Subject: Re: Streamlining
Posted by Amara Graps on Wed, 21 Mar 2001 18:53:40 GMT
View Forum Message <> Reply to Message

Vapuser wrote:
>
> Anyone have an good IDL streamlining code? Or know where I can get
> some? I'll even entertain the notions of call_external/linkimage code.
>

Dear William,

I honestly don't know if the following attached program
("test_strfield") is useful for you.

The attached code is a bit old (circa 1993), but I just ran it under
5.4, and it runs fine. It's well-documented, however it has not been
tested under different functions and datasets. It's not particularly
elegant IDL programming style either, but it works.

The code is simply the result of my own learning processes for how
stream functions work. (I'm not sure the "proper way" that
meteorologists conventionally define stream functions, maybe
someone can tell me... but in a few weeks.)

It tests out two approaches:

-the vector field approach
-the level sets approach

Please see the files:
http://www.amara.com/ftpstuff/streamlines1.txt
http://www.amara.com/ftpstuff/streamlines2.txt

for how I define these approaches.

Amara

P.S. I'm giving this code, because I have a question for the IDL group,
and I'm under a tight deadline for submitting my dissertation in the
next few weeks, so I thought I'd give something to the group before
I ask the group's assistance....!

--

********** **********

Amara Graps | Max-Planck-Institut fuer Kernphysik
Interplanetary Dust Group | Saupfercheckweg 1
+49-6221-516-543 | 69117 Heidelberg, GERMANY
* http://www.mpi-hd.mpg.de/dustgroup/~graps
********** **********
"Never fight an inanimate object." - P. J. O'Rourke

```
PRO STR_FIELD1,U,V,X,Y, Missing=Missing, Length=length, Dots=dots,  $
      Title=title, position=position, noerase=noerase, color=color,$
 xtitle=xtitle, ytitle=ytitle, xmargin=xmargin, ymargin=ymargin, $
 easy_view=easy_view, noaxis=noaxis
;
;+
; NAME:
; STR_FIELD
;
; PURPOSE:
; Produce a two-dimensional velocity field and streamlines plot.
;
;       See notes at http://www.amara.com/ftpstuff/streamlines1.txt
;       and consider using an integration method better than Euler.
;
; A directed arrow is drawn at each point showing the direction and
; magnitude of the field and streamlines connect the midpoints of the
; arrows. These streamlines are calculated by integrating tangents
;       to the vector field
;
; CATEGORY:
; Plotting, two-dimensional.
;
; CALLING SEQUENCE:
; STR_FIELD, U, V [, X, Y]
;
; INPUTS:
; U: The X component of the two-dimensional field.
;  U must be a two-dimensional array.
;
; V: The Y component of the two dimensional field.  Y must have
;  the same dimensions as X.  The vector at point (i,j) has a
;  magnitude of:
;
;   (U(i,j)^2 + V(i,j)^2)^0.5
;
;  and a direction of:
;
;   ATAN2(V(i,j),U(i,j)).
;
; OPTIONAL INPUT PARAMETERS:
;  X: Optional abcissae values.  X must be a vector with a length
```

```
;  equal to the first dimension of U and V.
;
; Y: Optional ordinate values.  Y must be a vector with a length
;  equal to the first dimension of U and V.
;
; KEYWORD INPUT PARAMETERS:
;      MISSING: Missing data value.  Vectors with a LENGTH greater
;  than MISSING are ignored.
;
; LENGTH: Length factor.  The default of 1.0 makes the longest (U,V)
;  vector the length of a cell.
;
; DOTS: Set this keyword to 1 to place a dot at each missing point.
;  Set this keyword to 0 or omit it to draw nothing for missing
;  points.  Has effect only if MISSING is specified.
;
; TITLE: A string containing the plot title.
;
;     POSITION: A four-element, floating-point vector of normalized
