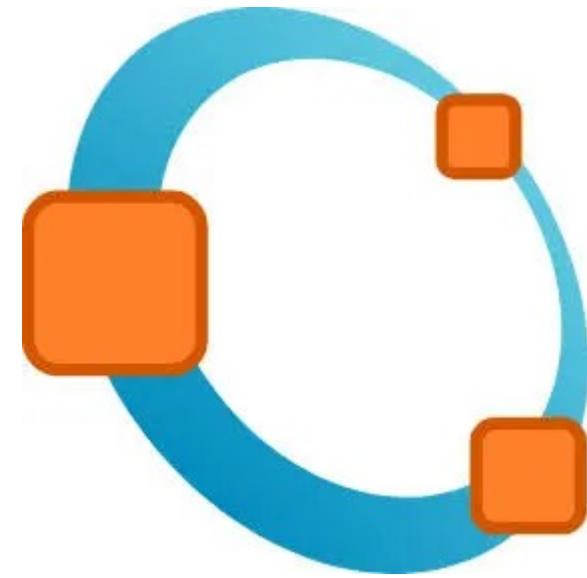


# Uma muito breve Introdução ao Octave



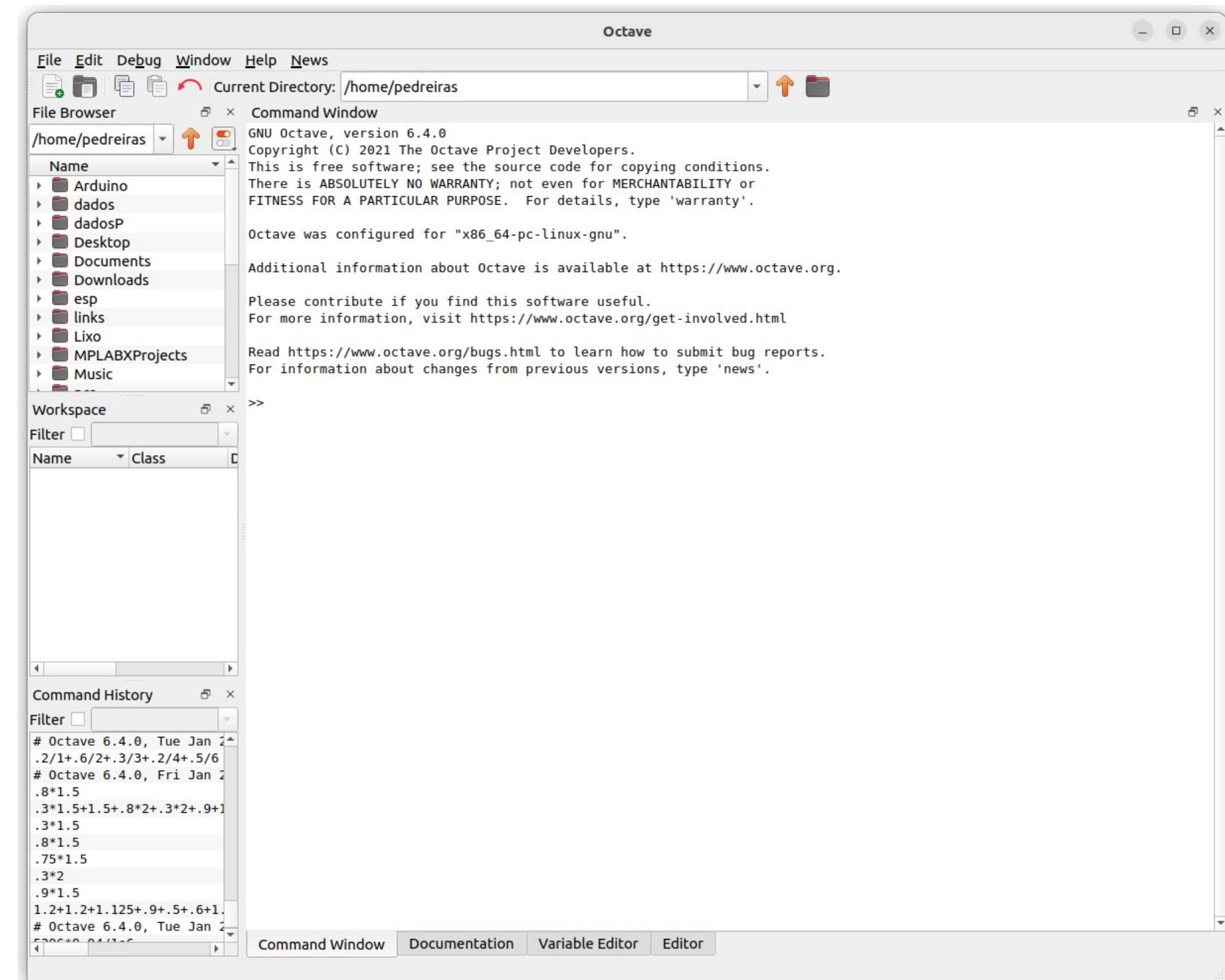
#43835 - Sensores e Sinais

Licenciatura em Engenharia de Computadores e Informática

Paulo Pedreiras ([pbrp@ua.pt](mailto:pbrp@ua.pt)), Pedro Fonseca ([pf@ua.pt](mailto:pf@ua.pt))  
UA/DETI/IT

