Subject: Re: How to do polar plots with logarithmic axis in radial coordinate?
Posted by Paul van Delst on Wed, 07 Feb 2001 14:21:36 GMT
View Forum Message <> Reply to Message

Charlie Zender wrote:
>
> Craig Markwardt wrote:
>
>> Could you simply take the ALOG10() logarithm of the data before
>> plotting it? Easier to re-label the axis than re-invent the world...
>>
>
> This would cause the radial coordinate to be negative-valued which
> would have unpleasant results. It's possible someone could get
> this method to work but I tried without success.

I did the following:

```
IDL> r=10.0^(findgen(100)/20.0)
IDL> theta=findgen(100)/5.
IDL> !p.charsize=2.5
IDL> plot, alog10(r), theta, xsty=4,ysty=4,/polar
IDL> axis,0,0,xax=0,xtickformat='expticks_log'
% Compiled module: EXPTICKS_LOG.
IDL> axis,0,0,yax=0,ytickformat='expticks_log'
```

where the expticks_log.pro is:

```
FUNCTION expticks_log, axis, index, value
  tickmark = '10!E' + $
             STRTRIM(STRING(value,FORMAT='(f5.1)'),2) + $
             '!N'
  RETURN, tickmark
END
```

The !P.CHARSIZE was just so I could see the exponents. Is this what you are looking for? Keep in mind that the format string for the expticks_log.pro won't work for exponents less than 0.0 - you'll have to get smart about checking for the range and then setting the format string based on that (rather than the f5.1 above, e.g.:

```
  CASE 1 OF

;   -- Exponent is less than zero ->
;   -- fractional ticklabel
    ( exponent LT 0 ): format = '( f' + $
                               STRTRIM(ABS(exponent)+2,2) + $
                               '.' + $
                               STRTRIM(ABS(exponent),2) + $
```

```
                        ' )'

;   -- Exponent is greater than or = to zero ->
;   -- whole number ticklabel
    ( exponent GE 0 ): format = '( i' + $
                        STRTRIM(ABS(exponent)+1,2) + $
                        ' )'

  ENDCASE
```

and then use the format string in the return value, e.g.

```
  RETURN, '10!E' + STRING( value, FORMAT = format ) + '!N'
```

or something similar depending on your tastes/needs.

Hope some of this is helpful, although I have to admit, the fact that IDL doesn't have a stock polar plotting routine that produces a circular graph with the radial and concentric circle tickmark axes is a bit ridiculous. Farting about with /POLAR and AXIS and whatnot is sort of like using OG to plot, x, y - and in the end you still end up with Cartesian-like axes.

Maybe there is a polar plotting routine out there in IDL software land somewhere. It is sorely needed.

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
pvandelst@ncep.noaa.gov Alexander Pope.

Subject: Re: How to do polar plots with logarithmic axis in radial coordinate?
Posted by Mirko Vukovic on Thu, 08 Feb 2001 16:05:26 GMT

View Forum Message <> Reply to Message

In article <3A8159F0.6BCE06BC@ncep.noaa.gov>,
  Paul van Delst <pvandelst@ncep.noaa.gov> wrote:
> Charlie Zender wrote:
>>
>> Craig Markwardt wrote:
>>
>>> Could you simply take the ALOG10() logarithm of the data before
>>> plotting it? Easier to re-label the axis than re-invent the

world...
>>>
>>
>> This would cause the radial coordinate to be negative-valued which
>> would have unpleasant results. It's possible someone could get
>> this method to work but I tried without success.
>

For cases of positive numbers with a huge range, I often use the arc hyperbolic sine function. It is approximately linear for arguments<1, and logarithmic for large arguments>1. I include it way at the end of the post (last two routines). I use it farily often, but never bothered to write and accompanying tick marking routine.

