Some notes on the possible under/overflow of the most common elementary functions

Jean-Michel Muller, Vincent Lefèvre

Some notes on the possible under/overflow of the most common elementary functions

Vincent Lefèvre and Jean-Michel Muller
Laboratoire LIP, Projet Arénaire
CNRS, Ecole Normale Supérieure de Lyon,
INRIA and Université Claude Bernard Lyon 1

May 25, 2007

Abstract

The purpose of this short note is not to describe when underflow or overflow must be signalled (it is quite clear that the rules are the same as for the basic arithmetic operations). We just want to show that for some of the most common functions and floating-point formats, in many cases, we can know in advance that the results will always lie in the range of the numbers that are representable by normal FP numbers, so that in these cases there is no need to worry about underflow or overflow. Note that when it is not the case, an implementation is still possible using a run-time test.

1 When the input argument is not near zero

The most tricky part is with the direct trigonometric functions of large arguments. Can a floating-point number be so close to an integer multiple of $\pi/2$ that its sine or cosine is smaller than the smallest normal FP number of the format being considered, or that its tangent is larger than the largest finite FP number?

A method based on the continued-fraction theory allows to find, in a given format, the FP number that is closest to a multiple of a given irrational constant. That method was first suggested by Kahan [1] for finding the precision with which range-reduction of trigonometric functions should be carried-on (a C-program that implements that method without needing extra-precision arithmetic was written by Kahan and McDonald). A similar method was found later by Smith [3]. That method is presented in Muller's book [2]. A good reference on continued fractions is Stark's book [4].

In the next sections, I give the results obtained for the formats defined by the new draft standard. I also include a Maple program that allows one to find the number (not near zero, of course) closest to a multiple of $\pi/2$ for any FP format.

1.1 Binary32 format

The FP number greater than 2^{-20} in the Binary32 format that is closest to an integer multiple of $\pi/2$ is

$$M_{Bin32} = 16367173 \times 2^{72}$$

we have,

$$\begin{aligned} \epsilon &= |M_{Bin32} - 492054812429041478249228356 05 \times \pi/2| \\ &= 1.6147697982476211\cdots \times 10^{-9} \approx 2^{-29.206} \end{aligned}$$

This means that if x is a Binary32 number of absolute value greater than 2^{-20}, then

- $|\sin(x)|$ or $|\cos(x)|$ will never be less than $\sin(\epsilon) \approx 2^{-29.206}$,
- $|\tan(x)|$ will never be less than $\tan(\epsilon) \approx 2^{-29.206}$ (same for the cotangent function),
- $|\tan(x)|$ will never be larger than $\tan(\pi/2-\epsilon) = 619283318.95061386\cdots \approx 2^{+29.206}$ (same for the cotangent function).

These values are representable as normal numbers in the Binary32 format. As a consequence, **the sine, cosine, tangent, cotangent, secant, cosecant of a Binary32 number not near zero cannot underflow nor overflow**.

1.2 Binary64 format

The FP number greater than 2^{-20} in the Binary64 format that is closest to an integer multiple of $\pi/2$ is

$$M_{Bin64} = 6381956970095103 \times 2^{797}$$

Its distance to the nearest integer multiple of $\pi/2$ is

$$\epsilon = 4.68716592425462761 11\cdots \times 10^{-19} \approx 2^{-60.8879}$$

This means that if x is a Binary64 number of absolute value greater than 2^{-20}, then

- $|\sin(x)|$ or $|\cos(x)|$ will never be less than $\sin(\epsilon) \approx 2^{-60.8879}$,
- $|\tan(x)|$ will never be less than $\tan(\epsilon) \approx 2^{-60.8879}$ (same for the cotangent function),
- $|\tan(x)|$ will never be larger than $\tan(\pi/2 - \epsilon) \approx 2^{+60.8879}$ (same for the cotangent function).

These values are representable as normal numbers in the Binary64 format. As a consequence, **the sine, cosine, tangent, cotangent, secant, cosecant of a Binary64 number not near zero cannot underflow nor overflow**.

1.3 Binary128 format

The FP number greater than 2^{-20} in the Binary128 format that is closest to an integer multiple of $\pi/2$ is

$$M_{Bin128} = 87948731350338293497021849247226 39 \times 2^{1852}$$

Its distance to the nearest integer multiple of $\pi/2$ is

$$\epsilon = 7.88136000827235 \times 10^{-38} \approx 2^{-123.2548}$$

This means that if x is a Binary128 number of absolute value greater than 2^{-20}, then

- $|\sin(x)|$ or $|\cos(x)|$ will never be less than $\sin(\epsilon) \approx 2^{-123.2548}$,
- $|\tan(x)|$ will never be less than $\tan(\epsilon) \approx 2^{-123.2548}$ (same for the cotangent function),
- $|\tan(x)|$ will never be larger than $\tan(\pi/2 - \epsilon) = 1.2688\cdots \times 10^{37} \approx 2^{+123.2548}$ (same for the cotangent function).

These values are representable as normal numbers in the Binary128 format. As a consequence, **the sine, cosine, tangent, cotangent, secant, cosecant of a Binary128 number not near zero cannot underflow nor overflow**.

