paper 8

Fast Fixed-Point Trig

Johann Borenstein
Haifa, Israel

Based on J. Baumgarner's article "Fixed-Point Trig by Derivation," *Forth Dimensions* IV/1, I have written a modified version for the sine function. While this version of **SIN** is approximately as long (in terms of compiled code) and almost as accurate as the original definition, it is about nine times faster (5.3 msec on my Z80A system running at 3.75 MHz). This considerable increase in speed has been achieved by optimal scaling of the series parameters, to the extent that no divisions are performed to evaluate the series. Also, the improved **SIN** is strictly in Forth.

To see how the modified version works, let's start with the basic Taylor-Maclaurin series expansion for the sine function as in (1).

By successively factoring out x and x^2, the series can be written as in (2).

When using a scaled integer x, each multiplication must be divided by the scaling

$$(1)\quad \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots$$

$$(2)\quad \sin x \approx x\left(1 - \frac{x^2}{6}\left(1 - \frac{x^2}{20}\left(1 - \frac{x^2}{42}\left(1 - \frac{x^2}{72}\right)\right)\right)\right)$$

$$(3)\quad \sin x \approx \frac{y}{K}\left(1 - \frac{y^2}{6K^2}\left(1 - \frac{y^2}{20K^2}\left(1 - \frac{y^2}{42K^2}\left(1 - \frac{y^2}{72K}\right)\right)\right)\right)$$

$$(4)\quad z = \frac{K_1}{K_2} X \quad ; \quad \frac{K_1}{K_2} >> 1$$

$$(5)\quad \sin x \approx \frac{K_2}{K_1} z\left(1 - \left(\frac{K_2}{K_1}\right)^2 \frac{z^2}{6}\left(1 - \left(\frac{K_2}{K_1}\right)\frac{z^2}{20}\left(1 - \left(\frac{K_2}{K_1}\right)\frac{z^2}{42}\left(1 - \frac{K_2}{K_1}\frac{z^2}{72}\right)\right)\right)\right)$$

$$(6)\quad xs = \frac{z * z}{k_1}$$

$$(7)\quad \frac{K_1}{K_2}\sin x = \frac{z}{K_1}\left(K_1 - \frac{K_2^2}{6}\frac{XS}{K_1}\left(K_1 - \frac{K_2^2}{20}\frac{XS}{K_1}\left(K_1 - \frac{K_2^2}{42}\frac{XS}{K_1}\left(K_1 - \frac{K_2^2 XS}{72}\right)\right)\right)\right)$$

$$(8)\quad 3784 \sin x = \frac{z}{2^{16}}\left(2^{16} - \frac{50\ XS}{2^{16}}\left(2^{16} - \frac{15\ XS}{2^{16}}\left(2^{16} - \frac{7.14\ XS}{2^{16}}\left(2^{16} - 4.2\ XS\right)\right)\right)\right)$$

$$(9)\quad \frac{1}{\text{scaling factor}} = \frac{1}{K_1/K_2} = \frac{1}{3784} = 0.026\ \%$$

$$(10)\quad a\ b\ --\ m \text{ with } m = 2^{16} - \frac{a * x^2 * b}{2^{16}}$$

(Continued on page 19)

(Continued from page 14)

factor k in order to prevent overflow. Here we choose y = kx and rewrite (2) as in (3).

For the reasons explained later on, we shall actually use two scaling factors (4) such that (2) becomes (5).

Defining a variable (6) as the repeatedly used square term, we can rewrite (5) as in (7).

As can be seen, there are five divisions by k_1, equations (6) and (7), which are rather time-consuming. In Forth there is a way that allows for extremely fast division by 2^{16}. Using assembly language, this corresponds to sixteen right shifts, whereas in Forth the simple **DROP** does the job. Therefore, a scaled multiplication with 2^{16} as the scaling factor may be coded as:

```
z z U* SWAP DROP
```

which is the same as

z z k_1 */with $k_1 = 2^{16}$ but much faster.

More time is saved when the division by the series factors (6, 20, 42, 72) is replaced by multiplications, as may be done by appropriate choice of k_2.

Here k_2 has been choosen as $(k_2)^2 = 300$ such that the series finally becomes as in (8) where $3784 = k_1/k_2 = 2^{16}/\sqrt{300}$

A little precision has been sacrificed (for the sake of speed) by using the integers 7 and 4 instead of the factors 7.14 and 4.2 in (8).

The average precision for the series is now 0.06%, and there is no point in trying to increase precision since it is in any case limited by the scaling factor according to (9).

Description of the Source Screens

Screen 1 holds the basic definition **SIN1** which evaluates the sine of values between zero and 5994 (3784 * PI/2 = 5944), corresponding to zero and 90°. The series may only be evaluated for arguments greater than 256 (= 4^0), since the scaled square of anything smaller than 256 is less than one (therefore zero for integers) and corrupts the series. Fortunately, the sine of very small angles is almost equal to the angle itself, so that argument itself may be used as the result. For the worst case (argument = 256) this simplification yields an error of 0.08 % which is only little more than the average error for the whole series. The term

$$\frac{z * z}{2^{16}}$$

is calculated and stored as XS. Then the innermost bracket $2^{16} - 4 * XS$ is calculated. **TERM1** is called for the remaining elements, where **TERM1** evaluates the frequently used expression (10).

The accumulated **TERMS** are multiplied by $z/2^{16}$ to obtain the scaled result. **DEG** scales whole-degree angles to the input range required by all the trigonometric functions f(z). **KTIMES** operates on the result of all trigonometric calculations and scales it to 1000 * f (z).

DEG and **KTIMES** are used for debugging only. They should not be used in a working application. Example:

Screen 2 holds definitions of additional trigonometric functions, all based on **SIN1**. **SIN1**, **COS1** and **TAN1** are about twice as fast as **SIN**, **COS** and **TAN** but accept input only in the range of $0 < z < 5944$ (angles between 0 and 90°), whereas **SIN**, **COS** and **TAN** accept any input between -2^{15} and $+2^{15} - 1$.

```
30 DEG SIN1 KTIMES — 500
90 DEG SIN1 KTIMES — 1000
```

Due to space limitations, Mr. Bornstein's code will appear in the next issue — Ed.

Index to Advertisers