Subject: adding a 4th dimension to 3D array during concatenation
Posted by wdolan on Tue, 15 Mar 2016 22:37:12 GMT
View Forum Message <> Reply to Message

So outside a loop I start out with an empty array (ex. array=[ ]). Then each time through the loop, I make an array with 3 dimensions (for example, array x has dimensions[91, 41, 33]), and then concatinate it to the previous array. (ex. array=[array, x]).

Lets say we run through the loop 16 times. What I'd like as a result is something that has dimensions like this [16, 91, 41, 33].

I'm not sure how to do this... I've looked at IDL coyote's concatenation tutorial, and still am having trouble.

I'm pretty new to coding period, so this is a challenge. Any ideas?

Subject: Re: adding a 4th dimension to 3D array during concatenation
Posted by Paul Van Delst[1] on Wed, 16 Mar 2016 15:00:28 GMT
View Forum Message <> Reply to Message

Hello,

On 03/15/16 18:37, Wayana Dolan wrote:
> So outside a loop I start out with an empty array (ex. array=[ ]).
> Then each time through the loop, I make an array with 3 dimensions
> (for example, array x has dimensions[91, 41, 33]), and then
> concatinate it to the previous array. (ex. array=[array, x]).
>
> Lets say we run through the loop 16 times. What I'd like as a result
> is something that has dimensions like this [16, 91, 41, 33].
>
> I'm not sure how to do this... I've looked at IDL coyote's
> concatenation tutorial, and still am having trouble.
>
> I'm pretty new to coding period, so this is a challenge. Any ideas?
>

It depends on what you want to do with your monster array after concatenation, and will each array be the same shape? (Even if the answer right now is yes, will they always be?)

Concatenation is a slow operation in IDL, and I have always found multi-dimensional concatenation similar to dealing with regular expression - counting all the [[['s and]]]'s to make sure they match up, etc. This is not a fault with IDL, IMO it's just that arrays, really, are not meant to have those sorts of things done to them.

So, why use an array?

Why not, say, a list?

```
IDL> array=list()
IDL> help, array
ARRAY LIST <ID=1 NELEMENTS=0>
IDL> x=findgen(91,43,33)
IDL> array.Add, x
IDL> x=findgen(14,17,36)
IDL> array.Add, x
IDL> help, array
ARRAY LIST <ID=1 NELEMENTS=2>
IDL> help, array[0]
<Expression> FLOAT = Array[91, 43, 33]
IDL> help, array[1]
<Expression> FLOAT = Array[14, 17, 36]
```

Or a hash? Works similarly.

Lists and hashes are data constructs that are designed to be added to
and extended. Arrays, not so much.

Anyhoo...

cheers,

paulv

Subject: Re: adding a 4th dimension to 3D array during concatenation
Posted by Jim Pendleton on Thu, 17 Mar 2016 01:46:25 GMT
View Forum Message <> Reply to Message

On Wednesday, March 16, 2016 at 9:00:35 AM UTC-6, Paul van Delst wrote:
> Hello,
>
> On 03/15/16 18:37, Wayana Dolan wrote:
>> So outside a loop I start out with an empty array (ex. array=[ ]).
>> Then each time through the loop, I make an array with 3 dimensions
>> (for example, array x has dimensions[91, 41, 33]), and then
>> concatinate it to the previous array. (ex. array=[array, x]).
>>
>> Lets say we run through the loop 16 times. What I'd like as a result
>> is something that has dimensions like this [16, 91, 41, 33].
>>
>> I'm not sure how to do this... I've looked at IDL coyote's
>> concatenation tutorial, and still am having trouble.

