Subject: Re: Generating a grid in the 3D,4D,5D...N space - Advice/Combinatory/Matrices
Posted by Markus Schmassmann on Tue, 14 Nov 2017 11:29:34 GMT
View Forum Message <> Reply to Message

On 11/14/2017 11:38 AM, Markus Schmassmann wrote:
> On 11/13/2017 03:50 PM, clement.feller@obspm.fr wrote:
>> I coming back to you for some advice on how to properly generate a
>> grid in an N-D space. I hope that this expression is the proper one in
>> english, but in any case, let me illustrate this by the following
>> exemple:
>>> a = indgen(3,3) & print, a
>> 0 1 2
>> 3 4 5
>> 6 7 8
>>
>> What I am looking for would be to find the clean and proper IDL way to
>> generate the following sets of combinations:
>> 0,1,2
>> 0,1,5
>> 0,1,8
>> 0,4,2
>> 0,4,5
>> .....
>> .....
>>
>> 6,7,2
>> 6,7,5
>> 6,7,8
>>
>> Now, I have found ways to do this for a 2D,3D,4D,5D space with either
>> nested loops (yuck! I know), or with combinations of rebin, reform and
>> transpose.
>> I've been successfully using those solutions for several weeks, yet I
>> wonder on how to expand this to a general case and in the proper IDL way.
>>
>> [...]
>>
>> I've playing around with nested indgen, looking for a repetitive
>> motive from the 2D to the 5D space when using rebin, reform, transpose
>> to assemble a generic command. But nothing much so far....
>>
>> Does anybody out there already had a go with such problem before or
>> any advice ?
> is this what you are looking for ?
>
> array=lindgen(n,long(n)^n)
> for k=0,n-1 do array[k,*]= $

> rebin((n*lindgen(n^(k+1))+k) mod (n^2),long(n)^n,/sample)
sorry, for n>5 you have an overflow, correct is:

```
array=bindgen(n,long(n)^n)
for k=0,n-1 do array[k,*]=rebin(byte( $
    (n*lindgen(long(n)^(k+1))+k) mod (n^2) ),long(n)^n, /sample )
```

it's also better on memory, for n=8 the index is only 16MB
