INSTITUTO DE MATEMÁTICA - UFRGS
MAT01169 - CÁLCULO NUMÉRICO
MATERIAL IMPRESSO PARA USO EM LABORATÓRIO
EXTRAIDO DE:

FUNDAMENTOS DE CÁLCULO NUMÉRICO EM SCILAB

1.2 Introdução a Scilab

Scilab é um poderoso ambiente computacional gráfico desenvolvido em INRIA (Institut National de Recherche en Informatique et Automatique), França. *Scilab* foi desenvolvido para aplicações em controle de sistemas e processamento de sinais. Atualmente, é distribuido gratuitamente (licensa GPL) em http://www.scilab.org, onde podem ser encontrados binários para instalação em variadas plataformas, como MS-Windows, Linux e FreeBSD. **Isso possibilita que você tenha *Scilab* em sua própria casa, sem custos de licenseamento !**

O principal atrativo de *Scilab* (depois, é claro, do fato de ser gratuito), é a fácil manipulação e visualização de matrizes de dados. Acrescente-se a isso a facilidade em definir novas funções, como muito convém em nossa disciplina de Cálculo Numérico.

Outro atrativo em *Scilab* é a possibilidade de interface com bibliotecas em Fortran ou C. Isso permite enormes flexibilidade e confiabilidade nas aplicações profissionais de Matemática Numérica.

1.2.1 Principais comandos do ambiente

Uma vez instalado e executado, *Scilab* mostrará o seguinte *prompt* (janela de comando):

$-->$

que então usaremos para fazer operações com variáveis pré-definidas e, opcionalmente, visualizar em janela gráfica. Nesta breve introdução, não está impressa a resposta que *Scilab* fornece para cada comando (ou sequência de comandos) dado, e então convém ao leitor executar *Scilab* e verificar por sí próprio o resultado dos exercícios que faremos a seguir.

O comando abaixo cria uma variável x e atribui a ela o valor 1 (independentemente de ela já existir ou não), que é impresso na tela:

$-->$ x = 1

O comando abaixo recria a variável x, atribuindo a ela o valor vetorial $\begin{bmatrix} 2 & 3 \end{bmatrix}$, mas suprimindo a impressão na tela (uso do ponto-e-vírgula ao final):

$-->$ x = [2 3];

Comentários em *Scilab* podem ser feitos usando barras duplas:

$-->$ x = exp(1); // definindo x = numero de Euler //

Tabela 1.1: Principais funções embutidas em Scilab

sqrt	power	poly	abs	sign	modulo
sin	cos	tan	cotg	sinh	cosh
asin	acos	atan	tanh	asinh	acosh
asinh	acosh	atanh	coth	real	imag
floor	ceil	round	conj	min	max
log	exp	log10	log2	mini	maxi

Tem fundamental importância o comando *help*, pois ele explica o uso (sintaxe) de todos os outros comandos. Por exemplo, o comando abaixo pede ajuda sobre o funcionamento da função *sqrt*, que calcula a raiz quadrada de um número dado.

$-->$ help sqrt

$-->$ x=-8; y = sqrt(x)

O exemplo acima também mostra que comandos podem dividir uma mesma linha, desde que separados por ponto-e-vírgula. A função *sqrt* acima é apenas uma das muitas que já estão definidas em *Scilab*. A tabela 1.1 apresenta as principais para Matemática Numérica. **Lembramos que em Scilab os cálculos são feitos em aritmética de números complexos, razão pela qual o usuário deve estar atento aos parâmetros fornecidos para algumas dessas funções.**

Temos ainda os operadores usuais $+, -, *, /, \wedge$, para adição, subtração, multiplicação e exponenciação, respectivamente.

Os comandos abaixo têm por objetivo avaliar a expressão

$$y = \frac{x}{1 + e^x \sqrt[3]{x}}$$

para $x = 4$, resultando $y = 0.0456261$.

