Guilherme Rosa Franzini
Alfredo Gay Neto
Carlos Eduardo Nigro Mazzilli
João Cyro André
Miguel Luiz Bucalem

Introdução à programação em ambientes MATLAB® e Octave

Material de apoio às disciplinas PEF 3302 e PEF 3401.

Universidade de São Paulo – USP

Escola Politécnica

São Paulo

2017

Resumo

Estas notas de aula visam dar suporte aos alunos das disciplinas de graduação PEF 3302 (Mecânica das Estruturas I) e PEF 3401 (Mecânica das Estruturas II) no que diz respeito à programação em ambientes MATLAB® e Octave. Ambas os ambientes são especializados em cálculo numérico e de grande poder. Desta forma, são ferramentas bastante empregadas em diversas universidades e centros de pesquisa.

Ressalta-se que o enfoque é na apresentação de conceitos e comandos elementares de programação, de sorte que aplicações mais complexas não estão aqui contempladas. Porém, espera-se que a base apresentada seja suficiente para que os alunos consigam buscar como resolver problemas que fogem do escopo destas notas de aula.

Este texto está dividido em quatro Capítulos. No Capítulo 1, apresenta-se uma série de comandos básicos para a solução de problemas bastante simples. Em seguida, no Capítulo 2, apresenta-se a resolução de problemas de aplicação em engenharia de estruturas. Por fim, os Capítulos 3 e 4 apresentam, respectivamente exercícios propostos de caráter geral e de caráter aplicado em engenharia de estruturas.

Antes do início do texto, salienta-se que a melhor forma de aprendizado destas ferramentas é a curiosidade e a proatividade por buscar por soluções para os problemas que são objeto de estudo dos leitores. Caso algum comando tenha sido transcrito de maneira incorreta, favor contatar a equipe de professores.

Palavras-chave: MATLAB®, Octave, introdução, aplicações em engenharia de estruturas.

Lista de ilustrações

Lista de tabelas

Sumário

1 Comandos básicos em ambientes MATLAB® e Octave

O objetivo deste apêndice é apresentar uma série de comandos básicos para familiarização com os ambientes MATLAB® e Octave[1] de programação. Cumpre ressaltar que este apêndice não detalhará os resultados obtidos, cabendo ao leitor a interpretação dos diversos comandos apresentados e também se aprofundar consultando o menu de ajuda do programa.

A Figura 1 ilustra a tela inicial do MATLAB®, enquanto a mesma tela referente ao ambiente Octave é apresentada na Figura 2. Nesta tela, três ambientes estão visíveis:

- **Janela de comando - *Command window***: Nesta tela, o usuário digita atribuição de variáveis e pode realizar operações matemáticas diversas.
- **Pasta de trabalho - *Current folder***: Antes de iniciar a programação, o usuário deve alterar a pasta onde o programa salvará os diversos arquivos. Isto pode ser feito por meio da barra de endereços. A pasta de trabalho exibe todos os arquivos que estão salvos na pasta indicada na barra de endereços.
- **Espaço de trabalho - *Workspace***: Aqui o programa mostrará todas as variáveis ativas. É possível conhecer o valor dessas variáveis por meio do espaço de trabalho. Uma outra forma de verificar o valor de uma variável consiste em digitar seu nome na janela de comando.

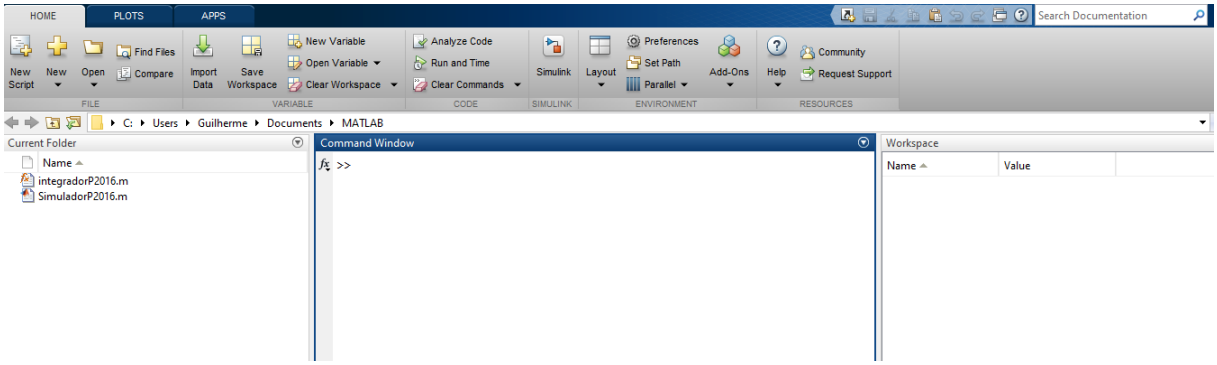

Figura 1 – Tela inicial do software MATLAB®.

