Personal Calculator Algorithms II: Trigonometric Functions

A detailed explanation of the algorithms used by HP hand-held calculators to compute sine, cosine, and tangent.

by William E. Egbert

BEGINNING WITH THE HP-35,[1,2] all HP personal calculators have used essentially the same algorithms for computing complex mathematical functions in their BCD (binary-coded decimal) microprocessors. While improvements have been made in newer calculators,[3] the changes have affected primarily special cases and not the fundamental algorithms.

This article is the second of a series that examines these algorithms and their implementation. Each article will present in detail the methods used to implement a common mathematical function. For simplicity, rigorous proofs will not be given, and special cases other than those of particular interest will be omitted.

Although tailored for efficiency within the environment of a special-purpose BCD microprocessor, the basic mathematical equations and the techniques used to transform and implement them are applicable to a wide range of computing problems and devices.

The Trigonometric Function Algorithm

This article will discuss the method of generating sine, cosine, and tangent. To minimize program length, a single function, tan θ, is generated first. Once tan θ is calculated, sin θ is found by the formula

$$\sin\theta = \frac{\pm\tan\theta}{\sqrt{1+\tan^2\theta}}.$$

It turns out (as will be explained later) that cot θ can easily be generated while generating tan θ. Then cos θ is calculated using the formula

$$\cos\theta = \frac{\pm\cot\theta}{\sqrt{1+\cot^2\theta}}.$$

It can be seen that these formulas are identical, except for the contangent replacing the tangent. Thus the same routine can solve for either sine or cosine depending on whether the argument is tangent or cotangent.

Scaling

Since θ and $\theta+n(360°)$ yield identical trigonometric functions, every angular argument is resolved to a positive angle between 0° and 360°. For reasons to be explained later, all calculations assume angles expressed in radians. An angle in degrees is first converted to radians by:

$$\theta_{rad} = \theta_{deg} \times \pi/180.$$

Angles expressed in grads are also converted using the appropriate scale factor.

Once θ is in radians, 2π is subtracted repeatedly from $|\theta|$ until the absolute remainder is between 0 and 2π. For large angles this would take a long time. In such cases $2\pi\times10^n$ can be subtracted in a process similar to division. Suppose an angle θ is expressed in scientific notation (e.g., 8.5×10^5). $2\pi\times10^n$, or $6.28...\times10^n$, is then repeatedly subtracted from θ until the result becomes negative (underflow). Thus $6.28...\times10^5$ is subtracted from 8.5×10^5 twice and underflow occurs. $6.28...\times10^5$ is then added to the negative remainder to give a number between 0 and $2\pi\times10^5$, in this case 2.2×10^5. The remainder is expressed now as 22×10^4 and the process is repeated, this time subtracting $2\pi\times10^4$. With this method, large angles are quickly resolved.

The problem with this scaling process is that in current computers numbers can be expressed only to a limited number of digits, so 2π and therefore $2\pi\times10^n$ cannot be expressed exactly. Error creeps in with each shift of the remainder. Thus, the larger the angle, the fewer significant digits remain in the scaled result. A rule of thumb for rough estimates is that for each count in the exponent, one digit of accuracy will be lost. For example, 5×10^5 when scaled will lose five digits of accuracy.

A negative argument is treated the same as a positive number until the end, when the scaling routine returns a number between 0 and -2π. Then 2π is added to the negative result, giving again a number between 0 and 2π. This addition of 2π causes a digit

to be lost, which results in asymmetry such as $\cos(86°) \neq \cos(-86°)$. Newer calculators obviate this problem by scaling to a number between 0 and $\pi/4$.

Vector Rotation

An angle can be expressed as a vector having X and Y components and a resultant **R** (see Fig. 1). If **R** is the unit vector, then $X=\cos\theta$ and $Y=\sin\theta$. However, regardless of the length of **R**, $Y/X=\tan\theta$ and $X/Y=\cot\theta$. This holds true for all values of θ from 0 to 2π. Thus, if some way could be found to generate X and Y for a given θ, all the trigonometric functions could be found.

