

```
...  linhas de código
numero_de_ararinhas = input('Quantas ararinhas tem no céu? ');% assim você comenta
...  linhas de código
```

Mas e se eu quiser agora printar um valor no terminal? É simples, existe a função **printf()** para isso. Sendo que as referências às variáveis para constuir uma string são `%d` (inteiros), `%f` (floats) e `%s` (strings).

```
...  linhas de código
printf('No céu, Lipinho contou %d ararinhas.', numero_de_ararinhas)
...  linhas de código
```

```
>> line = 'Adorei seu chapéu!\n';
>> printf(line)
Adorei seu chapéu!
```

Execução condicional e loops

Para esse tópico é bom ressaltar que as declarações das condicionais são também muito tranquilas, e os operadores lógicos são `||` (ou), `&&` (e), `==` (igual), `>` (maior que) etc.

```
...  linhas de código
a = 25;
if (a == 10 || a > 15)
   a = 14;
elseif (a != 14 && a <= -5)
   a = 40;
else
   a = 0;
endif % aqui declaramos final do condicional if
...  linhas de código
```

Além disso, algo que é de grande utilidade são os loops, como **while** e **for**.

```
...  linhas de código
for i = 1:n %lê-se de i = 1 até i = n
   A(i) = i**i; % IMPORTANTE: os arrays são indexados a partir do 1
endfor
...  linhas de código
```

Funções

As **funções** funcionam como uma forma de você manter o seu corpo de código principal organizado e limpo, e ao invés de repetir o mesmo código algumas vezes; ou até mesmo como forma de tornar seu código mais legível, é sempre bom utilizá-las.

```
...  linhas de código
while ehPrimo(n) % aqui chamamos a função ehPrimo()
   n = n+51;
endwhile
...  linhas de código
```

Enquanto isso, em um arquivo separado você poderia então construir uma função que pode ou não retornar algo, é importante que o nome do arquivo receba o nome da função e tenha extensão **.m**, além disso, você pode retornar mais de um valor na função, o que torna ela ainda mais útil e simples, mas lembre-se, cuidado!

```
function resp = ehPrimo(n) % resp é a variável que a função retornará
   resp = true;
   if n < 2
      resp = false;
   else
      for i = 2:sqrt(n)
         if mod(n,i) == 0 % em C seria if(n%i == 0)
            resp = false;
         endif
      endfor
   endif
endfunction
```

Operações matriciais

Para esse tópico é preciso que esteje bem familiar sobre identificar dimensões matrizes, para isso utilizo a definição $\mathbb{R}^{n\times m}$ para denotar uma matriz de n linhas por m colunas (observação: $\mathbb{R}^{1\times m}$ e $\mathbb{R}^{n\times 1}$ ambos tomaremos como vetores linha e coluna, respectivamente). Primeiro, você pode, por exempo, inicializar uma matriz de zeros ou uns.

```
...  linhas de código
A_array = zeros(1, 5); % ∈ ℝ^{1×5}
A_matrix = zeros(5, 7); % ∈ ℝ^{5×7}

A_result = A_array * A_matrix; % ℝ^{1×5} * ℝ^{5×7} ∈ ℝ^{1×7}
...  linhas de código
```

Não só isso como você pode declarar uma matriz para valores definidos e assim usa-se ',' (vírgula) ou ' ' (espaço) para separar elementos da mesma linha e colunas distintas, e ';' (ponto e vírgula) para separar linhas.

```
>> A = [1, 1, 2; 3, 4, 5; 7, 8, -5; 2, 1, 0] % ∈ ℝ^{4×3}
A =

   1   1   2
   3   4   5
   7   8  -5
   2   1   0

>> A(3,3) % acessando a posição A_{3,3}
ans = -5
```

Além da multiplicação de matrizes, você pode somar, subtrair, aplicar funções a elas, acessar um determinado elemento e até aplicar operadores tipo transposta ('), 'elemento a elemento' etc e este último, basta adicionar um '.' antes do operador entre duas matrizes.

```
>> A = sin( [0, pi/6, pi/4, pi/2] ) % ∈ ℝ^4
A =
   0.00000 0.50000 0.70711 1.00000
>> B = [1, 2, 3, 4]; % ∈ ℝ^4
>> C = A .* B % ∈ ℝ^4
C =
   0.00000 1.00000 2.12132 4.00000
>> C(4) % acessando a posição C_4
ans = 4.00000
```

Gráficos

A visualização de gráficos do OCTAVE é bem simples de utilizar, oferece muitas formas de edição, título, legendas, multiplas curvas, tamanho e nome dos eixos etc...

```
>> X = [0 :  0.1 :  10]; % vetor de 0 a 10 com espaçamento 0.1
>> Y = sin(X) + 2*sin(2*X) + 3*sin(3*X);
>> hold on % para manter aberto um gráfico
>> plot(X, Y, 'linewidth', 3) % plotando (x, y) com expessura 3
>> legend('função'); title('Título'); xlabel('X'); ylabel('Y');
```