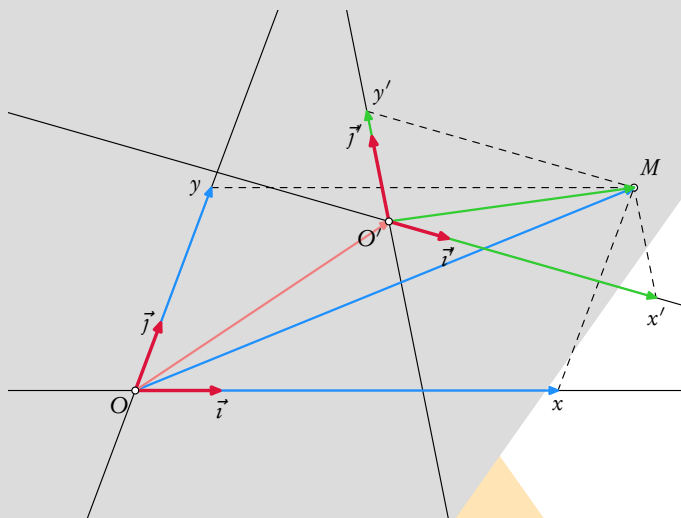


# mp-geom2d

Paquet METAPOST pour des figures de géométrie plane



**Contributeurs**  
Maxime CHUPIN  
Jean-Michel SARLAT

Version 1.1 du 22 octobre 2024

<https://gitlab.gutenberg-asso.fr/mchupin/mp-geom2d>

<https://ctan.org/pkg/mp-geom2d>

## 1 Objectif

`mp-geom2d` a été écrit avec le but de proposer des macros METAPOST permettant de réaliser une figure de géométrie *en collant* d'assez près à une description impérative :

*Soit A le point de coordonnées (2,3).*

*Soit B le point de coordonnées (4,5).*

*Trace la droite (A,B).*

....

Dans ce cadre, les objets géométriques sont le plus souvent nommés ( $A$ ,  $B$ , etc.) ou désignés par leur nature et leurs attributs (droite  $(A,B)$ , etc.). Pour ne pas avoir à dépasser ce mode de description, en particulier pour éviter d'avoir à déclarer le *type* de ces objets, le choix a été fait de les identifier par un *index*<sup>1</sup> dans des tables qui en précisent les caractéristiques.

**Note** — À ce jour, l'objectif n'est pas atteint, le développement est loin d'être achevé ; il est encore nécessaire de faire appel à des commandes METAPOST ou à des *macros intermédiaires* pour décrire une figure. Cela évoluera sans doute avec le temps, le temps de trouver une syntaxe satisfaisante...

## 2 Les fichiers

`mp-geom2d` est un ensemble d'outils pour la géométrie plane avec METAPOST [5]. Cet ensemble est organisé en plusieurs fichiers :

1. `geom2d.mp` : c'est le fichier principal, qui charge tous les autres ;
2. `geom2d-main.mp` : il contient les structures et fonctions générales ;
3. `geom2d-arc.mp` : contient tout ce qui concerne les arcs de cercles ;
4. `geom2d-c2d.mp` : contient tout ce qui concerne les courbes du second degré ;
5. `geom2d-fct.mp` : contient quelques fonctions mathématiques usuelles ;
6. `geom2d-lbl.mp` : contient les fonctions relatives aux labels ;
7. `geom2d-plt.mp` : contient des fonctions facilitant la représentation de fonctions mathématiques ;
8. `geom2d-rep.mp` contient différents outils pour le tracé de figure dans un repère ;
9. `geom2d-tra.mp` contient les fonctions permettant de gérer la transparence (code emprunté à Anthony PHAN) ;
10. `geom2d-svgnames.mp` fournit les 150 couleurs de la spécification SVG.

Nous allons, dans la suite, décrire plus en détails chacune de ces fonctions. Certaines fonctions s'appuient sur l'extension `graph.mp` présente dans toutes les bonnes distributions T<sub>E</sub>X.

## 3 Principe général de fonctionnement

`mp-geom2d` utilise des tables comme structure principale. Chaque objet est numéroté via le compteur `gdd0`, son type<sup>2</sup> est stocké dans la table `gddT[]` à la place `gddT[gdd0]`.

1. Le type `numeric`, qui est le type par défaut dans METAPOST, ne demande pas de déclaration préalable.

2. Les types sont propres à `mp-geom2d` et seront décrits plus tard.

Les propriétés des objets sont définies dans, là encore, des tables de type `numeric` qui sont `gddA[]`, `gddB[]`, ..., `gddF[]`.

Par exemple, pour un `Point` (type `mp-geom2d`), la première coordonnée se trouve dans `gddA[]` et la seconde dans `gddB[]` (les autres tables ne sont pas utilisées pour un tel objet).

Il y a trois tables particulières `gddP[]` qui est du type `path`, `gddPX[][]` qui est sa version étendue, et `gddS[]` qui est du type `string`. Nous verrons plus tard quelle est leur utilité.

Bien entendu, lors d'une utilisation classique de `mp-geom2d`, l'appel à toutes ces tables n'est pas chose courante. Les fonctions que nous allons décrire dans la suite de ce document permettent de ne pas avoir recours trop précisément à cette machinerie.

## 4 Tracés

`mp-geom2d` fournit des macros pour tracer des objets. Une commande permet de tracer la plupart des objets de `mp-geom2d` (qui sont décrits dans la section 5), ainsi que certains type `METAPOST`. Les exemples tout au long de la documentation permettront d'illustrer les commandes de tracé.

Pour comprendre comment fonctionne les représentations avec `mp-geom2d`, il nous faut tout d'abord décrire le mécanisme de gestion des unités.

### 4.1 Unité

`mp-geom2d` définit une unité générale avec la variable globale `gddU`. Ce `numeric` vaut par défaut 1 cm.

`gddU`

### 4.2 En place

Lors de l'utilisation des commandes de tracé, `mp-geom2d` utilise la macro

`gddEnPlace`

Cette macro permet de spécifier les transformations géométriques nécessaires au placement de l'objet, notamment le facteur d'échelle relatif à l'unité `gddU`.

En interne, cette macro est un `scantoken` de la variable globale `gddW` (`string`) qui contient les transformations.

### 4.3 Commandes de tracés

La macro de tracé la plus courante est la macro `trace`.

`trace(objet)`

- ⟨objet⟩** : — pour le côté METAPOST, un **path**, une **picture** ou un **pair**, et dans ce cas **trace** sera équivalent à **draw** ;
- du côté de mp-geom2d, n'importe quel des 6 objets définis jusque là : **triangle**, **segment**, **droite**, **cercle**, **ellipse**, **parabole**, **hyperbole**, **polygone**, **chemin**, **courbe** ou **vecteur** (pour ce dernier, une flèche sera ajoutée à l'extrémité du vecteur).

Lorsque **trace** est utilisée sur un objet de mp-geom2d, la macro fait appel à la commande METAPOST **draw** avec le mécanisme de mise en place décrit plus haut.  
mp-geom2d fournit un tracé avec flèche faisant non plus appel à **draw** mais à **drawarrow**

### **fleche(⟨objet⟩)**

- ⟨objet⟩** : — pour le côté METAPOST, un **path**, une **picture** ou un **pair**, et dans ce cas **trace** sera équivalent à **drawarrow** ;
- du côté de mp-geom2d, n'importe quel des 6 objets définis jusque là : **triangle**, **segment**, **droite**, **cercle**, **ellipse**, **polygone**, **chemin**, **courbe** ou **vecteur** (pour ce dernier, une flèche sera ajoutée à l'extrémité du vecteur).

On peut aussi colorier un objet grâce à la macro suivante.

### **colorie(⟨objet⟩)**

- ⟨objet⟩** : peut être :
- pour le côté METAPOST, uniquement un **path** et dans ce cas **trace** sera équivalent à **fill** ;
  - du côté de mp-geom2d, n'importe quel objet coloriable : **triangle**, **cercle**, **ellipse**, **polygone**, **chemin**, ou **courbe**.

Les commandes de tracés de mp-geom2d font appel aux macros de base de METAPOST : **draw** et **fill**. Ainsi, pour spécifier la couleur, on pourra utiliser la commande **withcolor** ou bien la macro **drawoptions**.

On peut aussi colorier avec de la transparence<sup>3</sup> avec la commande suivante.

### **colorieAvecTransparence(⟨objet⟩, ⟨couleur⟩, ⟨coefficient de transparence⟩)**

- ⟨objet⟩** : peut être :
- pour le côté METAPOST, uniquement un **path** et dans ce cas **trace** sera équivalent à **fill** ;
  - du côté de mp-geom2d, n'importe quel objet coloriable : **triangle**, **cercle**, **ellipse**, **polygone**, **chemin**, ou **courbe**.
- ⟨couleur⟩** : est une **color**.

---

3. En réalité, on simule de la transparence avec le code emprunté à Anthony Phan : <http://www-math.univ-poitiers.fr/~phan/metalpha.html>

*(coefficient de transparence)* : est un **numeric** compris entre 0 et 1, 0 pour un coloriage opaque et 1 pour un coloriage invisible.

Les macros ci-dessus, font appel à une macro plus bas niveau qui permet de retourner le **path** à tracer ou colorier pour tous les objets mp-geom2d.

**gddTraceObjet**(*objet*) → **path**

*objet* : **triangle**, **segment**, **vecteur**, **cercle**, **ellipse**, **parabole**, **hyperbole**, **polygone**, **chemin**, ou **courbe**.

Si l'objet n'est pas fermé, mp-geom2d fournit la macro suivante qui ajoute un **--cycle** à la suite du chemin :

**fermeture**(*objet*) → **path**

(fermé **--cycle**)

*objet* : — du côté de **METAPOST**, **path**;  
— du côté de **mp-geom2d**, **triangle**, **segment**, **vecteur**, **cercle**, **chemin**, ou **courbe**.

On pourra spécifier le **crayon** utilisé pour les tracés avec la commande suivante :

**avecCrayon**(*taille*), (*couleur*)

*taille* : facteur d'échelle **numeric** du **pen pencircle** de **METAPOST** utilisé ;  
*couleur* : couleur **color** à utiliser.

Cette macro est un raccourci pour le *classique* `withpen pencircle scaled ... withcolor ...`. On peut d'ailleurs évidemment toujours utiliser les commandes **METAPOST** pour le dessin.

On peut représenter les objets de type **point**. Pour cela, on dispose d'une commande qui trace un petit cercle coloré à l'endroit du point. La commande est la suivante :

**pointe**(*point*)

*point* : **point** ou **pair**

Cette macro utilise deux variables globales de mp-geom2d qui sont

**gddTaillePoint**

(**numeric**, valeur par défaut 3)

et

`gddCouleurPoint`

(*color*, valeur par défaut *white*)

Il existe deux autres types de pointage de point : la simple croix et le simple disque. Pour choisir un autre type de pointage (en utilisant la même commande `pointe`), on modifiera la variable globale suivante :

`gddPointType`

(*string*, valeur par défaut "*pointe*")

Peut valoir : "*pointe*", "*croix*" ou "*disque*".

## 4.4 Contenir les tracés

Il est souvent nécessaire de restreindre les tracés à une zone du plan  $\mathbb{R}^2$ . Pour cela, `mp-geom2d` fournit la commande `Fenetre` :

`Fenetre(Xmin,Ymin,Xmax,Ymin)`

Cette fonction trace un rectangle dont deux sommets opposés sont les points (*Xmin*, *Ymin*) et (*Xmax*, *Ymax*) avec la couleur METAPOST `background` et découpe l'image courante autour de ce cadre (avec la commande METAPOST `clip currentpicture`).

## 5 Les types

`mp-geom2d`, fournit plusieurs types d'objets. Le type d'objet est stocké dans la table `gddT[]`, et les tables `gddA[]` à `gddF[]`, ainsi que la table `gddX` contiennent les propriétés des objets.

Nous allons ici décrire chaque type de l'extension `mp-geom2d`, leurs propriétés respectives, ainsi que des méthodes associées directement à l'objet.

### 5.1 Le type `point`

Ce type correspond au point de l'espace euclidien.

#### 5.1.1 Constructeur

Pour être plus clair voici la fonction principale pour créer un tel objet :

```
1 vardef Point(expr a,b) =
2   gddT[incr gdd0] = "point";
3   gddA[gdd0] = a; gddB[gdd0] = b; gdd0
4 enddef;
```

```
Point( $\langle x \rangle, \langle y \rangle$ ) → point  
   $\langle x \rangle$  : numeric;  
   $\langle y \rangle$  : numeric.
```

Cette fonction « retourne » le compteur `gdd0` et crée dans la table de type une entrée `point` et les attributs (coordonnées) correspondants `a` et `b` dans les tables `gddA` et `gddB`.  
Avec un tel type de fonctionnement, la plupart des manipulations se fait sur des `numeric`s.  
En effet, pour déclarer un `point`, il suffit d'écrire

```
1 A = Point(2,3);
```

`A` prend alors la valeur courante de `gdd0`. C'est l'identifiant du point.  
On peut aussi définir un `point` par ses coordonnées *polaires* avec la commande suivante.

```
PointPolaire( $\langle r \rangle, \langle a \rangle$ ) → point  
   $\langle r \rangle$  : numeric, module du point;  
   $\langle a \rangle$  : numeric, angle par rapport à l'axe des abscisses.
```

On peut définir un point dans un repère défini lui par un point d'origine et deux vecteurs avec la fonction suivante :

```
PointDansRepere( $\langle x \rangle, \langle y \rangle, \langle o \rangle, \langle i \rangle, \langle j \rangle$ ) → point  
   $\langle x \rangle$  : numeric;  
   $\langle y \rangle$  : numeric;  
   $\langle o \rangle$  : point d'origine du repère;  
   $\langle i \rangle$  : vecteur (ou point) définissant l'axe des abscisses;  
   $\langle j \rangle$  : vecteur (ou point) définissant l'axe des ordonnées.
```

Le code suivant illustre l'utilisation de cette commande (avec des `points` au lieu de `vecteurs`, type décrit plus tard).

```
1 O = Point(0,0);  
2 I = Point(2,0);  
3 J = Point(0,3);  
4 A = PointDansRepere(2,3,0,I,J);
```

### 5.1.2 Macros associées

On peut récupérer l'abscisse et l'ordonnée d'un point avec les commandes suivantes :

`Abscisse( $\langle a \rangle$ )` → `numeric`  
 `$\langle a \rangle$`  : `point`, `vecteur` ou `pair`.

`Ordonnee( $\langle a \rangle$ )` → `numeric`  
 `$\langle a \rangle$`  : `point`, `vecteur` ou `pair`.

On pourra ajouter les abscisses, les ordonnées ou bien, à la manière des vecteurs, des points entre eux avec les commandes suivantes.

`AdditionAbscisses( $\langle a \rangle$ )` → `numeric`  
 `$\langle a \rangle$`  : `point` ou `pair`.

`AdditionOrdonnee( $\langle a \rangle$ )` → `numeric`  
 `$\langle a \rangle$`  : `point` ou `pair`.

et

`Addition( $\langle a \rangle$ ,  $\langle b \rangle$ )` → `point`  
 `$\langle a \rangle$`  : `point` ou `pair` ;  
 `$\langle b \rangle$`  : `point` ou `pair`.

On peut aussi calculer la longueur entre deux points grâce à la commande suivante :

`Longueur( $\langle a \rangle$ ,  $\langle b \rangle$ )` → `numeric`  
 `$\langle a \rangle$`  : `point` ou `pair` ;  
 `$\langle b \rangle$`  : `point` ou `pair`.

On peut calculer le point équidistant de deux points, le milieu :

`Milieu( $\langle a \rangle$ ,  $\langle b \rangle$ )` → `point`



$\langle a \rangle$  : point ou pair ;

$\langle b \rangle$  : point ou pair.

On peut réaliser la rotation d'un point autour de l'origine (0,0) avec la commande suivante :

**Rotation**( $\langle a \rangle, \langle b \rangle$ ) → point

$\langle a \rangle$  : point ou pair ;

$\langle b \rangle$  : numeric, l'angle de rotation en radian.

Si on veut réaliser la rotation d'un point autour d'un autre, on utilisera la commande suivante :

**RotationCentre**( $\langle a \rangle, \langle b \rangle, \langle c \rangle$ ) → point

$\langle a \rangle$  : point ou pair ;

$\langle b \rangle$  : point ou pair, centre de rotation ;

$\langle c \rangle$  : numeric, l'angle de rotation en radian.

On peut calculer l'isobarycentre d'une liste de points (ou de pair de METAPOST) avec la commande suivante.

**IsoBarycentre**( $\langle a \rangle, \langle b \rangle, \langle c \rangle, \text{ etc. } \rangle$ ) → point

$\langle a \rangle$  : point ou pair ;

$\langle b \rangle$  : point ou pair ;

etc.

On peut calculer le barycentre d'une liste de points associées à des poids (mais ici, il n'est pas possible d'utiliser des pair de METAPOST).

**Barycentre**( $(\langle A \rangle, \langle a \rangle), (\langle B \rangle, \langle b \rangle), (\langle C \rangle, \langle c \rangle), \text{ etc. } \rangle$ ) → point

$\langle A \rangle$  : point ;

$\langle a \rangle$  : numeric, poids associé à  $\langle A \rangle$  ;

$\langle B \rangle$  : point ;

$\langle b \rangle$  : numeric, poids associé à  $\langle B \rangle$  ;

etc.

On peut déterminer la bissectrice d'un secteur angulaire défini par trois points. La fonction retourne une **droite**.

**Bissectrice**( $\langle A \rangle, \langle B \rangle, \langle C \rangle$ )  $\rightarrow$  **droite**

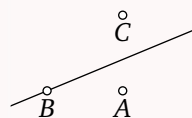
$\langle A \rangle$  : **point**;

$\langle B \rangle$  : **point**;

$\langle C \rangle$  : **point**.

### Exemple 1

```
1 input geom2d;
2 beginfig(1);
3 A = Point(1,0);
4 B = Point(0,0);
5 C = Point(1,1);
6 D = Bissectrice(A,B,C);
7 trace D;
8 marque.bot "A";
9 marque.bot "B";
10 marque.bot "C";
11 Fenetre(-0.5,-0.5,2,2);
12 endfig;
```



Le type **point** est évidemment lié au type METAPOST **pair**.  
mp-geom2d fournit des macros qui permettent de passer de **point** à **pair** et réciproquement.

**PointTOPair**( $\langle a \rangle, \langle b \rangle$ )  $\rightarrow$  **pair**

$\langle a \rangle$  : **numeric**, abscisse;

$\langle b \rangle$  : **numeric**, ordonnée.

**PairTOPoint**( $\langle p \rangle$ )  $\rightarrow$  **point**

$\langle p \rangle$  : **pair**.

Ces deux commandes sont complétées par deux autres qui assure qu'un élément est d'un type donné :

**PairImp**( $\langle p \rangle$ )  $\rightarrow$  **pair**

$\langle p \rangle$  : **point** ou **pair**.

`PointImp(p)` → `point`

*p* : `point` ou `pair`.

On peut aussi récupérer un point le long d'un objet `mp-geom2d` (décrit dans les sections suivantes). La macro suivante retourne un `point` le long du chemin (cyclique ou non) de l'objet le paramétrant entre 0 et 1.

`PointDe(o,t)` → `point`

*o* : n'importe quel objet `mp-geom2d` (même un `point` pour lequel lui-même est retourné);

*t* : `numeric` compris entre 0 et 1 (qui paramètre le chemin de l'objet entre 0 et 1).

## 5.2 Le type `vecteur`

Ce type correspond aux vecteurs définis à l'aide de deux coordonnées de l'espace euclidien.

### 5.2.1 Constructeurs

La fonction créatrice d'un tel objet est celle-ci

```
1 vardef Vecteur(expr a,b) =
2   save n; n = incr gdd0;
3   gddT[n] = "vecteur"; gddA[n] = a; gddB[n] = b; n
4 enddef;
```

`Vecteur(a,b)` → `vecteur`

*a* : `numeric`;

*b* : `numeric`.

Cette fonction a la même architecture que celle correspondante au `point` : elle retourne la valeur courante de `gdd0` après incrémentation, puis affecte le type `vecteur` à l'entrée correspondante dans la table `gddT`.

On peut aussi définir un vecteur à partir d'un `pair` METAPOST.

```
1 vardef VecteurP(expr a) =
2   save n; n = incr gdd0;
3   gddT[n] = "vecteur"; gddA[n] = xpart a; gddB[n] = ypart a; n
4 enddef;
```

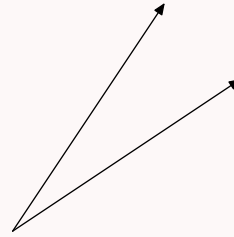
`VecteurP( $\langle a \rangle$ )` → vecteur

$\langle a \rangle$  : pair.

Un vecteur peut se définir alors comme ceci :

#### Exemple 2

```
1 input geom2d;  
2 beginfig(1);  
3 AB = Vecteur(2,3);  
4 pair a;  
5 a := (3,2);  
6 A = VecteurP(a);  
7 trace AB;  
8 trace A;  
9 endfig;
```



#### 5.2.2 Macros associées

Comme les objets de `mp-geom2d` ne sont pas des `numeric`s, les opérations classiques de l'espace vectoriel ne peuvent pas s'écrire avec les simples caractères `+`, `-`, et `*`.

`AdditionVecteur( $\langle a \rangle, \langle b \rangle$ )` → vecteur

$\langle a \rangle$  : vecteur;

$\langle b \rangle$  : vecteur.

`SoustractionVecteur( $\langle a \rangle, \langle b \rangle$ )` → vecteur

$\langle a \rangle$  : vecteur;

$\langle b \rangle$  : vecteur.

`ScalaireVecteur( $\langle k \rangle, \langle b \rangle$ )` → vecteur

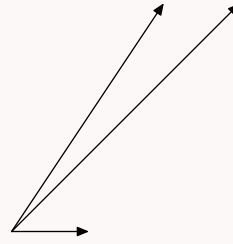
$\langle k \rangle$  : numeric;

$\langle b \rangle$  : vecteur.

On pourra alors faire les opérations classiques sur les vecteurs.