In vector geometry a useful formula results when one rotates a vector through a given angle. Let us suppose we have a vector whose angle is θ_1, and we know its components X_1 and Y_1 (see Fig. 2). The X_2 and Y_2 that result when the vector is rotated an additional angle θ_2 are given by:

$$X_2 = X_1 \cos\theta_2 - Y_1 \sin\theta_2$$
$$Y_2 = Y_1 \cos\theta_2 + X_1 \sin\theta_2$$

Dividing both sides of these equations by $\cos\theta_2$ gives:

$$\frac{X_2}{\cos\theta_2} = X_1 - Y_1\tan\theta_2 = X_2' \qquad (1)$$
$$\frac{Y_2}{\cos\theta_2} = Y_1 + X_1\tan\theta_2 = Y_2'$$

Note that X_2' and Y_2', while not the true values of X_2 and Y_2, both differ by the same factor, $\cos\theta_2$. Thus $Y_2'/X_2' = Y_2/X_2$. From Fig. 2 it is plain that the quotient Y_2'/X_2' is equal to $\tan(\theta_1+\theta_2)$. *Thus the tangent of a large angle can be found by manipulating smaller angles whose sum equals the large one.* Returning to equation 1 above, it can be seen that to generate X_2' and Y_2', X_1 and Y_1 need to be multiplied by $\tan\theta_2$ and added or subtracted as needed. If θ_2 is chosen so that $\tan\theta_2$ is a simple power of 10 (i.e., 1, 0.1, 0.01,...) then the multiplications simply amount to shifting X_1 and Y_1. Thus to generate X_2' and Y_2', only a shift and an add or subtract are needed.

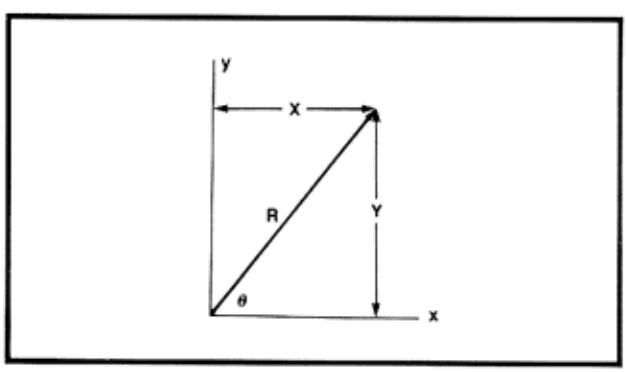

Fig. 1.

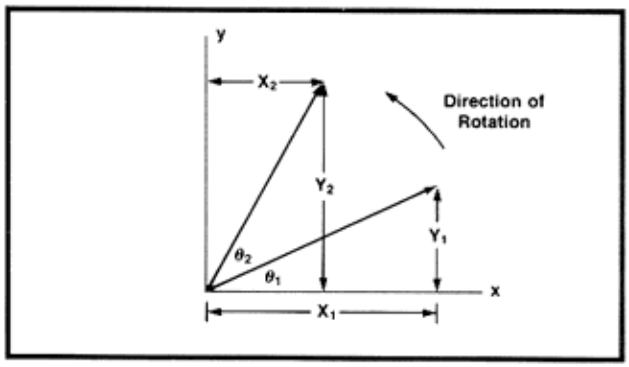

Fig. 2.

Pseudo-Division

The tangent of θ is found as follows. First θ is divided into a sum of smaller angles whose tangents are powers of 10. The angles are $\tan^{-1}(1) = 45°$, $\tan^{-1}(0.1) \approx 5.7°$, $\tan^{-1}(0.01) \approx 0.57°$, $\tan^{-1}(0.001) \approx 0.057°$, $\tan^{-1}(0.0001) \approx 0.0057°$, and so on.
This process is called pseudo-division. First, 45° is subtracted from θ until overdraft, keeping track of the number of subtractions. The remainder is restored by adding 45°. Then 5.7° is repeatedly subtracted, again keeping track of the number of subtractions. This process is repeated with smaller and smaller angles. Thus:

$$\theta = q_0 \tan^{-1}(1) + q_1 \tan^{-1}(0.1) + q_2 \tan^{-1}(0.01)... + r$$

The coefficients q_i refer to the number of subtractions possible in each decade. Each q_i is equal to or less than 10, so it can be stored in a single four-bit digit.

