UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
ESCOLA DE CIÊNCIAS E TECNOLOGIA
COMPUTAÇÃO NUMÉRICA

Apostila para o Minicurso de Introdução ao Scilab

Francisco Carlos G. da S. Segundo
Leonardo Vale de Araujo

Orientadores: Prof. Dr. Marconi Câmara Rodrigues

Prof. Dr. Diego Rodrigo Cabral Silva

Prof. Dr. José Josemar de Oliveira Júnior

Natal, RN, setembro de 2013

SUMÁRIO

1. INTRODUÇÃO AO SCILAB 3
2. O AMBIENTE DO SCILAB 5
   2.1. Introdução 5
   2.2. O Ambiente Gráfico do Scilab 6
   2.3. Variáveis Especiais 7
3. OPERAÇÕES BÁSICAS 8
   3.1. Operações Matemáticas e Lógicas 8
   3.2. Formato de Visualização dos Números 11
   3.3. Lista de Comandos e Funções Matemáticas 12
   3.4. Números Complexos 14
   3.5. Strings 14
   Exercícios Propostos 16
4. POLINÔMIOS 17
   Exercícios propostos 19
5. VETORES 20
   Exercícios propostos 23
6. MATRIZES 24
   Exercícios propostos 26
7. CONTROLE DE FLUXO 28
   7.1. Comandos para Iterações 28
      7.1.1.O Loop *for* 29
      7.1.2. O Loop *while* 30
   7.2. Comandos Condicionais 32
      7.2.1. Comandos *if-then-else* 32
      7.2.2. Comandos *select-case* 34
8. FUNÇÕES 36
   8.1. Funções definidas pelo usuário 37
9. GRÁFICOS 38
   9.1. Gráficos Bidimensionais 39
   9.2. Gráficos Tridimensionais 41
10. REFERÊNCIAS BIBLIOGRÁFICAS 44

1. INTRODUÇÃO AO SCILAB

O Scilab é um ambiente voltado para o desenvolvimento de software para resolução de problemas numéricos. O Scilab foi criado em 1990 por um grupo de pesquisadores do INRIA – Institut de Recherch e em Informatique e tem Automatique e do ENPC3 – École National e desPonts et Chaussées.

Desde 1994, quando passou a ser disponível na Internet, Scilab é gratuito, free software, e distribuído com o código fonte, open source software. Além da distribuição com o código fonte, existem, também, distribuições pré-compiladas do Scilab para vários sistemas operacionais. Na versão atual, Scilab está disponível para as seguintesplataformas:

- Plataforma GNU/Linux:

  - Scilab 5.4.1 – Linux 32 bits;
  - Scilab 5.4.1 – Linux 64 bits.

- Plataforma Windows XP, Vista, 7, 8:

  - Scilab 5.4.1 – Windows 32 bits;
  - Scilab 5.4.1 – Windows 64 bits.

- Plataforma Mac OS X:

  - Scilab 5.4.1 – Mac OS X.

Embora seja apresentado pelos seus mantenedores como um software CASCD – Computer Aided Control System Design - Projeto de Sistemas de Controle Auxiliado por Computador, Scilab é um ambiente para desenvolvimento ou prototipação de software numérico de propósitogeral.

O objetivo principal deste trabalho éapresentar o ambiente do Scilab através de um texto escritoem português com o objetivo de constituir um

material de referência para a disciplina Computação Numérica. Com este objetivo em mente, a ênfase maior é dada na apresentação das características do ambiente que serão utilizadas para a disciplina. Para esta apostila parte-se do princípio que o leitor já possua conhecimentos práticos, mesmo rudimentares, sobre programação.

O objetivo secundário, também relevante, é mostrar que a utilização de software livre e de código aberto, free/open source software, do ponto de vista do usuário, traz grandes vantagens como:

- A última versão do software está sempre disponível, geralmente através da Internet;
- O software pode ser legalmente utilizado, copiado, distribuído, modificado;
- Os resultados obtidos podem ser divulgados sem nenhuma restrição;
- Os programas desenvolvidos podem ser transferidos para outras pessoas sem imposiçõesou constrangimentos de quaisquer natureza;
- O acesso ao código fonte, evitando surpresas desagradáveis;
- O acesso à informação de alta qualidade, e
- A certeza de estar participando de uma comunidade cujo principal valor é a irrestrita difusão do conhecimento.

2. O AMBIENTE DO SCILAB

Neste Capítulo, são apresentadas algumas características do ambiente Scilab em plataforma. Em seguida, são mostrados exemplos de manipulação de arquivos e de diretórios apartir desse ambiente. O objetivo é a familiarização com o software.

2.1. Introdução

O Scilab é um ambiente de programação numérica bastante flexível. Suas principais características são:

- É um software de distribuição gratuita, com código fonte disponível. Sua linguagem é simples e de fácil aprendizado;
- Possui um sistema de auxílio ao usuário (help);
- É um ambiente poderoso para geração de gráficos bidimensionais e tridimensionais, inclusive com animação;
- Implementa diversas funções para manipulação de matrizes. As operações de concatenação, acesso e extração de elementos, transposição, adição e multiplicação de matrizes sãofacilmente realizadas;
- Permite trabalhar com polinômios, funções de transferência, sistemas lineares e grafos;
- Apresenta facilidades para a definição de funções;
- Permite o acesso a rotinas escritas nas linguagens FORTRAN ou C;
- Pode ser acessado por programas de computação simbólica como o Maple, que é um software comercial, ou o MuPAD, que é livre para uso em instituições de ensino/pesquisa;
- Suporta o desenvolvimento de conjuntos de funções voltadas para aplicações específicas, os chamados toolboxes.

2.2. O Ambiente Gráfico do Scilab

Após a realização dos procedimentos de instalação, é possível começar a trabalhar com Scilab. A tela inicial do Scilab é apresentada na Figura 2.1.

