Breve Introdução ao

MATLAB®

para uso em

Vibrações Mecânicas

O QUE É O MATLAB?

O MATLAB ("MATrix LABoratory") é um **ambiente computacional** com diversos programas, que pode ser usado para a resolução de uma variedade de problemas científicos e de engenharia, tais como:

→ diferenciação e integração numérica, equações diferenciais ordinárias e parciais, ajuste de curvas, equações não lineares e otimização;

Também pode resolver muitos tipos de problemas simbolicamente.

O MATLAB é bastante conveniente para cálculo e programação, bem como para geração de gráficos.

Pode-se executar uma única declaração de cada vez, ou uma lista de declarações, reunidas num arquivo de programa ("script file").

DADOS E OPERAÇÕES ELEMENTARES

O único tipo de **dado** no MATLAB é uma **matriz de valores complexos**. Assim, escalares, vetores com elementos inteiros e matrizes com valores reais são tratados como casos especiais de matrizes complexas.

Os **símbolos** usados nas **operações aritméticas básicas** de adição, subtração, multiplicação, divisão e exponenciação são, respectivamente, os seguintes: +, –, *, / e ^ .

Em qualquer expressão, os **cálculos** são **executados da esquerda para a direita**. A ordem de prioridade, da maior para menor, é: (1) exponenciação; (2) multiplicação e divisão (iguais); (3) adição e subtração (iguais).

PRECISÃO E VARIÁVEIS

PRECISÃO

O MATLAB usa **precisão dupla** (64 bits) durante os cálculos, mas mostra os resultados na tela em formato mais reduzido. Essa característica de apresentação pode ser alterada com o uso do comando *format*.

VARIÁVEIS

O **nome de uma variável** deve começar com uma letra, seguida por letras, dígitos e “underscores”. São usados os primeiros 63 caracteres. Maiúsculas e minúsculas são distinguidas. Nomes reservados não são aceitos.

Quando o MATLAB encontra um novo nome de variável, ele automaticamente cria a variável e aloca o espaço apropriado.

CRIAÇÃO DE VETORES E MATRIZES

Antes de se realizar operações aritméticas com vetores e matrizes, tais como adição e multiplicação, esses arranjos devem ser inseridos.

Para vetores, isso é feito da seguinte forma:

VETOR LINHA

≫ A = [1 2 3]

Um **vetor linha**, com n elementos, é tratado como uma **matriz (1 x n)**, ficando seus elementos entre colchetes e separados por espaços ou vírgulas.

Se não for acrescentado um ponto e vírgula no final da linha, o MATLAB apresenta os resultados da linha na tela, após ela ter sido lida.

CRIAÇÃO DE VETORES E MATRIZES (cont.)

VETOR COLUNA

$\gg$A = [1; 2; 3] ou $\gg$A = [1 2 3]'

Um **vetor coluna**, com n elementos, é tratado como uma **matriz (n x 1)**. Seus elementos podem ser digitados em uma única linha, usando ponto e vírgula para separá-los.

Alternativamente, pode ser usado um vetor linha, com um apóstrofo no colchete da direita, que representa a operação de transposição. Assim, o vetor linha torna-se um vetor coluna.

CRIAÇÃO DE VETORES E MATRIZES (cont.)

MATRIZ

Já para inserir uma matriz, por exemplo,

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

a seguinte especificação pode ser usada:

$\gg$A = [1 2 3; 4 5 6; 7 8 9]

Observa-se que o ponto e vírgula separa as linhas da matriz inserida.

VETORES COM ESTRUTURA ESPECIAL

Em certos casos, a estrutura especial de um vetor é usada para especificá-lo de uma maneira mais simples. Por exemplo, o comando

$\gg$ A = 1:10

cria o vetor linha

$A = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10]$.

Já o comando

$\gg$ A = 2:0.5:4

cria o vetor linha

$A = [2.0\quad 2.5\quad 3.0\quad 3.5\quad 4.0]$.

