数値解析 44－3
(1992. 12. 4)

象限三角関数

浜田 穂積

電気通信大学 情報工学科

三角関数の引き数は，数学ではラジアン単位が合理的であるが，工学の応用上では事実上直角単位が便利であることが多いことを，中京大・二宮教授が強く主張している．たしかにその主張で言う，三角関数ルーチンがラジアン単位と直角単位の両方の入口を持つべきという点は妥当であるが，基になる近似式が直角単位で作られているからというのは妥当でない．この論文は，近似式の引き数はラジアン単位が良いこと，また上述の主張の間接的な根拠となっている，大なる引き数を近似式の適用範囲へ還元する操作で発生する誤差を抑えるための，手間の少い精度の良い還元方法を示すことによって，適切でない認識を正すことを目的とする．

Quadrant Trigonometric Functions

HAMADA, Hozumi

Department of Computer Science
University of Electro-Communications
Chofu, Tokyo 182, Japan

Prof. Ninomiya strongly asserts himself that trigonometric functions using the argument of quadrant unit are convenient at the most of applications. It is reasonable that these functions should have the entry for both quadrant and radian unit, but the assertion that all the base approximations use the argument of quadrant unit is not reasonable. In this paper, we show that the base approximation should use the argument of radian unit, and the efficient method avoiding error at reducing the argument to the fundamental interval.

1. はじめに

三角関数の引き数は，数学ではラジアンを単位とする，すなわち単位円上の弧の長さを用いるのが合理的であるが，工学の応用上では事実上直角を単位とするのが便利であることが多いことを，中京大・二宮教授が強く主張している [4]．たしかにそこで主張しているように，三角関数ルーチンがラジアン単位と直角単位の両方の入口を持つべきという点は妥当であるが，基になる近似式が直角単位で作られているからというのは妥当でない．ここでは，後者についての誤解を解きたいと考える．

ここでの焦点は近似式が仮定する引き数の適用範囲とともに，近似式を作成するうえでラジアンを単位とするか，直角を単位とするかということと，もう一つは効率よく精度の高い引き数の基本区間への還元方法であるので，それぞれについて述べる．

2. 周期関数の区間の還元

よく知られているように，sin x，cos x は周期 2π を持ち，tan x は周期 π を持つ周期関数である．しかしながらこれらの関数の近似式は区間の幅 $\pi/2$ で計算するのが好都合であることはよく知られている．ここでは一般に，周期を（この場合の $\pi/2$ のように，周期に準じる場合も含めて）p とする．問題は周期 p が無理数等，計算精度では正しく表わせない場合である．こうするとき

$$x = p \cdot n + \xi \qquad (1)$$

ただし，n は整数，$|\xi| \leqq p/2$

を満たす n と ξ を定める．sin x と cos x の場合 $p = \pi/2$ であり，n を 4 を法とする 4 つの剰余のグループに分けて，以下の表にしたがって ξ を引数とする関数の近似式を用いて計算する．

n mod 4	sin x	cos x
0	$\sin\xi$	$\cos\xi$
1	$\cos\xi$	$-\sin\xi$
2	$-\sin\xi$	$-\cos\xi$
3	$-\cos\xi$	$\sin\xi$

しかしながら，$\pi/2$ のように浮動小数点表現で正しく表わせない場合には，(1) 式における ξ をその計算の精度に見合って正しく求めるには，多倍長の除算を必要とし効率的な処理が不可能と考えられているように思える．そこでよく用いられる方法は次のようなものである．

$$x/p = n + \theta \qquad (2)$$

ただし，n は整数，$|\theta| \leqq 0.5$

を満たす n と θ を定める．具体的には，x を p で割り，(2 進法で) 商の小数点以下第 1 位を 0 捨 1 入したものを n とし，商から n を引いたものを θ とする．実はこれでも θ をその計算の精度に見合って正しく求めるには，多倍長の除算を必要とすることも分っているが，(1) より (2) の計算の方が易しいと考えられているようである．DEC 社 (Digital Equipment Corp.) は，(2) の計算を行なって n と θ を別々のレジスタに入れる命令を特許にしている（ソフトウェアで行なえば逃れられる）．(2) の両辺に p を掛けると

$$\xi = p \cdot \theta \qquad (3)$$

という関係が得られる．これを上の表に適用すると，$\sin\frac{\pi}{2}\theta$，$\cos\frac{\pi}{2}\theta$ 等を実質的に計算に用いることになる．

