

Noções de Programação com GNU Octave

Julio Cesar Basilio Marcos Vinicius Issa

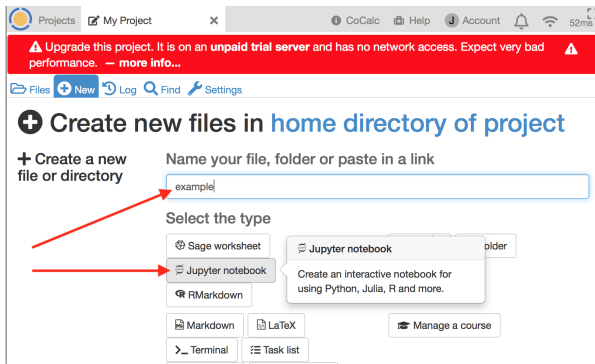
Victor Maudonet Americo Cunha

Universidade do Estado do Rio de Janeiro – UERJ



Noções de Programação com GNU Octave

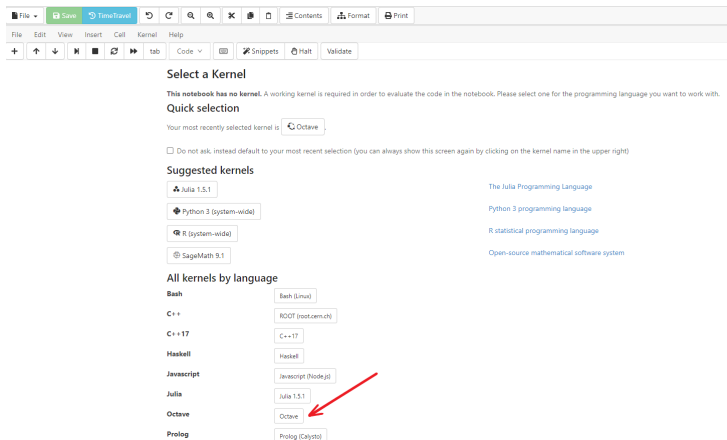
Primeiro clique em **(+)New** para criar um novo arquivo, nomeie seu arquivo e clique no **Jupyter Notebook**.



*Figura obtida em Cocalc.com.

Noções de Programação com GNU Octave

Em seguida, clique no Kernel que será utilizado. Escolha o Kernel **Octave**.



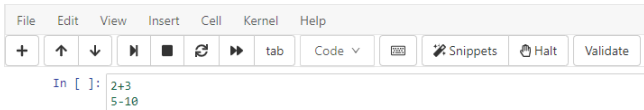
*Figura obtida em Cocalc.com.

Calculadora científica no GNU Octave

Inicialmente vamos entender algumas funções do GNU Octave que realizam as operações matemáticas básicas como numa calculadora científica no ambiente notebook Jupiter.

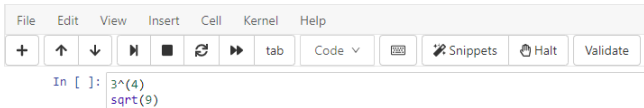
Soma, subtração, exponenciação e raiz quadrada

Da mesma maneira que em uma calculadora científica, a soma e a subtração no GNU Octave é realizado com os operadores $+$ e $-$.



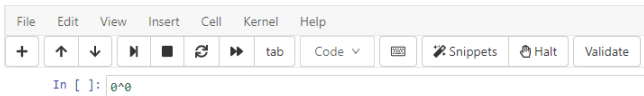
```
In [ ]: 2+3
        5-10
```

Já a exponenciação e a raiz quadrada são feitas com o **acento circunflexo** \wedge e com o comando **sqrt()**, respectivamente.



```
In [ ]: 3^(4)
        sqrt(9)
```

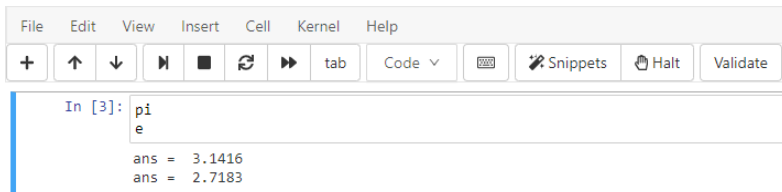
Experimente realizar a exponenciação com a base 0 (zero) e o expoente 0 (zero) e observe o resultado, conforme abaixo.



```
In [ ]: 0^0
```

Número π e o número de Euler

Esses são dois números fundamentais da matemática e que também são representados no GNU Octave. O número π é obtido pelo comando **pi**. Já o número de Euler é representado por **e**.



