Subject: Re: Array multiplication: implicit loop query
Posted by Craig Markwardt on Mon, 13 Aug 2001 13:58:08 GMT
View Forum Message <> Reply to Message

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.

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
```

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

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------