

Introdução à Programação com Matlab/Octave  
Recuperação Módulo 1  
Coltec - UFMG

Márcio Fantini Miranda

<b>1</b>	<b>Fundamentos da Matemática</b>	<b>3</b>
1.1	Conceitos Básicos . . . . .	3
1.1.1	Funções e Gráficos . . . . .	3
1.1.2	Revisão . . . . .	7
1.1.3	Operações Básicas . . . . .	8
1.1.4	Polinômios . . . . .	10
1.1.5	Operações Importantes com Polinômios . . . . .	14
1.1.6	Equações com Polinômios . . . . .	17
1.1.7	Numeração . . . . .	17
1.1.8	Matrizes . . . . .	17
1.1.9	Trigonometria e Geometria Plana . . . . .	18
1.1.10	Função Exponencial e Logaritmo Natural . . . . .	19
1.1.11	Somatórios . . . . .	19
1.2	Ajuste de Curvas . . . . .	21
1.2.1	Fundamentos . . . . .	21
1.2.2	Ajuste de Curva no Octave/Matlab . . . . .	22
1.2.3	polyfit() . . . . .	22
<b>2</b>	<b>Fundamentos da Programação</b>	<b>23</b>
2.1	Introdução . . . . .	23
2.1.1	Saída . . . . .	24
2.1.2	Entrada . . . . .	24
2.1.3	Decisões . . . . .	25
2.1.4	Repetições . . . . .	27
2.1.5	Resumo das Estruturas de Repetição . . . . .	27

---

2.2 Exercícios Básicos . . . . .	29
----------------------------------	----

<b>3 Lista Exercícios</b>	<b>30</b>
---------------------------	-----------

## Sobre Esse Texto e Lista de Exercícios

Esse texto contém uma revisão dos conceitos principais e uma Lista (capítulo 3) que é a avaliação da recuperação do Módulo 1.

Nos dois capítulos que se seguem, capítulos 1 e 2, são apresentadas revisões dos conceitos vistos no módulo 1. No capítulo 3 é apresentada a Lista de Exercícios, que deve ser entregue no local indicado no Classroom.

No capítulo 1 são apresentados os conteúdos básicos de Matemática necessários para nosso estudo de programação. Além de serem fundamentais para o trabalho com Octave, eles são necessários para o entendimento de vários conceitos de Eletrônica, Automação, Computação e programação.

Conceitos como geometria, gráficos e trigonometria estão associados à localização de pontos e figuras 2D ou 3D e também aos fundamentos da computação gráfica. Polinômios e Equações estão relacionados com aplicações em Física e Sistemas Dinâmicos. No Octave, polinômios estão diretamente relacionados com a forma de representação de vetores: ambos são armazenados em variáveis compostas, com seus valores entre colchetes. No trabalho com o Octave é essencial que se saiba trabalhar com polinômios, vetores e matrizes. Matrizes são a base para quase todo tipo de aplicação no Octave.

Nos capítulos 1 e 2 os "Exercícios Básicos" são opcionais. Eles são parte do estudo e dão base para a execução da Lista de Exercícios.

# 1

## Fundamentos da Matemática

### 1.1 Conceitos Básicos

Para termos sucesso na programação de computadores, na Eletrônica e na Automação precisamos ser bons em Matemática. Precisamos assimilar conceitos básicos que trazem consigo raciocínios abstratos e compreensões importantes sobre as abstrações matemáticas. O filósofo grego Platão diferenciava a matemática utilitária, importante para comerciantes e artesãos, da matemática abstrata, destinada a elite. E no nosso caso, temos que dominar alguns conceitos (obviamente relacionados com o nível médio de Ensino) dessa matemática abstrata. Nessa seção revemos alguns desses conceitos.

Nas subseções que se seguem são revistos conceitos importantes que nos ajudam na construção de scripts. Esses conceitos são apresentados no contexto do Octave.

#### 1.1.1 Funções e Gráficos

É de suma importância dominarmos as ferramentas que tratam funções e gráficos no octave. Lembre-se que para gerar um gráfico de uma função  $y = f(x)$  devemos criar um vetor para a **variável independente**  $x$  e calcular  $y$  para cada  $x$ . Por exemplo, podemos criar um vetor  $x$  que varia de -3 até 3, em intervalos regulares de 0.5 fazemos

```
x = -3:0.5:3;
```

Ou ainda, usando **linspace()**:

$x$	-3	-2.5	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2	2.5	3
$y$	9	6.25	4	2.25	1	0.25	0	0.25	1	2.25	4	6.25	9

Tabela 1.1: Exemplo de uma tabela da relação entre  $x$  (variável independente) e  $y = f(x) = x^2$ 

`N = 13`

`x = linspace(-5,5,N)`

No caso acima definimos o tamanho do vetor como sendo  $N = 21$ . Obviamente  $N$  pode ser qualquer tamanho. Uma vez criado um vetor, podemos associar a ele qualquer relação para criarmos uma segunda variável, **dependente** desse vetor. Esse é o conceito de função. Ou seja, quando dizemos que:

$$y = f(x)$$

estamos dizendo que temos uma nova variável,  $y$ , que se relaciona com  $x$  a partir de uma regra (dada por  $f(x)$ ). Se por exemplo tivermos  $f(x) = x^2$ , e desejarmos chamar  $f(x)$  de  $y$ , no Octave fazemos:

`y = x.^2`

Repare que temos que usar o operador `.` pois a operação de elevar ao quadrado tem que ser *ponto a ponto*. Ou seja, elemento a elemento do vetor  $x$ .

Uma vez **criado** (ou definido)  $x$  e **calculado** (ou obtido)  $y$ , temos um par de variáveis, que relacionam seus pontos. Em Matemática é como se tivéssemos uma tabela, como a dada na tabela 1.1.

Para traçarmos o gráfico dessa relação usamos a função `plot()`. Vide figura 1.1.

No Octave se usarmos simplesmente `plot(x,y)` ele irá unir os pontos, como mostrado na figura 1.1-(a). Na figura 1.1-(b) é mostrado o mesmo gráfico gerado com o comando:

`plot(x,y,'o')`

Nesse caso o comando diz algo do tipo "plotar o par  $x$ - $y$ , ponto a ponto (ou melhor dizendo, par a par), sem unir os pontos, usando o marcador círculo. Para ver os diferentes tipos de marcadores use `help plot`.

As variações para traçarmos gráficos são muitas. Podemos combinar pontos separados com linhas unindo pontos. Podemos mudar tamanho de linhas, de marcadores. Podemos escolher cores, etc. A única forma de sabermos sobre as possibilidades é fazendo os exercícios e estudando os exemplos. Na figura 1.2 temos os mesmos dados  $x$ – $y$  da tabela 1.1 gerando gráficos com algumas variações. Na figura 1.2-(a) usa-se um **marcardor círculo de tamanho 10**. O comando para isso é:

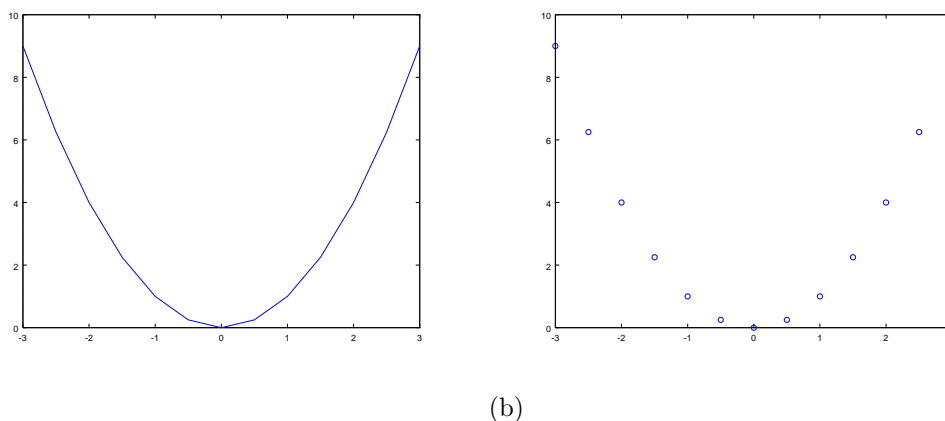


Figura 1.1: Dois gráficos para a relação  $y = f(x) = x^2$ , com  $x$  sendo um vetor de 13 pontos como dado na tabela 1.1. Na figura (a) usou-se o *default* da função `plot`, isto é, o padrão que a função adota, que é o de unir os pontos. Na figura (b) o plot foi feito com o comando `plot(x,y,'o')`, isto é, com o marcador círculo. Quando usamos marcadores, o gráfico é criado sem unir os pontos.

```
plot(x,y,'o','markersize',10)
```

Na figura 1.2-(b) apenas alteramos a espessura da linha que é usada para fazer o círculo. Para isso acrescentamos ao comando acima a propriedade *linewidth* com o tamanho 2 (No exercício você será convidado a experimentar outros tamanhos):

```
'linewidth',2
```

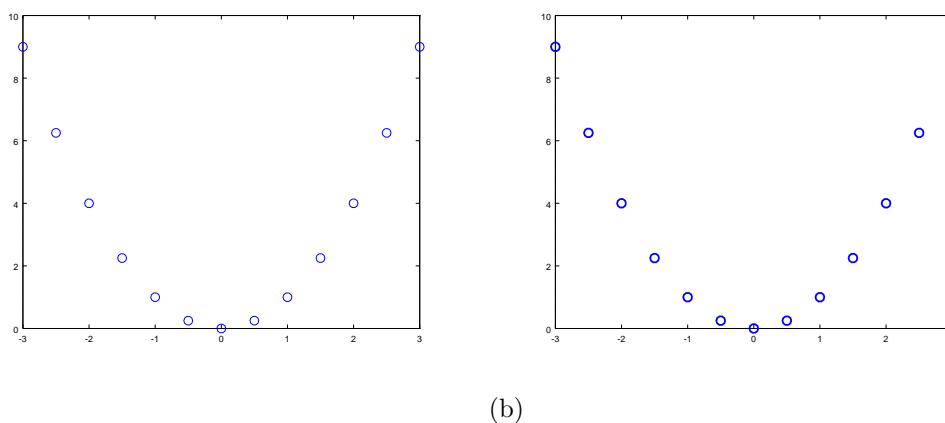
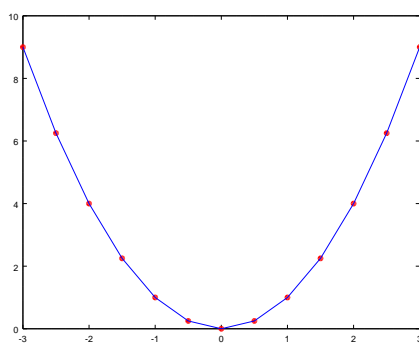
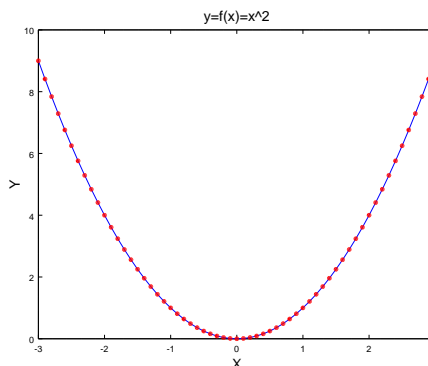


Figura 1.2: Gráficos dos mesmos vetores x-y usados na figura 1.1, criados a partir da tabela 1.1. Em (a) temos o marcador círculo de tamanho 10 e em (b) o mesmo marcador, também de tamanho 10 mas com a espessura da linha de tamanho 2.

Podemos também plotar as duas condições juntas: um gráfico só com os pontos e um gráfico com a linha. Veja a figura 1.3-a. Para isso plotamos o primeiro, usamos o comando **hold on** e depois plotamos o segundo. Se aumentarmos o número de pontos, podemos obter gráficos mais precisos. Repare na figura 1.3-b. Quando geramos um vetor, por exemplo fazendo



(a)



(b)

Figura 1.3: Na figura (a) temos os mesmos dados xy usados nas figuras 1.1 e 1.2. A única diferença são as cores do marcador, o tipo de marcador, que agora é um marcador do tipo ponto (.) de tamanho 8 no lugar do círculo de tamanho 10 e o fato de termos a linha e os pontos. Na figura (b) temos um vetor x de 61 pontos, que vai de -3 a 3, variando de 0.1 a 0.1.

$x = x_0 : \Delta : x_f$ , que é programado como:

$$x = x0 : delta : xf$$

e quisermos saber seu tamanho (quantos elementos ele tem), fazemos:

```
size(x)
```

### 1.1.2 Revisão

Os exemplos acima mostraram apenas uma parte do trabalho com funções e gráficos no Octave. As combinações e possibilidades são inúmeras. O que apresentamos acima foi mais uma motivação para você fazer a revisão e estudar os capítulos 1 a 4 da nossa apostila.

Além da função *plot()* usamos muito frequentemente a função *line()* para traçar uma linha entre dois pontos.

#### Exercícios Básicos

1. Faça um script para traçar os gráficos das figuras 1.1, 1.2 e 1.3.
2. Altere o tamanho do marcador, as cores, a espessura da linhas. Experimente diferentes casos para treinar usar os comandos. Coloque title e label nos eixos. Coloque grid.
3. Crie gráficos semelhantes aos dados figuras 1.1, 1.2 e 1.3 para as funções (repare que você vai ter que definir a faixa para a variável independente):

(a)

$$f(x) = x^3 \quad (1.1)$$

(b)

$$f(x) = \frac{1}{x} \quad (1.2)$$

(c)

$$f(t) = 2t^2 - t \quad (1.3)$$

(d)

$$y = \sin^2(\theta) \quad (1.4)$$

- (e) Faça os gráficos pedidos abaixo. Quando necessário, escolha a faixa de  $x$ . Use o comando *axis* para definir a faixa de valores nos eixos  $x$  e  $y$ .

- |   |   |
|---|---|
| i. Plotar o ponto (5,-5)                                    | v. Plotar a curva $ax + by + c$ para $a, b$ e   |
| ii. Plotar os pontos (-5,5), (5,-5), (5,5), (-5,-5).        | $c$ escolhidos por você (escolha a vontade! experimente diferentes valores).                    |
| iii. Plotar a reta $y = ax + b$ , com $a = -3$ ; $b = 4$ ;  | vi. Plotar a função $f(x) = \sin(2\pi fx)$ , sendo $f$ e $x$ escolhidos por você.               |
| iv. Repetir o item anterior, agora com $a = 3$ ; $b = -4$ ; | vii. Plotar a função $f(x) = \cos(2\pi fx)\sin(2\pi fx)$ , sendo $f$ e $x$ escolhidos por você. |



### 1.1.3 Operações Básicas

No Octave (assim como em qualquer linguagem ou ambiente de programação) é importante sabermos quais os **operadores** usamos para efetuar as operações aritméticas e quais usamos para as operações lógicas.

Esses **operadores** são definidos com símbolos (como os que usamos na Matemática) ou funções. Basicamente temos 3 tipos de operadores: Operadores Aritméticos, Operadores Relacionais e Operadores Lógicos

As tabelas 1.2 a 1.5 apresentam operadores aritméticos e algumas funções que efetuam operações que substituem os operadores em alguns casos.

