Index des commandes jps

par Jean-Paul Vignault
Groupe des Utilisateurs de Linux Poitevins (GULP)
(**jpv@melusine.eu.org**)
15 Janvier 2006

A

ABcercle

$A\ B\ C$ **ABcercle** $cerc \longrightarrow cerc$ est le cercle passant par les points A, B et C

ABCercle

$I\ A\ B$ **ABCercle** $- \longrightarrow$ trace l'arc du cercle $\mathcal{C}$ inscrit dans l'angle $\widehat{AIB}$ où $\mathcal{C}$ désigne le cercle de centre I passant par A

ABCercle*

$I\ A\ B$ **ABCercle*** $- \longrightarrow$ version étoilée de **ABCercle**

ABCercle_

$I\ A\ B$ **ABCercle_** $- \longrightarrow$ version underscore de **ABCercle**

ABpoint

$\alpha\ A\ B$ **ABpoint** $A' \longrightarrow$ le point A' est l'image du point B par l'homothétie de centre A, de rapport α. Autrement dit $\overrightarrow{AA'} = \alpha\overrightarrow{AB}$

abs

a **abs** $c \longrightarrow c = |a|$

addc

$z\ z'$ **addc** $Z \longrightarrow Z = z + z'$ est la somme des complexes z et z'

add

$a\ b$ **add** $c \longrightarrow c = a + c$

add

$nombre_1\ nombre_2$ **add** $nombre_3 \longrightarrow$ additionne $nombre_1$ et $nombre_2$

addm

$A\ B$ **addm** $M \longrightarrow$ additionne les matrices A et B et dépose le résultat sur la pile

addv3d

$\vec{u}\ \vec{v}$ **addv3d** $\vec{w} \longrightarrow \vec{w} = \vec{u} + \vec{v}$

addv

$u\ u'$ **addv** $\vec{U} \longrightarrow \vec{U} = \vec{u} + \vec{u}'$ est la somme des vecteurs $\vec{u}$ et $\vec{u}'$

angleA

- *angleA* : certaines connexions vous permettent de spécifier avec quel angle vous souhaitez la connexion au nœud. **valeur par défaut :** 0

angleB

- *angleB* : certaines connexions vous permettent de spécifier avec quel angle vous souhaitez la connexion au nœud. **valeur par défaut :** 0

angledroit

$A\ B\ C$ **angledroit** $- \longrightarrow$ dessine un angle droit en B

angle

$A\ B$ **angle** $\alpha \longrightarrow \alpha$ est l'angle en dégré défini par le vecteur $\overrightarrow{AB}$ dans le repère **orthonormé** jps.

Anp

$n\ p$ **Anp** $a \longrightarrow a = A_n^p = n \times (n-1) \times \cdots \times (n-p+1)$

apply

$[\ a_0 \ldots a_n\ ]\ f$ **apply** $[\ b_0 \ldots b_n\ ]$ ou $- \longrightarrow$ construit un nouveau tableau en répétant l'opération suivante : déposer l'élément a_i puis exécuter f, pour i variant de 0 à n. Si à la fin de cette opération le tableau est vide, alors il est enlevé de la pile.

apply

$[\ a_0 \ldots a_n\ ]\ f$ **apply** $[\ b_0 \ldots b_n\ ]$ ou $- \longrightarrow$ construit un nouveau tableau en répétant, pour i variant de 0 à n, l'opération suivante : déposer l'élément a_i puis exécuter f. Si à la fin de cette opération le tableau est vide, alors il est enlevé de la pile.

Apply

[$a_0 \dots a_n$] f n_0 n_1 **Apply** [$b_0 \dots b_n$] ou $- \longrightarrow$ L'éxécutable f prend n_1 arguments et on décale de n_0 à chaque itération. Ainsi le premier paramètre de la 2ème itération est x_{n_0+1}. Par exemple, les commandes **3 3 Apply** et **capply** sont équivalentes.

apply

[$a_0 \dots a_n$] *string* **apply** [$b_0 \dots b_n$] ou $- \longrightarrow$ Comme la précédente, mais l'exécutable est cette fois désigné par une chaîne de caractères.

arcangleA

- *arcangleA* : ce paramètre ne sert qu'avec les commandes **ncarc** et **pcarc**. **valeur par défaut :** 10

arcangleB

- *arcangleB* : ce paramètre ne sert qu'avec les commandes **ncarc** et **pcarc**. **valeur par défaut :** 10

arccos

a **arccos** $c \longrightarrow c = \mathrm{Arccos}\, a$ (en degrés)

Arccos

a **Arccos** $c \longrightarrow c = \mathrm{Arccos}\, a$ (en radians)

Arc

$I\ A\ \alpha$ **Arc** $- \longrightarrow$ trace au point A l'arc de cercle d'angle 2α centré en A

arc

$x\ y\ r\ ang_1\ ang_2$ **arc** $- \longrightarrow$ ajoute un arc dans le sens contraire des aiguilles d'une montre

arcn

$x\ y\ r\ ang_1\ ang_2$ **arcn** $- \longrightarrow$ ajoute un arc dans le sens des aiguilles d'une montre

arcnp

$I\ A\ B$ **arcnp** $- \longrightarrow$ trace entre les points A et B l'arc du cercle de centre I et de rayon IA (sens inverse du sens trigonométrique)

arcp

$I\ A\ B$ **arcp** $- \longrightarrow$ trace entre les points A et B l'arc du cercle de centre I et de rayon IA (sens trigonométrique)

arcsin

a **arcsin** $c \longrightarrow c = \mathrm{Arcsin}\, a$ (en degrés)

Arcsin

a **Arcsin** $c \longrightarrow c = \mathrm{Arcsin}\, a$ (en radians)

arctan

a **arctan** $c \longrightarrow c = \mathrm{Arctan}\, a$ (en degrés)

Arctan

a **Arctan** $c \longrightarrow c = \mathrm{Arctan}\, a$ (en radians)

arct

$x_1\ y_1\ x_2\ y_2\ r$ **arct** $r \longrightarrow$ ajoute un arc tangent

arcto

$x_1\ y_1\ x_2\ y_2\ r$ **arcto** $xt_1\ yt_1\ xt_2\ yt_2 \longrightarrow$ ajoute un arc tangent

argcosh

a **argcosh** $c \longrightarrow c = \mathrm{Arg\,ch}\, a$

arg

u **arg** $\theta \longrightarrow \theta \in]-180, 180]$ est l'angle que fait le vecteur $\vec{u}$ avec le vecteur unitaire de l'axe des abscisses

arg

z **arg** $\theta \longrightarrow \theta = \mathrm{Arg}(z) \in]-180, 180]$

argsinh

a **argsinh** $c \longrightarrow c = \mathrm{Arg\,sh}\, a$

argtanh

a **argtanh** $c \longrightarrow c = \mathrm{Arg\,th}\, a$

armA

- *armA* : certaines connexions commencent avec un bras de longueur *armA* (en picas). **valeur par défaut :** 10

armB

- *armB* : certaines connexions commencent avec un bras de longueur *armB* (en picas). **valeur par défaut :** 10

arrowangle

- *arrowangle* : angle en degrés de la rotation que doit subir la flèche avant d'être tracée par la commande **`arrow`**. Ce paramètre permet d'ajuster l'angulation lorsque les calculs automatiques laissent à désirer. **valeur par défaut :** 0

arrowscale

- *arrowscale* : exécutable donnant les facteurs d'échelles horizontale et verticale pour la tracé d'une flèche par **`arrow`**. **valeur par défaut :** { 1 1 }

`axeB`

zmin zmax ℓ **`axeB`** – $\longrightarrow$ [*zmin*; *zmax*] = étendue du pointille, ℓ = longueur du vecteur

`axeOxarrow`

– **`axeOxarrow`** – $\longrightarrow$ trace la flèche au bout de l'axe *Ox*

`axeOyarrow`

– **`axeOyarrow`** – $\longrightarrow$ trace la flèche au bout de l'axe *Oy*

`axeR`

xmin xmax ℓ **`axeR`** – $\longrightarrow$ [*xmin*; *xmax*] = étendue du pointille, ℓ = longueur du vecteur

`axeRVB`

min max ℓ **`axeRVB`** – $\longrightarrow$ [*min*; *max*] = étendue des pointillés, ℓ = longueur des vecteurs

`axesarrow`

– **`axesarrow`** – $\longrightarrow$ trace les flèches au bout des axes *Ox* et *Oy*

`axesymcercle`

cerc D **`axesymcercle`** $cerc'$ $\longrightarrow$ le cercle $cerc'$ est la symétrique du cercle *cerc* par rapport à la droite *D*

`axesymdroite`

d D **`axesymdroite`** d' $\longrightarrow$ la droite d' est la symétrique de la droite *d* par rapport à la droite *D*

`axesymell`

ell D **`axesymell`** ell' $\longrightarrow$ l'ellipse ell' est la symétrique de l'ellipse *ell* par rapport à la droite *D*

`axesympoint`

A D **`axesympoint`** A' $\longrightarrow$ le point A' est le symétrique du point *A* par rapport à la droite *D*

`axesympol`

pol D **`axesympol`** pol' $\longrightarrow$ le polygône pol' est la symétrique du polygône *pol* par rapport à la droite *D*

`axeV`

ymin ymax ℓ **`axeV`** – $\longrightarrow$ [*ymin*; *ymax*] = étendue du pointille, ℓ = longueur du vecteur

B

`baseeuler`

$a\ \{f\}\ x_0\ y_0\ h$ **`baseeuler`** $x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ dépose les points, calculés par la méthode d'Euler, de la courbe sur $[x_0, a]$ de la fonction *s* solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Plus précisement : $y_{i+1} = y_i + hy'_i$ où $y'_i = f(x_i, y_i)$, et $x_n = a$. Attention : on peut avoir $a < x0$, mais dans ce cas *h* doit être négatif

`baseeuler`

$a\ \{f\}\ x_0\ y_0\ n$ **`baseeuler`** $x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ *n* étant un entier, cette procédure dépose les points, calculés par la méthode d'Euler, de la courbe sur $[x_0, a]$ de la fonction *s* solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le pas *h* est calculé en fonction de l'entier *n*.

`baseeulermod`

$a\ \{f\}\ x_0\ y_0\ h$ **`baseeulermod`** $x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ dépose les points, calculés par la méthode d'Euler modifiée, de la courbe sur $[x_0, a]$ de la fonction *s* solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Attention : on peut avoir $a < x0$, mais dans ce cas *h* doit être négatif

`baseeulermod`

$a\ \{f\}\ x_0\ y_0\ n$ **`baseeulermod`** $x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ *n* étant un entier, cette procédure dépose les points, calculés par la méthode d'Euler modifiée, de la courbe sur $[x_0, a]$ de la fonction *s* solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le pas *h* est calculé en fonction de l'entier *n*.

