Guide des graphiques sous IDL

Stéphane Erard, IAS octobre 1999

(Notes pour les travaux pratiques de traitement d'images du DEA Astrophysique et techniques spatiales, Paris-7 Paris-11)

Publié dans Micro-Bulletin **70**, 76-110, 1997.

Mises à jour : http://www.ias.fr/cdp/infos/idl/idl.html

Réglage des sorties graphiques

Sélection de la sortie

L'instruction SET_PLOT sélectionne le système graphique utilisé :

```
set_plot, 'X'          ; sélectionne le terminal X courant
set_plot, 'PS'         ; sélectionne une sortie PostScript
```

D'autres pilotes existent pour diverses imprimantes ou systèmes d'écran, en particulier WIN et MAC qui sont des variantes du pilote X. Les librairies externes contiennent des routines initialisant certaines imprimantes, ainsi que des pilotes pour des systèmes graphiques particuliers normalement inconnus d'IDL. Le pilote Z est un écran virtuel conservé en mémoire, normalement utilisé pour les tracés en 3D, qui peut servir de terminal X sans accès physique à un écran (pour faire tourner une routine en batch par exemple) ; attention tout de même : cette sortie ne supporte pas le fenêtrage et bloquera en rencontrant une instruction WINDOW.

La sortie sélectionnée reste active tant qu'on n'en n'a pas définie une autre. En particulier, un fichier PS ouvert à l'intérieur d'une procédure n'est pas fermé lorsqu'on sort de cette procédure, et fermer le fichier ne suffit pas à rebasculer vers le terminal : les graphiques suivants sont écrit dans un nouveau fichier PS, `idl.ps` par défaut ; sous Unix, ceci a pour effet d'écraser le fichier précédent portant ce nom. Il vaut mieux ne pas accéder aux fichiers PS (pour les imprimer notamment) avant d'avoir sélectionné un autre pilote dans IDL (avec par exemple SET_PLOT, 'X').

En général la table de couleurs et le système de coordonnées sont perdus quand on change de sortie au milieu d'une session. Tous les réglages doivent donc être effectués après sélection de la sortie graphique. La table de couleurs peut toutefois être transmise avec :

```
set_plot, 'X', /copy
```

Si la nouvelle sortie a une table de couleurs plus petite que la précédente, seul le début de la table est recopié. Le mot-clef /INTERPOLATE permet de transmettre une table de couleurs cohérente.

Les problèmes de transfert de table de couleur deviennent vite pénibles quand on alterne souvent entre sorties X et PS. La bibliothèque GRAPHICS_DEVICES distribuée

avec ASTRON (dans le répertoire contrib/thompson) permet de gérer plusieurs sorties en parallèle, mais elle est assez délicate d'emploi.

Options des sorties graphiques

L'instruction DEVICE permet de modifier les options correspondant à la sortie choisie. On affiche les options de la sortie active en faisant HELP, /DEV.

On ouvre un fichier PS avec `device, filename='fichier.ps'`
et un fichier PS 256 couleurs avec `device,filename='fichier.ps',/color,bits=8`
Il faut fermer le fichier avant de pouvoir l'utiliser, avec `device, /close`

Cette même instruction permet entre autres d'imposer une police de caractères par défaut et ses attributs, et de définir le positionnement des graphiques dans les fichiers PS.

En sortie X, le contenu des fenêtres est effacé par un recouvrement et doit être rafraîchi périodiquement. Le type de rafraîchissement peut être contrôlé avec l'instruction suivante, également applicable à chaque fenêtre comme argument de l'instruction WINDOW ; l'option 1 est plus rapide mais ne marche pas toujours — elle est gourmande en mémoire :

```
device, retain= 1 ou 2
```

Sur les sorties écran, l'affichage d'une nouvelle image écrase normalement le contenu de la fenêtre (les nouveaux pixels remplacent les anciens). La fonction de recopie vidéo permet de combiner les pixels à afficher aux pixels déjà présents à l'écran :

```
device, set_graphics_function=N
```

où N est un code de 0 à 15 (voir le manuel de référence, p. 3-20 pour la liste des fonctions accessibles). La valeur par défaut est N=3 (remplacement) ; les routines graphiques un peu élaborées utilisent cette fonction par exemple pour superposer un curseur mobile à l'image. Quand une telle routine est interrompue en catastrophe, il peut être nécessaire de rétablir la valeur N=3.

Fenêtres

En sortie X ou équivalentes (Mac, Win…), trois instructions principales permettent de contrôler les fenêtres :

Window, n	Crée (ou recrée) et sélectionne la fenêtre n (n ∈ [0,31], 0 par défaut).

Wset, n Sélectionne comme fenêtre active la fenêtre n déjà ouverte.

Wshow gère l'affichage des fenêtres :

```
Wshow, n              ; affiche la fenêtre n en avant-plan, sans la sélectionner
Wshow, n, 0           ; masque la fenêtre n
Wshow, n, /iconic     ; la réduit en icône
Wshow, n, iconic=0    ; la réaffiche
```

Parmi les autres fonctions utiles :

Slide_image permet d'ouvrir une fenêtre à défilement pour les images plus grandes que l'écran. Ajouter le mot-clef CONGRID=0 accélère l'exécution.

Erase Efface le contenu d'une fenêtre. Automatique avant l'affichage d'un graphique, mais pas d'une image. On peut préciser l'indice de la couleur avec laquelle on remplit l'écran (couleur de fond, 0, par défaut). En PS, ERASE fait avancer d'une page.

Device, copy=[X0,Y0,dx,dy,X,Y,n] Recopie dans la fenêtre courante, à la position (X,Y), une zone de la fenêtre n commençant en (X0,Y0) d'étendue (dx,dy).

Une nouvelle fenêtre est ouverte avec des caractéristiques par défaut dont certaines sont réglées par l'appel de DEVICE. On peut affecter des caractéristiques différentes à une nouvelle fenêtre avec les arguments de WINDOW :

color=200 Fixe le nombre de couleurs disponibles. Cette option doit être donnée à la création de la première fenêtre d'une session, *elle n'est pas modifiable ensuite*. Si on ne l'utilise pas, on voit le nombre de couleurs varier d'une session à l'autre sans cause apparente (il dépend de l'utilisation du système et des autres applications actives).

xsize=x, ysize=y Fixent les dimensions de la fenêtre en pixels.

/free IDL ouvre une nouvelle fenêtre non utilisée (sinon, il recrée la fenêtre active ou ouvre la fenêtre 0 par défaut). Le numéro de cette fenêtre est disponible dans la variable !D.window.

/pixmap Indique une fenêtre « virtuelle » stockée en mémoire, qui n'est pas affichée immédiatement à l'écran (DEVICE, COPY permet de recopier une zone de cette fenêtre dans la fenêtre active).

Wdelete, n1, n2… Ferme une ou plusieurs fenêtres. C'est la seule façon de fermer une fenêtre pixmap, et il est important de le faire pour économiser la mémoire.

Écriture de fichiers PostScript

Il y a deux types de fichiers PostScript, normal ou encapsulé (EPS). Les fichiers PostScript normaux sont destinés à l'impression et peuvent contenir plusieurs pages. Ils contiennent des instructions de dimensionnement modifiables à l'ouverture du fichier avec les mots-clefs de DEVICE :

orientation=portrait Impose le format français (feuille debout, défaut).

orientation=landscape Impose le format italien (feuille couchée).

Xsize, Ysize Taille de la région d'affichage (en cm par défaut). Utilisables pour réduire les graphiques ou en changer l'aspect, ces paramètres sont indispensables pour afficher des images (voir plus loin).

Xoffset, Yoffset Règlent la position de la région d'affichage sur la feuille. Ces décalages sont calculés à partir du coin inférieur gauche du format français (XOFFSET correspond toujours à la petite dimension de la feuille, même en format italien).

La résolution du pilote PS est fixée à 1000 points par centimètre, mais les pixels sont redimensionnables (voir la section sur les images). En mode graphique on n'a normalement pas à s'en soucier, à ceci près que les traits sont peu épais et conviennent particulièrement mal à des transparents (jouer sur les mots-clefs X- Y- ZTHICK pour les axes, et sur le mot-clef THICK ou la variable !P.THICK pour les lignes).

Les fichiers encapsulés EPS sont destinés à être insérés dans des documents (traitements de texte…). Ils ne peuvent contenir qu'une seule page graphique, et pas d'indication de positionnement (XOFFSET et YOFFSET). Le redimensionnement est inutile pour les graphiques (ils occupent tout l'espace disponible comme sur écran), mais s'impose généralement pour les images (voir plus loin).

```
Device, filename='pinocchio.ps', /encapsulated, /preview
```

Un fichier EPS peut être importé dans Latex (avec la macro `idlplot_dvips.tex`), dans Word (Insère fichier… ou Insère image…) ou dans n'importe quel traitement de texte. Sur micro-ordinateur ces fichiers ne sont pas visualisables directement : seul l'espace réservé à la figure apparaît avec un commentaire. On peut les redimensionner dans Word ou ClarisWorks en cliquant le coin inférieur droit de cet espace avec la touche Majuscules enfoncée ; les deux dimensions varient alors en proportion (autrement, l'image est redécoupée). Il y a moyen de créer une image de qualité réduite associée au fichier, qui sera affichée à l'emplacement réservé dans un traitement de texte, en utilisant l'option /PREVIEW (n'existe pas dans la version VMS) ; cette prévisu est au format EPSI, qui n'est évidemment pas utilisé sur micro. Sur Mac, il suffit d'ouvrir le fichier EPS avec l'application MacGS (ou PS2EPS+), d'enregistrer et de fermer le fichier : celui-ci est enregistré avec un type EPS, et une prévisu au format PICT est ajoutée en ressource (non transmissible sur d'autres systèmes). Les traitements de texte affichent la prévisu PICT à l'écran ; lors de l'impression, c'est la partie EPS qui est transmise à l'imprimante. MacGS et PS2EPS+ permettent par ailleurs de reformater l'image et notamment de changer son orientation. Une application similaire existe sur PC.

Polices de caractères

Sur chaque système graphique IDL peut utiliser soit le jeux de polices spécifique du système, soit un jeu de polices vectorielles standard (Hershey, utilisé par défaut). Ces polices sont moins précises mais mieux adaptées aux graphiques (elles supportent mieux les affichages obliques). Chaque jeu possède une grande variété de polices avec différents attributs. En dehors des affichages obliques ou 3D, on a souvent intérêt à utiliser les polices de caractères système plutôt que les polices vectorielles d'IDL. On les sélectionne en faisant :

```
!P.font = 0              (-1 pour rétablir les polices vectorielles) ou
Plot, X, Y, font=0       pour une modification temporaire
```

On peut imposer une police par défaut et ses attributs (graisse, corps, chasse, romain ou italique…) dans DEVICE à l'ouverture d'un fichier PS. On change de police ou de niveau d'indices à l'intérieur d'une chaîne grâce à des séquences introduites par ! (pour le PostScript voir p. 3-37 du manuel de référence de la version 4 ; pour les polices vectorielles, voir le chapitre 9 du manuel utilisateur). À noter que ces séquences restent en action tant qu'elles ne sont pas explicitement inhibées (ça peut causer quelques surprises dans les affichages des axes de graphiques).

En sortie PostScript, ce sont des polices PS réduites qui tiennent lieux de polices système : les caractères accentués ne sont pas accessibles par défaut, mais seulement en utilisant les polices PS internationales avec le mot-clef /ISOLATIN1 dans DEVICE. Les caractères prétendus spéciaux (symboles et accentués) sont évidemment codés de façon différente en IsoLatin et dans les polices vectorielles ; il faut donc utiliser des variables contenant les codes correspondants pour avoir un affichage ambivalent écran-PS, ou utiliser systématiquement à l'écran des polices système qui suivent la norme IsoLatin1 — c'est généralement le cas des polices X.

Par contre, certains systèmes X-window (les machines HP en particulier) n'utilisent apparemment pas le codage IsoLatin dans les éditeurs de texte, et taper `xyouts, 'é'` dans un programme peut donner des résultats surprenants à l'affichage comme à l'impression (c'est également vrai, mais moins imprévisible, sur les Mac et les PC). Si on tient aux accents et aux symboles, il faut utiliser les codes numériques des caractères (faire `xyouts, string(233B)` dans l'exemple ci-dessus). Les polices et leurs codes sont visualisables de la façon suivante :

```
SHOWFONT, n, 'titre'        ; affiche la police vectorielle de type n
EFONT                       ; affiche les polices vectorielles proprement
a=XFONT()                   ; visualise les polices X
```

Les codes IsoLatin ne sont pas facilement accessibles sous IDL, mais disponibles sur tout bon micro-ordinateur (la fonction PS_SHOW_FONTS crée un fichier des polices PS de 70 pages, plutôt encombrant et un tantinet répétitif).