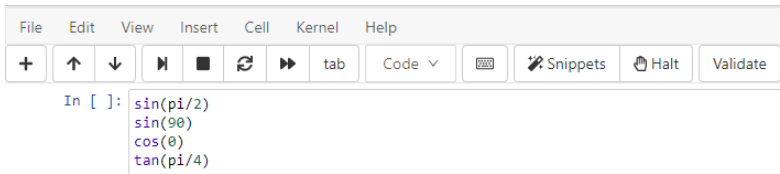
The image shows a screenshot of the GNU Octave command window. The menu bar at the top includes File, Edit, View, Insert, Cell, Kernel, and Help. Below the menu bar is a toolbar with icons for adding, navigating, executing, and other functions. The command window shows the input 'In [3]: pi' followed by 'e' on the next line. The output displays 'ans = 3.1416' for pi and 'ans = 2.7183' for e.

```
In [3]: pi
e
ans = 3.1416
ans = 2.7183
```

*Figura obtida em Cocalc.com.

Funções trigonométricas

As funções trigonométricas seno, cosseno e tangente são calculadas no GNU Octave através dos comandos **sin()**, **cos()** e **tan()**, respectivamente.



The image shows a screenshot of the GNU Octave software interface. At the top is a menu bar with the following options: File, Edit, View, Insert, Cell, Kernel, and Help. Below the menu bar is a toolbar with various icons for file operations, editing, and execution. The main area of the interface shows a command prompt with the following code entered:

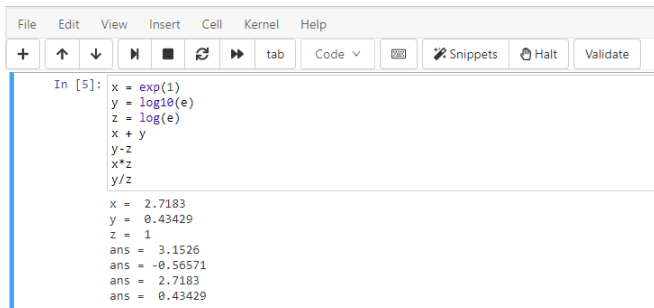
```
In [ ]: sin(pi/2)
sin(90)
cos(0)
tan(pi/4)
```

Observe que as funções trigonométricas no Octave são calculadas de acordo com o número em radianos entre parênteses. Logo **sin(90)** não será igual a 1. Para isso dever ser feito **sin($\frac{\pi}{2}$)**.

*Figura obtida em Cocalc.com.

Funções exponencial e logarítmica

A execução de uma função exponencial no GNU Octave é feita através do comando **exp()**. Já as funções logarítmicas em base natural ou na base 10 são feitas através dos comandos **log()** e **log10()**, respectivamente.



The screenshot shows the GNU Octave GUI. The menu bar includes File, Edit, View, Insert, Cell, Kernel, and Help. The toolbar contains icons for adding, navigating, and executing code. The command window shows the following input and output:

```
In [5]: x = exp(1)
        y = log10(e)
        z = log(e)
        x + y
        y - z
        x * z
        y / z

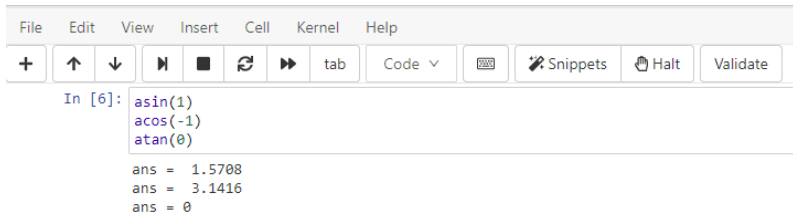
x = 2.7183
y = 0.43429
z = 1
ans = 3.1526
ans = -0.56571
ans = 2.7183
ans = 0.43429
```

Observe as operações de soma, subtração, multiplicação e divisão dessas funções, e analise os resultados para verificar se estão corretos.