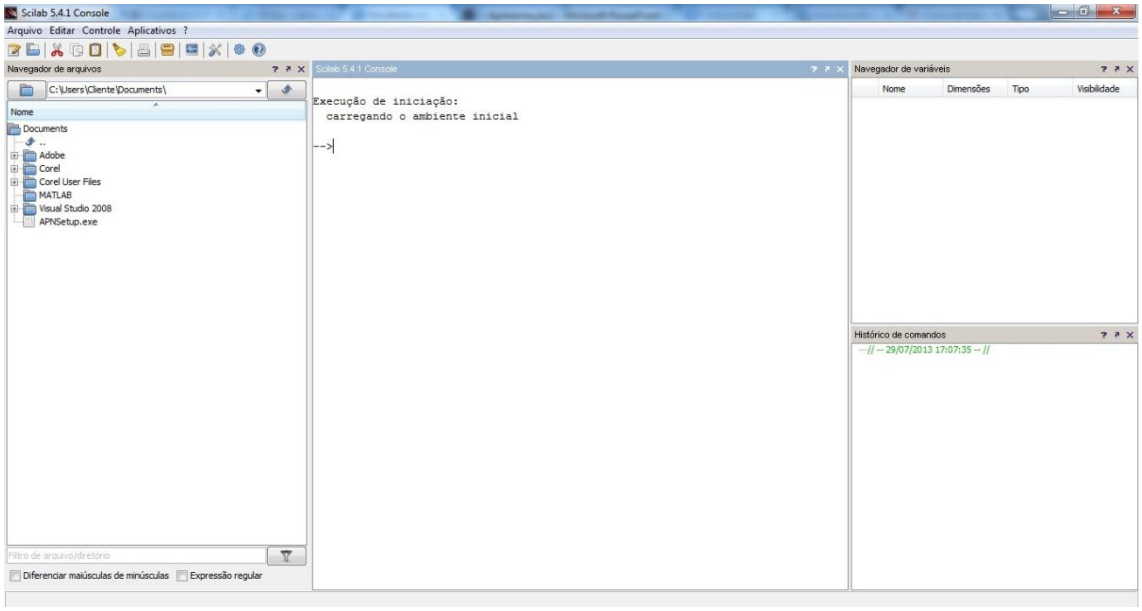

Figura 2.1 – Tela inicial do Scilab.

Podemos observar o prompt inicial, representado pelo símbolo -->, que faz a marcação da linha de comando e o cursor, ao lado do prompt. Nesse espaço, podem ser realizados cálculos e compilados programas sem serem salvas alterações. É através dele também que pode ser acessado o help do programa, através do comando help digitado no prompt. Será aberta uma janela depois de o comando ser executado (apertando a tecla ENTER), contento uma lista de todas as funções presentes.

Ainda na tela inicial estão presentes os menus dropdown com as opções Arquivo, Editar, Controle, Aplicativos e “?”. Cada menu possui as seguintesopções:

- **Arquivo:** Executar, Abrir um arquivo, Carregar ambiente, Salvar ambiente, Alterar o diretório atual, Exibir o diretório atual, Configuração de página, Imprimir e Sair.

- **Editar:** Recortar, Copiar, Colar, Esvaziar a área de transferência, Selecionar tudo, Mostrar/esconder a barra de ferramentas, Limpar histórico, Limpar o console e Preferências.
- **Controle:** Retomar, Abortar e Interromper.
- **Aplicativos:**SciNotes, Xcos, Tradutor de Matlab para Scilab, Gerenciador de módulos – ATOMS, Navegador de variáveis, Histórico de comandos e Navegador de arquivos.
- **"?"**: Ajuda do Scilab, Demonstrações do Scilab, Links da Web, ScilabEnterprises e Sobre o Scilab.

2.3. Variáveis Especiais

Existem variáveis que assumem valores predefinidos no Scilab. Elas podem ser vistas através do comando who ou what. Essas variáveis são protegidas e não podem ser apagadas. Algumas destas variáveis são prefixadas com o caractere %. A Figura 2.2 mostra o resultado desse comando.

```
Suas variáveis são:

                    home           scicos_autolib          scicos_utilslib
       graphic_exportlib              graphicslib              datatipslib
                uitreelib              scinoteslib                  jvmlib
                tclscilib                 atomslib              atomsguilib
            parameterslib   simulated_annealinglib    genetic_algorithmslib
           spreadsheetlib            demo_toolslib      external_objectslib
                  m2scilib  compatibility_functilib               arnoldilib
         windows_toolslib                    WSCI                  timelib
    special_functionslib                  sparselib     signal_processinglib
                      %s           polynomialslib           overloadinglib
            optimbaselib            neldermeadlib          optimizationlib
       linear_algebralib         output_streamlib                   iolib
         dynamic_linklib       data_structureslib                 cacsdlib
            functionslib  elementary_functionslib differential_equationlib
                     PWD                      %tk                       %F
                    %nan                     %inf                     SCI
                  TMPDIR                     %gui                   %fftw
                      %t                       %f                    %eps
                      %i                       %e                    %pi

 utilizando       7843 elementos de um total de   10000000.
 e           75 variáveis de um total de       9231.

Suas variáveis globais são:

           %modalWarning               %toolboxes           %toolboxes_dir

 utilizando         25 elementos de um total de        999.
 e            3 variáveis de um total de        767.

-->
```

Figura 2.2 –Saída do comando who.

3. OPERAÇÕES BÁSICAS

Scilab é um ambiente de programação numérica. Desse modo, existem duas formas de interação com o software: digitação diretamente no prompt, em que se tem uso de uma poderosíssima máquina de cálculo, ou como programação numérica propriamente dita, em que se delineiam linhas de código. Neste capítulo será abordada a primeira forma.

A primeira instrução a respeito do prompt é quanto ao uso do ponto e vírgula: ele faz com que o compilador interprete o fim da execução da linha de comando e esteja pronto para receber outra linha. Neste caso, o resultado da operação fica mascarado para o usuário. Se não for escrito o ponto e vírgula, a quebra de linha vai denotar fim da execução, mas o resultado será exibido para o usuário. Ambas as formas têm sua aplicação. Quando se está trabalhando com vetores de muitos elementos, não é conveniente esquecer o ponto e vírgula, pois isto pode acarretar perda de tempo e travamento da máquina (o Scilab tem uma ferramenta útil para evitar esse infortúnio: em algumas linhas de exibição, aparece uma mensagem perguntando se se desejacontinuar a exibição); já quando se quer testar uma operação, debugar um programa ou se as variáveis usadas forem de baixa ordem, o ponto e vírgula é por vezes necessário (no mínimo, facultativo).

Outra observação importante é que o Scilab é case sensitive. Quando oprogramador definir uma variável com letras maiúsculas e minúsculas, deve-se lembrar de manter a caixa das letras.

3.1. Operações Matemáticas e Lógicas

De um modo geral, as variáveis matemáticas usadas no Scilab são vetores complexos. O número imaginário é denotado por %i e, seguindo a notação padrão de um número complexo z = a+bi, tem-se as seguintes operações possíveis:

- Adição – é representado por “+”;
- Subtração – é representado por “-”;

- Multiplicação – é representado por “*”;
- Divisão à Direita – é representado por “/”;
- Divisão àEsquerda – é representado por “\”
- Potenciação – é representado por “^” ou “**”.

Exercício: Digite no console os seguintes comandos e visualize os resultados:

a) 5 + 7
b) 15 – 25
c) 9/2
d) 9\2
e) 3^2
f) 3**2

Os operadores lógicos também são bastante importantes que são dados por:

- a&b – representa E lógico (AND);
- a|b – representa OU lógico (OR);
- ~a – representa NÃO lógico (NOT);
- a==b – as duas variáveis são iguais.
- a~=b ou a<>b – as duas variáveis são diferentes.
- a<b – a é menor que b.
- a>b – a é maior que b.
- a<=b – a é menor ou igual a b.
- a>=b – a é maior ou igual a b.

