Universidade Federal de Minas Gerais
DEPARTAMENTO DE ENGENHARIA ELETRÔNICA
ELT129 – Oficina de Modelagem e Simulação

Professor: Leonardo Mozelli – lamoz@ufmg.br

Tutorial 1 – Comandos Básicos em Matlab

1 Introdução

Neste tutorial é apresentada uma série de comandos básicos em Matlab para manipulação de vetores, matrizes e gráficos, focando, sobretudo, em sinais e sistemas. São introduzidas operações com números complexos, gráficos de funções trigonométricas, e escala em gráficos. Além disso, funções para determinação das raízes de polinômios, decomposição em frações parciais, conversão de coordenadas e funções simbólicas em Matlab. Os exercícios exploram comandos relacionados.

2 Comandos básicos em Matlab e números complexos

Operações de adição, subtração, multiplicação, divisão e exponenciação podem ser realizadas usando os símbolos $+$, $-$, $*$, $/$ e $\wedge$.

O MATLAB predefine $i = j = \sqrt{-1}$ como uma constante complexa. Alternativamente, podem ser utilizados $1i$ ou $1j$. Um número complexo (a, b) ou $a + jb$ pode ser representado graficamente por um ponto cujas coordenadas Cartesianas são (a, b) em um plano complexo. Denotemos este número complexo z de tal forma que:

$$z = a + jb$$

Os números a (abscissa) e b (ordenada) referem-se, respectivamente, às partes real e imaginária de z. As partes real e imaginária de z podem ser expressas como:

$$\mathrm{Re}\{z\} = a$$
$$\mathrm{Im}\{z\} = b$$

Por exemplo:

```
z = -3-1i*4
```

atribui a constante complexa $-3 - i4$ à variável z.

Os componentes real e imaginário de z podem ser obtidos usando os operadores `real` e `imag`. No Matlab, a entrada para uma função é colocada entre parênteses após o nome da função:

```
z_real = real(z);
z_imag = imag(z);
```

Quando a linha termina com ponto e vírgula, o comando é avaliado, mas o resultado não é mostrado na tela. Isto pode ser útil quando se está calculando resultados intermediários ou construindo um *script*. Embora não seja mostrado, o resultado `z_real = -3` e `z_imag = -4` são calculados e estão disponíveis para operações adicionais.

Existem várias formas de computar o módulo de uma quantidade complexa. A trigonometria afirma que $z = -3 - i4$, que corresponde ao triângulo $-3 - 4 - 5$, tem módulo $|z| = |-3 - i4| = \sqrt{(-3)^2 + (-4)^2} = 5$.

O comando `sqrt` do Matlab é uma forma de computar a raiz quadrada:

```
>> z_mag = sqrt(z_real^2 + z_imag^2)
z_mag = 5
```

No Matlab, a maioria dos comandos com `sqrt` aceitam entradas de várias formas - incluindo constantes, variáveis e funções. De forma mais simples, o Matlab computa o valor absoluto de uma variável z por meio do comando `abs`:

```
>> z_mag = abs(z)
z_mag = 5
```

Além da magnitude, `z_mag`, notações polares requerem informação de fase. O comando `angle` fornece o ângulo em radianos de um número complexo.

```
>> z_rad = angle(z)
z_rad = -2.2143
```

Utilizando o Matlab também se pode calcular o ângulo em graus de duas formas diferentes:

```
>> z_deg = angle(z)*180/pi
z_deg = 126.8699

>> z_deg = rad2deg(angle(z))
z_deg = 126.8699
```

Note que a variável $pi = \pi$ é pré-definida. Na segunda forma de cálculo tem-se um exemplo de uma função sendo passada como argumento para outra função.

Ademais, é possível obter o ângulo de z usando a função arco-tangente com dois argumentos, `atan2`, da seguinte forma:

```
>> z_rad = atan2(z_imag,z_real)
z_rad = -2.2143
```

É preferível utilizar a função `atan2` ao invés de `atan`, pois a primeira determina corretamente em qual setor do plano complexo encontra-se o número z.