`basemilne`

a $\{f\}$ x_0 y_0 h **basemilne** x_0 y_0 x_1 y_1 $\ldots x_n$ y_n $\longrightarrow$ dépose les points, calculés par la méthode de Milne, de la courbe sur $[x_0, a]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Attention : on peut avoir $a < x0$, mais dans ce cas h doit être négatif

basemilne

a $\{f\}$ x_0 y_0 n **basemilne** x_0 y_0 x_1 y_1 $\ldots x_n$ y_n $\longrightarrow$ n étant un entier, cette procédure dépose les points, calculés par la méthode de Milne, de la courbe sur $[x_0, a]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le pas h est calculé en fonction de l'entier n.

baserungekutta

a $\{f\}$ x_0 y_0 h **baserungekutta** x_0 y_0 x_1 y_1 $\ldots x_n$ y_n $\longrightarrow$ dépose les points, calculés par la méthode de Runge-Kutta, de la courbe sur $[x_0, a]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Attention : on peut avoir $a < x0$, mais dans ce cas h doit être négatif

baserungekutta

a $\{f\}$ x_0 y_0 n **baserungekutta** x_0 y_0 x_1 y_1 $\ldots x_n$ y_n $\longrightarrow$ n étant un entier, cette procédure dépose les points, calculés par la méthode de Runge-Kutta, de la courbe sur $[x_0, a]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le pas h est calculé en fonction de l'entier n.

baton

A **baton** $- \longrightarrow$ dessine un baton parallèle à l'axe Oy, dont une extrémité est le point A, et dont l'autre est sur l'axe Ox

bbpict

A [*xscale yscale*] $\{\alpha\}$ *string* **bbpict** $- \longrightarrow$ Se place au point A, puis affiche l'objet désigné par la chaîne *string* au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

bbtexlabel3d

bbtexlabel3d $- \longrightarrow$ Analogue 3d de la commande **bbtexlabel**

bbtexlabel

A [*xscale yscale*] $\{\alpha\}$ **bbtexlabel** $- \longrightarrow$ Se place au point A, puis dessine le label TEX au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

bbtext3d

bbtext3d $- \longrightarrow$ Analogue 3d de la commande **bbtext**

bbtext

string A [*xscale yscale*] $\{\alpha\}$ **bbtext** $- \longrightarrow$ Se place au point A, puis affiche la chaîne *string* au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

bcpict

A [*xscale yscale*] $\{\alpha\}$ *string* **bcpict** $- \longrightarrow$ Se place au point A, puis affiche l'objet désigné par la chaîne *string* au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

bctexlabel3d

bctexlabel3d $- \longrightarrow$ Analogue 3d de la commande **bctexlabel**

bctexlabel

A [*xscale yscale*] $\{\alpha\}$ **bctexlabel** $- \longrightarrow$ Se place au point A, puis dessine le label TEX au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

bctext3d

bctext3d $- \longrightarrow$ Analogue 3d de la commande **bctext**

bctext

string A [*xscale yscale*] $\{\alpha\}$ **bctext** $- \longrightarrow$ Se place au point A, puis affiche la chaîne *string* au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

bezier_curve_

array **bezier_curve_** $- \longrightarrow$ ajoute au chemin courant la courbe de Bézier spécifiée par le tableau de points *array*

bezier_curve

array **bezier_curve** – ⟶ trace la courbe de Bézier spécifiée par le tableau de points *array*

binomiale

k n p **binomiale** *a* ⟶ $a = C_n^k p^k (1-p)^{n-k}$

bissectrice

A B C **bissectrice** *D* ⟶ *D* est la bissectrice de l'angle $\widehat{ABC}$

bissectrice

A B C **bissectrice** *D* ⟶ la droite *D* bissectrice de l'angle $\widehat{ABC}$

blanc

– **blanc** – ⟶ sélectionne la couleur blanc

bleu

– **bleu** – ⟶ sélectionne la couleur bleu

blpict

A [*xscale yscale*] {α} *string* **blpict** – ⟶ Se place à aguche du point *A*, puis affiche l'objet désigné par la chaîne *string* au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument {α} sont optionnels

bltexlabel3d

bltexlabel3d – ⟶ Analogue 3d de la commande **bltexlabel**

bltexlabel

A [*xscale yscale*] {α} **bltexlabel** – ⟶ Se place à aguche du point *A*, puis dessine le label TEX au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument {α} sont optionnels

bltext3d

bltext3d – ⟶ Analogue 3d de la commande **bltext**

bltext

string A [*xscale yscale*] {α} **bltext** – ⟶ Se place à aguche du point *A*, puis affiche la chaîne *string* au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument {α} sont optionnels

bnode

string **bnode** – ⟶ Déclare un nœud rectangulaire encadré dont le nom est défini par *string*

boxit_all

– **boxit_all** – ⟶ sélectionne l'encadrement systématique des objets affichés par l'environnement 'picture'. En particulier, la gestion de l'affichage du texte ou des labels TEX est concernée

boxit

– **boxit** – ⟶ sélectionne l'encadrement du prochain objet affiché par l'environnement 'picture'. En particulier, la gestion de l'affichage du texte ou des labels TEX est concernée

boxit_none

– **boxit_none** – ⟶ désélectionne l'encadrement des prochains objets affichés par l'environnement 'picture'.

brpict

A [*xscale yscale*] {α} *string* **brpict** – ⟶ Se place à droite du point *A*, puis affiche l'objet désigné par la chaîne *string* au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument {α} sont optionnels

brpict

A [*xscale yscale*] {α} *string* **brpict** – ⟶ Se place en bas à droite du point *A*, puis affiche l'objet désigné par la chaîne *string* avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument {α} sont optionnels

brtexlabel3d

brtexlabel3d – ⟶ Analogue 3d de la commande **brtexlabel**

brtexlabel

A [*xscale yscale*] {α} **brtexlabel** – ⟶ Se place à droite du point *A*, puis dessine le label TEX au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument {α} sont optionnels

brtexlabel

A [*xscale yscale*] $\{\alpha\}$ **brtexlabel** $- \longrightarrow$ Se place en bas à droite du point A, puis dessine le label TEX en cours avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

brtext3d

brtext3d $- \longrightarrow$ Analogue 3d de la commande **brtext**

brtext

string A [*xscale yscale*] $\{\alpha\}$ **brtext** $- \longrightarrow$ Se place à droite du point A, puis affiche la chaîne *string* au niveau de la *baseline*, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

brtext

string A [*xscale yscale*] $\{\alpha\}$ **brtext** $- \longrightarrow$ Se place en bas à droite du point A, puis affiche la chaîne *string* avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

bubblesort

$array_1$ **bubblesort** $array_2$ $\longrightarrow$ le tableau de réels $array_2$ est le résultat du tri à bulle sur le tableau de réels $array_1$.

C

CamView

$x\ y\ z$ **CamView** $X\ Y$ $\longrightarrow$ On projete le point 3d sur le plan de représentation de la caméra, selon le mode de représentation

capply

[$cerc_0 \ldots cerc_n$] f **capply** [$b_0 \ldots b_n$] ou $- \longrightarrow$ construit un nouveau tableau en répétant, pour i variant de 0 à n, l'opération suivante : déposer le cercle C_i puis exécuter f. Si à la fin de cette opération le tableau est vide, alors il est enlevé de la pile.

cbpict

A [*xscale yscale*] $\{\alpha\}$ *string* **cbpict** $- \longrightarrow$ Se place au point A, puis affiche l'objet désigné par la chaîne *string* centré, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

cbtexlabel3d

cbtexlabel3d $- \longrightarrow$ Analogue 3d de la commande **cbtexlabel**

cbtexlabel

A [*xscale yscale*] $\{\alpha\}$ **cbtexlabel** $- \longrightarrow$ Se place au point A, puis dessine le label TEX en cours centré, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

cbtext3d

cbtext3d $- \longrightarrow$ Analogue 3d de la commande **cbtext**

cbtext

string A [*xscale yscale*] $\{\alpha\}$ **cbtext** $- \longrightarrow$ Se place au point A, puis affiche la chaîne *string* centré, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

ccpict

A [*xscale yscale*] $\{\alpha\}$ *string* **ccpict** $- \longrightarrow$ Se place au point A, puis affiche l'objet désigné par la chaîne *string* centré, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

cctexlabel3d

cctexlabel3d $- \longrightarrow$ Analogue 3d de la commande **cctexlabel**

cctexlabel

A [*xscale yscale*] $\{\alpha\}$ **cctexlabel** $- \longrightarrow$ Se place au point A, puis dessine le label TEX en cours centré, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

cctext3d

cctext3d $- \longrightarrow$ Analogue 3d de la commande **cctext**

cctext

string A [*xscale yscale*] $\{\alpha\}$ **cctext** $- \longrightarrow$ Se place au point A, puis affiche la chaîne *string* centré, avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

ceiling

num$_1$ **ceiling** *num*$_2$ ⟶ plafond de *num*$_1$

Cercle_

$\alpha\ \beta$ *cerc* **Cercle_** – ⟶ ajoute au chemin courant la portion de cercle allant du point de paramètre α au point de paramètre β

Cercle

$\alpha\ \beta$ *cerc* **Cercle** – ⟶ trace la portion de cercle spécifiée

cercle_

cerc **cercle_** – ⟶ ajoute au chemin courant le cercle spécifié

cercle

cerc **cercle** – ⟶ trace le cercle spécifié

cframe_

$A\ L\ \ell$ **cframe_** – ⟶ ajoute au chemin courant trace le rectangle dont le point A est le centre, de dimension horizontale L et de dimension verticale ℓ

cframe

$A\ L\ \ell$ **cframe** – ⟶ trace le rectangle dont le point A est le centre, de dimension horizontale L et de dimension verticale ℓ

champvecteur

$f\ step_1\ step_2\ \ell$ **champvecteur** – ⟶ Trace les vecteurs de norme ℓ définis par $y' = f(x, y)$, en partant de $(xmin, ymin)$ et jusqu'à $(xmax, ymax)$ et en tenant compte des pas $step_1$ (sur Ox) et $step_2$ (sur O_y)

circleit_all

– **circleit_all** – ⟶ sélectionne l'encerclement systématique des objets affichés par l'environnement 'picture'. En particulier, la gestion de l'affichage du texte ou des labels TEX est concernée

circleit

– **circleit** – ⟶ sélectionne l'encerclement du prochain objet affiché par l'environnement 'picture'. En particulier, la gestion de l'affichage du texte ou des labels TEX est concernée

circleit_none

– **circleit_none** – ⟶ désélectionne l'encerclement des prochains objets affichés par l'environnement 'picture'.

circ

A **circ** – ⟶ dessine un point cerclé en A dans le repère jps

closepath

– **closepath** – ⟶ connecte le sous-chemin à son point de départ

clpict

A [*xscale yscale*] $\{\alpha\}$ *string* **clpict** – ⟶ Se place à aguche du point A, puis affiche l'objet désigné par la chaîne *string* centré, avec l'échelle $(xscale, yscale)$ et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

cltexlabel3d

cltexlabel3d – ⟶ Analogue 3d de la commande **cltexlabel**

cltexlabel

A [*xscale yscale*] $\{\alpha\}$ **cltexlabel** – ⟶ Se place à aguche du point A, puis dessine le label TEX en cours centré, avec l'échelle $(xscale, yscale)$ et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

cltext3d

cltext3d – ⟶ Analogue 3d de la commande **cltext**

cltext

string A [*xscale yscale*] $\{\alpha\}$ **cltext** – ⟶ Se place à aguche du point A, puis affiche la chaîne *string* centré, avec l'échelle $(xscale, yscale)$ et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

Cnode

string **Cnode** – ⟶ Déclare un nœud circulaire de rayon fixe *Circleradius* dont le nom est défini par *string*

cnode

string **cnode** – ⟶ Déclare un nœud circulaire dont le nom est défini par *string*

Cnp

$n\ p$ **Cnp** $c \longrightarrow c = C_n^p = A_n^p/p!$

coeffdir

D **coeffdir** $a \longrightarrow a$ est le coefficient directeur de la droite D si celle-ci n'est pas verticale, erreur sinon

ComputeCamera

$-$ **ComputeCamera** $- \longrightarrow$ *Compute vectors usefull to CamView.*

conjugue

z **conjugue** $\overline{z} \longrightarrow \overline{z}$ est le conjugué du complexe z

continu

$-$ **continu** $- \longrightarrow$ sélectionne le tracé de type *continu*

correlation

$array_1\ array_2$ **correlation** $r \longrightarrow$ le réel r est la coefficient de corrélation de la série double définie par les tableaux de réels $array_1$ et $array_2$

cosh

a **cosh** $c \longrightarrow c = \operatorname{ch} a$

cos

a **cos** $c \longrightarrow c = \cos a$ (a en degré)

Cos

a **Cos** $c \longrightarrow c = \cos a$ (a en radian)

cotanh

a **cotanh** $c \longrightarrow c = \coth a$

cotan

a **cotan** $c \longrightarrow c = \operatorname{cotan} a$ (a en degré)

coTan

a **coTan** $c \longrightarrow c = \operatorname{cotan} a$ (a en radian)

Courbe_

$a\ b\ \{f\}$ **Courbe_** $- \longrightarrow$ ajoute au chemin courant la courbe représentative de la fonction f pour x allant de a à b

Courbe

$a\ b\ \{f\}$ **Courbe** $- \longrightarrow$ trace la courbe représentative de la fonction f sur l'intervalle $[a;b]$

Courbe_

$a\ b\ proc$ **Courbe_** $- \longrightarrow$ ajoute au chemin courant la courbe représentative pour x allant de a à b de la fonction définie par l'exécutable *proc*

Courbe

$a\ b\ proc$ **Courbe** $- \longrightarrow$ trace la courbe représentative sur l'intervalle $[A;B]$ de la fonction définie par l'exécutable *proc*

courbe_

$\{f\}$ **courbe_** $- \longrightarrow$ ajoute au chemin courant la courbe représentative de la fonction f sur l'intervalle $[xmin;xmax]$

courbe

$\{f\}$ **courbe** $- \longrightarrow$ trace la courbe représentative de la fonction f sur l'intervalle $[xmin;xmax]$

courbe_

proc **courbe_** $- \longrightarrow$ ajoute au chemin courant la courbe représentative sur l'intervalle $[xmin;xmax]$ de la fonction définie par l'exécutable *proc*

courbe

proc **courbe** $- \longrightarrow$ trace la courbe représentative sur l'intervalle $[xmin;xmax]$ de la fonction définie par l'exécutable *proc*

