# INTER

# Manuel de référence

24 octobre 2012

# Table des matières

# Chapitre 1

# FONCTIONS

**OPERATIONS**

| | |
|---|---|
| + | addition. |
| − | soustraction. |
| * | multiplication. |
| / | division. |
| ** | exponentiation. |

**OPERATIONS DOUBLE PRÉCISION**

| | |
|---|---|
| DPLUS(x,y) | Addition. Les arguments sont numériques ou caractères, le résultat est numérique en simple précision. |
| DMINUS(x,y) | Soustraction. Même remarque que pour DPLUS. |
| | **Ex:** `dminus(jd(),"2400000") = 52534.500` |
| DMUL(x,y) | Multiplication. Même remarque que pour DPLUS. |
| DDIV(x,y) | Division. Même remarque que pour DPLUS. |

**OPERATIONS SUR NOMBRES DONNES SOUS FORME DE CHAINE DE CARACTERES**

| | |
|---|---|
| DCPLUS(x,y) | Addition. Les arguments sont caractères, le résultat est caractère. |
| DCMINUS(x,y) | Soustraction. Même remarque que pour DCPLUS. |
| DCMUL(x,y) | Multiplication. Même remarque que pour DCPLUS. |
| DCDIV(x,y) | Division. Même remarque que pour DCPLUS. |

**FONCTIONS MATHS**

| | |
|---|---|
| ABS(x) | retourne la valeur absolue de *x*. |
| EXP(x) | retourne l'exponentiel de *x*. |
| INT(x) | retourne la valeur entière de *x*. |
| LOG(x) | retourne le logarithme naturel de *x*. |
| LOG10(x) | retourne le logarithme base 10 de *x*. |
| MAX(x1,x2,...xn) | retourne le maximum parmis les argument *x1...xn*. Si un des arguments est un vecteur, la maximisation est faite pour chacun de ses élements et le résultat est un vecteur<br>**Ex:** `[1](:,:)=max([1](:,:),maxval)` |
| MIN(x1,x2,...xn) | retourne le minimum parmis les argument *x1...xn*. Si un des arguments est un vecteur, la minimisation est faite pour chacun de ses élements et le résultat est un vecteur |
| MOD(x,y) | retourne le modulo de *x* par *y*. |
| NINT(x) | retourne la plus proche valeur entière de *x*. |
| RAND(flag) | retourne un nombre aléatoire. Pour $flag = 0$ on a le prochain nombre, pour $flag = 1$ on redémarre le générateur et pour $flag = n$ on initialise le générateur en fonction de "n". |
| SQRT(x) | retourne la racine carrée de *x*. |

**FONCTIONS VECTEURS**

| | |
|---|---|
| VEC(arg1,arg2,...argn) | fabrique un vecteur en joignant tous les arguments.<br>**Ex:** `ny(1:5)=vec(1,2,3,nx(1:2))` |
| SETV(start,stop[,step]) | remplis un vecteur selon les paramètres de boucle.<br>**Ex:** `setv(1,5) = 1,2,3,4,5` |
| EXPAND(val\|vec, mul [,size]) | expand chaque position d'un vecteur ou d'une matrice. Le facteur multiplicatif est donné par "mul". Si "mul" est donné négatif, cette fonction fait une compression des pixels (moyenne). Si l'on travaille en 2D (matrice), il faut préciser la nombre de colonnes avec "size".<br>**Ex:** `[2](:,:)=expand([1](:,:),2,nx)` |

INMAXV(vec) . . . . . retourne l'index du premier plus grand élément de *vec.*

**Ex:** `inmaxv(vec(2,4,6,8,4,2)) = 4.0000000`

INMINV(vec) . . . . . retourne l'index du premier plus petit élément de *vec.*

**Ex:** `inminv(vec(2,4,6,8,4,2)) = 1.0000000`

MINV(vec) . . . . . retourne le plus petit élément de *vec.*

**Ex:** `minv(vec(2,4,6,8,4,2)) = 2.0000000`

MAXV(vec) . . . . . retourne le plus grand élément de *vec.*

**Ex:** `maxv(vec(2,4,6,8,4,2)) = 8.0000000`

SUM(vec) . . . . . retourne la somme des valeur contenus dans *vec.*

**Ex:** `sum(vec(2,4,6,8,4,2)) = 26.000000`

MEDIAN(vec) . . . . . retourne le median (nag :g07daf)

MEDDEV(vec) . . . . . retourne le sigma median (nag :g07daf)

STDDEV(vec) . . . . . retourne le sigma (nag :g07daf)

XBARY([mat]) . . . . . retourne le x barycentre

YBARY() . . . . . retourne le y barycentre (il faut utilise xbary(mat) avant)

CLEAN([mat],seuil,size,valrep) . . . . . patch une matrice centree de cote=2*size+1 contenant valrep pour chaque poinr plus eleve que seuil

**Ex:** `pour les cosmiques : clean([1],60000,1,0) defaut : 60000,1,0`

**FONCTIONS TRIGOS**

COS(alpha) . . . . . retourne le cosinus de *alpha* donné en radian.

COSD(alpha) . . . . . retourne le cosinus de *alpha* donné en degré.

SIN(alpha) . . . . . retourne le sinus de *alpha* donné en radian.

SIND(alpha) . . . . . retourne le sinus de *alpha* donné en degré.
TAN(alpha) . . . . . retourne la tangente de *alpha* donné en radian.
TAND(alpha) . . . . . retourne la tangente de *alpha* donné en degré.
ACOS(x) . . . . . retourne l'arc cosinus de *x* en radian.
ACOSD(x) . . . . . retourne l'arc cosinus de *x* en degré.
ASIN(x) . . . . . retourne l'arc sinus de *x* en radian.
ASIND(x) . . . . . retourne l'arc sinus de *x* en degré.
ATAN(x) . . . . . retourne l'arc tangente de *x* en radian.
ATAND(x) . . . . . retourne l'arc tangente de *x* en degré.
ATAN2(y,x) . . . . . retourne l'arc tangente selon *x,y* en radian.
ATAN2D(y,x) . . . . . retourne l'arc tangente selon *x,y* en degré.
DEG2RAD(x) . . . . . conversion degrés radians.
RAD2DEG(x) . . . . . conversion radians degrés.

**FONCTIONS LEXIQUES**

ATOR(str) . . . . . retourne le nombre écrit dans *str.*

**Ex:** `ator("12.2") = 12.200000`

CHAR(code) . . . . . retourne le caractère ASCII donné par *code.*

**Ex:** `char(48) = "0"`

FORMAT(val|vec|str, fmt) . . . . . retourne *val*|*vec*|*str* formatté le format *fmt.* Le format suit la syntaxe fortran.

**Ex:** `format(nx(1:3),"3i4") = " 100 100 100"`

ICHAR(str) . . . . . retourne le code ASCII du premier caractère de la chaîne *str.*

**Ex:** `char("abc") = 97.000000`

COMPAC(str) . . . . . retourne *str* sans espace.

**Ex:** `compac(" a bb ccc ") = "abbccc"`

STRIP(str) . . . . . supprime les blancs, les 0 non significatifs et le point non significatif sur une chaine numérique.

**Ex:** `strip(" 123.00000 ") = "123"`

INDEX(str1,str2) . . . . . retourne la position de *str2* dans *str1*.

**Ex:** `index("file.ext",".") = 5.0000000`

FIND(str1,str2) . . . . . retourne 1 si *str2* existe dans *str1*.

**Ex:** `find("file.ext","ext") = 1.0000000`

ITOA(val) . . . . . retourne le nombre *val* formatté en entier sans blanc.

**Ex:** `itoa(1234.56) = "1234"`

LCAT(str1, ..., strN) . . . . . retourne la concaténation des tous les arguments (type caratère) .

**Ex:** `lcat("file",".","ext") = "file.ext"`

LSCAT(str1,str2,...strN) . . . . . retourne la concaténation des tous les arguments (type caratère) en placant un espace entre chaque argument.

**Ex:** `lscat("filea","fileb","filec") = "filea fileb filec"`

LOWER(str) . . . . . retourne *str* mis en minuscule.

**Ex:** `lower("AbCdEF-1234") = "abcdef-1234"`

UPPER(str) . . . . . retourne *str* mis en majuscule.

**Ex:** `upper("AbCdEF-1234") = "ABCDEF-1234"`

PAD(str) . . . . . retourne *str* sans les espaces en début et fin de chaîne et avec une seule barre de soulignement ("_") pour chaque suites d'espaces entres les mots.

**Ex:** `pad(" a bb ccc ") = "a_bb_ccc"`

PARSE(str,No) ..... retourne le champ *No* de *str*, avec pour séparateur le premier caractère de *str*.

**Ex:** `parse("/ccd2/t4/beta",3) = "beta"`

SPARSE(str,No) ..... retourne le champ *No* de *str*, avec pour séparateur l'espace.

**Ex:** `sparse(" a bb ccc ",2) = "bb"`

TPARSE(str,No) ..... retourne le champ *No* de *str*, avec pour séparateur le tabulateur.

TRIM(str) ..... retourne *str* sans les espaces avant et apres le texte utile.

**Ex:** `trim(" a bb ccc ") = "a bb ccc"`

LTRIM(str) ..... retourne *str* sans les espaces à gauche du texte utile.

**Ex:** `ltrim(" a bb ccc ") = "a bb ccc "`

RTRIM(str) ..... retourne *str* sans les espaces à droite du texte utile.

**Ex:** `rtrim(" a bb ccc ") = " a bb ccc"`

ASSIGN(str[,prefix]) ..... assignation de variables selon une chaine formatée contenant une suite variable, contenu. On prefixe le nom de la variable si prefix est donnée

**Ex:** `assign("/texp/100/sn/20","e.")`

LEN(str) ..... retourne le nombre de caractères dans *str*.

**Ex:** `len("file.ext") = 8.0000000`

**OPERATEURS LOGIQUES**

.AND. ..... ET logique.

**Ex:** `if (flag.and.test) then`

.NOT. . . . . négation de la condition.

**Ex:** `if (.not.flag) then`

.OR. . . . . OU logique.
.EQ. . . . . égalité numérique.
.NE. . . . . non égalité numérique.
.GE. . . . . plus grand ou égal.
.GT. . . . . strictement plus grand.
.LE. . . . . plus petit ou égal.
.LT. . . . . strictement plus petit.

**FONCTIONS LOGIQUES**

AND(a,b) . . . . . retourne le résultat du 'and' binaire

**Ex:** `and(7,2) = 2.0000000`

OR(a,b) . . . . . retourne le résultat du 'or' binaire

**Ex:** `or(7,2) = 7.0000000`

XOR(a,b) . . . . . retourne le résultat du 'xor' binaire

**Ex:** `xor(7,2) = 5.0000000`

NOT(a) . . . . . retourne le résultat du 'not' binaire

**Ex:** `not(7) = -8.0000000`

LEQ(str1,str2) . . . . . retourne 1 si *str1* $=$ *str2*.
LCEQ(str1,str2) . . . . . retourne 1 si *str1* $=$ *str2*. Cette fonction est insensible aux minuscules et aux majuscules
LNE(str1,str2) . . . . . retourne 1 si *str1* $\neq$ *str2*.
LGE(str1,str2) . . . . . retourne 1 si *str1* $\geq$ *str2*.

LGT(str1,str2) ..... retourne 1 si *str1* $>$ *str2*.

LLE(str1,str2) ..... retourne 1 si *str1* $\leq$ *str2*.

LLT(str1,str2) ..... retourne 1 si *str1* $<$ *str2*.

LNO(str) ..... retourne 1 si *str* vaut "n", "no" ou "non".

LYES(str) ..... retourne 1 si *str* vaut "y", "yes", "o" ou "oui" .

ISNUM(xxx) ..... retourne 1 si "xxx" est un nombre (lexicalement parlant).

**Ex:** `isnum("1.2") = 1.0000000`

ISVNUM(xxx) ..... retourne 1 si "xxx" est une variable numérique.

**Ex:** `isvnum(nx(2)) = 1.0000000`

EXIST(fichier) ..... retourne 1 si le *fichier* existe.

**Ex:** `exist(lcat(getenv("HOME"),"/.login")) = 1.0`

**FONCTIONS ASTRO**

TIME() ..... donne l'heure de la machine (heures décimales).

**Ex:** `time() = 11.657222`

TS([tcl][,[j][,[m][,[a]]]]) ..... donne l'heure sidérale. Les arguments optionnels (heure, jour, mois, année) sont pris par défaut à l'instant courant, l'heure est donnée en heure décimale. Les variables LONGITU et FUSEAU sont utilisées pour le calcul, elles donnent la longitude du lieu et le décalage horaire par rapport a Greenwich (compté positif vers l'est).

**Ex:** `ts() = 9.7042542`

JD([j][,[m][,[a]]]) ..... retourne le jour julien formatté (1x,f11.3,1x). Il est alculé à partir des arguments optionnels (jour,mois,année) pris par défaut à l'instant courant. Remarque : le jour peut être fractionnaire.

**Ex:** `jd() = " 2452534.500 "`

| | |
|---|---|
| JD0IN([j][,[m][,[a]]]) . . . . . | initialise le jour julien de référence (jd0) pour les calculs de précéssion et de nutation. Ce calcul est fait à partir des arguments optionnels (jour,mois,année) pris par défaut à l'instant courant. Remarque : le jour peut être fractionnaire. Cette fonction retourne toujours 0. |
| PRECIN([j][,[m][,[a]]]) . . . . . | initialise les variables internes pour le calcul de la précession. La précéssion se fera à la date donnée calculée à partir des arguments optionnels (jour,mois,année) pris par défaut à l'instant courant. Remarque : le jour peut être fractionnaire. Cette fonction retourne toujours 0. |
| APREC(alpha,delta) . . . . . | précessionne la coordonnée donnée en alpha. Les coordonnées sont données en degré décimaux. JDOIN() et PRECIN() doivent être utilisés au préalable. On utilise DPREC pour la précéssion en delta. |
| DPREC(alpha,delta) . . . . . | précessionne la coordonnée donnée en delta. Les coordonnées sont données en degré décimaux. JDOIN() et PRECIN() doivent être utilisés au préalable. On utilise APREC pour la précéssion en alpha. |
| NUTIN([j][,[m][,[a]]]) . . . . . | initialise les variables internes pour le calcul de la nutation. La nutation se fera à la date donnée calculée à partir des arguments optionnels (jour,mois,année) pris par défaut à l'instant courant.Remarque : le jour peut être fractionnaire. Cette fonction retourne toujours 0. |
| ANUT(alpha,delta) . . . . . | nute la coordonnée donnée en alpha. Les coordonnées sont données en degré décimaux. JDOIN() et NUTIN() doivent être utilisés au préalable. On utilise DNUT pour la nutation en delta. |
| DNUT(alpha,delta) . . . . . | nute la coordonnée donnée en delta Les coordonnées sont données en degré décimaux. JDOIN() et NUTIN() doivent être utilisés au préalable. On utilise ANUT pour la nutation en alpha. |
| AHDE2AZ(angle_horaire, delta) . . . . . | calcul l'azimut. Les coordonnées sont données en degré décimaux. La variable LATITU est utilisée, elle donne la latitude du lieu. Convention : azimut=0 au Nord, croissant vers l'Est |
| AHDE2EL(angle_horaire, delta) . . . . . | calcul l'élévation. Les coordonnées sont données en degré décimaux. La variable LATITU est utilisée, elle donne la latitude du lieu. |

AZEL2DE(azimut, élévation) . . . . . calcul delta. Les coordonnées sont données en degré décimaux. La variable LATITU est utilisée, elle donne la latitude du lieu. Convention : azimut=0 au Nord, croissant vers l'Est

AZEL2AH(azimut, élévation) . . . . . calcul angle horaire. Les argument sont donnés en degré décimaux. La variable LATITU est utilisée, elle donne la latitude du lieu. Convention : azimut=0 au Nord, croissant vers l'Est

FZ(dist_zénitale) . . . . . calcul de la masse d'air (voir slalib).

REFCO(tdk, pmb, rh, [wl], [eps], [tlr], [hm], [phi]) . . . . . calcul des parametres AREFRA et AREFRA du modèle de réfraction atmospherique. AREFRA et AREFRA sont des variables du bloc de données.

REFRAC(dist_zénitale) . . . . . calcul de la réfraction atmosphérique pour La Silla.

ROTPOS(alpha,delta) . . . . . angle du derotateur moment courant. Remarque : utilise longit et latitu.

ROTVEL(alpha,delta) . . . . . vitesse du derotateur moment courant [deg/s]. Remarque : utilise longit et latitu.

ROTHPOS(ah,delta) . . . . . angle du derotateur. Remarque : utilise longit et latitu.

ROTHVEL(ah,delta) . . . . . vitesse du derotateur [deg/s]. Remarque : utilise longit et latitu.

**ANGLES ET HEURES**

ANGLE("angle") . . . . . retourne la valeur numérique d'un angle donné sous forme de chaîne de caractères. L'angle peut être donné sous n'importe quelle forme. Si aucune unité n'est donnée, cette fonction comprend des degrés.

**Ex:** `"3D12", "15 degre", "12h 15.4", "123sec", "12:23:22", "12h23m12.2"`

DDTOD(angle) . . . . . formatte *angle* en : "SDDDd MMm SSs ".

DDTOD2(angle) . . . . . formatte *angle* en : "SDDD:MM:SS ".

HTOHD("heure") . . . . . retourne la valeur numérique d'une heure donnée sous forme de chaîne de caractères. L'heure peut être donnée sous n'importe quelle forme. Si aucune unité n'est donnée, cette fonction comprend des heures. Exemple de format : "3D12", "15 degre", "12h 15.4", "123sec", "12:23:22", "12h23m12.2"

**Ex:** `angle("12 12") = 12.200000`

HDTOH(heure) . . . . . formatte *heure* en : "HHh MMm SS.Ss ".

**Ex:** `hdtoh(1.234567) = " 1h 14m 04.4s "`

HDTOH2(heure) . . . . . formatte *heure* en : "HH:MM:SS.S ".

**Ex:** `hdtoh2(1.234567) = " 1:14:04.4 "`

**MATRICES**

MAXSIZ(No_mat) . . . . . retourne la taille actuelle ou maximum (dans le cas d'une matrice en mémoire partagée) de la matrice No_mat.

NBCOU(No_mat) . . . . . retourne le nombre de couche pour la matrice *No_mat.*

**Ex:** `nbcou(1) = 20.000000`

NBMAT() . . . . . retourne le nombre de matrices accessibles.

**Ex:** `nbmat() = 22.000000`

NBPIX() . . . . . retourne le nombre de pixels alloués par l'ensemble des matrices en mémoire partagée.

NOCOU(str|num) . . . . . retourne le No de couche d'un No de matrice.

**Ex:** `nocou([1,3]) = 3.0000000`

NOMAT(str|num) . . . . . retourne le No de matrice d'un No de matrice.

**Ex:** `nomat([1,3]) = 1.0000000`

**FIT**

FITGAU(vec) . . . . . Fit une gaussienne sur les données de *vec.* retourne 4 paramètres plus le résidu, dans un vecteur 5 positions.

GENGAU(vec, nb_points) . . . . . génère une gaussienne selon les 4 premières valeurs du vecteur *vec.* (1=max, 2=centre, 3=forme, 4=background

**Ex:** `gengau(fitgau([1](34:46,1)),13)`

LSFIT(No_de_matrice) ..... résoud m équations à n inconnues. Travaille sur une matrice dont chaque ligne représente une équation (résultat en dernière colonne). Retourne un vecteur de taille (NX-1).

POLYF(xvec, yvec, degre [,pvec]).... fit polynômial à une inconnue de degré *degré*. Les points peuvent être pondéré par le vecteur pvec (attention pvec(i)>0). Retourne un vecteur de taille (DEGRE+1). Remarque : si *degré* est donné négatif, on donne à l'écran des messages concernant les résidus.

GENF(xvec, polyvec)..... fabrique f(xvec) avec polyvec comme coefficient du polynôme. Le calcul est effectué en double précision. Polyvec est vecteur de coefficents donné dans l'ordre des puissance à partir du degré zero.

**DIVERS**

APPNAME()..... retourne le nom de l'application.

**Ex:** `appname() = "inter"`

ENVDEF(evar)..... retourne 1 si la variable d'environnement est definie, 0 sinon.

**Ex:** `envdef("HOME") = 1`

GETENV(evar)..... retourne le contenu de la variable d'environnement *evar*.

**Ex:** `getenv("HOME") = "/home/albert"`

SETTENV("evar=val")..... assigne une variable d'environnement *evar*.

**Ex:** `stat=setenv("CAT=file.rdb")`

GETLU()..... retourne une unité logique inutilisée.

**Ex:** `local unit=getlu()`

SLEEP(sec) ..... Suspend le process durant un temps donné en seconde

**Ex:** `stat=sleep(1.5)`

SYSTEM(cmd) ..... retourne le résultat fournis par la commande system *cmd.*

**Ex:** `system("hostname") = "ouranos"`

VERS() ..... retourne la date de la dernière compilation d'Inter.

**Ex:** `vers() = "inter_SunOS_5.8 - Fri Sep 13 07 :38 :08 CLT 2002"`

USER() ..... retourne le No d'utilisateur.

**Ex:** `user() = 4730.0000`

GROUP() ..... retourne le numéro de groupe.

**Ex:** `group() = 4000.0000`

$(string) ..... retourne le contenu de la variable nommée dans string. Attention, en phase de compilation on ne sait pas si on fait une indirection sur une variable numérique ou caractère. En mode compilation et pour en tout cas autoriser les doubles indirections, Inter pose le résultat comme type caratère. Donc si l'on veut assigner une variable numérique il faut le faire en 2 temps : local x ; x=$(var) ; x=x+n

**Ex:** `$("nx") = 100.00000`

RM ?(string) ..... Cette fonction supprime les points d'interrogation en fin de ligne. (utile pour les commandes @@ et @@@ ...).

**Ex:** `rm?("@qq?? 1 2???") = "@qq?? 1 2"`

**COMMUNICATION**

CLEARSV() ..... Efface les flags d'erreur et de status ainsi que les code et message d'erreur ainsi que le texte de la commande courante dans le bloc de communication. Cette fonction permet de ne pas récupérer d'erreur trainant d'une précédente commande. Si on pose que l'on part d'une situation stable.

CONNECT() ..... Se (re)connecte sur les sémaphores du serveur courant s'ils ont été tués.

SELECT(server) ..... Séléctionne le serveur sur lequel vont travailler toutes les fonctions de communication. Rem : le mot–clé "myself" selectionne son propre bloc de communication

**Ex:** `i=select("ccd")`

SHMINIT() ..... initialise le bloc de communication en le vidant.

SHMGET(key) ..... lit un paramètre référencié par *key* dans le bloc de communication.

**Ex:** `nx=ator(shmget("XSIZE"))`

SHMPUT(key, content) ..... écrit un paramètre référencié par *key* et son contenu *content* dans le bloc de communication et initialise le bloc de communication. Après cette opération, le bloc ne contient qu'un paramètre.

**Ex:** `dummy=SHMPUT("XSIZE",itoa(nx))`

SHMADD(key, content) ..... ajoute un paramètre référencié par *key* et son contenu *content*, dans le bloc de communication .

SHMSHOW() ..... visualise le bloc de communication à l'écran.

SHMPCOD() ..... met le code d'erreur, l'erreur et le message d'erreur en shared memory.

SHMWAIT([timeout]) ..... suspend le client jusqu'à que le serveur soit prêt. (décrémente le sémaphore #0). Bloque le client si le serveur n'est pas prêt. Retourne un status différent de zéro si il y a eu une erreur sur la commande **précédente**. Dans ce cas (1=erreur sur serveur, 2=serveur déconnecté, 3=<CTRL>–C sur serveur, 101=SIGHUP, 102=SIGINT (<CTRL>–C), 113=SIGPIPE, 114=SIGALRM (timeout)) aucun message n'est affiché et l'erreur interne n'est pas activée

SHMCONT() ..... indique au serveur d'exécuter la commande placée dans "COMMAND" et lui signale de rendre la main. (incrémente le sémaphore #1)Le serveur se libérera par lui-même à la fin de la commande.

SHMACK() ..... indique au serveur d'exécuter la commande placée dans "COMMAND" et lui signale de ne pas rendre la main. (pose le sémaphore #2 à 1 et incrémente le sémaphore #1).

SHMWACK([timeout]) ..... attend que le serveur ait finis (après un SHMACK()). (Attend que le sémaphore #2 soit égal à zéro). Suspend le client tant que sa tâche n'est pas terminée. Retourne un status différent de zéro si il y a eu une erreur sur la commande en cours. En cas d'erreur (1=erreur sur serveur, 2=serveur déconnecté, 3=<CTRL>–C sur serveur, 101=SIGHUP, 102=SIGINT (<CTRL>–C), 113=SIGPIPE, 114=SIGALRM (timeout)) une message est affiché mais l'erreur interne n'est pas activée

SHMGACK() ..... retourne la valeur du flag ackno.

SHMFREE() ..... Rend la main (après un SHMWACK()). (incrémente le sémaphore #0) Apres cette commande, le serveur est accessible pour un autre client

SHMCMD(cmd [,to]) ..... Suspend le client, envoie une commande et libère le client. retourne un status différent de zéro si il y a eu une erreur sur la commande **précédente**. Dans ce cas (1=erreur sur serveur, 2=serveur déconnecté, 3=<CTRL>–C sur serveur, 101=SIGHUP, 102=SIGINT (<CTRL>–C), 113=SIGPIPE, 114=SIGALRM (timeout)) aucun message n'est affiché et l'erreur interne n'est pas activée

**Ex:** `i=shmcmd("@qq")`

SHMCMDW(cmd [,to_wait [,to_cmd]]) ..... Suspend le client, envoie une commande et attend la fin de la commande pour libérer le client. retourne un status différent de zéro si il y a eu une erreur sur la commande en cours. En cas d'erreur (1=erreur sur serveur, 2=serveur déconnecté, 3=<CTRL>–C sur serveur, 101=SIGHUP, 102=SIGINT (<CTRL>–C), 113=SIGPIPE, 114=SIGALRM (timeout)) une message est affiché mais l'erreur interne n'est pas activée

SHOWSEL() ..... affiche le nom de tous les serveurs possible ainsi que le serveur actuellement séléctionné.

**Ex:** `i=showsel()`

| | |
|---|---|
| SRVEXIS() | retourne 1 si le serveur existe. |
| SRVWAIT() | indique si le client a la main sur le serveur courant. Permet de traiter les erreurs dues aux time-out.<br>**Ex:** `if srvwait() i=shmfree()` |
| SRVWORK() | indique si le serveur est en train de travailler. retourne 0 s'il est sur le prompt (independement du fait qu'il soit ou non réservé. Retourne -1 s'il est occupé par un autre client, retourne +1 s'il est occupé par nous même. |
| SIGNAL(signal) | Envoie un signal au serveur. retourne le status de la fonction kill ou -1 si le serveur est inexistantex : 2=CTRL–C (voir "man -s5 signal") |
| SHMNCNT() | retourne le nombre de client en attente sur le serveur courant. |
| SHMZERO() | met le semaphore zéro à 1 (reset). |
| SHMZCNT() | retourne le nombre de process en attente de zero sur sem No. |
| SHMSRV() | retourne 1 si Inter a été lancé en mode serveur. |
| SHMCLI([name]) | retourne 1 si Inter a été lancé en mode client. Si name est précisé, alors retourne 1 si Inter est client de *name* |
| FETCH() | assigne les variable Inter selon les couples donnés dans le bloc de communication. |
| SHMGCOD() | retourne le code d'erreur (ascii). |
| SHMGERR() | retourne le code d'erreur (numérique). |
| SHMGID() | retourne l'identificateur du bloc de mémoire partagée. |
| SHMGMES() | retourne le texte du message d'erreur. |
| SHMGSRV() | retourne le nom du serveur courant. |
| SHMGSTA() | retourne le status d'erreur. |
| SEMAGET(No) | retourne la valeur du sémaphore *No.* |
| SEMASET(No, val) | Met le sémaphore *No* à *val.* |

**FICHIERS FITS**

| | |
|---|---|
| RFITS(file, key) | Lit un keyword *key* du fichier *file.* |

**LOGBOOK**

| | |
|---|---|
| INILOG(host) | Initialise une connection sur le logbook de la machine *host.* Retourne 0 en cas de succès ou -1 si la connection est impossible. Dans ce dernier cas, Inter ne voit pas d'erreur. L'initialisation peut se faire de manière externe à Inter en envoyant le signal SIGUP à Inter. |

**Ex:** `unix> kill -1 <pid_inter>`

**DEBUG**

ADD(variable) ..... retourne l'adresse de la *variable* numérique dans les tableaux d'Inter.

DIM(variable) ..... retourne la dimension d'une *variable.*

**TEMPÉRATURES CCD**

TEMP(flag) ..... retourne certaines températures du CCD. Cette fonction demande un argument (entre 1 et 4) et retourne une des température mesurée sur le sytème CCD en degré Kelvin. Pour $flag = 1$ on a le réservoir d'azote, pour $flag = 2$ on a ccd1, pour $flag = 3$ on a ccd2 et pour $flag = 4$ on a la paroi extérieure.

**Ex:** `write "T boitier = " TEMP(1)-27`

**COORDONNÉES**

CAL(x,y) ..... convertion de coordonnées cartésiennes en coordonnées équatorialles. retourne <alpha>. Cette conversion utilise des variables prefixées "ALI" ou "GSC". Elle permet de retrouver le coordonnée équatoriale alpha d'une position pointée sur l'afficheur après avoir utilisé la commande ALIGN.

**Ex:** `alpha=CAL(x,y)`

CDE(x,y) ..... convertion de coordonnées cartésiennes en coordonnées équatorialles. retourne <delta>. Cette conversion utilise des variables prefixées "ALI" ou "GSC". Elle permet de retrouver la coordonnée équatoriale delta d'une position pointée sur l'afficheur après avoir utilisé la commande ALIGN.

**Ex:** `delta=CDE(x,y)`

# Chapitre 2

# COMMANDES

## 2.1 ?

Affichage d'un fichier help.

---

**SYNTAXES:**

?<commande>

?<sujet>

---

**PARAMETRES:**

| | |
|---|---|
| <**commande**> | commande Inter |
| <**sujet**> | sujet d'interêt, ex : ?fonctions |

---

**DESCRIPTION:**

Les fichiers de help sont stockés dans un directory spécifique avec une extension ".hlp". La variable DIRHLP donne le chemin d'accès à ce directory. Ces fichiers sont affichés au moyen du "more" Unix et donc les commandes s'y rapportant sont accessibles.