O Matlab suporta todos os comandos de funções trigonométricas, por exemplo: `cos, sin, tan, sec, csc, cot, acos, asin, atan, asec, asinh, atanh, asech, acsch, acoth`. Assim como no comando `angle`, as funções trigonométricas utilizam a unidade radianos. O Matlab também tem suporte para argumentos complexos para qualquer função trigonométrica.

Funções trigonométricas com argumentos de valores complexos podem contradizer o que sempre foi ensinado em matemática. Por exemplo, uma afirmação comum é que $|\cos(x) \leq 1|$. Enquanto isso é verdadeiro para x real, não é necessariamente verdadeiro para x complexo. Isto pode ser verificado usando:

```
>> cos(i)
ans = 1.5431
```

Da mesma forma, a afirmação de que é impossível obter o logaritmo de um número negativo é falsa em Matlab. Por exemplo, o valor principal de $\ln(-1)$ que é $j\pi$, pode ser verificado pela equação de Euler. Logaritmos de base 10 e base exponencial são computados usando os comandos `log10` e `log`, respectivamente. Por exemplo:

```
>> log(-1)
ans = 0 + 3.1416i
```

3 Coordenadas polares e conversão de coordenadas

Números complexos podem ser expressos em termos de coordenadas polares. Se (r, θ) são as coordenadas polares de um ponto $z = a + jb$, então:

$$z = a + jb = r\cos\theta + jr\mathrm{sen}\,\theta = r(\cos\theta + j\mathrm{sen}\,\theta)$$

A fórmula de Euler diz que:

$$e^{j\theta} = \cos\theta + j\mathrm{sen}\,\theta$$

Logo, números complexos podem ser expressos na forma Cartesiana $a + jb$ ou na forma polar usando r e $e^{j\theta}$ tal que:

$$a = r\cos\theta; \quad b = r\mathrm{sen}\,\theta;$$
$$r = \sqrt{a^2 + b^2}; \quad \theta = \tan^{-1}\left(\frac{b}{a}\right)$$

Note que r é a distancia do ponto z até a origem. Por esta razão, r é também chamado de magnitude (ou valor absoluto) de z e é denotado $|z|$. Similarmente, θ é chamado de ângulo de z e é denominado de $\angle z$.

3.1 Exemplos

1) Expresse o número $2 + j3$ na forma polar.

$$|z| = r = \sqrt{2^2 + 3^2} = \sqrt{13}$$
$$\angle z = \tan^{-1}\left(\frac{3}{2}\right) \approx 0.9828$$

Assim, pode-se escrever $2 + j3 = \sqrt{13}e^{j0.9828}$

Em Matlab, pode-se converter os sistemas de coordenadas a partir dos comandos `cart2pol` e `pol2cart`. Por exemplo:

```
>> [z_rad, z_mag] = cart2pol(2,3)
z_rad = 0.9828
z_mag = 3.6056
>> z_deg = rad2deg(z_rad)
z_deg = 56.3099
```

Logo, $z = 2 + j3 = 3.6056e^{j0.9828} = 3.6056e^{j56.3099}$

2) Converta $z = 4e^{-j\frac{3\pi}{4}}$ para a forma cartesiana.

```
>> [z_real, z_imag] = pol2cart(-3*pi/4,4)
z_real = -2.8284
z_imag = -2.8284
```

Logo, $z = 4e^{-j\frac{3\pi}{4}} = -2.8284 - j2.8284$

4 Adição de senóides

Duas senóides de mesma frequência, mas de diferentes fases podem ser adicionadas para formar uma única senóide de mesma frequência. Este fato pode ser constatado a partir da identidade trigonométrica:

$$C\cos(\omega_0 t + \theta) = C\cos(\theta)\cos(\omega_0 t) - C\mathrm{sen}\,(\theta)\mathrm{sen}\,(\omega_0 t) = a\cos(\omega_0 t) + b\mathrm{sen}\,(\omega_0 t)$$

sendo $a = C\cos(\theta)$ e $b = -C\mathrm{sen}\,(\theta)$. Consequentemente, $C = \sqrt{a^2+b^2}$ e $\theta = \tan^{-1}\left(\frac{b}{a}\right)$. A Figura 1 mostra que C e θ são a magnitude e o ângulo, respectivamente, de um número complexo $a - jb$. Em outras palavras $a - jb = Ce^{j\theta}$. Para encontrar C e θ, pode-se converter $a - jb$ para a forma polar. A magnitude e o ângulo do número polar resultante são C e θ, respectivamente.

