

Tutorial GNU Octave/Matlab

Conhecimento e poder: ter paciência para alcançá-lo é fundamental.

Cássia Barbosa Teixeira

Turma A

RA 042565

Campinas, Novembro, 2005.

Sumário

1	Introdução.	2
1.1	Breve Histórico.	2
1.2	Como obtê-lo.	3
1.3	Iniciando o Octave.	3
1.4	A área de Trabalho.	3
2	Usando o Octave.	4
2.1	Operações básicas.	4
2.2	Definição de variáveis.	5
2.3	Formatação e precisão numérica.	6
2.4	Repetindo comandos anteriores.	6
2.5	Outros comandos úteis.	7
3	Recursos Gráficos	
3.1	A janela gráfica.	7
3.2	Gráficos bidimensionais.	8
3.3	Gráficos tridimensionais.	10
4	Funções.	12
4.1	Definição de funções.	12
5	Exercícios.	14
5.1	Exercícios	14
6.	Bibliografia.	24

1 Introdução

Esse tutorial apresenta os conceitos básicos do GNU Octave, uma importante ferramenta de cálculo científico que tem a vantagem de ser um software livre. Este tutorial mostra como usar o Octave em aplicações interessantes para o cálculo I, como limites, derivadas, integrais e traçado de gráficos simples. O tutorial aqui apresentado foi montado especialmente para a disciplina de Cálculo I ministrada pelo Prof^o Dr^o Márcio Rosa Imecc-Unicamp.

1.1 Breve Histórico

O Octave é um software livre, escrito por Eaton (1997) e por vários outros Colaboradores. Originalmente concebido como livro texto para estudantes de graduação de química para a resolução de equações químicas complexas; inicialmente foi escrita por James B. Rawlings da University of Wisconsin-Madison e John G. Ekerdt of the University of Texas. O Octave é um programa de linguagem aberta, logo muitas pessoas contribuem com sentenças de comando que são adicionados às versões em fase de teste, essas contribuições estão disponíveis no site da GNU Octave.

O software está disponível sob os termos da Licença Pública Geral do GNU (GPL) (Free Software Foundation, 1991).

O programa possui uma interface por linha de comandos para a solução numérica de problemas lineares e/ou não lineares e para implementar outros experimentos numéricos usando uma linguagem que é compatível com o programa comercial Matlab. O Matlab foi desenvolvido no início da década de 80 por Cleve Moler, no Departamento de Ciência da Computação da Universidade do Novo México, EUA. É um "software" interativo de alta performance voltado para o cálculo numérico. Faz análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos em ambiente fácil de usar, onde problemas e soluções são expressos somente como eles são escritos matematicamente, ao contrário da programação tradicional, os elementos básicos de informação é uma matriz que não requer dimensionamento. Além disso, as soluções dos problemas são expressas no MATLAB quase exatamente como elas são escritas matematicamente.

O Octave possui muitas ferramentas para a solução numérica de problemas comuns de álgebra linear, para a determinação de raízes de equações, polinômios e integração de equações diferenciais e equações diferenciais algébricas.

Programas como o Octave são usados freqüentemente no lugar de linguagens de programação científica como o C ou Fortran, por já trazerem embutidas muitas ferramentas numéricas e permitirem a visualização gráfica dos resultados de forma mais fácil.

1.2 Como obtê-lo

Existem muitas versões diferentes do Octave disponíveis para download:

- **Stable**

É a versão mais antiga (1991) e a mais usada, não apresenta muitos recursos.

- **Testing**

É a versão disponível atualmente com muitos recursos.

- **Development**

É a versão atualmente em teste.

Para obter qualquer uma destas versões pode-se fazer o download de um dos seguintes sites: <http://www.octave.org>, logo abaixo clique em DOWNLOAD, e escolha a versão mais adequada para a sua necessidade.

Para a instalação em sistemas Windows é preciso seguir algumas recomendações no site <http://home.tiscalinet.ch/paulsoderlind/Software/Software.html>, como compatibilidade e ferramentas.

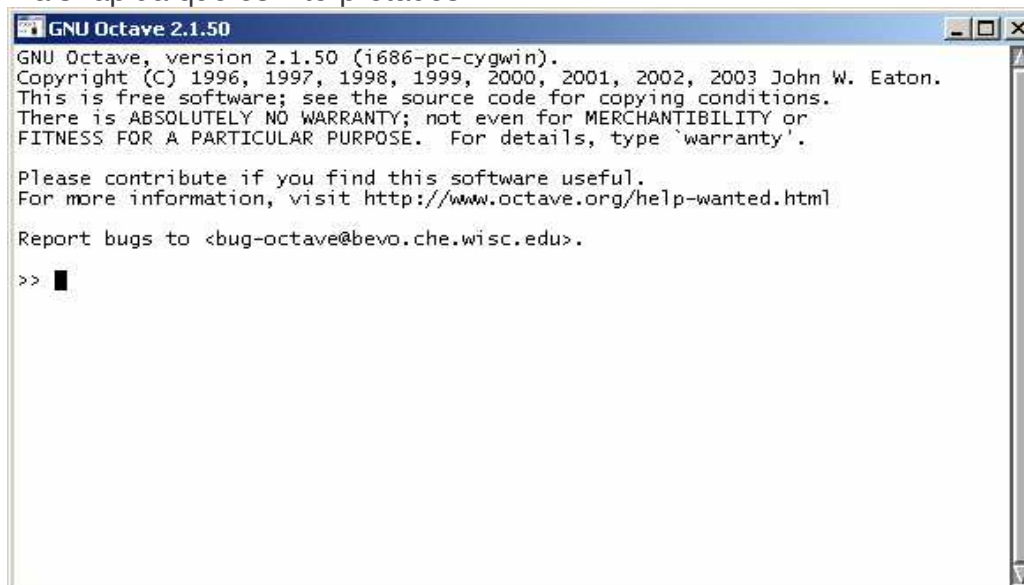
1.3 Iniciando o Octave

Nos sistemas Unix o Octave é iniciado digitando o comando octave. O programa então mostra uma mensagem inicial e um sinal de prontidão (>>), indicando que está pronto para aceitar comandos:

Para finalizar o programa, digita-se quit ou exit. Comentários podem ser colocados na área com o uso do caractere %, o comentário aparecerá em verde.

1.4 A área de trabalho

Conforme visto, o Octave tem uma interface baseada em linha de comando, onde os comandos são digitados, seguidos pela digitação da tecla Enter. O Octave é uma linguagem interpretada, o que significa que cada comando é convertido em código de máquina e executado. No caso de linguagens compiladas, o programa inteiro é convertido em código de máquina previamente, para depois ser executado. De forma geral os programas compilados são executados de forma mais rápida que os interpretados.



```
GNU Octave 2.1.50
GNU Octave, version 2.1.50 (i686-pc-cygwin).
Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003 John W. Eaton.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type `warranty`.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug-octave@bevo.che.wisc.edu>.

>> █
```

2 Usando o Octave

2.1 Operações básicas com o Octave

A forma mais simples de trabalhar com o Octave é digitar os comandos matemáticos, como em uma calculadora normal.

Exemplo: somar 2+3

```
>> 2+3
```

Teclando Enter, tem-se o resultado:

```
ans = 5
```

O valor calculado é exibido e guardado na variável ans, do inglês answer (resposta), e pode ser usado no cálculo seguinte:

```
>> ans+10
```

```
ans = 15
```

As operações aritméticas básicas são obtidas pelos seguintes operadores:

Tabela1: Operadores aritméticos.

Operador	Operação
-	subtração
*	multiplicação
/	divisão
^	potência

A ordem com que as operações são feitas é a usual, ou seja, parênteses são calculados primeiramente, seguidos de potência, multiplicação e divisão e, finalmente, adição e subtração.

O Octave possui uma série de funções matemáticas, sendo algumas delas apresentadas na Tabela 2.

Tabela2: Funções trigonométricas.

Função	Descrição	Função	Descrição
abs(x)	Módulo de x	sinh(x)	Seno hiperbólico
acos(x)	Arco cosseno é x	tan(x)	Tangente de x
cos(x)	Cosseno de (x em rad)	exp(n)	Função exponencial e ⁿ
cosh(x)	Cosseno hiperbólico	log10(x)	Logaritmo de x na base 10
Round(x)	Arredonda o valor de x		

Por exemplo, para calcular $1.2 \sin(40 + \ln(2.4)^2)$, digita-se

```
>> 1.2 * sin(40*pi/180 + log(2.4^2))
```

```
ans = 0.76618
```

Conforme se pode notar, os ângulos usados como argumentos nas funções trigonométricas devem estar em radianos. Para fazer a transformação de graus para radianos, pode-se multiplicar o valor por $\pi/180$. A operação de multiplicação deve ser sinalizada de forma explícita pelo uso do operador *, como indicado entre 1.2 e sin.

As constantes e, pi assumem valores já definidos, por exemplo:

```
>> e
```

```
e = 2.8173
```

```
>> pi
```

```
pi = 3.1416
```

Pode-se também mostrar mensagens na tela com o comando disp(""), assim:

```
>>disp("Octave para iniciantes!")  
Octave para iniciantes!
```

2.2 Definição de variáveis

É possível definir variáveis a serem usadas na sessão de trabalho, na definição dos nomes das variáveis, Octave diferencia letras maiúsculas de minúsculas. Por exemplo, *a* é diferente de *A*. Um exemplo:

```
>> a=3  
a = 3
```

Pressionando Enter o Octave confirma na tela o valor atribuído, a menos que seja colocado um caractere de ponto-e-vírgula (;) no final do comando

```
>> b=2.5;
```

Pode-se digitar vários comandos em uma mesma linha, separando-os por vírgula ou por ponto e vírgula:

```
>> a=3, b=2.5; c=7.5  
a = 3  
c = 7.5000
```

Uma vez definidas as variáveis, pode-se efetuar operações com as mesmas:

```
>> a+b  
ans = 5.5000
```

É possível definir números complexos especificando sua parte imaginária por meio da variável pré-definida *i*:

```
>> c=3+2i  
c = 3 + 2i
```

Além da variável, que como é de se esperar vale $p-1$, e da variável *ans*, que toma o resultado do último cálculo realizado, outras variáveis pré-definidas no Octave são π , que vale 3.14159... e *j*, que assim como *i* vale $p-1$. Essas variáveis podem ser redefinidas pelo usuário, mas para evitar enganos é melhor não fazê-lo. Da mesma forma, não é recomendável dar às variáveis nomes de funções, como *sin* e *cos*.

Para saber quais são as variáveis nomeadas, digita-se:

```
>> who  
*** dynamically linked functions:  
dispatch  
*** local user variables:  
a b c
```

Para remover da área de trabalho uma variável já atribuída, usa-se o comando *clear*, seguido do nome da variável, como em:

```
>> clear c
```

Para apagar todas as variáveis, digita-se:

```
>> clear all
```

2.3 Formatação e precisão numérica

O Octave normalmente exibe os números com seis algarismos significativos. Apesar de exibir-los dessa forma, o Octave trabalha internamente com uma precisão bem maior. Por isso, é bom guardar os resultados em variáveis, no lugar de digitar novamente os valores mostrados, para evitar erros nos resultados. O comando `format` permite selecionar a forma com que os algarismos são mostrados. Digitando `format long` o Octave passa a exibir os valores com pelos menos 16 algarismos significativos nas respostas dos cálculos efetuados.

```
>> c=1/3
c = 0.33333
>> format long
>> c
c = 0.3333333333333333
```

O comando `format` sem parâmetros faz o programa retornar ao seu modo normal de exibição com 5 casa decimais, o comando `format long` mostra o número com 16 casas decimais e o `format bank` com apenas 2 casas decimais:

```
>> format
>> c=1/3
c = 0.33333
```

Além dos números reais e complexos mostrados, outros números são reconhecidos e calculados pelo Octave:

Infinito (Inf) Resultado da divisão de um número por zero. Esta é uma resposta válida e pode ser usada nos cálculos e atribuída a uma variável, assim como outro número qualquer.

Not a Number (NaN) Resultado da divisão de zero por zero e outras operações que geram resultados indefinidos. Novamente, os resultados podem ser tratados como qualquer outro número, apesar dos resultados dos cálculos com seu uso gerarem sempre a resposta NaN.

Exemplo

```
>> 1 - 0.2 - 0.2 - 0.2 - 0.2 - 0.2
ans = 5.5511e-17
```

O resultado fornecido é bem pequeno, mas não chega a ser exatamente zero, que é a resposta correta. A razão para isto é que 0.2 não pode ser representado em binário usando um número finito de dígitos (em binário fica 0.0011001100...). Os programas de computador procuram representar esses números da forma mais próxima possível, mas o uso repetido dessas aproximações pode causar problemas.

2.4 Repetindo comandos anteriores

Octave mantém um registro de todos os comandos digitados durante a sessão, e pode-se usar as teclas de direção para rever os comandos precedentes. Para repetir um desses comandos, basta achá-los com essas teclas e pressionar `Enter`. Ao procurar um comando sabendo as primeiras as primeiras letras da linha digitada, pode-se pressionar essas letras e então pressionar `"`, de forma a encontrar todas as linhas começando com tais letras.

Uma vez que um comando foi recuperado, pode-se editá-lo antes de executá-lo outra vez. Pode-se usar `!` para mover o cursor através da linha, e digitar outros caracteres ou usar a tecla `Delete` para apagar os caracteres.

2.5 Outros comandos úteis

O Octave possui um sistema de ajuda integrado, que pode ser bastante útil para obter maiores informações sobre um comando ou encontrar uma função em particular. Para usar a ajuda basta digitar: `>> help nome_do_comando`

Para interromper a execução de comandos que estejam demorando muito para serem executados aplica-se Ctrl-C.

O comando `date` mostra a data atual em dia-mês-ano, o comando `clock` exibe a hora atual na forma ano, mês, dia, hora, minuto e segundo. Exemplo:

```
>>date
ans=19-nov-2005
>>clock
ans=2005 11 19 13 54 47
```

O Octave possui um comando que exibe os tópicos da ajuda:

```
>>help
```

Para obter ajuda informações sobre um tópico específico, digite `help tópico`

```
>>help plot
```

Para limpar a tela de comando basta ir ao menu **Edit** opção **Clear Command Window**, assim tudo o que já foi digitado será apagado.

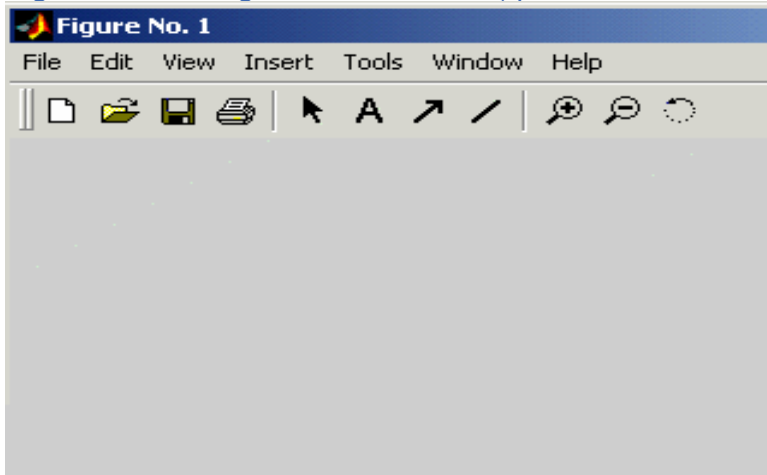
Após a digitação de um comando é conveniente usar ponto e vírgula, assim ao se pressionar a tecla enter não será mostrado o que este comando executou.

3 Recursos Gráficos

3.1 A janela gráfica

Além de um ambiente de trabalho baseado em linha de comando, quando usado em um ambiente XWindow, o Octave cria automaticamente uma janela separada para a apresentação de gráficos (Figura 1). Mesmo em ambiente texto é possível direcionar a saída gráfica para um arquivo, de forma a visualizá-lo posteriormente em outro aplicativo. Após aplicar o recurso `plot` ou outro qualquer, a janela gráfica aparecerá com, toda vez que uma modificação for feita na função o gráfico muda de aparência.

[Figura 1: Janela gráfica do MatLab \(que será usada como recurso gráfico\).](#)



3.2 Gráficos bidimensionais

O Octave pode plotar uma variedade de gráficos usando o Gnuplot e o ImageMagick para mostrar as imagens.

O comando básico para o traçado de gráficos bidimensionais é o `plot(x,y)`. Os parâmetros `x` e `y` são as coordenadas a serem traçadas. Se `x` e `y` forem um par de escalares, somente um ponto é traçado. Usando vetores, o programa irá traçar todos os pontos correspondentes a esses valores uni-los por linhas retas.

O primeiro passo para o traçado de gráficos 2D é formar uma tabela de coordenadas (`x`, `y`) da função a ser plotada. O procedimento para criar essa tabela usando vetores foi mostrado em seção anterior. Tome-se, como exemplo, a função $\cos(x)$, a ser traçada no intervalo entre 0 e 2π . Para construir um vetor `x`, com 100 valores dentro desse intervalo, pode-se usar o comando:

```
>> x=[0: 2*pi/100: 2*pi];
```

Outra opção é o uso do comando, este comando é o mais usado:

```
>> x=linspace(0, 2*pi);
```

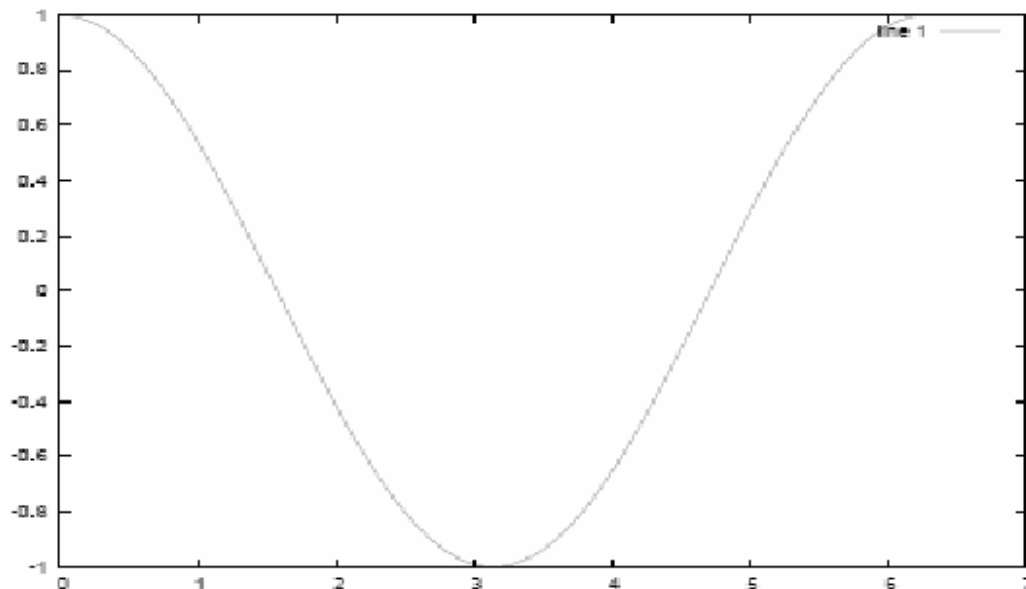
Para completar a tabela de coordenadas, determina-se o vetor `y` correspondente aos valores em `x`. Isso é feito simplesmente invocando a função com o comando:

```
>> y=cos(x);
```

Uma vez construída a tabela com as coordenadas (`x`, `y`), pode-se usar a função `plot`:

```
>> plot(x,y)
```

A Figura 2 mostra a curva obtida $y = \cos(x)$

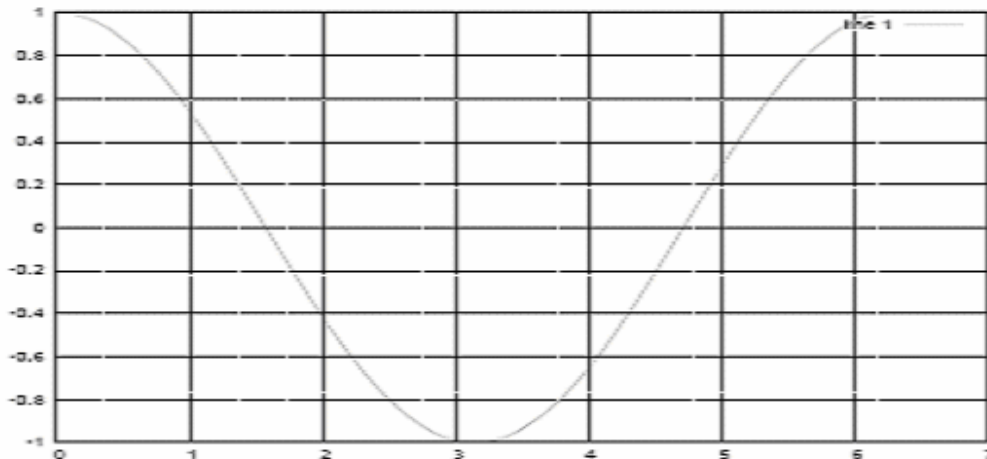


Linhas de grade podem ser adicionadas ao gráfico com o comando

```
>> grid on
```

O resultado é mostrado na Figura 3. O comando `grid off` retorna ao modo sem linhas de grade.

Figura 3: Gráfico de $y = \cos(x)$, com linhas de grade.



A aparência do gráfico pode ser modificada com os parâmetros adicionais, mostrados na Tabela 5, colocados entre aspas simples. Por exemplo, o comando `>> plot(x,y,'-kx')`

Tabela 5: Aparência do gráfico;

Símbolo	Cor	Símbolo	Marcador	Símbolo	Tipo de linha
b	azul	+	Sinal positivo	-	Linha cheia
w	branco	*	Asterisco	..	pontilhado
c	ciano	°	Diamante		
k	preto	x	Letra x		
g	verde				
r	vermelho				
m	magenta				

Traçar o gráfico com linha cheia, na cor preta, usando um “x” como marcador das coordenadas.

Os comandos `title`, `xlabel` e `ylabel` permitem escrever um título para o gráfico e um rótulo em cada um dos eixos. Deve-se passar como parâmetro para esses comandos uma seqüência de caracteres entre aspas:

```
>> title("Gráfico");
>>xlabel("x");
>> ylabel("y");
>> legend("Gráfico nº1111");
```

Para salvar o gráfico produzido em um arquivo de imagem do tipo PostScript.

Encapsulado (EPS), a seqüência de comandos é a seguinte:

Para fazer voltar para a tela, pode-se sair e entrar novamente no Octave ou executar o comando:

```
>> gset term x11
```

Em ambiente Windows, o parâmetro “x11” no comando acima deve ser substituído por “windows”. O comando `close` fecha a tela gráfica.

Para traçar gráficos de funções paramétricas, deve-se usar o comando `gset parametric`. Por exemplo, para traçar o gráfico da equação $x = \sin(3t)$ $y = \cos(5t)$, com o argumento t variando no intervalo $[-_, _]$, pode-se usar a seguinte seqüência de comandos:

```
>> gset parametric
>> t=-pi: pi/100: pi;
>> plot(sin(3*t),cos(5*t))
>> gset noparametric
```

O comando `gset` é usado novamente, para desativar o traçado de funções paramétricas.

Também podemos usar uma formatação de texto especial para símbolos como π , δ e α , existe uma seleção com mais de 75 desses símbolos. Na tabela 6 vemos os mais utilizados.

Tabela 6: Símbolos mais utilizados em cálculo.

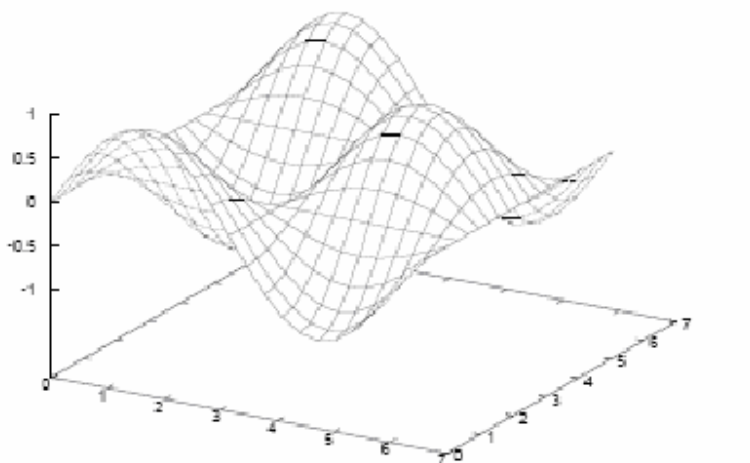
Seqüência	Símbolo	Seqüência	Símbolo
<code>\alpha</code>	α	<code>\theta</code>	θ
<code>\beta</code>	β	<code>\lambda</code>	λ
<code>\gamma</code>	γ	<code>\pi</code>	π
<code>\delta</code>	δ	<code>\int</code>	\int
<code>\epsilon</code>	ϵ	<code>\infty</code>	∞
<code>\omega</code>	ω	<code>\rho</code>	ρ

3.3 Gráficos tridimensionais (R^3)

O comando `mesh(x,y,z)` permite traçar a malha para gráficos tridimensionais, da forma $z = f(x, y)$. Por exemplo, a função $z = \cos(x)\sin(y)$ no intervalo $[0, 2\pi]$ dividido em 50 incrementos. Os comandos produzem a curva mostrada na Figura 4.

```
>> x=[0: 2*pi/50: 2*pi];
>> y=x;
>> z=cos(x)*sin(y);
>> mesh(x,y,z)
```

Figura 4: Gráfico de $z = \sin(x)\cos(x)$



Para mudar o ângulo de visão, pode-se clicar com o botão direito do mouse sobre a figura, e arrastá-la para uma nova posição.

