
Subject: Re: VELOVECT problem

Posted by [hcp](#) on Wed, 16 Aug 1995 07:00:00 GMT

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

In article <40nmmq\$aml@hugin.aau.dk>, rosentha@aauobs.obs.aau.dk (Colin Rosenthal) writes:

|> Is there a problem with VELOVECT in IDL 4.0? When I try to call
|> it with /NOERASE I get a "Keyword NOERASE not allowed in PLOTS"
|> error.
|>

Yes there is. RSI clearly know not a lot about their customers or they would not make it so hard to put (e.g.) wind vector arrows on top of a filled contour plot of wind strength. Fortunately velovect is written in IDL so you can copy it from /usr/local/rsi/idl/lib/ or wherever your system keeps it and hack your own copy. I'd change its name to something else so you know which version you are using.

The problem is that RSI have used the _EXTRA mechanism to ensure that any odd keywords you feed to velovect (e.g. /noerase) are passed on to plot. They pass them all on to plots and oplot too, where they can cause errors.

You therefore need to change the following.

The first lines of the program (now in file velo.pro) become

```
PRO VELO,U,V,X,Y, Missing = Missing, Length = length, Dots = dots, $  
    Color=color, noerase=noerase, xrange=xrange,yrange=yrange, $  
    xstyle=xstyle,ystyle=ystyle, _EXTRA = extra  
;
```

The plot statements

```
if n_elements(position) eq 0 then begin  
    plot,[x_b0,x_b1],[y_b1,y_b0],/nodata,/xst,/yst, $  
    color=color, _EXTRA = extra  
endif else begin  
    plot,[x_b0,x_b1],[y_b1,y_b0],/nodata,/xst,/yst, $  
    color=color, _EXTRA = extra  
endelse
```

must be changed to

```
if n_elements(position) eq 0 then begin  
    plot,[x_b0,x_b1],[y_b1,y_b0],/nodata, $  
    color=color, _EXTRA = extra,xstyle=xstyle,$  
    noerase=noerase,xrange=xrange,yrange=yrange,ystyle=ystyle  
endif else begin  
    plot,[x_b0,x_b1],[y_b1,y_b0],/nodata, $
```

```
        color=color, _EXTRA = extra, noerase=noerase,$
        xrange=xrange,yrange=yrange,ystyle=ystyle,xstyle=xstyle
    endelse
```

Doubtless there are other incompatible keywords to add to this list.

Why that if-else business is there I don't know, both plot statements look the same to me.

You could test the code with this short test program

```
; Test Program for velocity vectors on top of countour plot
n=29
x=findgen(n)*2*!Pi/n
y=x
vx=fltarr(n,n)
vy=vx
z=vx
for j=0,n-1 do begin
  for i=0,n-1 do begin
    vx(i,j)=cos(x(i))*sin(y(j))
    vy(i,j)=sin(x(i))*sin(y(j))*cos(3*y(j)*x(i))
    z=sqrt(vx*vx+vy*vy)
  endfor
endfor

ra=[-0.2,6.2]
contour,z,x,y,lev=(findgen(20)/20),xrange=ra,yrange=ra,/fil, /xst,/yst
velo,vx,vy,x,y,xrange=ra,yrange=ra,/noerase,/xst,/yst
end
; End of test program
```

Hope this works.
Hugh

P.S. RSI: if you are reading this, you might want to correct these errors in the next upgrade. My consulting fee is a mere \$800, all hard currency accepted.

Subject: Re: VELOVECT problem
Posted by [afl](#) on Wed, 16 Aug 1995 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

I wrote this as a "fix" to velovect. It allows one to place vectors over a pre-drawn plot, like a map background.

Please let me know if it works for you.
There is a nice text widget at the top which
provides usage information.

