Campus Pato Branco

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ - UTFPR
CURSO DE ENGENHARIA ELÉTRICA/COMPUTAÇÃO
DISCIPLINA: SISTEMAS LINEARES
PROFESSOR: CÉSAR R. CLAURE TORRICO

Alunos:	**Nota:**
1 -	
2 -	
3 -	**Data:**

Encontro 1

Introdução ao software *Matlab*

1.1 Objetivo:

Introduzir os conceitos e comandos fundamentais do programa ***Matlab***.

Apresentação de comandos básicos e específicos que serão usados na disciplina de Sistemas Lineares.

Resolução de exercícios como forma de fixação.

1.2 Programa *Matlab*:

- Abrindo o programa: Dois cliques no ícone 

.

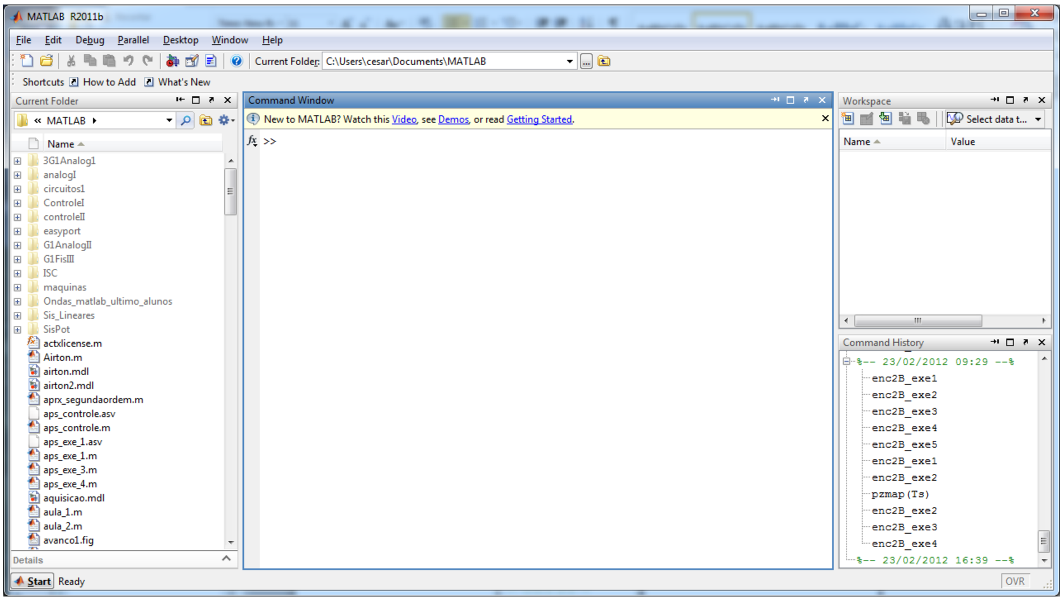

Figura 1: Tela inicial.

- Abrindo editor de programa: Clicar em "*New M-File*" .

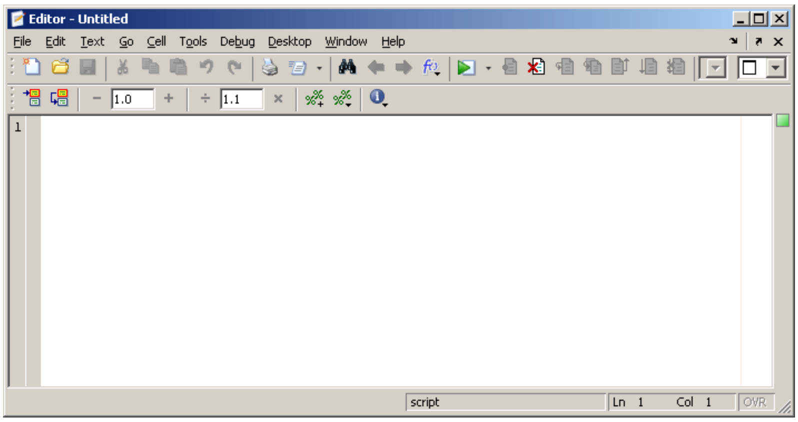

Figura 2: Tela do editor de programa.

- Pode-se escolher trabalhar no Editor de Programa (***Editor***) ou na Janela de Comando (***Command Window*** >>).
  Normalmente quando se deseja apenas testar alguma função utiliza-se a Janela de Comando, quando deseja-se "salvar" o programa feito utiliza-se o Editor de Programa.

