Subject: Re: Search single column of array - removing nasty loop
Posted by greg.addr on Tue, 29 Nov 2011 11:40:31 GMT
View Forum Message <> Reply to Message

I think this should be better...

```
q=where(array eq 0)
qq=array_indices(array,q)
for i=0,n_elements(q)-1 do array[qq[0,i],qq[1,i],qq[2,i],*]=0
```

If you often have many zeros in a single 4th-dim column, then it may be faster to check that the qq rows are unique before you make the loop.

Greg

Subject: Re: Search single column of array - removing nasty loop
Posted by Yngvar Larsen on Tue, 29 Nov 2011 12:40:20 GMT
View Forum Message <> Reply to Message

On Nov 29, 10:11 am, Rob <rj...@le.ac.uk> wrote:
> I was hoping that there was a nice way to do the following. I have a
> 4D array and I want to check if the 4th dimension contains a 0 in any
> of it's values for each value of the other 3 dimensions, if it does I
> want that whole column set to 0.
>
> This is how I'm doing it with a loop:
>
> FOR i=0, 1 DO BEGIN
> FOR k = 0, 359 DO BEGIN
> FOR j = 0, 5 DO BEGIN
> test = where(array[i,j,k,*] eq 0)
> IF max(test) gt -1 THEN array[i,j,k,*] = 0
> ENDFOR
> ENDFOR
> ENDFOR
>
> which is obviously horrible and slow.
>
> Any help/advice would be great.
>
> Thanks
>
> Rob

Interesting problem, which really depends a lot on the size of the 4th
dimension, and the number of expected zeros in the array.

(1) The original loop is better like this.

```
FOR k = 0, 359 DO FOR j = 0, 5 DO FOR i=0,1 DO $
  IF (total(array[i,j,k,*]) gt 0) THEN array[i,j,k,*] = 0
```

Here, we write it as a one-liner (no BEGIN/END), which is usually slightly faster in IDL. Also we avoid WHERE/MAX, and use instead TOTAL, which is very fast.

(2) If the number of zeros in the array is low, this one should be fast.

```
ind = where(array eq 0, count)
if (count gt 0) then begin
  dim = size(array, /dimensions)
  nrow = dim[0]*dim[1]*dim[2]
  ind mod= nrow
  ind = ind[uniq(ind[sort(ind)])] ; Unique rows containing zeros
  ind = array_indices(array,ind)
  for i=0L,n_elements(ind[0,*])-1 do
array[ind[0,i],ind[1,i],ind[2,i],*]=0
endif
```

(3) Since you are operating only on one dimension, it should really be the first one for efficiency reasons. So it is better to actually keep the data stored that way. If that is not possible, a transpose before and after the operation might help you:

```
array = transpose(temporary(array), [3,0,1,2]) ; Or keep the array
like this in the first place
dim = size(array, /dimensions)
nrow = dim[1]*dim[2]*dim[3]
array = reform(array, dim[0], nrow)
for ii=0L, nrow-1 do if (total(A[*,ii] eq 0) gt 0) then A[*,ii] = 0
array = reform(array, dim)
array = transpose(temporary(array), [1,2,3,0])
```

--
Yngvar

Subject: Re: Search single column of array - removing nasty loop
Posted by Yngvar Larsen on Tue, 29 Nov 2011 12:49:39 GMT
View Forum Message <> Reply to Message

On Nov 29, 1:40 pm, Yngvar Larsen <larsen.yng...@gmail.com> wrote:

> (1) The original loop is better like this.
>

> FOR k = 0, 359 DO FOR j = 0, 5 DO FOR i=0,1 DO $
> IF (total(array[i,j,k,*]) gt 0) THEN array[i,j,k,*] = 0

Bug fix. Should be:

FOR k = 0, 359 DO FOR j = 0, 5 DO FOR i=0,1 DO $
IF (total(array[i,j,k,*] eq 0) gt 0) THEN array[i,j,k,*] = 0

> (3) Since you are operating only on one dimension, it should really be
> the first one for efficiency reasons. So it is better to actually keep
> the data stored that way. If that is not possible, a transpose before
> and after the operation might help you:
[...]
> for ii=0L, nrow-1 do if (total(A[*,ii] eq 0) gt 0) then A[*,ii] = 0

Another bugfix. Should of course be
for ii=0L, nrow-1 do if (total(array[*,ii] eq 0) gt 0) then
array[*,ii] = 0