1.4 Decimal32 format

The FP number greater than 2^{-20} in the Decimal32 format that is closest to an integer multiple of $\pi/2$ is

$$M_{Dec32} = 4327189 \times 10^{42}$$

we have,

$$\begin{aligned} \epsilon &= M_{Dec32} \\ &\quad -2754774076171501944361681452576588198950125599968 \times \pi/2 \\ &= 1.8908070677421251636\cdots \times 10^{-10} \end{aligned}$$

This means that if x is a Decimal32 number of absolute value greater than 2^{-20}, then

- $|\sin(x)|$ or $|\cos(x)|$ will never be less than $\sin(\epsilon) \approx 1.89\cdots \times 10^{-10}$,
- $|\tan(x)|$ will never be less than $\tan(\epsilon) \approx 1.89\cdots \times 10^{-10}$ (same for the cotangent function),
- $|\tan(x)|$ will never be larger than $\tan(\pi/2 - \epsilon) \approx 5288746890$ (same for the cotangent function).

These values are representable as normal numbers in the Decimal32 format. As a consequence, **the sine, cosine, tangent, cotangent, secant, cosecant of a Decimal32 number not near zero cannot underflow nor overflow**.

1.5 Decimal64 format

The FP number greater than 2^{-20} in the Decimal64 format that is closest to an integer multiple of $\pi/2$ is

$$M_{Dec64} = 8919302781369317 \times 10^{296}$$

Its distance to the nearest integer multiple of $\pi/2$ is

$$\epsilon = 6.05527439099688\cdots \times 10^{-20}$$

This means that if x is a Decimal64 number of absolute value greater than 2^{-20}, then

- $|\sin(x)|$ or $|\cos(x)|$ will never be less than $\sin(\epsilon) \approx 6.05 \times 10^{-20}$,
- $|\tan(x)|$ will never be less than $\tan(\epsilon) \approx 6.05 \times 10^{-20}$ (same for the cotangent function),
- $|\tan(x)|$ will never be larger than $\tan(\pi/2 - \epsilon) \approx 1.65 \times 10^{19}$ (same for the cotangent function).

These values are representable as normal numbers in the Decimal64 format. As a consequence, **the sine, cosine, tangent, cotangent, secant, cosecant of a Decimal64 number not near zero cannot underflow nor overflow**.

1.6 Decimal128 format

The FP number greater than 2^{-20} in the Decimal128 format that is closest to an integer multiple of $\pi/2$ is

$$M_{Dec128} = 9308532438209917461067659354862169 \times 10^{4639}.$$

Its distance to the nearest integer multiple of $\pi/2$ is

$$\epsilon = 2.06901398954\cdots \times 10^{-38}$$

This means that if x is a Decimal128 number of absolute value greater than 2^{-20}, then

- $|\sin(x)|$ or $|\cos(x)|$ will never be less than $\sin(\epsilon) \approx 2.069 \times 10^{-38}$,
- $|\tan(x)|$ will never be less than $\tan(\epsilon) \approx 2.069 \times 10^{-38}$ (same for the cotangent function),
- $|\tan(x)|$ will never be larger than $\tan(\pi/2 - \epsilon) \approx 4.83\cdots \times 10^{37}$ (same for the cotangent function).

These values are representable as normal numbers in the Decimal128 format. As a consequence, **the sine, cosine, tangent, cotangent, secant, cosecant of a Decimal128 number not near zero cannot underflow nor overflow**.

1.7 Maple program

```
worstcases := proc(radix,precision,emin,emax)
 local DigitsRadix,epsilonmin,powerofradixoverC,e,a,Plast,r,Qlast,
       Q,P,NewQ,NewP,epsilon,
     numbermin,expmin,l;
     epsilonmin := 12345.0 ;
     DigitsRadix := (emax+precision+20);
     Digits := ceil(evalf(DigitsRadix*log(radix)/log(10)));
     printf("Intermediate precision: %a decimal digits\n",Digits);
     powerofradixoverC := 2*radix^(emin-precision)/Pi;
     for e from emin-precision+1 to emax-precision+1 do
         powerofradixoverC := radix*powerofradixoverC;
         a := floor(powerofradixoverC);
         Plast := a;
         r := 1/(powerofradixoverC-a);
         a := floor(r);
         Qlast := 1;
         Q := a;
         P := Plast*a+1;
         while Q < radix^precision-1 do
             r := simplify(1/(r-a));
             a := floor(evalf(r));
             NewQ := Q*a+Qlast;
             NewP := P*a+Plast;
             Qlast := Q;
             Plast := P;
             Q := NewQ;
             P := NewP
         od;
         epsilon :=
             evalf((Pi/2)*abs(Plast-Qlast*powerofradixoverC));
         if (e+precision-1) mod 50 = 0 then printf("I am processing exponents %a to %a
                                          \n",e+precision-1,min(e+precision+48,emax)) fi;
         if epsilon < epsilonmin then
             epsilonmin := epsilon; numbermin := Qlast;
             expmin := e
         fi
     od;
     printf("The FP number closest to a multiple of Pi/2 is %a * %a ^ %a\n",numbermin,
                                                              radix, expmin);
     printf("Its distance to a multiple of Pi/2 is %a\n",evalf(epsilonmin,15));
     l := evalf(log(epsilonmin)/log(radix),10);
     printf("which is %a ^ %a\n",radix,l)
 end;
```

2 When the input argument is near zero

First, with the usual functions if the value of the function at zero is a nonzero representable number, there is no problem[1]. So we have to deal with two cases.

