# Métodos Numéricos Computacionais

Prof. Darci Luiz Savicki

# INTRODUÇÃO

Embora existam diferenças entre os ambientes Octave e Matlab, as semelhanças na forma de programação permite executar todos os programas que iremos trabalhar nesta disciplina em ambos os programas.

Deixamos abaixo o link do programa para o Octave online, para aqueles que não desejam instalar o Octave e \ou o Matlab no seu computador. No Octave online, os códigos ficam armazenados na sua conta, e são rodados “na nuvem”.
https://octave-online.net/

No entanto, a maneira mais prática e fácil de usar o Octave é instalando ele no seu computador. Existem versões gratuitas para Windows e Linux.
https://www.gnu.org/software/octave/index - Linux
https://www.gnu.org/software/octave/download - Windows

# Gráfico de Funções

Esta secção ocupa-se de funções reais definidas no intervalo $(a, b)$. O comando `fplot(fun,lims)` traça o gráfico da função `fun` (definida por uma cadeia de caracteres) no intervalo (`lims(1)`,`lims(2)`). Por exemplo, para representar $f(x) = 1/(1+x^2)$ no intervalo $(-5, 5)$, podemos escrever

```
» fun ='1/(1+x.^2)'; lims=[-5,5]; fplot(fun,lims);
```

ou, mais directamente,

```
» fplot('1/(1+x.^2)',[-5 5]);
```

Dica:

Pratique os comando do octave / matlab.

Digite os comando acima e gere o gráfico função.

# Comando úteis

## Comando eval

O valor de uma função `fun` num ponto `x` determina-se escrevendo `y=eval(fun)`, depois de iniciar `x`. O valor correspondente é guardado em `y`. Notar que `x`, e portanto `y`, podem ser vectores. Ao usar este comando, a restrição é que o argumento da função `fun` deverá ser `x`. Se o argumento de `fun` tiver um nome diferente (o que acontece frequentemente quando este argumento é gerado no interior de um programa) o comando `eval` deverá substituir-se por `feval` (ver Observação 1.3).

```
» fun ='1./(1+x.^2)'; lims=[-5,5];
» fplot(fun,lims)
```

# Comando úteis

## Comando feval

(funções *inline*) O comando `inline`, cuja sintaxe mais simples é `g=inline(expr,arg1,arg2,...,argn)`, declara a função `g` que depende das cadeias `arg1,arg2,...,argn`. A cadeia `expr` contém a expressão de `g`. Por exemplo, `g=inline('sin(r)','r')` declara a função $g(r) = \sin(r)$. O comando abreviado `g=inline(expr)` supõe implicitamente que `expr` é uma função da variável por defeito `x`. Logo que uma função *inline* tenha sido declarada, poderemos calculá-la em qualquer conjunto de variáveis através do comando `feval`. Por exemplo, para calcular `g` nos pontos `z=[0 1]` podemos escrever

```
» feval('g',z);
```

Note-se que, contrariamente a `eval`, o comando `feval` não exige que o nome da variável (`z`) coincida com o nome simbólico (`r`) que figura no comando `inline`. •

## Programação em MATLAB

Expliquemos brevemente como escrever programas em MATLAB. Um novo programa deve ser colocado num ficheiro cujo nome inclua a extensão `.m`, chamado *m-file*. Estes ficheiros devem ser colocados num dos directórios onde o MATLAB procura automaticamente os *m-files*; a sua lista pode-se obter com o comando `path` (ver `help path` para aprender a adicionar um directório a esta lista). O primeiro directório examinado por MATLAB é o directório do trabalho em curso.

A este nível é importante distinguir entre *scripts* e *funções*. Um *script* é simplesmente uma colecção de comandos de MATLAB num *m-file* que pode ser usada interactivamente.

Uma *função* também se define num *m-file*, por exemplo `nome.m`, mas tem uma interface de entrada/saída bem definida, que se introduz com o comando `function`

```
function [out1,...,outn]=name(in1,...,inm)
```

onde `out1,...,outn` são as variáveis de saída e `in1,...,inm` são as variáveis de entrada.

# Exemplo de função que usa o comando “feval”

```
function [x]=itera(f,x,precisao,maxit)
E=input('usar erro absoluto ou erro relativo? EA=1  ER=2 ')
erro=1;
k=0;
while (erro>precisao)&(k<=maxit)
    k=k+1;
    fi=feval(f,x)
        EA=abs((x-fi));
        ER=abs(EA/x);
        if E==1, erro=EA; end
        if E==2, erro=ER; end
    x=fi;
end
if (k>maxit) & (erro>precisao)
    disp('Raiz nao encontrada com a precisão requerida!')
else
    disp('Raiz encontrada com a precisão requerida!')
endif
```

```
function y=f(x)
y=9.8*x/15*(1-exp(-135/x))-35;
endfunction
```

# Scripts x Functions
# (Roteiro x funções)

- No slide a seguir mostramos um script (roteiro), dentro do qual há uma chamada para uma function (função).
- Note que os roteiros (scripts) são apenas uma série de instruções.
- Já as “functions” tem na primeira linha:

```
function [saídas]=nome_da_função(entradas)
...
end
```

# Script

```
clear all  %é sempre prudente limpar as variáveis antigas, pois elas
d0=0;      %continuam valendo dentro do scrit
v0=0;
a=9.8;
t=3;
d=mruv(d0,v0,a,t)  %calcula apenas 1 vez a função mruv

graf=input('deseja graficar a função? 0=não  1=sim')
if (graf==1)
    tt=linspace(0,t,20);  %cria o vetor tt, de 0 a 3
    for i=1:20   %calcula apenas 100 vez a função mruv
        dd(i)=mruv(d0,v0,a,tt(i) );  %e armazena no vetor dd
    end
    plot(tt,dd), pause, hold on,
    plot(tt,dd,'o')
end
```

# Function

Dados de saída

Dados de entrada

```
function d=mruv(d0,v0,a,t)
    d=d0 + v0*t +1/2 * a*t^2;
end
```

# Tarefa 1)

1) Instale o Octave em seu computador. Escolha a versão de acordo com o seu sistema operacional (Linux ou Windows).

2) Digite o *script* e a *function* do slide anterior (em arquivos separados).

3)Execute o script e gere o gráfico correspondente.

# Tarefa 2)

1) Tome por base os programas mostrados nesta apresentação e crie um roteiro (script) que gere o gráfico das funções abaixo.

a) $y = (1-x)^2 \ , 0 \le x \le 1$

b) $y = sen(x) \ , \ 0 \le x \le \pi$

c) $y = sen\left(\frac{1}{x}\right) , \ 0 \le x \le 1$

OBS: O gráfico desta última função é particularmente interessante! Porque?