DACC | Departamento Acadêmico de Ciência da Computação

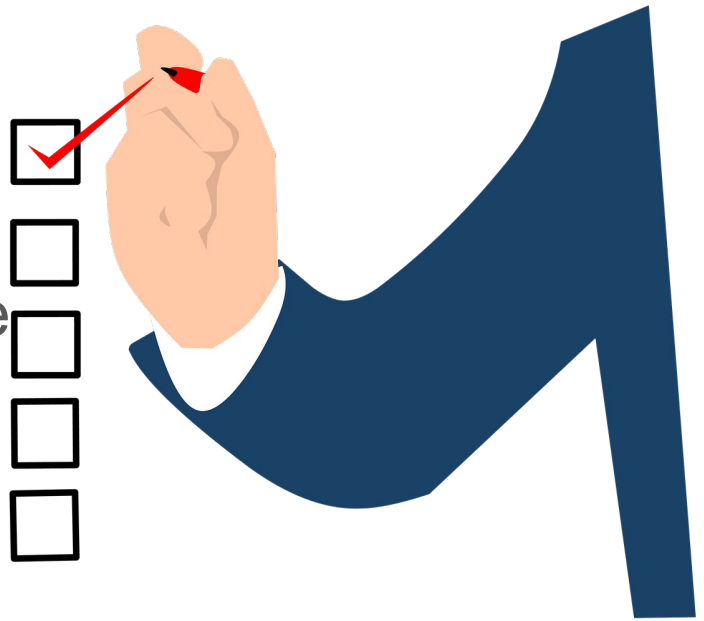
FUNDAÇÃO UNIVERSIDADE FEDERAL DE RONDÔNIA

Cálculo Numérico

Professor:
Dr. Lucas Marques da Cunha
lucas.marques@unir.br

Roteiro

1. Apresentação
2. Horário
3. Metodologia de Avaliação
4. Projeto Final
5. Cronograma de Aulas
6. Dúvidas e/ou sugestões
7. Introdução ao GNU Octave
   a. Operadores, Funções, Gráficos, Estruturas de controle de fluxo e decisão.
8. Exercícios práticos



Apresentação

1. Você já cursou Cálculo Numérico anteriormente? Se sim, qual foi sua experiência?
2. Você consegue enxergar a relação e/ou aplicação da Cálculo Numérico e a Computação?

Horário

COMPONENTE CURRICULAR: Cálculo Numérico

PERÍODO: 5º

CARGA HORÁRIA: 80H

HORÁRIO:

Presencial:
Segunda-feira (13h50min - 18h10min) - Tarde
Assíncrona:
Sábado (13h50min) - Tarde

Metodologia de avaliação

1. Durante o período o letivo serão realizadas 02 (duas) avaliações.
2. **A primeira a avaliação será composta de:**
   a. Prova escrita (peso 7,0)
   b. Atividades assíncronas (peso 3,0)
3. **A segunda avaliação será composta de:**
   a. Projeto Final (Peso 5,0)
   b. Prova escrita (Peso 3,0)
   c. Atividades assíncronas (Peso 2,0)
4. Se **média final < 60** o aluno fará avaliação repositiva nos termos regimentais da UNIR.
5. A **avaliação repositiva** irá substituir o menor das notas ($M1$ ou $M2$). Então se calculará novamente a **média final**.

Critérios para o desenvolvimento do projeto

Critérios para o desenvolvimento do projeto

1. A equipe deve ser composta por, no máximo, **3 membros**.
2. A equipe deverá selecionar um **tópico** referente aos conteúdos ministrados na disciplina:
   a. Representação de números;
   b. Zeros de funções;
   c. Sistemas lineares e não lineares;
   d. Interpolação Polinomial e Regressão Linear
   e. Derivação numérica
   f. Integração numérica

3. Como deve ser feito:
   a. Software aplicativo
   b. Short paper
   c. Publicação científica

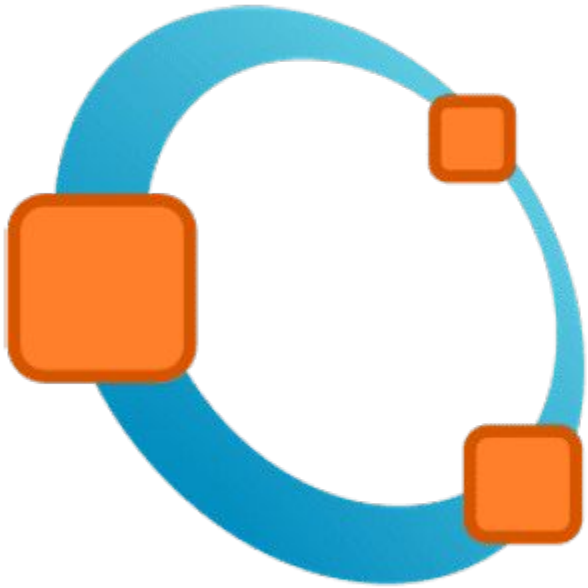
4. **Prazo para entrega: a definir!**

Cronograma de aulas

Dúvidas, sugestões?

