Subject: Re: multiplication
Posted by James Kuyper on Wed, 29 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

meron@cars3.uchicago.edu wrote:
>
> In article <38E0A379.34ADB7F7@wizard.net>, James Kuyper <kuyper@wizard.net> writes:
>> meron@cars3.uchicago.edu wrote:
...
>>> dum = where(a lt 0, ndum)
>>> sig = (-1)^ndum
>>> result = sig*exp(total(alog(abs(a))))
>>
>> You can't honestly be suggesting that this is a good technique?
>
> Good? No, only not as bad as using "for".
>
>> Ignore for a momement what happens if any element of 'a' is 0.
>
> That's the easiest to deal with. You're already checking for presence
> of negative elements, can check for zeroes as well. That should be
> the first thing, in fact, since if even one of the elements is 0, then
> the result is 0 and you can dispense with the rest of the evaluation.
>
>> That code performs two transcendental function evaluations per element
>> of 'a'.
>
> Yep, indeed.
>
>> IDL would have to be very badly engineered (which I suppose is possible),
>> for a 'for' loop to execute more slowly than your code.
>
> Well, I run a quick test, comparing the time it takes tto evaluate the
> product using both methods (it run on an old Vms Alpha, somebody may
> want to repeat it on a more modern platform. Being lazy, I'm simply
> filling an array with a constant element, then doing the
> multiplication. Here is the output
>
> IDL> speed, 1.00001, 100, 10
> "for" time = 0.0012000084 res = 1.00100
> "exp-log" time = 0.00019999743 res = 1.00100
>
> IDL> speed, 1.00001, 1000, 10
> "for" time = 0.012699997 res = 1.01006
> "exp-log" time = 0.0012000084 res = 1.01006
>
> IDL> speed, 1.00001, 10000, 10
> "for" time = 0.12589999 res = 1.10532

> "exp-log" time = 0.011699998 res = 1.10532
>
> IDL> speed, 1.00001, 100000, 10
> "for" time = 1.2583000 res = 2.72191
> "exp-log" time = 0.12850000 res = 2.72198
>
> The first input to SPEED is the array element, the second is the
> length of the array. the third is just telling SPEED how many times to
> repeat the test. As you can see, the above was tried for arrays with
> lengths ranging from 100 to 100000 and calculation using "for" loop is
> consistently an order of magnitude slower.

OK - I'd not bothered testing before, I didn't realize the disadvantage
of for loops was that large. Point taken.