No canto superior esquerdo, existe o comando *New script*. Este comando abrirá uma outra aba para que um *script* (isto é, um conjunto de comandos seja programado e, então,

[1] O Octave é um *software* gratuito e que pode ser obtido neste *link*.

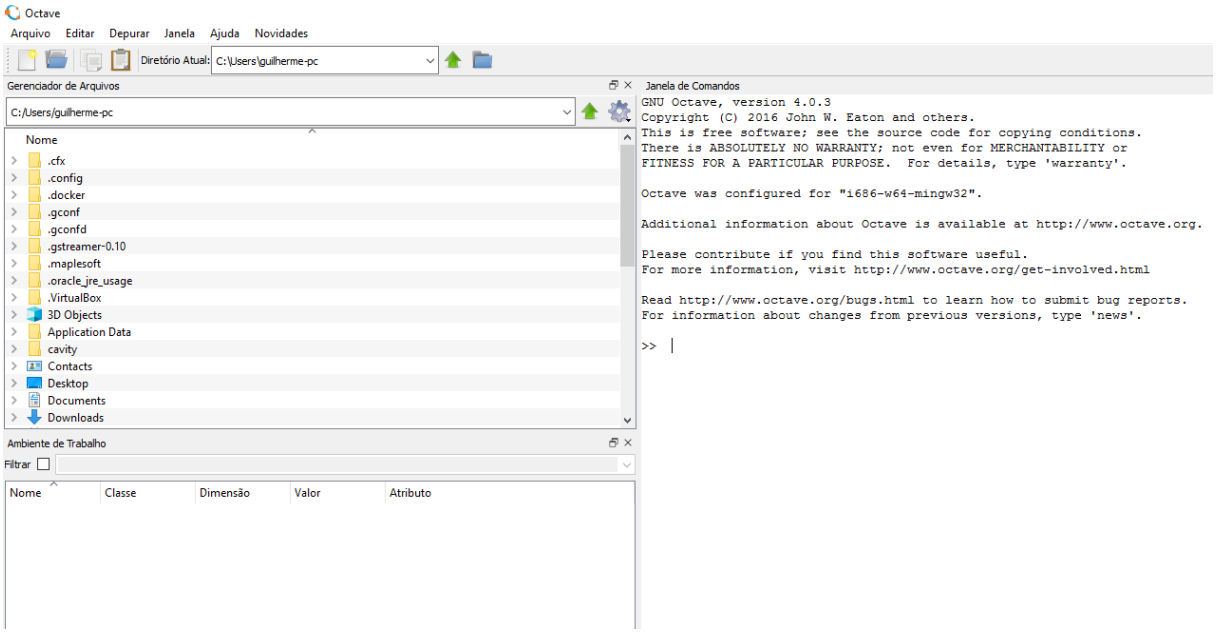

Figura 2 – Tela inicial do software Octave.

executado de uma única vez). Na maior parte das vezes, é interessante a programação de um *script* para executar as tarefas desejadas como, por exemplo, a integração numérica de uma equação diferencial ordinária. Uma vez o *script* finalizado, o usuário deve salvar o arquivo, produzindo uma *m-file* (arquivo com extensão .m) e executá-lo. Isto pode ser feito através do ícone destacado na Figura 3.

1.1 Comandos básicos

Aqui, apresenta-se uma série de comandos básicos que o usuário deverá testar na área de trabalho e compreender o resultado obtido. Por questão de organização, tais comandos estão ilustrados nas Tabelas 1 a 4. Recomenda-se que, uma vez compreendidos os comandos de uma dada tabela, o usuário limpe a área de trabalho, apague todas as variáveis e feche todas as figuras.

Antes de iniciar os testes, o usuário deve atentar para as seguintes observações:

- O separador decimal deve ser ponto (.) e não vírgula (,);
- Evite nomear arquivos e variáveis com caracteres como ç, acentos e espaços;
- Evite nomear arquivos e variáveis com primeiro caractere numérico;
- Alguns comandos foram propositalmente deixados incorretos. Cabe ao usuário a compreensão do erro;