---

**REMARQUE:**

On peut voir les commandes à disposition avec : SHOW /COMMAND

## 2.2 $

Exécute une commande système

---

**SYNTAXES:**

$ <commande>

$ <variable>

---

**PARAMETRES:**

| | |
|---|---|
| <**commande**> | commande Unix |
| <**variable**> | variable Inter contenant une commande Unix |

---

**DESCRIPTION:**

La commande peut être écrite dans une variable (au moyen de WRITE /KEYW)

---

**EXEMPLES:**

```
$ls -l *.f"

WRITE /KEYW=cmd "ls -l for" i ".dat" /FMT=a,i3.3,a
$cmd
```

---

**REMARQUE:**

Les commandes sont exécutées dans un shell. C'est à l'intérieur de celui-ci que sont exécutées les commandes avant de revenir dans l'interpréteur. Par exemple, la commande UNIX "cd" n'agit que pour l'environnement du shell et n'influencent aucunement celui de l'interpréteur. Le seul moyen d'accéder un fichier dans un autre directory est de préciser le nom complet de celui-ci.

## 2.3 @

Compile puis exécute une procédure contenue dans un fichier

---

**SYNTAXE:**

@<fichier> [<P1> [<P2> [...]]]

---

**PARAMETRES:**

| | |
|---|---|
| <**fichier**> | Nom d'un fichier contenant des commandes |
| <**Pn**> | argument recupérable par les commandes |

---

**DESCRIPTION:**

La procédure peut attendre jusqu'à 9 paramètres qui remplaceront toutes les termes de type {<n>}, <n> étant le No du paramètre concerné. Le remplacement a lieu à la compilation. Les paramètres absent peuvent êtres crées dans la procédure avec la commande DEFAULT. Un paramètre noté ' ?' force l'utilisation du défaut.

L'exécution d'une procédure se fait en deux temps.

1. Le fichier est entièrement lu et toutes les erreurs de syntaxe sont affichées.
2. S'il n'y a eu aucune erreur, la procédure est exécute. Sinon on retourne immédiatement en mode interpréteur.

Une procédure peut être relancée sans la recompiler (voir la commande EXECUTE)

---

**REMARQUE:**

Les fichiers de procédure ont l'extension ".prc" par défaut. Si une autre extension est utilisée, il faut la préciser.

---

**EXEMPLE:**

```
! Procedure de demonstration
!
! affiche une variable globale et l'incremente.
!
default 1=i
show {1}
{1}={1}+1
```

## 2.4 .

Affiche les variables du bloc de données utilisées par une commande Inter

---

**SYNTAXE:**

.<commande>

---

**PARAMETRE:**

<**commande**> nom d'une commande Inter

## 2.5 ACTION

Définis la stratégie à adopter lorsqu'une interruption survient durant l'exécution d'une procédure

---

**SYNTAXE:**

ACTION <type d'interruption> <opération> [<option>]

---

**PARAMETRES:**

<**type d'interruption**> Mot clé désignant le type d'interruption.

<**opération**> Mot clé indiquant le comportement de cette commande.

<**option**> Mot clé permettant le modifier le comportement de <opération>.

---

**DESCRIPTION:**

Actuellement la seule interruption gérée est le <CTRL>–C que l'on écrit : "CTRLC".

**Remarque importante : dans tout les cas d'interruption, la commande en cours se termine de façon normale et c'est seulement après que l'opération désirée est effectuée. Si la commande en cours dure dix minutes, les dix minutes devront s'écouler avant la prise de contrôle.**

Les opérations possibles sont :

**RETURN** retourne dans la routine appelante. Si cette dernière a également demandé un <RETURN> sur interruption alors on descend à nouveau d'un niveau et ainsi de suite jusqu'a la routine ne demandant pas de contrôle.

**NORETURN** supprime la gestion du <RETURN> sur interruption pour la routine en cours.

**ENDDO** finis la boucle DO–ENDDO en cours (exécute toutes les instructions jusqu'au ENDDO) en préservant l'indice de boucle sur sa dernière valeur. Les boucles incluant cette boucle seront aussi terminées si elles en ont fait la demande.

L'option "GENERAL" effectue la même demande sur toutes les boucles plus internes.

Ex: prompt> ACTION CTRLC ENDDO GENERAL

**NOENDDO** supprime la gestion du <ENDDO> sur interruption.

L'option "GENERAL" supprime la demande sur toutes les boucles plus internes.

**CALL** effectue un call directement après la commande courante puis continue la séquence normale. Le nom de la subroutine ainsi que les arguments sont donnés à la suite. ATTENTION aux variables locales, les arguments du CALL doivent être valides dans la procédure où

| | |
|---|---|
| | survient l'interruption. Un réenregistrement d'un CALL supprime le précédent.<br>Ex: prompt> ACTION CTRLC CALL HANDLER I |
| **NOCALL** | supprime l'effet de la gestion de l'interruption par CALL. |
| **HOLD** | suspend l'action de l'interruption, les signaux ne sont pas délivré au process (voir sighold()). Les signaux qui arrivent durant une séquence non interruptible sont délivrés avec l'opération <RELSE> ou annulés avec l'opération <CLEAR>.<br>Ex: prompt> ACTION CTRLC HOLD |
| **RELSE** | soumets les signaux arrivés durant une séquence non interruptible au process (voir sigrelse()). |
| **CLEAR** | annule les signaux arrivés durant une séquence non interruptible au process. |
| **PAUSE** | suspend le process jusqu'a l'arrivée du signal. |

Remarque:

*Les system calls sont interruptibles. Donc lorsque qu'un signal survient durant une attente (communication interprocess) le contrôle revient à Inter, et s'il survient durant un read() ou un write(), l'operation n'a pas lieu.*

## 2.6 AFF

Diverses opérations de contrôle sur l'afficheur.

---

**SYNTAXE:**

AFF /qualificateur

---

**DESCRIPTION:**

L'afficheur travaille en coordonnées world. Par défaut l'origine des ces coordonnées est $< 0; 0 >$ avec un pas de 1. dans les deux axes. On modifie le système de coordonnée avec /COORD.

L'afficheur doit être connecté au préalable pour pouvoir interagir avec Inter (voir AFF /CLIENT).

---

**QUALIFICATEURS A DISPOSITION:**

**/CLIENT**

Se connecte sur un afficheur libre ou en crée un si besoin. L'afficheur doit avoir été lancer par un Inter de même nom.

**/NOCATCH**

Lance un afficheur et s'y connecte

**/RESOLUTION=<resolution>**

Fixe la résolution d'affichage en bit au moment du lancement Compris entre 1 et 7 pour 2 à 128 couleurs

**/SIZE[=<x>[,<y>[,<win_x>[,<win_y>]]**

Donne les dimensions de l'image et de la fenêtre. Si les deux premiers termes sont nuls ou absents, la fenêtre prend la taille du ccd ($416 * 578$). S'ils sont négatifs, la taille de l'image est conservée, s'il sont positifs, l'image courante est perdue et la nouvelle image a la taille désirée. <win_x> et <win_y> donnent la taille de la fenêtre.

**/ZSIZE=<x>,<y>**

Donne les dimensions de la fenêtre zoom en pixel.

**/POS[=<x>[,<y>]]**

Donne la position de la fenêtre image sur l'écran (défaut 0 ;0)

**/ZPOS[=<x>[,<y>]]**

Donne la position de la fenêtre zoom sur l'écran (défaut 0 ;0)

**/REDISPLAY**

Rafraîchit l'image (comme le boutton REDSIPLAY).

**/CLEAR**

Efface la fenêtre et les données.

**/QUIT**

Tue l'afficheur.

**/CUTS=<lcut>,<hcut>**

Envoie les nouveaux cuts et réaffiche l'image.

**/LUT=<No lut>**

Place la lut selon son Numéro d'ordre (la première a le Numéro 0)

**/ITT=<No itt>**

Place la itt selon son Numéro d'ordre (le premier a le Numéro 0)

**/ZOOM=<x_world>,<y_world>[,<zoom_fact>]**

Zoom a l'endroit spécifié selon le facteur spécifié ou courant. ! ! ATTENTION la zone doit être affichée à l'écran au moment du zoom ! !.

**/SZOOM=<x_world>,<y_world>[,<zoom_fact>]**

Identique à /ZOOM mais zoom également les symboles

**/COORD[=<xstart>[,<ystart>[,<xstep>[,<ystep>]]]]**

Donne les coordonnées world de l'image.

**/CLOSE**

Met l'afficheur sous forme d'icône, le tableau de commande ainsi qu l'image disparaissent.

**/OPEN**

Ouvre l'icône pour afficher le tableau de commande ainsi qu l'image.

**/HIDE**

Cache l'image. Elle peut être réaffichée avec "/SHOW" ou par le bouton "REDISPLAY".

**/SHOW**

Réaffiche l'image cachée.

**/ZHIDE**

Cache l'image du zoom. Elle peut être réaffichée avec "/ZSHOW".

**/ZSHOW**

Réaffiche l'image du zoom cachée.

**/VGOP=<n>**

Niveau de verbosité du protocole de communication (0 à 9).

---

**REMARQUES:**

Les options caractérisant aff et la fenêtres sont placées dans la variable CMDAFF. Typiquement :

CMDAFF="-v 3 -Wp 100 200"

Les options de aff sont :

**-w win_x** donne la taille de la fenêtre par défaut selon X.

**-w win_y** donne la taille de la fenêtre par défaut selon Y.

**-v n** Mode Verbose du protocole de communication (0 à 9)

---

## VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE

| Variables | Mode | Description |
|---|---|---|
| CMDAFF | I | options pour l'afficheur (aff) |

## 2.7 ALIAS

Opérations sur les alias.

---

**SYNTAXES:**

ALIAS

ALIAS /CLEAR

ALIAS <alias>

ALIAS <alias>=<commande>

ALIAS <alias>=<commande> /TEMPORAIRE

ALIAS <alias> /CLEAR

---

**PARAMETRES:**

| | |
|---|---|
| <**alias**> | Nom de l'alias |
| <**commande**> | Commande avec (facultativement) paramètres et qualificateurs |

---

**DESCRIPTION:**

L'alias est un terme de 4 lettres au maximum qui est remplacé par <commande> s'il est placé au début d'une ligne.

Si le terme <commande> n'est pas précisé alors la traduction de <alias> est affichée.

Sans qualificateur ni paramètres, ALIAS affiche tous les alias.

---

**QUALIFICATEURS A DISPOSITION:**

**/CLEAR**

Tue un ou tous les alias. Si un alias existe plusieurs fois (utilisation de /TEMPORAIRE) seul sa dernière définition est tuée.

Ex: PROMPT> ALIAS SETV /CLEAR

Ex: PROMPT> ALIAS /CLEAR

**/TEMPORAIRE**

Crée un alias sans supprimer l'alias précédent du même nom.

## 2.8 ALIGN

Permet de trouver les paramètres de l'équation de transformation de coordonnées entre deux systèmes cartésien.

---

**SYNTAXE:**

ALIGN <Nombre de couple de coordonnées> [/qualificateur]

---

**DESCRIPTION:**

L'équation de transformation de coordonnées donne : la rotation, les translations et les facteurs d'échelle dans les deux axes.

Le but principal de cette commande et de permettre de transformer les coordonnées cartésienne d'une position relevée sur l'afficheur en une coordonnée équatoriale <alpha,delta> au moyen des fonctions "CAL" et "CDE".

Dans ce cas, la marche à suivre, pour initialiser les variables utilisées par ces deux fonctions, est la suivante :

- Créer les tableaux dans lesquels vont êtres stockés les coordonnées d'objets appairés entre l'image du champ ("X1" et "Y1") et la carte GSC ("X2" et "Y2").
- Remplir ces tableaux avec "GET" du côté afficheur et "GSC /XYGET" du côté GSC.
- Lancer "ALIGN n", "n" étant le nombre de couple
- Transformer les positions <x,y> en coordonnées équatoriales avec les fonctions "CAL" et "CDE"

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **ALIMET** | methode utilisee pour le fit |
| **ALISIG** | cte pour le passage de coord aff en coord gsc |
| **ALIF1** | cte pour le passage de coord aff en coord gsc |
| **ALIF2** | cte pour le passage de coord aff en coord gsc |

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **ALIANG** | cte pour le passage de coord aff en coord gsc |
| **ALIX0** | cte pour le passage de coord aff en coord gsc |
| **ALIY0** | cte pour le passage de coord aff en coord gsc |
| **ALIF1** | Voir plus haut sous "VARIABLES INTERACTIVES PRINCIPALES" |
| **ALIF2** | Voir plus haut sous "VARIABLES INTERACTIVES PRINCIPALES" |

---

---

**QUALIFICATEURS A DISPOSITION:**

**/METHODE=<méthode>**

Permet de donner la methode sur la ligne de commande. La valeur de <méthode> est définie plus haut (voir description de la variable ALIMET). <méthode> est conservée dans la variable ALIMET.

Ex: ALIGN /MET="E"

**/FSIG=<sigma>**

Permet de donner la valeur de sigma sur la ligne de commande. La signification <sigma> est définie plus haut (voir description de la variable ALISIG). <sigma> est conservée dans la variable ALISIG.

**/X1=<nom> /Y1=<nom> /X2=<nom> /Y2=<nom>**

Donne le nom des tableaux d'entrée. Par défaut ils sont"X1", "Y1", "X2" et "Y2"

---

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| ALIANG | O | cte pour le passage de coord aff en coord gsc |
| ALIF1 | I/O | cte pour le passage de coord aff en coord gsc |
| ALIF2 | I/O | cte pour le passage de coord aff en coord gsc |
| ALIMET | I/O | methode utilisee pour le fit |
| ALISIG | I/O | cte pour le passage de coord aff en coord gsc |
| ALITR | O | cte pour le passage de coord aff en coord gsc |
| ALIX0 | O | cte pour le passage de coord aff en coord gsc |
| ALIY0 | O | cte pour le passage de coord aff en coord gsc |

## 2.9 CALL

Appel à une subroutine

---

**SYNTAXE:**

CALL <subroutine_name> [<argument_1> ...]

---

**PARAMETRES:**

<**subroutine_name**> nom symbolique (maximum 10 caractères)

<**argument_n**> argument d'appel

---

**DESCRIPTION:**

Lors de l'appel si un argument de type numérique est écrit <arg>=V, il est passé par valeur et ne pourra pas être modifié par la subroutine.

Les arguments déclarés dans la sous-routine sont considéré par défaut comme des variables numériques. Les arguments de type caractères devront être notés : <arg>=C.

---

**EXEMPLE:**

```
...
CALL TEST I J "/image/demo.bdf"
...

...
SUBROUTINE TEST A=V B NAME=C
...
```

## 2.10 CASE

Aiguillage selon la valeur d'une expression numérique ou lexique

---

**SYNTAXE:**

CASE <expression> OF

---

**DESCRIPTION:**

Le test est effectué sur des valeurs numériques entières ou sur des chaînes de caractères. Le branchement est effectué sur le label correspondant à la valeur. Le choix du type de brachement retenu (numérique ou lexique) est donné par le type de <expression>, c'est lui qui détermine si on utilise le label du case sous sa forme numérique ou sous sa forme caractère. Si aucun label ne correspond à <expression>, le branchement est fait sur le label nommé "DEFAULT".

Remarque, le label **DEFAULT** est optionnel.

Attention, Si deux labels se suivent, le premier ne génere aucune action !

---

**EXEMPLES:**

```
get
case affret of
1::
   call next
2::
   call previous
3::
   call affich
endcase
```

```
local str="abcdef"
do i=1,len(str)
   case str(1)(i:i) of
   a::
      write debut
   f::
      write fin
   default::
      write str(1)(i:i)
   endcase
enddo
```

## 2.11 CLEAR

Effacement de la fenêtre graphique ou réinitialisation si le travail est sur papier

---

**SYNTAXE:**

CLEAR

## 2.12 COMPILE

Se met en mode compilation pour l'entrée d'une procédure tapée au clavier.

---

**SYNTAXE:**

COMPILE

---

**DESCRIPTION:**

Apres "COMPILE", le mode compilation est initialisé et la dernière procédureest oubliée.

Ensuite, chaque ligne est analysée pour déterminer si la syntaxe est correcte et si les variables utilisées existent dans le bloc de données. S'il y a une erreur, la commande courante n'est pas compilée et on reste mode compilation, permettant ainsi de retaper la commande correctement. Pour exécuter la procédure une fois finie, on utilise la commande "EXECUTE". Si l'on désire sortir du mode compilation en annulant la procédure on utilise la commande "INTERPRETEUR".

---

**REMARQUE:**

COMPILE ne peut pas être utilisé dans un fichier de procédure.

## 2.13 CONTOUR

Dessine une matrice de travail en couleur, en courbes de niveaux ou en projections 3D avec ombrage. Cette commande permet aussi l'acquisition de positions et de données.

---

**SYNTAXES:**

CONTOUR [<No matrice>] [/qualif]

CONTOUR [<No matrice>] /MIN=<min> /MAX=<max> [/qualif]

CONTOUR [<No matrice>] /STAT [/qualif]

---

**DESCRIPTION:**

Contour déssine 'PLNNIV' courbes de niveau écartée d'un pas définit par 'PLDELT' depuis le bas si PLDELT(1) est positif ou depuis le haut s'il est négatif. L'écart peut être linéaire ou logarithmique. Les limites selon Z sont fixées par les variables PLZMIN et PLZMAX, celles ci peuvent être calculées automatiquement grâce au qualificateur /STAT. Le pas peut être calculé automatiquement grâce au qualificateur /AUTO. Ainsi pour une premier dessin, la commande "CONTOUR /STAT /AUTO" fournit immédiatement une représentation correcte de la matrice.

Les paramètres de dessin sont passés directement à CONTOUR. Ils peuvent être remplacés par certains qualificateurs qui modifient les valeurs dans le bloc de données. Cela permet de ne pas les préciser lors du prochain appel.

Le type de représentation est donné par la variable GTYP. Pour les projection 3D, la position de l'observateur est défini par les variables GPHI et GTHETA.

On peut représenter le système de coordonnées équatorial sur une représentation 2D.

Les caractères des labels, graduation et titres sont controllés par les variables GFLAG, GT* et GL*. CONTOUR peut travailler en mode window.

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **GDRIVE** | marge pour la feuille 0=pas <0=mm >0=marge |
| **GTYP** | type de graphique (1-7) |
| **PLDELT** | pas inter niveaux |
| **PLNNIV** | nb de niveaux |
| **PLZMIN** | plot parameter |
| **PLZMAX** | plot parameter |
| Pour la représentation en 3D | |
| **GPHI** | 3D - angle dans le plan X-Y |
| **GTHETA** | 3D - Angle vertical |

---

---

**QUALIFICATEURS A DISPOSITION:**

**/LIN[=<n>]**

Mode linéaire. L'écart entre les courbes de niveaux est posé à une valeur constante : PLDELT(1). Si <n> est précisé, il donne la valeur du nouvel écart et PLDELT(1) est mis à jour.

| | |
|---|---|
| <**+n**> | pose PLNNIV niveaux a partir de PLZMIN |
| <**–n**> | pose PLNNIV niveaux a partir de PLZMAX |

**/LOG[=<n1>,<n2>]**

Mode Logarithmique. L'écart entre les lignes de niveaux est défini par la fonction suivante : $\text{écart}_\text{i} =< \text{n1} > * < \text{n2} >^{\text{i}-1}$

Ce mode utilise par défaut PLDELT(1) et PLDELT(2). Si <n1> et <n2> sont précisés, ils sont utilisés et PLDELT(1) et PLDELT(2) sont mis à jour.

| | |
|---|---|
| <**+n1**> | pose PLNNIV niveaux a partir de PLZMIN |
| <**–n1**> | pose PLNNIV niveaux a partir de PLZMAX |

**/MIN=<n>**

Pose le niveau inférieur en remplaçant PLZMIN.
Met à jour PLZMIN

**/MAX=<n>**

Pose le niveau supérieur en remplaçant PLZMAX.
Met à jour PLZMIN

**/AUTO**

Choisis le pas selon le mode défini par /LIN ou /LOG en annulant les paramètres donnés avec ceux-ci.
Met à jour PLDELT(1) ou PLDELT(1) et PLDELT(2)

**/STAT**

Détermine et met à jour PLZMIN et PLZMAX.

**/LTYPE=épaisseur[,dash[,color]]**

Pose l'épaisseur des lignes de contour en [mm]

**/PUT**

Numérote les pics dans l'ordre croissant selon les valeurs contenues dans les buffers XRET(i) et YRET(i), pour $i = 1, NPT$ (Uniquement avec GTYP=1,2,3).

**/MM=**$< xmm >$

Donne la taille de l'axe $< X >$ en millimètres.

**/NOAXIS**

Ne représente pas les axes.

**/LIST**

affiche les valeurs des niveaux.

Voir aussi tout les qualificateurs communs aux commandes graphiques.

---

## 1. TRACÉ DES COORDONNÉES ÉQUATORIALLES

**SYNTAXES:**

CONTOUR [<No matrice>] /AD [/NOLINE] [/qualif]

CONTOUR [<No matrice>] /ISO [/qualif]

**DESCRIPTION:**

Le mode remplace le tracé des graduations en pixel sur un dessin de type 2D (GTYP=1,2,3). Dans ce mode on spécifie la coordonnée du centre du champ ainsi que l'échelle dans les deux axes. On peut également tracer une grille ISO par dessus le tout.

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **ALPHA** | Position de l'objet clique (degres decimaux) |
| **DELTA** | Position de l'objet clique (degres decimaux) |
| **DETSCX** | taille d'un pixel en X |
| **DETSCY** | taille d'un pixel en Y |

**QUALIFICATEURS A DISPOSITION:**

**/AD**

Trace les coordonnées équatoriales sur le dessin. Le centre du champ est donné par ALPHA et DELTA et l'échelle par DETSCX et DETSCY ([sec d'arc/step]). Le champ est orienté Nord Sud.

**/NOLINE**

Ne trace ni les petits ni les grands cercles en mode AD.

**/DEGRE**

Gradue l'axe Alpha en degré.

**/ISO[=<alpha>[,<delta>[,<taille_X>[,<taille_Y>[,<mire_X>[,<mire_Y>]]]]]]**

Trace une mire ISO (grille) selon les arguments du qualif. C'est à dire :

| | |
|---|---|
| <**alpha**> | Donne le centre de la mire en alpha (donné sous forme de chaîne de caratères) |
| <**delta**> | Donne le centre de la mire en delta (donné sous forme de chaîne de caratères) |
| <**taille_X**> | Donne la taille de la mire selon alpha en [sec d'arc] |
| <**taille_Y**> | Donne la taille de la mire selon delta en [sec d'arc] |
| <**mire_X**> | Donne le nombre de cases dans la mire selon alpha |
| <**mire_Y**> | Donne le nombre de cases dans la mire selon delta |

Les défauts pour ce qualificateur sont : "0h", "0d", 32., 32., 32, 32

## 2. ACQUISITION DE POINTS

**SYNTAXE:**

CONTOUR [<No matrice>] /GET [/ECHO] [/qualif]

**DESCRIPTION:**

L'acquisition se fait au moyen de la souris sur un dessin de type 2D (GTYP=1,2,3) et permet de récupérer les positions X et Y ainsi que les valeurs de pixels.

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **RETMAX** | taille de XRET,YRET,ZRET |

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **NPT** | Nb de valeurs retournees |
| **XRET** | coordonees X en retour |
| **YRET** | coordonees Y en retour |
| **ZRET** | densite en retour |

**QUALIFICATEURS A DISPOSITION:**

**/GET[=<chaîne de caractère>]**

Recherche de position avec le réticule. Chaque validation du réticule pose soit le numéro d'ordre (défaut), soit un caractère pris successivement dans <chaîne de caractère>. Ce mode s'arrête en tapant 'E' ou 'e' ou lorsque tout les caractères de la chaîne on été entrés ou enfin lorsque RETMAX coordonnées on été entrées.
Le nombre de validation est rendu par NPT, les coordonnées par XRET(i) et YRET(i) et les valeurs pixel par ZRET(i).

**/ECHO**

Affiche en écho les positions relevées.

## 3. COULEUR et LÉGENDE

**SYNTAXES:**

CONTOUR [<No matrice>] /CSCALE=AFF [/LEGENDE] [/qualif]

CONTOUR [<No matrice>] /CSCALE=<No>[,AFF] [/LEGENDE] [/qualif]

CONTOUR [<No matrice>] /CSCALE=<nom> [/LEGENDE] [/qualif]

**DESCRIPTION:**

L'utilisation de la couleur (ou niveau de gris, si la couleur n'est pas possible) est activée pour GTYP=1,2,6,7. Le choix de la table de couleur se fait par le qualificateur /CSCALE (ColorSCALE). On a alors le choix entre les 6 tables prédéfinies par le logiciel UNIRAS, aux tables standards de Midas ou à la table actuellement affichée sur l'afficheur (voir "CLIENT /AFF", "AFF" et "PATCH").

Avec l'utilisation de l'afficheur (méthode la plus efficace), il faut uniquement charger la matrice sur l'afficheur (PATCH /FIT), et y selectionner les cuts et les couleurs adéquates. La commande "CONTOUR /CSCALE=AFF" se charge de rapatrier les couleurs et les cuts pour obtenir une copie fidèle de l'image sur l'afficheur.

La légende permet de visualiser la table de couleur utilisée selon différents aspects.

**VARIABLES INTERACTIVES PRINCIPALES:**

**PLNNIV** nb de niveaux

---

**QUALIFICATEURS A DISPOSITION:**

**/CSCALE[=<nom>]**

Donne le nom de l'échelle de couleur à utiliser. Si le nom est "aff" alors on lit la table de couleur de l'afficheur, sinon on lit dans le fichier "<nom>.lut" (fichier rdb contenant 356 niveaux. Colonnes : "R", "G" et "B"), ou dans le fichier binaire ("$AFFHOME/afflut.bin").

Les tables de couleurs a dispositions sont :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| rainbow | rainbow2 | rainbow3 | rainbow4 | standard | real | pseudo1 | pseudo2 |
| pastel | mousse | manycol | isophot | random | heat | staircase | ramp |
| light | | | | | | | |

**/LEGEND[=<xpos>,<ypos>][,option :argument[,...]]**

Pose la légende de l'échelle de couleur selon les options donnée sur le dessin courant(défaut), dans la prochaine window avec (**/NEXT**) ou à la position <xpos,ypos> si <xpos> et <ypos> sont donnés. Ces positions sont comprises en coordonnées world pour le dessin courant ou en millimètres pour la window suivante.

**<xpos>,<ypos>** position de la légende
défaut (std) :$< 1.04 * (nx - 1) * xstep >$,$< 0 >$
défaut (std,/next) :$< 4\%$fenêtre$>$,$< 4\%$fenêtre$>$
défaut (vbox) :$< 1.04 * (nx - 1) * xstep >$,$< 0 >$
défaut (vbox,/next) :$< 4\%$fenêtre$>$,$< 4\%$fenêtre$>$
défaut (hbox) :$< 0 >$,$< (ny - 1) * ystep - height >$
défaut (hbox,/next) :$< 4\%$fenêtre$>$,$< 4\%$fenêtre$>$

**TYPE :type** type de représentation

| | |
|---|---|
| **STD** | échelle standard PGPLOT (std) |
| **VBOX** | échelle continue verticale (vbox) |
| **HBOX** | échelle continue horizontale (hbox) |

**HEIGHT :h** hauteur de l'échelle std)
défaut (vbox) : $ny * ystep$
défaut (hbox,/next) : 92% de la fenêtre
défaut (hbox) : 5% de width

**WIDTH :w** largeur de l'échelle
défaut (vbox) : 5% de height
défaut (hbox) : $nx * xstep$
défaut (hbox,/next) : 92% de la fenêtre

**ABOVE :txt** texte du haut de l'échelle
défaut (vbox) : ""

défaut (hbox) : ""

**BELOW :txt** texte du bas de l'échelle
défaut (vbox) : ""
défaut (hbox) : ""

**/NEXT**

Avec /LEGENDE, place la légende dans la window suivante.

---

## 4. ÉCHELLE

---

**SYNTAXE:**

CONTOUR [<No matrice>] /SCALE [/qualif]

---

**DESCRIPTION:**

Dessin d'une échelle sur le dessin.

---

**VARIABLE INTERACTIVE PRINCIPALE:**

**DETSCX** taille d'un pixel en X

---

**QUALIFICATEUR A DISPOSITION:**

**/SCALE[=<xpos>[,<ypos>[,<taille>]]]**

Pose une échelle horizontale (barre avec valeur d'échelle) à la position <xpos>,<ypos> donnée relative à l'origine des axes (0 ;0) et du pas (XSTEP,YSTEP). La longueur de l'échelle est donnée par

$$< longueur >=< Taille > *DETSCX$$

avec <taille> donné en secondes d'arc.

L'échelle écrite s'arrondi automatiquement à des valeurs entières (sauf pour le cas d'une taille plus petite que la seconde) en secondes ("), minutes (') ou degrés (d).

La jeux de caractère utilisé est GLFONT, la taille des caractères est donnée par GLSIZ

Les défauts pour ce qualificateur sont : 1, 1, 1

---

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| ALPHA | I | Position de l'objet clique (degres decimaux) |
| DELTA | I | Position de l'objet clique (degres decimaux) |
| DETSCX | I | taille d'un pixel en X |
| DETSCY | I | taille d'un pixel en Y |
| GDRIVE | I | marge pour la feuille 0=pas <0=mm >0=marge |
| GFLAG | I | tailles des fontes titre et label automatique si = 0 |
| GLBCOL | I | index couleur background pour le label (<0 == tranparent) |
| GLCOL | I | index couleur des caracteres pour le label |
| GLFONT | I | No fonte des labels (1-4) (norm,roman,italic,script) |
| GLSIZ | I | taille de la fonte des labels si GFLAG =/= 0 |
| GMARGE | I | marge pour la feuille 0=pas <0=mm >0=marge |
| GPHI | I | 3D - angle dans le plan X-Y |
| GTBCOL | I | index couleur background pour le titre (<0 == tranparent) |
| GTCOL | I | index couleur des caracteres pour le titre |
| GTFONT | I | No fonte du titre (1-4) (norm,roman,italic,script) |
| GTHETA | I | 3D - Angle vertical |
| GTSIZ | I | taille de la fonte des titres si GFLAG =/= 0 |
| GTYP | I | type de graphique (1-7) |
| LWX | I | largeur de la window en X |
| LWY | I | largeur de la window en Y |
| NPT | I/O | Nb de valeurs retournees |
| NWX | I | nb de subwindow en X |
| NWY | I | nb de subwindow en Y |
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| PLDELT | I/O | pas inter niveaux |
| PLNNIV | I | nb de niveaux |
| PLZMAX | I/O | plot parameter |
| PLZMIN | I/O | plot parameter |
| RETMAX | I | taille de XRET,YRET,ZRET |
| WDX | I/O | No subwindow courante en X |
| WDY | I/O | No subwindow courante en Y |
| XRET | I/O | coordonees X en retour |
| XSTART | I | coord. world de depart en X de la matrice 1 |
| XSTEP | I | pas en X de la matrice 1 |
| YRET | I/O | coordonees Y en retour |
| YSTART | I | coord. world de depart en Y de la matrice 1 |
| YSTEP | I | pas en Y de la matrice 1 |
| ZRET | I/O | densite en retour |

## 2.14 DATE

Met à jour les variables du bloc concernant la date et l'heure.

