Subject: Re: Array multiplication: implicit loop query
Posted by Richard Younger on Mon, 13 Aug 2001 18:37:11 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
>

[...]

>
> 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
[...]

Wow. Thanks, JD. This looks like a wonderfully useful technique. Consequently, I want to try to figure out for myself how it works in a more general way. :-) It took me a little while working out examples to get a start, but let me see if my mental model is at all accurate.

It seems this method in multiple dimensions requires indexing using only one index dimension (i.e. A[6] instead of A[0,1,2] ). This makes things complicated, since modulo works nicely for cutting out the last dimension, and division works nicely for cutting out the first, but for getting stuff in between you have to do a combination of both.

conventions for the next few statements:

-take an array A with dimensions 1,2,...,n and dimension sizes S[1,2,...,n]
-you want to insert an expanded dimension of size Se

Therefore, if you want to put the existing array in dimensions (2 -> n+1) and expand on dimension 1, you can create an index array with dimensions (Se, S[1], S[2],...,S[n]), and divide your index array by Se.

If you want to put the existing array in dimensions (1 - > n) and expand on dimension n+1, you can create an index array with dimensions (S[1], S[2],...,S[n], Se) and take the modulo with Product(S[1],...,S[n]).

And your final array is just A with indices as above. I think you could expand multiple dimensions at the beginning or end by replacing Se with [Se1, Se2, ...] and the division with Se by Prod(Se1, Se2, ...), but I haven't tried it out.

However, if you want to expand on some interior dimension, call it k, things get complicated. It seems to me that you'd have to insert the new dimension at k in the index array and do some combination of division and modulo arithmetic (perhaps modulo Prod(S[1],..., S[k]) and division by (?) -- it would have to reduce to the two cases above) to get things to work out. Can't you just do one of the above and TRANSPOSE() to get the desired array? Hmm. I'm sure someone's figured this out already.

Thoughts?

Rich

P.S. On a side note, I recently reexamined histogram and I think I'm beginning to understand it. The breakthrough from total opacity came when I stopped thinking of REVERSE_INDICES as an array of numbers and started thinking of it as a data structure of pointers. Now I can start puzzling through those histogram examples instead of just having my eyes glaze over at the magic of it all.

--
Richard Younger