1.2.1 Funções Matemáticas Básicas:

O *Matlab* usa regras semelhantes as da matemática formal para as expressões:

```
>> a=3/4
a = 0.75
```

Caso não seja desejado mostrar o valor da resposta, deve ser colocado ao final de cada declaração o caractere ' **;** '. Desta forma a apresentação da resposta é suprimida. Pode-se também inserir um texto de comentário digitando antes do texto o caractere ' **%** '.

```
>> b = 2            % sem o caractere ';' no final da sentença o
resultado é apresentado.

b = 2
>> c = 3;           % com o caractere ';' no final da sentença o
resultado não é apresentado.

>> d = b+c          % o resultado é armazenado na variável 'd' e
é apresentado.
d = 5

>> b+c              % se nenhum nome é atribuído a uma variável
ela é armazenada em "ans".
ans = 5
```

Quanto às operações matemáticas básicas, o ***Matlab*** as realiza na seguinte ordem:

^ potenciação;

    * multiplicação; e / divisão;

        + adição; e - subtração.

As operações são realizadas da esquerda para direita seguindo a ordem de prioridade definida anteriormente, mas os parênteses podem afetar a ordem das operações.

```
>> 1+2^3/4*2                 % 1+{[(2^3)/4]*2}
ans = 5

>> 1+2^3/(4*2)               % 1+[(2^3)/(4*2)]
ans = 2

>> (1+2)^3/(4*2)             % [(1+2)^3]/(4*2)
ans = 3.3750
```

1.2.2 Variáveis e funções predefinidas:

O Matlab possui variáveis predefinidas que são apresentadas a seguir:

- *i* e *j* = $\sqrt{-1}$ (se usar i e j, limpar após o uso com o comando *clear*)
- pi = π (3.1416)
- Inf = ∞
- NaN = não número (ex.: 0/0)
- eps = 2.2251e-16 (usado para significar um número muito pequeno, próximo de zero)

```
>> 2*pi
ans = 6.2832
>> d=4/Inf
d = 0
>> z=2+2i
z = 2.0000 + 2.0000i
```

O Matlab possui ainda várias funções matemáticas predefinidas, tais como funções trigonométricas, exponenciais, logarítmicas, raízes quadrada, cúbica, funções lógicas booleanas, entre muitas outras.

```
>> u=sin(3*pi/4)  % o Matlab usa como default a medida de ângulo em radianos
u = 0.7071
>> v=sqrt(4)  % o comando sqrt (square root) executa a operação raiz quadrada
v = 2
>> abs(z)  % valor absoluto (módulo) da variável z definida anteriormente
ans = 2.8284
>> angle(z)  % ângulo (fase) do número variável z definida anteriormente
ans = 0.7854
>> exp(-1)  % exponencial
ans = 0.3679
>> log10(100)  % logaritmo na base 10
ans = 2
```

1.2.3 Polinômios:

A maneira de escrever um polinômio usando o ***Matlab*** é realizada digitando-se seus coeficientes em forma de vetor, conforme mostrado abaixo:

```
>> P = [1 6 8];     % declaração do polinômio P = s² + 6s + 8
```

As raízes são obtidas utilizando o comando "***roots***":

```
>> R = roots(P)
R =    -4
       -2
```

A utilização das raízes transformando-as em polinômio pode ser feita através do comando "***poly***".

```
>> S = poly([-2 -4])      % raízes do polinômio
S = 1  6  8
```

Para multiplicar dois ou mais polinômio pode ser utilizado o comando "*conv*".

```
>> P1=[1  4  8];          % polinômio P1 = s² + 4s + 8
>> P2=[1  2];             % polinômio P2 = s + 2
>> PM=conv(P1,P2)         % PM = (s² + 4s + 8)*(s + 2)
PM = 1  6 16 16           % PM = s³ + 6s² + 16s + 16
```

Para dividir dois polinômios utiliza-se o comando "***deconv***". Como esta operação pode não ser exata, ou seja, pode existir um resto, utiliza-se a seguinte forma:

```
>> [Q,R]=deconv(P1,P2)    % equivale operação (s2 + 4s + 8)/(s + 2)
Q = 1  2                  % Q = s + 2
R = 0  0  4               % R = 4
```

A variável Q armazena o quociente da divisão, ao passo que o resto é armazenado na variável R.

O comando **"[*r,p,k*]=*residue*(*num,den*)"** encontra os **resíduos** (r), **pólos** (p) e **termos diretos** (k) de uma expansão em frações parciais a partir da **razão entre dois polinômios**:

$$\frac{b(s)}{a(s)} = \frac{b_1 s^m + b_2 s^{m-1} + b_3 s^{m-2} + \cdots + b_{m+1}}{a_1 s^n + a_2 s^{n-1} + a_3 s^{n-2} + \cdots + a_{n+1}} = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \cdots + \frac{r_n}{s-p_n} + k(s)$$

$$\frac{b(s)}{a(s)} = \frac{5s^3 + 3s^2 - 2s + 7}{-4s^3 + 8s + 3} = \frac{-1,4167}{s-1,5737} + \frac{-0.6653}{s+1,1644} + \frac{1,3320}{s+0,4093} - 1,2500$$

```
>> b = [ 5 3 -2 7];
>> a = [-4 0 8 3];
>> [r, p, k] = residue(b,a)
r =
   -1.4167
   -0.6653
    1.3320
p =
    1.5737
   -1.1644
   -0.4093
k =
   -1.2500
```