This process of pseudo-division is one reason that all the trigonometric functions are done in radians. For accuracy, $\tan^{-1}(10^{-h})$ needs to be expressed to ten digits. In degrees, these constants are random digits and require considerable ROM (read-only memory) space to store. However, in radians, they become, for the most part, nines followed by sixes. Because of this, they can be generated arithmetically, thus using fewer ROM states. Also, in radians, $\tan^{-1}(1) = \pi/4$, which is needed anyway to generate π. The problem with using radians is that since π is an irrational number, scaling errors occur as discussed earlier. This means cardinal points do not give exact answers. For example, $\sin(720°) \neq 0$ when calculated this way but rather 4×10^{-9}. See reference 3 for a discussion of this point.

So far, a pseudo-quotient has been generated that represents the division of the given angle θ into smaller angles whose tangents are powers of 10. In many HP calculators the pseudo-quotient is five hexadecimal digits long. Each digit represents one series of subtractions and is a number from 0 to 10. For example, if θ were 359.9999°, the pseudo-quotient would be 77877, representing $\theta = 7\tan^{-1}(1) + 7\tan^{-1}(0.1) + 8\tan^{-1}(0.01) + 7\tan^{-1}(0.001) + 7\tan^{-1}(0.0001)$. There may also be a remainder r, which is the angle remaining after the previous partial quotient subtractions have taken place.

Tan θ can now be found using the vector rotation process discussed earlier.

Pseudo-Multiplication

To use equation 1 we need an initial X_1 and Y_1. These correspond to the X and Y of the residual angle r discussed previously. This angle is small (less than 0.001°), and for small angles in radians, $\sin\theta = \theta$ (another reason to use radians instead of degrees). Thus, to good accuracy, the initial Y_1 can be set to the residual angle, and the initial X_1 set to 1. Equation 1 can now be repeatedly applied, where θ_2 is the angle whose tangent is 10^{-j}. Each time equation 1 is applied, a new X_1 and Y_1 are generated, i.e., X_2' and Y_2'. The number of times equation 1 is applied is determined by the count in the pseudo-quotient digit for that θ. Thus if the original angle had a 3 in the pseudo-quotient digit corresponding to $\tan^{-1}0.1$, or 5.7°, equation 1 would be applied three times with X_1 and Y_1 being shifted one place right for tan ($\tan^{-1}0.1$) before the addition or subtraction. In this manner, new X_1 and Y_1 are formed as the vector is rotated the amount corresponding to the count in the pseudo-quotient digits which, of course, sum to the original angle θ.

Equation 1 shows that to generate X_2 requires a shift of Y_1 and a subtraction from X_1. Likewise Y_2 requires a shift of X_1 and an addition to Y_1. To implement this would require either two extra registers to hold the shifted values of X_1 and Y_1, or else shifting one register twice and the other once. It would be desirable to shift only one register once. Happily, this is possible. Consider the following: Let Y = 123 and X = 456. Suppose we want Y+(X×0.01). This can be obtained by keeping the decimal points in the same places and shifting X two places right.

$$\begin{array}{r} 123 \\ +\quad 4.56 \\ \hline 127.56 \end{array}$$

Now suppose instead of shifting X two places right, we multiply Y by 100, shifting it two places left. What happens?

$$\begin{array}{r} 12300 \\ 456 \\ \hline 12756 \end{array}$$

The digits in both answers are exactly the same. The only difference between the two is that the second answer is 100 times the correct value, which is the same value by which Y was multiplied before the addition. Thus to avoid shifting X, Y must be multiplied by 10^j.

Expanding this method to the problem at hand also helps us solve another problem, that of accuracy. During pseudo-division, the angle θ is resolved until a small angle r is left as the original Y value. Since this is done in fixed point arithmetic, zero digits are generated following the decimal point (e.g., .**00**123).