> Hope some of this is helpful, although I have to admit, the fact that IDL doesn't have a
> stock polar plotting routine that produces a circular graph with the radial and concentric
> circle tickmark axes is a bit ridiculous. Farting about with /POLAR and AXIS and whatnot
> is sort of like using OG to plot, x, y - and in the end you still end up with
> Cartesian-like axes.

agreed. In some of my applications, I use MAP for polar plotting. I'll excerpt parts of the code, but you will need to modify it for your applications. The code is part of an object, so some variables are stored as fields of SELF. But that is all they are, variables. If you provide them, they do not have to be parts of an object.

To give you some ideas as to what is involved, the following set-up the plot:

```
; convert rmin and rmax (stored as vector in Radial Frame Limits) to latitudes
   LatRange=self->r2Lat(RadialFrameLimits)
; convert min and max angle (can be 0 to 360) to longitude
   LonRange=self.FrameLimits[[1,3]]*!radeg
; store this as part of the object in which the whole things is done
    self.DataLatLonRanges=[LatRange[0],LonRange[0],LatRange[1],L onRange[1]]
; rotate map so that 0 angle points to the right
   Rotation=-90;+self.Orient*!radeg
   ;; the map is plotted without the default border
; put up the polar grid.  We can plot data over that, discussed below.
   map_set,90,0,Rotation,/Azimuthal,/iso,/noborder, $
     limit=self.DataLatLonRanges,NoErase=NoErase, $
     _extra=rPropertiesKeywordList
```

Now this requires two routines for conversion from data to latitudes
and back:

```
function Polar_PlotFrame::R2Lat,R
;@private
;
;function that converts radius to latitude.  This is used to translate
;data into units that MAP understands.


   Rmax=self.FrameLimits[2]
   LatRange=self.LatRange

   RelR=R/Rmax
   Lat=RelR*(LatRange[1]-LatRange[0])+LatRange[0]

   return,Lat
end

function Polar_PlotFrame::Lat2R,Lat
;@private
; function that converts from latitutde to radius

   Rmax=self.FrameLimits[1]
   LatRange=self.LatRange

   RelR=(Lat-LatRange[0])/(LatRange[1]-LatRange[0])
   R=RelR*Rmax

   return,R
end
```

For these to work, I need this somewhere at the start of the program.
Thus the map will have the latitude range from 90 (radius 0) to 0.
(max radius, to be determined later)

```
   self.LatRange = [90.,0.]
```

Finally, to plot the data, I do

```
; convert coords from data to latitude
    self.oPlotFrame-> AdjustCoords,*self.pIndependentVariable, $
     *self.pDependentVariable,AngleCoord,RadialCoord

; contour will work too!
    plots,AngleCoord,RadialCoord, $
     _extra=rPlotPropertiesKeywordList
```

And, finally, this needs the services of AdjustCoords:

```
pro Polar_PlotFrame::AdjustCoords,Phase,Mag,Lon,Lat
; converts coordinates from angle and radius to longitude and
; latitude.

    ;; convert negative magnitude to positive, and correct angle
    iNegMag =  where(Mag LT 0,cNegMag)
    CorrPhase = Phase
    CorrMag = Mag
    IF cNegMag NE 0 THEN BEGIN
       CorrPhase[iNegMag] = CorrPhase[iNegMag]+!pi
    CorrMag[iNegMag] = -CorrMag[iNegMag]
    ENDIF
    CorrPhase = CorrPhase MOD !twopi

    ;; radius to latitude
    Lat = self->r2lat(CorrMag)
    Lon = CorrPhase*!radeg

return
END
```

Now for the routines for arc sinh.

