Subject: Re: indexing 2-d vs. 3-d arrays...ARGH!
Posted by JD Smith on Tue, 19 Apr 2005 23:27:08 GMT
View Forum Message <> Reply to Message

On Tue, 19 Apr 2005 14:39:20 -0700, Henry wrote:

>
> Hey folks,
>
> This one tripped me up today. Wonder how many other places in code
> I've been messed up by it. I (vaguely) understand the logic behind it.
>
> Let's say you want to grab some points out of a 2-d image or a 3-d
> stack of images:
>
> To set the stage, some pretend images and indexes of the points to
> grab:
> arr2 = dindgen(10,10)
> arr3 = dindgen(10,10,5)
> index1 = [2,6]
> index2 = [3,7]
>
> Now,
> help, arr2[index1,index2]
> gives
> <Expression> DOUBLE = Array[2]
> which is what I would expect.
>
> BUT,
> help,arr3[index1, index2, 2]
> gives
> <Expression> DOUBLE = Array[2, 2]
> which is what screwed me up.
>
> IDL demands:
> help,arr3[index1, index2, replicate(2,2)]
> to get:
> <Expression> DOUBLE = Array[2]
>
>
> Get it? Got it? Good.

Why don't you have a read through:

http://dfanning.com/misc_tips/submemory.html

The bottom line? If all indexing arrays match in dimension, as a
special case, IDL "threads the list" and the result has the same size

as each index array. If they don't match, it "inflates the list", by matching things up one element at a time. So in your middle case, it sees:

[2,6],[3,7],2

and it says: OK, we're don't have matched index arrays. They must want 2,6,2; 6,7,2; 6,3,2; 6,7,2. You might argue you really wanted 2,3,2;6,7,2, and would therefore prefer IDL to sub-thread any subset of the indexing arrays which have matching dimensions. It's not a bad idea, it just isn't how IDL works.

JD

P.S. David, I hope you don't stop pulling those nuggets from the newgroup and sticking them on your webpage (or, perish the thought, let your webpage wither and die).