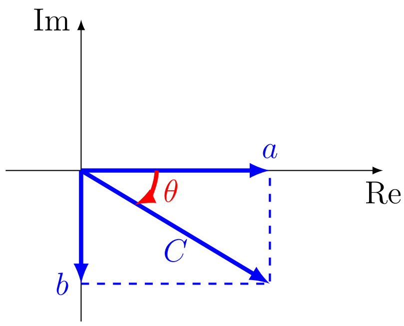

Figura 1: Adição de fasores de senóides.

A adição de senóides de mesma frequência pode ser clarificada pelo uso de fasores que representam as senóides. Uma senóide $C\cos(\omega_0 t + \theta)$ é representada por um fasor de comprimento C e ângulo θ com o eixo horizontal. A senóide $a\cos(\omega_0 t)$ é representada por um fasor horizontal de comprimento a ($\theta = 0$), enquanto $b\mathrm{sen}\,(\omega_0 t) = b\cos(\omega_0 t - \pi/2)$ é representada por um fasor vertical de comprimento b e um ângulo de $-\pi/2$ com a horizontal. A adição destes dois fasores resulta em um fasor de comprimento C e ângulo θ como mostrado na Figura 1.

4.1 Exemplo

Expresse $x(t) = \cos(\omega_0 t) - \sqrt{3}\mathrm{sen}\,(\omega_0 t)$ como uma senóide única.

Solução: Neste caso, $a = 1$ e $b = -\sqrt{3}$, logo

$$C = \sqrt{1^2 + (\sqrt{3})^2} = 2$$

$$\theta = \tan^{-1}\left(\frac{\sqrt{3}}{1}\right) = \frac{\pi}{3} = 60^\circ$$

Consequentemente, $x(t) = 2\cos(\omega_0 t + \pi/3)$. A amplitude C e a fase θ da senóide resultante são a magnitude e a fase do número complexo $1 - j\sqrt{3}$. Em Matlab:

```
>> a = 1; b = sqrt(3);
>> [theta, C] = cart2pol(a,-b)
theta = -1.0472
C = 2
>> theta_deg = rad2deg(theta)
theta_deg = -60
```

5 Operações com vetores

Considere a criação de vetores linha com elementos reais. Para isto, a notação `a:b:c` é usada, sendo `a` o valor inicial, `b` o passo, e `c` o valor final. Por exemplo:

```
>> k = 0:2:11
k = 0 2 4 6 8 10
>> k = 11:-10/3:0
k = 11.0000 7.6667 4.3333 1.0000
```

Se o tamanho do passo não é especificado, assume-se o passo igual a 1.

Para visualizar uma solução particular de um vetor deve-se especificar um índice. Por exemplo, o terceiro elemento de `k` é obtido a partir de:

```
>> k(3)
k = 4.3333
```

Similarmente, o segundo e o terceiro elementos de `k` são obtidos a partir de:

```
>> k(2:3)
k = 7.6667 4.3333
```

A palavra `end` é reservada no Matlab e é utilizada em múltiplas instâncias. Quando utilizada como indexador de um vetor, retorna o último elemento do vetor, por exemplo:

```
>> k(end)
k = 1.0000
>> k(2:end)
k = 7.6667 4.3333 1.0000
```

Adicionalmente, o comando `linspace(a,b)` pode ser utilizado para gerar 100 pontos igualmente espaçados, em escala linear, entre `a` e `b`. Um terceiro argumento opcional pode ser passado para a função informando o número de pontos a serem gerados no intervalo especificado. Assim, `linspace(a,b,N)` gera `N` pontos igualmente espaçados no intervalo $[a, b]$.

