
Subject: Problem array subscripting

Posted by [GRI](#) on Mon, 16 Aug 1999 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello everybody.

I have a little problem I want to solve using array subscripting (if possible), but I haven't been able to do it without using a FOR statement:

I have a 3D array (dimX, dimY, n_images), which represents a time-series of images.

I have a masking image (dimX, dimY), from which I extract some points I am interested in (a ROI) with the command WHERE.

I could then change the one dimensional subscripts returned by WHERE into two vectors, that give me the X and Y coordinates of the points.

What I want to do is:

```
index= INDGEN(n_images)
result= ARRAY[X, Y, index],
```

but I get an error:

```
% All array subscripts must be same size. Var = ARRAY.
```

I have read in the documentation I can't do

```
result= ARRAY[X,Y, 0:n_images-1]
```

because I get an (n,n,n_images); where n is the number of X (or Y) coordinates array, instead of the expected (n,n_images) array. According to IDL's help this combines each element in the first vector with all elements in the second vector, so it would work for me if my points were in a regular grid, but this is not the case!

With a FOR statement the solution would be easy:

```
n_points= N_ELEMENTS(X) ; number of points I am interested on.
result= FLTARR(n_points, n_images)
FOR i=0, n_points-1 DO $
  result[i,*]= ARRAY[X[i], Y[i], 0:n_images-1]
```

Is there a way to solve it with array subscripts, or should I go ahead with the FOR statement?.

Subject: Re: Problem array subscripting

Posted by [Craig Markwardt](#) on Mon, 16 Aug 1999 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

GRI <dibeas@etsit.upm.es> writes:

>
> I have a little problem I want to solve using array subscripting (if
> possible), but I haven't been able to do it without using a FOR
> statement:
>
> I have a 3D array (dimX, dimY, n_images), which represents a time-series
> of images.
>
> I have a masking image (dimX, dimY), from which I extract some points I
> am interested in (a ROI) with the command WHERE.
>
> I could then change the one dimensional subscripts returned by WHERE
> into two vectors, that give me the X and Y coordinates of the points.
>
> What I want to do is:
>
> index= INDGEN(n_images)
> result= ARRAY[X, Y, index],

First of all, I would argue that using a FOR loop in this situation is acceptable. Why? Because you are still vectorizing the majority of the computation at the image level. The extra computational overhead of the FOR loop for a few images will be small.

That said, if you *really* want to vectorize, then I think you need to take a different approach. Array subscripting with index lists can get tricky if you mix with other forms of subscripting at the same time. The WHERE command, operated on a two dimensional array, gives a *one* dimensional list of array indices. Therefore, it's usually best to refer to your data the same way, ala REFORM.

If MASK is (NX x NY) and ARRAY is (NX x NY x N_IMAGES) then this should work.

```
wh = where(MASK EQ 1, n_pix) ;; One-dimensional array
array = reform(array, nx*ny, n_images, /overwrite)
result = array(wh, *)
array = reform(array, nx, ny, n_images, /overwrite) ;; Revert to old dims.
```

It may look ugly, but it's pretty fast since the
REFORM(...,/OVERWRITE) operates on the data in place. The dimensions
of RESULT are (N_PIX x N_IMAGES) where N_PIX is the number of pixels
selected by WHERE.

Good luck,

Craig

--

Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response

Subject: Re: Problem array subscripting
Posted by [m218003](#) on Mon, 16 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

In article <37B7FF76.F5062466@etsit.upm.es>,
GRI <dibeas@etsit.upm.es> writes:
> Hello everybody.
>
> I have a 3D array (dimX, dimY, n_images), which represents a time-series
>
> result= ARRAY[X,Y, 0:n_images-1]
>
>
how about result = (ARRAY[X,Y,*]) [*,*,0:n_images-1]

You could also try my little arrex function which was developed from a
request on this newsgroup and can be found at
<http://www-as.harvard.edu/people/staff/mgs/idl/> (hopefully still there)

(download the tools library)

Regards,
Martin.

