Primeiro roteiro de exercícios no Scilab
Cálculo Numérico

Rodrigo Fresneda

13 de fevereiro de 2012

Guia para respostas:

Responda a todas as questões que estão em negrito ao longo do roteiro. Inclua sempre que possível todos os comandos relevantes dados ao console do Scilab. Entregue todos os programas pedidos como arquivos separados do Scilab (*.sce).

Data limite para entrega: 27/02/2012

1 Números de ponto flutuante no Scilab

1.1 grandezas básicas

O Scilab trabalha com a representação de ponto flutuante de precisão dupla (64bits) de acordo com o padrão IEE754. Isto quer dizer, entre outras coisas, que:

base β	2
precisão p	$52+1$
menor expoente	-1022
maior expoente	1023
maior núm. normalizado	
menor núm. normalizado	
epsilon da máquina eps	

Tabela 1: alguns parâmetros iee754-64bits

Questão 1: preencha os valores omissos da tabela com os resultados obtidos da primeira lista de exercícios. (na folha de respostas, inclua a sequência de input no Scilab que foi necessária para obter cada um dos resultados).
Verifique seus resultados comparando com os valores obtidos a partir da função "*number_properties*":

–>xmin=number_properties("tiny")

xmin =
2.22D-308
–>xmax=number_properties(“huge”)
xmax =
1.79D+308

O épsilon da máquina é uma constante no Scilab:

–>%eps
%eps =
2.220D-16

1.2 Controlando a precisão no display: o comando *format*

Por padrão, o Scilab mostra números reais como sequências de 10 caracteres, incluindo sinal, ponto decimal, e expoente. No console, armazene a constante π (no Scilab, $\%pi$) na variável pi:

–>pi=%pi
pi =
3.1415927

Verifique que o Scilab mostra 10 caracteres, incluindo o ponto decimal e o sinal de + (subentendido). Em seguida, conte quantos caracteres aparecem em $-pi$:

–>-pi
ans = - 3.1415927

Se quisermos aumentar o número de casas decimais do resultado, podemos usar o comando *format.* Por exemplo, para obter mais 6 casas decimais, fazemos:

–>format(16);pi
pi =
3.1415926535898

Para retornar ao padrão, basta fazer

–>format(10)
–>pi
pi =
3.1415927

O maior número de caracteres que o Scilab pode mostrar é 25. Utilize o comando *format* para obter

–>pi
pi =
3.1415926535897931159980

O valor de pi correto até a vigésima-segunda casa é

$$\pi = 3.1415926535897932384626$$

Repare que o valor de π mostrado pelo Scilab só é correto até a décima-quinta casa. Em geral, o Scilab representa números na base 10 com 15 a 17 casas corretas, aproximadamente. Isto ocorre por que esta é aproximadamente a precisão obtida na representação de ponto flutuante de precisão dupla dada por $\%eps$.

Verifique que o erro relativo entre o valor exato (com 22 casas) e o valor produzido pelo Scilab é nulo:

```
->pi22=3.1415926535897932384626
pi22 =
3.1415926535897931159980
->abs(pi-pi22)/abs(pi22)
ans =
0.
```

Isto aconteceu por que a representação de ponto flutuante do valor exato, pi22, é igual à representação de ponto flutuante de %pi. Os dígitos errados ao final, 1159980, aparecem devido à erros na conversão da base dois à base 10 na saída.

1.3 Arredondando 0.1[1]

Considere a representação decimal de 0.1 no Scilab com 25 caracteres:

```
->format(25)
->x=0.1
x =
0.1000000000000000055511
```

Este resultado está correto até a décima-sétima casa decimal.

Questão 2:Obtenha a representação de ponto flutuante da questão 13 da lista 1, $x = M*2^{e-52}$, para o número 0.1 no Scilab. Dica: no Scilab, a função *floor* faz o papel do operador $\lfloor\cdot\rfloor$:

$$\lfloor log_2 |x| \rfloor \longleftrightarrow floor(log2(abs(x)))$$

Após determinar o significante inteiro M e o expoente e, compare seus resultados com aquele obtido no exercício 2b da lista 1.

Questão 3:Para fazer isto, converta M para a representação binária utilizando a função *dec2bin*. Por exemplo,

```
->x=dec2bin(75)
x =
1001011
```

Leia o help do Scilab para esta função, e note que a variável de entrada deve necessariamente ser um inteiro ou uma matriz de inteiros, e a saída é uma variável tipo "string".

Questão 4:Verifique que 0.1 está contido entre os números $(M-1)*2^{e-52}$ e $M*2^{e-52}$.

1.4 Operações aritméticas de ponto flutuante

No Scilab, == é o operador booleano que compara dois números ou duas matrizes de números para determinar se são distintos. Por exemplo, como $1 \neq 2$, então

```
->1==2
ans =
```

[1] Faça os exercícios 11, 12 e 13 da lista 1 antes de prosseguir.