上に述べたこと，すなわち (1) より (2) による方が簡便であると考えられていることの真偽を検証しよう．方式 (2) を用いるとして，通常の浮動小数点除算を用いる場合，商 n が 0 の場合 θ は計算精度に見合って得られるが，それ以外の場合 n の表現に必要なビット数＋1 の 2 進桁数だけ少い精度しか得られない (n，θ の何れも符号を持つため +1 となる)．また，たまたま $|\theta|$ が非常に小さい場合，それ以上の桁落ちが発生するであろうことは容易に想像できる．これを補うために多倍長の除算を必要とする．DEC 社の除算命令では，θ の計算に精度の出るまで繰返し数を増せばよいことになるが，特許の文書では仮定されていない．一方，方式 (1) を用いる場合（このような命令は通常存在しないが，存在すると仮定すれば），ξ の精度は計算精度の分だけ自然に得られる．

次に除算命令の計算時間を検討する．除算の実

行は繰返し減算によるから，除算命令の計算時間は，引き放し法を用いる場合，加減算の繰返し回数，すなわち得るべき商の桁数に比例すると考えてよい．方式(2)による場合，IEEE倍精度システムでは商の桁数は64である．もし多倍長の除算を併用するなら，その数倍の時間を要する．もしDEC社の除算命令を用いてθをxの桁数まで求めると仮定すれば

$$>64+n\text{の桁数} \tag{4}$$

である．一方，方式(1)による場合は，単にnの桁数だけでよい．注目すべきは，データxの桁数には依らないということである．

ところで，何れの方式の場合も，pはxと同じ精度に丸めたものを用いるから，ξあるいはθはその丸め誤差×n程度の誤差を持つ．これが無視できないから補正したい場合を考える．方式(2)では除数もxの精度以上の精度を持つpで割る多倍長の除算による以外に方法はない．しかし方式(2)では以下に述べる簡便な方法がある．

除数pを

$$p=p_h+p_l \tag{5}$$

と2数に分解する．ここでp_hはpを計算精度に丸めたもので，p_lは$p-p_h$とする(計算に用いるときはやはりこれを計算精度に丸める)．p_lは，その浮動小数点の指数が，p_hの指数より計算精度の仮数の桁数以上小さい微小な数である．このとき，(1)の実質的除算を次の通りとする．

$$x=p_h\cdot n+\xi_h \tag{6}$$

(1)から(6)を引いて(5)を適用すると

$$\xi=\xi_h-p_l\cdot n \tag{7}$$

が得られる．この式において

$$|\xi_h|\gg|p_l\cdot n|$$

であるから，桁落ちは発生しない．具体的に言えば，より正確なξを得るために，(6)の除算を行い，そこで得られたnとξ_hを用いて(7)で補正する．これらの演算はすべてxと同じ計算精度によるのでよいから効率的である．

この簡便法には若干の懸念がある．(7)による補正の結果，(1)のξについての条件

$$|\xi|\leqq p/2$$

を満たさなくなる場合が極くまれに起こり得る．しかしこれは心配する必要がない．なぜなら，ξの範囲が拡大することを織り込んで，その後で用いる近似式のパラメタの適用範囲を広くしておけばよい．範囲の拡大は極くわずかであるから，最良近似式の最大誤差はほとんど増大しない(少くとも項数が増すことはない)．

(1)あるいは(6)を実行する除算命令を備えた計算機はほとんどない．著者は十数年前に，DEC社の命令が使用できないので，よい対策法はないかと相談を受けたことがある．そこでこの仕様を持つ命令を提案し(1979年)，日立製作所のある機種で採用された．DEC社は多くの特許を所有している訳ではないが，自社の所有する特許は決して他社に使用させない，ということは知る人ぞ知る存在である．またこの命令についてはかつて報告したことがある[2]．なおそれ以外にも，浮動小数点演算の結果を出す際になるべく情報を失わない様にという観点で，除算の結果として剰余も返す命令を用意すべきだという提案をする論文もある[3]．