A representação por vetores permite criar e explorar rapidamente vários sinais. Por exemplo, considere uma senóide de frequência 10 Hz descrita por $f(t) = \text{sen}\,(20\pi t + \pi/6)$. Dois ciclos da senóide são incluídos no intervalo $0 \leq t \leq 0.2$. Um vetor t é usado para representar 500 pontos deste intervalo, i.e.:

```
>> t = linspace(0,0.2,500);
% alternativamente: t = 0:0.2/499:0.2;
```

Logo, a função $f(t)$ pode ser avaliada nesses pontos como segue:

```
>> f = sin(20*pi*t+pi/6);
```

6 Gráficos simples

O comando `plot` é uma maneira conveniente de visualizar dados. O gráfico da função $f(t)$ pode ser obtido a partir de:

```
>> plot(t,f)
```

Rótulos de eixos podem ser adicionados usando:

```
>> xlabel('t');
>> ylabel('f(t)');
```

O comando `title` insere um título ao gráfico.

O Matlab conecta pontos em um gráfico usando linhas sólidas como padrão. Se, por exemplo, o argumento `'o'` for adicionado, cada ponto é representado por um círculo ao invés de linhas conectando os pontos.

```
>> plot(t,f,'o');
```

Muitos outros argumentos podem ser combinados possibilitando alterar cores, tipo de linha, espessura da linha, dentre outros. Utilize o comando

```
>> help plot
```

para ver a maioria das opções disponíveis.

Alguns comandos importantes para construção de gráficos tridimensionais são `plot3`, `mesh`, `surf` e `contour3`. A sintaxe para utilização desses comandos pode ser verificada acessando o *Help* do Matlab a partir dos comandos:

```
>> help plot3
>> help mesh
>> help surf
>> help contour3
```

ou acessando *Help – Product help*.

Os comandos `semilogx`, `semilogy` e `loglog` operam como o comando `plot`, mas usam escala logarítmica para os eixos das abscissas e ordenadas.

7 Operações com matrizes

Dados valores inteiros m, n, e o vetor x. A função `eye(m)` cria uma matriz identidade de dimensão $m \times m$. As funções `zeros(m,n)` e `ones(m,n)` criam matrizes de dimensão $m \times n$ com todos os elementos iguais a 0 e 1, respectivamente. A função `diag(x)` transforma o vetor x em uma matriz diagonal. Em geral, a criação de vetores e matrizes requer a especificação de cada um dos elementos.

Considere o vetor $r = [1\ 0\ 0]$ e a matriz A, 3×2, cuja primeira coluna é $[2\ 4\ 0]^T$, e a segunda coluna é $[3\ 5\ 6]^T$.

```
>> r = [1 0 0]
r = 1 0 0
>> A = [2 3; 4 5; 0 6]
A = 2 3
    4 5
    0 6
```

Vetores linha podem ser transformados em vetores coluna. O vetor coluna c pode ser obtido a partir do vetor linha r usando a operação de transposição:

```
>> c = r'
c = 1
    0
    0
```

Podemos concatenar o vetor c e a matriz A usando o comando:

```
>> B = [c A]
B = 1 2 3
    0 4 5
    0 0 6
```

O elemento da primeira linha e segunda coluna de B pode ser acessado através de:

```
>> B(1,2)
ans = 2
```

Pode-se ainda selecionar a segunda linha inteira de usando:

```
>> B(2,:)
ans = 0 4 5
```

O determinante e a inversa de uma matriz podem ser obtidos a partir dos comandos `det` e `inv`, respectivamente.

O produto matricial de duas matrizes $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ e $B = \begin{bmatrix} 9 & 8 \\ 7 & 6 \end{bmatrix}$ pode ser computado por meio do comando:

```
>> A*B
ans = 23  20
      55  48
```

Há também uma outra forma de multiplicação, usualmente, denominada produto ponto-a-ponto, na qual os elementos das posições correspondentes de duas matrizes são multiplicados, por exemplo:

```
>> A.*B
ans =   9  16
       21  24
```

A operação de divisão de matrizes (multiplicar uma matriz pela inversa da outra) pode ser definida de duas formas diferentes:

```
>> A/B      % A*inv(B)
ans =   4  -5
        5  -6
>> A\B      % inv(A)*B
ans =   -11   -10
         10     9
```

A divisão ponto-a-ponto também pode ser utilizada.