[1] One could argue that the value of the function at zero could be nonzero yet close to the underflow threshold: we remind here that we are not giving "general" rules, we are just dealing with the most common functions.

2.1 The function is null at zero

A frequent case with the common elementary functions is the case where the derivative of the function at zero is 1. Let us call f the function.

In many cases ($\log(1+x)$ for $x<0$, $\tan(x)$, $\arcsin(x)$, $\sinh(x)$, $\tanh^{-1}(x)$), $|f(x)|$ is (very slightly) larger than $|x|$ for x near zero. This means that near zero these functions shall not return a subnormal number (hence, we shall not signal underflow) if x is not a subnormal number. Still with the same functions, if x is a subnormal number, with any "reasonable" format (see below for an explanation of what is a "reasonable" format):

- in round-to-nearest, roundTowardsZero, roundTowardsPositive (for $x<0$) and roundTowardsNegative (for $x>0$) modes, the correctly rounded value of $f(x)$ will be a subnormal number;
- in roundTowardsPositive (for $x>0$) and roundTowardsNegative (for $x<0$) modes, $f(x)$ will be a subnormal number, unless $|x|$ is the largest subnormal number (i.e., $b^{emin}-b^{emin+1-p}$, where b is the base and p the precision). In that last case, the correctly rounded value of $f(x)$ will be $\text{sign}(x)$ times b^{emin} (hence, the correctly rounded value of $f(x)$ will be a normal number).

Now, when $|f(x)|$ is (very slightly) less than $|x|$ for x near zero (which happens for functions $\log(1+x)$ for $x<0$, $\sin(x)\arctan(x)$, $\sinh^{-1}(x)$, $\tanh(x)$), the situation is symmetrical. If x is a subnormal number, then the correctly rounded value of $f(x)$ is a subnormal (or zero) number too. Still with the same functions, if x is a normal number, then with any "reasonable" format:

- in round-to-nearest, roundTowardsPositive (for $x>0$) and roundTowardsNegative (for $x<0$) modes, the correctly rounded value of $f(x)$ will be a normal number too, so we shall not signal underflow;
- in roundTowardsZero, roundTowardsPositive (for $x<0$) and roundTowardsNegative (for $x>0$) modes, the correctly rounded value of $f(x)$ will be a normal number, unless $|x|$ is the smallest normal number b^{emin}. In that case, the correctly rounded value of $f(x)$ is $\text{sign}(x)$ times the largest subnormal number (ie., $\text{sign}(x)$ times $b^{emin}-b^{emin+1-p}$).

Now, let us explain with an example what we mean by a "reasonable" format. Consider function `arctan`, and rounding towards zero, with base b and precision p. We know that when computing $\arctan(b^{emin})$ (reminder: b^{emin} is the smallest normal number), we should return a subnormal number (hence, we have an underflow). Can this happen with larger input numbers? Let us try to see at which condition we are sure that $\arctan(nextUP(b^{emin}))$ is larger than b^{emin}. From the power series

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots$$

we know that

$$\arctan(nextUP(b^{emin})) \geq b^{emin}(1 + b^{-p+1}) - \frac{1}{3}b^{3emin}(1 + b^{-p+1})^3.$$

This will be larger than b^{emin} as soon as

$$b^{2emin+p-1}(1 + b^{-p+1})^3 \leq 3.$$

This condition is satisfied by far by all formats of IEEE-754 and the current draft format. For instance, with the Binary64 format, $b^{2emin+p-1}(1+b^{-p+1})^3 \approx 2.23\times 10^{-600}$, and with the Decimal128 format $b^{2emin+p-1}(1 + b^{-p+1})^3 \approx 10^{-12253}$.

2.2 The function goes to infinity at zero

Unless $\log(b)|emin + 1 - p| > b^{emax}(b - b^{1-p})$ (which would require a pretty stupid FP format), the absolute value of the logarithm of a positive FP number is much smaller than the overflow threshold.

Since $\csc(x) \sim 1/x$ near zero, it can overflow (the study is very similar to that of the previous section).

References

[1] W. Kahan. Minimizing q*m-n. text accessible electronically at `http://http.cs.berkeley.edu/~wkahan/`. At the beginning of the file "nearpi.c", 1983.

[2] Jean-Michel Muller. *Elementary Functions, Algorithms and Implementation.* Birkhäuser Boston, 2nd edition, 2006.

[3] R. A. Smith. A continued-fraction analysis of trigonometric argument reduction. *IEEE Transactions on Computers*, 44(11):1348–1351, November 1995.

[4] H. M. Stark. *An Introduction to Number Theory.* MIT Press, Cambridge, MA, 1981.