

Introdução ao GNU Octave

Profa. Dra. Anna Karina Fontes Gomes
Profa. Ma. Fernanda Luiz Teixeira

anna.gomes@ifsp.edu.br
fernanda.teixeira@ifsp.edu.br

06 e 08 de outubro de 2021



Sumário

O quê é GNU Octave ?

Introdução

Iniciando GNU Octave

Estruturas de controle de fluxo

Scripts e functions

Gráficos 2D

Plot de Matrizes

Sumário

O quê é GNU Octave ?

Introdução

Iniciando GNU Octave

Estruturas de controle de fluxo

Scripts e functions

Gráficos 2D

Plot de Matrizes

O quê é Octave ?

- ▶ **GNU Octave** é um aplicativo que foi desenvolvido originalmente com o propósito didático, com a intenção de criar um ambiente no qual a programação fosse mais rápida do que nas demais linguagens.
- ▶ É uma linguagem de **de alto nível**, destinada principalmente a cálculos numéricos. Fornece uma interface de linha de comando conveniente para resolver problemas lineares e não lineares numericamente.
- ▶ Seu desenvolvimento começou em 1988 e seguiu em constante desenvolvimento. Sua última versão (6.3.0) foi lançada em Julho de 2021.

O quê é Octave ?

Características básicas:

- ▶ Domínio público - Sua distribuição é feita de acordo com a licença GLP (*GNU General Public License*). Pode ser encontrada em:
<https://www.gnu.org/software/octave/>
<https://octave.sourceforge.io/>
- ▶ Sua linguagem é compatível com *MATLAB*® E *SciLab*®.
- ▶ São disponíveis versões para diferentes sistemas operacionais: Linux, Windows, Mac, entre outros.

Sumário

O quê é GNU Octave ?

Introdução

Iniciando GNU Octave

Estruturas de controle de fluxo

Scripts e functions

Gráficos 2D

Plot de Matrizes

Ambiente de trabalho e comandos básicos

- ▶ Janela de Comando (Command window): onde são realizadas as atribuições de variáveis e operações diversas.
- ▶ Pasta de trabalho (Current folder): indica todos os arquivos salvos na pasta indicada.
- ▶ Espaço de trabalho (Workspace): Exibe todas as variáveis ativas.

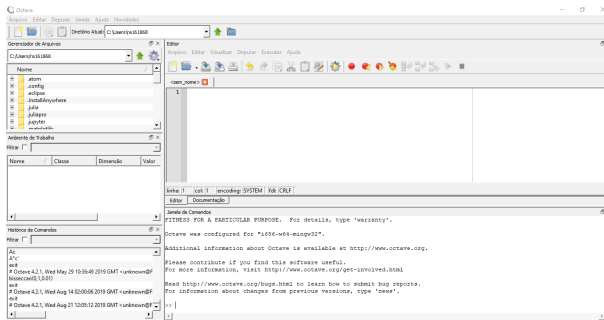


Figura: Interface gráfica do GNU Octave versão 4.2.1.

Ambiente de trabalho e comandos básicos

Comandos para funções básicas com escalares semelhantes aos adotados em calculadoras gráficas e científicas.

Tabela: Operações básicas em escalares.

Comando	Função desempenhada
$a+b$	Soma de dois escalares
$a-b$	Subtração de dois escalares
$a*b$	Multiplicação de dois escalares
a/b	Divisão de dois escalares $\frac{a}{b}$
a^b	Potenciação de dois escalares a^b
$\exp(a)$	Exponencial - $\exp(a)$
\sqrt{a}	Raiz quadrada - \sqrt{a}
$\sin(a)$	Seno de a - $\sin a$
$\cos(a)$	Cosseno de a - $\cos a$
$\log(a)$	Logaritmo de a na base e - $\ln a$
$\log_{10}(a)$	Logaritmo de a na base 10 - $\log a$.

Ambiente de trabalho e comandos básicos

Existem algumas variáveis que são intrínsecas ao GNU Octave:

Tabela: Constantes e seus valores.

Comando	Função desempenhada
<code>pi</code>	$\pi \cong 3.141592 \dots$
<code>i</code> ou <code>j</code>	Unidade imaginária = $\sqrt{-1}$, se não for atribuído outro valor para <code>i</code> e <code>j</code>
<code>inf</code>	Infinito
<code>NaN</code>	<i>Not-a-Number</i> - resultado numérico não definido ($\frac{0}{0}$)
<code>ans</code>	Resultado da última operação
<code>end</code>	se <code>a</code> for um vetor, <code>a(end)</code> representa o último elemento
<code>help (comando)</code>	Ajuda do GNU Octave
<code>version</code>	Versão do GNU Octave

Sumário

O quê é GNU Octave ?

Introdução

Iniciando GNU Octave

Estruturas de controle de fluxo

Scripts e functions

Gráficos 2D

Plot de Matrizes

- ▶ O GNU Octave trabalha essencialmente com um tipo de variável: `matriz`, que pode conter números (reais ou complexos) e textos.
 - ▶ Um escalar é uma matriz 1×1 ;
 - ▶ Um vetor é uma matriz $1 \times n$ (linha) ou $n \times 1$ (coluna).
- ▶ Não é necessário que sejam declaradas as variáveis e os respectivos tipos (inteiro, char, double, ...) para iniciá-las. Ao atribuir valor a variável, o programa aloca a memória automaticamente.

Matrizes

Para atribuir valores a uma variável digita-se os dados:

- ▶ Se for matriz ou vetor os elementos devem estar entre colchetes, [];
- ▶ Os elementos da mesma linha devem ser separados por espaço ou vírgula (,);
- ▶ Para trocar de linha usa-se ponto e vírgula (;) ou enter.

Janela de Comandos	Janela de Comandos
>> a=[1 2 3]	>> b=[1; 2; 3]
a =	b =
1 2 3	1
	2
	3
>> a=[1,2,3]	
a =	>> b=[1
1 2 3	2
	3]
>>	b =
	1
	2
	3

Figura: Vetor linha e vetor coluna.

Matrizes

Vetores também podem ser criados a partir de sequências de números, usando a notação início:passo:fim. Por exemplo,

```
Janela de Comandos
>> c=1:2:9
c =
    1    3    5    7    9

>> d=10:-1:1
d =
   10    9    8    7    6    5    4    3    2    1

>> e=1:7
e =
    1    2    3    4    5    6    7

>> |
```

Figura: Vetores.

Outra maneira de criar vetores é através do comando `linspace(a,b,n)` que gera n elementos igualmente espaçados entre a e b . Por exemplo,

```
Janela de Comandos
>> linspace(0,1,5)
ans =
    0.00000    0.25000    0.50000    0.75000    1.00000
>> |
```

Figura: Vetores.