;  coordinates for the rectangular plot window.
;  This vector has the form  [X0, Y0, X1, Y1], where (X0, Y0)
;  is the origin, and (X1, Y1) is the upper-right corner.
;
;      NOERASE: Set this keyword to inhibit erase before plot.
;
; COLOR: The color index used for the plot.
;
; EASY_VIEW: Set this keyword if you don't wish to have all streamlines
;  shown.  For example if you have more than 50 or 100 vectors in your
;  vector plot, the streamline plotting will add considerable time to
;  your plot.
;
; XMARGIN: A two-element vector that specifies the vertical margin between
;  the axis and the screen border in character units.
;
; YMARGIN: A two-element vector that specifies in the horzontal margin
;  between the map and screen border in character units.
;
; NOAXIS: Set this keyword if you wish to have no axes drawn (i.e.
;  if you will be calling this routine on top of another plot.
;
; OUTPUTS:
; None.
;
; COMMON BLOCKS:
; None.
;
; SIDE EFFECTS:
```

```
; Plotting on the selected device is performed.  System
; variables concerning plotting are changed.
;
; RESTRICTIONS:
; None.
;
; PROCEDURE:
; Straightforward.  The system variables !XTITLE, !YTITLE and
; !MTITLE can be set to title the axes.
;
; MODIFICATION HISTORY:
; VEL_FIELD: DMS, RSI, Oct., 1983.
;
; VEL_FIELD: For Sun, DMS, RSI, April, 1989.
;
; VEL_FIELD: Added TITLE, Oct, 1990.
;
; VEL_FIELD: Added POSITION, NOERASE, COLOR, Feb 91, RES.
;
; Created STR_FIELD as modification of VEL_FIELD,
; Nov/Dec 1993, Amara Graps NASA-Ames
;-
;
        on_error,2                      ;Return to caller if an error occurs
        s = size(u)
        t = size(v)
        if s(0) ne 2 then begin
baduv:   message, 'U and V parameters must be 2D and same size.'
                endif
        if total(abs(s(0:2)-t(0:2))) ne 0 then goto,baduv
;
        if n_params(0) lt 3 then x = findgen(s(1)) else $
                if n_elements(x) ne s(1) then begin
badxy:                  message, 'X and Y arrays have incorrect size.'
                        endif
        if n_params(1) lt 4 then y = findgen(s(2)) else $
                if n_elements(y) ne s(2) then goto,badxy
;
        if n_elements(missing) le 0 then missing = 1.0e30
        if n_elements(length) le 0 then length = 1.0

        mag = sqrt(u^2+v^2)             ;magnitude.
 stream = mag

 ;--------  Subscripts of good elements  ----------------------
        nbad = 0                        ;# of missing points
        if n_elements(missing) gt 0 then begin
                good = where(mag lt missing)
```

```
            if keyword_set(dots) then bad = where(mag ge missing, nbad)
        endif else begin
            good = lindgen(n_elements(mag))
        endelse

        mag = mag(good)                ;Discard missing values
        maxmag = max(mag)
        ugood = u(good)
        vgood = v(good)

 ;Define grid for vector and streamline plotting
        xleft = min(x)
        xright = max(x)
        ybottom = min(y)
        ytop = max(y)
 delta_x = abs(xleft - xright)
 delta_y = abs(ybottom - ytop)
 nsteps = 300  ;# of integration steps per streamline
 delta_max = delta_x > delta_y ;choose greater delta
 h = delta_max / 30. ;step-size

 xstream = fltarr(n_elements(ugood), nsteps, 2)
 ystream = fltarr(n_elements(vgood), nsteps, 2)
 xmid = fltarr(n_elements(ugood))
 ymid = fltarr(n_elements(vgood))

        sina = length * (xright-xleft)/s(1)/maxmag*ugood ;sin & cosine components.
        cosa = length * (ytop-ybottom)/s(2)/maxmag*vgood
        ;Note: x0, x1, y0, y1 are boundaries for determining streamline limits