# Ferramenta e linguagem de alto nível para cálculo numérico

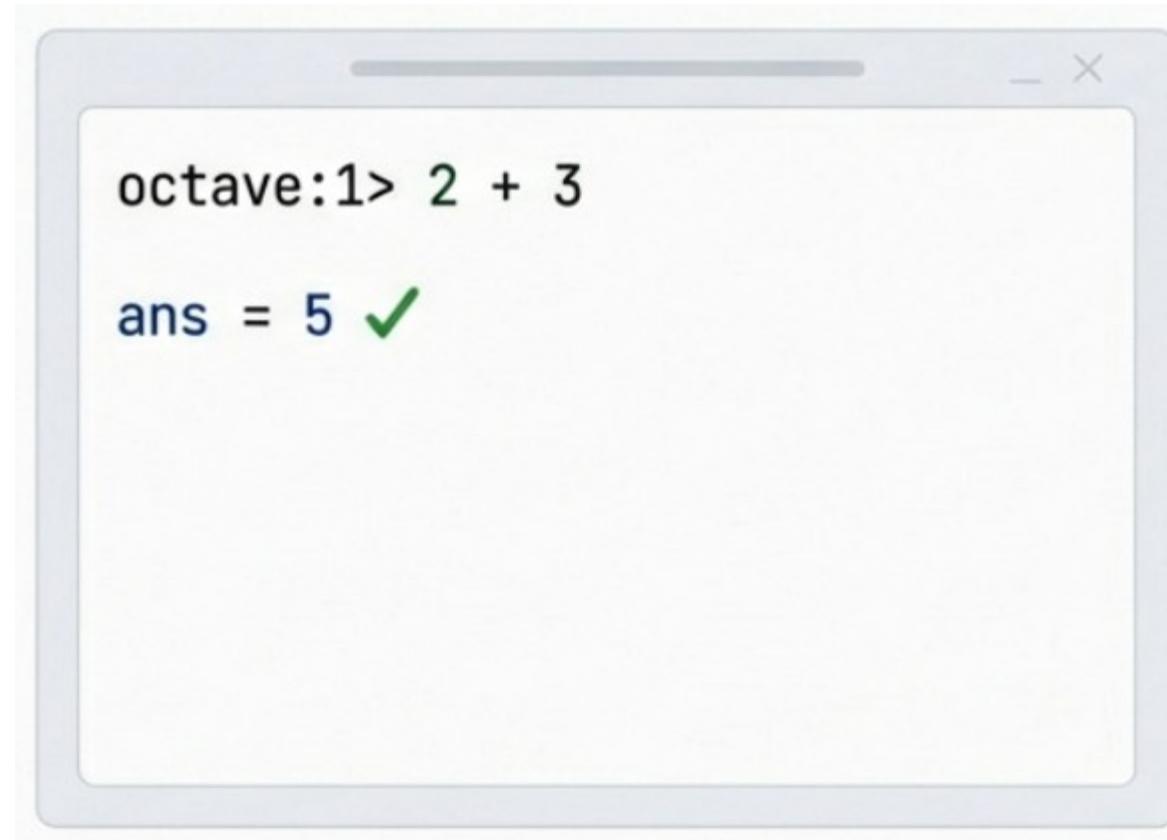
- O “irmão”: razoavelmente compatível com o Matlab, especialmente em uso elementar
- Licença: GNU General Public License (Free software).
- Filosofia: Software aberto, sem garantias. Usar e verificar.