Matrizes

Para entrar com a matriz abaixo e armazená-la na variável A:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
Janela de Comandos
>> A=[1 2 3; 4 5 6; 7 8 9]
A =

     1     2     3
     4     5     6
     7     8     9

>> |
```

Figura: Matriz A.

Matrizes

Um elemento específico da matriz pode ser acessado especificando a linha e a coluna do elemento desejado, fazendo $A(\text{linha}, \text{coluna})$.

```
Janela de Comandos
>> A=[1 2 3; 4 5 6; 7 8 9]
A =

     1     2     3
     4     5     6
     7     8     9

>> A(2,3)
ans = 6
>> |
```

Figura: Elemento da matriz A.

Matrizes

É possível extrair submatrizes de uma matriz dada. Por exemplo,

```
Janela de Comandos
>> A=[1 2 3; 4 5 6; 7 8 9]
A =

     1     2     3
     4     5     6
     7     8     9

>> B=A(1:2,3)
B =

     3
     6

>> C=A(1:2,2:3)
C =

     2     3
     5     6

>> D=triu(A)
D =

     1     2     3
     0     5     6
     0     0     9
```

Figura: Submatrizes de A.

- ▶ *B* armazena os elementos das linhas 1 e 2 da coluna 3;
- ▶ *C* armazena os elementos das linhas 1 e 2 das colunas 2 e 3;
- ▶ *D* extrai a matriz triangular superior de *A*. (teste $E=\text{tril}(A)$!)

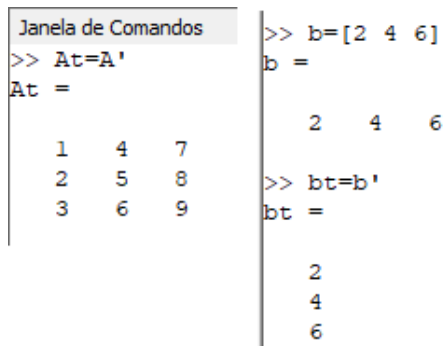
Matrizes

Algumas matrizes especiais:

- ▶ Criar uma matriz unitária de dimensão $n \times n$, `E=ones(n)`;
- ▶ Criar uma matriz de elementos nulos $n \times n$, `F=zeros(n)`;
- ▶ Criar uma matriz identidade de dimensão n , `G=eye(n)`;
- ▶ Criar um vetor com os elementos da diagonal de uma matriz A , `u=diag(A)`;
- ▶ Criar uma matriz diagonal com os elementos da diagonal de uma matriz A , `H=diag(diag(A))`;
- ▶ Criar uma matriz de elementos randômicos (aleatórios entre 0 e 1) de dimensão n , `I=rand(n)`.

Operações com matrizes

- A transposição de uma matriz ou vetor é indicado por um apóstrofe:



The image shows two side-by-side MATLAB command windows. The left window is titled 'Janela de Comandos' and shows the command `>> At=A'` followed by the output `At =` and a 3x3 matrix: $\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$. The right window shows the command `>> b=[2 4 6]` followed by the output `b =` and the row vector $[2 \ 4 \ 6]$. Below that, it shows the command `>> bt=b'` followed by the output `bt =` and the column vector $\begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$.

```
Janela de Comandos
>> At=A'
At =
     1     4     7
     2     5     8
     3     6     9

>> b=[2 4 6]
b =
     2     4     6

>> bt=b'
bt =
     2
     4
     6
```

Figura: Transposta.

Operações com matrizes

- ▶ Nas operações com matrizes devem ser respeitadas as regras usuais da matemática
 - ▶ Multiplicação por escalar

Janela de Comandos

```
>> b=[2 4 6]
```

```
b =
```

```
2    4    6
```

```
>> c=5*b
```

```
c =
```

```
10   20   30
```

```
>> |
```

Operações com matrizes

► Soma/Subtração de matrizes

```
Janela de Comandos
>> A
A =
    1    2    3
    4    5    6
    7    8    9

>> B=ones(3)
B =
    1    1    1
    1    1    1
    1    1    1

>> C=A+B
C =
    2    3    4
    5    6    7
    8    9   10

>> D=A-B
D =
    0    1    2
    3    4    5
    6    7    8
```

- Multiplicação de matrizes e potenciação também seguem as regras usuais.

Operações com matrizes

- **Operações elemento-a-elemento:** Colocando um ponto antes do operador ($.*$, $.^*$, $./$, ...) resulta que os elementos serão operados termo-a-termo entre as matrizes.

```
Janela de Comandos
```

<pre>>> A</pre> <pre>A =</pre> <table border="0"><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></table>	1	2	3	4	5	6	7	8	9	<pre>>> A.*B</pre> <pre>ans =</pre> <table border="0"><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></table>	1	2	3	4	5	6	7	8	9	<pre>>> (A.^2) ./A</pre> <pre>ans =</pre> <table border="0"><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></table>	1	2	3	4	5	6	7	8	9
1	2	3																											
4	5	6																											
7	8	9																											
1	2	3																											
4	5	6																											
7	8	9																											
1	2	3																											
4	5	6																											
7	8	9																											
<pre>>> B=ones(3)</pre> <pre>B =</pre> <table border="0"><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	1	1	1	<pre>>> A.^2</pre> <pre>ans =</pre> <table border="0"><tr><td>1</td><td>4</td><td>9</td></tr><tr><td>16</td><td>25</td><td>36</td></tr><tr><td>49</td><td>64</td><td>81</td></tr></table>	1	4	9	16	25	36	49	64	81										
1	1	1																											
1	1	1																											
1	1	1																											
1	4	9																											
16	25	36																											
49	64	81																											

Operações com matrizes

Alguns comandos adicionais para operações com matrizes:

- ▶ Matriz inversa de uma matriz A : `inv(A)`;
- ▶ Determinante e o número de condição de uma matriz A :
`det(A)`;
- ▶ Traço de uma matriz A : `trace(A)`;
- ▶ Dimensão de uma matriz A : `[m, n] = size(A)`.
Dimensão de um vetor b : `nelementos = length(b)`.

Exercícios

1. Dados as seguintes matrizes e vetores:

$$A = [1 \ 2 \ 3; \ 4 \ 10 \ 6; \ 7 \ 8 \ 19]$$

$$B = [4 \ 5 \ 6; \ 1 \ 2 \ 3; \ 8 \ 7 \ 6]$$

$$c = [4 \ 5 \ 6]$$

$$d = [4; \ 5; \ 6]$$

Calcule:

(a) $D = A + B$

(b) $E = A * B$

(c) $F = A .* B$

(d) $G = A * c'$

(e) $J = \text{inv}(A)$

(f) $I = J * A$

(g) $L = B * d$

(h) Calcule o traço de A.

(i) Calcule a soma dos elementos de c.

Sumário

O quê é GNU Octave ?

Introdução

Iniciando GNU Octave

Estruturas de controle de fluxo

Scripts e functions

Gráficos 2D

Plot de Matrizes

Durante a execução de um programa, frequentemente é necessário realizar operações repetitivas ou percorrer vetores e matrizes, alterando seus valores com base em algum tipo de regra, que precisa ser conferida em algum momento. Veremos alguns estruturas que desempenham essas funções.

- ▶ As estruturas básicas para se implementar um *loop* são `for` e `while`.
- ▶ E as estruturas usadas para testar condições pré-definidas são `if`, `ifelse` e `else`.

Controle de fluxo

Para a função `for`, o *loop* é mantido enquanto a variável `i` não assumir o valor `N`.

Sintaxe básica do `for`