*Figura obtida em Cocalc.com.

Funções trigonométricas inversas

As funções trigonométricas inversas de seno, cosseno e tangente são calculadas no GNU Octave através dos comandos **asin()**, **acos()** e **atan()**, respectivamente.



The screenshot shows the GNU Octave interface. The menu bar includes File, Edit, View, Insert, Cell, Kernel, and Help. Below the menu bar is a toolbar with icons for adding new files, navigating between files, running the code, and other standard IDE functions. The command window shows the following input and output:

```
In [6]: asin(1)
        acos(-1)
        atan(0)

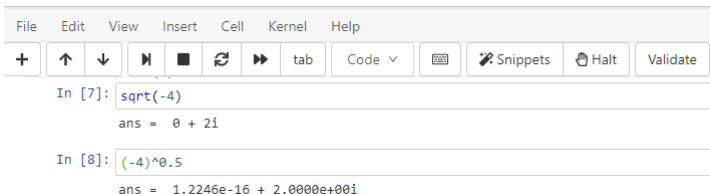
ans = 1.5708
ans = 3.1416
ans = 0
```

Observe que as funções trigonométricas inversas são calculadas de acordo com um número dentro dos parênteses que deve estar entre $[-1, +1]$.

*Figura obtida em Cocalc.com.

Números complexos

Os números complexos também são representados no GNU Octave. Essa representação é feita sempre com um número real + ou – um número seguido de **i** (parte imaginária).



```
File Edit View Insert Cell Kernel Help
+ ↑ ↓ ⏮ ■ ⏭ tab Code ▾ ⌨ Snippets ⏹ Halt Validate

In [7]: sqrt(-4)
ans = 0 + 2i

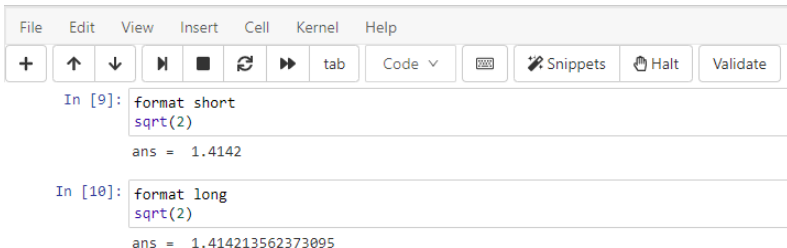
In [8]: (-4)^0.5
ans = 1.2246e-16 + 2.0000e+00i
```

Para resultar num número complexo, basta realizar a raiz quadrada de um número negativo. Outra forma de realizar a raiz quadrada é fazer a exponenciação a potência 0.5. Porém, como podemos observar acima, esse formato produz um resíduo da ordem de 10^{-16} no lugar do zero.

*Figura obtida em Cocalc.com.

Formatação e precisão numérica

O GNU Octave normalmente exibe os números com seis algarismos significativos. O comando **format** permite selecionar a forma com que os algarismos são mostrados. Digitando **format long** o Octave passa a exibir os valores com 16 dígitos nas respostas dos cálculos efetuados. Já o comando **format short** mostra o número com 5 dígitos.



```
In [9]: format short
sqrt(2)

ans = 1.4142

In [10]: format long
sqrt(2)

ans = 1.414213562373095
```

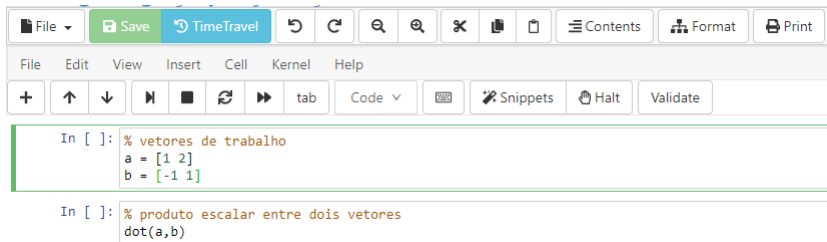
*Figura obtida em Cocalc.com.

Operações vetoriais e matriciais no GNU Octave

Agora vamos entender algumas funções do GNU Octave que realizam as operações matemáticas entre vetores e matrizes no ambiente notebook Jupiter.