Courbeparam_

$a\ b\ proc_1\ proc_2$ **Courbeparam_** $- \longrightarrow$ ajoute au chemin courant la courbe paramétrée définie, pour t variant de a à b, par les exécutables $proc_1$ et $proc_2$

Courbeparam

a b $proc_1$ $proc_2$ **Courbeparam** – ⟶ trace sur l'intervalle $[a;b]$ la courbe paramétrée définie par les exécutables $proc_1$ et $proc_2$

Courbeparam_

a b $\{X\}$ $\{Y\}$ **Courbeparam_** – ⟶ ajoute au chemin courant la courbe paramétrée $t \mapsto (X(t);Y(t))$ pour t variant de a à b

Courbeparam

a b $\{X\}$ $\{Y\}$ **Courbeparam** – ⟶ trace la courbe paramétrée $t \mapsto (X(t);Y(t))$ sur l'intervalle $[a;b]$

courbeparam_

$proc_1$ $proc_2$ **courbeparam_** – ⟶ ajoute au chemin courant la courbe paramétrée définie sur l'intervalle $[tmin;tmax]$ par les exécutables $proc_1$ et $proc_2$

courbeparam

$proc_1$ $proc_2$ **courbeparam** – ⟶ trace sur l'intervalle $[tmin;tmax]$ la courbe paramétrée définie par les exécutables $proc_1$ et $proc_2$

courbeparam_

$\{X\}$ $\{Y\}$ **courbeparam_** – ⟶ ajoute au chemin courant la courbe paramétrée $t \mapsto (X(t);Y(t))$ sur l'intervalle $[tmin;tmax]$

courbeparam

$\{X\}$ $\{Y\}$ **courbeparam** – ⟶ trace la courbe paramétrée $t \mapsto (X(t);Y(t))$ sur l'intervalle $[tmin;tmax]$

covariance

$array_1$ $array_2$ **covariance** c ⟶ le réel c est la covariance de la série double définie par les tableaux de réels $array_1$ et $array_2$

cpoint

α *cerc* **cpoint** M ⟶ dépose sur la pile les coordonnées du point M du cercle *cerc* correspondant à l'angle α

crpict

A [*xscale yscale*] $\{\alpha\}$ *string* **crpict** – ⟶ Se place à droite du point A, puis affiche l'objet désigné par la chaîne *string* centré, avec l'échelle $(xscale, yscale)$ et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

crtexlabel3d

crtexlabel3d – ⟶ Analogue 3d de la commande **crtexlabel**

crtexlabel

A [*xscale yscale*] $\{\alpha\}$ **crtexlabel** – ⟶ Se place à droite du point A, puis dessine le label TEX en cours centré, avec l'échelle $(xscale, yscale)$ et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

crtext3d

crtext3d – ⟶ Analogue 3d de la commande **crtext**

crtext

string A [*xscale yscale*] $\{\alpha\}$ **crtext** – ⟶ Se place à droite du point A, puis affiche la chaîne *string* centré, avec l'échelle $(xscale, yscale)$ et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

currentpoint

– **currentpoint** x y ⟶ renvoie les coordonnées du point courant

curveto

x_1 y_1 x_2 y_2 x_3 y_3 **curveto** – ⟶ ajoute une section cubique de Bézier

cyan

– **cyan** – ⟶ sélectionne la couleur cyan

D

dapply

[$d_0 \dots d_n$] f **dapply** [$b_0 \dots b_n$] ou – ⟶ construit un nouveau tableau en répétant, pour i variant de 0 à n, l'opération suivante : déposer la droite d_i puis exécuter f. Si à la fin de cette opération le tableau est vide, alors il est enlevé de la pile.

dashpoint

point **dashpoint** – ⟶ dessine le point spécifié avec projection sur les axes en pointillé

dashpoints

[*point*$_1$ ... *point*$_n$ **]** **dashpoints** $- \longrightarrow$ dessine les points spécifiés avec projection sur les axes en pointillé

dbpict

A [*xscale yscale*] $\{\alpha\}$ *string* **dbpict** $- \longrightarrow$ Se place en bas du point *A*, puis affiche l'objet désigné par la chaîne *string* avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

dbtexlabel3d

dbtexlabel3d $- \longrightarrow$ Analogue 3d de la commande **dbtexlabel**

dbtexlabel

A [*xscale yscale*] $\{\alpha\}$ **dbtexlabel** $- \longrightarrow$ Se place en bas du point *A*, puis dessine le label TEX en cours avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

dbtext3d

dbtext3d $- \longrightarrow$ Analogue 3d de la commande **dbtext**

dbtext

string A [*xscale yscale*] $\{\alpha\}$ **dbtext** $- \longrightarrow$ Se place en bas du point *A*, puis affiche la chaîne *string* avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

dcpict

A [*xscale yscale*] $\{\alpha\}$ *string* **dcpict** $- \longrightarrow$ Se place en bas du point *A*, puis affiche l'objet désigné par la chaîne *string* avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

dctexlabel3d

dctexlabel3d $- \longrightarrow$ Analogue 3d de la commande **dctexlabel**

dctexlabel

A [*xscale yscale*] $\{\alpha\}$ **dctexlabel** $- \longrightarrow$ Se place en bas du point *A*, puis dessine le label TEX en cours avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

dctext3d

dctext3d $- \longrightarrow$ Analogue 3d de la commande **dctext**

dctext

string A [*xscale yscale*] $\{\alpha\}$ **dctext** $- \longrightarrow$ Se place en bas du point *A*, puis affiche la chaîne *string* avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

defcercle

cerc name **defcercle** $- \longrightarrow$ associe le cercle *cerc* au nom *name*

defdroite

d name **defdroite** $- \longrightarrow$ associe la droite *d* au nom *name*

defpoint3d

x y z lit **defpoint3d** $- \longrightarrow$ Associe le littéral *lit* au point (x, y, z)

defpoint

x y name **defpoint** $- \longrightarrow$ affecte le nom *name* au couple de nombres (x, y)

demidroite

demidroite *A B option* $\longrightarrow -$: Trace la demi-droite $[AB)$. *option* est une chaine de caractères optionnelle indiquant le type de terminaison de ligne en *A*

diamcercle

A B **diamcercle** *cerc* $\longrightarrow$ *cerc* est le cercle de diamètre $[AB]$

diamond

A **diamond** $- \longrightarrow$ dessine un losange au point *A* dans le repère *jps*

dianode

string **dianode** $- \longrightarrow$ Déclare un nœud losange (diamond) dont le nom est défini par *string*

dich_solve

a b ε $\{f\}$ **dich_solve** $x \longrightarrow \{f\}$ est un exécutable, le produit $f(a) \times proc(b)$ est négatif, et ε est un réel strictement positif. Alors x est un réel tel que $f(x+\varepsilon) \times f(x-\varepsilon) < 0$

dimmatrix

M **dimmatrix** *m n* $\longrightarrow$ dépose sur la pile les dimensions de la matrice *M* (*m* lignes, *n* colonnes)

dir

α **dir** $\vec{u} \longrightarrow \vec{u}$ est un vecteur correspondant à l'angle α (en degrés)

dir

α **dir** $\vec{v} \longrightarrow \vec{v}$ est le vecteur de norme 1 d'angle $(\widehat{\vec{u},\vec{v}}) = \alpha$ où $\vec{u}$ désigne le vecteur unitaire de l'axe des abscisses

distance3d

$A\ B$ **distance3d** $d \longrightarrow$ calcule la distance $d = AB$

distance

$A\ B$ **distance** $\ell \longrightarrow$ le nombre réel ℓ est la distance séparant les points A et B

divc

$z\ z'$ **divc** $Z \longrightarrow Z = z/z'$ est le quotient des complexes z et z'

div

$a\ b$ **div** $c \longrightarrow c = a/b$

dlpict

A [*xscale yscale*] $\{\alpha\}$ *string* **dlpict** $- \longrightarrow$ Se place en bas à gauche du point A, puis affiche l'objet désigné par la chaîne *string* avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

dltexlabel3d

dltexlabel3d $- \longrightarrow$ Analogue 3d de la commande **dltexlabel**

dltexlabel

A [*xscale yscale*] $\{\alpha\}$ **dltexlabel** $- \longrightarrow$ Se place en bas à gauche du point A, puis dessine le label TEX en cours avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

dltext3d

dltext3d $- \longrightarrow$ Analogue 3d de la commande **dltext**

dltext

string A [*xscale yscale*] $\{\alpha\}$ **dltext** $- \longrightarrow$ Se place en bas à gauche du point A, puis affiche la chaîne *string* avec l'échelle (*xscale*, *yscale*) et après une rotation d'angle α. Le tableau d'échelle et l'argument $\{\alpha\}$ sont optionnels

dotangle

- *dotangle* : paramètre indiquant l'angle en degrés de la rotation à appliquer pour les dessins de type point. **valeur par défaut :** 0

dotscale

- *dotscale* : exécutable indiquant les échelles horizontale et verticale à appliquer pour les dessins de type point. **valeur par défaut :** { 1 1 }

dotsize

- *dotsize* : dimension en points postcript paramétrant la taille des dessins de type point. **valeur par défaut :** 4

dotted

$-$ **dotted** $- \longrightarrow$ sélectionne le tracé de type *pointilles*

down

$-$ **down** $\vec{u} \longrightarrow \vec{u}$ est le vecteur $(0, -1)$

droite

d **droite** $- \longrightarrow$ trace la droite d

dupc

cerc **dupc** *cerc cerc* $\longrightarrow$ duplique le cercle au dessus de la pile

dupd

D **dupd** $D\ D \longrightarrow$ duplique la droite au dessus de la pile

dupmatrix

M **dupmatrix** $M\ M' \longrightarrow$ dépose une nouvelle instance de M sur la pile

dupp3d

A **dupp3d** $A\ A \longrightarrow$ Dupplique le point 3d au dessus de la pile

dupp

A **dupp** *A A* ⟶ duplique le point au dessus de la pile

dupv3d

u **dupv3d** *u u* ⟶ Dupplique le vecteur *u* au dessus de la pile

dx_boxit

- *dx_boxit* : taille, en points postscript, de l'espace horizontal entre un objet et son cadre placé par l'environnement 'picture' . **valeur par défaut :** 0

dy_boxit

- *dy_boxit* : taille, en points postscript, de l'espace vertical entre un objet et son cadre placé par l'environnement 'picture' . **valeur par défaut :** 0

E

ecarttype

[$a_0 \dots a_n$] **ecarttype** σ ⟶ le réel σ est l'écart-type de la série des a_i.

ell2pol

ell **ell2pol** *pol* ⟶ le polygône *pol* est constitué des 4 sommets de l'ellipse

ella

ell **ella** *a* ⟶ *a* est la longueur du demi-axe horizontal de l'ellipse *ell*

ellangle

ell **ellangle** α ⟶ α est l'angle de l'ellipse *ell*

ellb

ell **ellb** *b* ⟶ *b* est la longueur du demi-axe vertical de l'ellipse *ell*

ellcentre

ell **ellcentre** *A* ⟶ le points *A* est le centre de l'ellipse *ell*

ellipseangle

- *ellipseangle* : angle que fait le premier axe d'une ellipse avec l'horizontale lorsque l'on trace cette ellipse avec la syntaxe allégée des commandes **ellipse** et **Ellipse**. **valeur par défaut :** 0

Ellipse_

α β *ell* **Ellipse_** – ⟶ ajoute au chemin courant la portion de l'ellipse entre les points de paramètres respectifs α et β (dans ce sens).

Ellipse

α β *ell* **Ellipse** – ⟶ trace la portion de l'ellipse entre les points de paramètres respectifs α et β.

