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