Agora, a conversão da expansão em frações parciais para polinômio pode ser feita assim:

```
>> [b,a] = residue(r,p,k)
b =
   -1.2500   -0.7500    0.5000   -1.7500
a =
    1.0000   -0.0000   -2.0000   -0.7500
```

$$\frac{b(s)}{a(s)} = \frac{-1.25s^3 - 0.75s^2 + 0.50s - 1.75}{s^3 - 2.00s - 0.75}$$

O polinômio é dado no formato normalizado!

1.2.4 Vetores:

Um vetor é a forma adequada de representação de sinais amostrados. A seguir algumas formas de geração de vetores:

```
>> t=0:0.01:5;
```

O código anterior cria um vetor "`t`" iniciando em 0 e terminando em 5, com intervalo de amostragem de 0.01.

```
>> n=0:100;
```

O código anterior cria um vetor "`n`" iniciando em 0 e terminando em 100, com intervalo unitário.

```
>> x=linspace(0,20);
```

O código anterior gera um vetor "`x`" de 100 pontos, igualmente espaçados entre 0 e 20.

```
>> y=linspace(0,20,30);
```

O código anterior gera um vetor "`y`" de 30 pontos, igualmente espaçados entre 0 e 20.

1.2.5 Gráfico de funções 2D:

Para plotar uma função de tempo continuo é necessário dois vetores de igual dimensão, um representando o eixo vertical e o outro o eixo horizontal:

```
t=0:0.01*pi:4*pi;
ft=4*sin(t);
plot(t,ft);
```

Para funções de tempo discreto substituímos o comando `plot` por `stem`:

```
n=0:26;        %O intervalo é discreto
fn=4*sin(pi*n/13);
stem(n,fn);
```

Dentre as funções periódica de interesse podemos citar:

- `sin`, $\rightarrow$ seno
- `cos`, $\rightarrow$ coseno
- `square`, $\rightarrow$ gera onda quadrada

- `sawtooth`, → gera onda dente de serra

Maiores detalhes podem ser obtidos com o comando "`help plot`" ou "`help stem`"

Exemplo de aplicação de algumas operações de tempo contínuo na variável independente:

```
clear all;
clc;

t= -10:0.01:10
ft = 4*sawtooth(t) % sinal original
ft_r = 4*sawtooth(-t) % sinal refletido
ft_d = 4*sawtooth(t-2) %sinal deslocado em 2 para a direita
ft_c = 4*sawtooth(2*t) %sinal comprimido em 2 vezes

plot(t,ft)
title('Sinal original');

figure % comando para apresentar as figuras em janelas separadas
plot(t,ft_r);
title('Sinal refletido');

figure % comando para apresentar as figuras em janelas separadas
plot(t,ft_d);
title('Sinal deslocado em 2 para a direita');

figure % comando para apresentar as figuras em janelas separadas
plot(t,ft_c);
title('Sinal comprimido em 2 vezes');
```

Exemplo de aplicação de algumas operações de tempo discreto na variável independente:

```
clear all;
clc;

n= -20:20
fn = 4*sin(4*pi*n/19) % sinal original
fn_r = 4*sin(4*pi*(-n)/19) % sinal refletido
fn_d = 4*sin(4*pi*(n-3)/19) %sinal deslocado em 3 para a direita
fn_c = 4*sin(4*pi*(2*n)/19) %sinal comprimido em 2 vezes

stem(n,fn)
title('Sinal original');

figure % comando para apresentar as figuras em janelas separadas
stem(n,fn_r);
title('Sinal refletido');

figure % comando para apresentar as figuras em janelas separadas
stem(n,fn_d);
title('Sinal deslocado em 3 para a direita');

figure % comando para apresentar as figuras em janelas separadas
stem(n,fn_c);
title('Sinal comprimido em 2 vezes');
```

1.2.6 Decomposição de funções em degraus e rampas:

Muitas vezes para representação de funções em algum programa computacional é necessário decompor em funções básicas do tipo degraus e rampas.

Por exemplo, seja a função apresentada na Fig. 3.