Ellipse

α β *I a b* **Ellipse** – ⟶ trace la portion spécifiée de l'ellipse de centre *I* et de demi-axes *a* et *b*. L'angle que fait le premier axe avec l'horizontale est déterminé par le paramètre *ellipseangle*

ellipse_

ell **ellipse_** – ⟶ ajoute au chemin courant le chemin de l'ellipse spécifiée

ellipse

ell **ellipse** – ⟶ trace l'ellipse spécifiée

ellipse

I a b **ellipse** – ⟶ trace l'ellipse de centre *I* et de demi-axes *a* et *b*. L'angle que fait le premier axe avec l'horizontale est déterminé par le paramètre *ellipseangle*

e

– **e** $2,718$ ⟶ le nombre *e*

enode

x y string **enode** – ⟶ Déclare un nœud vide (*emptynode*) au point de coordonées (x, y) et dont le nom est défini par *string*

Epoint

α *ell* **Epoint** *A* ⟶ *A* est le point de l'ellipse *ell* tel que $(\vec{u}, \overrightarrow{\Omega A}) = \alpha$, où Ω désigne le centre de l'ellipse, et $\vec{u}$ le vecteur de base de l'axe *Ox*.

epoint

t ell **epoint** *A* ⟶ *A* est le point de l'ellipse *ell* de paramètre *t* (avec le paramétrage $(a \cos t, b \sin t)$)

eqc

$z\ z'$ **eqc** *bool* ⟶ le booléen *bool* vaut **true** si les complexes *z* et z' sont égaux, **false** sinon.

eqp

$A\ B$ **eqp** *bool* $\longrightarrow$ le booléen *bool* vaut **true** si les points A et B sont confondus, **false** sinon

Euler

$a\ b\ \{f\}\ x_0\ y_0\ h$ **Euler** $x_{-n}\ y_{-n}\ \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ dépose les points, calculés par la méthode d'Euler, de la courbe sur $[a, b]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est un réel positif, il détermine le pas entre chaque point calculé

Euler

$a\ b\ \{f\}\ x_0\ y_0\ n$ **Euler** $x_{-n}\ y_{-n}\ \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ n étant un entier, cette procédure dépose les points, calculés par la méthode d'Euler, de la courbe sur $[a, b]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est calculé en fonction de n.

euler

$\{f\}\ x_0\ y_0\ h$ **euler** $x_{-n}\ y_{-n}\ \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ dépose les points, calculés par la méthode d'Euler, de la courbe sur $[xmin, xmax]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est un réel positif, il détermine le pas entre chaque point calculé

euler

$\{f\}\ x_0\ y_0\ n$ **euler** $x_{-n}\ y_{-n}\ \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ n étant un entier, cette procédure dépose les points, calculés par la méthode d'Euler, de la courbe sur $[xmin, xmax]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est calculé en fonction de n.

Eulermod

$a\ b\ \{f\}\ x_0\ y_0\ h$ **Eulermod** $x_{-n}\ y_{-n}\ \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ dépose les points, calculés par la méthode d'Euler modifiée, de la courbe sur $[a, b]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est un réel positif, il détermine le pas entre chaque point calculé

Eulermod

$a\ b\ \{f\}\ x_0\ y_0\ n$ **Eulermod** $x_{-n}\ y_{-n}\ \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ n étant un entier, cette procédure dépose les points, calculés par la méthode d'Euler modifiée, de la courbe sur $[a, b]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est calculé en fonction de n.

eulermod

$\{f\}\ x_0\ y_0\ h$ **eulermod** $x_{-n}\ y_{-n}\ \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ dépose les points, calculés par la méthode d'Euler modifiée, de la courbe sur $[xmin, xmax]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est un réel positif, il détermine le pas entre chaque point calculé

eulermod

$\{f\}\ x_0\ y_0\ n$ **eulermod** $x_{-n}\ y_{-n}\ \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow$ n étant un entier, cette procédure dépose les points, calculés par la méthode d'Euler modifiée, de la courbe sur $[xmin, xmax]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est calculé en fonction de n.

exch_l

$M\ i\ j$ **exch_l** $- \longrightarrow$ échange les lignes d'indice i et d'indice j dans la matrice M

exec

any **exec** $- \longrightarrow$ exécute un objet arbitraire

Exp

a **Exp** $c \longrightarrow c = \exp(a) = e^a$

exp

$a\ n$ **exp** $c \longrightarrow c = a^n$

exposant

string **exposant** $- \longrightarrow$ affiche la chaîne *string* dans la police courante, après une réduction à 70% de la taille courante et un déplacement vertical (40% de *fontsize*) par rapport au point courant

F

factorielle

n **factorielle** $b \longrightarrow b = a!$

Fillcourbe

$a\ b\ \{f\}$ **Fillcourbe** $- \longrightarrow$ remplit, suivant les indications de *fillstyle*, le domaine plan délimité par l'axe Ox, la courbe représentative de f, et les droites verticales $x = a$ et $x = b$

Fillcourbe

$a\ b$ *proc* **Fillcourbe** $- \longrightarrow$ remplit, suivant les indications de *fillstyle*, le domaine plan délimité par l'axe Ox, la courbe représentative de la fonction numérique définie par *proc*, et les droites verticales $x = a$ et $x = b$

`fillcourbe`

$\{f\}$ **`fillcourbe`** $- \longrightarrow$ remplit, suivant les indications de $fillstyle$, le domaine plan délimité par l'axe Ox, la courbe représentative de f, et les droites verticales $x = xmin$ et $x = xmax$

`fillcourbe`

proc **`fillcourbe`** $- \longrightarrow$ remplit, suivant les indications de $fillstyle$, le domaine plan délimité par l'axe Ox, la courbe représentative de la fonction numérique définie par *proc*, et les droites verticales $x = xmin$ et $x = xmax$

`Fillcourbes`

$a\ b\ \{f\}\ \{g\}$ **`Fillcourbes`** $- \longrightarrow$ remplit, suivant les indications de $fillstyle$, le domaine plan délimité par les courbes représentatives des fonctions f et g, et les droites verticales $x = a$ et $x = b$

`Fillcourbes`

$a\ b\ proc_1\ proc_2$ **`Fillcourbes`** $- \longrightarrow$ remplit, suivant les indications de $fillstyle$, le domaine plan délimité par l'axe Ox, les courbes représentatives des fonctions numérique définies par $proc_1$ et $proc_2$, et les droites verticales $x = a$ et $x = b$

`fillcourbes`

$\{f\}\ \{g\}$ **`fillcourbes`** $- \longrightarrow$ remplit, suivant les indications de $fillstyle$, le domaine plan délimité par les courbes représentatives des fonctions f et g, et les droites verticales $x = xmin$ et $x = xmax$

`fillcourbes`

$proc_1\ proc_2$ **`fillcourbes`** $- \longrightarrow$ remplit, suivant les indications de $fillstyle$, le domaine plan délimité par l'axe Ox, les courbes représentatives des fonctions numérique définies par $proc_1$ et $proc_2$, et les droites verticales $x = xmin$ et $x = xmax$

`flattenpath`

$-$ **`flattenpath`** $- \longrightarrow$ convertit les courbes en suites de segments de droites

`floor`

num_1 **`floor`** $num_2 \longrightarrow$ plancher de num_1

$fontsize$

- $fontsize$: taille, en points postscript, de la prochaine fonte chargée. **valeur par défaut :** $12,5$

$frameangle$

- $frameangle$: angle que fait avec l'horizontale le côté « bas » d'un rectangle que l'on trace avec la syntaxe allégée d'une commande **`frame`** ou dérivée. **valeur par défaut :** 0

`frame_`

$A\ B$ **`frame_`** $- \longrightarrow$ ajoute au chemin courant le rectangle dont les points A et B sont repectivement les coins inférieur droit et supérieur gauche

`frame`

$A\ B$ **`frame`** $- \longrightarrow$ trace le rectangle dont les points A et B sont repectivement les coins inférieur droit et supérieur gauche

`fresnelC`

x **`fresnelC`** $y \longrightarrow y$ est l'image de x par la fonction de Fresnel $x \mapsto \displaystyle\int_0^x \cos\left(\frac{\pi t^2}{2}\right)\,dt$

`fresnelS`

x **`fresnelS`** $y \longrightarrow y$ est l'image de x par la fonction de Fresnel $x \mapsto \displaystyle\int_0^x \sin\left(\frac{\pi t^2}{2}\right)\,dt$

`fuz`

$[\ a_0 \dots a_n\ ]\ [\ b_0 \dots b_n\ ]$ **`fuz`** $[\ a_0\ b_0 \dots a_n\ b_n\ ] \longrightarrow$ fusionne les 2 tableaux de même tailles donnés en entrée

G

`genMi`

$/M\ n$ **`genMi`** $M1\ M2 \dots Mn \longrightarrow$ dépose les valeurs des noms $/M1$, $/M2$, ..., $/Mn$ sur la pile

`genMiname`

$/M\ n$ **`genMiname`** $M1\ M2 \dots Mn \longrightarrow$ dépose les noms $/M1$, $/M2$, ..., $/Mn$ sur la pile

`genpolyreg`

$I\ r\ n\ \alpha$ **`genpolyreg`** $array \longrightarrow array$ est le tableau des n points sommets du polygône régulier de centre I tel que l'angle entre le vecteur unité sur Ox et le vecteur IA soit de α degrés (où A est le « premier » sommet du polygône)

`GetCamPos`

– **GetCamPos** *x y z* ⟶ Dépose sur la pile les coordonnées de la caméra

GetCamUp

– **GetCamUp** $U_x\ U_y\ U_z$ ⟶ *Set Camera Up vector.*

GetCamVec

– **GetCamVec** $V_x\ V_y\ V_z$ ⟶ *Get Camera Looking vector.*

get_Ci

M i **get_Ci** *array* ⟶ le tableau *array* représenta la colonne d'indice *i* de la matrice *M*

get_ij

M i j **get_ij** *a* ⟶ *a* est le coefficient d'indice (i, j) de la matrice *M*

get_Li

M i **get_Li** *array* ⟶ *array* représente la ligne d'indice *i* de la matrice *M*

getp3d

getp3d ⟶

getp

$[\ A_0 \dots A_n\ ]\ i$ **getp** A_i ⟶ donne le point d'indice *i* du tableau de points donné en entrée.

gradangle

- *gradangle* : Angle utilisé pour les gradients de couleurs. **valeur par défaut :** 0

gradientfill

gradientfill $couleur_1$ $couleur_2$ *m* – ⟶ remplit le domaine d'incrustation en cours par un gradient de couleur passant de $couleur_1$ à $couleur_2$. Le nombre *m* appartient à [0; 1]; il indique à quel moment doit atteindre la couleur $couleur_2$

gris

– **gris** – ⟶ sélectionne la couleur gris

H

Hachcourbe

a b $\{f\}$ **Hachcourbe** – ⟶ hachure le domaine plan délimité par l'axe *Ox*, la courbe représentative de *f*, et les droites verticales $x = a$ et $x = b$

Hachcourbe

a b proc **Hachcourbe** – ⟶ hachure le domaine plan délimité par l'axe *Ox*, la courbe représentative de la fonction numérique définie par *proc*, et les droites verticales $x = a$ et $x = b$

hachcourbe

$\{f\}$ **hachcourbe** – ⟶ hachure le domaine plan délimité par l'axe *Ox*, la courbe représentative de *f*, et les droites verticales $x = xmin$ et $x = xmax$

hachcourbe

proc **hachcourbe** – ⟶ hachure le domaine plan délimité par l'axe *Ox*, la courbe représentative de la fonction numérique définie par *proc*, et les droites verticales $x = xmin$ et $x = xmax$

Hachcourbes

a b $\{f\}$ $\{g\}$ **Hachcourbes** – ⟶ hachure le domaine plan délimité par les courbes représentatives des fonctions *f* et *g*, et les droites verticales $x = a$ et $x = b$

Hachcourbes

a b $proc_1$ $proc_2$ **Hachcourbes** – ⟶ hachure le domaine plan délimité par l'axe *Ox*, les courbes représentatives des fonctions numérique définies par $proc_1$ et $proc_2$, et les droites verticales $x = a$ et $x = b$

hachcourbes

$\{f\}$ $\{g\}$ **hachcourbes** – ⟶ hachure le domaine plan délimité par les courbes représentatives des fonctions *f* et *g*, et les droites verticales $x = xmin$ et $x = xmax$

hachcourbes

$proc_1$ $proc_2$ **hachcourbes** – ⟶ hachure le domaine plan délimité par l'axe *Ox*, les courbes représentatives des fonctions numérique définies par $proc_1$ et $proc_2$, et les droites verticales $x = xmin$ et $x = xmax$

hachure

– **hachure** – ⟶ hachure l'ensemble de la fenêtre courante

hadjust

- *hadjust* : taille, en points postscript, du décalage horizontal appliqué, s'il y a lieu par les commandes de positionnement de l'environnement 'picture' . **valeur par défaut :** $3,75$

hangle

- *hangle* : l'angle en degrés que font les hachures avec l'horizontale. **valeur par défaut :** -45

hcolor

- *hcolor* : couleur d'un hachure. **valeur par défaut :** $\{\}$

`homcercle`

cerc I α **`homcercle`** $cerc' \longrightarrow$ le cercle $cerc'$ est l'image du cercle *cerc* par l'homothétie de centre I, de rapport α.

`homell`

ell I α **`homell`** $ell' \longrightarrow$ l'ellipse ell' est l'image de l'ellipse *ell* par l'homothétie de centre I, de rapport α.

`hompoint`

A I α **`hompoint`** $A' \longrightarrow$ le point A' est l'image du point A par l'homothétie de centre I, de rapport α. Autrement dit $\overrightarrow{IA'} = \alpha\overrightarrow{IA}$

`hompol`

pol I α **`hompol`** $pol' \longrightarrow$ le polygône pol' est l'image du polygône *pol* par l'homothétie de centre I, de rapport α.

hstep

- *hstep* : l'espace en points postscript séparant 2 hachures. **valeur par défaut :** 7

hwidth

- *hwidth* : l'épaisseur du trait en points postscript pour une hachure. **valeur par défaut :** $0,8$

I

`IAcercle`

I A **`IAcercle`** *cerc* $\longrightarrow$ *cerc* est le cercle de centre I passant par A

`idiv`

a b **`idiv`** $q \longrightarrow q$ est le quotient de la division euclidienne de a par b

`idmatrix`

m n **`idmatrix`** $M \longrightarrow$ dépose une nouvelle matrice identité (m,n) sur la pile

`ifelse`

bool $proc_1$ $proc_2$ **`ifelse`** $- \longrightarrow$ exécute $proc_1$ si *bool* est *true*, $proc_2$ si *bool* est *false*

`if`

bool proc **`if`** $- \longrightarrow$ exécute *proc* si *bool* est *true*

`indice`

string **`indice`** $- \longrightarrow$ affiche la chaîne *string* dans la police courante, après une réduction à 70% de la taille courante et un déplacement vertical (20% de *fontsize*) par rapport au point courant

`intercercle`

$cerc_1$ $cerc_2$ **`intercercle`** A $A' \longrightarrow$ les points A et A' sont les points d'intersection du cercle $cerc_1$ avec le cercle $cerc_2$, triés par la fonction *ordonnepoints*. Comme d'habitude, l'appel de cette fonction provoque une erreur si ces cercles n'ont pas de point commun.

`interdroitecercle`

D *cerc* **`interdroitecercle`** A $A' \longrightarrow$ les points A et A' sont les points d'intersection de la droite D avec le cercle *cerc*, triés par la fonction *ordonnepoints*

`interdroiteell`

D *ell* **`interdroiteell`** A $A' \longrightarrow$ les points A et A' sont les points d'intersection de la droite D avec l'ellipse *ell*, triés par la fonction *ordonnepoints*

`interdroite`

D D' **`interdroite`** $A \longrightarrow$ si les droites D et D' sont sécantes, alors A est leur point d'intersection. Erreur sinon

J

`jaune`

$-$ **`jaune`** $- \longrightarrow$ sélectionne la couleur jaune

`jtoppoint`

X Y **`jtoppoint`** x $y \longrightarrow$ Reçoit les coordonnées (X,Y) dans le repère *jps* et renvoie les coordonnées (x,y) dans le repère postscript

L

lambdav3d

$\lambda\ \vec{u}$ **lambdav3d** $\vec{v} \longrightarrow$ Le vecteur $\vec{v}$ vérifie $\vec{v} = \lambda\vec{u}$

left

$-$ **left** $\vec{u} \longrightarrow \vec{u}$ est le vecteur $(-1, 0)$

ligne3d

array **ligne3d** $- \longrightarrow$ Analogue 3d de la commande **ligne**

ligne_

array **ligne_** $- \longrightarrow$ ajoute au chemin courant la ligne définie par le tableau de points *array*

ligne

array string **ligne** $- \longrightarrow$ trace la ligne définie par le tableau de points *array*. Les extrémités de la ligne sont décrites par la chaîne optionnelle *string*

linearc

- *linearc* : indique au format le rayon (dans le repère jps) de l'arc de cercle reliant deux segments de droites pour les commandes **ligne**, **polygone**, ainsi que les **frame** et dérivés. . **valeur par défaut :** 0

line

A B string **line** $- \longrightarrow$ trace le segment de droite $[AB]$. Les extrémités du segment sont décrites par la chaîne optionnelle *string*

lineto

$x\ y$ **lineto** $- \longrightarrow$ ajoute une ligne droite jusqu'en (x, y)

ln

a **ln** $c \longrightarrow c = \ln a$

log_bande

i **log_bande** $- \longrightarrow i$ est un entier. trace 10 traits horizontaux et 10 traits verticaux sur $\left]10^{i-1}; 10^{i}\right]$ en utilisant les commandes **hrule** et **vrule**

logicNInput

- *logicNInput* : Nombre d'entrées de la cellule. **valeur par défaut :** 2

logicUnit

- *logicUnit* : Échelle pour le dessin du symbole. **valeur par défaut :** $0,5$

logicWireLength

- *logicWireLength* : Longueur des pattes de raccordement (unité jps). **valeur par défaut :** $0,5$

log

a **log** $c \longrightarrow c = \log a$

log_seq

$a\ b$ **log_seq** $10^a\ 2.10^a\ 3.10^a \ldots 9.10^a\ 10^{a+1}\ 2.10^{a+1} \ldots 9.10^{a+1} \ldots 9.10^b \longrightarrow a$ et b sont des entiers. Génère une séquence pour une échelle logarithmique de graduations entre 10^a et 10^b

log_xbande

i **log_xbande** $- \longrightarrow i$ est un entier. trace 10 traits verticaux $\left]10^{i-1}; 10^{i}\right]$ en utilisant la commande **vrule**

log_xmark

n **log_xmark** $- \longrightarrow n$ est un entier. affiche la numérotation correspondant à 10^n sur l'axe Ox

log_ybande

i **log_ybande** $- \longrightarrow i$ est un entier. trace 10 traits horizontaux $\left]10^{i-1}; 10^{i}\right]$ en utilisant la commande **hrule**

log_ymark

n **log_ymark** $- \longrightarrow n$ est un entier. affiche la numérotation correspondant à 10^n sur l'axe Oy

loop

proc **loop** $- \longrightarrow$ exécute *proc* un nombre indéfini de fois

M

magenta

$-$ **magenta** $- \longrightarrow$ sélectionne la couleur magenta

mapnc

$[\ lit_0\ lit_1 \ldots lit_n\ ]\ [\ C_0\ C_1 \ldots C_n\ ]$ **mapnc** $- \longrightarrow$ associe, pour $0 \leqslant i \leqslant n$, le littéral lit_i et le cercle C_i dans le dictionnaire courant

mapnp

[lit_0 lit_1 ... lit_n] [A_0 A_1 ... A_n] **mapnp** $-$ $\longrightarrow$ associe, pour $0 \leqslant i \leqslant n$, le littéral lit_i et le point A_i dans le dictionnaire courant

mapnu

[lit_0 lit_1 ... lit_n] [a_0 a_1 ... a_n] **mapnu** $-$ $\longrightarrow$ associe, pour $0 \leqslant i \leqslant n$, le littéral lit_i et la valeur unaire a_i dans le dictionnaire courant

marked

A B n **marked** $-$ $\longrightarrow$ marque le segment $[AB]$ avec n traits inclinés

marks

$-$ **marks** $-$ $\longrightarrow$ inscrit toute la numérotation des axes Ox et Oy

marks

$-$ **marks** $-$ $\longrightarrow$ numérote les graduations sur les axes Ox et Oy

{masque-_

$-$ **masque-_** $-$ $\longrightarrow$ ajoute au chemin courant le rectangle de coin inférieur gauche le point $(xmin, ymin)$ et de coin supérieur droit $(xmax, ymax)$. Le chemin est parcouru dans le sens des aiguilles d'une montre

{masque_

$-$ **masque_** $-$ $\longrightarrow$ ajoute au chemin courant le rectangle de coin inférieur gauche le point $(xmin, ymin)$ et de coin supérieur droit $(xmax, ymax)$. Le chemin est parcouru dans le sens inverse des aiguilles d'une montre

max

a b **max** c $\longrightarrow$ c est le plus grand des deux nombres a et b

max

a b **max** c $\longrightarrow$ c est le plus grand des deux nombres a et b

Mayer

$array_1$ $array_2$ **Mayer** d $\longrightarrow$ la droite d est la droite de Mayer définie par les tableaux de réels $array_1$ et $array_2$ définissant respectivement les abscisses et les ordonnées d'un nuage de points

mediane

[a_0 ... a_n] **mediane** m $\longrightarrow$ le réel m est la médiane de la série des a_i.

mediatrice

A B **mediatrice** D $\longrightarrow$ D est la médiatrice du segment $[AB]$

methodetrapeze

a b $\{f\}$ n **methodetrapeze** $real$ $\longrightarrow$ $real$ est une approximation de l'intégrale de $f(x)$ entre a et b, calculée avec la méthode des trapèzes pour $n+1$ points (n est un entier)

mframe_

A L ℓ **mframe_** $-$ $\longrightarrow$ ajoute au chemin courant le rectangle dont le point A est le milieu du côté inférieur, de dimension horizontale L et de dimension verticale ℓ

mframe

A L ℓ **mframe** $-$ $\longrightarrow$ trace le rectangle dont le point A est le milieu du côté inférieur, de dimension horizontale L et de dimension verticale ℓ

milieu3d

A B **milieu3d** I $\longrightarrow$ I est le milieu de $[AB]$

milieu

A B **milieu** I $\longrightarrow$ le point I est le milieu du segment $[AB]$

Milne

a b $\{f\}$ x_0 y_0 h **Milne** x_{-n} y_{-n} ... x_{-1} y_{-1} x_0 y_0 x_1 y_1 ... x_n y_n $\longrightarrow$ dépose les points, calculés par la méthode de Milne, de la courbe sur $[a, b]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est un réel positif, il détermine le pas entre chaque point calculé

Milne

a b $\{f\}$ x_0 y_0 n **Milne** x_{-n} y_{-n} ... x_{-1} y_{-1} x_0 y_0 x_1 y_1 ... x_n y_n $\longrightarrow$ n étant un entier, cette procédure dépose les points, calculés par la méthode de Milne, de la courbe sur $[a, b]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est calculé en fonction de n.

milne

$\{f\}$ x_0 y_0 h **milne** x_{-n} y_{-n} $\ldots x_{-1}$ y_{-1} x_0 y_0 x_1 y_1 $\ldots x_n$ y_n $\longrightarrow$ dépose les points, calculés par la méthode de Milne, de la courbe sur $[xmin, xmax]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est un réel positif, il détermine le pas entre chaque point calculé

milne

$\{f\}$ x_0 y_0 n **milne** x_{-n} y_{-n} $\ldots x_{-1}$ y_{-1} x_0 y_0 x_1 y_1 $\ldots x_n$ y_n $\longrightarrow$ n étant un entier, cette procédure dépose les points, calculés par la méthode de Milne, de la courbe sur $[xmin, xmax]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est calculé en fonction de n.

min

a b **min** c $\longrightarrow$ c est le plus petit des deux nombres a et b

min

a b **min** c $\longrightarrow$ c est le plus petit des deux nombres a et b

mixte

$-$ **mixte** $-$ $\longrightarrow$ sélectionne le tracé de type trait mixte

mod

a b **mod** r $\longrightarrow$ r est reste de la division euclidienne de a par b

module

z **module** r $\longrightarrow$ le réel $r = |z|$

moveto

x y **moveto** $-$ $\longrightarrow$ définit le point courant à (x, y)

moyenne

[a_0 a_1 $\ldots$ a_n] **moyenne** m $\longrightarrow$ m est la moyenne arithmétique des nombres $a_0, a_1, \ldots, a_n$

moyenne

[$a_0 \ldots a_n$] **moyenne** m $\longrightarrow$ le réel m est la moyenne arithmétique de la série des a_i. $m = \left(\sum_{i=0}^{n} a_i\right) / (n+1)$.

mulc

z z' **mulc** Z $\longrightarrow$ $Z = zz'$ est le produit des complexes z et z'

mul

a b **mul** c $\longrightarrow$ $c = a \times b$

mulm

A B **mulm** M $\longrightarrow$ multiplie les matrices A et B et dépose le résultat sur la pile

mulv3d

$\vec{u}$ λ **mulv3d** $\vec{v}$ $\longrightarrow$ $\vec{v} = \lambda\vec{u}$

mulv

u a **mulv** $\vec{U}$ $\longrightarrow$ $\vec{U} = a\vec{u}$ où a est un nombre réel

N

ncangle

string$_1$ *string*$_2$ *option* **ncangle** $-$ $\longrightarrow$ Trace en B, et suivant l'angle *angleB* un bras de longueur *armB*, puis elle connecte ce bras en A, suivant l'angle *angleA* par un double segment à angle droit.

ncangles

string$_1$ *string*$_2$ *option* **ncangles** $-$ $\longrightarrow$ Trace en A (resp. B), et suivant l'angle *angleA* (resp. *angleB*) un bras de longueur *armA* (resp. *armB*), puis elle connecte les 2 bras par un double segment à angle droit (en partant de B).

ncarc

string$_1$ *string*$_2$ *option* **ncarc** $-$ $\longrightarrow$ Connecte les nodes avec une courbe de Bézier, en utilisant le paramètre *nodesep*. La courbe se connecte en A avec un angle *arcangleA* par rapport à la droite (AB) et se connecte en B avec un angle $-$*arcangleB* par rapport à la droite (AB).

ncbar

string$_1$ *string*$_2$ *option* **ncbar** $-$ $\longrightarrow$ Trace d'abord les bras de longueurs respectives *armA* et *armB* à un angle *angleA*. Ensuite, l'un des bras est étendu puis connecté, de telle façon que la ligne finale soit composée de 3 segments à angle droit.

nccurve

string$_1$ *string*$_2$ *option* **nccurve** $-$ $\longrightarrow$ Trace une courbe de Bézier entre les nodes A et B. Les paramètres *angleA* et *angleB* sont utilisés.

ncdiagg

string$_1$ *string*$_2$ *option* **ncdiagg** $- \longrightarrow$ Trace d'abord, en A et à l'angle *angleA*, le bras de longueur *armA*. Ensuite, ce bras est directement connecté au point B. Le paramètre *linearc* est utilisé pour arrondir les angles.

ncdiag

string$_1$ *string*$_2$ *option* **ncdiag** $- \longrightarrow$ Trace d'abord les bras de longueurs respectives *armA* et *armB* à des angles respectifs *angleA* et *angleB*. Ensuite, ces bras sont connecté par une ligne droite. Le paramètre *linearc* est utilisé pour arrondir les angles.

ncline

string$_1$ *string*$_2$ *option* **ncline** $- \longrightarrow$ Trace une simple ligne entre les nodes A et B. Seul le paramètre *nodesep* est utilisé.

neg

a **neg** $c \longrightarrow c = -a$

newmatrix

$m\ n$ **newmatrix** $M \longrightarrow$ dépose une nouvelle matrice nulle (m, n) sur la pile

newpath

$-$ **newpath** $- \longrightarrow$ initialise et vide le chemin courant

newton_solve

$x_0\ \varepsilon\ \varepsilon\ \{f\}\ \{f'\}$ **newton_solve** $x \longrightarrow \{f\}$ et $\{f'\}$ sont des exécutables, et f' désigne la fonction dérivée de f. Le réel x est obtenu par la méthode des tangentes de Newton, avec la valeur initiale x_0 et la tolérance ε

node

string **node** $- \longrightarrow$ Déclare un nœud rectangulaire dont le nom est défini par *string*

nodesep

- *nodesep* : espace en picas séparant le point de connexion et l'extrémité du trait de connexion. **valeur par défaut :** 3

noir

$-$ **noir** $- \longrightarrow$ sélectionne la couleur noir

nomme_noeud

tnode string **nomme_noeud** *tnode* $\longrightarrow$ Positionne à *string* le champ *nom* du nœud d'arbre *tnode*

non_relie

tnode **non_relie** *tnode* $\longrightarrow$ Positionne à *false* le champ adapté du nœud d'arbre *tnode*

normalize3d

$\vec{u}$ **normalize3d** $\vec{v} \longrightarrow$ Sinonyme de **unitaire3d** : si $\vec{u} = \vec{0}$, alors $\vec{v} = \vec{0}$, sinon $\vec{v} = \vec{u}/||\vec{u}||$

normal

u **normal** $v \longrightarrow$ le vecteur v vérifie $\vec{u} \cdot \vec{v} = 0$. Plus présisément, si $\vec{u}(a, b)$ alors $\vec{v}(-b, a)$.

norme3d

$\vec{u}$ **norme3d** $r \longrightarrow r$ est la norme du vecteur $\vec{u}$

norme

u **norme** $r \longrightarrow$ le réel $r = ||\vec{u}||$

nullc

z **nullc** *bool* $\longrightarrow$ le booléen *bool* vaut **true** si le complexe z est nul, **false** sinon.

O

O

$-$ **O** $O_x\ O_y \longrightarrow$ dépose sur la pile les coordonnées de l'origine du repère *jps*

orange

$-$ **orange** $- \longrightarrow$ sélectionne la couleur orange

ordonnepoints

$A\ B$ **ordonnepoints** $A'\ B' \longrightarrow$ range les points A et B par ordre d'ordonnée décroissante si possible, par ordre d'abscisse décroissante sinon

ordorig

D **ordorig** $b \longrightarrow b$ est l'ordonnée à l'origine de la droite D si celle-ci n'est pas verticale, erreur sinon

orthoproj

$A\ D$ **orthoproj** $A' \longrightarrow$ le point A' est le projeté orthogonal du point A sur la droite D

orthoprojplane3d

$M\ A\ \vec{v}$ **orthoprojplane3d** M' ⟶ Le point M' est le projeté du point M sur le plan P défini par le point A et le vecteur $\vec{v}$, normal à P.

ovalnode

string **ovalnode** – ⟶ Déclare un nœud ovale (elliptique) dont le nom est défini par *string*

OX

OX D ⟶ dépose la droite $D = Ox$ sur la pile

OY

OY D ⟶ dépose la droite $D = Oy$ sur la pile

P

pangle

$A\ B$ **pangle** α ⟶ α est l'angle en dégré défini par le vecteur $\overrightarrow{AB}$ dans le repère postscript

papply

$[\ A_0 \dots A_n\ ]\ f$ **papply** $[\ b_0 \dots b_n\ ]$ ou – ⟶ construit un nouveau tableau en répétant, pour i variant de 0 à n, l'opération suivante : déposer le point A_i puis exécuter f. Si à la fin de cette opération le tableau est vide, alors il est enlevé de la pile.

parallelopoint

$A\ B\ C$ **parallelopoint** D ⟶ le point D tel que $ABCD$ soit un parallélogramme

paral

$D\ A$ **paral** D' ⟶ D' est la droite parallèle à la droite D passant par le point A

pcangle

$A\ B$ *option* **pcangle** – ⟶ Comme **ncangle**

pcangles

$A\ B$ *option* **pcangles** – ⟶ Comme **ncangles**

pcarc

$A\ B$ *option* **pcarc** – ⟶ Comme **ncarc**

pcbar

$A\ B$ *option* **pcbar** – ⟶ Comme **ncbar**

pccurve

$A\ B$ *option* **pccurve** – ⟶ Comme **nccurve**

pcdiagg

$A\ B$ *option* **pcdiagg** – ⟶ Comme **ncdiagg**

pcdiag

$A\ B$ *option* **pcdiag** – ⟶ Comme **ncdiag**

pcline

$A\ B$ *option* **pcline** – ⟶ Comme **ncline**

perp

$D\ A$ **perp** D' ⟶ D' est la droite perpendiculaire à la droite D passant par le point A

pictdict

name **pictdict** $x\ y$ ⟶ dépose sur la pile les coordonnées associées au nom *name* dans le dictionnaire *Pictdict*

pi

– **pi** $3,141\,59$ ⟶ le nombre π

plot

$[\ A_0\ A_1 \dots A_n\ ]$ **plot** – ⟶ affiche les points A_i du tableau en utilisant la commande *dotstyle*

plot

$[\ A_0\ A_1 \dots A_n\ ]$ *proc* **plot** – ⟶ affiche les points A_i du tableau en utilisant la procédure *proc*

plus3d

A **plus3d** – ⟶ Analogue 3d de la commande **plus**

plus

A **plus** – ⟶ dessine une croix + au point A dans le repère *jps*

point3d

A **`point3d`** – ⟶ Analogue 3d de la commande **`point`**

`pointilles`

– **`pointilles`** – ⟶ sélectionne le tracé de type *tiret court*

`point`

A **`point`** – ⟶ dessine un point en A dans le repère jps

`point`

point **`point`** – ⟶ dessine le point spécifié

`points3d`

array **`points3d`** – ⟶ Analogue 3d de la commande **`points`**

`points`

`[` $point_1 \ldots point_n$ **`]`** **`points`** – ⟶ dessine les points spécifiés

`Poisson`

$x\ \lambda$ **`Poisson`** y ⟶ y est l'image de x par la loi de Poisson de paramètre λ

`pol2ell`

pol **`pol2ell`** *ell* ⟶ le polygône *pol* est constitué des 4 sommets de l'ellipse *ell*

`polygone*3d`

array **`polygone*3d`** – ⟶ Analogue 3d de la commande **`polygone*`**

`polygone3d`

array **`polygone3d`** – ⟶ Analogue 3d de la commande **`polygone`**

`polygone_`

array **`polygone_`** – ⟶ ajoute au chemin courant le polygône défini par le tableau de points *array*

`polygone`

array **`polygone`** – ⟶ trace le polygône défini par le tableau de points *array*

`popc`

cerc **`popc`** – ⟶ enlève le cercle au sommet de la pile

`popd`

D **`popd`** – ⟶ enlève la droite au sommet de la pile

`popp`

A **`popp`** – ⟶ enlève le point au sommet de la pile

`printmatrix`

$A\ x\ y\ M$ **`printmatrix`** – ⟶ Affiche la matrice M. Le coefficient a_{00} est affiché en A, et on utilise les décalages $(x, 0)$ et $0, y$ pour les autres coefficients

`projx`

A **`projx`** A' ⟶ le point A' est le projeté orthogonal du point A sur l'axe Ox

`projy`

A **`projy`** A' ⟶ le point A' est le projeté orthogonal du point A sur l'axe Oy

`ptojpoint`

$x\ y$ **`ptojpoint`** $X\ Y$ ⟶ Reçoit les coordonnées (x, y) dans le repère postscript et renvoie les coordonnées (X, Y) dans le repère jps

`put_ij`

$M\ i\ j$ *any* **`put_ij`** – ⟶ affecte le coefficient a_{ij} de la matrice M à *any*

`put_Li`

$M\ i\ L$ **`put_Li`** – ⟶ remplace dans la matrice M la ligne d'indice i par L

Q

`qplanxy`

– **`qplanxy`** – ⟶ Trace un quadrillage du plan XY

quadrillagegray

- *quadrillagegray* : niveau de gris pour le quadrillage tracé avec la commande **`quadrillage`**. **valeur par défaut :** $0,4$

`quadrillage`

– **`quadrillage`** – ⟶ Trace un quadrillage simple. Les paramètres sont *xstquadrillage*, *ystquadrillage*, *quadrillagegray* et *quadrillagewd*

`Quadrillage`

xs_0 ys_0 { *color* } **`Quadrillage`** – ⟶ Trace un quadrillage simple, de couleur *color*, avec les pas sur x et y définis par le couple (xs_0, ys_0). L'argument $\{color\}$ est optionnel. Avec cette syntaxe, l'épaisseur du trait est l'épaisseur courante.