Subject: Re: Problem array subscripting
Posted by [davidf](#) on Tue, 17 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Henry Chapman (chapman9@llnl.gov) writes:

```
> This can be solved with a little trick I posted awhile back (some time
> after a long thread about indices). The trick is to index an extra
> dimension.
>
> Example:
>
> IDL> array = bindgen(16,16,5)
> IDL> index = indgen(5)
> IDL> x = [0,1,2]
> IDL> y = [8,9,10,11]
>
> IDL> result = array[x, y, index, 0]
```

The trouble with this trick is that I don't understand how it works. It's fine for production code, don't get me wrong, where a little smoke and mirrors can even be elegant. I just don't want to get up in front of an IDL programming class and have to explain it. Do you have any theories?

Cheers,

David

--

David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Problem array subscripting
Posted by [Henry Chapman](#) on Tue, 17 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

This can be solved with a little trick I posted awhile back (some time after a long thread about indices). The trick is to index an extra dimension.

Example:

```
IDL> array = bindgen(16,16,5)
IDL> index = indgen(5)
IDL> x = [0,1,2]
IDL> y = [8,9,10,11]
```

```
IDL> result = array[x, y, index, 0]
```

```
IDL> help,result  
RESULT      BYTE      = Array[3, 4, 5]
```

```
IDL> print,result[*,* ,0]  
128 129 130  
144 145 146  
160 161 162  
176 177 178
```

You could also do

```
IDL> result = array[x, y, *, 0]
```

Henry.

--

Henry Chapman <mailto:chapman9@llnl.gov>
Information Science & Technology Program
Lawrence Livermore National Lab
L-395, 7000 East Ave., Livermore CA 94550

Subject: Re: Problem array subscripting
Posted by [GRI](#) on Wed, 18 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

I think I did not explain myself very well the first time... I'll try again

What I want to do is to be able to get the evolution along time of some pixels of a given image series. For instance, I want to see what values pixel at the origin (x=0, y=0) takes as time goes by.

Suppose I define:

```
array= INDGEN(2,2,5) ; dimensions (x,y,t)
```

I want to extract vectors along the time dimensions, i.e. of length 5, at any given combination of x and y.

For instance, I may need the vector at x=0, y=0 (from t=0 to t=4), and also another vector at x=1, y=1 (I obtain the x and y coordinates from a masking image)

This vectors would be, in this case [0,4,8,12,16] and [3,7,11,15,19].

When using a combination of two vectors to index ARRAY like

```
array[[0,1],[0,1],*]
```

what IDL does is combine each value in the first vector with all elements in the second vector ([0,0], [0,1], [1,0], [1,1]), and all values in the third dimension, as indicated by the asterisk, and so I get (in this case) the original array.

In a general case this would be ok if I wanted to profile the array along the time dimension at a regular grid ([0,0], [10,0], [20,0], [0,10], [10,10], [20,10] ...)

The only method I have proved it works like I want is the 'ugly' one proposed by Craig Markwardt. I have also seen it is really faster than using a for loop when the number of vectors to extract is high. Looks like the use of the /OVERWRITE keyword makes it work very fast, since as IDL's help says it only changes the data descriptor, not the data itself.

The other two methods give me the same results as indexing the time series directly with [[0,1],[0,1],*].

I haven't been able to download Martin Schultz's arrex function (might be a problem with my browser, the tools library is there, but instead of downloading it my Netscape opens it like if it were text, which is not).

Thanks everybody for your help.

P.S.: I hope I have explained this time, but as I read this again it looks a lot like my original post... well, at least this time I have included the little example!

Subject: Re: Problem array subscripting
Posted by [davidf](#) on Wed, 18 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Henry Chapman (chapman9@llnl.gov) writes a lengthy explanation for his array subscripting trick:

Well, as I say, I'm not anxious to get up in front of an IDL programming class and explain it. :-(

Cheers,

David

P.S. But I *would* consider giving them your e-mail address so *you* can explain it. :-)

--

David Fanning, Ph.D.