Here is the asinh, that can handle scalars and vectors

```
;+
; return the inverse hyperbolic sine of the argument.  The calculation
is
; performed in double precision because of the addition of 1 under the
; square root.  It might be better to test for size and return the
; approximate value of the sqarre root.
;
; Written by Mirko Vukovic, around 1990
;-
 FUNCTION ASINH,ARG

;create the result array
type=size(arg)
type_res = type
dim = type(0)
type_res(dim+2) = 32
res = m$replicate(type_res)
;fill it in with results
index1 = where (abs(arg) lt 1.d3,count)
```

```
if count ne 0  then  $
 res(index1) = alog(arg(index1)+sqrt(arg(index1)^2+1.d00))
index2 = where(arg le -1.d3,count)
if count ne 0 then $
 res(index2) = -alog(-2.*arg(index2))
index3 = where(arg ge 1.d3,count)
if count ne 0 then  $
 res(index3) = alog(2.*arg(index3))


; bring result to original type of the argument
if type(dim+2) ne 32 then res=float(res)
return,res


end
```

It requires mv_replicate, similar to IDL's replicate, but can handle
scalars (there is probably a better way)

```
;+
; NAME:
;       MV_REPLICATE
;
; PURPOSE:
;       To emulate the REPLICATE function of the old version of IDL
;
; CATEGORY:
;       Variable massaging
;
; CALLING SEQUENCE:
;       result=MV_REPLICATE(INFO,type=type)
;
; INPUTS:
;       INFO - a vector, of SIZE-like properties
;
; OPTIONAL INPUT PARAMETERS:
;       None
;
; KEYWORD PARAMETERS:
;       TYPE -- (optional) integer assigns the type of the variable.
If not
;       present, the type present in INFO is assigned to the varible.
;       The value of type should be
;  1: binary
;  2: integer
;  3: long integer
;  4: floating
;  5: double precision
```

```
;  6: complex
;
;
; OUTPUTS:
;       RESULT - a variable specified according to INFO
;
; OPTIONAL OUTPUT PARAMETERS:
;       None
;
; COMMON BLOCKS:
;       None
;
; SIDE EFFECTS:
;       If INFO does not have the total number of elements in the
;       variable, that is added to it.
;
; RESTRICTIONS:
;       None
;
; PROCEDURE:
;       Straightforward.  Checking is done to see if the total number
of
;       elements in the variable is present in INFO.  If not, it is
;       calculated and added to it.
;
; MODIFICATION HISTORY:
;       Written and performed by Mirko Vukovic, sometimes around 1990
;
;-
function mv_replicate,info,type=type

nod = info(0)
infod = n_elements(info)

; make the INFO array complete
if infod ne nod+3 then begin  ; total no. of elemets is
missing
 t=1
 for i=1,nod do t=t*info(i)
 info=[info,t]
endif

if info(0) ne 0 then begin   ; this is for an array
        if keyword_set(type) then res=make_array(size=info,type=type) $
        else res = make_array(size=info)
endif else begin
 if keyword_set(type) then begin
 case type of ; and this for a scalar, info(1) has variable type
```

```
info
 0:  begin
  print, 'MV_REPLICATE: cannot make variable of undefined
type.'
  stop
  end
 1: res=0b
 2: res=0
 3: res=long(0)
 4: res=0.
 5: res=0.d00
 6: res=complex(0.,0.)
 else:  begin
  print, 'MV_REPLICATE: cannot make structure or string
variable.'
  stop
  end
 endcase
 endif else begin
 case info(1) of ; and this for a scalar, info(1) has variable
type info
 0:  begin
  print, 'MV_REPLICATE: cannot make variable of undefined
type.'
  stop
  end
 1: res=0b
 2: res=0
 3: res=long(0)
 4: res=0.
 5: res=0.d00
 6: res=complex(0.,0.)
 else:  begin
  print, 'MV_REPLICATE: cannot make structure or string
variable.'
  stop
  end
 endcase
 endelse
endelse
return,res
end
```

Sent via Deja.com
http://www.deja.com/

Subject: Re: How to do polar plots with logarithmic axis in radial coordinate?
Posted by Charlie Zender on Thu, 08 Feb 2001 17:33:25 GMT
View Forum Message <> Reply to Message