;
;print, 'xtitle = ', xtitle
;print, 'ytitle = ', ytitle

        ;--------------  plot to get axes  ---------------

 ;Set up some of the keywords
        if n_elements(title) le 0 then title = ''
 if n_elements(xtitle) le 0 then xtitle = 'U(x,y)'
 if n_elements(ytitle) le 0 then ytitle = 'V(x,y)'
 if n_elements(xmargin) le 0 then xmargin = !x.margin
 if n_elements(ymargin) le 0 then ymargin = !y.margin
        if n_elements(color) eq 0 then color = !p.color

        if n_elements(position) ne 0 then begin
           plot,[xleft,xright],[ytop,ybottom],/nodata,/xst,/yst,title=t itle, $
            noerase=noerase, color=color, xtitle=xtitle, ytitle = ytitle,$
     xmargin=xmargin, ymargin = ymargin, position = position
```

```
        endif else $
    if keyword_set(noaxis) then begin
     plot,[xleft,xright],[ytop,ybottom],/nodata,$
  noerase=noerase, color=color, xmargin=xmargin, $
  ymargin = ymargin, xstyle=14, ystyle=14
 endif else $
          plot,[xleft,xright],[ytop,ybottom],/nodata,/xst,/yst,$
    title=title, noerase=noerase, color=color, xtitle=xtitle, $
    ytitle=ytitle, xmargin=xmargin, ymargin = ymargin


;
 ;---  place vectors with arrows --
        r = .3                      ;len of arrow head
        angle = 22.5 * !dtor          ;Angle of arrowhead
        st = r * sin(angle)           ;sin 22.5 degs * length of head
        ct = r * cos(angle)
;
        for i=0,n_elements(good)-1 do begin     ;Each point
                x0 = x(good(i) mod s(1))        ;get coords of start & end
                dx = sina(i)
                x1 = x0 + dx
  xmid(i) = (x1 + x0)/2.0
                y0 = y(good(i) / s(1))
                dy = cosa(i)
                y1 = y0 + dy
  ymid(i) = (y0 + y1)/2.0
                plots,[x0,x1,x1-(ct*dx-st*dy),x1,x1-(ct*dx+st*dy)], $
                     [y0,y1,y1-(ct*dy+st*dx),y1,y1-(ct*dy-st*dx)], $
                     color=color, thick = 2.0
                endfor
        if nbad gt 0 then $             ;Dots for missing?
                oplot, x(bad mod s(1)), y(bad / s(1)), psym=3, color=color

 ;----------integrate for streamlines----------

 ;Set up for showing a select set of streamlines
 nskip = 1
        if keyword_set(easy_view) then begin

  nshow_streams = 20.  ;# of streamlines to show for easy viewing
  ntotal = n_elements(good) ;# of potential streamlines (1 per vector)

  ;determine increment value for "for loop" based on nshow_streams & ntotal
  if ntotal gt nshow_streams then nskip = fix(ntotal/nshow_streams)

 end ;if easy_view
```

```
istep = 0  ;step counter

for i = 0, n_elements(good) - 1, nskip do begin   ;each point (field vector)

 for j = 0, 1 do begin ;forward and backward streamlines

  curr_x = xmid(i)
  curr_y = ymid(i)
  xstream(i,istep, j) = curr_x
  ystream(i,istep, j) = curr_y
  out_of_bounds = 0 ;false

  while ((istep le nsteps-2) and (out_of_bounds eq 0)) do begin

   ;Calculate gx, gy (deriv f'ns of x,y)
   case 1 of

   istep eq 0: begin
    ;1st step only
    gx = sina(i)
    gy = cosa(i)
    end ;case istep eq 0

   else: begin

    ;Calc distance of all other vectors from current point
    tempx = xmid - curr_x
    tempy = ymid - curr_y
    distance = sqrt(tempx^2 + tempy^2)

    ;Sort to get nearest ones
    neari = sort(distance)  ;array indices
    nearest = distance(neari) ;sorted by distance from curr
    nearest_gx = sina(neari) ;function in x direction at i
    nearest_gy = cosa(neari) ;function in y direction at i

    ;Choose nearest 5 vectors, and use to calculate a weighted