```
for i = 1:N  
    Operações  
end
```

Para a função `while`, o *loop* é mantido enquanto a condição de saída não é atingida.

Sintaxe básica do `while`

```
while (condição satisfeita)  
    Operações  
end
```

Controle de fluxo

A condição 1 do comando `if` é testado e, se for cumprida, realiza os comandos e ignora os comandos seguintes. Senão, a condição 2 é testada, se verdadeira os comandos dentro do `elseif` são realizados. Caso condição 2 seja falsa, o programa executa as instruções contidas no `else`. Essa estrutura pode conter quantos `elseif` forem necessários.

Sintaxe básica do `if`

```
if (condição 1)
    Comandos
elseif (condição 2)
    Comandos
else
    Comandos
end
```

Exemplo for

É comum construções com estruturas de `for` em operações envolvendo vetores e matrizes. Por exemplo, a construção das matrizes A e B a seguir:

```
for i=1:10;  
    for j=1:10;  
        A(i,j)=i+j;  
        B(i,j)=i-j;  
    end  
end  
>> A  
>> B  
>> C=A+B
```

Exemplo while

O laço `while` é executado se a condição testada for verdadeira. Quando o teste se tornar falso o laço terminará. Por exemplo,

```
a = 1;
b = 15;
while a<b;
    a = a+1;
    b = b-1;
end
disp('fim do loop, a=b')
>> a
>> b
```

Exemplo if

As declarações if, else são usadas em situações condicionadas.
Por exemplo,

```
for i=1:5;
  for j=1:5;
    if i == j
      A(i,j)=2;
    if abs(i-j) == 1
      A(i,j)=1;
    else
      A(i,j)=0;
    end
  end
end
end
>> A
```

Comandos e elementos relacionais

Tabela: Comandos e elementos relacionais.

Comando	Função desempenhada
>	Maior
<	Menor
>=	Maior ou igual
<=	Menor ou igual
==	Testa se as variáveis têm conteúdos iguais
=	Testa se as variáveis são diferentes
&&	Operação e lógica
	Operação ou lógico
	Não lógico

Sumário

O quê é GNU Octave ?

Introdução

Iniciando GNU Octave

Estruturas de controle de fluxo

Scripts e functions


Gráficos 2D

Plot de Matrizes

- ▶ Um *script* é um conjunto de comandos que segue uma ordem pré-estabelecida e que pode ser armazenado como um arquivo, para ser reutilizado.
- ▶ Os arquivos devem ser salvos em *m-files*, arquivos com extensão *.m*.
- ▶ Para executar esses comandos, podemos:
 - ▶ Abrir a pasta que contém o arquivo na pasta de trabalho, digitar o nome do arquivo na janela de comando e apertar enter;
 - ▶ Digitar na janela de comando o caminho até o local onde está armazenado o script.

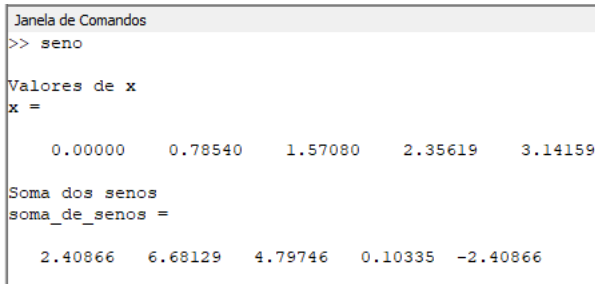
Exemplo de Scripts

O *script* a seguir calcula a soma de senos de diferentes frequência. Note que x varia de 0 a π com passo de $\frac{\pi}{4}$.

```
*seno.m   
1  % Cálculo a soma de senos  
2  
3  x=0:(pi/4):pi;  
4  
5  sen1=3*sin(2*x);  
6  sen2= sin(5*x+pi/2);  
7  sen3=5*sin(x+pi/11);  
8  
9  soma_de_senos =sen1+sen2+sen3;  
10  
11 display('Valores de x')  
12  
13 x  
14  
15 display('Soma dos senos')  
16  
17 soma_de_senos
```

Exemplo de Scripts

Digitando seno na janela de comandos, o *script* será executado:



The screenshot shows a command window titled 'Janela de Comandos'. The user has entered the command '>> seno'. The script output is as follows:

```
Janela de Comandos
>> seno

Valores de x
x =
    0.00000    0.78540    1.57080    2.35619    3.14159

Soma dos senos
soma_de_senos =
    2.40866    6.68129    4.79746    0.10335   -2.40866
```

Figura: Resultado do *Script* de soma de senos.

Functions

- ▶ Em geral, as *functions* são definidas em termo dos parâmetros que recebem (entradas) e dos parâmetros que retornam (saídas).
- ▶ Vantagem das *functions* é a redução de partes repetitivas no código e o reaproveitamento em outras situações.
- ▶ São armazenadas em arquivos `.m` e, obrigatoriamente, o arquivo e a função devem possuir o mesmo nome.

Sintaxe de uma *function*

```
function[saída1,..., saídaM] = nome[entrada1,...,  
entradaN]  
...  
endfunction
```

Exemplo de *functions*

O exemplo a seguir mostra uma função usada para calcular a primeira derivada de um polinômio de grau qualquer. A entrada da função (`deriva_pol`) é um vetor com os coeficientes do polinômio (`coef`) e a saída é um vetor com os coeficientes do polinômio derivado (`deriva`).