Fanning Software Consulting

Phone: 970-221-0438 E-Mail: davidf@dfanning.com

Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Problem array subscripting

Posted by [Henry Chapman](#) on Wed, 18 Aug 1999 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

David Fanning wrote:

```
>> IDL> result = array[x, y, index, 0]
```

```
>
```

```
> The trouble with this trick is that I don't understand
> how it works. It's fine for production code, don't get
> me wrong, where a little smoke and mirrors can even be
> elegant. I just don't want to get up in front of an IDL
> programming class and have to explain it. Do you have
> any theories?
```

Well, here's what's written in "Building IDL applications" on pp 62,63:

---- [begin quote from building.pdf] ---

"Extra" Dimensions

When creating arrays, IDL eliminates all size 1, or "degenerate", trailing dimensions. Thus, the statements

```
A = INTARR(10, 1)
HELP, A
```

print the following:

```
A INT = Array(10)
```

This removal of superfluous dimensions is usually convenient, but it can cause problems when attempting to write fully general procedures and functions. Therefore, IDL allows you to specify "extra" dimensions for

an array as long as the extra dimensions are all zero. For example, consider a vector defined as follows:

```
ARR = INDGEN(10)
```

The following are all valid references to the sixth element of ARR:

```
X = ARR[5]
X = ARR[5, 0]
X = ARR[5, 0, 0, *, 0]
```

Thus, the automatic removal of degenerate trailing dimensions does not cause problems for routines that attempt to access the resulting array.

----- [end quote] ---

So, IDL says that it's all Kosher to put in the extra dimension.

But, I guess, that doesn't quite answer all of it, since the above doesn't address the behaviour of subscripting. Let me quote some more from building.pdf (p 68):

---- [start quote]

When combining two subscript arrays, each element of the first array is combined with the corresponding element of the other subscript array. The two subscript arrays must have the same number of elements. The resulting subscript array has the same number of elements as its constituents. For example, the expression A[[1, 3], [5, 9]] yields the elements A[1,5] and A[3,9].

---- [end quote]

That is, when you try and do

```
result = array[x, y]
```

```
IDL wants to return result[0] = array[x[0], y[0]], ....
result[i] = array[x[i], y[i]], ...
```

And this may well be what you want. So how does one switch between these behaviours? There is a hint on p. 67 of the same good book:

-- [start quote]

When combining an array subscript with a subscript range, the result is

>
> Suppose I define:
>
> array= INDGEN(2,2,5) ; dimensions (x,y,t)
>
> I want to extract vectors along the time dimensions, i.e. of length 5,
> at any given combination of x and y.
>
> For instance, I may need the vector at x=0, y=0 (from t=0 to t=4), and also
> another vector at x=1, y=1 (I obtain the x and y coordinates from a masking
> image)
>
> This vectors would be, in this case [0,4,8,12,16] and [3,7,11,15,19].
>
> When using a combination of two vectors to index ARRAY like
>
> array[[0,1],[0,1],*]
>
> what IDL does is combine each value in the first vector with all
> elements in the second vector ([0,0], [0,1], [1,0], [1,1]), and all
> values in the third dimension, as indicated by the asterisk, and so
> I get (in this case) the original array.
>
> In a general case this would be ok if I wanted to profile the array along
> the time dimension at a regular grid ([0,0], [10,0], [20,0], [0,10],
> [10,10], [20,10] ...)
>
> The only method I haved proved it works like I want is the 'ugly' one
> proposed by Craig Markwardt. I have also seen it is really faster than
> using a for loop when the number of vectors to extract is hight. Looks
> like the use of the /OVERWRITE keyword makes it work very fast, since as
> IDL's help says it only changes the data descriptor, not the data itself.
>
> The other two methods give me the same results as indexing the time series
> directly with [[0,1],[0,1],*].
>
> I haven't been able to download Martin Schultz's arrex function (might be a
> problem with my browser, the tools library is there, but instead of
> downloading
> it my Netscape opens it like if it were text, which is not).

I've been scratching my head over this one for a while, and I don't see how this particular problem can be solved using traditional IDL array indexing syntax, even with added degenerate dimensions. I don't think arrex will solve it either (I'll be happy to be proven wrong).

