Subject: Re: Cumulative max() in *arbitrary* dimension?
Posted by JDS on Fri, 16 Mar 2012 21:26:44 GMT
View Forum Message <> Reply to Message

On Saturday, March 10, 2012 12:21:48 PM UTC-5, Heinz Stege wrote:
> First: On Fri, 09 Mar 2012 18:58:13 +0100, I wrote:
>
>>   for i=off,ns-off,off do a[i]=a[i-off:i-1]>a[i:i+off-1]
>
> There is a mistake. It has to read:
>
>   for i=off,n_elements(a)-off,off do a[i]=a[i-off:i-1]>a[i:i+off-1]
>
>
> Second: My contribution => my work. ;-) I measured the times for an
> array a=byte(randomu(seed,60,400,3000),/long). I tested the following
> versions:
>
> [1] i1=0 &i2=off-1 & $
>   for i=1,s[d]-1 do a[i*off]=a[i1:i2]>a[(i1+=off):(i2+=off)]
> [2] for i=1,s[d]-1 do $
>   a[i*off]=a[(i-1)*off:i*off-1]>a[i*off:(i+1)*off-1]
> [3] for i=off,n_elements(a)-off,off do a[i]=a[i-off:i-1]>a[i:i+off-1]
>
> I ran each version 2000 times and found no significant differences in
> the run-time. There only seems to be a slight trend for [2] beeing the
> slowest. However it does not have any practical relevance. Here are
> the details:
>
> [1] 134.3 (+/-0.2) ms (calculation of i1 and i2 included)
> [2] 134.9 (+/-0.2) ms
> [3] 134.2 (+/-0.2) ms
>
> The given errors are statistical standard-errors ("1 sigma").

Thanks, Heinz. When compared to processing 10^5 or 10^6 numbers within each loop iteration (which is what makes this a fast loop), the indexing math isn't really significant. Yours is a bit easier on the eye though!

JD