Personal Calculator Algorithms III: Inverse Trigonometric Functions

A detailed description of the algorithms used in Hewlett-Packard hand-held calculators to compute arc sine, arc cosine, and arc 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 third of a series that examines these algorithms and their implementation. Each article presents in detail the methods used to implement a common mathematical function. For simplicity, rigorous proofs are not given, and special cases other than those of particular interest are 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.

Inverse Trigonometric Functions

This article will discuss the method of generating $\sin^{-1}$, $\cos^{-1}$, and $\tan^{-1}$. An understanding of the trigonometric function algorithm is assumed. This was covered in the second article of this series and the detailed discussion will not be repeated here.[4]

To minimize program length, the function $\tan^{-1}A$ is always computed, regardless of the inverse trigonometric function required. If $\sin^{-1}A$ is desired, $A/\sqrt{1-A^2}$ is computed first, since

$$\sin^{-1} A = \tan^{-1} \frac{A}{\sqrt{1-A^2}}.$$

For $\cos^{-1}A$, $\sin^{-1} A$ is computed as above and then $\cos^{-1} A$ is calculated using

$$\cos^{-1} A = \pi/2 - \sin^{-1}A.$$

Cos^{-1} is found in the range $0 \leq \theta \leq \pi$ and $\sin^{-1}$ and $\tan^{-1}$ are computed for the range $-\pi/2 \leq \theta \leq \pi/2$. The $\tan^{-1}$ routine solves only for angles between 0 and $\pi/2$, since $-\tan A = \tan(-A)$. Thus A may be assumed to be positive and the sign of the input argument becomes the sign of the answer. All angles are calculated in radians and converted to degrees or grads if necessary.

General Algorithm

A vector rotation process similar to that used in the trigonometric routine is used in the inverse process as well. A vector expressed in its X and Y components can easily be rotated through certain specific angles using nothing more than shifts and adds of simple integers. In the algorithm for $\tan^{-1}|A|$, the input argument is $|A|$, or $|\tan\theta|$, where θ is the unknown. Letting $\tan\theta = Y_1/X_1$, $|A|$ can be expressed as $|A|/1$, where $Y_1 = |A|$ and $X_1 = 1$. A vector rotation process (see Fig. 1) is then used to rotate the vector clockwise through a series of successively smaller angles θ_i, counting the number of rotations for each angle, until the Y_2 component approaches zero. If q_i denotes the number of rotations for θ_i then

$$|\theta| = q_o + q_1\theta_1 + \ldots + q_i\theta_i + \ldots$$

This process is described in detail below.

Vector Rotation

To initialize the algorithm, A and 1 are stored in fixed-point format in registers corresponding to Y_1

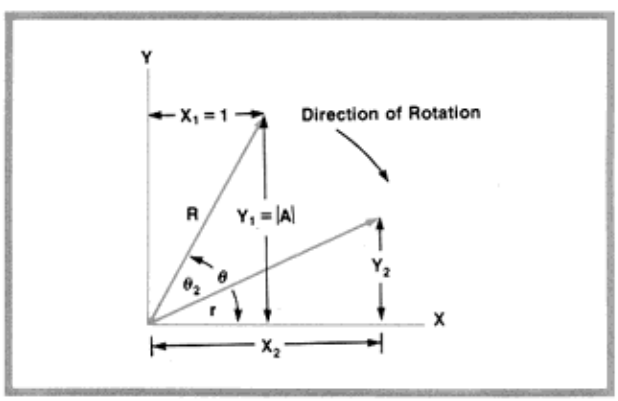

Fig. 1. *Vector rotation.*

and X_1. This is done in such a way as to preserve as many digits of A as possible when the exponent of A differs from zero.

At this point the sign of A is saved and $Y_1 = |A|$. Now comes the vector rotation (see Fig. 1). If the vector **R** is rotated in a clockwise direction, Y_2 becomes smaller and smaller until it passes zero and becomes negative. As soon as Y_2 becomes negative, we know that we have rotated **R** just past the desired angle θ. Thus to find θ, **R** is simply rotated clockwise until Y_2 becomes negative. The amount of rotation is remembered and is equal to the desired angle $\theta = \tan^{-1}|A|$. To rotate **R**, the following formula is used.