Tabela 1.2: Operadores Aritméticos e Algumas Funções

operador	o que faz
—	subtração
+	adição
*	multiplicação
/	divisão
$\wedge$	potenciação
função	o que faz
mod(x,y)	resto da divisão inteira entre x e y
sqrt(x)	raiz quadrada de x
exp(x)	$e^x$

Os operadores relacionais são usados para comparar valores. A tabela 1.3 apresenta os símbolos usados para os operadores relacionais enquanto que a tabela ?? apresenta funções do Octave que relacionam valores.

### Operadores Lógicos

A importância dos operadores lógicos é óbvia: eles trabalham com relações lógicas entre variáveis ou entre bits e bytes. A tabela 1.5 apresenta os símbolos usados para as principais operações lógicas.

### Dicas de Estudo

É importante saber usar os operadores e principalmente entender o contexto que eles devem ser usados. Importante também dominar as operações aritméticas. Esses conceitos são

Tabela 1.3: Operadores Relacionais

Operador	Descrição (relação determinada pelo operador)
<	menor que
<=	menor ou igual a
>	maior que
>=	maior ou igual a
==	igual
~=	diferente
!=	diferente

Tabela 1.4: Funções Relacionais

função	operador (o que ela faz)
eq	==
le	$\leq$
ge	$\geq$
lt	<
ne	$\neq$
isequal	retorna verdadeiro se (X1,X2,...) são todas iguais

apresentados no capítulo 2 da nossa apostila.

### Exercícios Básicos

1. Efetue as operações abaixo no papel e depois confira o resultado usando o Octave. Usamos o  $\times$  para indicar multiplicação e usamos ponto no lugar da vírgula para indicar números com casas decimais. Lembre-se que usamos a notação “0x” para indicar o número na base 16. Usamos também a notação **0b** na frente do número para indicar que ele está na base 2.

(a)  $0.1 \times 0.1$

(d)  $0.1^{-3}$

(g) **0b**11011 OR **0b**0011100

(b)  $\frac{0.1}{100}$

(e)  $\frac{1}{7} + \frac{1}{3} + 0.1$

(h) **0b**11111 OR **0x**ABCD

(c)  $\frac{1 \times 10^{-3}}{2 \times 10^2}$

(f) **0x**AF AND **0x**12

(i) not **0b**11001100

Tabela 1.5: Operadores Lógicos

&	and elemento por elemento
&&	and lógico para escalares e expressões lógicas
	or elemento por elemento
	or para esclares
~	not
!	not

### 1.1.4 Polinômios

Em linhas gerais, numa definição nada formal, polinômios são funções específicas que são formados por combinações de uma dada variável elevada a expoentes de diferentes valores, multiplicados por constantes (os denominados coeficientes). Por exemplo, um polinômio de grau 3, na variável  $x$  é dado (genericamente) por:

$$p(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0x^0 \quad (1.5)$$

Sendo as constantes  $a_n$ , com  $n = 3, 2, 1, 0$  os coeficientes do polinômio. Esses coeficientes podem assumir quaisquer valores, inclusive 0. Obviamente se o coeficiente  $a_3$  for nulo o polinômio não mais será de ordem 3. A ordem do polinômio é dada pelo maior expoente cujo coeficiente é não nulo.

No Octave vimos que podemos definir um polinômio armazenando apenas os coeficientes em um vetor. Atenção, muitos confundem quando trabalhamos com equações, expressões e polinômios achando que temos que definir a variável  $x$ . Não é esse o caso, pois com os coeficientes conseguimos armazenar todas as informações que precisamos.

O polinômio genérico de ordem 3 dado na equação (1.5) é armazenado no octave como (obviamente, dados os valores de  $a_3, \dots, a_0$ ):

```
p = [a3,a2,a1,a0]
```

Se por exemplo quisermos definir o polinômio

$$p1(x) = 2x^3 - x^2 + 3 \quad (1.6)$$

teríamos que fazer:

```
p1 = [2 -1 0 3];
```

Ou se você preferir:

```
a3=2;a2=-2;a1=0;a0=3;
p1 = [a3,a2,a1,a0];
```

(você pode também trocar a vírgula por espaço)

Quando definimos um polinômio como no exemplo acima, estamos também definindo um vetor (que nada mais é que uma variável composta, que armazena valores que são indexados). Lembre-se que nesse caso, cada coeficiente armazenado será acessado a partir de seu índice.

```
p1      = [a3  a2  a1  a0]
%posicao(índice) = 1  2  3  4
```

Dessa forma se fizermos  $p1(2)$  obteremos o valor  $a2$ . E assim por diante.

Algumas funções para operações com polinômios são essenciais para nossos cálculos e programas no Octave. A seguir repetimos da apostila, capítulo 2, a parte que explica como trabalhar com polinômios no Octave.

Acompanhe os exemplos a seguir. Veja a explicação do professor e as vídeo-aulas sobre polinômio e vetores no Octave.

- Exemplo: Polinômio de 3 ordem (ou grau 3):

$$p(x) = x^3 - 10x^2 + 4x + 1$$

- Esse polinômio possui os coeficientes:

$$\begin{bmatrix} 1 & -10 & 4 & 1 \end{bmatrix}$$

- O polinômio  $q(y) = -2x^4 + 3x$  é um polinômio de 4 ordem, com os coeficientes:

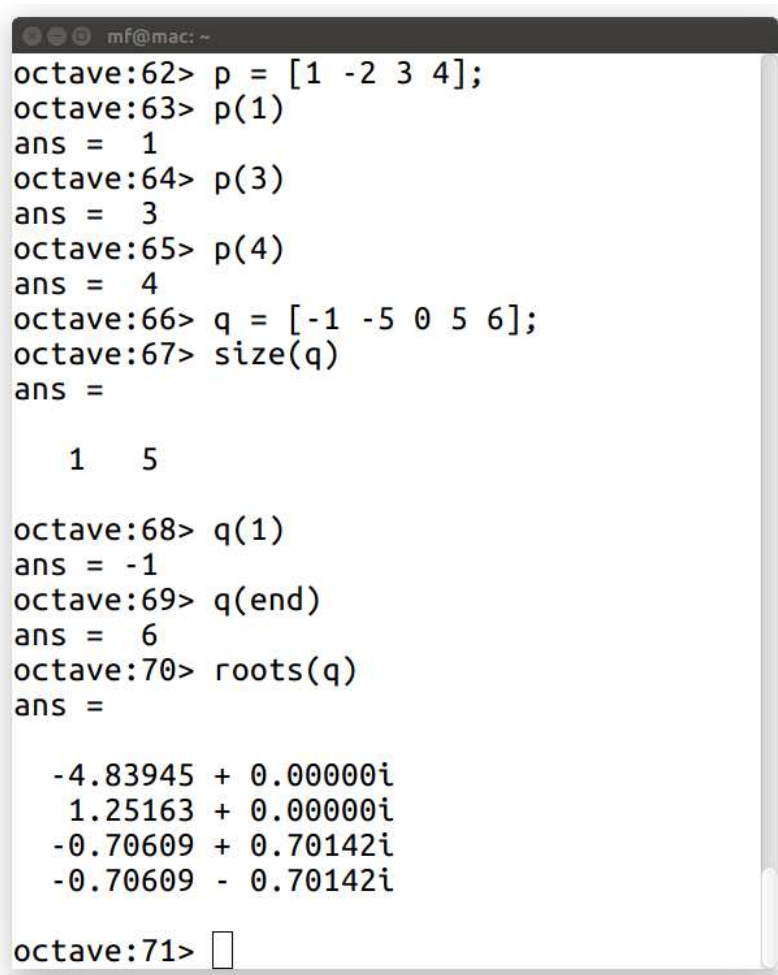
$$\begin{bmatrix} -2 & 0 & 0 & 3 & 0 \end{bmatrix}$$

- Repare a forma que organizamos os coeficientes. A ordem que eles aparecem dentro dos colchetes é EXTREMAMENTE IMPORTANTE

**Conclusão e explicação:** Os coeficientes do polinômio são organizados entre colchetes, começando da esquerda para direita, da maior para a menor ordem.

É importante entender que a variável que é definida para armazenar o polinômio possui a mesma estrutura de um vetor. Na verdade usamos o termo “vetor” para todo tipo de variável que tem mais de um valor. Nem sempre ela (a variável) vai significar um vetor (no plano ou no espaço n-dimensional).

Usamos a notação da variável indexada para acessar os coeficientes. Veja o exemplo na figura 1.4 e acompanhe a explicação dos comandos e resultados.



```
mf@mac: ~  
octave:62> p = [1 -2 3 4];  
octave:63> p(1)  
ans = 1  
octave:64> p(3)  
ans = 3  
octave:65> p(4)  
ans = 4  
octave:66> q = [-1 -5 0 5 6];  
octave:67> size(q)  
ans =  
  
    1    5  
  
octave:68> q(1)  
ans = -1  
octave:69> q(end)  
ans = 6  
octave:70> roots(q)  
ans =  
  
   -4.83945 + 0.00000i  
    1.25163 + 0.00000i  
   -0.70609 + 0.70142i  
   -0.70609 - 0.70142i  
  
octave:71> 
```

Figura 1.4: Tela do Octave com comandos de polinômio

**Explicação da Figura 1.4**

- Na linha 62 definimos o polinômio  $p(x) = x^3 - 2x^2 + 3x + 4$
- Nas linhas 63, 64 e 65, definimos, respectivamente, os valores das posições 1, 3 e 4 dessa variável.  $p(1)$  retorna o valor da posição 1, que é a posição do coeficiente de maior ordem. (4) mostra o valor da última posição.
- Na linha 66 definimos outro polinômio, dado por  $q(x) = -x^4 - 5x^3 + 5x + 6$ .
- Na linha 67 obtemos o tamanho de  $q$ .
- na linha 61 pedimos o valor da posição 1 da variável  $q$ . Esse valor é -1, que é o valor da primeira posição da variável. Aqui chamamos 1 de índice. Ou seja o valor da variável no índice (ou posição) 1 é -1.
- Na linha 69 obtemos o valor da última posição da variável. Usamos o termo *end* para acessar a última posição. Podíamos também usar  $q(5)$ .
- Na linha 70 calculamos as raízes da equação  $q(x) = 0$ , ou  $-x^4 - 5x^3 + 5x + 6 = 0$ . Essa é uma equação de quarto grau. E portanto possui 4 raízes. No caso a resposta foram duas raízes reais e duas raízes complexas. O número  $j$  é o número complexo equivalente a  $\sqrt{-1}$ . Veremos os números complexos no próximo capítulo.

### 1.1.5 Operações Importantes com Polinômios

#### Multiplicação

No Octave fazemos multiplicação de polinômios usando a função **conv()**. O termo *conv* vem de **convolução** que é uma operação entre dois polinômios ou entre dois **sinais**. Veremos mais a frente no curso o significado de um sinal e o que significa fazer a **convolução** entre dois sinais. Matematicamente falando **convolução** é um **operador** que cria uma terceira função, a partir de duas outras, segundo uma regra de superposição de seus valores. Essa operação de convolução se aplica a polinômios, funções e sinais.

No nosso caso, entendemos a convolução como multiplicação de polinômios. Vejamos. Sejam  $p_1(x)$  e  $p_2(x)$  dados por:

$$p_1(x) = 2x - 1; \quad p_2(x) = -x + 3$$

Então,  $p_1 \times p_2$  (ou  $p_1 \cdot p_2$ , ou ainda  $p_1 p_2$ ) será:

$$\begin{aligned} p_1 p_2 &= (2x - 1) \cdot (-x + 3) \\ &= 2x(-x) + 2x(3) + (-1)(-x) + (-1)(3) \\ &= -2x^2 + 6x + x - 3 \\ &= -2x^2 + 7x - 3 \end{aligned}$$

#### Raízes de equações

Se fizermos uma igualdade com polinômios definimos uma equação. Ou seja, para  $p_1(x)$  dado acima, podemos fazer

$$p_1(x) = 0$$

que gera uma equação de primeiro grau:

$$2x - 1 = 0$$

Uma questão de interesse em Matemática, com aplicações em Engenharia, Física e áreas afins é saber quando um polinômio vale zero. Ou seja, quais os valores de  $x$  (no caso de um polinômio na variável  $x$ ) tornam  $p(x) = 0$ . Responder essa pergunta é resolver a equação  $p(x) = 0$ .

Achar raízes de equações de primeiro e segundo grau é fácil. Mas e de um polinômio de grau  $n > 2$ ?

No Octave isso é simples. Basta usar a função **roots()**. Veja o exemplo a seguir:

**Exemplo 1** Encontre as raízes das equações  $p_1(x) = 0$  e  $p_2(x) = 0$ , sendo:

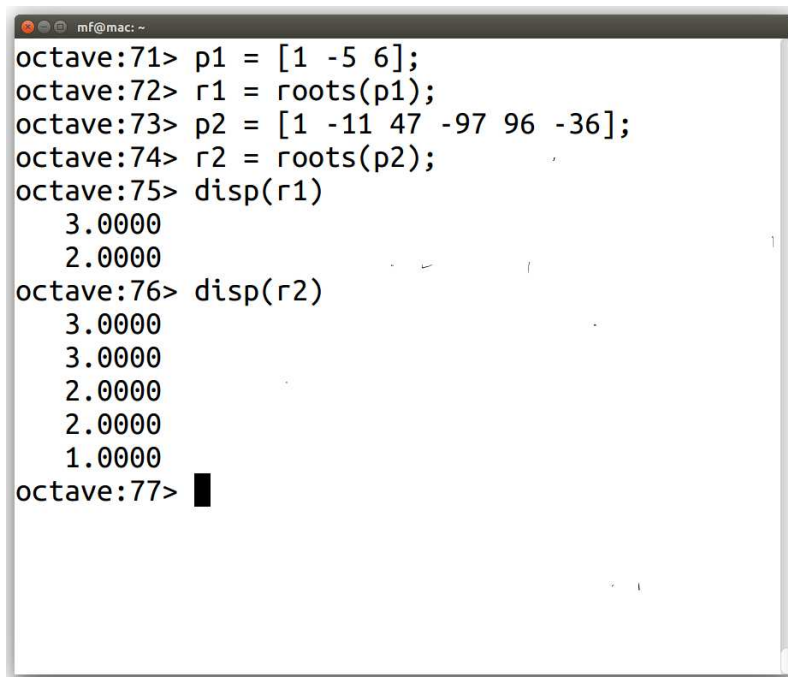
$$p_1(x) = x^2 - 5x + 6 \quad \text{e} \quad p_2(x) = x^5 - 11x^4 + 47x^3 - 97x^2 + 96x - 36$$

Para resolver essas equações no Octave/Matlab basta usar a função `roots()`.

```
p1 = [1 -5 6];  
r1 = roots(p1);  
p2 = [1 -11 47 -97 96 -36];  
r2 = roots(p2);
```

Nesse caso `r1` e `r2` possuem as raízes das equações pedidas. Para visualizá-las use `disp(r1)` e `disp(r2)`. Veja a figura 1.5.

É importante nesse exemplo reparar que para calcular as raízes de uma equação de qualquer grau basta definirmos o polinômio e usarmos a função `roots()` para ele. No caso entende-se que ao usar `roots(p)` para um dado polinômio  $p$  estamos calculando as raízes de  $p = 0$ .



```
octave:71> p1 = [1 -5 6];  
octave:72> r1 = roots(p1);  
octave:73> p2 = [1 -11 47 -97 96 -36];  
octave:74> r2 = roots(p2);  
octave:75> disp(r1)  
    3.0000  
    2.0000  
octave:76> disp(r2)  
    3.0000  
    3.0000  
    2.0000  
    2.0000  
    1.0000  
octave:77> █
```

Figura 1.5: Tela do Octave para Exemplo de cálculo de raízes



### Calculando Polinômios para um Valor Específico

Uma outra aplicação com polinômios é o cálculo do valor do polinômio para um dado valor da variável. Por exemplo, se temos  $p_1(x) = x^2 - 5x + 6$  se desejarmos saber quanto vale  $p_1(1)$  basta substituímos o valor 1 no polinômio. Ou seja,  $p_1(x)$  para  $x = 1$  é dado por  $p_1(1)$  e vale:

$$p_1(1) = 1^2 - 5(1) + 6 = 1 - 5 + 6 = 2;$$

logo  $p_1(x = 1)$ , que escrevemos  $p_1(1)$  vale 2, isto é  $p_1(1) = 2$ .

No Octave podemos usar uma função para fazer esse cálculo (mas podemos fazer usando o cálculo exemplificado acima também. Usar a função simplifica nossos cálculos). A função que calcula o valor do polinômio em um dado valor é a **polyval()**. Podemos entender seu nome com algo próximo de “valor do polinômio”.

Ou seja, para o  $p_1(x)$  dado acima podíamos fazer:

```
p1 = [1 -5 6]
polyval(p1,1)
```

Nesse caso estamos “passando para a função”o polinômio e o valor que queremos calculá-lo.

### Adição de Polinômios

A operação de adição de polinômios é muito simples. Basta usar o operador soma! (+). Veja:

```
p1 = [1 2]; p2 = [2 3];
p3 = p1+p2;
```

Nesse caso  $p3$  valerá:

```
p3 = [3 5];
```

O que equivale, na Matemática:

$$p_3(x) = (x + 2) + (2x + 3) = 3x + 5$$

Simples assim!!

### 1.1.6 Equações com Polinômios

Como visto acima, se quisermos resolver uma equação do tipo

$$p(x) = 0$$

para um polinômio genérico  $p(x)$ , basta usarmos a função **roots()**. É importante você entender quando e como usar a função **roots** e entender o que ela retorna.

### 1.1.7 Numeração

É importante conhecer as representações nas bases 2, 8, 10 e 16. Para isso estude o capítulo ?? dessa apostila.

### 1.1.8 Matrizes

Saber trabalhar com matrizes e vetores está na essência de quase tudo no Octave. A seguir são apresentados alguns conceitos básicos. Você deve estudar com muita atenção o capítulo 6 da nossa apostila.

- Matrizes, numa definição bem simples (ou simplista) são ARRANJOS ORDENADOS de números.
- Os números são organizados dentro de uma estrutura “criada” para que eles (os números) possam ser INDEXADOS.
- Veja o exemplo no slide a seguir.

$$M_1 = \begin{bmatrix} -2 & 3 \\ 3.5 & 20 \end{bmatrix} \quad G = \begin{bmatrix} -2 & 3 & 4 \\ 5 & 6 & -7 \end{bmatrix} \quad (1.7)$$

$$P = \begin{bmatrix} -2 & 3 \\ 3.5 & 20 \\ 0 & 0.1 \\ 10 & 100 \end{bmatrix} \quad Q = \begin{bmatrix} -2 \\ 5 \\ -6 \\ 0.1 \\ -\pi \end{bmatrix} \quad M_2 = \begin{bmatrix} -2 & 5 & -6 & 0.1 \end{bmatrix} \quad (1.8)$$

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (1.9)$$

A matriz  $M_1$

$$M_1 = \begin{bmatrix} -2 & 3 \\ 3.5 & 20 \end{bmatrix} \quad (1.10)$$

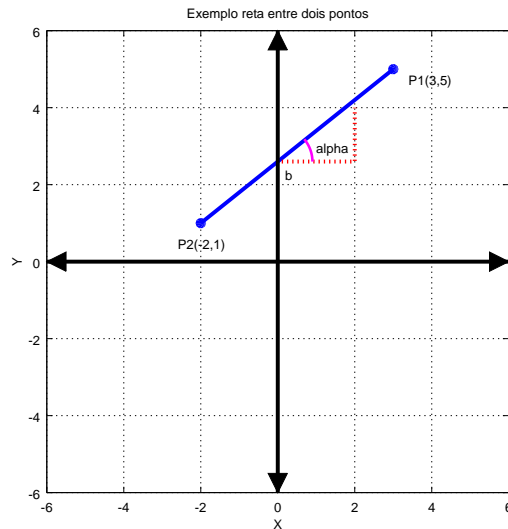


Figura 1.6: Exemplo de um segmento de reta entre dois pontos, destacando-se o ponto  $b$  e o ângulo  $\alpha$ , relacionado com o coeficiente  $a$  da reta.

é escrita no Octave (e também no Matlab) como;

$$M1 = [-2, 3; 3.5, 20]$$

- vírgula separa as colunas e ponto-e-vírgula separa as linhas.

Cada elemento da matriz é acessado por um índice. Veja o exemplo:

$$M1 = [-2, 3; 3.5, 20]$$

$$M1(1, 1) \rightarrow -2; M1(2, 1) \rightarrow 3.5$$

$$M1(i, j) \rightarrow i \text{ acessa a linha e } j \text{ acessa a coluna}$$

### 1.1.9 Trigonometria e Geometria Plana

O capítulo 10 da apostila do curso contém informações fundamentais sobre as regras de geometria plana e geometria analítica, que lida com coordenadas, pontos, retas, distâncias, etc. no Plano Cartesiano, como mostrado na figura 1.6. Nessa figura temos um segmento de reta, destacando-se sua inclinação e os dois pontos.

#### Exercício Básico

1. Faça um script para Gerar o gráfico da figura 1.6.

### 1.1.10 Função Exponencial e Logaritmo Natural

Usamos com frequência o número  $e$ , que é dado no Octave pela função exponencial  $e^x$ , definida com o a função:

`exp(x)`

Usamos também com frequência o Logaritmo Natural ou Logaritmo Neperiano,  $\ln(x)$ , que no octave é dado por:

`log(x)`

Veja nos capítulos 2 e 3 da apostila como trabalhar com esses dois conceitos.

### 1.1.11 Somatórios

Os somatórios são estudados em detalhes no Capítulo 9 da nossa apostila. Estude com muito carinho esse capítulo.

#### O que é um Somatório

Em Matemática usamos o conceito de **Somatório** como sendo uma soma de uma função, definida num intervalo com um valor inicial e um valor final.

Assim, pode-se definir uma soma dos quadrados de números naturais de 1 a 10, por exemplo. Isso poderia ser representado assim:

$$S = 1^2 + 2^2 + 3^2 + \dots + 10^2$$

Na Matemática, usa-se o símbolo  $\Sigma$  (letra grega **sigma** maiúscula) para indicar uma soma desse tipo.

Ou seja, a letra grega  $\Sigma$  indica um SOMATÓRIO.

$$\Sigma = \text{somatório}$$

Assim, a soma dada acima pode ser representada como:

$$S = \sum_{n=1}^{n=10} n^2$$

A equação acima pode ser lida da seguinte forma:

*Faça a soma de  $n$  elevado a 2, com  $n$  variando de 1 até 10. Ou ainda, faça a soma do quadrado dos números 1 a 10.*

Repare que nessa notação, o que vem embaixo da letra  $\Sigma$  é o valor inicial de  $n$  e na parte superior de  $\Sigma$  vem o valor final de  $n$ . E considera-se que  $n$  varia de 1 em 1.