```
deriva_pol.m ✖  
1 % Função calcula a derivada de primeira  
2 % ordem de um polinômio qualquer.  
3  
4 function[deriva]=deriva_pol(coef)  
5  
6 ncoef = length(coef);  
7  
8 if ncoef ==1  
9     deriva = 0;  
10 else  
11     for i=1:(ncoef-1)  
12         deriva(i)=coef(i)*(ncoef - i);  
13     end  
14 end  
15 %display('Coeficientes da derivada de primeira ordem')  
16 fprintf('Coeficientes da derivada de primeira ordem do polinomio de grau %d\ ',ncoef-1)  
17 endfunction
```

Figura: *Function* que deriva polinômios.

Exemplo de *functions*

Para testar o código foi utilizado o polinômio

$$p(x) = 2x^5 + 3x^4 + 4x^3 + x^2 + 0x + 10$$

```
Janela de Comandos
>> coef=[2 3 4 1 0 10]
coef =

    2    3    4    1    0   10

>> deriva_pol(coef)
Coeficientes da derivada de primeira ordem do polinomio de grau 5 ans =

   10   12   12    2    0
```

Figura: Resultado da *Function* que deriva polinômios.

Algumas observações:

- ▶ O símbolo de porcentagem é utilizado para fazer comentários no código;
- ▶ O comando `clc` limpa a janela de comandos;
- ▶ Os comandos `clear` e `clear all` limpam o ambiente de trabalho;

Exercícios

1. Defina uma nova *function* chamada *areacirculo* que calcula a área de um círculo. A única entrada é o raio do círculo.
2. Defina uma nova *function* chamada *bhaskara* que determina as raízes de uma equação do segundo grau qualquer, imprimindo a mensagem 'Não possui raiz' se $\Delta < 0$, 'Possui 1 raiz' se $\Delta = 0$, e 'Possui 2 raízes' se $\Delta > 0$.
3. Defina uma nova *function* chamada *fatorial* que determina o fatorial de um número qualquer.

Sumário

O quê é GNU Octave ?

Introdução

Iniciando GNU Octave

Estruturas de controle de fluxo

Scripts e functions

Gráficos 2D

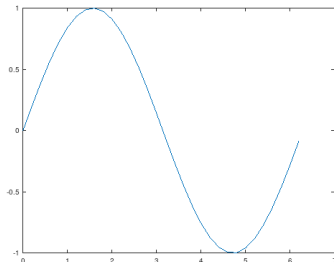
Plot de Matrizes

- ▶ Vamos aprender a fazer gráficos a partir de funções que definimos ou vetores de dados
- ▶ Existem alguns tipos de visualização, de acordo com a dimensão do que se deseja visualizar ou o que se deseja representar
- ▶ Veremos aqui forma de visualizar funções e dados de entrada

- ▶ A função `plot` é mais comumente usada para criar gráficos 2D
- ▶ Em sua forma mais simples, ela recebe como argumentos um vetor com os valores das abscissas (eixo x) e um vetor das ordenadas (eixo y)
- ▶ Ou seja, ela trabalha com a idéia do plano Cartesiano que conhecemos
- ▶ Vejamos um exemplo.

Função Plot

```
>> x = 0:0.5:2*pi;  
>> y = sin(x);  
>> plot(x,y)
```

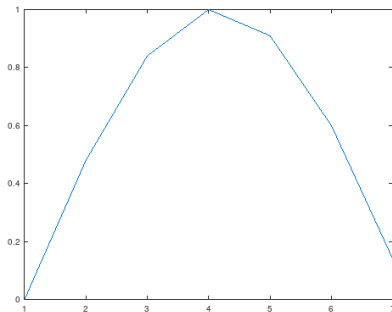


- As dimensões dos vetores x e y devem ser **iguais!**

Função Plot

- Se y é um vetor dado, `plot(y)` produz um gráfico com abscissa variando em uma unidade

```
>> y = [0.0 0.48 0.84 1.0 0.91 0.6 0.14];  
>> plot(y)
```

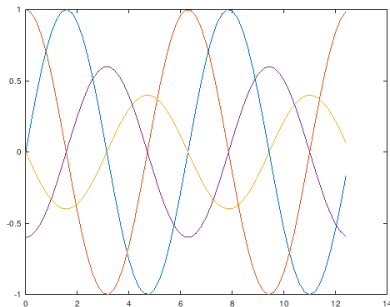


Função Plot

- Também é possível usar a função `plot` com múltiplos argumentos, para criar múltiplos plots no mesmo gráfico

```
>> x = 0:0.2:4*pi;
```

```
>> plot(x,sin(x),x,cos(x),x,0.4*sin(x+pi),x,0.6*cos(x+pi))
```



- ▶ Outra forma de criar múltiplos plots em um gráfico é usando o comando `hold on`

```
>> x = 0:0.2:4*pi;  
>> plot(x,sin(x))  
>> hold on  
>> plot(x,cos(x))  
>> plot(x,0.4*sin(x+pi))  
>> plot(x,0.6*cos(x+pi))  
>> hold off
```

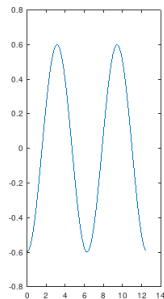
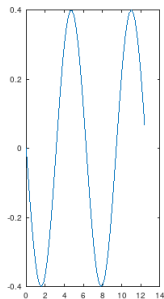
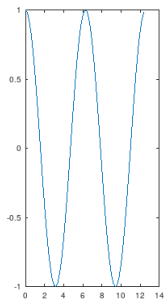
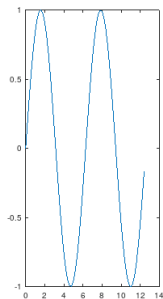
- ▶ o comando `hold off` finaliza a adição de gráficos

Função Plot

- ▶ É possível adicionar vários gráficos em uma mesma janela
- ▶ Para isso, utilizamos a função `subplot(linhas, colunas, gráfico)`
- ▶ Vamos colocar as funções em 4 gráficos na mesma linha

```
>> subplot(1,4,1)
>> plot(x,sin(x))
>> subplot(1,4,2)
>> plot(x,cos(x))
>> subplot(1,4,3)
>> plot(x,0.4*sin(x+pi))
>> subplot(1,4,4)
>> plot(x,0.6*cos(x+pi))
```

Função Plot

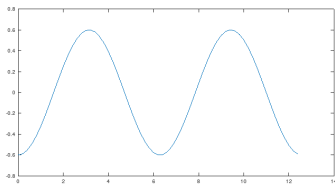
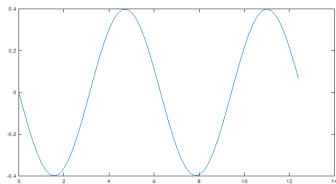
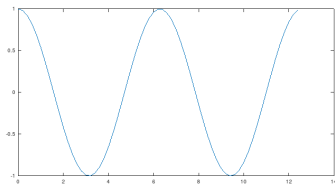
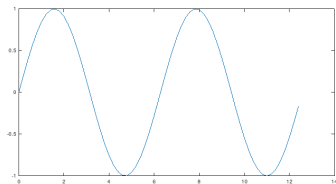


Função Plot

- Vamos colocar as funções em 4 gráficos em duas linhas e duas colunas:

```
>> subplot(2,2,1)
>> plot(x,sin(x))
>> subplot(2,2,2)
>> plot(x,cos(x))
>> subplot(2,2,3)
>> plot(x,0.4*sin(x+pi))
>> subplot(2,2,4)
>> plot(x,0.6*cos(x+pi))
```

Função Plot



Estilos de Linha e Símbolos

- ▶ Podemos alterar as cores e tipos de linha dos gráficos, adicionando um argumento na função `plot`
- ▶ Ou seja, `plot(x,y,'estilo')`. Temos as opções de estilo:

Tipo de linha
-
—
- - - - -
- . - . - . - .

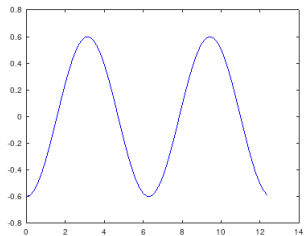
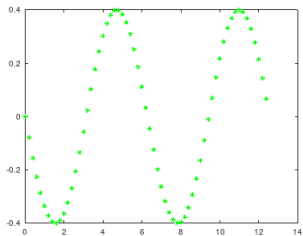
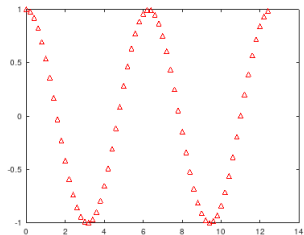
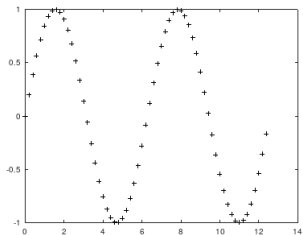
Tipo de ponto	
.
*	*****
o	ooooo
+	+++++
x	xxxxx
^	△△△△
v	▽▽▽▽
s	□□□□□

Cores	
y	amarelo
m	magenta
c	cian
r	vermelho
g	verde
b	azul
w	branco
k	preto

- Vamos adicionar cores e pontos ao gráfico anterior

```
>> subplot(2,2,1)
>> plot(x,sin(x),'+k')
>> subplot(2,2,2)
>> plot(x,cos(x),'^r')
>> subplot(2,2,3)
>> plot(x,0.4*sin(x+pi),'*g')
>> subplot(2,2,4)
>> plot(x,0.6*cos(x+pi),'b')
```

Função Plot

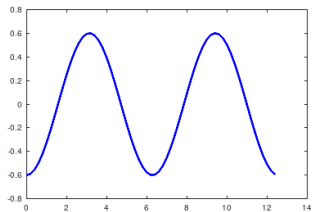
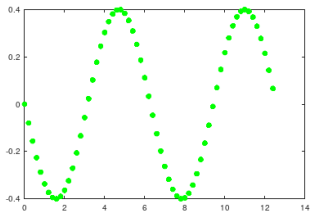
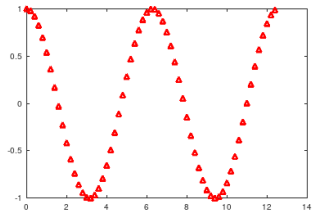
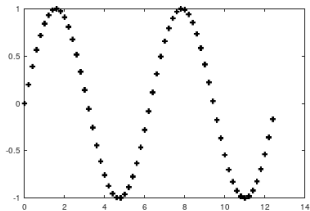


Função Plot

- ▶ Se desejarmos alterar a espessura da linha ou do ponto, adicionamos mais um argumento à função plot
- ▶ Isto é, `plot(x,y,'estilo','linewidth',espessura)`. A espessura é dada por um número, quanto maior o número, maior a espessura

```
>> subplot(2,2,1)
>> plot(x,sin(x),'+k','linewidth',3)
>> subplot(2,2,2)
>> plot(x,cos(x),'^r','linewidth',3)
>> subplot(2,2,3)
>> plot(x,0.4*sin(x+pi),'*g','linewidth',3)
>> subplot(2,2,4)
>> plot(x,0.6*cos(x+pi),'b','linewidth',3)
```


Função Plot

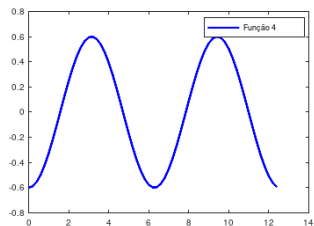
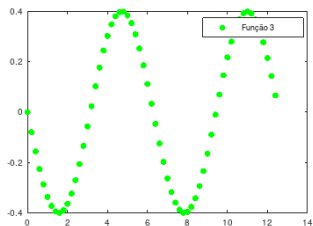
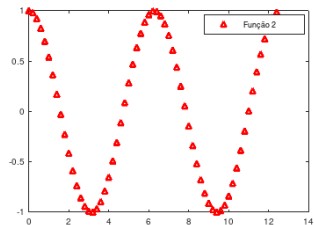
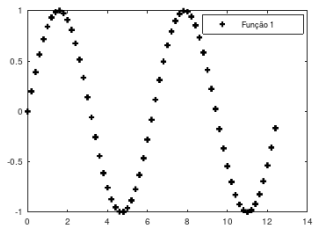


Legenda

- ▶ Adicionando mais um argumento à função plot, criamos uma legenda para o gráfico
- ▶ Isto é, `plot(x,y,'estilo;legenda;')`

```
>> subplot(2,2,1)
>> plot(x,sin(x),'+k;Função 1;', 'linewidth',3)
>> subplot(2,2,2)
>> plot(x,cos(x),'^r;Função 2;', 'linewidth',3)
>> subplot(2,2,3)
>> plot(x,0.4*sin(x+pi),'*g;Função 3;', 'linewidth',3)
>> subplot(2,2,4)
>> plot(x,0.6*cos(x+pi),'b;Função 4;', 'linewidth',3)
```

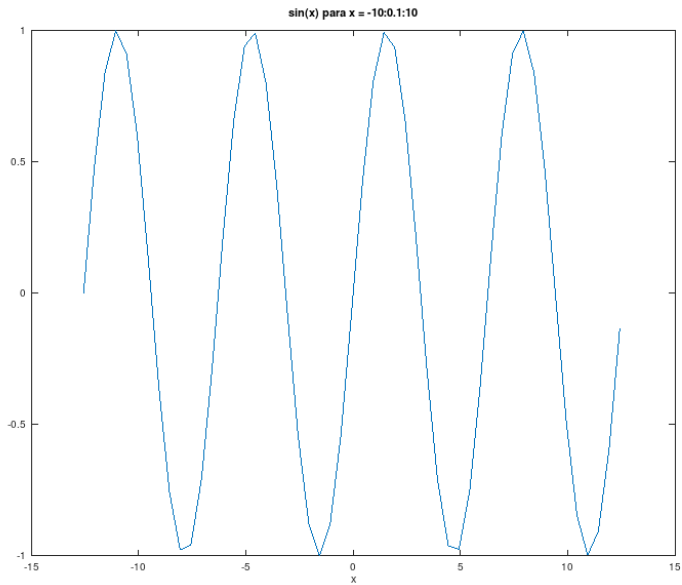
Função Plot



- ▶ Algumas outras opções são interessantes para ajustar o gráfico, como nome do eixo, título e domínio.
- ▶ Os comandos `xlabel` e `ylabel` nomeiam os eixos do gráfico
- ▶ Para adicionar título, usamos `title`

