Subject: Re: Specification for a new array slicing function
Posted by Liam Gumley on Thu, 20 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan wrote:
> IMO, the use of keyword parameters for START, STRIDE and
> COUNT is a bit "wordy" for my liking. And these items
> are really essential to the routine as such. So why not
> use positional parameters?

I agree that START, STRIDE, COUNT are somewhat wordy. However I'd like to be able to specify them in any combination, e.g.
START only, or
STRIDE only, or
COUNT only, or
START and STRIDE, or
START and COUNT, or
STRIDE and COUNT, or
START and STRIDE and COUNT.
I'm not sure I know a clean way to allow these combinations, other than using optional keywords. Does anyone else have any thoughts? If optional positional parameters were used, there would have to be a hierarchy, as Stein suggests below.

> For something that really ought to be a part of the IDL
> syntax, I would also like a shorter name (despite the
> possibility for name conflicts), like "arex", short
> for array_extract.

How about ARRGET and ARRPUT? (Sounds a bit like FORTRAN-77 to me...)

> My suggestion would be something a bit more like the native
> Fortran 9X syntax (not that I actually *know* exactly how that
> syntax works!) , e.g.:
> a(0:5:2,:,5:9) would be translated into
> arex(a,[0,5,2],-1,[5,9])
> I.e., each positional parameter signifies the extraction
> operation for one array dimension. There are some issues
> that I would like to clear up, though: What exactly does
> the 0:5:2 sequence mean? Does it mean elements 0:5, sampled
> with a stride of 2? Or does it mean 5 elements sampled with
> a stride of 2, starting from 0? Or is it START:STRIDE:COUNT,
> meaning 2 elements, sampled with a stride of 5?
> Anyway, the three elements in each parameter appear in
> "optionality" order: start [, stride [, count]] (if that's
> what the syntax is supposed to be).

If it was done this way, I'd prefer that the positional parameters be

defined the same way as in my original spec.

> Looking at the example above, you may wonder what the "-1" is
> doing there... Well, the idea is that one could use a
> nonnegative *scalar* parameter to signify extraction of a
> slice at a given position, whilst -1 really means "*", in IDL
> notation.

If you use the keyword method, then omitting the keyword means read 'everything' in the START, STRIDE, or COUNT context.

> I've always disliked the way this works:
> a = fltarr(5,5,5)
> surface,a(*,3,*)
> % SURFACE: Array must have 2 dimensions: <FLOAT Array[5, 1, 5]>.
> I mean - if I'm extracting an "image" out of a "cube", why
> would I want the last dimension to stick around...???
> So, I would like to be able to say
> surface,arex(a,-1,3,-1)

This could always be an option. The default method could be to leave dangling dimensions, with an optional keyword Boolean flag that specifies all dangling dimensions be removed. I think leaving the dangling dimensions alone by default is a good idea, since you might want to modify the extracted array, and then re-insert it back into the original array. That might be difficult if the dangling dimensions are lost.

> I would also like to see a corresponding index function,
> returning the one-dimensional indices to the extracted
> elements instead of the elements themselves. This could
> be used for assignments. I.e.:
> a(arexi(a,-1,[3],[0,2])) = data_block

That shouldn't be a problem.

Liam E. Gumley
Space Science and Engineering Center, UW-Madison
http://cimss.ssec.wisc.edu/~gumley