Produto escalar com dois vetores

Primeiro definimos os vetores **a** e **b**, que serão utilizados na operação. O vetor no GNU Octave é definido com seus elementos separados por um "espaço" e todos eles entre colchetes [].



```
In [ ]: % vetores de trabalho
a = [1 2]
b = [-1 1]

In [ ]: % produto escalar entre dois vetores
dot(a,b)
```

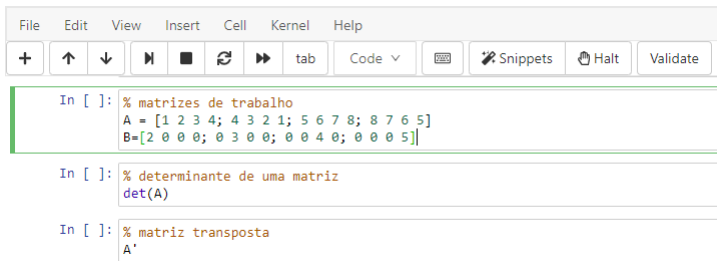
Em seguida, o produto escalar é realizado utilizando o comando **dot()** com os vetores **a** e **b** para o produto escalar.

*Figura obtida em Cocalc.com.

Determinante e transposta de uma matriz

Novamente o primeiro passo é definir as matrizes **A** e **B** que serão utilizados para calcular o determinante e a sua transposta. A matriz no GNU Octave tem suas linhas separadas por **ponto e vírgula** ;, seus elementos separados por um "espaço" e tudo entre colchetes [].

Para calcular a determinante de uma matriz é utilizado o comando **det()**, conforme indicado abaixo. Já a matriz transposta é obtida inserindo um **apóstrofo** ' ao lado da matriz original.



```
File Edit View Insert Cell Kernel Help
+ ↑ ↓ ⏮ ■ ⏭ tab Code ▾ ⌨ Snippets ⏹ Halt Validate

In [ ]: % matrizes de trabalho
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]

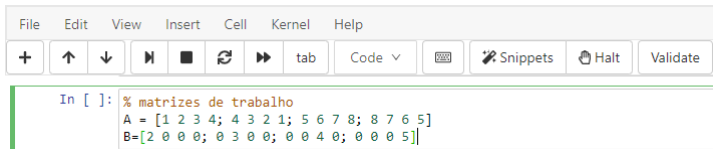
In [ ]: % determinante de uma matriz
det(A)

In [ ]: % matriz transposta
A'
```

*Figura obtida em Cocalc.com.

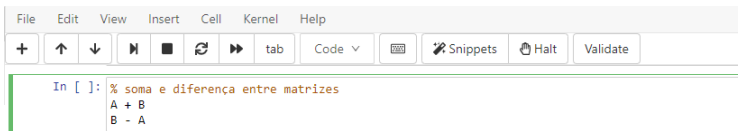
Soma e diferença entre matrizes

Para esta operação são utilizadas novamente as matrizes **A** e **B**.



```
In [ ]: % matrizes de trabalho
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]
```

O cálculo da soma e da diferença entre matrizes no GNU Octave é realizado da forma clássica, com os operadores $+$ e $-$, conforme abaixo:

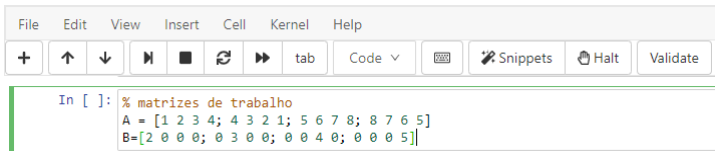


```
In [ ]: % soma e diferença entre matrizes
A + B
B - A
```

*Figura obtida em Cocalc.com.

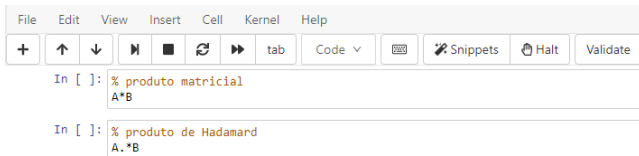
Produto Matricial e produto de Hadamard

Para esta operação são utilizadas novamente as matrizes **A** e **B**.



```
In [ ]: %matrizes de trabalho
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]
```

O cálculo de um produto matricial é realizado com o operador `*` entre as matrizes. Já o produto de Hadamard (produto entre os elementos de mesma "posição" das matrizes) utiliza o operador `.*` entre as matrizes, conforme abaixo:



```
In [ ]: %produto matricial
A*B

In [ ]: %produto de Hadamard
A.*B
```

*Figura obtida em Cocalc.com.