$-->$ x=4; y= x/(1 + exp(x)*x∧(1/3))

Os comandos abaixo têm por objetivo avaliar a expressão

$$y = \frac{\sec(x)}{1 + \tan(x)^3}$$

para $x = \pi/6$, resultando $y = 0.9683429$.

$-->$ x=%pi/6; y= 1/(cos(x)*(1 + tan(x)**3))

Uma atenção especial deve ser dada à operação de exponenciação (operadores $\wedge$ ou **), pois tal operação é feita em aritmética de números complexos e pode resultar valores indesejados no caso de raízes (potências fracionárias em geral). **É de fundamental importância que o leitor revise seu entendimento sobre o tópico raízes, desenvolvido no Ensino Médio, para saber como contornar dificuldades que naturalmente aparecerão em ambientes computacionais para Matemática Numérica.** Os comandos abaixo exemplificam isso:

$-->$ x = -9; y = x∧(3/2)

−−> x = -8; y = x∧(5/3)

Explique por quê $y = -27i$ é a resposta esperada para o primeiro caso, mas , entretanto, $y = 16 - 27.712813i$ **não** é a resposta esperada para o segundo caso, apesar de ser uma das três respostas matematicamente corretas.

Outros exemplos de surpresas que *Scilab* proporciona:

−−> x = -1; y = log(x)

−−> x = 2; y = asin(x)

1.2.2 Definindo novas funções

Scilab proporciona um método muito simples e rápido de definir novas funções: através do comando *function* (use *help function* para saber mais detalhes).

Observe, nos comandos abaixo, como podemos resolver o problema de calcular raízes ímpares reais de números negativos, (no caso exemplificado, a raiz cúbica):

−−> function u = cbrt(x)

−−> u = sign(x)*abs(x)∧(1/3);

−−> endfunction

−−> x = -8; y = cbrt(x)∧5

−−> x = -8; y = x∧(5/3)

Em particular, o leitor é encorajado a entender uso e significado das funções sinal (*sign*) e valor absoluto (*abs*).

Observe, nos comandos abaixo (que é assumido que a função *cbrt* descrita acima esteja definida), como podemos definir e avaliar a função: $f : \mathbb{R}^2 \to \mathbb{R}$ definida por

$$f(t, y) = \frac{t}{1 + \sqrt[3]{y}}$$

para $t = 1.5$ e $y = -3.5$.

−−> function u=f(t,y)

−−> u = t/(1 + cbrt(y));

−−> endfunction

−−> f(1.5, -3.5)

Observe, nos comandos abaixo, como podemos definir a função: $F : \mathbb{R}^2 \to \mathbb{R}^2$ definida por

$$F\left(\left[\begin{array}{c} x \\ y \end{array}\right]\right) = \left[\begin{array}{c} x^3 + y \\ y + 1/x \end{array}\right]$$

−−> function u=F(vx)

−−> x = vx(1); y = vx(2);

−−> u = [x**3+y; y+1/x];

−−> endfunction

−−> x = [0.5; 2]

−−> y = F(x)

1.2.3 Repetição, vetores e operações vetoriais

Em *Scilab*, vetores terão suma importância na resolução de alguns problemas. Veremos a seguir exemplos de como definir e manipular vetores em *Scilab*. Um vetor linha tem suas componentes dispostas horizontalmente, ao passo que em um vetor coluna as componentes são dispostas verticalmente. A troca de um arranjo para outro é feita com o comando de transposição aspa simples: x' transpõe o vetor x.

Os comandos abaixo mostram como resolver a tarefa de montar um vetor com os 6 primeiros quadrados perfeitos. Para tal, usaremos o comando de repetição *for* (use *help for* para maior detalhamento sobre isso).