;This "front end" procedure is a help tool for velovect.

pro vec_help

```
base = widget_base(title='VEC_HELP', /row)
list = widget_text(base, xsize=85, ysize=20, scroll=1, $
    font='-adobe-helvetica-medium-r-normal--14-140-75-75-p-77-is o8859-1', $
    value=[$
'===== ', $
'                >>>> MAP_VEC HELP MENU
<<<<', $
'', $
'        The map_vec procedure plots vectors within an existing plot
space.', $
'        Usually a map is produced, then vectors are overlayed using
velovect.', $
'', $
'-----', $
'USAGE:   map_vec, u, v, x, y, missing=missing, length=length, maxvec=maxvec,
round=round, $, $
'          dots=dots, color=color, everyx=everyx, everyy=everyy,
arrowsize=arrowsize, $, $
'          limit=limit, refvec=refvec, refunit=refunit,
refposx=refposx, $, $
'          refposy=refposy, reflabely=reflabely, charsize=charsize,
help=help', $
'===== ', $
'', $
'PARAMETERS:', $
'  u,v..... 2-D data arrays specifying the zonal and meridional
velocity components.', $
'  x,y..... Meridional and zonal location vectors (longitude,
latitude).', $
'', $
'KEYWORDS:', $
'  missing..... Do not plot vectors with a magnitude greater or equal to
this value.', $
'  length..... Length factor. The default of 1.0 makes the longest
vector the', $
'    length of one grid cell.', $
'  maxvec..... Maximum vector magnitude for scaling. Default is
```

```

computed', $
'   round..... Round computed maxvec so it is divisible by 5.', $
'   dots..... After setting dots to 1, a dot is placed at missing data
points.', $
'   color..... Color index for drawing vectors. Default is zero.', $
'   everyx..... Plot every 2nd, 3rd, 4th... vector in the x direction.
Default is 1.', $
'   everyy..... Plot every 2nd, 3rd, 4th... vector in the y direction.
Default is 1.', $
'   arrowsize..... Factor by which default arrowsize is multiplied.', $
'   limit..... vector containing [latmin, lonmin, latmx, lonmax] for
plotting area.', $
'', $
'   refvec..... After setting refvec to 1, a "max vector magnitude"
arrow appears on the plot.', $
"   refunit..... String variable containing unit of the reference vector.
Default is 'm s!U-1!n.'", $
'   refposx..... Ending x position (in data coords.) for drawing refvec.
Default is computed.', $
'   refposy..... Y position (in data coords.) for drawing refvec.
Default is computed.', $
'   reflabely..... Y position (in data coords.) for drawing label beneath
refvec. Default is computed.', $
'   charsize..... Character size for reference vector label. Default is
.88', $
'   help..... Bring up this help menu.', $
'
.....
.....'+$
'
.....'+$
'.....', $
'', $
'   EXAMPLE: Plotting global taux, tauy data.', $
"       taux = rz('/data/obs/nmc/stress/taux_9_90', xloc=x, yloc=y)", $
"       tauy = rz('/data/obs/nmc/stress/tauy_9_90')", $
'       gplot, taux, x, y, /nodata ... or ... map_set, 0, 180,
/cont, /cyl', $
'', $
'       map_vec, taux, tauy, x, y, missing=999., everyx=2, everyy=1,
length=5, /dots, $', $
"       /refvec, refunit='dynes cm !e-2!n', refposx=340.,
refposy=-100., $", $
'       reflabely=-106.', $
'   '])

widget_control, /realize, base

end

```

```

pro map_vec, u, v, x, y, missing=missing, length=length, maxvec=maxvec,
round=round, $
    dots=dots, color=color, everyx=everyx, everyy=everyy,
arrowsize=arrowsize, $
    limit=limit, refvec=refvec, refunit=refunit, refposx=refposx,
refposy=refposy, $
    reflabely=reflabely, charsize=charsize, help=help