Produto entre uma matriz e um vetor

Para esta operação é utilizada a matriz **A** abaixo.

```
File Edit View Insert Cell Kernel Help
+ ↑ ↓ ⏮ ■ ↺ ⏭ tab Code ▾ 📄 Snippets ⏻ Halt Validate

In [ ]: %matrizes de trabalho
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]
```

Já o vetor utilizado nessa operação deve ser definido antes da operação, conforme descrito anteriormente e é apresentado a seguir.

```
File Edit View Insert Cell Kernel Help
+ ↑ ↓ ⏮ ■ ↺ ⏭ tab Code ▾ 📄 Snippets ⏻ Halt Validate

In [ ]: %produto matriz vetor
b = [1; 1; 1; 1]
A*b
```

O cálculo de um produto entre uma matriz e um vetor é realizado da mesma forma que um produto matricial, utilizando o operador ***** entre eles.

*Figura obtida em Cocalc.com.

Obter blocos (sub-matrizes) de uma matriz

Para esta operação é utilizada a matriz **A** abaixo.

```
File Edit View Insert Cell Kernel Help
+ ↑ ↓ ⏮ ■ ↺ ⏭ tab Code ▾ 📄 Snippets 🛑 Halt Validate

In [ ]: %matrizes de trabalho
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]
```

Para essa operação é preciso saber **A(linha, coluna)**, ou seja, do lado esquerdo da virgula deve estar o número da(s) linha(s) e do lado direito o número da(s) coluna(s) que se deseja. Como queremos sub-matrizes dessa matriz A, então utilizamos o operador : para dizer qual intervalo de linhas e colunas serão selecionadas, conforme abaixo.

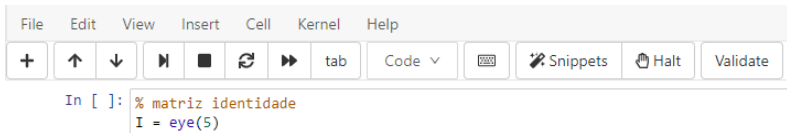
```
File Edit View Insert Cell Kernel Help
+ ↑ ↓ ⏮ ■ ↺ ⏭ tab Code ▾ 📄 Snippets 🛑 Halt Validate

In [3]: %blocos de matrizes
A1 = A(1:2,1:2)
A2 = A(1:2,3:4)
A3 = A(3:4,1:2)
A4 = A(3:4,3:4)
```

*Figura obtida em Cocalc.com.

Matriz identidade

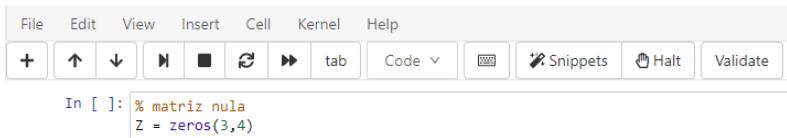
A matriz onde os elementos da diagonal principal são todos iguais a 1 e os elementos restantes são iguais a zero, é chamada de matriz identidade. Para obter essa matriz no GNU Octave é utilizada o comando **eye()** juntamente com a dimensão da matriz quadrada, do lado esquerdo da virgula o número de linhas e do lado direito o número de colunas, conforme abaixo.



*Figura obtida em Cocalc.com.

Matriz nula

A matriz onde todos os elementos são iguais a 0 (zero) é chamada de matriz nula. Para obter essa matriz no GNU Octave é utilizada o comando **zeros()** juntamente com a dimensão da matriz, conforme abaixo.