8 Expansão em frações parciais

Existem várias técnicas para computar a expansão em frações parciais de uma função racional $F(x) = N(x)/D(x)$. Uma situação comum em análise de sistemas dinâmicos em que precisamos expandir uma função racional em frações parciais é quando queremos obter a resposta no domínio do tempo contínuo a partir de uma função de transferência e da tabela de transformadas de Laplace. Em Matlab usamos a função `residue`, cuja forma básica é:

```
>> [R,P,K] = residue(N,D)
```

Os dois vetores de entrada N e D especificam os coeficientes dos polinômios do numerador e denominador, respectivamente. O vetor R contém os coeficientes de cada fração parcial; o vetor P contém as raízes correspondentes de cada fração parcial. Para uma raiz repetida r vezes, as r frações parciais são ordenadas em potências ascendentes. Quando a função racional não é própria, o vetor K contém os termos diretos que são ordenados em potências descendentes da variável independente.

Considere encontrar a expansão em frações parciais de:

$$F(x) = \frac{x^5 + \pi}{x^4 - \sqrt{8}x^3 + \sqrt{32}x - 4}$$

Note que é complicado obter o resultado desta expansão à mão. Em Matlab, podemos encontrar a solução fazendo:

```
>> [R,P,K] = residue([1 0 0 0 0 pi],[1 -sqrt(8) 0 sqrt(32) -4])
R = 7.8888
    5.9713
    3.1107
    0.1112
P = 1.4142
    1.4142
    1.4142
    -1.4142
K = 1.0000  2.8284
```

Escrito na forma padrão, a expansão em fração parcial de $F(x)$ é:

$$F(x) = x + 2.8284 + \frac{7.8888}{x - \sqrt{2}} + \frac{5.9713}{(x - \sqrt{2})^2} + \frac{3.1107}{(x - \sqrt{2})^3} + \frac{0.1112}{x + \sqrt{2}}$$

9 Funções simbólicas

O Matlab usa objetos simbólicos para representar variáveis e expressões. Uma expressão simbólica é uma expressão que contém objetos simbólicos. Um objeto simbólico é uma estrutura de dados do tipo cadeia de caracteres (string).

O processamento simbólico é mais preciso que o processamento numérico visto que as operações com valores numéricos introduzem erros de arredondamento que se acumulam em operações sucessivas. Erros de arredondamento surgem porque a precisão numérica é limitada pelo número de dígitos utilizados em cada operação.

Objetos simbólicos são criados através dos comandos `sym` e `syms`:

```
>> sym('expressao');
>> syms x y z;
```

Para representar a expressão $2k + \sqrt{5} = 1$ faz-se:

```
>> e = sym('2*k + sqrt(5) = 1');
```

Para criar uma variável simbólica f para a expressão $\frac{x-y}{x-2}$ utiliza-se:

```
>> f = (x-y)/(x-2);
```

Note que f é uma variável simbólica quando a expressão contém ao menos uma variável simbólica (definida previamente).

Também é possível converter um valor numérico $\frac{1+\sqrt{5}}{2}$ em uma constante simbólica fazendo:

```
>> c = sym('(1+sqrt(5))/2');
```

A variável simbólica c guarda a expressão, e não o seu resultado. A constante simbólica c pode ser reconvertida em um valor numérico a partir de:

```
>> double(c);
```

A função `ezplot` pode ser usada no seguinte formato `ezplot(f,[a,b])` para criar o gráfico de $f = f(x)$ no intervalo $a < x < b$. Por exemplo, para criar o gráfico da função seno no intervalo $-2\pi < x < 2\pi$ usa-se:

```
>> ezplot('sin(x)', [-2*pi, 2*pi]);
```

Para traçar uma nova curva no mesmo gráfico, por exemplo, a função cosseno, utiliza-se o comando `hold` da seguinte forma:

```
>> hold on    % segura a figura selecionada
>> ezplot('cos(x)', [-2*pi, 2*pi]);
```

10 Exercícios Computacionais

1. Explore os comandos `cart2pol` e `pol2cart`. Escolha dois números complexos: z_1 (na forma Cartesiana) e z_2 (na forma polar).

   a. Converta-os e trace os respectivos gráficos usando `compass`.

   b. Determine $z_1 z_2$.

   c. Determine z_1/z_2. Mostre o resultado na janela de comando.