MATRIZES ESPECIAIS

Algumas matrizes especiais são referenciadas por comandos particulares, como exposto na tabela abaixo.

comando	resultado	comentário
$\gg$ A = eye(3)	$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	Matriz identidade de ordem 3.
$\gg$ A = ones(3)	$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	Matriz quadrada de ordem 3, com todos os elementos iguais a 1.
$\gg$ A = zeros(2,3)	$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	Matriz retangular 2x3, com todos os elementos iguais a 0.

OPERAÇÕES ELEMENTARES COM MATRIZES

Matrizes podem somadas, subtraídas e multiplicadas pelo uso dos sinais

$+, \; - \;$ e $\; * \;$,

respectivamente.

Por exemplo, para somar as matrizes A e B, de modo a obter a matriz C, usa-se o comando

$\gg$ C = A + B

Já a matriz D, que decorre da multiplicação de A por B, é obtida por

$\gg$ D = A * B

FUNÇÕES MATLAB

O MATLAB tem um grande número de funções embutidas, tais como:

sqrt(x) – raiz quadrada de x

sin(x) – seno de x

cos(x) – cosseno de x

tan(x) – tangente de x

log10(x) – logaritmo de x na base 10

ln(x) – logaritmo neperiano de x

exp(x) – função exponencial de x

Essas e outras funções podem ser combinadas, como se verá abaixo.

FUNÇÕES MATLAB (cont.)

Para gerar um vetor y que contenha 11 valores associados com a função

$$y = e^{-2x}\cos x\ ,$$

com

$$x = 0,\ 0.1,\ 0.2,\ ...,\ 1.0\ ,$$

executam-se os seguintes comandos:

```
≫ x = 0:0.1:1
≫ y = exp(-2*x).*cos(x)
```

O **ponto antes do sinal de multiplicação** permite que os valores das funções exponencial e cosseno sejam multiplicados em correspondência.

NÚMEROS COMPLEXOS

O MATLAB usa a **álgebra de números complexos** automaticamente.

Os símbolos i e j podem ser usados para representar a parte imaginária, sem necessidade de um asterisco entre eles e um número. Por exemplo,

≫ z = 1 – 3i

gera a variável complexa z, cuja a parte real é 1 e a parte imaginária é –3.

O **módulo** e o **argumento** (em radianos) **de** um certo **número complexo** z, tal que z = a + ib, podem ser determinados da seguinte forma:

≫ mz = abs(z) (módulo de z = $\sqrt{a^2 + b^2}$)

≫ az = angle(z) (argumento de z= $\arctan(b/a)$)

ARQUIVOS M

Como visto até agora, o MATLAB pode ser usado em **modo interativo**, com a digitação imediata de cada comando pelo teclado. Nesse modo, ele executa as operações como se fosse uma calculadora mais avançada.

Porém, se o mesmo conjunto de comandos tiver que ser repetido várias vezes, com valores diferentes dos parâmetros de entrada, executar um programa será mais rápido e eficiente.

Um **programa MATLAB** é uma sequência de comandos de interesse, escritas em um módulo próprio e executadas como um único bloco.

Um arquivo de programa é dito “script file” ou “m-file” (**arquivo m**).

ARQUIVOS M (cont.)

Todo **arquivo de programa** deve ter um **nome**, que deverá terminar com a extensão .m (um ponto seguido da letra m).

Um arquivo m típico, denominado fibo.m, é dado apresentado abaixo.

```
% m-file to compute Fibonacci numbers
f=[1 1];
i=1;
while f(i) + f(i+1) < 1000
   f(i+2)=f(i)+f(i+1);
   i=i+1;
end
f
```

ARQUIVOS M (cont.)

Um **arquivo m** também pode ser usado para conter **funções** (**sub-rotinas**), que serão utilizadas na linha de comando ou por outros programas.

