Subject: Faster alternative to MIN_CURVE_SURF in IDL contouring
Posted by info on Mon, 11 Oct 1993 22:13:34 GMT
View Forum Message <> Reply to Message

In response to complaints about the speed of MIN_CURVE_SURF when used with a large number of points, we have written a new User's Library function, TRI_SURF. TRI_SURF is a much more efficient method of interpolating a smooth surface over both regularly gridded and scattered points for larger datasets. This procedure is included at the end of this posting.

It uses the built-in routines TRIANGULATE and TRIGRID(/QUINTIC) to fit C1 smooth quintic polynomials to the surface. For example, TRI_SURF will interpolate a 25x25 matrix to 100x100 in about 4 seconds on a Sun IPX.

_____Cut Here_____

```
; $Id: tri_surf.pro,v 1.1 1993/10/08 15:37:53 dave Exp $

FUNCTION TRI_SURF, z, x, y, REGULAR = regular, XGRID=xgrid, $
 XVALUES = xvalues, YGRID = ygrid, YVALUES = yvalues, $
 GS = gs, BOUNDS = bounds, NX = nx0, NY = ny0, $
 MISSING = missing, EXTRAPOLATE = extrapolate
;
; Copyright (c) 1993, Research Systems, Inc.  All rights reserved.
; Unauthorized reproduction prohibited.
;+
; NAME:
; TRI_SURF
;
; PURPOSE:
; This function Interpolates a regularly or irregularly gridded
; set of points with a C1 smooth quintic surface.
;
; CATEGORY:
; Mathematical functions, Interpolation, Surface Fitting
;
; CALLING SEQUENCE:
; Result = TRI_SURF(Z [, X, Y])
;
; INPUTS:
; X, Y, Z:  arrays containing the X, Y, and Z coordinates of the
;    data points on the surface. Points need not be
;    regularly gridded. For regularly gridded input data,
```

```
;    X and Y are not used: the grid spacing is specified
;    via the XGRID and YGRID (or XVALUES and YVALUES)
;    keywords, and Z must be a two dimensional array.
;    For irregular grids, all three parameters must be
;    present and have the same number of elements.
;
; KEYWORD PARAMETERS:
;  Input grid description:
; REGULAR:  if set, the Z parameter is a two dimensional array
;    of dimensions (N,M), containing measurements over a
;    regular grid. If any of XGRID, YGRID, XVALUES, YVALUES
;    are specified, REGULAR is implied. REGULAR is also
;    implied if there is only one parameter, Z. If REGULAR is
;    set, and no grid (_VALUE or _GRID) specifications are
;    present, the respective grid is set to (0, 1, 2, ...).
; XGRID:    contains a two element array, [xstart, xspacing],
;    defining the input grid in the X direction. Do not
;    specify both XGRID and XVALUES.
; XVALUES:  if present, XVALUES(i) contains the X location
;    of Z(i,j). XVALUES must be dimensioned with N elements.
; YGRID:    contains a two element array, [ystart, yspacing],
;    defining the input grid in the Y direction. Do not
;    specify both YGRID and YVALUES.
; YVALUES:  if present, YVALUES(i) contains the Y location
;    of Z(i,j). YVALUES must be dimensioned with N elements.
;
;  Output grid description:
; GS:   If present, GS must be a two-element vector [XS, YS],
;    where XS is the horizontal spacing between grid points
;    and YS is the vertical spacing. The default is based on
;    the extents of X and Y. If the grid starts at X value
;    Xmin and ends at Xmax, then the default horizontal
;    spacing is (Xmax - Xmin)/(NX-1). YS is computed in the
;    same way. The default grid size, if neither NX or NY
;    are specified, is 26 by 26.
; BOUNDS:   If present, BOUNDS must be a four element array containing
;    the grid limits in X and Y of the output grid:
;    [Xmin, Ymin, Xmax, Ymax]. If not specified, the grid
;    limits are set to the extent of X and Y.
; NX:       The output grid size in the X direction. NX need not
;      be specified if the size can be inferred from GS and
;    BOUNDS. The default value is 26.