Enquanto o comando `mesh(x,y,z)` representa o gráfico por meio de uma malha, o comando `surf(x,y,z)` representa a função tridimensional como uma superfície, adicionando à malha efeitos de cores e profundidade. Uma vez que essa função é acionada, as chamadas subsequentes à função `mesh` irão também mostrar uma superfície com os mesmos efeitos de profundidade, a não ser que o comando `clf` seja usado antes para limpar a janela gráfica, ou então o comando `close` seja usado para fechá-la.

A seqüência de comandos usados para o gráfico da Figura 4 é válida para funções do tipo $z = f(x)g(y)$. Para outros tipos de funções um outro procedimento é necessário. Considere-se, por exemplo, a função $z = (x - 3)^2 - (y - 2)^2$.

Uma seqüência de comandos para traçar o gráfico dessa equação é a seguinte:

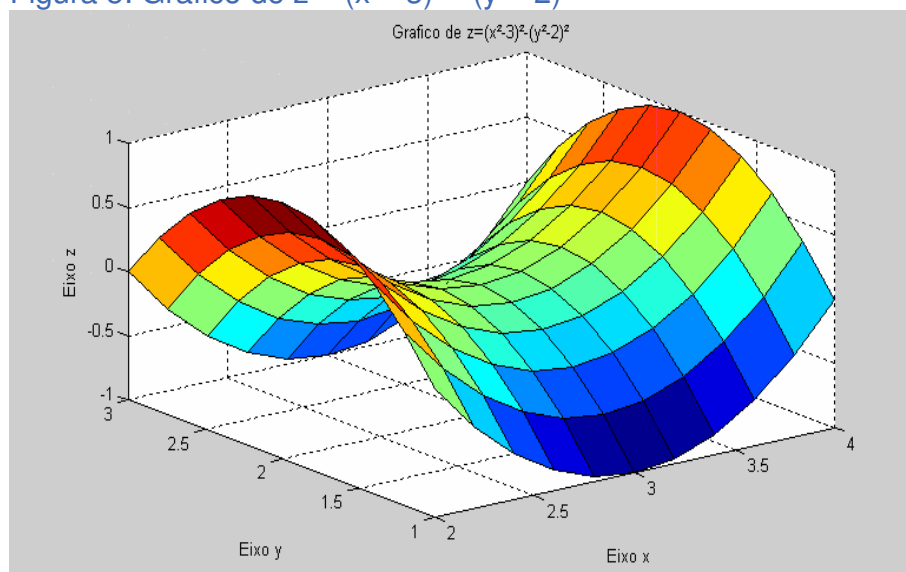
```
>> x = 2: 0.2: 4;  
>> y = 1: 0.2: 3;  
>> [xx,yy] = meshgrid(x,y);  
>> z = (xx-3).^2 - (yy-2).^2;  
>> surf(x,y,z)
```

Esses comandos produzem a forma de sela, mostrados na Figura 5.

O comando `meshgrid`, usado no exemplo acima, recebe dois vetores de coordenadas x e y e retorna duas matrizes correspondentes às coordenadas da malha.

As linhas de `xx` são cópias de x e as colunas de `yy` são cópias de y .

Figura 5: Gráfico de $z = (x - 3)^2 - (y - 2)^2$



O Octave também pode traçar gráficos em escalas especiais:

- loglog gráfico com base log nos eixos.
- semilogy gráfico com eixo x linear e eixo y logarítmico.
- semilogx gráfico com eixo y linear e eixo x logarítmico.
- Polar gráfico em coordenadas polares.

4 Funções

4.1 Definição de funções

O Octave tem certas regras para nomear as variáveis. Os nomes de variáveis devem ser iniciados por letras, não podem conter espaços nem caracteres de pontuação. O Octave diferencia letras maiúsculas de minúsculas. Alguns nomes são de uso restrito, como π , `inf`(infinito), `ans`(variável usada para armazenar os resultados), etc.

Uma função no Octave tem a forma geral:

```
function [lista-saída] = nome(lista-entrada)
    comandos da função
endfunction
```

onde lista-saída é uma lista de parâmetros de saída da função, separados por vírgula; lista-entrada é uma lista de parâmetros de entrada, separados por vírgula; nome é o nome dado à função.

Uma função pode ser criada digitando-a no ambiente de trabalho, ou criando um arquivo com a função e salvando-o no diretório de trabalho. O arquivo deve ter o mesmo nome dado à função e a extensão “.m”. Para declarar um intervalo usa-se o caracter “:” que deve estar entre o início e o fim do intervalo.

Como exemplo, considere-se a criação de uma função, chamada de `somaprod`, que recebe dois valores retorna a soma e o produto entre esses dois valores. A função fica da seguinte forma:

```
function [soma, produto] = somaprod(a,b)
% Funcao de exemplo
% Recebe dois parâmetros e calcula
% a soma e o produto entre os mesmos
soma = a+b;
produto = a*b;
endfunction
```

Após salvar a função no arquivo `somaprod.m`, a mesma pode ser usada como se fosse uma função pré-existente no Octave:

```
>> [s,p]=somaprod(3,2)
s = 5
p = 6
```

Para que a função possa ser usada também para realizar as operações de soma e produto em uma lista de valores ao mesmo tempo, o operador de produto (*) deve ser substituído pelo operador de multiplicação elemento por elemento (*). Assim, a evocação da função com a passagem de vetores, que nesse caso funcionam como listas de valores, retorna também listas de resultados, para cada variável de saída:

```
>> [s,p]=somaprod ([1 2],[3 4])
s =
4 6
p =3 8
```

Tabela 7: manipulação de funções.

Função	Operação
diff(f)	Calcula a derivada indefinida de f.
int(f)	Calcula a intergral indefinida de f
compose(f,g)	Determina a composta de f e g.
Finverse(expr)	Expande uma expressão.
finverse(expr)	Determina a inversa funcional da expressão expr.
Simple(expr)	Procura uma foram simples de escrever a expressão.
petty(expr)	Exibe a expressão numa forma mais bonita.
Simplify(expr)	Simplifica a expressão expr.
Solve (expr)	Acha as soluções da equação $expr=0$
Syms x y z a	Define as variáveis simbólicas x, y, z, e a.

Tabela8: Comandos para otimização

fmin	Minimiza a função de uma variável
fzero	Encontra o zero da função de uma variável
fmins	Minimiza uma função de muitas variáveis

Tabela9: Comandos para integração usando quadratura

quad	Calcula a integral numericamente, método para baixar ordem
quadb	Calcula a integral numericamente, método para aumentar ordem

Tabela10: Cores utilizadas em recursos gráficos.

Função	Descrição	Função	Descrição
Hsv	cores saturadas	Copper	tons acobreados.
Hot	Preto-vermelho-amarelo-branco	Flag	Vermelho, branco, azul e preto alternados.
Gray	linear de tons de cinza.	Cool	Tons de ciano e magenta
Bone	tons de cinza levemente azulados.	lines	cores que usa comando plot
Jet	variante do mapa hsv	Pink	Tons pastéis de rosa.