Usualmente, como sabemos que a variável que terá seu valor mudado é a variável que está dentro do somatório ( $\Sigma$ ) usamos uma notação mais simplificada:

$$S = \sum_{n=1}^{10} n^2$$

ou ainda

$$S = \sum_n^{10} n^2$$

Nesse caso já sabemos que quem irá variar de 1 a 10 é a variável  $n$ .

## 1.2 Ajuste de Curvas

### 1.2.1 Fundamentos

A ideia do "ajuste de curvas" é simples: dado um conjunto de dados (geralmente visualizados num gráfico  $x \times y$ ) é possível obter um polinômio de grau  $n$  a partir de operações matemáticas, relacionadas com a minimização de erros.

Ou seja, de posse de dados reais é possível definir algoritmos que encontrem uma representação matemática, a partir de operações que busquem reduzir o erro entre o valor real e o valor aproximado, dado pela representação matemática.

Para ficar mais claro essa idéia, veja a figura 1.7. Nela é mostrado conjunto de dados e uma reta de ajuste. Essa reta foi encontrada a partir de operações matemáticas que tentaram minimizar o erro entre o valor obtido pela reta e o valor real.

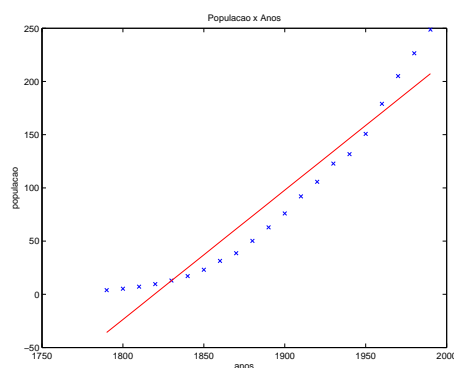


Figura 1.7: Exemplo de ajuste com polinômio de primeiro grau ( $y = ax + b$ )

Obviamente podemos encontrar um polinômio melhor. Possivelmente para esse conjunto de dados, a reta não seja a melhor opção.

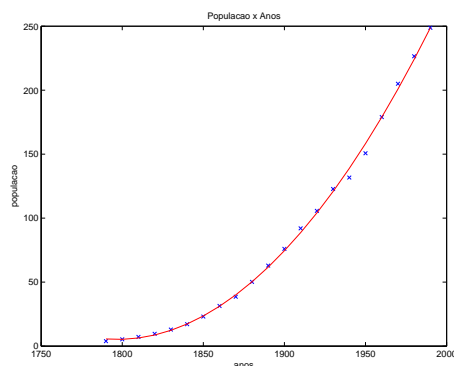


Figura 1.8: Exemplo de ajuste com polinômio de segundo grau ( $y = ax^2 + bx + c$ )

A figura 1.8 mostra o ajuste da curva com um polinômio de segundo grau. Veja que o erro entre o valor da curva encontrada e os dados reais está muito menor.

A ideia de todo ajuste é essa: encontrar uma representação matemática que mais se aproxime do (ou que mais represente o) dado real.

### 1.2.2 Ajuste de Curva no Octave/Matlab

No Octave é muito simples fazer ajuste de curvas. Basta usar o comando **polyfit** (ajuste de polinômios).

A regra geral é:

```
polinomio = polyfit(dadosX,dadosY,ordem);
```

Ou seja, o comando **polyfit** é chamado com os dados do eixo X, depois do eixo Y e depois a ordem que se deseja obter o polinômio.

Para o exemplo dado nas duas figuras acima, temos:

```
p1 = polyfit(x,y,1) %obtem polinomio de 1o grau  
p1 = [1.2157 -2.2120e+03]
```

Ou seja, o Octave dá como resultado a reta  $y = ax + b$ , sendo  $a = 1.2157$  e  $b = 22120$ .

Repare que temos que saber usar polinômios no Octave. Lembr-se que o polinômio é salvo no Octave com os seus coeficientes em ordem decrescente.

No caso do polinômio de segundo grau, para o exemplo acima, o Octave retornou:

$$p(x) = 0.0065x^2 - 23.5097x + 2.1130 \times 10^4$$

### 1.2.3 polyfit()

Existem muitos algoritmos para se encontrar representações matemáticas para “massa de dados”. E existem muitas representações possíveis. Essa nota de aula apresentou uma versão muito simplificada de como fazer esses ajustes. Dependendo do tipo de dado e das relações entre  $x$  e  $y$  usar o comando **polyfit()** simplesmente não é suficiente. Algumas vezes você terá que considerar o erro de estimação. Outras terá que trabalhar os dados antes de aplicar o algoritmo.

O **polyfit** usa o que chamamos de “método dos mínimos quadrados”. Para saber mais, use o *help polyfit*.

# 2

## Fundamentos da Programação

### 2.1 Introdução

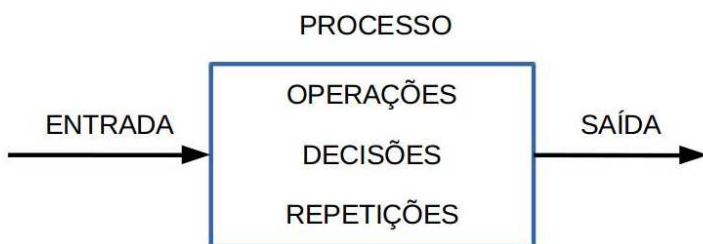


Figura 2.1: Estrutura básica de um programa de computador

Lembre-se sempre de pensar o script dentro da formulação padrão de um programa de computador (figura 2.1). Ao pensar no script, pense em qual estrutura você vai precisar. Quais as variáveis. O que você precisa ler e o que precisa mostrar. Lembre-se sempre do que faz cada parte do programa.

Nas seções seguintes são revistos os principais conceitos representados na figura 2.1.



### 2.1.1 Saída

A **saída** pode ser feita de várias formas. Um comando **plot()** é uma saída. Sairas na tela podem ser feitas com **printf()** ou **disp()**. Podemos também salvar uma figura com o comando **print**. Todas essas funções caracterizam uma saída. Como vimos, as mais usadas são mesmo **printf()** ou **disp()**.

### 2.1.2 Entrada

A **Entrada** é preferencialmente feita via função **input()**. Ou no caso de quem já sabe escrever sua própria função, a entrada de uma função são os parâmetros para ela passados. O Octave aceita também usarmos em um script uma variável que está no **workspace** (ou no ambiente de trabalho, ou ainda na memória do Octave).

- Lembre-se que **input()** pode ser usado para receber um número ou um string (array de caracteres). Ou ainda apenas um caractere. No caso de leitura de um caractere ou um string, use 's' no final.

### 2.1.3 Decisões

As **decisões** alteram o fluxo normal do programa. Lembrem-se da estrutura if-else. Podemos ter três casos (na verdade são variações de uma única estrutura).