---

**SYNTAXES:**

DATE

DATE /ECHO

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **FDATE** | date formattée (TCL) |
| **SECOND** | seconde (TCL) |
| **MINUTE** | minute (TCL) |
| **HOUR** | heure (TCL) |
| **DAY** | jour (TCL) |
| **MONTH** | mois (TCL) |
| **YEAR** | annee (TCL) |
| **FDATE** | date formattée (TCL) |
| **UTC_SECOND** | seconde (UTC) |
| **UTC_MINUTE** | minute (UTC) |
| **UTC_HOUR** | heure (UTC) |
| **UTC_DAY** | jour (UTC) |
| **UTC_MONTH** | mois (UTC) |
| **UTC_YEAR** | annee (UTC) |
| **UTC** | Coordinated Universal Time (heures decimales) |
| **TCL** | temps civil local (heures decimales) |

---

**QUALIFICATEUR A DISPOSITION:**

**/ECHO**

Affiche la date à l'écran en même temps.

---

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| DAY | O | jour (TCL) |
| DTUNIX | O | delta tunix depuis le lancement de inter |
| FDATE | O | date formattée (TCL) |
| HOUR | O | heure (TCL) |
| MINUTE | O | minute (TCL) |
| MONTH | O | mois (TCL) |
| SECOND | O | seconde (TCL) |
| TCL | O | temps civil local (heures decimales) |
| TUNIXUS | O | microsecondes de TUNIX |
| TUNIX | O | temps Unix nb de seconde depuis 1.1.1970 0h UTC |
| UTC_DAY | O | jour (UTC) |
| UTC_FDATE | O | date formattée (UTC) |
| UTC_HOUR | O | heure (UTC) |
| UTC_MINUTE | O | minute (UTC) |
| UTC_MONTH | O | mois (UTC) |
| UTC_SECOND | O | seconde (UTC) |
| UTC_YEAR | O | annee (UTC) |
| UTC | O | Coordinated Universal Time (heures decimales) |
| YEAR | O | annee (TCL) |

## 2.15 DEBUG

Permet la sortie des messages de test à l'écran par catégorie et selon un niveau de debugging dans Inter et dans les commandes d'application.

---

**SYNTAXES:**

DEBUG

DEBUG /ON | /OFF | /LEVEL=<n> /ALL

DEBUG /ON | /OFF | /LEVEL=<n> /<category>

avec : /<category> ==

- /CMD=<commande>
- /PROC
- /IPC
- /SIGNAL
- /REHASH
- /STORE
- /DEPTH
- /EVAL
- /MALLOC

---

**DESCRIPTION:**

Sans qualificateurs, DEBUG affiche la liste des catégories ayant le debugging "ON". Avec les qualificateurs il permet de selectionner la ou les catégories et le niveau de debbugging.

Pour chaque utilisation de `DEBUG` il faut préciser un niveau de debugging et la catégorie pour laquelle il est destiné.

---

### 1) Niveau de debug

---

**QUALIFICATEURS A DISPOSITION:**

**/ON**

Mode Debug `ON`, niveau = 1

**/OFF**

Mode Debug `OFF`, niveau = 0

**/LEVEL=<n>**

Donne le niveau de debugging.

---

## 2) Categories

**QUALIFICATEURS A DISPOSITION:**

**/ALL**

Toutes les catégories

**/CMD=<commande>**

Mode Debug pour la commande d'application <commande>, La commande peut être raccourcie au cas de non synonyme. Attention on ne peut donner qu'une commande à la fois. Il faut donc lancer `DEBUG` autant de fois que l'on désire de commande.

**/PROC**

Affiche les opérations liées aux fichiers de procédure (open(), close(), ...

**/IPC**

Affiche les opérations liées à la communication inter–process.

**/SIGNAL**

Affiche les opérations liées aux signaux.

## 3) Categories (Pour initiés)

**QUALIFICATEURS A DISPOSITION:**

**/REHASH**

Affiche les opérations liées au hashing des tables de variables.

**/STORE**

Affiche les opérations liées au stockage des variables.

**/DEPTH**

Affiche les opérations liées au management des niveaux logiques.

**/MALLOC**

Affiche les opérations liées aux allocations mémoire.

**/EVAL**

Affiche les opérations liées à l'évaluation des fonctions.

## 2.16 DECONV

Déconvolution d'une image 2D par des méthodes itératives

---

**SYNTAXE:**

DECONV [<No mat image>] [<No mat PSF>][/qualificateurs]

---

**PARAMETRES:**

| | |
|---|---|
| <**No mat image**> | matrice à deconvoluer ou convoluer |
| <**No mat PSF**> | matrice contenant la PSF |

---

**DESCRIPTION:**

L'image à déconvoluer se trouve dans la matrice donnée sur la couche 1. Le résultat de la déconvolution apparaît sur la couche 2. La couche 3 est un tableau intermédiaire de calcul, la couche 4 est le masque d'image (il indique la position des points éliminés. Si elle n'est pas spécifiée, la PSF doit être dans la matrice 1. Si la matrice à déconvoluer se trouve dans la 1 alors la PSF doit être dans la matrice suivante.

DECONV permet d'utiliser 5 algorithmes (NALGO=1...5). Les paramètres qu'ils utilisent ne sont donnés qu'approximativement :

SIMPLE : power = .5à .8

LUCY : power = .7 à 1

FRIEDEN : power = 1 à l'infini

GAUSS : power = 1, wdcv = .01 à 1

LUCY_CAP : power = .7 à 1, wdcv = .5 à 1, sigma = 1 à 5

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **NALGO** | choix d'un algorithme |
| **NSTEP** | Nb d'iteration |
| **POWER** | Puissance |
| **SIGMA** | Ecart type |
| **WDCV** | ECONV Coeff pour GAUSS |

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **FLUX** | Flux total sur la matrice image |

---

---

**QUALIFICATEURS A DISPOSITION:**

**/INIT**

Initialise le programme : création du masque, élimination des points, calcul du flux sur l'image, affectation (objet initial)=(image).

**/STOP**

Restitue l'image (restauration des points)

**/CONVOL**

Uniquement la convolution (conservation du flux sur l'image totale).

---

## VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE

| Variables | Mode | Description |
|---|---|---|
| FLUX | O | Flux total sur la matrice image |
| NALGO | I | choix d'un algorithme |
| NSTEP | I | Nb d'iteration |
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| POWER | I | Puissance |
| SIGMA | I | Ecart type |
| WDCV | I | ECONV Coeff pour GAUSS |

## 2.17 DEFAULT

Donne les défauts des paramètres manquant à l'appel d'une procédure

---

**SYNTAXES:**

DEFAULT <No du paramètre>=<chaîne de caractères>

DEFAULT <No du paramètre>=<angle> /ANGLE

---

**DESCRIPTION:**

Cette commande a deux fonctions. Elle permet, d'une part, d'attribuer une valeur de défaut à un argument de procédure absent ou donné sous forme de point d'interrogation. Ou, d'autre part, de tester le type de l'argument quand il est donné (celui-ci doit être le même que celui du défaut).

Rappel : les arguments d'une procédure remplacent les termes donnés sous la forme suivante : "{n}" avec "n" comme numéro d'argument (1-9).

---

**QUALIFICATEUR A DISPOSITION:**

**/ANGLE**

Précise que le défaut doit représenter un angle.

Ex: 12 (nombre) ou "12h24m15s" (chaîne de caractères) sont également admis.

---

**EXEMPLES:**

```
DEFAULT 1=XSTART+1
DEFAULT 2=12    /ANGLE
DEFAULT 3="alpha du centaure"
```

---

**REMARQUE:**

DEFAULT ne doit apparaître que dans une procédure de type macro.

## 2.18 DO

Initialise une boucle DO...ENDDO, d'un usage identique au fortran

---

**SYNTAXES:**

DO <index>=<start>,<stop>

DO <index>=<start>,<stop>,<step>

---

**PARAMETRES:**

| | |
|---|---|
| <**index**> | Nom d'une variable d'Inter donnant l'index de boucle. |
| <**start**> | Expression donnant le valeur de <index> en début de boucle. |
| <**stop**> | Expression donnant la valeur test pour finir la boucle. |
| <**step**> | Expression donnant la valeur d'incrémentation de <index> à chaque boucle. Cette valeur est calculée à l'initialisation de la boucle. |

---

**EXEMPLE:**

```
local name="albert"
do i=1,len(name)
   write name(1)(i:i)
enddo
```

## 2.19 ECHO

Permet l'écho des commandes en mode compilation ou en mode exécution.

---

**SYNTAXES:**

ECHO

ECHO /ON

ECHO /OFF

---

**DESCRIPTION:**

Sans qualificateurs, ECHO affiche son status (on ou off).

---

**QUALIFICATEURS A DISPOSITION:**

**/ON**

Se met en mode écho.

**/OFF**

Quitte le mode écho.

## 2.20 ELIMINATION

Elimine les points d'une image ne répondant pas à certains critères ou remplace les points éliminés par les valeurs de la fonction théorique calculée.

---

**SYNTAXE:**

ELIMINATION [<No de matrice>] [/qualificateurs]

---

**DESCRIPTION:**

L'élimination des points s'opère de manière non destructive. Cela permet ainsi de retrouver la valeur initiale de chaque point en effectuant l'opération réciproque.

Les critères de rejet sélectionnés sont écrits dans la variable ELIM (de type caractère) et signifient :

**MM** rejet des points dont la valeur est hors des limites définies par ELMIN et ELMAX.

**FU** rejet des points dont l'écart à la fonction théorique est trop importante.

L'élimination des points dépend du type de bruit s'ajoutant au signal. Celui-ci peut être constant ou proportionnel au signal (au nb de photons). La variable NOISE (de type caractère) contiendra les termes 'CONSTANT' ou 'PHOTON' selon le type de bruit. Par défaut le bruit est pris constant.

Dans le cas d'un **bruit constant** les points sont éliminés lorsque l'écart à la fonction est supérieur à :

$$écart\_max = ELFACT * CHI$$

ELFACT est égal à l'écart absolu si le qualificateur /ABS est précisé.

Dans le cas d'un **bruit de photon** l'écart à la fonction maximum admis est calculé par :

$$\mathrm{abs(écart\_max)} = ELFACT * \sqrt{\mathrm{RMS}^2 + \frac{\max(0, \mathrm{valeur\_pixel})}{\mathrm{GAIN}}}$$

**EL** rejet des points a l'intérieur d'une ellipse de rayon RELIM définie par les paramètre de la fonction (orientation=PFCT(5), ellipticité=PFCT(4)).

Si les termes 'MM', 'FU' ou 'EL' sont suffixés par 'OUT' (exemple : 'MMOUT'), alors l'élimination a lieu sur les points ne répondant pas au test. Par défaut, les termes sont compris 'MMIN' 'FUIN' ou 'ELIN' même si la terminaison 'IN' n'est pas précisée.

La fonction de référence est décrite par les variables FONCT et PFCT(1) à PFCT(10). Elle peut être choisie interactivement avec le qualificateur /FONCTION. Dans ce cas elle remplace la fonction décrite dans le bloc de donnée.

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **ELIM** | type d'elimination |
| **FONCT** | nom de la fct |
| **PFCT** | parametre fonction |
| **GAIN** | Gain photoelectrons/ADU |
| **CHI** | Sigma du fit |
| **RMS** | Read-out noise in ADU |
| **ELFACT** | facteur pour CHI ou val ABS |
| **ELMIN** | seuil inferieur |
| **ELMAX** | seuil superieur |
| **NOISE** | bruit(PHOTON-CONST) |
| **RELIM** | rayon d'elimination pour ELIM="EL" |

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **FONCT** | Voir plus haut sous "VARIABLES INTERACTIVES PRINCIPALES" |
| **PFCT** | Voir plus haut sous "VARIABLES INTERACTIVES PRINCIPALES" |

---

**QUALIFICATEURS A DISPOSITION:**

**/ELIMINATION**

Elimine les points (Opération par défaut).

**/RESTORE**

Redonne leurs valeurs initiales aux points éliminés.

**/REPLACE**

Donne les valeurs de la fonction théorique aux points à éliminer et aux points déjà éliminés.

**/FONCTION[=[<fonction>,][<P4> ...[,<P10>]]]**

Décrit la fonction théorique remplaçant les valeurs par défaut.
Met a jour les variables FONCT et PFCT(4) à PFCT(10) s'il sont précisés. C'est le seul cas où une modification dans le bloc de données a lieu.

**/ABS**

Utilise la variable ELFACT comme écart absolu maximal dans le cas d'un bruit constant.

**/LIST**

Affiche des statistiques sur le travail effectué.

---

**REMARQUE:**

La facon d'éliminer les points est décrite dans le fichier de type include "elim.inc".
ELTST est aussi initialisée dans ce fichier et sert au test des points éliminés.

---

## VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE

| Variables | Mode | Description |
|---|---|---|
| CHI | I | Sigma du fit |
| ELFACT | I | facteur pour CHI ou val ABS |
| ELIM | I | type d'elimination |
| ELMAX | I | seuil superieur |
| ELMIN | I | seuil inferieur |
| FONCT | I/O | nom de la fct |
| GAIN | I | Gain photoelectrons/ADU |
| NOISE | I | bruit(PHOTON-CONST) |
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| PFCT | I/O | parametre fonction |
| RELIM | I | rayon d'elimination pour ELIM="EL" |
| RMS | I | Read-out noise in ADU |
| XSTART | I | coord. world de depart en X de la matrice 1 |
| XSTEP | I | pas en X de la matrice 1 |
| YSTART | I | coord. world de depart en Y de la matrice 1 |
| YSTEP | I | pas en Y de la matrice 1 |

## 2.21 ELSE

C’est le ELSE du IF

---

**REMARQUE:**

Le "ELSE IF" n’est pas supporté.

---

**EXEMPLE:**

```
if (i.gt.maxval) then
   write "Erreur: Dépassement"
else
   call nextstep
endif
```

## 2.22 ENDCASE

Ordre de contrôle pour la fin d'un aiguillage type CASE

---

**EXEMPLE:**

```
get
case affret of
  1::
     call next
  2::
     call previous
  3::
     call affich
endcase
```

## 2.23 ENDDO

Termine une boucle DO...ENDDO, d'un usage identique au fortran

---

**EXEMPLE:**

```
local name="albert"
do i=1,len(name)
    write name(1)(i:i)
enddo
```

## 2.24 ENDIF

C’est le ENDIF du IF

---

**EXEMPLE:**

```
if (i.gt.maxval) then
    write "Erreur: Dépassement"
else
    call nextstep
endif
```

## 2.25 ENDPROC

Ordre de contrôle pour la fin d'une procédure ou d'une subroutine

---

**REMARQUE:**

Cette ordre est optionnel, il est utilisé pour séparer les procédures et les subroutines dans un même fichier.

---

**EXEMPLE:**

```
do i=1,10
   call show i
enddo
endproc

subroutine show value
write "compteur = " value
call maxi value
return
endproc

subroutine maxi val
if val.eq.10 write "c'est fini"
return
```

## 2.26 ERREUR

Définis la stratégie à adopter lorsque survient une erreur.

---

**SYNTAXES:**

ERREUR
ERREUR /ON
ERREUR /OFF
ERREUR /SET "<message>"
ERREUR /SET /LOGBOOK "<message>"
ERREUR /CODE=<code> <argument> ...
ERREUR /CODE=<code> /LOGBOOK <argument> ...
ERREUR /PROPAGATE /qualif

---

**DESCRIPTION:**

Elle peut soit aborter la procédure en cours, soit simplement afficher l'erreur.On peut également simuler une erreur et afficher un message d'erreur.
Sans qualificateurs, ERREUR affiche le mode courant (on ou off).

---

**QUALIFICATEURS A DISPOSITION:**

**/ON**

S'arrête lors d'une erreur.

**/OFF**

Continue lors d'une erreur.

**/FPEON**

S'arrête lors d'une erreur de Floating Point Exception (défaut).

**/FPEOFF**

Continue lors d'une erreur de Floating Point Exception.

**/SET**

Génère une erreur. Affiche le message s'il est donné.

**/CODE=<code>**

Génère une erreur. Affiche le message selon le code et les arguments.

**/LOGBOOK**

Avec /SET ou /CODE envoie le message sur le logbook.

**/PROPAGATE**

En mode serveur, le code d'erreur fournis par son serveur est propagé pour un usage par le client. Sinon c'est <code> ou "int_nomess" qui est utilisé.

---

---

**REMARQUE:**

INTER s’initialise en mode arrêt sur erreur.

## 2.27 EVALUE

Commande similaire à "SET" mais permet l'évaluation des expressions contenue dans les chaînes de caractères.

---

**SYNTAXE:**

EVALUE <dest>=<expression> [<dest>=<expression>] ...

---

**PARAMETRES:**

| | |
|---|---|
| <**dest**> | nom d'une variable Inter |
| <**expression**> | expression numérique ou caractère |

---

**EXEMPLE:**

```
qualif="/expr=i+1"
...
EVALUE i=qualif(7:)
```

## 2.28 EXECUTE

Exécute la dernière procédure lancée avec "@" ou termine et lance la procédure tapée au clavier (voir commande COMPILE).

---

**SYNTAXE:**

EXECUTE [/qualificateur]

---

**DESCRIPTION:**

EXECUTE termine la procédure entrée au clavier (s'il n'y a pas d'erreur dans l'arrangement des boucles IF..ELSE..ENDIF, DO..ENDDO, ...) puis l'exécute.
Une procédure n'a besoin d'être compilée qu'une seule fois, EXECUTE la relance sans avoir besoin de la retaper ou de réutiliser la commande "@".

---

**QUALIFICATEURS A DISPOSITION:**

**/VERIFY**

Les lignes de commandes sont affichées en écho

**/CONTINUE**

Poursuit le procédure après un $< CTRL > -C$ depuis l'endroit où elle était interrompue.

## 2.29 EXIT

Sort d'inter en sauvant le bloc de données.

---

**SYNTAXE:**

EXIT

---

**REMARQUE:**

Ferme les images et les tables ouvertes.

## 2.30 EXTRACT

Extrait une partie d'un fichier image ou d'une image de l'afficheur pour la placer dans l'une des matrices de travail.

---

**SYNTAXES:**

EXTRACT <fichier> [<No matrice>] [/qualif]

EXTRACT AFF [<No matrice>] [/qualif]

---

**PARAMETRES:**

| | |
|---|---|
| <**fichier**> | Nom du fichier à lire |
| <**No matrice**> | Numéro de matrice de destination |

---

**DESCRIPTION:**

Si <nom> vaut "AFF" (ou "aff") alors l'extraction a lieu sur l'image de l'afficheur. Sinon, <nom> est soit, par défaut, le nom d'une image Midas (extension "bdf"), soit un fichier FITS (extension "fit" ou "fits"). Dans tout les cas, l'image est découpée selon les options spécifiées avec les qualificateurs (un pour chaque axe).

Les valeurs données avec les options indiquent :

– des coordonnées 'world'. Attention, l'origine d'une image de l'afficheur est toujours en $< 0., 0. >$
  Ex: EXTRACT /X=(S :21200,E :34100) ...

– des numéros de lignes ou de colonnes si l'option est précédée d'un '@'. Attention, on numérote les lignes et les colonnes à partir de 1.
  Ex: EXTRACT /X=(@S :12,@E :32) ...

Ces deux modes peuvent être mélangés :

Ex: EXTRACT /X=(C :23612,@N :20) ...

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **NX** | nb de pixel en X de la matrice 1 |
| **NY** | nb de pixel en Y de la matrice 1 |
| **XSTART** | coord. world de depart en X de la matrice 1 |
| **YSTART** | coord. world de depart en Y de la matrice 1 |
| **XSTEP** | pas en X de la matrice 1 |
| **YSTEP** | pas en Y de la matrice 1 |
| **XCORIG** | coord w. de dep en X de l'image d'origine 1 |
| **YCORIG** | coord w. de dep en Y de l'image d'origine 1 |

---

**QUALIFICATEURS A DISPOSITION:**

**/X**

Donne les paramètres de découpage selon l'axe X

| | |
|---|---|
| **S :n** | début |
| **E :n** | fin |
| **N :n** | largeur |
| **C :n** | centre |
| **P :n** | pas d'échantillonnage |

Sans autre précision, les paramètres sont compris par défaut comme "@S" et "@N".

**/Y**

Donne les paramètres de découpage selon l'axe Y.
Ce qualificateur possède les mêmes arguments que /X

**REMARQUES:**

L'extension BDF est mise par défaut.

EXTRACT travaille avec des valeurs internes entières pour l'accès aux colonnes et aux lignes de l'image. Mais tous ces paramètres lui sont envoyés en valeurs réelles, que ce soit en coordonnées 'world' ou en position (colonne ;ligne).

Dans toutes les opérations les réels sont convertis en entiers avec la fonction fortran NINT.
Par exemple : avec une image ayant un pas de 40, si l'échantillonnage est demandé à 59, toutes les colonnes sont prises, et s'il est de 60, une colonne sur 2 est transférée.

S'il y a des paramètres redondants, EXTRACT est arrêté.

Si les options concernant la largeur ou l'échantillonnage sont négatives, les valeurs sont prises positives et un message est envoyé.

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| MDCOMM | I/O | commentaire |
| MDDATE | I/O | date d'acquisition |
| NFIELD | I/O | No de champ |
| NPLAT | I/O | No de cliche |
| NREP | I/O | |
| NX | I/O | nb de pixel en X de la matrice 1 |
| NY | I/O | nb de pixel en Y de la matrice 1 |
| XCORIG | I/O | coord w. de dep en X de l'image d'origine 1 |
| XREF | I/O | X reference |
| XSTART | I/O | coord. world de depart en X de la matrice 1 |
| XSTEP | I/O | pas en X de la matrice 1 |
| YCORIG | I/O | coord w. de dep en Y de l'image d'origine 1 |
| YREF | I/O | Y reference |
| YSTART | I/O | coord. world de depart en Y de la matrice 1 |
| YSTEP | I/O | pas en Y de la matrice 1 |

## 2.31 FILE

Opérations de base sur fichiers ASCII formattés.

---

**SYNTAXES:**

FILE <nom_du_fichier> [/qualif]

FILE /unit=<unit> [/qualif]

---

**DESCRIPTION:**

Permet l'ouverture en début ou en fin de fichier, la fermeture ou l'impression du status(si aucun qualificateur est donné) de fichiers utilisés sous INTER par la commande WRITE, selon l'usage en Fortran.

---

**QUALIFICATEURS A DISPOSITION:**

**/UNIT=<unit>**

Précise le numéro d'unité logique sous lequel le fichier pourra être accédé par la suite. Ce numéro doit être compris entre 50 et 80 et doit être précisé lors de l'ouverture. La fonction **GETLU()** peut être utilisée pour obtenir un numéro d'unité valide.

**/OPEN**

Ouvre le fichier de manière standard (STATUS='UNKNOWN' par défaut).

**/STATUS=<status>**

Précise le status d'ouverture du fichier ('OLD' ou 'NEW').

**/CLOSE**

Ferme le fichier.

**/LIST**

Affiche le status des unités logiques ouvertes entre 50 et 80.

**/ALL**

Utilisé avec /CLOSE, il ferme tout les fichiers ouverts.

Utilisé avec /LIST, il affiche le status des unités logiques ouvertes entre 0 et 100.

**/APPEND**

Se place sur l'EOF.

**/REWIND**

Se place au début du fichier.

**/BACKSPACE**

Revient d'une ligne en arrière.

---

**REMARQUE:**

Si seul le nom du fichier ou seul son numéro d'unité est donné alors le status du fichier est affiché.

## 2.32 FIMAGE

Diverses opérations sur fichier image FITS

---

**COMMANDES DE BASE**

---

**SYNTAXE:**

FIMAGE <nom> [<No_de_matrice>] [/qualificateur]

---

**PARAMETRES:**

<**nom**> Nom d'un fichier fits

<**No_de_matrice**> Numéro de matrice de destination

---

**DESCRIPTION:**

Ces commandes permettent de lire une image complète, de stocker une matrice dans un nouveau fichier, de lister l'entête d'un fichier fits et de lire ou d'écrire des keywords du fichier.

---

**REMARQUE:**

la lecture partielle d'une image FITS se fait avec la commande EXTRACT.

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **NX** | nb de pixel en X de la matrice 1 |
| **NY** | nb de pixel en Y de la matrice 1 |
| **XSTART** | coord. world de depart en X de la matrice 1 |
| **YSTART** | coord. world de depart en Y de la matrice 1 |
| **XSTEP** | pas en X de la matrice 1 |
| **YSTEP** | pas en Y de la matrice 1 |
| **XCORIG** | coord w. de dep en X de l'image d'origine 1 |
| **YCORIG** | coord w. de dep en Y de l'image d'origine 1 |

---

---

**QUALIFICATEURS A DISPOSITION:**

**/READ**

Lecture de l'image complète

**/WRITE**

Ecriture de la matrice complète. En plus des keywords standards (SIMPLE, BITPIX, NAXIS, NAXISn, EXTEND), les keywords suivants sont écrits : CRVALn, CRPIXn, CDELTn, DATE, BSCALE, BZERO.

Il est recommandé de rajouter les keywords :

| | |
|---|---|
| **ORIGIN** | nom de l'organisation ayant crée le fichier |
| **DATE-OBS** | date de l'observation (DDMMYY) en TU. |
| **TELESCOP** | nom du télescope |
| **INSTRUME** | nom de l'instrument ayant servit à l'acquisition |
| **OBSERVER** | nom de l'observateur |
| **OBJECT** | nom de l'objet observé |
| **EQUINOX** | l'équinoxe en année pour lequel les coordonnées sont exprimées |

**/REPLACE**

Tue le fichier s'il existe avant de le créer. (Rem : Fimage n'écrit pas sur un fichier existant).

**/BYTE**

Avec /WRITE, sauve l'image en entiers 8 bits non–signés.

**/SHORT**

Avec /WRITE, sauve l'image en entier 16 bits signés.

**/INT**

Avec /WRITE, sauve l'image en entier 32 bits signés.

**/FLOAT**

Avec /WRITE, sauve l'image en flottant 32 bits. (C'est le défault)

**/DOUBLE**

Avec /WRITE, sauve l'image en flottant 64 bits.

**/RKEY=<keyword>[,<variable>]**

Lit le contenu de <keyword> dans une variable. Si la variable n'est pas donnée l'affichage a lieu à l'écran, avec le commentaire.

Attention, le nom de la variable doit être donné entre double guillemets.

**/WKEY=<keyword>,<expression>[,<commentaire>]**

Ecrit le résultat de l'expression dans <keyword> avec des commentaires optionnels.

**/MKEY=<keyword>,<expression>[,<commentaire>]**

Modifie le contenu de <keyword>. Si <commentaire> est donné, le commentaire est modifié.

**/RMKEY=<keyword>**

Enlève le keyword <keyword>.

**/HEADER**

Liste l'entête

**/COMMENT=<texte>**

Ajoute une ligne de type "COMMENT"

**/HISTORY=<texte>**

Ajoute une ligne de type "HISTORY"

**/RCOMMENT**

Liste uniquement les lignes COMMENT et les lignes sans keywords

**/RHISTORY**

Liste uniquement les lignes HISTORY

**/DATE**

Pose ou met à jour la date courante (dd/mm/yy)

## MACRO–COMMANDES

**SYNTAXES:**

FIMAGE <nom> /RHEADER[=<modèle>]

FIMAGE <nom> /RHEADER[=<modèle>] /FNAME [/UNSET]

FIMAGE <nom> /WHEADER=<modèle>

FIMAGE <nom> /WHEADER=<modèle> /FNAME

FIMAGE <nom> /RHEADER | /WHEADER /NOWARNING

FIMAGE <nom> /RHEADER | /WHEADER /ECHO

FIMAGE <nom> /RHEADER | /WHEADER /INQUIRE

FIMAGE <modèle> <nom> /CPHEADER

FIMAGE <nom> <No_de_matrice> <modèle> /WRITE /CPHEADER

**PARAMETRES:**

| | |
|---|---|
| <**nom**> | Nom d'un fichier fits |
| <**modèle**> | Nom d'un fichier fits contenant les déscripteurs de référence |

---

**DESCRIPTION:**

Ces macros commandes permettent un contrôle plus général pour l'accès aux headers Fits en lecture et écriture.

Le but est de pour voir lire ou écrire les keywords Fits dans les variables Inter et vice–versa d'une manière.

Pour réaliser la correspondance entre les nom des keywords Fits et des variables Inter, on écrit le nom de la variable Inter dans le commentaire du keyword Fits. Exemple :

```
FNAME  =  value                / iname / commentaires
```

Ansi lorsque l'on lit ou écrit le keyword "FNAME", on récupère ou stocke la variable Inter "iname". Dans le cas ou "iname" n'est pas présent, et si le quelificateur "/FNAME" est donné, on accéde la variable Inter nommée "FNAME"

La lecture ou écriture de header peut se faire également au moyen d'un fichier "modèle" annexe. C'est lui qui contient la liste des keywords Fits que l'on veut traiter. Ce fichier peut soit être éditable (avec la syntaxe Fits exacte, taille du descripteur et position du "=", mais la ligne peut être plus courte), soit être un fichier Fits existant répondant (si possible) à la syntaxe donnée ci–dessus.

Dans le cas d'une lecture, si le "modèle" contient des keywords Fits inexistant dans le fichier que l'on lit, la variable Inter reste inchangée sauf si le qualificateur "/UNSET" est donné. Dans ce cas les variables numérique sont données à 9999.999 et les variables caractères sont données à "9999.999"

---

**QUALIFICATEURS A DISPOSITION:**

**/RHEADER[=<modèle>]**

Lecture d'un header Fits avec transfert des variable dans les variables Inter. Si le modèle est présent, on ne lit que les keyword donné dans le modèle.

**/WHEADER=<modèle>**

Ecriture des variables Inter dans un header FITS. Dans ce cas le modèle est obligatoire et doit être différent du fichier de sortie.

**/CPHEADER**

Copie tout les descripteurs d'un fichier <modèle> dans le fichier de sortie.

**/CPHEADER /WRITE**

Copie tout les descripteurs d'un fichier <modèle> dans fichier de sortie au moment de la création du fichier de sortie (gain de temps). ATTENTION à l'ordre de paramètre (voir ci-dessus).

**/FNAME**

Utilise le nom Fits si le nom Inter n'est pas donné selon la syntaxe ci–dessus.

**/UNSET**

En lecture uniquement pose la variable Inter à 9999.999 ou "9999.999" si le keyword Fits recherché est inexistant.

**/NOWARNING**

En situation normale, FIMAGE émet des warnings pour tout les problèmes rencontrés (variable Inter inexistante, keyword Fits inexistant, ...), mais ne génère aucune erreur. Avec /NORWARNING, aucun message n'apparaît.

**/ECHO**

Echo des opérations effectuées.

**/INQUIRE**

Echo des opérations qui seront effectuées, mais ne les effectue pas.

