Subject: Re: Padding arrays - vector subscripts not working
Posted by James[2] on Fri, 02 Jul 2010 18:51:27 GMT
View Forum Message <> Reply to Message

Well, I see that this was already asked by hradliv in 2007 and
answered by JD Smith, among others. Still, it seems strange that the
natural method I tried to use doesn't work.

Subject: Re: Padding arrays - vector subscripts not working
Posted by penteado on Fri, 02 Jul 2010 19:29:00 GMT
View Forum Message <> Reply to Message

On Jul 2, 3:51 pm, James <donje...@gmail.com> wrote:
> Well, I see that this was already asked by hradliv in 2007 and
> answered by JD Smith, among others. Still, it seems strange that the
> natural method I tried to use doesn't work.

It is not strange.

X[1,1,1] = Y

and

X[diff] = Y with diff=[1,1,1]

are different things. X[diff] is the same as X[[1,1,1]].

To see the difference, consider this case (where I defined X so that
its elements are not all equal)

IDL> X = bindgen(5,5,5)

```
IDL> print,X[1,1,1]
  31
IDL> print,X[31]
  31
```

Both of these print lines are accessing the same (one) element of X,
the one at the (3D) indexes 1,1,1, which, as it happens, is the 32nd
element of the array:

```
IDL> print,array_indices(X,31)
       1       1       1
```

Now,

IDL> diff=[1,1,1]

```
IDL> print,X[diff]
  1   1   1
```

X[diff], which is the same as X[[1,1,1]], is a 3-element array, where each element has the element of X with the (1D) index 1, that is, this is 3 copies of the second element of X:

```
IDL> print,array_indices(X,1)
         1           0           0
IDL> print,X[1,0,0]
  1
```

X[diff] is the same as X[1,0,0], repeated 3 times.

Now, looking at your assignments:

X[1,1,1]=Y

is assigning 12 elements of X (Y has 12 elements), sequentially, starting at X[1,1,1]. Which is working only because X has 125 elements, and this assignment is using 12 elements, starting at the 32nd, thus it fits inside X (it is assigning from to the 32nd to the 43rd element of X the 12 values in Y)

X[diff], since it is X[[1,1,1]] is attempting to assign the 12 values of Y into 3 values (since X[[1,1,1]] is a 3-element array), thus it fails. Only if you tried to assign to X[diff] a 3-element array (or a scalar, which would be repeated), would this have worked.

If this is seeming too abstract, redo these examples with small 2D arrays (which print nicely), and look at the whole X array, and how it changes after assignments.

Subject: Re: Padding arrays - vector subscripts not working
Posted by penteado on Fri, 02 Jul 2010 20:01:42 GMT

View Forum Message <> Reply to Message

On Jul 2, 3:45 pm, James <donje...@gmail.com> wrote:
> By the way, I would like my final program to work on arrays with any
> number of dimensions, so I'd rather avoid a kludge like X[diff[0],
> diff[1], diff[2]] = Y.

As I wrote above, X[diff[0],diff[1],diff[2]] is not the same as X[diff]. What you seem to want is the equivalent of X[diff[0], diff[1], diff[2]] = Y, but working for any number of dimensions. That is, assign the elements of Y to a contiguous piece of X that starts at some location you calculate with the N-dimensional indexes. So it is

just a matter of converting from those N indexes the a 1D index of
that element:

IDL> strides=[1L,product(size(x,/dimensions),/integer,/cumulative)]
IDL> start_index=total(strides*diff,/integer)

Then you can do X[start_index]=Y

Subject: Re: Padding arrays - vector subscripts not working
Posted by James[2] on Sat, 03 Jul 2010 20:52:00 GMT
View Forum Message <> Reply to Message

On Jul 2, 1:01 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:
> On Jul 2, 3:45 pm, James <donje...@gmail.com> wrote:
>
>> By the way, I would like my final program to work on arrays with any
>> number of dimensions, so I'd rather avoid a kludge like X[diff[0],
>> diff[1], diff[2]] = Y.
>
> As I wrote above, X[diff[0],diff[1],diff[2]] is not the same as
> X[diff]. What you seem to want is the equivalent of X[diff[0],
> diff[1], diff[2]] = Y, but working for any number of dimensions. That
> is, assign the elements of Y to a contiguous piece of X that starts at
> some location you calculate with the N-dimensional indexes. So it is
> just a matter of converting from those N indexes the a 1D index of
> that element:
>
> IDL> strides=[1L,product(size(x,/dimensions),/integer,/cumulative)]
> IDL> start_index=total(strides*diff,/integer)
>
> Then you can do X[start_index]=Y

Paulo, your explanation makes perfect sense. I guess I was hoping
there was a more elegant way than calculating every single 1d index
that will be used, but your method looks good. Thank you!

Subject: Re: Padding arrays - vector subscripts not working
Posted by penteado on Sat, 03 Jul 2010 21:01:47 GMT
View Forum Message <> Reply to Message

On Jul 3, 5:52 pm, James <donje...@gmail.com> wrote:
> Paulo, your explanation makes perfect sense. I guess I was hoping
> there was a more elegant way than calculating every single 1d index
> that will be used, but your method looks good. Thank you!

You do not need to calculate every 1D index. Only the first index is needed. The single assignment

X[start_index]=Y

will assign all the 12 elements of Y to 12 elements of X (from start_index to start_index+11).

Subject: Re: Padding arrays - vector subscripts not working
Posted by James[2] on Tue, 06 Jul 2010 18:38:14 GMT
View Forum Message <> Reply to Message

OK, I tried using your method, but it seemed to only take the first dimension of Y. Here's the function I used:

```
function pad, array, dims

  ;error checking, etc. removed for more compact newsgroup post

  type = size(array, /type)
  out = make_array(dims, type=type)

  ;calculate where to put the original array in the new
  sz = size(array, /dimensions)
  diff = (dims - sz)/2

  strides = [1L, product(dims, /integer, /cumulative)]
  start = total(strides*diff, /integer)

  out[start] = array
  return, out
end
```

And here is an example of its output:

```
IDL> test = replicate(4,3,3)
IDL> padtest = pad(test, [5,6])
IDL> print, padtest
      0       0       0       0       0
      0       4       4       4       0
      0       0       0       0       0
      0       0       0       0       0
      0       0       0       0       0
      0       0       0       0       0
```
