Subject: Re: indexing
Posted by JDS on Wed, 08 Apr 2009 22:35:56 GMT
View Forum Message <> Reply to Message

On Apr 2, 3:20 pm, Jeremy Bailin <astroco...@gmail.com> wrote:
> On Apr 2, 12:49 pm, "Dick Jackson" <d...@d-jackson.com> wrote:
>
>
>
>> Hi all,
>
>> Good one, Chris. How about this for getting the subset:
>
>> PRO IndexXYPairsOverZ
>
>> a = Transpose(IndGen(10,10,10), [2,1,0]) ; Array where, e.g. a[3,0,4]=304
>
>> xy = [[3,4],[2,1],[3,7]]
>> z0 = 1
>> z1 = 3
>
>> dims=Size(a,/Dim)
>> a = Reform(a, dims[0]*dims[1], dims[2], /Overwrite) ; Reshape a temporarily
>> xyIndices = xy[0,*]+xy[1,*]*dims[0]
>> subset = a[xyIndices, z0:z1]
>> a = Reform(a, dims, /Overwrite) ; Restore a's shape
>
>> Help, subset
>> Print, subset
>
>> END
>

This is an really cool way of doing it, but it's nothing that
straightforward REBIN can't handle:

```
 nxy=dims[0]*dims[1] & nz=z1-z0+1
 t=[nz,n_elements(xy)/2]
 indices=rebin(xy[0,*]+dims[0]*xy[1,*],t) + rebin(nxy*(z0+lindgen
(nz)),t)
```

I regard the computation of an index array as a *benefit* not a
liability here and in many cases. The reason? IDL happily computes
its own array of indices for you behind the scenes when you use the
higher-order indexing function, e.g. a[xyIndices, z0:z1]. Especially
problematic are statements like a[*,*,1:3] (as above) which don't just
make an index vector, but one which is much larger than needed,
wasting time and memory. The advantage of computing the index vector

yourself is that you can re-use it without throwing it away and computing it all over again (which is what IDL would do).

What was extremely interesting about this problem was the relative performance of these two methods for very large lists of xy indices and large arrays. For small to moderate lists of xy pairs, the REFORM method Dick presented was roughly 2x faster for me. However, as the size of the xy list got large, the REBIN method catches up and eventually overtakes the REFORM method. Over about 5 million xy pairs by 100 z planes, the REBIN method keeps getting faster compared to the IDL-native calculations of the indices. Probably a memory usage difference, or perhaps related to the use of the thread pool (dual proc system). Still, getting IDL to do all the index computation almost entirely internally, as Dick's method does, seems to be a real benefit at least for some problem sizes.

JD