Procédures et fonctions sur les images

On se souviendra utilement que la moindre opération est environ dix fois plus rapide sur des entiers que sur des réels. Les images sont destinées à toujours finir sous forme entière et sont par définition de gros tableaux ; sauf à enchaîner de nombreuses opérations, on a donc toujours intérêt à travailler sur des variables entières en cas de ralentissement (le type le plus rapide est curieusement l'entier court, pas le byte).

Affichage et lecture à l'écran

Les routines intégrées d'IDL transmettent des tableaux sur la sortie graphique active ou relisent les informations affichées dans une fenêtre. Sur une sortie X, chaque

élément de tableau est affiché sur un pixel (c'est différent en PostScript, voir plus loin). Les sorties graphiques fonctionnent toujours sur 8 bits (avec trois tableaux pour les sorties 24 bits), on a donc intérêt à stocker les images dans des tableaux d'octets — en tous cas à éviter les réels. La discussion qui suit suppose que le table de couleurs active est sur 256 niveaux (voir plus loin pour les sorties 24 bits).

TV, image — Recopie le tableau Image sur la sortie active. Les valeurs du tableau sont directement traduites en indices de couleur, et la table de couleurs est réutilisée cycliquement (Image doit donc en principe être un tableau d'octets). Par défaut le tableau est copié à l'envers, c'est à dire avec la première ligne en bas de l'écran (ce qui permet de manipuler les images de façon naturelle). On inverse ce défaut en mettant la variable système !ORDER à 1. Sur une sortie 24 bits, on peut spécifier le plan couleur sur lequel on affiche.

TV, image, Pos — Avec pos=0, affiche l'image en haut à gauche de la fenêtre. La position 1 est immédiatement à droite… on se décale d'une ligne vers le bas quand il n'y a plus la place d'afficher une image complète. Permet de juxtaposer facilement des séries d'images de même dimension. En sortie PostScript, cette fonction ne marche qu'à condition de préciser la taille des images individuelles (ajouter xsize = Xnbplot*!d.x_size et la même chose en y) : elles se recouvrent sinon.

TV, image, x0, y0 — Copie le tableau à partir de la position (x0,y0), par défaut en pixels, qu'on peut spécifier dans différents systèmes de coordonnées (ou en centimètres) en utilisant les mots-clefs adéquats.

TVSCL, image — Similaire à TV, mais prend la peine d'ajuster le maximum du tableau au nombre de couleurs disponibles. C'est surtout utile quand on ne sait pas bien ce que contient l'image et pour afficher des images codées sur plus de valeurs qu'il n'y a de couleurs disponibles, ou dont la dynamique est faible. Pour conserver un contrôle précis des valeurs affichées, il vaut mieux utiliser TV et forcer les bornes de l'échelle de couleur à des valeurs connues (voir le paragraphe sur les tables de couleur).

Device, copy=[x,y,xs,ys,x0,y0,fen] — Recopie dans la fenêtre courante la zone (x:x+xs-1,y:y+ys-1) de la fenêtre pixmap FEN (ouverte par WINDOW, fen, /pixmap) à partir de la position (x0,y0) de la

fenêtre active. Cette copie est instantanée, c'est la bonne façon de créer des animations rapides (voir aussi la fonction XANIMATE).

tab=TVRD(x0, y0, Nx, Ny) Lit la zone (x0:x0+Nx-1,y0:y0+Ny-1) de l'écran et la stocke dans tab. Permet de récupérer facilement une image modifiée à l'écran.

Rdpix,ima, x0,y0 Lit IMA en balayant la fenêtre où elle est affichée, et donne sur la fenêtre de commande IDL la position du pointeur (x,y) et la valeur du tableau en (x,y). Si l'image n'est pas affichée à partir de l'origine de la fenêtre, il faut spécifier les coordonnées (x0,y0) du coin bas-gauche de l'image pour avoir une bonne correspondance entre le tableau et la position du curseur.

Curval, ima Similaire à Rdpix mais affiche directement les coordonnées astro (α, δ) si on fournit un en-tête FITS (dans ASTRON).

TVlist, image Affiche la valeur des pixels d'une boite réglable centrée sur le curseur (dans ASTRON).

Profiles,ima, x0,y0 Lit IMA en balayant la fenêtre où elle est affichée, et affiche dans une fenêtre séparée des profils en ligne ou en colonne. Il faut spécifier la position d'affichage (x0,y0) de ima sur la fenêtre, comme pour RDPIX.

Zoom, coef Affiche un zoom de la fenêtre active dans un autre fenêtre. On sélectionne la zone à agrandir avec le pointeur ; le coefficient d'agrandissement est modifiable. Marche seulement avec des écrans 8-bits. La fonction ZOOM_24 est équivalente pour des écrans 24-bits.

TVCRS, x0,y0 Place le curseur en (x0,y0).

Cursor, X, Y,Wait Retourne dans (x,y) la position du pointeur dans différents systèmes de coordonnées. L'argument WAIT force à attendre qu'un bouton de la souris soit pressé (utilisé pour des procédures interactives).

Curfull, X, Y Identique, mais utilise un curseur plein écran. Dans la bibliothèque ASTRON.

Rddat Boucle sur la fonction CURSOR, c'est l'équivalent de RDPIX pour un graphique ou une carte (dans la biliothèque Lowell).

Box_cursor Sélecteur rectangulaire qu'on déplace à la souris. Retourne la position et la taille du rectangle dans les paramètres.

Sélection de pixels et de régions d'intérêt

Les fonctions suivantes permettent de définir des zones particulières à l'intérieur d'un tableau ou d'une image :

Where(ima gt seuil) Retourne la liste des éléments de ima (indicés selon une seule dimension) qui vérifient cette condition. Les valeurs elles-mêmes sont ima(where(ima gt seuil)). L'argument de WHERE peut être une fraction d'un tableau plus grand (par exemple where(ima(*,3:5) lt max)), mais le résultat ne peut être réutiliser que pour indexer le même morceau de tableau. Le résultat vaut -1 si aucun élément ne satisfait à la condition, il faut donc le tester avant de le réutiliser pour indicer le tableau :

```
ind=where(tab ne 0)
if ind(0) ne (-1) then tab(ind) = 1./tab(ind)
```

ou :

```
ind=where(tab ne 0, count)
if count ne 0 then tab(ind) = 1./tab(ind)
```

Polyfillv(X,Y,Nx,Ny) Retourne la liste des éléments d'un tableau Nx × Ny contenus à l'intérieur d'un polygone dont les sommets ont pour coordonnées X et Y.

Label_region(image) Identifie les régions connexes non nulles de Image et leur affecte un indice. Cette fonction permet de reconnaître des objets séparés sur un fond sombre — si celui-ci est fluctuant comme un fond de ciel, il faut commencer par seuiller l'image. Quand l'image ne contient qu'un seul objet, on récupère plus simplement les pixels intéressants avec Where(image gt seuil).

Profile(ima,x,y, xstart=x0, ystart=y0) Retourne un profil entre deux points cliqués à la souris ; les coordonnées des points extraits sont dans x et y. (x0,y0) sont les coordonnées écran du coin bas-gauche de l'image.

Defroi(Nx,Ny,x0,y0) Définit une région polygonale sur l'image Nx × Ny affichée à la position (x0,y0). La région est définie interactivement en cliquant ses sommets à la souris, et coloriée en sortant de la fonction. La fonction retourne la liste des pixels qui appartiennent à la région définie (comme pour POLYFILLV) ; on peut inhiber ce calcul et ne récupérer que les coordonnées des sommets avec le mot-clef /NOREGION.

Extrac(A,C1,S1…) Extrait un sous-tableau de longueur S1 à partir de C1 dans la première dimension, etc… C'est plus lent que l'adressage direct du tableau, mais les zones situées hors du tableau sont mises à 0.

Superposition d'images

Ces fonctions permettent de recaler des images comparables (poses télescopiques successives, poses à travers différents filtres, imagerie spectrale…) :

Flick, A, B, V Affiche alternativement dans la même fenêtre les tableaux A et B, à la vitesse V (approximative). Permet de comparer deux images.

Blink, Fen,V Affiche alternativement dans la fenêtre active le contenu des fenêtres dont les indices sont dans le tableau FEN (fonction de la bibliothèque ASTRON).

Correl_Optimize Calcule les corrélations de deux images pour différentes valeurs de décalage en pixels, et renvoie le décalage optimal. Utilise deux routines de bas niveau qui peuvent être intéressantes directement : Correl_Images et Corrmat_Analyze (bibliothèque ASTRON).

Statistiques élémentaires

Sans rentrer dans le détail des fonctions statistiques, voici le minimum indispensable pour des images. Ces fonctions sont beaucoup plus stables à partir de la version 5 :

Min et Max(ima,n) Renvoient les valeurs extrêmes d'un tableau, et l'indice de l'élément correspondant dans n. Les mots-clefs MAX et MIN permettent de ne trier qu'une seule fois le tableau pour récupérer les

deux valeurs (faire : mini = min(ima,max=maxi)). La fonction MINMAX de ASTRON est semblable.

Moment(ima) Renvoie les quatre premiers moments. On peut également récupérer l'écart moyen absolu et l'écart-type dans des variables avec les mots-clefs MDEV et SDEV. Attention, en version 4 il arrive que cette fonction produise des résultats erronés quand on l'applique à des sous-tableaux. La fonction STDEV des versions 3 (dans le répertoire « Obsolete » de la version 4.0) était plus stable.

Stddev(ima) Calcule correctement l'écart-type (et VARIANCE calcule la variance).

Mve, ima Affiche les principaux paramètres de l'image (dans la bibliothèque ESRG_UCSB).

Avg(ima) et Sigma(ima) Calculent la moyenne et l'écart-type d'un tableau, éventuellement par ligne ou par colonne (dans ASTRON).

PCA Est une routine de calcul d'analyse en composantes principales (destinée à des tableaux de données), qui peut s'utiliser sur des séries d'images spectrales recalées spatialement (dans ASTRON). La version 5 d'IDL contient une routine PCOMP similaire. Ni l'une ni l'autre ne tiennent compte du bruit sur les données. Avec un jeu de données très corrélées l'ACP est peu informative ; on a donc intérêt à travailler sur des rapports d'images pour évacuer les variations de luminosité.

CLUSTER Classification par clusters d'un tableau de données, à partir de la version 4.

Transformations géométriques des images

Les fonctions suivantes opèrent des transformations sur les matrices qui contiennent les images. Certaines sont en fait des fonctions mathématiques utilisables sur des matrices quelconques.

Shift(tab,S1,S2) Décale les colonnes de S1 positions et les lignes de S2 positions. Le décalage est circulaire (pas de perte d'information sur les bords, mais ce peut être un problème pour des images si le fond n'est pas uniforme). S'il n'y a qu'un argument précisé, le décalage opère sur

les indices en série du tableau (la fin d'une ligne se retrouve au début de la suivante). Pour recentrer une série d'images télescopiques sur un objet, il vaut mieux découper la partie utile si on en connaît à peu près les limites : on réduit ainsi l'espace mémoire occupé, et on s'évite des problèmes sur les bords.

Transpose(tab) Symétrie du tableau selon la diagonale haut-gauche/bas-droite. L'image est affichée par TV avec la première ligne en bas, donc l'image apparaît renversée par rapport à l'autre diagonale…

Reverse(tab, N) Inverse tab selon la dimension N (1 ou rien pour les lignes, 2 pour les colonnes, et 3 pour la troisième dimension).

Rotate(tab, N) Combine les deux précédentes : retourne tab par pas de 90° (N=0 à 3), et transpose en plus pour N=4 à 7. Le résultat est immédiat, c'est la méthode normale pour redresser une image inversée.

Rot(tab,Angle,coef,x0,y0) Tourne tab d'un angle quelconque autour de (x0,y0) — par défaut autour du centre du tableau. Le point (x0,y0) est recentré, sauf si le mot-clef /PIVOT est utilisé. Attention, la rotation du tableau s'effectue dans le sens direct, mais l'image affichée est tournée dans le sens rétrograde ! Cette fonction est rapide si on l'utilise telle quelle (interpolation au plus proche voisin), mais le résultat peut être sévèrement dégradé. Les mots-clefs /INTERP et /CUBIC forcent l'utilisation de méthodes d'interpolation plus satisfaisantes, mais nettement plus longues. On peut appliquer simultanément un coefficient d'échelle, mais le résultat est un tableau de même taille que l'entrée, et il y a donc facilement perte d'information sur les bords (cette fonction est utile pour compenser de légères dilatations dans une série d'images, notamment entre différents filtres). Par défaut, les pixels absents du tableau original prennent les valeurs des pixels du bord le plus proche (c'est à dire que les bordures sont recopiées telles quelles pour combler les trous) ; le mot-clef MISSING permet d'attribuer une valeur constante à ces pixels.