Introdução ao GNU Octave

UNIR



GNU Octave

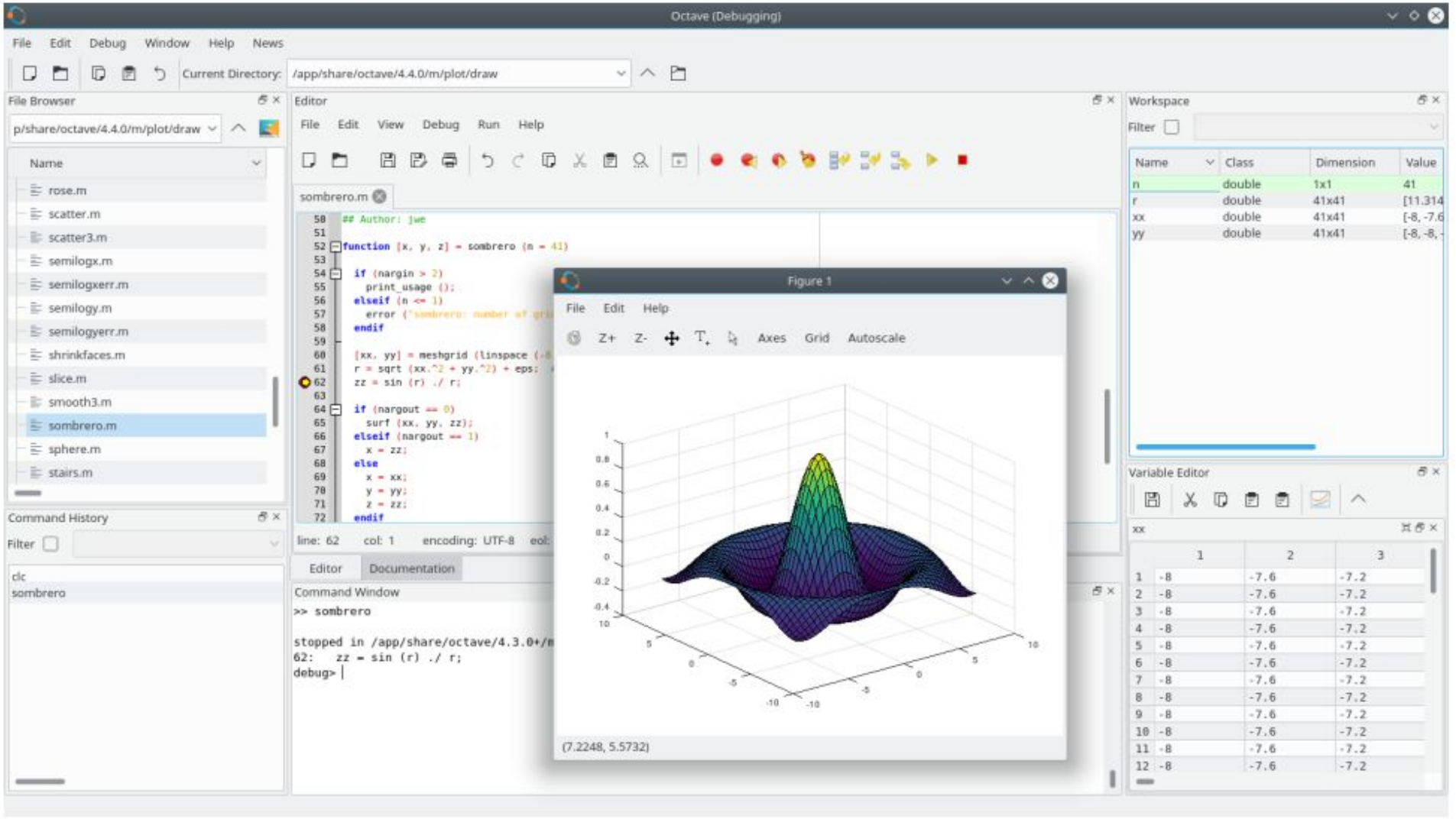
Objetivos de aprendizagem

- Aprender ferramentas e IDE’s para programação científica Octave;
- Estudar conceitos teóricos e práticos sobre variáveis e operadores aritméticos.
- Estudar conceitos teóricos e práticos sobre vetores e matrizes.

O que é Octave?

- O Octave (oficialmente GNU-Octave) é uma **linguagem de alto nível interpretada** que destina-se principalmente para cálculos numéricos.
- O Octave é um “clone” **open-source** do Matlab.
- O Octave pode ser usado como uma “calculadora com muitos recursos” ou como um ambiente de programação.
- Interativo e expansível;
- Disponível online: https://octave-online.net/

Qual a cara do Octave?



Comandos básicos

- **Símbolo >>**
  - Ao abrir o Octave, o símbolo >> aparece na Janela de Comandos.
  - Ele indica que o programa está aguardando comandos.
  - Se o símbolo >> não estiver aparecendo, significa que o Octave ainda está rodando algum programa.

Exercício

Digite: `2+3 <enter>`

Resposta: `ans = 5`

A resposta (ANSwer) do comando que você digitou é 5.

Agora digite: `2+3*5 <enter>`

Resposta: `ans = 17`

E agora: `(2+3)*5`

Resposta: `ans = 25`

GNU Octave

Comandos básicos

- **Janela de comandos**
  - A Janela de Comandos é onde você usa o Octave de forma interativa.

```
Janela de Comandos
>> 1/7
ans =  0.14286
>> format long
>> 1/7
ans =    1.42857142857142e-01
>>
```

Comentário, variáveis e o “;”

```
>> # O Octave ignora tudo que vier após o caracter #, assim
>> # isso é uma ótima forma de acrescentar comentários
>>
>> p1 = 8.6;   # note que o valor de p1 não é exibido
>> p2 = 7.3    # porém o valor de p2 sim
p2 =  7.3000
>> mt = 9.5;
>>
>> m = 0.3 * p1 + 0.4 * p2 + 0.3 * mt;
>> m
m =  8.3500
>>
```

O comando **diary**

- O comando **diary** permite gravar em um arquivo texto um diário (ou um registro) de tudo que for feito dentro do Octave.

```
>> y = sin(8*pi/9)
y =  0.34202
>> diary registro.txt     # Salvando a partir daqui
>> x = 28;
>> v = sin(x)^2 + cos(x)^2
v =  1
>> diary off              # Encerra o salvamento em disco
>> x+y
ans =  28.342
```

Operadores aritméticos

Operação	Símbolo	Exemplo
Adição	+	5 + 3
Subtração	–	5 – 3
Multiplicação	*	5 * 3
Divisão	/	5 / 3
Exponenciação	^	5 ^ 3 (significa $5^3 = 125$)

Operações matemáticas

- Ordem em que o Octave faz as operações:

Ordem	Operação Matemática
Primeiro	Parênteses. Para vários parênteses, o que estiver por dentro é executado primeiro
Segundo	Exponenciação
Terceiro	Multiplicação, divisão (mesma ordem)
Quarto	Adição e subtração

- Se duas ou mais operações tiverem a mesma ordem de precedência, a expressão mais à esquerda será executada primeiro.

Operadores lógicos

- No GNU Octave, o valor lógico verdadeiro é escrito como **true** e o valor lógico falso como **false**.
- Temos os seguintes operadores lógicos disponíveis:
  - & e lógico
  - | ou lógico
  - ! negação
  - == igualdade
  - != diferente
  - < menor que
  - > maior que
  - <= menor ou igual que
  - >= maior ou igual que

Funções matemáticas elementares

UNIR

- `exp(x) - e^x`
- `abs(x) - valor absoluto`
- `log(x) - logaritmo natural (base e)`
- `log10(x) - logaritmo na base 10`
- `sqrt(x) - raiz quadrada`
- `factorial(x) - x!`
- `sin(x) - seno (x em radianos)`
- `sind(x) - seno (x em graus)`

GNU Octave

Funções matemáticas elementares

UNIR

- `cos(x) - coseno (x em radianos)`
- `cosd(c) - coseno (x em graus)`
- `tan(x) - tangente (x em radianos)`
- `tand(x) - tangente (x em graus)`
- `cot(x) - cotangente (x em radianos)`
- `cotd(x) - cotangente (x em graus)`
- `mod(x,y) - resto da divisão inteira entre x e y.`

GNU Octave

Exercícios

1. **Experimente as operações abaixo no Octave. Entenda os resultados.**
   a. ~0
   b. ~1
   c. ~4
   d. [1 0 1] & [1 0 0]
   e. [1 3 4] & [1 2 1]
   f. 1 + (4 <=5)
   g. x = 4; y = x; x == y
   h. x = 3; y = 4; x == y
   i. 1 < 4; y = ans; z = y == 1

GNU Octave

Exercícios

2. **Informe o resultado das duas expressões abaixo:**
   a. [1 1 1 1] & [0 0 1 0];
   b. [1 1 1 1] | [0 0 1 0];
3. **Informe o resultado das expressões:**
   a. ~4
   b. !0
   c. mod(4,4)
   d. mod(4.5,5)
   e. x = 1:10; y = x .^2;

Limpar memória e comandos

- **Comando `clc` ou ctrl+l:**
  - Limpa os comandos exibidos na Janela de Comandos.
- **Comando clear:**
  - Limpa a memória (todas as operações realizadas).

Undefined variáveis

- **O Octave é case sensitive, ou seja, faz distinção entre letras maiúsculas e minúsculas.**
- **Erro comum:** declara-se uma variável com letras minúsculas e na hora de utilizá-la, usa-se letra maiúscula.
- Exemplo:

  >> a = 2;

  >> A

- Aparecerá a mensagem: `error: 'A' undefined near line 1 column 1`

Regras para nomes de variáveis

UNIR

- Podem conter até 63 caracteres;
- Podem conter letras, números e o caractere sublinhar;
- Devem iniciar com uma letra;
- Evite usar nomes de funções nativas do Octave para nomear variáveis (p.ex. cos, sin, exp, sqrt, etc.)
- **O Octave é case sensitive, ou seja, faz distinção entre letras maiúsculas e minúsculas.**

Exercícios

- Resolver a equação quadrática:

1) $3x^2 - 2x - 1 = 0 \rightarrow (a = 3, b = -2, c = -1)$

$\triangle = (-2)^2 - 4 \cdot 3 \cdot (-1)$

$\triangle = 4 + 12 = 16$

$$x = \frac{2 \pm \sqrt{16}}{6} \begin{cases} x_1 = \frac{2+4}{6} = \frac{6}{6} = 1 \\ x_2 = \frac{2-4}{6} = -\frac{2}{6} = -\frac{1}{3} \end{cases}$$

2) $-x^2 + 6x - 9 = 0 \rightarrow (a = -1, b = 6, c = -9)$

$\triangle = 6^2 - 4 \cdot (-1) \cdot (-9)$

$\triangle = 36 - 36 = 0$

$$x = \frac{-6 \pm 0}{-2}$$

$$x_1 = x_2 = \frac{-6}{-2} = 3$$

Utilize `format long`.

