Representação de um real no sistema base-b

Projeto 1

6 de abril de 2009

O objetivo deste projeto é construir uma função no Scilab que realize a mudança de base de um número real.

A entrada é um número real (“double”) e a base para a representação. A saída é formada por dois vetores, um com os dígitos da parte inteira e outro com os da parte fracionária.

Sabemos que o algoritmo para determinar os dígitos de um número inteiro utiliza sucessivas divisões, enquanto que o algoritmo para determinar os dígitos de um número real positivo menor do que 1, utiliza sucessivas multiplicações. Assim, o primeiro passo consiste em isolar a parte inteira da parte fracionária.

A parte inteira de um número x representado em base-b possui a estrutura

$$[x] = d_n b^n + d_{n-1} b^{n-1} + \ldots + d_1 b + d_0,$$

onde $[x]$ representa a parte inteira de x e d_n, d_{n-1},..., d_0 são os dígitos da parte inteira na base b. O logaritmo desse número é tal que

$$\begin{aligned} \ln[x] &= \ln\left(b^n\left(d_n + d_{n-1}b^{-1} + \ldots + d_1 b^{-n+1} + d_0 b^{-n}\right)\right) \\ &= n\ln b + \ln\left(d_n + d_{n-1}b^{-1} + \ldots + d_1 b^{-n+1} + d_0 b^{-n}\right). \end{aligned}$$

Dividindo por $\ln b$,

$$\frac{\ln[x]}{\ln b} = \log_b[x] = n + \log_b\left(d_n + d_{n-1}b^{-1} + \ldots + d_1 b^{-n+1} + d_0 b^{-n}\right). \tag{1}$$

Necessariamente, $1 \leq d_n < b$, portanto

$$1 \leq d_n + d_{n-1}b^{-1} + \ldots + d_1 b^{-n+1} + d_0 b^{-n} < b$$

e consequentemente

$$0 \leq \log_b\left(d_n + d_{n-1}b^{-1} + \ldots + d_1 b^{-n+1} + d_0 b^{-n}\right) < 1,$$

ou seja, o segundo termo no lado direito de (1) é fracionário. Assim, tomando a parte inteira de $\log_b x$ temos n que é o número de dígitos, subtraído de uma unidade, na parte inteira de x :

$$[\log_b[x]] = n. \tag{2}$$

A expressão (3) será importante pois ela nos permite saber quantas operações de divisão serão necessárias para obter os dígitos da parte inteira de um número real x.

Voltamos anosa atenção par parte fracionária do número x, $x - [x]$. Sabemos que a representação da parte fracionária pode envolver infinitos dígitos não nulos. Dessa forma, será necessário implementar um critério de parada para o algoritmo. Também sabemos que a parte fracionária é armazenada na forma de um ponto flutuante de 64bits ("double") dos quais, 52bits são utilizados na representação do significando o que possibilita representar a parte fracionária do número em base 2 com 53 dígitos no total[1]. Portanto, qualquer informação contida a partir do 54º dígito é irremediavelmente perdida. Adiante veremos como esse fato motiva a adoção de um critério de parada para o algoritmo que cuida da parte fracionária.

O pior caso aconteceria se a representação binária da parte fracionária contiver dígitos não nulos a partir do 55º dígito (se o 54º dígito for não nulo, ocorre arredondamento no 53º)[2].

d_1	d_2	d_3	$\ldots$	d_{53}	0	d_{55}	$\ldots$

Nesse caso, a maior diferença seria $2^{-54} \times 2^p$, onde $p \leq 0$ é o expoente da representação binária da parte fracionária do número x. Assim, o número 2^{p-54} não poderia ser representado corretamente na base b e a sua representação (não normalizada)

$$0,000\ldots d_{-m}d_{-m-1}\ldots_b \times b^0$$

permite observar que apenas os $m-1$ primeiros termos do procedimento são confiáveis. O número m é determinado a partir da igualdade

$$2^{p-54} = d_{-m} \times b^{-m} + d_{-m-1} \times b^{-m-1} + \ldots \tag{3}$$

e de que a parte fracionária de x, dada por $x - [x]$, possui representação binária da forma[3]

$$0,1\ldots_2 \times 2^p.$$

Essa última expressão permite obter o expoente p:

$$p = \lfloor 1 + \log_2 (x - [x]) \rfloor \,,$$

[1] Lembre-se de que o primeiro dígito em um ponto flutuante normalizado deve ser maior do que 0. Como no sistema binário os únicos algarismos são 0 e 1, necessariamente o primeiro dígito será sempre 1 e portanto não há necessidade de se utilizar um dos registros para representá-lo.

[2] Por exemplo, o número $\frac{1}{2} + 2^{-54}$ possui a representação

$$0.50000000000000005551115123125782702118158340454101 5625.$$

Se esse valor for utilizado passado para uma variável no Scilab (testado na versão 5.1 para Linux x86-64), a mesma é armazenada como

$$0.5 \quad .$$

Já o valor $\frac{1}{2} + 2^{-53}$, que possui a representação

$$0.5000000000000001110223024625156540423631668090820 3125,$$

é armazenado exatamente.

[3] Se não for nula.

onde o símbolo $\lfloor\cdot\rfloor$ representa o inteiro arredondado no sentido $-\infty$ (por exemplo, $\lfloor -0.1\rfloor = -1$ e $\lfloor 0.1\rfloor = 0$). E a expressão (3), ou melhor, o seu logaritmo em base b, permite obter o valor de m:

$$\begin{aligned}(p-54)\log_b 2 &= \log_b\left(b^{-m}\left(d_{-m} + d_{-m-1}\times b^{-1} + \ldots\right)\right)\\ &= -m + \log_b\left(d_{-m} + d_{-m-1}\times b^{-1} + \ldots\right)\end{aligned}$$

Ou seja,

$$m = (54-p)\log_b 2 + \log_b\left(d_{-m} + d_{-m-1}\times b^{-1} + \ldots\right).$$

O segundo termo do lado direito da expressão anterior satisfaz a desigualdade

$$0 < \log_b\left(d_{-m} + d_{-m-1}\times b^{-1} + \ldots\right) < 1.$$

E esse termo é o que falta para tornar o número $(54-p)\log_b 2$ em um inteiro – m é um número inteiro. Assim, temos finalmente que

$$m = \lceil (54-p)\log_b 2\rceil\,,$$

ou ainda

$$m = \lceil (54 - \lfloor 1 + \log_2(x - [x])\rfloor)\log_b 2\rceil\,, \qquad (4)$$

onde $\lceil\cdot\rceil$ representa o inteiro arredondado no sentido $+\infty$ (por exemplo, $\lceil -0.1\rceil = 0$ e $\lceil 0.1\rceil = 1$).

As expressões (2) e (4), respectivamente, permitem calcular o número de dígitos da parte inteira de x na base b, dado por $n+1$, e o número de dígitos da parte fracionária de x na base b que podem ser obtidos através de cálculos com pontos flutuantes binários de 64bits, dado por $m-1$.