Rebin(tab,Nx,Ny) Réduit ou dilate tab d'un facteur entier dans chaque dimension, indépendamment (Nx et Ny doivent être des multiples ou sous-multiples entiers des dimensions originales). Par défaut les nouvelles valeurs sont interpolées entre les anciennes. Avec le mot-

clef /SAMPLE, la routine se contente d'agrandir le tableau (interpolation au plus proche voisin, plutôt que linéaire). C'est la méthode normale pour agrandir une image trop petite. Dans les cas très délicats où l'on veut garder une photométrie exacte, on peut utiliser la fonction BOXAVE(tab, dX,dY) d'ASTRON qui travaille entièrement en réels, et est donc beaucoup plus lente (dX et dY sont alors les dimensions des zones à moyenner).

Congrid(tab,Nx,Ny) Ré-échantillonne tab aux dimensions Nx, Ny quelconques (fonctionne également en 1D et 3D). Par défaut le tableau est seulement agrandi (interpolation au plus proche voisin). On force l'interpolation bilinéaire ou cubique avec /INTERP et /CUBIC. Cette routine est beaucoup plus longue que REBIN, surtout si on interpole. S'il s'agit d'ajuster une image à la taille d'un graphique, il vaut nettement mieux modifier les dimensions du graphique que celles de l'image. Utiliser REBIN avec un coefficient entier si la taille précise du résultat est indifférente.

Expand, tab,Nx,Ny, Resultat Une alternative à CONGRID, apparemment plus régulière (interpolation bilinéaire seulement, ne travaille que sur des tableaux). On peut écranter à une valeur maximum.

Poly_2D(tab,P,Q) Transformation polynomiale d'une image (les coordonnées x et y des points sont transformées selon des polynômes dont les coefficients sont dans les tableaux P et Q respectivement). Cette fonction permet un grand nombre de transformations (décalages, dilatations…), elle est notamment utile pour modifier les coordonnées d'une image déformée, en utilisant des points de référence (assez difficile à utiliser pour ça tout de même). On peut ou non modifier les dimensions du tableau final, et utiliser différents types d'interpolation.

Warp_tri(X0,Y0,X1,Y1,Image) La bonne solution pour faire des corrections géométriques à partir de points de référence : les positions dans l'image (X1,Y1) sont recalées sur les positions connues (X0,Y0).

Les tableaux de données incomplet doivent fréquemment être interpolés sur une grille régulière avant affichage, de façon à reconstituer une pseudo-image. C'est en particulier le cas quand on doit calculer des isolignes à partir d'un tableau de mesures spatiales échantillonnées de manière aléatoire. Ce maillage est effectué en deux temps :

```
Triangulate, X, Y, TR, Lim   ; retourne dans les tableaux TR et Lim la structure du
                             ; réseau de points défini par les coordonnées X et Y.
gs = [(max(X)-min(X))/50.,(max(Y)-min(Y))/50.]  ; Définit les échelles en X et
                                                ; Y (le coefficient 50 est obligatoire).
Z1=Trigrid(X,Y,Z,TR,gs,extra=Lim) ; Interpole Z. EXTRA permet de traiter les bords
                                  ;du tableau proprement.
Contour, Z1, zrange=[min(Z),max(Z)]    ; Affiche des contours réguliers.
```

Un système de maillage sphérique utilisant ces fonctions est disponible depuis la version 4.0. La routine SPH_SCAT a des capacités semblables, qui servent notamment à l'interpolation en symétrie sphérique.

Fonctions de traitement d'images

Les fonctions suivantes servent au filtrage ou la mise en évidence de structures dans des images : elles requièrent en général des arguments à deux dimensions.

a=Ima GT 25 — Seuillage (renvoie un tableau binaire). Utiliser les opérateurs logiques pour seuiller une image, pas > et < qui sont des troncatures.

Median(Ima, L) — Filtrage médian sur un voisinage de largeur L. Réduit le bruit additif sans perte de détails. Attention, par défaut cette fonction et la suivante ne traitent pas les bords sur une largeur L/2 ; à partir de la version 4, le mot-clef /EDGE_TRUNCATE permet de traiter également les bords, ce qui est indispensable dans les processus itératifs.

Smooth(Ima, L) — Lissage par moyenne mobile, supprime les hautes fréquences. Ima - Smooth(ima,5, /edge_truncate) / 1.2 fait un masque flou très convenable (rehausse les contrastes locaux, mais introduit des artefacts là où les gradients sont importants).

Filter_image(ima,Smooth=L) — Effectue des séquences de lissages médians ou moyens en traitant proprement les bords des images avec l'option /ALL. Sait également convoluer par une gaussienne de demi-largeurs quelconques (dans ASTRON).

LeeFilt(Ima, N, sig) — Filtrage de Lee, en principe moins violent que la moyenne mobile : ajoute une image lissée sur un intervalle 2N+1, avec un poids dépendant de sig (le résultat a la même moyenne que l'image).

Sigrange(ima,fraction=f) Renvoie une version du tableau IMA où les valeurs sont forcées à l'intérieur d'un intervalle contenant une fraction f des valeurs d'origine (filtre les valeurs extrêmes). Utile en particulier pour l'affichage, ou pour déterminer rapidement les valeurs extrêmes réelles dans une image. Dans ASTRON.

Sigma_filter(Ima,N,Nsig) Supprime les valeurs aberrantes : remplace chaque point par la moyenne de son voisinage de largeur N s'il s'en écarte de plus de Nsig × écart-type.

Convol(Ima,Noy,Ech) Convolue une image par un noyau et permet le filtrage linéaire. Par défaut le noyau est centré sur le point courant de l'image (fonction de filtrage ; faire CENTER=0 pour convoluer au sens mathématique). Le résultat est divisé par le troisième argument au cours du calcul ; prendre Ech=TOTAL(Noy) si Ima est un tableau d'octets pour ne pas saturer et éviter les problèmes d'arrondis (on ne peut pas le faire après exécution de la routine). Le résultat a les dimensions de Ima, mais les bords ont une valeur nulle sur une bande de demi-largeur du noyau. Les mots-clefs EDGE_WRAP et EDGE_TRUNCATE permettent de calculer les bords de l'image soit en repliant celle-ci, soit en répliquant les lignes et colonnes externes ; le résultat est alors comparable à l'image d'entrée sur les bords (c'est indispensable dans les processus itératifs qui finissent sinon par grignoter toute l'image, mais ça n'est disponible qu'à partir de la version 4).

Convolve(Ima,noy) Convolution par multiplication des transformées de Fourier. Cet algorithme est plus rapide pour les noyaux de taille supérieure à environ 25×25, à condition que les deux matrices aient des dimensions égales à une puissance de 2. Le mot-clef /CORRELATE permet de calculer la corrélation de deux images. Les mots-clefs FT_PSF et FT_IMAGE permettent de passer les transformées du noyau et de l'image lors d'un appel successif à la routine, ce qui permet de gagner beaucoup de temps dans les processus itératifs ou les traitements en série (dans la bibliothèque ASTRON).

Dilate, Erode Suppriment respectivement les trous et les structures isolées dans des images binaires. Utilisés comme opérations élémentaires dans des traitements plus complexes.

Thin — Renvoie le squelette de l'image (amincit les traits). A pratiquer sur une image seuillée.

Roberts, Sobel — Deux algorithmes de détection de bords légèrement différents (gradients isotropes par convolution avec des noyaux différents).

Max_entropy — Déconvolution itérative d'une image par une fonction d'étalement (PSF). Dans ASTRON, qui contient plusieurs routines permettant d'estimer la PFS depuis l'image (GETPSF…), ou d'en fabriquer une (PSF_GAUSSIAN). Cette fonction et la suivante s'utilisent à l'intérieur d'une boucle, en contrôlant à la main les critères de convergence.

Max_likelihood — Déconvolution itérative d'une image bruitée par une PSF (algorithme de Richardson-Lucy). Dans ASTRON.

Analyse d'images CCD et d'imagerie spectrale

Outre les bibliothèques spécifiques d'un instrument, il existe plusieurs systèmes de routines générales, qu'on peut utiliser par exemple pour analyser en temps réel des images télescopiques.

ASTRON — La bibliothèque de la Nasa contient réellement tout ce qui est nécessaire à l'analyse et au pré-traitement des images CCD, en modules séparés. Elle n'offre pas d'interface globale de traitement.

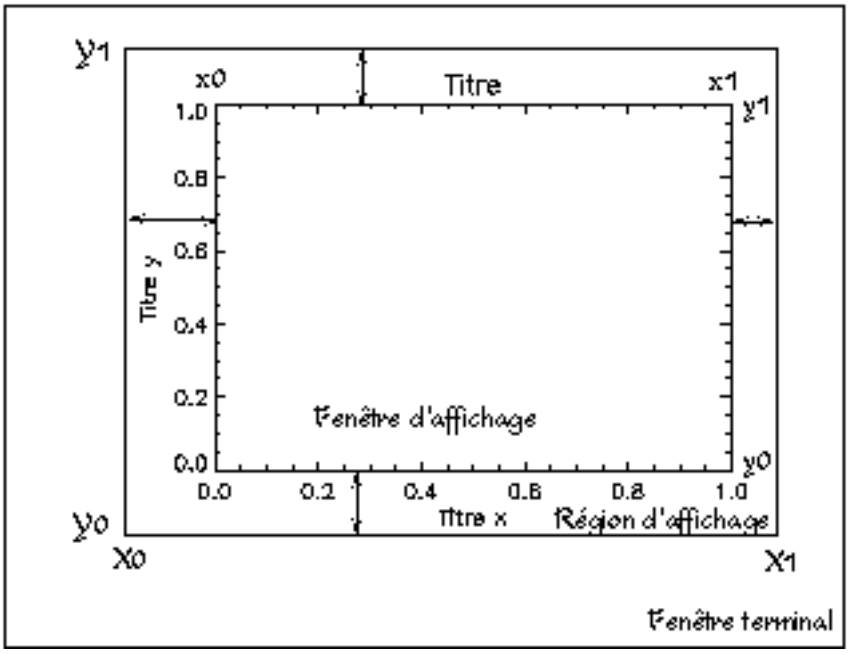
Look, ima — Module d'analyse d'image simple, mais évolutif. Le tableau ima contient l'image préalablement chargée, ou plusieurs images empilées dans la troisième dimension. On peut ajouter des routines quelconques au menu lors de l'appel (routine utilisateur distribuée avec ASTRON).

LOWELL — Bibliothèque de l'observatoire du même nom, incluant des modules de réduction d'images et de photométrie planétaire. Elle contient des routines utiles notamment pour le calcul automatique des fonds et des PLU et pour la correction des rayons cosmiques – mais l'ensemble des traitements est assez spécifique. Elle est incompatible avec ASTRON à cause de nombreuses homonymies.

CCDPhot	Traitement d'images CCD. C'est l'interface finale de la bibliothèque Lowell. Il est assez difficile d'emploi et utilise apparemment des formats particuliers. Le module ITOOL peut être utilisé directement pour l'analyse d'images.
SIPS	Module de traitement d'images spectrale de bonne qualité, facilement utilisable pour des images télescopiques composites (gratuit, mais distribué sous licence par l'Université du Colorado). Écrite pour IDL 3.6, elle fonctionne en versions 4 et 5 à condition de transférer le module Xmenu du répertoire "Obsolete" vers une répertoire accessible.
ENVI	Module d'imagerie spectrale sophistiqué, dérivé de SIPS. Il existe une version réduite (FREELOOK) qui ne permet que de visualiser des images et d'extraire des spectres (logiciel commercial).

Systèmes de coordonnées graphiques

IDL utilise simultanément trois systèmes de coordonnées : écran, normalisées, utilisateur (device, normal et data), mesurées respectivement en pixels, de 0.0 à 1.0 et dans l'unité des données affichées. Certaines distances sont par ailleurs gérées en taille de caractères courants ou en cm. IDL est capable de gérer des graphiques en manipulant des objets à trois dimensions ; ces fonctions ne sont pas détaillées ici.



