Aula 2P - Comandos básicos do Matlab aplicados a PDS

Bibliografia

- HAYKIN, S. S.; VAN VEEN, B. **Sinais e sistemas**, Bookman, 2001. ISBN 8573077417. Páginas 71-76.
- INGLE, V. K.; PROAKIS, J. G. **Digital signal processing using Matlab®**, 2nd ed., Thomson, 2007. ISBN 0495073113. Páginas 7-17.

1. Introdução

O Matlab é uma ferramenta muito útil no estudo de problemas e no desenvolvimento de projetos em Engenharia sendo utilizado em universidades e empresas ao redor do mundo.

Na área de Engenharia Elétrica e, mais precisamente, em Processamento de Sinais vem adquirindo um caráter quase fundamental.

O principal motivo deste sucesso é a utilização maciça de vetores e matrizes para representar dados de uma forma simples (Matlab = *Mat*rix *Lab*oratory). Esta forma de representação praticamente elimina a necessidade de utilização de laços FOR ou WHILE simplificando e acelerando muito os programas. EM OUTRAS PALAVRAS, EM MATLAB, SEMPRE QUE POSSÍVEL (OU SEJA, QUASE SEMPRE!) NÃO UTILIZE LAÇOS FOR OU WHILE!

O objetivo desta aula é (re) ver alguns conceitos básicos de programação em Matlab. Durante o curso veremos muitos outros detalhes técnicos.

Lembre-se: sempre que você ficar na dúvida sobre a utilização de um comando, a função <`help` *`comando`*> pode lhe ajudar.

2. Gerando vetores

2.1. O operador :

O operador : é utilizado para gerar e acessar elementos de um vetor.

```
Vetor = valor inicial: passo: valor final
```

Quando o passo é unitário, ele pode ser omitido.

- Exemplos de utilização

A. gerar um vetor `x` com os números inteiros de zero a cinco

```
>> x = 0:5
x =
0     1     2     3     4     5
```

b. gerar um vetor `y` indo de 0 a 1 com passo de 0.1.

```
>> y = 0:0.1:1
```

```
y =
      0      0.1000      0.2000      0.3000      0.4000      0.5000
0.6000  0.7000    0.8000    0.9000    1.0000
```

c. mostrar o segundo elemento do vetor x

```
>> x(2)
ans =
     1
```

Exercício

1. Gerar um vetor `x` de números pares de 0 a 50.

Comandos:

2.2. A função `linspace`

A função `linspace` é uma forma prática de se gerar vetores quando sabemos quantos pontos ele deve ter.

```
Vetor = linspace (valor inicial, valor final, no. de pontos)
```

- Exemplos de utilização

A. Gere um vetor de 1000 pontos com valores entre zero e 1 igualmente espaçados.

```
>> v = linspace(0,1,1000);
```

b. Repita o exercício anterior, mas com os valores em ordem decrescente.

```
>> v = linspace(1,0,1000);
```

Exercício

2. Gere um vetor `x` de 5000 pontos com valores entre 0 e `2*pi`.

Comandos:

2.3. Vetores especiais

Existem vetores pré-definidos pelo Matlab e que são muito úteis. Dois deles são o `ones(num.linhas, num.colunas)` e o `zeros(num.linhas, num. Colunas)` que geram, como os nomes dizem, vetores constituídos de uns e de zeros respectivamente.

- Exemplos de aplicação

A. Gere um vetor constituído de 10 zeros.

```
>> x = zeros(1,10)
```

```
x =
     0     0     0     0     0     0     0     0     0     0
```

b. Gere um vetor constituído por 5000 uns.

```
>> y = ones(1,5000);
```

Exercício

3. Gere uma matriz 2x2 constituída por zeros.

Comandos:

2.4. Concatenação de vetores

Uma ferramenta muito interessante do Matlab é a possibilidade de combinar vetores para formar outros (concatenar vetores). Veja os seguintes exemplos.