### Exemple 3

```
1 input geom2d;  
2 beginfig(1);  
3 AB = Vecteur(2,3);  
4 BC = Vecteur(1,0);  
5 AC = AdditionVecteur(AB,BC);  
6 kAC = ScalaireVecteur(3.0,AC);  
7 trace AB;  
8 trace BC;  
9 trace AC;  
10 endfig;
```



On peut aussi calculer le produit scalaire de deux vecteurs avec la commande suivante.

`ProduitScalaire( $\langle a \rangle, \langle b \rangle$ )` → `numeric`

$\langle a \rangle$  : `vecteur`;

$\langle b \rangle$  : `vecteur`.

On peut aussi calculer la norme euclidienne d'un vecteur.

`Norme( $\langle a \rangle$ )` → `numeric`

$\langle a \rangle$  : `vecteur`.

La commande suivante permet d'obtenir l'angle, en radian, entre  $[0, \pi]$  formé entre deux vecteurs. Si on note  $u$  et  $v$  les deux vecteurs de  $\mathbf{R}^2$ , l'angle calculé par la commande suivante est obtenu avec la formule suivante :

$$\theta = \arccos\left(\frac{u \cdot v}{\|u\| \|v\|}\right).$$

`Angle( $\langle a \rangle, \langle b \rangle$ )` → `numeric` (dans  $[0, \pi]$ )

$\langle a \rangle$  : `vecteur`;

$\langle b \rangle$  : `vecteur`.

### 5.3 Le type `segment`

Les segments sont définis par deux points de  $\mathbf{R}^2$ .

### 5.3.1 Constructeur

La fonction créatrice de cet objet est celle-ci

```
1 vardef Segment (expr a,b) =  
2   save n; n = incr gdd0;  
3   gddT[n] = "segment"; gddA[n] = PointImp(a); gddB[n] = PointImp(  
4     b); n  
5 enddef;
```

`Segment(a,b)` → `segment`

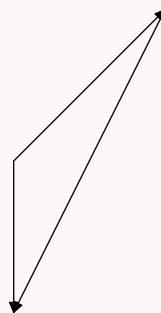
*a* : `point`;

*b* : `point`.

Un segment se définit alors comme ceci :

#### Exemple 4

```
1 input geom2d;  
2 beginfig(1);  
3 A = Point(2,3);  
4 B = Point(4,5);  
5 C = Point(2,1);  
6 AB = Segment(A,B);  
7 BC = Segment(B,C);  
8 AC = Segment(A,C);  
9 fleche AB;  
10 fleche BC;  
11 fleche AC;  
12 endfig;
```



### 5.3.2 Macros associées

On peut calculer la longueur d'un segment avec la macro suivante.

`LongueurSegment(a)` → `numeric`

*a* : `segment`.

On peut aussi convertir un segment en vecteur avec la macro suivante qui fait la différence des coordonnées des deux points définissant le segment.

`SegmentTOVecteur(a)` → `vecteur`

*a* : `segment`.

## 5.4 Le type **droite**

Une droite est simplement définie par deux points.

### 5.4.1 Constructeur

Ainsi la fonction créatrice de cet objet est la suivante

```
1 vardef Droite (expr a,b) =  
2   save n; n = incr gdd0;  
3   gddT[n] = "droite"; gddA[n] = PointImp(a); gddB[n] = PointImp(b  
4   ); n  
5 enddef;
```

**Droite**( $\langle a \rangle, \langle b \rangle$ )  $\rightarrow$  **droite**

$\langle a \rangle$  : **point**;  
 $\langle b \rangle$  : **point**.

Lors de la représentation des droites (voir section 4.3), le caractère *infini* de la droite est géré par la variable globale **gddExtensionDroite** qui vaut 10 par défaut.

### 5.4.2 Macros associées

La macro suivante permet, sous forme d'un triplet, et donc d'une **color**, d'obtenir les coefficients d'une droite donnée. Pour une droite ( $D$ ), la macro donne le triplet  $(u, v, w) \in \mathbb{R}^3$  tel que  $\forall (x, y) \in (D), ux + vy + w = 0$ .

**EquationDroite**( $\langle a \rangle$ )  $\rightarrow$  **color**

$\langle a \rangle$  : **droite**.

On peut calculer la projection d'un point sur une droite avec la commande suivante.

**ProjectionPointSurDroite**( $\langle p \rangle, \langle a \rangle$ )  $\rightarrow$  **point**

$\langle p \rangle$  : **point**;  
 $\langle a \rangle$  : **droite**.

On peut obtenir l'ordonnée relative d'un point sur une droite avec la macro suivante.

**OrdonneeRelativePointDroite**( $\langle p \rangle, \langle a \rangle$ )  $\rightarrow$  **numeric**

$\langle p \rangle$  : **point**;

$\langle a \rangle$  : droite.

Le calcul de la distance d'un point à une droite se fait avec la macro suivante.

**DistancePointDroite**( $\langle p \rangle$ ,  $\langle a \rangle$ )  $\rightarrow$  numeric

$\langle p \rangle$  : point;

$\langle a \rangle$  : droite.

La macro suivante permet d'obtenir la droite perpendiculaire à une droite passant par un point.

**DroitePerpendiculaire**( $\langle a \rangle$ ,  $\langle p \rangle$ )  $\rightarrow$  droite

$\langle a \rangle$  : droite;

$\langle p \rangle$  : point.

On pourra obtenir le point d'intersection de deux droites avec la macro suivante.

**IntersectionDroites**( $\langle a \rangle$ ,  $\langle b \rangle$ )  $\rightarrow$  point

$\langle a \rangle$  : droite;

$\langle b \rangle$  : droite.

La macro suivante permet de reporter un point sur une droite avec une certaine longueur (signée), à la manière d'un compas.

**ReportSurDroite**( $\langle P \rangle$ ,  $\langle D \rangle$ ,  $\langle l \rangle$ )  $\rightarrow$  point

$\langle P \rangle$  : point;

$\langle D \rangle$  : droite;

$\langle l \rangle$  : numeric (peut être négatif).

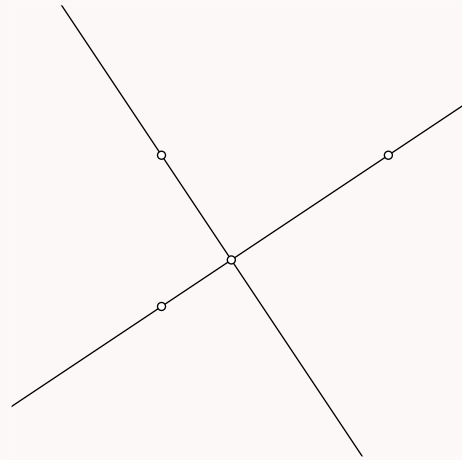
Le côté du report par rapport au point  $\langle P \rangle$  dépend de la définition de la droite : pour une droite définie par deux points  $A$  et  $B$ , alors la direction du report sera suivant le vecteur  $\overline{AB}$ . Pour changer de direction, il suffira de multiplier le paramètre de distance  $\langle l \rangle$  par  $-1$ .

Voici un exemple permettant d'illustrer l'utilisation de quelques commandes relatives aux droites.



### Exemple 5

```
1 input geom2d;
2 beginfig(1);
3 A = Point(0,0);
4 B = Point(3,2);
5 AB = Droite(A,B);
6 trace AB;
7 pointe A;
8 pointe B;
9 C = Point(0,2);
10 DC = DroitePerpendiculaire(AB,C);
11 trace DC;
12 pointe C;
13 D = ProjectionPointSurDroite(C,AB
14 );
15 pointe D;
16 Fenetre(-2,-2,4,4);
17 endfig;
```



## 5.5 Le type **cercle**

Un cercle est défini par un point et un rayon.

### 5.5.1 Constructeurs

La fonction créatrice de base de cet objet est la suivante

```
1 vardef Cercle (expr a,b) =
2   save n; n = incr gdd0;
3   gddT[n] = "cercle"; gddA[n] = PointImp(a); gddB[n] = b; n
4 enddef;
```

**Cercle**( $\langle a \rangle, \langle b \rangle$ )  $\rightarrow$  cercle

$\langle a \rangle$  : point ou pair, centre du cercle;

$\langle b \rangle$  : numeric, rayon du cercle.

Deux autres fonctions permettent de définir des cercles. La fonction **CercleCP** définit le cercle par un centre et un point par lequel passe le cercle.

**CercleCP**( $\langle a \rangle, \langle b \rangle$ )  $\rightarrow$  cercle

$\langle a \rangle$  : point ou pair, centre du cercle;

$\langle b \rangle$  : point, tel que  $\langle b \rangle - \langle a \rangle$  est le rayon.

On peut aussi définir un cercle par deux points définissant son diamètre. La fonction correspondante est la suivante

**CercleD**( $\langle a \rangle, \langle b \rangle$ )  $\rightarrow$  **cercle**

$\langle a \rangle$  : **point** ou **pair**;

$\langle b \rangle$  : **point** ou **pair**, tel que  $(\langle a \rangle - \langle b \rangle)$  est le diamètre du cercle.

On peut aussi définir un cercle comme le cercle qui passe par trois points avec la macro suivante.

**CercleTroisPoints**( $\langle a \rangle, \langle b \rangle, \langle c \rangle$ )  $\rightarrow$  **cercle**

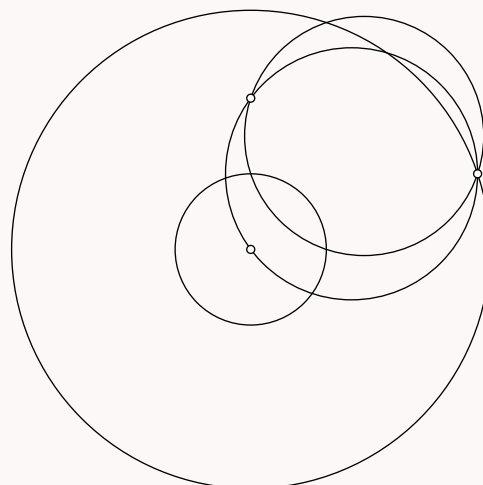
$\langle a \rangle$  : **point** ou **pair**;

$\langle b \rangle$  : **point** ou **pair**;

$\langle c \rangle$  : **point** ou **pair**.

#### Exemple 6

```
1 input geom2d;
2 beginfig(1);
3 A = Point(3,1);
4 B = Point(0,2);
5 O = Point(0,0);
6 r = 1.0;
7 COr = Cercle(0,r);
8 COA = CercleCP(0,A);
9 CAB = CercleD(A,B);
10 COAB = CercleTroisPoints(0,A,B);
11 trace COr;
12 trace COA;
13 trace CAB;
14 trace COAB;
15 pointe A; pointe B; pointe O;
16 endfig;
```



#### 5.5.2 Macros associées

De nombreuses macros sont associées aux cercles.

On pourra récupérer le rayon d'un cercle avec la commande suivante.

**Rayon**( $\langle a \rangle$ )  $\rightarrow$  **numeric**

$\langle a \rangle$  : **cercle**.

On pourra obtenir le centre (`point`) d'un cercle avec la commande :

`Centre(a)` → `point`

*a* : `cercle`, `ellipse` ou `hyperbole`.

On peut calculer l'intersection entre deux cercles grâce à la macro suivante.

`IntersectionCercles(a,b)` → `point`

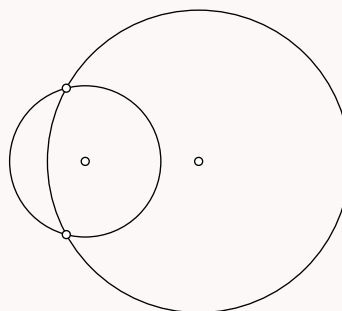
*a* : `cercle`;

*b* : `cercle`.

Cette macro ne donnera qu'un seul point d'intersection. Pour obtenir les deux points d'intersection, il faudra inverser l'ordre d'appel sur les cercles comme l'exemple suivant l'illustre.

#### Exemple 7

```
1 input geom2d;
2 beginfig(1);
3 A = Point(0,0);
4 CA = Cercle(A,1);
5 B = Point(1.5,0);
6 CB = Cercle(B,2);
7 D1 = IntersectionCercles(CA,CB);
8 D2 = IntersectionCercles(CB,CA);
9 trace CA; trace CB;
10 pointe A; pointe B; pointe D1;
    pointe D2;
11 endfig;
```



Si il n'existe pas d'intersection, alors la compilation échouera avec l'erreur ! `Pythagorean subtraction X.XXXX+-+X.XXXX has been replaced by 0`. Si les deux cercles sont confondus, alors l'erreur sera ! `Division by zero`..

`mp-geom2d` fournit aussi une macro permettant d'obtenir les intersections entre une droite et un cercle.

`IntersectionDroiteCercle(d,c,n)` → `point`

*d* : `droite`;

*c* : `cercle`;

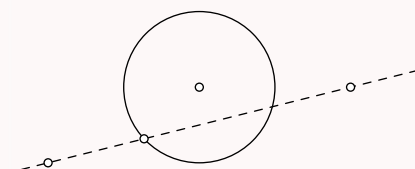
*n* : `numeric` qui vaut 1 ou 2 suivant le point d'intersection que l'on souhaite (s'il n'existe qu'un point d'intersection alors les deux valeurs renvoient le même point).

S'il n'existe pas d'intersection entre la droite et le cercle, vous obtiendrez l'erreur !  
Intersection between line and circle does not exist.

L'exemple suivant permet d'illustrer cette macro.

### Exemple 8

```
1 input geom2d;  
2 beginfig(1);  
3 A = Point(2,2);  
4 C_A = Cercle(A,1);  
5 B = Point(0,1);  
6 C = Point(4,2);  
7 BC = Droite(B,C);  
8 E1 = IntersectionDroiteCercle(BC,  
   C_A,1);  
9 trace C_A;  
10 trace BC dashed evenly;  
11 pointe A; pointe B; pointe C;  
   pointe E1;  
12 Fenetre(-0.5,-0.5,5,4);  
13 endfig;
```



On peut obtenir les tangentes intérieures et extérieures communes à deux cercles avec les deux macros suivantes. Là encore, comme il existe deux tangentes, pour obtenir les deux, on inversera l'ordre d'appel des deux cercles.

**TangenteCommuneExterieur( $\langle a \rangle, \langle b \rangle$ )** → droite

$\langle a \rangle$  : cercle;

$\langle b \rangle$  : cercle.

**TangenteCommuneInterieur( $\langle a \rangle, \langle b \rangle$ )** → droite

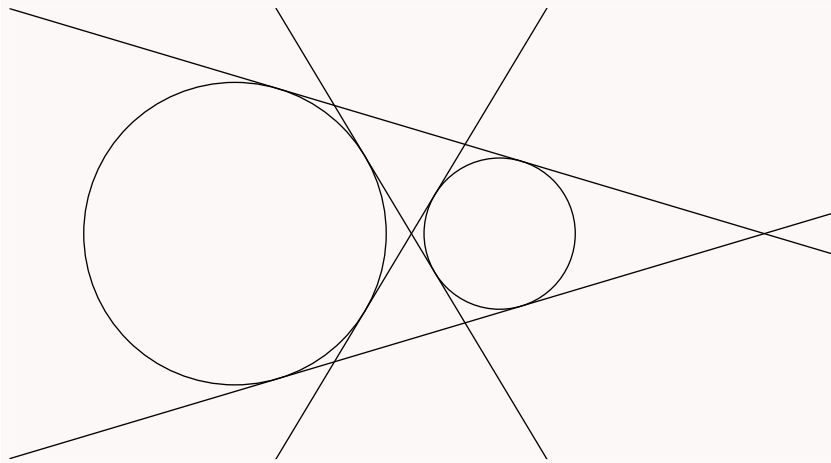
$\langle a \rangle$  : cercle;

$\langle b \rangle$  : cercle.

L'exemple suivante illustre l'utilisation de ces deux macros.

### Exemple 9

```
1 input geom2d;  
2 beginfig(1);  
3 C1 = Cercle((-2,0),2);  
4 C2 = Cercle((1.5,0),1);  
5 T1 = TangenteCommuneExterieur(C1,C2);  
6 T2 = TangenteCommuneExterieur(C2,C1);  
7 T3 = TangenteCommuneInterieur(C1,C2);  
8 T4 = TangenteCommuneInterieur(C2,C1);  
9 trace C1; trace C2;  
10 trace T1; trace T2;  
11 trace T3; trace T4;  
12  
13 Fenetre(-5,-3,6,3);  
14 endfig;
```



La macro suivante permet de calculer l'axe radical (**droite**) de deux cercles<sup>4</sup>.

**AxeRadical**( $\langle a \rangle, \langle b \rangle$ )  $\rightarrow$  **droite**

$\langle a \rangle$  : **cercle**;

$\langle b \rangle$  : **cercle**.

La macro suivante permet d'obtenir le centre radical de trois cercles.

**CentreRadical**( $\langle a \rangle, \langle b \rangle, \langle c \rangle$ )  $\rightarrow$  **point**

$\langle a \rangle$  : **cercle**;

$\langle b \rangle$  : **cercle**;

<sup>4</sup> La méthode de construction a été largement inspirée du code de T. Thurston dans son document *Drawing with METAPOST* [6].

`<c> : cercle.`

La macro suivante permet de calculer l'axe de similitude de trois cercles.

`AxeDeSimilitude(<a>,<b>,<c>) → droite`

`<a> : cercle;`

`<b> : cercle;`

`<c> : cercle.`

## 5.6 Le type **ellipse**

### 5.6.1 Constructeurs

Les ellipses peuvent être définies de plusieurs façons. Tout d'abord, on peut la définir avec son centre, un des points de l'ellipse sur le grand axe (appelé vertex), et un des points de l'ellipse sur le petit axe (appelé co-vertex). Cependant, lors de la création d'une ellipse de nombreux attributs sont calculés.

Le code du constructeur est le suivant :

```
1 vardef Ellipse(expr C,A,B) =
2   % C : centre
3   % A : vertex
4   % B : co-vertex
5   save n,a,b,c,e,_tmp,slope;
6   pair _tmp;
7   n = incr gdd0;
8   gddT[n] = "ellipse"; gddA[n] = PointImp(C); gddB[n] = PointImp(
9     A);
10  gddC[n] = PointImp(B);
11  % calcul des deux foyers
12  a = Longueur(C,A);
13  b = Longueur(C,B);
14  c = sqrt(a**2-b**2);
15  e = c/a;
16  _tmp = e*(Pt(A)-Pt(C));
17  % les foyers
18  gddD[n] = PairTOPoint(Pt(C)+_tmp);
19  gddE[n] = PairTOPoint(Pt(C)-_tmp);
20  gddX[n][1] = a;
21  gddX[n][2] = b;
22  gddX[n][3] = e;
23  % angle du demi-grand axe
24  slope = angle(PairImp(A)-PairImp(C));
```

```

24 gddX[n][4] = slope;
25 n
26 endif;

```

**Ellipse(*c*, *a*, *b*)** → ellipse

- <c>** : point ou pair, le centre;
- <a>** : point ou pair, le vertex;
- <b>** : point ou pair, le co-vertex.

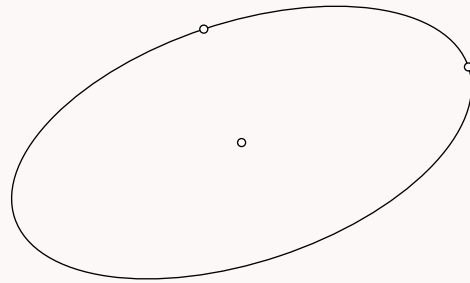
Cette macro s'utilise comme le montre l'exemple suivant.

#### Exemple 10

```

1 input geom2d;
2 beginfig(1);
3 C = Point(0,0);
4 A = Point(3,1);
5 B = Rotation((1.5,0.5),Pi/2);
6 E = Ellipse(C,A,B);
7 trace E;
8 pointe A; pointe B; pointe C;
9 endfig;

```



On pourra définir une ellipse avec la donnée de ses deux foyers et du demi-grand axe avec la commande suivante :

**EllipseF(*A*, *B*, *a*)** → ellipse

- <A>** : point ou pair, premier foyer;
- <B>** : point ou pair, deuxième foyer;
- <a>** : numeric, le demi-grand axe.

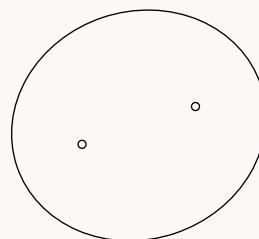
Cette macro s'utilise comme le montre l'exemple suivant.

#### Exemple 11

```

1 input geom2d;
2 beginfig(1);
3 F1 = Point(3,1);
4 F2 = Point(1.5,0.5);
5 E = EllipseF(F1,F2,1.7);
6 trace E;
7 pointe F1; pointe F2;
8 endfig;

```



### 5.6.2 Macros associées

On pourra obtenir le centre (**point**) d'une ellipse avec la commande :

```
Centre(a) → point  
a : cercle, ellipse ou hyperbole.
```

On obtient le vertex et le co-vertex avec les commandes suivantes :

```
Vertex(a) → point  
a : ellipse.
```

```
CoVertex(a) → point  
a : ellipse.
```

La commande suivante permet d'obtenir les deux foyers selon l'entier passé en argument.

```
Foyer(a, n) → point  
a : ellipse;  
n : numeric, entier qui vaut 1 ou 2 pour obtenir chaque foyer.
```

On peut obtenir le demi-grand axe et le demi-petit axe avec les commandes suivantes.

```
DemiGrandAxe(a) → numeric  
a : ellipse.
```

```
DemiPetitAxe(a) → numeric  
a : ellipse.
```

On obtient l'excentricité, souvent notée  $e$ , avec la commande suivante.



`Excentricite(a)` → `numeric`

*a* : `ellipse`, `parabole` ou `hyperbole`.

Pour obtenir l'inclinaison (coefficient directeur de la droite passant par les foyers de l'ellipse), on utilisera la commande suivante.

`Inclinaison(a)` → `numeric`

*a* : `ellipse`, `parabole` ou `hyperbole`.

Pour obtenir la tangente (`droite`) à un point de l'ellipse, on utilisera la commande suivante.