Say you have the following arrays defined:

```
a = indgen(2, 2, 5)
x = [0, 1]
y = [0, 1]
z = [0, 1, 2, 3, 4]
```

When you tell IDL to extract `a[x, y, z]`, it extracts the following elements:

```
0, 0, 0
1, 0, 0
0, 1, 0
1, 1, 0
0, 0, 1
1, 0, 1
0, 1, 1
1, 1, 1
0, 0, 2
1, 0, 2
0, 1, 2
1, 1, 2
0, 0, 3
1, 0, 3
0, 1, 3
1, 1, 3
0, 0, 4
1, 0, 4
0, 1, 4
1, 1, 4
```

You can see that it loops through the supplied index arrays an element at a time, with the leftmost index array in the innermost loop. The elements you really wanted were

```
0, 0, 0
1, 1, 0
0, 0, 1
1, 1, 1
0, 0, 2
1, 1, 2
0, 0, 3
1, 1, 3
0, 0, 4
1, 1, 4
```

I don't believe you can extract this sub-array using normal array indexing. It requires creative array re-arrangement, as Craig demonstrated rather elegantly IMHO.

Array indexing is always good for a long thread in this newsgroup.....

Cheers,
Liam.

--

Liam E. Gumley
Space Science and Engineering Center, UW-Madison
<http://cimss.ssec.wisc.edu/~gumley>

Subject: Re: Problem array subscripting
Posted by [davidf](#) on Thu, 19 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Henry Chapman (chapman9@llnl.gov) writes:

>> P.S. But I *would* consider giving them your e-mail address
>> so *you* can explain it. :-)
>
> Is that your definition of being famous?

Uh, no. My definition of being famous is to throw
one of Stein Vidar's no-loop constructions up on the
board and say to the class, "Like that, of course!" :-)

Someone pointed out to me the other day that I...
uh, *overused* the phrase "of course" in my book.
They said nothing is "of course" to a beginning IDL
user. So I want you to know that I went through the
whole damn book and eliminated all 5,438 instances
(or whatever it was) of them. I'm calling this the
2nd (Less Pedantic) Edition. :-)

Cheers,

David

--

David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Problem array subscripting

Posted by [Henry Chapman](#) on Thu, 19 Aug 1999 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

David Fanning wrote:

- > P.S. But I *would* consider giving them your e-mail address
- > so *you* can explain it. :-)

Is that your definition of being famous?

Henry.

--

Henry Chapman <mailto:chapman9@llnl.gov>
Information Science & Technology Program
Lawrence Livermore National Lab
L-395, 7000 East Ave., Livermore CA 94550

Subject: Re: Problem array subscripting

Posted by [Mirko Vukovic](#) on Thu, 19 Aug 1999 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

In article <37BB1623.DEC726F8@etsit.upm.es>,

GRI <dibeas@etsit.upm.es> wrote:

... stuff deleted

- > I haven't been able to download Martin Schultz's arrex function (might be a
- > problem with my browser, the tools library is there, but instead of
- > downloading
- > it my Netscape opens it like if it were text, which is not).
- >

In that case you can use Save As (under file menu) to save the file.

If I remember correctly (I cannot test this now, as I am typing this in Netscape), as long as you select the file type as something else other than html, you should get a usefull text file, which after some editing on your part becomes you .pro file.

good luck

Mirko

Sent via Deja.com <http://www.deja.com/>

Share what you know. Learn what you don't.

Subject: Re: Problem array subscripting
Posted by [davidf](#) on Fri, 20 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Martin.Schultz@dkrz.de (m218003@modell3.dkrz.de) writes:

>> [...]. So I want you to know that I went through the
>> whole damn book and eliminated all 5,438 instances
>> (or whatever it was) of them [of course]. I'm calling this the
>> 2nd (Less Pedantic) Edition. :-)
>
> Wow! That's 48942 characters! Enough space for a whole new chapter I should
> think ;-)

Gee, I hadn't thought of that! I did notice, however,
that the cost of shipping the book to Germany went down
by a third. :-)

Cheers,

David

--

David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Problem array subscripting
Posted by [m218003](#) on Fri, 20 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

In article <MPG.12261704372f4d149898ab@news.frii.com>,
davidf@dfanning.com (David Fanning) writes:
> Henry Chapman (chapman9@llnl.gov) writes:
>
> . So I want you to know that I went through the
> whole damn book and eliminated all 5,438 instances
> (or whatever it was) of them . I'm calling this the
> 2nd (Less Pedantic) Edition. :-)

Wow! That's 48942 characters! Enough space for a whole new chapter I should
think ;-)

Martin
>

—