Les fenêtres sont divisées en deux zones : la « fenêtre d'affichage » (zone délimitée par les axes) et la « région d'affichage » (qui contient les axes et la place réservées aux titres ; elle ne correspond pas nécessairement à la fenêtre écran…). Pour un fichier PS, la feuille elle-même tient le rôle de la fenêtre, et seule une partie en est exploitée.

Variables systèmes pour les graphiques

Les variables systèmes sont des structures dont le nom commence par !. L'ensemble des paramètres d'affichage et de gestion des sorties graphiques est contenu dans certaines de ces variables : d'une part les valeurs par défaut des quantités modifiables par mots-clefs dans les routines d'affichage, d'autre part les dimensions des zones d'affichage et les transformations entre systèmes de coordonnées. La plupart ne

sont pas modifiables directement, et peuvent seulement être lues. Celles qui sont modifiables peuvent retourner à leur valeurs par défaut en leur affectant la valeur 0. Plus généralement, la procédure CLEANPLOT d'ASTRON réinitialise les variables graphiques, et indique les changements effectués. Les variables systèmes les plus utiles sont :

!D.x_size et !D.y_size Dimensions de la fenêtre en pixels.

!D.x_vsize et !D.y_vsize Dimensions de la partie visible de la fenêtre en pixels (intersection avec l'écran)

!D.n_colors Nombre de couleurs disponibles (déterminé à l'ouverture de la première fenêtre de la session).

!D.x_ch_size et !D.y_ch_size Largeur et hauteur des caractères normaux en pixels.

!Order Ordre de transfert des images (utilisé par les routines TV uniquement). 0 : première ligne en bas (défaut), 1 : première ligne en haut.

!P.multi Gestion du multi-affichage :
P(0) : nb de graphiques restants sur la page
P(1) : nb de graphiques en largeur
P(2) : nb de graphiques en hauteur
P(3) : nb de graphiques en Z
P(4) : ordre d'affichage lignes/colonnes
!P.multi = 0 rétablit la situation initiale.

!P.position Limites de la « fenêtre d'affichage » (les axes), [x0, y0, x1, y1], en coordonnées normales. Ce vecteur est modifiable, mais n'est pris en compte que si x0 est différent de x1.

!P.region Limites de la « région d'affichage » (axes + titres), [X0, Y0, X1, Y1], en coordonnées normales. Ce vecteur est modifiable, mais n'est pris en compte que si X0 est différent de X1. Il est initialisé à la taille de la fenêtre terminal si un seul graphique est affiché sur une page. !P.region = 0 rétablit la taille de la région d'affichage.

!P.clip Limites, en coordonnées normales, du masque d'affichage : seuls les graphiques destinés à l'intérieur de cette zone sont effectivement affichés. Ce vecteur est modifiable, et initialisé sur la fenêtre d'affichage. L'agrandir permet par exemple d'afficher des points en dehors des axes.

!P.noclip Désactive le masque d'affichage (« clipping ») s'il vaut 1.

!MAP Contient la matrice de transformation des coordonnées cartographiques (utilisateur) vers les coordonnées normales. Cette matrice est établie au moment où l'on définit la projection avec MAP_SET. Elle est perdue quand on change de fenêtre, il faut la

	stocker si on veut la récupérer ensuite. La définition de cette structure a été modifiée à partir de la version 5, et les routines plus anciennes qui l'utilise ne fonctionnent plus.
!x.style	Force l'utilisation exacte des limites d'axes (sans arrondi) s'il vaut 1 (et !y.style, !z.style).
!x.crange	[Xmin,Xmax] en coordonnées utilisateur après éventuel arrondi.
!x.s	Coefficients de transformation entre coordonnées utilisateur et normales : $Xn = S_1 * Xd + S_0$
!x.window	Limites de la fenêtre d'affichage (axes) en coordonnées normales [x0,x1]. Lisible, mais non modifiable (modifier !P.position).
!x.region	Limites de la région d'affichage (axes + titres) en coordonnées normales [X0,X1]. Lisible, mais non modifiable.
!x.margin	Espace entre la fenêtre d'affichage et la région d'affichage (où sont écrits les titres), en taille de caractères normaux : [X0-x0,x1-X1].
!x.omargin	Idem autour de l'ensemble de la fenêtre pour le multi-affichage.
!Mouse.Button	Indique quel bouton de souris a été utilisé au dernier appel de la fonction CURSOR. Jusqu'à la version 3.6, IDL utilisait la variable d'erreur !ERR pour stocker cette information.

Des variables !Y et !Z de structure identique à !X jouent sur les paramètres des deux autres axes. La variable !P contient également l'équivalent de la plupart des mots-clefs de style des fonctions graphiques (THICK, CHARSIZE…) et définit les valeurs par défaut de ces paramètres. Attention avec !x.window et !x.region : ces variables sont mises à jour à l'exécution des fonctions graphiques, pas quand on modifie !P ; elles ne sont pas non plus réinitialisées à l'ouverture d'une fenêtre, il ne faut donc les utiliser pour définir la taille des axes.

La bibliothèque Graphics_devices de SERTS permet de gérer simultanément plusieurs sorties graphiques en gardant trace des systèmes de coordonnées définis. Elle peut rendre service dans certains cas, mais est assez délicate d'emploi.

La fonction CONVERT_COORD permet les conversions, et gère également les transformations à 3D :

```
NV=convert_coord(X,Y,/data,/to_normal)
```

Les coordonnées d'entrée peuvent être dans une seule matrice à deux ou trois colonnes. Le résultat est toujours un vecteur à trois colonnes, la dernière étant à zéro si la transformation est à deux dimensions (c'est important quand on réutilise les points).

D'autres variables systèmes sont utiles pour les conversions, notamment :

!Pi et !Dpi	(simple et double précision)
!DtoR	= π/ 180. ; conversion degrés en radians
!Radeg	= 180./π ; conversion radians en degrés

Graphiques à deux dimensions

Les procédures suivantes travaillent par défaut avec des coordonnées utilisateur, matérialisées par des axes (x,y) ; à part OPLOT toutes acceptent également des coordonnées dans les autres systèmes, ou des coordonnées 3D.

Plot, X, Y	Définit un système d'axes et affiche Y en fonction de X (les deux vecteurs doivent avoir les mêmes dimensions). Les arguments peuvent être donnés en pixels ou en coordonnées normales, mais cette fonction définit toujours un nouveau système de coordonnées utilisateur, basé sur les données à afficher.
Oplot, x, y	Affiche Y en fonction de X en réutilisant le système de coordonnées utilisateur actif (typiquement, établi par le PLOT précédent). La fonction accepte des arguments polaires, mais ne comprend que les coordonnées utilisateur. Comme pour PLOT, les arguments sont nécessairement des vecteurs : pour afficher un seul point, il faut faire oplot, [3], [5], Psym=4 (il est plus simple d'utiliser PLOTS).
Plots, x, y	est une instruction de dessin. Elle fonctionne comme OPLOT par défaut, mais accepte également des arguments scalaires (définit la position du pointeur graphique) dans les trois systèmes de coordonnées d'IDL. Le mot-clef /CONTINUE permet le chaînage avec la position précédente ; le mot-clef COLOR permet de tracer les segments successifs dans différentes couleurs.
Contour,ima,X,Y	Trace les isolignes de ima. Les contours ainsi calculés peuvent être récupérés dans un tableau. Le graphique résultant peut être annoté,

les niveaux calculés choisis à l'avance… Le graphique est affiché en fonction des numéros d'indices (i,j), mais on peut fournir les coordonnées des axes dans X et Y. Les tableaux irréguliers doivent être ré-échantillonnés au préalable pour obtenir un résultat correct (voir TRIGRID plus haut). Si le tableau est régulier et complet, faire d'abord un REBIN : ça accélère le calcul et ça fournit un lissage de bonne qualité. On peut remplir les contours avec des couleurs en utilisant l'option /FILL ; dans les cas où les contours ouverts sont mal rendus, utiliser plutôt /CELL_FILL (qui est beaucoup plus lente). Les couleurs peuvent être spécifiées avec le mot-clef C_Colors=tab.

Contour_RMS, ima — Calcule des isolignes distantes d'un écart-type (dans la bibliothèque WINDT).

Polar_Contour — Calcule les contours d'un tableau en coordonnées polaires.

Polyfill, X, Y — Trace un polygone dont les sommets ont pour coordonnées X et Y. Le polygone peut être colorié ou rempli par un motif quelconque (y compris une image) avec le mot-clef PATTERN. Les coordonnées (X,Y) peuvent être fournies selon les trois systèmes de coordonnées d'IDL ce qui permet de superposer des données spatialement étendues sur un graphique, un contour, une carte, une image… Cette fonction n'utilise pas le masque d'affichage, et affiche donc les données qui dépassent des axes (on doit les filtrer explicitement pour s'en débarrasser).

PWidget, x, y — Outil de tracé graphique interactif. Capable d'afficher plusieurs vecteurs Y(N,M) en fonction de X(N), puis d'ajouter des symboles, des annotations, des titres… et de modifier les styles d'affichage à partir de l'écran. Le bouton PRINT écrit un fichier adapté au type de l'imprimante sélectionnée.

Graffer — Un autre outil graphique interactif (routine utilisateur disponible sur le réseau).

Annotate — Celui-ci est capable d'ajouter dans un fichier PS des commandes graphiques insérées sur l'écran, et donc d'annoter une image affichée au préalable.

Options de PLOT

PLOT efface par défaut la fenêtre active (sauf si le mot-clef /NOERASE est présent) et établit toujours un nouveau système de coordonnées. Si on veut superposer plusieurs courbes, c'est donc la première instruction à donner (il en va de même par défaut pour CONTOUR, mais le mot-clef /OVERPLOT inhibe l'effacement et conserve le système de coordonnées précédent).

Si X n'est pas fourni, le vecteur Y est affiché en fonction de ses numéros d'indice (ligne après ligne pour une matrice). Par défaut, les points affichés sont reliés par une ligne continue, les limites des axes sont arrondies à des valeurs simples et l'axe Y part de 0. Les options principales sont :

MAX_VALUE et MIN_VALUE Fixent l'intervalle pris en compte pour le vecteur Y.

/YNOZERO Fait partir l'axe Y du minimum au lieu de 0.

/POLAR Indique des coordonnées polaires (R, Θ dans cet ordre).

/Xlog et /Ylog Indiquent un axe logarithmique.

NSUM Affiche une moyenne mobile des vecteurs, *i.e.* réduit le nombre de points et lisse rapidement l'affichage.

PSYM Trace les points à l'aide de symboles prédéfinis, sans les relier entre eux (on peut rajouter une ligne après coup avec OPLOT). La taille des symboles est choisie avec le mot-clef SYMSIZE (en unités caractères). PSYM=8 indique un symbole utilisateur défini auparavant avec USERSYM (la fonction PLOTSYM de ASTRON permet d'en charger 8 types prédéfinis, la fonction SYMBOLS de WINDT en définit 32), PSYM=10 est un mode histogramme.

XRange et YRange Fixent les limites des axes (par défaut, elles sont ensuite arrondies). Ces paramètres permettent d'avoir des axes décroissants.

XStyle et YStyle Déterminent les options sur les axes (c'est équivalent à modifier !X.Style…). Les valeurs sont les sommes des options choisies :

1 : supprime l'arrondi sur les limites de l'axe.
2 : étend les limites de l'axe de 5%.
4 : supprime l'affichage de l'axe.
8 : supprime l'axe secondaire (la boîte).
16 : comme /Ynozero (Y seulement).

Position — Vecteur à quatre éléments déterminant la position des axes dans la fenêtre (modification temporaire de !P.position).

/NoData — N'affiche que les axes, sans données. En conjonction avec Xstyle=(YStyle=4), définit le système de coordonnées sans rien afficher.

/Ticklen — Force l'affichage d'une grille (ce mot-clef contrôle en fait la longueur des traits sur les axes).

De nombreux autres mots-clefs permettent de définir l'ensemble du graphique comme on l'entend, en particulier TITLE, XTITLE, CHARSIZE… La fonction FORMAT_AXIS_VALUES retourne une chaîne de labels formatés à partir de valeurs numériques, qu'on affiche avec le mot-clef YTICKNAME de PLOT.