ところで，上記の命令を備えない計算機では方式(2)を用いる以外に方法がないかと言えば，そうではない．ξ_hを正しく求めるには，nを得た後で

$$\xi_h=x-p_h\cdot n \tag{8}$$

とする．ただし，$p_h\cdot n$が正しくなければならないので，条件をつける必要がある．すなわち，nとして許すことのできる値の範囲をあらかじめ定める．たとえば

$$-255\leqq n\leqq 255$$

としたとする．このときnは2進8桁で表わせるので，p_hを得るための丸めを，計算精度から8桁引いた桁数を得る位置でする．そのようにしてもp_lによる補正が十分すぎる程効果を与えてくれる．しかし，nの可能な桁数が多くて剰余ξの範囲の拡大が許容できない場合は，上記の還元過程をξについてもう一度行ない，得られた整数商を前段の還元過程で得られた商に加えればよい．このようにしても，長精度の除数による多倍長除

算をまともに行なうのと比べれば効率的である．

3．近似式の引き数

工学分野の数値計算では，$\sin\frac{\pi}{2}x$，$\cos\frac{\pi}{2}x$ という形の公式で用いられることが多いのは確かである．この場合の x は $\frac{\pi}{2}$ すなわち直角，あるいは四分円を単位にして測ったものであるといえる．そこで直接これらの関数値を求める数学関数ルーチンを用意すべきという主張 [4] は納得できる．四分円 (Quadrant) を単位にして測った sin とか cos と言う意味で SINQ とか COSQ という入口名でライブラリに登録するのは妥当である．現に著者の提案で，日立製作所の汎用大型計算機のための数学関数ライブラリには用意されている．

とはいえ，これらの関数を用意したいという発想にはさらにいくつかの動機がある．一つには還元が高速で誤差も少いことである．もう一つの動機は，前章で言う方式 (2) しか還元法がないと仮定すれば，近似式は $\sin\frac{\pi}{2}\theta$，$\cos\frac{\pi}{2}\theta$ のためのもの (以下 Q 型近似式と呼ぶ) を用いるのが自然であるから，入口としての SINQ とか COSQ がないことはむしろバランスを欠いていると考えるからであろう．SINQ，COSQ での還元過程は簡単で

$$x=n+z \quad n\text{は整数}, |z|\leqq 0.5 \qquad (9)$$

を満たす n と z を定めればよい．

しかしながら，還元後の狭い区間に於ける関数値の計算に Q 型近似式を用いることには納得できないものがある．近似値の計算にはいくつかの方法がある．平方根の計算法に用いられる非線形方程式の解法によるものもあるし，CORDIC(Coordinate Rotation Digital Computer) の様にやや特殊な方法もあるが，もっとも一般的なのは近似式による方法であろう．

まず CORDIC から検討する．CORDIC の計算法の解説書で，Q 型近似式に対応する説明をしているものはない．しかし，その中で用いられる定数 $\tan^{-1}2^{-i}$ を $\pi/2$ で割ったものを用いれば，原理的には可能である．しかし，x が小さいとき

$$\sin x\fallingdotseq x,\ \cos x\fallingdotseq 1 \qquad (10)$$

という事実を用いた半止法 [1] を用いることができない．小さい引数が残ったときそれに $\pi/2$ を掛けなければならないが，それは処理時間にとって多大な負担となる．CORDIC は所詮ラジアンを単位とする引数のためのものと言える．

式 (10) が効果的に使えるという意味では近似式も同様である．ラジアンを単位とする場合を考える．近似式は次のようである．

$$\sin x\sim a_0x+a_1x^3+a_2x^5+\cdots$$

最良近似式の場合は $a_0\fallingdotseq 1$ である．この式の右辺は次のように計算する．

$$((\cdots+a_1)\times(x^2)+(a_0-1))\times x+x$$

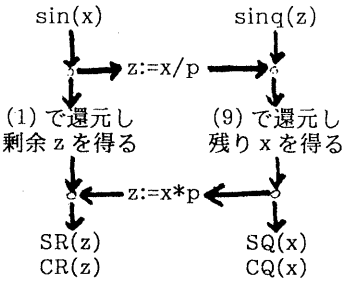
x の値が小さいときは，Taylor 展開の場合も，最良近似式の場合も，上の式の最後の加算の第 1 オペランドは，その指数が第 2 オペランドのそれと比べてかなり小のため無視されて x となる．したがって，(10) の条件を自然に満たしてくれる．

