Subject: Re: efficient kernel or masking algorithm ?
Posted by John-David T. Smith on Thu, 30 Nov 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> "J.D. Smith" <jdsmith@astro.cornell.edu> writes:
> ...
>> While I'm on the gripe train, why shouldn't we be able to consistently
>> perform operations along any dimension of an array we like with relevant
>> IDL routines. E.g., we can total along a single dimension. All due
>> respect to Craig's CMAPPLY function, but some of these things should be
>> much faster. Resorting to summed logarithms for multiplication is not
>> entirely dubious, but why shouldn't we be able to say:
> ...
>
> Agree! Agree! Agree! For once we are griping in synchrony :-)
>
> These are exactly the kinds of operations that would be enhanced by
> vectorization, but they can't as IDL stands now.
>
> By the way, CMAPPLY doesn't use summed logarithms any more. It uses
> my bestest algorithm that came out of the recent newsgroup discussion.
>

Ahh yes, multiplication by decimation. Must have missed that one. I simply read the comment in your code without looking at the details:

;; *** Multiplication
(newop EQ '*'): begin ;; Multiplication (by summation of logarithms)

Did you do some time testing and find all that shifted indexing was really faster than the logarithm? This I suspect will be very architecture dependent. Looks neat though.

Maybe I'll write up the 100 lines of C it would take for a shared library to do dimensional multiply, sum, add, median, min, max, and, or, mode, variance, etc., and send it to RSI. The problem with all of this specific "vector-aware" coding, is that it reveals a dirty secret of IDL's. It was built to do some vector operations fast, but was never a truly generic vector language like APL or J.

But sinceDavid hasn't written a book on either of those, we'll just have to slog through with what we have. <insert disremembered sarcasm code>

JD