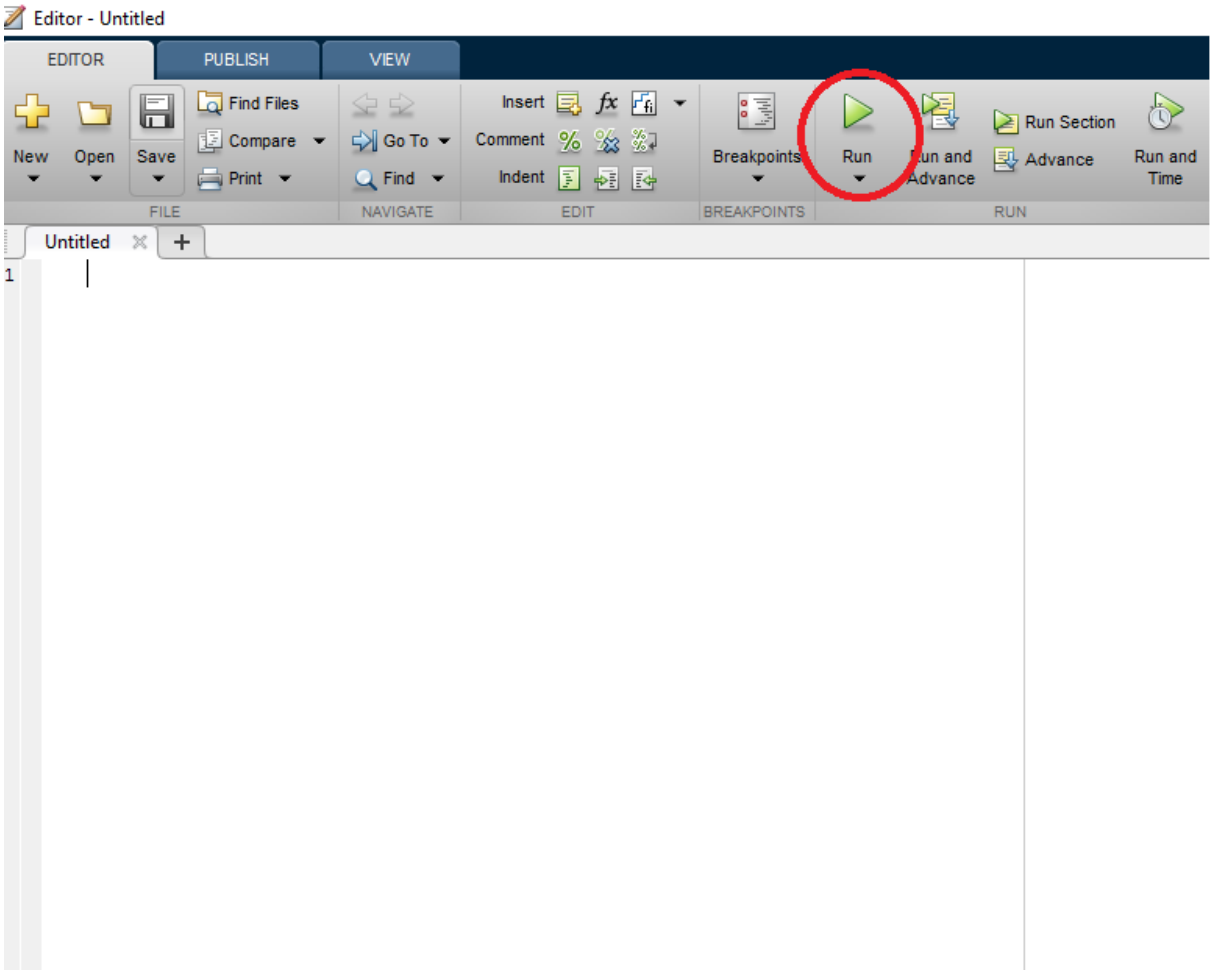

Figura 3 – Tela de programação de um *script* em ambiente MATLAB®.

- Tenha o hábito de comentar seu código. Isto pode ser feito colocando um % e escrevendo seu comentário logo em seguida.

O MATLAB®, assim como o Octave, conta com dois formatos de ajuda ao usuário. Para uso do primeiro formato, o usuário deve digitar $help < NomedaFuncao >$ na janela de comando. Por exemplo, digitando *help plot*, o programa oferece uma descrição da função *plot*, além de alguns exemplos de uso. O segundo tipo de ajuda, mais completo, pode ser acessado por meio da barra de tarefas, localizada na porção superior da tela.

A Tabela 1 apresenta alguns comandos elementares de manipulação de variáveis, ilustrado operações triviais. Já a Tabela 2 também apresenta comandos elementares, porém focados na manipulação de vetores e matrizes. Note que o MATLAB e o Octave consideram a primeira posição em vetores como o índice 1 e não 0.

O usuário também poderá usar comandos como *while*, *if/else* e *for*, já bastante conhecidos de outras linguagens de programação. No entanto estes comandos sempre devem ser finalizados com um *end*.

Tabela 1 – Comandos básicos de manipulação de variáveis.

