Subject: Re: Array multiplication: implicit loop query
Posted by John-David T. Smith on Mon, 13 Aug 2001 14:43:29 GMT
View Forum Message <> Reply to Message

Richard Younger wrote:
>
> "Bill B." wrote:
>>
>> george@apg.ph.ucl.ac.uk (george Millward) wrote in message
>
>>> It
>>> would seem that, to get this to work I need to make
>>> Pres=fltarr(30,91,40).
>>
>> Yes.
>>
>> IDL> a = indgen(20,20)
>> IDL> b = indgen(20)
>> IDL> c = b * a
>> IDL> help, c
>> C INT = Array[20]
>>
>> I believe you need to REPLICATE 'Pres' as needed.
>>
>
> I've been converted to REBIN, myself.
> (see the group archives for the dimensional juggling tutorial by JD
> Smith this past spring)
>

Don't abandon those subscripting array inflation techniques just yet though! While rebin/reform is conceptually simpler (especially for more than 2 dimensions), the old lindgen() method still has its place. When, you ask? Well, rebin works only with numeric data. If you have an array of structures, pointers, or objects, you'll need to fall back on the ancestral methods.

The idea is simple. Construct an array of indices of the size you're after, and use "mod" and "/" to massage it into the correct form for indexing into the original array. If you have many such arrays to inflate, it may even be competitive in speed (since you have to precompute the index array only once).

In 2D it's simple.

IDL> a=findgen(5)
IDL> inds=lindgen(5,10)
IDL> big_a=a[inds mod 5] ; across

```
IDL> inds=lindgen(10,5)
IDL> big_a=a[inds/10] ; down
```

for higher dimensions, it quickly becomes cumbersome (try it and see).

JD

P.S. Here's an example of this method's use in the field... a little function I cooked up to find where in one vector elements of another vector do *not* exist. As a bonus, not a histogram in there.

```
function where_not_array,A,B,cnt,IA_IN_B=iA_in_B

   Na = n_elements(a)
   Nb = n_elements(b)
   l = lindgen(Na,Nb)
   AA = A(l mod Na)
   BB = B(l / Na)

   if keyword_set(iA_in_B) then $
    wh = where(total(AA ne BB,2) eq Nb,cnt) $
   else wh = where(total(AA ne BB,1) eq Na,cnt)

   return,wh
end
```