>>
>> I'm pretty new to coding period, so this is a challenge. Any ideas?
>>
>
> It depends on what you want to do with your monster array after
> concatenation, and will each array be the same shape? (Even if the
> answer right now is yes, will they always be?)
>
> Concatenation is a slow operation in IDL, and I have always found
> multi-dimensional concatenation similar to dealing with regular
> expression - counting all the [[['s and]]]'s to make sure they match
> up, etc. This is not a fault with IDL, IMO it's just that arrays,
> really, are not meant to have those sorts of things done to them.
>
> So, why use an array?
>
> Why not, say, a list?
>
> IDL> array=list()
> IDL> help, array
> ARRAY LIST <ID=1 NELEMENTS=0>
> IDL> x=findgen(91,43,33)
> IDL> array.Add, x
> IDL> x=findgen(14,17,36)
> IDL> array.Add, x
> IDL> help, array
> ARRAY LIST <ID=1 NELEMENTS=2>
> IDL> help, array[0]
> <Expression> FLOAT = Array[91, 43, 33]
> IDL> help, array[1]
> <Expression> FLOAT = Array[14, 17, 36]
>
> Or a hash? Works similarly.
>
> Lists and hashes are data constructs that are designed to be added to
> and extended. Arrays, not so much.
>
> Anyhoo...
>
> cheers,
>
> paulv

To add to Paul's excellent suggestion about lists and the perils of array concatenation, I'll add the following. If you actually need an array at the end of your accumulation, you can use the List.ToArray() method. Use the /NO_COPY keyword to get extra points for efficiency.

IDL> a = fltarr(91, 41, 33)

```
IDL> b = a
IDL> c = list()
IDL> c.add, a, /NO_COPY
IDL> c.add, b, /NO_COPY
IDL> d = c.toarray(/NO_COPY)
IDL> help, d
D               FLOAT     = Array[2, 91, 41, 33]
```

Jim P.

Subject: Re: adding a 4th dimension to 3D array during concatenation
Posted by wlandsman on Thu, 17 Mar 2016 03:46:16 GMT
View Forum Message <> Reply to Message

On Tuesday, March 15, 2016 at 6:37:18 PM UTC-4, Wayana Dolan wrote:
You can go ahead with the list suggestions but if you do know how many arrays you want to concatenate, then it is even more efficient to build your array beforehand.

```
outarr = fltarr(91,41,33,16)
for i=0,15 do begin
   ....   Make an array x with dimensions (91,41,33)
   outarr[0,0,0,i] = x
endfor

x = transpose(x, [3,0,1,2])
```

Now for efficiency reasons it is better to make the last dimension 16 rather than the first. But if you really want the first dimension to be the "counting" dimension then you can use TRANSPOSE() as above to rearrange things.

> So outside a loop I start out with an empty array (ex. array=[ ]). Then each time through the loop, I make an array with 3 dimensions (for example, array x has dimensions[91, 41, 33]), and then concatinate it to the previous array. (ex. array=[array, x]).
>
> Lets say we run through the loop 16 times. What I'd like as a result is something that has dimensions like this [16, 91, 41, 33].
>
> I'm not sure how to do this... I've looked at IDL coyote's concatenation tutorial, and still am having trouble.
>
> I'm pretty new to coding period, so this is a challenge. Any ideas?

Subject: Re: adding a 4th dimension to 3D array during concatenation
Posted by Guilherme Gualda on Thu, 17 Mar 2016 05:52:14 GMT

Hi,

It may not be the most efficient, but it is easy to do what you want if you use reform. In your example, if each array x you create has dimensions [sx, sy, sz], then you can use:

array = [array, reform(x, 1, sx, sy, sz)]

Hope this helps!

Best,

Guil

On Tuesday, March 15, 2016 at 5:37:18 PM UTC-5, Wayana Dolan wrote:
> So outside a loop I start out with an empty array (ex. array=[ ]). Then each time through the loop, I make an array with 3 dimensions (for example, array x has dimensions[91, 41, 33]), and then concatinate it to the previous array. (ex. array=[array, x]).
>
> Lets say we run through the loop 16 times. What I'd like as a result is something that has dimensions like this [16, 91, 41, 33].
>
> I'm not sure how to do this... I've looked at IDL coyote's concatenation tutorial, and still am having trouble.
>
> I'm pretty new to coding period, so this is a challenge. Any ideas?