Thanks, I diddled with this all day and finally came up with something
reasonable that I hate but I'll post it in case anyone can improve on it.
Basically it takes the log10 of the data if the dynamic range is greater
than an order of magnitude, and call a special tick marking routine that
just plots the positive exponents on the positive x axis.

```
if(ffc) then abc=phz_fnc_ffc else abc=phz_fnc
abc=[reverse(abc(1:n_elements(abc)-1)),abc]
if (max(abc)/min(abc)) gt 10 then xpn_flg=1 else xpn_flg=0; Plot data by exponent
if xpn_flg then begin
 scl_xpn=abs(floor(min(alog10(abc))))+1
 print,'scl_xpn = ',scl_xpn
 scl=10.0^scl_xpn
 abc=alog10(scl*abc)
endif; endif xpn_flg
abc_min=min(abc)
abc_max=max(abc)
rng_x=[abc_min,abc_max]
rng_x=[-abc_max,abc_max]
ord=ngl_dgr*!pi/180.0
ord=[-1.0*reverse(ord(1:n_elements(ord)-1)),ord]
ord_min=min(ord)
ord_max=max(ord)
rng_y=[ord_min,ord_max]
rng_y=rng_x
if prn then chr_sz=2.0 else chr_sz=1
ttl=string(aer_lng_nm)
x_ttl=''
y_ttl='!5Scattering Angle !7H!5 (radians)'

if prn then begin
 mrg_top=0.6 ; 2 is default
 if top then mrg_top=mrg_top+1.1
 mrg_btm=0.5 ; 4 is default
 if btm then mrg_btm=mrg_btm+2.7
 if not top then ttl=''
 if not btm then x_ttl=''
 fl_nm_out='/data/zender/ps/phz_fnc_plr.ps'
 open_ps,fl_nm=fl_nm_out,x_sz=x_sz,y_sz=y_sz,/ps,/color
endif; endif prn

; polar plot phz_fnc
print,'abc = ',abc
print,'abc_min = ',abc_min,', abc_max = ',abc_max
xlg_flg=0
```

```
ylg_flg=0
 plot,abc,ord,xlog=xlg_flg,ylog=ylg_flg,tit=ttl,xtit=x_ttl,xr
ange=rng_x,yrange=rng_y,xmargin=[mrg_lft,mrg_rgt],ymargin=[m
rg_btm,mrg_top],xstyle=5,ystyle=5,thick=2.0,charsize=chr_sz,
linestyle=0,color=clr_blk_idx,/nodata,/polar,psym=2,position =aspect(1.0)
 oplot,abc,ord,max_value=1.0e20,thick=2.0,linestyle=0,color=c lr_blk_idx,/polar
if xpn_flg then tck_fmt_fnc='plr_tck_fmt' else tck_fmt_fnc='pst_tck_fmt'
 axis,0.0,0.0,xaxis=0,xlog=0,xtick_get=x_tck,xstyle=1,charsiz
e=0.75*chr_sz,color=clr_blk_idx,xtickformat=tck_fmt_fnc ; IDL 3.6 UG p. 14-24
 axis,0.0,0.0,yaxis=0,ylog=0,ytick_get=y_tck,ystyle=1,charsiz
e=0.75*chr_sz,color=clr_blk_idx,ytickformat='null_fmt' ; IDL 3.6 UG p. 14-24

; Draw circles
max_dat=max(abc)
for idx=0,n_elements(x_tck)-1 do begin
 crc_val=circle(0,0,x_tck(idx))
 if idx ne 0 then plots,crc_val,color=2
endfor; end loop over tck

if info then begin
ln_lgn_x1=0.07
ln_lgn_dx=0.07
ln_lgn_x2=ln_lgn_x1+ln_lgn_dx
txt_lgn_x=ln_lgn_x2+0.01
txt_lgn_sz=1.75
lgn_y_top=0.77
lgn_dy=0.05
lgn_y=lgn_y_top-indgen(6)*lgn_dy
if xpn_flg then  xyouts,txt_lgn_x+0.45,0.22,'!5log!I10!N[10!E'+auto_sng(scl_x
pn,0)+'!N!8p!5(!7H!5)!5]',color=clr_blk_idx,size=0.75*chr_sz ,/NORMAL
xyouts,txt_lgn_x+0.45,lgn_y(0),'!7k!5 = '+auto_sng(wvl_dbg*1.0e6,2)+'
!7l!5m',color=clr_blk_idx,size=0.75*chr_sz,/NORMAL
if ffc then xyouts,txt_lgn_x,lgn_y(0),'!8D!5!Inmr!N = '+auto_sng(dmt_nmr*1.0e6,2)+'
!7l!5m',color=clr_blk_idx,size=0.75*chr_sz,/NORMAL
if ffc then xyouts,txt_lgn_x,lgn_y(1),'!7r!5 =
'+auto_sng(psd_gsd_anl,2),color=clr_blk_idx,size=0.75*chr_sz ,/NORMAL
if not ffc then xyouts,txt_lgn_x,lgn_y(0),'!8D!5 = '+auto_sng(dmt_dbg*1.0e6,2)+'
!7l!5m',color=clr_blk_idx,size=0.75*chr_sz,/NORMAL
endif; endif info

if prn then close_ps,fl_nm=fl_nm_out else begin
print,'Hit any key to continue, or q to quit ...'
foo=get_kbrd(1)
if foo eq 'q' then goto,end_of_procedure
endelse; endif not prn
```