```

```

;
;
; NAME:
; MAP_VEC
;
;
;
; PURPOSE:
; Produce a two-dimensional velocity field plot.
;
; A directed arrow is drawn at each point showing the direction and
; magnitude of the field.
;
;
; CATEGORY:
; Plotting, two-dimensional.
;
;
; CALLING SEQUENCE:
; MAP_VEC, 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:
;
;  $\text{SQRT} ( U(i,j)^2 + V(i,j)^2 )$ 
;
; and a direction of:
;
;  $\text{ATAN2} ( V(i,j), U(i,j) )$ 
;
;
; OPTIONAL INPUT PARAMETERS:
; X: Optional abscissae 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.
;
; COLOR: The color index used for the arrows.
;
;   EVERY: Plot every ____ vector. Default is 1.
;
;   REFARROW: Plot a maximum vector magnitude. Default is 0 (NO).
;
;   REFUNIT: Unit of max arrow. Default is m s!e-1!n (i.e., m/s)
;
;   REFPOSX: X starting position of maxxarrow. Default is computed.
;
;   REFPOSY: Y position of maxxarrow. Default is computed.
;
;   REFLABELY: Y position of refveclabel. Default is computed.
;
;
; OUTPUTS:
; None.
;
;
; COMMON BLOCKS:
; None.
;
;
; SIDE EFFECTS:
; Plotting on the selected device is performed.
;
;
; RESTRICTIONS:
; None.
;
;
;

```

```

; PROCEDURE:
; Straightforward. The system variables !XTITLE, !YTITLE and
; !MTITLE can be set to title the axes.
;
;
; MODIFICATION HISTORY:
; DMS, RSI, Oct., 1983.
;
; For Sun, DMS, RSI, April, 1989.
;
; Added TITLE, Oct, 1990.
;
; Added POSITION, NOERASE, COLOR, Feb 91, RES.
;
; Added all keywords past color.      June 1993, Andrew F. Lough
; Draw to current plot area (usually a map).

```

```

on_error, 2      ;Return to caller if an error occurs
help = keyword_set(help)
if (help eq 1 or n_params() eq 0) then begin
    vec_help
    message, 'Stopped for help in map_vec.pro'
endif

```

```

s = size(u)
t = size(v)

```

```

; Check size of input arrays.

```

```

if n_params(1) lt 4 then $
    message, 'Must specify u, v, x, y'

```

```

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_elements(x) ne s(1) or n_elements(y) ne s(2)) then $
    message, 'X and Y arrays have incorrect size.'

```

```

; Set some default values.

```

```

if n_elements(missing) le 0 then missing = 1.0e10
if n_elements(length) le 0 then length = 1.0
if n_elements(maxvec) le 0 then maxvec = -1.
if n_elements(round) le 0 then round = 0

```

```

if n_elements(everyx) le 0 then everyx = 1
if n_elements(everyy) le 0 then everyy = 1
if n_elements(arrowsize) le 0 then arrowsize = 1.
if n_elements(limit) le 0 then limit = [-100.,-10000.,100.,10000.]
if n_elements(title) le 0 then title = ""
if n_elements(charsize) le 0 then charsize = .88
if n_elements(color) le 0 then color = 0

latmin = limit(0)      ;Get geographical limits
lonmin = limit(1)      ;for plotting the data
latmax = limit(2)
lonmax = limit(3)

mag = sqrt(u^2 + v^2)   ;Get magnitude of all vectors

; Get subscripts of good and bad elements.
; good = those points where vector magnitude is less than missing value.
; bad = those points where vector magnitude is greater or equal to missing value.
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

; Using good points, find maximum vector magnitude for scaling.
if (maxvec eq -1.) then begin
    mag2 = mag(good)      ;Get magnitude of good points.
    if (max(mag2) le 2) then maxvec = fix(max(mag2))
    if (max(mag2) gt 2) then maxvec = fix(max(mag2)+1)
    if (round eq 1) then begin
        maxvec = fix((max(mag2)+1)/5)*5 ;Round maxvec dwn to number
divisible by 5
        print, 'Max vector length for scaling is made divisible by 5:
',maxvec
    endif
    if (maxvec eq 0.) then maxvec = 1
endif

;Grid spacing needed for scaling vectors
deltax = (max(x) - min(x)) / (s(1)-1)
deltay = (max(y) - min(y)) / (s(2)-1)

; Plot vectors and arrow heads (loop through all points).
for j = 0, s(2)-2, fix(everyy) do begin
    for i = 0, s(1)-1, fix(everyx) do begin

