Subject: Re: Interpolation of missing data
Posted by Martin Schultz on Tue, 20 Jan 1998 08:00:00 GMT
View Forum Message <> Reply to Message

R. Kyle Justice wrote:
>
> I have a 2-D array with missing data. Is there an easy way to
> interpolate the missing values?
>
> I would like to replace a missing value with the average of
> its neighbors.
>
> Kyle J.

This may not be exactly what you want, but you could try to
"re-sample" your data as an array and use the TRI_SURF (or
MIN_CURVE_SURF) function. I have a piece of code that does something
like

```
; create x and y vectors that match with zz array
goodx = findgen(nx+3)/(nx+2)*(xrange(1)-xrange(0))+xrange(0)
goody = findgen(ny+3)/(ny+2)*(yrange(1)-yrange(0))+yrange(0)

; little trick to get the indices of valid zz's in goodx and goody
xind = (ind mod (nx+2))
yind = (ind/(nx+2))      ; integer division !!

newx = reform(goodx(xind),n_elements(ind))
newy = reform(goody(yind),n_elements(ind))


zzz = TRI_SURF(newz,newx,newy,gs=[dx,dy],  $
          bounds=[xrange(0),yrange(0),xrange(1),yrange(1)] )
```

(for regular readers: this turned out to be the best solution to my
contour problem that I described earlier - but I must warn of the use
of MIN_CURVE_SURF: it takes *forever* [i.e. I did not want to wait more
than 3 minutes for a data set of ~1000 points and interrupted])

Regards,
Martin

PS: another solution (which would involve a loop [nasty word ;-)])
would be to compute the averages of surrounding grid boxes like

```
ind = where(data eq MISSING)   ; supply your code for missing data
```

```
  if (ind(0) ge 0) then begin
    for i=0,n_elements(ind)-1 do begin
       x = (i mod (NX+2))  ; get indices in data array
       y = (i/(NX+2))      ; integer division !!
       ; create index array for neighbouring points
       xind = [ x-1>(-1), x, x+1<NX, x ]
       yind = [ y, y-1>(-1), y, y+1<NY ]
       ; find out valid neighbours
       ok = where(xind ne MISSING and yind ne MISSING)
       if (ok(0) ge 0) then $
          data(x,y) = total(data(xind(ok),yind(ok))/ $
                    float(n_elements(ok))
    endfor
  endif
```

This would of course only work if at least one neighbour is a valid data point. In case you are not familiar with the < and > operators: they are great to limit value ranges, I just recently understood them and loved them immediately! Please NOTE: I did not test this code, but it should give you something to start with at least.


---

Dr. Martin Schultz
Department for Earth&Planetary Sciences, Harvard University
186 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318
fax  : (617)-495-4551

e-mail: mgs@io.harvard.edu
IDL-homepage: http://www-as.harvard.edu/people/staff/mgs/idl/


---