−−> x = zeros(1,6); // vetor linha com 6 elementos //

−−> for i=1:6

−−> x(i) = i∧2;

−−> end

−−> x

observe também que uma alternativa, que também será bastante usada, é

−−> function u = f(p)

−−> u = p*p;

−−> endfunction

−−> i = [1:1:6]; // inicia em 1, varia de 1 em 1, termina em 6 //

−−> x = feval(i,f); // aplica f a cada componente do vetor i // □

Os comandos abaixo montam uma tabela de valores $(x, \sqrt{x^2 + 1})$ para $x = 1,\ 1.1,\ 1.2, \ldots, 2.0$:

−−> function u = f(x)

−−> u = sqrt(x*x + 1);

−−> endfunction

−−> x=[1:0.1:2]'; // vetor de abscissas //

−−> y=feval(x,f); // vetor de ordenadas via feval //

−−> [x y]

Os comandos abaixo montam uma tabela de valores (x, y), onde

$$y = \frac{1}{1 + e^x}$$

para $x = 1,\ 1.1,\ 1.2, \ldots, 2.0$:

−−> function u = f(x)

−−> u = 1/(1 + exp(x));

−−> endfunction

−−> x=[1:0.1:2]'; // vetor de abscissas //

−−> y=feval(x,f); // vetor de ordenadas via feval //

−−> [x y]

1.2.4 Avaliando sequências ou recursões.

Uma tarefa numérica bastante comum é a de avaliar uma expressão que define uma sequência ou mesmo uma recursão.

Por exemplo, considere a tarefa de avaliar a soma de uma progressão geométrica de termo inicial unitário e razão q dada. Na forma de sequência temos:

$$a_n = 1 + q + q^2 + q^3 + \ldots + q^n$$

para $n = 1, 2, 3, \ldots, N$, onde N também é dado. Uma possibilidade, talvez a mais natural, é usar uma estrutura de dados do tipo array (vetor) e avaliar para cada valor de n entre 1 e N, observando que a expressão pode ser re-escrita como uma recursão: $a_1 = 1 + q$,

$$a_n = a_{n-1} + q^n, n = 2, 3, \ldots, N$$

e em Scilab, para $N = 20$ e $q = 0.6$, tal poderia ser feito:

−−> N = 20; q = 0.6; a = (1+q); // a=a(1)=(primeiro elemento) //

−−> for n=2:N; a(n) = a(n-1) + q∧n; end; a

Entretanto, como nosso objetivo é encontrar apenas a_N (para o último valor estabelecido para n), então estaríamos usando mais memória do que o necessário ao aplicar a estratégia acima, uma vez que a tarefa poderia ser cumprida SEM usar uma estrutura de dados, MAS através da atualização recursiva de 1 única variável, como mostra a estratégia abaixo:

−−> N = 20; q = 0.6; an = 1; // an é a variavel que sera atualizada //

−−> for n=1:N; an = an + q∧n; end; an

Mais um aperfeiçoamento pode ser feito: é possível evitar o uso da potenciação (essa é uma operação frequentemente considerada como em média 8 vezes mais demorada que qualquer das 4 operações aritméticas básicas). Para tal, re-escrevemos a sequência inicial como

$$a_n = 1 + q(1 + q + q^2 + q^3 + \ldots + q^{n-1})$$
$$a_n = 1 + q \cdot a_{n-1}$$

para $n = 1, 2, 3, \ldots, N$, desde que seja estabelecido que $a_0 = 1$. Em Scilab:

−−> N = 20; q = 0.6; an = 1; // variavel que sera atualizada //

−−> for n=1:N; an = 1 + q*an; end; an

Comparando com a primeira tentativa, essa última tem as vantagens de usar apenas operações básicas e de usar (alocar) apenas a quantidade mínima necessária de memória.

1.2.5 Plotando dados na janela gráfica

Scilab representa dados bi-dimensionais graficamente (plota) através do comando *plot2d*, e dados tri-dimensionais através do comando *plot3d*.

O comando que mais usaremos é o *plot2d*, no contexto de uma tabela ou gráfico de uma função. Mais detalhadamente, *Scilab* representa vetores de dados em uma janela gráfica, opcionalmente unindo os pontos por linhas, e então dando ao usuário a impressão de tratar-se da plotagem de uma curva. Para uma funcionalidade maior desse comando, que logo se fará necessária, o leitor é encorajado a usar ***help plot2d***.