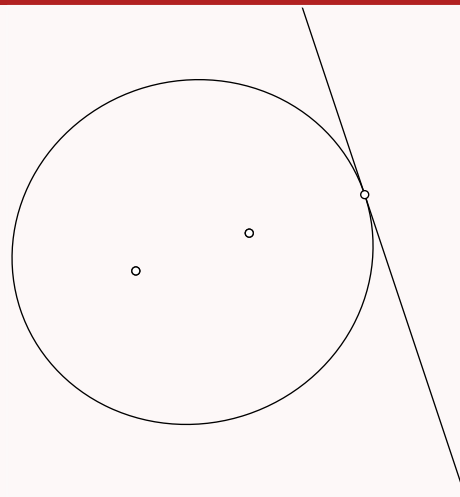
`TangenteEllipse(e,p)` → `droite`

*e* : `ellipse`;

*p* : `point` ou `pair`.

### Exemple 12

```
1 input geom2d;
2 beginfig(1);
3 F1 = Point(3,1);
4 F2 = Point(1.5,0.5);
5 E = EllipseF(F1,F2,2.4);
6 pointe F1; pointe F2;
7 trace E;
8 M' = PointDe(E,0.5);
9 D = TangenteEllipse(E,M');
10 trace D;
11 pointe M';
12 pointe Foyer(E,1);
13 pointe Foyer(E,2);
14 Fenetre(-0.2,-2.5,6,4);
15 endfig;
```



Pour obtenir les tangentes (`droite`) passant par un point extérieur à l'ellipse, on utilisera la commande suivante. Si le point choisi n'est pas extérieur à l'ellipse, alors il y aura une erreur de compilation.

`TangenteExterieurEllipse(e),(p),(n)` → droite

*e* : ellipse;

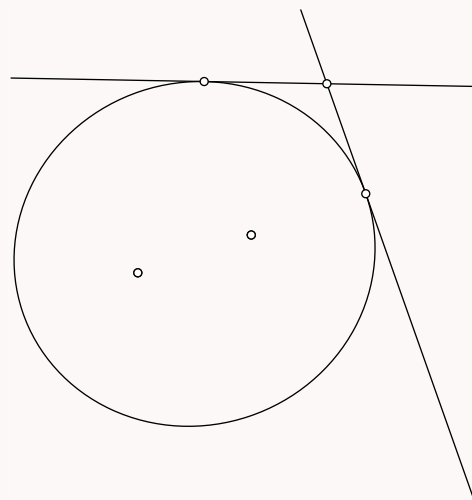
*p* : point ou pair;

*n* : numeric qui vaut 1 ou 2 pour choisir la tangente parmi les deux possibles.

Les points de tangence sont accessibles comme deuxième point de la définition de la droite grâce à la table `gddB[]`.

### Exemple 13

```
1 input geom2d;
2 beginfig(1);
3 F1 = Point(3,1);
4 F2 = Point(1.5,0.5);
5 E = EllipseF(F1,F2,2.4);
6 pointe F1; pointe F2;
7 trace E;
8 M = Point(4,3);
9 D1 = TangenteExterieurEllipse(E,
10   M,1);
11 trace D1;
12 pointe gddB[D1];
13 D2 = TangenteExterieurEllipse(E,
14   M,2);
15 trace D2;
16 pointe gddB[D2];
17 pointe M;
18 pointe Foyer(E,1);
19 pointe Foyer(E,2);
20 Fenetre(-0.2,-2.5,6,4);
21 endfig;
```



## 5.7 Le type parabole

Le constructeur de base de la parabole définit cet objet à partir du foyer et de la directrice. Le code du constructeur est le suivant :

```
1 vardef ParaboleFD(expr F,D) =
2   % F : foyer (point)
3   % D : directrice (droite)
4   save u, v, w, d, i, n,_tmp,slope;
5   pair _tmp;
6   n = incr gdd0;
7   (u,v,w) = EquationDroite(D);
8   % ordonnée relative
9   d := u * gddA[F] + v * gddB[F] + w;
10  gddT[n] := "parabole";
```

```

11 gddX[n][1] := D; % on stocke la directrice
12 gddX[n][2] := D; % on stocke la directrice (compatibilité avec
    hyperbole)
13 % sommet
14 _tmp := (((-d/2)*(u,v)) shifted PairImp(F));
15 gddB[n] = PointImp(_tmp);
16 gddC[n] = PointImp(_tmp);
17 % le foyer (doublé pour compatibilité)
18 gddD[n] := F;
19 gddE[n] := F;
20 gddX[n][3] := 1.0; % excentricité
21 % angle du demi-grand axe
22 slope = angle(PairImp(gddA[D])-PairImp(gddB[D]))+90;
23 gddX[n][4] = slope;
24 i := -gddC2Dparam-1;
25 gddP[n] := (
26   ((i*(v,-u)+((i*i-d*d)/(2d))*(u,v))
27   for i:= -gddC2Dparam upto gddC2Dparam:
28     ..(i*(v,-u)+((i*i-d*d)/(2d))*(u,v))
29   endfor)) shifted PairImp(F);
30   n
31 enddef;

```

**ParaboleFD( $\langle f \rangle, \langle d \rangle$ )** → chemin

$\langle f \rangle$  : est le foyer (**point**) de la parabole;

$\langle d \rangle$  : est la directrice (**droite**) de la parabole.

Lors de la représentation d'une parabole, le caractère *infini* est géré par la variable globale `gddC2Dparam` qui vaut 15 par défaut.

### 5.7.1 Fonctions associées

On obtient l'excentricité, souvent notée  $e$ , avec la commande suivante.

**Excentricite( $\langle a \rangle$ )** → numeric

$\langle a \rangle$  : **ellipse**, **parabole** ou **hyperbole**.

Pour obtenir l'inclinaison (coefficient directeur de la droite axe de symétrie de la parabole), on utilisera la commande suivante :

**Inclinaison( $\langle a \rangle$ )** → numeric

$\langle a \rangle$  : ellipse, parabole ou hyperbole.

On peut obtenir le sommet de la parabole avec la commande suivante :

**Sommet**( $\langle a \rangle, \langle n \rangle$ ) → point

$\langle a \rangle$  : parabole ou hyperbole ;

$\langle n \rangle$  : 1 ou 2 (**numeric**), argument utile pour l'hyperbole qui possède deux sommets.  
Ici, peu importe la valeur de  $\langle n \rangle$ , l'unique sommet sera donné.

On peut obtenir la directrice de la parabole avec la commande suivante :

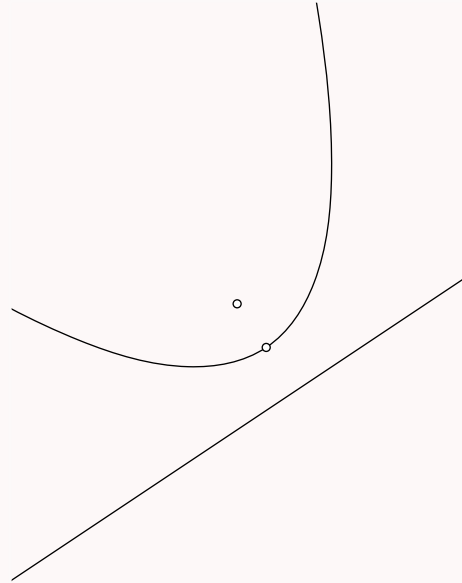
**Directrice**( $\langle a \rangle, \langle n \rangle$ ) → droite

$\langle a \rangle$  : parabole ou hyperbole ;

$\langle n \rangle$  : 1 ou 2 (**numeric**), argument utile pour l'hyperbole qui possède deux directrices.  
Ici, peu importe la valeur de  $\langle n \rangle$ , l'unique directrice sera donnée.

## Exemple 14

```
1 input geom2d;
2 beginfig(1);
3 A = Point(0,0);
4 B = Point(3,2);
5 AB = Droite(A,B);
6 F = Point(-1,1);
7 Para = ParaboleFD(F,AB);
8 trace AB;
9 trace Para;
10 pointe Sommet(Para,1);
11 pointe F;
12 Fenetre(-4,-5,2,5);
13 endfig;
```



## 5.8 Le type **hyperbole**

Le constructeur de base de l'hyperbole définit cet objet à partir du foyer et de la directrice. Le code du constructeur est le suivant :

```
1 vardef HyperboleFD(expr F,D,e) =
2   % F : foyer (point)
3   % D : directrice (droite)
4   % e : exentricité (numeric)
5   % pm : +1 ou -1 pour les deux branches
6   save u, v, w, d, i, n, _tmp, slope, aa, bb;
7   pair _tmp;
8   n = incr gdd0;
9
10  (u,v,w) = EquationDroite(D);
11  d := u * gddA[F] + v * gddB[F] + w;
12  gddT[n] := "hyperbole";
13
14  % sommets
15  _tmp := -(d+1*sqrt(e*e*d*d))/(e*e-1)*(u,v)shifted (
16    ProduitScalaire(F,Vecteur(v,-u))*(v,-u));
17  gddB[n] := PointImp(_tmp);
```

```

17  _tmp := -(d-1*sqrt(e*e*d*d))/(e*e-1)*(u,v)shifted
18  (ProduitScalaire(F,Vecteur(v,-u))*(v,-u));
19  gddC[n] := PointImp(_tmp);
20  % centre comme milieu des deux sommets
21  gddA[n] = Milieu(gddB[n],gddC[n]);
22  % le foyer (doublé pour compatibilité)
23  gddD[n] := F;
24  gddE[n] := RotationCentre(F, gddA[n], Pi);
25  gddX[n][3] := e; % excentricité
26  % directrices
27  gddX[n][1] := D; % on stocke la directrice
28  gddX[n][2] := Droite(RotationCentre(gddA[D], gddA[n], Pi),
    RotationCentre(gddB[D], gddA[n], Pi));
29  % angle de l'axe
30  slope = angle(PairImp(gddA[D])-PairImp(gddB[D]))+90;
31  gddX[n][4] = slope;
32  % cercle principale
33  gddX[n][5] = CercleD(gddB[n],gddC[n]);
34  % asymptotes
35  aa = IntersectionDroiteCercle(D,gddX[n][5],1);
36  bb = IntersectionDroiteCercle(D,gddX[n][5],2);
37  gddX[n][6] = Droite(gddA[n],aa);
38  gddX[n][7] = Droite(gddA[n],bb);
39  % tracés des moitiés
40  i := -gddC2Dparam-1;
41  gddPX[n][1] := (
42  (
43  (i*(v,-u)-(d+sqrt(e*e*d*d+i*i*(e*e-1)))/(e*e-1)*(u,v))
44  for i:= -gddC2Dparam upto gddC2Dparam:
45  ..(i*(v,-u)-(d+sqrt(e*e*d*d+i*i*(e*e-1)))/(e*e-1)*(u,v))
46  endfor
47  ) shifted (ProduitScalaire(F,Vecteur(v,-u))*(v,-u))
48  );
49  gddPX[n][2] := (
50  (
51  (i*(v,-u)-(d-sqrt(e*e*d*d+i*i*(e*e-1)))/(e*e-1)*(u,v))
52  for i:= -gddC2Dparam upto gddC2Dparam:
53  ..(i*(v,-u)-(d-sqrt(e*e*d*d+i*i*(e*e-1)))/(e*e-1)*(u,v))
54  endfor
55  ) shifted (ProduitScalaire(F,Vecteur(v,-u))*(v,-u))
56  );
57  n
58  enddef;

```

On peut voir dans ce constructeur l'utilisation de la table de `path` étendu `gddPX[][]` qui permet d'associer plusieurs `paths` à un seul objet.

**HyperboleFD**( $\langle f \rangle, \langle d \rangle, \langle e \rangle$ )  $\rightarrow$  chemin

- $\langle f \rangle$  : est le foyer (**point**) de la parabole ;
- $\langle d \rangle$  : est la directrice (**droite**) de la parabole ;
- $\langle e \rangle$  : est l'excentricité de l'hyperbole.

### 5.8.1 Fonctions associées

On pourra obtenir le centre (**point**) d'une hyperbole avec la commande :

**Centre**( $\langle a \rangle$ )  $\rightarrow$  point

- $\langle a \rangle$  : **cercle**, **ellipse** ou **hyperbole**.

On obtient l'excentricité, souvent notée  $e$ , avec la commande suivante.

**Excentricite**( $\langle a \rangle$ )  $\rightarrow$  numeric

- $\langle a \rangle$  : **ellipse**, **parabole** ou **hyperbole**.

Pour obtenir l'inclinaison (coefficient directeur de la droite passant par les foyers de l'hyperbole), on utilisera la commande suivante.

**Inclinaison**( $\langle a \rangle$ )  $\rightarrow$  numeric

- $\langle a \rangle$  : **ellipse**, **parabole** ou **hyperbole**.

On peut obtenir les sommets de l'hyperbole avec la commande suivante :

**Sommet**( $\langle a \rangle, \langle n \rangle$ )  $\rightarrow$  point

- $\langle a \rangle$  : **parabole** ou **hyperbole** ;
- $\langle n \rangle$  : 1 ou 2 (**numeric**), pour choisir l'un des deux sommets.

On peut obtenir les directrices de l'hyperbole avec la commande suivante :

**Directrice**( $\langle a \rangle, \langle n \rangle$ )  $\rightarrow$  droite

- $\langle a \rangle$  : **parabole** ou **hyperbole** ;

$\langle n \rangle$  : 1 ou 2 (**numeric**), pour choisir l'une des deux directrices.

Pour obtenir le *cercle principal* de l'hyperbole, on pourra utiliser la commande suivante :

**CerclePrincipale**( $\langle h \rangle$ ) → **cercle**  
 $\langle h \rangle$  : **hyperbole**.

On pourra aussi obtenir les deux asymptotes de l'hyperbole avec la commande suivante :

**AsymptoteHyperbole**( $\langle h \rangle, \langle n \rangle$ ) → **droite**  
 $\langle h \rangle$  : **hyperbole** ;  
 $\langle n \rangle$  : 1 ou 2 (**numeric**), pour choisir l'une des deux asymptotes.

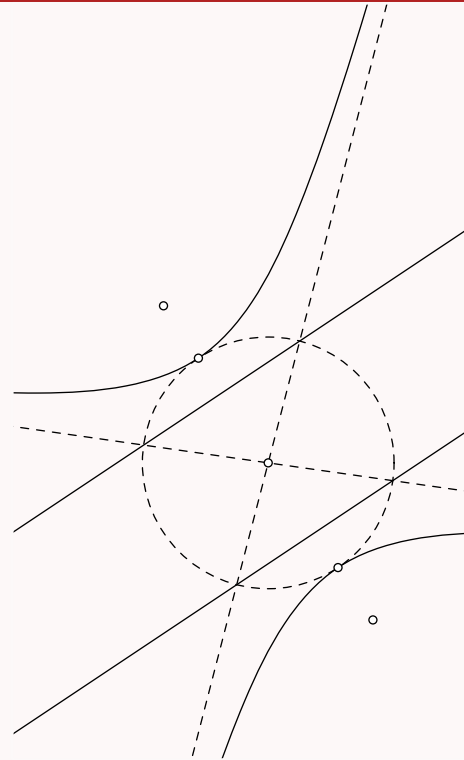
Parce qu'une hyperbole est constituée de deux parties disjointes, on ne peut pas, comme pour les autres objets `mp-geom2d`, utiliser directement la commande **trace** directement sur l'objet. Il faudra utiliser la commande suivante pour obtenir le **path** associé à une des deux parties de l'hyperbole.

**DemiHyperbole**( $\langle h \rangle, \langle n \rangle$ ) → **path**  
 $\langle h \rangle$  : **hyperbole** ;  
 $\langle n \rangle$  : 1 ou 2 (**numeric**), pour choisir l'une des deux parties.



### Exemple 15

```
1 input geom2d;
2 beginfig(1);
3 A = Point(0,0);
4 B = Point(3,2);
5 AB = Droite(A,B);
6 F = Point(-1,1);
7 Hyper = HyperboleFD(F,AB,1.5);
8 trace AB;
9 trace DemiHyperbole(Hyper,1);
10 trace DemiHyperbole(Hyper,2);
11 trace Directrice(Hyper,2);
12 trace CerclePrincipale(Hyper)
   dashed evenly;
13 trace AsymptoteHyperbole(Hyper,1)
   dashed evenly;
14 trace AsymptoteHyperbole(Hyper,2)
   dashed evenly;
15 pointe Centre(Hyper);
16 pointe Foyer(Hyper,1);
17 pointe Foyer(Hyper,2);
18 pointe Sommet(Hyper,1);
19 pointe Sommet(Hyper,2);
20 Fenetre(-3,-5,3,5);
21 endfig;
```



## 5.9 Le type **triangle**

Les triangles sont définis par trois points de  $\mathbb{R}^2$ .

### 5.9.1 Constructeur

La fonction créatrice de cet objet est la suivante

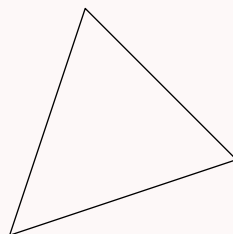
```
1 vardef Triangle (expr a,b,c) =
2   save n; n = incr gdd0; gddT[n] = "triangle";
3   gddA[n] = PointImp(a); gddB[n] = PointImp(b); gddC[n] =
   PointImp(c); n
4 enddef;
```

**Triangle**( $\langle a \rangle, \langle b \rangle, \langle c \rangle$ )  $\rightarrow$  numeric  
 $\langle a \rangle$ : point;  
 $\langle b \rangle$ : point;  
 $\langle c \rangle$ : point.

Ici, on voit l'appel à la troisième table `gddC` pour stocker le troisième point.  
Un triangle se définit alors comme ceci :

#### Exemple 16

```
1 input geom2d;  
2 beginfig(1);  
3 A = Point(3,1);  
4 B = Point(1,3);  
5 C = Point(0,0);  
6 ABC = Triangle(A,B,C);  
7 trace ABC;  
8 endfig;
```



### 5.9.2 Macros associées

La macro suivante permet de calculer l'aire d'un triangle.

`AireTriangle(a)` → `numeric`  
*a* : `triangle`.

On peut calculer aussi l'orthocentre d'un triangle avec la commande suivante.

`Orthocentre(a)` → `point`  
*a* : `triangle`.

La macro suivante calcule le cercle inscrit d'un triangle.

`CercleInscrit(a)` → `cercle`  
*a* : `triangle`.

On peut aussi obtenir le cercle circonscrit avec la macro suivante.

`CercleCirconsrit(a)` → `cercle`  
*a* : `triangle`.

On peut aussi calculer les cercles exinscrits. Soient *A*, *B* et *C* trois points définissant un triangle *ABC*<sup>5</sup>. La commande suivante permet d'obtenir, au choix, un des trois cercles exinscrits.

---

5. Évidemment, les points peuvent s'appeler autrement.

`CercleExinscrit(a,n)` → cercle

*a* : triangle;

*n* : numeric qui vaut 1, 2 ou 3. Si *n*= 1, c'est le cercle exinscrit tangent au côté [BC] du triangle, si *n*= 2, c'est celui tangent au côté [AC] et si *n*= 3, c'est celui tangent au côté [AB].

On peut aussi obtenir le cercle d'Euler d'un triangle avec la commande suivante.

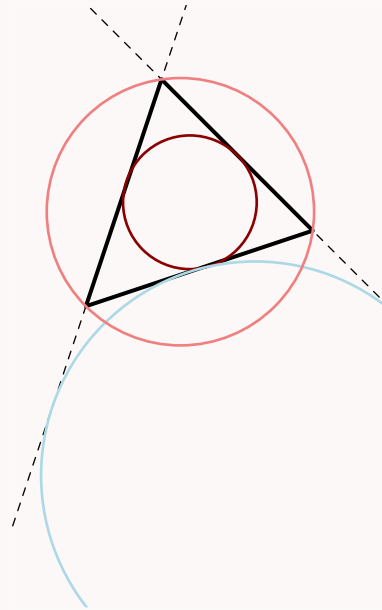
`CercleEuler(a)` → cercle

*a* : triangle.

Voici un exemple d'illustration des quelques unes des macros relatives aux triangles.

#### Exemple 17

```
1 input geom2d;
2 beginfig(1);
3 A = Point(3,1);
4 B = Point(1,3);
5 C = Point(0,0);
6 AB = Droite(A,B);
7 BC = Droite(B,C);
8 ABC = Triangle(A,B,C);
9 trace AB dashed evenly;
10 trace BC dashed evenly;
11 trace ABC avecCrayon(1.5,black);
12 Euler = CercleEuler(ABC);
13 trace Euler avecCrayon(1,DarkRed)
14 ;
15 C_E1 = CercleExinscrit(ABC,2);
16 trace C_E1 avecCrayon(1,LightBlue
17 );
18 C_C = CercleCirconscriit(ABC);
19 trace C_C avecCrayon(1,LightCoral
20 );
21 Fenetre(-1,-4,4,4);
22 endfig;
```



## 5.10 Le type polygone

Les triangles sont définis par  $N$  points de  $\mathbf{R}^2$ .

### 5.10.1 Constructeurs

La fonction créatrice de cet objet est la suivante

```

1 vardef Polygone (text plist) =
2   save n,_point,i; n = incr gdd0; gddT[n] = "polygone";
3   i:=1;
4   for _point = plist:
5     gddX[gdd0][i] = PointImp(_point);
6     i:=i+1;
7   endfor
8   gddA[n] = i-1; % nombre de côté
9   gddB[n] = IsoBarycentre(plist); % centre
10  n
11 enddef;

```

**Polygone**(*liste*) → **polygone**

*liste* : est une liste de **point** ou de **pair**.

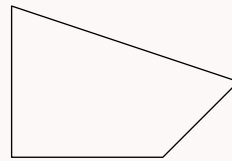
Ici, on voit l'appel à la *double* table **gddX** pour stocker les *N* points.  
Un polygone se définit alors comme ceci :

#### Exemple 18

```

1 input geom2d;
2 beginfig(1);
3 A = Point(2,0);
4 B = Point(3,1);
5 C = Point(0,2);
6 D = Point(0,0);
7 P = Polygone(A,B,C,D);
8 trace P;
9 endfig;

```



On pourra aussi construire des polygones réguliers avec la commande suivante.