5 Exercícios

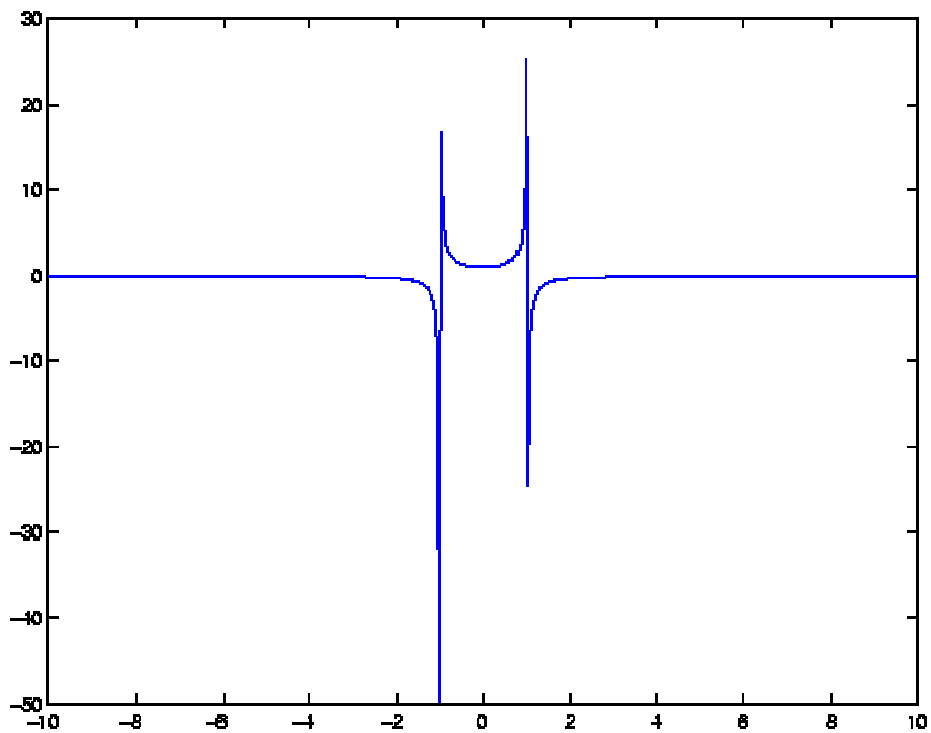
Alguns exercícios do livro “Cálculo com geometria analítica” do Edwards e Penney (maiores informações na bibliografia) são resolvidos usando os recursos mostrados anteriormente

5.1 Exercícios recomendados

A seguir, temos a resolução de alguns interessantes.

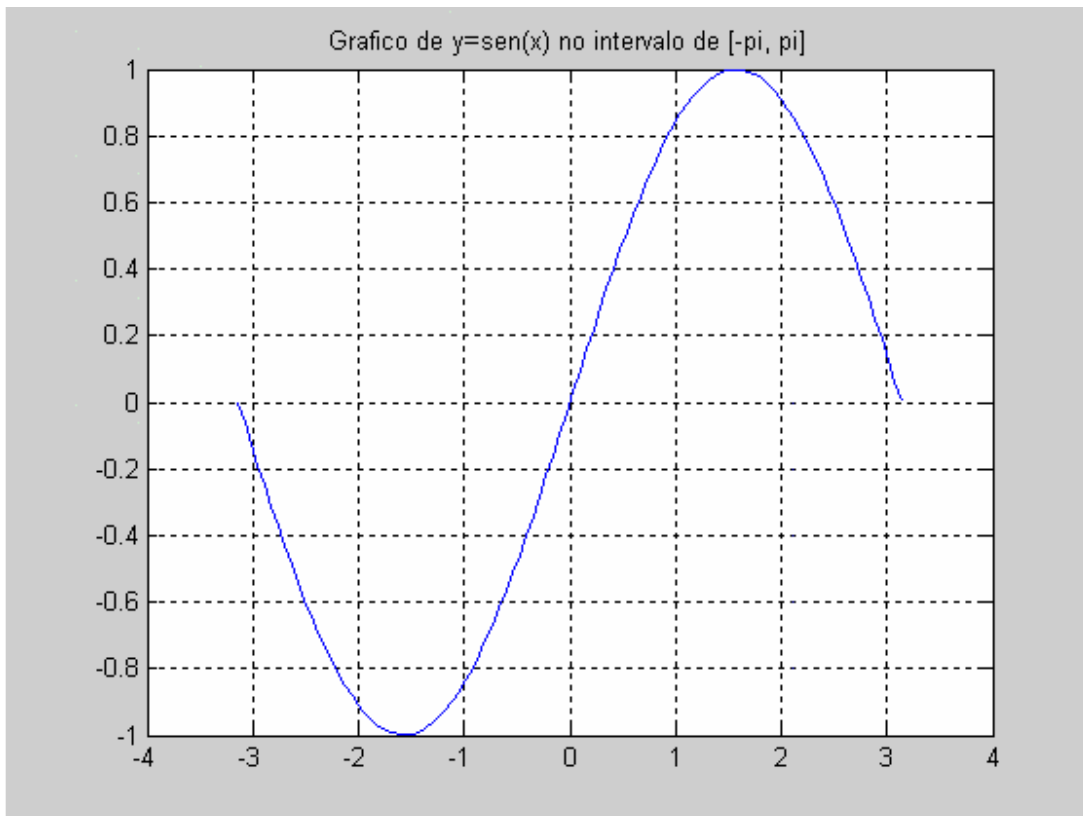
(1)

```
 $f(x) = 1/(1-x^2)$  %é a função dada;  
>>syms x %declarando uma variável simbólica;  
>> f=1/(1-x^2); %aparência da função na janela gráfica;  
>> plotf1(f,[-10,10],200); %comando plotf1 plota a função dada no intervalo declarado[-10,10];
```



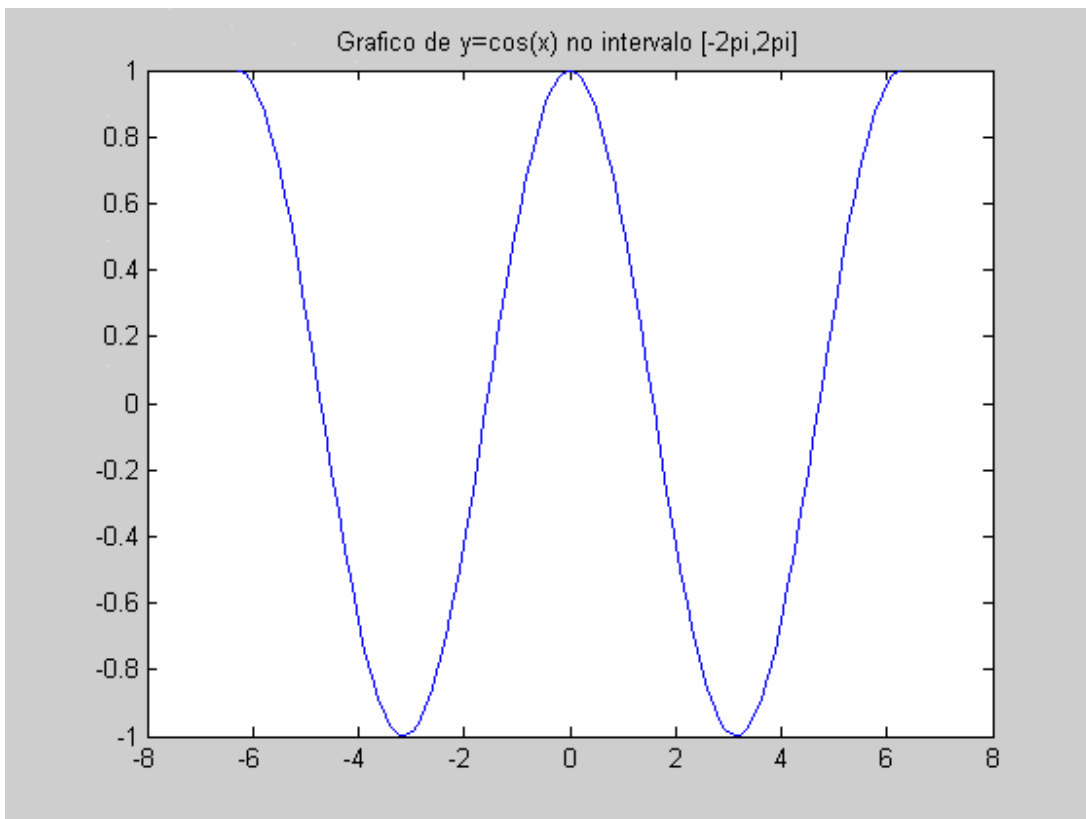
(2)

```
>> x=linspace(-pi,pi,100);  
>> y=sin(x);  
>> plot(x,y);  
>> title('Grafico de y=sen(x) no intervalo de [-pi, pi]');  
>> grid on;
```