    ;average. Our weights for the nearest points are inversely
    ;proportional to their distance from the current vector.
    ;The 1st vector nearest(0) is the current point, which we don't
    ;want.
    ;
    ;Equation being solved for the weights is:
    ;k/d1 + k/d2 + k/d3 +k/d4 = 1;
    ;d1, d2, d3, d4 =  nearby vector distances from current vector
    ;w1 = k/d1, w2 = k/d2, w3 = k/d3, w4 = k/d4
    ; w1, w2, w3, w4 = weights for nearby
    ;vectors, and k = a proportionality constant which we must calculate.
```

```
  d1 = nearest(1) & d2 = nearest(2) & d3 = nearest(3) & d4 = nearest(4)
  k = (d1*d2*d3*d4) / (d2*d3*d4 + d1*d3*d4 + d1*d2*d4 + d1*d2*d3)
  w1 = k/float(d1) & w2 = k/float(d2) & w3 = k/float(d3) & w4 = k/float(d4)

  ;x-component
  d1x = nearest_gx(1) & d2x = nearest_gx(2)
  d3x = nearest_gx(3) & d4x = nearest_gx(4)
  gx = (w1*d1x + w2*d2x + w3*d3x + w4*d4x)/4.0

  ;y-component
  d1y = nearest_gy(1) & d2y = nearest_gy(2)
  d3y = nearest_gy(3) & d4y = nearest_gy(4)
  gy = (w1*d1y + w2*d2y + w3*d3y + w4*d4y)/4.0

 end ;case else i
endcase

istep = istep + 1

;Apply Eulers algorithm to integrate to next (forward or backward) step in
;streamline
case j of
 0: begin
  ;forward
  xstream(i,istep, j) = curr_x + gx * h
  ystream(i,istep, j) = curr_y + gy * h
  end

 1: begin
  ;backward
  xstream(i,istep, j) = curr_x - gx * h
  ystream(i,istep, j) = curr_y - gy * h
  end
endcase
curr_x = xstream(i,istep, j)
curr_y = ystream(i,istep, j)

;See if out of bounds to end integration
case 1 of
 curr_x le xleft: out_of_bounds = 1 ;true
 curr_x ge xright: out_of_bounds = 1
 curr_y le ybottom: out_of_bounds = 1
 curr_y ge ytop: out_of_bounds = 1
 else: out_of_bounds = 0   ;false
endcase

end ;while istep le nsteps
```

```
  oplot, xstream(i,0:istep, j), ystream(i,0:istep, j)
  istep = 0

  endfor ;j (forward and backward streamline)
 endfor ;i (each wind vector)

end

 ;************************************************************ *****
PRO STR_FIELD2,U,V,X,Y, Missing=Missing, Length=length, Dots=dots,  $
       Title=title, position=position, noerase=noerase, color=color,$
 xtitle=xtitle, ytitle=ytitle, xmargin=xmargin, ymargin=ymargin, $
 easyview=easyview, noaxis=noaxis, nolabels=nolabels
;
;+
; NAME:
; STR_FIELD2
;
; PURPOSE:
; Produce a two-dimensional velocity field plot with streamlines
;       by finding the level sets of stream functions.
;
;       See notes at http://www.amara.com/ftpstuff/streamlines2.txt
;
; A directed arrow is drawn at each point showing the direction and
; magnitude of the field and streamlines show the field in a mass-
; conservation way.
;
; CATEGORY:
; Plotting, two-dimensional.
;
; CALLING SEQUENCE:
; STR_FIELD, U, V [, X, Y]
;
; INPUTS:
; U: The X component of the two-dimensional field.
;  U must be a two-dimensional array.
;
; V: The Y component of the two dimensional field.  Y must have
;  the same dimensions as X.  The vector at point (i,j) has a
;  magnitude of:
;
;   (U(i,j)^2 + V(i,j)^2)^0.5
;
;  and a direction of:
;
;   ATAN2(V(i,j),U(i,j)).
```

```
;
; OPTIONAL INPUT PARAMETERS:
;  X: Optional abcissae values.  X must be a vector with a length
;  equal to the first dimension of U and V.
;
; Y: Optional ordinate values.  Y must be a vector with a length
;  equal to the first dimension of U and V.
;
; KEYWORD INPUT PARAMETERS:
;      MISSING: Missing data value.  Vectors with a LENGTH greater
;  than MISSING are ignored.
;
; LENGTH: Length factor.  The default of 1.0 makes the longest (U,V)
;  vector the length of a cell.
;
; DOTS: Set this keyword to 1 to place a dot at each missing point.