GNU Octave

Vetores e Matrizes

- Seja a seguinte matriz A 3x4:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 0 & 8 & 7 \end{bmatrix}$$

- Neste exemplo a matriz A é uma matriz de três linhas e quatro colunas.
- Para criar a matriz A no octave, executar:
  - >> A = [1 2 3 4; 5 6 7 8; 9 0 8 7]
- Observe que o ponto e vírgula ; serve para separar as linhas da matriz.

Componentes de uma matriz

- Na linguagem matemática, a matriz definida anteriormente tem os seguintes componentes:

$$a_{11} = 1, a_{12} = 2, a_{13} = 3, a_{14} = 4$$
$$a_{21} = 5, a_{22} = 6, a_{23} = 7, a_{24} = 8$$
$$a_{31} = 9, a_{32} = 0, a_{33} = 8, a_{34} = 7$$

- No octave é possível ter acesso ao componente `aij` pondo na linha de comando A(i, j).
- Assim, por exemplo:

```
>> A(1,1)
ans = 1
>> A(2,4)
ans = 8
```

Linhas e colunas de uma matriz

- No octave é possível selecionar linha ou coluna de uma matriz.
- Por exemplo:

```
>> A(1, :)
ans = 1 2 3 4
>> A(3, :)
>> ans = 9 0 8 7
```

Por sua vez, a coluna 2 é obtida com o comando:

```
>> A(:, 2)
2
6
0
```

Definição de um vetor coluna

- Um vetor coluna é uma matriz com uma coluna e uma ou mais linhas. Por exemplo:

$$v = \begin{bmatrix} 4 \\ 3 \\ 10 \end{bmatrix}$$

- executar na linha de comandos:

```
>> v = [4; 3; 10]
```

Definição de um vetor linha

- Um vetor linha (ou vetor deitado) é uma matriz com uma linha e uma ou mais colunas. Por exemplo:

```
w = [1 -2 3 -4]
```

- executar na linha de comandos:

```
>> w = [1 -2 3 -4]

w =

1 -2 3 -4
```

Componentes de um vetor

- Na linguagem matemática, o vetor coluna v tem os seguintes componentes:

$$v = \begin{bmatrix} 4 \\ 3 \\ 10 \end{bmatrix}$$

```
v1 = 4, v2 = 3, v3=10
```

- No octave, o componente v(*i*) pode ser obtido com o comando v(i). Assim, por exemplo:

```
>> v(1)

ans = 4
```

Transposta de uma matriz

- Seja a seguinte matriz definida no Octave:

```
>> A = [1 2 3 4; 5 6 7 8; 9 0 8 7];
```

- A transposta de A e obtida com o comando:

```
>> A'
ans =

   1   5   9
   2   6   0
   3   7   8
   4   8   7
```

GNU Octave

Transposta de uma matriz

- A transposta de um vetor coluna é um vetor linha. A transposta de um vetor linha é um vetor coluna.
- Experimente executar a seguinte sequência de comandos:

```
>> v = [1; 2; 3]

>> w = [1 2 3]

>> v'

>> w'
```

Combinação linear de matrizes e vetores

- Sejam as seguintes matrizes, vetores definidos no Octave:

```
>> A = [1 2; 3 4; 5 6; 7 8];
>> B = [9 0; 1 -2; 3 -4; 5 -6];
>> a = [1; 2; 3; 4];
>> b = [5; -6; 7; -8];
```

- A combinação linear de vetores 3a - 2b calcula-se em octave da seguinte forma:

```
>> 3 * a - 2 * b
```

GNU Octave

Combinação linear de matrizes e vetores

- A combinação linear de matrizes 3A - 2B é:

```
>> 3*A - 2*B
ans =

  -15    6
    7   16
    9   26
   11   36
```

Produto de matrizes

- Sejam as seguintes matrizes e vetores definidos no Octave:

```
>> A = [1 2 3 4; 5 6 7 8; 9 0 8 7];
>> B = [1 2; 3 4; 5 6; 7 8];
>> w = [1; 2; 3; 4];
>> v = [1 2 3];
```

Produto de matrizes

- O produto AB (matriz x matriz) pode ser calculado com o comando:

```
>> A * B
```

- cujo resultado é:

```
ans =
```

$$\begin{matrix} 50 & 60 \\ 114 & 140 \\ 98 & 122 \end{matrix}$$

Produto de matrizes

- O produto Aw (matriz x vetor) se obtém fazendo:

```
>> A   * w
```

- cujo resultado é:

```
ans =
30
70
61
```

- O produto vA (vetor x matriz) pode-se obter com:

```
>> v * A
```

cujo resultado é:

```
ans =
38 14 41 41
```

Sistemas Lineares

- **Resolução de um sistema linear “no chute”**
- Seja o seguinte sistema linear:

$$\begin{array}{rcrcrcl} 12x_1 & + & 3x_2 & - & 4x_3 & = & 11 \\ 24x_1 & - & x_2 & - & 6x_3 & = & 29 \\ -12x_1 & - & 17x_2 & + & 14x_3 & = & 45 \end{array}$$

- O sistema acima pode ser escrito como Ax = b onde:

$$A = \begin{bmatrix} 12 & 3 & -4 \\ 24 & -1 & -6 \\ -12 & -17 & 14 \end{bmatrix} \quad e \quad b = \begin{bmatrix} 11 \\ 29 \\ 35 \end{bmatrix}$$

Sistemas Lineares

- **Resolução de um sistema linear com o algoritmo de eliminação**
- Seja o seguinte sistema linear:

$$A = \begin{bmatrix} 12 & 3 & -4 \\ 24 & -1 & -6 \\ -12 & -17 & 14 \end{bmatrix} \quad e \quad b = \begin{bmatrix} 11 \\ 29 \\ 35 \end{bmatrix}$$

- Para resolução, deve-se gerar a matriz aumentada:

$$\begin{bmatrix} 12 & 3 & -4 & 11 \\ 24 & -1 & -6 & 29 \\ -12 & -17 & 14 & 45 \end{bmatrix}$$

GNU Octave

Sistemas Lineares

- **Passo 01:**
  - Definir pivot = MA(1, 1)
  - Definir multiplicador = -MA(2,1)/pivot
  - Substituir L2 por multiplicador x L1 + L2
  - Definir multiplicador = -MA(3,1)/pivot
  - Substituir L3 por multiplicador x L1 + L3
  - Matriz resultante:

$$\begin{matrix} 12 & 3 & -4 & 11 \\ 0 & -7 & 2 & 7 \\ 0 & -14 & 10 & 56 \end{matrix}$$

Sistemas Lineares

- **Passo 02:**
  - Definir pivot = MA(2, 2)
  - Definir multiplicador = -MA(2,1)/pivot
  - Substituir L2 por multiplicador x L1 + L2
  - Definir multiplicador = -MA(3,1)/pivot
  - Substituir L3 por multiplicador x L1 + L3
  - Matriz resultante:

$$\begin{matrix} 12 & 3 & -4 & 11 \\ 0 & -7 & 2 & 7 \\ 0 & -14 & 10 & 56 \end{matrix}$$

Estrutura de decisão e repetição

- **Instrução de decisão if.**
- Execute o código abaixo e informe qual saída é apresentada no Octave. Explique!

```
i = 2;
if ( i == 1 )
    disp ( " Hello ! " );
elseif ( i == 2 )
    disp ( " Goodbye ! " );
elseif ( i == 3 )
    disp ( " Tchau ! " );
else
    disp ( " Au Revoir ! " );
endif
```

Estrutura de decisão e repetição

- **Instrução de repetição for.**
- Execute o código abaixo e informe qual saída é apresentada no Octave. Explique!