**PolygoneRegulier**(*N*), (*rayon*), (*rotation*), (*translation*) → **polygone**

*N* : entier (**numeric**) indiquant le nombre de points ;

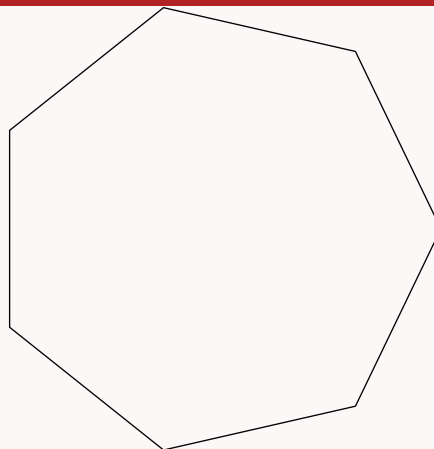
*rayon* : **numeric**, rayon du cercle circonscrit ;

*rotation* : **numeric**, rotation du polygone ;

*translation* : (point) ou **vecteur** (ou même **pair**), déplacement du centre du polygone.

### Exemple 19

```
1 input geom2d;  
2 beginfig(1);  
3 P = PolygoneRegulier(7,3,0,  
   origine);  
4 trace P;  
5 endfig;
```



#### 5.10.2 Macros associées

On pourra obtenir le nombre de côté d'un polygone avec la commande suivante.

`NombreCotesPolygone(a)` → `numeric`  
*a* : polygone.

On pourra avoir accès aux différents sommets du polygone, numérotés à partir de 1, avec la macro suivante.

`PointPolygone(a,i)` → `point`  
*a* : polygone;  
*i* : `numeric`, entier plus grand que 1, permettant d'accéder au *i*<sup>e</sup> point du polygone.

#### 5.11 Le type `chemin`

Pour ce type particulier, `mp-geom2d` stocke le `path` dans la table `gddP` réservée. Ainsi la fonction créatrice de ce type d'objet est la suivante

```
1 vardef Chemin (expr p) =  
2 gddT[incr gdd0] = "chemin"; gddP[gdd0] = p; gdd0  
3 enddef;
```

`Chemin(p)` → `chemin`

$\langle p \rangle$  : un **path** de METAPOST

### 5.11.1 Macros associées

La macro suivante construit, à partir d'une liste de **points**, une ligne brisée de type **chemin**.

**LigneBrisee**( $\langle liste \rangle$ )  $\rightarrow$  **chemin**

$\langle liste \rangle$  : est une liste de **point** ou de **pair**.

## 5.12 Le type **courbe**

Pour ce type particulier, `mp-geom2d` stocke une chaîne de caractère (un nom de fichier à exploiter avec l'extension `graph.mp`) dans la table **gddS** réservée.

### 5.12.1 Constructeur

Ainsi la fonction créatrice de ce type d'objet est la suivante

```
1 vardef CourbeDat (expr s) =  
2   gddT[incr gdd0] = "courbe"; gddS[gdd0] = s; gdd0  
3 enddef;
```

**CourbeDat**( $\langle s \rangle$ )  $\rightarrow$  **courbe**

$\langle s \rangle$  : un nom de fichier sous forme de **string** de METAPOST

## 6 Arc de cercle

Avec `mp-geom2d`, quelques commandes permettent de travailler avec les arcs de cercle. Tout d'abord, on peut décider de ne représenter qu'un arc de cercle d'un **cercle** précédemment défini. Ceci se fait avec la commande suivante qui prend en argument un **cercle** et deux angles en radian, et retourne un **path** de METAPOST correspondant à l'arc de cercle compris entre les deux angles donnés (l'angle 0 étant parallèle à l'axe  $Ox$ ).

**gddTraceArcDeCercle**( $\langle C \rangle, \langle a \rangle, \langle b \rangle$ )  $\rightarrow$  **path**

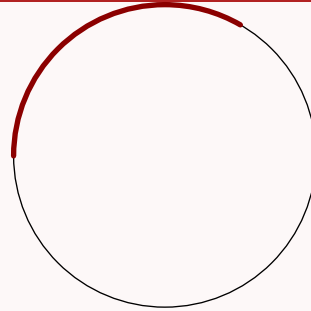
$\langle C \rangle$  : **cercle** dont on souhaite représenter un arc ;

$\langle a \rangle$  : premier angle en radian (**numeric**) ;

$\langle b \rangle$  : deuxième angle en radian (**numeric**).

### Exemple 20

```
1 input geom2d;  
2 beginfig(1);  
3 C = Cercle(origine,2);  
4 trace C;  
5 trace gddTraceArcDeCercle(C,Pi/3,  
  Pi) avecCrayon(2,DarkRed);  
6 endfig;
```



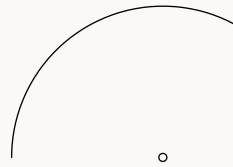
On peut aussi définir un objet `chemin`. Pour cela on pourra utiliser les deux commandes suivantes. La première permet de définir un arc de cercle à partir d'un point, d'un rayon et de deux angles.

$\text{Arc}(\langle P \rangle, \langle r \rangle, \langle a \rangle, \langle b \rangle) \rightarrow \text{chemin}$

- $\langle P \rangle$  : point ou pair centre de l'arc de cercle ;
- $\langle r \rangle$  : rayon de l'arc de cercle (numeric) ;
- $\langle a \rangle$  : premier angle en radian (numeric) ;
- $\langle b \rangle$  : premier angle en radian (numeric).

### Exemple 21

```
1 input geom2d;  
2 beginfig(1);  
3 A = Point(2,2);  
4 Ac = Arc(A,2,Pi/3,Pi);  
5 pointe A;  
6 trace Ac;  
7 endfig;
```



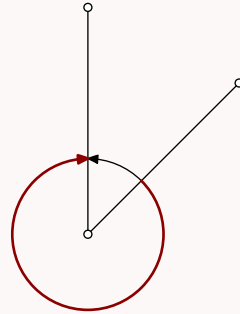
La deuxième commande définissant un `chemin` est la suivante. Elle prend comme arguments trois points, notons les  $C$ ,  $A$  et  $B$  : le centre de l'arc de cercle  $C$ , et les points  $A$  et  $B$ . Elle prend ensuite comme argument le rayon de l'arc défini entre les segments  $[C,A]$  et  $[C,B]$ .

$\text{ArcEntrePoints}(\langle P \rangle, \langle r \rangle, \langle A \rangle, \langle B \rangle, \langle s \rangle) \rightarrow \text{chemin}$

- $\langle P \rangle$  : point ou pair centre de l'arc de cercle ;
- $\langle r \rangle$  : rayon de l'arc de cercle (numeric) ;
- $\langle A \rangle$  : premier point ou pair ;
- $\langle B \rangle$  : deuxième point ou pair ;
- $\langle s \rangle$  : numeric,  $-1$  ou  $1$  suivant le sens choisi.

### Exemple 22

```
1 input geom2d;
2 beginfig(1);
3 C = Point(0,0);
4 A = Point(2,2);
5 B = Point(0,3);
6 Cab = ArcEntrePoints(C,1,A,B,1);
7 Cba = ArcEntrePoints(C,1,A,B,-1);
8 trace Segment(C,A);
9 trace Segment(C,B);
10 pointe A; pointe B; pointe C;
11 fleche Cab;
12 fleche Cba avecCrayon(1,DarkRed);
13 endfig;
```



## 7 Quelques transformations

### 7.1 Homothétie

On peut réaliser une homothétie sur n'importe quel objet mp-geom2d avec la commande suivante.

**Homothetie**( $\langle p \rangle$ ,  $\langle o \rangle$ ,  $\langle k \rangle$ )  $\rightarrow$  du même type que  $\langle p \rangle$

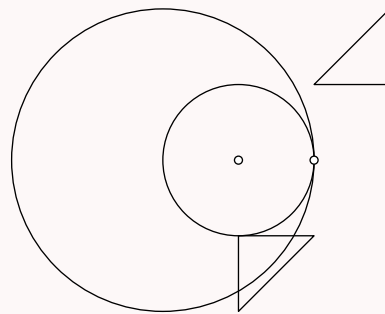
$\langle p \rangle$  : un objet mp-geom2d;

$\langle o \rangle$  : **point** ou **pair**, centre de l'homothétie;

$\langle k \rangle$  : **numeric**, facteur de l'homothétie.

### Exemple 23

```
1 input geom2d;
2 beginfig(1);
3 A = Point(1,2);
4 B = Point(0,2);
5 C_B = Cercle(B,1);
6 T = Triangle((1,1),(0,0),(0,1));
7 C_H = Homothetie(C_B, A, 2);
8 T_H = Homothetie(T, A, -1);
9 trace C_B; trace C_H;
10 trace T; trace T_H;
11 pointe A; pointe B;
12 endfig;
```



### 7.2 Symétrie axiale

On peut réaliser une symétrie axiale sur n'importe quel objet mp-geom2d avec la commande suivante.



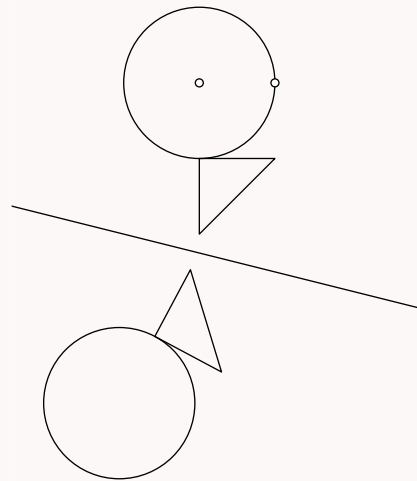
**SymetrieAxiale**( $\langle p \rangle, \langle d \rangle$ )  $\rightarrow$  du même type que  $\langle p \rangle$

$\langle p \rangle$  : un objet mp-geom2d;

$\langle d \rangle$  : **droite** définissant l'axe de symétrie.

#### Exemple 24

```
1 input geom2d;
2 beginfig(1);
3 A = Point(1,2);
4 B = Point(0,2);
5 C_B = Cercle(B,1);
6 T = Triangle((1,1),(0,0),(0,1));
7 D = Droite((-1,0),(3,-1));
8 C_H = SymetrieAxiale(C_B, D);
9 T_H = SymetrieAxiale(T, D);
10 trace C_B; trace C_H;
11 trace T; trace T_H;
12 trace D;
13 pointe A; pointe B;
14 Fenetre(-2.5,-3.5,3,3.2);
15 endfig;
```



### 7.3 Symétrie centrale

On peut réaliser une symétrie centrale sur n'importe quel objet mp-geom2d avec la commande suivante.

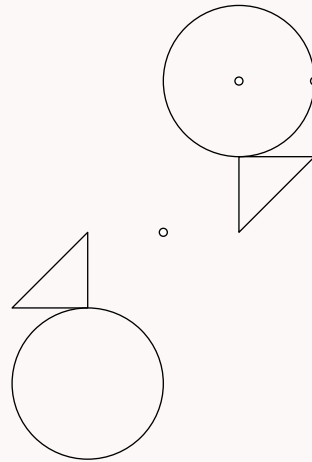
**SymetrieCentrale**( $\langle p \rangle, \langle d \rangle$ )  $\rightarrow$  du même type que  $\langle p \rangle$

$\langle p \rangle$  : un objet mp-geom2d;

$\langle d \rangle$  : **point** définissant le centre de symétrie.

### Exemple 25

```
1 input geom2d;
2 beginfig(1);
3 A = Point(1,2);
4 B = Point(0,2);
5 C_B = Cercle(B,1);
6 T = Triangle((1,1),(0,0),(0,1));
7 D = Point(-1,0);
8 C_H = SymetrieCentrale(C_B, D);
9 T_H = SymetrieCentrale(T, D);
10 trace C_B; trace C_H;
11 trace T; trace T_H;
12 pointe D;
13 pointe A; pointe B;
14 Fenetre(-3.5,-3.5,3,3.2);
15 endfig;
```



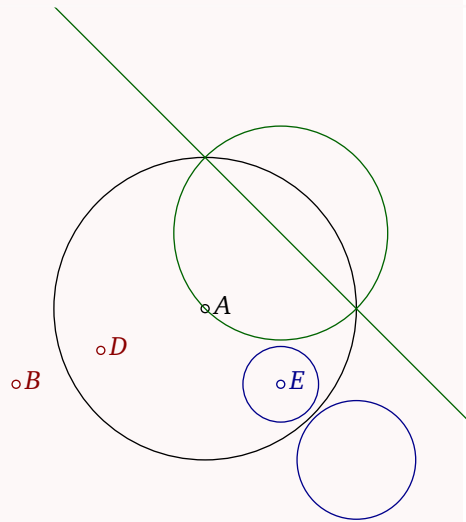
## 7.4 Inversion

On peut calculer l'inversion d'un point, d'un cercle ou d'une droite par rapport à un cercle avec la macro suivante.

**Inversion**( $\langle p \rangle$ ,  $\langle c \rangle$ )  $\rightarrow$  point  
 $\langle p \rangle$  : point, cercle ou droite;  
 $\langle c \rangle$  : cercle.

## Exemple 26

```
1 input geom2d;
2 beginfig(1);
3 A = Point(2,2);
4 C = Cercle(A,2);
5 trace C;
6 marque.rt "A";
7 B = Point(-0.5,1);
8 D = Inversion(B,C);
9 drawoptions(withcolor DarkRed);
10 marque.rt "B"; marque.rt "D";
11
12 drawoptions(withcolor DarkBlue)
13 ;
14 E = Point(3,1);
15 CE = Cercle(E,0.5);
16 trace CE;
17 marque.rt "E";
18 iCE = Inversion(CE,C);
19 trace iCE;
20
21 drawoptions(withcolor DarkGreen
22 );
23 d = Droite((3,3),(4,2));
24 trace d;
25 Cd = Inversion(d,C);
26 trace Cd;
27 Fenetre(-1,-1,5.5,6);
endfig;
```



## 8 Annotations et labels

### 8.1 Signes

mp-geom2d fournit la macro suivante pour marquer un angle droit formé par trois points :

**SigneOrtho**( $\langle a \rangle, \langle b \rangle, \langle c \rangle, \langle x \rangle$ )

$\langle a \rangle, \langle b \rangle, \langle c \rangle$  : sont les trois points formant l'angle droit  $\widehat{ABC}$ ;

$\langle x \rangle$  : est la « taille » (numeric) du signe d'orthogonalité.

On peut aussi réaliser des marques entre deux points ou sur un segment, un vecteur, ou tout type d'objet. Pour cela, la première macro est la suivante :

### Marque(*a*, *b*, *n*)

- a* : premier **point** formant le segment à marquer ;
- b* : deuxième **point** formant le segment à marquer ;
- n* : est le type (**numeric**) de marque, il y en a quatre *n*= 1, 2, 3 ou 4.

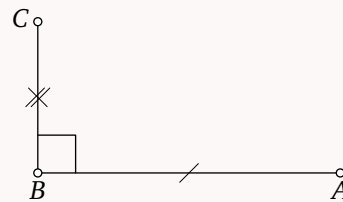
On peut aussi marquer n'importe quel type de trait avec la macro suivante :

### MarqueTrait(*a*, *n*)

- a* : est n'importe quel objet mp-geom2d ou même un **path** METAPOST ;
- n* : est le type (**numeric**) de marque, il y en a quatre *n*= 1, 2, 3 ou 4.

#### Exemple 27

```
1 input geom2d;
2 beginfig(1);
3 A = Point(4,0);
4 B = Point(0,0);
5 C = Point(0,2);
6 trace Segment(B,A);
7 trace Segment(B,C);
8 marque.bot "A";
9 marque.bot "B";
10 marque.lft "C";
11 trace SigneOrtho(A,B,C,0.5);
12 trace Marque(A,B,1);
13 trace MarqueTrait(Segment(B,C),3)
14 ;
15 endfig;
```



## 8.2 Labels

Comme le code produit avec mp-geom2d est un code METAPOST [5], on peut utiliser les outils classiques de labélisation. Cependant, pour permettre plus de flexibilité (et notamment une compatibilité avec Lua<sub>TEX</sub> et **luamplib** [1] ou **minim-mp** [4]), si le code est compilé avec METAPOST, alors le package **latexmp** [2] est chargé et fournit la commande `textext()`.

De plus, mp-geom2d fournit quelques commandes pour se faciliter la vie.

On peut marquer les points avec la commande suivante. Cette commande `pointe` le point (avec la commande `pointe`, voir section 4.3) et inscrit un label.

### marque.<place> "<nom>"

- place* : peut être les classiques placement de METAPOST : `top`, `bot`, `rt`, `lft`, `+urt`, `ulft`, `lrt`, `llft`.

**<nom>** : entre double quote, doit être un nom de variable de point. Le nom sera composé en mode mathématique (entre \$). Si le nom contient un `_`, tout ce qui sera après sera mis en indice (ex. `A_be` deviendra  $A_{be}$ ). Si le nom est `alpha`, `beta`, `gamma` ou `delta`, alors le résultat donnera la lettre grecque composée en mode mathématique.

`mp-geom2d` fournit aussi une adaptation au classique `label` de METAPOST.

`gddLabel.<place>( <matériel> , <point> )`

**<place>** : peut être les classiques placement de METAPOST : `top`, `bot`, `rt`, `lft`, `+urt`, `ulft`, `lrt`, `llft`.

**<matériel>** : classiquement ce qu'on donne à `label`, une chaîne de caractères, ou une `picture` (qui peut-être produite par exemple avec `btex ... etex` ou, puisque `latexmp` est chargé par `mp-geom2d`, `textext()`).

**<point>** : doit être un `point` de `mp-geom2d`.

On peut aussi étiquetter un `chemin`, une `courbe` ou un `path` avec la macro suivante :

`EtiquetteChemin.<place>( <matériel> , <chemin> , <position> )`

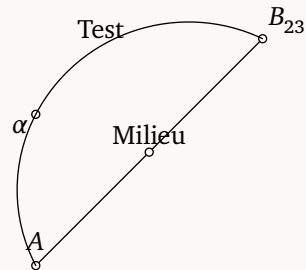
**<place>** : peut être les classiques placement de METAPOST : `top`, `bot`, `rt`, `lft`, `+urt`, `ulft`, `lrt`, `llft`.

**<matériel>** : classiquement ce qu'on donne à `label`, une chaîne de caractères, ou une `picture` (qui peut-être produite par exemple avec `btex ... etex` ou, puisque `latexmp` est chargé par `mp-geom2d`, `textext()`).

**<chemin>** : doit être un `chemin`, une `courbe`, ou un `path`.

### Exemple 28

```
1 input geom2d;
2 beginfig(1);
3 A = Point(0,0);
4 B_23 = Point(3,3);
5 alpha = Point(0,2);
6 trace Segment(A,B_23);
7 C = Milieu(A,B_23);
8 pointe C;
9 gddLabel.top(texttext("Milieu"),C)
  ;
10 marque.top "A";
11 marque.urrt "B_23";
12 marque.llft "alpha";
13 P = Chemin(Pt(A)..Pt(alpha)..Pt(
  B_23));
14 trace P;
15 EtiquetteChemin.top("Test",P,0.6)
  ;
16 endfig;
```



## 9 Repère

mp-geom2d fournit un ensemble de commandes permettant de faciliter le dessin de repère.

La commande principale est la définition du repère, c'est-à-dire la boîte dans laquelle le dessin sera représenté.

**Repere**( $\langle l \rangle$ ,  $\langle h \rangle$ ,  $\langle ox \rangle$ ,  $\langle oy \rangle$ ,  $\langle ux \rangle$ ,  $\langle uy \rangle$ )

$\langle l \rangle$  : **numeric**, largeur du repère (en unité **gddU**);

$\langle h \rangle$  : **numeric**, hauteur du repère (en unité **gddU**);

$\langle ox \rangle$  : **numeric**, distance (en unité **gddU**) de l'origine (**point** (0,0)) par rapport au bord gauche;

$\langle oy \rangle$  : **numeric**, distance (en unité **gddU**) de l'origine (**point** (0,0)) par rapport au bord bas;

$\langle ux \rangle$  : **numeric**, unité de l'axe ( $Ox$ ) (en unité **gddU**);

$\langle uy \rangle$  : **numeric**, unité de l'axe ( $Oy$ ) (en unité **gddU**).

Ce commande ne retourne, ni ne trace rien. Elle sert à spécifier quelques variables internes de définition du dessin. Elle modifie d'ailleurs le comportement de la macro **gddEnPlace** (voir page 3). Noter que cette commande impose le fait que l'origine (c'est-à-dire le **point** (0,0)) soit à l'intérieur du repère.

Cette macro s'accompagne de deux autres, elles aussi *silencieuses*, ne servant qu'à :

— sauvegarder la **picture** courante;

— construire une **picture** avec le contenu se trouvant entre **Debut** et **Fin**;

- rogner (avec `clip`) la `picture` aux cadre du repère construit ;
- ajouter la `picture` courant à celle sauvegarder ;
- enfin rétablir le fonctionnement de `gddEnPlace` comme avant l'utilisation de `Repere`

Ces deux commandes sont appelées `Debut` et `Fin`.

`Debut`

`Fin`

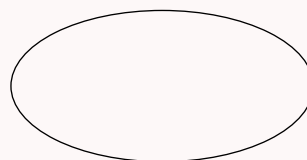
Ainsi, il est très simple d'illustrer le fait qu'à un changement de base orthogonale prêt, un cercle est une ellipse.

#### Exemple 29

```

1 input geom2d;
2 beginfig(1);
3 A = Point(1,1);
4 CA = Cercle(A,2);
5 Repere(5,5,1,1,1,0.5);
6 Debut;
7 trace CA;
8 Fin;
9 endfig;

```



On dispose de deux commandes pour tracer les axes du repère. La première trace les axes passant par l'origine.

`Axes`

(*trace un ensemble d'éléments sur le repère*)

Cette commande inscrit aussi les labels des axes des abscisses et des ordonnées qui sont stockées dans les variables globales dédiées suivantes :

`gddXlabel`

(*string, valeur par défaut "\$x\$"*)

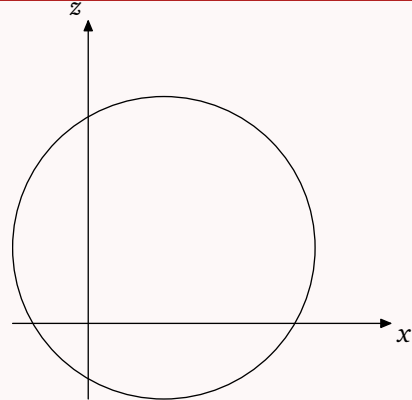
`gddYlabel`

(*string, valeur par défaut "\$y\$"*)

Attention, ces commandes doivent s'utiliser avant l'appel à `Debut`. L'exemple suivant permet d'illustrer le tracé des axes.