- Caso 1:

```
comandos
...
if (condição)
    comandos se condição for verdadeira
end
continua o programa
comandos
....
```

- Caso 2:

```
comandos
comandos
...
if (condição)
    comandos se condição for verdadeira
else
comandos se condicao for falsa
end
continua o programa
comandos
....
```

- Caso 3:

```
comandos
comandos
...
if (condição 1)
    comandos se condição for verdadeira
```

```
elseif(condição 2)
comandos se condição 2 for verdadeira
else
comandos se condicao 1 for falsa
end
continua o programa
comandos...
```

### Comando switch():

1. Ainda dentro da categoria de estrutura de decisões temos o comando **switch()**. Veja no capítulo 7 da nossa apostila.
2. Usamos muito o switch quando fazemos menu. Uma combinação de **while()** com switch é muito útil para definirmos um menu de opções. Estude, entenda e aplique o exemplo dado a seguir:

```
%repeticao para criar menu
opcao = 1;

while(opcao != 0)
    menu;      % chama script que cria menu
    opcao = input('Escolha a opcao do menu: ');
    switch(opcao)
        case 1
            disp('voce digitou 1')
        case 2
            disp('voce digitou 2')
        case 3
            disp('voce digitou 3')
        case 0
            printf("voce digitou 0...vou sair\n");
        otherwise
            disp('entre com valores entre 0 e 3');
    end
end
printf("Sai do while (opcao = %i)\n",opcao);
```

### 2.1.4 Repetições

As **repetições** são usadas para alterar o fluxo natural do script, mantendo-se o programa executando um mesmo conjunto de instruções enquanto uma determinada condição for satisfeita. Como vimos nos capítulos

#### Algumas Aplicações com Repetições

- Somatórios
- Exercícios com vetores e matrizes
- Figuras geométricas com símbolos e com números.
- Loops para execução contínua de ações.
- Ordenação e Busca.

Dicas:

- Sempre é possível usar `for` e `while` indistintamente, mas em algumas situações é melhor usar a estrutura `for` e em outras a estrutura `while`.
- Sempre que `for` repetir um número determinado de vezes, use a estrutura `for`.
- Sempre que `for` usar uma repetição com uma condição use a estrutura `for`.
- Em um somatório com número determinado de pontos é mais adequado se usar o `for`.
- Em um somatório com uma condição de parada é melhor usar o `while`.
- Para acessar índices de matrizes e vetores, geralmente é melhor usar `for`.
- Arrays bi-dimensionais são usadas com dois loops `for`: um para variar as linhas e outro para as colunas

### 2.1.5 Resumo das Estruturas de Repetição

1. Estrutura `for` para preencher um vetor:

```
N = input("qual o tamanho do vetor? ");
for i = 1 : N
    x = input("entre com um numero: ");
    v(i) = x;
```

```
endfor  
disp(v)
```

2. Estrutura for para mostrar elementos do vetor:

```
for i=1:5  
    printf("v[%i] = %i\n",i,v(i));  
end
```

3. Estrutura for para preencher uma matriz aleatoria,  $2 \times 3$

```
for i=1:2  
    for j=1:3  
        M(i,j)=int16(5*rand());  
    end  
end  
disp(M);
```

4. Somando 10 primeiros números naturais usando while

```
n = 1;soma=0;  
while (n <= 10)  
    soma = soma + n;  
    n = n + 1;  
end
```

5. Somando 10 primeiros números naturais usando for

```
soma=0;  
for i=1:10  
    soma = soma + i;  
end
```

---

## 2.2 Exercícios Básicos

1. Faça um programa para ler dados para um vetor de tamanho 4. O programa deve usar um laço for para carregar os dados dentro do vetor.
2. Repite o exercício acima, agora usando um estrutura while.
3. Faça um programa para gerar duas matrizes  $3 \times 3$  aleatórias, com números inteiros entre 1 e  $N$ , com  $N$  dado pelo usuário
4. Faça um script para somar os 20 primeiros números naturais pares. Use loop for.
5. Repita o exercício anterior usando loop while.

# 3

## Lista Exercícios

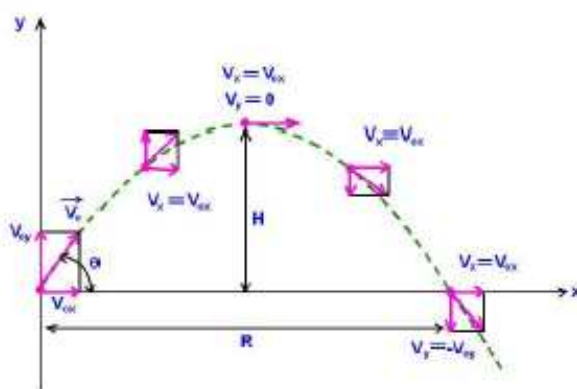


Figura 3.1: Diagrama mostrando o comportamento de um corpo (projétil) arremessado de um ângulo  $\theta_0$ .

### Questão 1

A figura 3.1 apresenta um gráfico do comportamento temporal de um corpo arremessado a partir de um ângulo  $\theta_0$ . A equação (3.1) apresenta o valor da altura  $y$  em função da distância horizontal percorrida ( $x$ ) e de parâmetros iniciais do lançamento e de constantes:  $\theta_0$ , ângulo inicial de arremesso, em radianos;  $g$  (aceleração da gravidade),  $y_0$  e  $v_0$ . Essa função define a trajetória de um objeto arremessado em um ângulo  $\theta_0$ , velocidade inicial  $v_0$  e altura inicial  $y_0$ .

$$y = \tan(\theta_0)x - \frac{g}{2v_0^2 \cos^2(\theta_0)}x^2 + y_0 \quad (3.1)$$

A partir dessas informações faça um script para:

- receber os dados da equação (3.1):  $\theta_0$ ,  $y_0$  e  $v_0$ ;
- criar um vetor  $x = x_0 : \Delta : x_f$  ( $x_0 = 0$ ) a partir dos dados lidos ( $\Delta$  e  $x_f$ ). De posse de todos os dados o programa deve gerar o valor de  $y$ . Considere  $g = 10m/s^2$ . Calculado  $y$ , o programa deve plotar o gráfico  $x \times y$ .
- Depois de plotar o primeiro gráfico, deve-se ler novamente outro valor de ângulo. Mantenha o gráfico anterior e plote (na mesma figura) o resultado para esse novo ângulo (todos os outros parâmetros devem ser os mesmos). Caso o segundo ângulo seja igual ao primeiro, deve-se sair do programa, com uma mensagem de erro.
- Assim, ao final do script, deve-se gerar uma figura com dois gráficos (na mesma figura) para ângulos diferentes. Coloque título, *labels* nos eixos e use **grid**.

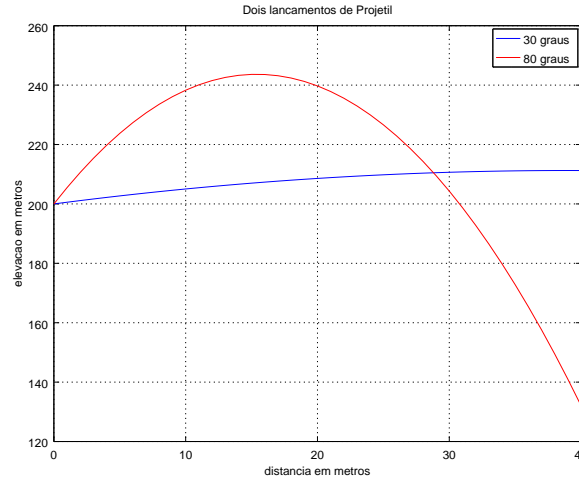


Figura 3.2: Diagrama mostrando dois gráficos com ângulos  $\theta_0$  iguais a 30 e 80. Os dados dessa simulação foram:  $v_0 = 30m/s$ ,  $y_0 = 200m$ ,  $x_f = 40$ ,  $\Delta$  (incremento) = 1.