2. Determine a expressão para uma senóide exponencialmente convergente que oscila 3 vezes por segundo e cuja amplitude decresce 50% a cada 2 segundos. Trace o gráfico do sinal entre $-2 \leq t \leq 2$.

3. Trace os gráficos de

   a. $x_1(t) = \text{Re}(2e^{(-1+j2\pi)t})$

   b. $x_2(t) = \text{Im}(3 - e^{(1-j2\pi)t})$

   c. $x_3(t) = 3 - \text{Im}(e^{(1-j2\pi)t})$

4. Trace o gráfico de $x(t) = \cos(t)\text{sen}\,(20t)$. Escolha uma faixa para t.

5. Decomponha as frações parciais

   a. $F(s) = \dfrac{6(s+1)}{s(s+4.46)(s+0.13)}$

   b. $F(s) = \dfrac{s^2+2s+3}{(s+1)^2}$

   c. $F(s) = \dfrac{6(s+34)}{s(s^2+10s+34)}$

   Mostre as funções decompostas obtidas na janela de comando.

6. Trace em apenas um gráfico as funções $x_1 = \cos(t)\text{sen}\,(20t)$, $x_2 = \cos(t)$ e $x_3 = \text{sen}\,(20t)$. Considere faixa e amostragem pertinentes para a variável t. Use legenda e cores para discriminar as diferentes funções.

7. Trace o gráfico de $f(t) = -3\cos(\omega_0 t) + 4\text{sen}\,(\omega_0 t)$ para um valor de ω_0 fixo e t variável. Comente o ocorrido. Obtenha a expressão equivalente de $f(t)$, conforme visualizada no gráfico.

8. Dadas as matrizes

$$A = \begin{bmatrix} 1 & 1 & 6 \\ 5 & -2 & 1 \\ -8 & 2 & -3 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 9 \\ -5 & -1 \\ 9 & 2 \end{bmatrix}$$

   a. Determine se A e B são matrizes quadradas. Obs.: O algoritmo deve ser válido para avaliar matrizes de quaisquer dimensões.

   b. Quais elementos contêm o valor 2?

   c. Quais elementos contêm valores negativos?

9. Calcule o determinante, a inversa e o traço da matriz simbólica

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

   Obs.: Traço é a soma dos elementos da diagonal principal.

10. Defina uma função $f(x)$ simbólica de quarta ordem e calcule suas derivadas de primeira e segunda ordem. Use os comandos `diff` e `pretty`.

11. Calcule as derivadas de primeira ordem com relação à x dos elementos da matriz simbólica

$$M = \begin{bmatrix} ax & bx^2 \\ cx^3 & dy \end{bmatrix}$$

12. Dada a função $p(x) = (x^2 - 1)(x - 2)(x - 3)$. Explore os comandos `expand` e `factor`. Comente.

Dica: Os comandos `size`, `roots`, `syms`, `fprintf`, `solve`, `hold on/off`, `legend`, `atan` e `trace` podem ser úteis.

A Alguns comandos úteis em computação simbólica

- `simplify`: simplifica uma expressão simbólica
- `collect`: reescreve uma expressão como um polinômio
- `subs`: substitui uma variável por um valor
- `int`: integral (operação inversa da derivação diff)
- `limit`: limite de uma expressão
- `taylor`: expansão em série de Taylor
- `poly2sym`: converte os coeficientes de um polinômio em polinômio simbólico
- `solve`: solução simbólica de equações
- `dsolve`: solução simbólica de equações diferenciais