### Exemple 30

```
1 input geom2d;
2 beginfig(1);
3 A = Point(1,1);
4 CA = Cercle(A,2);
5 Repere(5,5,1,1,1,1);
6 gddYlabel:="$z$";
7 Axes;
8 Debut;
9 trace CA;
10 Fin;
11 endfig;
```



On pourra aussi définir un repère en utilisant une syntaxe permettant de spécifier les abscisses et les ordonnées extrémales.

**RepereMinMax**( $\langle xmin \rangle$ ,  $\langle xmax \rangle$ ,  $\langle ymin \rangle$ ,  $\langle ymax \rangle$ ,  $\langle ux \rangle$ ,  $\langle uy \rangle$ )

$\langle xmin \rangle$  : **numeric**, abscisse minimum (en unité **gddU**).

$\langle xmax \rangle$  : **numeric**, abscisse maximum (en unité **gddU**).

$\langle ymin \rangle$  : **numeric**, ordonnée minimum (en unité **gddU**).

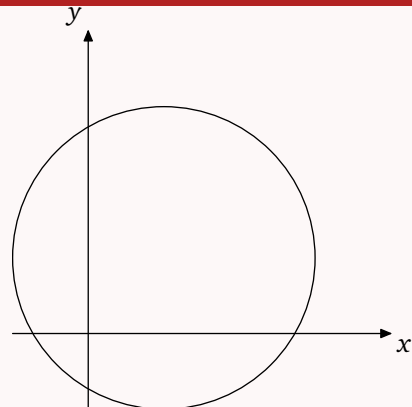
$\langle ymax \rangle$  : **numeric**, ordonnée maximum (en unité **gddU**).

$\langle ux \rangle$  : **numeric**, unité de l'axe ( $Ox$ ) (en unité **gddU**).

$\langle uy \rangle$  : **numeric**, unité de l'axe ( $Oy$ ) (en unité **gddU**).

### Exemple 31

```
1 input geom2d;
2 beginfig(1);
3 A = Point(1,1);
4 CA = Cercle(A,2);
5 RepereMinMax(-1,4,-1,4,1,1);
6 Axes;
7 Debut;
8 trace CA;
9 Fin;
10 endfig;
```



mp-geom2d fournit aussi de quoi tracer les axes sur le bord du cadre avec la commande suivante.

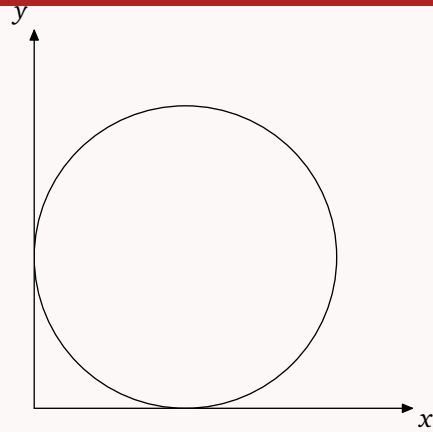


## AxesBords

(trace un ensemble d'éléments sur le repère)

### Exemple 32

```
1 input geom2d;  
2 beginfig(1);  
3 A = Point(1,1);  
4 CA = Cercle(A,2);  
5 RepereMinMax(-1,4,-1,4,1,1);  
6 AxesBords;  
7 Debut;  
8 trace CA;  
9 Fin;  
10 endfig;
```



Les commandes suivantes permettent de graduer les axes (classiques ou sur le bord). Là encore, ce sont des commandes qui n'ont pas d'arguments ni ne retournent rien et qui tracent.

## Graduations

(trace un ensemble d'éléments sur le repère)

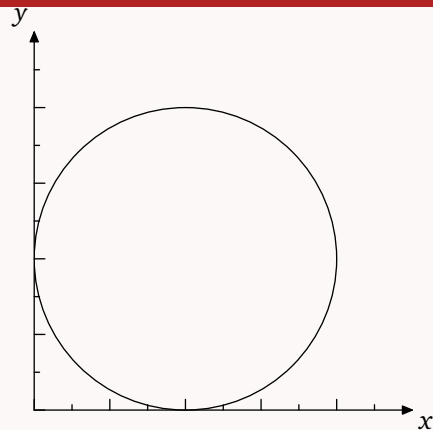
## GraduationsBords

(trace un ensemble d'éléments sur le repère)

On utilisera la version en cohérence avec les axes choisis.

### Exemple 33

```
1 input geom2d;  
2 beginfig(1);  
3 A = Point(1,1);  
4 CA = Cercle(A,2);  
5 RepereMinMax(-1,4,-1,4,1,1);  
6 AxesBords;  
7 GraduationsBords;  
8 Debut;  
9 trace CA;  
10 Fin;  
11 endfig;
```



Si on désire marquer les unités, mp-geom2d propose la macro suivante (qui n'est utilisable que si l'on ne place pas les axes sur le bord).

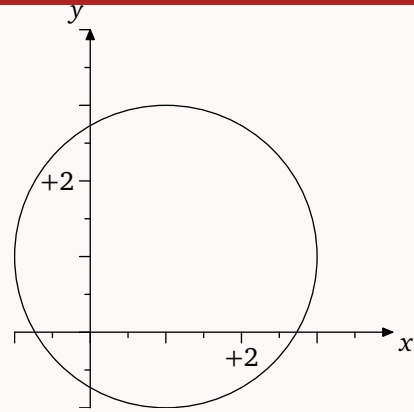
`Unites(<unité>)`

(trace un ensemble d'éléments sur le repère)

`<unité>` : *numeric*, valeur à inscrire sur les deux axes.

### Exemple 34

```
1 input geom2d;
2 beginfig(1);
3 A = Point(1,1);
4 CA = Cercle(A,2);
5 RepereMinMax(-1,4,-1,4,1,1);
6 Axes;
7 Debut;
8 Graduations;
9 Unites(2);
10 trace CA;
11 Fin;
12 endfig;
```



On peut aussi ajouter une grille sur notre repère avec la macro suivante.

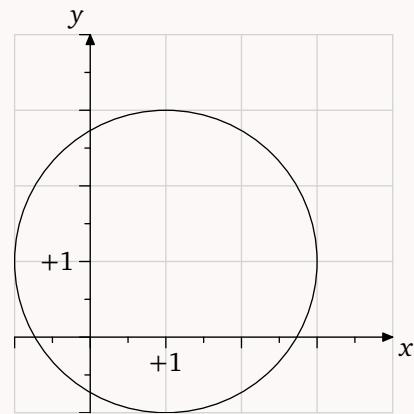
`GrilleRepere`

(trace un ensemble d'éléments sur le repère)

Dans l'exemple suivant, on règle la couleur de la grille avec la commande MetaPost `drawoptions`.

### Exemple 35

```
1 input geom2d;
2 beginfig(1);
3 A = Point(1,1);
4 CA = Cercle(A,2);
5 RepereMinMax(-1,4,-1,4,1,1);
6 drawoptions(withcolor LightGrey);
7 GrilleRepere;
8 drawoptions();
9 Axes;
10 Debut;
11 Graduations;
12 Unites(1);
13 trace CA;
14 Fin;
15 endfig;
```



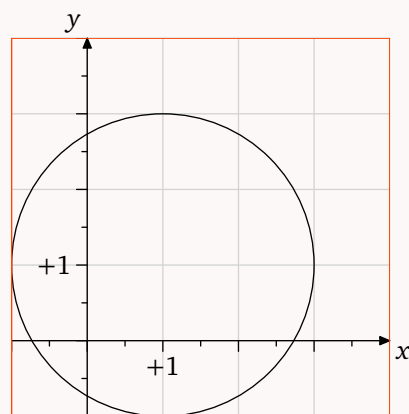
On pourra aussi ajouter un cadre au repère avec la macro suivante.

`CadreRepere` → `path`

Dans l'exemple suivant, on règle la couleur de la grille avec la commande MetaPost `drawoptions`.

### Exemple 36

```
1 input geom2d;
2 beginfig(1);
3 A = Point(1,1);
4 CA = Cercle(A,2);
5 RepereMinMax(-1,4,-1,4,1,1);
6 drawoptions(withcolor LightGrey);
7 GrilleRepere;
8 drawoptions();
9 Axes;
10 Debut;
11 Graduations;
12 Unites(1);
13 trace CA;
14 trace CadreRepere withcolor
    OrangeRed;
15 Fin;
16 endfig;
```



## 10 Quelques constantes et fonctions mathématiques

`mp-geom2d` définit deux constantes mathématiques `Pi` et `_E` pour les constantes  $\pi \simeq 3.14159265$  et  $e = 2.71828183$ .

De plus, le package définit quelques fonctions mathématiques de la variable réelle :

`sin( $\langle x \rangle$ )`

`cos( $\langle x \rangle$ )`

`tan( $\langle x \rangle$ )`

`exp( $\langle x \rangle$ )`

`ln( $\langle x \rangle$ )`

`ch( $\langle x \rangle$ )`

`sh( $\langle x \rangle$ )`

`th( $\langle x \rangle$ )`

`arcsin( $\langle x \rangle$ )`

`arccos( $\langle x \rangle$ )`

`arctan( $\langle x \rangle$ )`

## 11 Représentation de courbes et de fonctions

mp-geom2d fournit aussi quelques macros facilitant la représentation simple de courbes et de fonctions mathématiques.

### 11.1 Fonction de la variable réelle

Pour représenter une fonction de la variable réelle, on utilisera la macro suivante :

**Representation**(*fonction*)(*ti*),(*tf*),(*n*) → path

**<fonction>** : est une macro METAPOST qui définit la fonction mathématique d'une variable réelle que l'on souhaite représenter ;

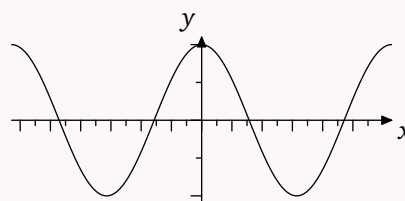
**<ti>** : est la valeur (**numeric**) initiale de la variable à partir de laquelle on souhaite construire la représentation de la fonction ;

**<tf>** : est la valeur (**numeric**) finale de la variable jusqu'à laquelle on souhaite construire la représentation de la fonction ;

**<n>** : est le nombre de pas (**numeric**) utilisé pour la discrétisation de la représentation.

#### Exemple 37

```
1 input geom2d;
2 beginfig(1);
3 RepereMinMax(-2Pi,2Pi
4   , -1.1,1.1,0.4,1);
5 Axes;
6 Debut;
7 Graduations;
8 trace Representation(cos,-2Pi,2
9   Pi,100);
10 Fin;
11 endfig;
```



### 11.2 Courbe paramétrique

Pour représenter une courbe plane définie par deux fonctions décrivant l'abscisse et l'ordonnée en fonction d'un paramètre, on utilisera la macro suivante :

**Courbe**(*fct\_abscisse*)(*fct\_ordonnee*)(*ti*),(*tf*),(*n*) → path

**<fct\_abscisse>** : est une macro METAPOST qui définit la fonction mathématique d'une variable réelle décrivant l'évolution de l'abscisse des points de la courbe que l'on souhaite représenter ;

**<fct\_ordonnee>** : est une macro METAPOST qui définit la fonction mathématique d'une variable réelle décrivant l'évolution de l'ordonnée des points de la courbe que l'on souhaite représenter ;

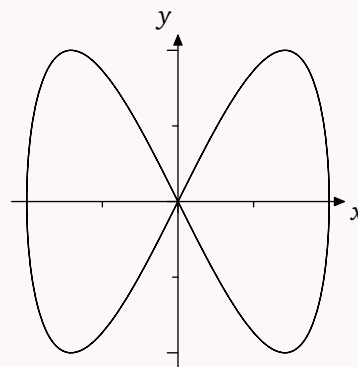
$\langle ti \rangle$  : est la valeur (**numeric**) initiale de la variable à partir de laquelle on souhaite construire la représentation de la fonction ;

$\langle tf \rangle$  : est la valeur (**numeric**) finale de la variable jusqu'à laquelle on souhaite construire la représentation de la fonction ;

$\langle n \rangle$  : est le nombre de pas (**numeric**) utilisé pour la discrétisation de la représentation.

### Exemple 38

```
1 input geom2d;
2 vardef f_a(expr t)= cos(t) enddef
  ;
3 vardef f_o(expr t)= sin(2*t)
  enddef;
4
5 beginfig(1);
6 RepereMinMax
  (-1.1,1.1,-1.1,1.1,2,2);
7 Axes;
8 Debut;
9 Graduations;
10 trace Courbe(f_a,f_o,-2Pi,2Pi
  ,100);
11 Fin;
12 endfig;
```



Pour représenter une courbe plane définie par une fonction décrivant les coordonnées polaires en fonction d'un paramètre, on utilisera la macro suivante :

**CourbeEnPolaires**( $\langle fonction \rangle$ )( $\langle ti \rangle$ , $\langle tf \rangle$ , $\langle n \rangle$ )  $\rightarrow$  path

$\langle fonction \rangle$  : est une macro METAPOST qui définit la fonction mathématique d'une variable réelle décrivant l'évolution des coordonnées polaires des points de la courbe que l'on souhaite représenter ;

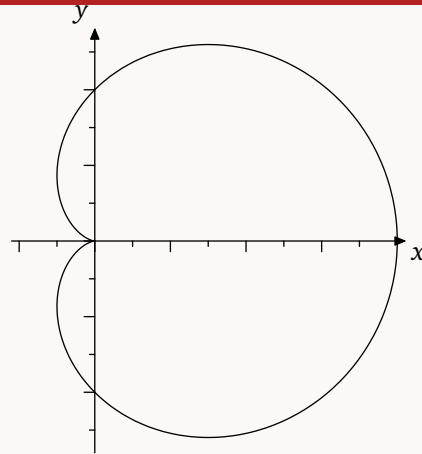
$\langle ti \rangle$  : est la valeur (**numeric**) initiale de la variable à partir de laquelle on souhaite construire la représentation de la fonction ;

$\langle tf \rangle$  : est la valeur (**numeric**) finale de la variable jusqu'à laquelle on souhaite construire la représentation de la fonction ;

$\langle n \rangle$  : est le nombre de pas (**numeric**) utilisé pour la discrétisation de la représentation.

### Exemple 39

```
1 input geom2d;
2 a := 2;
3 vardef cardioide(expr t)= a*(1+
   cos(t)) enddef;
4 beginfig(1);
5 ReperMinMax
   (-1.1,4.1,-2.8,2.8,1,1);
6 Axes;
7 Debut;
8 Graduations;
9 trace CourbeEnPolaires(
   cardioide,0,2Pi,100);
10 Fin;
11 endfig;
```



## 11.3 Champs de vecteurs

mp-geom2d fournit des macros pour la représentation des champs de vecteurs.

Tout d'abord, on peut tracer des champs de vecteurs associée à une équation différentielle du premier ordre pour une fonction  $y$  de la variable  $x$  :

$$y' = F(x, y).$$

La macro associée est la suivante :

**ChampVecteurs**(*fonction*)(*x*), (*y*), (*px*), (*py*), (*dx*), (*couleur*) → **path**

**fonction** : est une macro METAPOST qui définit une fonction mathématique de  $\mathbf{R}^2$  dans  $\mathbf{R}$  ;

**x** : est une valeur (**numeric**), en unité **gddU**, qui permet de décaler la grille des vecteurs suivant la direction  $x$ , les points étant tous les  $\langle x \rangle + i\langle px \rangle$  pour  $i$  entier ;

**y** : est une valeur (**numeric**), en unité **gddU**, qui permet de décaler la grille des vecteurs suivant la direction  $y$ , les points étant tous les  $\langle y \rangle + i\langle py \rangle$  pour  $i$  entier ;

**px** : est la valeur (**numeric**), en unité **gddU**, du pas de la grille suivant l'axe  $x$  pour la représentation des vecteurs ;

**py** : est la valeur (**numeric**), en unité **gddU**, du pas de la grille suivant l'axe  $y$  pour la représentation des vecteurs ;

**dx** : est la norme (**numeric**), en unité **gddU**, des vecteurs du champ de vecteur ;

**couleur** : est la couleur (**color**) utilisée pour tracer les vecteurs.

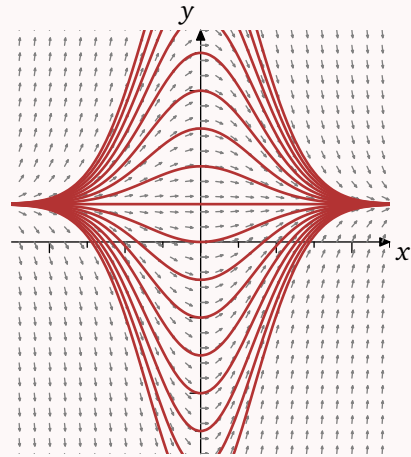
Comme cette macro utilise la macro **drawarrow** de METAPOST, il faudra jouer avec le paramètre **ahlength** pour régler la taille du triangle des flèches (voir [5]).

### Exemple 40

```

1 input geom2d;
2 vardef F(expr x,y) = x - 2 * x *
  y enddef;
3 vardef f(expr x) = 1/2 + a * exp
  (- x*x) enddef;
4 beginfig(1);
5 RepereMinMax
  (-2.5,2.5,-2.8,2.8,1,1);
6 Axes;
7 Debut;
8 Graduations;
9 ahlength := 1;
10 ChampVecteurs(F
  ,0,0,0.2,0.2,0.1,0.5white);
11 % Courbes intégrales
12 for n = 0 upto 16:
13   a := (n/2) - 4;
14   trace Representation(f
  ,-2.5,2.5,50)
15   avecCrayon(1,(0.7,0.2,0.2));
16 endfor
17 Fin;
18 endfig;

```



Sur le même modèle, on peut tracer des champs de vecteurs d'une fonction de  $\mathbf{R}^2$  dans  $\mathbf{R}^2$ . La macro associée est la suivante :

**ChampVecteursDD**(*fonction*)(*x* , *y* , *px* , *py* , *dx* , *couleur*) → *path*

*fonction* : est une macro METAPOST qui définit une fonction mathématique de  $\mathbf{R}^2$  dans  $\mathbf{R}^2$  (donc qui retourne une *pair*) ;

*x* : est une valeur (*numeric*), en unité *gddU*, qui permet de décaler la grille des vecteurs suivant la direction *x*, les points étant tous les  $\langle x \rangle + i \langle px \rangle$  pour *i* entier ;

*y* : est une valeur (*numeric*), en unité *gddU*, qui permet de décaler la grille des vecteurs suivant la direction *y*, les points étant tous les  $\langle y \rangle + i \langle py \rangle$  pour *i* entier ;

*px* : est la valeur (*numeric*), en unité *gddU*, du pas de la grille suivant l'axe *x* pour la représentation des vecteurs ;

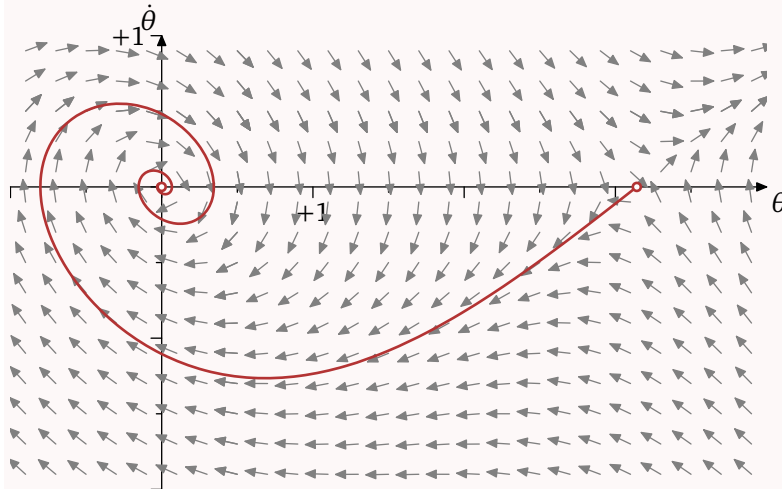
*py* : est la valeur (*numeric*), en unité *gddU*, du pas de la grille suivant l'axe *y* pour la représentation des vecteurs ;

*dx* : est la norme (*numeric*), en unité *gddU*, des vecteurs du champ de vecteur ;

*couleur* : est la couleur (*color*) utilisée pour tracer les vecteurs.