## Questão 2

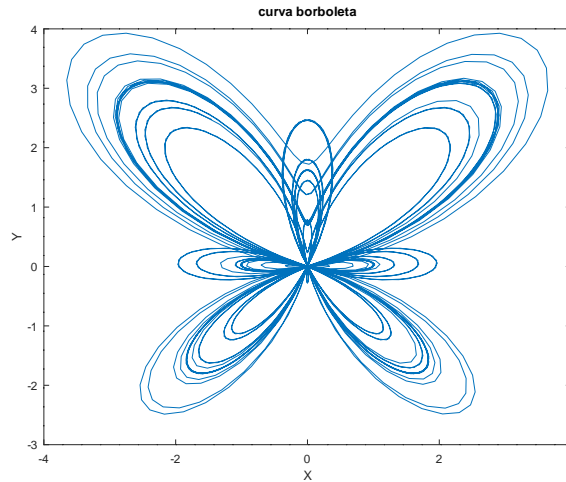


Figura 3.3: Curva "borboleta" da função dada pela equação 3.2

A "curva borboleta" é criada pelas funções parametrizadas dadas pela equação (3.2).

$$\begin{cases} x = \text{sen}(t) \left( e^{\cos(t)} - 2\cos(4t) - \text{sen}^5\left(\frac{t}{12}\right) \right) \\ y = \cos(t) \left( e^{\cos(t)} - 2\cos(4t) - \text{sen}^5\left(\frac{t}{12}\right) \right) \end{cases} \quad (3.2)$$

Faça um script para gerar os valores de  $x$  e  $y$  para  $0 \leq t \leq 100$  com  $\Delta t = 1/16$ . Como os valores gerados faça os gráficos:

1.  $x \times y$ ,
2.  $x \times t$
3.  $y \times t$ .

Use subplot para criar os três gráficos em uma única figura. Coloque labels e título.

Faça uma outra figura (figure(2)) e plote apenas o gráfico de  $x \times y$ . Coloque título e labels. Salve essa segunda figura como borboleta.jpg.

**Questão 3**

A curva da borboleta dada em (3.2) pode ser escrita em coordenadas polares como:

$$r = e^{\sin(\theta)} - 2\cos(4\theta) - \sin^5\left(\frac{2\theta - \pi}{24}\right) \quad (3.3)$$

Faça um script para calcular  $r$  a partir de  $\theta$ , com  $0 \leq \theta \leq 8\pi$  com  $\Delta\theta = \pi/32$ . Use a função **polar** para traçar gráfico dessa função parametrizada dada na equação 3.3). Use o **help polar** para saber como usar a função.

**Questão 4**

Faça um script Octave para gerar um triângulo formado pelas letras do alfabeto, na ordem crescente: um 'a' na linha 1, dois 'bs' na linha 2, três 'cs' na linha 3 e assim por diante. O programa deve aceitar um número inteiro  $n$  (no máximo 29) que definirá o número de linhas. A figura abaixo mostra o resultado para  $n = 6$ .

```
a
b b
c c c
d d d d
e e e e e
f f f f f f
```

**Questão 5**

Faça um script Octave para gerar um triângulo de base  $n$  e altura  $n$ , sendo  $n \leq 15$  inteiro dado pelo usuário. A cada linha do triângulo deve-se escrever antes a frase "linha  $i$ " sendo  $i$  o número da linha. Use `\t` para tabular os asteriscos. A figura abaixo é um exemplo para  $n = 7$

```
linha 1:      *
linha 2:      * *
linha 3:      * * *
linha 4:      * * * *
linha 5:      * * * * *
linha 6:      * * * * * *
linha 7:      * * * * * * *
```

---

**Questão 6**

Fazer um script que calcule e escreva a soma dos 30 primeiros termos da série abaixo:

$$\frac{480}{10} - \frac{475}{11} + \frac{470}{12} - \frac{465}{13} + \dots$$

---

**Questão 7**

Fazer um script para calcular e escrever o valor do número  $\pi$ , com precisão de 0.0001, usando a série:

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \dots \quad (3.4)$$

- Para definir a precisão do  $\pi$  calculado por você, efetue a diferença dele com o  $\pi$  do Octave (variável *pi*).

---

**Questão 8**

Faça um programa para efetuar o somatório dado abaixo. Ao final, o programa deve imprimir na tela: "O somatório vale: xxxxx", onde xxxxx é o valor calculado.

$$S = \sum_{i=-10}^{10} \frac{2i+1}{i+1} + \sum_{i=-10}^{10} \frac{i+1}{i-1}$$

---

---

**Questão 9**

---

$v(m/s)$	10	20	30	40	50	60	70	80
$F(N)$	25	70	380	550	610	1220	830	1450

---

Tabela 3.1: Dados do experimento com o túnel de vento, com a força medida (em Newtons) e velocidade (m/s)

Um "túnel de vento" é um experimento no qual a pessoa fica suspensa em um túnel (um tubo) que é submetido a ventos de diferentes velocidades, com o objetivo de medir a força de resistência do ar. Em um experimento realizado foram obtidos os resultados dados na tabela 3.1. Para esses dados fazer um script para:

1. Escrever os dados da tabela em uma matriz,  $V$ .
  2. Criar o gráfico de  $v \times F$ , com símbolos (não usar os pontos). Use círculos pretos como símbolos. Coloque título e label nos eixos. Use grid.
  3. Obter um polinômio de grau  $n$  (escolha um  $n$  que melhor adeque os dados).
  4. Plotar a figura com os dados e com o polinômio obtido.
-

**Questão 10**

Faça um script para ajustar uma curva para os dados fornecidos no arquivo *lista04\_dados01.mat*. Para isso siga os passos dados abaixo.

- a) Carregue o arquivo *lista04\_dados01.mat* no Octave ou o arquivo *lista04\_dados01\_matlab.mat* no Matlab. Repare quais as variáveis foram carregadas.
- b) Dentre as variáveis carregadas está a matriz  $M$ . Verifique o tamanho dessa matriz. Ela contém dados do censo norte-americano entre os anos de 1790 e 1990. Plote o gráfico com os anos da abscissa. Coloque títulos e label nos eixos. Salve a figura como *census.jpg*.
- c) Obtenha um ajuste linear, pela reta  $y = ax + b$ . Use o comando *polyfit* para obter os coeficientes  $a$  e  $b$  da reta.
- d) Obtida a reta, calcule um novo vetor,  $y_1$ , agora usando como função a reta obtida e como entrada o valor de  $x$  (anos). Trace, no mesmo gráfico, a função  $y_1 = f(x)$ , usando uma cor diferente.
- e) Salve a nova figura como *census2.jpg*.
- f) Repita o procedimento, agora criando uma nova figura e fazendo o ajuste para um polinômio de segunda ordem. Salve a figura, plotando os dados originais e o resultado do ajuste no mesmo gráfico. Salve com o nome de *census3.jpg*.
- g) Repita o item acima, para um polinômio de ordem 4. Salve a figura como *census4.jpg*.
- h) Ao enviar as soluções da lista de exercício, todas essas figuras devem ser enviadas no seu arquivo ZIP.

---

**Questão 11**

Faça um script que crie um menu como o mostrado abaixo. O programa deve ficar sendo executado enquanto o usuário não escolher a opção de sair. O programa deve executar as três tarefas pedidas: cálculo da área do círculo, cálculo da área do triângulo e cálculo da área do trapézio. Cabe ao programador definir as restrições e condições para o programa.

Digite a opcao:

- (a) Área do Círculo
- (b) Área do Triângulo
- (c) Área do Trapézio
- (d) Sair

---

**Questão 12**

Faça um script Octave/Matlab para gerar um vetor de 12 números aleatórios, inteiros, entre 0 e 100 armazená-los em um vetor. Depois o programa deve, a partir desse vetor criado, gerar outros dois vetores: um com os números ímpares e outro com os números pares. O script deve mostrar todos os 3 vetores como vetores coluna. Deve mostrar ainda a soma dos elementos do vetor par e do vetor ímpar. E deve também mostrar os maior e o menor elemento de cada vetor.