```
>> x = -4*pi:0.5:4*pi;  
>> plot (x, sin (x));  
>> title ("sin(x) para x = -10:0.1:10");  
>> xlabel ("x");  
>> ylabel ("sin (x)");
```

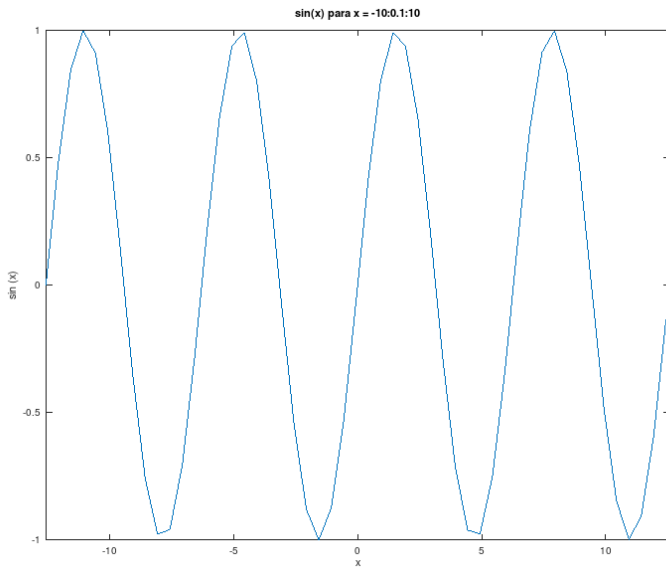
Opções



- ▶ Observamos que o domínio do gráfico não está ajustado
- ▶ Usamos o comando `axis([x1 x2 y1 y2])` para definir os intervalos de x e y
- ▶ Nesse caso, temos `axis([-4*pi 4*pi -1 1])`

```
>> x = -4*pi:0.5:4*pi;  
>> plot (x, sin (x));  
>> axis([-4*pi 4*pi -1 1])  
>> title ("sin(x) para x = -10:0.1:10");  
>> xlabel ("x");  
>> ylabel ("sin (x)");
```

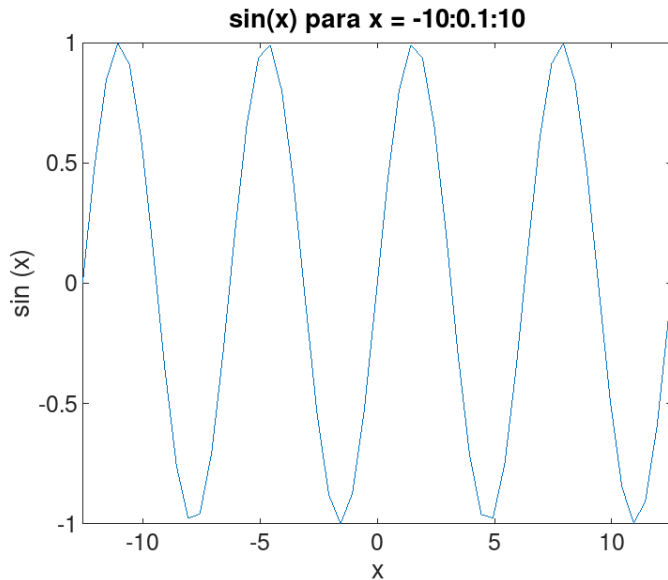
Opções



- ▶ Para finalizar, aumentamos a fonte do texto do gráfico
- ▶ Usamos o comando

```
set(gca,'fontsize',tamanho da fonte)
```

```
>> x = -4*pi:0.5:4*pi;  
>> plot (x, sin (x));  
>> axis([-4*pi 4*pi -1 1])  
>> title ("sin(x) para x = -10:0.1:10");  
>> xlabel ("x");  
>> ylabel ("sin (x)");  
>> set(gca,'fontsize',22)
```

- Temos ainda outras funções de gráfico 2D

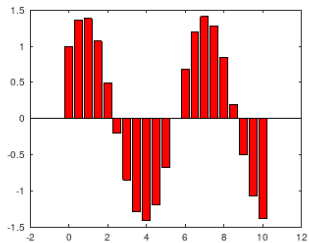
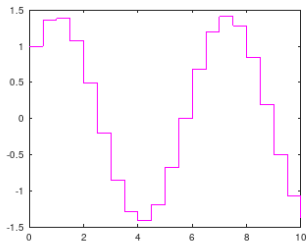
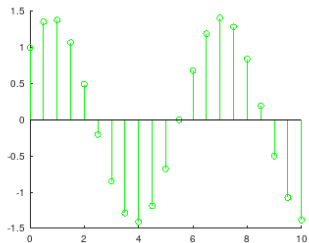
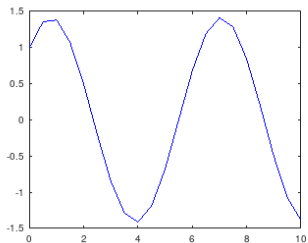
Comando	Descrição
<code>bar</code>	gráfico de barras
<code>stem</code>	sequência discreta
<code>stairs</code>	plot em degraus

Exercício

- Plote a função $\sin(x) + \cos(x)$ para $x=0:0.5:10$ com as funções `plot`, `bar`, `stem` e `stairs` em cores diferentes, utilizando o `subplot`.

```
>> x = 0:0.5:10;  
>> y = sin(x) + cos(x);  
>> subplot(2,2,1)  
>> plot(x,y,'b')  
>> subplot(2,2,2)  
>> stem(x,y,'g')  
>> subplot(2,2,3)  
>> stairs(x,y,'m')  
>> subplot(2,2,4)  
>> bar(x,y,'r')
```

Exercício

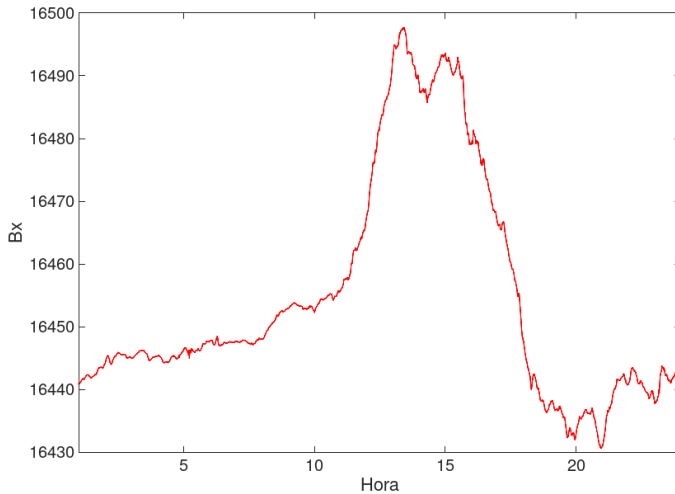


Exercício

- ▶ Baixe o arquivo do link tinyurl.com/oficinaOctave
- ▶ Leia o arquivo com o comando `load`
- ▶ Plote a primeira coluna do arquivo, ajuste o domínio e adicione o nome dos eixos