--
Yngvar

Subject: Re: Search single column of array - removing nasty loop
Posted by Heinz Stege on Tue, 29 Nov 2011 17:53:00 GMT
View Forum Message <> Reply to Message

Hi Rob,

no loop necessary:

```
array=(randomu(seed,2,6,360,42)-.1)>0.   ; sample array
array=reform(array,n_elements(array)/42,42,/overwrite)
ii=where(min(array,dim=2) eq 0.,count)
if count ge 1 then array[ii,*]=0.
array=reform(array,2,6,360,42,/overwrite)
```

Heinz

Subject: Re: Search single column of array - removing nasty loop
Posted by rjp23 on Wed, 30 Nov 2011 09:34:32 GMT
View Forum Message <> Reply to Message

On Nov 29, 5:53 pm, Heinz Stege <public.215....@arcor.de> wrote:

> Hi Rob,
>
> no loop necessary:
>
> array=(randomu(seed,2,6,360,42)-.1)>0. ; sample array
> array=reform(array,n_elements(array)/42,42,/overwrite)
> ii=where(min(array,dim=2) eq 0.,count)
> if count ge 1 then array[ii,*]=0.
> array=reform(array,2,6,360,42,/overwrite)
>
> Heinz

That's perfect. Out of interest, why the /overwrite? I've not seen reform used with that before.

Thanks to everyone for their help :-)

Rob

Subject: Re: Search single column of array - removing nasty loop
Posted by Yngvar Larsen on Wed, 30 Nov 2011 20:15:38 GMT
View Forum Message <> Reply to Message

On Nov 29, 6:53 pm, Heinz Stege <public.215....@arcor.de> wrote:
> Hi Rob,
>
> no loop necessary:
>
> array=(randomu(seed,2,6,360,42)-.1)>0. ; sample array
> array=reform(array,n_elements(array)/42,42,/overwrite)
> ii=where(min(array,dim=2) eq 0.,count)
> if count ge 1 then array[ii,*]=0.
> array=reform(array,2,6,360,42,/overwrite)

Hm. The /OVERWRITE keyword to REFORM was new to me. Thanks!

Silly me. I have somehow always imagined that the compiler was smart enough to do this (i.e. not copy any data, only alter the internal IDL descriptor of the ARRAY variable) automatically when input and output to REFORM is the same variable. But a bit of profiling shows this is not at all the case. This will be _very_ useful many places in my operational code...

A small comment to the code above: "where(min(array,dim=2) eq 0.)" obviously only works if array contains only non-negative data. If not, "where(total(array eq 0, 2) gt 0)" will do the trick also for floating point data containing negative numbers, with more or less the same

performance.

--
Yngvar

Subject: Re: Search single column of array - removing nasty loop
Posted by rjp23 on Thu, 01 Dec 2011 10:37:13 GMT
View Forum Message <> Reply to Message

On Nov 30, 8:15 pm, Yngvar Larsen <larsen.yng...@gmail.com> wrote:
> On Nov 29, 6:53 pm, Heinz Stege <public.215....@arcor.de> wrote:
>
>> Hi Rob,
>
>> no loop necessary:
>
>> array=(randomu(seed,2,6,360,42)-.1)>0. ; sample array
>> array=reform(array,n_elements(array)/42,42,/overwrite)
>> ii=where(min(array,dim=2) eq 0.,count)
>> if count ge 1 then array[ii,*]=0.
>> array=reform(array,2,6,360,42,/overwrite)
>
> Hm. The /OVERWRITE keyword to REFORM was new to me. Thanks!
>
> Silly me. I have somehow always imagined that the compiler was smart
> enough to do this (i.e. not copy any data, only alter the internal IDL
> descriptor of the ARRAY variable) automatically when input and output
> to REFORM is the same variable. But a bit of profiling shows this is
> not at all the case. This will be _very_ useful many places in my
> operational code...
>
> A small comment to the code above: "where(min(array,dim=2) eq 0.)"
> obviously only works if array contains only non-negative data. If not,
> "where(total(array eq 0, 2) gt 0)" will do the trick also for floating
> point data containing negative numbers, with more or less the same
> performance.
>
> --
> Yngvar

Thanks, that explains why a few results were coming out slightly
differently as there are a few negative values.

Also, the code fails when the final column only has 1 element in it.

IDL> help, array

ARRAY DOUBLE = Array[4320, 1]
IDL> help, total(array eq 0, 2)
% TOTAL: For input argument <BYTE Array[4320]>, Dimension must be
1.

