1 Visualisation using procedure `plot_nao`

1.1 Introduction

A NAO can be visualised using procedure `plot_nao`, which can represent it by:

- xy graphs (1D and 2D NAOs)
- bar-charts and histograms (1D and 2D NAOs)
- color-coded z-images and maps (2D NAOs)
- RGB z-images and maps (3D NAOs with three layers for red, green and blue)
- tiled images (multiple color-coded z-images on a page) (3D NAOs)

If there are additional dimensions then it is possible to generate multiple frames which can be animated. See examples.

The right mouse button displays a menu. The left mouse button saves (x,y,z) values (z-images only) which can be written to a file.

The Tcl code is in the files `plot_nao.tcl` and `plot_nao_procs.tcl`.

1.2 Usage

`plot_nao` *expression ?options?*

1.2.1 Options

Most options can be set using either the above *command-line* or the *options menu*. Command-line options are as follows:

`-barwidth` *float*: (bar chart only) width of bars in x-coordinate units (Default: 1.0)

`-buttonCommand` *script*: executed when button pressed with z-plots (Default: `"lappend Plot_nao::${window_id}::save [set Plot_nao::${window_id}::xyz]"`)

`-colors` *list*: Colors of xy graphs or bars. (Default: `black red green blue yellow orange purple grey aquamarine beige`)

`-columns` *int*: (tiled-plot only) number of columns of tiles on page

`-dash` *list*: Dash patterns of xy-graph lines. (Default: `""` i.e. all full lines) Each element is `""` for full line, `" "` for no line, or standard Tk dash pattern (See entry for *canvas* in Tk manual).

`-discrete` 0 or 1: 1 = Discrete colors between major z tick marks. (Default: 0)

`-filename` *name with extension* `.ps, .gif, .jpeg,` *etc.:* File produced by `-print` (Default: Print rather than writing a file)

`-fill` 0 or 1: 1 = Scale PostScript to fill page. (Default: 0)

`-font_standard` *font*: Main font. (Default: `"courier 10"`)

`-font_title` *font*: Font for title. (Default: `"courier 16"`)

`-gap_height` *int*: height (pixels) of horizontal white gaps (Default: 20)

`-gap_width` *int*: width (pixels) of vertical white gaps (Default: 20)

`-geometry` *string*: If specified then use to create new toplevel window. (E.g. `"-geometry +0+0"` for top left corner)

`-height` *int*: Desired height (screen units). Not used for tiled-plot.

- Type `xy/bar`: Height of whole window (Default: automatic)
- Type `z`: Image height (can be '*min max*' for range) (Default: NAO dim if within limits)

`-key` *int*: width (pixels) of key. No key if 0 or blank. (Default: 30)

`-labels` *list*: Labels of tiles or xy-graphs

`-menu` 0 or 1: 0 = Start with menu bar at top hidden. (Default: 1)

`-orientation P, L` *or* `A:` P = portrait, L = landscape, A = automatic (Default: A)

`-overlay C, L, S, N` or `"E` *expression*`"`: Define overlay. C = coast, L = land, S = sea, N = none, E = *expr* (Default: N)

`-oversize_prompt` 0 or 1: 1 = prompt if image is larger than screen. (Default: 1)

`-ovpal` *expression*: Overlay palette in same form as main palette (Default: `black white red green blue`)

`-palette` *expression*: Main palette defining color map for 2D image. This is matrix with 3 or 4 columns and up to 256 rows. If there are 4 columns then the first gives color indices in range 0 to 255. Values can be whole numbers in range 0 to 255 or fractional values from 0.0 to 1.0. `""` = black-to-white. (Default: blue-to-red)
`-paperheight` *distance*: E.g. `11i` = 11 inch (Default: `297m` = 297 mm (A4))
`-paperwidth` *distance*: E.g. `8.5i` = 8.5 inch (Default: `210m` = 210 mm (A4))
`-parent` *string*: parent window (Default: `""` i.e. create toplevel window)
`-print` 0 or 1: 1 = automatic print/write (for batch processing) (Default: 0)
`-printer` *string*: name (Default: `env(PRINTER)` if defined, else any printer)
`-range` *expression*: defines scaling (Default: auto scaling)
`-rank` 1, 2 or 3: rank of sub-arrays to be displayed (Default: `3 <<< rank`(*data*))
`-scaling` 0 or 1: 0 = Start with scaling widget hidden. (Default: 1)
`-symbols` *list*: Symbol drawn at each point of xy-graph. Can be `plus, square, circle, cross, splus, scross, triangle` or single character (e.g. `"*"`) (Default: `""` i.e. none)
`-title` *string*: title (Default: NAO label (if any) else *expression*
`-type` *string* plot-type (`bar, tile, xy` or `z`)
If rank is 1 then default type is `"xy"`
If rank is 2 and n_rows <= 8 then default type is `"xy"`
If rank is 2 and n_rows > 8 then default type is `"z"`
If rank is 3 then default type is `"z"`
`-width` *int*: Desired width (screen units). Not used for tiled-plot.
- Type `xy/bar`: Width of whole window (Default: automatic)
- Type `z`: Image width (can be '*min max*' for range) (Default: NAO dim if within limits)