Exercício: Digite no console os seguintes comandos e visualize os resultados.

Para a = %T e b = (0==1):

a) a&b
b) a|b
c) ~a

Para a = 5 e b = 8:

d) a==b
e) a~=b
f) a<b
g) a>b

Outro ponto importante a ser considerado nas operações matemáticas é a ordem de precedência. A Tabela 3.1 mostra qual a ordem de prioridade utilizada pelo Scilab.

Tabela 3.1 – Ordem de prioridade do Scilab.

Prioridade	**Operações**
1ª	Resolver parênteses
2ª	Avaliar função
3ª	Potenciação
4ª	Menos e mais unário
5ª	Multiplicação e divisão
6ª	Adição e subtração

Vale salientar que para a mesma prioridade resolve-se a expressão mais à esquerda.

Exercício: Digite no console os seguintes comandos e visualize os resultados.

a) A = 1 + 2/3 + 4
b) B = (1 + 2)/(3 + 4)
c) C = 1 + 2/(3 + (4+1)/2)
d) D = 3*2/5
e) E = 5^2/10

3.2. Formato de Visualização dos Números

O comando format modifica a quantidade de dígitos com que os números são mostradosno Scilab. Por exemplo, o comando:

```
-->format(5)
```

fará com que todas os números sejam visualizados em 5 posições (incluindo o pontodecimal e um espaço para o sinal). Por exemplo:

```
-->sqrt(3)
ans =
1.73
```

Para aumentar o número de posições para 16, usa-se:

```
-->format(16)
-->sqrt(3)
ans =
1.7320508075689
```

A raiz de 3 foi mostrada ocupando 16 posições (sendo uma posição para o ponto, umespaço reservado para o sinal, uma posição para a parte inteira e 13 posições para a partefracionária).

O comando format('e') mostra os números em notação científica. Por exemplo:

```
-->format('e')
-->2*%pi/10
ans =
6.283185307E-01
```

6.283185307E-01 significa $6.283185307x10^{-1}$. Para retornar ao formato inicial usa-se:

```
-->format('v')
```

que é chamado de “formato de variável”. Existem outras formas de usar o comandoformat:

```
-->format('v',10)
```

mostra os números em formato de variável com 10 posições.

```
-->format('e',8)
```

mostra os números em notação científica com 8 posições.

3.3. Lista de Comandos e Funções Matemáticas

É importante que os usuários do Scilab conheçam algumas das funções básicas. A Tabela 3.2 mostra alguns comandos para a melhor utilização do Espaço de Trabalho e para o gerenciamento de arquivos em diretórios. Já a Tabela 3.3 mostra uma lista de funções matemáticas normalmente utilizadas pelos usuários.

Tabela 3.2 – Comandos utilizados no Scilab.

Comandos	Funcionalidade
clc	Limpa a tela do console
clear	Limpa as variáveis armazenadas
help	Ajuda do Scilab
pwd	Mostra o diretório de trabalho
chdir	Muda de diretório
dir	Mostra o que tem no diretório
save	Salva variáveis para uso posterior
load	Carrega o arquivo salvo

Tabela 3.3 – Funções matemáticas do Scilab.

`abs(x)`	Valor absoluto.
`acos(x)`	Arco co-seno.
`acosh(x)`	Arco co-seno hiperbólico.
`asin(x)`	Arco seno.
`asinh(x)`	Arco seno hiperbólico.
`atan(x)`	Arco tangente.
`atanh(x)`	Arco tangente hiperbólico.
`conj(x)`	Conjugado.
`cos(x)`	Co-seno.
`cosh(x)`	Co-seno hiperbólico.
`exp(x)`	Exponencial: e^x.
`imag(x)`	Parte imaginária de um número complexo.
`log(x)`	Logaritmo natural.
`log10(x)`	Logaritmo na base 10.
`real(x)`	Parte real de um número complexo.
`modulo(x,y)`	Resto da divisão de x por y.
`sign(x)`	Função sinal: retorna o valor -1, +1 ou zero conforme o argumento x seja negativo, positivo ou nulo, respectivamente.
`sin(x)`	Seno.
`sinh(x)`	Seno hiperbólico.
`sqrt(x)`	Raiz quadrada.
`tan(x)`	Tangente.
`tanh(x)`	Tangente hiperbólica.

Exercício: Digite no console os seguintes comandos e visualize os resultados.

a) cos(2*%pi)
b) %e^2
c) abs(-5)
d) modulo(8,3)
e) modulo(6,3)
f) sign(-4)

g) sign(5)

3.4. Números Complexos

Não é necessário manuseio especial em Scilab para números complexos. As operaçõescom números complexos são tão fáceis como nos reais. A unidade imaginária é representada por %i, ou seja, %i é igual a sqrt(-1).

Exemplos:

```
x = 3 + 4*%i
y = 1 - %i
z1 = x - y
z2 = x * y
z3 = x / y
real(z1)                 // Parte real de z1
imag(z1)                 // Parte imaginária de z1
abs(x)                   // Valor absoluto do número complexo
atan(imag(x),real(x))    // Argumento do número complexo
conj(z2)                 // Conjugado
sin(x)                   // Seno de um número complexo
```

3.5. Strings

Strings são usados para toda e qualquer informação composta de caracteres alfanuméricose/ou caracteres especiais (exemplo, #, $, &, %, ?, !, @, <, ~, etc). Os strings são envolvidos por aspas duplas ou simples.

Exemplos:

```
-->a = "abcd"
a =
abcd
```

```
-->b = 'efgh'
b =
efgh

-->c = "Maria e Jose"
c =
Maria e Jose
```

Uma das atividades mais comuns em programação é a concatenação de strings. Concatenaçãoé a junção de dois ou mais strings. Isto pode ser feito com o operador “+”.

```
-->a + b // Concatena abcd com efgh
ans =
abcdefgh

-->n = "Pedro"
n =
Pedro

-->m = "Paulo"
m =
Paulo

-->m + n // Concatena Paulo com Pedro sem espaço entre eles
ans =
PauloPedro

-->m + " e " + n // Concatena Paulo com Pedro inserindo espaço entre eles
ans =
Paulo e Pedro
```

Muitas vezes precisamos armazenar informações que contém as aspas. Isto pode serfeito repetindo as aspas. Exemplos:

-->n = "O oráculo disse ""conheça-te a ti mesmo"" para Socrátes."
n =
O oráculo disse "conheça-te a ti mesmo" para Socrátes.

