Subject: Re: Array multiplication: implicit loop query
Posted by bennetsc on Wed, 15 Aug 2001 06:57:32 GMT
View Forum Message <> Reply to Message

In article <onelqg12q7.fsf@cow.physics.wisc.edu>,
Craig Markwardt <craigmnet@cow.physics.wisc.edu> wrote:
>
> george@apg.ph.ucl.ac.uk (george Millward) writes:
>>
>> I have inserted the "rebin" function and this works fine.
>> I am still a little intrigued as to why IDL works this way - it still
>> seems to me that my orginal combination of 3D and 1D arrays should
>> yield a 3D array. Not a problem - we all live with "features" of
>> programming languages - just wondering.
>
> Heh, the reason is pretty simple, if not intuitive. When confronted
> with operations between arrays of different sizes, IDL will *truncate*
> the longest array to the shortest size. I think this rule is pretty
> general but somebody will probably pipe in with an exception.

    Yup. See below.
>
> That "limitation" can sometimes be used to your advantage. For
> example, when doing finite differencing, a[i+1] - a[i] for each
> element, normally you would write that like:
>
> diff = a(1:*) - a(0:n_elements(a)-2)
>
> The expression A(0:N_ELEMENTS(A)-2) removes that last element of the
> array A. But that is a little obscure. It's almost "better" to say,
>
> diff = a(1:*) - a(0:*) ; or
> diff = a(1:*) - a

    Here's an exception, sort of:

 a[*] = a[1:*] - a

The above will, in fact, get you an error message from IDL to the
effect that n_elements(RHS) ne n_elements(LHS). For consistency with
Craig's examples, one might like to see IDL understand that this
example should be handled in the same manner as

 a[0:n_elements(a[1:*] - a) - 1] = a[1:*] - a
             ^^^^^^^^^^^   ^^^^^^^^^^

In other words, one might want IDL to truncate the vector length meant
by "*" on the LHS to match the shorter RHS vector length.

>
> It saves keystrokes, avoids subscript clutter, etc. But why does it

That it does. Craig's latter example--the one with "- a" instead of "- a[0:n_elements(a) - 2]" may also get you some nasty surprises, perhaps undetected till too late, if you ever change earlier parts of the program in certain ways.

> work? The answer is that A(1:*) has one less element than A, so when
> A appears by itself in the above equation, it is automatically
> truncated by one, achieve the desired result.
>
That's right, but incomplete. The exception, as noted above, is that the truncation of vector length in an assignment statement applies *only* to the RHS. One might argue that the exception does not apply in a case like

> diff = a(1:*) - a(0:*) ; or
> diff = a(1:*) - a

but this is because a new target is being created to which the evaluated expression will be assigned, rather than storing the evaluated expression into part of an existing target. n_elements(diff) is undefined until after the first time the assignment is completed, so it's really not the same situation.

Scott Bennett, Comm. ASMELG, CFIAG
College of Oceanic and Atmospheric Sciences
Oregon State University
Corvallis, Oregon 97331