; NY:       The output grid size in the Y direction. See NX.
;
;  Others:
;       EXTRAPOLATE: If set, extrapolate the surface to points outside
;  the convex hull of input points.  Has no effect if
;  input points are regularly gridded.
```

```
; MISSING: If set, points outside the convex hull of input points
;  are set to this value.  Default is 0.  Has no effect
;  if input points are regularly gridded.
;
; OUTPUTS:
; This function returns a two-dimensional floating point array
;  containing the interpolated surface, sampled at the grid points.
;
; RESTRICTIONS:
; The output grid must enclose convex hull of the input points.
; PROCEDURE:
; This routine is similar to MIN_CURVE_SURF but the surface
; fitted is a smooth surface, not a minimum curvature surface.  This
; routine has the advantage of being much more efficient
; for larger numbers of points.
;
; The built-in IDL routines TRIANGULATE and TRIGRID(/QUINTIC) are
; used.
;
; EXAMPLES:
; Example 1: Irregularly gridded cases
; Make a random set of points that lie on a gaussian:
;   n = 15    ;# random points
;   x = RANDOMU(seed, n)
;   y = RANDOMU(seed, n)
;   z = exp(-2 * ((x-.5)^2 + (y-.5)^2))   ;The gaussian
;
;  get a 26 by 26 grid over the rectangle bounding x and y:
;   r = TRI_SURF(z, x, y) ;Get the surface.
;
;  Or: get a surface over the unit square, with spacing of 0.05:
;   r = TRI_SURF(z, x, y, GS=[0.05, 0.05], BOUNDS=[0,0,1,1])
;
;  Or: get a 10 by 10 surface over the rectangle bounding x and y:
;   r = TRI_SURF(z, x, y, NX=10, NY=10)
;
; Example 2: Regularly gridded cases
; Make some random data
;   z = randomu(seed, 5, 6)
;
; interpolate to a 26 x 26 grid:
;   CONTOUR, TRI_SURF(z, /REGULAR)
;
; MODIFICATION HISTORY:
; DMS, RSI, October, 1993.  Written.
;-
```

```
ON_ERROR, 2
s = size(z)  ;Assume 2D
nx = s(1)
ny = s(2)

reg = keyword_set(regular) or (n_params() eq 1)

if n_elements(xgrid) eq 2 then begin
 x = findgen(nx) * xgrid(1) + xgrid(0)
 reg = 1
endif else if n_elements(xvalues) gt 0 then begin
 if n_elements(xvalues) ne nx then $
  message,'Xvalues must have '+string(nx)+' elements.'
 x = xvalues
 reg = 1
endif

if n_elements(ygrid) eq 2 then begin
 y = findgen(ny) * ygrid(1) + ygrid(0)
 reg = 1
endif else if n_elements(yvalues) gt 0 then begin
 if n_elements(yvalues) ne ny then $
  message,'Yvalues must have '+string(ny)+' elements.'
 y = yvalues
 reg = 1
endif

if reg then begin
 if s(0) ne 2 then message,'Z array must be 2D for regular grids'
 if n_elements(x) ne nx then x = findgen(nx)
 if n_elements(y) ne ny then y = findgen(ny)
 x = x # replicate(1., ny) ;Expand to full arrays.
 y = replicate(1.,nx) # y
 endif

n = n_elements(x)
if n ne n_elements(y) or n ne n_elements(z) then $
 message,'x, y, and z must have same number of elements.'

if n_elements(bounds) lt 4 then begin ;Bounds specified?
 xmin = min(x, max = xmax)
 ymin = min(y, max = ymax)
 bounds = [xmin, ymin, xmax, ymax]
 endif

TRIANGULATE, x, y, tr, b

if n_elements(gs) lt 2 then begin ;GS specified?  No.
```

```
    if n_elements(nx0) le 0 then nx = 26 else nx = nx0 ;Defaults for nx and ny
    if n_elements(ny0) le 0 then ny = 26 else ny = ny0
    gs = [(bounds(2)-bounds(0))/(nx-1.), $
    (bounds(3)-bounds(1))/(ny-1.)]
  endif else begin   ;GS is specified?
    if n_elements(nx0) le 0 then $
 nx = ceil((bounds(2)-bounds(0))/gs(0)) + 1 $
    else nx = nx0
    if n_elements(ny0) le 0 then $
 ny = ceil((bounds(3)-bounds(1))/gs(1)) + 1 $
    else ny = ny0
  endelse


if N_ELEMENTS(missing) le 0 then missing = 0

if KEYWORD_SET(extrapolate) then $
    return, TRIGRID(x,y,z, tr, /QUINTIC, gs, bounds, $
      MISSING=missing, EXTRAPOLATE=b)

return, TRIGRID(x,y,z, tr, /QUINTIC, gs, bounds, MISSING=missing)
end
```