--
Charlie Zender zender@uci.edu (949) 824-2987/FAX-3256, Department of
Earth System Science, University of California, Irvine CA 92697-3100

Subject: Re: How to do polar plots with logarithmic axis in radial coordinate?
Posted by Charlie Zender on Thu, 08 Feb 2001 17:34:10 GMT
View Forum Message <> Reply to Message

Thanks for the suggestion, I might steal your idea of using
arc hyperbolic sin instead of log....

Mirko Vukovic wrote:

> In article <3A8159F0.6BCE06BC@ncep.noaa.gov>,
> Paul van Delst <pvandelst@ncep.noaa.gov> wrote:
>
>> Charlie Zender wrote:
>>
>>> Craig Markwardt wrote:
>>>
>>>
>>>> Could you simply take the ALOG10() logarithm of the data before
>>>> plotting it? Easier to re-label the axis than re-invent the
>>>
> world...
>
>>> This would cause the radial coordinate to be negative-valued which
>>> would have unpleasant results. It's possible someone could get
>>> this method to work but I tried without success.
>>
>
> For cases of positive numbers with a huge range, I often use the arc
> hyperbolic sine function. It is approximately linear for arguments<1,
> and logarithmic for large arguments>1. I include it way at the end of
> the post (last two routines). I use it farily often, but never
> bothered to write and accompanying tick marking routine.
>
>
>> Hope some of this is helpful, although I have to admit, the fact that
>
> IDL doesn't have a
>
>> stock polar plotting routine that produces a circular graph with the
>
> radial and concentric
>