PLOT peut s'utiliser pour définir seulement le système d'axes et les coordonnées ; OPLOT s'utilise normalement pour ajouter des segments sur un PLOT précédent :

```
Plot, X, Y, /nodata    ; définit la longueur des axes et les affiche
Oplot, lambda1, L1     ; affiche un premier spectre
Oplot, lambda2, L2     ; affiche un deuxième spectre
```

Autres fonctions graphiques

Les bibliothèques contributives en sont truffées, notamment ASTRON, MERON, FKK, WINDT et IMAGE_DISPLAY (dans SERTS). Celles qui servent souvent sont :

Surface,tab — Affiche une représentation 3D en fils de fer de tab. Le mot-clef SHADES permet de superposer une quatrième variable colorant la grille. Établit des coordonnées 3D. Les valeurs affichées en x et y sont par défaut les indices du tableau. On peut les fournir dans deux variables optionnelles ; si ces valeurs imposées sont irrégulièrement espacées, le tableau est localement dilaté pour rendre un affichage linéaire en x et y, mais n'est pas rééchantillonné (dans les versions récentes d'IDL en tous cas).

Shade_surf — Identique à la précédente, mais le rendu est grisé et continu, simulant une source de lumière. On remplacer celle-ci par une quatrième variable colorant la surface.

Set_Shading	Modifie l'orientation par défaut de la source de lumière pour SHADE_SURF. Le mot-clef VALUE permet de restreindre le domaine de couleurs utilisé par Shade_Surf (similaire à la mise à l'échelle dans TVSCL).
XSurface,Tab	Un outil interactif (widget) de tracé de surfaces regroupant les fonctions de SURFACE et SHADE_SURF. Permet de régler les orientations…
Velovect, Vx,Vy	Affiche la carte d'un champ vectoriel.
Errplot, X, E0,E1	Ajoute des barres d'erreur à un graphique. La plus souple parmi les nombreuses routines similaires des bibliothèques contributives.
Errbars, X, Y	Identique mais autorise les erreurs en X et Y. Appartient à la bibliothèque MERON.
histo=Hist_2D(a,b)	Retourne un diagramme de corrélation discret (fonction de densité) de a et b (entiers). On l'affiche ensuite en faisant TVSCL, histo.
Axis	Rajoute des axes où l'on veut. Permet notamment d'avoir plusieurs échelles d'unités sur le même graphique.
Arrow,x0,y0,x1,y1	Trace une flèche paramétrable entre les points mentionnés.
XYouts,X,Y,Ch	Affiche une chaîne de caractères Ch en X/Y (comprend les tableaux de paramètres pour les affichages multiples).
Legend	Trace à peu de frais différents types de légendes graphiques paramétrables. Dans la bibliothèque ASTRON (il en existe une version plus vieille et moins performante dans FKK).
Put_color_scale	Ajoute une échelle de couleur continue ou discrétisée sur un graphique. Dans la bibliothèque ESRG_UCSB.

Le tracé d'un histogramme propre peut être scabreux. La fonction HISTOGRAM(TAB) retourne l'histogramme de Tab ; on peut fixer les bornes avec MIN et MAX, la largeur des classes avec BINSIZE (1 par défaut). On doit recalculer à la main le vecteur en abscisses :

```
histo=histogram(Tab, binsize=b, omin=mi, omax=ma)
x= findgen(n_elements(histo))*b+mi
PLOT, x, histo, /PSYM=10
```

La procédure PLOTHIST de la bibliothèque ASTRON est similaire mais plus compacte ; elle utilise le mot-clef BIN pour définir la largeur des intervalles, et tous les mots-clefs habituels de PLOT pour la mise en forme. Attention à ne pas définir un intervalle (bin) trop petit par rapport à la dynamique, le calcul devient vite très long.

Projections cartographiques

La routine MAP_SET ne fait pas partie du noyau d'IDL mais de la bibliothèque utilisateur ; c'est probablement ce qui explique qu'elle soit curieusement écrite. MAP_SET a été complètement réécrite pour la version 5 d'IDL. La nouvelle version s'utilise de façon semblable, mais souffre de moins de limitations — par contre toutes les routines utilisant directement la variable !MAP (contenant la matrice de projection) sont incompatibles d'une version d'IDL à l'autre. MAP_SET requiert trois arguments un peu déroutants :

```
Map_set, lat0, long0, rot, position=[xmin,ymin,xmax,ymax], limit=[… ]
```

lat0 et long0 — Les coordonnées du centre du plan de projection. La bizarrerie est que la latitude est fournie en premier alors qu'elle représente une ordonnée ; c'est l'inverse dans toutes les autres fonctions qui demandent des coordonnées, telles que PLOT, mais c'est la notation traditionnelle. Toutes les longitudes utilisées par IDL suivent la convention terrestre (-180° à 180° en allant vers l'est), qui diffère de la convention planétaire et céleste (0° à 360 en allant vers l'ouest, sauf pour la Lune où les deux conventions sont admises), et n'a rien à voir avec les coordonnées en alpha/delta.

rot — La direction du pôle nord à partir du centre de projection. Cet angle est compté en degrés dans le sens horaire, à partir de midi (les autres angles d'IDL sont toujours donnés dans le sens direct).

Position — N'accepte que des coordonnées normales pour définir l'emplacement et la taille du graphique à l'écran. Le mot-clef /ISOTROPIC permet de conserver le rapport des deux dimensions sans avoir à calculer le vecteur position (à partir d'IDL 4).

Limites de la carte — On donne soit explicitement les coordonnées des deux coins opposés (vecteur à 4 éléments, latmin, longmin, latmax, longmax : attention, la latitude apparaît en premier) soit les coordonnées de

points situés sur les bords de la carte (dans l'ordre gauche, haut, droit, bas) ; il faut dans ce cas les recalculer pour les projections non cylindriques ou transverses (quand les bords verticaux ne sont pas des méridiens). C'est notamment important pour les projections sinusoïdales (le standard planétaire) quand elles ne sont pas centrées sur l'équateur : il faut impérativement recalculer les limites de longitude à mi-hauteur des bords de la carte à tracer. Si ceux-ci sont dans le fond de ciel, l'affichage correct est impossible.

Un autre piège se dissimule dans cette fonction : la zone définie par le mot-clef position est occupée par le cadre entourant la carte, l'espace réservé à la carte elle-même est plus petit. La routine agrandit en fait la zone à tracer (définie par limit) de 2% dans chaque direction sans prévenir. Si l'on veut superposer la carte à une image il faut en tenir compte en imposant le même grandissement aux dimensions de la carte (*ie*. au vecteur position) : ainsi l'image correspondra exactement aux limites en longitude et latitude qu'on fixe (voir exemple plus bas). À partir de la version 5 la fonction accepte un mot-clef NOBORDER qui inhibe l'affichage du cadre et utilise tout l'espace requis pour la carte, mais le résultat n'est pas bien beau.

Il arrive que MAP_SET produise une erreur "Coordinate system not established" ; c'est généralement parce qu'il y a un problème avec le vecteur Position (obligatoirement donné en coordonnées normales).

Un problème avec la version 5 : les labels affichés en dehors de la carte sont masqués par défaut. Le mot-clef CLIP_TEXT=0 de Map_grid supprime le masquage, mais n'est pas transmis par map_set (un bug ?). Il faut donc appeler map_grid séparément avec cette option.

Dernier problème, responsable de bien des cheveux arrachés : dans certains cas les limites de la carte ne sont pas celles qu'on spécifie, et la grille et les limites de continents ne s'affichent que sur une partie de la carte. Ceci se produit notamment quand on demande l'affichage d'une région limitée en fixant une longitude minimum supérieure à la longitude maximum (ce qui est parfaitement légitime, la longitude minimum étant simplement la limite ouest). On peut apparemment toujours s'en sortir en jouant sur les limites (±360°) *et* en déclarant explicitement la longitude centrale :

```
; suppose que les longitudes limites sont comprises entre -180 et +180
if (lomin gt lomax) then lomax = lomax+360.
longcent = (lomax + lomin)/2.
latcent = (lamin + lamax)/2
map_set, latcent, longcent,limit=[lamin, lomin, lamax, lomax]
```

MAP_SET manipule 22 projections différentes en version 5.2 (12 seulement dans la version 4) ; il va de soi qu'on ne les choisit pas au hasard, mais en fonction de ce qu'on veut montrer (certaines conservent les surfaces, d'autres les distances, certaines permettent de représenter les deux hémisphères complets, et parfois les données disponibles ou l'usage commandent…). Cette version 5.2 est enfin capable, mais très peu, de projeter sur un ellipsoïde plutôt qu'une sphère — mais seulement avec certaines projections peu utiles. IDL inclut une carte des côtes terrestres que l'on peut superposer à une projection quelconque, ainsi qu'une carte politique et une carte terrestre à haute résolution (installée en option et donc pas toujours disponible). Les mots-clefs permettent de disposer convenablement les labels de la grille et de modifier le style de présentation.

Autres fonctions cartographiques

MAP_IMAGE projette une image sur une grille cartographique quelconque ; les bords de l'image doivent correspondre à des longitudes et latitudes constantes, c'est donc nécessairement une carte en projection cylindrique ou l'image d'une zone assez petite pour que la courbure du globe soit négligeable. Elle convient pour projeter des images aériennes, mais pas pour des images télescopiques — celles-ci doivent être ré-échantillonnées en premier lieu. Le résultat de cette fonction est une image déformée selon la projection cartographique courante ; la fonction renvoie par ailleurs les coordonnées écrans où afficher cette image avec TV.

MAP_PATCH Remplit la même fonction mais donne dans certains cas des résultats plus satisfaisants.

LATLON ouvre une petite fenêtre dans laquelle s'inscrivent les coordonnées (latitude et longitude) des points cliqués. Cette fonction est ancienne et n'est pas maintenue par IDL ; elle se trouve dans le répertoire « Obsolete », et ne fonctionne effectivement pas toujours très bien, mais rend des services. Cette fonction utilise toujours la fenêtre 1, ce qui peut poser des problèmes. La même fonction peut s'obtenir point par point en faisant :

```
Cursor, Lon, Lat & print, lat, lon.
```

LL_ARC_DISTANCE Calcule les coordonnées d'un point à partir de sa distance à un point origine.

La bibliothèque ASTRON contient des fonctions de cartographie célestes, utilisables aussi pour la cartographie des planètes :

GCIRC est l'inverse de LL_ARC_DISTANCE (calcule la distance le long d'un grand cercle connaissant les coordonnées de deux points).

WCSSPH2XY, WCSXY2SPH, GLACTC, EULER et ASTRO sont les fonctions de conversions de coordonnées célestes (ASTRO est interactive et trace les cartes). Ces fonctions utilisent 25 projections cartographiques différentes, et gèrent les coordonnées célestes (même convention que les cartes planétaires). WCS_DEMO permet d'enregistrer des commandes de projections.

Les fonctions AITOFF, AITOFF_GRID, EQPOLE, EQPOLE_GRID gèrent les projections célestes courantes.

Graphiques et images

Un problème courant est de superposer une image (affichée avec TVSCL) et un graphique (affiché par PLOT ou CONTOUR) ou une grille cartographique (affichée par MAP_SET). Ces routines utilisent des systèmes de coordonnées différents qu'il faut ajuster. La méthode à suivre est différente selon qu'on affiche sur écran X ou qu'on écrit un fichier PostScript.

Les instructions de positionnement à utiliser sont :

- Dans TV ou TVSCL les arguments X0 et Y0 qui imposent la position du coin inférieur gauche de l'image (en pixels par défaut, ou dans les autres coordonnées, voire même en centimètres avec les mots-clefs appropriés). Ces routines n'effacent pas le contenu de la fenêtre mais le recouvrent :

```
TV, ima, x0, y0
```

- Dans PLOT, CONTOUR ou MAP_SET le mot-clef POSITION qui définit la taille et la position des axes dans la fenêtre terminal. Ce vecteur est en coordonnées normales par défaut (obligatoire avec MAP_SET), mais peut être forcé pour PLOT et CONTOUR en pixels avec /DEVICE (qui doit apparaître après le vecteur position ; une indication de coordonnées différentes peut apparaître avant le mot-clef POSITION, mais concerne alors l'unité des données à afficher). Il faut imposer l'option /NOERASE pour ne pas écraser l'image précédente, et forcer l'utilisation de

l'intervalle exact des données avec XSTYLE=1, YSTYLE=1 (sinon les longueurs des axes sont arrondies) :

```
PLOT, X, Y, position=[x0,y0,x1,y1], /noerase, /xstyle, /ystyle
```

- Quand on écrit un fichier PostScript, il est plus facile de tirer parti des propriétés du pilote PS : les pixels sont redimensionnables, on peut donc ajuster la taille de l'image à celle du graphique sans perte de qualité. En pratique, cela veut dire distinguer entre les pilotes PS et X…

Superposer une image à un contour

Si l'on peut sans problème calculer les niveaux d'intensité d'une image et les afficher sous forme de graphique, il est délicat de superposer correctement l'image aux contours. La fonction IMAGE_CONT de la bibliothèque utilisateur permet d'automatiser ce travail :

```
Image_cont, Image, /window
```

donne rapidement un bon résultat : le rapport des deux dimensions est préservé, et l'image n'est pas ré-échantillonnée (ce qui gagne du temps et ne dégrade pas le résultat : à l'écran la fenêtre prend les dimensions de l'image, en PS l'image est dilatée aux dimensions du graphique). En pratique si l'on ne veut n'afficher que certaines lignes, ou récupérer dans une variable les coordonnées des lignes calculées, il faut utiliser CONTOUR et recaler les graphiques à la main. La routine CONT_IMAGE de la bibliothèque WINDT peut être une alternative simple dans certaines situations.

