

---

Subject: Re: zero-padding an array of arbitrary dimensionality (replacing execute in vm)

Posted by [Allan Whiteford](#) on Thu, 19 Jul 2007 16:37:26 GMT

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

---

Vince,

```
data=fltarr(43,45,23,12) ; <-- arbitrary
data[1,1,1,1]=1000. ; <-- so we know we get the right answer
```

```
tmp=size(data)
tmp=tmp[1:tmp[0]]
idx=1
for i=0,n_elements(tmp)-2 do idx=idx+product(tmp[0:i])
```

```
print,data[idx] ; We get element [1,1,1,1]
```

Helpful?

Probably doesn't work for 1D arrays.

Thanks,

Allan

hradilv wrote:

```
> I would like to zero-pad an array programmatically without knowing in
> advance what the dimensionality is of the array.
```

```
>
```

```
> For example, in 2D, for data = some fltarr (31,31) I could do
```

```
> dims = size(data,/dimensions)
```

```
> zpad = fltarr(dims[0]+1,dims[1]+1)
```

```
> zpad[1,1] = data
```

```
>
```

```
> For arb. dimensionality I have:
```

```
>
```

```
> dsize = size(data)
```

```
> ndim = dsize[0]
```

```
> dim = dsize[1:ndim]
```

```
> dtmp = make_array(dim+2,value=0,type=dsize[ndim+1])
```

```
>
```

```
> cmd = 'dtmp['
```

```
> for n=0L, ndim-1 do cmd = cmd+'1,'
```

```
> cmd = strmid(cmd,0,strlen(cmd)-1)+']=data'
```

```
> result = execute(cmd)
```

```
>
```

```
> But this won't work in the vm. So I need to somehow figure out the
```

```
> position of the [1,1,1,...] index for an arbitrary dimensionality.
```

>  
> Clear enough? TIA!  
> Vince  
>

---

Subject: Re: zero-padding an array of arbitrary dimensionality (replacing execute in vm)

Posted by [Michael Galloy](#) on Thu, 19 Jul 2007 22:11:03 GMT

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

---

On Jul 19, 10:37 am, Allan Whiteford  
<allan.rem...@phys.remove.strath.ac.remove.uk> wrote:  
> data=fltarr(43,45,23,12) ; <-- arbitrary  
> data[1,1,1,1]=1000. ; <-- so we know we get the right answer  
>  
> tmp=size(data)  
> tmp=tmp[1:tmp[0]]

This is easier:

```
tmp = size(data, /dimensions)  
  
> idx=1  
> for i=0,n_elements(tmp)-2 do idx=idx+product(tmp[0:i])  
>  
> print,data[idx] ; We get element [1,1,1,1]  
>  
> Helpful?
```

That calculates the index, but I don't see how to use that. A straight-forward

```
data[idx] = origArray
```

doesn't seem to work. Am I missing some other way to copy the original array into the padded array?

It's ugly, but I think this should do the trick:

```
function zeropad, arr  
  compile_opt strictarr  
  on_error, 2  
  
  ndims = size(arr, /n_dimensions)  
  if (ndims lt 1) then message, 'must be an array'  
  
  padded = make_array(dimension=size(arr, /dimensions) + 2L, $
```

```
        type=size(a, /type))
case ndims of
1 : padded[1] = arr
2 : padded[1, 1] = arr
3 : padded[1, 1, 1] = arr
4 : padded[1, 1, 1, 1] = arr
5 : padded[1, 1, 1, 1, 1] = arr
6 : padded[1, 1, 1, 1, 1, 1] = arr
7 : padded[1, 1, 1, 1, 1, 1, 1] = arr
8 : padded[1, 1, 1, 1, 1, 1, 1, 1] = arr
endcase

return, padded
end
```

Mike

--

[www.michaelgalloy.com](http://www.michaelgalloy.com)

---

---

Subject: Re: zero-padding an array of arbitrary dimensionality (replacing execute in vm)

Posted by [Vince Hradil](#) on Thu, 19 Jul 2007 22:11:45 GMT

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

---

On Jul 19, 11:37 am, Allan Whiteford

<[allan.rem...@phys.remove.strath.ac.remove.uk](mailto:allan.rem...@phys.remove.strath.ac.remove.uk)> wrote:

```
> Vince,
>
> data=fltarr(43,45,23,12) ; <-- arbitrary
> data[1,1,1,1]=1000.    ; <-- so we know we get the right answer
>
> tmp=size(data)
> tmp=tmp[1:tmp[0]]
> idx=1
> for i=0,n_elements(tmp)-2 do idx=idx+product(tmp[0:i])
>
> print,data[idx]      ; We get element [1,1,1,1]
>
> Helpful?
>
> Probably doesn't work for 1D arrays.
>
> Thanks,
>
> Allan
>
> hradilv wrote:
```

```
>> I would like to zero-pad an array programmatically without knowing in
>> advance what the dimensionality is of the array.
>
>> For example, in 2D, for data = some fltarr (31,31) I could do
>> dims = size(data,/dimensions)
>> zpad = fltarr(dims[0]+1,dims[1]+1)
>> zpad[1,1] = data
>
>> For arb. dimensionality I have:
>
>> dsize = size(data)
>> ndim = dsize[0]
>> dim = dsize[1:ndim]
>> dtmp = make_array(dim+2,value=0,type=dsize[ndim+1])
>
>> cmd = 'dtmp['
>> for n=0L, ndim-1 do cmd = cmd+'1,'
>> cmd = strmid(cmd,0,strlen(cmd)-1)+']=data'
>> result = execute(cmd)
>
>> But this won't work in the vm. So I need to somehow figure out the
>> position of the [1,1,1,...] index for an arbitrary dimensionality.
>
>> Clear enough? TIA!
>> Vince
```

Yes. I was just having some trouble wrapping my head around it.  
Thanks!

---