>> circle tickmark axes is a bit ridiculous. Farting about with /POLAR
>
> and AXIS and whatnot
>
>> is sort of like using OG to plot, x, y - and in the end you still end
>
> up with
>
>> Cartesian-like axes.
>
>
> agreed. In some of my applications, I use MAP for polar plotting.
> I'll excerpt parts of the code, but you will need to modify it for your
> applications. The code is part of an object, so some variables are
> stored as fields of SELF. But that is all they are, variables. If you
> provide them, they do not have to be parts of an object.
>
> To give you some ideas as to what is involved, the following set-up the
> plot:
> ; convert rmin and rmax (stored as vector in Radial Frame Limits) to
> latitudes
> LatRange=self->r2Lat(RadialFrameLimits)
> ; convert min and max angle (can be 0 to 360) to longitude
> LonRange=self.FrameLimits[[1,3]]*!radeg
> ; store this as part of the object in which the whole things is done
> self.DataLatLonRanges=[LatRange[0],LonRange[0],LatRange[1],L onRange
> [1]]
> ; rotate map so that 0 angle points to the right
> Rotation=-90;+self.Orient*!radeg
> ;; the map is plotted without the default border
> ; put up the polar grid. We can plot data over that, discussed below.
> map_set,90,0,Rotation,/Azimuthal,/iso,/noborder, $
> limit=self.DataLatLonRanges,NoErase=NoErase, $
> _extra=rPropertiesKeywordList
>
> Now this requires two routines for conversion from data to latitudes
> and back:
>
> function Polar_PlotFrame::R2Lat,R
> ;@private
> ;
> ;function that converts radius to latitude. This is used to translate
> ;data into units that MAP understands.
>
>
> Rmax=self.FrameLimits[2]
> LatRange=self.LatRange
>

```
>    RelR=R/Rmax
>    Lat=RelR*(LatRange[1]-LatRange[0])+LatRange[0]
>
>    return,Lat
>  end
>
>  function Polar_PlotFrame::Lat2R,Lat
>  ;@private
>  ; function that converts from latitutde to radius
>
>    Rmax=self.FrameLimits[1]
>    LatRange=self.LatRange
>
>    RelR=(Lat-LatRange[0])/(LatRange[1]-LatRange[0])
>    R=RelR*Rmax
>
>    return,R
>  end
>
>  For these to work, I need this somewhere at the start of the program.
>  Thus the map will have the latitude range from 90 (radius 0) to 0.
>  (max radius, to be determined later)
>
>    self.LatRange = [90.,0.]
>
>  Finally, to plot the data, I do
>  ; convert coords from data to latitude
>     self.oPlotFrame-> AdjustCoords,*self.pIndependentVariable, $
>      *self.pDependentVariable,AngleCoord,RadialCoord
>
>  ; contour will work too!
>     plots,AngleCoord,RadialCoord, $
>      _extra=rPlotPropertiesKeywordList
>
>
>  And, finally, this needs the services of AdjustCoords:
>
>  pro Polar_PlotFrame::AdjustCoords,Phase,Mag,Lon,Lat
>  ; converts coordinates from angle and radius to longitude and
>  ; latitude.
>
>     ;; convert negative magnitude to positive, and correct angle
>     iNegMag =  where(Mag LT 0,cNegMag)
>     CorrPhase = Phase
>     CorrMag = Mag
>     IF cNegMag NE 0 THEN BEGIN
>        CorrPhase[iNegMag] = CorrPhase[iNegMag]+!pi
>     CorrMag[iNegMag] = -CorrMag[iNegMag]
```

```
>      ENDIF
>      CorrPhase = CorrPhase MOD !twopi
>
>      ;; radius to latitude
>      Lat = self->r2lat(CorrMag)
>      Lon = CorrPhase*!radeg
>
>  return
>  END
>
>
>
>
>  Now for the routines for arc sinh.
>
>  Here is the asinh, that can handle scalars and vectors
>  ;+
>  ; return the inverse hyperbolic sine of the argument.  The calculation
>  is
>  ; performed in double precision because of the addition of 1 under the
>  ; square root.  It might be better to test for size and return the
>  ; approximate value of the sqarre root.
>  ;
>  ; Written by Mirko Vukovic, around 1990
>  ;-
>   FUNCTION ASINH,ARG
>