Exemple : détermination d'un disque planétaire sur une image.

```
px0=20.       ; Décale la région graphique des bords de 20 pixels
py0=20.
sz=size(ima)

tvscl, ima, px0,py0  ; affiche l'image dans la région graphique

                     ; Superpose la grille sur l'image, et établit un système de
                     ;  coordonnées utilisateur en pixels (provenant de l'image).
                     ; Stocke le niveau 400 dans cont
                     ;  (obligatoirement en coordonnées normales).
contour, ima, levels=[400], path_xy=cont, /noerase,xstyle=1, ystyle=1,$
  position=[px0,py0,px0+sz(1)-1,py0+sz(2)-1],/device

              ; Transforme les coordonnées du contour en pixels et  calcule le centre.
cont0=convert_coord(cont,/normal,/to_data)
nelt=n_elements(cont)/2
moy=cont0#replicate(1,nelt)/nelt
```

```
                    ; superpose le contour à l'image et pointe le centre.
oplot,  cont0(0,*),cont0(1,*)
oplot,  [moy(0)],[moy(1)],psym=4
```

Une version PostScript de cette routine ressemblerait à :

```
contour, [[0,1],[0,1]], /nodata, xstyle=4, ystyle=4     ; n'affiche rien mais définit
                                                        ; la taille de la fenêtre
px=!x.window*!D.x_vsize             ; Limites (en pixels) de la fenêtre graphique
py=!y.window*!D.y_vsize
Dx=px(1)-px(0)                      ; Taille de la fenêtre
Dy=py(1)-py(0)
sz=size(ima)

f=float(sz(1))/float(sz(2))  *  Dy/Dx
If (f le 1.) then Dy=Dy/f else Dx=Dx*f
tvscl, ima, px0,py0, Xsize=Dx, Ysize=Dy   ; redimensionne l'image sur le graphique

contour, ima, levels=[400], path_xy=cont, /noerase,xstyle=1, ystyle=1,$
  position=[px0,py0,px0+Dx,py0+Dy],/device
```

Superposer une image à une carte

La routine MAP_SET définit un système de coordonnées cartographiques (longitude et latitude dans cet ordre) qui est compris comme coordonnées utilisateur (x,y). Cette transformation est généralement complexe, et fait intervenir une matrice de rotation stockée dans la variable !MAP. La fonction CONVERT_COORD permet d'effectuer simplement les conversions d'un système à l'autre, quel que soit le type de projection utilisée. Les fonctions PLOT, OPLOT, PLOTS et CONTOUR peuvent utiliser le système de coordonnées géographiques pour ajouter des graphiques sur une carte. Une image doit par contre être recalée pour que les coordonnées pixel correspondent aux coordonnées utilisateur.

La fonction MAP_IMAGE projette une image sur la grille cartographique courante quand on connaît les limites géographiques de l'image ; cette solution très simple doit être évitée pour afficher des images astronomiques, qui sont ré-échantionnées et souvent fortement dégradées au cours du processus. Il vaut beaucoup mieux calculer la position et les limites précises de la carte en fonction de la taille de l'image, ce qui est un peu plus compliqué.

Les images astronomiques sont fréquemment tournées d'un angle arbitraire, et doivent être redressées pour y superposer une grille cartographique. Si l'angle compté dans le sens direct est ANG, deux solutions sont possibles :

```
tvscl, rot(ima, ang)
```

```
map_set, 0, 0, 0, /grid, /ortho      ; redresse l'image en mettant le N en haut

tvscl, ima
map_set,0,0,-ang,/grid,/ortho      ; oriente la grille sur l'image.
```

La première solution suppose qu'on affiche les images de bas en haut, *ie*. que !ORDER=0 ; la seconde est plus rapide et n'altère pas la qualité de l'image, mais modifie l'orientation du graphique. La dégradation due à la rotation est généralement acceptable, contrairement à celle qui provient d'un redimensionnement.

Exemple : cartes cylindrique et Mercator à partir d'une image télescopique planétaire (orthographique).

```
; ** 1ere étape : On superpose la grille géographique sur l'image
sz=size(ima3)              ; taille image
px0=15.
py0=15.
px=[px0,px0+sz(1)-1]    ; coord (en pixels) pour la nv fenêtre
py=[py0,py0+sz(2)-1]    ; (décale un peu les bords)
tvscl, ima3, px(0), py(0)

qx=px/!D.x_vsize           ; transformation en coord normales pour Map_set
qy=py/!D.y_vsize
del=(qx(1)-qx(0))*0.01          ; Allonge les axes de 2% pour compenser
qx(0)=qx(0)-del                 ; l'agrandissement similaire des limites
qx(1)=qx(1)+del                 ; en long et lat effectué dans MAP_SET
del=(qy(1)-qy(0))*0.01
qy(0)=qy(0)-del
qy(1)=qy(1)+del
                                ; Superpose la grille et définit les coord lat/long
; Les coordonnées du centre sont données par les éphémérides et l'heure TU du cliché
; (les éphémérides donnent les longitudes en convention planétaire).
; L'angle de rotation est l'azimut du pôle Nord plus l'angle de la caméra dans le
; plan (compté pour Map_set dans le sens rétro à partir de midi).
; Selon l'image, le système d'exploitation et le périphérique de sortie, la grille peut-
; être décalée de 1 pixel sur chacun des bords indépendamment. A regarder de près si
; on a besoin de précision (affiner les valeurs de px et py plus haut).

lat0=19.21
long0=140.          ; Attention, IDL veut les longitudes en convention terrestre
rot0=-135.
map_set,lat0,long0,rot0,/ortho,/grid,position=[qx(0),qy(0),qx(1),qy(1)],   $
  /noerase


; ** 2e étape : On échantillonne le plan de la carte cylindrique, et on va chercher
; le pixel correspondant dans l'image : c'est la méthode la plus précise.
inci=60.                   ; N'utilise que le centre du disque
dim=80                     ; Ré-échantillonne en long/lat, i.e. reconstitue une
rap=inci*2./dim            ; projection cylindrique autour du centre
lat=(lon=fltarr(dim*dim))
for i=0,dim-1 do begin
```

```
  j=i*dim
   lon(j:j+dim-1)=findgen(dim)*rap+long0-inci
   lat(j:j+dim-1)=replicate(i*rap+lat0-inci,dim)
endfor

xy=convert_coord(lon,lat,/data,/to_device)        ; Renvoie une matrice à 3 dimensions
xy(0,*)=xy(0,*)-px(0)          ; Les points (lat,lon) qui ne sont pas visibles
xy(1,*)=xy(1,*)-py(0)          ; ont une valeur xy=10^12
Plan=fltarr(dim,dim)

maxx=max(xy(0,where(xy(0,*) le sz(1)))) ; Sélectionne les points de l'échantillon
maxy=max(xy(1,where(xy(1,*) le sz(2)))) ; qui sont sur le disque.

ind=where((xy(1,*) le maxy) and (xy(0,*) le maxx))
plan(ind)=ima3(xy(3*ind),xy(1+3*ind))        ; Projection cylindrique

window,2
pla=rebin(plan,dim*4,dim*4)     ; affiche un agrandissement
tvscl, pla                      ; de la projection cylindrique

; ** 3e étape : On reprojette en Mercator (ou en n'importe quoi d'autre)
lamin=min(lat) & lamax=max(lat)
lomin=min(lon) & lomax=max(lon)
minipla=min(pla(where(pla gt 400)))   ; mini sur le disque (même limite que
                                      ; pour le contour)
; Recalcule une projection de Mercator centrée sur l'équateur
; Map_image dégrade un peu le résultat, et ne s'impose
; pas pour passer de cylindrique en Mercator - mais c'est obligatoire pour
; passer à certaines autres projections.

window, 3
Map_set,0,long0,/grid, /mercator, /label, limit=[-85,long0-180, 85, long0+180]
pla1=Map_image(pla,startx,starty,latmin=lamin,latmax=lamax, $
   lonmin=lomin,lonmax=lomax, /bilinear)
tvscl, pla1>minipla, startx, starty ; Affiche l'image au bon endroit
```

Remarques importantes pour les sorties PostScript

- L'équivalent en PostScript d'une fenêtre X est la partie de la feuille qu'on utilise. Ses dimensions en centimètres sont fixées à l'ouverture du fichier. La feuille est par défaut au format B5, avec des marges importantes. On peut régler la taille et la position de la partie utilisée avec Device :

```
DEVICE, Xsize=Tx,Ysize=Ty,Xoff=Ox,Yoff=Oy, filename="Tartempion.ps"
```

- Par défaut, les images affichées par TV sont redimensionnées pour occuper toute une dimension de la feuille en préservant le rapport des dimensions. Les coordonnées normales varient toujours de 0 à 1 dans la zone utilisée.

- Pour une image de taille (Nx,Ny) destinée à être insérée dans un texte (fichier EPS), on a intérêt à ajuster les dimensions du fichier pour éviter les blancs. Il y a

normalement 1000 pixels par centimètre, l'ajustement se fait donc en imposant Tx=Nx/1000. et Ty=Ny/1000 dans DEVICE. TV affichera ensuite l'image sur toute cette zone.

- Inversement, on peut vouloir restreindre l'affichage à une partie de la zone disponible, notamment pour superposer une image à un graphique, pour afficher une série d'images côte à côte (avec TV, ima, 0…), ou pour modifier le rapport des dimensions. La fonction TV et ses dérivées acceptent en sortie PS deux mots-clefs supplémentaires, ignorés lors des affichages écran : XSIZE et YSIZE fixent les dimensions de l'image, dans la même unité que les arguments de position (en pixels par défaut).

Pour afficher une image avec des dimensions similaires en PS et à l'écran :

```
sz=size(ima)
TV, ima,xsize=sz(1)*!D.x_px_cm/40., ysize=sz(2)*!D.x_px_cm/40.
```

Le nombre de pixels par centimètres étant généralement de 40 à l'écran, ce réglage fournit une sortie PS de la taille de la fenêtre X (ce rapport peut dépendre de la machine). Si on doit décaler l'image du bord, c'est plus délicat (penser à mettre des constantes de type réel…) :

```
if keyword_set(ps) then begin
  set_plot,'ps'
  device,filename='CiSiArrangia.ps', bits=8, /landscape
endif
else window, /free, xsize=640, ysize=512
; coefPS=!D.x_px_cm/40.          ; graphique PS de la même taille que sur l'écran
coefPS=min([!D.X_vsize/640.,!D.Y_vsize/512.])          ; image sur toute la feuille
px0=20.
py0=20.
sz=size(ima)
TV, d, px0*coefPS,py0*coefPS,xsize=sz(1)*coefPS, ysize=sz(2)*coefPS
```

- Il faut parallèlement imposer les dimensions correctes aux instructions de position des graphiques qu'on superpose. Le plus simple est d'utiliser systématiquement des coordonnées normales, on évite ainsi également les problèmes liés à la taille des marges :

```
qx=[px0,px0+sz(1)]/!D.x_vsize*coefPS          ; Limites en coord normales
qy=[py0,py0+sz(2)]/!D.y_vsize*coefPS
plot, X,Y,/noerase, position=[qx(0),qy(0),qx(1),qy(1)], /normal
```

- Une série de TV ou TVSCL sans redimensionnement superpose les images sur la même feuille. On peut insérer un saut de page dans le fichier PS avec ERASE.

- Les fichiers PostScript sont écrits par défaut en Noir et Blanc sur 4 bits (16 niveaux de gris). On peut écrire en 256 niveaux de gris en ajoutant BITS=8 dans DEVICE et en couleurs avec /COLOR. La feuille est au format français à moins qu'on ne fasse /LANDSCAPE dans DEVICE (il peut alors se passer des choses curieuses en EPS).