---

## VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE

| Variables | Mode | Description |
|---|---|---|
| NX | I/O | nb de pixel en X de la matrice 1 |
| NY | I/O | nb de pixel en Y de la matrice 1 |
| XCORIG | I/O | coord w. de dep en X de l'image d'origine 1 |
| XSTART | I/O | coord. world de depart en X de la matrice 1 |
| XSTEP | I/O | pas en X de la matrice 1 |
| YCORIG | I/O | coord w. de dep en Y de l'image d'origine 1 |
| YSTART | I/O | coord. world de depart en Y de la matrice 1 |
| YSTEP | I/O | pas en Y de la matrice 1 |

## 2.33 FIT1

Ajustement du profil stellaire sur les données d'une matrice.

---

**SYNTAXE:**

FIT1 <No de matrice> [/qualificateurs]

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **PARI** | param initial courant H |
| **PARER** | erreur maximale pour H |
| **PARIT** | nombre d'iterations pour H |
| **PARST** | pas d'iterations pour H |
| **GAIN** | Gain photoelectrons/ADU |
| **RMS** | Read-out noise in ADU |

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **PAR** | param courant H |
| **CHI** | Sigma du fit |
| **DTIME** | CPU temps en sec dernier programme |
| **NIMAX** | nombre d'iterations utilise |
| **NUSED** | nombre de pixels utilises |

---

**QUALIFICATEURS A DISPOSITION:**

**/L**

Affichage des itérations à l'écran.

**/W**

Pondération statistique des points.

**/PINCH**

Pondération fortes des points centraux.

**/FILE=<file>**

Fichier utilisé pour le debug.

---

**UTILISATION**

Le profil ajuste la forme suivante :

$$GAUSM(x,y) = H * e^{(R^2 * B)} * (1 - C * (B * R^D)) + BG$$

*où*

$$R = (xtr^2 + ytr^2 * E)$$

*et*

$$xtr = (x - x0) * cos(Fi) + (y - y0) * sin(Fi)$$
$$ytr = (y - y0) * cos(Fi) + (x - x0) * sin(Fi)$$

Les paramètres B,C,D définissent le profil radial :
– 1/B est approximativement la largeur a mi-hauteur
– C est approximativement 1
– D est approximativement 2
Les paramètres E et Fi sont l'excentricité et l'orientation de l'ellipse isophotale (coupe horizontale) :
– E <1.,infini>
– Fi angle en radians à partir de l'axe X+ dans le sens contraire des aiguilles d'une montre
– H est la hauteur du maximum et X0, Y0 sa position en coordonnées "world"
– BG est le fond de ciel local (considéré comme uniforme à travers du champ couvert par la matrice)
Le fit est controlé par les paramètres suivants :

**PARI(1 :9)** Les paramètre initiaux de la fonction à ajuster dans l'ordre H,X0,Y0,E,FI,B,C,D,BG

**PARIT(1 :9)** Le NOMBRE D'ITTERATION pour chaque paramètre.
Si un ou plusieurs PARIT=0, le paramètre correspondant n'est pas ajusté. Il garde sa valeur initale.

**PARST(1 :9)** Le facteur de relaxation pour chaque paramètre.
A chaque ittération la modification proposée par LSQ est multipliée par ce facteur. Lorsqu'on ajuste les para- metres de forme il est recommandé de réduire les PARST.
Valeurs suggérées :=0.5,0.5,0.5,0.2,0.2,0.1,0.1,0.1,0.5

**PARER(1 :9)** L'ERREUR RELATIVE tolérée sur chaque paramètre.
Lorsque la correction proposée par LSQ est $<$ on cesse d'ittérer ce paramètre.

**PRINCIPE DE FONCTIONNEMENT**

La commande procède en deux phases :

1. Initialisation :
   Les paramètres de la fonction stockées dans PAR(1 :9) sont mis aux valeurs de PARI(1 :9).
2. Itteration :
   La fonction est approximée par son développement linéaire en paramètres dont PARIT .ne. 0. On ajuste les différences des paramètres par moindre carré et on met à jour les PAR.

   $$PAR(1:9) = PAR(1:9) + dPAR(1:9) * PARST(1:9)$$

   nottez que dPAR n'est pas une VARIABLE et reste inacessible à l'utilisateur
   On vérifie pour tout les paramètres de la fonction que la condition de continuation du fit est satisfaite. A savoir :

   $PARIT > 0$ et $abs(dPAR)/PAR > PARER$

   Si ce n'est pas le cas le fit du paramètre est stoppé. Lorsque AUCUN paramètre n'est à ajuster la commande s'arrete.

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| CHI | O | Sigma du fit |
| DTIME | O | CPU temps en sec dernier programme |
| GAIN | I | Gain photoelectrons/ADU |
| NIMAX | O | nombre d'iterations utilise |
| NUSED | O | nombre de pixels utilises |
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| PARER | I | erreur maximale pour H |
| PARIT | I | nombre d'iterations pour H |
| PARI | I | param initial courant H |
| PARST | I | pas d'iterations pour H |
| PAR | I/O | param courant H |
| RMS | I | Read-out noise in ADU |
| XSTART | I | coord. world de depart en X de la matrice 1 |
| XSTEP | I | pas en X de la matrice 1 |
| YSTART | I | coord. world de depart en Y de la matrice 1 |
| YSTEP | I | pas en Y de la matrice 1 |

## 2.34 FREAD

Lecture sur fichier formatté.

---

**SYNTAXE:**

FREAD <VAR1> [<VAR2> [<VAR3> ... [<VAR20>]]] /qualif

---

**PARAMETRE:**

<**VARn**> nom d'une variable Inter, pas de vecteur.

---

**DESCRIPTION:**

Lit des nombres ou des chaînes de caractères séparées par des espaces ou des tabulateurs et/ou la ligne entière.

La variable EOF est mis 1 lors du dépassement de fin de fichier. Cette variable **doit** être testée juste après une lecture pour être valide.

---

**EXEMPLE:**

```
fread /unit=34 str
if eof goto next
```

---

**QUALIFICATEURS A DISPOSITION:**

**/UNIT=<unit>**

Lit dans le fichier ouvert sous le numéro d'unité <unit> par la commande FILE.

**/FILE=<file>**

Lit dans le fichier <file> ouvert par la commande FILE.

Attention, cette option peut être plus lente que l'option /UNIT car un INQUIRE fortran est effectué chaque fois pour déterminer le numéro d'unité logique du fichier.

**/REJECT=<texte>**

Rejette les lignes qui commencent par <texte> (jusqu'à 10 caractères).

**/LINE**

Stocke la ligne entière dans la premiere variable de la liste.

Ex: FREAD /UNIT=34 /LINE line col

Place la ligne dans "line" et lit la première colonne dans "col".

## 2.35 FRES

Calcul des résidus par rapport a une fonction.

**SYNTAXE:**

FRES [<No de matrice>] [/qualificateurs]

**DESCRIPTION:**

les résidus sont :

$$residu(1) = \frac{\sum weight_i * delta_i}{\sum weight_i} \quad (2.1)$$

$$residu(2) = \frac{\sum weight_i * abs(delta_i)}{\sum weight_i} \quad (2.2)$$

$$residu(3) = \frac{\sum weight_i * delta_i * rayon_i}{\sum weight_i} \quad (2.3)$$

$$residu(4) = \frac{\sum weight_i * delta_i^2}{\sum weight_i} \quad (2.4)$$

$$residu(5) = \frac{\sum weight_i * delta_i^2 * rayon_i}{\sum weight_i} \quad (2.5)$$

Les points sont pondérés selon la valeur du pixel ($z_i$) :

$$weigth_i = \frac{1}{rms^2 + \frac{z_i}{gain}} \quad (2.6)$$

On peut également determiner les résidus en fonction de secteur (jusqu'a 128) (SRES(1 :NSRES)). On effectue une transformation de Fourier sur ces valeur et on retourne les 9 premiers coefficients (TFA) ainsi que leur index de leur classement en ordre décroissant (TFAI).

**REMARQUE:**

Cette commande gère les points éliminés.

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **VOLR** | rayon d'integration |
| **FONCT** | nom de la fct |
| **PFCT** | parametre fonction |
| **GAIN** | Gain photoelectrons/ADU |
| **RMS** | Read-out noise in ADU |

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **RESIDU** | Residu –> DELTA |
| **SRES** | residu=f(segment) –> WI*DELTA**2 |
| **NSRES** | nb de segment (nb de SRES valable) |
| **TFA** | Coeff de la TF angulaire |
| **TFAI** | Index Coeff de la TF angulaire |
| **NTFA** | Nb de coeff utile dans TFA(i) |

**QUALIFICATEURS A DISPOSITION:**

**/RAYON**

Effectue le calcul uniquement sur les points inclus dans le cercle de rayon VOLR (en coord. world), centré en PFCT(2) ;PFCT(3).

**/CENTER**

Pose le centre du cercle au milieu de la matrice (voir /RAYON).

**/CENTER=(**$X_o, Y_o$**)**

Pose le centre du cercle en $< X_o; Y_o >$.

**/CENTER=(@X :**$Delta_x$**,@Y :**$Delta_y$**)**

Calcule les coordonnées du centre en offset pixel depuis le coin bas–gauche de la matrice. L'offset (1 ;1) donne le premier pixel bas-gauche.

**/NB[=<nb>]**

Calcul les résidus en fonction de secteur. On donne le nombre de secteurs a considérer sous la forme :

$$nb\_de\_secteur = 2^{nb}$$

Par défaut <nb> vaut 3 (8 secteurs) et on ne prend que 128 secteurs au maximum.

**/ADJUST**

Prend PFCT(5) comme origine des secteurs (en [radian]). Par défaut l'origine est a 0.

**/NOWEIGHT**

Empêche la pondération

**/LIST**

Affiche des statistiques sur le travail effectué. Lors de la recherche des résidus sur des secteurs, et en mode DEBUG, cette commande fournit quelques indications de test.

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| FONCT | I | nom de la fct |
| GAIN | I | Gain photoelectrons/ADU |
| NSRES | O | nb de segment (nb de SRES valable) |
| NTFA | O | Nb de coeff utile dans TFA(i) |
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| PFCT | I | parametre fonction |
| RESIDU | O | Residu –> DELTA |
| RMS | I | Read-out noise in ADU |
| SRES | O | residu=f(segment) –> WI*DELTA**2 |
| TFAI | O | Index Coeff de la TF angulaire |
| TFA | O | Coeff de la TF angulaire |
| VOLR | I | rayon d'integration |
| XSTART | I | coord. world de depart en X de la matrice 1 |
| XSTEP | I | pas en X de la matrice 1 |
| YSTART | I | coord. world de depart en Y de la matrice 1 |
| YSTEP | I | pas en Y de la matrice 1 |

## 2.36 GENERATION

Crée une ou plusieurs étoiles synthétiques dans une matrice de travail ou dans un fichier image MIDAS de type BDF.

---

**SYNTAXE:**

GENERATION [<No de matrice>] [/qualificateurs]

---

**DESCRIPTION:**

Il peut exister plusieurs fonctions permettant de générer l'image de l'étoile. Ces fonctions ont jusqu'a 10 paramètres. La fonction est nommée dans FONCT(1) et les paramètres sont décrits par les variables PFCT(1) à PFCT(10), ce sont les paramètres par défaut. On a comme paramètres communs a toutes les fonctions :

| | |
|---|---|
| PFCT(1) | hauteur |
| PFCT(2) | coordonnée centrale en X |
| PFCT(3) | coordonnée centrale en Y |

les autres paramètres dépendent de la fonction.

GENERATION fonctionne selon deux modes.
- **mode interactif :** on pose une étoile après l'autre en appelant GENERATION à chaque fois. Les paramètres non résolus sont pris dans les valeurs par défaut.
- **mode automatique :** les paramètres décrivant les étoiles a générer sont placés dans une table MIDAS.
  Les paramètres dont les colonnes ne sont pas nommées dans la table sont pris dans les valeurs par défaut. Les paramètres nuls ne sont pas pris en compte et c'est la précédente valeur qui est utilisée ou la valeur par défaut.

La fonction et ses paramètres sont choisis selon les priorités suivantes :
- ceux fournis par les variables du bloc de données.
- ceux donnés sur la ligne de commande avec /FUNCTION.
- ceux donnés dans la table avec /TABLE.

Les étoiles peuvent êtres sommées ou soustraites de l'image principale. Elles sont additionnées par défaut et soustraites si le qualificateur /MINUS est précisé.

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **FONCT** | nom de la fct |
| **PFCT** | parametre fonction |
| **SEUILG** | seuil de calcul de la fonctio |

---

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **FONCT** | Voir plus haut sous "VARIABLES INTERACTIVES PRINCIPALES" |
| **PFCT** | Voir plus haut sous "VARIABLES INTERACTIVES PRINCIPALES" |

---

**QUALIFICATEURS A DISPOSITION:**

**/MINUS**

L'image résultante est soustraite a l'image originale au lieu d'être sommée.

**/IMAGE=<image>**

Précise le nom d'un fichier image MIDAS existant

**/TABLE=<table>**

Précise le nom d'une table MIDAS contenant les paramètres des étoiles a créer. Cette table peut contenir les colonnes suivantes pour remplacer les valeurs par défauts :
X, Y, INT, FUNCTION, P4, P5, P6, P7, P8, P9 et P10.

**/X=<x> /Y=<y> /H=<h>**

Permet de changer les valeurs par défaut :

| | |
|---|---|
| **/H=<h>** | hauteur |
| **/X=<x>** | coordonnée selon l'axe X |
| **/Y=<y>** | coordonnée selon l'axe Y |

**/FUNCTION[=[<fonction>][,][<P4> ...[,<P10>]]]**

Décrit la fonction remplaçant les valeurs par défaut.
Met a jour les variables FONCT et PFCT(4) à PFCT(10) s'il sont précisés. C'est le seul cas où une modification dans le bloc de données a lieu.

**/SELECT**

Seule les lignes sélectionnées dans la table sont utilisée.
(Qualificateur par défaut)

**/NOSELECT**

Toutes les lignes de la table sont utilisées

**/CFUNCTION=<nom>**

Donne le nom de la colonne FUNCTION

**/CX=<nom> /CY=<nom> /CINT=<nom>**

Donne le nom des colonnes dont les défauts sont X, Y et INT

**/CP4=<nom> /CP5=<nom> /CP6=<nom> /CP7=<nom> /CP8=<nom> /CP9=<nom> /CP10=<nom>**

Donne le nom des colonnes des paramètres 4 à 10

---

**REMARQUE:**

La fonction est calculée sur une zone carrée. La taille de cette zone est calculée expérimentalement en déterminant a quelle distance du centre la valeur de la fonction est inférieure a SEUILG. Si SEUILG vaut zéro, il est posé a 0.001 .

## VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE

| Variables | Mode | Description |
|---|---|---|
| FONCT | I/O | nom de la fct |
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| PFCT | I/O | parametre fonction |
| SEUILG | I | seuil de calcul de la fonctio |
| XSTART | I | coord. world de depart en X de la matrice 1 |
| XSTEP | I | pas en X de la matrice 1 |
| YSTART | I | coord. world de depart en Y de la matrice 1 |
| YSTEP | I | pas en Y de la matrice 1 |

## 2.37 GET

Lit les coordonnées pointées par la souris sur l'image de l'afficheur.

---

**SYNTAXE:**

GET

GET /ECHO

GET /TIMEOUT=<delai> /RETURN=<valeur>

---

**DESCRIPTION:**

Attend un click sur la souris. Puis rend le numéro du bouton appuyé, les coordonnées pixel (origine <0 ;0>), les coordonnés world (origine <xstart_aff ;ystart_aff>) et la valeur du pixel pointé.

---

**QUALIFICATEURS A DISPOSITION:**

**/ECHO**

Affiche à l'écran les données issues du get.

**/TIMEOUT=<delai>**

Rends la main apres le <delai> donné en seconde. Dans ce cas les coordonnées (AFFPCU, AFFWCU) et la valeur du pixel (AFFVAL) sont retournées à "-9999". Le numéro du bouton (AFFRET) est retourné à zéro, ou à la valeur précisée par le qualificateur "/RETURN". S'il y a eu un déplacement du curseur sur l'image ou le zoom ou si une touche du clavier est enfoncée, le timer du timeout est relancé lorsqu'il arrive à terme.

**/RETURN=<valeur>**

Donne la valeur de retour (AFFRET) en cas de retour par timeout.

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **AFFPCU** | coord pixel X curs No 1 |
| **AFFWCU** | coord world X curs No 1 |
| **AFFVAL** | valeur du pixel |
| **AFFRET** | code de retour lors d'un get |

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| AFFPCU | O | coord pixel X curs No 1 |
| AFFRET | O | code de retour lors d'un get |
| AFFVAL | O | valeur du pixel |
| AFFWCU | O | coord world X curs No 1 |

## 2.38 GLOBAL

Permet de créer des variables globales depuis une procédure

---

**SYNTAXES:**

GLOBAL <variable> ...

GLOBAL <variable>[=<expression>] ...

---

**DESCRIPTION:**

Ces variables sont reconnues à tout les niveaux d'imbication de procédure et au niveau interactif. D'un usage similaire à la commande LOCAL, elle ne peut toutefois s'utiliser que si aucune variable globale n'a été initilisée.

Son usage principal est de l'utiliser dans des procédure d'initialisations.

---

**EXEMPLE:**

```
global i j k
global ramp(10)=setv(1,10)
global name=""
global instru="telescope"
global maxval=minval+minval*0.5
```

---

**REMARQUE:**

Souvenez-vous :

```
PROMPT > GLOBAL i=10 j=20
PROMPT > GLOBAL i=20 j=i k=i+j
PROMPT > SHOW i j k
   I(001) =  20.000000
   J(001) =  10.000000
   K(001) =  30.000000
```

---

**QUALIFICATEURS A DISPOSITION:**

**/N=<n>**

Initialise la taille de la table de hashing à <n>. <n> vaut 200 par défaut. Si on sait que l'on va utiliser plus de 200 variables pour une profondeur de procédure, il est interressant de donner ce qualificateur. Mais si les table deviennent trop petites l'augmentation des tables est faites de manière automatique. Donc pas de soucis particulier....

## 2.39 GOTO

GOTO avec des labels symboliques.

---

**SYNTAXE:**

GOTO <label>

---

**EXEMPLE:**

```
goto stop
...
stop:
...
```

## 2.40 GSC

Permet divers commandes sur le générateur de cartes stellaires "gsc" ou "xgsc".

---

**SYNTAXE:**

GSC [/qualif]

---

**DESCRIPTION:**

Le générateur de carte accède une partie des données du Guide Star Catalog, lequel regroupe plus de 18 millions d'objets stellaires ayant jusqu'à une magnitude limite de 16. Il contient les coordonnées alpha et delta pour l'an 2000 et la magnitude.
La connection avec ce logiciel peut ce faire en tout temps. Il suffit de tapez la commande "GSC /CLIENT" sous Inter. Si la connection ne se réalise pas il faut toujours essayer une deuxième fois avant de chercher plus loin ! !

---

**QUALIFICATEURS A DISPOSITION:**

**/CLIENT**

Lance GSC et s'y connecte.

**/ALPHA=<alpha>**

Donne la position alpha du centre du champ.

**/DELTA=<delta>**

Donne la position delta du centre du champ.

**/YEAR=<année>**

Donne l'année des coordonnées d'entrée.

**/SCALE=<scale>**

Donne l'échelle du champ selon le format utilisé avec gsc.

**/MAGNITUDE=<min>,<max>**

Donne les magnitudes limites à représenter.
Si <max> n'est pas donné alors on prend :

$$< max >=< min > +2$$

$$< min >=< min > -1$$

**/GET**

Récupère dans ALPHA, DELTA, CALPHA, CDELTA et MAGNI la coordonnée et la magnitude de l'objet le plus proche du curseur au moment du clic sur le bouton du milieu de la souris. Si on tape <SPACE> alors le curseur n'est pas déplacé, la coordonnée est prise à la position courante et MAGNI est rendu à 99. Dans ces deux cas STATUS est rendu a 0. Un clic sur le bouton de gauche affiche les informations sur l'étoile la plus proche alors qu'un click sur le bouton de droite annule l'opération et retourne STATUS à 1.

**/ECHO**

Affiche la position et la magnitude de l'objet pointé.

**/XYGET**

Récupère dans GSCX et GSCY la position <x ;y> du curseur ainsi que dans les variables "GSC*" les paramètres servant a la conversion <x ;y> en <alpha ;delta>. L'emploi des boutons de la souris et de la touche <Return> est le même que pour "/GET". Cette fonctionnalité est utilisées en conjonction avec la commande ALIGN qui permet de retrouver les coordonnées alpha et delta d'une position <x ;y> sur l'afficheur.,

La conversion s'effectue avec les fonctions intrinsèques suivante :

$$alpha = CAL(x, y)$$

$$delta = CDE(x, y)$$

**/NOAFFICH**

La zone n'est pas dessinée lors de la définition d'un nouveau champ.

**/AFFICH**

Affiche le champ courant. Cette opération a lieu par défaut lorsque l'on définit un nouveau champ.

**/CLEAR**

Efface le contenu de la fenêtre contenant le champ.

**/MGET**

Recherche au maximum 20 objets les plus proches du centre du du champ. Retourne le nombre d'objets dans NPT, les coordonnées écran dans XRET(i) et YRET(i) des objets triées par ordre croissant de distance au centre (DRET(i)) et leurs magnitudes GSC dans ZRET(i).

**/ADGET**

Récupère dans ALPHA, DELTA, CALPHA, CDELTA et MAGNI, la coordonnée et la magnitude de l'objet dont la position écran est donnée dans XRET(1) et YRET(1).

La séquence suivante retourne la coordonnée de l'objet le plus proche du centre :

```
GSC /MGET /ADGET
```

**/IGET**

Dump de l'image dans le fichier zgsc.dump. Format ascii 2 bytes par pixel. Recupère le tilt

**/EXTRACT**

Récupère l'image (4bits) de GSC dans une matrice

**/CENTER**

retourne les coordonnées du centre et l'échelle

**/CENTER**

retourne les coordonnées du centre et l'échelle

**/SIZE[=<x>[,<y>]]**

Donne la taille de la fenêtre image (défaut 512∗512)

**/POS[=<x>[,<y>]]**

Place la fenêtre image sur l'écran (défaut 0 ;0)

**/PRINT**

Lance l'impression

**/TITLE=<titre>**

Donne le titre sur le document imprimé

**/LABEL=<No_label>,<label>**

Donne le label de la ligne <No_label> sur le document imprimé. Le No de ligne est compris dans la gamme (0...9), le label de la ligne 0 contient par défaut les coordonnées et la date du centre du champ.

**/QUIT**

Tue gsc.

**/CLOSE**

Met gsc sous forme d'icône, le tableau de commande ainsi qu l'image disparaissent.

**/OPEN**

Ouvre l'icône pour afficher le tableau de commande ainsi qu l'image.

**/HIDE**

Cache l'image. Elle peut être réaffichée avec "/SHOW" ou par le bouton "REDISPLAY".

**/SHOW**

Réaffiche l'image après un "/HIDE" ou après avoir enlevé la punaise.

**/VGOP=<n>**

Niveau de verbosité du protocole de communication (0 à 9).

---

**REMARQUE:**

Les options caractérisant aff et la fenêtres sont placées dans la variable CMDGSC. Typiquement :

CMDGSC="-v 2 -Wp 100 200"

Les options de gsc sont décrites sur une page de "man" (voir man gsc ou man xgsc).

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| ALPHA | I/O | Position de l'objet clique (degres decimaux) |
| CALPHA | I/O | ALPHA sous forme caractere |
| CDELTA | I/O | DELTA sous forme caractere |
| CMDGSC | I | options pour gsc |
| DELTA | I/O | Position de l'objet clique (degres decimaux) |
| DETSCX | I/O | taille d'un pixel en X |
| DETSCY | I/O | taille d'un pixel en Y |
| DRET | I/O | distance au centre |
| GSCA0 | I/O | te pour la translation de <x ;y> en <a ;d> |
| GSCAX | I/O | te pour la translation de <x ;y> en <a ;d> |
| GSCAY | I/O | te pour la translation de <x ;y> en <a ;d> |
| GSCBX | I/O | te pour la translation de <x ;y> en <a ;d> |
| GSCBY | I/O | te pour la translation de <x ;y> en <a ;d> |
| GSCD0 | I/O | te pour la translation de <x ;y> en <a ;d> |
| GSCX0 | I/O | te pour la translation de <x ;y> en <a ;d> |
| GSCX | I/O | te pour la translation de <x ;y> en <a ;d> |
| GSCY0 | I/O | te pour la translation de <x ;y> en <a ;d> |
| GSCY | I/O | te pour la translation de <x ;y> en <a ;d> |
| MAGNI | I/O | retour magnitude objet pointe |
| NPT | I/O | Nb de valeurs retournees |
| NX | I/O | nb de pixel en X de la matrice 1 |
| NY | I/O | nb de pixel en Y de la matrice 1 |
| STATUS | O | status pour usage general |
| TILT | I/O | Tilt des images dss |
| XRET | I/O | coordonees X en retour |
| YRET | I/O | coordonees Y en retour |
| ZRET | I/O | densite en retour |

## 2.41 IF

Intruction de contrôle, d'usage similaire au fortran

---

**SYNTAXES:**

IF <expression> THEN

IF <expression> <commande>

---

**PARAMETRES:**

| | |
|---|---|
| <**expression**> | expression dont le résultat est un numérique |
| <**commande**> | commande Inter |

---

**DESCRIPTION:**

Les instructions suivant le IF sont exécutée si le résultat de <expression> est différent de 0.
Il est essentiel de préserver des espaces autour de <expression>.

---

**EXEMPLES:**

```
if (i.gt.maxval) then
   write "Erreur: Dépassement"
else
   call nextstep
endif
```

ou

```
if (i.gt.maxval) write "Erreur: Dépassement"
```

## 2.42 INPUT

Redirige l'entrée standard d'inter sur un fichier.

---

**SYNTAXE:**

INPUT <fichier>

---

**DESCRIPTION:**

Les lignes de commandes sont lues dans un fichier et immédiatement interprétée. Il n'y a pas de compilation, donc les ordres de contrôle sont interdits.

Le fichier a une extension '.prc' et doit se trouver dans un des directories écris dans la variable DIRPRC.

Dans ce mode, l'arrêt sur erreur est pris en compte (voir la commande ERREUR).

## 2.43 INQUIRE

Permet l'entrée d'une variable en posant une question.

---

**SYNTAXE:**

INQUIRE <variable>=<texte>

---

**DESCRIPTION:**

La variable peut être de type caractère ou numérique. Si on tape <RETURN> comme réponse, la variable n'est pas modifié. Si on demande un vecteur numérique, on est questionné pour chaque position de celui-ci.

Ex: a(1 :3)="entrez 3 valeurs "

Ex: passwd="Entrez le mot de passe "

Les réponses peuvent être données sous forme d'expression.

Si une réponse est incorrecte (type incorrect) la question est reposée.

## 2.44 INTERPRETE

Initialisation interne.

---

**SYNTAXE:**

INTERPRETE

---

**DESCRIPTION:**

INTERPRETE s'utilise soit pour sortir du mode compilation interactif, soit pour regagner le niveau zéro de l'accès aux variables après un <Ctrl-C>.

En effet, Inter recherche les variables selon le niveau de procédure pour lequel ils sont définis. Puis, s'ils n'ont pas été trouvé la recherche se fait au niveau zéro (block de données et variables locales définis en interactif). Lors d'une interruption par un <Ctrl–C>, la procédure est suspendue et on se retrouve en mode interactif. Malgré le fait de se retrouver en interactif, les variables de la procédures restent accessibles. On remarque donc que s'il y a des synonymes entre les variables de la procédure et ceux du niveau zéro, ces derniers ne sont plus accessibles. La commande INTERPRETE permet d'oublier la procédure en cours et d'accéder uniquement les variables du niveau zéro.

## 2.45 LOAD

Charge un fichier image sur l'afficheur.

---

**SYNTAXE:**

LOAD <Nom du fichier>

---

**DESCRIPTION:**

Ainsi, si l'image est ouverte par Inter (on le voit avec IMAGE /STATUS), il faut la fermer (IMAGE /CLOSE) avant d'effectuer un LOAD.

De même, si l'image est présente sur l'afficheur, il ne faut par la charger sous Inter par "EXTRACT image".

Ces prescriptions forcent la commande LOAD à être utilisée uniquement pour, par exemple, lire des coordonnées, extraire une partie de l'image (EXTRACT AFF), charger des images pour une démonstration, ...

---

**REMARQUE:**

Attention : Inter et l'afficheur ne doivent pas partager une image (problèmes liés à l'ouverture d'une image par plusieurs process).

## 2.46 LOCAL

Permet de créer des variables locales à une procédure

---

**SYNTAXES:**

LOCAL <variable> ...

LOCAL <variable>=<expression> ...

---

**DESCRIPTION:**

Ces variables sont reconnues dans la procédure courante et dans les macros (proc) lancées depuis cette procédure. Si on utilise cette commande au niveau interactif (présence du prompt) ces variables sont accessibles tout au long de la session.

On ne peut pas detruire une variable locale déclarée au niveau interatif.

De plus, on peut créer des variables dimensionnées et leur donner une valeur d'initialisation.

Les variables sont crées numériques par défaut, on crée une variable de type caractère en donnant une initialisation de type caractères.

---

**EXEMPLE:**

```
local i j k
local ramp(10)=setv(1,10)
local name=""
local instru="telescope"
local maxval=minval+minval*0.5
```

---

**REMARQUE:**

Souvenez-vous :

```
PROMPT > LOCAL i=10 j=20
PROMPT > LOCAL i=20 j=i k=i+j
PROMPT > SHOW i j k
   I(001) =  20.000000
   J(001) =  10.000000
   K(001) =  30.000000
```

---

---

**QUALIFICATEURS A DISPOSITION:**

**/N=<n>**

Initialise la taille de la table de hashing à <n>. <n> vaut 200 par défaut. Si on sait que l'on va utiliser plus de 200 variables pour une profondeur de procédure, il est interressant de donner ce qualificateur. Mais si les table deviennent trop petites l'augmentation des tables est faites de manière automatique. Donc pas de soucis particulier....

**/GLOBAL**

Initialise une variable globale. Utilisez plutot la commande GLOBAL

## 2.47 MATRIX

Divers utilitaires interagissant avec les matrices de travail, tel que :

1. Affichage
2. Allocation et désallocation
3. Opérations numériques
4. Accès aux fichiers
5. Accès aux bases de données type RDB

---

### 1. AFFICHAGE

---

**SYNTAXES:**

MATRIX [<No matrice>] [/VIEW]

MATRIX [<No matrice>] /OK

MATRIX [<No matrice>] /ELIM

MATRIX [<No matrice>] /SAMPLE

---

**DESCRIPTION:**

Lors d'un affichage à l'écran, on peut donner la taille de celui-ci au moyen des variables MATNLI (nb de lignes) et MATNCL (nb de colonnes, donné en caractères). A ce moment là, les valeurs sont arrondies à des valeurs entières. Le nombre de digits affichables est donné par la variable MATNDG.