`Quadrillage`

[xs_0 ys_0 xs_1 ys_1 xs_2 ys_2] { *color* } **`Quadrillage`** – ⟶ Trace un triple quadrillage, de couleur *color*, avec les pas sur x et y définis par les xs_i et ys_i. L'argument $\{color\}$ est optionnel, de même que les couples (xs_2, ys_2) et (xs_1, ys_1). Les épaisseurs de trait sont relevées dans le tableau *quadrillagewidth*

`Quadrillage`

[xs_0 ys_0 xs_1 ys_1 xs_2 ys_2] { *color* } **`Quadrillage`** – ⟶ Trace un triple quadrillage, de couleur *color*, avec les pas sur x et y définis par les xs_i et ys_i. L'argument $\{color\}$ est optionnel, de même que les couples (xs_2, ys_2) et (xs_1, ys_1). Les épaisseurs de trait sont relevées dans le tableau *quadrillagewidth*

quadrillagewd

- *quadrillagewd* : épaisseur du trait pour le quadrillage tracé avec la commande **`quadrillage`**. **valeur par défaut :** $0,25$

Quadrillagewidth

- *Quadrillagewidth* : tableau définissant les 3 épaisseurs de traits pour le quadrillage triple tracé par la commande **`Quadrillage`**. **valeur par défaut :** [0, 7 0, 4 0, 2]

`quadrilleXYZ`

xmin xmax ymin ymax zmin zmax **`quadrilleXYZ`** – ⟶ Effectue un quadrillage d'unité 1 sur le produit $[xmin; xmax] \times [ymin; ymax] \times [zmin; zmax]$

R

`rand`

– **`rand`** *int* ⟶ génère un entier au hasard

`rcurveto`

dx_1 dy_1 dx_2 dy_2 dx_3 dy_3 **`rcurveto`** – ⟶ **`curveto`** relatif

`rect`

A **`rect`** – ⟶ colorie en niveau de gris un rectangle dont une base est porté par l'axe *Ox*, et tel que le point *A* soit le milieu du côté opposé

`rect`

A **`rect`** – ⟶ dessine un rectangle dont une base est porté par l'axe *Ox*, et tel que le point *A* soit le milieu du côté opposé

`regxy`

$array_1$ $array_2$ **`regxy`** *d* ⟶ *d* est la droite de régression des x en y de la série double définie par les tableaux de réels $array_1$ et $array_2$

`regyx`

$array_1$ $array_2$ **`regyx`** *d* ⟶ *d* est la droite de régression des y en x de la série double définie par les tableaux de réels $array_1$ et $array_2$

`repeat`

int proc **`repeat`** – ⟶ exécute *proc int* fois

representationtype

- *representationtype* : Chaîne de caractère spécifiant le type de perspective : (perspective) ou (ortho). **valeur par défaut :** (perspective)

`reversepath`

– **`reversepath`** – ⟶ renverse la direction du chemin courant

`rframe_`

A L ℓ **`rframe_`** – ⟶ ajoute au chemin courant le rectangle dont le point *A* est le coin inférieur droit, de dimension horizontale *L* et de dimension verticale ℓ

`rframe`

A L ℓ **`rframe`** – ⟶ trace le rectangle dont le point *A* est le coin inférieur droit, de dimension horizontale *L* et de dimension verticale ℓ

`right`

– **right** $\vec{u}$ $\longrightarrow$ $\vec{u}$ est le vecteur $(1, 0)$

rlineto

dx dy **rlineto** – $\longrightarrow$ **lineto** relatif

rmoveto

dx dy **rmoveto** – $\longrightarrow$ **moveto** relatif

Rnode

string **Rnode** – $\longrightarrow$ Déclare un nœud rectangulaire avec un encadrement non dessiné et dont le nom est défini par *string*

rollp

n p **rollp** – $\longrightarrow$ considère la pile comme une file circulaire de *n* points, et la tourne de *p* crans

romberg

a b $\{f\}$ ε **romberg** *real* $\longrightarrow$ *real* est une approximation de l'intégrale de $f(x)$ entre *a* et *b*, calculée avec la méthode de Romberg pour une valeur de convergence fixée à ε

rootnode

x y tnode **rootnode** *tnode* $\longrightarrow$ Dépose les coordonnées (x, y) dans le champ adapté du nœud d'arbre *tnode*, et le nomme *A* si celui-ci n'a pas de nom.

rose

– **rose** – $\longrightarrow$ sélectionne la couleur rose

rotatecercle

cerc I α **rotatecercle** *cerc'* $\longrightarrow$ le cercle *cerc'* est l'image du cercle *cerc* par la rotation de centre *I* et d'angle α

rotatedroite

D I α **rotatedroite** *D'* $\longrightarrow$ la droite *D'* est l'image de la droite *D* par la rotation de centre *I* et d'angle α

rotateell

ell I α **rotateell** *ell'* $\longrightarrow$ l'ellipse *ell'* est l'image de l'ellipse *ell* par la rotation de centre *I* et d'angle α

rotatepoint

A I α **rotatepoint** *A'* $\longrightarrow$ le point *A'* est l'image du point *A* par la rotation de centre *I* et d'angle α

rotatepol

pol I α **rotatepol** *pol'* $\longrightarrow$ le polygône *pol'* est l'image du polygône *pol* par la rotation de centre *I* et d'angle α

rouge

– **rouge** – $\longrightarrow$ sélectionne la couleur rouge

round

num_1 **round** num_2 $\longrightarrow$ arrondit num_1 à l'entier le plus proche

\#rpn\#

#rpn# $expr_1$ $\longrightarrow$ $expr_2$: $expr_2$ est l'écriture en notation polonaise inverse de l'expression en notation cartésienne $expr_1$

rptojpoint

x y **rptojpoint** *X Y* $\longrightarrow$ Reçoit les coordonnées (x, y) dans le repère BB (*real postscript*) et renvoie les coordonnées (X, Y) dans le repère *jps*

Rungekutta

a b $\{f\}$ x_0 y_0 *h* **Rungekutta** $x_{-n}\ y_{-n} \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1 \ldots x_n\ y_n$ $\longrightarrow$ dépose les points, calculés par la méthode de Runge-Kutta, de la courbe sur $[a, b]$ de la fonction *s* solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre *h* est un réel positif, il détermine le pas entre chaque point calculé

Rungekutta

a b $\{f\}$ x_0 y_0 *n* **Rungekutta** $x_{-n}\ y_{-n} \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1 \ldots x_n\ y_n$ $\longrightarrow$ *n* étant un entier, cette procédure dépose les points, calculés par la méthode de Runge-Kutta, de la courbe sur $[a, b]$ de la fonction *s* solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre *h* est calculé en fonction de *n*.

rungekutta

$\{f\}$ x_0 y_0 *h* **rungekutta** $x_{-n}\ y_{-n} \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1 \ldots x_n\ y_n$ $\longrightarrow$ dépose les points, calculés par la méthode de Runge-Kutta, de la courbe sur $[xmin, xmax]$ de la fonction *s* solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre *h* est un réel positif, il détermine le pas entre chaque point calculé

rungekutta

$\{f\}\ x_0\ y_0\ n$ **rungekutta** $x_{-n}\ y_{-n}\ \ldots x_{-1}\ y_{-1}\ x_0\ y_0\ x_1\ y_1\ \ldots x_n\ y_n \longrightarrow n$ étant un entier, cette procédure dépose les points, calculés par la méthode de Runge-Kutta, de la courbe sur $[xmin, xmax]$ de la fonction s solution de l'équation différentielle $y' = f(x, y)$ vérifiant $s(x_0) = y_0$. Le nombre h est calculé en fonction de n.

S

sarc

$x\ y\ r\ ang_1\ ang_2$ **sarc** $- \longrightarrow$ ajoute un arc dans le sens contraire des aiguilles d'une montre (coordonnées dans le repère jps)

sarcn

$x\ y\ r\ ang_1\ ang_2$ **sarcn** $- \longrightarrow$ ajoute un arc dans le sens des aiguilles d'une montre (coordonnées dans le repère jps)

scalprod3d

$\vec{u}\ \vec{v}$ **scalprod3d** $s \longrightarrow$ Produit scalaire : $s = \vec{u} \cdot \vec{v}$

scalprod

$\vec{u}\ \vec{v}$ **scalprod** $\vec{u} \cdot \vec{v} \longrightarrow$ Le produit scalaire de $\vec{u}$ par $\vec{v}$

ScreenDist

- *ScreenDist* : Distance par rapport à l'écran. **valeur par défaut :** 0.1

scurveto

$x_1\ y_1\ x_2\ y_2\ x_3\ y_3$ **scurveto** $- \longrightarrow$ ajoute une section cubique de Bézier (coordonnées dans le repère jps)

setangle_repere

α **setangle_repere** $- \longrightarrow$ Instruction destinée au script $jps2ps$. Elle indique que l'angle entre les axes Ox et Oy est de α degrés. Attention, α doit être lisible en clair par le script

setborder

b **setborder** $- \longrightarrow$ Indique, en points postripts, a taille de la bordure entourant l'image. Cette instruction est interprétée par le script $jps2ps$ pour déterminer la BoundingBox.

SetCamPos

$x\ y\ z$ **SetCamPos** $- \longrightarrow$ Positionne la caméra au point (x, y, z)

SetCamUp

$U_x\ U_y\ U_z$ **SetCamUp** $- \longrightarrow$ *Set Camera Up vector.*

SetCamVec

$V_x\ V_y\ V_z$ **SetCamVec** $- \longrightarrow$ *Set Camera Looking vector.*

setcmykcolor

cyan magenta yellow black **setcmykcolor** $- \longrightarrow$ définit la couleur CMYK. Les nombres *red*, *green* et *blue* sont des réels compris entre 0 et 1

setCourierBoldItalic

$-$ **setCourierBoldItalic** $- \longrightarrow$ sélectionne la police CourierBoldItalic

setCourierBold

$-$ **setCourierBold** $- \longrightarrow$ sélectionne la police CourierBold

setCourierItalic

$-$ **setCourierItalic** $- \longrightarrow$ sélectionne la police CourierItalic

setCourier

$-$ **setCourier** $- \longrightarrow$ sélectionne la police Courier

setellipseangle

α **setellipseangle** $- \longrightarrow$ affecte la valeur α au paramètre *ellipseangle*

setfontsize

n **setfontsize** $- \longrightarrow$ affecte la valeur n au paramètre $fontsize$

setformat

k **setformat** $- \longrightarrow$ Instruction destinée au script $jps2ps$. Elle indique que k est le rapport entre largeur et hauteur de l'image. Attention, k doit être lisible en clair par le script

setframeangle

α **setframeangle** $- \longrightarrow$ affecte la valeur α au paramètre $frameangle$

setheight

ℓ **setheight** $-\longrightarrow$ Affecte la valeurs ℓ à la variable *height* (taille verticale, en points poscripts, de la fenêtre de dessin). La première occurence de cette instruction dans le fichier *jps* est interprétée par le script pour déterminer la BoundingBox.

setmkstep

t_1 t_2 **setmkstep** $-\longrightarrow$ définit respectivement les pas t_1 et t_2 pour la numérotation sur les axes *Ox* et *Oy*

setorigine

x y **setorigine** $-\longrightarrow$ affecte les coordonnées (x, y) au point origine du repère *jps*

setPalatinoBoldItalic

$-$ **setPalatinoBoldItalic** $-\longrightarrow$ sélectionne la police PalatinoBoldItalic

setPalatinoBold

$-$ **setPalatinoBold** $-\longrightarrow$ sélectionne la police PalatinoBold

setPalatinoItalic

$-$ **setPalatinoItalic** $-\longrightarrow$ sélectionne la police PalatinoItalic

setPalatino

$-$ **setPalatino** $-\longrightarrow$ sélectionne la police Palatino

setquadrillagegray

g **setquadrillagegray** $-\longrightarrow$ affecte la valeur g au paramètre *quadrillagegray*. On doit avoir $0 \leqslant g \leqslant 1$

setresolution

n **setresolution** $-\longrightarrow$ affecte l'entier n à la variable *resolution* qui contrôle le nombre de points de calculs pour la représentation d'une courbe de fonction numérique

setrgbcolor

red green blue **setrgbcolor** $-\longrightarrow$ définit la couleur RGB. Les nombres *red*, *green* et *blue* sont des réels compris entre 0 et 1

setrotate

α **setrotate** $-\longrightarrow$ Instruction destinée au script *jps2ps*. Elle indique que l'image finale devra subir une rotation d'angle α (en degrés). Le calcul de la BoundingBox tient compte de cette rotation. Attention, α doit être lisible en clair par le script

setsubtkstep

t_1 t_2 **setsubtkstep** $-\longrightarrow$ définit respectivement les pas t_1 et t_2 pour les sous-tirets sur les axes *Ox* et *Oy*

setSymbolBoldItalic

$-$ **setSymbolBoldItalic** $-\longrightarrow$ sélectionne la police SymbolBoldItalic

setSymbolBold

$-$ **setSymbolBold** $-\longrightarrow$ sélectionne la police SymbolBold

setSymbolItalic

$-$ **setSymbolItalic** $-\longrightarrow$ sélectionne la police SymbolItalic

setSymbol

$-$ **setSymbol** $-\longrightarrow$ sélectionne la police Symbol

settailletangente

x **settailletangente** $-\longrightarrow$ affecte la valeur réelle a à la variable *tailletangente* qui gère la taille des tangentes tracées par **tangente**. La taille est exprimée en unités de l'axe *Ox*

setTimesBoldItalic

$-$ **setTimesBoldItalic** $-\longrightarrow$ sélectionne la police TimesBoldItalic

setTimesBold

$-$ **setTimesBold** $-\longrightarrow$ sélectionne la police TimesBold

setTimesItalic

$-$ **setTimesItalic** $-\longrightarrow$ sélectionne la police TimesItalic

setTimes

$-$ **setTimes** $-\longrightarrow$ sélectionne la police Times

settkstep

t_1 t_2 **settkstep** $-\longrightarrow$ définit respectivement les pas t_1 et t_2 pour les tirets sur les axes *Ox* et *Oy*

settrange

$a\ b$ **settrange** $- \longrightarrow$ affecte respectivement les valeurs réelles a et b aux variables $tmin$ et $tmax$

settvar

a **settvar** $- \longrightarrow$ affecte la valeur réelle a à la variable t

setwidth

L **setwidth** $- \longrightarrow$ Affecte la valeurs L à la variable $width$ (taille horizontale, en points postscript, de la fenêtre de dessin). La première occurence de cette instruction dans le fichier jps est interprétée par le script pour déterminer la BoundingBox.

