Subject: Re: simple array math question
Posted by JD Smith on Thu, 23 Jan 2003 16:53:33 GMT
View Forum Message <> Reply to Message

On Fri, 17 Jan 2003 14:46:58 -0700, JD Smith wrote:

> On Thu, 16 Jan 2003 20:47:46 -0700, Craig Markwardt wrote:
>
>
>> Heinz Stege <reply_to_posting@arcor.de> writes:
>>> On Thu, 16 Jan 2003 14:05:27 -0600, "Sean Raffuse" <sean@me.wustl.edu>
>>> wrote:
>>>
>>>> >> a=[[1,2,3],[4,5,6],[7,8,9]]
>>>>
>>>> >> b=[1,2,3]
>>>>
>>>> What is the best (read, fastest) way to multiply b by each individual
>>>> row of a? I would like to return a result of:
>>>>
>>>> [[1,4,9],[4,10,18],[7,14,27]]
>>>
>>>
>>> result=a*b(*,intarr(3))
>>
>> WOW! I've never seen that! It scares me how cool that is. :-)
>>
>> Craig
>
>
> I may have to add that to the REBIN/REFORM tutorial. I'll see how fast
> it is first. It's definitely one of the more readeable ways to add a
> new trailing dimension. Doesn't work for leading or in-the-middle
> dimensions, as far as I can tell.
>
>

OK, I've performed some speed tests comparing the two methods:

Expanding 1st dimension:

```
d=findgen(j,k)

e=rebin(reform(d,1,j,k),i,j,k,/SAMPLE)
 vs.
e=(reform(d,1,j,k))[intarr(i),*,*]
```

Expanding middle dimension:

```
d=findgen(i,k)

e=rebin(reform(d,i,1,k),i,j,k,/SAMPLE)
 vs.
e=(reform(d,i,1,k))[*,intarr(j),*]
```

Expanding last dimension:

```
d=findgen(i,j)

e=rebin(d,i,j,k,/SAMPLE)
 vs.
e=d[*,*,intarr(k)]
```

I tested this for all permutations of (i,j,k)=(100,200,500)

When your arrays fit in memory, the rebin(reform) method is 2.9-3.5 times faster. Actually, it's remarkably close to 3.0 in all cases. However, when you begin to run out of memory, the intarr() method really begins to suffer, up to 25 times slower. I suspect this is because all the index arrays must be pre-computed in memory when "*" is used.

The one convenience of the slower method: you don't need to keep track of and enter the other two dimensions. However, since REBIN/REFORM (as of v5.5) now take a single dimension argument, this problem is minimized; you can save yourself the trouble like this:

```
dim=size(d,/dimension)
e=rebin(reform(d,[1,dim]),[i,dim],/SAMPLE)
```

or

```
e=rebin(d,[dim,k],/SAMPLE)
```

or

```
 e=rebin(reform(d,[dim[0],1,dim[2:*]]),[dim[0],j,dim[2:*]],/S AMPLE)
```

and this has the added advantage that you don't even need to know how many dimensions your arrays have, just where you'd like to add a dimension of some size.

JD