Algumas funções para manipulação de strings são mostradas da Tabela 3.4. Exemplos:

Tabela 3.4 – Funções para manipulação de strings do Scilab.

`convstr`	Retorna os caracteres de um string convertidos para maiúscula ou minúscula.
`length`	Comprimento de um string.
`part`	Extrai caracteres de um string.
`strindex`	Procura a posição de um string dentro de outro.
`strcat`	Concatena strings.
`string`	Converte número para string.
`evstr`	Converte string em número. Também avalia expressões aritméticas.
`eval`	Converte string em número. Também avalia expressões aritméticas.
`strsubst`	Substitui uma parte de um string por um outro string.

Exercícios Propostos

1 – Para os valores de h = 0.25, d = 2, p = 3, a = 5 e b = -1 quais são os valores das expressões abaixo:

a) $c = (h + 0.5d)\ln\left(\frac{2h}{p}\right)$

b) $z = 2e^{x \sin x\pi}$

c) $m = 2\left(y^2 + \frac{p}{p - 1 + p^2}\right)$

d) $s = \sqrt{\frac{\sin^{a+b} x}{a + b}}$

e) $g = L\left[0.5\pi r^2 - r^2 \arcsin(h/r) - h(r^2 - h^2)\right]^{1/2}$

2 – Para os valores de A = 11, B = 5, C = -4 e D = 2 quais são os valores das expressões abaixo:

a) `3*modulo(A,3)-C`
b) `2^(2*abs(C))/8`
c) `(A/B-fix(A/B)+sign(C)+2.8)^(15/B)`
d) `sqrt(cos(A)^2+sin(A)^2) + sin(D*%pi/4)`
e) `(A+C)/A * round(sign(C)+D/4)-fix(D/1.5)`

3 – Faça um programa no editor que a partir da temperatura fornecida pelo usuário, o programa converta a temperatura de graus Celsius para Fahrenheit. A fórmula é dada por: F = 9*C/5 + 32. Na saída, apresente no console o texto: “A temperatura é __ F”.

4 – Elabore um programa para calcular a resistência equivalente entre dois resistores R1 e R2 em paralelo. A fórmula é dada por: Req = (R1*R2)/(R1+R2). Apresente o resultado usando o comando disp.

4. POLINÔMIOS

Os polinômios são criados no Scilab através da utilização da função “poly”. Vale salientar quepolinômios de mesma variável podem ser somados, subtraídos, multiplicados e divididos entresi. Por exemplo, o polinômio $p = s^2 - 3s + 2$, que possui raízes 1 e 2, pode ser criado através do comando:

```
-->// Polinomio definido pelas suas raizes
-->p = poly([1 2], 's')
p =
2 - 3s + s^2
```

Com a função roots, comprova-se que as raízes de p são, realmente, 1 e 2:

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

ans =

! 1. !

! 2. !

Um polinômio também pode ser criado a partir da especificação de seus coeficientes. Porexemplo, o polinômio q = 2s + 1 é criado através do comando:

-->// Polinomio definido pelos seus coeficientes

-->q = poly([1 2], ’s’, ’coeff’)

q =

1 + 2s

-->roots(q) // Obtendo as raizes do polinomio q

ans =

- 0.5

Para complementar o exemplo, os dois polinômios podem ser multiplicados, divididos, somados ou subtraídos como mostra a sequência de comandos:

-->p * q // Multiplicacao

ans =

$2 + s - 5s^2 + 2s^3$

-->p / q // Divisao

ans =

$2 - 3s + s^2$

---

1 + 2s

-->[r, q] = pdiv(p,q) // Efetuando a divisao: q=quociente, r=resto

q =

- 1.75 + 0.5s

r =

3.75

-->p + q // Adicao
ans =
$3 - s + s^2$

-->p - q // Subtracao
ans =
$1 - 5s + s^2$

Para obter valores de polinômios, utiliza-se a função horner:

-->x = poly(0, 'x')
x =
x

-->p = x^2 - 3*x + 5 // definindo o polinomio
p =
$5 - 3x + x^2$

-->horner(p, 2) // avaliando o polinomio em x = 2
ans =
3.

Exercícios propostos

1 – Calcule as raízes dos seguintes polinômios:

a) $p(x) = -\frac{7}{3}x^4 - \frac{16}{3}x^2 + 25x$

b) $p(x) = x^7 - 9x^6 + 2$

2 – Calcule os valores das operações abaixo:

$$a)\ y1 = (x^5 + 3x^4 + 6x^3 + 4x^2 + x + 1)/(x+1)$$

$$b)\ y2 = (x^4 + 2x^3 + 4x^2 + 1).(x+1)$$

$$c)\ y3 = (x^2 + 2x + 2).(x^7 + 1)$$

$$d)\ y4 = (x^2 + 2x + 1)/(x+1)$$

5. VETORES

Considerando R o conjunto dos números reais. É possível dizer que x é um vetor de dimensão n em R, indicado por $x \in R^n$, se, e somente se,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Nessa definição, cada um dos elementos do vetor x, x_i, pertence a R,

$$x_i \in R$$

O elemento xi é o i-ésimo elemento do vetor x.O vetor x definido anteriormente é um vetor coluna.

No Scilab, os vetores são criados colocando-se seus componentes entre colchetes. Oselementos de um vetor coluna são separados por ponto e vírgula. Assim:

```
-->x = [ 1; 2; 3] // vetor coluna. Elementos separados por ;
x =
! 1. !
```

! 2. !
! 3. !

Um vetor linha, y, de dimensão n em R pode ser escrito na forma:

$$y = [y_1 \quad y_2 \quad \cdots \quad y_n]$$

No Scilab, os componentes de um vetor linha são separados por espaço ou por vírgula.

-->y = [1 2 3] // vetor linha; Elementos separados por espaco
y =
! 1. 2. 3. !

-->z = [4, 5, 6] // vetor linha; Elementos separados por virgula
z =
! 4. 5. 6. !

Vetores podem ser multiplicados ou divididos por quantidades escalares. Vetoresde mesma dimensão podem ser somados ou subtraídos. Para exemplificar algumas dessas operações, são considerados os vetores:

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \text{e} y = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

-->x = [1; 2; 3] // Definindo o vetor x
x =
! 1. !
! 2. !
! 3. !

-->y = [4; 5; 6] // Definindo o vetor y
y =

```
! 4. !
! 5. !
! 6. !

-->size(x) // Dimensao do vetor x
ans =
! 3. 1. !

-->size(y) // Dimensao do vetor y
ans =
! 3. 1. !

-->3 * x // Multiplicando o vetor x por uma constante
ans =
! 3. !
! 6. !
! 9. !

-->x / 2 // Dividindo o vetor x por uma constante
ans =
! 0.5 !
! 1. !
! 1.5 !

-->x + y // Somando os dois vetores
ans =
! 5. !
! 7. !
! 9. !
```

Exercícios propostos

1 – Construa um vetor com elementos crescentes que começa em 10 e termina em 20, com passo de 5.

