

Re: Efficiency of math.h

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2004-02/4731.html

From: Christian Bau (christian.bau_at_cbau.freemove.co.uk)

Date: 02/26/04

Date: Wed, 25 Feb 2004 23:08:02 +0000

In article <403D0A18.5030204@jpl.nasa.gov>,
"E. Robert Tisdale" <E.Robert.Tisdale@jpl.nasa.gov> wrote:

> *CBFalconer wrote:*
>
> > *Peter Ammon wrote:*
> >
> > ... *snip* ...
> >
> > > *Why not get libm from GNU and find out? I wouldn't be surprised*
> > > *if you could beat acos() with "good enough" precision over a*
> > > *particular range (whatever you're interested in).*
> > > *Try using a Taylor expansion.*
> >
> > *Not if you want speed you don't.*
> > *With some slight trigonometry,*
> > *you should be able to convert cos(angle) into tan(angle).*
> > *Use this as input to an arctan(value) routine,*
> > *which can in turn be built in various ways.*
> > *One uses a leveled Tschebychev approximation over 0 to 1.*
>
> *Not if you want speed you don't.*
>
> *This method requires reading a bunch of high precision*
> *floating-point numbers from system memory*
> *which can actually require more time than computing*
> *Taylor coefficients in registers on-the-fly.*
>
> *You're out of your depth here Chuck.*

Tisdale, you are clueless. As usual.

The fastest approach would be to split the argument range into two parts, $\text{abs}(x) \leq c$ and $\text{abs}(x) > c$ for some suitably chosen c , probably somewhere around 0.7 or 0.8. For $\text{abs}(x) \leq c$, approximate $\text{acos } x$ by a polynomial of the form $(\pi - ax - bx^3 - cx^5\dots)$. For $x > c$, define $f(z) = \text{acos}(1 - z^2)$. Approximate $f(z)$ by a polynomial of the form $ax + bx^3 + cx^5 + dx^7$. Calculate $\text{acos}(x) = f(\sqrt{1-x})$. For x

comp.lang.c: Re: Efficiency of math.h

$\leq -c$ use $\arccos(x) = \pi - \arccos(-x) = \pi - f(\sqrt{1+x})$. Take a higher polynomial depending on how much precision you want. Finding the coefficients is left as an exercise to the reader :)

The substitution $z = \sqrt{1-x}$ nicely catches the behavior of $\arccos x$ for $|\arccos(x)|$ close to $\pi/2$, where the first derivative is unbounded and avoids excessive rounding errors.