5 Arrays

5.1 Array constructors

| brackets:  | pow=2 = [1.,2,4,8];  |
|           | mat=[0.1.][2.,3.];   |
| indgen:   | n=indgen(10) (integers 0 to 9) |
| findgen:  | xx=findgen(10) (floats 0 to 9.) |
| linspace: | x - linspace(-2,2,Nx) (NxD equidistant points from -2 to 2.) |
| spread:   | x=spread(0,5,Ny) xx=spread(x,1,Ny) & yy=spread(y,0,Ny) (replicate data in given direction(s)) |
| make_array: | zz=make_array(Nx,Ny) |
| rebin:    | change size, but not rank (dimensionality) |
| reform:   | change rank, not size; reform(x) removes any degenerate dimensions |

5.2 Array slices

If f is an array of shape [20,17,9], then
- f[*,*] is f
- f[*,0:5,6:*] has shape [20,6,3]
- f[*,0:5,6] has shape [20,6]
- f[*,0:6:*] has shape [20,1,3]
- reform(f[*,0,6:*]) has shape [20,3]
- f[*,0,6] has shape [20]
- f[19,0,6] has shape 1 (is a scalar)

Round brackets: f(1,2,3) is the same as f[1,2,3].

Array syntax is much faster than explicit loops.

5.3 Array inquiries

n_elements(xx) returns total number of elements (or 0 if xx is undefined).

size(xx) returns detailed info:

| scalar:  | [0.,type,1] |
| 1d array: | [1.,Nx, type, n_elements] |
| 2d array: | [2.,Nx,Ny, type, n_elements] |
| 3d array: | [3.,Nx,Ny,Nz, type, n_elements] |

where type is 2 for short integers, 3 for long integers, 4 for floats, 5 for doubles, 6 for complex and 7 for strings.
6 Ploting

6.1 Plotting routines

1-d data: plot, x, f:
   opplot, x, g to plot second curve onto this graph

2-d scalar data: surface, f, x, y:
   shade_surf, f, x, y
   contour, f, x, y
   contourfill, f, x, y

2-d vector data: velovect, vx, vy, x, y
   or (considerable improved)
   w_velovect*, vx, vy, x, y:
   vel, vx, vy, x, y

2-d 3-vector data:
   plot_3d_vect*, vx, vy, vz, x, y
   or simply plot_3d_vect*, v, x, y

6.2 Plotting keywords

X RANGE, Y RANGE: plotting range \([x_{\text{min}}, x_{\text{max}}]\)

X STYLE, Y STYLE: type of axis (1: strict; 3: add 2%)

TITLE, XTITLE, YTITLE: top title and axes titles

XLOG, YLOG: flag (0/1) for (semi-)logarithmic plotting

PSYM: symbol for data points: 0 (none – connect points with line), 1 (+), 2 (•), 3 (□), 4 (○), 5 (∆), 6 (◇), 7 (×), 8 (user defined), 10 (histogram). With PSYM=5 SYM, points are plotted with symbol SYM and connected by line

LINESTYLE: type of line connecting points: 0 (——), 1 (—.), 2 (—-), 3 (-----), 4 (………), 5 (———)

To set a keyword to 1, use it with /KEYWORD. Example:

```idl
plot, x, f, /YLOG, X RANGE=[0.1,10], $ TITLE='Pressure', PSYM=4, LINE=2
```

6.3 Multiple plots + Window management

Set !p.multi = [N_COLS, N_ROWS] to combine \(N_{\text{cols}} \times N_{\text{rows}}\) plots in one window; set !p.multi = 0 for single-plot mode.

```idl
!p.multi=[0,2,3]
for i=0,3 do plot, x, f[i,*]
```

window: create window with a given number:

```idl
window, w
```

wset: switch to given window:

```idl
wset, 0
```

6.4 Hardcopy

Switch to PostScript device and back with

```idl
psa & plot, [0,1] & psa
```

Keywords:

```idl
psa`, FILE='1.ps', /LANDSCAPE, THICK=3 or
```

```idl
psa`, /FULLPAGE, /NOPSFONTS
```

7 System variables

Most graphics keywords have corresponding system variables to set default values. E.g.:

```idl
!p.title = 'Temperature' & !x.range = [0,2]
```

Use help, /struct on !p, !x, !y, !z and !d. The search path for files is given by !path.

7.1 Save/restore graphics state

save_state, restore_state` allow to temporarily modify the graphics state:

```idl
save_state
!p.multi=[0,2,2] & !x.range=[0,2]
for i=0,3 do plot, x, f[i,*]
restore_state
```

```idl
restore_state`, /full reverts fully back.
```

8 Files; running

@file1 includes file file1.pro at cmd line or in script.

```idl
.run run a file: .run file2.pro
```

```idl
.rnew like .run, but clears all variables first
```

```idl
.continue continue after STOP or [cmd-
```

file1 must have no final end; file2 needs one.

9 Help

9.1 help command

```idl
help – info about all variables
```

help, var – variable var

help, NAME='x*', – variables 'x', 'xx', 'x1', etc.

help, /STRUCT – structure variables

help, /RECALL – command line history

help, /KEYS – keyboard settings

help, /DEVICE – graphics device

help, /FUNCTIONS – compiled functions

help, /PROCEDURES – compiled procedures

help, /SOURCE – file names for compiled procedures/functions

9.2 Built-in help tool

Available with '?' (from within IDL) or 'idlhelp' (from the Shell). Indispensable, but strictly suboptimal.

9.3 Online docs

Starting point for PDF manuals: $IDL/docs/onlineguide.pdf. Where $IDL is something like /user/local/rsi/idl/or/opt/idl/idl_actual.

9.4 WWW

http://www.dfanning.com/ (very useful)

news:comp.lang.idl-pvwave (newsgroup)

10 Diverse

10.1 Reading formatted data from a file

data = input_table('list.dat')

col1 = data[0,*] & col2=data[1,*]

10.2 Adding a legend

```idl
esrg.legend`, ['Curve 1', 'Curve 2'], $ LINE=[0,1], SPOS='t1', /BOX
```

SPOS specifies the position (top left in the example).

10.3 Setting default values

To initialise a variable only if it is undefined, do

```idl
default', X, 20
```

if (n_elements(X) le 0) then X = 20

Revision: 1.8 © 2004 Wolfgang Dobler

May be freely distributed under the GNU Free Documentation License.