Subject: Re: How to create a 2D mask that automatically half's an irregularly shaped 2D array from top to bottom?
Posted by Jeremy Bailin on Sun, 04 Dec 2011 06:15:47 GMT
View Forum Message <> Reply to Message

> When you say the output is a little strange, what exactly does the input
> mask do? My solution assumes that each row has only one string of 1s in
> it - if it has more, then it won't break up each individual string, but
> will just cut out the last half of them.... a correct solution would
> require some fancy footwork with label_region.

Here's a solution that should work for any arbitrary mask. I'm sure it's not as fast as the original, but it should be pretty efficient... there is a for loop, but it is at most a loop through the columns rather than the rows (and depending on the mask, could be much shorter).

For reference, it's inspired by part of JD's double-histogram solution to the chunk indexing problem, although it doesn't actually use a double histogram.

```
masksize = size(inmask, /dimen)

; expand the mask with a column of 0s on each side
; so that shift will not cycle 1s around
mask_expand = bytarr(masksize[0]+2, masksize[1])
mask_expand[1:masksize[0],*] = inmask

; find the left and right ends of each string of 1s
; by checking to see if it changes when shifted left
; or right by one column
leftends = where(mask_expand and not shift(mask_expand,1), nstring)
rightends = where(mask_expand and not shift(mask_expand,-1))
; find the midpoints by averaging
midpts = rebin([[leftends],[rightends]], nstring)
; and how long is each string?
lengths = midpts - leftends + 1

; clear mask_expand so we can fill it up again
mask_expand[*] = 0

; histogram the lengths so we can loop through them and use
; reverse indices to know which string has which length
hlen = histogram(lengths, min=1, reverse_indices=lenri)

; single indices are easy
if hlen[0] gt 0 then mask_expand[leftends[lenri[lenri[0]:lenri[1]-1]]]=1
```

```
; loop through length counts
for j=1, n_elements(hlen)-1 do if hlen[j] gt 0 then begin
   ; which ones have this length?
   vec_inds = lenri[lenri[j]:lenri[j+1]-1]
   ; make a 2D array of numbers starting at each left end of this length
   ; and incrementing along the second dimension. This array contains
   ; all of the indices into mask_expand that we want to set
   vec_inds = rebin(leftends[vec_inds], hlen[j], j+1, /sample) + $
     rebin(lindgen(1,j+1), hlen[j], j+1, /sample)
   ; and set them
   mask_expand[vec_inds] = 1
endif

; finally get rid of those extra columns of 0s we added at the beginning
outmask = temporary(mask_expand[1:masksize[0],*])
```

-Jeremy.