(3)

```
>> x=linspace(-2*pi,2*pi,100);  
>> y=cos(x);  
>> plot(x,y);  
>> title('Grafico de y=cos(x) no intervalo [-pi,pi]');  
>> title('Grafico de y=cos(x) no intervalo [-2pi,2pi]');
```



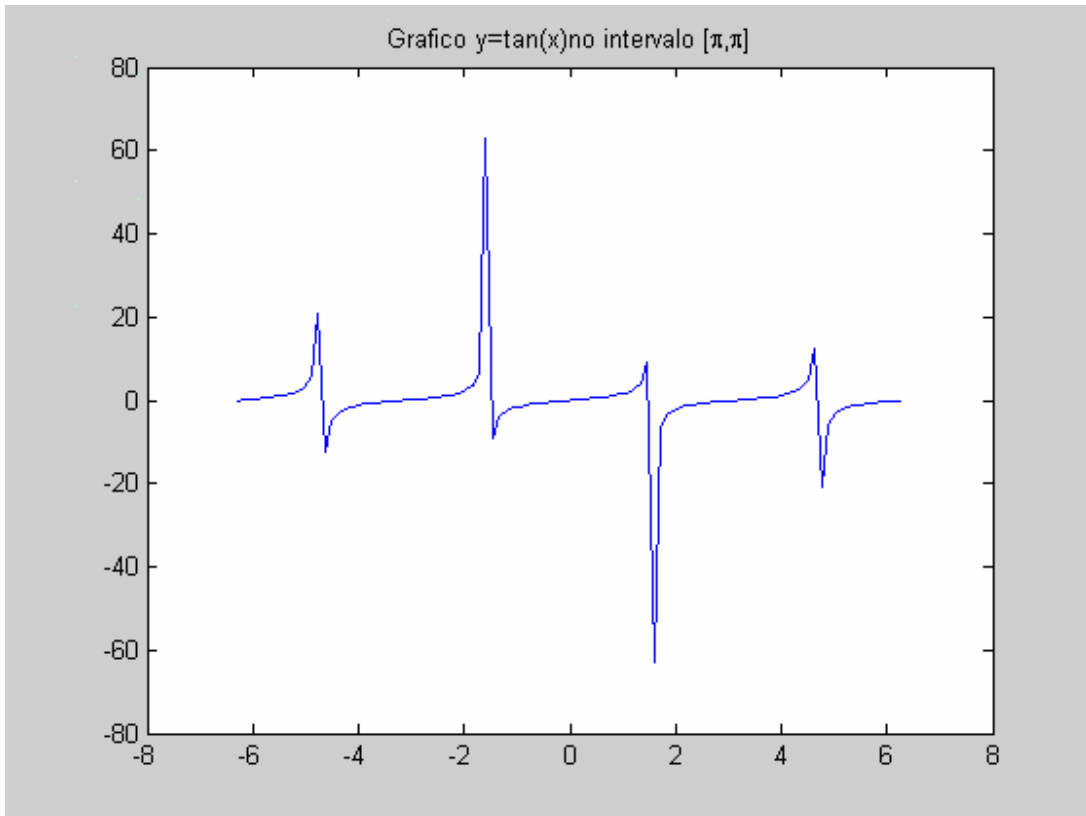
(4)

```
>> x=linspace(-2*pi,2*pi,100);
```

```
>> y=tan(x);
```

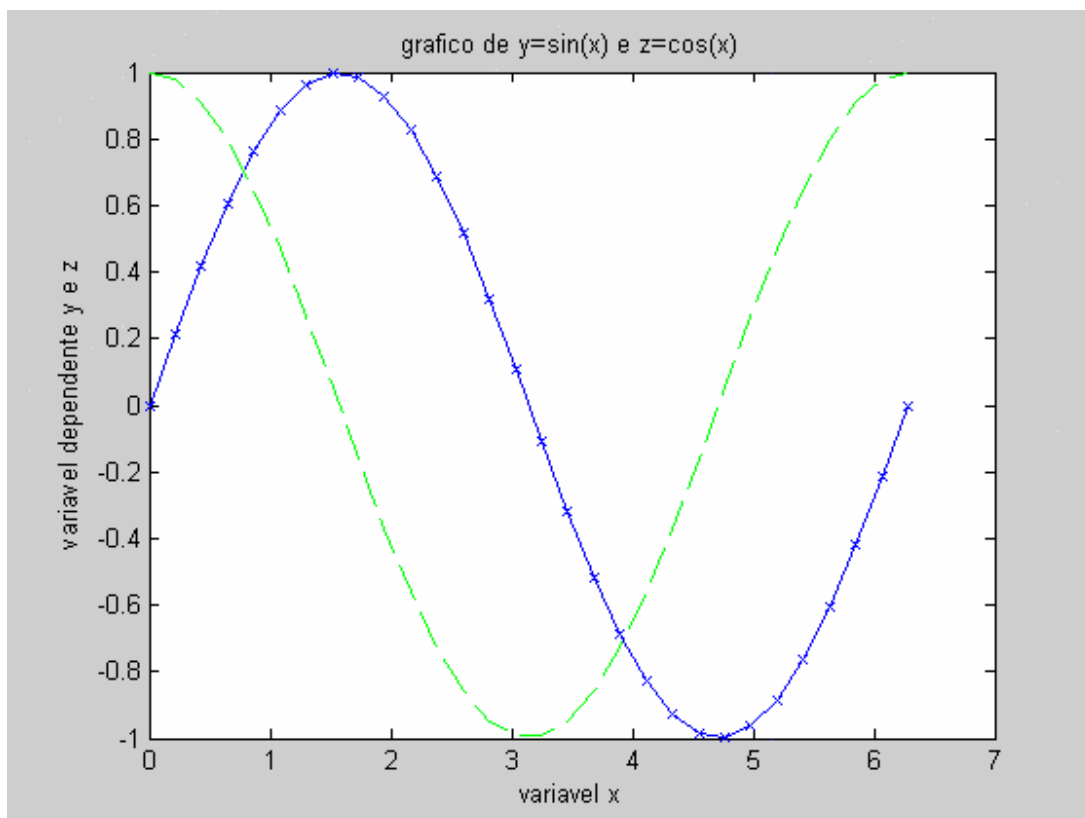
```
>> plot(x,y);
```

```
>> title('Grafico y=tan(x)no intervalo [ $\pi$ , $\pi$ ]'); %a notação \pi serve para mostrar o carácter  $\pi$ .
```



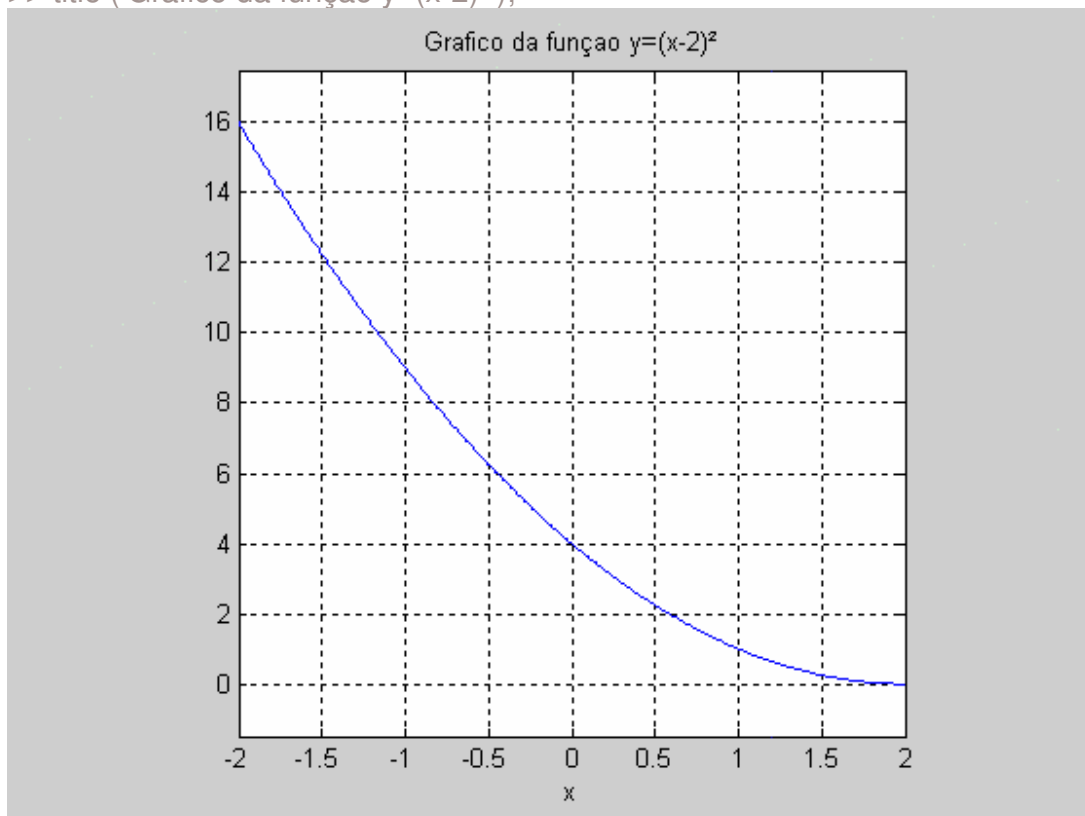
(5)

```
>> x=linspace(0,2*pi,30);  
>> y=sin(x);  
>> z=cos(x);  
>> plot(x,y,'-bx',x,z,'--g'); %aquí estamos definindo curva de seno com linha contínua e cor azul, a curva do cosseno tem linha tracejada e cor verde.  
>> title('grafico de y=sin(x) e z=cos(x)');  
>> xlabel('variavel x'); %aqui nomeia-se o eixo x  
>> ylabel('variavel dependente y e z'); %aquí nomeia-se o eixo y
```