Os comandos abaixo definem uma tabela (x, y), onde $y = x\cos(\pi x)$, para x no intervalo $[0, 1]$, e plotam na janela gráfica. É escolhido um conjunto de pontos igualmente espaçados por 0.1 unidades.

−−> function u = f(x)
−−> u = x*cos(%pi*x);
−−> endfunction
−−> x=[0:0.1:1]'; // criamos vetor de abscissas //
−−> y = feval(x,f); // feval monta vetor de ordenadas usando f(x) //
−−> [x y]
−−> plot2d(x,y,-1); // plota usando pontos //
−−> plot2d(x,y,1); // plota interligando por linha solida //

Os comandos abaixo mostram como plotar o gráfico da função raiz cúbica em *Scilab*:

−−> function y = cbrt(x)
−−> y = sign(x)*abs(x)**(1/3);
−−> endfunction
−−> x=[-2:.01:2]'; // vetor de abscissas //
−−> y=feval(x,cbrt); // vetor de ordenadas //
−−> plot2d(x,y,1); // linha solida //

conforme pode ser visto na Figura 1.1.

Os comandos abaixo montam uma tabela de valores (x, y), $y = e^{-x}/\sqrt{|x|}$, para $x \in [-2, 1]$. Observe que o ponto $x = 0$ é evitado, pela escolha de ponto inicial e espaçamento. Observe ainda que, por tratar-se de função descontínua, não tem sentido interligar os pontos por linhas.

−−> function u = f(x)
−−> u = exp(-x)/sqrt(abs(x));
−−> endfunction
−−> x=[-2.005:0.01:1.005]'; // vetor de abscissas //
−−> y=feval(x,f); // usamos feval para montar vetor de ordenadas //
−−> plot2d(x,y,-1)

conforme pode ser visto na Figura 1.2.

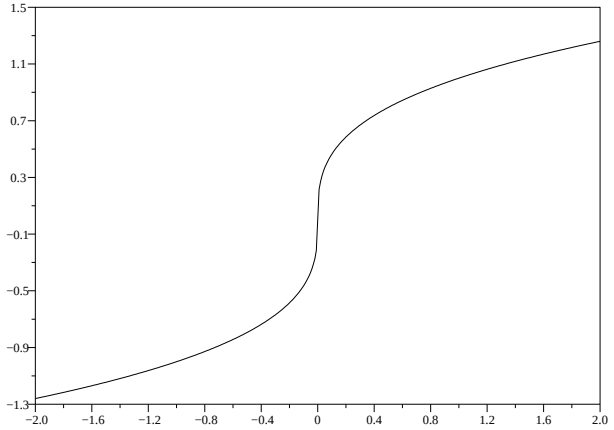

Figura 1.1: Plotagem da função raiz cúbica no intervalo [-1,1].

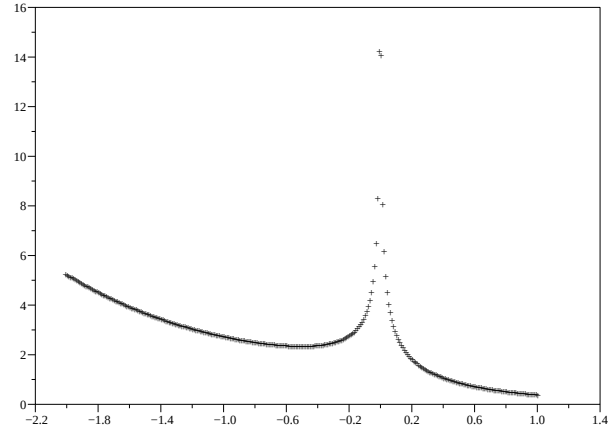

Figura 1.2: Plotagem da função $f(x) = e^{-x}/\sqrt{|x|}$ no intervalo [-2,1].