Comando	Resultado
$a = 1$	
$b = 3;$	
$c = a * b$	
$d = a \wedge b + c;$	
$e = a/b$	
$f = a/b + c$	
$g = a/(b + c)$	
$h = 3 \wedge c$	
$k = exp(1)$	
m=sqrt(2)	
rand	
figure	
clc	
clear all	
close all	

A Tabela 3 apresenta outros comandos de manipulação de vetores e matrizes, porém em um nível de profundidade maior do que aqueles apresentados na Tabela 2. Já a Tabela 4 ilustra o uso das diversas funções trigonométricas e de manipulação de números complexos disponíveis. É importante ressaltar que o uso das funções trigonométricas exige que os ângulos sejam fornecidos em rad.

Tanto o MATLAB® quanto o Octave são capazes de manipular arquivos-texto. A Tabela 5 ilustra alguns comandos necessários para salvar resultados de interesse em um arquivo-texto e também como fazer sua leitura.

Também é interessante exibir na forma gráfica os resultados obtidos. Para tanto, considere os comandos apresentados na Tabela 6. Tais comandos necessitam do arquivo salvo por ocasião dos estudos realizados na Tabela 5. Note que o usuário deve saber, *a priori*, a ordenação das colunas da matriz com os resultados de interesse.

1.2 Construindo funções

Conforme visto no exemplo anterior, caso fosse necessário gerar um grande número de séries temporais, o usuário deveria repetir exaustivamente o comando de criação de séries temporais. Visando facilitar esta tarefa repetitiva, é possível construir funções. Uma forma bastante fácil de construir funções é acionar a criação de um novo *script* e iniciar com a seguinte sequência de comando:

$$function[saidas] = NomedaFuncao(entradas)$$

Tabela 2 – Comandos básicos de manipulação de vetores e matrizes.

Comando	Resultado
$a = 1$	
$b = 3;$	
$v1 = [a \quad b]$	
$v2 = [a; b]$	
$v3 = v1'$	
$v4 = v1 * v2$	
$v5 = 34 + v1$	
$v2 * v1$	
$v6 = linspace(0, 10)$	
$v7 = linspace(0, 10, 50)$	
$v8 = [0 : 0.01 : 5];$	
$N = length(v8)$	
$eye(3, 3)$	
$M1 = [1 \quad 2 \quad 3; 6 \quad 7 \quad 8; 1 \quad -14 \quad 16]$	
$M2 = [v1; v1]$	
$M3 = 4 + M2$	
$M4 = 5 * M2$	
$M5 = M2. \wedge 2$	
$M6 = inv(M1)$	
det(M1)	

O termo *saidas* indica qual(is) variável(is) a função irá devolver. Já o termo *entradas* indica qual(is) variável(is) são necessárias para que a *saidas* seja calculado.

Antes de usar a função, você deverá salvar o arquivo. É fundamental que o nome do arquivo .m seja exatamente igual ao nome da função criada. Para usar sua função dentro de um *script* principal, basta digitar $[saidas] = NomedaFuncao(entradas)$.

Um último comentário deve ser feito. A menos que você declare uma determinada variável como global ou como uma saída, variáveis internas às funções não são ecoadas na janela de comandos e no espaço de trabalho.

1.3 Construção de gráficos tridimensionais

É possível utilizar o MATLAB® e Octave para gerar gráficos tridimensionais. Vejamos algumas funções.

Além de curvas no espaço, é possível também construir superfícies dadas como funções de duas variáveis. Uma forma de construir tais curvas é utilizar o comando *meshgrid*.

Tabela 3 – Outros comandos de manipulação de vetores e matrizes.

Comando	Resultado
$x = linspace(0, 10)$	
$y = x \wedge 2$	
$y = x. \wedge 2$	
$int = trapz(x, y)$	
$z = y + 45$	
$w = 12 * y$	
$std(w)$	
$mean(w)$	
$abs(w)$	
$max(w)$	
$[valor, pos] = max(w)$	
$min(w)$	
$aux = find(x > 5)$	
$aux2 = find(x > 2 \quad \& \quad x < 8)$	
$aux3 = find(x > 2 \quad \mid \quad x < 8)$	
$x(1)$	
$x(2)$	
$x(3 : 10)$	
$M = magic(5)$	
$M(1, 1)$	
$M(3, 2)$	
$M(2 : 4, 1 : 3)$	
$M(:, 1)$	
$M(:, 3 : 5)$	

Desejamos, por exemplo, construir o gráfico da função:

$$f(x, y) = \frac{\sin(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}; \quad -8 < x < 8, -8 < y < 8. \tag{1.1}$$

A Tabela 8 apresenta uma série de comandos necessários para a visualização desta função, bem como outras formas representativas que podem ser de interesse do leitor.

Uma outra forma de exibir funções de duas variáveis consiste em usar um gráfico de cores. Suponha que desejemos representar a função dada pela Equação 1.1 por meio de um gráfico de cores.

