Subject: How does REFORM work in PV-Wave
Posted by jeyadev on Tue, 30 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

I have a question about the way reform works on 3d arrays.

First, the specific problem. I have a data file that has 8 columns and 56 rows. Each column corresponds to an independent variable. The The 56 rows correspond to an 8 x 7 array of points on a plane. What I would like to do is to read the data file and then make a new 3d array so that each plane of this 3d array corresponds to one of the independent variables (so that there are 8 separate 8 x 7 planes).

After some experimentation I found that the solution was

newdata = reform(data, 8, 8, 7)

But, I do not understand why this works. In fact, I find it quite counter intuitive and I do not understand why the 'plane' index should be the first one (more below). The fact that there is a degeneracy here made my experimentation easier! There are only 3 possibilities, instead of 6. But, that raises the more interesting question. Also, I understand that the first index of newdata identifies the *plane* for each variable and I can do, for example,

surface, reform(newdata(1,*,*))

to see how the second variable varies over the plane. Now,

if I had N variables on a grid with C columns and R rows and the data file was in the format

r1 c1 v1 v2 v3 ... vN
r2 c1 v1 v2 v3 ... vN
.....
.....
R c1 v1 v2 v3 ... vN
r1 c2   ....
r2 c2   ....

etc. In the above, you can ignore the first two columns -- they are there just to show the order. The actual data and just C x R rows of N elements each.

what would be the reform command be? (With that I should be able to figure out the answer for what it should be in the row and column orders were reversed in the original data file!)

I guess what I am missing is the *order* in which elements are stored.
I think that PV-Wave stores 2d arrays in the 'row' format (first index
varies fastest), but what about higher dimensional arrays?

thanks

--

Surendar Jeyadev jeyadev@wrc.xerox.com

Subject: Re: How does REFORM work in PV-Wave
Posted by jeyadev on Wed, 01 Dec 1999 08:00:00 GMT
View Forum Message <> Reply to Message

In article <mgs-52612D.20571630111999@news.silcom.com>,
Mike Schienle <mgs@ivsoftware.com> wrote:
> In article <8214p7kcq1@news.wrc.xerox.com>, jeyadev@wrc.xerox.com
> wrote:
>
> ...
>> I guess what I am missing is the *order* in which elements are stored.
>> I think that PV-Wave stores 2d arrays in the 'row' format (first index
>> varies fastest), but what about higher dimensional arrays?
>
> You can probably find more than you wanted to know abot row and column
> order by visiting the IDL FAQ at <http://www.ivsoftware.com:8000/FAQ/>.

Thanks, but I think that I need an FAQ to use the FAQ :-(
I got everything I do not need (e.g. What is IDL?, Where can I contact
them?, ... Do open contours fill?).

> Select the "Search FAQ" button. Enter the word "major" in the "Question"
> field and press the "Start Search" button. You'll be treated to a fairly
> detailed discussion on column- and row-major, as well as memory access
> into the arrays.
>
> Folks, as long as we're on the PV-WAVE subject, I have an old FAQ for
> PV-WAVE that was done in 1995. It's at
> <http://www.ivsoftware.com/pvwave_faq.html>. If someone would like to

More readable (I least I get to see what is in there), but it does not
answer my question. (Sob!)

> take it over, drop me a line. If someone just wants to update it, I can
> put it into the same format as the IDL FAQ and serve it from this site.

Appreciate the help.
--

Surendar Jeyadev jeyadev@wrc.xerox.com

Subject: Re: How does REFORM work in PV-Wave
Posted by thompson on Thu, 02 Dec 1999 08:00:00 GMT
View Forum Message <> Reply to Message

jeyadev@wrc.xerox.com (Surendar Jeyadev) writes:

> In article <mgs-52612D.20571630111999@news.silcom.com>,
> Mike Schienle <mgs@ivsoftware.com> wrote:
>>
>> You can probably find more than you wanted to know abot row and column
>> order by visiting the IDL FAQ at <http://www.ivsoftware.com:8000/FAQ/>.
>> Select the "Search FAQ" button. Enter the word "major" in the "Question"
>> field and press the "Start Search" button. You'll be treated to a fairly
>> detailed discussion on column- and row-major, as well as memory access
>> into the arrays.

> Found it, at last, by listing all the questions, but I know all *that*
> stuff.

> My question was what happens beyond 2 dimensions and how REFORM treats
> a 2d to 3d coversion. I will simplify my question in the hope that some
> kind soul will help me out.

> Let us say that I have the data file

> 1 13
> 2 14
> 3 15
> 4 16
> 5 17
> 6 18
> 7 19
> 8 20
> 9 21
> 10 22
> 11 23
> 12 24

> and that the first column represents data for a variable that is defined
> on a 3 x 4 (i.e. 3 column and 4 rows) grid and the second column is for
> another variable on the same grid. Assume that the data is stored in the
> the array odat(2,12).

> What is I want to do is the following: I want to create a 3 data array
> with two planes of 3 x 4 elements so that each plane contains the the data
> for one variable.

> The REAL QUESTION: The command

> data = reform(odat,2,3,4)

> seems to do the job. For example

> WAVE> a = data(0,*,*)
> WAVE> info, a
> A   INT   = Array(1, 3, 4)
> WAVE> a = reform(a)
> WAVE> info, a
> A   INT   = Array(3, 4)
> WAVE> print, a
>   1   2   3
>   4   5   6
>   7   8   9
>   10   11   12

> which is exactly what I want. Now, what I would like to know is why the
> number of planes (2) had to be the *first* index in the reform statement.

> thanks
> --

> Surendar Jeyadev   jeyadev@wrc.xerox.com

Surendar:

The basic answer is as follows. Your odat(2,12) array is really stored as a series of numbers. You can see the way the array is stored by the command

IDL> print,odat(*)
       1   13   2   14   3   15   4   16   5
      17   6   18   7   19   8   20   9   21
      10   22   11   23   12   24

By formatting this into a (2,12) array, you tell IDL to organize it into the indices

 odat(0,0) = 1
 odat(1,0) = 13
 odat(0,1) = 2
 odat(1,1) = 14

odat(0,2) = 3
odat(1,2) = 15
odat(0,3) = 4
    etc.

If you then reformat it into a (2,3,4) array, it will be stored as

odat(0,0,0) = 1
odat(1,0,0) = 13
odat(0,1,0) = 2
odat(1,1,0) = 14
odat(0,2,0) = 3
odat(1,2,0) = 15
odat(0,0,1) = 4
    etc.

The leftmost index always increases most rapidly, and the rightmost index always increases most slowly. The REFORM() function doesn't rearrange the numbers in memory--it just changes how they're interpreted.

William Thompson