Exemple : Superposition de données spatiales sur une carte planétaire.

```
Pro Mars

tab=bytarr(1440,723)
lat=fltarr(4)
lon=fltarr(4)

if keyword_set(ps) then begin
  set_plot,'ps'
  !P.font=0                          ; Utilise les polices PostScript
  device,filename='Aurorae.ps',/color,bits=8,/landscape,/Bold
endif
else window, /free, xsize=850, ysize=420

; coefPS=!D.x_px_cm/40.               ; image de la même taille que sur l'écran
coefPS=min([!D.X_vsize/850.,!D.Y_vsize/420.]) ; image sur toute la feuille

restore, 'Mars.idl'     ; Relit la carte d'albédo NASA-PDS dans le tableau a
                        ; La carte complète fait 1440 x 720, en projection sinusoïdale

e=a(440:799,280:439)   ; correspond à long: [-70;20], lat: [-20;20] à l'équateur
a=1                          ; Libère la place
d=rebin(e,720,320)
d=Bytscl(d, Top=211)       ; utilise les 211 premières couleurs pour la carte

sz=size(d)
px0 =20.                     ; décale un peu du bord
py0 =20.
                                ; Retour en coordonnées normales pour Map_set
qx=[px0,px0+sz(1)+0.5]/!D.x_vsize*coefPS
qy=[py0,py0+sz(2)+0.5]/!D.y_vsize*coefPS
del=(qx(1)-qx(0))*0.01          ; Compense l'allongement des limites dans
qx(0)=qx(0)-del                 ; MAP_SET par un agrandissement similaire
qx(1)=qx(1)+del                 ; de la fenêtre d'affichage.
del=(qy(1)-qy(0))*0.01
qy(0)=qy(0)-del
qy(1)=qy(1)+del


la=[-20,20]
lo=[-70, 20]                              ; Limites à l'équateur
lax=(la(0)+la(1))/2.                      ; Latitude au milieu des bords
lo1=lo/cos(lax*!dtor)        ;Long  au centre des bords E et W en projection sinusoïdale
print, lo1, la
c0=[lax,lo1(0)]             ; coordonnées des milieux des 4 bords
c1=[la(1),(lo(0)+lo(1))/2.]
c2=[lax,lo1(1)]
c3=[la(0),(lo(0)+lo(1))/2.]
```

```
                                ; Affiche la carte
TV, d, px0*coefPS,py0*coefPS,xsize=sz(1)*coefPS, ysize=sz(2)*coefPS
                                ; Calcule les coordonnées utilisateur et affiche la grille
map_set, /sinu,/grid, limit=[c0,c1,c2,c3],/noerase, /label,title=$
 'Observations ISM dans Aurorae Planum!C',latlab=-72,lonlab=-22,$
  londel=10.,latdel=10.,position=[qx(0),qy(0),qx(1),qy(1)],$
 glinethick=2.,glinestyle=0

numspec=121*25                  ; nombre de pixels ISM
restore, filename='aurorae.idl' ; fichier coordonnées/réflectance ISM dans TAB
nbf=0
maxi=max(ima)
mini=min(ima(where(ima ne 0))) ; code le niveau sur les couleurs 213 à 227
col=(bytscl(tab(0,*), min=mini, max=maxi, top = 15)+213)<227


for j=0,numspec-1 do begin      ; on fait une boucle à cause du Polyfill
  nbf=nbf+1
 for i=1,4 do begin
      lon(i-1)=tab(2*i,j)/100.          ; Relit les coordonnées lat/long des quatre
      lat(i-1)=tab(2*i+1,j)/100.        ;  coins des pixels.
 endfor
 if (lon(1) ge -70.) then $             ; Filtre ce qui est en dehors de la carte
        polyfill,lon,lat,color=col(j)   ; Trace le pixel
endfor
print, 'Nb de pixels:', nbf
                                        ; Écrit une légende automatique
colors=reverse(findgen(15)+213)         ; Couleurs réflectance
citems=strarr(15)
citems(0)=string(format='(F5.2)',maxi/32537./2.)
citems(7)=string(format='(F5.2)',(maxi+mini)/32537./4.)
citems(14)=string(format='(F5.2)',mini/32537./2.)
legend, citems,/fill,psym=8+intarr(15),colors=colors,char=1.5,$
   pos=[qx(1)+0.01,qy(1)],/normal
xyouts, qx(1)-0.015,qy(1)+0.02,/normal, 'Réflectance ISM'

if keyword_set(ps) then begin
 device,/close
 set_plot, 'x'
  !P.font=-1
endif

end                                     ; Et voilà
```

Gestion des tables de couleurs

Le nombre de couleurs disponibles dépend de la sortie graphique. Sur la plupart des terminaux X et en PostScript, les couleurs sont codées sur 8 bits (256 niveaux). Certains écrans travaillent avec trois plans couleurs indépendants (256^3 niveaux), et on

peut écrire des fichiers 24 bits en PostScript. Il faut savoir que les imprimantes à jet d'encre reproduisent mal les images en couleurs ou niveaux de gris, même en PostScript 8 bits — et que les imprimantes non PostScript ne disposent souvent que de 16 couleurs.

Codage des couleurs

Les couleurs utilisées sont toujours définies en interne par leurs niveaux dans les trois composantes fondamentale, rouge, vert et bleu, chacun codé sur 8 bits. Sur la plupart des sorties graphiques, seules 256 couleurs différentes sont accessibles simultanément parmi les 256^3 possibles. De tels systèmes utilisent une table de couleurs à 256 entrées, qui consiste en un tableau de trois colonnes contenant la définition RVB de chaque couleur (système indexé). Modifier cette table (ie modifier la définition des couleurs) modifie les affichages en temps réel à l'écran (affichage dynamique). La table de couleurs indéxée est généralement partagée entre les applications actives, toutes les couleurs ne sont donc pas accessibles à IDL.

Les affichages 24 bits utilisent un autre système de codage, où les composantes RVB sont spécifiées explicitement en chaque point de l'image. Dans ce cas, les couleurs peuvent être soient définies directement, (mode TrueColor), soit affectées à travers une table de couleurs, et donc modifiables à travers celle-ci (mode DirectColor). Dans le premier cas une modification de la table de couleurs ne modifie pas l'affichage à l'écran (affichage statique). Le fonctionnement du mode DirectColor (disponible sous Unix uniquement) dépend en fait du système d'exploitation, et peut-être dynamique ou statique.

Les mots-clefs de DEVICE permettent de savoir dans quel mode graphique se trouve IDL, et de modifier celui-ci avant affichage de la première fenêtre de la session :

```
Device, GET_VISUAL_NAME=name, GET_VISUAL_DEPTH=depth
        ; retourne le mode et le nombre de couleurs en fonction
Device, Pseudo_Color=8      ; force un affichage 8 bit dynamique
Device,  True_Color=24      ; force un affichage 24 bits statique
```

Forcer le mode d'affichage permet en principe de retrouver un fonctionnement cohérent sur différents systèmes. À noter que les modes d'affichage étaient moins clairement définis avant la version 5.1, et que des problèmes persistent sous Windows en version 5.2. Par ailleurs les modes d'affichage 16 bits sur Mac et PC laissent à désirer.

Modèles colorimétriques reconnus par IDL

IDL travaille en interne avec le système RVB (RGB), mais on peut spécifier les couleurs en HLS ou HSV (parfois nommé HSI). Chaque modèle décompose les couleurs en trois composantes. En RVB, les trois indices représentent l'intensité croissante de chacune des primaires rouge, vert et bleu sur une échelle allant de 0 à 255 ; les gris sont situés le long d'une diagonale de l'espace RVB. Les deux autres systèmes décrivent séparément l'intensité globale sur une composante (L ou V) et la couleur sur les deux autres.

L (luminosité) varie de 0.0 à 1.0 entre le noir le blanc, et représente une quantité de gris (c'est la moyenne des composantes RVB, donc une distance le long de la diagonale de l'espace RVB). La valeur 0.5 correspond au maximum d'intensité de la couleur.

V (valeur, ou intensité) varie de 0.0 à 1.0 entre le noir et une couleur intense (équivalent à L mais ne provoque pas de fondu au blanc).

S (saturation) varie de 0.0 à 1.0 et représente l'inverse de la dilution de la couleur dans le blanc (c'est une distance à la diagonale RVB : les couleurs du spectres sont saturées).

H (nuance, pour hue) est un angle qui varie entre 0.0 et 360.0 degrés (autour de la diagonale RVB). La valeur 0 correspond au rouge, 120 au vert et 240 au bleu.

Manipulation des tables de couleurs

IDL contient une quarantaine de tables de couleurs prédéfinies. Une session démarre toujours avec la table 0 — niveaux de gris réguliers. Les instructions de chargement les plus utiles sont :

Loadct, n	Charge la table de couleurs standard N. On peut préciser l'intervalle d'indices à réaffecter à l'aide des mots-clefs BOTTOM et NCOLORS. Une liste des tables existantes est retournée quand on ne fournit pas d'argument.
Xloadct	Outil de sélection et de modification des tables de couleurs standard.
Modifyct	Modifie et sauve la table de couleurs courante, qui peut être rappelée ensuite grâce à LOADCT.
TVLCT, R, V, B, I	Charge une table de couleurs personnelle à partir de l'indice I. Les couleurs sont définies par leur proportion de rouge, vert et bleu (R, V, et B sont des vecteurs dont la dimension est égale au nombre de

couleurs modifiées). On peut définir les couleurs en systèmes HLS ou HSV avec les mots-clefs correspondants. Avec le mot-clef /GET on charge la table courante dans les vecteurs sans rien modifier.

Xpalette et Color_Edit Outils de construction interactifs de table de couleurs.

Les instructions de conversion et de calcul de tables sont :

Color_convert Conversion de RGB à HLS ou HSV et inversement.

Ct_Luminance Calcule l'intensité des indices de la table courante.

HLS et HSV Écrivent des tables variant régulièrement entre les limites fournies dans ces deux systèmes.

Pour éviter les problèmes de cohabitation entre routines graphiques, il est conseillé de ne modifier les tables de couleurs qu'à travers le bloc COMMON géré par IDL (c'est parfois indispensable dans les widget), plutôt qu'avec TVLCT :

```
        ; On déclare le bloc
COMMON COLORS, R_ORIG, G_ORIG, B_ORIG, R_CURR, G_CURR, B_CURR
        ; on modifie les valeurs courantes
R_Curr(cur_idx) = value
        ; et on charge la table
tvlct, r_curr(cur_idx), g_curr(cur_idx), b_curr(cur_idx), cur_idx
```

Ajuster la dynamique

Le problème est de faire correspondre un domaine de valeurs intéressant dans l'image [Mini, Maxi] avec un domaine d'indices de la table [Cmin, Cmax]. Par défaut, on utilise tous les indices disponibles (Cmin=0 et Cmax=!D.N_colors). On peut modifier la dynamique de l'image ou celle de l'écran :

TVSCL, image > Maxi < Mini est un peu lent à exécuter mais facile à écrire…

Ima=BYTSCL(ima) plus rapide, ré-échelonne IMA entre 0 et 255 (ou la valeur introduite par le mot-clef TOP). Les options MIN et MAX permettent de ne ré-échelonner qu'un intervalle de valeurs (le min et le max de l'image sont utilisés sinon). Pour obtenir 15 couleurs entre 100 et 114, faire :

```
col=(bytscl(A),min=mini,max=maxi,top=15)+100)<114
```

De cette façon, la dernière classe est normalement peuplée (avec top=14 et sans l'écrêtage, elle ne contient que le maximum).

Stretch, Cmin, Cmax, G Ré-échelonne la table de couleurs entre les indices CMIN et CMAX. Le troisième argument est une correction de gamma. Sans argument, recharge la table non modifiée. Les indices < Cmin pointent la première couleur, les indices > Cmax pointent la dernière, ou la première si l'option /CHOP est activée.

Multi, coef est du même genre, mais permet d'obtenir des tables cycliques (coef est le nombre de fois où la table est parcourue entre 0 et !D.N_colors).

H_EQ_CT, ima Modifie la table courante pour obtenir une égalisation d'histogramme de ima (qui doit être une image en octets). La routine H_EQ_INT est semblable mais interactive.

Ima1=HIST_EQUAL(Ima) Inversement, cette fonction retourne l'image égalisée.