O comando *pcolor* é ideal para representar os valores de uma matriz por meio de cores. Note que sua sintaxe tem como dois primeiros parâmetros os vetores e um terceiro que é a matriz a ser representada. Neste exemplo, os elementos das matrizes estão relacionados a elementos de dois vetores.

Este último exemplo introduz uma série de novos conceitos quando comparados com os comandos até aqui apresentados. Em primeiro lugar, nota-se o uso do comando

Tabela 4 – Uso de funções trigonométricas e de números complexos.

Comando	Resultado
pi	
$x = linspace(0, 2 * pi)$	
$y = cos(x)$	
$z = sin(x)$	
$w = tan(x)$	
help atan2	
help asin	
$y. \wedge 2 + z. \wedge 2$	
i	
j	
$exp(i * x)$	
$a = 1;$	
$b = 4;$	
$g = a + i * b$	
$modg = abs(g)$	
$argg = angle(g)$	

Tabela 5 – Manipulação de arquivos-texto.

Comando	Resultado
$x = linspace(0, 2 * pi)$	
$y = cos(x)$	
$z = sin(x)$	
$w = tan(x)$	
$Matriz = [x' \quad y' \quad z' \quad w'];$	
save Resultados.txt Matriz -ascii	
clc	
clear all	
close all	
load Resultados.txt	

for, bastante comum em programação. Cada comando *for* deve ser encerrado por um comando *end*. As variáveis contadoras *contx* e *conty* variam da primeira posição dos vetores associados até a última. Note que, diferente de outras linguagens de programação, o MATLAB® utiliza 1 para a primeira posição do vetor.

Um outro comando bastante útil apresentado é o *set*. Este comando está associado a trocar alguma propriedade de um elemento. No caso, foi definida uma variável h como o resultado da função *pcolor*. Em seguida, foi trocada da variável h a propriedade *edgecolor* para *none*. Para que você tenha acesso a todas as propriedades de um elemento, digite *get(NomedoElemento)*.

Tabela 6 – Comandos para geração de gráficos bidimensionais.

Comando	Resultado

load Resultados.txt
x=Resultados(:,1);
y=Resultados(:,2);
z=Resultados(:,3);
w=Resultados(:,4);
Fig1=figure;
close(Fig1);
Fig2=figure('units','normalized','position',[.1 .1 .8 .8],'color','w');
plot(x,y,'k')
hold on
plot(x,z,'r')
plot(x,w,'g')
xlabel('x')
ylabel('funcao')
legend('y','z','w')
title('Resultados graficos')
set(gca,'ylim',[-1.2 1.2])
set(gca,'fontsize',18)	

Tabela 7 – Primeira sequência de comandos para geração de gráficos bidimensionais.

Comando	Resultado

t=linspace(0,2*pi);
x=cos(t);
y=sin(t);
z=t;
plot3(x,y,z,'k');
xlabel('x');
ylabel('y');
zlabel('z');	

Tabela 8 – Sequência de comandos para superfícies dadas por funções de duas variáveis.

Comando	Resultado
$range = linspace(-8, 8, 41);$ $[X, Y] = meshgrid(range, range)$ $Z = sin(sqrt(X. \wedge 2 + Y. \wedge 2))./sqrt(X. \wedge 2 + Y. \wedge 2);$ $surf(X, Y, Z);$ $surface(X, Y, Z);$ $waterfall(X, Y, Z);$ $contour(X, Y, Z);$	

Tabela 9 – Sequência de comandos para produzir gráficos de cores para funções de duas variáveis.

Comando	Resultado
$vetorx = linspace(-8, 8, 41);$	
$vetory = linspace(-8, 8, 41);$	
$forcontx = 1 : length(vetorx);$	
$forconty = 1 : length(vetory);$	
$xcalc = vetorx(contx);$	
$ycalc = vetory(conty);$	
$Matriz(contx, conty) = sin(sqrt(xcalc^2 + ycalc^2))/sqrt(xcalc^2 + ycalc^2);$	
end	
end	
$pcolor(vetorx, vetory, Matriz);$	
$h = pcolor(vetorx, vetory, Matriz);$	
$set(h,' edgecolor',' none');$	
colorbar;	

1.4 Integração de equações diferenciais ordinárias

Tanto o MATLAB® quanto o Octave são ferramentas bastante poderosas para a integração numérica de equações diferenciais ordinárias lineares ou não-lineares. Estas classes de equações são encontradas em diversos problemas de engenharia, abrangendo desde tópicos relativos à engenharia civil como, por exemplo, dinâmica de estruturas até temas atinentes à engenharia elétrica (análise de circuitos elétricos). Esta Seção apresentará uma série de equações diferenciais resolvidas numericamente.