2 – Construa um vetor com elementos crescentes que começa em 10 e termina em 70 com 7 elementos.

3 – Gere uma sequência de números pares começando em 4 e não ultrapassando 15.

4 – Crie um vetor A de 15 elementos aleatórios e a partir deste, crie outros vetores de acordo com os seguintes critérios:

a) B1 = Conter somente os elementos de A maiores que 0,5. Calcule quantos elementos são.

b) B2 = Conter somente os elementos de A menores que 0.2. Calcule quantos elementos são.

c) B3 = Conter os elementos de A em Ordem Crescente.

d) B4 = Conter os elementos de A em Ordem Decrescente.

Obs.: use o comando apropos para encontrar o comando que ordena o vetor em ordem crescente ou decrescente.

5 – Extraia os quatro últimos elementos do vetor a e armazene na variável c:

a = [1 30 100 3 10 30 90 12 3 4 5 30 30 0.5]

Obs.: Use o comando length e $.

6 – Crie uma expressão que calcule o valor de y para as equações abaixo. Defina x variando de 2 a 19 espaçados de 5 em 5 unidades.

$a)\ y = x^2$

$b)\ y = (x-1)^3$

$c)\ y = (x^2-1)^3$

$d)\ y = \frac{\log(3x)}{2} + x^2$

$e)\ y = \frac{\ln(3x)}{\sqrt{2}} - \sum x$

$f)\ y = e^{-4x}$

$g)\ y = sen(x^2)$

$h)\ y = \tan^3(100\,\pi\, x)$

$i)\ y = \frac{sen(x)}{x}$

$j)\ y = \frac{\log[sen(x)]}{2} + \cot(x-1)$

$l)\ y = \left[\sum_{i=0}^{n} sen(x_i)\right].\left[\sum_{i=0}^{n} \cos^3(x_i)\right]$

$m)\ y = \frac{\mathrm{Re}[\log(-x)]}{\mathrm{Im}[\log(-x^2)]}$

Obs.: Use o comando disp para mostrar os resultados no formato y (para os valores de x iguais a __) = resultado.

6. MATRIZES

Matrizes usam dois índices para individualizar elementos. Elas são construídas utilizando colchetes. Cada linha da matrizé separada por um ponto e vírgula e cada elemento de uma linha é separado por espaço(ou vírgula). Por exemplo, a seguinte matriz:

$$A = \begin{bmatrix} 2 & 3 & 4 \\ 4 & 5 & 2 \end{bmatrix}$$

Poderia ser construída pelo comando:

```
-->a = [2 3 4; 4 5 2]
a =
! 2. 3. 4. !
```

! 4. 5. 2. !

Nos exemplos a seguir, para fixar conceitos, a matriz A é digitada com os elementos de suas linhas separados por espaços enquanto a matriz B é digitada com os elementos de suas linhas separados por vírgula. Assim,

```
-->// Matriz A - Elementos das linhas separados por espaço
-->A = [1 2 3; 5 -8 9]
A =
! 1. 2. 3. !
! 5. - 8. 9. !

-->// Matriz B - Elementos das linhas separados por virgulas
-->B = [1, 2, 3; 4, 5, 6]
B =
! 1. 2. 3. !
! 4. 5. 6. !

-->size(A) // Dimensao da matriz A
ans =
! 2. 3. !

-->size(B) // Dimensao da matriz B
ans =
! 2. 3. !
```

A Tabela 6.1 mostra uma lista com as possíveis operações utilizando matrizes e a Tabela 6.2 apresenta uma lista de comandos normalmente utilizados quando trabalha-se com matrizes:

Tabela 6.1 – Operações matemáticas com matrizes.

Operação	Comando
Adição	A + B
Subtração	A – B
Multiplicação	A * B
Divisão pela direita	B / A
Divisão pela esquerda	A \ B
Exponenciação	A ^ 3
Multiplicação individual	A.*B
Divisão individual	A./B
Exponenciação individual	A.^3

Tabela 6.2 – Operações matriciais.

Operação	Comando
Determinante	det(A)
Transposta	A’
Inversa	inv(A)
Diagonal	diag(A)
Traço	trace(A)

Exercícios propostos

1 – Encontrar o valor de x1, x2 e x3.

$$\begin{bmatrix} -1 & 2 & 0 \\ \frac{1}{2} & 9 & 3 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \\ 7 \end{bmatrix}$$

2 – Um construtor tem contratos para construir 3 estilos de casa: moderno, mediterrâneo e colonial. A quantidade de material empregada em cada tipo de casa é dada pela tabela:

	Ferro	Madeira	Vidro	Tinta	Tijolo
Moderno	5	20	16	7	17
Mediterrâneo	7	18	12	9	21
Colonial	6	25	8	5	13

Faça um programa que armazene os valores da tabela acima em uma matriz chamada material e que seja capaz de:

a) Perguntar quantas casas e de qual tipo se deseja construir (usar comando input e armazenar no vetor qcasas).

b) Mostrar a quantidade de material de cada tipo a ser utilizado no projeto total.

c) O preço total do projeto de construção das casas, supondo que os preços por unidade de ferro, madeira, vidro, tinta e tijolo sejam, respectivamente, 15, 8, 5, 1 e 10 (Armazenar esses preços no vetor preço).

d) Mostre também o preço unitário de cada casa.

7. CONTROLE DE FLUXO

Uma das características mais importante do Scilab é a facilidade com que o usuário pode criar seus próprios programas.

Apesar de simples, a linguagem Scilab disponibiliza a maioria das estruturas das linguagens de programação convencionais. A diferença principal é que, na programação Scilab, não há a necessidade da declaração prévia dos tipos das variáveis que serão utilizadas ao longo do programa.

Um fator a ser levado em consideração é que Scilab é um interpretador de comandos. Os programas escritos na linguagem Scilab são, portanto, normalmente executados em um tempo maior que os programas semelhantes escritos em linguagens compiláveis. Este fato é mais relevante quando precisamos desenvolver programas para a realização de simulações ou de otimizações. Nesses casos, pode ser conveniente escrever o código responsável pela lentidão do processamento em uma linguagem convencional (no caso, C ou FORTRAN) e rodar esse código dentro do ambiente Scilab.

Deve ser enfatizado, entretanto, que a vantagem na utilização do Scilab advém da facilidade de prototipação de programas e da disponibilidade de uma poderosa biblioteca de funções gráficas. Como sempre ocorre nessas situações, cabe ao usuário encontrar a sua solução de compromisso.

7.1. Comandos para Iterações

Existem dois comandos que permitem a realização de iterações, loops, no Scilab: o loop implementado com o comando *for* e o loop implementado com o comando *while*.

