Subject: Re: The IDL way, summing variable sized slices of array.
Posted by d19997919 on Wed, 04 Apr 2012 15:38:33 GMT
View Forum Message <> Reply to Message

On Wednesday, April 4, 2012 3:59:54 PM UTC+1, Russell wrote:
> On Apr 4, 4:00 am, d19997...@gmail.com wrote:
>> Hi,
>>
>> I've recently been learning how to use REBIN/REFORM etc. to do the heavy lifting rather than loops, (saving me at least an order of magnitude in execution time in some of the code I'm working with). I have a situation now where I don't know if it's possible to completely remove loops so I was hoping someone more experienced could illuminate me.
>>
>> In essence the problem is that I have a 3D array which I want to reduce to a 2D array by summing over elements of the first dimension. This wouldn't be an issue apart from the fact that the range of elements that I wish to sum over varies depending on the value of the second dimension.
>>
>> In code what I have at the moment looks a bit like:
>>
>> d=DBLARR(nt,nl,ne) ;Array of data
>> t=DBLARR(nt) ; Array of "axis" values
>> b=DBLARR(nl,2) ;Array of summation limits
>> p=DBLARR(nl,ne) ; Array of answer
>>
>> ;<<BIT OF CODE TO FILL THESE ARRAYS>>
>>
>> FOR i=0L,nl-1 DO BEGIN
>> tmp=TOTAL(d[b[i,0]:b[i,1],i,*],1) ;Sum elements
>> p[i,*]=tmp/TOTAL(t[b[i,0]:b[i,1]]) ;Store sum divided by sum of axis (i.e. get average value over summation range)
>> ENDFOR
>> RETURN,p
>>
>> Any help will be appreciated,
>>
>> Note whilst nl is not necessarily large in the cases I'll be looking at, i'm still interesting in "the IDL way" for this as part of my learning!
>>
>> Thanks,
>> David
>
> Not, sure... This sounds like the rare case were loops are useful.
> BTW, I'm guessing you have a loop to fill the arrays? If so, then why
> not stick this part in that loop?
>
> Russell

Actually I've managed to eliminate loops in the other bit of code to fill the arrays (and this code actually consists of a few other functions).

One way I've thought of doing this is to create a logical array which has the same dimensions as d and is zero outside the b index range and 1 inside. I can then multiply d by this and then just sum over the whole array, seen as d is now zero outside the region of interest it shouldn't effect the result of the call to TOTAL.

Any tips on the best way to achieve this?