Nesse caso, a geração do arquivo se dá de forma independente, mas a execução só ocorre via linha de comando, ou em parte específica do programa principal de interesse (**arquivo de função não é executável!**).

Por exemplo, a solução de uma equação quadrática

$$ax^2 + bx + c = 0$$

pode ser obtida, de forma geral, com a **criação de** uma **função** associada **e** a **execução de** um **programa** correspondente, tal como exposto abaixo.

ARQUIVOS M (cont.)

a) Função raizes_quadraticas (sub-rotina)

```
function [x1, x2]=raizes_quadraticas(a,b,c)
% função para encontrar as raízes de uma equação quadrática
dis=b^2-4*a*c;  % dis = discriminante
if (dis < 0.0)
  x1=(-b+i*sqrt(-dis))/(2*a);
  x2=(-b-i*sqrt(-dis))/(2*a);
  disp('As raízes são complexas conjugadas.');
elseif (abs(dis) < 1e-8)  % dis = 0.0
  x1=-b/(2*a);    x2=-b/(2*a);
  disp('As raízes são idênticas.');
```

```
else (dis > 0.0)
  x1=(-b+sqrt(dis))/(2*a);
  x2=(-b-sqrt(dis))/(2*a);
  disp('As raízes são reais e distintas.');
end
```

b) Programa equacao_quadratica (principal)

```
clc; clearvars; close all;

[x1,x2]=raizes_quadraticas(2,2,1)
```

Como, pelo programa, a = 2, b = 2 e c = 1, o resultado será

As raízes são complexas conjugadas.

$x1 = -0.5000 + 0.5000i \qquad x2 = -0.5000 - 0.5000i$

GERAÇÃO DE GRÁFICOS

Para se gerar um gráfico x-y em MATLAB, **define-se um vetor x**, de valores da variável independente x (abscissa), **e um vetor y**, de valores da variável dependente y (ordenada), correspondentes aos valores de x.

Na sequência, o gráfico x-y pode ser gerado com o comando plot(x,y).

Para gerar, por exemplo, o gráfico da função $y = x^2 + 1$, na faixa $0 \le x \le 3$, os seguintes comandos podem ser utilizados:

```
≫x = 0:0.2:3;

≫y = x.^2 + 1;

≫ plot(x,y)
```

GERAÇÃO DE GRÁFICOS (cont.)

Nas duas primeiras linhas acima, são gerados os vetores x e y (com incrementos de 0,2 para x), enquanto que na terceira linha é gerado o gráfico (conectando os pontos por linhas retas, que é a opção padrão).

Ao adicionar as cinco linhas abaixo, inserem-se os eixos x e y e uma grade.

```
≫hold on
≫x1 = [0 3]; y1 = [0 0]; x2 = [0 0]; y2 = [0 10];
≫plot(x1,y1, '-b',x2,y2,'-b')
≫grid on
≫hold off
```

SOLUÇÃO DE EQUAÇÕES DIFERENCIAIS ORDINÁRIAS

Para a solução de um sistema de equações diferenciais ordinárias de primeira ordem, o MATLAB tem diversas funções construídas com base nos **métodos de Runge-Kutta**.

A função MATLAB **ode23** implementa uma combinação de métodos de Runge-Kutta de segunda e terceira ordens, enquanto a função **ode45** tem como base uma combinação de métodos de quarta e quinta ordens.

Para usar essas funções na resolução de uma equação diferencial ordinária de ordem n, a equação deve ser previamente convertida para um sistema de n equações diferenciais ordinárias, todas de primeira ordem.

