Subject: Re: Matrix Multiplication in IDL
Posted by R.G. Stockwell on Mon, 04 Aug 2003 20:56:25 GMT
View Forum Message <> Reply to Message

"K Banerjee" <kbanerjee_NO_SPAM@ucwphilly.rr.com> wrote in message
news:fozXa.94997$852.70192@twister.nyc.rr.com...
> Folks,
>
> In IDL:
>
> VB> a = indgen(2, 4) - 2
> VB> print, a ; Actually prints transpose(a)
> -2 -1
> 0 1
> 2 3
> 4 5

IDL is column major, the above does not print transpose(a), it prints a.
i.e. a = indgen(2 columns, 4 rows)

Cheers,
bob

Subject: Re: Matrix Multiplication in IDL
Posted by James Kuyper on Mon, 04 Aug 2003 21:06:02 GMT
View Forum Message <> Reply to Message

K Banerjee wrote:
>
> Folks,
>
> In IDL:
>
> VB> a = indgen(2, 4) - 2
> VB> print, a ; Actually prints transpose(a)
> -2 -1
> 0 1
> 2 3
> 4 5

Not quite. You've defined a as being a 2x4 matrix, and that's exactly
what it prints. Specifically, you'll find that the elements are printed
as

a(0,0) a(1,0)

a(0,1) a(1,1)
a(0,2) a(1,2)
a(0,3) a(1,3)

If you interpret that as transpose(a), then you haven't adapted your thinking yet to IDL's way of handling arrays. To print transpose(a), type:

print, transpose(a)

> Keeping the above points in mind, I am trying to understand the
> IDL command:
>
> transpose(G)##G
>
> (In my case, G is (296 x 4).)

The fundamental confusing thing is that IDL's array operators use the opposite convention from IDL itself, as to the meaning of rows and columns. Rather than getting hung up on the defintions of rows, columns, and transposes, let's just investigate how it works.

a = INTARR(l,m)
b = INTARR(n,o)
c = a # b
d = b ## a

The matching rule is:

m eq n

Both c and d are (l,o) arrays. The contents of c[i,j] are the same as the contents of d[i,j]. Both of them are

total(transpose(a[i,*])*b[*,j]))

> In IDL, the above matrix product turns out to be (4 x 4). However,
> I was expecting the matrix product to be (296 x 296) since I
> interpret the above IDL command as carrying out the matrix
> multiplication:

To get a (296,269) array out of g, you must type either

h = g # TRANSPOSE(g)

or

h = TRANSPOSE(g) ## g