```
>> f = load('vss20191007pmin.dat');  
>> n = length(f(:,1));  
>> x = linspace(0,24,n)  
>> plot(x,f(:,1),'r','linewidth',2)  
>> axis([1 24])  
>> xlabel ("Hora");  
>> ylabel ("Bx");  
>> set(gca,'fontsize',20)
```

Exercício



Sumário

O quê é GNU Octave ?

Introdução

Iniciando GNU Octave

Estruturas de controle de fluxo

Scripts e functions

Gráficos 2D

Plot de Matrizes

Imagens

- ▶ Matrizes são aplicáveis a diversas situações do dia-a-dia
- ▶ Um tipo de matriz muito utilizado por nós é uma imagem
- ▶ Uma imagem pode ser transformada em uma matriz e vice-versa
- ▶ Cada pixel da imagem é associado a uma entrada da matriz

Imagens

- ▶ Escolha uma imagem qualquer e salve no diretório ou baixe a imagem em tinyurl.com/oficinaOctave1
- ▶ O Octave lê essa imagem com o comando `imread('image')`

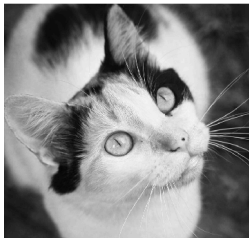
```
>> img = imread('cat.png');  
>> imshow(img)
```



- ▶ Podemos observar que a matriz da imagem tem 3 dimensões

Imagens

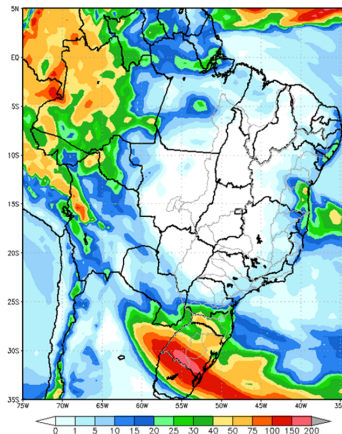
- ▶ Vamos visualizar as matrizes
`img(:,:,1)`, `img(:,:,2)` e `img(:,:,3)`



- ▶ Cada uma dessas matrizes representa um canal RGB (Red, Blue, Green)

Imagens

- ▶ Quando lidamos com dados numéricos, não tratamos mais desses canais
- ▶ Dados numéricos podem ser obtidos de **simulações numéricas**
- ▶ Uma simulação numérica que usamos diariamente é a **previsão de tempo**



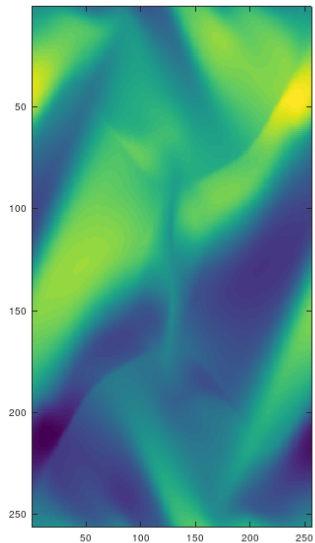
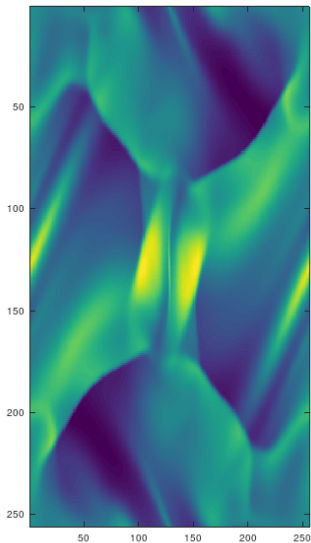
Imagens

- ▶ Para cada valor (x,y) é associado uma posição da matriz
- ▶ A cada posição da matriz, é associada uma cor
- ▶ Essas cores não são mais obtidas por RGB, mas por uma paleta de cores pré-definida chamada `colorbar`
- ▶ Nesse caso, as imagens são visualizadas com a função `imagesc`, que permite a visualização após a aplicação de uma escala de cores

- ▶ Faça o download dos dados em tinyurl.com/oficinaOctave2
- ▶ Vamos carregar os arquivos e visualizá-los

```
>> var1 = load('density.dat');  
>> var2 = load('By.dat');  
>> subplot(1,2,1)  
>> imagesc(var1)  
>> subplot(1,2,2)  
>> imagesc(var2)
```

Imagens



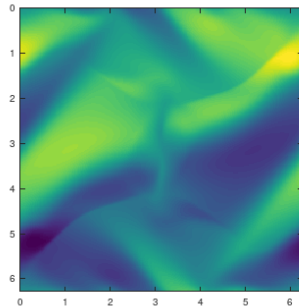
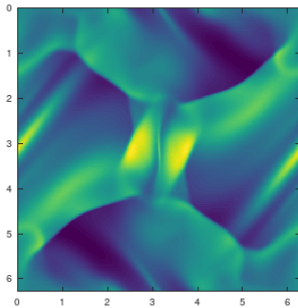
Imagens

- ▶ Essas imagens possuem domínio quadrado $[0 \ 2\pi \ 0 \ 2\pi]$ e dimensão 256x256, então vamos ajustá-las

```
>> x = linspace(0,2*pi,256);  
>> y = linspace(0,2*pi,256);  
>> subplot(1,2,1)  
>> imagesc(x,y,var1)  
>> axis([0 2*pi 0 2*pi], 'square')  
>> subplot(1,2,2)  
>> imagesc(x,y,var2)  
>> axis([0 2*pi 0 2*pi], 'square')
```

- ▶ O comando square dentro do axis, força a imagem a ser quadrada

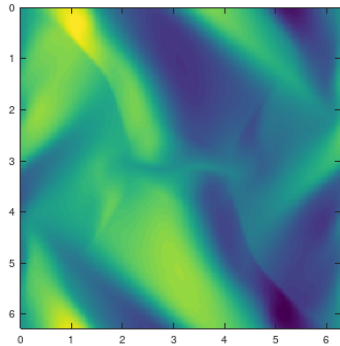
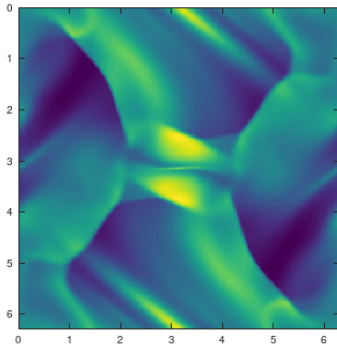
Imagens



- As imagens estão rotacionadas 90 graus, vamos ajustá-las utilizando a função `rot90`