# Interacção via *Prompt* ou janela de comandos

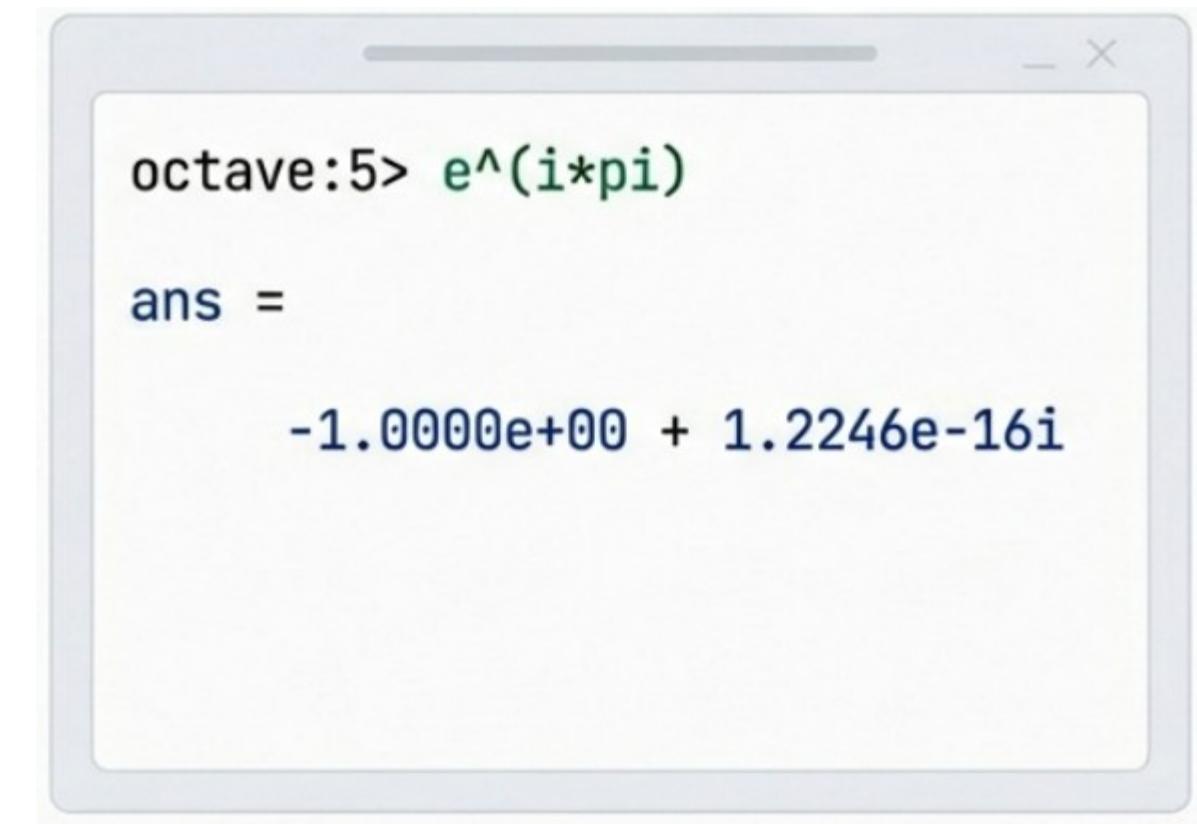
- possui interface de texto e gráfica

## Aritmética elementar



```
octave:1> 2 + 3
ans = 5 ✓
```

## E não só ...



```
octave:5> e^(i*pi)
ans =
-1.0000e+00 + 1.2246e-16i
```

**Constantes**

—

**Inf** (Infinity)

**NaN**(Not a Number)

**I,j** (Imaginary unit)

**pi** (3.14159...)

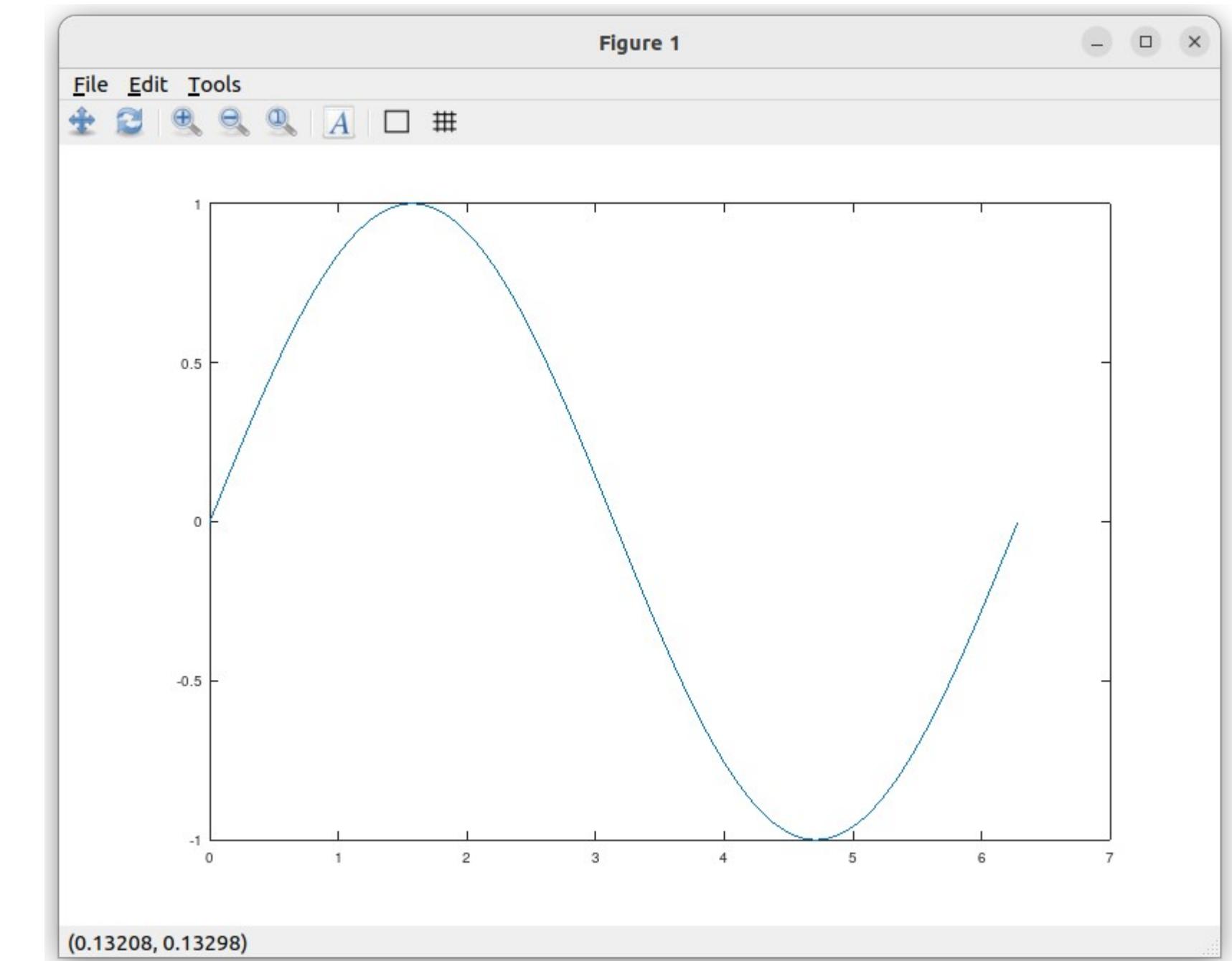
# Visualizando dados numéricos

```
x = linspace(0, 2*pi, 100);  
y = sin(x);  
plot(x, y);
```

