

Noções de Programação para Computação Científica

Prof. Americo Cunha

Universidade do Estado do Rio de Janeiro – UERJ

americo.cunha@uerj.br

www.americocunha.org



Computadores: máquinas para fazer cálculos!

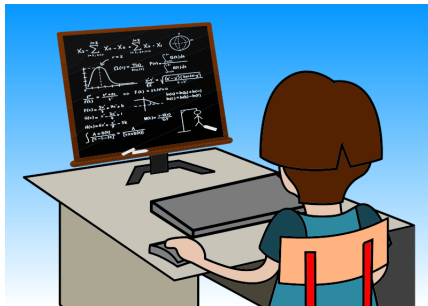
Computadores surgiram como ferramentas para fazer cálculos matemáticos que são difíceis (ou mesmo impossíveis) de serem executados a mão por seres humanos.



* 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^AT_EX

* 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



https://play.google.com/store/apps/details?id=verbosus.anoclite&hl=es_NI

<https://apps.apple.com/br/app/anoc-pro-octave-editor/id597864931>



CoCalc – Plataforma Computacional na Nuvem

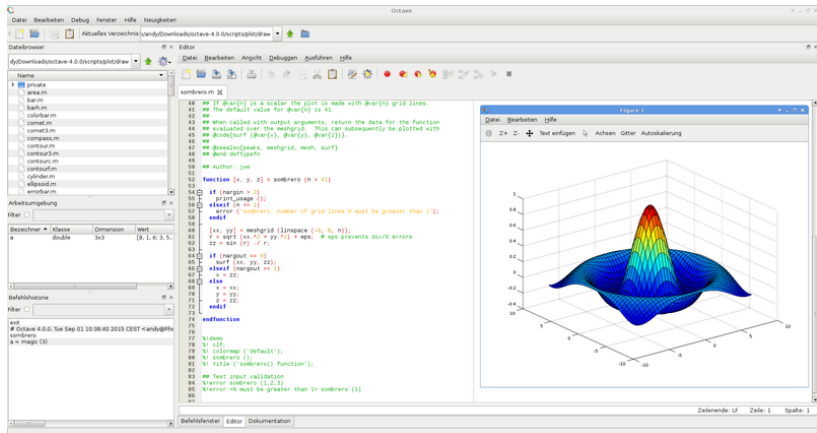


Collaborative Calculation in the Cloud

<https://cocalc.com>



GNU Octave – Software p/ Desktop



<https://gnu.org/software/octave>



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/logarítmo:

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

- produto escalar:

dot

- produto vetorial:

cross

```
>> 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:
 $C(\cdot, \cdot)$

```
>> 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
```



Linguagem de Programação



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
```



Condicionais

- condicional:
if else

```
>> x = 2
x = 2
>> y = 3
y = 3
>>
>> if x == y || x < 0
    display('x e y sao iguais ou x eh negativo.')
else
    display('x e y sao diferente ou x nao eh negativo.')
end
x e y sao diferente ou x nao eh negativo.
```



Laços

- laço:
while

```
>> k = 0
k = 0
>> N = 3
N = 3
>> x = 0.0
x = 0
>> while k <= N && x < 30.0
      x = x + sqrt(k)
      k = k + 1
end
x = 0
k = 1
x = 1
k = 2
x = 2.414213562373095
k = 3
x = 4.146264369941973
k = 4
```



Laços

- laço:
for

```
>> N = 4
N = 4
>> soma = 0.0
soma = 0
>> prod = 1.0
prod = 1
>> for i = 1:N
    i
    soma = soma + i
    prod = (1.2)*prod
end
i = 1
soma = 1
prod = 1.2000000000000000
i = 2
soma = 3
prod = 1.4400000000000000
i = 3
soma = 6
prod = 1.7280000000000000
i = 4
soma = 10
prod = 2.0736000000000000
```



Funções e scripts

- **funções:**
executam tarefas
- **scripts:**
chamam funções

```
>> % função
>> function [x1,x2] = eq2(a,b,c)
    x1 = (-b + sqrt(b^2 - 4*a*c))/(2*a);
    x2 = (-b - sqrt(b^2 - 4*a*c))/(2*a);
end
>>
>> % script
>> a = 2.0
a = 2
>> b = 1.0
b = 1
>> c = 1.0
c = 1
>>
>> [x1,x2] = eq2(a,b,c)
x1 = -0.25000 + 0.66144i
x2 = -0.25000 - 0.66144i
```


Como citar esse material?

A. Cunha, *Noções de Programação para Computação Científica*, Universidade do Estado do Rio de Janeiro – UERJ, 2021.

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