Since zero digits do not convey information except to indicate the decimal point, the remainder is shifted left one place (multiplied by 10) during each decade of pseudo-division. This preserves an extra digit of accuracy with each decade. The final remainder is equal to $r\times10^4$ if the pseudo-quotient is five digits long.

To demonstrate mathematically the implementation that requires only a single register shift, return to equation 1 and replace $\tan\theta_2$ by 10^{-j}. This substitution is legal because $\theta_2 = \tan^{-1}(10^{-j})$, where j is the decade digit.

$$\begin{aligned} X_2' &= X_1 - Y_1\times10^{-j} \\ Y_2' &= Y_1 + X_1\times10^{-j} \end{aligned} \qquad (2)$$

Now let $Z = Y_1\times10^j$, or $Y_1 = Z\times10^{-j}$. Substituting in equation 2,

$X_1' = X_1 - Z\times10^{-2j}$

$Y_2' = Z\times10^{-j} + X_1\times10^{-j}$

Multiplying the second equation by 10^j gives:

$Y_2'\times10^j = Z + X_1$

The left-hand side ($Y_2'\times10^j$) is in the correct form to be the new Z for the next iteration. Thus for each iteration within a decade:

$$\begin{aligned} &X_2' = X_1 - Z\times10^{-2j} \\ &Y_2'\times10^j = Z + X_1 \end{aligned} \qquad (3)$$

X_2' becomes the new X_1

$Y_2'\times10^j$ becomes the new Z

Since the shifted remainder ($r\times10^4$) is desired as Z for the first iteration, the original j is 4.

To implement equation 3, X_1 and Z are stored in two registers. $Z\times10^{-2j}$ is formed and stored in a third register. X_1 is added to Z to form the new Z. This leaves X_1 undisturbed so that $Z\times10^{-2j}$ can be subtracted from it to form the new X_2'.

This implementation saves extra shifts and increases accuracy by removing leading zeros in Z. The only register shifted is Z.

After equation 3 has been applied the number of times indicated by one pseudo-quotient digit, Z is shifted right one place, and a new pseudo-quotient digit is fetched. This in effect creates $Y_1 \times 10^j$, where j is one less than before. Again equation 3 is applied, and the process is repeated until all five pseudo-quotient digits have been exhausted. The result is an X and a Y that are proportional to the cosine and sine of the angle θ. Because the final j is zero, the final Y (=Z) is correctly normalized with respect to X.

So far, then, an X and a Y have been generated by a pseudo-multiply operation consisting of shifts and additions. If tan θ is required, Y/X is generated, which is the correct answer. For sin θ, Y/X is calculated, and for cos θ, X/Y is calculated. Then either X/Y or Y/X is operated on by the routine described at the beginning of this article. The only difference between the computation for sin θ and that for cos θ is whether X and Y are exchanged.

In summary, the computation of trigonometric functions proceeds as follows:

1. Scale the input angle to a number in radians between 0 and 2π.
2. Using the pseudo-division process divide the scaled number into groups of selected smaller angles.
3. With the pseudo-multiply process of equation 3 applied once for each angle resulting from the division of the input argument, generate an X and a Y that are proportional to the sine and cosine of the input angle.
4. With X and Y, compute the required function using elementary operations.
5. Round and display the answer.

The calculator is now ready for another operation.

References

1. T.M. Whitney, F. Rodé, and C.C. Tung, "The 'Powerful Pocketful': An Electronic Calculator Challenges the Slide Rule," Hewlett-Packard Journal, June 1972.
2. D.S. Cochran, "Algorithms and Accuracy in the HP-35," Hewlett-Packard Journal, June 1972.
3. D.W. Harms, "The New Accuracy: Making $2^3=8$," Hewlett-Packard Journal, November 1976.

William E. Egbert

Bill Egbert is a project leader at HP's Corvallis, Oregon Division. He produced this series of algorithm articles as part of his work on the HP-67 and HP-97 Programmable Calculators. He was project leader for the HP-67 and did micro-programming for both calculators. Bill received his BSEE degree from Brigham Young University in 1973 and his MSEE from Stanford University in 1976. He's been with HP since 1973. Born in Fallon, Nevada, he's married, has two small children, and lives in Corvallis.