Gera pontos igualmente espaçados

Operações matemáticas vectorizadas

Gera gráfico

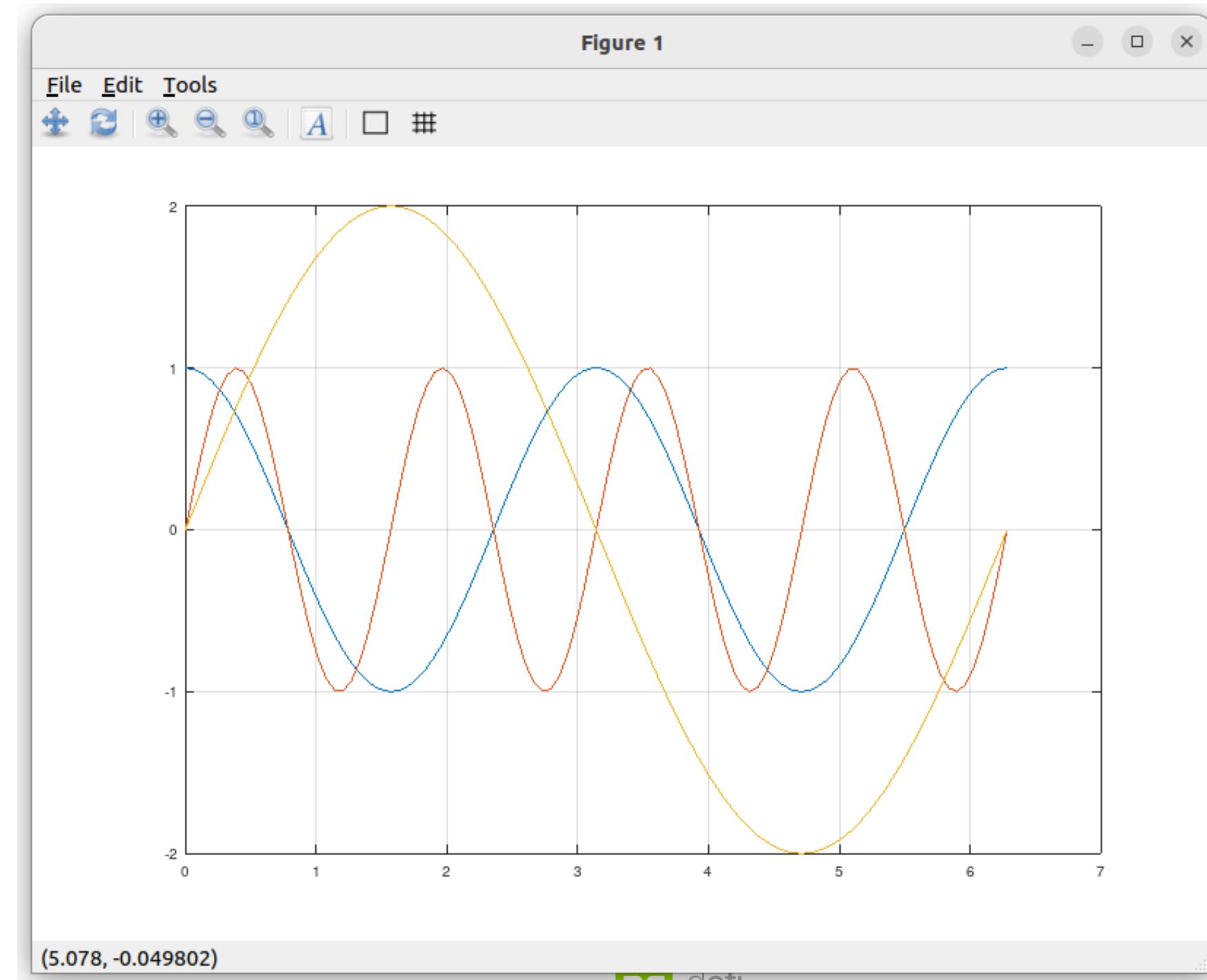


# Os gráficos podem ser elaborados

## Super-imposição, grelhas, ...

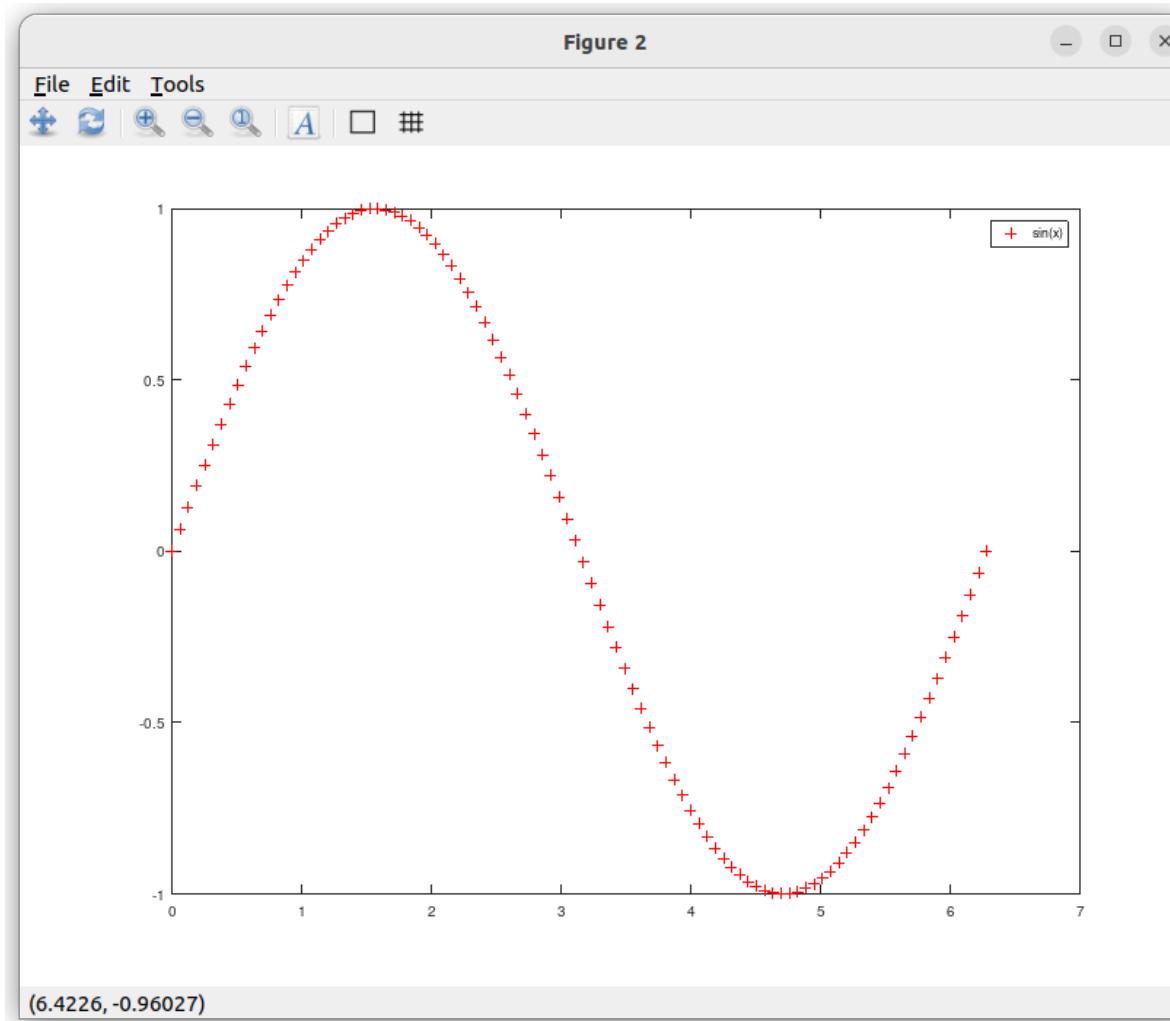
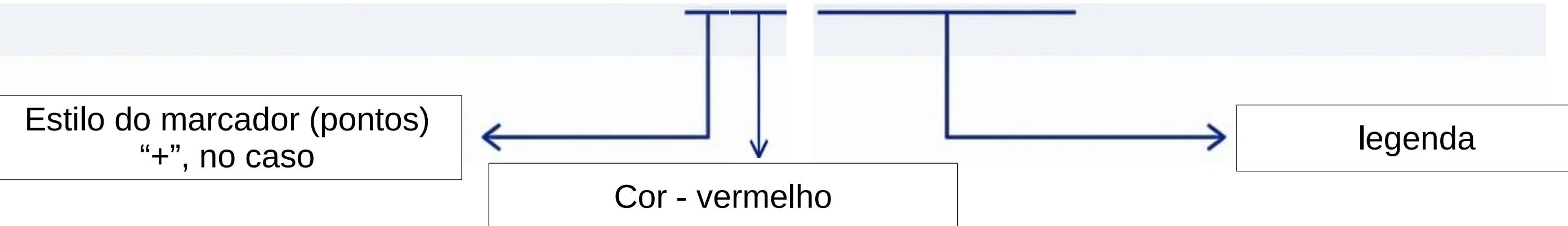
...

```
grid on;  
hold on;  
plot(x,cos(2*x));  
plot(x, sin(4*x));  
plot(x,2*sin(x));
```



# Melhorando a legibilidade ...

```
plot(x, y, '+r;sin(x);')
```



Exemplos:  
->`plot(x,y,'-k;sin(x);')` - Linhas a preto  
->`plot(x,y,'-sy;sin(x);')` - Linhas e quadrados, a amarelo

Comandos adicionais

- `title('Texto com titulo');`
- `xlabel('texto com legenda – eixo dos xx');`
- `ylabel('texto com legenda – eixo dos yy');`
- `legend('show');` % Mostrar a legenda no gráfico

# A base de representação é matricial

Vector linha

[ 1 , 2 ]

|   |   |
|---|---|
| 1 | 2 |
|---|---|

Vector coluna

[ 1 ; 2 ]

|   |
|---|
| 1 |
| 2 |

Matriz

[ 1 , 2 ;  
3 , 4 ]

|   |   |
|---|---|
| 1 | 2 |
| 3 | 4 |

Geradores rápidos

zeros(n)

ones(n)

eye(n)

rand(n)

# Indexação de vectores/matrizes

```
A = [1 2 3;  
      4 5 6;  
      7 8 9]
```

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

O operador “:” start:step:stop

```
>> 1:1:10  
ans =
```

```
1 2 3 4 5 6 7 8 9 10
```

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

A(1:2, 1:2)

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

A(:, end)



1:2:10

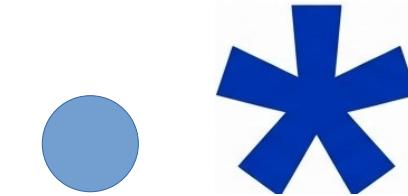
# O operador “.”