一方，Q 型近似式では，上のように $a_0\fallingdotseq 1$ という訳にはいかない．最悪の場合は，sin x を Q 型近似式を用いて計算する場合である．x を還元過程で $\pi/2$ で割って，近似式で $\pi/2$ で掛けるから，x の値が小さいとき常に x になるということはとても保証できない．誤差に関して万全を期したいときは，Q 型近似式を使うことはできない．

指数関数，対数関数についても同様で，近似式は e^x-1，$\log_e(1+x)$ の形のものを使うべきである．Intel 社の i8087 浮動小数点 coprocessor のように，基本区間における関数値を求める機能として，底に 2 を使うのは望ましくない．

4．区間の還元と変数変換

プログラマの用いる数学関数ライブラリの入口名を，ラジアン (Radian) 単位の引数の方を sin，cos とし，四分円 (Quadrant) 単位の引数の方を sinq，cosq とする．また，基本区間における近似式を同様に SR，CR および SQ，CQ とする．区間の還元法は (1) あるいは (9) を用いるものとする．そこで，区間の還元と変数変換を組合せて，全体としての関数値計算法を図示すれば以下の通りとなろう．関数は sin の場合を示す．また $\frac{\pi}{2}$ を p で表わす．



この図で分るように，SR，CR，SQ，CQ のすべてを用意できれば，左の列，右の列のそれぞれ縦に下に降りればよいが，SR，CR の組あるいは SQ，CQ の組だけで済ませたいときは，左の列から右の列，あるいはその逆への変数変換を必要とする．そのとき，図に記入した矢印の方向のみに進み得るとした場所での変換を行なう．

5. 関数ルーチンの構成プログラム例

前の章で言及した関数の処理のアルゴリズムを具体的に示すため，SR，CR の組だけを用意するとして，Pascal によるプログラムを以下に示す．

```
const p=π/2;

function quot(var x:real;p:real):integer;
{還元法(1)を実行する関数．剰余をxに入れて，整数商を関数値として返す．}
  var n:integer;
begin n:=round(x/p); x:=x-p*n; quot:=n end;

function sin(x:real):real;
   var n:integer; z:real;
begin n:=quot(x,p)mod 4+4;
  if odd(n) then z:=CR(x) else z:=SR(x);
  n:=n div 2;
  if odd(n) then sin:=-z else sin:=z end;

function cos(x:real):real;
   var n:integer; z:real;
begin n:=quot(x,p)mod 4+5;
  if odd(n) then z:=CR(x) else z:=SR(x);
  n:=n div 2;
  if odd(n) then cos:=-z else cos:=z end;

function sinq(x:real):real;
   var n:integer; z:real;
begin n:=round(x); x:=(x-n)*p; n:=n mod 4+4
  if odd(n) then z:=CR(x) else z:=SR(x);
  n:=n div 2;
  if odd(n) then sinq:=-z else sinq:=z end;

function cosq(x:real):real;
   var n:integer; z:real;
begin n:=round(x); x:=(x-n)*p; n:=n mod 4+5
  if odd(n) then z:=CR(x) else z:=SR(x);
  n:=n div 2;
  if odd(n) then cosq:=-z else cosq:=z end;
```

6. おわりに

これまで述べたことをまとめると，以下の結論を得る．

a) 象限三角関数 SINQ，COSQ を，数学関数ライブラリの入口名として用意するのは妥当である．

b) 基本区間における近似式の引数としてはラジアン単位の引数のものを用いる方が望ましい．

c) これらの点は，指数関数，対数関数の場合も同様である．

d) 基本区間への還元操作は，無理数の周期でも簡単で，方式(1)を用いる方が，精度，処理時

間の両点でむしろ望ましい.

e) 式(1)の処理を行なう機械命令を，計算機に用意させることが望ましい.

参考文献

[1] 一松信：初等関数の数値計算. 教育出版 (1974), p.235

[2] 浜田穂積：初等関数ルーチンの一構成法. 情報処理学会第31回全国大会講演予稿集 (Sep. 1985), pp.1435-1436

[3] Bohlender,G., Matula,D.W. et al.: Semantics for Exact Floating Point Operations, Proc. of 10th Symp. on Comp. Arith., IEEE, (June 1991), pp.22-26

[4] 二宮市三：象限三角関数. 日本応用数理学会平成4年度年会研究発表予稿集 (Oct. 1992), pp.65-66