A

```
for i = 1:5
    disp(i);
endfor
```

B

```
for j = 1:2:8
    disp(j);
endfor
```

C

```
for k = 10:-3:1
    disp(k);
endfor
```

D

```
for i = 1:3
    for j = 1:3
        disp([i,j]);
    endfor
endfor
```

Estrutura de decisão e repetição

- **Instrução de repetição while.**
- Execute o código abaixo e informe qual saída é apresentada no Octave. Explique!

```
s = 0;
i = 1;
while ( i <= 10 )
   s = s + i;
   i = i + 1;
endwhile
```

Funções

- **Exemplo 1:** Seja $f(x) = x^3 - 9x + 3$. Vamos analisar o sinal desta função.
- Construindo uma tabela de valores para f(x) e considerando apenas os sinais, temos:

x	$-\infty$	-100	-10	-5	-3	-1	0	1	2	3
$f(x)$	-	-	-	-	+	+	+	-	-	+

Funções

- **No editor:**

```
function [y] = f(x)
   y=x.^3-9*x+3;
endfunction
```

- Para podermos chamar essa função na janela de comandos, o arquivo com a função deve ter o mesmo nome dado à função (ex. f) e a extensão .m.
- Além disso, é necessário que estejamos com a pasta na qual a função foi salva aberta.

Funções

- **Na janela de comandos:**

```
>> f(0)
ans=3
>> [y]=f(0)
y=3
```

Funções

- Podemos calcular o valor de f(x) para vários valores de x.

```
>> x=-5:5
```

```
>> y = f(x)
```

- Qual resultado é apresentado após a execução dos comando acima?
- Para introduzir mais pontos entre -5 e 5, podemos fazer:

```
>> x=-5:0.5:5
```

A estrutura SE

- Uma das estruturas que podemos utilizar no Octave é a se, dada pela instrução if seguida de uma expressão:

```
> if expressao
>   bloco_de_comandos;
> end
```

As estruturas SE, SENÃO, SENÃO SE

- Uma das estruturas que podemos utilizar no Octave é a se, senão e senão se, dadas pelas instruções if, else e elsif seguida de uma expressão:

```
> if expressao
>   bloco_de_comandos;
> end
```

```
> if expressao
>   bloco_de_comandos;
> else
>   outro_bloco;
> end
```

```
> if expressao
>   bloco_de_comandos;
> elseif outra_expressao
>   outro_bloco;
> else
>   outro_bloco_ainda;
> end
```

As estruturas SE, SENÃO, SENÃO SE

UNIR

- Exemplo:

```
> limite_baixo = 18;
> limite_alto = 60;
> idade_usuario = input('Digite sua idade: ');
> if idade_usuario < limite_baixo
>   disp('Usuário fora do grupo (limite menor). Desculpe!');

> elseif idade_usuario > limite_alto
>   disp('Usuário fora do grupo (limite maior). Desculpe!')
> else
>   disp('Usuário entra no grupo!')
> end
```

GNU Octave

A estrutura SWITCH

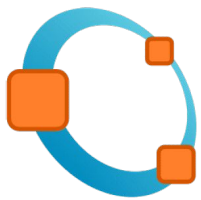
- Imagine que precisamos classificar os elementos em cinco grupos entre 18 e 60 anos.
- Para evitar o uso excessivo de ifs e elses, utilizaremos a estrutura SWITCH.

```
> switch expressao
>   case cond_1
>      bloco_comandos;
>   case cond_2
>      outro_bloco;
>   case cond_3
>      mais_um_bloco;
    ...
>   otherwise
>      outro_bloco_ainda;
> end
```

A estrutura SWITCH

- Exemplo 01:

```
letra_base = input('Indique a letra da base [ACGT]: ', 's');
switch letra
    case {'a' 'A'}
        disp('Adenina');
    case {'c' 'C'}
        disp('Citosina');
    case {'g' 'G'}
        disp('Guanina');
    case {'t' 'T'}
        disp('Timina');
    otherwise
        disp('Não existe nenhuma base correspondente!');
    end
end
```



GNU Octave

A estrutura SWITCH

- Exemplo 02:

```
> idade_usuario = input('Digite sua idade: ');
> switch (idade_usuario)
>   case num2cell([18:21])
>      disp('Usuário no Grupo 1');
>   case num2cell([22:30])
>      disp('Usuário no Grupo 2');
>   case num2cell([31:40])
>      disp('Usuário no Grupo 3');
>   case num2cell([41:50])
>      disp('Usuário no Grupo 4');
>   case num2cell([51:60])
>      disp('Usuário no Grupo 5');
>   otherwise
>      disp('Usuário nao cumpre requerimentos. Desculpe!');
> end
```

GNU Octave

A estrutura PARA

- Uma das estruturas de repetição é a declaração para, dada pela instrução **for** :

```
> for iter = intervalo
>   bloco_comandos;
> end
```

- Exemplo:

```
> for iter = 1:10
>   disp(sin(iter))
> end
```

GNU Octave

A estrutura PARA

- Exemplo 01: Execute o código abaixo e indique sua saída:

```
> tam_grupo = 8;
> idade_grupo = zeros(1, tam_grupo);
> for elem = 1:tam_grupo
>   idade_grupo(elem) = input('Digite sua idade: ');
> end
```

GNU Octave

A estrutura PARA

```
function classgruposid()
    %{
    classgruposid()

    CLASSGRUPOSID() recebe as idades de um grupo de pessoas com
    tamanho predefinido (tam_grupo) e as classifica em cinco
    grupos distintos utilizando a estrutura switch.
    %}

    tam_grupo = 8;
    idade_grupo = zeros(1, tam_grupo);
    for elem = 1:tam_grupo
        idade_grupo(elem) = input('Digite sua idade: ');
        switch (idade_grupo(elem))
            case num2cell([18:21])
                disp('Usuário no Grupo 1');
            case num2cell([22:30])
                disp('Usuário no Grupo 2');
            case num2cell([31:40])
                disp('Usuário no Grupo 3');
            case num2cell([41:50])
                disp('Usuário no Grupo 4');
            case num2cell([51:60])
                disp('Usuário no Grupo 5');
            otherwise
                disp('Usuário nao cumpre requerimentos.
                    Desculpe!');
        end
    end
end
```

GNU Octave

A estrutura enquanto

- Outra estrutura de repetição é enquanto, representada pela instrução while :

```
> while condicao
>   bloco_de_comandos;
> end
```

A estrutura enquanto

- **Exemplo 01:** Uma série geométrica é uma soma cujos termos sucessivos possuem razão constante entre si.

```
function soma = seriegeomwhile(termos)
    %{
    soma = seriegeomwhile(termos)

    SERIEGEOMWHILE calcula a soma da série geométrica
    1/2 + 1/4 + 1/8 + 1/16 + ... utilizando do
    e recebe como argumento a quantidade de termos
    a serem somados, retornando o valor da soma.
    %}

    soma = 0;
    cont = 1;
    while (cont != termos)
        soma = soma + (1/2)^cont;
        cont++;
    end
end
```

GNU Octave

Atividade

1. Utilize o while para criar a função **classgruposid2()** e melhorar o sistema de classificação, tomando as idades de usuários sem estipular um tamanho fixo para o grupo.