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **MATNCL** | nombre de colonne sur l'ecran |
| **MATNLI** | nombre de lignes |
| **MATNDG** | nombre de digits |

---

**QUALIFICATEURS A DISPOSITION:**

**/VIEW**

Affiche la matrice en partant du coin bas gauche (défaut si aucun qualificateur n'est donné).

**/OK**

Affiche uniquement les points non éliminés.

**/ELIM**

Affiche uniquement les points éliminés.

**/SAMPLE=<échantillonnage en x>[,<échantillonnage en y>]**

Permet l'échantillonnage lors de l'affichage à l'écran.

## 2. ALLOCATION ET DESALLOCATION

**SYNTAXES:**

MATRIX [<No matrice>] /SET

MATRIX [<No matrice>] /FREE

MATRIX /FREE /ALL

MATRIX /LIST

**DESCRIPTION:**

Permet la gestion de l'allocation des matrices ansi que la visualisation de l'arrangement des matrices en mémoire.

**QUALIFICATEURS A DISPOSITION:**

**/SET**

Initialise une matrice selon les variables NX et NY.

**/FREE**

Tue une matrice.

**/ALL**

Avec /FREE, tue toutes les matrices.

**/LIST**

Liste la taille des matrices actuelles et affiche une représentation de la mémoire dynamique utilisée par les matrices et les images midas ouvertes.

## 3. OPERATIONS NUMERIQUES

**SYNTAXES:**

MATRIX [<No matrice>] /CLEAR

MATRIX [<No matrice>] /ABS

**DESCRIPTION:**

Divers opérations numériques.

**QUALIFICATEURS A DISPOSITION:**

**/CLEAR**

Remplit la matrice de zéro.

**/ABS**

Donne la valeur absolue à chaque point.

## 4. ACCES AUX FICHIERS

**SYNTAXES:**

MATRIX [<No matrice>] /READ

MATRIX [<No matrice>] /READ /FMT

MATRIX [<No matrice>] /READ /COLUMNS [/VERIF]

MATRIX [<No matrice>] /READ /DIRECT [/RECL] [/FLOAT]

MATRIX [<No matrice>] /READ /DIRECT [/RECL] [/[UN]SIGNED] /INT

MATRIX [<No matrice>] /READ /DIRECT [/RECL] [/[UN]SIGNED] /SHORT

MATRIX [<No matrice>] /READ /DIRECT [/RECL] [/[UN]SIGNED] /BYTE

MATRIX [<No matrice>] /READ [/SEQUENTIAL] [/RECL] [/FLOAT]

MATRIX [<No matrice>] /READ [/SEQUENTIAL] [/RECL] [/[UN]SIGNED] /INT

MATRIX [<No matrice>] /READ [/SEQUENTIAL] [/RECL] [/[UN]SIGNED] /SHORT

MATRIX [<No matrice>] /READ [/SEQUENTIAL] [/RECL] [/[UN]SIGNED] /BYTE

MATRIX [<No matrice>] /SAVE

MATRIX [<No matrice>] /SAVE /FMT

MATRIX [<No matrice>] /SAVE /DIRECT /FLOAT

MATRIX [<No matrice>] /SAVE /DIRECT /[UN]SIGNED /INT

MATRIX [<No matrice>] /SAVE /DIRECT /[UN]SIGNED /SHORT

MATRIX [<No matrice>] /SAVE /DIRECT /[UN]SIGNED /BYTE

MATRIX [<No matrice>] /SAVE [/SEQUENTIAL] [/FLOAT]

MATRIX [<No matrice>] /SAVE [/SEQUENTIAL] [/[UN]SIGNED] /INT

MATRIX [<No matrice>] /SAVE [/SEQUENTIAL] [/[UN]SIGNED] /SHORT

MATRIX [<No matrice>] /SAVE [/SEQUENTIAL] [/[UN]SIGNED] /BYTE

---

**DESCRIPTION:**

Matrix permet l'accès aux fichiers binaires ou Ascii en lecture ou écriture.

---

**QUALIFICATEURS A DISPOSITION:**

**/SAVE[=<nom de fichier>]**

Sauve le contenu de la matrice dans un fichier. Ecrit par défaut dans le fichier "mat.out" sous forme flottante et accès séquentiel.

**/READ[=<nom de fichier>]**

Lit le contenu du fichier et le place dans une matrice. Lit par défaut le fichier "mat.out" sous forme flottante et accès séquentiel.

Les lignes commençant par un "#", dans les fichiers formattés, ne sont pas prises en compte.

La taille de la matrice est mise à jour selon le nombre de colonnes et de lignes lues dans le fichier.

**/RECL=<longueur des lignes en bytes>**

Avec /READ uniquement et pour les fichiers binaires, donne la longueur des lignes en bytes. Pris par défaut comme : NX∗nb_de_bytes_pour_un_élément

Rappel : flottant=4bytes entier∗4=4bytes entier∗2=2bytes bytes=1byte

Ex: : MAT /READ=file.dat /DIRECT /RECL=24

**/COLUMNS[=<No de colonne> | <Nom de colonne>,...]**

Avec /READ uniquement, indique qu'on lit un fichier formatté et donne, optionnellement, les numéros ou les noms des colonnes à lire. **Le nombre de colonnes décrites est limité à 40.**

Ex: : MAT /READ=file.dat /COLUMNS=3,2,"alpha","delta"

Si aucun numéro ou nom de colonne n'est donnée, tout les nombres sont pris, et le fichier n'a pas besoin d'être mis en colonnes (lecture en format libre).

Si on donne le numéro de colonne, le fichier n'a pas besoin d'être mis en colonnes (lecture en format libre).

Si on donne le nom de colonne, celui-ci doit être spécifié dans le fichier sur une ligne commencant par "#d" et suivit du nom des colonnes. Dans ce cas le format du fichier est très strict :

Les colonnes commencent sur le premier caractère du nom de la colonne et se termine juste avant le début de la colonne suivante. De plus la première colonne commence avec le premier caractère. Exemple :

```
#resultats
#d no   count  moyenne  delta  alpha
 0001    10    5.02     1.0044 2.435325
 0002    14    5.01     12.453 2.678768
 0003    11    5.01     8.4354 2.858576
```

**/VERIFY**

Avec /COLUMNS spécifiés avec des noms de colonnes, donne les limites des colonnes à lire (debug).

**/FMT[=<format>]**

Avec /SAVE, définit le format d'écriture pour une sortie en colonne dans un fichier formatté (défaut "G13.7").

On donne le format pour l'inscription d'une ou plusieurs colonnes. Attention tout les formats doivent être soit entier (I, O, Z) ou réel (F, G).

Les parenthèses sont facultatives.

Ex : /FMT="(1x,i3,z2)", /FMT=f6.1, ...

Avec /READ, signale que le fichier est formatté. Si /COLUMNS est déjà précisé, /FMT n'est pas necéssaire.

**/SEQUENTIAL**

Mode d'accès aux enregistrements du fichier binaire. C'est le défaut.

(rem : ce mode place 4 bytes en début et fin de chaque enregistrement.)

**/DIRECT**

Mode d'accès aux enregistrements du fichier binaire.

(rem : le fichier contient uniquement les données. Il n'y a pas de structure, pas même d'information sur le nombre de colonnes ou de lignes)

**/SIGNED**

Ecrit des entiers∗4, des entiers∗2 ou des bytes signés. C'est le défaut.

**/UNSIGNED**

Ecrit des entiers∗4, des entiers∗2 ou des bytes non signés.

**/FLOAT**

Ecrit ou lit du binaire flottant.

Ex: : MAT /SAVE=file.dat /FLOAT /DIRECT

**/INT**

Ecrit ou lit du binaire entier∗4 (imprécis au dessus de 16777216).

Ex: : MAT /READ=file.dat /INT /DIRECT /RECL=1024

**/SHORT**

Ecrit ou lit du binaire entier∗2 (0...+65536 ou -32768...+32767).

Ex: : MAT /READ=file.dat /SHORT /UNSIGNED /DIRECT

**/BYTE**

Ecrit ou lit du binaire byte (0...+255 ou -128...+127).

Ex: : MAT /SAVE=file.dat /BYTE /UNSIGNED /SEQUENTIAL

---

## 5. ACCES AUX BASES DE DONNEES TYPE RDB

---

**SYNTAXES:**

MATRIX [<No matrice>] /RDB /READ

MATRIX [<No matrice>] /RDB /READ /COLUMNS [/VERIF]

MATRIX [<No matrice>] /RDB /SAVE

---

**DESCRIPTION:**

Matrix permet l'accès aux fichiers RDB en lecture ou écriture. **Attention, les fichiers RDB traités sous Inter doivent contenir uniquement des champs numériques.** Il faut donc fabriquer une table intermédiaire avec les colonnes numériques si la table originale contient des chaînes de caractères.

---

**QUALIFICATEURS A DISPOSITION:**

**/RDB**

Avec /READ, lit un fichier de type rdb (base de données).

**/RDB[=<fichier_à_entête_rdb>]**

Avec /SAVE, sauve la matrice dans un fichier rdb, le nom des colonnes est donné par défaut selon le format suivant : "cXXX" ; avec XXX comme No de colonne.

L'entête rdb peut être reprise tel quel d'un fichier rdb existant en précisant son nom.

**/READ /SAVE /COLUMNS /VERIF**

Voir plus haut.

---

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| MATCOU | I | Nb de couches pour la matrice(i) |
| MATNCL | I | nombre de colonne sur l'ecran |
| MATNDG | I | nombre de digits |
| MATNLI | I | nombre de lignes |
| MATSIZ | I | Nb de pixel pour la matrice(i) |
| NX | I/O | nb de pixel en X de la matrice 1 |
| NY | I/O | nb de pixel en Y de la matrice 1 |
| XSTART | I | coord. world de depart en X de la matrice 1 |
| XSTEP | I | pas en X de la matrice 1 |
| YSTART | I | coord. world de depart en Y de la matrice 1 |
| YSTEP | I | pas en Y de la matrice 1 |

## 2.48 MOM

.

---

**SYNTAXE:**

MOM [<No matrice>] [<BGFR>][/qualificateur]

---

**QUALIFICATEURS A DISPOSITION:**

**/FREE**

**/FIXE**

**/LIST**

---

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| BGCLIP | I | clip. fact. : sigma*BGCLIP |
| BGCONV | I | clip. converg : \|diffBG/sigma\|<BGCONV |
| BGFLAG | I/O | flag du background |
| BGFR | I/O | fraction centrale |
| BGMED | I | BG=BGMED*median+(1-BGMED)*mean |
| BGSIGM | I/O | sigma of last background |
| BG | I/O | valeur initiale du background |
| HIN | O | pic central initial |
| NM | I | Numero de la matrice courante |
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| XIN | O | x-init pic 1 |
| XSTART | I | coord. world de depart en X de la matrice 1 |
| XSTEP | I | pas en X de la matrice 1 |
| YIN | O | y-init peak 1 |
| YSTART | I | coord. world de depart en Y de la matrice 1 |
| YSTEP | I | pas en Y de la matrice 1 |

## 2.49 NEXTWINDOW

Passe automatiquement à la window suivante lors d'opérations graphiques. Gère l'impression sur l'imprimante ou l'effacement de l'écran lorsque la page est remplie.

---

**SYNTAXE:**

NEXTWINDOW

---

**DESCRIPTION:**

Cette commande utilise les variables WDX,WDY,NWX,NWY,LWX et LWY pour connaître la position de la window courante et met simplement à jour WDX et WDY qui sont les coordonnées de la prochaine window. Si la position de la nouvelle window est sur la page suivante alors la page graphique est effacée, ou l'ordre d'impression est envoyé, dépendant du type de station.

Si la dernière page n'est pas éjectée, il faut envoyer la commande Inter "PRINT".

---

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| LWX | I | largeur de la window en X |
| LWY | I | largeur de la window en Y |
| NWX | I | nb de subwindow en X |
| NWY | I | nb de subwindow en Y |
| WDX | I/O | No subwindow courante en X |
| WDY | I/O | No subwindow courante en Y |

## 2.50 NOISE

Somme un bruit gaussien sur une matrice de travail ou dans un fichier MIDAS

---

**SYNTAXE:**

NOISE [<No matrice>] [/qualificateurs]

---

**DESCRIPTION:**

Le bruit est calculé par la fonction :

$$\text{bruit} = \text{bruit\_gaussien} \cdot \sqrt{\text{rms}^2 + \frac{\text{valeur\_pixel}}{\text{gain}}}$$

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **GAIN** | Gain photoelectrons/ADU |
| **RMS** | Read-out noise in ADU |
| **RANDOM** | initialisation du generateur |

---

**QUALIFICATEURS A DISPOSITION:**

**/IMAGE=<image>**

pour préciser une image MIDAS

**/START**

Initialise le générateur de nombre aléatoire avec la valeur de RANDOM

---

**REMARQUE:**

Le bruit est gaussien de moyenne nulle et de variance = 1.

---

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| GAIN | I | Gain photoelectrons/ADU |
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| RANDOM | I | initialisation du generateur |
| RMS | I | Read-out noise in ADU |

## 2.51 ONED

Dessine le profil radial du contenu d'une matrice de travail

---

**SYNTAXE:**

ONED [<No matrice>] [/qualif=(option[,...])]

---

**DESCRIPTION:**

Le profil radial représente sur un graphique à deux dimensions les valeurs de chaque point d'une image ramenées le long d'un axe. La position en X donne le rayon elliptique calculé de la sorte :

$$\begin{aligned} dX &= (X_i - X_c) \\ dY &= (Y_i - Y_c) \\ X_{ellip} &= dX * cos\varphi + dY * sin\varphi \\ Y_{ellip} &= dY * cos\varphi - dX * sin\varphi \\ R_{ellip} &= \sqrt{X_{ellip}^2 + \varepsilon^2 * Y_{ellip}^2} \end{aligned}$$

$$\begin{aligned} avec : & \\ \epsilon &= PAR(4) \\ \varphi &= PAR(5) \end{aligned}$$

Et la position Y affiche soit des valeurs moyennes avec barres d'erreur pour un échantillonnage donné par PLRSTP soit les valeurs de chaque points lorsque PLRSTP vaut 0. De plus le profil théorique peut être dessiné par dessus les points.

ONED peut rejeter des points s'ils sont à l'extérieur de la zone de travail. Cette zone est définie entre deux angles, PLALMN et PLALMX (angles mini et maxi exprimés en degré), et entre deux rayons (PLRMIN et PLRMAX) pris en compte seulement avec l'option /RDEF. La zone peut être doublée selon une symétrie centrale si l'angle d'ouverture est plus petit que 180 degré avec l'option /SYM ou la variable PLSYM.

L'image peut être tournée d'un angle (PAR(5)) avant son analyse si le référent est donné "PHI" avec l'option /REF ou la variable PLREF. Aucune rotation n'est effectuée si le référent est "X".

L'echelle verticale est fixée automatiquement si on ne précise pas /ZDEF. Dans ce cas la valeur maximum vaut le plus haut point ou la valeur maximum de la fonction (PAR(1) si on donne /FUNC.

Les caractères des labels, graduation et titres sont controllés par les variables GFLAG, GT* et GL*

---

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **GDRIVE** | marge pour la feuille 0=pas <0=mm >0=marge |
| **PAR** | param courant H |
| **PLALMN** | plot parameter |
| **PLALMX** | plot parameter |
| **PLAR** | axe en R 0=lin, 0 sinon log |
| **PLAZ** | axe en Z 0=lin, 0 sinon log |
| **PLREF** | axe de reference |
| **PLRMAX** | plot parameter |
| **PLRMIN** | plot parameter |
| **PLRSTP** | plot parameter |
| **PLSR** | plot parameter |
| **PLSYM** | Symetrie 0=pas de symetrie, 0 sinon symetrie centrale |
| **PLZMAX** | plot parameter |
| **PLZMIN** | plot parameter |

---

**QUALIFICATEURS A DISPOSITION:**

**/REF=<référent>**

Nom du référent (axe sur lequel sont ramenés les points) : "X" ou "PHI". S'il est précisé, il est stocké dans la variable PLREF.

**/SYM**

Relève les points et dessinne selon une symétrie centrale. Ce qualificateur est mémorisé dans la variable "plsym" par une valeur logique ($plsym = 0$ pas de symétrie, $plsym \neq 0$ symétrie centrale). Celle-ci est utilisée si aucun qualificateur (/SYM ou /NOSYM) n'est donné.

Pour le fonctionnement de cette option voir aussi /RDEF.

**/NOSYM**

Relève les points et dessinne sans tenir compte d'aucune symétrie (mêmes remarques que pour "/SYM").

**/ZLIN**

Fabrique un axe linéaire selon l'axe Z. Ce qualificateur est mémorisé dans la variable "plaz" par une valeur logique ($plaz = 0$ axe linéaire, $plaz \neq 0$ axe logarithmique). Celle-ci est utilisée si aucun qualificateur (/ZLIN ou /ZLOG) n'est donné.

**/ZLOG**

Fabrique un axe logarithmique selon l'axe Z (mêmes remarques que pour "/ZLIN").

**/RLIN**

Fabrique un axe linéaire pour le rayon. Ce qualificateur est mémorisé dans la variable "plar" par une valeur logique ($plar = 0$ axe linéaire, $plar \neq 0$ axe logarithmique). Celle-ci est utilisée si aucun qualificateur (/ZLIN ou /ZLOG) n'est donné.

**/RLOG**

Fabrique un axe logarithmique pour le rayon (mêmes remarques que pour "/RLIN").

**/RDEF**

Utilise les variables PLRMIN et PLRMAX pour définir les rayons minimum et maximum de la zone de recherche. Par défaut le rayon maximum est le plus grand rayon possible et le rayon minimum est nul pour une symétrie unique et égal a –rayon maximun pour une symétrie double (voir /SYM).

**/ZDEF**

Utilise les variables PLZMIN et PLZMAX pour fixer l'échelle selon l'axe Z.

**/FUNC**

Tracé de la fonction théorique (PSF) sur le graphe (selon PAR( :)).

**/LTYPE=<No>**

Type de ligne.

**/LIST**

Affiche une représentation des point pris en compte sur l'écran alpha. Exemple :

```
------------------------------------------------------------
                                                      <> <>
                                                   <> <> <>
                                                54 56 58 <> <>
 <>                                          49 51 53 55 58 <>
 <>  3  6  8                              44 46 48 51 53 56 <>
  2  5  7 10 13 15 18                 39 41 44 46 49 52 54 57
  2  5  8 11 14 17 19 22 25 ?? ?? ?? 37 40 42 45 48 51 54 57
  2  5  8 11 14 16 19 22 24          37 40 43 45 48 51 54 57
  1  4  7 10 ?? 15 17 19                44 47 49 52 55 58
 <>  3  5  8 10 ?? 14                               54 56 <>
 <> <>  3  5  7
 <> <> <>  2
 <> <> <>
 <> <>
------------------------------------------------------------
```

Les cases vides représentent les points rejetés selon l'angle.
Les cases "<>" représentent les points rejetés selon le rayon.
Les cases "??" représentent les points rejetés selon le rayon.
Les nombre représente le numéro de case (modulo 100) de l'histogramme depuis le rayon

minimum.
Si $PLRSTP = 0$ : Les cases "[]" représentent les points pris en compte

De plus, si la matrice est trop grande, on échantillonne pour avoir un affichage sur 80 colonnes.

Voir aussi tout les qualificateurs communs aux commandes graphiques.

---

**REMARQUES:**

- La variable PLSR permet de définir l'echelle selon R, si elle est différente de zero, d'après la formule :

$$RAYON\_MAX = \frac{1.}{PLSR} + RAYON\_MIN$$

- on considère au plus 10000 points pour $PLRSTP = 0$
- on considère au plus 150 case pour $PLRSTP \neq 0$

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| GDRIVE | I | marge pour la feuille 0=pas <0=mm >0=marge |
| GFLAG | I | tailles des fontes titre et label automatique si = 0 |
| GLBCOL | I | index couleur background pour le label (<0 == tranparent) |
| GLCOL | I | index couleur des caracteres pour le label |
| GLFONT | I | No fonte des labels (1-4) (norm,roman,italic,script) |
| GLSIZ | I | taille de la fonte des labels si GFLAG =/= 0 |
| GMARGE | I | marge pour la feuille 0=pas <0=mm >0=marge |
| GTBCOL | I | index couleur background pour le titre (<0 == tranparent) |
| GTCOL | I | index couleur des caracteres pour le titre |
| GTFONT | I | No fonte du titre (1-4) (norm,roman,italic,script) |
| GTSIZ | I | taille de la fonte des titres si GFLAG =/= 0 |
| LWX | I | largeur de la window en X |
| LWY | I | largeur de la window en Y |
| NWX | I | nb de subwindow en X |
| NWY | I | nb de subwindow en Y |
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| PAR | I | param courant H |
| PLALMN | I | plot parameter |
| PLALMX | I | plot parameter |
| PLAR | I/O | axe en R 0=lin, 0 sinon log |
| PLAZ | I/O | axe en Z 0=lin, 0 sinon log |
| PLREF | I/O | axe de reference |
| PLRMAX | I | plot parameter |
| PLRMIN | I | plot parameter |
| PLRSTP | I | plot parameter |
| PLSR | I | plot parameter |
| PLSYM | I/O | Symetrie 0=pas de symetrie, 0 sinon symetrie centrale |
| PLZMAX | I | plot parameter |
| PLZMIN | I | plot parameter |
| WDX | I | No subwindow courante en X |
| WDY | I | No subwindow courante en Y |
| XSTART | I | coord. world de depart en X de la matrice 1 |
| XSTEP | I | pas en X de la matrice 1 |
| YSTART | I | coord. world de depart en Y de la matrice 1 |
| YSTEP | I | pas en Y de la matrice 1 |

## 2.52 PARABOLIC

Fit parabolique sur les données d'une matrice

---

**SYNTAXES:**

PARABOLIC [<No matrice>] [/FIT]

PARABOLIC [<No matrice>] /GENERATION [/LIST]

---

**DESCRIPTION:**

Fit un polynôme du type

$$Z = a + bX + cY + dXY + eX^2 + fY^2$$

$avec$

$$a = ppar(1)$$
$$b = ppar(2)$$
$$c = ppar(3)$$
$$d = ppar(4)$$
$$d = ppar(5)$$
$$e = ppar(6)$$

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **PPAR** | param parabole cte |
| **XCENTR** | centre de l'image |
| **YCENTR** | centre de l'image |

---

**QUALIFICATEURS A DISPOSITION:**

**/FIT**

Fit une surface parabolique. Retourne les 6 paramètres de cette parabole dans PPAR. Et détermine le centre (derivées nulles).

**/GENERATION**

Génère une surface parabolique selon PPAR.

**/LIST**

Affiche les résultats.

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| PPAR | I/O | param parabole cte |
| XCENTR | I/O | centre de l'image |
| XSTART | I | coord. world de depart en X de la matrice 1 |
| XSTEP | I | pas en X de la matrice 1 |
| YCENTR | I/O | centre de l'image |
| YSTART | I | coord. world de depart en Y de la matrice 1 |
| YSTEP | I | pas en Y de la matrice 1 |

## 2.53 PATCH

Patch une matrice sur l'afficheur (opération par défaut) ou sur un fichier image de type BDF.

---

**SYNTAXES:**

PATCH [No de matrice]

PATCH [No de matrice] /FIT

PATCH [No de matrice] /LOCK

PATCH [No de matrice] /IMAGE

PATCH [No de matrice] /IMAGE /NEW

PATCH [No de matrice] /IMAGE /FIT

---

**DESCRIPTION:**

PATCH tient compte des coordonnées des pixels lors du transfert (coordonnées world). Ainsi, les systèmes de coordonnées doivent êtres superposables.

---

**REMARQUES:**

- Concernant l'afficheur :
  - Une seule exception : l'afficheur autorise la matrice à avoir un pas égal à un multiple du pas de l'image courante (créant ainsi un effet zoom).
  - Si la fenêtre n'existe pas encore, elle est crée à la taille de la matrice avec un système de coordonnée égal.
  - Si la fenêtre existe déjà on peut la ramener à la taille de la matrice avec /FIT
- Concernant une image :
  - Si l'image de destination n'existe pas, on la crée avec "/NEW" et dans ce cas sa taille et son système de coordonnée est donné par les variables "BDF***".
  - Si on désire que l'image ait la taille de la matrice, on crée l'image avec /FIT. Les commentaires et autres paramètres annexes sont pris dans les variables "BDF***".

---

**VARIABLES INTERACTIVES PRINCIPALES:**

Ces variables sont utilisées pour la création d'une image.

| | |
|---|---|
| **BDFIDE** | DF - ident |
| **BDFCUN** | DF - cunit |
| **BDFNPI** | DF - npix |
| **BDFSTA** | DF - start |
| **BDFSTE** | DF - step |
| **BDFCUT** | DF - cuts |

---

---

**QUALIFICATEURS A DISPOSITION:**

**/IMAGE=<image MIDAS>**

Effectue le patch sur un fichier type BDF.

**/NEW**

Utilisé avec /IMAGE, il crée une image selon les paramètres contenus dans les variable "BDF***".

**/FIT**

Utilisé avec /IMAGE, il crée une image ayant la taille de la matrice patchée, son système de coordonnée et certains paramètres contenus dans les variable "BDF***".

Utilisé sur l'afficheur, la fenêtre prend la taille et le système de coordonnée de la matrice.

**/LOCK**

Utilisé sur l'afficheur uniquement, fonctionne come "/FIT" mais conserve la taille de la fenêtre.

---

## VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE

| Variables | Mode | Description |
|---|---|---|
| BDFCUN | I | DF - cunit |
| BDFCUT | I | DF - cuts |
| BDFIDE | I | DF - ident |
| BDFNPI | I | DF - npix |
| BDFSTA | I | DF - start |
| BDFSTE | I | DF - step |
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| XSTART | I | coord. world de depart en X de la matrice 1 |
| XSTEP | I | pas en X de la matrice 1 |
| YSTART | I | coord. world de depart en Y de la matrice 1 |
| YSTEP | I | pas en Y de la matrice 1 |

## 2.54 PATH

Permet de modifier la variable DIRPRC lors de la compilation.

---

**SYNTAXE:**

PATH <nouveau chemin d'accès aux procédures>

---

**DESCRIPTION:**

En effet, une commande du type *SET DIRPRC="test"* n'a aucune action à la compilation. Cette commande remédie à ce problème.

## 2.55 POINTE

Pointe une coordonnée world sur l'image de l'afficheur.

---

**SYNTAXE:**

POINTE X_world Y_world /qualificateur

---

**PARAMETRES:**

| | |
|---|---|
| **X_world** | Coordonnées selon X donnée selon le pas et le départ de l'image sur l'afficheur. |
| **Y_world** | Coordonnées selon Y donnée selon le pas et le départ de l'image sur l'afficheur. |

---

**DESCRIPTION:**

Place le curseur à la position donnée, recentre l'image si le point est hors fenêtre.

---

**QUALIFICATEUR A DISPOSITION:**

**/IMAGE**

Centre l'image mais ne place pas le curseur. Utile lorsque l'image est plus grande que la fenêtre.

## 2.56 PRINT

Termine le dessin courant et l'envoie directement sur l'imprimante pour le driver MPOST.

---

**SYNTAXE:**

PRINT

---

**DESCRIPTION:**

Cette commande est automatique hors du mode window (1 dessin par page) ou en fin de page en mode window avec l'utilisation de la commande NEXTWINDOW.

Pour les driver d'écran (MSVIEW, MX11) cette commande n'a d'effet qu'en mode window, où un effacement de l'écran a lieu.

Si un fichier est produit, son nom se trouve dans la variable LASTF.

## 2.57 PROCEDURE

Listing des procédures à dispositions et création de documentation.

---

**SYNTAXE:**

PROCEDURE [pseudo-wildcard] [/qualif]

---

**DESCRIPTION:**

PROCEDURE affiche par défaut la première ligne des fichiers de procédure se trouvant dans les directories nommés dans la variable "DIRPRC" et commencant par un commentaire. La liste des fichiers peut être limitée aux fichiers contenant dans leur nom la pseudo-wildcard si elle est donnée (appelée pseudo car elle ne doit pas contenir de 'regular expression').

Il est possible de limiter l'accès de recherche à certain directories en placant un séparateur " :" dans la liste des directories. Dans ce cas, les directories suivant le " :" ne sont pas accédé. Exemple :

DIRPRC=". prc : call"

Le comportement de PROCEDURE est modifié par les qualificateurs suivant.

---

**QUALIFICATEURS A DISPOSITION:**

**/HEADER**

Liste l'entête du fichier, la déclaration "SUBROUTINE" et les défauts.
L'entête est le début du fichier dont les lignes commencent par un point d'exclamation.
Les défaut sont les lignes d'instruction qui suivent l'entête et qui contiennent le mot "DEFAULT".

**/FULL**

Liste le fichier complet.

**/N=<nb de lignes>**

Liste au maximum le nombre de lignes donné. Donné avec "/HEADER" il limite la longueur de l'entête.

**/ALL**

Prend en compte les fichier n'ayant pas de commentaires sur la première ligne. Donné avec "/HEADER" il n'affiche que la première ligne des fichiers sans commentaires.

**/GREP=<mot>**

N'affiche que les fichiers qui contiennent <mot> dans la zone demandée (1ère ligne, header, ...).

**/TEX**

Fabrique une sortie formattable par "latex" dans le fichier "proc.tex". Le document ainsi créé comporte un chapitre par directory. Si le fichier "INTRO.tex" existe dans le directory visité, le texte qu'il contient est mélangé avec la sortie en cours. Le nom du chapitre est fournit par défaut (comme étant le nom du directory) si le fichier "INTRO.tex" n'existe pas ou si celui-ci ne contient pas la déclaration "/chapter" sur sa première ligne.

Le fichier "proc.tex" doit donc être traité par latex avant d'être imprimé s'il ne contient pas d'erreur.

Il est important de noter que les commentaires des procédures sont écrit en latex.

---

**REMARQUE:**

Cette commande lance le programme "exe/infoprc" avec les paramètres adéquats.

## 2.58 QUIT

Sort de l'interpréteur sans sauver le bloc de données.

---

**SYNTAXE:**

QUIT

---

**REMARQUE:**

Ferme les images et les tables ouvertes.

## 2.59 READ

Lit un bloc de données standard ou un bloc de données partiel.

---

**SYNTAXE:**

READ <nom du bloc> [/qualif]

---

**DESCRIPTION:**

La lecture d'un bloc entraîne le remplacement des valeurs des variables en mémoire. Les variables courantes ne sont pas sauvées (voir SAVE pour le sauvetage).

La lecture d'un bloc standard (extension ".blk" ou ".ref") modifie les variables standard (niveau 0) quelque soit le niveau d'imbrication de la procédure.

La lecture de bloc partiel (autre extension) modifie les variables standard (niveau 0) si le nom correspond ou cree des variables au niveau d'imbrication de la procédure courant. ou au niveau 0 si le qualifiquateur "/global" est précisé.

---

**QUALIFICATEUR A DISPOSITION:**

**/GLOBAL**

Permet de lire un bloc de données non standard dans une procédure pour faire des variables globales.