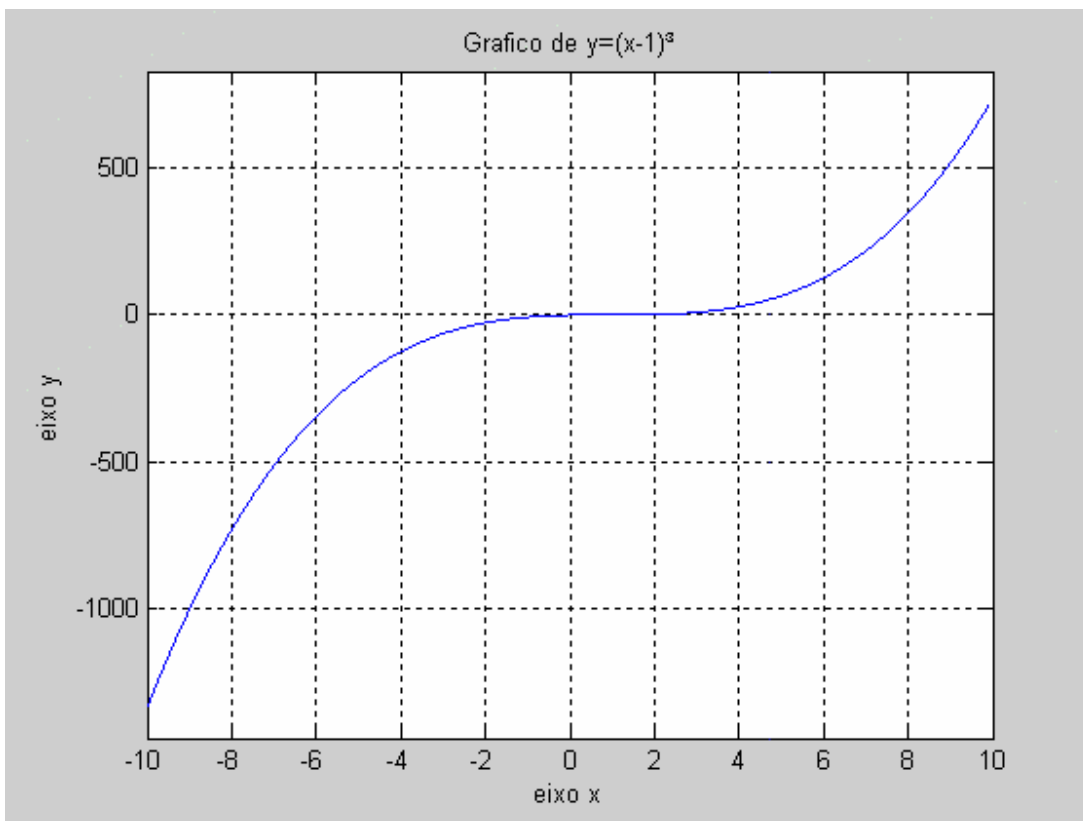
(6)

```
>> istr = '(x-2)^2';  
>> ezplot (istr, [-2 2]);  
>> axis square  
>> grid  
>> title ('Gráfico da função  $y=(x-2)^2$ ');
```



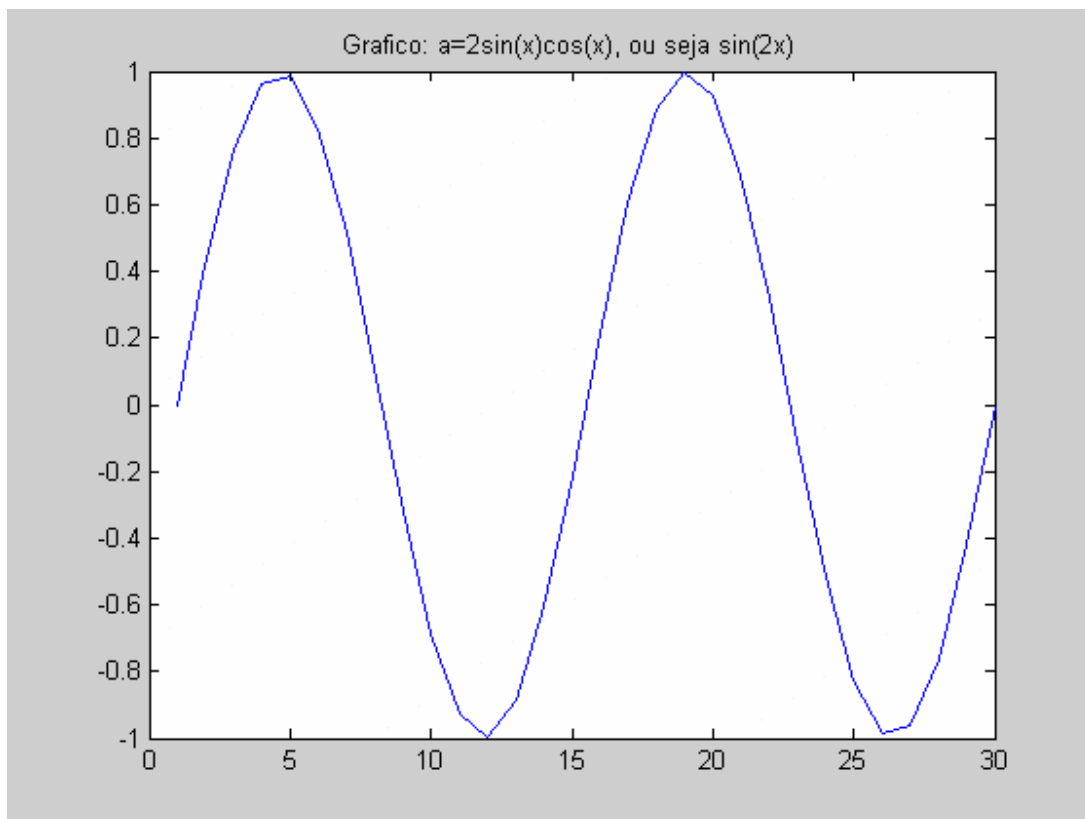
(7)

```
>> istr = '(x-1)^3';  
>> ezplot (istr, [-10 10]);  
>> grid  
>> title (Gragico de y=(x-1)^3);  
>> xlabel ('eixo x');  
>> ylabel ('eixo y');
```