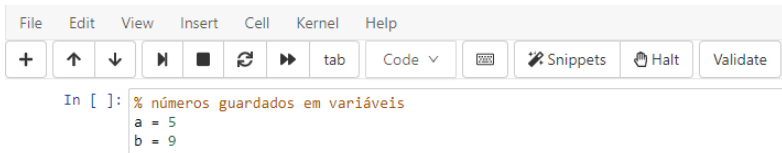
The image shows a screenshot of the Cocalc.com web interface. At the top, there is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, and Help. Below the menu is a toolbar with various icons for file operations (add, up, down), execution (run, stop, refresh, next), and editing (tab, code dropdown, keyboard icon). To the right of the toolbar are buttons for 'Snippets', 'Halt', and 'Validate'. The main area of the interface displays a code editor with the following text:

```
In [ ]: % matriz nula
        Z = zeros(3,4)
```

*Figura obtida em Cocalc.com.

Operações lógicas e relacionais

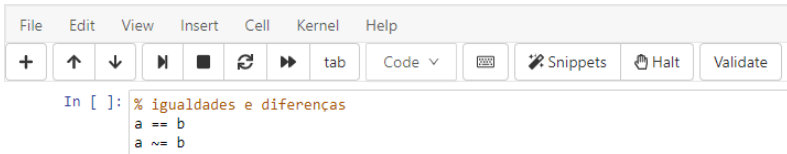
É possível guardar valores em variáveis ($=$) ,



The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for adding, navigating, and executing code. Below the toolbar is a code cell containing the following text:

```
In [ ]: % números guardados em variáveis  
a = 5  
b = 9
```

verificar igualdades ($==$), diferenças ($\sim =$) e



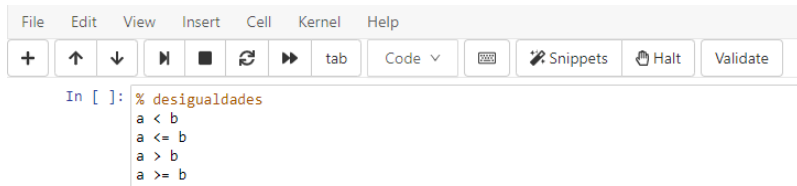
The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for adding, navigating, and executing code. Below the toolbar is a code cell containing the following text:

```
In [ ]: % igualdades e diferenças  
a == b  
a ~ = b
```

*Figura obtida em Cocalc.com.

Operações lógicas e relacionais

Desigualdades ($<$, $<=$, $>$, $>=$).

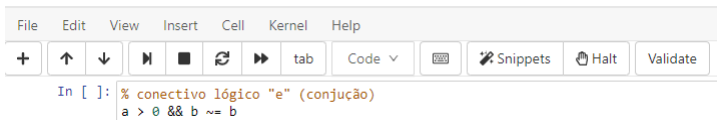


The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for adding, navigating, and executing code. The code cell contains the following text:

```
In [ ]: % desigualdades  
a < b  
a <= b  
a > b  
a >= b
```

Conectivos lógicos:

“e” (disjunção)



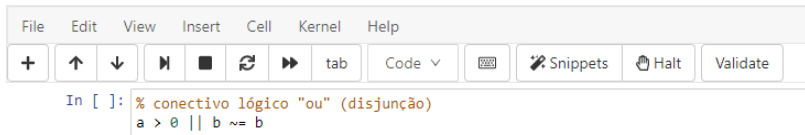
The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for adding, navigating, and executing code. The code cell contains the following text:

```
In [ ]: % conectivo lógico "e" (conjunção)  
a > 0 && b ~= b
```

*Figura obtida em Cocalc.com.

Operações lógicas e relacionais

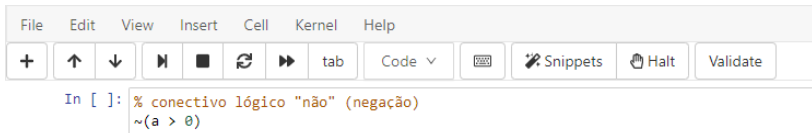
“ou” (disjunção)



The screenshot shows the Cocalc interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for adding, navigating, and executing code. Below the toolbar, a code cell is visible with the following content:

```
In [ ]: % conectivo lógico "ou" (disjunção)
a > 0 || b ~= b
```

“não” (negação)



The screenshot shows the Cocalc interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for adding, navigating, and executing code. Below the toolbar, a code cell is visible with the following content:

```
In [ ]: % conectivo lógico "não" (negação)
~(a > 0)
```