Atividade

```
function idade_grupo = classgruposid2()
    idade_grupo = -1;
    while (idade_grupo(end) != 0)
        idade_grupo(end+1) = input('Digite sua idade: ');
        switch (idade_grupo(end))
            case num2cell([18:21])
                disp('Usuário no Grupo 1');
            case num2cell([22:30])
                disp('Usuário no Grupo 2');
            case num2cell([31:40])
                disp('Usuário no Grupo 3');
            case num2cell([41:50])
                disp('Usuário no Grupo 4');
            case num2cell([51:60])
                disp('Usuário no Grupo 5');
            otherwise
                disp('Usuário nao cumpre requerimentos.
                        Desculpe!');
        end
    end
    disp('Fim do processamento!')
end
```

GNU Octave

PRODUZINDO GRÁFICOS NO OCTAVE

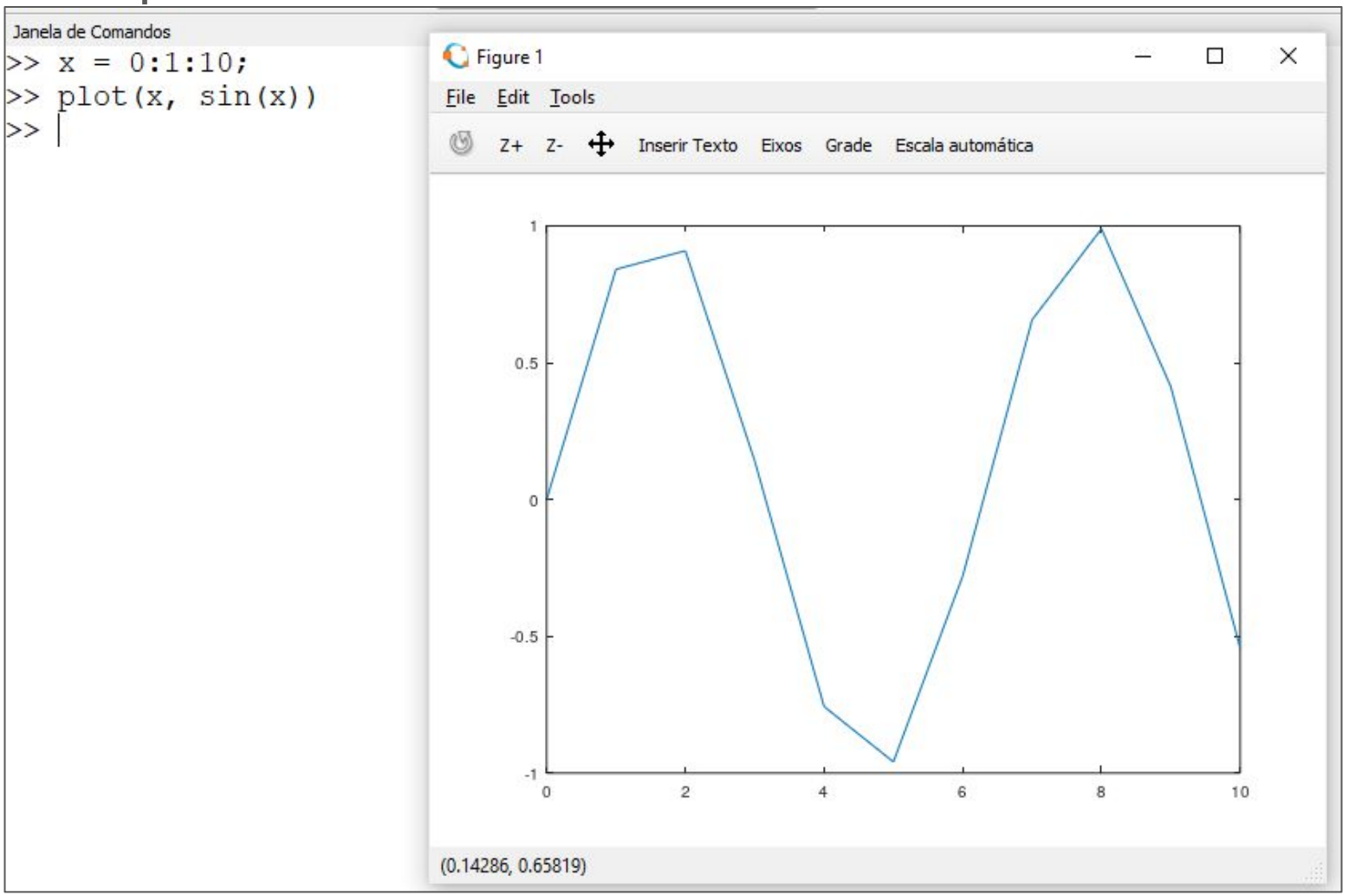
UNIR

- A função elementar para gráficos bidimensionais, **plot()** , é versátil e adaptável para diferentes situações.
- Sua forma mais simples é **plot(f)** , em que f é a função desejada.
- Pensando no plano cartesiano, o eixo X mostra o intervalo que começa em zero e termina com o número de elementos de f , ou seja, **numel(f)**.
- O eixo Y representa os valores da função f correspondentes a cada valor presente no eixo X.

GNU Octave

PRODUZINDO GRÁFICOS NO OCTAVE

- Exemplo 01:

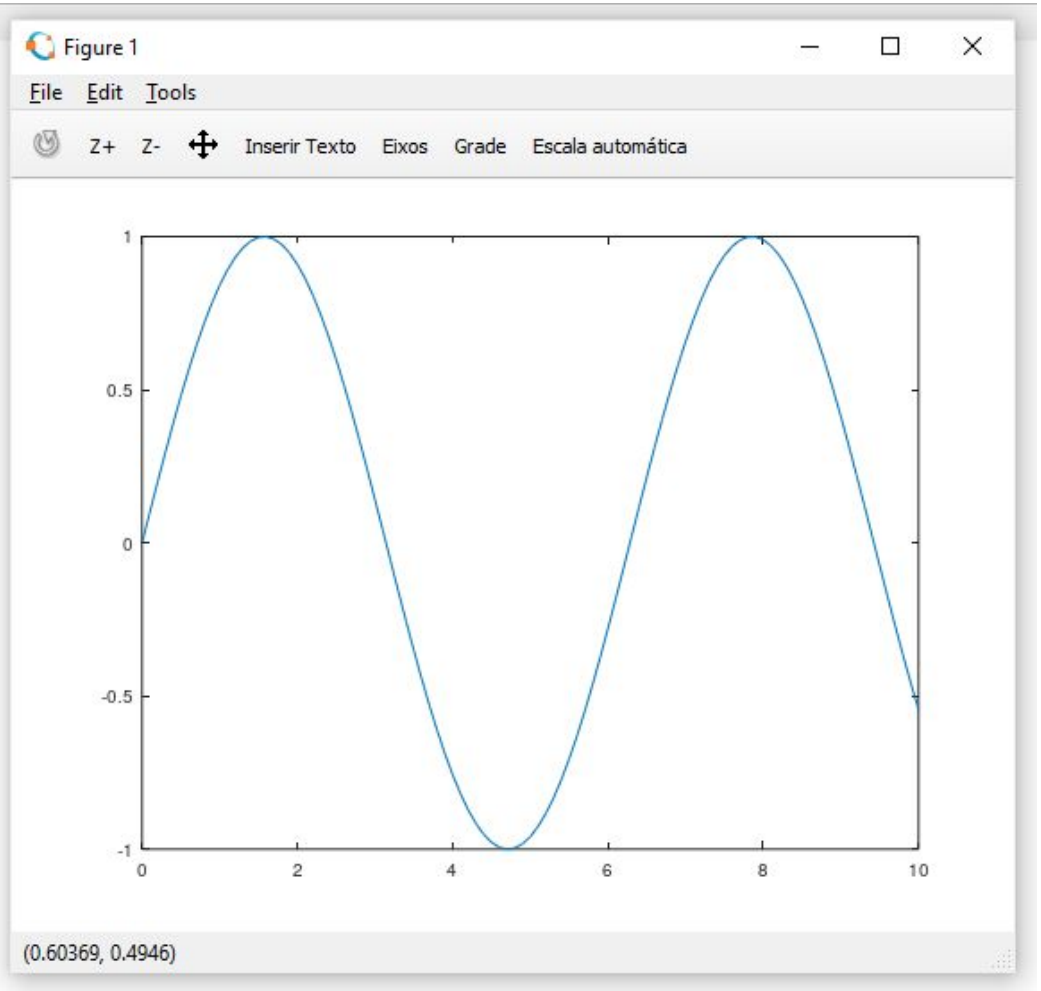


PRODUZINDO GRÁFICOS NO OCTAVE

- Exemplo 02:

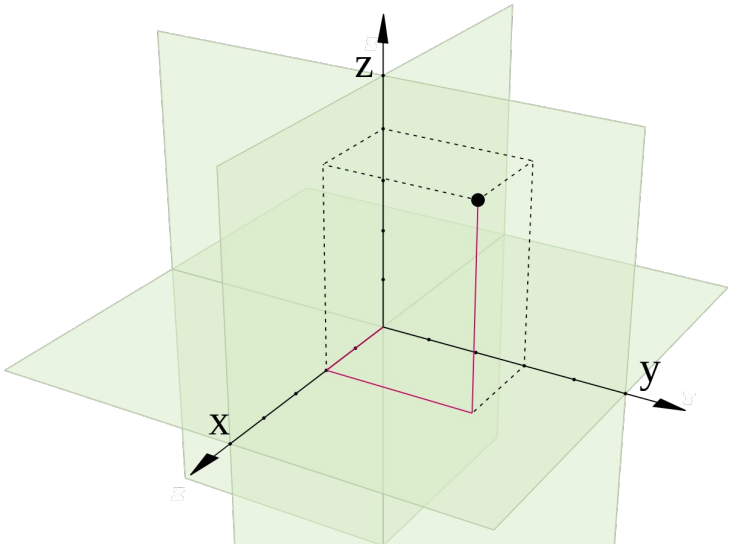
Janela de Comandos

```
>> x = 0:0.1:10;
>> plot(x, sin(x))
>>
```



PRODUZINDO GRÁFICOS NO OCTAVE

- Por sua vez, **plot3()** é a função elementar para gráficos tridimensionais.
- Sua forma mais simples é **plot3(x,y,z)**.
- As variáveis x, y e z representam os eixos X (comprimento), Y (largura) e Z (altura), respectivamente.

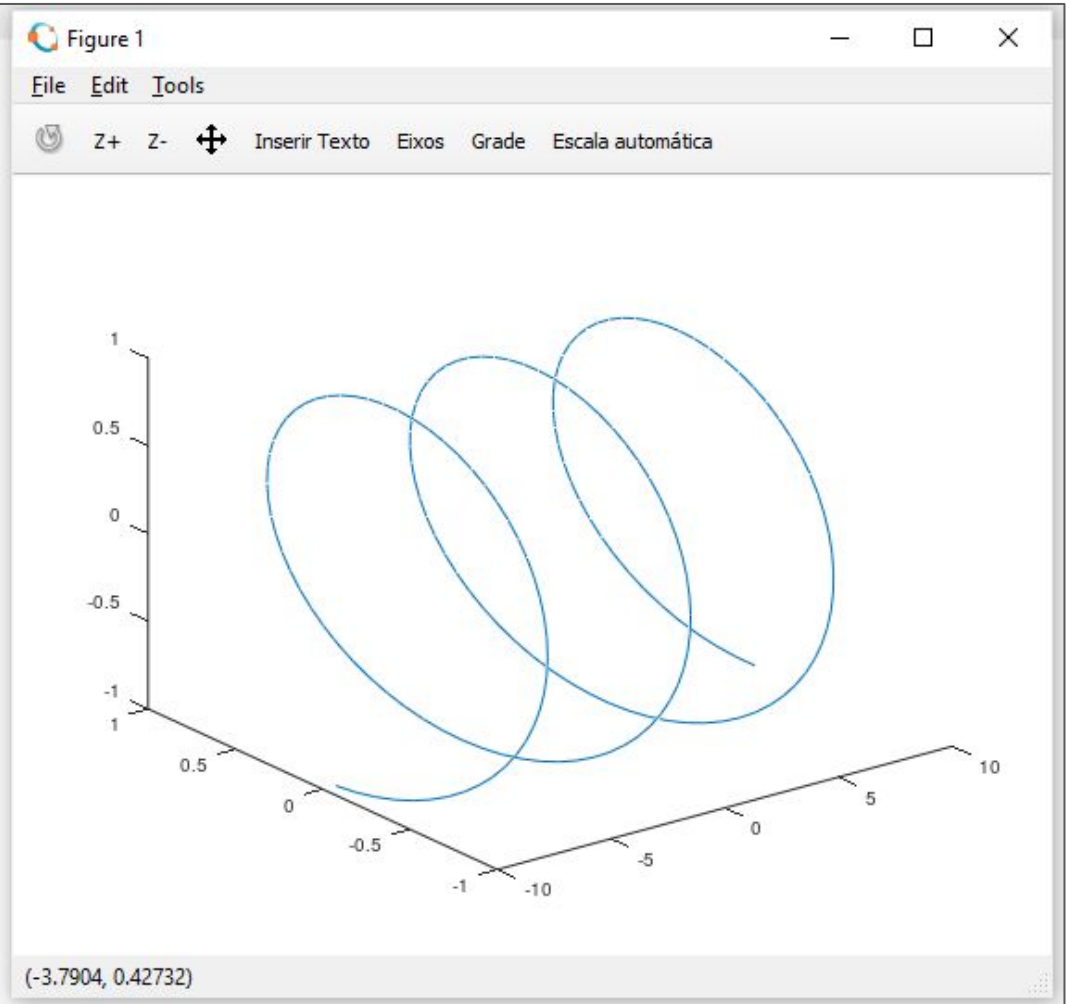


PRODUZINDO GRÁFICOS NO OCTAVE

- Exemplo:

Janela de Comandos

```
>> t= -3*pi:0.1:3*pi;
>> sint = sin(t);
>> cost = cos(t);
>> plot3(t,sint,cost);
>>
```



Modificando a aparência dos gráficos

- **Mira** (`'+'`);
- **Círculo** (`'o'`);
- **Estrela** (`'*'`);
- **Ponto** (`'.'`);
- **Cruz** (`'x'`);
- **Quadrado** (`'s'`);
- **Diamante** (`'d'`);
- **Triângulos voltados para cima, baixo, direita** e **esquerda** (respectivamente, `'^'`, `'v'`, `'>'` e `'<'`);
- **Pentagramas** (`'p'`);
- **Hexagramas** (`'h'`).

```
> x = linspace(-2*pi,2*pi,200);
> plot(x,cos(x),'o');
```

GNU Octave

- **azul** (é o padrão, e seu argumento é `'b'`);
- **preto** (`'k'`);
- **vermelho** (`'r'`);
- **verde** (`'g'`);
- **amarelo** (`'y'`);
- **magenta** (`'m'`);
- **ciano** (`'c'`);
- **branco** (`'w'`).

```
> x = linspace(0,2*pi,100);
> plot(x,exp(x),'r');
```