Subject: Re: Search single column of array - removing nasty loop
Posted by Yngvar Larsen on Thu, 01 Dec 2011 12:00:26 GMT
View Forum Message <> Reply to Message

On Dec 1, 11:37 am, Rob <rj...@le.ac.uk> wrote:
> On Nov 30, 8:15 pm, Yngvar Larsen <larsen.yng...@gmail.com> wrote:
>
>
>
>
>
>
>
>
>
>
>> On Nov 29, 6:53 pm, Heinz Stege <public.215....@arcor.de> wrote:
>
>>> Hi Rob,
>
>>> no loop necessary:
>
>>> array=(randomu(seed,2,6,360,42)-.1)>0. ; sample array
>>> array=reform(array,n_elements(array)/42,42,/overwrite)
>>> ii=where(min(array,dim=2) eq 0.,count)
>>> if count ge 1 then array[ii,*]=0.
>>> array=reform(array,2,6,360,42,/overwrite)
>
>> Hm. The /OVERWRITE keyword to REFORM was new to me. Thanks!
>
>> Silly me. I have somehow always imagined that the compiler was smart
>> enough to do this (i.e. not copy any data, only alter the internal IDL
>> descriptor of the ARRAY variable) automatically when input and output
>> to REFORM is the same variable. But a bit of profiling shows this is
>> not at all the case. This will be _very_ useful many places in my
>> operational code...
>
>> A small comment to the code above: "where(min(array,dim=2) eq 0.)"
>> obviously only works if array contains only non-negative data. If not,
>> "where(total(array eq 0, 2) gt 0)" will do the trick also for floating
>> point data containing negative numbers, with more or less the same
>> performance.
>

>> --
>> Yngvar
>
> Thanks, that explains why a few results were coming out slightly
> differently as there are a few negative values.
>
> Also, the code fails when the final column only has 1 element in it.
>
> IDL> help, array
> ARRAY DOUBLE = Array[4320, 1]
> IDL> help, total(array eq 0, 2)
> % TOTAL: For input argument <BYTE Array[4320]>, Dimension must be
> 1.

If the final column has only 1 element, the operation is not necessary
at all since all elements are already 0 :)

IDL sometimes behaves rather idiotic with singleton dimensions:

IDL> help, fltarr(4320, 1)
<Expression> FLOAT = Array[4320]

This is a problem when arrays are expected to be 2D, and suddenly are
automatically 1D. You can avoid it by adding an explicit REFORM
statement at the appropriate place in the code:

```
;; Force ARRAY to be 2D always
if (size(array, /n_dimensions) eq 1) then $
  array = reform(array, n_elements(array), 1, /overwrite)
```

--
Yngvar

Subject: Re: Search single column of array - removing nasty loop
Posted by rjp23 on Thu, 01 Dec 2011 12:10:12 GMT
View Forum Message <> Reply to Message

On Dec 1, 12:00 pm, Yngvar Larsen <larsen.yng...@gmail.com> wrote:
> On Dec 1, 11:37 am, Rob <rj...@le.ac.uk> wrote:
>
>
>
>
>
>
>
>
>

>
>> On Nov 30, 8:15 pm, Yngvar Larsen <larsen.yng...@gmail.com> wrote:
>
>>> On Nov 29, 6:53 pm, Heinz Stege <public.215....@arcor.de> wrote:
>
>>>> Hi Rob,
>
>>>> no loop necessary:
>
>>>> array=(randomu(seed,2,6,360,42)-.1)>0. ; sample array
>>>> array=reform(array,n_elements(array)/42,42,/overwrite)
>>>> ii=where(min(array,dim=2) eq 0.,count)
>>>> if count ge 1 then array[ii,*]=0.
>>>> array=reform(array,2,6,360,42,/overwrite)
>
>>> Hm. The /OVERWRITE keyword to REFORM was new to me. Thanks!
>
>>> Silly me. I have somehow always imagined that the compiler was smart
>>> enough to do this (i.e. not copy any data, only alter the internal IDL
>>> descriptor of the ARRAY variable) automatically when input and output
>>> to REFORM is the same variable. But a bit of profiling shows this is
>>> not at all the case. This will be _very_ useful many places in my
>>> operational code...
>
>>> A small comment to the code above: "where(min(array,dim=2) eq 0.)"
>>> obviously only works if array contains only non-negative data. If not,
>>> "where(total(array eq 0, 2) gt 0)" will do the trick also for floating
>>> point data containing negative numbers, with more or less the same
>>> performance.
>
>>> --
>>> Yngvar
>
>> Thanks, that explains why a few results were coming out slightly
>> differently as there are a few negative values.
>
>> Also, the code fails when the final column only has 1 element in it.
>
>> IDL> help, array
>> ARRAY DOUBLE = Array[4320, 1]
>> IDL> help, total(array eq 0, 2)
>> % TOTAL: For input argument <BYTE Array[4320]>, Dimension must be
>> 1.
>
> If the final column has only 1 element, the operation is not necessary
> at all since all elements are already 0 :)
>
> IDL sometimes behaves rather idiotic with singleton dimensions:

>
> IDL> help, fltarr(4320, 1)
> <Expression>   FLOAT   = Array[4320]
>
> This is a problem when arrays are expected to be 2D, and suddenly are
> automatically 1D. You can avoid it by adding an explicit REFORM
> statement at the appropriate place in the code:
>
> ;; Force ARRAY to be 2D always
> if (size(array, /n_dimensions) eq 1) then $
>   array = reform(array, n_elements(array), 1, /overwrite)
>
> --
> Yngvar

I'm not sure if that's the solution as the array was already 2D:

>> IDL> help, array
>> ARRAY   DOUBLE   = Array[4320, 1]

Subject: Re: Search single column of array - removing nasty loop
Posted by Yngvar Larsen on Thu, 01 Dec 2011 12:44:42 GMT
View Forum Message <> Reply to Message

On Dec 1, 1:10 pm, Rob <rj...@le.ac.uk> wrote:
> On Dec 1, 12:00 pm, Yngvar Larsen <larsen.yng...@gmail.com> wrote:
>
>
>
>
>
>
>
>
>
>
>> On Dec 1, 11:37 am, Rob <rj...@le.ac.uk> wrote:
>
>>> On Nov 30, 8:15 pm, Yngvar Larsen <larsen.yng...@gmail.com> wrote:
>
>>>> On Nov 29, 6:53 pm, Heinz Stege <public.215....@arcor.de> wrote:
>
>>>> > Hi Rob,
>
>>>> > no loop necessary:
>
>>>> > array=(randomu(seed,2,6,360,42)-.1)>0. ; sample array
>>>> > array=reform(array,n_elements(array)/42,42,/overwrite)

>>>> > ii=where(min(array,dim=2) eq 0.,count)
>>>> > if count ge 1 then array[ii,*]=0.
>>>> > array=reform(array,2,6,360,42,/overwrite)
>
>>>> Hm. The /OVERWRITE keyword to REFORM was new to me. Thanks!
>
>>>> Silly me. I have somehow always imagined that the compiler was smart
>>>> enough to do this (i.e. not copy any data, only alter the internal IDL
>>>> descriptor of the ARRAY variable) automatically when input and output
>>>> to REFORM is the same variable. But a bit of profiling shows this is
>>>> not at all the case. This will be _very_ useful many places in my
>>>> operational code...
>
>>>> A small comment to the code above: "where(min(array,dim=2) eq 0.)"
>>>> obviously only works if array contains only non-negative data. If not,
>>>> "where(total(array eq 0, 2) gt 0)" will do the trick also for floating
>>>> point data containing negative numbers, with more or less the same
>>>> performance.
>
>>>> --
>>>> Yngvar
>
>>> Thanks, that explains why a few results were coming out slightly
>>> differently as there are a few negative values.
>
>>> Also, the code fails when the final column only has 1 element in it.
>
>>> IDL> help, array
>>> ARRAY   DOUBLE   = Array[4320, 1]
>>> IDL> help, total(array eq 0, 2)
>>> % TOTAL: For input argument <BYTE   Array[4320]>, Dimension must be
>>> 1.
>
>> If the final column has only 1 element, the operation is not necessary
>> at all since all elements are already 0 :)
>
>> IDL sometimes behaves rather idiotic with singleton dimensions:
>
>> IDL> help, fltarr(4320, 1)
>> <Expression>   FLOAT   = Array[4320]
>
>> This is a problem when arrays are expected to be 2D, and suddenly are
>> automatically 1D. You can avoid it by adding an explicit REFORM
>> statement at the appropriate place in the code:
>
>> ;; Force ARRAY to be 2D always
>> if (size(array, /n_dimensions) eq 1) then $
>>   array = reform(array, n_elements(array), 1, /overwrite)

>
>> --
>> Yngvar
>
> I'm not sure if that's the solution as the array was already 2D:
>
>>> IDL> help, array
>>> ARRAY DOUBLE = Array[4320, 1]

