
Subject: Blanking out regions

Posted by [Victorpoe](#) on Thu, 05 Feb 2004 08:43:05 GMT

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

I need to draw a contour plot of data on irregular 2d grid and blank out regions outside the boundaries of multiconnected non-convex domain A in R2.

; Draw contours

Contour, MyData, X, Y, /IRREGULAR

; Draw boundary

Oplot, BoundaryA

;Blank out exterior

?

Is there a way to fill the complement of A to circumscribed rectangle?

Any advice will be appreciated.

Victor.

Subject: Re: Blanking out regions

Posted by [hjalti](#) on Tue, 10 Feb 2004 13:42:28 GMT

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

Victorpoe@hotmail.com (Victor Podsechin) wrote in message news:<20b63ed6.0402050043.3aa30c29@posting.google.com>...

> I need to draw a contour plot of data on irregular 2d grid and blank
> out regions outside the boundaries of multiconnected non-convex domain
> A in R2.

> ; Draw contours

> Contour, MyData, X, Y, /IRREGULAR

> ; Draw boundary

> Oplot, BoundaryA

> ;Blank out exterior

> ?

> Is there a way to fill the complement of A to circumscribed rectangle?

> Any advice will be appreciated.

> Victor.

If I remember correctly it was David Fanning who discussed some time ago how this can be done in case you have a masking array, in that instance terrain elevation where the oceans with value zero were to be blanked out. I have adapted Fanning's example for my own work and added a method for creating a masking array from a polygon. I include below the part of my program that does this job, and some comments within.

; You read your data into an array, then do

thisDevice = !D.Name ; Store the present device in a variable

xsize=500 & ysize=600 ; Define the dimensions of your plotting device

```

Set_Plot, 'Z' ; Set the Z-buffer as the current device
Device, Set_Resolution=[xsize, ysize]

; Here I read the masking polygon, a list of (x,y) coordinates
openr, lun, '../kortagr/boundary.xy', /get_lun
readf, lun, dummy
readf, lun, npoints
boundary=fltarr(2, npoints)
readf, lun, boundary
free_lun, lun
; Now the vertices of the polygon are stored in the array boundary

plot, boundary[0,*], boundary[1,*], /nodata, xstyle=1, ystyle=1,
/isotropic
; Plot the boundary with /nodata keyword
map=tvrd() ; Now read the image (axes) from the Z-buffer for later
use.
mask1=intarr(xsize,ysize)
id_mask1=where(map ne 0)
mask1[id_mask1]=1 ; set mask1 to one where the axes are
; Do the contour plot
contour, z, x, y, /irregular, nlevels=nlevels, /overplot,
c_labels=replicate(1, nlevels);, c_colors=c_colors, /fill, /overplot,
max_value=100, min_value=-100.

map=tvrd() ; read again the image from the Z-buffer

Set_Plot, thisDevice
window, xsize=xsize, ysize=ysize ; make a visible window

; Convert the boundary coordinates to device-coordinates
res=convert_coord(boundary[0,*], boundary[1,*], /data, /to_device)
res=fix(res[0:1,*]) ; Change to integer type - to be used as array
indices(actually not necessary).

id_mask=polyfillv(res[0,*], res[1,*], xsize, ysize); create an array
of all the array indices within the polygon defined by 'res'

mask=intarr(xsize, ysize)
mask[id_mask]=1
mask=mask+mask1 ; Region inside the polygon + the axes are to be
plotted

map=map*mask; Blanks map where mask is zero
tv, map ; puts map to the plotting window
END

```

This is it, hope it was helpful.

Regards, Hjalti

Subject: Re: Blanking out regions

Posted by [Victorpoe](#) on Mon, 23 Feb 2004 13:41:20 GMT

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

hjalti@vatnaskil.is (Hjalti Sig) wrote in message

news:<e1330fff.0402100542.4df549a3@posting.google.com>...

> If I remember correctly it was David Fanning who discussed some time
> ago how this can be done in case you have a masking array, in that
> instance terrain elevation where the oceans with value zero were to be
> blanked out. I have adapted Fanning's example for my own work and
> added a method for creating a masking array from a polygon. I include
> below the part of my program that does this job, and some comments
> within.

> -----

> ; You read your data into an array, then do

>

> thisDevice = !D.Name ; Store the present device in a variable

> xsize=500 & ysize=600 ; Define the dimensions of your plotting device

> Set_Plot, 'Z' ; Set the Z-buffer as the current device

> Device, Set_Resolution=[xsize, ysize]

>

> ; Here I read the masking polygon, a list of (x,y) coordinates

> openr, lun, '../kortagr/boundary.xy', /get_lun

> readf, lun, dummy

> readf, lun, npoints

> boundary=fltarr(2, npoints)

> readf, lun, boundary

> free_lun, lun

> ; Now the vertices of the polygon are stored in the array boundary

>

> plot, boundary[0,*], boundary[1,*], /nodata, xstyle=1, ystyle=1,

> /isotropic

> ; Plot the boundary with /nodata keyword

> map=tvrd() ; Now read the image (axes) from the Z-buffer for later

> use.

> mask1=intarr(xsize,ysize)

> id_mask1=where(map ne 0)

> mask1[id_mask1]=1 ; set mask1 to one where the axes are

> ; Do the contour plot

> contour, z, x, y, /irregular, nlevels=nlevels, /overplot,

> c_labels=replicate(1, nlevels);, c_colors=c_colors, /fill, /overplot,

> max_value=100, min_value=-100.