- Exemplos de aplicação:

A. Gere um vetor de cinco zeros seguidos por cinco uns.

```
>> vector = [zeros(1,5) ones(1,5)]
vector =
     0     0     0     0     0     1     1     1     1     1
```

B. Gere um vetor contendo os números inteiros entre zero e 10 em ordem crescente seguidos pelos mesmos em ordem decrescente.

```
>> x = [0:10 10:-1:0]
x =
0     1     2     3     4     5     6     7     8     9    10
10     9     8     7     6     5     4     3     2     1     0
```

Exercício

4. Construa um vetor constituído pelos números pares de 0 a 10 seguido pelos números ímpares de 0 a 10.

Comandos:

2.5. Operações entre vetores

O Matlab permite somar (+), subtrair (-), multiplicar (.*) , dividir (./) vetores. Essas operações são realizadas elemento a elemento e só podem ser aplicadas entre vetores de mesmo comprimento.

Além disso, quase todas as suas funções (trigonométricas, exponenciais e outras) podem ser aplicadas a um vetor sendo que elas operam também elemento a elemento.

- Exemplos de aplicação

a. Sendo `x = [2 3 7]` e `y = [0 -1 3]` escreva a resposta de cada um desses comandos executados no Matlab.

I) `x + y` `[2 2 10]`

ii) `x – y` `[2 4 4]`

iii) `x.*y` `[0 -3 21]`

b. Como gerar a partir do vetor `x = 0:0.001:1` um vetor com números de 1 a 11?

`V = 10*x+1`

Exercício

5. Sendo `x = [2.1 -2 3]` e `y = [0 -1 3]`, escreva o vetor resultante das seguintes operações:

i) `3*x` ii) `x.*y` iii) `x./y` iv) `y.^2`

Respostas:

3. Gráficos

Outra característica muito interessante do Matlab para um engenheiro é a facilidade de se construir gráficos complicados com ele de uma maneira muito simples. Os dois comandos mais utilizados são:

```
plot(vetor.abscissa, vetor.ordenada, 'modo');
stem(vetor.abscissa, vetor.ordenada);
```

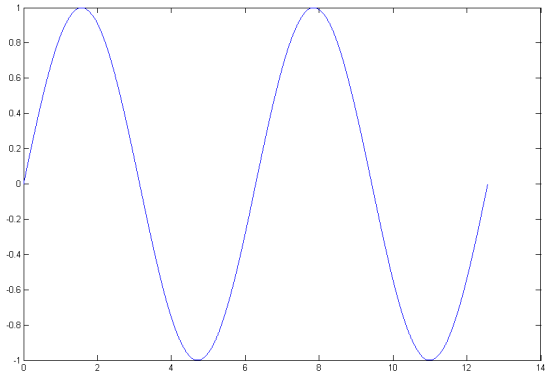
O comando `plot` traça um gráfico colocando seu primeiro argumento no eixo horizontal e seu segundo argumento no eixo vertical. A "string" 'modo' indica a forma como o gráfico será traçado. Veja `help plot` para mais detalhes.

`Stem` traça um gráfico da seqüência em seu segundo argumento como palitos com círculos no valor dos dados usando seu primeiro argumento como abscissa. Veja os exemplos.

- Exemplos de aplicação

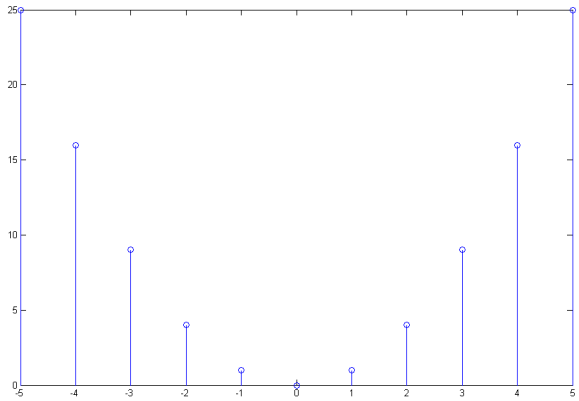
a. Faça um gráfico da função $y(t) = \sin(t)$ para $x \in [0,4\pi]$

```
>> t = linspace(0,4*pi,5000);
>> y = sin(t);
>> plot(t,y)
```



b. Faça um gráfico da função $y[n] = n^2$ para $n \in Z, -5 \leq x \leq 5$.

```
>> n = -5:5;
>> y = n.^2;
>> stem(n,y)
```