Inicialmente, considera-se o problema de vibrações forçadas e amortecidas, governado pela Equação 1.2 e sujeita às condições iniciais indicadas:

$$\ddot{u} + 2\zeta\omega\dot{u} + \omega^2 u = \frac{F_0}{m}\cos(\bar{\omega}t), u(0) = 0.1, \dot{u}(0) = 0 \tag{1.2}$$

Aqui, será descrito o uso da função *ode45* do MATLAB®, que utiliza o Método de Runge-Kutta para integração numérica. Em ambiente Octave, a mesma sequência de comandos deve ser aplicada, porém substituindo o comando *ode45* para *lsode*. Além disso, a sintaxe da função *lsode* exige que a variável de integração (no caso, o tempo) seja declarada após a variável que representa a incógnita.

Em ambos os ambientes de programação, é necessário transformar a Equação 1.1 (equação de segunda ordem) em um sistema de equações de primeira ordem. Isto é feito por meio da seguinte mudança de variáveis $x_1 = u$ e $x_2 = \dot{u}$. Esta mudança permite escrever a Equação 1.2 na seguinte forma:

$$\dot{x_1} = \dot{u} = x_2 \tag{1.3}$$

$$\dot{x_2} = \ddot{u} = -2\zeta\dot{u} - \omega^2 u = -2\zeta x_2 - \omega^2 x_1 + \frac{F_0}{m}\cos(\bar{\omega}t) \tag{1.4}$$

ou, na forma matricial, por:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b} \tag{1.5}$$

$$\mathbf{x} = \{x_1 \quad x_2\}^t \tag{1.6}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & -2\zeta\omega \end{bmatrix} \tag{1.7}$$

$$\mathbf{b} = \begin{bmatrix} 0 \\ \frac{F_0}{m}\cos(\bar{\omega}t) \end{bmatrix} \tag{1.8}$$

Para a integração numérica da Equação 1.2, serão construídos um *script* principal e também uma função anônima. O *script* principal terá a seguinte sequência de comandos:

Tabela 10 – Sequência de comandos para a integração numérica de uma equação diferencial ordinária.

Comando	Resultado

clc;
clear all;
close all;
zeta=0.01;
w=2;
m=1;
F0=1;
wbarra=1.2*w;
T0=0;
Tf=30;
dt=0.01;
u0=0.1;
up0=0;
$t = [T0 : 0.01 : Tf];$
$x0 = [u0; up0];$
$f = @(t, x) \quad Integrador(t, x, zeta, w, F0, m, wbarra);$
$[tsim, xsim] = ode45(f, t, x0);$
$u = xsim(:, 1);$
$up = xsim(:, 2);$	

A Tabela 10 apresenta os comandos a serem utilizados no ambiente MATLAB®. Para o uso em Octave, o usuário deverá utilizar $f = @(x, t) \quad Integrador(t, x, zeta, w, F0, m, wbarra);$ e $[xsim] = lsode(f, x0, t);$ ao invés de algumas das linhas mostradas na referida Tabela. Já a função[2] será dada pela sequência apresentada na Tabela 11

Tabela 11 – Sequência de comandos para a geração de uma função utilizada na rotina de integração numérica.

Comando	Resultado
$function \quad xp = Integrador(t, x, zeta, w, F0, m, wbarra);$ $A = [0 \quad 1; -w^2 \quad -2 * zeta * w];$ $b = [0; F0/m * cos(wbarra * t)];$ xp=A*x+b;	

Note que o leitor pode, internamente à função, definir o deslocamento u e a velocidade $\dot{u}$ acessando as posições correspondentes da variável vetorial x. Este procedimento pode ser interessante na resolução de equações diferenciais ordinárias não-lineares.

[2] A segunda e a terceira linha da função podem ser substituídas pela sequência de comandos $xp(1,1) = x(2); xp(2,1) = -2*zeta*w*x(2) - w^2*x(1) + F0/m*cos(wbarra*t)$. Estas expressões correspondem exatamente à equação diferencial de primeira ordem definida na forma matricial.

Retomando os comandos apresentados na Tabela 10, a variável $tsim$ contém o vetor de tempo utilizado na integração numérica. No caso, foi exigido que a função $ode45$ utilizasse o vetor de tempo definido pelo usuário. A variável $xsim$ é, neste caso, uma matriz com duas colunas e número de linhas igual ao número de posições do vetor $tsim$. A primeira coluna contém a série temporal referente à primeira posição do vetor x (aqui, definido como o deslocamento u). Já a segunda coluna contém a variável correspondente à segunda posição do vetor x, no caso a velocidade $\dot{u}$.