Couleurs réelles et simulations 8 bits

Les terminaux possédant suffisamment de mémoire graphique (VRam) permettent de travailler sur trois plans couleur simultanément. Les terminaux X en 24 bits sont encore chers, mais un ordinateur de bureau récent fait ça très bien pour un prix modique. Une telle machine peut d'ailleurs être utilisée pour faire tourner IDL sur une station de travail distante dans de très bonnes conditions : il suffit pour cela d'une carte Ethernet, de 4 Mo de Vram (pour gérer un écran de 17 pouce en 24 bits) et d'un émulateur X (Exceed sous Windows NT ou MacX sur Macintosh, qui interfacent tous deux naturellement les fonctions terminal et le système local). Les pilotes graphiques d'IDL reconnaissent spontanément les capacités d'un tel système.

Les images 24 bits sont affichées avec un codage indépendant sur chaque plan de couleur : la couleur de chaque pixel est codée sur trois octets, donnant respectivement les intensités en rouge, vert et bleu ; l'image est donc un tableau à trois dimensions. Sur une sortie 24 bits, on utilise soit une table de couleurs régulière, soit une table lue avec l'image (si elle provient d'un fichier JPEG par exemple), et l'on affiche celle-ci avec TV en indiquant avec le mot-clef TRUE dans quel ordre est rangé le tableau :

```
TVLCT, indgen(256), ingen(256), indgen(256) ; table linéaire en RVB
TV, ima24, TRUE=3                           ; Affiche l'image en mode 24 bits, BSQ
```

TRUE	Entrelacement	Dimensions de Image	Nom du mode
1	par pixel	(3,N,M)	BIP (band interleaved by pixel)
2	par ligne	(N,3,M)	BIL (band interleaved by line)
3	par image	(N,M,3)	BSQ (band sequential)

(les noms de modes de stockage mentionnés ci-dessus sont utilisés notamment dans les archives de données planétaires de la Nasa).

On peut alternativement charger trois images RVB indépendantes :

```
TVLCT, indgen(256), ingen(256), indgen(256) ; table linéaire en RVB
TV, imageR, Channel= 1                      ; Affiche l'image R dans le canal rouge…
TV, imageV, Channel= 2
TV, imageB, Channel= 3
```

Les canaux 1, 2, 3 sont affectés respectivement aux plans RVB ; chacune des tables donne la proportion de la couleur R, V ou B qui correspond à un indice (permet de corriger les nuances affichées dans une image, ou de faire des composites trichromes). L'indication de canal 0 force l'écriture sur les trois plans couleurs en même temps (affichage Noir et Blanc). Les écrans 8 bits ne tiennent pas compte de ce mot-clef (la routine TVRD peut fonctionner différemment sur des écrans 8 ou 24 bits). On sélectionne une couleur particulière en calculant son indice de l'une des façons suivantes :

```
plot, X, Y, color = R + 256L * (G + 256L * B)
 ; L'indication d'entier long (256L) est nécessaire pour obtenir un résultat correct
plot, X, Y, color='RRVVBB'x
; ou plus simplement on définit la couleur par son code hexa
```

La possibilité de travailler dans trois plans couleur indépendants est particulièrement intéressante pour superposer des informations visuelles. Dans l'exemple des observations de Mars plus haut, on aimerait représenter la carte Nasa en noir et blanc sur 256 niveaux de gris, la réflectance mesurée par ISM par une variation de couleurs, et une troisième information indépendamment ; ce serait immédiat en travaillant dans le système HLS sur 24 bits. On peut néanmoins reconstituer ce résultat en travaillant en 8 bits, pour peu que l'image résultante ne comporte pas plus de 256 couleurs différentes. La fonction COLOR_QUAN calcule une table de couleurs 8 bits basée sur les couleurs les plus fréquentes d'une image particulière, et retourne une image codée selon cette table. Le mot-clef COLOR permet de choisir le nombre d'entrées de la table synthétique (!D.N_COLOR par défaut). Avec le mot-clef CUBE (racine cubique du nombre d'entrées, de 2 à 6) on définit une table de moins bonne qualité, mais utilisable pour afficher d'autres images.

Exemple : calcul d'une image composite couleur à partir de trois images monochromes.

```
x=read_fits('SaturneI.fits',h)      ; lit trois images télescopiques à travers des
y=read_fits('SaturneV.fits',h) ;filtres B, V, I (elles doivent être calées en position)
z=read_fits('SaturneB.fits',h)
col=!D.N_colors
x=bytscl(x,TOP=col)                 ;Les étalonne (on suppose que la région
y=bytscl(y,TOP=col)                 ; la plus brillante est blanche — la même
z=bytscl(z,TOP=col)                 ; intensité dans les trois filtres)
trichro=color_quan(z,x,y,r, g, b) ;Retourne l'image codée et la table de couleurs
tv, trichro                         ; Affiche l'image.
tvlct, r, g, b                      ; Charge la table.
```

Particularités des terminaux X

En général, les 256 niveaux de couleur ne sont pas accessibles : IDL utilise par défaut une table de couleurs partagée entre les applications actives, et réquisitionne les entrées libres. Au mieux, on n'a donc accès qu'aux couleurs qui ne sont pas utilisées par le gestionnaire de fenêtres X, aux alentours de 220 dans les cas favorables, et ce nombre varie souvent d'une session sur l'autre. Le nombre de couleurs disponibles est conservé dans !D.N_colors. On peut tout de même utiliser les 256 indices de couleurs, mais au-delà de D.N_colors-1, tous correspondent à la dernière couleur disponible.

La seule façon de retrouver une table de même dimension d'une session sur l'autre est de fixer un nombre de couleurs. Il faut ouvrir la *première* fenêtre de la session avec :

```
Window, 0, colors=220
```

S'il n'y a pas suffisamment de couleurs disponibles, IDL crée une table personnelle qui peut recouvrir partiellement celle utilisée par le gestionnaire de fenêtres. Tout marche normalement, mais l'affichage de l'écran est entièrement modifié (et souvent illisible) dès qu'on active une fenêtre IDL, et réciproquement. Pour éviter ce problème, il faut minimiser les entrées couleur utilisées par le gestionnaire X, et lancer IDL sans autre application graphique active (Netscape est à bannir en particulier). Les sessions IDL lancées à travers un rlogin semblent utiliser systématiquement une table personnelle.

Il peut être génant dans certains cas d'avoir trop de couleurs disponibles en sortie X ou sur un micro-ordinateur, notamment pour utiliser des routines écrites pour affichage 8 bits. Sur Mac ou PC, il suffit de régler le nombre de couleurs du système à 256 avant de lancer IDL. Sur terminal X, il faut taper :

```
Device, decomposed=0
Device, PseudoColor=8, decomposed=0 ; ou ceci, dès le début de la session IDL
```

Ceci n'est pas cependant équivalent à utiliser un écran 8 bits. En particulier, les modifications de la table de couleurs ne sont pas reflétées à l'écran, et ne s'appliquent qu'aux images et graphiques affichés ensuite.

Particularités du PostScript

- La table de couleurs doit être chargée avant l'affichage, elle n'a pas valeur rétroactive comme sur un terminal X.

- Les graphiques sont affichés en noir sur fond blanc, mais ce n'est pas le cas pour les images. Si une image possède un fond uniforme, il faut le mettre en blanc avant de l'afficher : c'est plus lisible, et ça épargne la cartouche noire de l'imprimante (pas conseillé pour les images astronomiques en fait, à cause des halos autour des objets).

```
ima(where(ima eq 0B)) = 255B   ; si le dernier indice correspond au blanc
TVscl, ima                     ;    (la plupart des tables standard)
```

- En noir comme en couleur, les fichiers PS sont écrits en 4 bits par défaut. Pour disposer de 256 niveaux de couleur (ou de gris), il faut l'imposer :

```
Set_plot, 'PS'
Device, filename='Tanti_Grigi.ps', bits=8
```

Les valeurs acceptables sont 1, 2, 4, ou 8. Des problèmes peuvent se produire sur les bordures d'une image affichée sur 2 ou 4 bits (le nombre de colonnes doit être pair en 4 bits, multiple de 4 en 2 bits). Attention, la plupart des imprimantes laser PostScript NB ne disposent que de 16 niveaux de gris (les LaserWriter de l'IAS en particulier).

- Contrairement aux sorties X, le pilote PS dispose toujours des 256 couleurs quand on a demandé une sortie sur 8 bits. En général, il faut donc éviter d'utiliser la même table de couleur (avec set_plot, 'PS', /copy), sous peine de saturer les valeurs claires.

- Les fichiers sont en Noir et Blanc par défaut. La couleur se déclare explicitement, et il ne faut pas tenter d'envoyer un fichier couleur sur une imprimante NB :

```
Set_plot, 'PS'
Device, filename='A_Colori.ps', /color, bits=8
```

- Pour repasser en mode Noir et Blanc après une image couleur :

```
Device, color=0
```

- Les fichiers en couleurs réelles s'ouvrent comme les autres, sur 8 bits (exemple précédent). La seule différence est dans le mot-clef TRUE de TV et TVSCL.

Quelques remarques sur les impressions couleur pour finir. Le résultat est souvent décevant par rapport aux sorties écran, généralement de qualité très inférieure avec une imprimante à jet d'encre, et pas toujours convaincant même avec une très bonne imprimante. Il faut se souvenir que certaines couleurs affichables sur l'écran ne sont pas imprimables telles quelles : les imprimantes fonctionnent en synthèse soustractive, les écrans en synthèse additive ; la conversion dépend du système d'impression, et le rendu est généralement beaucoup moins lumineux. Certains préfèrent utiliser des photos d'écran pour les figures d'article, mais c'est également une technique difficile.

Une simple recette de base : si l'image contient des couleurs complexes il faut essayer d'optimiser la table de couleurs pour l'impression. En système HLS, la teinte H est obtenue en juxtaposant deux trames de couleurs, la luminosité L est modifiée en ajoutant une trame blanche plus ou moins serrée, et la saturation avec une trame noire. Ces trames utilisent des motifs différents, propres à l'imprimante, qui ne sont pas simples à reprogrammer et peuvent être plus ou moins adaptés à une image donnée. On voit souvent des imprimantes de même modèle utiliser des trames différentes, il n'est donc pas inutile de faire des essais sur chacune (c'est le cas des diverses HP DeskJet de l'IAS notamment). Ces juxtapositions de trames sortent particulièrement mal, surtout sur de petites surfaces ou des lignes, et on a tout intérêt à éliminer au moins la trame noire en utilisant des couleurs presque saturées. Si les couleurs de l'image sont arbitraires (codes de valeurs, graphiques…), il faut choisir une à une les couleurs qui sortent le mieux (le plus simple est d'écrire une routine qui affiche la table de couleurs, et de l'envoyer sur l'imprimante qu'on utilise) ; si les couleurs représentent des variations physiques continues, il n'est généralement pas illicite de décaler la table entière pour optimiser l'impression.

Enregistrement de fichiers images

L'enregistrement d'images à l'aide des fonctions d'écriture en 8 bits (write_gif, write_bmp…) apporte parfois quelques surprises. Ces routines stockent toujours la table de couleurs après l'image, en imposant le nombre d'entrées du pilote d'écran actif. Sur un terminal X 8 bits le nombre d'entrées de la table est généralement inférieur à 256 niveaux ; il est contenu dans la variable !D.n_colors, la table enregistrée est complétée à 256 en répétant sa dernière valeur. Si l'image elle-même est stockée sous forme de tableau d'octets entre 0 et 256, les niveaux situés entre !D.n_colors et 256 se verront tous

attribuer la même couleur, c'est à dire que l'image sera saturée dans les tons clairs. Il n'y a apparemment rien à faire pour récupérer les niveaux manquants même sur une autre machine, bien que les niveaux corrects soient présents dans le fichier. L'affichage du fichier enregistré produit un résultat similaire à :

```
TV, image < !D.n_colors
```

Il faut donc ré-échelonner l'image sur le nombre de couleurs disponibles avant de l'enregistrer :

```
x=bytscl(x,TOP=!D.N_colors)
```

Ce qui réduit la dynamique de l'image… Quand il est important de préserver le signal sur bruit il faut forcer l'utilisation d'une table de 256 indices. On peut le faire à l'ouverture de la première fenêtre de la session IDL, avec :

```
window, 0, color=256
```

Cette méthode implique une bascule de la table de couleurs un peu pénible chaque fois qu'on active une fenêtre graphique. Une autre méthode qui devrait marcher est de passer par le pilote Z qui dispose toujours de 256 couleurs :

```
set_plot, 'Z'
loadct, 0                              ; recharge une table 256 niveaux
write_gif, "truc.gif", image
set_plot, 'x'
```