Alguns comandos interessantes:

I) `grid` – coloca linhas de grade no gráfico

ii) `title` – permite acrescentar um título ao gráfico

iii) `xlabel` – permite acrescentar um título no eixo das abscissas

iv) `ylabel` – permite acrescentar um título no eixo das ordenadas

v) `hold on` – não apaga o gráfico atual antes de fazer o seguinte

Exercícios

6. Faça um gráfico de $y[n] = \sin^2\left(\frac{\pi}{12}n\right)$ e $z[n] = \cos^2\left(\frac{\pi}{12}n\right)$ para $-30 \leq n \leq 30$ na mesma figura. O gráfico de `y[n]` deverá ficar em azul e o de `z[n]` em vermelho.

Comandos:

4. *Scripts*

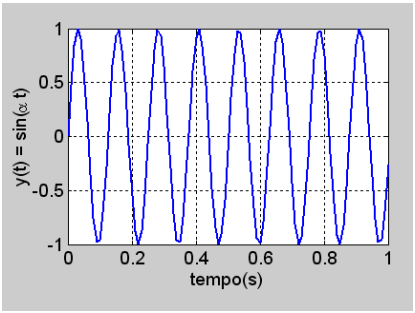
- Até este ponto, todas as nossas interações com o Matlab têm sido através da linha de comando. Entramos comandos ou funções na linha de comando e o Matlab interpreta nossa entrada e toma a ação apropriada. Este é o modo de operação preferencial quando nossa sessão de trabalho é curta e não repetitiva.
- No entanto, o real poder do Matlab para análise e projeto de sistemas vêm da sua habilidade de executar uma longa seqüência de comandos armazenados num arquivo. Estes arquivos são chamados de **arquivos-M** porque seus nomes têm a forma *nomearq.m*.
- Um *script* é um tipo de arquivo-M. *Scripts* são arquivos-textos comuns e podem ser criados usando um editor de texto.
- Um *script* é uma seqüência de comandos e funções comuns usados na linha de comando. Um *script* é invocado na linha de comando digitando-se o nome do arquivo. *Scripts* podem invocar outros *scripts*. Quando um *script* é invocado, o Matlab executa os comandos e funções no arquivo como se eles tivessem sido digitados diretamente na linha de comando.
- O *script* opera sobre as variáveis do espaço de trabalho.

- Suponha por exemplo que desejemos fazer um gráfico da função $y(t) = \sin \alpha t$ em que α é uma variável que queremos variar.
- Usando o editor de texto do Matlab (basta ditar `edit` na linha de comando), podemos escrever um *script* chamado `plotdata.m` como mostrado a seguir.

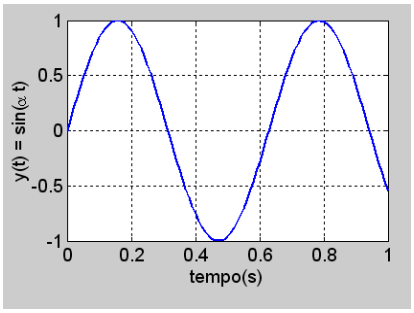
```
% Este e um script para fazer um grafico da funcao y = sin(alfa*t)
% O valor de alfa precisa existir no espaco de trabalho antes
% de se chamar este script
t = 0:0.01:1;
y = sin(alfa*t);
plot(t,y);
xlabel ('tempo(s)');
ylabel('y(t) = sin(\alpha t)');
grid on;
```

- É importante salvar o *scritpt* no mesmo diretório em que se está trabalhando na linha de comando. Caso contrário, ao tentar executar o *script* o Matlab não encontrará o arquivo e exibirá uma mensagem de erro. Este erro é muito comum quando estamos começando a trabalhar com *scripts*.
- Uma vez digitado e salvo é muito fácil utilizar o *script*. Veja os exemplos a seguir:

```
>> alfa = 50;
>> plotdata
```




```
>> alfa = 10;
>> plotdata
```