`-xaxis` 0 or 1: Draw x-axis? 0 = no, 1 = yes. (Default: 1)
`-xflip` 0 or 1: Flip left-right? 0 = no, 1 = yes. (Default: 0)
`-xlabel` *string*: x-axis label (Default: name of last (final) dimension)
`-xproc` *string*: name of procedure to format x-axis tick values (Default: none)
`-xticks` *expression*: Major tick marks of x-axis (Default: automatic)
`-yaxis` 0 or 1: Draw y-axis? 0 = no, 1 = yes. (Default: 1)
`-yflip 0, 1, ascending or geog`: Flip upside down? (Default: `geog`)
0 = no,
1 = yes,
`ascending` = 'if y ascending',
`geog` = 'if ascending and (y_dim_name = `latitude` or y_unit = `degrees_north` (or equivalent))'
`-ylabel` *string*: y-axis label (Default: name of 2nd-last dimension)
`-yproc` *string*: name of procedure to format y-axis tick values (Default: none)
`-yticks` *expression*: Major tick marks of y-axis (Default: automatic)
`-zlabels` *list*: z-axis labels of values 0, 1, 2, ... (Default: none)
`-zticks` *expression*: Major tick marks of z-axis (Default: automatic)

1.3 Examples

You can cut and paste the following examples into tkcon or wish.

1.3.1 x-y graphs and bar-charts

```
nap "sales = {
    {2 5 1 3 5 7 9 1 2 9 1 0}
    {9 2 5 5 3 9 2 0 8 8 3 8}
}"
nap "month = 1 .. 12"
$sales set coord "" month
$sales set label "Car sales"
plot_nao sales -labels "Joe Mary" -xtick "1..12" -dash {{} .} -symbols "plus cross"
proc format_x x {lindex {{} Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec} $x}
```

```
plot_nao sales -labels "Joe Mary" -xtick "1..12" -type bar -xproc format_x
```

1.3.2 Scattergram

```
nap "x = {1.1 3.2 2.0 5.9 7.7 4.5 6.3}"
nap "y = {5.0 4.1 9.9 3.7 1.2 2.1 4.5}"
$y set coo x
plot_nao y -symbols plus -dash " "
```

1.3.3 Color-coded z-images

These examples define and use the 2D NAO `z`.

```
nap "n = 200"
nap "x = n ... 0.0 .. 10.0"
nap "y = x(-)"
nap "z = sin(x) * sin(reshape(n#y, 2#n))"
$z set coo y x
plot_nao z
plot_nao z -zticks "-1 .. 1 ... 0.2" -discrete 1; # discrete colors
plot_nao "nint(z+1)" -zlabels {{zero (0)} {one (1)} {two}}; # labelled values 0, 1, 2
```

1.3.4 RGB z-images, tiles and animation

These examples define and use the 3D NAO `z3d`. This is defined using `z` defined above.

```
nap "z3d = z /// z*z // z**3 // z**4"
plot_nao z3d; # layer 0 = red, layer 1 = green, layer 2 = blue, layer 3 is ignored
plot_nao z3d -type tile -labels {z {z * z} z**3 z**4} -title "powers of z"
set frames [plot_nao z3d -rank 2]; # create four 2D frames
animate $frames; # animate these frames
```

1.3.5 Printing and writing files

Note that the `-print 1` option can be used to print and write automatically in batch mode operation. It may be necessary to specify '`-geometry +0+0`' to ensure the window is entirely visible.

```
nap "x = 200 ... 0 .. 4p"
nap "y = sin x"
$y set coo x
plot_nao y -print 1; # Print on default printer
plot_nao y -print 1 -geometry +0+0 -filename sin.ps; # write postscript file sin.ps
plot_nao y -print 1 -geometry +0+0 -filename sin.jpeg; # write JPEG image file sin.jpeg
```