Programming with DM_Plot Object Graphics

Yiming Qiu(yiming.qiu@nist.gov) 9/2010

Where

The IDL program *dm_plot__define.pro* can be found at DAVE directory *\dave\programs\modules\DCS\dcs_mslice*. All of the other programs that are directly or indirectly used by dm_plot are also included in DAVE.

Command Line Calling

1-D data

Plot (xdat1, ydat1, [yerr1]): xdat1, ydat1, and yerr1 are all 1d arrays

tmp = obj_new('dm_plot', xdat1, ydat1, [yerr=yerr1, color=color, hidelegend=hidelegend, isotropic=isotropic, legend=legend, legdpos=legdpos, linestyle=linestyle, psym=psym, showxgrid=showxgrid, showygrid=showygrid, symsize=symsize, thick=thick, title=title, xlog=xlog, xran=xran, xtit=xtit, ylog=ylog, ytit=ytit, layer=layer, …])

tmp->draw

overplot (xdat2, ydat2, [yerr2])

tmp->add_plot,xdat2, ydat2, [yerr=yerr2, color=color, legend=legend, linestyle=linestyle, psym=psym, symsize=symsize, thick=thick, layer=layer, …]

tmp->draw

psym: a string or a number

0-'no symbol' 1-'plus' 2-'star' 3-'dot' 4-'diamond'
5-'triangle' 6-'square' 7-'cross' 8-'greater' 9-'less'
plus 'circle' and 'bullet', which is a filled circle.

legdpos: a two-element array [xposition, yposition] for the initial legend position, both xposition and yposition are between 0 and 1.

linestyle: a string or a number

0-'solid' 1-'dotted' 2-'dashed' 3-'dash dot'
4-'dash dot dot dot' 5-'long dash' 6-'no line'

color: a string or [r, g, b]

Recognized color strings: 'black', 'white', 'red', 'blue', 'magenta', 'green', 'cyan', 'yellow', 'grey', 'dark grey', 'dark red', 'dark blue', 'dark magenta', 'dark green', 'dark cyan', 'dark yellow'

xlog,ylog,zlog: if set, the corresponding axis is in log scale

2-D data

(xdat, ydat, zdat, [idat]): xdat is a 1d array with nx elements; ydat is a 1d array with ny elements; zdat and idat are 2d arrays with [nx, ny] size.

tmp = obj_new('dm_plot', xdat, ydat, zdat, [idat, colotable=coltable, hideintbar=hideintbar, isotropic=isotropic, iran=iran, ilog=ilog, misscol=misscol, shading=shading, showxgrid=showxgrid, showygrid=showygrid, surfplot=surfplot, title=title, usepolygon=usepolygon,

xlog=xlog, xran=xran, xtit=xtit, ylog=ylog, yran=yran, ytit=ytit, zlog=zlog, zmax=zmax, zmin=zmin, zran=zran, ztit=ztit, …])

tmp->draw

For 2D contour plot, usepolygon keyword is strongly recommended to speed up the plotting. For contour plot, idat is not necessary. For surface plot(with keyword surfplot set), if idat is absent, zdat is used for the color intensity.

3-D data

(xdat, ydat, zdat, idat): xdat is a 1d array with nx elements; ydat is a 1d array with ny elements; zdat is a 1d array with nz elements; idat is a 3d arrays with [nx, ny, nz] size.

tmp = obj_new('dm_plot', xdat, ydat, zdat, idat, [colotable=coltable, hideintbar=hideintbar, interpolate=interpolate, iran=iran, ilog=ilog, misscol=misscol, opacity=opacity,shading=shading, showxgrid=showxgrid, showygrid=showygrid, title=title, volclip=volclip,xlog=xlog, xran=xran, xtit=xtit, ylog=ylog, yran=yran, ytit=ytit, zlog=zlog, zmax=zmax, zmin=zmin, zran=zran, ztit=ztit, …])

tmp->draw

interpolate: 0 or 1

0-nearest neighbor 1-trilinear

opacity: a number between 0 and 1, the same as *alpha_channel* in the IDLgrVolume object.

volclip: a integer between -1 and 7

-1-no clip	0-octant	1-quadrant	2-X-	3-X+
4-Y-	5-Y+	6-Z-	7-Z+	

Widget/Object Programming---Stand-alone Plot Window

1.calling from the parent widget/object

tmp = obj_new('dm_plot', …, group_leader=group_leader, parentobj=parentobj, parenthandler=parenthandler)

group_leader: a widget ID of the parent widget

parentobj: an object instance of the parent object

parenthandler: a string specifying the event handling routine of the parent object. This keyword must be combined with parentobj keyword. Default is "event". The parent object must have an event handler:

obj->parenthandler, event

2. events

keep event:

{plot_keep, ID:tlb, TOP:group_leader, HANDLER:tlb, OBJECT:self}

make current event:

{plot_make_current, ID:tlb, TOP:group_leader, HANDLER:tlb, OBJECT:self}

kill event:

{plot_kill, ID:tlb, TOP:group_leader, HANDLER:tlb, OBJECT:self}

Use the event structure name ‘plot_keep’, ‘plot_make_current’, or ‘plot_kill’ to identify the events.

type = tag_names(event,/structure)

Widget/Object Programming---Compound Widget

1.When used as a compound widget, the menu bar is replaced by a pop-up window when the right mouse button is clicked over the upper left quadrant of the plot window.

2.calling from the parent widget/object

plotWin = obj_new(‘dm_plot’,/compound,widgetbase=widgetbase,
group_leader=group_leader,xsize=xsize,ysize=ysize,…)

widgetbase is the base widget ID where the plot window is to be placed, and group_leader is the top-level-base widget ID. group_leader keyword is optional if you don’t want to have the window resize events. For details about how to set up and handle the window resize events, you can check the test_plot_compound program at the very end of dm_plot__define.pro.

In the event handling routine of the calling program, unhandled events need to be passed to the dm_plot widget by calling

plotWin->event, event

3.useful routines

plotWin->saveas,’ps’	:save the plot as a postscript file
plotWin->saveas,’printer’	:print the plot
plotWin->add_text,textstring,xpos,ypos,fontsize=fontsize,color=color	:add textstring to the plot window at normalized position [xpos,ypos]
plotWin->remove_text	:remove all additional text strings from the plot