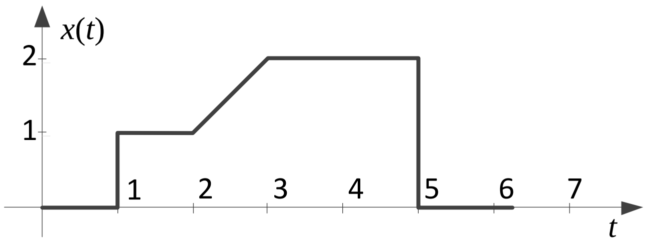

Figura 3: Função arbitrária.

A decomposição em degraus e rampas está dada pela equação:

$$x(t) = u(t-1) + r(t-2) - r(t-3) - 2u(t-5)$$

Para plotar esta função no matlab, segue-se a sequência de comandos apresentado a seguir:

```
t=-7:0.001:7;
xt=heaviside(t-1)+(t-2).*heaviside(t-2)-(t-3).*heaviside(t-3)-2*heaviside(t-5);
plot(t,xt);
title('Sinal original');

t2=-t; %espelhamento do tempo para geração do sinal refletido
xt_r=heaviside(t2-1)+(t2-2).*heaviside(t2-2)-(t2-3).*heaviside(t2-3)-
2*heaviside(t2-5); %sinal refletido
figure;
plot(t,xt_r);
title('Sinal refletido');
```

1.2.7 Exercícios a serem entregues no dia da primeira prova (12/abril/2016)

Utilizando o "***Editor***" no ***Matlab*** faça os exercícios.

1) Obtenha as raízes dos polinômios:
   a. $6s^3 + 3s^2 - 5s$
   b. $s^6 + 4s^3 + 1$
   c. $(s^3 + 2s - 10)(2s^2 - 10s)$

2) Obtenha os polinômios que possuem as seguintes raízes.
   a. $s = -3$ e $s = -8$
   b. $s = -4$, $s = 5$ e $s = 2$
   c. $s = -5$, $s = -6+9j$ e $s = -6-9j$

3) Expandir as seguintes $B(s)/A(s)$ em frações parciais utilizando o Matlab.

   a. $\dfrac{B(s)}{A(s)} = \dfrac{128}{4s^2 + 32s + 64}$

   b. $\dfrac{B(s)}{A(s)} = \dfrac{s+8}{(s+2)^4}$

c. $\dfrac{B(s)}{A(s)} = \dfrac{(s-1)(s+3)}{(s+6)(s+5)(s+2)}$

4) Criar duas funções senoidais de tempo contínuo com as seguintes características:
$x_1(t)$ com amplitude de 40 e frequência de 30Hz. (Plotar para conferência)
$x_2(t)$ com amplitude de 5 e frequência de 40Hz. (Plotar para conferência)

Plotar os sinais a seguir:

a. $y(t) = 2x_1(t-\pi/4)$
b. $y(t) = 3x_1(t)+4\,x_2(t)$
c. $y(t) = x_1(t)x_2(t)$
d. $y(t) = -2\,x_2(-t/2)$
e. $y(t) = 3x_1(-3t+\pi/3)$

5) Criar duas funções senoidais de tempo discreto com as seguintes características:
$x_1(n)$ com amplitude de 20 e período de 15 amostras. (Plotar para conferência)
$x_2(n)$ com amplitude de 2 e período de 10 amostras. (Plotar para conferência)

Plotar os sinais a seguir:

a. $y[n] = 3x_1[n]$
b. $y[n] = 4x_1[n]-2x_2[n]$
c. $y[n] = 2x_1[-n]\,x_2[-n]$
d. $y[n] = -3x_1[-n]+2x_2[n]$
e. $y[n] = x_1[-5n+3]$

6) Dadas as funções $x(t)$ e $y(t)$ apresentadas na Figura 4.

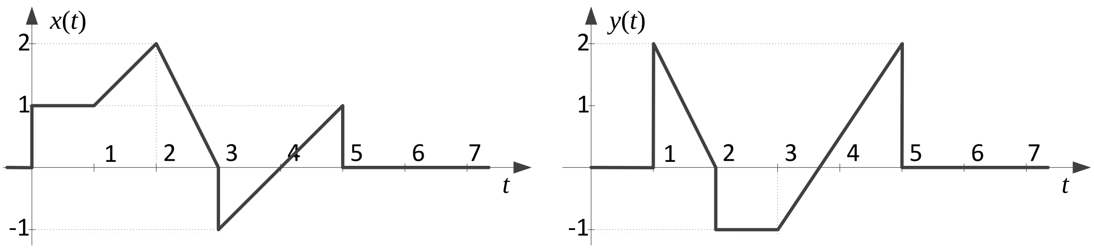

Figura 4:

a. Representar como um somatório de rampas e degraus e logo em seguida plotar para verificação.
b. Obter e plotar $z(t) = x(2t-1)$
c. Obter e plotar $z(t) = x(t-1)\,y(t+1)$
d. Obter e plotar $z(t) = -\,x(-2t-1)+y(t-1)$