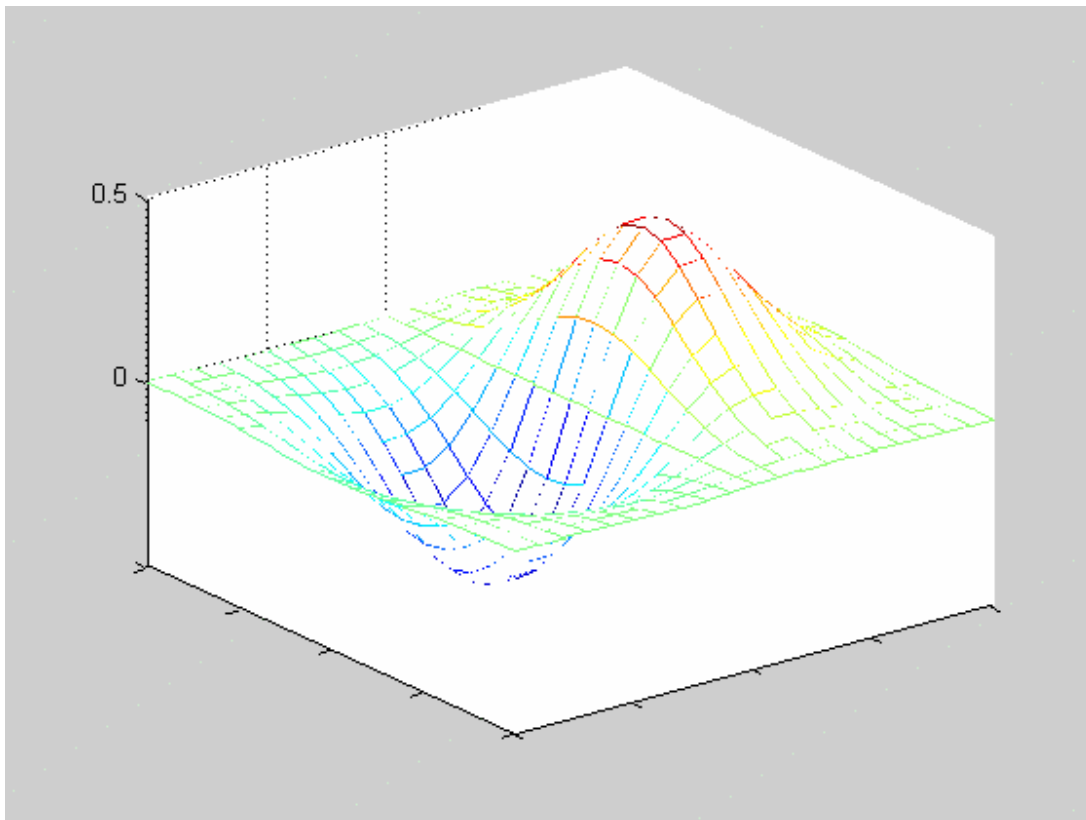
(8)

```
>> x = linspace(0,2*pi,30);  
>> y = sin(x);  
>> z = cos(x);  
>> a = 2*sin(x).*cos(x);  
>> plot(a);  
>> title('Gráfico: a=2sin(x)cos(x), ou seja sin(2x)');
```



(9) Gráfico em 3D

```
>>[X,Y] = meshgrid(-2:.2:2, -2:.2:2);  
>>Z = X .* exp(-X.^2 - Y.^2);  
>>mesh(X,Y,Z)  
>>surf(X,Y,Z)
```



(10) A derivada

```
>> syms x
>> y=(x.^2-x);
>> pretty(diff(y))
>> 2x-1
```

(11)

```
>> syms x
>> y=cos(x);
>> pretty(diff(y))
-sin(x)
```

(12)

```
>> syms x
>> y= cosh(x);
>> pretty(diff(y))
sinh(x)
```

(13) A integral

```
>> syms x
>> y=x.^3+x.^2+x
>> int (y)
ans =
1/4*x^4+1/3*x^3+1/2*x^2
```

(14) %Valor absoluto da integral

```
>> x=linspace(0,4); %aqui faz-se a declaração do intervalo de integração.
>> y= x;
>> area=trapz(x,y) % o valor absoluto pode ser interpretado como o valor da área
%no intervalo dado
>> area=
8.0000
>> int y
>> 1/2x^2 %é a integral procurada
```

(15) página 307 exercício 37

```
>>x=linspace(0,4);
>> y=sqrt(32*x)-x.^3;
>> int y
>> área= -33.85
```

(16) Volume

```
>> área=dblquad('sqrt(32*x)-x.^3',0,4) %o comando dblquad calcula o volume pó  
%aproximação de quadrados
```

```
>>area=  
85.3334
```

(17) F zero %calcula o zero do cosseno entre 1 e 2

```
>> x = fzero(@cos,[1 2])
```

```
x =
```

```
1.5708
```

(18) Raízes

```
%introduzir os coeficientes como uma matriz linha
```

```
>>p=[1 5 6];
```

```
>> roots(p)
```

```
ans =
```

```
-3.0000
```

```
-2.0000
```

(19) Derivada de polinômio

```
>>p=[1 5 6];
```

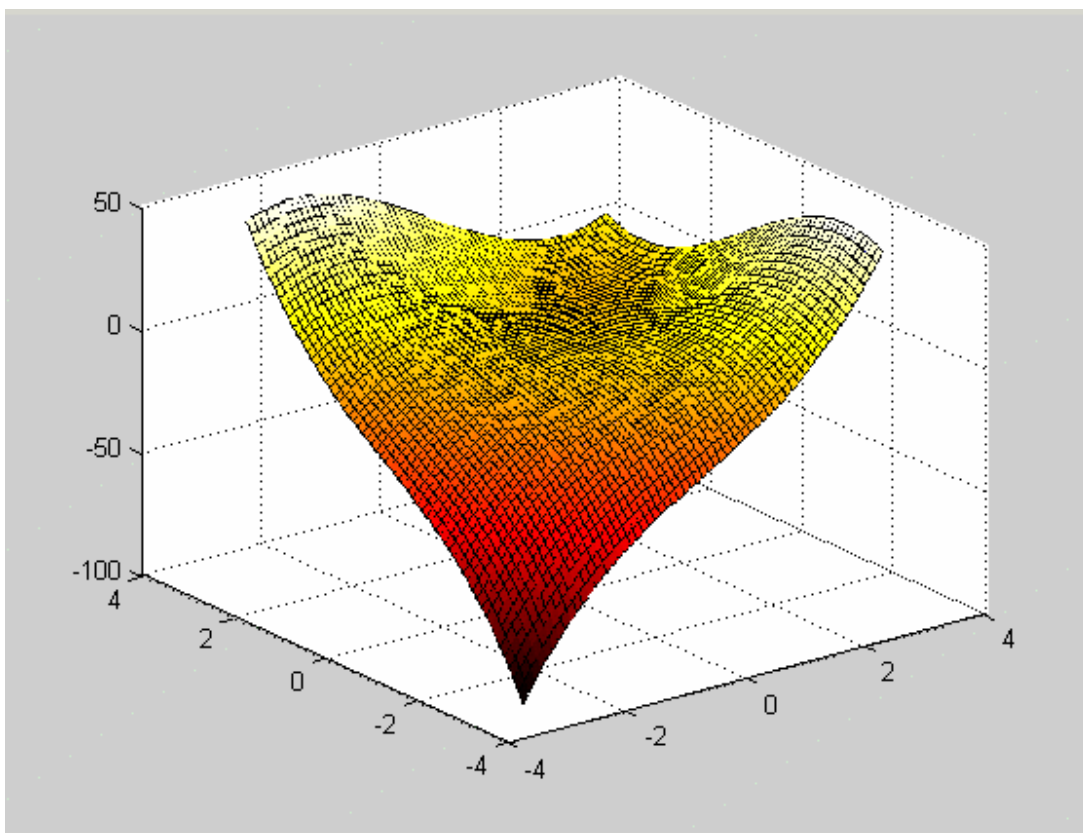
```
>> y=polyder(p) %são os coeficientes da derivada
```

```
y =
```

```
2 5
```


(20) Mais recursos para colorir gráficos

```
>> x=-3:0.1:3; y=x; [X,Y]=meshgrid(x,y); Z=X.^3 + Y.^3 - 5*X.*Y + 1/5;  
>> surf(X,Y,Z), colormap([summer]);  
>> surf(X,Y,Z);colormap([hot]);
```



6 Bibliografia

- Crawford, D. Gnuplot online documentation.
<http://www.gnuplot.info/docs/gnuplot.html>. 1998.
- Eaton, J. W. GNU Octave - A high-level interactive language for numerical computation.
<http://www.octave.org>. 1997.
- Free Software Foundation. GNU General Public License. Version 2.
<http://www.fsf.org/licensing/licenses/index.html>. Jun 1991.
- "*Cálculo com Geometria Analítica*", de Edwards e Penney, 4ª edição, 1999. LTC-
editora, Michigan.
- Hanselman, D; Littlefield, B; "*Matlab Completo*", Ed Prentice Hall, 2003.