Subject: Re: Matrix algebra and index order, A # B vs A ## B
Posted by on Tue, 27 Mar 2012 07:45:40 GMT
View Forum Message <> Reply to Message

Den tisdagen den 27:e mars 2012 kl. 00:41:18 UTC+2 skrev Craig Markwardt:
> On Monday, March 26, 2012 9:45:51 AM UTC-4, Mats Löfdahl wrote:
>> On Monday, March 26, 2012 3:00:05 PM UTC+2, David Fanning wrote:
>>> Mats Löfdahl writes:
>>>
>>>> IDL has two operators for matrix multiplication, # and ##.
>>>> The former assumes the matrices involved have colum number as
>>>> the first index and row number as the second, i.e., A_{rc} =
>>>> A[c,r] with mathematics on the LHS and IDL on the RHS. The
>>>> latter operator makes the opposite assumption, A_{rc} = A[r,c].
>>>>
>>>> I believe much headache can be avoided if one chooses one
>>>> notation and sticks with it. If it were only me, I'd choose
>>>> the A_{rc} = A[r,c] notation. But it isn't only me, because
>>>> I like to take advantage of IDL routines written by others.
>>>> So, has there emerged some kind of consensus among influential
>>>> IDL programmers (those that write publicly available
>>>> routines that are widely used - thank you BTW!) for
>>>> which convention to use?
>>>
>>> Yes, the consensus that has emerged is that no operation
>>> is more fraught with ambiguity, anguish, and frustration
>>> than trying to translate a section of linear algebra code
>>> from a paper or textbook (say on Principle Components
>>> Analysis) to IDL than almost anything you can imagine!
>>> It's like practicing backwards writing in the mirror.
>>>
>>> And, of course, while you are doing it you have the
>>> growing realization that there is no freaking way you
>>> are EVER going to be able to write the on-line
>>> documentation to explain this dog's dish of a program
>>> to anyone else. :-(
>>>
>>> The solution, of course, is to stick with the ##
>>> notation for as long as it makes sense, then throw
>>> in a couple of # signs whenever needed to make the
>>> math come out right. :-)
>>
>> It's that bad? :o)
>>
>> One thing that had me wondering is the documentation for Craig Markwardt's qrfac routine:
>>
>>
>> ; Given an MxN matrix A (M>N), the procedure QRFAC computes the QR

>> ; decomposition (factorization) of A. This factorization is useful
>> ; in least squares applications solving the equation, A # x = B.
>> ; Together with the procedure QRSOLV, this equation can be solved in
>> ; a least squares sense.
>> ;
>> ; The QR factorization produces two matrices, Q and R, such that
>> ;
>> ;   A = Q ## R
>> ;
>> ; where Q is orthogonal such that TRANSPOSE(Q)##Q equals the identity
>> ; matrix, and R is upper triangular.
>>
>> The ## operator for the matrix-matrix multiplications but # for matrix-vector multiplication! But then I thought this might be IDL 1D arrays being interpreted as row vectors so x # A is actually just another way of writing A ## transpose(x). And the former would be more efficient. Am I on the right track here...?
>
> I believe I double checked the notation of QRFAC when I wrote it way back when.
>
> Maybe you need to read this part of the documentation as well....
>
> ;   Note that the dimensions of A in this routine are the
> ;   *TRANSPOSE* of the conventional appearance in the least
> ;   squares matrix equation.

Yes, but that doesn't help much when "the conventional appearance" is not defined...

> The transposed matrix means you flip all the #'s: # <--> ##.
>
> I realize this is very confusing, but unfortunately I inherited this code from somewhere else (MPFIT), so it retains the warts of the original.
>
> By the way, there's an example provided with the documentation, which you could test the notation for yourself.

Yes, of course. Sorry, I realize I gave the impression that I had problems running the qrfac program. I don't, trial and error solved that problem. But it got me thinking about it and I thought it might be nice to find out the most common convention and then perhaps stay sane by writing wrappers around routines that use the opposite convention (if I stumble upon any). At least when that can be done without much time penalty.

Anyway, if my notation on the math side is right I believe qrfac uses the A[r,c] notation. So that's one data point in favor of the ## operator. But the comment about "the conventional appearance" is then a data point against?