F
Isto é, a saída é o booleano F (false), como esperado, ao passo que
–>1==1
ans =
T
e a saída é o booleano T (true).
Utilize o operador == para comparar as seguintes operações aritméticas de ponto flutuante:
–>0.9 / 3 * 3 == 0.9
–>(0.1 + 0.2) + 0.3 == 0.1 + (0.2 + 0.3)
–>(0.1 * 0.2) * 0.3 == 0.1 * (0.2 * 0.3)
–>0.3 * (0.1 + 0.2) == (0.3 * 0.1) + (0.3 * 0.2)

Questão 5: qual o valor de cada uma das sentenças acima? explique os resultados obtidos.

1.5 números extremos

Vimos que number_properties("huge") é o maior número normalizado representável:
–>number_properties("huge")
ans =
1.7976931348623157D+308
Verifique o que acontece se somarmos à décima-quinta casa desse número uma unidade:
–>number_properties("huge")+10^(-15+308)
No Scilab, existem números menores que o menor número normalizado, chamados de "denormalizados". O menor deles pode ser obtido usando a função
–>number_properties("tiniest")
ans =
4.94065645841246544D-324
Verifique o resultado de dividir esse número por dois.

Questão 6: o que ocorreu em cada uma das situações acima?

1.6 o épsilon da máquina

Na seção 1.1 você determinou qual o valor de eps para para representação de ponto flutuante utilizada pelo Scilab. Esse número em geral está relacionado com a distância relativa entre números normalizados vizinhos:
–>1+%eps==1
ans =
F
O resultado acima nos diz que 1+%eps é um número diferente de 1, mas
–>1+%eps/2==1
ans =
T

nos diz que 1+%eps/2 corresponde ao mesmo número de ponto flutuante que 1. A função *nearfloat* permite encontrar os números representáveis mais próximos de um dado número real. Por exemplo, o número representável que sucede $x = 1$ é

->x1=nearfloat("succ",1)
x1 =
1.0000000000000002220446

Questão 7: calcule a distância relativa entre x1 e 1, e interprete o resultado à luz das comparações feitas no início da subseção.

1.7 o editor de scripts do Scilab

Até aqui só utilizamos o console interativo do Scilab para executar comandos individuais. Agora vamos utilizar o editor de scripts do Scilab 5.3.3, o SciNotes. No menu do console, escolha Aplicativos>SciNotes. Uma nova janela irá se abrir. Nessa janela, copie a sequência de comandos abaixo, e em seguida salve o arquivo no seu disco rígido.

Algoritmo 1 sistema simples de ponto flutuante

```
emin=-2
emax=3
Mmin=4
Mmax=7
base=2
t=2
f = [];
for e = emax : -1 : emin
        for M = - Mmax : - Mmin
                f ( $ +1) = M * base ^( e - t);
        end
end
f ( $ +1) = 0;
for e = emin : emax
        for M = Mmin : Mmax
                f ( $ +1) = M * base ^( e - t);
        end
end
```

Este algoritmo calcula todos os números representáveis no sistema $F(base, t, -emin, emax)$ de acordo com a notação do exercício 12 da lista 1, i.e., os números representáveis são da forma $x = M * 2^{e-t}$, em que M e e são inteiros que satisfazem $2^t \leq |M| \leq 2^{t+1}$ e $emin \leq e \leq emax$. Estude o algoritmo com atenção e entenda seu funcionamento.

Agora execute-o (no menu do SciNotes, escolha Executar> "file with no echo"). No console, se não houve erro, você deve ler algo semelhante a

->exec('/caminho para arquivo/ep1.sce', -1)

–>

Todos os números representáveis foram guardados no vetor f, em ordem crescente. Para ver o conteúdo de f , basta executar f no console:

–>f
f =
- 14.
- 12.
- 10.
- 8.
etc.

No console, digite a sequência de comandos:

–>y=zeros(1,49);plot(f,y,'*')

Uma nova janela deve se abrir, com um gráfico que representa os números do sistema acima com o símbolo *. Repare como a densidade de pontos é maior próximo à origem.

Leia o help do Scilab para os comandos dados, e entenda seu funcionamento.

1.8 algoritmo para conversão de decimal para binário

Abaixo há o esqueleto de um algoritmo que converte um número inteiro positivo na base 10 para a base 2 de acordo com o algoritmo visto em sala de aula. Complete o algoritmo, e verique se funciona corretamente utilizando números conhecidos por você e/ou compare com a função *dec2bin* do Scilab.

Algoritmo 2 algoritmo para converter decimal inteiro para binário

```
function str = decbin(N) // decbin é função de uma variável
        Ni=N
        str="" // essa variavel é uma string
        while Ni>=1
                str=string(?????)+str //concatenação de strings
                Ni=?????
        end
endfunction
```