Right. I suspected something like that. That's why I qualified it with
"...at the appropriate place in the code" :)

Your problem is this rather strange behavior:

IDL> help, array
ARRAY FLOAT = Array[4320, 1]
IDL> help, array eq 0
<Expression> BYTE = Array[4320]

So the solution is:
```
;;...
tmp = array eq 0
;; Force TMP to be 2D always
if (size(tmp, /n_dimensions) eq 1) then $
  tmp = reform(tmp, n_elements(tmp), 1, /overwrite)
ii = where(total(tmp, 2) gt 0, count)
;;...
```

--
Yngvar

Subject: Re: Search single column of array - removing nasty loop
Posted by Yngvar Larsen on Thu, 01 Dec 2011 12:46:38 GMT
View Forum Message <> Reply to Message

On Dec 1, 1:10 pm, Rob <rj...@le.ac.uk> wrote:
> On Dec 1, 12:00 pm, Yngvar Larsen <larsen.yng...@gmail.com> wrote:
>
>
>
>
>
>
>
>
>

>> On Dec 1, 11:37 am, Rob <rj...@le.ac.uk> wrote:
>
>>> On Nov 30, 8:15 pm, Yngvar Larsen <larsen.yng...@gmail.com> wrote:
>
>>>> On Nov 29, 6:53 pm, Heinz Stege <public.215....@arcor.de> wrote:
>
>>>> > Hi Rob,
>
>>>> > no loop necessary:
>
>>>> > array=(randomu(seed,2,6,360,42)-.1)>0. ; sample array
>>>> > array=reform(array,n_elements(array)/42,42,/overwrite)
>>>> > ii=where(min(array,dim=2) eq 0.,count)
>>>> > if count ge 1 then array[ii,*]=0.
>>>> > array=reform(array,2,6,360,42,/overwrite)
>
>>>> Hm. The /OVERWRITE keyword to REFORM was new to me. Thanks!
>
>>>> Silly me. I have somehow always imagined that the compiler was smart
>>>> enough to do this (i.e. not copy any data, only alter the internal IDL
>>>> descriptor of the ARRAY variable) automatically when input and output
>>>> to REFORM is the same variable. But a bit of profiling shows this is
>>>> not at all the case. This will be _very_ useful many places in my
>>>> operational code...
>
>>>> A small comment to the code above: "where(min(array,dim=2) eq 0.)"
>>>> obviously only works if array contains only non-negative data. If not,
>>>> "where(total(array eq 0, 2) gt 0)" will do the trick also for floating
>>>> point data containing negative numbers, with more or less the same
>>>> performance.
>
>>>> --
>>>> Yngvar
>
>>> Thanks, that explains why a few results were coming out slightly
>>> differently as there are a few negative values.
>
>>> Also, the code fails when the final column only has 1 element in it.
>
>>> IDL> help, array
>>> ARRAY DOUBLE = Array[4320, 1]
>>> IDL> help, total(array eq 0, 2)
>>> % TOTAL: For input argument <BYTE Array[4320]>, Dimension must be
>>> 1.
>
>> If the final column has only 1 element, the operation is not necessary
>> at all since all elements are already 0 :)
>

>> IDL sometimes behaves rather idiotic with singleton dimensions:
>
>> IDL> help, fltarr(4320, 1)
>> <Expression>   FLOAT   = Array[4320]
>
>> This is a problem when arrays are expected to be 2D, and suddenly are
>> automatically 1D. You can avoid it by adding an explicit REFORM
>> statement at the appropriate place in the code:
>
>> ;; Force ARRAY to be 2D always
>> if (size(array, /n_dimensions) eq 1) then $
>>   array = reform(array, n_elements(array), 1, /overwrite)
>
>> --
>> Yngvar
>
> I'm not sure if that's the solution as the array was already 2D:
>
>
>
>
>
>
>
>>> IDL> help, array
>>> ARRAY   DOUBLE   = Array[4320, 1]

Right. I suspected something like that. That's why I qualified it with
"...at the appropriate place in the code" :)

Your problem is this rather strange behavior:

IDL> help, array
ARRAY   FLOAT   = Array[4320, 1]
IDL> help, array eq 0
<Expression>   BYTE   = Array[4320]

So the solution is:
;;...
tmp = array eq 0
;; Force TMP to be 2D always
if (size(tmp, /n_dimensions) eq 1) then $
   tmp = reform(tmp, n_elements(tmp), 1, /overwrite)
ii = where(total(tmp, 2) gt 0, count)
;;...

--