>

> map=tvrd() ; read again the image from the Z-buffer

```

>
> Set_Plot, thisDevice
> window, xsize=xsize, ysize=ysize ; make a visible window
>
> ; Convert the boundary coordinates to device-coordinates
> res=convert_coord(boundary[0,*], boundary[1,*], /data, /to_device)
> res=fix(res[0:1,*]) ; Change to integer type - to be used as array
> indices(actually not necessary).
>
> id_mask=polyfillv(res[0,*], res[1,*], xsize, ysize); create an array
> of all the array indices within the polygon defined by 'res'
>
> mask=intarr(xsize, ysize)
> mask[id_mask]=1
> mask=mask+mask1 ; Region inside the polygon + the axes are to be
> plotted
>
> map=map*mask; Blanks map where mask is zero
> tv, map ; puts map to the plotting window
> END
>
>
> This is it, hope it was helpful.
> Regards, Hjalti

```

Thanks you, Hjalti,
 your method works fine for me also.
 I found another way to blank regions using polyfill procedure.
 I assume that a boundary of domain A in R2 is closed and
 counter-clockwise ordered.

Then it can be divided into four segments:

```

(xmin,*) - (*, ymin),
(*,ymin) - (xmax,*),
(xmax,*) - (*, ymax),
(*,ymax) - (xmin,*).

```

These segments can be closed by adding the points - corners of circumscribed rectangle:

```

(xmin, ymin)
(xmax, xmin)
(xmax, ymax)
(xmin,ymax).

```

By applying polyfill procedure to these closed segments the regions outside the boundary will be blanked.

Victor.

Subject: Re: Blanking out regions

Posted by [hjalti](#) on Tue, 24 Feb 2004 13:40:37 GMT

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

Victorpoe@hotmail.com (Victor Podsechin) wrote in message
news:<20b63ed6.0402230541.48b5d75e@posting.google.com>...

> [hjalti@vatnaskil.is](#) (Hjalti Sig) wrote in message

news:<e1330fff.0402100542.4df549a3@posting.google.com>...

>> If I remember correctly it was David Fanning who discussed some time
>> ago how this can be done in case you have a masking array, in that
>> instance terrain elevation where the oceans with value zero were to be
>> blanked out. I have adapted Fanning's example for my own work and
>> added a method for creating a masking array from a polygon. I include
>> below the part of my program that does this job, and some comments
>> within.

>> -----

>> ; You read your data into an array, then do

>>

>> thisDevice = !D.Name ; Store the present device in a variable

>> xsize=500 & ysize=600 ; Define the dimensions of your plotting device

>> Set_Plot, 'Z' ; Set the Z-buffer as the current device

>> Device, Set_Resolution=[xsize, ysize]

>>

>> ; Here I read the masking polygon, a list of (x,y) coordinates

>> openr, lun, '../kortagr/boundary.xy', /get_lun

>> readf, lun, dummy

>> readf, lun, npoints

>> boundary=fltarr(2, npoints)

>> readf, lun, boundary

>> free_lun, lun

>> ; Now the vertices of the polygon are stored in the array boundary

>>

>> plot, boundary[0,*], boundary[1,*], /nodata, xstyle=1, ystyle=1,

>> /isotropic

>> ; Plot the boundary with /nodata keyword

>> map=tvrd() ; Now read the image (axes) from the Z-buffer for later

>> use.

>> mask1=intarr(xsize,ysize)

>> id_mask1=where(map ne 0)

>> mask1[id_mask1]=1 ; set mask1 to one where the axes are

>> ; Do the contour plot

>> contour, z, x, y, /irregular, nlevels=nlevels, /overplot,

>> c_labels=replicate(1, nlevels);, c_colors=c_colors, /fill, /overplot,

>> max_value=100, min_value=-100.

>>

>> map=tvrd() ; read again the image from the Z-buffer

>>

>> Set_Plot, thisDevice

>> window, xsize=xsize, ysize=ysize ; make a visible window

```

>>
>> ; Convert the boundary coordinates to device-coordinates
>> res=convert_coord(boundary[0,*], boundary[1,*], /data, /to_device)
>> res=fix(res[0:1,*]) ; Change to integer type - to be used as array
>> indices(actually not necessary).
>>
>> id_mask=polyfillv(res[0,*], res[1,*], xsize, ysize); create an array
>> of all the array indices within the polygon defined by 'res'
>>
>> mask=intarr(xsize, ysize)
>> mask[id_mask]=1
>> mask=mask+mask1 ; Region inside the polygon + the axes are to be
>> plotted
>>
>> map=map*mask; Blanks map where mask is zero
>> tv, map ; puts map to the plotting window
>> END
>>
>>
>> This is it, hope it was helpful.
>> Regards, Hjalti
>
>
> Thanks you, Hjalti,
> your method works fine for me also.
> I found another way to blank regions using polyfill procedure.
> I assume that a boundary of domain A in R2 is closed and
> counter-clockwise ordered.
> Then it can be divided into four segments:
> (xmin,*) - (*, ymin),
> (*,ymin) - (xmax,*),
> (xmax,*) - (*, ymax),
> (*,ymax) - (xmin,*).
> These segments can be closed by adding the points - corners of
> circumscribed rectangle:
> (xmin, ymin)
> (xmax, xmin)
> (xmax, ymax)
> (xmin,ymax).
> By applying polyfill procedure to these closed segments the regions
> outside the boundary will be blanked.
> Victor.

```

OK, I may try this. But I was thinking - I guess the simplest way to blank out regions is making a regular grid of your data, then converting your (x,y) values to index numbers and same to the blanking polygon, then finding the index numbers to be blanked using polyfillv, assigning NaN to the corresponding z-values, and then doing countour

plotting. By this you can avoid manipulating images in the z-buffer. Generally your data array has much smaller dimensions than the plotting device, so this may result in a less refined blanking, but I have not tried this out.

Hjalti
