

---

Subject: Re: vector plots on maps

Posted by [Andy Loughe](#) on Wed, 04 Feb 1998 08:00:00 GMT

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

---

> is there a routine by which I can overplot a vector field on a map?

I wrote this, but don't use it much. Have fun...

;This "front end" procedure is a help tool for map\_vec.

```
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 overlaid  
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 .78', $
'    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.
Loughe
; 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 = .78
  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

hsize=float(arrowsize)*(!d.x_size/100.)*(mag(i,j)/(2.*maxvec ))
      arrow_andy, x0, y0, x1, y1, color=color, /data,
hsize=hsize
      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 = .4 ;Length of arrow head
  angle = 20. * !dior ;Half-co-angle of arrowhead
  st = r * sin(angle) ;Sin 20 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(reflably) gt 0 then ylabel=reflably

; 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

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

    xyouts, .5*(xx0+xx1), yy0-.015, reftitle, align=.5,
charsize=charsize, color=color, /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

.....
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

PRO ARROW_ANDY, x0, y0, x1, y1, HSIZE = hsize, COLOR = color, HTHICK =
hthick, $
    THICK = thick, DATA = data, DEVICE = device, NORMALIZED = norm, $
    SOLID = solid
;+
; NAME:   ARROW
; PURPOSE: Draw a vector(s) with an arrow head
; CATEGORY: Graphics
; CALLING SEQUENCE:
; ARROW, x0, y0, x1, y1
; INPUTS:
; (x0, y0) = coordinates of beginning of vector(s). May be arrays
; or scalars. Coordinates are in DEVICE coordinates
; unless otherwise specified.
; (x1, y1) = coordinates of endpoint (head) of vector.
; x0, y0, x1, y1 must all have the same number of elements.
; KEYWORD PARAMETERS:
; DATA - if set, implies that coordinates are in data coords.
; NORMALIZED - if set, coordinates are specified in normalized coords.
; HSIZE = size of arrowhead. Default = 1/64th the width of the device,
; (!D.X_SIZE / 64.).
; If the size is positive, it is assumed to be in device
; coordinate units. If it is NEGATIVE, then the head length
; is set to the vector length * abs(hsize), giving heads

```

```

; proportional in size to the bodies. The size is defined as
; the length of each of the lines (separated by 60 degrees)
; that make the head.
; COLOR = drawing color. Default = highest color index.
; HTHICK = thickness of heads. Default = 1.0.
; SOLID = if set, make a solid arrow, using polygon fills, looks better
; for thick arrows.
; THICK = thickness of body. Default = 1.0.
;
;
; OUTPUTS:
; No explicit outputs.
; SIDE EFFECTS:
; RESTRICTIONS:
; PROCEDURE:
; Straightforward.
; Examples:
; Draw an arrow from (100,150) to (300,350) in DEVICE units.
; ARROW, 100, 150, 300, 350
;
; Draw a sine wave with arrows from the line Y=0 to
; sin(x/4).
; X = FINDGEN(50)
; Y = SIN(x/4) ;Make sin wave
; PLOT, X, Y
; ARROW, X, REPLICATE(0,50), X, Y, /DATA
; MODIFICATION HISTORY:
; DMS, Feb, 1992.
; DMS, Sept, 1992. Added /SOLID.
;-

; Draw an arrow with a head from (x0,y0) to (x1, y1). Params may be
; vectors.

; Set up keyword params

if n_elements(thick) eq 0 then thick = 1.
if n_elements(hthick) eq 0 then hthick = thick

;Head size in device units
if n_elements(hsize) eq 0 then arrowsize = !d.x_size/64. * (hthick/2. >
1) $
else arrowsize = float(hsize)
if n_elements(color) eq 0 then color = !P.color

angle = 20.0 ;degrees for head angle

mcost = - cos( !dtor * angle )
sint = sin( !dtor * angle )

```

```
msint = - sint
```

```
for i = 0, n_elements(x0)-1 do begin      ;Each vector
  if keyword_set(data) then $           ;Convert?
    p = convert_coord([x0(i),x1(i)], [y0(i),y1(i)], /data, /to_dev) $
  else if keyword_set(norm) then $
    p = convert_coord([x0(i),x1(i)], [y0(i),y1(i)], /norm, /to_dev) $
  else p = [[x0(i), y0(i)], [x1(i), y1(i)]]
```

```
xp0 = p(0,0)
xp1 = p(0,1)
yp0 = p(1,0)
yp1 = p(1,1)
```

```
dx = float(xp1-xp0)
dy = float(yp1-yp0)
zz = sqrt(dx^2 + dy^2) ;Length
```

```
if zz gt 1e-6 then begin
  dx = dx/zz ;Cos th
  dy = dy/zz ;Sin th
endif else begin
  dx = 1.
  dy = 0.
  zz = 1.
endelse
if arrowsize gt 0 then a = arrowsize $ ;a = length of head
else a = -zz * arrowsize
```

```
xxp0 = xp1 + a * (dx*mcost - dy * msint)
yyp0 = yp1 + a * (dx*msint + dy * mcost)
xxp1 = xp1 + a * (dx*mcost - dy * sint)
yyp1 = yp1 + a * (dx*sint + dy * mcost)
```

```
;print, xxp0, yyp0, xxp1, yyp1
;if ( ( xp0 le 900) and (yp0 le 900) and (xp1 le 900) and (yp1 le 900)
) $
;then begin
```

```
if keyword_set(solid) then begin ;Use polyfill?
  b = a * mcost*.9 ;End of arrow shaft (Fudge to force join)
  plots, [xp0, xp1+b*dx], [yp0, yp1+b*dy], /DEVICE, $
  COLOR = color, THICK = thick
  polyfill, [xxp0, xxp1, xp1, xxp0], [yyp0, yyp1, yp1, yyp0], $
  /DEVICE, COLOR = color
```

```
endif else begin
```

```
plots, [xp0, xp1], [yp0, yp1], /DEVICE, COLOR = color, THICK = thick,  
$  
    noclip=0  
plots, [xxp0,xp1,xxp1],[yyp0,yp1,yyp1], /DEVICE, COLOR = color, $  
    THICK = hthick, noclip=0  
endelse  
  
;endif  
  
ENDFOR  
  
end
```

---