- operações elemento a elemento



**Multiplicação  
matricial “normal”**

```
>> [1 2] * [3 ; 4]  
ans = 11
```

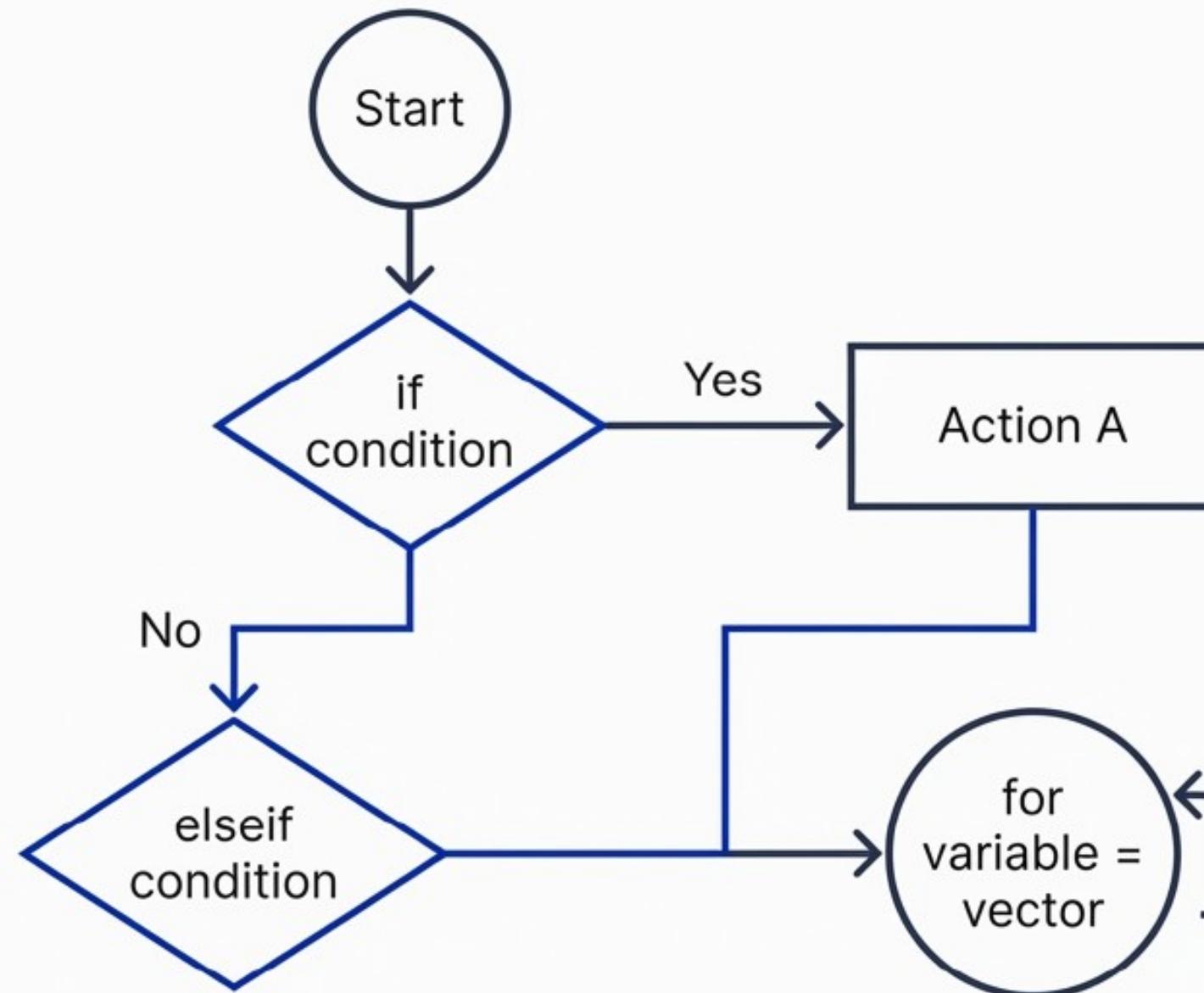


**Multiplicação  
Elemento-a-elemento**

```
>> [1 2] .* [3 4]  
ans =  
3 8
```

Incompatibilidade nas dimensões dos vectores geram erros como 'Nonconformant arguments'.

# Estruturas de controlo/programação



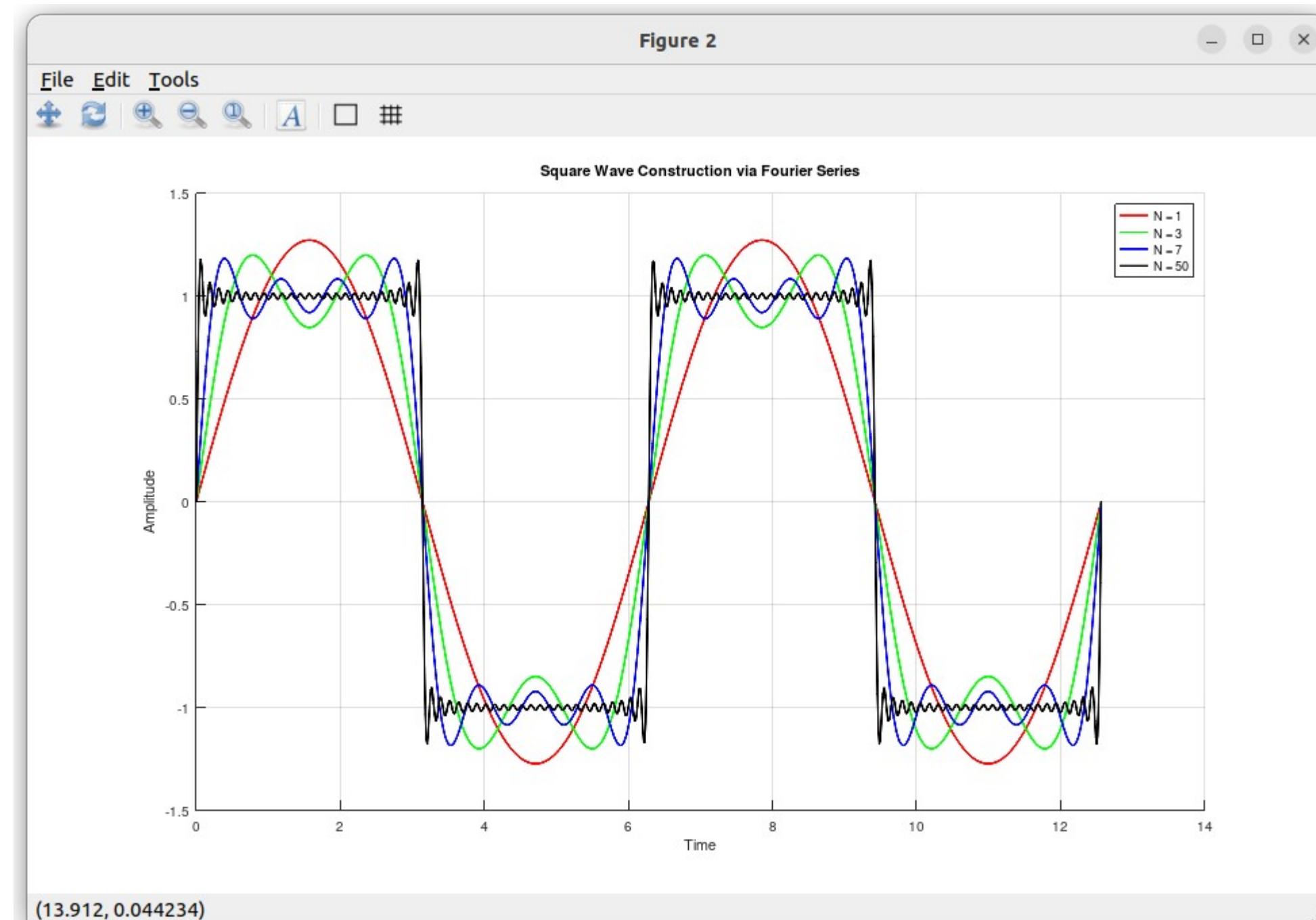
`for ... end`

`while ... endwhile`

`if ... elseif ... else ... endif`

`break / continue`

# Exemplo: Aproximação de onda quadrada por série de Fourier



# Exemplo: Aproximação de onda quadrada por série de Fourier

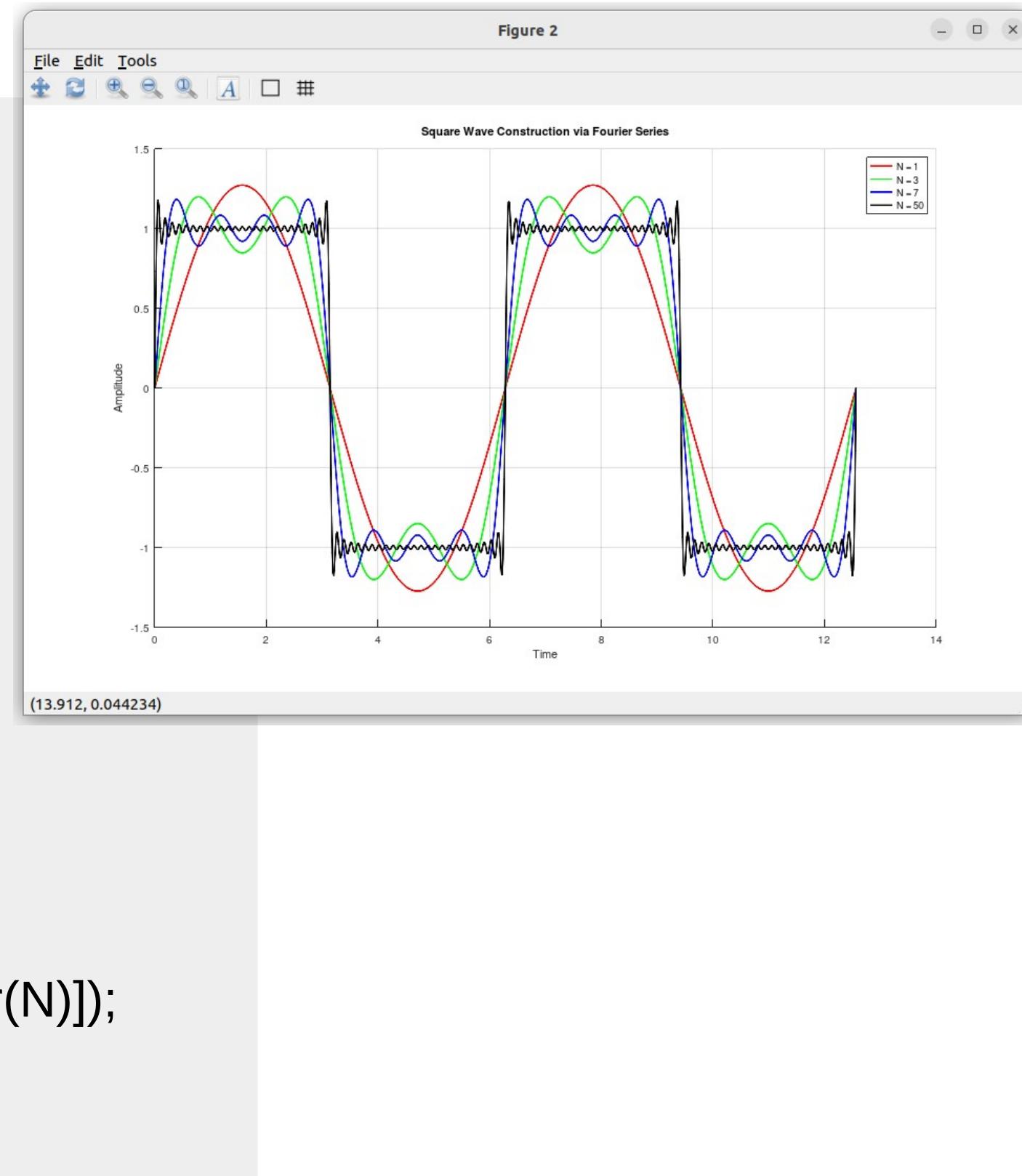
```
clear; clc; clf;

t = linspace(0, 4*pi, 1000); % Time vector (two periods)
n_terms = [1, 3, 7, 50];    % Number of harmonics to plot
colors = {'r', 'g', 'b', 'k'};

hold on;
for i = 1:length(n_terms)
    N = n_terms(i);
    f = zeros(size(t));

    % Summing the odd harmonics: (4/pi) * sum( sin(n*t) / n )
    for n = 1:2:N
        f = f + (4/pi) * (sin(n * t) / n);
    end

    plot(t, f, colors{i}, 'LineWidth', 1.5, 'DisplayName', ['N = ', num2str(N)]);
end
... % Title, axis labels, ...
```



# Scripts e Funções

## Anatomia de um ficheiro “.m”

### Script

- Ficheiro de texto simples que contém uma sequência de comandos Octave.
- Quando se executa um script, o Octave executa as linhas exactamente como se fossem digitadas na janela de comando, uma a uma.
- As variáveis são globais: qualquer variável criada num script permanece na área de trabalho após o script terminar.
- Sem entradas/saídas: os scripts não aceitam argumentos;

**Utilização:** ideal para automatizar uma série de etapas repetitivas, como carregar dados, realizar um cálculo e mostrar o resultado.