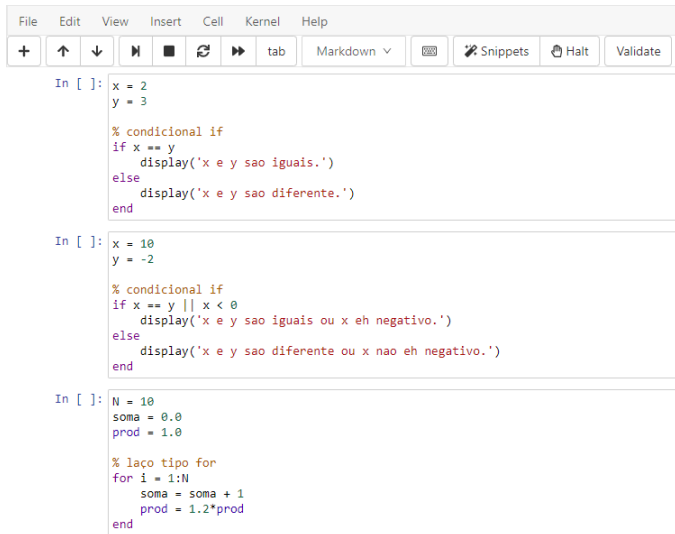
*Figura obtida em Cocalc.com.

Condicionais e laços

Condicionais são estruturas em que a sequência de comandos em seu interior é executada se uma determinada condição lógica for satisfeita. Em uma estrutura condicional, não se sabe em princípio se o interior da estrutura será executado. O comando utilizado para a construção de uma estrutura condicional é o “if – else”.

Já os laços são estruturas de repetição que permitem efetuar uma sequência de instruções múltiplas vezes. Os comandos utilizados para executar os laços são “for” e “while”.

Condicionais e laços



```
File Edit View Insert Cell Kernel Help
+ ↑ ↓ ⏮ ■ ↺ ⏭ tab Markdown 📄 Snippets ⏹ Halt Validate

In [ ]: x = 2
        y = 3

        % condicional if
        if x == y
            display('x e y sao iguais.')
        else
            display('x e y sao diferente.')
        end

In [ ]: x = 10
        y = -2

        % condicional if
        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

In [ ]: N = 10
        soma = 0.0
        prod = 1.0

        % laço tipo for
        for i = 1:N
            soma = soma + 1
            prod = 1.2*prod
        end
```

*Figura obtida em Cocalc.com.

Condicionais e laços



The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for adding, navigating, and executing code. Below the toolbar are two code cells, each starting with 'In []:'.

```
In [ ]: k = 0
        N = 15
        x = 0.0

        % laço tipo while
        while k <= N && x < 30.0
            x = x + sqrt(k)
            k=k+1
        end
```

```
In [ ]: k = 0
        N = 15
        x = 0.0

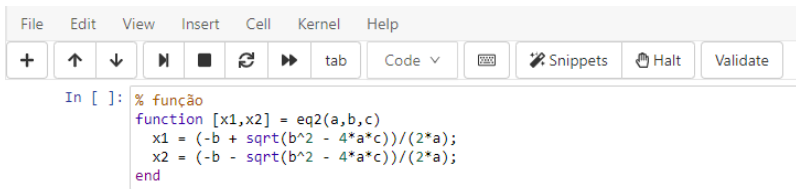
        % laço tipo while
        while k <= N
            x = x + sqrt(k)
            k=k+1
        end
```

*Figura obtida em Cocalc.com.

Funções e Scripts no GNU Octave

O programa a seguir calcula as raízes de uma equação do segundo grau. Implemente esse programa no ambiente GNU Octave e teste-o para diferentes valores de a , b e c .

I. Primeiro deve-se definir a função que usa a fórmula de resolução de uma equação do segundo grau:



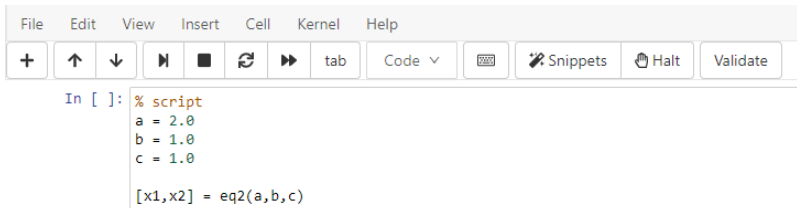
The image shows a screenshot of the GNU Octave environment. At the top is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. Below the menu bar is a toolbar with various icons for file operations, execution, and editing. The main area displays a code editor with the following text:

```
In [ ]: % 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
```

*Figura obtida em Cocalc.com.