### Exemple 41

```
1 input geom2d;
2 gddXlabel := "\theta";
3 gddYlabel := "\dot{\theta}";
4 numeric b,c;
5 b:=0.5;
6 c=1.0;
7
8 vardef F(expr x,y) = (y,-b*y-c*sin(x)) enddef;
9
10 beginfig(1);
11 Repere(10,6,2,4,2,2);
12 Axes;
13 Unites(1);
14 Debut;
15 Graduations;
16 trajectoire := CourbeDat("solution0",0);
17 ChampVecteursDD(F,0.5,0.5,0.2,0.2,0.15,0.5white);
18 trace trajectoire avecCrayon(1,(0.7,0.2,0.2));
19 pointe Point(0,0) avecCrayon(1,(0.7,0.2,0.2));
20 pointe Point(3.1415,0) avecCrayon(1,(0.7,0.2,0.2));
21 Fin;
22 endfig;
```





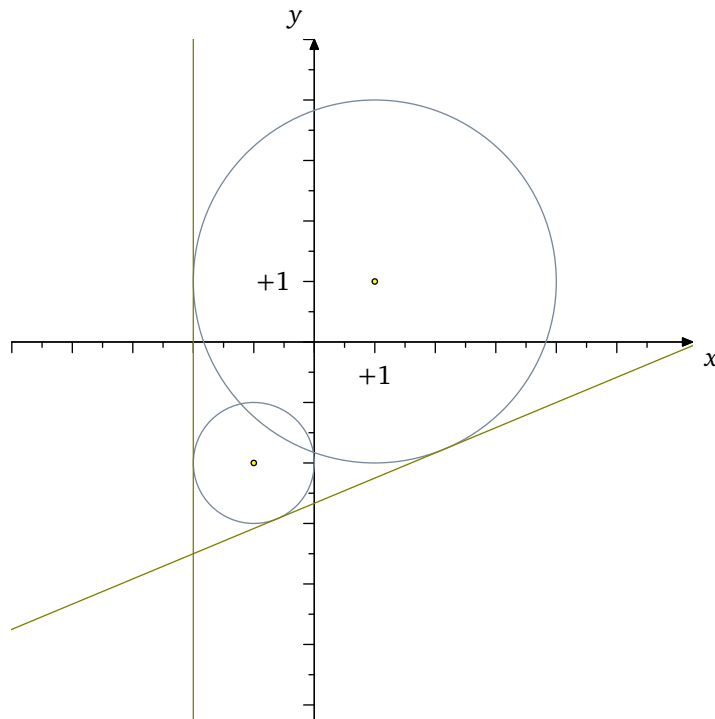
## 12 Couleurs svgnames

 AliceBlue	 AntiqueWhite	 Aqua	 Aquamarine
 Azure	 Beige	 Bisque	 Black
 BlanchedAlmond	 Blue	 BlueViolet	 Brown
 BurlyWood	 CadetBlue	 Chartreuse	 Chocolate
 Coral	 CornflowerBlue	 Cornsilk	 Crimson
 Cyan	 DarkBlue	 DarkCyan	 DarkGoldenrod
 DarkGray	 DarkGreen	 DarkGrey	 DarkKhaki
 DarkMagenta	 DarkOliveGreen	 DarkOrange	 DarkOrchid
 DarkRed	 DarkSalmon	 DarkSeaGreen	 DarkSlateBlue
 DarkSlateGray	 DarkSlateGrey	 DarkTurquoise	 DarkViolet
 DeepPink	 DeepSkyBlue	 DimGray	 DimGrey
 DodgerBlue	 FireBrick	 FloralWhite	 ForestGreen
 Fuchsia	 Gainsboro	 GhostWhite	 Gold
 Goldenrod	 Gray	 Green	 GreenYellow
 Grey	 Honeydew	 HotPink	 IndianRed
 Indigo	 Ivory	 Khaki	 Lavender
 LavenderBlush	 LawnGreen	 LemonChiffon	 LightBlue
 LightCoral	 LightCyan	 LightGoldenrod	 LightGoldenrodYellow
 LightGray	 LightGreen	 LightGrey	 LightPink
 LightSalmon	 LightSeaGreen	 LightSkyBlue	 LightSlateBlue
 LightSlateGray	 LightSlateGrey	 LightSteelBlue	 LightYellow
 Lime	 LimeGreen	 Linen	 Magenta
 Maroon	 MediumAquamarine	 MediumBlue	 MediumOrchid
 MediumPurple	 MediumSeaGreen	 MediumSlateBlue	 MediumSpringGreen
 MediumTurquoise	 MediumVioletRed	 MidnightBlue	 MintCream
 MistyRose	 Moccasin	 NavajoWhite	 Navy
 NavyBlue	 OldLace	 Olive	 OliveDrab
 Orange	 OrangeRed	 Orchid	 PaleGoldenrod
 PaleGreen	 PaleTurquoise	 PaleVioletRed	 PapayaWhip
 PeachPuff	 Peru	 Pink	 Plum
 PowderBlue	 Purple	 Red	 RosyBrown
 RoyalBlue	 SaddleBrown	 Salmon	 SandyBrown
 SeaGreen	 Seashell	 Sienna	 Silver
 SkyBlue	 SlateBlue	 SlateGray	 SlateGrey
 Snow	 SpringGreen	 SteelBlue	 Tan
 Teal	 Thistle	 Tomato	 Turquoise
 Violet	 VioletRed	 Wheat	 White
 WhiteSmoke	 Yellow	 YellowGreen	



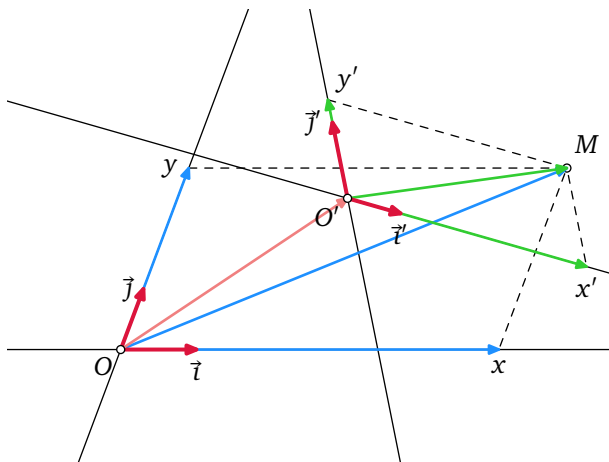
## 13 Galerie

### 13.1 Repère et tangentes extérieures



```
1 input geom2d;
2
3 labeloffset := 6;
4 gddTaillePoint := 2;
5 gddCouleurPoint := Yellow;
6
7 beginfig(1);
8 Repere(9,9,4,5,0.8,0.8);
9 Axes;
10 Debut;
11 Axes;
12 Graduations; Unites(1);
13
14 C1 = Cercle((1,1),3);
15 C2 = Cercle((-1,-2),1);
16
17 T1 = TangenteCommuneExterieur(C1,C2);
18 T2 = TangenteCommuneExterieur(C2,C1);
19
20 drawoptions(withcolor LightSlateGrey);
21 trace C1;
22 trace C2;
23
24 drawoptions(withcolor Olive);
25 trace T1;
26 trace T2;
27
28 drawoptions();
29 pointe Point(1,1);
30 pointe Point(-1,-2);
31
32 Fin;
33 endfig;
34 end
```

## 13.2 Vecteur dans un repère



```

1 input geom2d;
2
3 labeloffset := 4;
4 gddU:=1.cm;
5
6 beginfig(1);
7 O = Point(0,0);
8 I = Point(1,0);
9 J = Point(0.3,0.8);
10 M = PointDansRepere(5,3,0,I,J);
11 H = PointDansRepere(5,0,0,I,J);
12 K = PointDansRepere(0,3,0,I,J);
13 O' = Point(3,2);
14 I' = Point(3.7,1.8);
15 J' = Point(2.8,3);
16
17 pair Mt;
18 Mt = CoordonneesRepere(M,O',I',J');
19 H' = PointDansRepere(xpart Mt,0,0',I',J');

```

```

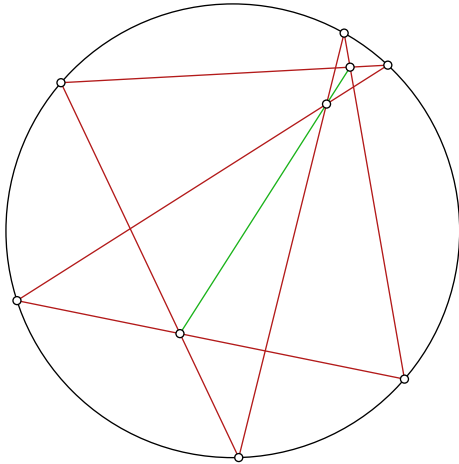
20 K' = PointDansRepere(0,ypart Mt,O',I',J');
21
22 Repere(8,6,1.5,1.5,1,1);
23 Debut;
24 trace Droite(O,I);
25 trace Droite(O,J);
26 trace Droite(O',I');
27 trace Droite(O',J');
28
29 trace Pt(K)--Pt(M)--Pt(H) dashed evenly;
30 trace Pt(K')--Pt(M)--Pt(H') dashed evenly;
31
32 marque.urt "M";
33
34 drawoptions(withpen pencircle scaled 1pt withcolor DodgerBlue);
35 fleche Segment(O,K);
36 fleche Segment(O,H);
37 fleche Segment(O,M);
38
39 drawoptions(withpen pencircle scaled 1pt withcolor LimeGreen);
40 fleche Segment(O',M);
41 fleche Segment(O',H');
42 fleche Segment(O',K');
43
44 drawoptions(withpen pencircle scaled 1.5pt withcolor Crimson);
45 fleche Segment(O,I);
46 fleche Segment(O,J);
47 fleche Segment(O',I');
48 fleche Segment(O',J');
49
50 drawoptions(withpen pencircle scaled 1pt withcolor LightCoral);
51 fleche Segment(O,O');
52
53 drawoptions();
54 marque.llft "O";
55 marque.llft "O'";
56
57 label.bot(texttext("\(x\)"), PtR(H));
58 label.lft(texttext("\(y\)"), PtR(K));
59 label.bot(texttext("\(\vec{\imath}\)"), PtR(I));
60 label.lft(texttext("\(\vec{j}\)"), PtR(J));

```

```
61 label.bot(texttext("\(x'\)"), PtR(H'));
62 label.urt(texttext("\(y'\)"), PtR(K'));
63 label.bot(texttext("\(\vec\imath'\)"), PtR(I'));
64 label.lft(texttext("\(\vec\jmath'\)"), PtR(J'));
```

```
66
67 Fin;
68 endfig;
69 end
```

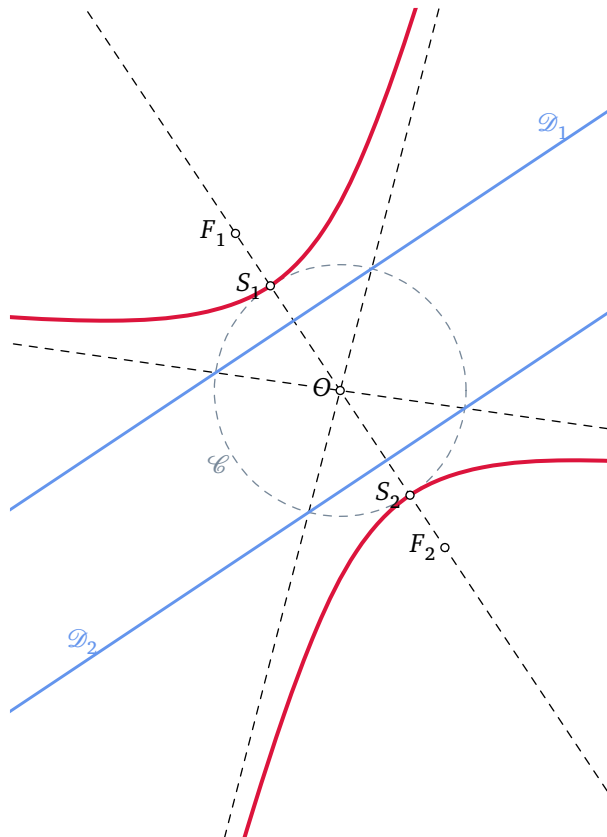
### 13.3 Théorème de Pascal



```
1 input geom2d;  
2  
3 beginfig(1);  
4 C = Cercle(origine,3);
```

```
5 for i:=1 upto 6:  
6   rd := uniformdeviate(1.0/6)+(i-1)/6;  
7   P[i] := PointDe(C,rd);  
8 endfor;  
9 D1 = Droite(P1,P3); S1 = Segment(P1,P3);  
10 D2 = Droite(P3,P5); S2 = Segment(P3,P5);  
11 D3 = Droite(P6,P2); S3 = Segment(P6,P2);  
12 D4 = Droite(P4,P6); S4 = Segment(P4,P6);  
13 D5 = Droite(P5,P2); S5 = Segment(P5,P2);  
14 D6 = Droite(P1,P4); S6 = Segment(P1,P4);  
15 I1 = IntersectionDroites(D1,D3);  
16 I2 = IntersectionDroites(D2,D4);  
17 I3 = IntersectionDroites(D5,D6);  
18 PL = Segment(I1,I2);  
19 trace C;  
20 drawoptions(withcolor (0.7,0.1,0.1));  
21 trace S1; trace S2; trace S3; trace S4; trace S5; trace S6;  
22 drawoptions(withcolor (0.1,0.7,0.1));  
23 trace PL;  
24 drawoptions();  
25 pointe(P1); pointe P2; pointe P3; pointe P4; pointe P5; pointe P6;  
26 pointe I1; pointe I2; pointe I3;  
27 endfig;  
28 end.
```

## 13.4 Hyperbole



```

1 input geom2d;
2
3 beginfig(1);

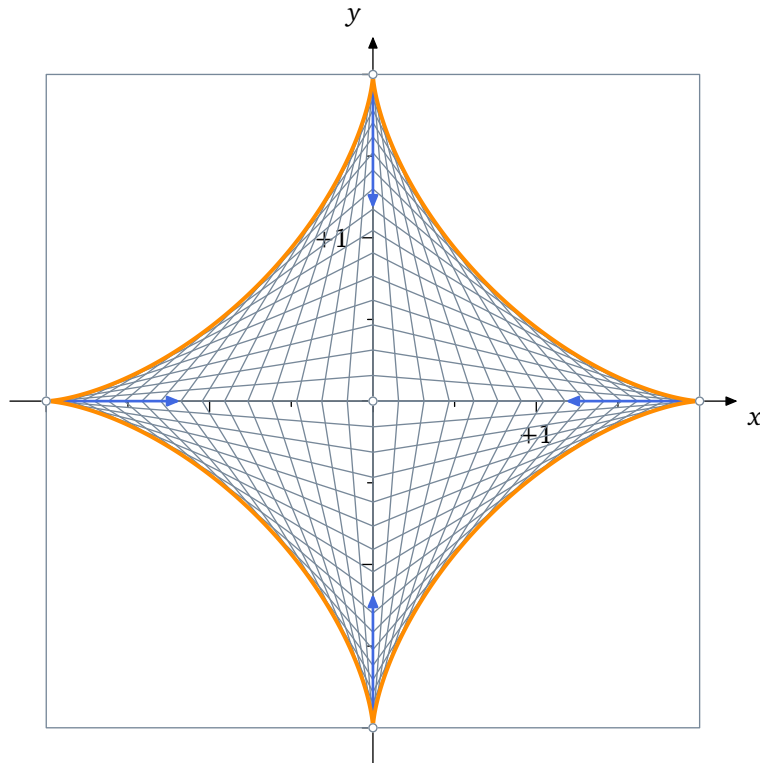
```

```

4 A = Point(0,0);
5 B = Point(3,2);
6 AB = Droite(A,B);
7 F_1 = Point(-1,1);
8 Hyper = HyperboleFD(F_1,AB,1.5);
9 O = Centre(Hyper);
10 F_2 = Foyer(Hyper,2);
11 Axe = Droite(F_1,F_2);
12 S_1 = Sommet(Hyper,1);
13 S_2 = Sommet(Hyper,2);
14
15 trace Axe dashed evenly;
16 C = CerclePrincipale(Hyper) ;
17 trace C avecCrayon(0.5,LightSlateGray) dashed evenly;
18 A_1 = AsymptoteHyperbole(Hyper,1);
19 A_2 = AsymptoteHyperbole(Hyper,2);
20 trace A_1 dashed evenly;
21 trace A_2 dashed evenly;
22
23 D_1 = Directrice(Hyper,1);
24 D_2 = Directrice(Hyper,2);
25
26 trace D_1 avecCrayon(1.1,CornflowerBlue);
27 trace D_2 avecCrayon(1.1,CornflowerBlue);
28 trace DemiHyperbole(Hyper,1) avecCrayon(1.5,Crimson);
29 trace DemiHyperbole(Hyper,2) avecCrayon(1.5,Crimson);
30
31 marque.lft "O";
32 marque.lft "S_1";
33 marque.lft "S_2";
34 marque.lft "F_1";
35 marque.lft "F_2";
36 label.lft(texttext("\(\mathcal{C}\)"),Pt(PointDe(C,0.6)) gddEnPlace)
   withcolor LightSlateGray;
37 label.top(texttext("\(\mathcal{D}_1\)",Pt(PointDe(D_1,0.47))
   gddEnPlace) withcolor CornflowerBlue;
38 label.top(texttext("\(\mathcal{D}_2\)",Pt(PointDe(D_2,0.46))
   gddEnPlace) withcolor CornflowerBlue;
39 Fenetre(-4,-7,4,4);
40 endfig;
41 end.

```

### 13.5 Astroïde comme enveloppe de droites

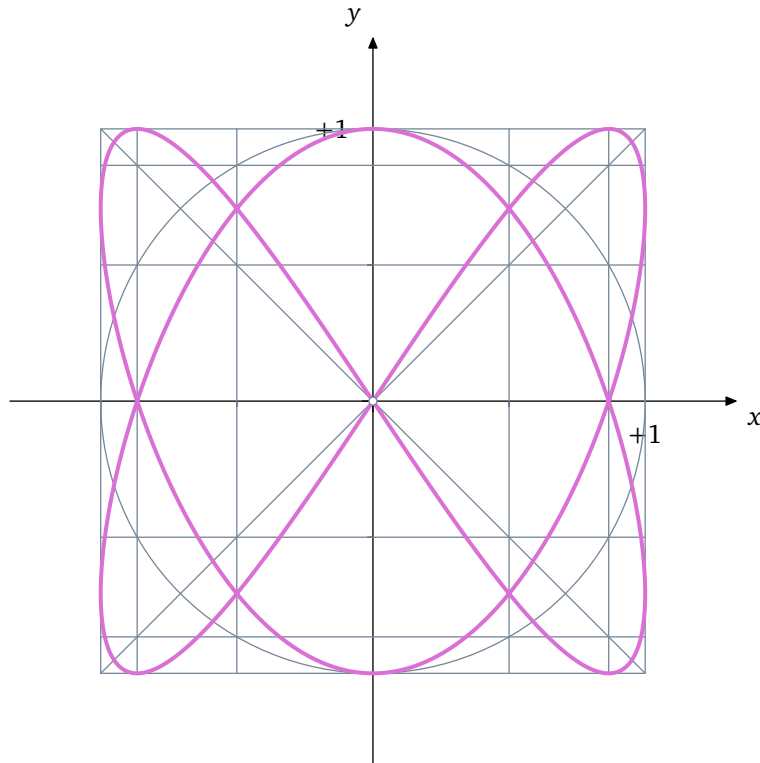


```
1 input geom2d;
2
```

```
3
4 labeloffset := 6;
5 gddU:=1.2cm;
6
7 vardef f(expr t) = 2*cos(t)*cos(t)*cos(t) enddef;
8 vardef g(expr t) = 2*sin(t)*sin(t)*sin(t) enddef;
9
10
11 beginfig(1);
12 Repere(8,8,4,4,1.8,1.8);
13 Axes;
14 Debut;
15 Graduations; Unites(1);
16 drawoptions(withpen pencircle scaled 0.5 withcolor LightSlateGrey);
17 trace ((-2,-2)--(-2,2)--(2,2)--(2,-2)--cycle);
18
19 nb = 80;
20 pas = 2*Pi/nb;
21 for i=0 upto nb: trace (0,2*sin(i*pas))--(2*cos(i*pas),0); endfor;
22
23 fleche Segment((2,0),(1.2,0)) avecCrayon(1,RoyalBlue);
24 fleche Segment((-2,0),(-1.2,0)) avecCrayon(1,RoyalBlue);
25 fleche Segment((0,2),(0,1.2)) avecCrayon(1,RoyalBlue);
26 fleche Segment((0,-2),(0,-1.2)) avecCrayon(1,RoyalBlue);
27
28 trace Courbe(f,g,0,2*Pi,500) avecCrayon(1.5,DarkOrange);
29
30 pointe Point(0,0);
31 pointe Point(2,0);
32 pointe Point(0,2);
33 pointe Point(0,-2);
34 pointe Point(-2,0);
35
36 Fin;
37 endfig;
38 end
```



## 13.6 Courbe de Lissajous

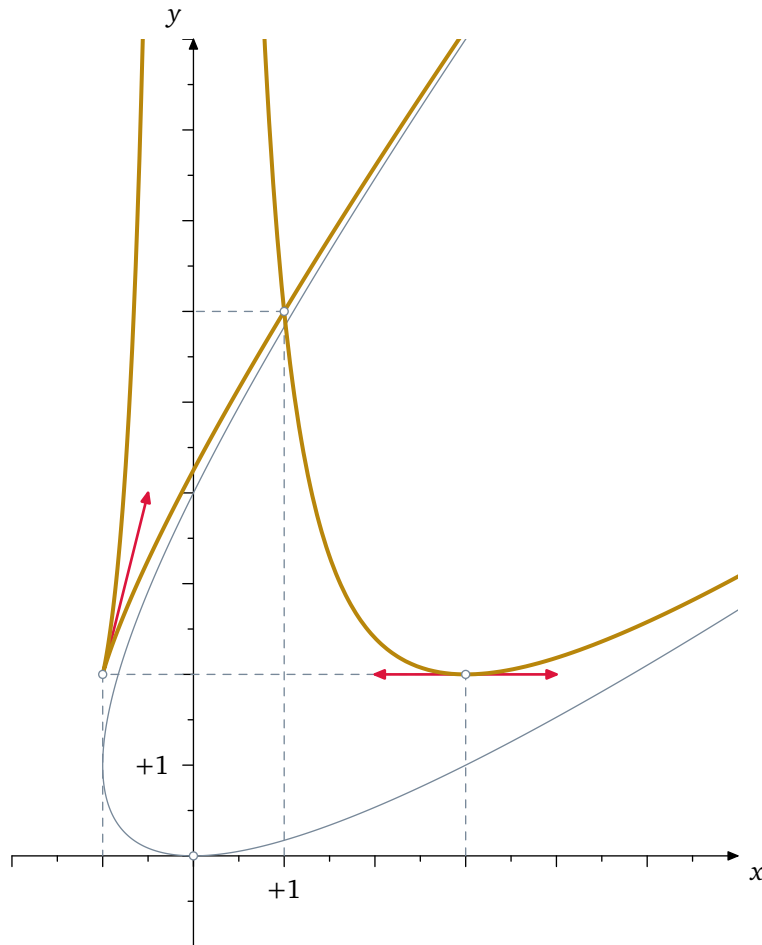


```

1 input geom2d;
2
3 labeloffset := 6;
4 gddU:=1.2cm;
5
6 vardef f(expr t) = sin(2*t+Pi/3) enddef;
7 vardef g(expr t) = sin(3*t) enddef;
8
9 beginfig(1);
10 Repere(8,8,4,4,3,3);
11 Axes;
12 Debut;
13 Graduations; Unites(1);
14
15 drawoptions(withcolor LightSlateGrey);
16 trace ((-1,-1)--(-1,1)--(1,1)--(1,-1)--cycle);
17 trace ((-1,-1)--(1,1));
18 trace (1,-1)--(-1,1);
19 trace Cercle(origine,1);
20 trace ((-1,0.5)--(1,0.5));
21 trace ((-1,-0.5)--(1,-0.5));
22 trace ((0.5,-1)--(0.5,1));
23 trace ((-0.5,-1)--(-0.5,1));
24 trace ((-1,sqrt(3)/2)--(1,sqrt(3)/2));
25 trace ((-1,-sqrt(3)/2)--(1,-sqrt(3)/2));
26 trace ((sqrt(3)/2,-1)--(sqrt(3)/2,1));
27 trace ((-sqrt(3)/2,-1)--(-sqrt(3)/2,1));
28
29 trace Courbe(f,g,0,2*Pi,500) avecCrayon(1.5,Orchid);
30
31 pointe Point(0,0);
32
33 Fin;
34 endfig;
35 end