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

$$\frac{Y_2}{\cos\theta_2} = Y_1 - X_1 \tan\theta_2 = Y_2' .$$

This equation is the same as equation 1 of the article on trigonometric functions,[4] except that the plus and minus signs are exchanged because **R** is rotated in the opposite direction. As before, $\tan\theta_2$ is chosen such that the implementation requires a simple shift and add ($\tan\theta_2 = 10^{-j}$). To find θ, **R** is initially rotated with $\tan\theta_2 = 1$ ($\theta_2 = 45°$). Y_2' soon becomes negative and the number of successful rotations is stored as the first digit of what is known as the pseudo-quotient. Y_2' is then restored to the last value it had before becoming negative and **R** is rotated again, this time through a smaller angle, i.e., $\tan\theta_2 = 0.1$ ($\theta_2 \approx 5.7°$). This process is repeated with the angle of rotation becoming smaller and smaller until five pseudo-quotient digits have been generated.

At the end of each series of rotations, Y_2 is multiplied by 10 to preserve accuracy.

Pseudo-Multiplication

It is now time to shift gears and add up all the small angles represented by the pseudo-quotient digits. There remains a residual angle ,r, represented by the final X_2' and Y_2'. Since the residual angle is small, we would like to say $Y_2' = \sin r = r$. However, this is true only if $X_2' = 1$. Unfortunately, X_2' in this case is the product of all the $1/\cos\theta$ terms resulting from several applications of equation 1. However, Y_2' is this same product times Y_2. Thus $Y_2'/X_2' = Y_2/1$. Therefore, the final Y_2' is divided by the final X_2' and the result is sin r, which for small angles in radians is approximately equal to r, the residual angle.

With the residual angle as the first partial sum, θ is generated by adding the angles represented by the digits of the pseudo-quotient. This is exactly the reverse of the pseudo-division operation in the trigonometric routine. Thus:

$$\theta = q_0 \tan^{-1}(1) + q_1 \tan^{-1}(0.1) + q_2 \tan^{-1}(0.01) + \ldots + r \qquad (2)$$

Each coefficient q_i refers to the count in a particular pseudo-quotient digit.

The result of this summation process, also called pseudo-multiplication, is an angle θ that is equal to $\tan^{-1}|A|$, where $|A|$ is the input argument to the $\tan^{-1}$ routine. At this point the original sign of A is appended to θ. For $\tan^{-1}$ this angle is normalized, converted to degrees or grads if necessary, and displayed. Recall that for $\sin^{-1}$, $A/\sqrt{1-A^2}$ was first generated. Thus for $\sin^{-1}$, the result of the $\tan^{-1}$ routine is again simply normalized, converted to degrees or grads if necessary, and displayed. For $\cos^{-1}$, the $\tan^{-1}$ routine returns $\sin^{-1}$. $\cos^{-1}$ is then simply found as $\pi/2 - \sin^{-1}A$.

Summary

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

1. Calculate $A/\sqrt{1-A^2}$ if the desired function is $\sin^{-1} A$ or $\cos^{-1} A$.
2. Place $|A|$ and 1 in fixed-point format into appropriate registers, while preserving the sign of A.
3. Repeatedly rotate the vector with A=Y and 1=X clockwise using equation 1 until Y approaches zero. The number of rotations and the amount of each rotation is stored as a pseudo-quotient along the way.
4. Using the pseudo-multiplication process of equation 2, sum all of the angles used in the rotation to form $|\theta|$.
5. Append the proper sign to the answer and calculate $\cos^{-1} A = \pi/2 - \sin^{-1} A$ if required.
6. Convert to the selected angle mode, and 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.
4. W.E. Egbert, "Personal Calculator Algorithms II: Trigonometric Functions," Hewlett-Packard Journal, June 1977.

Bill Egbert is a project manager at HP's Corvallis (Oregon) Division.