Quelques éléments de Yorick (http://yorick.sourceforge.net)

Pour l'essentiel, la syntaxe de Yorick (les expressions, les structures de contrôle, les boucles, les tests, *etc.*) sont semblables au C. Les principales différences sont :

- les variables sont dynamiquement typées et allouées ;
- le résultat d'une opération mettant en jeu des tableaux est un tableau (l'opération étant appliquée élément par élément) ; par exemple :
  Yorick : `a = b + c*d;`
  C : `for (i = 0; i < n; ++i) a[i] = b[i] + c[i]*d[i];`
- les indices des tableaux commencent à 1 et la liste des indices est donnée entre parenthèses ; Yorick : `a(i,j,k)` → C : `a[k-1][j-1][i-1]`

Général

```
#include "fichier.i";              // interprète le contenu de fichier.i
help, fn;                         // aide sur la fonction fn
info, expr;                       // information sur l'expression expr
about, "regex";                   // recherche d'aide
dbexit;                           // sortir du mode debug
x = expr;                         // donner une valeur à une variable
expr;                             // affiche le résultat de l'expression expr
numberof(x)                       // nombre d'éléments dans x
dimsof(x)                         // dimensions de x
a = array(type, dim1, dim2, ...); // création d'un tableau
i = where(test);                  // liste des indices où test est vrai
indgen(n);                        // listes des indices de 1 à n

func f(arg1, arg2, ...)           // création d'une fonction f
{
  ...;
  return expr;
}
```

Syntaxe sur les tableaux

```
x(i)                  // le i-ème élément de x (i = 1 pour le premier)
x(i:j)                // les éléments de i à j de x
x(i:j:s)              // les éléments de i à j par pas s de x
x(0)                  // le dernier élément de x
x(-1)                 // l'avant dernier élément de x
x(*)                  // tous les éléments de x (en un tableau 1-D)
img(i,)               // la i-ème colonne de img (supposé être un tableau 2D)
img(,j)               // la j-ème ligne de img (supposé être un tableau 2D)
img(,avg:11:59:3)     // la moyenne (avg) des lignes 11 à 59 par pas de 3
[]                    // rien ou vide
[1.0, 5.0, 15.0]      // un tableau 1-D de 3 valeurs
[[1,2,3],[4,5,6]]     // un tableau 2-D de dimensions 3 par 2
x(dif,..)             // différence finie le long de la 1ère dimension
x(,dif,..)            // différence finie le long de la 2ème dimension
```

Graphismes

Note : la plupart des commandes graphiques s'appliquent à la fenêtre courante.

```
window, n;                        // sélectionner fenêtre n (la créer si elle
                                  // n'existe pas)
show, img;                        // afficher une image (simple)
show, img, n;                     // afficher une image img dans la fenêtre n
pli, img;                         // afficher l'image img
plg, y;                           // afficher la courbe
plg, y, x;                        // afficher la courbe y(x)
fma;                              // effacer la liste graphique
limits;                           // ré-initialiser les limites
limits, xmin, xmax, ymin, ymax;   // fixer les limites
limits, square=0/1;               // axes x/y de même aspect ratio
logxy, 0/1, 0/1;                  // axes linéaires/logarithmiques
palette, "stern.gp";              // changer de palette de couleur
```

Utilitaires

```
img = fits_read("input.fits");    // lire un fichier FITS
fits_write, "output.fits", img;   // écrire un fichier FITS
img_pad(a, dims, just=1);         // étendre une image a par des zéros
abs2(z);                          // module carré de z
z = fft(x);                       // FFT de x
x = ifft(z);                      // FFT inverse de z (Hermitique)
fft_plg, y;                       // afficher une courbe en géométrie FFT
fft_pli, z;                       // afficher une image en géométrie FFT
u = fft_dist(dimsof(z));          // longueur des fréquences spatiales
conj(z);                          // conjugué de z
help, cg;                         // aide pour les gradients conjugués
histo_plot, a, binsize=0.1, color="red";
                                  // afficher un histogramme des valeurs de a
w = wavelet(img, n);              // décomposition de img en n plans
                                  // d'ondelettes
```