Funções e Scripts no GNU Octave

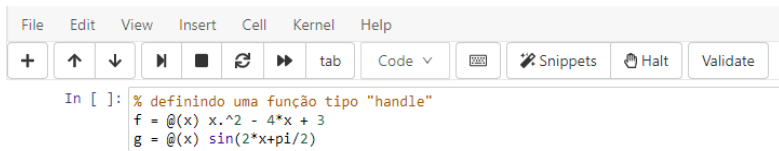
II. Em seguida executar o script principal com a definição dos parâmetros:



*Figura obtida em Cocalc.com.

Representação gráfica de funções

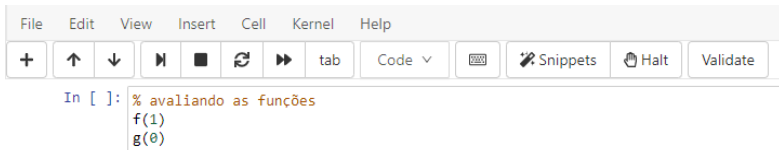
Definindo uma função tipo “handle”



The screenshot shows the Cocalc web interface. At the top is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. Below the menu is a toolbar with icons for adding a new cell, navigating between cells, running the current cell, and other actions. The main area contains a code editor with the following text:

```
In [ ]: % definindo uma função tipo "handle"
f = @(x) x.^2 - 4*x + 3
g = @(x) sin(2*x+pi/2)
```

Avaliando funções



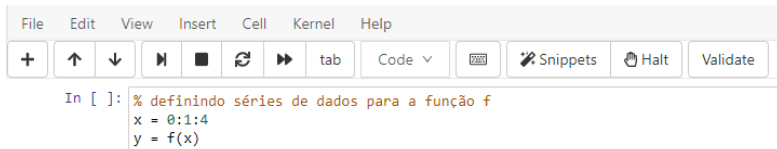
The screenshot shows the Cocalc web interface with the same menu and toolbar as the previous image. The code editor now contains the following text:

```
In [ ]: % avaliando as funções
f(1)
g(0)
```

*Figura obtida em Cocalc.com.

Representação gráfica de funções

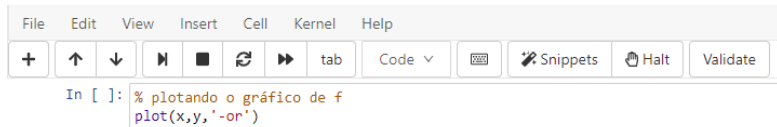
Definindo séries de dados para a função f



The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for adding, navigating, and executing code. The code cell contains the following text:

```
In [ ]: % definindo séries de dados para a função f
x = 0:1:4
y = f(x)
```

Plotando o gráfico de f



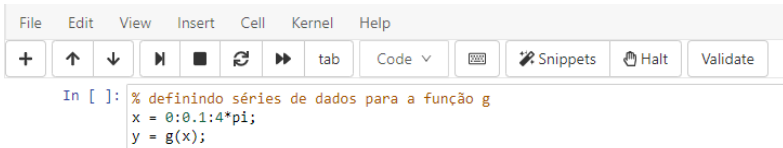
The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for adding, navigating, and executing code. The code cell contains the following text:

```
In [ ]: % plotando o gráfico de f
plot(x,y,'-or')
```

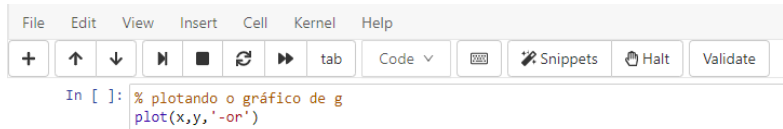
*Figura obtida em Cocalc.com.

Representação gráfica de funções

Definindo séries de dados para a função g



Plotando o gráfico de g



*Figura obtida em Cocalc.com.

Mais informações sobre o CoCalc

<https://doc.cocalc.com>