USING NUMERICAL APPROXIMATIONS AND MODELING TO CREATE A GRAPHING CALCULATOR IN FLASH

Paul Bouthellier
Department of Mathematics and Computer Science
University of Pittsburgh-Titusville
504 East Main Street
Titusville, PA 16354
pbouthe@pitt.edu

Introduction:

Students often use calculators and computers without understanding how function values are derived. To illustrate the mathematics behind such approximations, in this paper algorithms are discussed which approximate values for trigonometric, hyperbolic, exponential, and logarithmic functions.

Students will be shown how functions such as sin(x), cos(x), tan(x), $\sin^{-1}(x)$, $\cos^{-1}(x)$, $\tan^{-1}(x)$, sinh(x), cosh(x), tanh(x), $\sinh^{-1}(x)$, $\cosh^{-1}(x)$, $\tanh^{-1}(x)$, e^x, ln(x), and a^x are approximated by the CORDIC (Coordinate Rotation Digital Computer) method. A more efficient method using minimax polynomials is also given. These algorithms will be implemented in Flash programs which will simulate a calculator.

The programs in this presentation were written in the ActionScript language of Flash. The programs were saved in the .swf format and embedded in web pages. Students were not required to learn how to program in ActionScript-just to be able to alter the code and re-run the programs. This worked well after walking through a few examples.

CORDIC-Coordinate Rotation Digital Computer

The CORDIC method was invented in 1959 by Jack E. Volder as a faster method to calculate trigonometric values on early machines which did not have a software multiplier [1].

By using $\theta_i = \tan^{-1}(2^{-i})$ $i = 0,1,2,3,...,41$ all angles in the 1st quadrant can be covered with a maximum error of less than $2.605*10^{-11} \Rightarrow$ maximum error of sin(x) and cos(x) of $5*10^{-13}$.

$\theta_0 = \tan^{-1}(2^{-0}) = 45^\circ$, $\theta_1 = \tan^{-1}(2^{-1}) = 26.565^\circ$,..., $\theta_5 = \tan^{-1}(2^{-5}) = 1.7899^\circ$...

For example: $60^\circ = \theta_0 + \theta_1 - \theta_2 + \theta_3 - \theta_4 - \theta_5 + \ldots$.

Using

$$\begin{aligned}\sin(u \pm v) &= \sin(u)\cos(v) \pm \cos(u)\sin(v)\\ \cos(u \pm v) &= \cos(u)\cos(v) \mp \sin(u)\sin(v)\end{aligned} \tag{1}$$

which may be written

$$\begin{aligned}\sin(u \pm v) &= \cos(v)[\sin(u) \pm \tan(v)\cos(u)]\\ \cos(u \pm v) &= \cos(v)[\cos(u) \mp \tan(v)\sin(u)]\end{aligned} \tag{2}$$

Starting with

$$\begin{aligned}\sin(\,u_0\,) &= \cos(\,\theta_0\,)\\ \cos(\,u_0\,) &= \cos(\,\theta_0\,)\end{aligned} \tag{3}$$

and iterating (2) we get [1]

$$\begin{aligned}\sin(u_{42}) &= \prod_{i=0}^{41}\cos(\theta_i)s_{42}\\ \cos(u_{42}) &= \prod_{i=0}^{41}\cos(\theta_i)c_{42}\end{aligned} \tag{4}$$

Setting

$$s_0 = c_0 = \prod_{i=0}^{41}\cos(\theta_i) \tag{5}$$

s_{42} and c_{42} which approximate sin(x) and cos(x) respectively with a maximum error of $5*10^{-13}$ are computed as follows:

$$\begin{aligned}c_{i+1} &= c_i - 2^{-i}D_i s_i\\ s_{i+1} &= s_i + 2^{-i}D_i c_i\end{aligned} \tag{6}$$

where

$$D_i = \begin{cases} 1 & \textit{if we add the angle}\\ -1 & \textit{if we subtract the angle}\end{cases} \tag{7}$$

Having computed sin(x) and cos(x) we can then compute tan(x). (Note: in order to get a desired degree of accuaracy on tan(x), one needs to compute the necessary accuracy required for sin(x) and cos(x), which in general will be different, and then run the algorithm with the required number of iterations.)

Computing $\sin^{-1}(x)$, $\cos^{-1}(x)$, $\tan^{-1}(x)$

Inverse tangent

Define initial values:

$$\begin{aligned} &s(in) = \tan^{-1}(x) \\ &c(os) = 1 \\ &\Rightarrow \frac{s}{c} = \tan^{-1}(x) \end{aligned} \tag{8}$$

in (6) and drive s→0 adding and subtracting the corresponding angles.. This computes $\theta = \tan^{-1}(x)$. $\sin^{-1}(x)$ and $\cos^{-1}(x)$ can then be computed from $\tan^{-1}(x)$.

Computing sinh(x), cosh(x), and tanh(x)

Find

1. sinh(x) and cosh(x) on [0, 1] →e^x on [0, 1]
2. storing e^1→ e^x and e^{-x} for all x→
3. sinh(x), cosh(x), and tanh(x) for all x

Use

$$\begin{aligned} \sinh(u \pm v) &= \cosh(v)[\sinh(u) \pm \tanh(v)\cosh(u)] \\ \cosh(u \pm v) &= \cosh(v)[\cosh(u) \pm \tanh(v)\sinh(u)] \end{aligned} \tag{9}$$

Defining:

$$\begin{aligned} &\theta_i = \tanh^{-1}((\frac{3}{2})^{-i})\ i = 1, 2, 3, \ldots \\ &\theta_1 = \tanh^{-1}((\frac{3}{2})^{-1}) = .8042 \\ &\theta_2 = \tanh^{-1}((\frac{3}{2})^{-2}) = .47775 \\ &\theta_3 = \tanh^{-1}((\frac{3}{2})^{-3}) = .305454 \\ &\quad\vdots \end{aligned} \tag{10}$$

By using powers of (3/2) instead of 2 we can cover the interval 0 to 1 without having to use duplicates of any angles.

Where

$$s_1 = c_1 = \prod_{i=1}^{72} \cosh(\theta_i) \tag{11}$$

s_i and c_i are computed as follows:

$$\begin{aligned} c_{i+1} &= c_i + (\frac{3}{2})^{-i-1} D_i s_i \\ s_{i+1} &= s_i + (\frac{3}{2})^{-i-1} D_i c_i \end{aligned} \tag{12}$$

where

$$D_i = \begin{cases} 1 & \textit{if we add the angle} \\ -1 & \textit{if we subtract the angle} \end{cases} \tag{13}$$

s_{72}=sinh(x) and c_{72}=cosh(x) with a maximum error of $3.23*10^{-13}$ on [0, 1].

The inverse hyperbolic functions can be computed in a similar fashion to the inverse trigonometric functions described above.

As $\ln(x) = 2\tanh^{-1}(\frac{x-1}{x+1})$ and $a^x = e^{x \ln a}$, ln(x) and powers may now be computed.

Minimax Polynomials-Much faster approximations when a multiplier is available

Given a function f(x) the nth degree polynomial p(x) which satisfies

$$\min_{p \in P_n} (\max_{a \le x \le b} | f(x) - p(x) |) \tag{14}$$

is called the n^{th} order minimax polynomial of f(x) on the interval [a, b].

```
> minimax(sin(x),x=0..Pi/2,[10,0]);
-0.11989779143 10^-11 + (1.00000000186750 + (-0.48080844307 10^-8 + (-0.1666666183567628 + (-0.249222475568 10^-6 + (
  0.008334084753389323 + (-0.14164429551428 10^-5
  + (-0.0001966903087260213 + (-0.1355176683237023 10^-5 + (0.3426105525245168 10^-5 - 0.1919209214135189 10^-6 x) x) x) x) x) x) x) x) x)
  x
```

Figure 1-Minimax polynomial for sin(x) on [0, π/2] from Maple

Where b(x) is the minimax polynomial computed above, Figure 2 below shows the error of maximum error of approximation on the interval [0, π/2] is approximately 10^{-12}.

```
> plot(b(x)-sin(x),x=0..Pi/2);
```

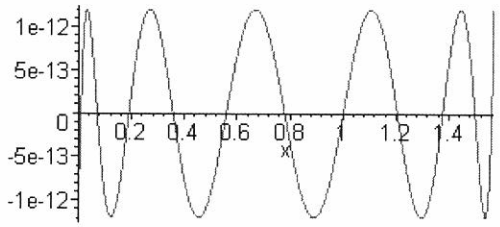

Figure 2-Error of the minimax polynomial for sin(x) on [0, π/2] from Maple

On trials the minimax polynomial computed approximations for sin(x) approximately 30 times as fast as the CORDIC method. (Note: This takes into account the CORDIC method also computed cos(x) as part of the algorithm.)

Generating Graphs and Rotations using Numerical Approximations

It should be noted that the algorithms illustrated above were accurate to 12 or 13 decimal places. In terms of graphing on calculators or computers-which are only accurate to the pixel-far less accuracy is needed and hence fewer iterations and lower order polynomials may be used to greatly speed up computations.

Conclusions

Our objective was to show students some methods by which calculators and computer packages can approximate the elementary functions.

References

[1] Richard Parris, *Elementary Functions and Appoximations*, Phillips Exeter Academy, math.exeter.edu/rparris/peanut/cordic.pdf.

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