Plotando vários gráficos em uma janela

UNIR

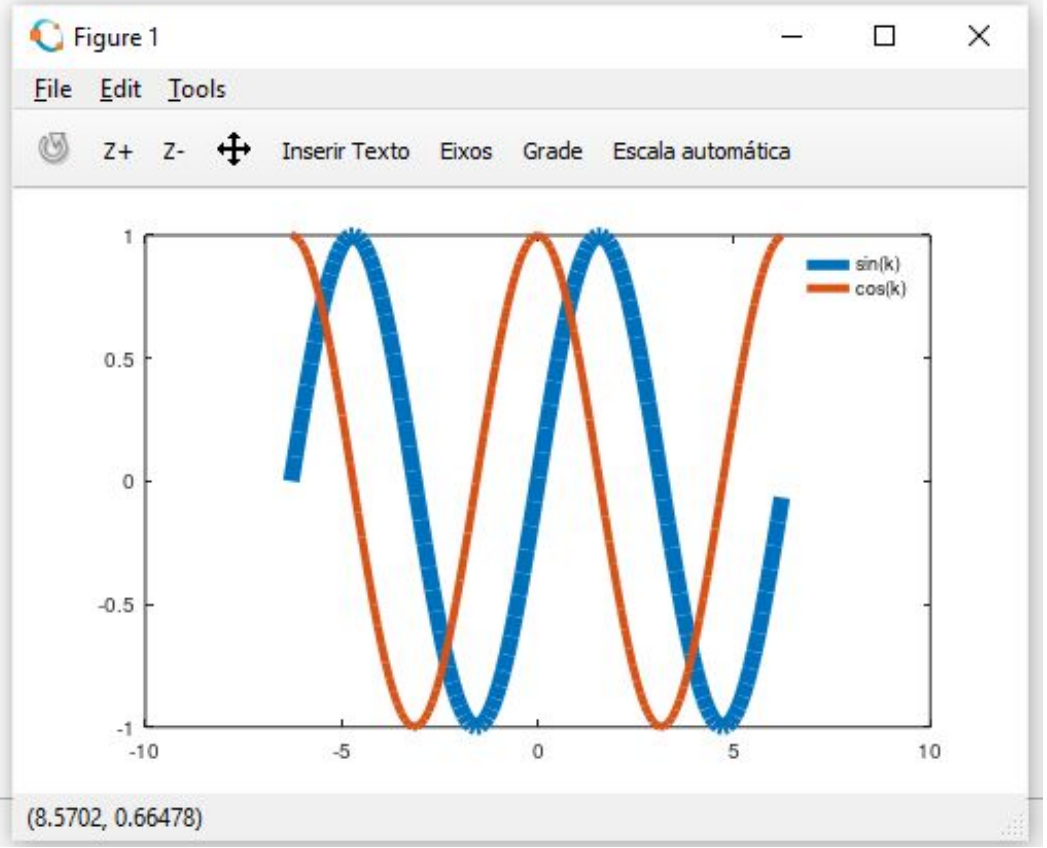
- Para plotar mais de um gráfico utilizando apenas **plot()** ou **plot3()** , devemos informar as funções na seguinte ordem:
  - **Primeiro gráfico:** variável do eixo X, variável do eixo Y, opções (cor, tamanho, símbolo etc.);
  - **Segundo gráfico:** variável do eixo X, variável do eixo Y, opções;
  - **Terceiro gráfico:** variável do eixo X etc.
- Assim por diante, para o **quarto gráfico** e os seguintes.

GNU Octave

Plotando vários gráficos em uma janela

grafico.m

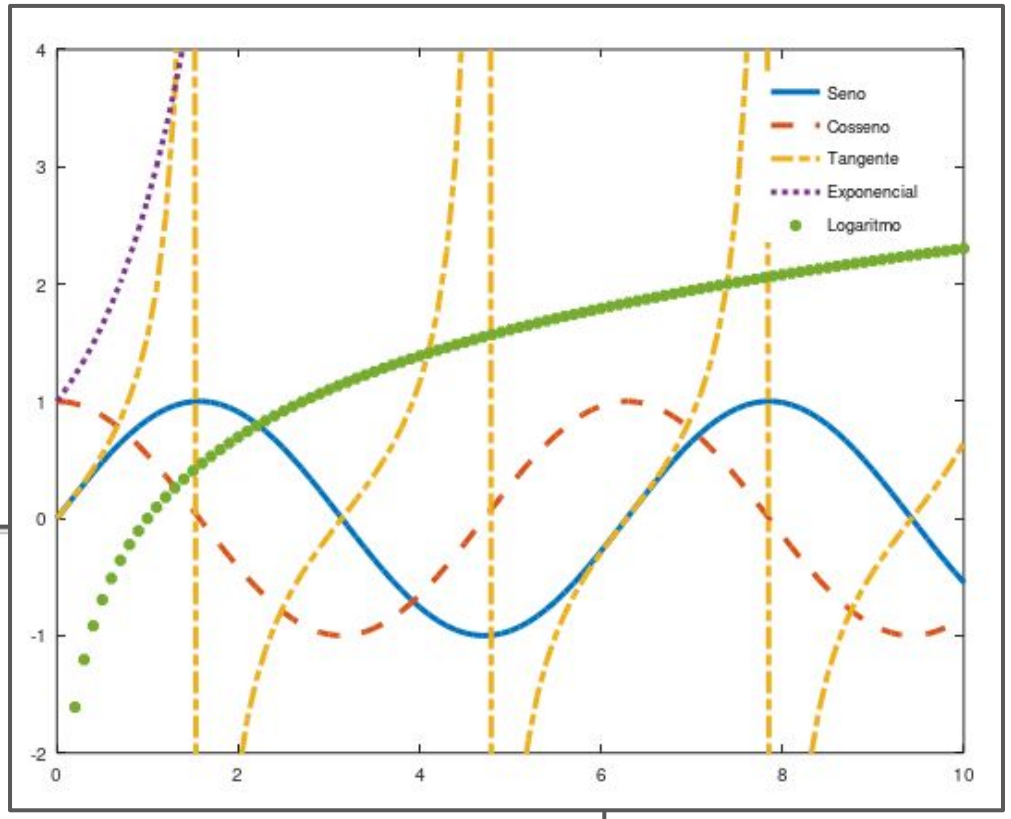
```
1  k = -2*pi:0.1:2*pi;
2  plot(k,sin(k),'linewidth',8,k,cos(k),'linewidth',4)
3  legend('sin(k)','cos(k)')
4  legend('boxoff')
```



linha: 4 col: 1 codificação: SYSTEM fdl: CRLF

Mantendo os gráficos na mesma janela gráfica

GNU Octave




```
1  x = 0:0.1:4*pi;
2  plot(x,sin(x),'linewidth',3)
3  hold all
4  plot(x,cos(x),'--','linewidth',3)
5  plot(x,tan(x),'-.','linewidth',3)
6  plot(x,exp(x),':','linewidth',3)
7  plot(x,log(x),'*','linewidth',3)
8  axis([0,10,-2,4])
9  legend('Seno','Cosseno','Tangente','Exponencial','Logaritmo')
10 legend('boxoff')
11 hold off
```

