Subject: Assignment, Form 5 (what they didn't tell you in the manuals)
Posted by David Ritscher on Wed, 18 Nov 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Today's matrix subscripting puzzler:
I'm noting that there are some extra rules govorning assignement statements where the right side contains subscripting expressions in more than one of the dimensions of a matrix. Given a 3-D array,
array(indgen(5), indgen(5), indgen(5)) yields a 5-element vector
array(indgen(5), 0, indgen(5)) yields a 5 x 1 x 5 matrix

I'm trying to create an index that allows me to extract vectors of information from a matrix that has been stored. Here is a simple example of the form of what I'm trying to extract. The matrix 'index' will put the matrix into the order I want.

```
threeD = indgen(3,4,5)
twoD = indgen(3,5)
index = [[indgen(15) mod 3], [indgen(15) / 3]]
```

This 'index' allows me to make a vector out of the ascending elements of the twoD matrix:

```
info, twoD(index(*,0), index(*,1))
<Expression>    INT       = Array(15)
print, twoD(index(*,0), index(*,1))
      0       1       2       3       4       5       6       7
8       9      10      11      12      13      14
```

How do I do the same thing, with the threeD matrix?:

```
info, threeD(index(*,0), *, index(*,1))
<Expression>    INT       = Array(15, 4, 15)
info, threeD(index(*,0), 0, index(*,1))
<Expression>    INT       = Array(15, 1, 15)
```

if I want to get a vector that would extract the first element of the second dimension, as I had hoped the above would do, I can use:

```
info, threeD(index(*,0), intarr(15), index(*,1))
print, threeD(index(*,0), intarr(15), index(*,1))
```

How do I do this, where I get a 2-D result where the second dimension of the incoming matrix becomes the second (and final) dimension of my outgoing matrix?

Here is a solution, done in long-hand:

```
print, [[ threeD(index(*,0), intarr(15), index(*,1))], $
       [threeD(index(*,0), intarr(15)+1, index(*,1))], $
       [threeD(index(*,0), intarr(15)+2, index(*,1))], $
```

```
        [threeD(index(*,0), intarr(15)+3, index(*,1))] ]
```

I can also do it with for-loops:

```
out = lonarr(15, 4)
for i=0,3 do out(0,i) = threeD(index(*,0), intarr(15)+i, index(*,1))
```

this does it, too:

```
for i=0,3 do out(0,i) = (reform(threeD(*,i,*)))(index(*,0), index(*,1))
```

I won't go into the details, but my actual matrices contain information on every heart beat, stored over a 5-day period. I'm trying to string this information together, in a channel-by-channel order.

Every now and then, with IDL and PV-Wave, one notes that you just can't quite get there from here... Does anyone see a true matrix solution to this?

Thanks,

David Ritscher

--
Cardiac Rhythm Management Laboratory
Department of Medicine
University of Alabama at Birmingham
B168 Volker Hall - 1670 University Boulevard
Birmingham AL 35294-0019
Tel: (205) 975-2122   Fax: (205) 975-4720
Email: davidNO.ritscherSPAM@bigfoot.com