7.1.1.O Loop *for*

O comando for tem a forma geral:

```
For variavel = vetor_linha
    instrucao_1
    instrucao_2
    ... ...
    instrucao_n
    end
```

No ambiente Scilab, a forma acima é equivalente a:

```
-->for variavel=vetor_linha
-->     instrucao_1
-->     instrucao_2
-->     instrucao_n
-->end
```

Como mostrado no exemplo:

```
-->Loop for em linha de comando
-->for k = 1:3
-->     a = k + 1
-->end
a =
2.
a =
3.
a =
4.
```

Como é possível visualizar no exemplo, no loop *for*o comportamento das iterações é baseado no conteúdo do vetor linha.

Exercício: Elabore um programa para calcular o fatorial do valor digitado pelo usuário. E no final mostre, “O fatorial de ___ é = ___”. Obs.: Utilize os comandos *input* e *disp* ou *printf* para inserir o valor a ser calculado do fatorial e expor o resultado. Utilize o laço *for*.

7.1.2. O Loop *while*

O comando *while* tem a forma geral:

```
While condicao
  instrucao_1
  instrucao_2
    ... ...
  instrucao_n
    end
```

A forma acima é equivalente à forma:

```
-->while condicao
-->     instrucao_1
-->     instrucao_2
-->     instrucao_n
-->end
```

no ambiente Scilab.O loop baseado no *while* realiza uma sequência de instruções enquanto uma determinada condição estiver sendo satisfeita. A condição geralmente inclui comparações entre objetos. Na Tabela 7.1, são apresentados os operadores que permitem fazer comparações entre valores de objetos no Scilab.

Tabela 7.1 –Operadores condicionais.

Operadores	Significado
== ou =	igual a
<	menor do que
>	maior do que
<=	menor ou igual a
>=	maior ou igual a
<> ou ~=	diferente

A seguir, é apresentado um exemplo da utilização do loop baseado no comando *while*:

```
-->x = 1;
-->while x < 14
-->x = x * 2
-->end
x =
2.
x =
4.
x =
8.
x =
16.
```

Exercício: A série de Fibonacci se define como tendo os dois primeiros elementos iguais a um e cada elemento seguinte é igual a soma dos dois elementos imediatamente anteriores.

$$1,1,2,3,5,8,13, \ldots$$

Faça um programa que realize a série até determinado número fornecido pelo usuário.

7.2. Comandos Condicionais

O Scilab implementa dois tipos de comandos condicionais: *if-then-else* e *select-case*.

7.2.1. Comandos *if-then-else*

O comando *if-then-else* tem duas formas. Na forma mais simples, o comando é escrito como:

```
if condicao_1 then
sequencia_de_instrucoes_1
else
sequencia_de_instrucoes_2
end
```

Enquanto na sua forma mais geral o comando é escrito como:

```
if condicao_1 then
sequencia_de_instrucoes_1
elseif condicao_2
sequencia_de_instrucoes_2
... ... ... ... ... ...
elseifcondicao_n
sequencia_de_instrucoes_n
else
sequencia_de_instrucoes_n+1
end
```

A forma acima é equivalente à forma:

```
-->if condicao_1 then
```

```
-->     sequencia_de_instrucoes_1
-->elseif condicao_2
-->     sequencia_de_instrucoes_2
-->elseif condicao_n
-->     sequencia_de_instrucoes_n
-->else
-->     sequencia_de_instrucoes_n+1
-->end
```

no ambiente Scilab.A condicao_1 do comando *if-then-else* avalia uma expressão. Se esta expressão for verdadeira, true, será executada a instrução ou instruções subsequentes. Se for falsa, false, será executada a instrução ou instruções após o *else* ou o *elseif*, conforme o caso. Alguns exemplos da utilização do condicional *if-then-else*:

```
-->x = -1
x =
- 1.
-->if x < 0 then
-->     y = -x // apresenta a resposta
y =
1.
-->else
-->     y = x
-->end

-->// Outra forma
-->x = 1 // Inicializando
x =
1.
-->if x > 0 then, y = -x, else, y=x, end
y =
- 1.
-->x = -1
```

x =

- 1.

Exercício: Faça um programa que receba o peso e altura de uma pessoa e calcule o Índice de Massa Corporal (IMC) definido como:

$$IMC = Peso(Kg)/Altura(m)$$

O programa deve mostrar ao usuário a situação de Obesidade com a Tabela da Associação Brasileira para o Estudo da Obesidade.

Cálculo IMC	Situação
Abaixo de 18,5	Você está abaixo do peso ideal
Entre 18,5 e 24,9	Parabéns — você está em seu peso normal!
Entre 25,0 e 29,9	Você está acima de seu peso (sobrepeso)
Entre 30,0 e 34,9	Obesidade grau I
Entre 35,0 e 39,9	Obesidade grau II
40,0 e acima	Obesidade grau III

7.2.2. Comandos *select-case*

O condicional *select-case* tem a forma geral:

Select variavel_de_teste
case expressao_1
sequencia_de_instrucoes_1
case expressao_2
sequencia_de_instrucoes_2
... ...
Case expressao_n
sequencia_de_instrucoes_n
else
sequencia_de_instrucoes_n+1

end

A forma acima é equivalente à forma:

```
-->select variavel_de_teste
-->case expressao_1
-->     sequencia_de_instrucoes_1
-->case expressao_2
-->     sequencia_de_instrucoes_2
-->case expressao_n
-->     sequencia_de_instrucoes_n
-->else
-->     sequencia_de_instrucoes_n+1
-->end
```

O condicional *select-case* compara o valor de uma variável de teste com as várias expressões dos case. Serão executadas as instruções que possuírem o valor da expressão do case igual ao valor da variável de teste. Um exemplo de utilização do condicional *select-case*:

```
-->x = -1
x =
- 1.
-->select x
-->case 1
--> y = x + 5
-->case -1
--> y = sqrt(x)
y =
i
-->end

-->x = 1
x =
```

1.

```
-->select x, case 1, y = x+5, case -1, y = sqrt(x), end
y =
      6
```

Exercícios: Uma máquina de café automática normalmente é capaz de preparar café, cappuccino e chocolate quente. Faça um programa que pergunte ao cliente qual o seu pedido, informe quais ingredientes misturou e quando a operação terminar volte ao seu estado inicial. Caso o pedido não exista informe ao usuário.

Ingredientes:

- Café – Água, Café, Açúcar
- Capuccino – Água, Café, Leite e Açúcar
- Chocolate Quente – Água, Chocolate e Açúcar.