B Compilação de principais comandos em Matlab

```
clear        % Apaga variavels do "Workspace"
close all    % Fecha todos os graficos eventualmente abertos
clc          % Limpa janela de comando

a=1
b=3
who  % Exibe variaveis da area de trabalho

pause  % Pausa execucao do script; tecle espaco na janela de
       % comando para continuar

help linspace  % Obtem ajuda sobre comando linspace

pause

% ---- Vetores e graficos ----

x=linspace(0,100,1e4);  % Cria um vetor de 0 a 100 com 10^4
                        % elementos igualmente espacados
Nx=length(x)            % Comprove dimensao do vetor x (10^4?)
y=logspace(-5,2,1e4);   % Equivalente a 10.^linspace(-5,2,1e4)
Ny=length(y)            % Comprove dimensao do vetor y (10^4?)

pause

tic                     % Inicia o relogio
for i=1:length(x)
    zn(i)=x(i)+y(i);
end
toc                     % Exibe o tempo

tic                     % Reinicia o relogio
z=x+y;                  % z = zn
toc                     % Exibe o tempo
% Observe a diferenca de tempo
% Operacoes com vetores sao muito mais rapidas

pause

figure(1)    % Abre figura de numero 1 para plotagem
plot(x)      % Plota variavel x
plot(y)      % Plota variavel y. A variavel x continua plotada?

pause

hold on      % Retem as curvas ja plotadas para que
             % nao desaparecam ao plotar novas curvas
plot(x,'r') % Plota a variavel x em vermelho (red)
```

```
plot(z,'k') % Veja as outras cores com "help plot"
legend('y','x','z')

pause

figure(2)
titulo='Parabola'; % titulo e' uma variavel "string"
plot(x,x.^2)
xlabel('x')     % Define nome para os eixos x e y e o titulo
                % da Figura
ylabel('x^2')
title(titulo)
grid            % O que faz este comando?

pause

% ---- Salvar a figura 1 em figura1.pdf e figura1.png ----

figure(1)
print -dpdf 'figura1'
print -dpng 'figura1'

pause

figure(3)
plot3(x,y,x.*y)
grid

pause

figure(4)
subplot(2,2,1)
plot(x,2*x,'m--')

subplot(2,2,2)
plot(x,cos(0.5*x),'k.')

subplot(2,2,3)
plot(x,10.^(-x),'rx')

subplot(2,2,4)
plot(x,ones(size(x)),'o')

pause

% ---- Indexacao ----

a=x(end/2:end); % Metade final do vetor x
b=x(1:5)        % Primeiros 5 elementos
c=find(x>99.96) % indices dos elementos maiores que 99,96
```

```
d=x(c)              % Elementos maiores que 99,96
e=x([3 8 10 4])     % Elementos com indice 3,8,10 e 4

pause

w=ones(size(b)) % Vetor de uns
w=w+b
ze=zeros(1,length(w)) % Vetor de zeros
w=rand(1,length(w))   % Vetor de numeros aletorios
                      % uniformemente distribuidos
                      % entre 0 e 1

w=randn(1,length(w))  % Vetor de numeros aleatorios com
                      % distribuicao normal (Gaussiana),
                      % media 0 e variancia 1

w(1:3)=[]             % Apagamos os 3 primeiros elementos do
                      % vetor w

pause

clear

% ---- Matrizes ----

v=[1,3,5]
size(v)
w=[1,3,5]'
w=v'
size(w)

pause

A=w*v
size(A)

A(1,2)
col2=A(:,2)
size(col2)

pause

A*A
A.*A
A*inv(A)

pause

B=A^2
B=A.^2
```

```
pause

eye(5)      % Matriz identidade
randn(2,3)  % Matriz aleatoria

pause

% ---- Concatenacao de matrizes ----

B=[A zeros(size(A)); ones(size(A)) eye(size(A))]

B=[B B]

pause


% ---- Graficos de superficie ----

x=-5:0.5:5;
y=-5:0.5:5;

[X,Y]=meshgrid(x,y);

figure(5)

surf(X,Y,X.^2+Y.^2)

pause

% ---- Controle de fluxo ----

x=rand(10,1)
aux=Inf;
for i=1:length(x)
    if x(i)<= aux
        aux=x(i);
    elseif x(i)<0
        disp(x(i))
    else
        disp(-x(i))
    end
end

aux

% ---- Polinomios ----

p=[1 4 -2 3]
x=2
```

```
y=polyval(p,x) % Avalia o polinomio de coeficientes p
               % x^3 + 4x^2 - 2x + 3 no ponto x

pause

p2=[3 2 1];
pn=conv(p,p2) % Produto dos polinomios

pause

[p,r]=deconv(pn,p2) % Divisao de pn por p2 com resto r

pause

r=roots(p) % Raizes do polinomio

pause

% ---- Numeros complexos ----

imag(r)
real(r)
angle(r)
abs(r)
```