Subject: Re: What are the rules for automatic removal of singleton dimensions, and can I have a way of disabling them, please?
Posted by on Wed, 26 Dec 2012 09:25:09 GMT
View Forum Message <> Reply to Message

Den måndagen den 24:e december 2012 kl. 00:01:24 UTC+1 skrev Tom Grydeland:
> Hello all,
>
>
>
> I was trying to visualize subsections of a windowing function when this bit me.
>
>
>
> I was creating piecewise results in an array res[ix, iy, ii, jj], where the partial results were sums up to some value in the final two indices
>
>
>
> for ii = 0, Kx-1 do begin
>
> for jj = 0, Ky-1 do begin
>
> visualize, total(total(res[*,*,0:ii,0:jj], 4), 3)
>
> endfor
>
> endfor
>
>
>
> but the problem is that IDL (arbitrarily, IMO) discards trailing singleton dimensions on my indexing, so that res[*,*,0:ii,0:jj] ends up as a two-dimensional array when both ii and jj are zero, and a three-dimensional index on subsequent cases of jj being zero, which again causes the calls to 'total' to fail. The innermost portion of these loops become extraordinarily messy if trying to fix this problem.
>
>
>
> I've fixed my program by reordering the indices (to [ii, jj, ix, iy]), but I am still miffed that this should be necessary.
>
>
>
> Similarly, when I concatenate arrays of dimensions [x, j] and [y, j], I expect a result with dimensions [x+y, j], even when j is equal to 1. I'm trying to write programs independent of the actual value of j, but these arbitrary removals of singleton dimensions make my task that much harder.

>
>
>
> Is there a way to disable this stripping of singleton dimensions?

I thought the idea with the stripping of dimensions was that it shouldn't matter whether these dimensions are there or not. I mena, you can still address a[2,2,0] when a=fltarr(5,5). So shouldn't the fix really be to make total() work for non-existing trailing dimensions?

(Or, if just to make your code less messy, to write a mytotal() function that takes care of the exceptions.)