**/ECHO**

Echo des variables lues dans un bloc de donnée.

---

**REMARQUES:**

Si le bloc de donnée n'est pas compatible avec la version de l'interpréteur, l'interpréteur questionne l'utilisateur pour une mise a jour du bloc (retrait des variables superflues et rajout des nouvelles variables selon les valeurs qu'elles ont dans BLOCK.REF).

S'il le bloc n'existe pas, alors aucune modification n'est effectuée.

L'extension ".blk" est mise par défaut.

## 2.60 RETURN

Retour d’une subroutine

## 2.61 SAVE

Permet le sauvetage, du bloc de données, d'une partie du bloc de données, de la forme compilée de la procédure courante (debugging) et de la liste des variables (debugging)

---

**SYNTAXES:**

SAVE

SAVE [<fichier>] [/qualif]

---

**DESCRIPTION:**

SAVE travaille par défaut avec le bloc de données. Un bloc standard est reconnu par son extension ".blk" ou ".ref".

Si le fichier a une autre extension, cela signifie que l'on veut faire un sauvetage partiel des variables. Dans ce cas, on sauve uniquement les variables décrits dans ce bloc (qui suit le standard des blocs de données). Les variables sauvées sont celles accessibles au moment de la commandes. Un message est émis pour chaque variable inexistante, mais cela ne génère pas d'erreur.

---

**QUALIFICATEURS A DISPOSITION:**

**/BLOCK**

Sauve le bloc (qualificateur par défaut)

**/REFERENCE=<bloc_name>**

Dans le cas d'un sauvetage partiel, lorsque le fichier de destination n'existe pas, <bloc_name> donne la référence des variables à sauver. Le format de ce fichier doit suivre le standard des blocs de données.

**/GLOBAL**

Sauve les variables globales dans "block.glo"

**/NEW**

le fichier de destination est crée, il ne doit pas exister.

**/OLD**

le fichier de destination doit exister.

**/UNKNOWN**

le fichier de destination est crée, s'il n'existe pas.

**/ASMB**

Sauve la forme compilée pour le debugging.

**/KW**

Sauve les variables pour le debugging.

---

---

**REMARQUES:**

- Le qualif /NEW est utilisé par défaut
- Le fichier procédure de défaut est : procedure.asm
- Les extensions par défaut sont :
  - .asm pour les procédures
  - .blk pour les blocs de données
  - .kw pour les variables

## 2.62 SEARCH

Affiche les variables du bloc dont le nom ou le commentaire contient la chaîne de caractères donnée en entrée.

---

**SYNTAXE:**

SEARCH <chaîne de caractères>

---

**QUALIFICATEUR A DISPOSITION:**

**/COMMENTAIRE**

effectue la recherche sur la partie commentaire du bloc de donnée

## 2.63 SERVER

Gère l'accès à des serveurs communiquant sous GOP

---

**SYNTAXES:**

SERVER /CONNECT[=<socket_name>] [/TEST]
SERVER /CONNECT[=<socket_name>] /COMMAND=<commande_unix>
SERVER /CONNECT /PORT=<No_port> [/HOST=<hostname>]
SERVER /CONNECT /PORT=... [/HOST=...] /COMMAND=<commande_unix> [/TEST]
SERVER /WRITE=<texte> [/CI[=<Channel_Identifier>]]
SERVER /READ [/CI[=<Channel_Identifier>]]
SERVER /DISCONNECT [/CI[=<Channel_Identifier>]]
SERVER /VGOP=<Verbosité_GOP> [/CI[=<Channel_Identifier>]]

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **SRVCI** | identificateur de canal |

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **SRVANS** | reponse du serveur |

---

**DESCRIPTION:**

SERVER permet l'accès à des serveurs communiquant sous GOP dont le protocole se limite au passage de chaînes de caractères ASCII.
Les paramètres de connections sont simplement le nom de la socket (<socket_name>) pour les connections sur la même machine ou le No du port (<No_port>) et optionellement le nom de la machine (<hostname>), prise par défaut à "localhost", pour les connections au travers d'Internet.
La commande de connection retourne un identificateur de canal (`CI`) (<Channel_Identifier>) dans la variable `SRVCI`. Ce `CI` est utilisé par la suite pour les accès au serveur ainsi connecté. S'il n'est pas précisé avec `/CI`, la variable `SRVCI` est prise par défaut.
Un erreur apparaît si le serveur n'est pas présent. Dans ce cas, et si <commande_unix> a été précisé avec `/COMMAND` cette commande est envoyée en background dans le but de lancer le serveur. SERVER rajoute l'option "`-M`" à la commande pour permettre un connection de type MASTER (voir sources). Si la connection se fait au travers d'Internet la commande doit lancer un remote shell et donc "`rsh <host>`" doit être précisé.
Lors de l'écriture, <texte> est envoyé au serveur, ce texte doit être composé en fonction de ce qu'attend le serveur dans le cas de commande ou sans aucune contrainte particuliaire dans le cas d'un serveur de type logbook.
Lors d'une lecture, le texte reçu est placé dans la variable SRVANS. Lors d'une lecture, les messages GOP d'un status autre que "`OPOK`" sont affichés et génèrent une erreur. Les messages de classe "`DBUG`" ou "`INFO`" sont affichés jusqu'à l'arrivée d'un message d'une autre classe.

---

---

**QUALIFICATEURS A DISPOSITION:**

**/CONNECT[=<socket_name>]**

Connection. Si <socket_name> est précisé la connection est faire sur Socket Unix (même machine). Retourne <Channel_Identifier> dans SRVCI.

**/DISCONNECT**

Déconnection du serveur

**/TEST**

Avec `/CONNECT` teste la connection et l'effectue si le serveur est présent. Retourne -1 dans la variable `srvci` si le serveur est absent, mais sans générer d'erreur.

**/PORT=<No_port>**

Donne le Numéro de port pour une connection Internet.

**/HOST=<hostname>]**

Donne le nom de la machine pour une connection Internet. Pris par défaut à "localhost" (machine locale).

**/COMMAND=<commande_unix>**

Commande pour le lancement facultatif du serveur.

**/CI=<Channel_Identifier>**

Précise l'identificateur de canal pour les opérations de lecture, écriture, déconnection et changement de verbosité. Pris par défaut à `srvci`.

**/WRITE=<texte>**

Envoi <texte> sur le serveur.

**/READ**

Lecture sur le serveur

**/MWRITE=<matrice>**

Envoi d'une matrice sur le serveur. Les valeurs de la matrice doivent être compris entre 1 et 16. utilisé pour envoyer des images 4 bits sur serveur GOP

**/TIMEOUT=<timeout>**

Timeout sur la conenction ou sur les opération de read

**/VGOP=<Verbosité_GOP>**

Change la verbosité de GOP (0–9)

**/SYNC=0|1**

Choix de la synchronisation des headers et des paquets (protocole GOP)

---

---

**REMARQUE:**

Quelque soit le type de serveur, ce dernier doit être capable de répondre à la commande "TEST" et renvoyer un acknowledge de maximum 8bytes. Cette commande permet de tester les connections redondantes

## 2.64 SET

Permet de modifier le contenu de variables du bloc de données courant ou d'effectuer des opérations de base sur les matrices de travail

---

**SYNTAXES:**

SET <dest>=<expression> ...

SET <dest> ... /VERIF

---

**DESCRIPTION:**

Voici plusieurs exemples d'utilisation

– pour les variables numériques
  Ex: SET START(2)=PARAM(N)
  Ex: SET STOP(N)=START(N)+STEP(N)*NX(N)
  Ex: SET A=12*(LOG(N-1))
– pour les variables de type caractère
  Ex: SET USER(1)=ARTHUR
  Ex: SET USER(3)=ARTHUR(1)
  Ex: SET USER(I+1)="WOLFGAND AMADEUS"

Il est possible de modifier plusieurs positions d'une variable dimensionnée en précisant les indices de départ et de fin. Dans ce cas on vérifie l'existence de toutes les variables mis en jeu avant les modification. Si une erreur apparaît les changements n'ont pas lieu.

– toutes les variables prennent la même valeur
  Ex: SET INDEX(4 :9)=0
– le tableau est modifie variable par variable
  Ex: SET INDEX(4 :9)=VEC(4,SIN(2),A+B,SQRT(10),8,9)
– les variables d'un tableau prennent les valeurs des variables d'un autre tableau. Attention, dans ce cas le nombre de variables doit être le même de part et d'autre du signe égal.
  Ex: SET START(1 :10)=STOP(21 :30)

SET permet également de déposer une valeur dans une case de matrice ou d'effectuer des opérations simples sur des matrices entières. L'opération d'écriture d'un pixel s'écrit :

Ex: [1](2,3)=24 (matrice 1, colonne 2, ligne 3)

Les opérations sur matrices permettent cinq opérateurs : $+$, $-$, $*$, $/$ et &. Ils opèrent entre 2 matrices ou entre une matrice et une variable. L'opérateur & remplace les points éliminés de la matrice opérande 1 par les points de la matrice opérande 2 s'ils ne sont pas déjà éliminés. Dans le fonctionnement standard seuls les points superposables des matrices mis en jeu sont soumis au calcul (coordonnées utilisateur). Avec l'utilisation du qualificateur /PIXEL, seules les coordonnées <colonne ;ligne> sont prisent en compte.

– en coordonnées worlds
  Ex: [1]=[2]
  Ex: [1]=[1]+[2]
  Ex: [1]=[2]*100

– en coordonnées pixel
  Ex: [1]=[2] /PIXEL
  Ex: [1]=[2]+[3] /PIXEL

---

**QUALIFICATEURS A DISPOSITION:**

**/VERIFY**

Le contenu de la variable a modifier est affiché et l'on peut soit la conserver tel quel en tapant <RETURN> ou la modifier.

Ex: SET /VERIFY START

**/ALL**

S'utilise uniquement avec /VERIFY pour modifier toutes les positions d'une variable dimensionnée.

Ex: SET /VERIFY /ALL START

**/WORLD**

Permet l'assignation selon les coordonnées des pixels (pour les opérations sur matrices).

---

**REMARQUES:**

Il n'est pas nécessaire d'écrire le mot SET . En effet, INTER reconnaît l'assignation de paramètres si le signe égal est présent dans le premier terme de la ligne de commande.

Ex: PROMPT> A=A+1

Souvenez-vous :

```
PROMPT > i=10 j=20
PROMPT > i=20 j=i k=i+j
PROMPT > SHOW i j k
   I(001) =  20.000000
   J(001) =  10.000000
   K(001) =  30.000000
```

## 2.65 SHOW

Affiche le contenu d'une variable ou l'évaluation une expression

---

**SYNTAXES:**

SHOW <expression> ...

SHOW <variable> ...

---

**QUALIFICATEURS A DISPOSITION:**

**/ALL**

Affiche toutes les positions d'une variable dimensionnée

Ex: SHOW ALL NX

**/NOSTOP**

En mode courant le défilement des variables s'interrompt lorsque l'écran est plein. Ce qualificateur permet un affichage en continu.

Ex: SHOW /NOSTOP VAR(20 :58)

**/COMMAND**

Affiche toutes les commandes de l'interpréteur contenue en mémoire.

Ex: SHOW /COMMAND

## 2.66 STOP

Retourne en mode interprétation depuis une procédure

---

**SYNTAXE:**

STOP

## 2.67 SUBROUTINE

Point d'entrée d'une subroutine

---

**SYNTAXE:**

SUBROUTINE <nom> [<argument>[=<type>]] ...

---

**DESCRIPTION:**

Les subroutines sont d'un usage similaire au Fortran. Leur déclaration peut se faire a tout moment mais elle ne peut être faite qu'en début de fichier ou qu'après une fin de procédure (voir ENDPROC). Elle permettent le passage de variables **non dimensionnées** de type numérique ou caractère.

Les arguments déclarés dans la subroutine sont considérés par défaut comme des variables numériques. Les arguments de type caractères devront être notés : $< argument >= C$.

## 2.68 SUM

Somme les points d'une matrice et soustrait le fond de ciel.

---

**SYNTAXE:**

SUM [<No de matrice>] [/qualificateurs]

---

**DESCRIPTION:**

La sommation est effectuée soit sur la matrice complète, soit a l'intérieur ou à l'extérieur d'un cercle ou d'un polygone. Les pixels sont pris partiellement lorsqu'ils sont traversé par le cercle.

SUM renvoie la somme des points, la surface utilisée par les points non éliminés, la surface des points éliminés et le rapport des surfaces (cercle/matrice).

Le fond de ciel est soustrait par défaut (PFCT(9)). Le qualificateur /BG empêche cette soutraction.

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **PFCT** | parametre fonction |
| **VOLR** | rayon d'integration |

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **SUMC** | somme |
| **SUMFR** | pourcent surface sommee / surface matrice |
| **SUMR** | surface non-utilisee en pixel**2 |
| **SUMU** | surface utilisee en pixel**2 |

---

**QUALIFICATEURS A DISPOSITION:**

**/RAYON**

Somme les points en ou hors du cercle de rayon VOLR. Centré en <PFCT(2) ;PFCT(3)> ou selon le qualif /CENTER.

Si le qualificateur /CENTER est précisé, il n'est pas nécessaire de donner /RAYON.

**/CENTER**

Pose le centre du cercle au milieu de la matrice .

**/CENTER=(**$X_o, Y_o$**)**

Pose le centre du cercle en $< X_o; Y_o >$ (en coordonnées utilisateur).

**/CENTER=(@X :**$Delta_x$**,@Y :**$Delta_y$**)**

Calcule les coordonnées du centre en offset pixel depuis le coin bas–gauche de la matrice. L'offset (1 ;1) donne le premier pixel bas-gauche.

**/POLY=<nb_coord>,<X_coord>,<Y_coord>**

Somme les points strictement inclus dans le polygone à <nb_coord> $-1$ sommets dont les coordonnées sont donnée dans les vecteurs <X_coord> et <Y_coord> (coordonnée world).

Attention,

<X_coord>(<nb_coord>) = <X_coord>(1)

<Y_coord>(<nb_coord>) = <Y_coord>(1)

Ex: pour un rectangle : SUM /POLY=5,"X1","Y1"

**/MASK**

Utilisé uniquement avec /POLY. Fabrique le masque du polygone dans la matrice de couche directement supérieure.

**/IN**

Somme les points à l'intérieur du cercle ou du polygone (défaut).

**/OUT**

Somme les points à l'extérieur du cercle ou du polygone.

**/BG**

Empêche la soustraction du fond de ciel.

**/LIST**

Affiche des statistiques sur le travail effectué.

---

## VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE

| Variables | Mode | Description |
|---|---|---|
| NX | I | nb de pixel en X de la matrice 1 |
| NY | I | nb de pixel en Y de la matrice 1 |
| PFCT | I | parametre fonction |
| SUMC | O | somme |
| SUMFR | O | pourcent surface sommee / surface matrice |
| SUMR | O | surface non-utilisee en pixel**2 |
| SUMU | O | surface utilisee en pixel**2 |
| VOLR | I | rayon d'integration |
| XSTART | I | coord. world de depart en X de la matrice 1 |
| XSTEP | I | pas en X de la matrice 1 |
| YSTART | I | coord. world de depart en Y de la matrice 1 |
| YSTEP | I | pas en Y de la matrice 1 |

Avec :
- somme = somme des points non-éliminés
- Surface totale = surface de la région pris en compte
- Surface matrice = $nx * ny * xstep * ystep$
- surface utile = partie non éliminée de la surface totale
- surface restante = partie éliminée de la surface totale

## 2.69 SYMBOLE

Diverses actions sur l'image de l'afficheur concernant les symboles et leurs labels.

---

**SYNTAXE:**

SYMBOLE
SYMBOLE /qualif

---

**DESCRIPTION:**

L'action par défaut est de poser un symbole, à la position du curseur, lorsque l'on clique sur un des bouttons de la souris. Le symbole peut être tracé grâce au qualificateur /POSITION.
Les symbole peuvent être des croix, des carrés vides ou pleins, des rectangles ou des cercles de taille variable. Les symboles peuvent être labellés par un texte ou un numéro.
Les paramètres du symbole se trouvent dans le bloc de données.
Dans les tous les cas la coordonnée (world) du centre du symbole est rendue dans les keywords AFFWCU et dans le premier seulement cas on retourne la coordonnées (pixel) dans AFFPCU, le contenu du pixel (AFFVAL) et le numéro du bouton (AFFRET). La taille du symbole (`AFFDIM`) peut être donnée en coordonées world ou coordonnées pixels.
Types de symboles :

| | |
|---|---|
| **1** | croix centrée (dimension= $2 * AFFDIM + 1$) |
| **2** | carré vide centré (coté= $2 * AFFDIM + 1$) |
| **3** | carré plein centré (coté= $2 * AFFDIM + 1$) |
| **4** | cercle centré (diamètre= $2 * AFFDIM + 1$) |
| **5** | rectangle centré (largeur= $2 * AFFDIM(1) + 1$, hauteur= $2 * AFFDIM(2) + 1$) |
| **6** | carré vide avec l'origine sur le coin bas-gauche (coté= $AFFDIM$) |
| **7** | rectangle avec l'origine sur le coin bas-gauche (largeur= $AFFDIM(1)$, hauteur= $AFFDIM(2)$) |

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **AFFCOL** | couleur des symboles |
| **AFFCTR** | type de centrage |
| **AFFDIM** | taille des symboles |
| **AFFPCU** | coord pixel X curs No 1 |
| **AFFRET** | code de retour lors d'un get |
| **AFFTYP** | typ symb 1:cr 2:car 3:bx 4:crcl |
| **AFFVAL** | valeur du pixel |
| **AFFWCU** | coord world X curs No 1 |
| **AFFWID** | largeur des symboles |

---

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **AFFPCU** | Voir plus haut sous "VARIABLES INTERACTIVES PRINCIPALES" |
| **AFFWCU** | Voir plus haut sous "VARIABLES INTERACTIVES PRINCIPALES" |
| **AFFVAL** | Voir plus haut sous "VARIABLES INTERACTIVES PRINCIPALES" |
| **AFFRET** | Voir plus haut sous "VARIABLES INTERACTIVES PRINCIPALES" |

---

**QUALIFICATEURS A DISPOSITION:**

**/POSITION=X_world,Y_world**

Place le symbole selon la coordonnée donnée.

**/DPIXEL**

Comprend la variable `AFFDIM` en pixel. Le défaut est `AFFDIM` en world coordonnée.

**/LABEL=<valeur numérique>**

Affiche le nombre selon le mode de centrage.

**/LABEL=<chaire de caractères>**

Affiche la chaîne de caractères selon le mode de centrage.

---

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| AFFCOL | I | couleur des symboles |
| AFFCTR | I | type de centrage |
| AFFDIM | I | taille des symboles |
| AFFPCU | I/O | coord pixel X curs No 1 |
| AFFRET | I/O | code de retour lors d'un get |
| AFFTYP | I | typ symb 1 :cr 2 :car 3 :bx 4 :crcl |
| AFFVAL | I/O | valeur du pixel |
| AFFWCU | I/O | coord world X curs No 1 |
| AFFWID | I | largeur des symboles |

## 2.70 VERBOSE

Permet l'affichage conditionnel des messages adressés à l'écran par la commandes WRITE /VERBOSE.

---

**SYNTAXE:**

VERBOSE

VERBOSE /ON

VERBOSE /OFF

---

**DESCRIPTION:**

Sans qualificateurs, VERBOSE affiche le mode courant

---

**QUALIFICATEURS A DISPOSITION:**

**/ON**

Se met en mode verbose.

**/OFF**

Quitte le mode verbose.

---

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| VERBOS | I/O | selon VERBOSE /on /off |

## 2.71 VERSION

Affiche la date à laquelle a eu lieu la dernière compilation de l'interpréteur avec un nouveau bloc de donnée

---

**SYNTAXE:**

VERSION

---

**DESCRIPTION:**

Le bloc de données contient en première position la variable AAAAAA(1). Il contient la date de la dernière modification du bloc de donnée (variables rajoutées ou enlevées). Cette variable est mise à jour par le programme INICODE. Si la date n'est pas la même alors le bloc est mis à jour.

## 2.72 VOLUME

Calcule le volume défini par l'intégrale de la fonction théorique limité par le rayon VOLR.

---

**SYNTAXE:**

VOLUME [/qualificateurs]

---

**DESCRIPTION:**

Il peut exister plusieurs fonctions permettant de calculer le volume de l'étoile. Ces fonctions peuvent avoir jusqu'a 10 paramètres. La fonction est nommée dans VFONCT(1) et les paramètres sont décrits par les variables PFCT(1) à PFCT(10), ce sont les paramètres par défaut. La fonction est choisie selon les priorités suivantes :
– la fonction fournie avec les variables du bloc de données.
– la fonction donnée sur la ligne de commande.

---

**VARIABLES INTERACTIVES PRINCIPALES:**

| | |
|---|---|
| **VFONCT** | fonction |
| **PFCT** | parametre fonction |
| **VOLM0** | magnitude zero |
| **VOLR** | rayon d'integration |

---

**VARIABLES RESULTATS PRINCIPALES:**

| | |
|---|---|
| **VOLM** | magnitude |
| **VOL** | volume sous la fonction |

---

**QUALIFICATEURS A DISPOSITION:**

**/FONCTION[=[<fonction>,][<P4> ...[,<P10>]]]**

Précise le nom de la fonction ou certains des arguments à utiliser. Met a jour les variables VFONCT et PFCT(4) à PFCT(10) si les arguments correspondants sont donnés.

**/NB=nb**

Décrit la finesse avec laquelle la fonction est approximée. Cette valeur vaut 10 par défaut.

**/ERMAX=ermax**

Décrit l'erreur maximum acceptable pour l'approximation de la fonction. Cette valeur vaut 0.0001 par défaut.

**/LIST**

Affiche un compte rendu des opérations.

---

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| PFCT | I/O | parametre fonction |
| VFONCT | I/O | fonction |
| VOLM0 | I | magnitude zero |
| VOLM | O | magnitude |
| VOLR | I | rayon d'integration |
| VOL | O | volume sous la fonction |

## 2.73 WRITE

Ecriture formattée.

---

**SYNTAXE:**

WRITE <EXP1> [<EXP2> [<EXP3> ... [<EXP20>]]] [/qualif]

---

**PARAMETRES:**

<**EXPn**> expression de type numérique ou caractère.

---

**DESCRIPTION:**

Ecrit des variables, des vecteurs ou des textes à l'écran, sur une fenêtre graphique, dans un fichier, dans une variable ou dans une fenêtre d'un moniteur.

---

**QUALIFICATEURS A DISPOSITION:**

**/FMT=<format_1>[,<format_2>[,...]]**

Ce qualificateur permet de donner le format d'écriture de chaque élément.

Si un élément n'a pas de format ou le format ne correspond pas au type de l'élément (numérique ou caractère), on prend alors un format par défaut. Celui-ci est 'A' pour les chaîne de caractères et pour les nombres, on prend le dernier format numérique. Si ce dernier n'existe pas, on prend alors le format 'G13.7'.
Les formats respectent la syntaxe fortran.

On a à disposition :

**A, An, In, In.n, Fn.m, En.m, Gn.m, Zn, Zn.m, On** et On.m

Ex: WRITE "Itération No " VAR /FMT=A,I1

Un facteur multiplicatif peut être placé devant chacun d'entre eux et il faut alors placer le format entre guillemets . Si ce facteur est donné par une expression numérique, il faut le donner entre les signes < et >.

Ex: WRITE I NX(1 :I-1) DATA(1 :2) /FMT=I2,"<I-1>I4","2F6.2"

**Attention : Les parenthèses ne sont pas acceptées dans un format.**

**/TERM**

Affiche à l'écran. C'est l'opération par défaut si aucun qualificateur concernant la sortie du texte n'est donné.

**/REVERSE**

Le texte est affiché en reverse vidéo.

**/VERBOSE**

Affiche à l'écran uniquement en mode verbose (voir VERBOSE /ON).

**/KEYWORD=<variable>**

Permet l'écriture formattée dans une variable de type caractère du bloc de donnée.

**/CHAMP=<champ>**

Ecrit dans tout les champs des moniteur nommés <champ> (voir SCREEN).

**/UNIT=<unit>**

Ecrit dans le fichier ouvert sous le numéro d'unité <unit> par la commande FILE.

**/FILE=<file>**

Ecrit dans le fichier <file> ouvert par la commande FILE.

Attention, cette option peut être plus lente que l'option /UNIT car un INQUIRE fortran est effectué chaque fois pour déterminer le numéro d'unité logique du fichier.

**/OPEN**

L'écriture dans le fichier précisé par "/UNIT" ou par "/FILE" se fait uniquement si celui-ci est ouvert. Il n'y a pas de message d'erreur si le fichier est fermé.

**/WINDOW**

Ecrit le texte dans la window courante de la sortie graphique. La gestion du saut de ligne dans la window est gérée automatiquement. Par contre il est préférable que la window ait été accédée apres la commande NEXTWINDOW ou que soit utilisé le qualificateur /TOP.

**/TOP**

Ecrit sur la première ligne du haut de la window courante.

**/POS=<X_mm>,<Y_mm>**

Ecrit à la position <X_mm,Y_mm> sur la feuille entière ou sur la window (selon le mode).

**/LOGBOOK**

Ecrit sur le log–book.

**/CODE=<code>**

Ecrit les paramètres selon le format <code> (voir base de données des messages).

**/QUOTATIONMARKS**

Entour tout les argument caratères de double guillemets

---

## VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE

| Variables | Mode | Description |
|---|---|---|

## 2.74 ZERNIKE

Génère une image selon la fonction de Zernike (anlyse de front d'onde.

---

**SYNTAXE:**

ZERNIKE [<No de matrice>] [/qualificateurs]

---

**DESCRIPTION:**

La fonction de Zernike est décrite par ses 25 coefficients contenu dans les variables `ZERCOF(1 :25)`

---

**VARIABLES INTERACTIVES PRINCIPALES:**

**ZERCOF** coefficient

---

**QUALIFICATEURS A DISPOSITION:**

**/RADIUS=<rayon>**

Donne le rayon du front d'onde à construire.

**/OBSCURATION=<raport primaire secondaire>**

Donne le diamètre de la zone centrale

---

**VARIABLES DU BLOC UTILISEES PAR CETTE COMMANDE**

| Variables | Mode | Description |
|---|---|---|
| NX | I/O | nb de pixel en X de la matrice 1 |
| NY | I/O | nb de pixel en Y de la matrice 1 |
| ZERCOF | I | coefficient |

# Chapitre 3

# PROFILS

## 3.1 PSF : point spread function

$$P_i(r) = BG + H\frac{e(-Br_i^2)}{(1+CBr_i^D)}$$

$où$

$$r_i^2 = x_i'^2 + \varepsilon^2 y_i'^2$$

$avec$

$$x_i' = (x_i - X_c)cos\varphi + (y_i - Y_c)sin\varphi$$
$$y_i' = (y_i - Y_c)cos\varphi - (x_i - X_c)sin\varphi$$

Les paramètres sont :

| | | |
|---|---|---|
| H | PFCT(1) | hauteur |
| $X_c$ | PFCT(2) | Coordonnée X centrale |
| $Y_c$ | PFCT(3) | Coordonnée Y centrale |
| $\varepsilon$ | PFCT(4) | Ellipticité |
| $\varphi$ | PFCT(5) | Inclinaison |
| B | PFCT(6) | paramètre de forme |
| C | PFCT(7) | paramètre de forme |
| D | PFCT(8) | paramètre de forme |
| BG | PFCT(9) | fond de ciel |

## 3.2 GAUSSM1

$$P_i(r) = Hr_i \frac{e^{-(min(70, Br_i^2)}}{(1 + CBr_i^D)}$$

$où$

$$r_i^2 = x_i'^2 + \varepsilon^2 y_i'^2$$

$avec$

$$x_i' = (x_i - X_c)cos\varphi + (y_i - Y_c)sin\varphi$$
$$y_i' = (y_i - Y_c)cos\varphi - (x_i - X_c)sin\varphi$$

Les paramètres sont :

| | | |
|---|---|---|
| H | PFCT(1) | hauteur |
| $X_c$ | PFCT(2) | Coordonnée X centrale |
| $Y_c$ | PFCT(3) | Coordonnée Y centrale |
| $\varepsilon$ | PFCT(4) | Ellipticité |
| $\varphi$ | PFCT(5) | Inclinaison |
| B | PFCT(6) | paramètre de forme |
| C | PFCT(7) | paramètre de forme |
| D | PFCT(8) | paramètre de forme |
| BG | PFCT(9) | fond de ciel |