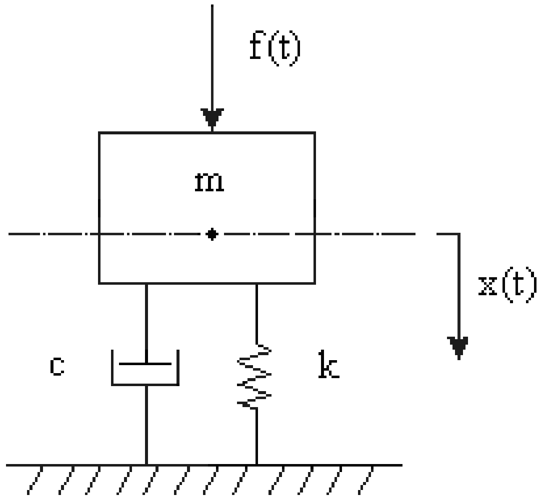
SOLUÇÃO DE EQUAÇÕES DIFERENCIAIS (cont.)

Como exemplo, considere a resolução do problema de valor inicial composto pela **equação diferencial ordinária** (EDO)

$$\boxed{m\ddot{x}(t)+c\dot{x}(t)+kx(t)=f(t)} \rightarrow \boxed{\ddot{x}(t)+\frac{c}{m}\dot{x}(t)+\frac{k}{m}x(t)=\frac{1}{m}f(t)},$$

junto com as **condições iniciais** $\boxed{x(0)=x_0 \text{ e } \dot{x}(0)=v_0}$.

Essa equação descreve o comportamento dinâmico de um **sistema mecânico linear com um grau de liberdade** e amortecimento viscoso, como ilustrado na figura ao lado.



SOLUÇÃO DE EQUAÇÕES DIFERENCIAIS (cont.)

Para resolver esse problema numericamente, transforma-se a EDO num sistema de 2 equações de ordem 1, pela definição das variáveis de estado

$$x_1(t) = x(t) \quad \text{e} \quad x_2(t) = \dot{x}(t) \ ,$$

onde $x_1(t)$ representa o deslocamento e $x_2(t)$ a velocidade.

Por essas definições, e omitindo a dependência no tempo, tem-se que

$$\boxed{\dot{x}_1 = x_2 \quad \text{e} \quad \dot{x}_2 = -\frac{c}{m}x_2 - \frac{k}{m}x_1 + \frac{1}{m}f} \text{, com } \boxed{x_1(0) = x_0 \text{ e } x_2(0) = v_0} \ .$$

Para um sistema em que m = 100 kg, c = 100 kg/s e k = 2000 N/m, sendo x_0 = 0,01 m, v_0 = 0,1 m/s e f(t) = 150sen(6t), o programa e a função correspondentes em MATLAB são apresentados abaixo.

SOLUÇÃO DE EQUAÇÕES DIFERENCIAIS (cont.)

a) Programa solumgdl (principal)

```
% solução numérica de sistema com 1 gdl sob excitação harmônica
clc; clearvars; close all

ts=[0 15];
ci=[0.01  0.1];

[t,x]=ode45('seh',ts,ci);

plot(t,x(:,1),'r-',t,x(:,2),'b-')
xlabel('tempo (s)')
ylabel('deslocamento(m), velocidade(m/s)')
legend('deslocamento','velocidade')
grid on
```

SOLUÇÃO DE EQUAÇÕES DIFERENCIAIS (cont.)

b) Função seh (sub-rotina, não executável)

```
function f=seh(t,x)
% sistema com 1 gdl sob excitação harmônica, ci's não nulas

m=100; c=100; k=2000; % sistema
f0=150; wf=6; % força

f=[x(2); -(c/m)*x(2)-(k/m)*x(1)+(1/m)*f0*sin(wf*t)];
```

A função nomeada seh contém o sistema de equações de primeira ordem, correspondente à equação diferencial de interesse (f contém $\dot{x}_1$ e $\dot{x}_2$).

Nela também são incluídos os parâmetros do sistema e as características de amplitude e frequência da força harmônica atuante sobre o sistema.

SOLUÇÃO DE EQUAÇÕES DIFERENCIAIS (cont.)

Após executar o programa solumgdl, que contém as sub-rotinas ode45 (nativa) e seh (criada), obtém-se o gráfico abaixo.