```

## 13.7 Étude de fonctions

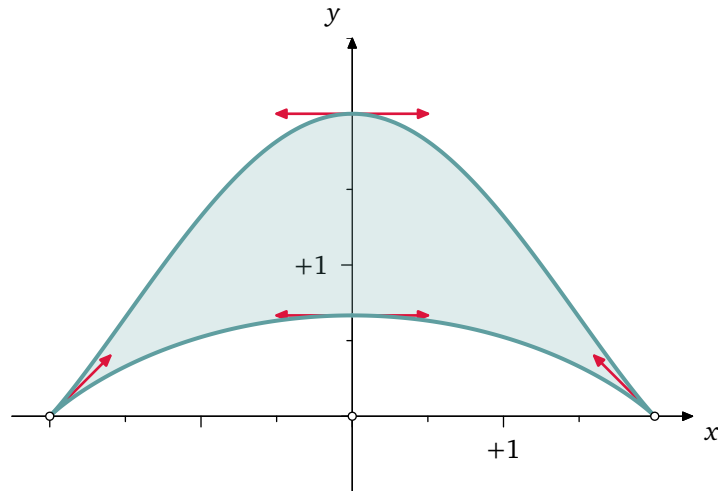


```

1 input geom2d;
2
3 labeloffset := 6;
4 gddU:=1.2cm;
5
6 vardef f(expr t) = t*t-2*t enddef;
7 vardef g(expr t) = t*t + 1/(t*t) enddef;
8 vardef fp(expr t) = 2*t+t*t enddef;
9 vardef gp(expr t) = t*t enddef;
10
11 beginfig(1);
12
13 Repere(8,10,2,1,1,1);
14 Axes;
15 Debut;
16 Graduations; Unites(1);
17
18 drawoptions(withcolor LightSlateGrey);
19 trace (3,0)--(3,2)--(-1,2)--(-1,0) dashed evenly;
20 trace (1,0)--(1,6)--(0,6) dashed evenly;
21 trace Courbe(fp,gp,-5,5,500);
22
23 fleche Segment((-1,2),(-1,2)+(0.5,2)) avecCrayon(1,Crimson);
24 fleche Segment((3,2),(4,2)) avecCrayon(1,Crimson);
25 fleche Segment((3,2),(2,2)) avecCrayon(1,Crimson);
26
27 trace Courbe(f,g,-10,-0.05,300) avecCrayon(1.5,DarkGoldenrod);
28 trace Courbe(f,g,0.05,10,300) avecCrayon(1.5,DarkGoldenrod);
29
30 pointe Point(0,0);
31 pointe Point(-1,2);
32 pointe Point(3,2);
33 pointe Point(1,6);
34
35 Fin;
36 endfig;
37 end

```

## 13.8 Le bicorne



```

1 input geom2d;
2
3 labeloffset := 6;
4 gddU:=1cm;
5
6 vardef f(expr t) = 2*sin(t) enddef;
7 vardef g(expr t) = 2*cos(t)*cos(t)/(2-cos(t)) enddef;
8 vardef h(expr t) = -t*(1+f(t)) enddef;

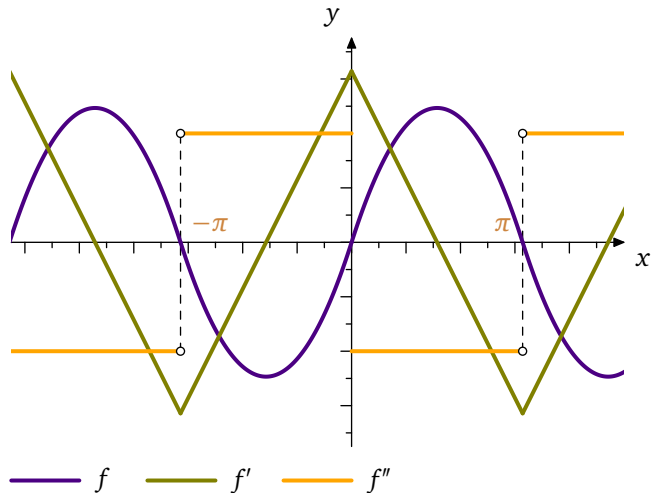
```

```

9
10 def traceDoubleVecteur(expr o,d)= drawdblarrow ((o-d)--(o+d))
    gddEnPlace enddef;
11
12 path bicorne;
13 bicorne = Courbe(f,g,-Pi,Pi,200)--cycle;
14
15 beginfig(1);
16
17 Repere(9,6,4.5,1,2,2);
18 Axes;
19 Debut;
20 Axes;
21 Graduations; Unites(1);
22
23 colorieAvecTransparence(bicorne,CadetBlue,0.2);
24
25 traceDoubleVecteur((0,2),(0.5,0)) avecCrayon(1,Crimson);
26 traceDoubleVecteur((0,2/3),(0.5,0)) avecCrayon(1,Crimson);
27 fleche Segment((2,0),(1.6,0.4)) avecCrayon(1,Crimson);
28 fleche Segment((-2,0),(-1.6,0.4)) avecCrayon(1,Crimson);
29
30 trace bicorne avecCrayon(1.5,CadetBlue);
31
32 pointe Point(0,0);
33 pointe Point(2,0);
34 pointe Point(-2,0);
35
36 Fin;
37 endfig;
38 end

```

## 13.9 Une fonction et ses dérivées



```

1 input geom2d;
2
3 labeloffset := 6;
4 gddU:=0.9cm;
5
6 vardef f(expr x) = x*(Pi-x) enddef; % f
7 vardef g(expr x) = Pi-2*x enddef; % f'
8 vardef h(expr x) = -2 enddef; % f''
9
10 beginfig(1);
11
12 Repere(9,6,5,3,.8,0.8);
13 Axes;
14 Debut;
15 Graduations;
16
17 trace (Pi,-2)--(Pi,2) dashed evenly;
18 trace (-Pi,-2)--(-Pi,2) dashed evenly;
19
20 trace Representation(f,0,Pi,100) avecCrayon(1.5,Indigo);

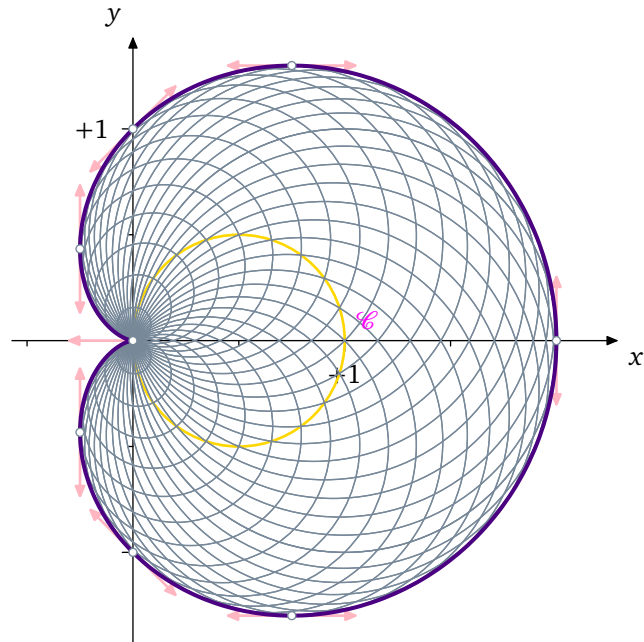
```

```

21 trace (Representation(f,0,Pi,100) scaled -1) avecCrayon(1.5,Indigo)
22 ;
23 trace (Representation(f,0,Pi,100) shifted (-2*Pi,0)) avecCrayon
24 (1.5,Indigo);
25 trace (Representation(f,0,Pi,100) scaled -1 shifted (2*Pi,0))
26 avecCrayon(1.5,Indigo);
27
28 trace Representation(g,0,Pi,100) avecCrayon(1.5,Olive);
29 trace (Representation(g,0,Pi,100) xscaled -1) avecCrayon(1.5,Olive)
30 ;
31 trace (Representation(g,0,Pi,100) shifted (-2*Pi,0)) avecCrayon
32 (1.5,Olive);
33 trace (Representation(g,0,Pi,100) xscaled -1 shifted (2*Pi,0))
34 avecCrayon(1.5,Olive);
35
36 trace Representation(h,0,Pi,100) avecCrayon(1.5,Orange);
37 trace (Representation(h,0,Pi,100) scaled -1) avecCrayon(1.5,Orange)
38 ;
39 trace (Representation(h,0,Pi,100) shifted (-2*Pi,0)) avecCrayon
40 (1.5,Orange);
41 trace (Representation(h,0,Pi,100) scaled -1 shifted (2*Pi,0))
42 avecCrayon(1.5,Orange);
43
44 pointe Point(Pi,2);
45 pointe Point(Pi,-2);
46 pointe Point(-Pi,2);
47 pointe Point(-Pi,-2);
48
49 label.urt(texttext("\(-\pi\)"), (-Pi,0) gddEnPlace) withcolor Peru;
50 label.ulft(texttext("\(\pi\)"), (Pi,0) gddEnPlace) withcolor Peru;
51 Fin;
52
53 trace (0,-0.5)--(1,-0.5) avecCrayon(1.5,Indigo);
54 label.rt(texttext("(f)"), (1,-0.5) gddEnPlace);
55 trace (2,-0.5)--(3,-0.5) avecCrayon(1.5,Olive);
56 label.rt(texttext("(f')"), (3,-0.5) gddEnPlace);
57 trace (4,-0.5)--(5,-0.5) avecCrayon(1.5,Orange);
58 label.rt(texttext("(f'')"), (5,-0.5) gddEnPlace);
59
60 endfig;
61
62 end

```

## 13.10 Cardioïde



```

1 input geom2d;
2
3 labeloffset := 6;
4 gddU:=0.8cm;
5
6 vardef r(expr t) = 1+cos(t) enddef;
7 vardef rp(expr t) = (r(t)*cos(t),r(t)*sin(t)) enddef;
8
9 def traceDoubleVecteur(expr o,d)= drawdblarrow ((o-d)--(o+d))
   gddEnPlace enddef;
10
11 beginfig(1);
12 Reperre(10,10,2,5,3.5,3.5);

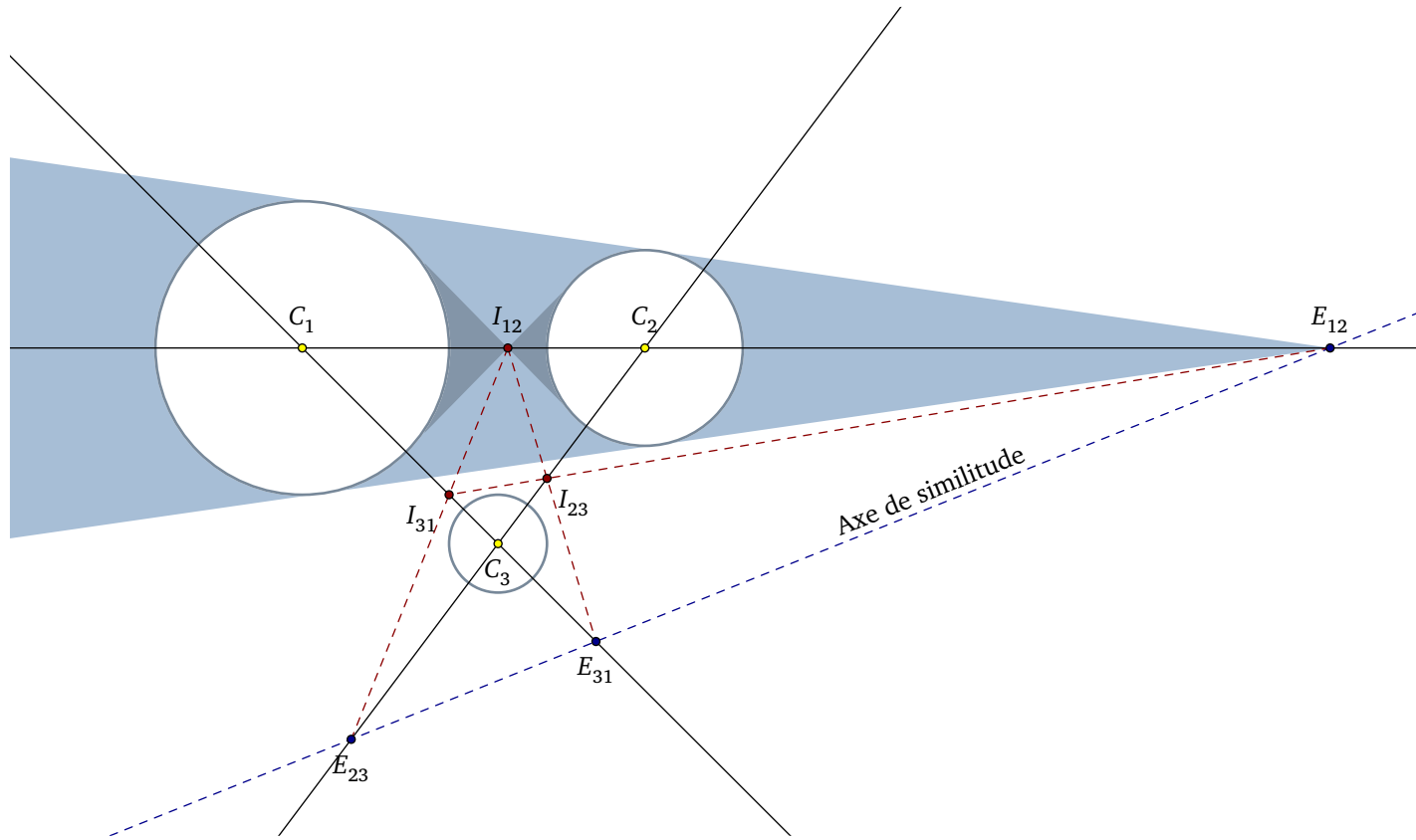
```

```

13 Axes;
14 Debut;
15 Graduations; Unites(1);
16
17 drawoptions(withcolor LightSlateGrey);
18 draw fullcircle shifted (0.5,0) gddEnPlace avecCrayon(1,Gold);
19
20 nb = 80;
21 pas = 2Pi / nb;
22 for i=0 upto nb:
23   theta := i * pas;
24   draw (fullcircle scaled 2cos(theta)
25     shifted (cos(theta)*cos(theta),cos(theta)*sin(theta)))
26     gddEnPlace;
27 endfor;
28
29 traceDoubleVecteur(rp(0),(0,0.3)) avecCrayon(1,LightPink);
30 traceDoubleVecteur(rp(Pi/3),(0.3,0)) avecCrayon(1,LightPink);
31 traceDoubleVecteur(rp(Pi/2),(0.2,0.2)) avecCrayon(1,LightPink);
32 traceDoubleVecteur(rp(2Pi/3),(0,0.3)) avecCrayon(1,LightPink);
33 traceDoubleVecteur(rp(-Pi/3),(0.3,0)) avecCrayon(1,LightPink);
34 traceDoubleVecteur(rp(-Pi/2),(0.2,-0.2)) avecCrayon(1,LightPink);
35 traceDoubleVecteur(rp(-2Pi/3),(0,0.3)) avecCrayon(1,LightPink);
36
37 fleche Segment(origine,(-0.3,0)) avecCrayon(1,LightPink);
38 trace CourbeEnPolaires(r,-Pi,Pi,100) avecCrayon(1.5,Indigo);
39
40 pointe Point(0,0);
41 pointe Point(2,0);
42 pointe PairTOPoint(rp(Pi/3));
43 pointe PairTOPoint(rp(-Pi/3));
44 pointe PairTOPoint(rp(2Pi/3));
45 pointe PairTOPoint(rp(-2Pi/3));
46 pointe Point(0,1);
47 pointe Point(0,-1);
48
49 label.urt(texttext("\(\mathcal{C}\)"),PtR(Point(1,0))) withcolor
   Magenta;
50 Fin;
51 endfig;
52 end

```

### 13.11 Axe de similitude



```

1 input geom2d;
2 %%% depuis Drawing with Metapost de Toby Thurston
3 labeloffset := 6;
4 gddTaillePoint := 3;
5 gddCouleurPoint := Yellow;
6 gddU:=0.65cm;
7 beginfig(1);
8 C1 = Cercle((-4,0),3);
9 C2 = Cercle((3,0),2);
10 C3 = Cercle((0,-4),1);
11
12 T1 = TangenteCommuneExterieur(C1,C2);
13 T2 = TangenteCommuneExterieur(C2,C1);
14 T3 = TangenteCommuneInterieur(C1,C2);
15 T4 = TangenteCommuneInterieur(C2,C1);
16
17 T5 = TangenteCommuneExterieur(C2,C3);
18 T6 = TangenteCommuneExterieur(C3,C2);
19 T7 = TangenteCommuneInterieur(C2,C3);
20 T8 = TangenteCommuneInterieur(C3,C2);
21
22 T9 = TangenteCommuneExterieur(C1,C3);
23 T10 = TangenteCommuneExterieur(C3,C1);
24 T11 = TangenteCommuneInterieur(C1,C3);
25 T12 = TangenteCommuneInterieur(C3,C1);
26
27 E12 = IntersectionDroites(T1,T2);
28 E23 = IntersectionDroites(T5,T6);
29 E31 = IntersectionDroites(T9,T10);
30 I12 = IntersectionDroites(T3,T4);
31 I23 = IntersectionDroites(T7,T8);
32 I31 = IntersectionDroites(T11,T12);
33
34 path t[];
35 t1 :=(gddTraceObjet T1) gddEnPlace;
36 t2 := (gddTraceObjet T2) gddEnPlace;
37 t3 := (-10*gddU,-10*gddU)--(-10*gddU,10*gddU);
38 fill buildcycle(t1, t3,reverse t2) withcolor 1.4*LightSlateGrey;
39
40 t4 :=(gddTraceObjet T3) gddEnPlace;
41 t5 := (gddTraceObjet T4) gddEnPlace;
42 t6 := (1.5*gddU,-10*gddU)--(1.5*gddU,10*gddU);
43 fill buildcycle(t4, t5,reverse t6) withcolor 1.1*LightSlateGrey;
44

```

```

45 t7 :=(gddTraceObjet T3) gddEnPlace;
46 t8 := (gddTraceObjet T4) gddEnPlace;
47 t9 := (-1.5*gddU,-10*gddU)--(-1.5*gddU,10*gddU);
48 fill buildcycle(t7, t8,reverse t9) withcolor 1.1*LightSlateGrey;
49
50 drawoptions(withpen pencircle scaled 1pt withcolor LightSlateGrey);
51 colorie C1 withcolor white;
52 colorie C2 withcolor white;
53 colorie C3 withcolor white;
54 trace C1; trace C2; trace C3;
55
56 drawoptions();
57 trace Droite(Centre(C1),Centre(C2));
58 trace Droite(Centre(C3),Centre(C2));
59 trace Droite(Centre(C1),Centre(C3));
60
61 D_E = Droite(E12,E23);
62 trace D_E dashed evenly withcolor DarkBlue;
63
64 E_I = Segment(E12,I31);
65 trace E_I dashed evenly withcolor DarkRed;
66 I_E1 = Segment(I12,E31);
67 trace I_E1 dashed evenly withcolor DarkRed;
68 I_E2 = Segment(I12,E23);
69 trace I_E2 dashed evenly withcolor DarkRed;
70
71
72 drawoptions();
73 pointe Centre(C1);
74 pointe Centre(C2);
75 pointe Centre(C3);
76 gddCouleurPoint := DarkBlue;
77 pointe E12; pointe E31; pointe E23;
78 gddCouleurPoint := DarkRed;
79 pointe I12; pointe I31; pointe I23;
80
81 label.top(btex $\E_{12}$ etex,PointTOPair(E12) gddEnPlace);
82 label.bot(btex $\E_{31}$ etex,PointTOPair(E31) gddEnPlace);
83 label.bot(btex $\E_{23}$ etex,PointTOPair(E23) gddEnPlace);
84
85 label.top(btex $\I_{12}$ etex,PointTOPair(I12) gddEnPlace);
86 label.llft(btex $\I_{31}$ etex,PointTOPair(I31) gddEnPlace);
87 label.lrt(btex $\I_{23}$ etex,PointTOPair(I23) gddEnPlace);
88
89

```

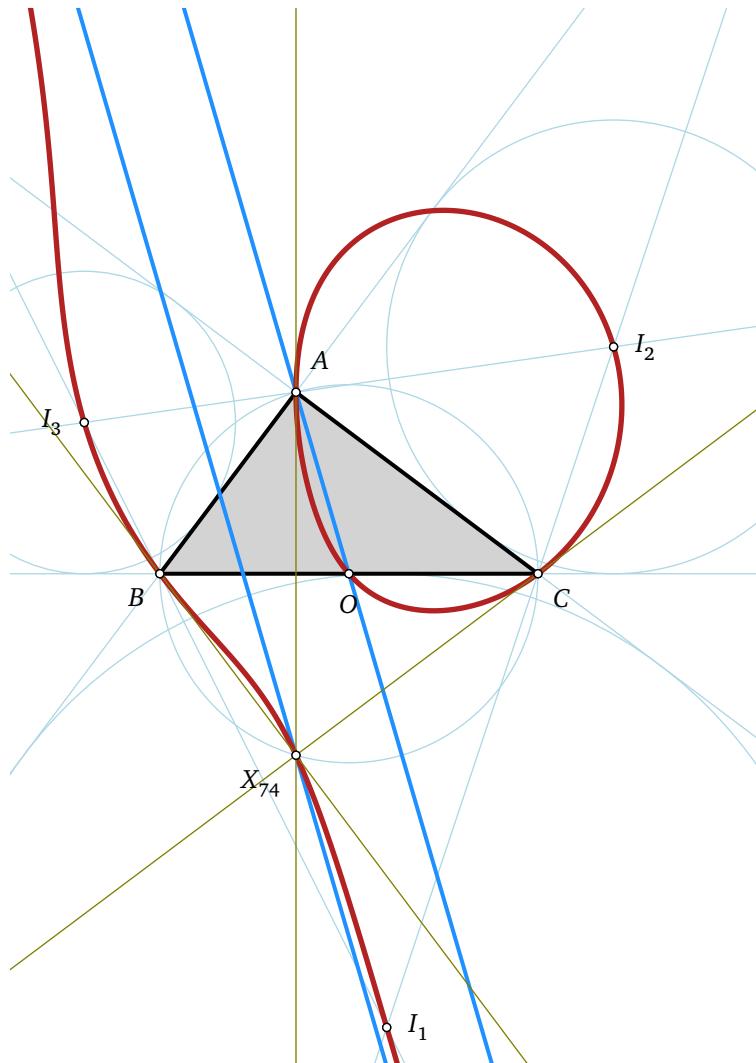
```

90 label.top(btex $C_{1}$ etex,PointTOPair(Centre(C1)) gddEnPlace);           ,Pt(E12)]
91 label.top(btex $C_{2}$ etex,PointTOPair(Centre(C2)) gddEnPlace);           95 gddEnPlace +(0,3));
92 label.bot(btex $C_{3}$ etex,PointTOPair(Centre(C3)) gddEnPlace);           96 Fenetre(-10,-10,19,7);
93                                                                           97 endfig;
94 draw texttext("Axe de similitude") rotated (22) shifted (0.5[Pt(E23) 98 end.