# Chapitre 4

# DICTIONNAIRE DES VARIABLES

| | |
|---|---|
| **AAAAAAAAAA** | code |
| **AFFCOL** | couleur des symboles |
| **AFFCTR** | type de centrage |
| **AFFDIM** | taille des symboles |
| **AFFPCU** | coord pixel X curs No 1 |
| **AFFRET** | code de retour lors d'un get |
| **AFFTYP** | typ symb 1:cr 2:car 3:bx 4:crcl |
| **AFFVAL** | valeur du pixel |
| **AFFWCU** | coord world X curs No 1 |
| **AFFWID** | largeur des symboles |
| **ALIANG** | cte pour le passage de coord aff en coord gsc |
| **ALIF1** | cte pour le passage de coord aff en coord gsc |
| **ALIF2** | cte pour le passage de coord aff en coord gsc |
| **ALIMET** | methode utilisee pour le fit |
| **ALISIG** | cte pour le passage de coord aff en coord gsc |
| **ALITR** | cte pour le passage de coord aff en coord gsc |
| **ALIX0** | cte pour le passage de coord aff en coord gsc |
| **ALIY0** | cte pour le passage de coord aff en coord gsc |
| **ALPHA** | Position de l'objet clique (degres decimaux) |
| **AREFRA** | coefficient de refraction (A) |
| **BDFCUN** | DF - cunit |
| **BDFCUT** | DF - cuts |
| **BDFIDE** | DF - ident |
| **BDFNPI** | DF - npix |
| **BDFSTA** | DF - start |
| **BDFSTE** | DF - step |
| **BG** | valeur initiale du background |
| **BGCLIP** | clip. fact.: sigma*BGCLIP |
| **BGCONV** | clip. converg: \|diffBG/sigma\|<BGCONV |
| **BGFLAG** | flag du background |
| **BGFR** | fraction centrale |
| **BGMED** | BG=BGMED*median+(1-BGMED)*mean |
| **BGSIGM** | sigma of last background |
| **BLKCRT** | bloc de donnees courant |
| **BLKTIM** | date du sauvetage de ce block |
| **BREFRA** | coefficient de refraction (B) |
| **CALPHA** | ALPHA sous forme caractere |
| **CDELTA** | DELTA sous forme caractere |
| **CHI** | Sigma du fit |
| **CHRONO** | de chronometre interne en mu sec |
| **CI** | variable a disposition |
| **CJ** | variable a disposition |
| **CK** | variable a disposition |
| **CMDAFF** | options pour l'afficheur (aff) |

| | |
|---|---|
| **CMDGSC** | options pour gsc |
| **DAY** | jour (TCL) |
| **DEBUG** | selon DEBUG /on /off |
| **DELTA** | Position de l'objet clique (degres decimaux) |
| **DETSCX** | taille d'un pixel en X |
| **DETSCY** | taille d'un pixel en Y |
| **DIRHLP** | 2eme directory pour help |
| **DIRPRC** | directory pour procedure |
| **DRET** | distance au centre |
| **DTIME** | CPU temps en sec dernier programme |
| **DTUNIX** | delta tunix depuis le lancement de inter |
| **ELFACT** | facteur pour CHI ou val ABS |
| **ELIM** | type d'elimination |
| **ELMAX** | seuil superieur |
| **ELMIN** | seuil inferieur |
| **EOF** | end of file |
| **ERREUR** | erreur |
| **ERRNAM** | nom pour le fichier d'erreur |
| **ERRSTAT** | status (==1) si erreur /on |
| **FDATE** | date formattée (TCL) |
| **FLUX** | Flux total sur la matrice image |
| **FONCT** | nom de la fct |
| **FPESTAT** | status (==1) si erreur /fpeon |
| **FPE_ERREUR** | floating point execption |
| **FUSEAU** | heures de decalage par rapport au TU (+ouest) |
| **GAIN** | Gain photoelectrons/ADU |
| **GANG** | angle en deg de la ligne de base (normal=0) pour le texte |
| **GBCOL** | index couleur background pour le texte (<0 == tranparent) |
| **GCOL** | index couleur des caracteres pour le texte |
| **GDRIVE** | marge pour la feuille 0=pas <0=mm >0=marge |
| **GFLAG** | tailles des fontes titre et label automatique si = 0 |
| **GFONT** | No fonte courante (1-4) (norm,roman,italic,script) |
| **GJUS** | justification (g->d)=(0 - 0.5 - 1) |
| **GLBCOL** | index couleur background pour le label (<0 == tranparent) |
| **GLCOL** | index couleur des caracteres pour le label |
| **GLFONT** | No fonte des labels (1-4) (norm,roman,italic,script) |
| **GLSIZ** | taille de la fonte des labels si GFLAG =/= 0 |
| **GMARGE** | marge pour la feuille 0=pas <0=mm >0=marge |
| **GPHI** | 3D - angle dans le plan X-Y |
| **GSCA0** | te pour la translation de <x;y> en <a;d> |
| **GSCAX** | te pour la translation de <x;y> en <a;d> |
| **GSCAY** | te pour la translation de <x;y> en <a;d> |
| **GSCBX** | te pour la translation de <x;y> en <a;d> |
| **GSCBY** | te pour la translation de <x;y> en <a;d> |
| **GSCD0** | te pour la translation de <x;y> en <a;d> |
| **GSCX** | te pour la translation de <x;y> en <a;d> |
| **GSCX0** | te pour la translation de <x;y> en <a;d> |
| **GSCY** | te pour la translation de <x;y> en <a;d> |
| **GSCY0** | te pour la translation de <x;y> en <a;d> |
| **GSIZ** | hauteur de la font courante |
| **GTBCOL** | index couleur background pour le titre (<0 == tranparent) |
| **GTCOL** | index couleur des caracteres pour le titre |
| **GTFONT** | No fonte du titre (1-4) (norm,roman,italic,script) |
| **GTHETA** | 3D - Angle vertical |
| **GTSIZ** | taille de la fonte des titres si GFLAG =/= 0 |
| **GTYP** | type de graphique (1-7) |
| **HIN** | pic central initial |
| **HOUR** | heure (TCL) |
| **I** | variable |
| **J** | variable |
| **K** | variable |
| **LASTF** | nom du dernier fichier cree |

| | |
|---|---|
| **LATITU** | latitude du lieu d'observation |
| **LONGIT** | longitude du lieu d'observation (+est) |
| **LWX** | largeur de la window en X |
| **LWY** | largeur de la window en Y |
| **MAGNI** | retour magnitude objet pointe |
| **MATCOU** | Nb de couches pour la matrice(i) |
| **MATNCL** | nombre de colonne sur l'ecran |
| **MATNDG** | nombre de digits |
| **MATNLI** | nombre de lignes |
| **MATSIZ** | Nb de pixel pour la matrice(i) |
| **MDCOMM** | commentaire |
| **MDDATE** | date d'acquisition |
| **MINUTE** | minute (TCL) |
| **MONTH** | mois (TCL) |
| **NALGO** | choix d'un algorithme |
| **NFIELD** | No de champ |
| **NIMAX** | nombre d'iterations utilise |
| **NM** | Numero de la matrice courante |
| **NOISE** | bruit(PHOTON-CONST) |
| **NPLAT** | No de cliche |
| **NPT** | Nb de valeurs retournees |
| **NREP** | |
| **NSRES** | nb de segment (nb de SRES valable) |
| **NSTEP** | Nb d'iteration |
| **NTFA** | Nb de coeff utile dans TFA(i) |
| **NUSED** | nombre de pixels utilises |
| **NWX** | nb de subwindow en X |
| **NWY** | nb de subwindow en Y |
| **NX** | nb de pixel en X de la matrice 1 |
| **NY** | nb de pixel en Y de la matrice 1 |
| **PAR** | param courant H |
| **PARER** | erreur maximale pour H |
| **PARI** | param initial courant H |
| **PARIT** | nombre d'iterations pour H |
| **PARST** | pas d'iterations pour H |
| **PFCT** | parametre fonction |
| **PLALMN** | plot parameter |
| **PLALMX** | plot parameter |
| **PLAR** | axe en R 0=lin, 0 sinon log |
| **PLAZ** | axe en Z 0=lin, 0 sinon log |
| **PLDELT** | pas inter niveaux |
| **PLNNIV** | nb de niveaux |
| **PLREF** | axe de reference |
| **PLRMAX** | plot parameter |
| **PLRMIN** | plot parameter |
| **PLRSTP** | plot parameter |
| **PLSR** | plot parameter |
| **PLSYM** | Symetrie 0=pas de symetrie, 0 sinon symetrie centrale |
| **PLZMAX** | plot parameter |
| **PLZMIN** | plot parameter |
| **POWER** | Puissance |
| **PPAR** | param parabole cte |
| **PROMPT** | prompt |
| **RANDOM** | initialisation du generateur |
| **RELIM** | rayon d'elimination pour ELIM="EL" |
| **RESIDU** | Residu –> DELTA |
| **RETMAX** | taille de XRET,YRET,ZRET |
| **RMS** | Read-out noise in ADU |
| **SECOND** | seconde (TCL) |
| **SEUILG** | seuil de calcul de la fonctio |
| **SHAMEM** | indicateur de shared memory. Si diff de zero on alloue en shared memory |
| **SIGMA** | Ecart type |

| | |
|---|---|
| **SRES** | residu=f(segment) –> WI*DELTA**2 |
| **SRVANS** | reponse du serveur |
| **SRVCI** | identificateur de canal |
| **STATUS** | status pour usage general |
| **SUMC** | somme |
| **SUMFR** | pourcent surface sommee / surface matrice |
| **SUMR** | surface non-utilisee en pixel**2 |
| **SUMU** | surface utilisee en pixel**2 |
| **TCL** | temps civil local (heures decimales) |
| **TFA** | Coeff de la TF angulaire |
| **TFAI** | Index Coeff de la TF angulaire |
| **TILT** | Tilt des images dss |
| **TUNIX** | temps Unix nb de seconde depuis 1.1.1970 0h UTC |
| **TUNIXUS** | microsecondes de TUNIX |
| **USE_GRAPH** | si existe, alors initalisation graph |
| **UTC** | Coordinated Universal Time (heures decimales) |
| **UTC_DAY** | jour (UTC) |
| **UTC_FDATE** | date formattée (UTC) |
| **UTC_HOUR** | heure (UTC) |
| **UTC_MINUTE** | minute (UTC) |
| **UTC_MONTH** | mois (UTC) |
| **UTC_SECOND** | seconde (UTC) |
| **UTC_YEAR** | annee (UTC) |
| **VERBOS** | selon VERBOSE /on /off |
| **VFONCT** | fonction |
| **VOL** | volume sous la fonction |
| **VOLM** | magnitude |
| **VOLM0** | magnitude zero |
| **VOLR** | rayon d'integration |
| **WDCV** | ECONV Coeff pour GAUSS |
| **WDX** | No subwindow courante en X |
| **WDY** | No subwindow courante en Y |
| **XCENTR** | centre de l'image |
| **XCORIG** | coord w. de dep en X de l'image d'origine 1 |
| **XIN** | x-init pic 1 |
| **XREF** | X reference |
| **XRET** | coordonees X en retour |
| **XSIZE** | taille de l'espace graphique en mm |
| **XSSIZE** | taille de l'espace graphique désiré en mm (0==defaut) |
| **XSTART** | coord. world de depart en X de la matrice 1 |
| **XSTEP** | pas en X de la matrice 1 |
| **YCENTR** | centre de l'image |
| **YCORIG** | coord w. de dep en Y de l'image d'origine 1 |
| **YEAR** | annee (TCL) |
| **YIN** | y-init peak 1 |
| **YREF** | Y reference |
| **YRET** | coordonees Y en retour |
| **YSIZE** | taille de l'espace graphique en mm |
| **YSSIZE** | taille de l'espace graphique désiré en mm (0==defaut) |
| **YSTART** | coord. world de depart en Y de la matrice 1 |
| **YSTEP** | pas en Y de la matrice 1 |
| **ZERCOF** | coefficient |
| **ZRET** | densite en retour |

# Chapitre 5

# REFERENCES CROISEES

| Variables | lues par | mises à jour par |
|---|---|---|
| AAAAAAAAAA | main | — |
| ALIMET | align | align |
| BDFCUN | patch | — |
| BDFIDE | patch | — |
| BLKCRT | main | — |
| BLKTIM | main | — |
| CALPHA | gsc | gsc |
| CDELTA | gsc | gsc |
| CI | — | — |
| CJ | — | — |
| CK | — | — |
| CMDAFF | aff | — |
| CMDGSC | gsc | — |
| DIRHLP | main | — |
| DIRPRC | main | — |
| ELIM | elimination | — |
| ERRNAM | main | — |
| FDATE | — | date |
| FONCT | elimination fres generation | elimination generation |
| GDRIVE | contour oned print | — |
| LASTF | main | — |
| MDCOMM | extract | extract |
| MDDATE | extract | extract |
| NOISE | elimination | — |
| PLREF | oned | oned |
| PROMPT | main | — |
| SRVANS | server | server |
| TUNIX | — | date |
| USE_GRAPH | — | — |
| UTC_FDATE | — | date |
| VFONCT | volume | volume |
| AFFCOL | symbole | — |
| AFFCTR | symbole | — |
| AFFDIM | symbole | — |
| AFFPCU | symbole | get symbole |
| AFFRET | symbole | get symbole |
| AFFTYP | symbole | — |
| AFFVAL | symbole | get symbole |
| AFFWCU | symbole | get symbole |
| AFFWID | symbole | — |
| ALIANG | — | align |
| ALIF1 | align | align |
| ALIF2 | align | align |
| ALISIG | align | align |
| ALITR | — | align |
| ALIX0 | — | align |
| ALIY0 | — | align |
| ALPHA | contour gsc | gsc |
| AREFRA | main | — |
| BDFCUT | patch | — |
| BDFNPI | patch | — |
| BDFSTA | patch | — |
| BDFSTE | patch | — |
| BGCLIP | mom | — |
| BGCONV | mom | — |
| BGFLAG | mom | mom |
| BGFR | mom | mom |
| BGMED | mom | — |
| BGSIGM | mom | mom |
| BG | mom | mom |

| **Variables** | **lues par** | **mises à jour par** |
|---|---|---|
| BREFRA | main | — |
| CHI | elimination | fit1 |
| CHRONO | main | — |
| DAY | — | date |
| DEBUG | debug | debug |
| DELTA | contour gsc | gsc |
| DETSCX | contour gsc | gsc |
| DETSCY | contour gsc | gsc |
| DRET | gsc | gsc |
| DTIME | — | fit1 |
| DTUNIX | — | date |
| ELFACT | elimination | — |
| ELMAX | elimination | — |
| ELMIN | elimination | — |
| EOF | main | — |
| ERREUR | erreur | erreur |
| ERRSTAT | erreur | erreur |
| FLUX | — | deconv |
| FPESTAT | erreur | erreur |
| FPE_ERREUR | erreur | erreur |
| FUSEAU | main | — |
| GAIN | elimination fit1 fres noise | — |
| GANG | print | — |
| GBCOL | print | — |
| GCOL | print | — |
| GFLAG | contour oned | — |
| GFONT | print | — |
| GJUS | print | — |
| GLBCOL | contour oned | — |
| GLCOL | contour oned | — |
| GLFONT | contour oned | — |
| GLSIZ | contour oned | — |
| GMARGE | contour oned print | — |
| GPHI | contour | — |
| GSCA0 | gsc | gsc |
| GSCAX | gsc | gsc |
| GSCAY | gsc | gsc |
| GSCBX | gsc | gsc |
| GSCBY | gsc | gsc |
| GSCD0 | gsc | gsc |
| GSCX0 | gsc | gsc |
| GSCX | gsc | gsc |
| GSCY0 | gsc | gsc |
| GSCY | gsc | gsc |
| GSIZ | print | — |
| GTBCOL | contour oned | — |
| GTCOL | contour oned | — |
| GTFONT | contour oned | — |
| GTHETA | contour | — |
| GTSIZ | contour oned | — |
| GTYP | contour | — |
| HIN | — | mom |
| HOUR | — | date |
| I | main | — |
| J | main | — |
| K | main | — |
| LATITU | main | — |
| LONGIT | main | — |
| LWX | contour nextwindow oned | — |
| LWY | contour nextwindow oned | — |

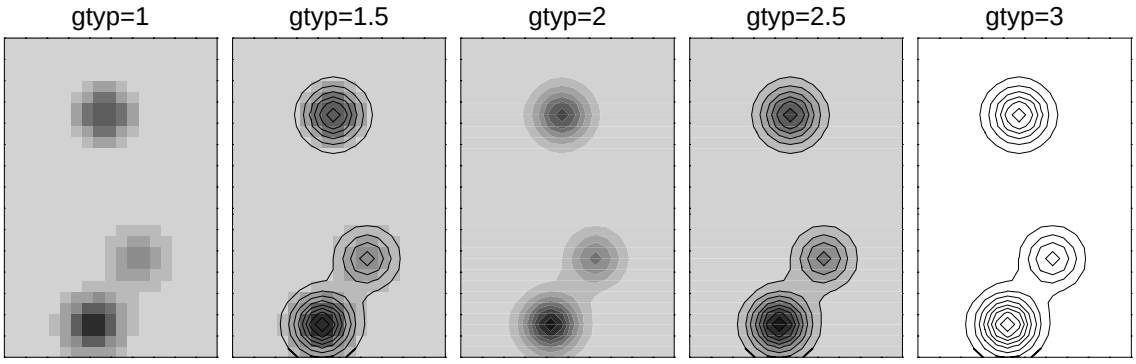
| **Variables** | **lues par** | **mises à jour par** |
|---|---|---|
| MAGNI | gsc | gsc |
| MATCOU | matrix | — |
| MATNCL | matrix | — |
| MATNDG | matrix | — |
| MATNLI | matrix | — |
| MATSIZ | matrix | — |
| MINUTE | — | date |
| MONTH | — | date |
| NALGO | deconv | — |
| NFIELD | extract | extract |
| NIMAX | — | fit1 |
| NM | mom | — |
| NPLAT | extract | extract |
| NPT | contour gsc | contour gsc |
| NREP | extract | extract |
| NSRES | — | fres |
| NSTEP | deconv | — |
| NTFA | — | fres |
| NUSED | — | fit1 |
| NWX | contour nextwindow oned | — |
| NWY | contour nextwindow oned | — |
| NX | contour deconv elimination extract fimage fit1 fres generation gsc matrix mom noise oned parabolic patch server sum zernike | extract fimage gsc matrix zernike |
| NY | contour deconv elimination extract fimage fit1 fres generation gsc matrix mom noise oned parabolic patch server sum zernike | extract fimage gsc matrix zernike |
| PARER | fit1 | — |
| PARIT | fit1 | — |
| PARI | fit1 | — |
| PARST | fit1 | — |
| PAR | fit1 oned | fit1 |
| PFCT | elimination fres generation sum volume | elimination generation volume |
| PLALMN | oned | — |
| PLALMX | oned | — |
| PLAR | oned | oned |
| PLAZ | oned | oned |
| PLDELT | contour | contour |
| PLNNIV | contour | — |
| PLRMAX | oned | — |
| PLRMIN | oned | — |
| PLRSTP | oned | — |
| PLSR | oned | — |
| PLSYM | oned | oned |
| PLZMAX | contour oned | contour |
| PLZMIN | contour oned | contour |
| POWER | deconv | — |
| PPAR | parabolic | parabolic |
| RANDOM | noise | — |
| RELIM | elimination | — |
| RESIDU | — | fres |
| RETMAX | contour | — |
| RMS | elimination fit1 fres noise | — |
| SECOND | — | date |
| SEUILG | generation | — |
| SHAMEM | main | — |
| SIGMA | deconv | — |
| SRES | — | fres |
| SRVCI | server | server |
| STATUS | — | gsc |

| Variables | lues par | mises à jour par |
|---|---|---|
| SUMC | — | sum |
| SUMFR | — | sum |
| SUMR | — | sum |
| SUMU | — | sum |
| TCL | — | date |
| TFAI | — | fres |
| TFA | — | fres |
| TILT | gsc | gsc |
| TUNIXUS | — | date |
| UTC_DAY | — | date |
| UTC_HOUR | — | date |
| UTC_MINUTE | — | date |
| UTC_MONTH | — | date |
| UTC_SECOND | — | date |
| UTC_YEAR | — | date |
| UTC | — | date |
| VERBOS | verbose | verbose |
| VOLM0 | volume | — |
| VOLM | — | volume |
| VOLR | fres sum volume | — |
| VOL | — | volume |
| WDCV | deconv | — |
| WDX | contour nextwindow oned | contour nextwindow |
| WDY | contour nextwindow oned | contour nextwindow |
| XCENTR | parabolic | parabolic |
| XCORIG | extract fimage | extract fimage |
| XIN | — | mom |
| XREF | extract | extract |
| XRET | contour gsc | contour gsc |
| XSIZE | main | — |
| XSSIZE | clear | — |
| XSTART | contour elimination extract fimage fit1 fres generation matrix mom oned parabolic patch sum | extract fimage |
| XSTEP | contour elimination extract fimage fit1 fres generation matrix mom oned parabolic patch sum | extract fimage |
| YCENTR | parabolic | parabolic |
| YCORIG | extract fimage | extract fimage |
| YEAR | — | date |
| YIN | — | mom |
| YREF | extract | extract |
| YRET | contour gsc | contour gsc |
| YSIZE | main | — |
| YSSIZE | clear | — |
| YSTART | contour elimination extract fimage fit1 fres generation matrix mom oned parabolic patch sum | extract fimage |
| YSTEP | contour elimination extract fimage fit1 fres generation matrix mom oned parabolic patch sum | extract fimage |
| ZERCOF | zernike | — |
| ZRET | contour gsc | contour gsc |

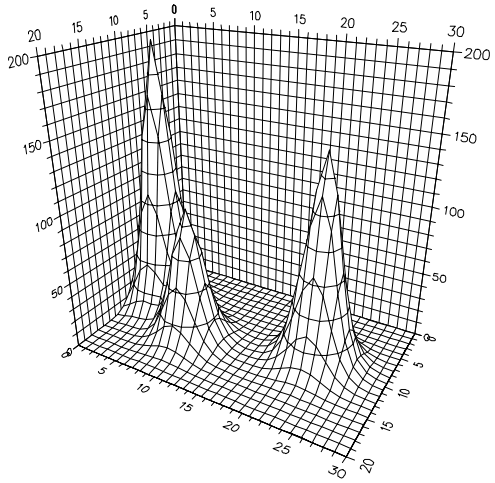
# Chapitre 6

# Annexe Graphique

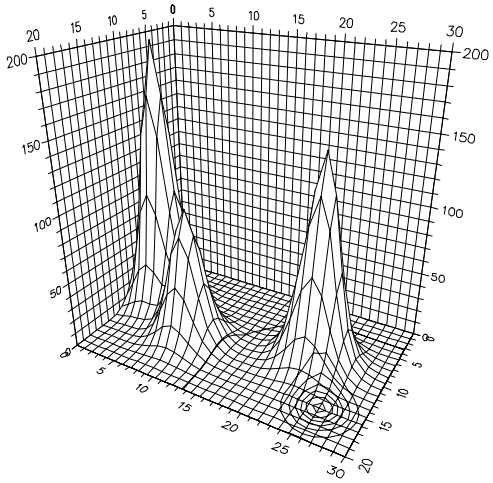
## 6.1 Type de graphiques à disposition (gtyp)



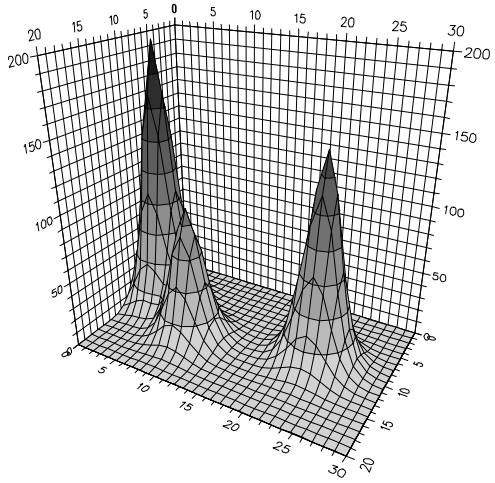

gtyp=4



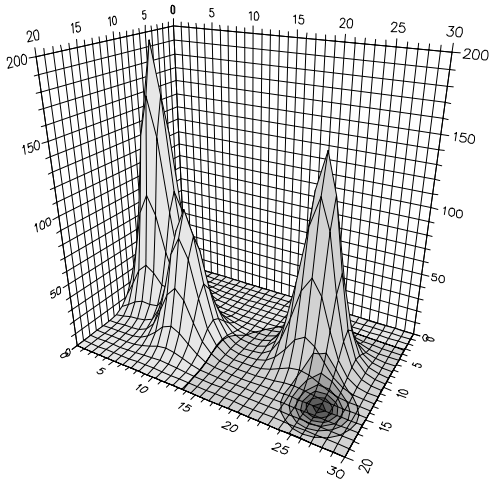
gtyp=5



gtyp=6

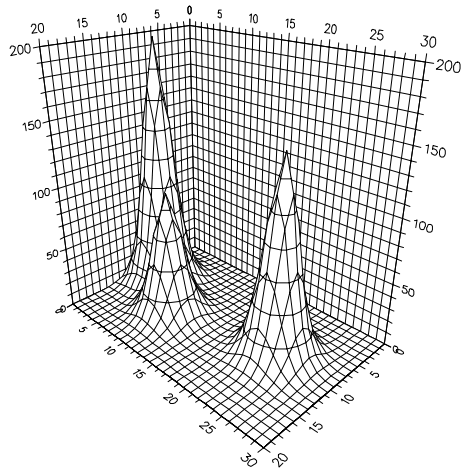


gtyp=7

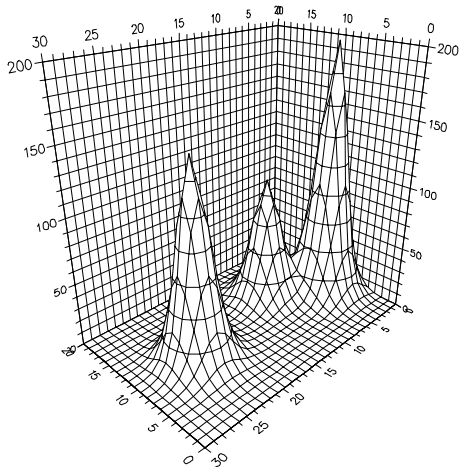


## 6.2 Orientation des dessins 3D (gphi)

gphi=45

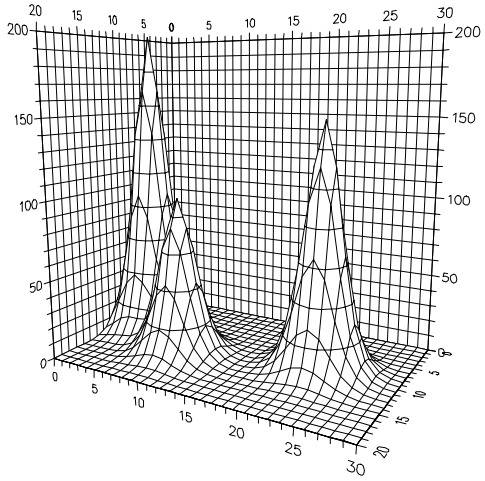


gphi=135

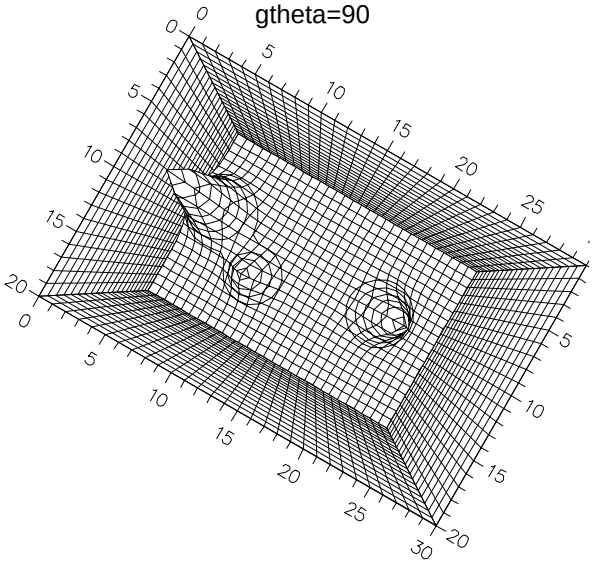


## 6.3 Elévation des dessins 3D (gtheta)

gtheta=10

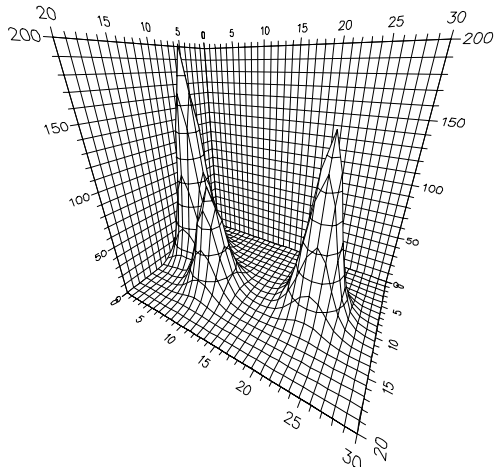


gtheta=90

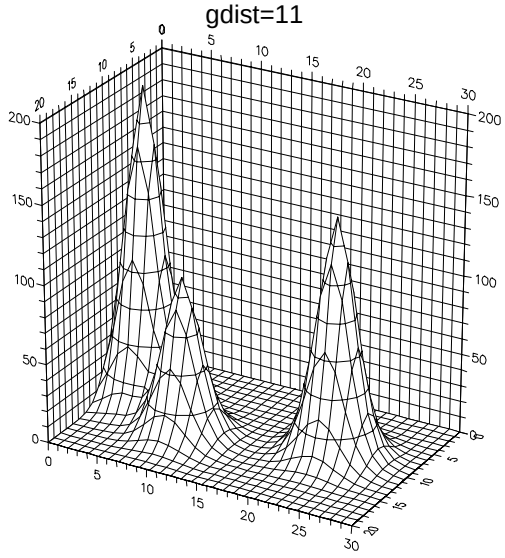


## 6.4 Distance des dessins 3D (gdist)

gdist=1

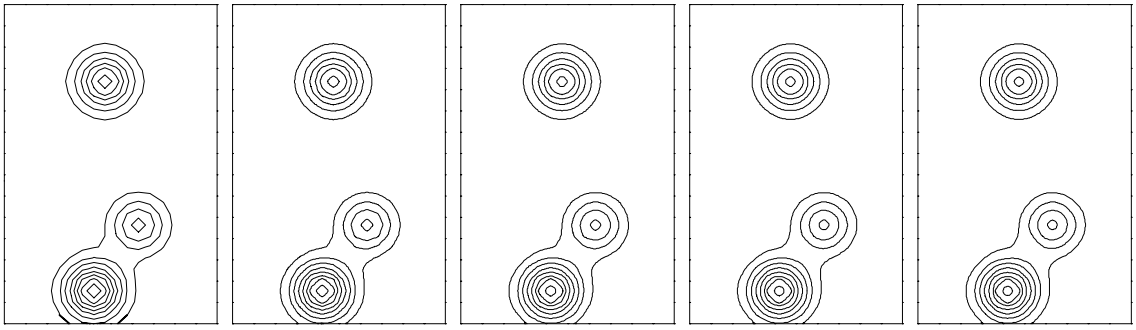


gdist=11



## 6.5 Smoothing

/smooth=1 /smooth=3 /smooth=5 /smooth=7 /smooth=9



## 6.6 Gestion de l'épaisseur des lettres (gbold, gtbold, glbold)

Paramètres globaux:

```
gang   =   0.00      gexp   =   1.00
gskew  =  90.00
ghjus  =      0      gvjus  =      1
gspmod =      0      gspfac =   0.00
gfcol  =      0      gocol  =      1
gfont  = "comp"
```

GBOLD: 0.01

gsiz: 2[mm] abcdef ABCDEF 1234567890
gsiz: 3[mm] abcdef ABCDEF 1234567890
gsiz: 4[mm] abcdef ABCDEF 1234567890
gsiz: 5[mm] abcdef ABCDEF 1234567890
gsiz: 6[mm] abcdef ABCDEF 123456
gsiz: 7[mm] abcdef ABCDEF 1
gsiz: 8[mm] abcdef ABCDE

GBOLD: 0.05

gsiz: 2[mm] abcdef ABCDEF 1234567890
gsiz: 3[mm] abcdef ABCDEF 1234567890
gsiz: 4[mm] abcdef ABCDEF 1234567890
gsiz: 5[mm] abcdef ABCDEF 1234567890

GBOLD: 0.10

gsiz: 2[mm] abcdef ABCDEF 1234567890
gsiz: 3[mm] abcdef ABCDEF 1234567890
gsiz: 4[mm] abcdef ABCDEF 1234567890
gsiz: 5[mm] abcdef ABCDEF 1234567890

## 6.7 Gestion des modes d'écriture (gspmod & gspfac)

Paramètres globaux:

```
gang  =   0.00     gexp  =   1.00
gbold =   0.00     gskew =  90.00
ghjus =      0     gvjus =      1
gfcol =      1     gocol =      1
gfont = "comp"
```

gspmod= 0

"AVAWAIavai" gspfac= 0.00
"AVAWAIavai" gspfac= 0.50
"AVAWAIavai" gspfac= 1.00
"AVAWAIavai" gspfac=-0.20