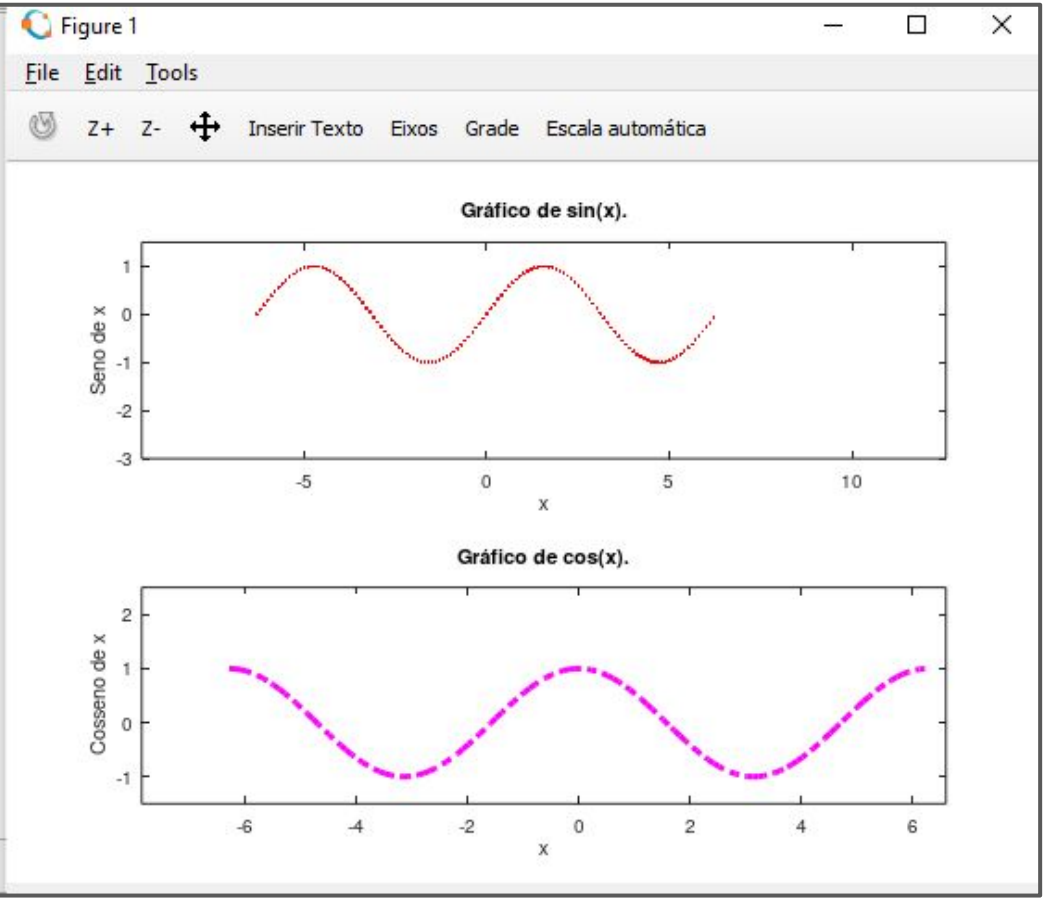
Plotando gráficos em áreas diferentes

UNIR

GNU Octave

```
1  x = -2*pi:0.1:2*pi;
2  subplot(2,1,1)
3  plot(x,sin(x),'.r','linewidth',3)
4  axis([-3*pi,4*pi,-3,1.5])
5  title('Gráfico de sin(x).')
6  xlabel('x')
7  ylabel('Seno de x')
8  subplot(2,1,2)
9  plot(x,cos(x),'-.m','linewidth',3)
10 axis([-2.5*pi,2.1*pi,-1.5,2.5])
11 title('Gráfico de cos(x).')
12 xlabel('x')
13 ylabel('Cosseno de x')
```

inha: 13 col: 1 codificação: SYSTEM fdl: CRLF



Chamando uma função no código de outra função

```
function grafico_quad()
  %{
  grafico_quad()
  GRAFICO_QUAD plota o gráfico da função do segundo grau
  Y = A*X^2+B*X+C, com A (coef_a), B (coef_b) e C (coef_c)
  fornecidos pelo usuário, além das raízes X1 (x0_1) e X2
  (x0_2), obtidas com o auxílio das funções DISCR e RAIZES.
  %}
  disp('*** Gráfico de Y = A*X^2+B*X+C, com X = [-50, 50]. ***');
  coef_a = input('Digite o valor do coeficiente A: ');
  coef_b = input('Digite o valor do coeficiente B: ');
  coef_c = input('Digite o valor do coeficiente C: ');
  % Chamando a função discr() com os argumentos obtidos.
  val_delta = discr(coef_a, coef_b, coef_c);
  % Obtendo X1 e X2 por meio da função raizes().
  [raiz_1, raiz_2] = raizes(val_delta, coef_a, coef_b);
  % Calculando os pontos usando a função quadratica().
  [eixo_x, eixo_y] = quadratica(coef_a, coef_b, coef_c);
  % Plotando os gráficos com a função parabola().
  parabola(eixo_x, eixo_y, raiz_1, raiz_2);
endfunction
```

Essa função esboça o gráfico de uma função do segundo grau, marcando também suas raízes como pontos vermelhos no gráfico.

Chamando uma função no código de outra função

```
1  function delta = discr(vala, valb, valc)
2    %{
3    delta = discr(vala, valb, valc)
4    DISCR calcula o discriminante (delta) da equação
5    quadrática A*X^2 + B*X + C = 0, a partir dos valores
6    de A (vala), B (valb) e C (valc).
7    %}
8    delta = valb^2 - 4*vala*valc;
9  endfunction
```

A função discr() calcula o discriminante da equação quadrática definida pelos argumentos A (vala), B (valb) e C (valc), retornando a variável delta.

Chamando uma função no código de outra função

```
1  function [raiz1, raiz2] = raizes(delta, vala, valb)
2    %{
3    [raiz1, raiz2] = raizes(delta, vala, valb)
4    RAIZES encontra as raízes X1 (raiz1) e X2 (raiz2) da
5    equação quadrática A*X^2 + B*X + C = 0 pela fórmula de
6    Bhaskara utilizando o discriminante (delta) e os
7    coeficientes A (vala) e B (valb).
8    %}
9    raiz1 = (-valb + sqrt(delta))/(2*vala);
10   raiz2 = (-valb - sqrt(delta))/(2*vala);
11 endfunction
```

A função raizes() calcula as raízes da equação quadrática por meio da fórmula de Bhaskara. Seus argumentos são o discriminante delta e os coeficientes vala e valb . As variáveis resultantes são as raízes X1 (raiz1) e X2 (raiz2).

Chamando uma função no código de outra função

```
1  function [eixox, eixoy] = quadratica(vala, valb, valc)
2    %{
3    [eixox, eixoy] = quadratica(vala, valb, valc)
4    QUADRATICA calcula os valores da função quadrática
5    Y = A*X^2 + B*X + C para todos os valores de X (eixox),
6    definido entre -50 e 50.
7    %}
8    % O valor da equação quadrática depende dos valores de x.
9    eixox = -50:0.01:50;
10   % Como eixox é um vetor, usamos .* e .^ para multiplicação
11   % e exponencial por todos os elementos
12   eixoy = vala.*eixox.^2 + valb.*eixox + valc;
13 endfunction
```

A função quadratica(), por sua vez, define X entre -50 e 50 (eixox), e calcula os valores de Y por meio da função quadrática (eixoy). Os argumentos de entrada são os coeficientes vala, valb e valc.

Chamando uma função no código de outra função

```
1  function parabola(eixox, eixoy, raiz1, raiz2)
2    %{
3    parabola(eixox, eixoy, raiz1, raiz2)
4    PARABOLA é responsável por plotar o gráfico da função
5    quadrática e também as raízes X1 (raiz1) e X2 (raiz2),
6    obtidas por meio da função RAIZES.
7    %}
8    disp('As raízes são: ');
9    disp(raiz1);
10   disp(raiz2);
11   % Gráfico da função e das raízes.
12   plot(eixox, eixoy, 'linewidth', 2)
13   hold on;
14   plot(raiz1, 0, 'r*', 'linewidth', 4)
15   plot(raiz2, 0, 'r*', 'linewidth', 4)
16   hold off;
17 endfunction
```

A função parabola() é responsável por plotar os gráficos da equação quadrática e de suas raízes. Seus argumentos são os valores dos eixos X e Y (eixox e eixoy) e as raízes da equação (raiz1 e raiz2).

Atividade

1. Crie uma função de nome area_circunferencia() que receba o valor de raio como argumento, e retorne a variável area .
2. Crie uma função de nome volume_esfera() que receba o valor de raio como argumento e retorne a variável volume .
3. Crie uma função que, a partir do valor de um raio dado pelo usuário, calcule a área da circunferência e o volume da esfera definidos por esse raio. Para isso, essa função deve utilizar as funções area_circunferencia() e volume_esfera() criadas anteriormente.
4. Modifique a função quadratica() para que o valor dos pontos inicial e final de eixox seja recebido como argumento da função.
5. Modifique a função grafico_quad() de forma que ela utilize a função quad(), em vez de discr() e raizes(), no cálculo das raízes da equação quadrática.

Dúvidas, sugestões?