;  Set this keyword to 0 or omit it to draw nothing for missing
;  points.  Has effect only if MISSING is specified.
;
; TITLE: A string containing the plot title.
;
;     POSITION: A four-element, floating-point vector of normalized
;  coordinates for the rectangular plot window.
;  This vector has the form  [X0, Y0, X1, Y1], where (X0, Y0)
;  is the origin, and (X1, Y1) is the upper-right corner.
;
;      NOERASE: Set this keyword to inhibit erase before plot.
;
; COLOR: The color index used for the plot.
;
; EASYVIEW: Set this keyword if you don't wish to have all vectors
;  shown.  For example if you have more than ~200 vectors in your
;  vector plot, the plotting will add considerable time.
;
; XMARGIN: A two-element vector that specifies the vertical margin between
;  the axis and the screen border in character units.
;
; YMARGIN: A two-element vector that specifies in the horzontal margin
;  between the map and screen border in character units.
;
; NOAXIS: Set this keyword if you wish to have no axes drawn (i.e.
;  if you will be calling this routine on top of another plot or map.
;
; NOLABELS: Set this keyword if you wish to *not* have the contours
;  of the streamlines labeled with their values.
;
; OUTPUTS:
; None.
```

```
;
; COMMON BLOCKS:
; None.
;
; SIDE EFFECTS:
; Plotting on the selected device is performed.  System
; variables concerning plotting are changed.
;
; RESTRICTIONS:
; None.
;
; PROCEDURE:
; Straightforward for vectors.  The streamlines are calculated
; using Bob Chatfield's and John Vastano's mass-conservation method.
; (i.e. producing level sets of the stream function)
;
; MODIFICATION HISTORY:
; VEL_FIELD: DMS, RSI, Oct., 1983.
;
; VEL_FIELD: For Sun, DMS, RSI, April, 1989.
;
; VEL_FIELD: Added TITLE, Oct, 1990.
;
; VEL_FIELD: Added POSITION, NOERASE, COLOR, Feb 91, RES.
;
; Created STR_FIELD as modification of VEL_FIELD,
; Nov/Dec 1993, Amara Graps NASA-Ames
;-
;
        on_error,2                      ;Return to caller if an error occurs
        s = size(u)
        t = size(v)
        if s(0) ne 2 then begin
baduv:   message, 'U and V parameters must be 2D and same size.'
                endif
        if total(abs(s(0:2)-t(0:2))) ne 0 then goto,baduv
;
        if n_params(0) lt 3 then x = findgen(s(1)) else $
                if n_elements(x) ne s(1) then begin
badxy:                  message, 'X and Y arrays have incorrect size.'
                        endif
        if n_params(1) lt 4 then y = findgen(s(2)) else $
                if n_elements(y) ne s(2) then goto,badxy
;
        if n_elements(missing) le 0 then missing = 1.0e30
        if n_elements(length) le 0 then length = 1.0

        mag = sqrt(u^2+v^2)             ;magnitude.
```

```
;--------  Subscripts of good elements  ----------------------
      nbad = 0                    ;# of missing points
      if n_elements(missing) gt 0 then begin
            good = where(mag lt missing)
            if keyword_set(dots) then bad = where(mag ge missing, nbad)
      endif else begin
            good = lindgen(n_elements(mag))
      endelse

      mag = mag(good)               ;Discard missing values
      maxmag = max(mag)
      ugood = u(good)
      vgood = v(good)

;Define grid for vector and streamline plotting
      xleft = x(0)
      xright = x(n_elements(x)-1)
      ybottom = y(0)
      ytop = y(n_elements(y)-1)
range_x = (xright - xleft)
range_y = (ytop - ybottom)

;Set grid for stream function
nsteps_x = n_elements(x);# of steps in each dimension
nsteps_y = n_elements(y);# of steps in each dimension