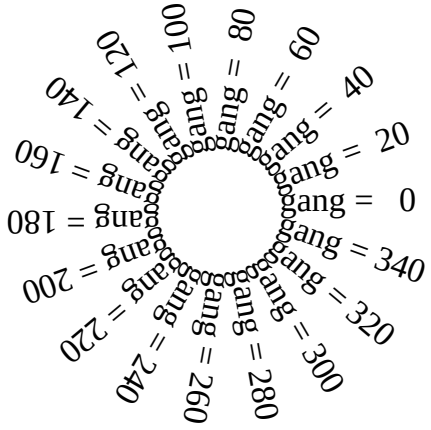
gspmod= 1

"AVAWAIavai" gspfac= 0.00
"AVAWAIavai" gspfac= 0.50
"AVAWAIavai" gspfac=
"AVAWAIavai" gspfac=-0.20

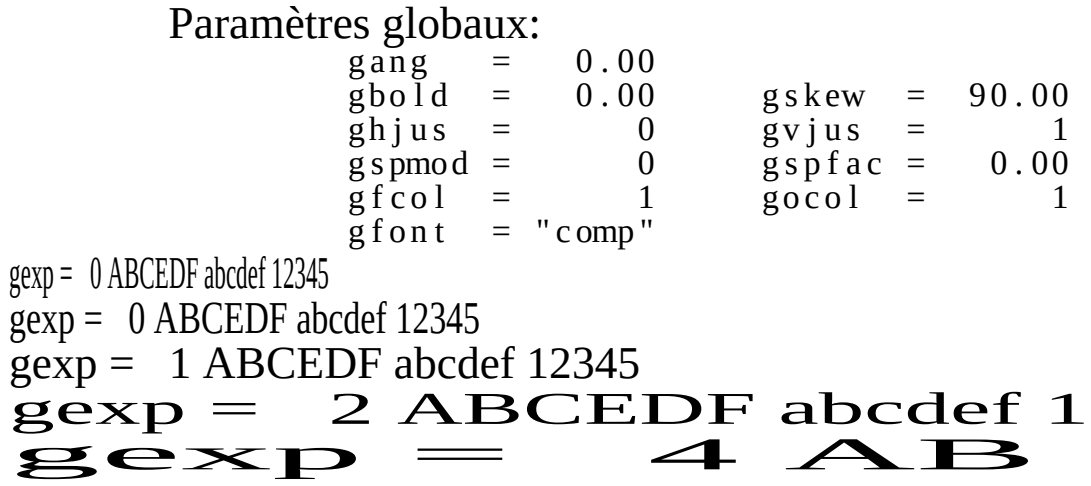
## 6.8 Gestion de l'inclinaison du texte (gang)

Paramètres globaux:

```
gexp   =   1.00
gskew  =  90.00
ghjus  =      0
gvjus  =      1
gfcol  =      1
gocol  =      1
gspmod =      0
gspfac =   0.00
gfont  = "comp"
```



## 6.9 Gestion de l'élongation des caractères (gexp, gtexp, glexp)



## 6.10 Gestion de l'inclinaison des caractères (gskew, gtskew, glskew)

Paramètres globaux:

```
gang   =   0.00      gexp   =   1.00
gbold  =   0.00
ghjus  =      0      gvjus  =      1
gspmod =      0      gspfac =   0.00
gfcol  =      1      gocol  =      1
gfont  = "comp"
```

ABCEDF abcdef 12345  gskew = 90

*ABCEDF abcdef 12345  gskew = 80*

*ABCEDF abcdef 12345  gskew = 70*

*ABCEDF abcdef 12345  gskew = 60*

*ABCEDF abcdef 12345  gskew = 50*

*ABCEDF abcdef 12345  gskew = 40*

*ABCEDF abcdef 12345  gskew = 30*

*ABCEDF abcdef 12345  gskew = 20*

*ABCEDF abcdef 12345  gskew = 10*

ABCEDF abcdef 12345  gskew = 0

ABCEDF abcdef 12345  gskew = -10

ABCEDF abcdef 12345  gskew = -20

ABCEDF abcdef 12345  gskew = -30

ABCEDF abcdef 12345  gskew = -40

ABCEDF abcdef 12345  gskew = -50

ABCEDF abcdef 12345  gskew = -60

ABCEDF abcdef 12345  gskew = -70

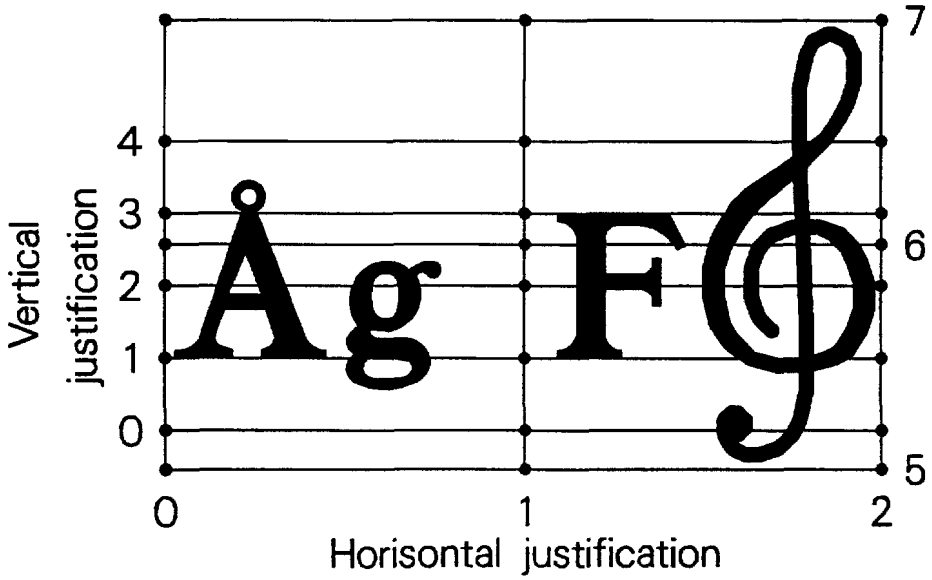
ABCEDF abcdef 12345  gskew = -80

ABCEDF abcdef 12345  gskew = -90

## 6.11 Choix de jeux de caractères (gfont, gtfont, glfont)

SIMP= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê
COMP= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê
*ITAL= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê*
**DUPL= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê**
**TRIP= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê**
CART= ABCDEFGHIJKLMNOPQRSTUVWXYZ 1234567890
SMAL= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê
SCRI= abcdefghijklmNOP2RSTUVWXYZ 1234567890
SANS= abcdefghijklmNOP2RSTUVWXYZ 1234567890
***TRII= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê***
*SMAI= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê*
HΣEE= αβγδεζηϑικλμνΞΟΠΡΣΤΥΦΧΨΩ 12345678
TINH= αβγδεζηϑικλμνΞΟΠΡΣΤΥΦΧΨΩ 1234567890 á
ΓΑΣΗ= ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ 123456789
TNAH= αβγδεζηϑικλμνΞΟΠΡΣΤΥΦΧΨΩ 12345
ЭУРИ= абэдйфгжичклмНОПШРСТЮВЩХУЗ 12345
**GOTG= abcdefghijklmNOPQRSTUVWXYZ 12345678**
**GOTE= abcdefghijklmNOPQRSTUVWXYZ 12345678**
GOUI= abcdefghijklmNOPQRSTUVWXYZ 12345678
VSIM= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê
**PSIM= abcdefghijklmNOPQRSTUVWXYZ 1234567890** áéèê
FUTU= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê
CENB= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê
**SWIB= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê**
SWIL= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê
SWIM= abcdefghijklmNOPQRSTUVWXYZ 1234567890 áéèê

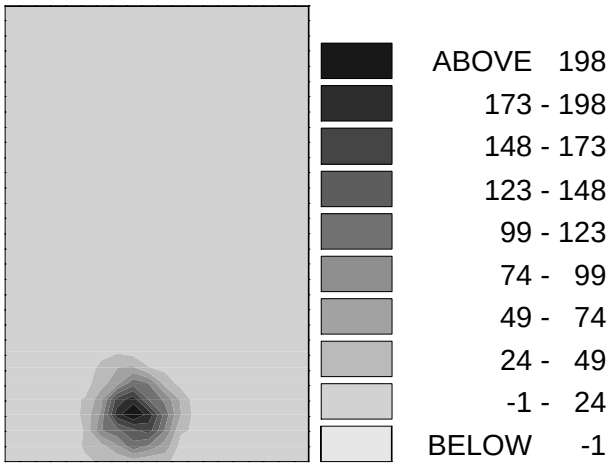
## 6.12 Gestion du centrage (ghjus & gvjus)




**Text justification.**

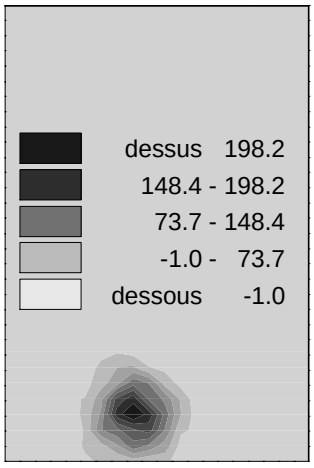
## 6.13 Affichage d'une legende type standard

Remarque : nx=20 ny=30 xstep=1 ystep=1




```
/legende=

type   :std
```

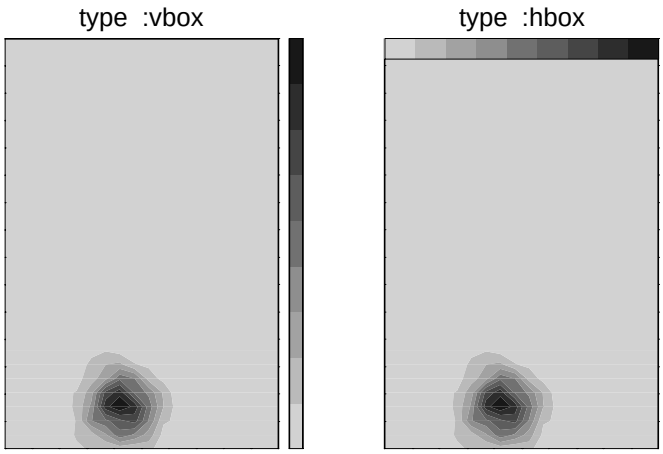



```
/legende=

xpos   :    1.0
ypos   :   10.0
type   :std
height:    4.0
nth    :      3
above  :dessus
below  :dessous
dec    :      1
```

## 6.14 Affichage d'une legende type hbox ou vbox

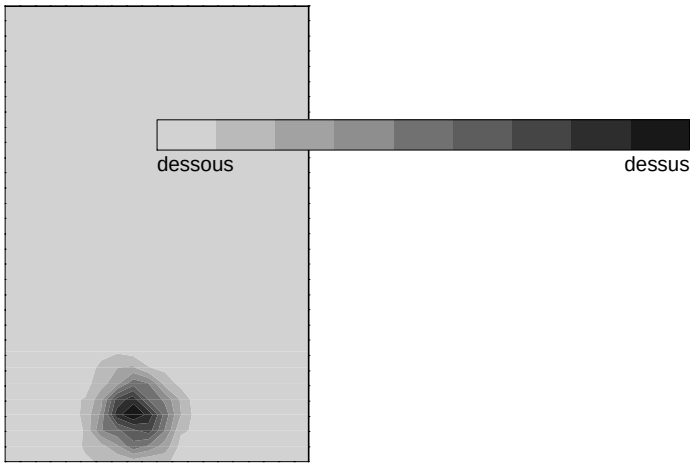
Remarque : nx=20 ny=30 xstep=1 ystep=1




```
Position des
légendes par
défaut pour
type:
       - vbox
       - hbox
```


```
/legende=

xpos   :    2.0
ypos   :   10.0
type   :vbox
height:   18.0
width  :    5.0
above  :dessus
below  :dessous
```




```
/legende=

xpos   :   10.0
ypos   :   22.5
type   :hbox
height:    2.0
width  :   35.0
above  :dessus
below  :dessous
```

# Chapitre 7

# Annexe MIDAS

## 7.1 Liste de erreurs midas

```
ERR_NORMAL   0   successful return status
ERR_DSCNPR   1   descriptor not present
ERR_SIMFIL   2   too many files opened simultaneously
ERR_LOGFIL   3   problems with logfile
ERR_OPSSYS   4   Operating system error
ERR_DSKFUL   5   disk full
ERR_FRMNAC   6   frame not accessible
ERR_INPINV   7   input invalid
ERR_CATOVF   8   overflow in general catalog
ERR_DSCBAD   9   descriptor bad
ERR_KEYBAD  10   keyword bad
ERR_KEYOVL  11   overflow in keyword data area
ERR_KEYOVN  12   overflow in keyword name table
ERR_FILNAM  13   invalid syntax for file name
ERR_FILBAD  14   file handling error
ERR_CATBAD  15   bad catalog
ERR_DSCOVF  16   descriptor data overflow ( > 32767)
ERR_INSBAD  17   bad Midas installation
ERR_FMTBAD  18   bad (not matching) binary data format
ERR_VERNOR  19   unrecognized Midas file version
ERR_TBLFUL  20   Table errors start at 20
ERR_TBLMEM  21    error allocating dynamic mem.
ERR_TBLMAP  22    error mapping the table
ERR_TBLCOV  23    column overflow
ERR_TBLENT  24    table not found wrong init.
ERR_TBLCOL  25    wrong column number
ERR_TBLROW  26    wrong row number
ERR_TBLKEY  27    identifier not found
```

```
ERR_TBLFMT  28    error in column format
ERR_TBLIMP  29    not implemented
ERR_TBLRFM  30    error reformatting table
ERR_TBLDNM  31    duplicate table name
ERR_TBLABL  32    illegal label
ERR_TBLINI  33    error in table init
ERR_TABVER  34    old VMS Midas table format

ERR_CBYTES   -1   bytes per element are not exact
ERR_ALRDON   -2   task already done - nothing to do
ERR_NODATA   -3   no data available
ERR_KEYTYP   -4   wrong type of keyword
ERR_CATENT   -5   no entry in catalog
ERR_MESOVF   -6   overflow in error message stack
ERR_BADLCK   -7   keyword is locked by another process
ERR_KEYNPR   -8   keyword not present
```

## 7.2 Définition des formats

```
D_I1_FORMAT     1L  I1 = 1 byte
D_I2_FORMAT     2L  I2 = 16 bit integer
D_UI2_FORMAT  102L  I2 = 16 bit unsigned integer
D_I4_FORMAT     4L  I4 = 32 bit integer
D_R4_FORMAT    10L  R4 = 32 bit floating point
D_R8_FORMAT    18L  R8 = 64 bit floating point
D_L1_FORMAT    21L  L1 = 1 byte logical
D_L2_FORMAT    22L  L2 = 16 bit logical
D_L4_FORMAT    24L  L4 = 32 bit logical
D_C_FORMAT     30L  1 byte character
D_X_FORMAT     40L  1 byte flags
D_P_FORMAT     50L  pointers
```

# Chapitre 8

# QUICKREF

| Fonctions | Descriptions |
|---|---|
| * | multiplication. |
| ** | exponentiation. |
| + | addition. |
| − | soustraction. |
| / | division. |
| .AND. | ET logique. |
| .EQ. | égalité numérique. |
| .GE. | plus grand ou égal. |
| .GT. | strictement plus grand. |
| .LE. | plus petit ou égal. |
| .LT. | strictement plus petit. |
| .NE. | non égalité numérique. |
| .NOT. | négation de la condition. |
| .OR. | OU logique. |
| ABS(x) | retourne la valeur absolue de *x*. |
| ACOS(x) | retourne l'arc cosinus de *x* en radian. |
| ACOSD(x) | retourne l'arc cosinus de *x* en degré. |
| ADD(variable) | retourne l'adresse de la *variable* numérique dans les tableaux d'Inter. |
| AHDE2AZ(angle_horaire, delta) | calcul l'azimut. |
| AHDE2EL(angle_horaire, delta) | calcul l'élévation. |
| AND(a,b) | retourne le résultat du 'and' binaire |
| ANGLE("angle") | retourne la valeur numérique d'un angle donné sous forme de chaîne de caractères. |
| ANUT(alpha,delta) | nute la coordonnée donnée en alpha. |
| APPNAME() | retourne le nom de l'application. |
| APREC(alpha,delta) | précessionne la coordonnée donnée en alpha. |
| ASIN(x) | retourne l'arc sinus de *x* en radian. |
| ASIND(x) | retourne l'arc sinus de *x* en degré. |
| ASSIGN(str[,prefix]) | assignation de variables selon une chaine formatée contenant une suite variable, contenu. |
| ATAN(x) | retourne l'arc tangente de *x* en radian. |
| ATAN2(y,x) | retourne l'arc tangente selon *x,y* en radian. |
| ATAN2D(y,x) | retourne l'arc tangente selon *x,y* en degré. |
| ATAND(x) | retourne l'arc tangente de *x* en degré. |
| ATOR(str) | retourne le nombre écrit dans *str*. |
| AZEL2AH(azimut, élévation) | calcul angle horaire. |
| AZEL2DE(azimut, élévation) | calcul delta. |
| CAL(x,y) | convertion de coordonnées cartésiennes en coordonnées équatorialles. |
| CDE(x,y) | convertion de coordonnées cartésiennes en coordonnées équatorialles. |
| CHAR(code) | retourne le caractère ASCII donné par *code*. |
| CLEAN([mat],seuil,size,valrep) | patch une matrice centree de cote=2*size+1 contenant valrep pour chaque poinr plus eleve que seuil |
| CLEARSV() | Efface les flags d'erreur et de status ainsi que les code et message d'erreur ainsi que le texte de la commande courante dans le bloc de communication. |
| COMPAC(str) | retourne *str* sans espace. |
| CONNECT() | Se (re)connecte sur les sémaphores du serveur courant s'ils ont été tués. |
| COS(alpha) | retourne le cosinus de *alpha* donné en radian. |
| COSD(alpha) | retourne le cosinus de *alpha* donné en degré. |
| DCDIV(x,y) | Division. Même remarque que pour DCPLUS. |
| DCMINUS(x,y) | Soustraction. Même remarque que pour DCPLUS. |
| DCMUL(x,y) | Multiplication. Même remarque que pour DCPLUS. |
| DCPLUS(x,y) | Addition. Les arguments sont caractères, le résultat est caractère. |
| DDIV(x,y) | Division. Même remarque que pour DPLUS. |
| DDTOD(angle) | formatte *angle* en : "SDDDd MMm SSs ". |
| DDTOD2(angle) | formatte *angle* en : "SDDD:MM:SS ". |
| DEG2RAD(x) | conversion degrés radians. |
| DIM(variable) | retourne la dimension d'une *variable*. |
| DMINUS(x,y) | Soustraction. Même remarque que pour DPLUS. |
| DMUL(x,y) | Multiplication. Même remarque que pour DPLUS. |
| DNUT(alpha,delta) | nute la coordonnée donnée en delta |
| DPLUS(x,y) | Addition. Les arguments sont numériques ou caractères, le résultat est numérique en simple précision. |

| Fonctions | Descriptions |
|---|---|
| DPREC(alpha,delta) | précessionne la coordonnée donnée en delta. |
| ENVDEF(evar) | retourne 1 si la variable d'environnement est definie, 0 sinon. |
| EXIST(fichier) | retourne 1 si le *fichier* existe. |
| EXP(x) | retourne l'exponentiel de *x*. |
| EXPAND(val\|vec, mul [,size]) | expand chaque position d'un vecteur ou d'une matrice. |
| FETCH() | assigne les variable Inter selon les couples donnés dans le bloc de communication. |
| FIND(str1,str2) | retourne 1 si *str2* existe dans *str1*. |
| FITGAU(vec) | Fit une gaussienne sur les données de *vec*. |
| FORMAT(val\|vec\|str, fmt) | retourne *val*\|*vec*\|*str* formatté le format *fmt*. |
| FZ(dist_zénitale) | calcul de la masse d'air (voir slalib). |
| GENF(xvec, polyvec) | fabrique f(xvec) avec polyvec comme coefficient du polynôme. |
| GENGAU(vec, nb_points) | génère une gaussienne selon les 4 premières valeurs du vecteur *vec*. (1=max, 2=centre, 3=forme, 4=background |
| GETENV(evar) | retourne le contenu de la variable d'environnement *evar*. |
| GETLU() | retourne une unité logique inutilisée. |
| GROUP() | retourne le numéro de groupe. |
| HDTOH(heure) | formatte *heure* en : "HHh MMm SS.Ss ". |
| HDTOH2(heure) | formatte *heure* en : "HH:MM:SS.S ". |
| HTOHD("heure") | retourne la valeur numérique d'une heure donnée sous forme de chaîne de caractères. |
| ICHAR(str) | retourne le code ASCII du premier caractère de la chaîne *str*. |
| INDEX(str1,str2) | retourne la position de *str2* dans *str1*. |
| INILOG(host) | Initialise une connection sur le logbook de la machine *host*. |
| INMAXV(vec) | retourne l'index du premier plus grand élément de *vec*. |
| INMINV(vec) | retourne l'index du premier plus petit élément de *vec*. |
| INT(x) | retourne la valeur entière de *x*. |
| ISNUM(xxx) | retourne 1 si "xxx" est un nombre (lexicalement parlant). |
| ISVNUM(xxx) | retourne 1 si "xxx" est une variable numérique. |
| ITOA(val) | retourne le nombre *val* formatté en entier sans blanc. |
| JD([j][,[m][,[a]]]) | retourne le jour julien formatté (1x,f11.3,1x). |
| JD0IN([j][,[m][,[a]]]) | initialise le jour julien de référence (jd0) pour les calculs de précéssion et de nutation. |
| LCAT(str1, ..., strN) | retourne la concaténation des tous les arguments (type caratère) . |
| LCEQ(str1,str2) | retourne 1 si $str1 = str2$. Cette fonction est insensible aux minuscules et aux majuscules |
| LEN(str) | retourne le nombre de caractères dans *str*. |
| LEQ(str1,str2) | retourne 1 si $str1 = str2$. |
| LGE(str1,str2) | retourne 1 si $str1 \geq str2$. |
| LGT(str1,str2) | retourne 1 si $str1 > str2$. |
| LLE(str1,str2) | retourne 1 si $str1 \leq str2$. |
| LLT(str1,str2) | retourne 1 si $str1 < str2$. |
| LNE(str1,str2) | retourne 1 si $str1 \neq str2$. |
| LNO(str) | retourne 1 si *str* vaut "n", "no" ou "non". |
| LOG(x) | retourne le logarithme naturel de *x*. |
| LOG10(x) | retourne le logarithme base 10 de *x*. |
| LOWER(str) | retourne *str* mis en minuscule. |
| LSCAT(str1,str2,...strN) | retourne la concaténation des tous les arguments (type caratère) en placant un espace entre chaque argument. |
| LSFIT(No_de_matrice) | résoud m équations à n inconnues. |
| LTRIM(str) | retourne *str* sans les espaces à gauche du texte utile. |
| LYES(str) | retourne 1 si *str* vaut "y", "yes", "o" ou "oui" . |
| MAX(x1,x2,...xn) | retourne le maximum parmis les argument *x1...xn*. |
| MAXSIZ(No_mat) | retourne la taille actuelle ou maximum (dans le cas d'une matrice en mémoire partagée) de la matrice No_mat. |
| MAXV(vec) | retourne le plus grand élément de *vec*. |
| MEDDEV(vec) | retourne le sigma median (nag :g07daf) |
| MEDIAN(vec) | retourne le median (nag :g07daf) |
| MIN(x1,x2,...xn) | retourne le minimum parmis les argument *x1...xn*. |
| MINV(vec) | retourne le plus petit élément de *vec*. |
| MOD(x,y) | retourne le modulo de *x* par *y*. |
| NBCOU(No_mat) | retourne le nombre de couche pour la matrice *No_mat*. |
| NBMAT() | retourne le nombre de matrices accessibles. |
| NBPIX() | retourne le nombre de pixels alloués par l'ensemble des matrices en mémoire partagée. |

| Fonctions | Descriptions |
|---|---|
| NINT(x) | retourne la plus proche valeur entière de *x*. |
| NOCOU(str\|num) | retourne le No de couche d'un No de matrice. |
| NOMAT(str\|num) | retourne le No de matrice d'un No de matrice. |
| NOT(a) | retourne le résultat du 'not' binaire |
| NUTIN([j][,[m][,[a]]]) | initialise les variables internes pour le calcul de la nutation. |
| OR(a,b) | retourne le résultat du 'or' binaire |
| PAD(str) | retourne *str* sans les espaces en début et fin de chaîne et avec une seule barre de soulignement ("_") pour chaque suites d'espaces entres les mots. |
| PARSE(str,No) | retourne le champ *No* de *str*, avec pour séparateur le premier caractère de *str*. |
| POLYF(xvec, yvec, degre [,pvec]) | fit polynômial à une inconnue de degré *degré*. |
| PRECIN([j][,[m][,[a]]]) | initialise les variables internes pour le calcul de la précession. |
| RAD2DEG(x) | conversion radians degrés. |
| RAND(flag) | retourne un nombre aléatoire. |
| REFCO(tdk, pmb, rh, [wl], [eps], [tlr], [hm], [phi]) | calcul des parametres AREFRA et AREFRA du modèle de réfraction atmospherique. |
| REFRAC(dist_zénitale) | calcul de la réfraction atmosphérique pour La Silla. |
| RFITS(file, key) | Lit un keyword *key* du fichier *file*. |
| RM ?(string) | Cette fonction supprime les points d'interrogation en fin de ligne. (utile pour les commandes @@ et @@@ ...). |
| ROTHPOS(ah,delta) | angle du derotateur. Remarque : utilise longit et latitu. |
| ROTHVEL(ah,delta) | vitesse du derotateur [deg/s]. Remarque : utilise longit et latitu. |
| ROTPOS(alpha,delta) | angle du derotateur moment courant. Remarque : utilise longit et latitu. |
| ROTVEL(alpha,delta) | vitesse du derotateur moment courant [deg/s]. Remarque : utilise longit et latitu. |
| RTRIM(str) | retourne *str* sans les espaces à droite du texte utile. |
| SELECT(server) | Séléctionne le serveur sur lequel vont travailler toutes les fonctions de communication. |
| SEMAGET(No) | retourne la valeur du sémaphore *No*. |
| SEMASET(No, val) | Met le sémaphore *No* à *val*. |
| SETTENV("evar=val") | assigne une variable d'environnement *evar*. |
| SETV(start,stop[,step]) | remplis un vecteur selon les paramètres de boucle. |
| SHMACK() | indique au serveur d'exécuter la commande placée dans "COMMAND" et lui signale de ne pas rendre la main. |
| SHMADD(key, content) | ajoute un paramètre référencié par *key* et son contenu *content*, dans le bloc de communication . |
| SHMCLI([name]) | retourne 1 si Inter a été lancé en mode client. Si name est précisé, alors retourne 1 si Inter est client de *name* |
| SHMCMD(cmd [,to]) | Suspend le client, envoie une commande et libère le client. |
| SHMCMDW(cmd [,to_wait [,to_cmd]]) | Suspend le client, envoie une commande et attend la fin de la commande pour libérer le client. |
| SHMCONT() | indique au serveur d'exécuter la commande placée dans "COMMAND" et lui signale de rendre la main. |
| SHMFREE() | Rend la main (après un SHMWACK()). |
| SHMGACK() | retourne la valeur du flag ackno. |
| SHMGCOD() | retourne le code d'erreur (ascii). |
| SHMGERR() | retourne le code d'erreur (numérique). |
| SHMGET(key) | lit un paramètre référencié par *key* dans le bloc de communication. |
| SHMGID() | retourne l'identificateur du bloc de mémoire partagée. |
| SHMGMES() | retourne le texte du message d'erreur. |
| SHMGSRV() | retourne le nom du serveur courant. |
| SHMGSTA() | retourne le status d'erreur. |
| SHMINIT() | initialise le bloc de communication en le vidant. |
| SHMNCNT() | retourne le nombre de client en attente sur le serveur courant. |
| SHMPCOD() | met le code d'erreur, l'erreur et le message d'erreur en shared memory. |
| SHMPUT(key, content) | écrit un paramètre référencié par *key* et son contenu *content* dans le bloc de communication et initialise le bloc de communication. |
| SHMSHOW() | visualise le bloc de communication à l'écran. |
| SHMSRV() | retourne 1 si Inter a été lancé en mode serveur. |
| SHMWACK([timeout]) | attend que le serveur ait finis (après un SHMACK()). |
| SHMWAIT([timeout]) | suspend le client jusqu'à que le serveur soit prêt. |
| SHMZCNT() | retourne le nombre de process en attente de zero sur sem No. |
| SHMZERO() | met le semaphore zéro à 1 (reset). |
| SHOWSEL() | affiche le nom de tous les serveurs possible ainsi que le serveur actuellement séléctionné. |

| Fonctions | Descriptions |
|---|---|
| SIGNAL(signal) | Envoie un signal au serveur. |
| SIN(alpha) | retourne le sinus de *alpha* donné en radian. |
| SIND(alpha) | retourne le sinus de *alpha* donné en degré. |
| SLEEP(sec) | Suspend le process durant un temps donné en seconde |
| SPARSE(str,No) | retourne le champ *No* de *str*, avec pour séparateur l'espace. |
| SQRT(x) | retourne la racine carrée de *x*. |
| SRVEXIS() | retourne 1 si le serveur existe. |
| SRVWAIT() | indique si le client a la main sur le serveur courant. |
| SRVWORK() | indique si le serveur est en train de travailler. |
| STDDEV(vec) | retourne le sigma (nag :g07daf) |
| STRIP(str) | supprime les blancs, les 0 non significatifs et le point non significatif sur une chaine numérique. |
| SUM(vec) | retourne la somme des valeur contenus dans *vec*. |
| SYSTEM(cmd) | retourne le résultat fournis par la commande system *cmd*. |
| TAN(alpha) | retourne la tangente de *alpha* donné en radian. |
| TAND(alpha) | retourne la tangente de *alpha* donné en degré. |
| TEMP(flag) | retourne certaines températures du CCD. |
| TIME() | donne l'heure de la machine (heures décimales). |
| TPARSE(str,No) | retourne le champ *No* de *str*, avec pour séparateur le tabulateur. |
| TRIM(str) | retourne *str* sans les espaces avant et apres le texte utile. |
| TS([tcl][,[j][,[m][,[a]]]]) | donne l'heure sidérale. |
| UPPER(str) | retourne *str* mis en majuscule. |
| USER() | retourne le No d'utilisateur. |
| VEC(arg1,arg2,...argn) | fabrique un vecteur en joignant tous les arguments. |
| VERS() | retourne la date de la dernière compilation d'Inter. |
| XBARY([mat]) | retourne le x barycentre |
| XOR(a,b) | retourne le résultat du 'xor' binaire |
| YBARY() | retourne le y barycentre (il faut utilise xbary(mat) avant) |
| $(string) | retourne le contenu de la variable nommée dans string. |