>  ;create the result array
>  type=size(arg)
>  type_res = type
>  dim = type(0)
>  type_res(dim+2) = 32
>  res = m$replicate(type_res)
>  ;fill it in with results
>  index1 = where (abs(arg) lt 1.d3,count)
>  if count ne 0  then  $
>   res(index1) = alog(arg(index1)+sqrt(arg(index1)^2+1.d00))
>  index2 = where(arg le -1.d3,count)
>  if count ne 0 then $
>   res(index2) = -alog(-2.*arg(index2))
>  index3 = where(arg ge 1.d3,count)
>  if count ne 0 then  $
>   res(index3) = alog(2.*arg(index3))
>
>
>  ; bring result to original type of the argument
>  if type(dim+2) ne 32 then res=float(res)
>  return,res
```

```
>
>
> end
>
> It requires mv_replicate, similar to IDL's replicate, but can handle
> scalars (there is probably a better way)
> ;+
> ; NAME:
> ;       MV_REPLICATE
> ;
> ; PURPOSE:
> ;       To emulate the REPLICATE function of the old version of IDL
> ;
> ; CATEGORY:
> ;       Variable massaging
> ;
> ; CALLING SEQUENCE:
> ;       result=MV_REPLICATE(INFO,type=type)
> ;
> ; INPUTS:
> ;       INFO - a vector, of SIZE-like properties
> ;
> ; OPTIONAL INPUT PARAMETERS:
> ;       None
> ;
> ; KEYWORD PARAMETERS:
> ;       TYPE -- (optional) integer assigns the type of the variable.
> If not
> ;       present, the type present in INFO is assigned to the varible.
> ;       The value of type should be
> ;  1: binary
> ;  2: integer
> ;  3: long integer
> ;  4: floating
> ;  5: double precision
> ;  6: complex
> ;
> ;
> ; OUTPUTS:
> ;       RESULT - a variable specified according to INFO
> ;
> ; OPTIONAL OUTPUT PARAMETERS:
> ;       None
> ;
> ; COMMON BLOCKS:
> ;       None
> ;
> ; SIDE EFFECTS:
```

```
>  ;       If INFO does not have the total number of elements in the
>  ;       variable, that is added to it.
>  ;
>  ; RESTRICTIONS:
>  ;       None
>  ;
>  ; PROCEDURE:
>  ;       Straightforward.  Checking is done to see if the total number
>  of
>  ;       elements in the variable is present in INFO.  If not, it is
>  ;       calculated and added to it.
>  ;
>  ; MODIFICATION HISTORY:
>  ;       Written and performed by Mirko Vukovic, sometimes around 1990
>  ;
>  ;-
>  function mv_replicate,info,type=type
>
>  nod = info(0)
>  infod = n_elements(info)
>
>  ; make the INFO array complete
>  if infod ne nod+3 then begin  ; total no. of elemets is
>  missing
>   t=1
>   for i=1,nod do t=t*info(i)
>   info=[info,t]
>  endif
>
>  if info(0) ne 0 then begin   ; this is for an array
>          if keyword_set(type) then res=make_array(size=info,type=type) $
>          else res = make_array(size=info)
>  endif else begin
>   if keyword_set(type) then begin
>   case type of ; and this for a scalar, info(1) has variable type
>  info
>   0:  begin
>    print, 'MV_REPLICATE: cannot make variable of undefined
>  type.'
>    stop
>    end
>   1: res=0b
>   2: res=0
>   3: res=long(0)
>   4: res=0.
>   5: res=0.d00
>   6: res=complex(0.,0.)
>   else:  begin
```

```
>    print, 'MV_REPLICATE: cannot make structure or string
>  variable.'
>    stop
>    end
>   endcase
>   endif else begin
>   case info(1) of ; and this for a scalar, info(1) has variable
>  type info
>   0:  begin
>    print, 'MV_REPLICATE: cannot make variable of undefined
>  type.'
>    stop
>    end
>   1: res=0b
>   2: res=0
>   3: res=long(0)
>   4: res=0.
>   5: res=0.d00
>   6: res=complex(0.,0.)
>   else:  begin
>    print, 'MV_REPLICATE: cannot make structure or string
>  variable.'
>    stop
>    end
>   endcase
>   endelse
>  endelse
>  return,res
>  end
>
>
>  Sent via Deja.com
>  http://www.deja.com/
```

--
Charlie Zender zender@uci.edu (949) 824-2987/FAX-3256, Department of
Earth System Science, University of California, Irvine CA 92697-3100