8. FUNÇÕES

Quando o tamanho de um programa estende-se a centenas de linhas, o programa torna-se difícil de compreender e administrar. Por isso, dividir um grande programa computacional em partes menores para facilitar a compreensão (ou legibilidade) do problema é umatarefa comum em programação de computadores. No Scilab, este trecho menor do programa é chamado de função. Funções são também chamadas de sub-rotinas, módulos, subprogramas ou subalgoritmos.

Funções são usadas também para evitar repetição do mesmo código no programa.Por exemplo, suponha que seu programa tenha a tarefa de por em ordem crescente várias listas de números. Em vez de repetir o código toda vez que for realizar esta tarefa,você escreve uma função para ordenar listas numéricas e depois chama a mesma função sempre que for ordenar uma lista. Neste sentido, as funções apresentam as seguintes vantagens:

a) Você escreve o código somente uma vez;

b) Você pode reutilizar a função em outros programas;

c) Uma vez que você tem corrigido todos os erros do programas (i.e.,depurado o programa), ele funcionara corretamente não importa quantas vezes você usea função.

Em resumo, funções são usadas para:

1. Dividir um grande programa em programas menores;
2. Repetir uma tarefa que é realizada frequentemente sem ter que repetir o mesmo código em vários lugares;
3. Aumentar a legibilidade do programa.

No Scilab já existem muitas funções prontas (pré-definidas), algumas delas elementares (*cos(x)* e *sin(x)*) e outras específicas para aplicações em engenharia, matemática,física, e na estatística. O objeto de estudo deste capítulo são as funções definidas pelo usuário. Isto é, funções que são elaboradas pelos próprios usuários do Scilab.

8.1. Funções definidas pelo usuário

A forma geral de uma função é:

```
function [y1,y2,...,ym] = nomedafuncao(x1,x2,x3,...,xn)

<comandos>...

endfunction
```

Onde,

`function`	Palavra reservada que indica o inicio de uma função.
`nomedafuncao`	o nome da função é definido pelo usuário.
`x1, x2, x3,..., xn`	parâmetros de entrada.
`y1, y2, y3,..., ym`	parâmetros de saída.
`<comandos>`	Comandos do Scilab a serem executados pela função.

Em que,

- *function*: Palavra reservada que indica o inicio de uma função.

- *nomedafuncao:* o nome da função é definido pelo usuário.
- *x1, x2, x3,..., xn*: parâmetros de entrada.
- *y1, y2, y3,..., ym*: parâmetros de saída.
- <*comandos*>: Comandos do Scilab a serem executados pela função.

A declaração:

function [y1,y2,...,ym] = nomedafuncao(x1,x2,x3,...,xn)

é o cabeçalho da função e serve, entre outras coisas, para dar o nome da função e definira lista de parâmetros de entrada e saída (também chamados de parâmetros formais).Quando há apenas um parâmetro de saída, os colchetes podem ser omitidos.

Cada parâmetro da lista de parâmetros de saída de uma função é necessário aparecer a esquerda de pelo menos um comando de atribuição da função.Quando a função termina, o valor contido nos parâmetros de saída é retornado ao programa que chamou a função.Uma chamada (ou ativação) de função é a solicitação explícita para executar uma função.

Exercícios:

1 – Altere o programa do Fatorial feito anteriormente para função.

2 – Faça uma função que é inserida como parâmetros de entrada 5 valores e nos parâmetros de saída o resultado da soma desses valores e da multiplicação.

9. GRÁFICOS

Neste capítulo são apresentados alguns comandos que podem ser utilizados para traçar gráficos bidimensionais e tridimensionais. Informações

mais detalhadas sobre todos os comandos disponíveis na biblioteca gráfica do Scilab podem ser acessadas através do navegador de help. A Tabela 9.1 mostra alguns desses comandos:

Tabela 9.1 –Comandos utilizados para manipulação de gráficos.

Comando	Descrição
plot	Gráfico linear
loglog	Gráfico em escala logaritmica
semilogx	Gráfico em escala semilog
semilogy	Gráfico em escala semilog
fill	Preenche polígonos 2D
polar	Gráfico em coordenada polar
bar	Gráfico em barras
stem	Gráfico de seqüência discreta
stairs	Gráfico em degraus
hist	Gráfico em forma de histograma
cdfplot	Gráfico em forma de distribuição de probabilidade

9.1. Gráficos Bidimensionais

Gráficos bidimensionais podem ser gerados através da utilização da função *plot*. A forma mais simples da função *plot* é:

plot([x],y)

em que x e y podem ser matrizes ou vetores reais. Os colchetes, [e], envolvendo x indicam que este parâmetro é opcional. Alguns pontos devem ser considerados sobre os parâmetros x e y:

1. Se x e y são vetores, a função *plot* permite traçar o gráfico de y em função de x. É importante observar que os dois vetores devem ter a mesma dimensão, isto é, os dois vetores devem ter o mesmo número de elementos;
2. Se x é um vetor e y é uma matriz, a função *plot* permite traçar o gráfico de cada coluna da matriz y em função do vetor x. Neste caso, o número de

elementos das colunas da matriz y deve ser igual ao número de elementos do vetor x;

3. Se x e y são matrizes, a função *plot* permite traçar o gráfico de cada coluna da matriz y em função de cada coluna da matriz x. Neste caso, as matrizes devem ter as mesmas dimensões;

4. Se y é um vetor, a função *plot* permite traçar o gráfico do vetor y em função do vetor [1:size(y)];

5. Se y é uma matriz, a função *plot* permite traçar o gráfico da matriz y em função do vetor [1:size(y)].

Existem alguns passos normalmente adotados para a criação de gráficos:

1 – Defina um vetor com nome “t” que varia de –pi a pi com passo de 0.01.

2 – Defina o vetor “y” igual ao seno de t.

3 – Gere o gráfico

4 – Crie grade no gráfico com o comando grid();

Nas figuras abaixo serão mostrados alguns exemplos com os códigos desenvolvidos e a forma de onda obtida.

```
clear;
clc;

t = [-%pi:0.01:%pi];
t1 = [3*%pi:0.01:5*%pi];
y = sin(t);
y1 = sin(2*t);
plot(t,y,'r',t,y1,'g',t1,y1,'c');
```

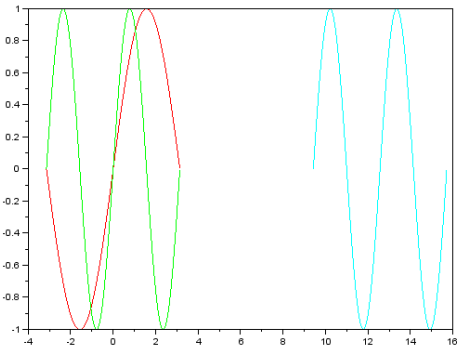

Figura 9.1 – Gráfico da função seno.

```
clear;
clc;

t = [-%pi:0.01:%pi];
t1 = [3*%pi:0.01:5*%pi];
y = sin(t);
y1 = sin(2*t);
plot(t,y,'r');
figure()
plot(t,y1,'g');
figure()
plot(t1,y1,'c');
```