- Ao escrever *scripts* é sempre interessante utilizar comentários, linhas que começam com %. Se você escrever linhas de comentário antes do começo das instruções do *script* Ao utilizar o comando `help` *`nomearq`* o Matlab apresenta estas linhas na tela. Por exemplo,

```
>> help plotdata
  Este e um script para fazer um grafico da funcao y = sin(alfa*t)
  O valor de alfa precisa existir no espaco de trabalho antes
  de se chamar este script
```

5. Funções

- Assim como os *scripts*, as funções definidas pelo usuário estão entre os recursos mais importantes e utilizados do Matlab. Uma função é um *script* que recebe um ou mais parâmetros do teclado e pode devolver um ou mais parâmetros ou executar uma tarefa.
- O formato de uma função no Matlab é o seguinte

```
function [outarg1, outarg2,...] = fname(inarg1, inarg2,...)
% Um comentário
% Mais um comentário
....
(código executável)
....
```

- `fname` é o nome da função criada e deve ser o nome do arquivo m em que foi gravado o arquivo. `inarg1, inarg2,...` são os argumentos de entrada e `outarg1, outarg2,...` são os argumentos de saída.
- A seguir damos um exemplo bastante simples de função. A função `somateste` recebe dois argumentos `a`, `b` e retorna a soma deles.

```
function res = somateste(a,b);
%Funcao para somar dois numeros a e b
res = a+b;
```

- Uma vez que você tenha salvado este arquivo como `somateste` no diretório corrente, você pode usá-lo como nos exemplos a seguir:

```
>> somateste(2, 4)
ans =
     6
>> a = 5;
>> b = -3;
>> res = somateste(a,b)
```

```
res =
     2
```

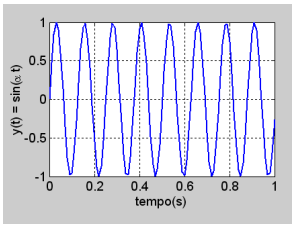
Exercícios

7. (a) Digite o *script* `plotdata` da página 7 e gere os gráficos dos exemplos da mesma página.

(b) Reescreva o *script* `plotdata` visto acima de forma que ele seja uma função que recebe a variável `alfa`. Ou seja, escreva uma função que faça um gráfico da função $y(t) = \sin \alpha t$ no intervalo $0 \le t \le 1$ e α é um parâmetro escolhido pelo usuário. Por exemplo, o comando:

```
>> plotdada(50)
```

deve gerar o gráfico



Resposta (listagem):

8. Gere um vetor `x` de 1000 valores aleatórios com distribuição uniforme no intervalo $[0,1]$.

Dica: use a função `rand` (não sabe como usar? Para que serve o `help`?). Escute as amostras armazenadas no vetor `x` usando o comando `sound(x)`.

Comandos:

9. Escreva uma seqüência de comandos do Matlab que forneça um vetor contendo 100 valores aleatórios uniformemente distribuídos no intervalo -1 a 1 e que faça um gráfico deste sinal.

Comandos:

10. Escreva uma seqüência de comandos Matlab que gere um gráfico do sinal $x[n]=\cos\left(\frac{\pi}{8}n\right)+0{,}2r[n]$ onde $r[n]$ é um vetor de números aleatórios com distribuição uniforme entre -1 e 1. Faça $0 \le n \le 99$. *(Dica: use o comando* `rand`*).*

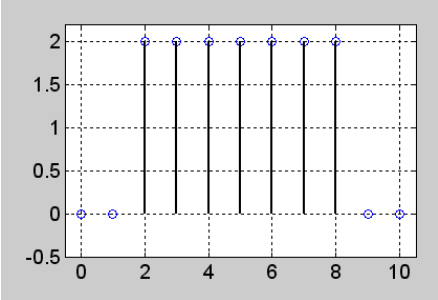
Comandos:

11. Escreva uma função Matlab chamada `pulso2graf` cujas entradas sejam dois números inteiros a e b com $a < b$. A função deverá fazer o gráfico de um pulso com amplitude 2 no intervalo $a \le n \le b$. O gráfico deve começar em $a-2$ e terminar em $b+2$.

Por exemplo, ao digitarmos:

```
>> pulso2graf(2,8);
```

devemos obter a figura



Listagem da função: