

# Computação Científica com GNU Octave

Prof. Americo Cunha

Universidade do Estado do Rio de Janeiro – UERJ


[americo.cunha@uerj.br](mailto:americo.cunha@uerj.br)

[www.americocunha.org](http://www.americocunha.org)



 @AmericoCunhaJr

 @AmericoCunhaJr

 @AmericoCunhaJr



# Computadores: máquinas para fazer cálculos!

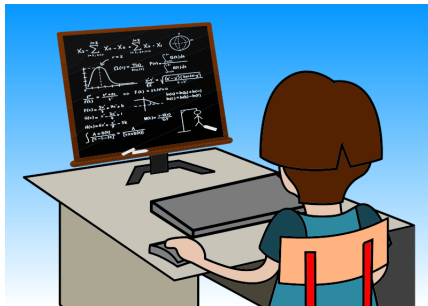
Computadores surgiram como ferramentas para fazer cálculos matemáticos que são difíceis (ou mesmo impossíveis) de serem executados a mão por seres humanos.



\* Figura obtida em <http://clipart-library.com/clipart/n1493907.htm>

# Computadores: máquinas para fazer cálculos!

Computadores surgiram como ferramentas para fazer cálculos matemáticos que são difíceis (ou mesmo impossíveis) de serem executados a mão por seres humanos.



**Eles ainda são (muito) usados com esse propósito!**

\* Figura obtida em <http://clipart-library.com/clipart/n1493907.htm>

# Linguagem de Programação

Uma *linguagem de programação* pode ser vista como um “idioma” para permitir a comunicação entre pessoas e máquinas.

```
1  /*
2  * This line basically imports the "stdio" header file, part of
3  * the standard library. It provides input and output functionality
4  * to the program.
5  */
6  #include <stdio.h>
7
8  /*
9  * Function (method) declaration. This outputs "Hello, world\n" to
10 * standard output when invoked.
11 */
12 void sayHello(void) {
13     // printf() in C outputs the specified text (with optional
14     // formatting options) when invoked.
15     printf("Hello, world!\n");
16 }
17
18 /*
19 * This is a "main function". The compiled program will run the code
20 * defined here.
21 */
22 int main(void)
23 {
24     // Invoke the sayHello() function.
25     sayHello();
26     return 0;
27 }
```



# Existe um zoológico de “idiomas” de programação!

Assim como existem vários idiomas no mundo, diferentes linguagens de programação estão disponíveis:



Perl

L<sup>A</sup>T<sub>E</sub>X

\* Existem muitas outras linguagens além dessas acima.



1. Ambiente avançado para computação científica
2. Linguagem de programação de alto nível
3. Simples, intuitivo e gratuito



# Como usar o GNU Octave?



# Anoc – App para celular

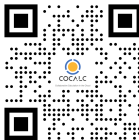




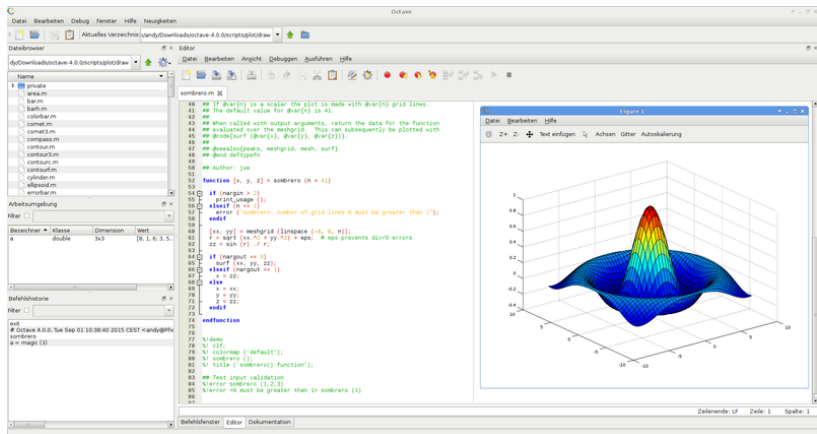
# CoCalc – Plataforma Computacional na Nuvem



Collaborative Calculation in the Cloud



# GNU Octave – Software p/ Desktop



# Ambiente para Computação Científica



# Calculadora Científica

- operações aritméticas:  
+, -, \*, /
- potenciação/radiciação:  
^, sqrt
- funções trigonométricas:  
sin, cos, tan  
asin, acos, atan

```
>> 2+3
ans = 5
>> 5-10
ans = -5
>> 3^4
ans = 81
>> sqrt(3)
ans = 1.7321
>> e
ans = 2.7183
>> pi
ans = 3.1416
>> sin(pi/2)
ans = 1
>> sin(90)
ans = 0.89400
>> cos(0)
ans = 1
>> tan(pi/4)
ans = 1.00000
>> asin(1)
ans = 1.5708
>> acos(-1)
ans = 3.1416
>> atan(0)
ans = 0
```



# Calculadora Científica

- exponenciação/logaritmo:

exp, log

- declaração de variáveis:

x, y, z

- números complexos:

i

- precisão numérica:

format short

format long

```
>> x = exp(1)
x = 2.7183
>> y = log10(e)
y = 0.43429
>> z = log(e)
z = 1
>> x + y
ans = 3.1526
>> y-z
ans = -0.56571
>> x*z
ans = 2.7183
>> y/z
ans = 0.43429
>> sqrt(-4)
ans = 0 + 2i
>> format short
>> sqrt(2)
ans = 1.4142
>> format long
>> sqrt(2)
ans = 1.414213562373095
```



# Vetores

- declaração de vetores:

a, b

- produto escalar:

dot

- produto vetorial:

cross

```
>> a = [1; 2; 0]
a =

     1
     2
     0

>> b = [-1; 1; 0]
b =

    -1
     1
     0

>> dot(a,b)
ans = 1
>> cross(a,b)
ans =

     0
    -0
     3
```



# Matrizes

- declaração de matrizes:  
A, B

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

```
A =
```

1	2	3	4
4	3	2	1
5	6	7	8
8	7	6	5

```
>> B = [2 0 0 0; 0 3 0 0; 0 0 4 0; 0 0 0 5]
```

```
B =
```

2	0	0	0
0	3	0	0
0	0	4	0
0	0	0	5



# Matrizes

- soma matricial:

$A+B$

- diferença matricial:

$A-B$

- produto matricial:

$A*B$

```
>> A+B
```

```
ans =
```

3	2	3	4
4	6	2	1
5	6	11	8
8	7	6	10

```
>> A-B
```

```
ans =
```

-1	2	3	4
4	0	2	1
5	6	3	8
8	7	6	0

```
>> A*B
```

```
ans =
```

2	6	12	20
8	9	8	5
10	18	28	40
16	21	24	25



# Matrizes

- produto de Hadamard:

$A \cdot B$

- matriz transposta:

$A'$

- determinante:

$\det$

```
>> A.*B
ans =

     2     0     0     0
     0     9     0     0
     0     0    28     0
     0     0     0    25

>> A'
ans =

     1     4     5     8
     2     3     6     7
     3     2     7     6
     4     1     8     5

>> det(A)
ans = 0
```



# Matrizes

- matriz identidade:  
eye
- matriz nula:  
zeros
- produto matriz-vetor:  
 $A*b$
- indexando elementos:  
 $C(:, \cdot)$

```
>> I = eye(3)
I =

Diagonal Matrix

    1    0    0
    0    1    0
    0    0    1

>> Z = zeros(3,4)
Z =

    0    0    0    0
    0    0    0    0
    0    0    0    0
```

```
>> b = [1; 1; 1; 1]
b =

    1
    1
    1
    1

>> C = A*b
C =

    10
    10
    26
    26

>> C(3,1)
ans = 26
```



# Lógica booleana

- **Negação**

$p$	$\text{não } p$
1	0
0	1

- **Conjunção**

$p$	$q$	$p \text{ e } q$
1	1	1
1	0	0
0	1	0
0	0	0

- **Disjunção**

$p$	$q$	$p \text{ ou } q$
1	1	1
1	0	1
0	1	1
0	0	0

- **Condicional**

$p$	$q$	Se $p$ então $q$
1	1	1
1	0	0
0	1	1
0	0	1

- **Bicondicional**

$p$	$q$	$p \text{ se e só se } q$
1	1	1
1	0	0
0	1	0
0	0	1



# Operações lógicas e relacionais

- igualdade e diferença:

`==`, `~=`

- desigualdades:

`<`, `<=`, `>`, `>=`

- conectivos lógicos (e, ou, não):

`&&`, `||`, `~`

```
>> a = 5
a = 5
>> b = 9
b = 9
>> a == b
ans = 0
>> a ~= b
ans = 1
>> a < b
ans = 1
>> a <= b
ans = 1
>> a > b
ans = 0
>> a >= b
ans = 0
>> a > 0 && b ~= b
ans = 0
>> a > 0 || b ~= b
ans = 1
>> ~(a > 0)
ans = 0
```



# Linguagem de Programação



# Funções e scripts

- **funções:** Executam tarefas individuais

```
1 % funcao
2 function [x1,x2] = eq2(a,b,c)
3     x1 = 0.5*(-b + sqrt(b^2-4*a*c));
4     x2 = 0.5*(-b - sqrt(b^2-4*a*c));
5 end
```

- **scripts:** Executam tarefas complexas

```
1 % script
2 clc
3 clear
4
5 a = 1
6 b = -5
7 c = 6
8
9 disp('As raizes sao:')
10 [x1,x2] = eq2(a,b,c)
11
12 disp('Soma e produto das raizes:')
13 x1+x2
14 x1*x2
```



# Condicional

- if — else

```
1  % script de teste p/ if-else
2  clc
3  clear
4
5  x = 2
6  y = 3
7
8  if x == y || x < 0
9      disp('x eh igual a y ou x eh negativo');
10 else
11     disp('x nao eh igual a y e x nao eh negativo');
12 end
```



# Laços

- while

```
1  % script de teste p/ while
2  clc
3  clear
4
5  k = 1
6  N = 3
7  x = 0.0
8
9  while k <= N && x < 30.0
10     x = x + sqrt(k)
11     k = k + 1
12 end
```

- for

```
1  % script de teste p/ for
2  clc
3  clear
4
5  N = 4
6  soma = 0.0
7  prod = 1.0
8
9  for i = 1:N
10     soma = soma + i
11     prod = prod*(1+i)
12 end
```





## Como citar esse material?

A. Cunha, *Computação Científica com GNU Octave*,  
Universidade do Estado do Rio de Janeiro – UERJ, 2023.



 @AmericoCunhaJr



@AmericoCunhaJr



@AmericoCunhaJr

Essas notas de aula podem ser compartilhadas nos termos da licença Creative Commons BY-NC-ND 3.0, com propósitos exclusivamente educacionais.