div_x = nsteps_x-1
div_y = nsteps_y-1
delta_x = range_x / float(div_x)
delta_y = range_y / float(div_y)
stream = fltarr(nsteps_x, nsteps_y)
stream(0,0) = 0.0 ;initialize first point
xbeg = fltarr(n_elements(ugood))
ybeg = fltarr(n_elements(vgood))

      sina = length * (xright-xleft)/s(1)/maxmag*ugood ;sin & cosine components.
      cosa = length * (ytop-ybottom)/s(2)/maxmag*vgood
      ;Note: x0, x1, y0, y1 are streamline boundaries

      ;--------------  plot to get axes  ---------------

;Set up some of the keywords
      if n_elements(title) le 0 then title = ''
if n_elements(xtitle) le 0 then xtitle = 'U(x,y)'
if n_elements(ytitle) le 0 then ytitle = 'V(x,y)'
if n_elements(xmargin) le 0 then xmargin = !x.margin
if n_elements(ymargin) le 0 then ymargin = !y.margin
      if n_elements(color) eq 0 then color = !p.color
```

```
        if n_elements(position) ne 0 then begin
           plot,[xleft,xright],[ytop,ybottom],/nodata,/xst,/yst,title=t itle, $
            noerase=noerase, color=color, xtitle=xtitle, ytitle = ytitle,$
     xmargin=xmargin, ymargin = ymargin, position = position
        endif else $
    if keyword_set(noaxis) then begin
     plot,[xleft,xright],[ytop,ybottom],/nodata, $
  noerase=noerase, color=color, xstyle=5,ystyle=5,$
  xmargin=xmargin,ymargin=ymargin
 endif else $
           plot,[xleft,xright],[ytop,ybottom],/nodata,/xst,/yst,$
    title=title, noerase=noerase, color=color, xtitle=xtitle, $
    ytitle=ytitle, xmargin=xmargin, ymargin = ymargin

 ;Set up for showing a select set of vectors
 nskip = 1
        if keyword_set(easyview) then begin
  nshow_vecs = 200. ;# of vectors to show for easy viewing (EMPIRICAL!)
  ntotal = n_elements(good) ;# of potential vectors
  ;Determine increment value for "for loop" based on nshow_vecs & ntotal
  if ntotal gt nshow_vecs then nskip = fix(ntotal/nshow_vecs)
  end ;if easyview
;
 ;----- Calculate vectors -------------
        r = .3                      ;len of arrow head
        angle = 22.5 * !dtor          ;Angle of arrowhead
        st = r * sin(angle)           ;sin 22.5 degs * length of head
        ct = r * cos(angle)
;
 ;First calculate so that we have the full vector xbeg, ybeg
        for i=0,n_elements(good)-1, 1 do begin     ;Each point
                x0 = x(good(i) mod s(1))        ;get coords of start & end
                dx = sina(i)
                x1 = x0 + dx
    xbeg(i) = x0
                y0 = y(good(i) / s(1))
                dy = cosa(i)
                y1 = y0 + dy
    ybeg(i) = y0
                endfor

 ;-------Place vectors with arrows-----------------------
        for i=0,n_elements(good)-1, nskip do begin     ;Each point
                x0 = x(good(i) mod s(1))        ;get coords of start & end
                dx = sina(i)
                x1 = x0 + dx
                y0 = y(good(i) / s(1))
```

```
            dy = cosa(i)
            y1 = y0 + dy
            plots,[x0,x1,x1-(ct*dx-st*dy),x1,x1-(ct*dx+st*dy)], $
                 [y0,y1,y1-(ct*dy+st*dx),y1,y1-(ct*dy-st*dx)], $
       color=color, thick = 1.25,noclip=0
             endfor
      if nbad gt 0 then $            ;Dots for missing?
            oplot, x(bad mod s(1)), y(bad / s(1)), psym=3, color=color

 ;-----------Create a streamline grid----------

;CURRENTLY NOT USED--------------------
 ;Set up for showing a select set of streamlines
 nskip = 1
       if keyword_set(easyview) then begin
  nshow_streams = 20.  ;# of vectors to show for easy viewing
  ntotal = n_elements(good) ;# of potential streamlines
