A Note on Finding Difficult Values to Evaluate Numerically

Gaston H. Gonnet
ETH Informatik
Zurich, Switzerland

September 1, 2002

Abstract

This is an application of the LLL lattice reduction algorithm to find values which are difficult to evaluate correctly for a particular function $f(x)$. The difficulty comes from the fact that $f(x)$ will lie very close to 1/2 ulp (unit in the last place) and hence to get the correct rounding the computation may require extra precision.

1 Introduction

We want to find a value x, close to a value y, such that $f(x)$ when evaluated to a certain precision is as close as possible to 1/2 ulp. More formally, we define the following terms. Let B be the base at which we are representing our values. A value y is approximated by a floating point number in base B. Let M be the number of digits (base B) in the representation. Let $mant(x)$ and $expo(x)$ be the mantissa (as an integer) and the exponent base B of the representation of x.

$$expo(x) = \lfloor \log_B x \rfloor - M + 1, \qquad mant(x) = [x/B^{expo(x)}]$$

$$x = mant(x) B^{expo(x)}$$

As a running example we will consider the computation of $\sin 3.14$. For this we have:

$$B = 2, \qquad M = 53$$

$$mant(3.14) = 7070651414971679, \quad expo(3.14) = -51$$

$$y = mant(3.14) B^{expo(3.14)} = \frac{7070651414971679}{2251799813685248}$$

$$y - 3.14 = .124344978758... \times 10^{-15}$$

$$\sin y = .0015926529164868281957...$$

$$mant(\sin y) = 7344815187169908, \quad expo(\sin y) = -62$$

The way we have defined these values, a ulp for the value x is given by $B^{expo(x)}$. Notice that 3.14 cannot be represented exactly in a binary base, and y is its closest approximation. Similarly, $\sin y$ cannot be computed exactly, it is approximated by

$$\sin y \approx s_y = mant(\sin y) B^{expo(\sin y)} = \frac{1836203796792477}{1152921504606846976}$$

In this case, if we return s_y for $\sin y$, the error made in ulps is:

$$\frac{|\sin y - s_y|}{B^{expo(s_y)}} = .1368513561...\text{ulps}$$

So in terms of the example, we want to find an x, close to y, such that the resulting minimum error is as large as possible, or

$$\frac{|\sin x - s_x|}{B^{expo(s_x)}} = 0.5$$

2 Building the lattice matrix

We will now build a matrix with three row vectors as follows:

$$\begin{bmatrix} F\left(\frac{f(y)}{B^{expo(f(y))}} - \frac{1}{2}\right) & 0 & F \\ \\ F\frac{f'(y)B^{expo(y)}}{B^{expo(f(y))}} & 1 & 0 \\ \\ F & 0 & 0 \end{bmatrix}$$

where F is a size-tuning parameter. When we apply the LLL algorithm, we will view this matrix as 3 rows and will find integer linear combinations of these rows which have minimal height (or norm).

For our example the matrix is

$$\begin{bmatrix} F\left(\frac{\sin y}{B^{expo(s_y)}} - \frac{1}{2}\right) & 0 & F \\ \\ F\frac{\cos y B^{expo(y)}}{B^{expo(s_y)}} & 1 & 0 \\ \\ F & 0 & 0 \end{bmatrix} = \begin{bmatrix} 7344815187169907.363148...F & 0 & F \\ \\ -2047.997402...F & 1 & 0 \\ \\ F & 0 & 0 \end{bmatrix}$$

The first row represents the ulps of the function and of the final result being 1/2 ulp. Its third entry, by F being sufficiently large, will ensure that this row will be used only once.

The second row represents the increment on the function value when we change the argument y by 1 ulp. At this point we will assume that we approximate the change in $f(y)$ in a linear way, just by the derivative of $\sin y$, i.e. $\cos y$. The change is scaled to be in ulps in the result.

The third row is to eliminate all the integer ulps, which are not interesting, what is interesting is the fractional part. This row serves the purpose of computing $\pmod 1$.

A linear combination of these rows, where the first row is included only once, has the following interpretation: the first column measures how many ulps we are away from 1/2 ulp, the second column, by how many ulps we have incremented the y value and the third column how many times we included the function value minus the 1/2 ulp. So the goal is to find a short vector with its third component equal to F in absolute value (it cannot be smaller).

Since the short vectors are measured by their norm, we need to weight the first column more or less to obtain a closer approximation to 1/2 ulp. We will run two examples, for $F = 1000$ we obtain the reduced lattice (after using the LLL algorithm):

$$\begin{bmatrix} 2.59742... & 1 & 0 \\ 129.86363... & -335 & 0 \\ 46.26316... & -122 & 1000 \end{bmatrix}$$

This means that if we subtract 122 ulps to y, the result will be about 46.26316/1000 away from 1/2 ulp.

$$\frac{\sin(y - 122B^{expo(y)})}{B^{expo(s_y)}} = 7344815187419763.546263...$$

and for $F = 10^{10}$ the reduced lattice is

$$\begin{bmatrix} -74696.19177... & -385 & 0 \\ -20054.74739... & -133979 & 0 \\ 25589.00637... & 25270 & 10000000000 \end{bmatrix}$$

and now we are much closer to 1/2 a ulp

$$\frac{\sin(y + 25270B^{expo(y)})}{B^{expo(s_y)}} = 7344815135417013.50000210...$$

Finally, let i be the second value of the row in the reduced matrix which contains F as a last element (if it contains $-F$, then multiply the row by -1). Then for each F we find an x

$$x = y + iB^{expo(y)}$$

At some point, if we keep increasing F, we will find that the approximations to 1/2 ulp do not improve further. This will be due to the approximation with the derivative, which will have an error $O(\frac{i^2 B^{2expo(y)}}{B^{expo(s_y)}})$, where i is the number of ulps in the increment. Since $i = O(\sqrt{F})$, the error due to the linear approximation of the derivative is $O(F)$. The closest proximity to 1/2 ulp will be obtained by $F = O(B^{2/3(expo(s_y)-2expo(y))})$.

In principle, the F value of the first column does not need to be the same as the value in the top third column. The value in the third column could be smaller, but there is no gain in making this change.