2 Exercícios resolvidos - aplicações em engenharia de estruturas

2.1 Exercício 1

Considere uma viga bi-apoiada de comprimento 1m e seção transversal retangular de dimensões $b = h = 10$mm, conforme apresentado na Figura 4. Esta viga é feita em aço ($E = 210$GPa) e está submetida a um carregamento uniforme $q = 1000$N/m. Construa uma rotina que calcule numericamente e trace os diagramas de força cortante $V(x)$ e de momento fletor $M(x)$. Use a convenção dos sinais comumente adotada. Usando o Teorema dos Esforços Virtuais, construa o diagrama de momento fletor virtual adequado δM e calcule numericamente o deslocamento no meio do vão. Compare com a solução analítica $w(L/2) = 5qL^4/384EI$. Os conceitos acerca do Teorema dos Esforços Virtuais podem ser encontrados em AndrÉ et al. (2011).

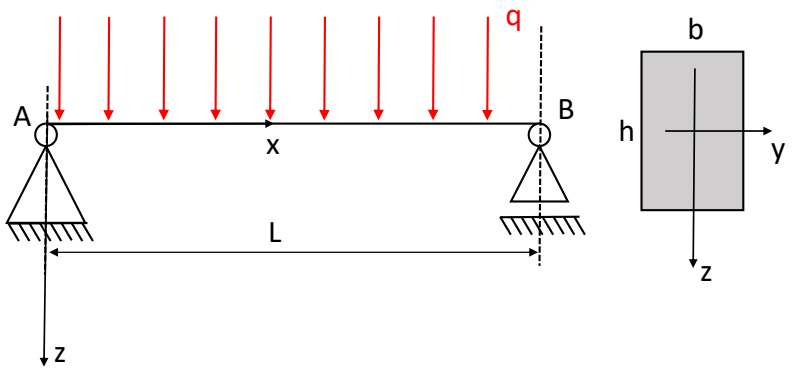

Figura 4 – Esquema ilustrativo para o exemplo 1. Barra bi-apoiada, submetida a carregamento uniformemente distribuído.

```
clc
clear all
close all

L=1;%m
q=1000;%N/m
b=10; %mm
h=10; %mm
E=210e9;
I=(b*h^3/12)*10^(-12);
EI=E*I; %Nm^2
vetorx=linspace(0,L); %vetor com posições ao longo da barra
vetorq=q*ones(1,length(vetorx)); % carregamento q(x)
Va=q*L/2; %força cortante em A, de coordenada x=0
Ma=0;


vetorV(1)=Va;
for cont=2:length(vetorx)
vetorV(cont)=-trapz(vetorx(1:cont),vetorq(1:cont))+Va;
end
vetorM(1)=Ma;
for cont=2:length(vetorx)
vetorM(cont)=trapz(vetorx(1:cont),vetorV(1:cont))+Ma;
end

Fig=figure('units','normalized','position',[.1 .1 .8 .8],'color','w');
subplot(3,1,1)
plot(vetorx,vetorq,'k','linewidth',1.5)
xlabel('x[m]','fontsize',18)
ylabel('q[N/m]','fontsize',18)
set(gca,'fontsize',16)

subplot(3,1,2)
plot(vetorx,vetorV,'k','linewidth',1.5)
xlabel('x[m]','fontsize',18)
ylabel('V[N]','fontsize',18)
set(gca,'fontsize',16)

subplot(3,1,3)
plot(vetorx,vetorM,'k','linewidth',1.5)
xlabel('x[m]','fontsize',18)
ylabel('M[Nm]','fontsize',18)
set(gca,'fontsize',16,'ydir','reverse')

%% Cálculo do deslocamento no meio do vão via teorema dos esforços virtuais
for cont=1:length(vetorx)
if vetorx(cont)<=L/2
vetordeltaM(cont)=1/2*vetorx(cont);
else
vetordeltaM(cont)=L/4-L/2*(vetorx(cont)-L/2);
end
end

delta=trapz(vetorx,vetorM.*vetordeltaM/EI)
valor_teor=5*q*L^4/(384*EI)


Fig=figure('units','normalized','position',[.1 .1 .8 .8],'color','w');
plot(vetorx,vetordeltaM,'k','linewidth',1.5)
xlabel('x[m]','fontsize',18)
ylabel('\delta M [Nm]','fontsize',18)
set(gca,'fontsize',16,'ydir','reverse')
```

3 Exercícios propostos - temas gerais

3.1 Exercício 1

Construa um *script* que é iniciado limpando a janela de comandos, apagando todas as variáveis e fechando todas as figuras.

O seu *script* deve gerar séries temporais para a função $u(t) = Ae^{-\zeta\omega_n t}\cos(\omega_d t + \phi)$, para $0 < t < 50$ para os seguintes conjuntos de parâmetros:

- Série 1: $A = 1, \phi = 90°, \omega_n = 2\pi, \zeta = 0{,}05$ e $\omega_d = \omega_n\sqrt{1-\zeta^2}$
- Série 2: $A = 1, \phi = 90°, \omega_n = 2\pi, \zeta = 0{,}15$ e $\omega_d = \omega_n\sqrt{1-\zeta^2}$

Exiba, em um mesmo gráfico e com legendas, as duas séries temporais.

