Macros de géométrie plane avec `Asymptote`

D. Comin

6 février 2009

Table des matières

1 Les constantes.

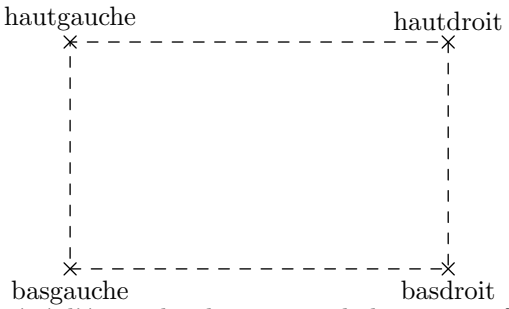
1.1 Les dimensions.

Par défaut, l'unité est le cm, ce qui est cohérent pour faire de la géométrie plane. L'usage de la commande `size` perturbe les unités et la figure tracée n'est plus en vraie grandeur ...

La figure est plus ou moins contenue dans un chemin fermé appelé `cadre` dont les coordonnées des sommets sont récupérables par les quatre «pairs» suivants : `hautgauche, hautdroit, basgauche, basdroit`.

La commande ▶ `figure(pair basgauche,pair hautdroit)` permet de fixer le cadre. Il n'est, par défaut, pas tracé.

```
import geoplane;
 figure((0,0),(5,3),black+dashed);
\\voir plus bas pour la macro nomme
nomme("hautgauche",hautgauche,nord);
nomme("hautdroit",hautdroit,nord);
nomme("basdroit",basdroit,sud);
nomme("basgauche",basgauche,sud);
```

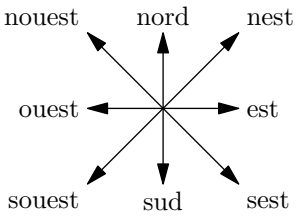


La constante `croix` définit la taille des points tracés à l'écran, les dimensions de la croix en fait.

1.2 Les directions

`Asymptote` connaît les huit principaux points cardinaux comme variables de type «pair» : N pour le nord, S pour le sud,etc. Ces directions sont pratiques pour nommés les points mais interdisent d'utiliser certaines lettres : N,S,W,E...Ces directions sont renommées comme suit.

```
 import geoplane;
figure((-2,-2),(2,2));
draw((0,0)--nord,Arrow);
label("nord",nord,nord);
draw((0,0)--sud,Arrow);
label("sud",sud,sud);
draw((0,0)--est,Arrow);
label("est",est,est);
draw((0,0)--ouest,Arrow);
label("ouest",ouest,ouest);
draw((0,0)--nest,Arrow);
label("nest",nest,nest);
draw((0,0)--sest,Arrow);
label("sest",sest,sest);
draw((0,0)--nouest,Arrow);
label("nouest",nouest,nouest);
draw((0,0)--souest,Arrow);
label("souest",souest,souest);
```

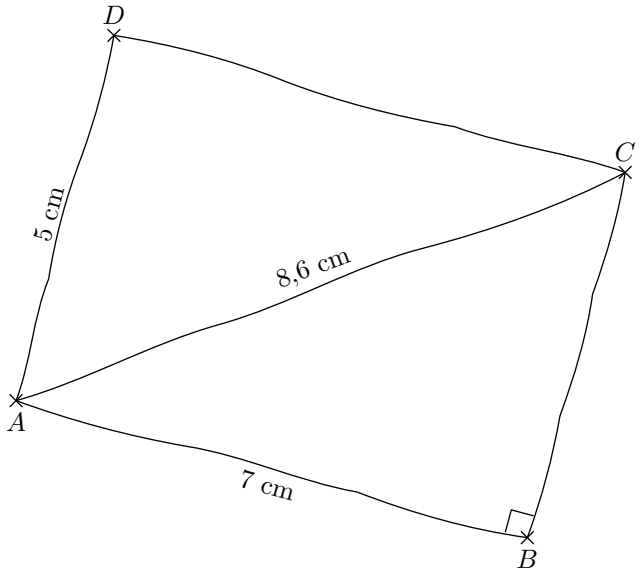


1.3 Le tracé à «main levée»

Il suffit de mettre la constante `mainlevee` de type `bool` à la valeur `true` pour avoir un dessin qui ressemble à un croquis à main levée.

```
import geoplane;
//dessin à main levée
mainlevee=true;
pair A=(0,0);
path tt=rectangle(A,7,5,angle=-15);
draw(tt);
nomme("$A$",sommet(tt,0),sud);
nomme("$B$",sommet(tt,1),sud);
nomme("$C$",sommet(tt,2),nord);
nomme("$D$",sommet(tt,3),nord);
```

```
draw(segment(sommet(tt,0),sommet(tt,2)));
etiquette(sommet(tt,0),sommet(tt,2),"8,6 cm");
etiquette(sommet(tt,0),sommet(tt,3),"5 cm");
etiquette(sommet(tt,0),sommet(tt,1),"7 cm",dessus=false);
angledroit(sommet(tt,0),sommet(tt,1),sommet(tt,2));
```



2 Les points

2.1 Tracé

▶ `pointe(pair A,pen p=currentpen)` trace une croix en A sans nommer le point.

▶ `nomme(Label L, pair position,pen p=currentpen)` trace le point sur le pair «position» et place le texte contenu dans le «label» L.

```
import geoplane;
figure((-2,-2),(2,2));
pair A,B,C;
A=(-1.5,1.5);
B=(-1,0);
C=(1,1.5);
pointe(A);
nomme("et hop",B,est,blue);
nomme("$C$",C,sud);
```

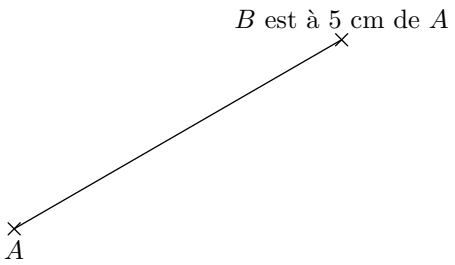
On notera la présence des directions est,sud,... qui permettent de placer l'étiquette de texte : "et hop" est à l'est de la position B.

2.2 Construction

▶ `pointdistant(pair A,real distance, real angle)` crée un «pair» situé à une certaine distance de A, (AB) faisant un angle donné avec l'horizontale.

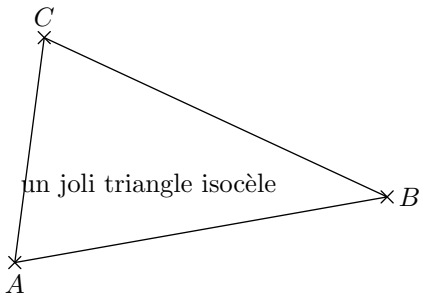
▶ `point_angle_dist(pair A, pair B, real angle, real distance)` renvoie M un point à *distance* de A et tel que $\widehat{MAB} = angle^\circ$

```
import geoplane;
figure((-1,-1),(5,5));
pair A,B;
A=(0,0);
B=pointdistant(A,5,30);
nomme("$A$",A,sud);
nomme("$B$ est \‘a 5 cm de $A$",B,nord);
draw(A--B);
```



► `compas(pair A,pair B, real a, real b)` crée un «pair» situé à distances données de A et B.

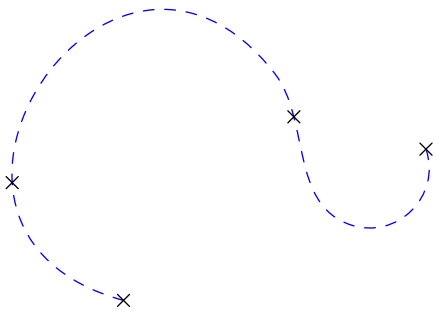
```
import geoplane;
figure((-1,-1),(5,5));
pair A,B,C;
A=(0,0);
B=pointdistant(A,5,10);
C=compas(A,B,3,5);
nomme("$A$",A,sud);
nomme("$B$",B,est);
nomme("$C$",C,nord);
draw(A--B--C--cycle);
label("un joli triangle isoc\‘ele",(A+B+C)/3,sud);
```



► `milieu(pair A, pair B)` ,bon là, ça va aller.

► `pointsur(path chemin, real r)` renvoie un «pair» situé sur la chemin en fonction de $r \in [0..1]$: 0 correspond à l'origine et 1 à l'extrémité.

```
import geoplane;
figure((-0.5,-0.5),(4,5));
path c=(0,0)..(2,3)..(3,1)..(4,2);
draw(c,blue+dashed);
pointe(pointsur(c,0));
pointe(pointsur(c,0.2));
pointe(pointsur(c,0.7));
pointe(pointsur(c,0),1);
```



2.3 Points particuliers

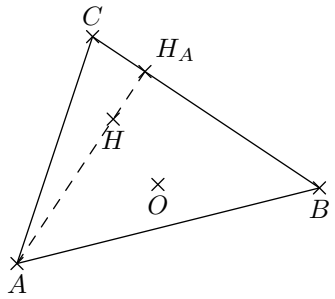
► `orthocentre(pair C, pair A, pair B)` renvoie l'orthocentre du triangle ABC.

► `circonscrit(pair C, pair A, pair B)` renvoie le centre du cercle circonscrit à ABC.

► `projortho(pair M, pair A,pair B)` renvoie le projeté orthogonal de M sur (AB).

► `inscritpair A, pair B, pair C` renvoie le centre du cercle inscrit dans ABC.

```
import geoplane;
figure((-0.5,-0.5),(4,5));
pair A,B,C,O,H,Ha;
A=(0,0);
B=(4,1);
C=(1,3);
draw(A--B--C--cycle);
nomme("$A$",A,sud);
nomme("$B$",B,sud);
nomme("$C$",C,nord);
H=orthocentre(A,B,C);
nomme("$H$",H,sud);
O=circonscrit(A,B,C);
nomme("$O$",O,sud);
Ha=projortho(A,B,C);
draw(Ha--A,dashed);
nomme("$H_A$",Ha,nest);
```



3 Mesure et codage

3.1 Cotation

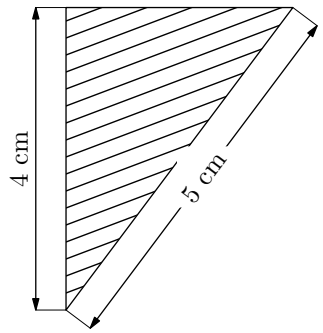
▶ `cotemilieu(pair A,pair B, string texte, real d,bool trait=false,pen sty=black)` trace une flèche de A à B à d mm au dessus de (AB), le texte est au milieu.

▶ `cote(pair A,pair B, string texte, real d,bool trait=false,pen sty=black)` trace une flèche de A à B à d mm au dessus de (AB), le texte est au dessus.

▶ `etiquette(pair A, pair B, string txt,bool dessus=true,pen sty=currentpen)` place le texte txt le long de [AB].

▶ `hachurage(path p,real espace, real angle, pen pen=currentpen)` remplit avec des hachures espacées de "espace" mm, avec un angle de "angle" ° le chemin fermé p.

```
import geoplane;
figure((-0.5,-0.5),(4,5));
pair A,B,C;
A=(0,0);
B=(0,4);
C=(3,4);
draw(A--B--C--cycle);
cote(A,B,"4 cm",4,trait=true);
cotemilieu(A,C,"5 cm",-2,trait=true);
hachurage(A--B--C--cycle,2mm,20);
```



3.2 Codage des longeurs et des angles

▶ `code(pen sty=invisible,int trait ...pair[] K)` code une série de segments dont les extrémités sont contenues dans une matrice de type «pair[]» mais il suffit d'ecrire `code(2,A,B,B,C,D,E)` pour coder les segments $[AB]$, $[BC]$ et $[DE]$ de deux traits. Le paramètre "trait" précise le codage :

Si "trait" vaut 1, 2 ou 3 le segment est codé par des traits ...

Si "trait" vaut 4 le segment est codé par un tilde.

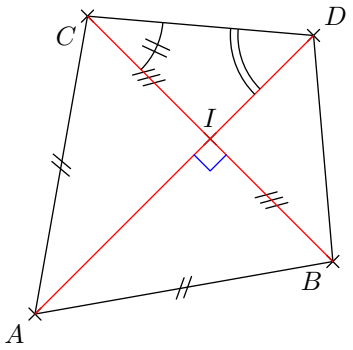
Si "trait" vaut 5 le segment est codé par un cercle.

▶ `codemilieu(pair A, pair B, int trait)` est similaire à `code` mais plus simple syntaxiquement.

▶ `codeangle(pair A,pair B, pair C, int trait,int nbarc=1)` marque l'angle $\widehat{ABC}$ par des traits et un ou plusieurs arcs de cercle.

▶ `angledroit(pair A,pair B,pair C,real taille=3mm, pen p=black)` code l'angle $\widehat{ABC}$, supposé droit.

```
import geoplane;
figure((-0.5,-0.5),(4,5));
pair A,B,C,D,I;
A=(0,0);
B=pointdistant(A,4,10);
C=pointdistant(A,4,80);
D=compas(C,B,3,3);
I=extension(C,B,A,D);
draw(A--B--D--C--cycle);
nomme("$A$",A,souest);
nomme("$B$",B,souest);
nomme("$C$",C,souest);
nomme("$D$",D,nest);
nomme("$I$",I,nord);
code(2,C,A,A,B);
codemilieu(C,B,3);
draw(A--D,red);
draw(B--C,red);
codeangle(B,C,D,2);
codeangle(C,D,I,0,2);
angledroit(A,I,B,blue);
```



4 Quadrillages

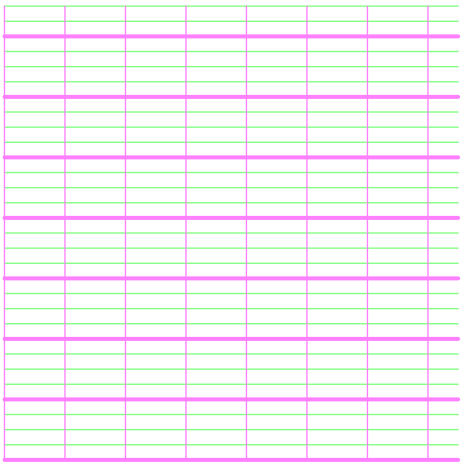
4.1 les papiers

▶ `millimetre(pen sty=orange)` trace du papier millimétré dans les limites du `cadre`. La couleur par defaut est "orange".

▶ `carreau(real cote=0.5, pen sty=orange)` trace un quadrillage 5 mm × 5mm par défaut.

▶ `seyes()` trace un morceau de cahier d'écolier.

```
import geoplane;
figure((0,0),(6,6));
seyes();
```



4.2 les fractions

Les commandes suivantes tracent des fractions à partir d'un quadrillage en coloriant un nombre n de parts.

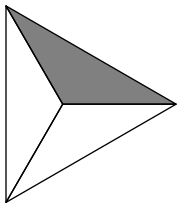
▶ `void fractionRect(int l,int h, int n=0, pen remplissage=grey, pen trait=currentpen)` trace un quadrillage rectangulaire. l désigne la largeur et h la hateur du quadrillage.

▶ `void fractionCercle(int l,int n=0, pen remplissage=grey, pen trait=currentpen)` trace des parts de camembert [1]. l désigne le nombre total de parts.

▶ `void fractionPoly(int l,int n=0, pen remplissage=grey, pen trait=currentpen)` trace un polygone régulier à l côtés séparé en l parts.

[1] ou tout autre fromage circulaire

```
import geoplane;
figure((0,0),(3,3));
fractionPoly(3,1);
```



5 les objets de type path

5.1 triangle et quadrilatères

Pour les triangles, la longueur a est située en face du point A, b et c sont placés ensuite dans le sens trigonométrique. L'angle $\widehat{alpha}$ est l'angle en A, $\widehat{beta}$ en B.

▶ `triangle3c(pair A, real a, real b, real c, bool dessus=true,real angle=0)` renvoie un path, le triangle dont les côtés sont a b et c en cm.

▶ `triangle1c(pair A, real c, real alpha, real beta,bool dessus=true,real angle=0)` renvoie un path, le triangle de côté c en cm.et d'angles adjacents alpha et beta

▶ `triangle2c(pair A, real c, real b, real alpha, bool dessus=true,real angle=0)` renvoie un path, le triangle de côtés adjacents c et b en cm.et formant un angle alpha.

▶ `rectangle(pair A, real a, real b,bool diagonale=false,real angle=0)` renvoie un path, le rectangle de côtés adjacents a et b en cm. Si diagonale=true, b est la diagonale.

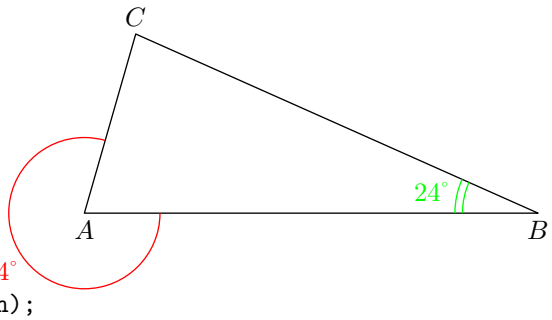
▶ `parallelogramme(pair A, real a, real b, real alpha, bool diagonale=false, real angle=0)` renvoie un path, le parallelogramme de côtés adjacents a et b en cm formant un angle alpha.

Si diagonale vaut true, alpha est la diagonale en cm.

▶ `pair sommet(path c,int n)` renvoie le sommet n du triangle ou quadrilatère. La numérotation commence à 0.

▶ `path polygone(...pair[] K)` renvoie un polygone, utile pour tracer des triangles ou quadrilatères définis par des sommets dans le style «à main levee». Il suffit pour cela de donner les sommets ; `path poly=polygone(A,B,C,D,E)`.

```
import geoplane;
pair A=(0,0);
path t=triangle1c(A,6,74,24);
draw(t);
pair B=sommet(t,1);
pair C=sommet(t,2);
label("$A$",A,sud);
label("$B$",B,sud);
label("$C$",C,nord);
codeangle(C,A,B,"$74 \degres$",0,sty=red);
codeangle(C,B,A,"$24 \degres$",trait=0,2,green);
```



5.2 Droites et segments

▶ La commande `segment(pair A, pair B,real a=0)` renvoie le chemin [AB] et permet de faire dépasser le trait de a cm de part et d'autre des extrémités.

▶ `droite(pair A, pair B)` définit un «path» passant par A et B mais contenu dans le `cadre`, le résultat est plus élégant – à mon sens – que `drawline`.

▶ `demidroite(pair A, pair B)` renvoie, elle, la demi-droite $[AB)$

▶ La fonction `perpendiculaire(pair A, pair B, pair M)` (*resp.* `parallele`) retourne la droite perpendicularire (*resp.* parallèle) à (AB) et passant par M.

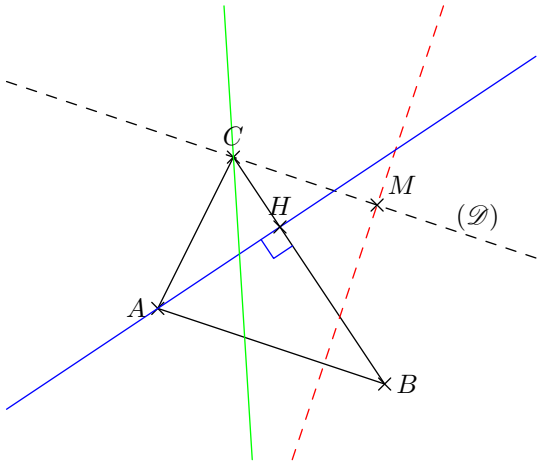
On dispose aussi des fonctions suivantes qui renvoient toutes des droites :

▶ `hauteur(pair C, pair A, pair B)` : la hauteur issue de C de ABC.

▶ `mediatrice(pair A, pair B)` : la médiatrice de [AB]

▶ `bissectrice(pair A, pair B, pair C)` :la bissectrice de l'angle ABC.

```
import geoplane;
usepackage("mathrsfs");
figure((-1,-1),(6,5));
pair A,B,C,M,H;
A=(1,1);
B=(4,0);
C=(2,3);
draw(A--B--C--cycle);
nomme("$A$",A,ouest);
nomme("$B$",B,est);
nomme("$C$",C,nord);
path d=parallele(A,B,C);
draw(d,dashed);
M=pointsur(d,0.3);
nomme("$M$",M,nest);
path dd=perpendiculaire(C,M,M);
draw(dd,dashed+red);
label("$(\mathscr{D})$",1.7*(M-C)+C,nord);
draw(hauteur(A,C,B),blue);
draw(bissectrice(A,C,B),green);
H=projortho(A,C,B);
nomme("$H$",H,nord);
angledroit(A,H,B,blue);
```



5.3 Les cercles

Les cercles ne sont pas contenus dans le `cadre` (le moyen n'a pas été encore trouvé ...), ils peuvent en déborder. La commande suivante permet de ne tracer qu'une partie de crecle.

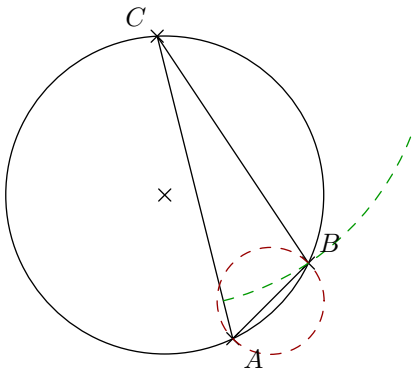
▶ `arc(pair B, pair A, real s, real t)` renvoie un chemin qui est l'arc de centre B, passant par A et avec un angle autour de A, le 0 étant sur A.

▶ `cercle(pair O, pair A)` renvoie le cercle de centre O et de rayon A.

▶ `cercleR(pair O, real R)` définit le cercle de centre O et de rayon R.

▶ `cercleD(pair A, pair B)` donne le cercle de diamètre [AB].

```
import geoplane;
pair A,B,C;
figure((-1,-1),(5,5));
A=(3,0);
B=(4,1);
C=(2,4);
nomme("$A$",A,sest);
nomme("$B$",B,nest);
nomme("$C$",C,nouest);
pointe(circonscrit(A,B,C));
draw(cercle(circonscrit(A,B,C),A));
draw(A--B--C--cycle);
draw(cercleD(A,B),dashed+0.6*red);
draw(arc(C,B,-20,45),green*0.6+dashed);
```



5.4 Nommer les paths

La commande

▶ `nomchemin(string txt,path chemin, real pos=0.9, pair direction, pen p=currentpen)` place à proximité d'un objet `path` son nom . Le texte est placé par défaut à le position 0,9 du chemin, au sens de la fonction `pointsur`.

6 Repérage

6.1 Axes gradués

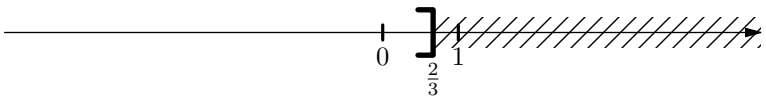
▶ `void inequation(string txt="", real valeur, real crochet, real zone, pen pen=currentpen)` permet de tracer l'axe gradué, hachuré à partir de `valeur` dans la direction définie par `zone` :

si `zone` $= -1$; la partie vers les abscisses négatifs est hachurée.

si `zone` $= 1$; la partie vers les abscisses positifs est hachurée.

`crochet` fonctionne de la même façon.

```
import geoplane;
figure((-5,-1),(5,1));
inequation("$\frac{2}{3}$",2/3,-1,1);
```



De maninière similaire au type `repere` (6.2), on définit le type `axe` *ie* un point, un vecteur directeur de l'axe et les informations comme le nom de l'origine, etc.

```
struct axe {
pair origine ;
pair abscisse;
string originetxt="$O$" ;
string abscissetxt="1";
}
```

L'axe par défaut est appelé `canonique` :

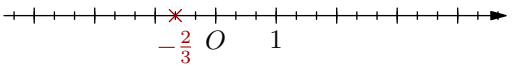
```
axe canonique;
canonique.origine=(0,0);
canonique.abscisse=(1,0);
canonique.originetxt="$O$";
canonique.abscissetxt="1";
```

Par commodité, l'axe courant est contenu dans `axecourant`.

▶ `void graduation(axe grad=axecourant, real debut, real fin, real intermediaire=0, pen sty=currentpen)` trace un axe gradué, les valeurs `début` et `fin` sont exprimées en fonction du vecteur unité, `intermediaire` est une fraction du vecteur unité.

▶ `void abscisse(axe grad=axecourant, string txtdessous, string txtdessus="", real x, bool croix=true, pen sty=currentpen)` trace un point qui correspond à une abscisse particulière sur un axe

```
import geoplane;
figure((-5,-1),(5,1));
axe axe;
axe.origine=(0,0);
axe.abscisse=(0.8,0);
axecourant=axe;
graduation(-3.5,4.8,1/3);
abscisse("$-\frac{2}{3}$",-2/3,red*0.6);
```



6.2 Repères

Le type `repere` contient les informations nécéssaires à un repère du plan :

```
struct repere {
pair origine ;
pair abscisse;
pair ordonnee;
string originetxt ;
string abscissetxt;
string ordonneetxt;
}
```

origine contient les coordonnées de l'origine.

abscisse contient les coordonnées du vecteur de l'axe des abscisses.

ordonnee contient les coordonnées du vecteur de l'axe des ordonnées.

originetxt contient le nom de l'origine.

abscissetxt contient le nom de du vecteur de l'axe des abscisses.

ordonneetxt contient le nom de du vecteur de l'axe des ordonnées.

Le repère par défaut est définit par `reperecourant`, une constante de type `repere`. C'est, sauf définition de l'utilisateur, le repère « canonique » de `Asymptote` .

```
repere canonique;
canonique.origine=(0,0);
canonique.abscisse=(1,0);
canonique.ordonnee=(0,1);
canonique.originetxt="$O$";
canonique.abscissetxt="$\vec{i}$";
canonique.ordonneetxt="$\vec{j}$";
```

Une fois le repere définit, il reste à tracer les axes et eventuellement la base :

▶ `axes(repere rep,int graduation=1,pen sty=currentpen)` trace les axes si graduation=1, gradué toutes les unités, 2 toutes les demi-unités, 3 tous les dixièmes.

▶ `base(repere rep,bool vecteur=true,pen sty=currentpen)` trace une base avec les vecteurs. Si `vecteur` est `true`, le nom des vecteurs apparaissent.

6.3 Les points

▶ `pair place(string nom,real x, real y,pair direction, repere rep=reperecourant, bool trait=false, pen sty=currentpen)` renvoie et place un pair avec les coordonnées (x ;y) dans le repère rep mais dont les coordonnées pour `Asymptote` peuvent être différentes.

▶ `pair position(real x, real y,repere rep=reperecourant)` renvoie un pair avec les coordonnées (x ;y) dans le repère rep mais dont les coordonnées pour `Asymptote` peuvent être différentes.

7 Les courbes et fonctions

7.1 Courbes

▶ `tracepara (real f(real),real g(real),real a, real b, repere rep=reperecourant, int precision=500, pen sty=currentpen+linewidth(0.5))` trace des courbes paramètrées.

▶ `tracepolaire (real r(real), real a, real b, repere rep=reperecourant, int precision=500, pen sty=currentpen+linewidth(0.5))` trace des courbes polaires.

7.2 Fonctions

▶ `tracefonction (real f(real), real a, real b, repere rep=reperecourant, int precision=500, pen sty=currentpen+linewidth(0.5))` trace des fonctions.

▶ `path tangente(real a, real f(real), real h=0.01, repere rep=reperecourant)` trace la tangente en a à f.

▶ `path morceau (real f(real), real a, real b, repere rep=reperecourant, int precision=500)` renvoie un morceau de courbe si tout le tracé est contenu dans le cadre. Cette fonction est pratique pour hachurer des zones ou calculer des intersections de courbes.

▶ `nomfonction(string txt, real f(real), real abscisse, repere oij=reperecourant, pair diri, pen p=currentpen)` place le nom du graphe de la fonction f.

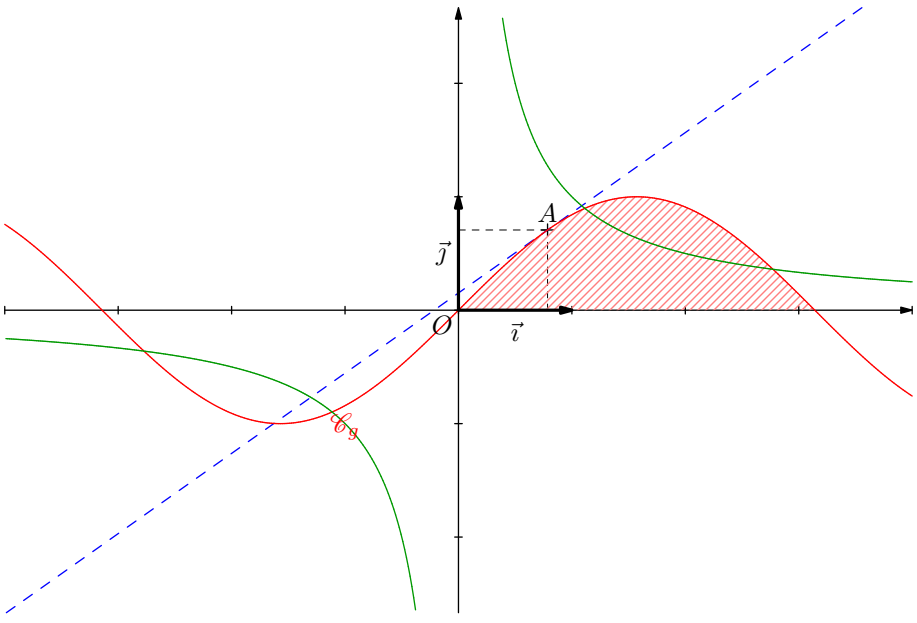
Exemple de tracé de fonctions :

```
import geoplane;
figure((-6,-4),(6,4));
pair A;
repere oij;
oij.origine=(0,0);
oij.abscisse=(1,0);
oij.ordonnee=(0,1);
oij.originetxt="$O$";
oij.abscissetxt="$\vec{\imath}$";
oij.ordonneetxt="$\vec{\jmath}$";
reperecourant=oij;
real f2(real x){return sin(x);}

axes(oij,1);

A=place("$A$",pi/4,f2(pi/4),nord,oij,trait=true);
hachurage(morceau(f2,0,pi/4,200)--position(pi/4,0)--oij.origine--cycle,2,45,grey);
draw(tangente(pi/4,f2),blue+dashed);
tracefonction(f2,-1.99,2,oij,red);
//commencer le trace à -1,99 permet d'éviter une division par zéro

nomfonction("$\mathscr{C}_g$",f2,-1,sud,red);
base(oij);
```



Exemple de tracé de courbes paramétrées :

```
import geoplane;
figure((-6,-6),(6,6));

repere oij;
oij.origine=(0,0);
oij.abscisse=(1.2,0);
```

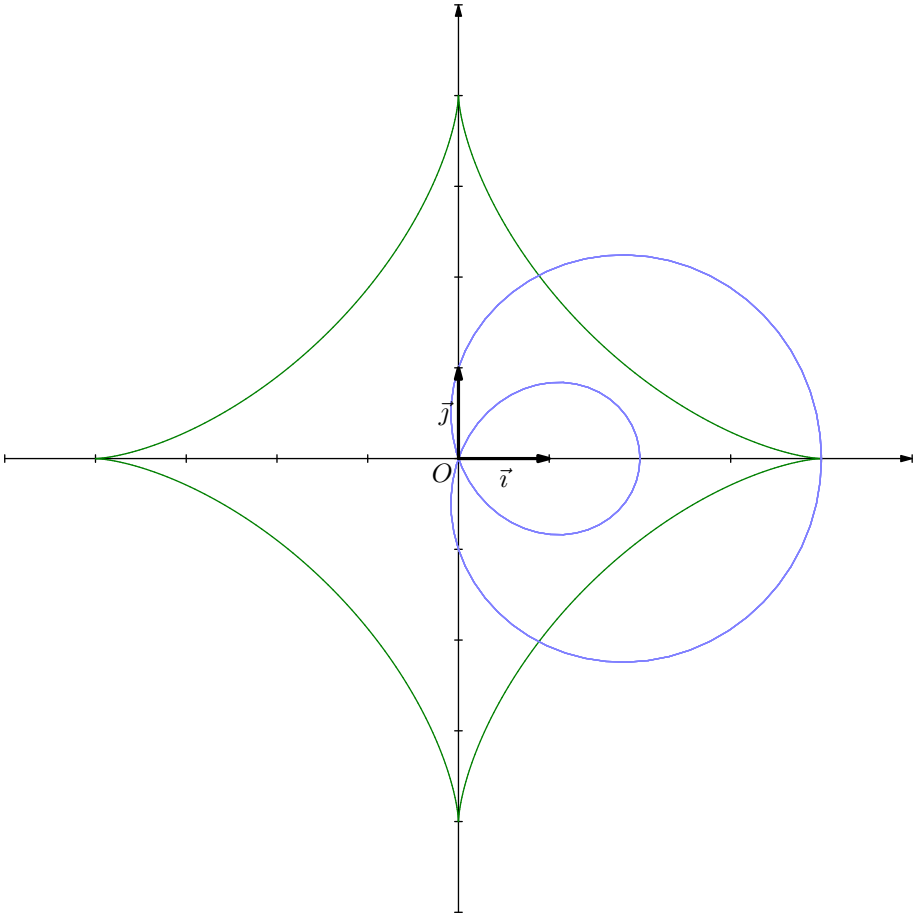
```
oij.ordonnee=(0,1.2);
oij.originetxt="$O$";
oij.abscissetxt="$\vec{\imath}$";
oij.ordonneetxt="$\vec{\jmath}$";
reperecourant=oij;

real x1(real t){return 4*(cos(t)^3);}
real y1(real t){return 4*(sin(t)^3);}
real r(real t){return 1+3*cos(t);}

axes(oij,1);

tracepara(x1,y1,-3.14,3.14,oij,0.5*green);
tracepolaire(r,0,10*pi,lightblue);

base(oij);
```

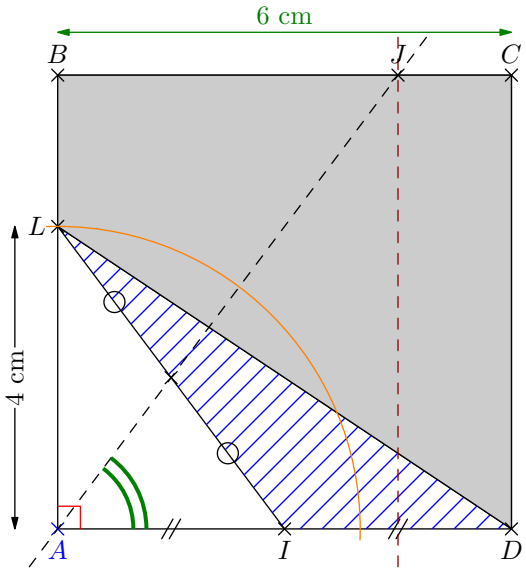


Et un dernier exemple plus complet ...

```
import geoplane;
import math;
import markers;
import geometry;
import patterns;

pair A,B,C,D,I,L,J;

figure((-0.5,-0.5),(6.5,6.5));
A=(0,0);
B=(0,6);
C=(6,6);
D=C-B+A;
I=milieu(A,D);
L=A+4/6*(B-A);
draw(A--B--C--D--cycle);
draw(L--I);
draw(L--D);
angledroit(L,A,I,red);
cotemilieu(A,L,"4 cm",2mm);
cote(B,C,"6 cm",2mm,0.5*green);
hachurage(I--L--D--cycle,2mm,45,blue);
filldraw(L--D--C--B--cycle,0.8*white);
nomme("$A$",A,p=blue,sud);
nomme("$B$",B,nord);
nomme("$C$",C,nord);
nomme("$D$",D,sud);
nomme("$I$",I,sud);
nomme("$L$",L,ouest);
pointe(milieu(L,I));
codemilieu(L,I,5);
draw(droite(A,milieu(L,I)),dashed);
J=intersectionpoint(droite(A,milieu(L,I)),B--C);
nomme("$J$",J,nord);
codemilieu(A,D,2);
path d;
d=perpendiculaire(A,D,J);
draw(d,0.6*red+dashed);
draw(arc(A,L,2,-92),orange);
codeangle(I,A,J,0,2,green*0.5+1.5bp);
```


6 cm
B
J
C
L
4 cm
A
I
D