### Função

- Bloco de código reutilizável, projectado para realizar uma tarefa específica.
- Normalmente recebe entradas e retorna saídas.
- Espaço de trabalho privado. As variáveis criadas dentro de uma função são “locais”.
- Definida por palavras-chave: um ficheiro de função deve começar com a palavra-chave “function”.

**Utilização:** ideal para fórmulas matemáticas ou lógica que precisa de usar repetidamente com argumentos distintos

# Scripts e Funções

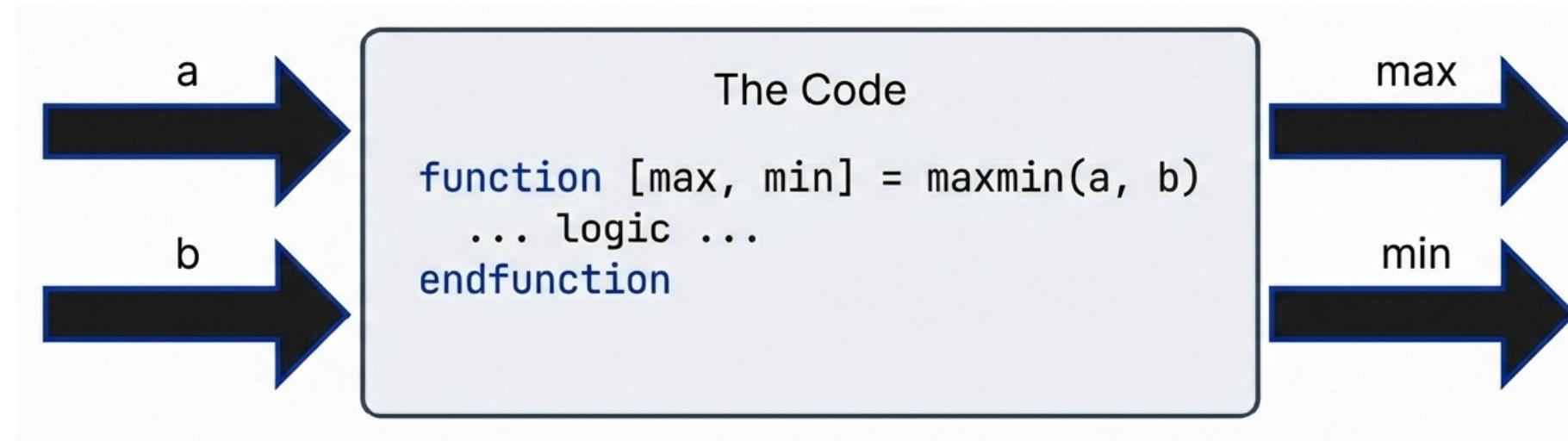
Anatomia de um ficheiro “.m”

```
% Returns the maximum value of a vector
function [max] = maxValue(v)
    n=length(v); % Get vector size
    m=v(1);      % Init variable that holds the max value

    for k=1:1:n    % Iterate over the vector elements
        if(v(k) > m) % If current value greater than max, update max
            m=v(k);
        endif

    endfor
    max=m;          % Assign computed value to return variable
endfunction
```

Similar a funções “C”



- Ao contrário de C, suporta nativamente múltiplos valores de retorno
- O nome do ficheiro tem que coincidir com o nome da função

# Representação de polinómios como vectores

- Permite calcular valores, raízes, multiplicar, ...

$$-2x^3 - 1x^2 + 0x + 1$$



Representado como

$$p = [-2, -1, 0, 1]$$

`polyval(p, x)`

Evaluate

`conv(p, q)`

Multiply polynomials

`roots(p)`

Solve for zero

`union / intersect`

Set operations

# Outras facilidades importantes para engenharia

## Álgebra linear

`inv (A)` : Inversa

`eig (A)` : Valores próprios

`A \ B` : Divisão esquerda

---

## Equações Diferenciais Ordinárias (ODE).

`lsode(fen, x0, t)`

# Algumas notas gerais

## 1. Vectorizar

Se um loop puder ser substituído por operações matriciais, deve fazer-se

“A \* B” é ordens de grandeza mais rápido que efectuar a correspondente multiplicação num loop

## 2. Explorar

Usar 'help command' e 'doc command'.

Há inúmeros comandos, que são, versáteis e complexos. Não (tentar sequer) decorar!

help inv; doc eig;

## 3. Packages

As packages dão acesso a funções especializadas (e.g. processamento de sinal/imagem, controlo, ....)

Ver  
<https://gnu-octave.github.io/packages/>

E o comando “pkg”

# Bibliografia

- Elaborado com a ajuda do “NotebookLM”
- Bibliografia base
  - GNU Octave Manual (<https://www.gnu.org/software/octave/doc/interpreter/>) (acedido em 2026/02/02)
  - Octave Programming Tutorial, Henri Amuasi (updated by Carl Scheffler and Mike Pickles), [https://en.wikibooks.org/wiki/Octave\\_Programming\\_Tutorial#](https://en.wikibooks.org/wiki/Octave_Programming_Tutorial#) (acedido em 2026/01/30)