```
>> x = linspace(0,2*pi,256);  
>> y = linspace(0,2*pi,256);  
>> var1 = rot90(var1);  
>> var2 = rot90(var2);  
>> subplot(1,2,1)  
>> imagesc(x,y,var1)  
>> axis([0 2*pi 0 2*pi], 'square')  
>> subplot(1,2,2)  
>> imagesc(x,y,var2)  
>> axis([0 2*pi 0 2*pi], 'square')
```

Imagens



Imagens

- ▶ Vamos adicionar uma barra de cores `colorbar` e alterar a escala de cores das imagens `colormap`
- ▶ Algumas opções de paleta de cores:

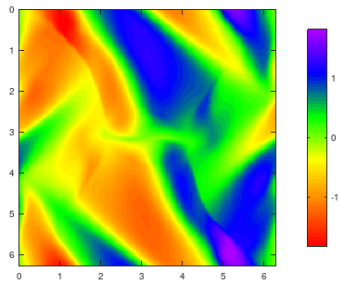
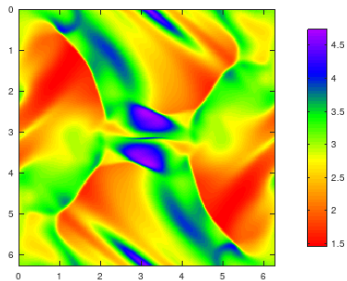
Colormaps		
viridis	summer	colorcube
jet	autumn	ocean
cubehelix	winter	lines
hsv	gray	prism
rainbow	bone	hot
cool	copper	pink
spring	flag	white

Imagens

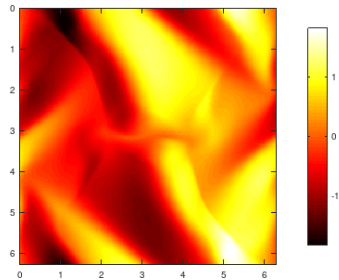
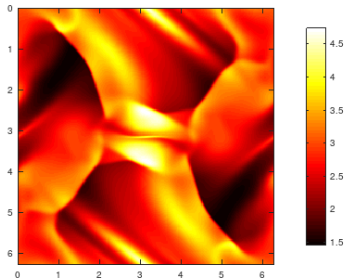
- ▶ A função `imagesc` é adicionada depois de cada imagem
- ▶ A função `colormap` é adicionada no final do código

```
>> x = linspace(0,2*pi,256);  
>> y = linspace(0,2*pi,256);  
>> var1 = rot90(var1);  
>> var2 = rot90(var2);  
>> subplot(1,2,1)  
>> imagesc(x,y,var1)  
    >> colorbar  
>> axis([0 2*pi 0 2*pi], 'square')  
>> subplot(1,2,2)  
>> imagesc(x,y,var2)  
    >> colorbar  
>> axis([0 2*pi 0 2*pi], 'square')  
    >> colormap(rainbow)
```

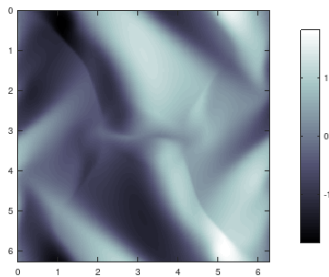
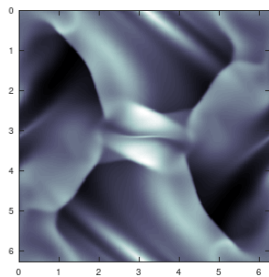
Imagens - RAINBOW



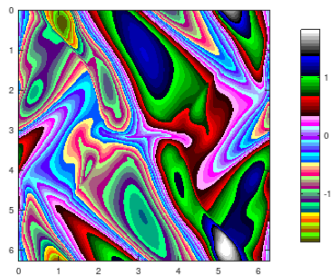
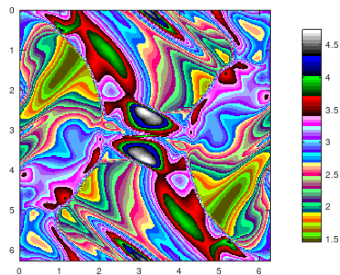
Imagens - HOT



Imagens - BONE



Imagens - COLORCUBE

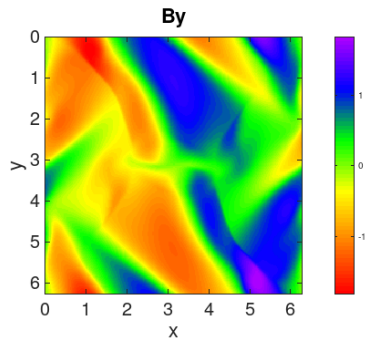
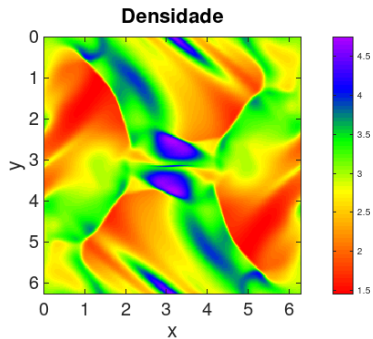


- Vamos aumentar a fonte do gráfico com `set(gca,'fontsize',20)`, adicionar o nome dos eixos e o título

```
>> subplot(1,2,1)
>> imagesc(x,y,var1)
>> title('Densidade')
>> xlabel('x')
>> ylabel('y')
>> colorbar
>> axis([0 2*pi 0 2*pi],'square')
>> set(gca,'fontsize',20)
```

```
>> subplot(1,2,2)
>> imagesc(x,y,var2)
>> title('By')
>> xlabel('x')
>> ylabel('y')
>> colorbar
>> axis([0 2*pi 0 2*pi],'square')
>> colormap(rainbow)
>> set(gca,'fontsize',20)
```

Imagens - COLORCUBE



- ▶ Para finalizar, salvamos a imagem criada com o comando:
- ▶ `print('figura.png', '-dpng');`

C'est fini

Referências

- ▶ http://www2.fct.unesp.br/docentes/cartogalo/_Softwares_Free/Octave/Tutorial_Octave_Unesp.pdf
- ▶ <https://octave.sourceforge.io/octave/function/colormap.html>
- ▶ https://edisciplinas.usp.br/pluginfile.php/256601/mod_resource/content/1/apostila_matlab_octave.pdf
- ▶ <http://panorama.comerc.com.br/en/2019/09/12/dry-weather-prevails-across-brazil/>