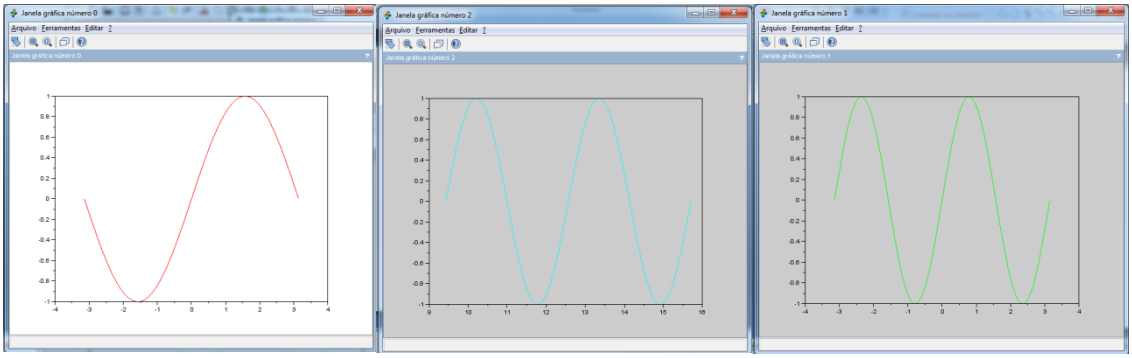

Figura 9.2 – Mudança de cores.

```
t = [-%pi:0.01:%pi];
t1 = [3*%pi:0.01:5*%pi];
y = sin(t);
y1 = sin(2*t);
y2 = sin(3*t);
y3 = sin(5*t);

subplot(2,2,1)
plot(t,y,'r');
subplot(2,2,2);
plot(t,y1,'g');
subplot(2,2,3);
plot(t1,y2,'c');
subplot(2,2,4);
plot(t1,y3,'c');
```

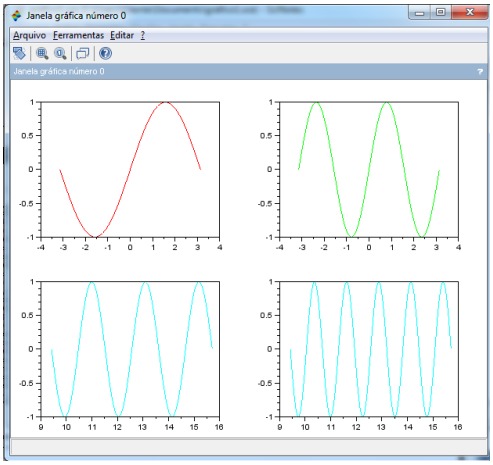

Figura 9.3 – Vários gráficos por janela.

9.2. Gráficos Tridimensionais

O comando *plot3d()* permite traçar gráficos de superfícies:

$$z = f(x, y)$$

Na notação Scilab, as variáveis independentes x e y são vetores de dimensões n1 e n2,respectivamente, e a variável dependente z é uma matriz

de dimensãon1xn2. Então, por essa definição, o elemento z(i,j) é o valor da superfície no ponto (x(i), y(j)).

Para exemplificar o uso de *plot3d()*, considere a função:

$$\cos(x) * \sin(y)$$

no intervalo [0, 2π] com incremento igual a 0.3. Logo:

```
-->x = [0:0.3:2*%pi]';
-->y = x;
-->z = cos(x) * sin(x');
-->plot3d(x, x, z)
```

gera o seguinte gráfico:

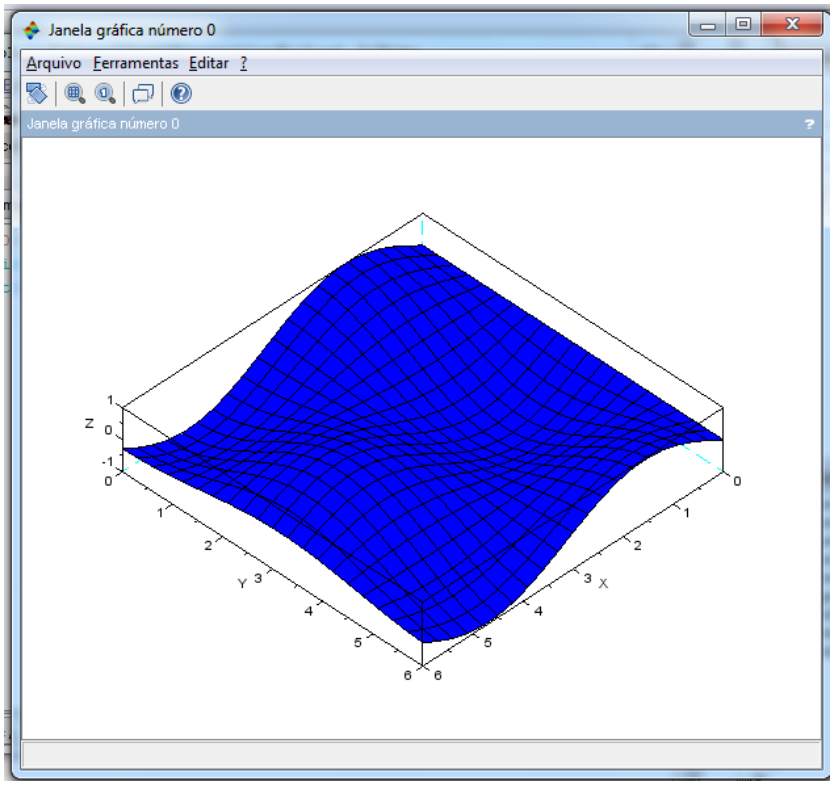

Figura 9.4 – Exemplo de gráfico 3-D.

Além da função *plot3d()* as seguintes funções permitem traçar gráficos tridimensionais especiais:

- param3d - permite traçar curvas paramétricas;
- hist3d - permite traçar histogramas 3-D;
- contour - permite traçar curvas de n´nível para uma função 3-D.

10. REFERÊNCIAS BIBLIOGRÁFICAS

[1] PIRES, Paulo Sérgio da Motta. Introdução ao Scilab Versão 3.0. disponível em http://www.dca.ufrn.br/~pmotta/sciport-3.0.pdf.

[2] LACERDA, E. G. M. Programando com Scilab. Departamento de Engenharia de Computação e Automação – UFRN. Disponível em http://www.dca.ufrn.br/~estefane/academica/progsci.pdf

[3] JÚNIOR, Vicente A. de Souza. Mini-Curso de Matlab. UFRN.

[4] DANUSIO, Gadelha Filho. Scilab 5.x. Universidade Federal do Ceará – UFC.