setxmkstep

t **setxmkstep** $- \longrightarrow$ définit le pas pour la numérotation sur l'axe Ox

setxrange

$x_1\ x_2$ **setxrange** $- \longrightarrow$ Affecte respectivement les valeurs x_1 et x_2 à $xmin$ et $xmax$ (amplitude horizontale de la fenêtre de dessin). La première occurence de cette instruction dans le fichier jps est interprétée par le script pour déterminer la BoundingBox.

setxstquadrillage

p **setxstquadrillage** $- \longrightarrow$ affecte la valeur p au paramètre $xstquadrillage$

setxsubtkstep

t **setxsubtkstep** $- \longrightarrow$ définit le pas pour les sous-tirets sur l'axe Ox

setxtkstep

t **setxtkstep** $- \longrightarrow$ définit le pas pour les tirets sur l'axe Ox

setxunit

u **setxunit** $- \longrightarrow$ Spécifie, en nombre de points postscripts par unité, l'échelle sur l'axe Ox. La première occurence de cette instruction dans le fichier jps est interprétée par le script pour déterminer la BoundingBox.

setxvar

a **setxvar** $- \longrightarrow$ affecte la valeur réelle a à la variable x

setxyrapport

α **setxyrapport** $- \longrightarrow$ Instruction destinée au script $jps2ps$. Elle indique que le rapport entre l'unité sur Ox et l'unité sur Oy est α. Attention, α doit être lisible en clair par le script

setymkstep

t **setymkstep** $- \longrightarrow$ définit le pas pour la numérotation sur l'axe Oy

setyrange

$y_1\ y_2$ **setyrange** $- \longrightarrow$ Affecte respectivement les valeurs y_1 et y_2 à $ymin$ et $ymax$ (amplitude verticale de la fenêtre de dessin). La première occurence de cette instruction dans le fichier jps est interprétée par le script pour déterminer la BoundingBox.

setystquadrillage

p **setystquadrillage** $- \longrightarrow$ affecte la valeur p au paramètre $ystquadrillage$

setysubtkstep

t **setysubtkstep** $- \longrightarrow$ définit le pas pour les sous-tirets sur l'axe Oy

setytkstep

t **setytkstep** $- \longrightarrow$ définit le pas pour les tirets sur l'axe Oy

setyunit

v **setyunit** $- \longrightarrow$ Spécifie, en nombre de points postscripts par unité, l'échelle sur l'axe Oy. La première occurence de cette instruction dans le fichier jps est interprétée par le script pour déterminer la BoundingBox.

simpson

$a\ b\ \{f\}\ n$ **simpson** $real \longrightarrow real$ est une approximation de l'intégrale de $f(x)$ entre a et b, calculée avec la méthode de Simpson pour $n+1$ points (n est un entier pair)

sinh

a **sinh** $c \longrightarrow c = \operatorname{sh} a$

sin

a **sin** $c \longrightarrow c = \sin a$ (a en degré)

Sin

a **Sin** $c \longrightarrow c = \sin a$ (a en radian)

slineto

$x\ y$ **slineto** $- \longrightarrow$ ajoute une ligne droite jusqu'en (x, y) dans le repère jps

smoveto

$x\ y$ **smoveto** $- \longrightarrow$ définit le point courant à (x, y) dans le repère jps

smulm

$M\ \alpha$ **smulm** $M' \longrightarrow M'$ est la matrice produit de la matrice M par le scalaire α

solve2nddegre

$a\ b\ c$ **solve2nddegre** $x_1\ x_2 \longrightarrow$ Les réels x_1 et x_2 sont les racines réelles de l'équation $ax^2 + bx + c = 0$, où $a \neq 0$ et où $b^2 - 4ac \geqslant 0$

solve_syst

$B\ A$ **solve_syst** $X \longrightarrow A$ est une matrice carrée de déterminant non nul, B est un vecteur colonne, et X est le vecteur colonne solution de l'équation matricielle $AX = B$

solve_syst

$B\ A$ **solve_syst** $X \longrightarrow A$ est une matrice carrée de déterminant non nul et X est l'unique vecteur solution de l'équation $AX = B$

solve_trig

$B\ A$ **solve_trig** $X \longrightarrow A$ est une matrice triangulaire supérieure et X est l'unique vecteur solution de l'équation $AX = B$

sqrt

a **sqrt** $c \longrightarrow c = \sqrt{a}$

square

A **square** $- \longrightarrow$ dessine un carré au point A dans le repère jps

srcurveto

$dx_1\ dy_1\ dx_2\ dy_2\ dx_3\ dy_3$ **srcurveto** $- \longrightarrow$ **scurveto** relatif

srlineto

$dx\ dy$ **srlineto** $- \longrightarrow$ **slineto** relatif

srmoveto

$dx\ dy$ **srmoveto** $- \longrightarrow$ **smoveto** relatif

subc

$z\ z'$ **subc** $Z \longrightarrow Z = z - z'$ est la différence des complexes z et z'

sub

$a\ b$ **sub** $c \longrightarrow c = a - c$

subticks

$-$ **subticks** $- \longrightarrow$ trace tous les sous-tirets des axes Ox et Oy

subv3d

$\vec{u}\ \vec{v}$ **subv3d** $\vec{w} \longrightarrow \vec{w} = \vec{u} + \vec{v}$

subv

$u\ u'$ **subv** $\vec{U} \longrightarrow \vec{U} = \vec{u} - \vec{u}'$ est la différence des vecteurs $\vec{u}$ et $\vec{u}'$

sum

$[\ a_0 \dots a_n\ ]$ **sum** $s \longrightarrow$ le réel s est la somme $s = \sum_{i=0}^{n} a_i$

surfaceparam3d

$xmin\ pas_x\ xmax\ ymin\ pas_y\ ymax\ f$ **surfaceparam3d** $- \longrightarrow$ Dessine la surface $f(x, y) = z$ sur $[xmin; xmax] \times [ymin; ymax]$. f est un éxécutable.

symcercle

$cerc\ I$ **symcercle** $cerc' \longrightarrow$ le cercle $cerc'$ est le symétrique du cercle $cerc$ par rapport au point I

symell

$ell\ I$ **symell** $ell' \longrightarrow$ l'ellipse ell' est la symétrique de l'ellipse ell par rapport au point I

sympoint

$A\ I$ **sympoint** $A' \longrightarrow$ le point A' est le symétrique du point A par rapport au point I

sympol

pol I **sympol** *pol'* ⟶ le polygône *pol'* est le symétrique du polygône *pol* par rapport au point *I*

T

tab3dto2d

*array*1 **tab3dto2d** *array*2 ⟶ transforme un tableau de points 3d en tableau de points 2d

tailletangente

- *tailletangente* : la taille, exprimée en unités de l'axe *Ox*, des tangentes tracées par **tangente**.. **valeur par défaut :** 1

tangente

x string **tangente** – ⟶ trace la tangente à la courbe de la fonction f au point d'abscisse x, où f est l'exécutable désigné par la chaîne de caractères *string*. Attention, le calcul utilise l'exécutable f' dont le nom est obtenu en adjoignant à la chaîne *string* le caractère `'`. L'exécutable f' doit donc être défini.

tanh

a **tanh** $c \longrightarrow c = \operatorname{th} a$

tan

a **tan** $c \longrightarrow c = \tan a$ (a en degré)

Tan

a **Tan** $c \longrightarrow c = \tan a$ (a en radian)

Tbc

string/*lit* **Tbc** *tnode* ⟶ Construit un nœud d'arbre rectangulaire encadré dont le contenu, centré, est soit *string*, soit le label TEX défini par *lit*

Tb

string/*lit* **Tb** *tnode* ⟶ Construit un nœud d'arbre rectangulaire encadré dont le contenu est soit *string*, soit le label TEX défini par *lit*

TCc

string/*lit* **TCc** *tnode* ⟶ Construit un nœud d'arbre circulaire de rayon fixe *Circleradius* dont le contenu, centré, est soit *string*, soit le label TEX défini par *lit*

Tcc

string/*lit* **Tcc** *tnode* ⟶ Construit un nœud d'arbre circulaire dont le contenu, centré, est soit *string*, soit le label TEX défini par *lit*

TC

string/*lit* **TC** *tnode* ⟶ Construit un nœud d'arbre circulaire de rayon fixe *Circleradius* dont le contenu est soit *string*, soit le label TEX défini par *lit*

Tc

string/*lit* **Tc** *tnode* ⟶ Construit un nœud d'arbre circulaire dont le contenu est soit *string*, soit le label TEX défini par *lit*

Tdiac

string/*lit* **Tdiac** *tnode* ⟶ Construit un nœud d'arbre en forme de losange et dont le contenu, centré, est soit *string*, soit le label TEX défini par *lit*

Tdia

string/*lit* **Tdia** *tnode* ⟶ Construit un nœud d'arbre en forme de losange et dont le contenu est soit *string*, soit le label TEX défini par *lit*

\#tex\#

#tex# *expr* ⟶ – : compile l'expression *expr* avec TEX et affecte le résultat pour l'utilisation des commandes *labeltex*

Tf

– **Tf** *tnode* ⟶ Construit un nœud d'arbre fantôme (contenu vide et non relié au nœud père)

ticks

– **ticks** – ⟶ trace tous les tirets des axes *Ox* et *Oy*

times

A **times** – ⟶ dessine une croix au point *A* dans le repère *jps*

tnparametres

– **usecolor** – ⟶ charge le package *color*

uselabo

– **uselabo** – ⟶ charge le package *labo*

V

vadjust

- *vadjust* : taille, en points postscript, du décalage vertical appliqué, s'il y a lieu par les commandes de positionnement de l'environnement 'picture' . **valeur par défaut :** 3,75

variance

$[\, a_0 \dots a_n \,]$ **variance** $v \longrightarrow$ le réel v est la variance de la série des a_i.

vecteur3d

$A\ B$ **vecteur3d** $u \longrightarrow u = \overrightarrow{AB}$

vecteur

$A\ B$ **vecteur** $\vec{u} \longrightarrow A$ et B sont des points, et $\vec{u} = \overrightarrow{AB}$

vectprod3d

$\vec{u}\ \vec{v}$ **vectprod3d** $\vec{w} \longrightarrow \vec{w} = \vec{u} \wedge \vec{v}$

verticale?

D **verticale?** *bool* ⟶ vrai si la droite D est verticale, faux sinon

vert

– **vert** – ⟶ sélectionne la couleur vert

view_square_matrix

M **view_square_matrix** $a_{1,1} \dots a_{1,n}$ **()** $a_{1,1} \dots a_{1,n}$ **()** $\dots$ **()** $a_{n,1} \dots a_{n,n}$ ⟶ dépose sur la pile les coefficients de la matrice carrée M

W

wedge_

$\alpha\ \beta\ A\ r$ **wedge_** – ⟶ ajoute au chemin courant la portion de camembert de centre A, de rayon r, délimité par les angles α et β

wedge

$\alpha\ \beta\ A\ r$ **wedge** – ⟶ trace la portion de camembert de centre A, de rayon r, délimité par les angles α et β

widthangledroit

- *widthangledroit* : taille, en points postscript, d'un des côté de l'angle droit tracé par **angledroit**. **valeur par défaut :** 5

withcontrolpoints

– **withcontrolpoints** – ⟶ active le tracé des points de contrôle lors du dessin d'une courbe de Bézier par **draw** ou **bezier_curve**

withoutcontrolpoints

– **withoutcontrolpoints** – ⟶ désactive le tracé des points de contrôle lors du dessin d'une courbe de Bézier par **draw** ou **bezier_curve**

X

xdpoint

$x\ D$ **xdpoint** $A \longrightarrow$ si la droite D n'est pas verticale, alors A est le point de D d'abscisse x. Erreur sinon

xmark

x **xmark** – ⟶ inscrit la numérotation au point d'abscisse x de l'axe Ox

xmarks

– **xmarks** – ⟶ inscrit toute la numérotation de l'axe Ox

xmarkstyle

string A **xmarkstyle** – ⟶ Procédure utilisée par les commandes **xmark** et dérivées pour inscrire la chaîne *string* au point A

xmax

- *xmax* : borne supérieure sur l'axe Ox. **valeur par défaut :** 5

xmin

- *xmin* : borne inférieure sur l'axe Ox. **valeur par défaut :** -5

xstquadrillage

- *xstquadrillage* : pas sur l'axe Ox pour le quadrillage tracé avec la commande **quadrillage**. **valeur par défaut :** 1