  ;Determine increment value for "for loop" based on nshow_streams & ntotal
  if ntotal gt nshow_streams then nskip = fix(ntotal/nshow_streams)
  end ;if easyview
;-------------------------------------

 ;Fill up the Stream function grid by moving "UP" (i.e. constant x)
  yrow = 1
 while yrow le nsteps_y-1 do begin

  ;Get u{0,n} and u{0,n+1}
  gx = fltarr(2)
  gx(0) = u(0,yrow-1)
  gx(1) = u(0,yrow)

  ;Use Trapezoidal Rule with the two "gx" evaluations
  stream(0,yrow) = -0.5*(gx(0) + gx(1))*delta_y + stream(0,yrow-1)
  yrow = yrow + 1
  end ;while filling "UP"

 ;Now fill up the Stream grid by moving "ROW by ROW" (i.e. constant y)
 yrow = 0
 while yrow le nsteps_y-1 do begin

   istep = 1
  while istep le nsteps_x-1 do begin

   ;Find v{0,n} and v{0,n+1}
   gy = fltarr(2)
   gy(0) = v(istep-1, yrow)
   gy(1) = v(istep,yrow)
```

```
   ;Use Trapezoidal Rule with the two "gy" evaluations
   stream(istep,yrow) = 0.5*(gy(0) + gy(1))*delta_x + stream(istep-1,yrow)

   istep = istep + 1

   end; while istep

  yrow = yrow + 1

  end ;while filling "ROW by ROW" (yrow)

 ;----------Contour the streamline grid----------

 ;Set up levels and colors for contour
 ncontour = 25
 nclevel = ncontour+1
 datamin = min(stream)
 datamax = max(stream)
 dinc = (datamax-datamin)/ncontour
 levels = datamin+dinc*findgen(nclevel)
 clabels = intarr(nclevel)
 for i=0,ncontour do clabels(i) = 0
 for i = 0,ncontour,2 do clabels(i) = 1
 cinc = 200./ncontour
 colors=30.+cinc*findgen(nclevel)

 ;Now contour
 if keyword_set(nolabels) then begin
  contour,stream,x,y,levels=levels, c_colors=colors,/noerase, $
  xstyle=5, ystyle=5, xmargin = xmargin, ymargin= ymargin
 endif else $
  contour,stream,x,y,levels=levels, c_colors=colors,c_charsize=1.0,$
  c_labels=clabels,/follow, /noerase, xstyle=5, ystyle=5, $
  xmargin = xmargin, ymargin= ymargin

end   ;of procedure str_field2


 ;********************************************************** *****
PRO TEST_STRFIELD

;This program initializes the "u" and "v"
;vector components of a vector field to test the stream
;and velocity field routine; str_field.pro.
;-----------------------------------------------------
;Amara Graps 11-19-93 Updated: 12-30-93
;-----------------------------------------------------
```

```
u = fltarr(6,6)
v = fltarr(6,6)

;Equation for 2-D vector field is Fvec = -y i + x j
;where i and j are unit vectors pointing in ijk coords
;F1(x,y)i = -y --> our "u" vector
;F2(x,y)j = x --> our "v" vector

x=indgen(6)
y=indgen(6)

;fill in edge points
for i = 0, 5 do u(0,i) = float(-i)
for i = 0, 5 do v(i,0) = float(i)

;fill in some intermediate points
u(1,1) = -1.
for i = 0,5 do u(i,1) = -1.
u(2,2) = -2.
for i = 0,5 do u(i,2) = -2.
u(2,3) = -3.
for i = 0,5 do u(i,3) = -3.
u(4,4) = -4.
for i = 0,5 do u(i,4) = -4.
u(5,5) = -5.
for i = 0,5 do u(i,5) = -5.

v(1,1) = 1.
for i = 0,5 do v(1,i) = 1.
v(2,2) = 2.
for i = 0,5 do v(2,i) = 2.
v(3,3) = 3.
for i = 0,5 do v(3,i) = 3.
v(4,4) = 4.
for i = 0,5 do v(4,i) = 4.
v(5,5) = 5.
for i = 0,5 do v(5,i) = 5.

print, 'testing str_field'

if !version.os eq 'MacOS' then begin
 device, get_screen = ssize
 xsize = ssize(0)*.7
 ysize = ssize(1)*.8
 window, xsize = xsize, ysize = ysize
 end

loadct, 12
```

```
print, 'Testing Vector Field Streamline approach'
str_field1, u, v, xmargin=[5,5],ymargin=[0,0]

stop, 'Press .con to continue..'

print, 'Testing Level Sets of Stream Function approach'
str_field2,u,v,x,y,xmargin=[5,5],ymargin=[5,5],/nolabels

end
;*******************************************************
```

File Attachments

1) test_strfield.pro, downloaded 121 times