3.2 Exercício 2

Construa cinco séries temporais análogas à do exercício anterior, porém considerando três séries adicionais:

- Série 3: $A = 1, \phi = 90°, \omega_n = 2\pi, \zeta = 0{,}25$ e $\omega_d = \omega_n\sqrt{1-\zeta^2}$
- Série 4: $A = 1, \phi = 90°, \omega_n = 2\pi, \zeta = 0{,}35$ e $\omega_d = \omega_n\sqrt{1-\zeta^2}$
- Série 5: $A = 1, \phi = 90°, \omega_n = 2\pi, \zeta = 0{,}75$ e $\omega_d = \omega_n\sqrt{1-\zeta^2}$

Exiba as cinco séries em um mesmo gráfico. Não se esqueça de dar rótulos aos eixos e também de identificar as legendas corretamente.

3.3 Exercício 3

Considere a equação:

$$D = \frac{1}{\sqrt{(1-\beta^2)^2 + (2\zeta\beta)^2}} \tag{3.1}$$

Considerando o intervalo $0 < \beta < 6$, construa um *script* que mostre, em um mesmo gráfico, a função D para os mesmos valores de ζ adotados nos Exercícios 1 e 2.

3.4 Exercício 4

Considere a função dada por $f(x, y) = ax^2 + by^2$. Considerando $a = 1$ e $b = 2$, construa uma rotina que construa o gráfico de $f(x, y)$ no domínio $-1 < x < 1 \times -1 < y < 1$.

3.5 Exercício 5

Considere a Equação de van der Pol dada por $\ddot{q} + \epsilon(q^2 - 1)\dot{q} + q = 0$, sendo $\epsilon = 0{,}01$. Construa um simulador para a obtenção numérica da solução $q(t)$ para um determinado par de condições iniciais $q(0)$ e $\dot{q}(0)$. Obtenha, em um mesmo gráfico, as séries temporais $q(t)$ para as seguintes condições iniciais: $q(0) = \dot{q}(0) = 0$; $q(0) = 0.1, \dot{q}(0) = 0$ e $q(0) = 5, \dot{q}(0) = 0$.

3.6 Exercício 6

Considere a Equação de Mathieu dada por $\ddot{u} + (1 + \delta \sin(nt))u = 0$. Considerando como condições iniciais $u(0) = 0.1$ e $\dot{u}(0) = 0$ e também $\delta = 0.2$, obtenha as séries temporais para $n = 1$ e $n = 2$.

4 Exercícios propostos - aplicações em engenharia de estruturas

4.1 Exercício 1

Considere uma viga bi-apoiada, submetido ao carregamento uniforme $q = 10\text{kN/m}$. Esta viga possui seção transversal retangular de dimensões $b = 10\text{cm}$ e $h = 30\text{cm}$ e comprimento $L = 3\text{m}$. O momento fletor é dado por: $M(x) = (qx/2)(L - x)$.

Mostre o diagrama de momentos fletores ao longo da viga, bem como a máxima tensão normal de tração ao longo do comprimento. Dados $I_y = bh^3/12$ e $\sigma(z) = M(x)z/I_y$.

4.2 Exercício 2

A equação de uma estrutura modelada como um sistema de um grau de liberdade é dada por:

$$m\ddot{u} + c\dot{u} + ku = F_0 \cos(\bar{\omega}t) \tag{4.1}$$

Sabe-se que a frequência natural do sistema é $\omega = \sqrt{k/m}$ e sua taxa de amortecimento dada por $\zeta = c/2m\omega$ valem, respectivamente $2\pi\text{rad/s}$ e 0,05. Sabe-se que nenhuma força externa atua nesta estrutura. Sabendo-se que em $t = 0$ o seu deslocamento é $u = 0.1\text{m}$ e sua velocidade é nula, obtenha a série temporal de deslocamento $u(t)$

4.3 Exercício 3

Com base no mesmo exemplo anteriormente estudado, simule o problema de vibração livre condição para os seguintes valores de ζ: $\zeta = 0; 0{,}01; 0{,}10; 1; 1{,}2$. Mantenha os demais parâmetros como indicados no Exercício 2.

4.4 Exercício 4

Com base no mesmo exemplo anteriormente estudado, retome o problema forçado harmonicamente. Obtenha a série temporal de deslocamento na condição de ressonância para dois valores de taxa de amortecimento: $\zeta = 0$ e $\zeta = 0{,}05$.

Referências

ANDRÉ, J. C.; MAZZILLI, C. E. N.; BUCALEM, M. L.; CIFÚ, S. *Lições em Mecânica das Estruturas: Trabalhos virtuais e energia*. [S.l.]: Oficina de Textos, 2011.