```

```

; Get scaled vector componets.
; length = If three, the longest vector covers three grid cells.
; u,v = The actual velocity components.
; maxvec = The maximum wind component.

    if (mag(i,j) lt missing) then begin
        dx = length * (deltax) * (u(i,j)/maxvec) ;Get (u,v) components
        dy = length * (deltay) * (v(i,j)/maxvec) ;that will be plotted

        x0 = x(i) ;Get beginning/ending coords of vector
        x1 = x0 + dx
        y0 = y(j)
        y1 = y0 + dy

        if (y0 ge latmin and y1 le latmax and $
            x0 ge lonmin and x1 le lonmax) then begin
            arrow_andy, x0, y0, x1, y1, color=color, /data, $

        hsize=float(arrowsize)*(!d.x_size/94.)*(mag(i,j)/(2.*maxvec) )
        endif
    endif
endfor
endfor

; Plot a "maximum vector" arrow at the bottom righthand corner of the plot
if n_elements(refvec) gt 0 then begin

; Determine new arrow head size and angle for the reference vector
    r = .35 ;Length of arrow head
    angle = 15. * !dtor ;Half-co-angle of arrowhead
    st = r * sin(angle) ;Sin 15 degs * length of head
    ct = r * cos(angle)

    dx = length * deltax ;Get x-component
    dy = length * deltay ;Get y-component
    dy=0

; Compute (or use supplied value) for the ending x position, and the
; y position of the reference vector.
    if n_elements (refposx) le 0 then refposx= max(x) - dx*3.
    if n_elements (refposy) le 0 then refposy= min(y) - (y(3)-y(0))

    x0 = refposx - dx
    y0 = refposy
    x1 = x0 + dx
    y1 = y0 + dy ;No y variation (reference vector is horizontal)

```

```

; Find default (x,y) positions for max vector label
  xlabel = (x0 + x1)/2.
  ylabel = y0 - (y(4)-y(0))
  if n_elements(reflabely) gt 0 then ylabel=reflabely

; Determine (or use supplied value) unit of the reference vector
  if n_elements(refunit) le 0 then refunit= 'm s!U-1!n'

  max_label = strcompress(string(maxvec), /remove_all) ;Get maxvec
  reftitle = max_label + ' '+ refunit
  xyouts, xlabel, ylabel, reftitle, align=.5, charsize=charsize,
color=color

; Found it more universally acceptable to plot refvec in normalized coords.
  new_coords = convert_coord (x0, y0, /data, /to_norm)
  xx0 = new_coords(0) & yy0 = new_coords(1) ;refvec starts here

  new_coords = convert_coord (x1, y1, /data, /to_norm)
  xx1 = new_coords(0) & yy1 = new_coords(1) ;refvec ends here

  new_coords = convert_coord ( x1-(ct*dx-st*dy), y1-(ct*dy+st*dx),$
    /data, /to_norm)
  xx2 = new_coords(0) & yy2 = new_coords(1) ;top arrowheads end here

  new_coords = convert_coord ( x1-(ct*dx+st*dy), y1-(ct*dy-st*dx),$
    /data, /to_norm)
  xx3 = new_coords(0) & yy3 = new_coords(1) ;bot arrowheads end here

  plots,[xx0, xx1, xx3, xx1, xx3], $
    [yy0, yy1, yy3+2.*(yy1-yy3), yy1, yy3], color=color, $
    thick=1., /norm

endif

; Place dots at missing data points
  if nbad gt 0 then $
    oplot, x(bad mod s(1)), y(bad/s(1)), psym=3, color=color

end

```