```



### 13.12 Tracé d'une cubique



```

1 input geom2d;
2 gddU := 0.5cm;
3 labeloffset := 8pt;
4 gddTailleLabel := 1;
5
6 beginfig(1);
7 A := Point(3.6,4.8);
8 B := Point(0,0);
9 C := Point(10,0);
10 T := Triangle(A,B,C);
11 X_74 := Point(3.6,-4.8);
12
13 C1 := CourbeDat("K001-1",0);
14 C2 := CourbeDat("K001-2",1);
15
16 CE1 := CercleExinscrit(T,1);
17 CE2 := CercleExinscrit(T,2);
18 CE3 := CercleExinscrit(T,3);
19 I_1 := Centre(CE1);
20 I_2 := Centre(CE2);
21 I_3 := Centre(CE3);
22 CC := CercleCirconscri(T);
23 O := Centre(CC);
24
25 drawoptions(withcolor LightBlue);
26 trace Droite(A,B);
27 trace Droite(B,C);
28 trace Droite(C,A);
29 trace Droite(I_1,I_2);
30 trace Droite(I_2,I_3);
31 trace Droite(I_3,I_1);
32 trace CE1;
33 trace CE2;
34 trace CE3;
35 trace CC;
36 colorie T withcolor LightGrey;
37 drawoptions(withpen pencircle scaled 1.5);
38 trace T;

```

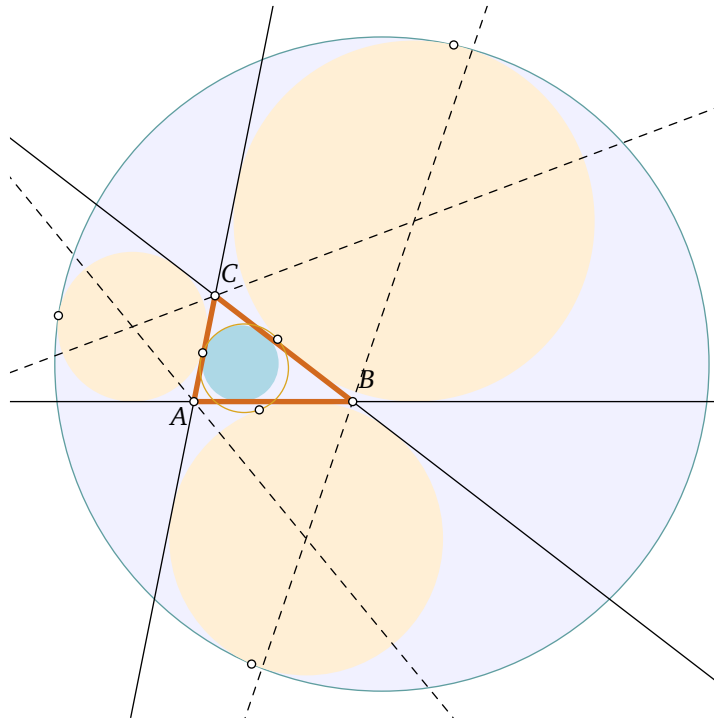
```

39 drawoptions(withcolor DodgerBlue withpen pencircle scaled 1.5);
40 trace Droite(0,A);
41 trace Droite(X_74, Point(2.2,0));
42 drawoptions(withpen pencircle scaled 2);
43 trace C1 withcolor FireBrick;
44 trace C2 withcolor FireBrick;
45 drawoptions(withcolor Olive);
46 trace Droite(A,X_74);
47 trace Droite(B,X_74);
48 trace Droite(C,X_74);
49 drawoptions();
50 pointe A;
51 pointe B;
52 pointe C;
53 pointe I_1;
54 pointe I_2;
55 pointe I_3;

56 pointe 0;
57 pointe X_74;
58
59 marque.urt "A";
60 marque.llft "B";
61 marque.lrt "C";
62 marque.rt "I_1";
63 marque.rt "I_2";
64 marque.lft "I_3";
65 marque.bot "0";
66 marque.llft "X_74";
67
68 Fenetre(-4,-13,16,15);
69 endfig;
70
71 end

```

### 13.13 Apollonius



```

1 input geom2d;
2 gddU:=0.35cm;
3 beginfig(1);
4 % nos trois points
5 A = Point(0,0);
6 B = Point(6,0);
7 C = Point(0.8,4);
8 T_ABC = Triangle(A,B,C);

```

```

9
10 C_I = CercleInscrit(T_ABC);
11 C_A = CercleExinscrit(T_ABC,2);
12 C_B = CercleExinscrit(T_ABC,3);
13 C_C = CercleExinscrit(T_ABC,1);
14
15 d_AB = Droite(A,B);
16 d_BC = Droite(B,C);
17 d_CA = Droite(C,A);
18
19 I = Centre(C_I);
20 I_C_A = Centre(C_A);
21 I_C_B = Centre(C_B);
22 I_C_C = Centre(C_C);
23 d_CAA = Droite(A,I_C_A);
24 d_CAB = Droite(B,I_C_B);
25 d_CAC = Droite(C,I_C_C);
26
27 A_S = AxeDeSimilitude(C_A,C_B,C_C);
28 P_CA = ProjectionPointSurDroite(I_C_A,A_S);
29 P_CB = ProjectionPointSurDroite(I_C_B,A_S);
30 P_CC = ProjectionPointSurDroite(I_C_C,A_S);
31
32 P_A = Inversion(P_CA,C_A);
33 P_B = Inversion(P_CB,C_B);
34 P_C = Inversion(P_CC,C_C);
35
36 C_R = CentreRadical(C_A,C_B,C_C);
37 % les neuf points pour les cercles 'deuler (tangent intérieur)
38 % et 'dapollonius (tangent extérieur)
39 D1 = Droite(C_R,P_A);
40 P1 = IntersectionDroiteCercle(D1,C_A,1);
41 Q1 = IntersectionDroiteCercle(D1,C_A,2);
42
43 D2 = Droite(C_R,P_B);
44 P2 = IntersectionDroiteCercle(D2,C_B,1);
45 Q2 = IntersectionDroiteCercle(D2,C_B,2);
46
47 D3 = Droite(C_R,P_C);
48 P3 = IntersectionDroiteCercle(D3,C_C,2);

```

```

49 Q3 = IntersectionDroiteCercle(D3,C_C,1);
50
51 % le cercle 'dapollonius
52 Apol = CercleTroisPoints(P1,P2,P3);
53 colorie Apol withcolor 1.05*Lavender;
54 trace Apol withcolor CadetBlue;
55
56 % cercle inscrit
57 colorie C_I withcolor LightBlue;
58 % cercles exinscrits
59 colorie C_A withcolor PapayaWhip;
60 colorie C_B withcolor PapayaWhip;
61 colorie C_C withcolor PapayaWhip;
62
63 trace d_AB;
64 trace d_BC;
65 trace d_CA;
66
67 trace d_CAA dashed evenly;
68 trace d_CAB dashed evenly;
69 trace d_CAC dashed evenly;
70

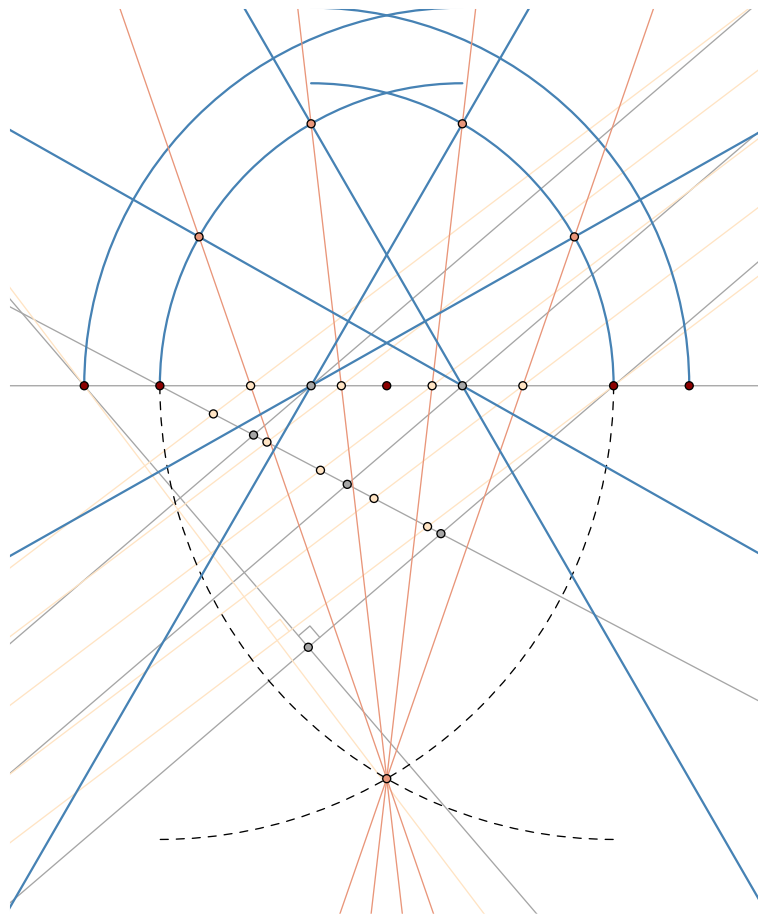
```

```

71 trace T_ABC withpen pencircle scaled 2 withcolor Chocolate;
72
73 C_E = CercleEuler(T_ABC);
74 trace C_E withcolor Goldenrod;
75
76 pointe P1;
77 pointe P2;
78 pointe P3;
79
80 pointe Q1;
81 pointe Q2;
82 pointe Q3;
83
84
85 marque.llft "A";
86 marque.urt "B";
87 marque.urt "C";
88 Fenetre(-7,-12,20,15)
89 endfig;
90 end.

```

## 13.14 Épure d'ogive



```

1 input geom2d;
2
3 beginfig(1);
4 % les points de base pour l'ogive
5 O = Point(0,0);
6 A = Point(-3,0);
7 A2 = Point(-4,0);
8 B = Point(3,0);
9 B2 = Point(4,0);
10 % la droite de base de l'ogive
11 AB = Droite(A,B);
12 % on se prend une droite quelconque
13 Ct = Point(0.8,-2); % point quelconque
14 AC = Droite(A,Ct); % droite quelconque
15 % on prend trois longueurs égales sur la droite
16 longueurTier := 1.4;
17 C1 = ReportSurDroite(A,AC,longueurTier);
18 C2 = ReportSurDroite(C1,AC,longueurTier);
19 C3 = ReportSurDroite(C2,AC,longueurTier);
20 % on définit la droite qui passe par B et le dernier point
21 BC = Droite(B,C3);
22 % on construit une droite perpendiculaire à BC
23 D = Point(-4,0);
24 perpD = DroitePerpendiculaire(BC,D);
25 DD = IntersectionDroites(perpD,BC);
26 % on projette C2 et C1 sur la droite AB pour diviser [AB] en 3
    parties
27 % égales
28 perpC2 = DroitePerpendiculaire(perpD,C2);
29 perpC1 = DroitePerpendiculaire(perpD,C1);
30 D1 = IntersectionDroites(AB,perpC1);
31 D2 = IntersectionDroites(AB,perpC2);
32 % on construit l'enveloppe de l'ogive
33 C_DA = CercleCP(D2,A);
34 C_DB = CercleCP(D1,B);
35 C_DA2 = CercleCP(D2,A2);
36 C_DB2 = CercleCP(D1,B2);
37 % on construit le point P pour la projection finale
38 C_AB = CercleCP(A,B);
39 C_BA = CercleCP(B,A);
40 P = IntersectionCercles(C_BA,C_AB);

```

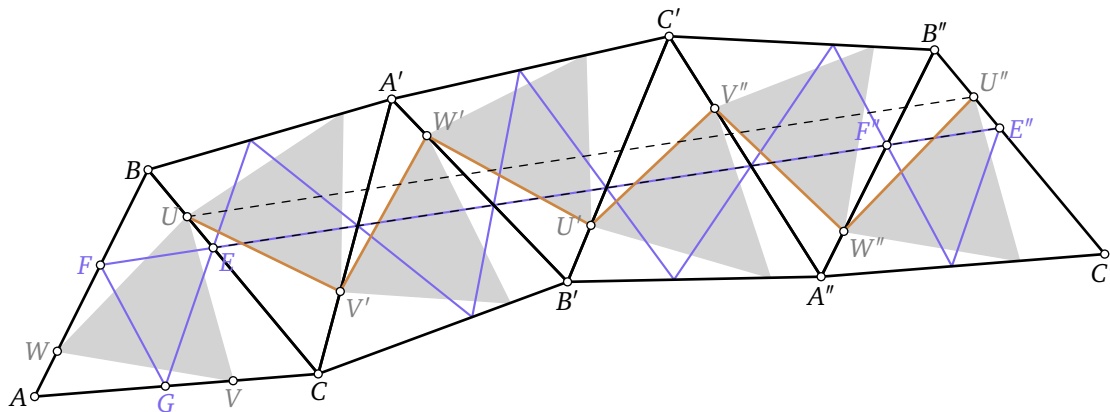
```

41 % on prend 5 longueurs égales sur la droite quelconque du début
42 longueurCinq :=0.8;
43 F1 = ReportSurDroite(A,AC, longueurCinq);
44 F2 = ReportSurDroite(F1,AC, longueurCinq);
45 F3 = ReportSurDroite(F2,AC, longueurCinq);
46 F4 = ReportSurDroite(F3,AC, longueurCinq);
47 F5 = ReportSurDroite(F4,AC, longueurCinq);
48 % on projette les points sur AB pour diviser [AB] en 5 parties
   égale
49 BF5 = Droite(B,F5);
50 perpF5 = DroitePerpendiculaire(BF5,A2);
51 CC = IntersectionDroites(BF5,perpF5);
52 perpF4 = DroitePerpendiculaire(perpF5,F4);
53 perpF3 = DroitePerpendiculaire(perpF5,F3);
54 perpF2 = DroitePerpendiculaire(perpF5,F2);
55 perpF1 = DroitePerpendiculaire(perpF5,F1);
56 G1 = IntersectionDroites(AB,perpF1);
57 G2 = IntersectionDroites(AB,perpF2);
58 G3 = IntersectionDroites(AB,perpF3);
59 G4 = IntersectionDroites(AB,perpF4);
60 % on projette les Gi sur l'ogive avec le point P
61 PG1 = Droite(P,G1);
62 PG2 = Droite(P,G2);
63 PG3 = Droite(P,G3);
64 PG4 = Droite(P,G4);
65 I1 = IntersectionDroiteCercle(PG1,C_DA,1);
66 I2 = IntersectionDroiteCercle(PG2,C_DA,2);
67 I3 = IntersectionDroiteCercle(PG3,C_DB,2);
68 I4 = IntersectionDroiteCercle(PG4,C_DB,2);
69 % à partir des 2 points issus de la division en 3 de [AB]
70 % on trace les séparations des pierres qui constituent l'ogive
71 Dvoute1 = Droite(D2,I1);
72 Dvoute2 = Droite(D2,I2);
73 Dvoute3 = Droite(D1,I3);
74 Dvoute4 = Droite(D1,I4);
75
76 % les tracés
77 drawoptions(withcolor 1.3*Grey);
78 trace perpC2; trace perpC1;
79 trace AB;
80 trace AC;
81 trace BC;
82 trace perpD;
83 trace SigneOrtho(B,DD,A2,0.2);
84
85 drawoptions(withcolor Bisque);
86 trace BF5;
87 trace perpF5;trace perpF4;trace perpF3;trace perpF2;trace perpF1;
88 trace SigneOrtho(B,CC,A2,0.2);
89
90 drawoptions(withcolor SteelBlue withpen pencircle scaled 0.85pt);
91 trace gddTraceArcDeCercle(C_DA,Pi/2,Pi);
92 trace gddTraceArcDeCercle(C_DA2,Pi/2,Pi);
93 trace gddTraceArcDeCercle(C_DB2,0,Pi/2);
94 trace gddTraceArcDeCercle(C_DB,0,Pi/2);
95
96 drawoptions();
97 trace gddTraceArcDeCercle(C_AB,0,-Pi/2) dashed evenly;
98 trace gddTraceArcDeCercle(C_BA,Pi,3Pi/2) dashed evenly;
99
100 drawoptions(withcolor DarkSalmon);
101 trace PG1; trace PG2; trace PG3; trace PG4;
102
103 drawoptions(withcolor SteelBlue withpen pencircle scaled 0.85pt);
104 trace Dvoute1; trace Dvoute2; trace Dvoute3; trace Dvoute4;
105 drawoptions();
106
107 gddCouleurPoint := DarkRed;
108 pointe O;
109 pointe A;
110 pointe B; pointe D;
111 pointe B2;
112
113 gddCouleurPoint := Bisque;
114 pointe F1; pointe F2; pointe F3; pointe F4; pointe F5;
115 pointe G1; pointe G2;pointe G3;pointe G4;
116 gddCouleurPoint := 1.3*Grey;
117 pointe DD;
118 pointe C1; pointe C2; pointe C3;
119 pointe D1; pointe D2;
120
121 gddCouleurPoint := DarkSalmon;
122 pointe P;
123 pointe I1; pointe I2; pointe I3; pointe I4;
124 Fenetre(-5,-7,5,5);
125 endfig;
126 end.

```

### 13.15 Triangle orthique et problème de Fagnano

Figure 21 de [3].



```

1 input geom2d;
2 gddU:=1.5cm;
3 beginfig(1);
4 numeric A[],B[],C[],E[],F[],G[],U[],W[],V[],T[],R[],P[];
5 A[1] = Point(0,0);
6 B1 = Point(1,2);
7 C1 = Point(2.5,0.2);
8 W1 = PointDe(Segment(A1,B1),0.2);
9 U1 = PointDe(Segment(B1,C1),0.23);
10 V1 = PointDe(Segment(C1,A1),0.3);
11 G1 = ProjectionPointSurDroite(B1,Droite(A1,C1));
12 F1 = ProjectionPointSurDroite(C1,Droite(A1,B1));
13 E1 = ProjectionPointSurDroite(A1,Droite(B1,C1));
14
15
16
17 for i:=0 step 3 until 3:
18
19 D[1+i] = Droite(B[1+i],C[1+i]);

```

```

20 A[2+i] = SymetrieAxiale(A[1+i],D[1+i]);
21
22 B[2+i] = B[1+i];
23 C[2+i] = C[1+i];
24
25 W[2+i] = SymetrieAxiale(W[1+i],D[1+i]);
26 V[2+i] = SymetrieAxiale(V[1+i],D[1+i]);
27 U[2+i] = U[1+i];
28
29 G[2+i] = SymetrieAxiale(G[1+i],D[1+i]);
30 F[2+i] = SymetrieAxiale(F[1+i],D[1+i]);
31 E[2+i] = E[1+i];
32
33 D[2+i] = Droite(A[2+i],C[2+i]);
34 B[3+i] = SymetrieAxiale(B[2+i],D[2+i]);
35 A[3+i] = A[2+i];
36 C[3+i] = C[2+i];
37 W[3+i] = SymetrieAxiale(W[2+i],D[2+i]);
38 U[3+i] = SymetrieAxiale(U[2+i],D[2+i]);

```

```

39 V[3+i] = V[2+i];
40 E[3+i] = SymetrieAxiale(E[2+i],D[2+i]);
41 F[3+i] = SymetrieAxiale(F[2+i],D[2+i]);
42 G[3+i] = G[2+i];
43
44 D[3+i] = Droite(A[3+i],B[3+i]);
45 C[4+i] = SymetrieAxiale(C[3+i],D[3+i]);
46 A[4+i] = A[3+i];
47 B[4+i] = B[3+i];
48 U[4+i] = SymetrieAxiale(U[3+i],D[3+i]);
49 V[4+i] = SymetrieAxiale(V[3+i],D[3+i]);
50 W[4+i] = W[3+i];
51 E[4+i] = SymetrieAxiale(E[3+i],D[3+i]);
52 G[4+i] = SymetrieAxiale(G[3+i],D[3+i]);
53 F[4+i] = F[3+i];
54 endfor;
55
56 for i:=1 upto 7:
57   T[i] = Triangle(A[i],B[i],C[i]);
58   Q[i] = Triangle(U[i],V[i],W[i]);
59   P[i] = Triangle(E[i],F[i],G[i]);
60   colorie Q[i] avecCrayon(0.5,LightGray);
61   trace P[i] avecCrayon(0.8,MediumSlateBlue);
62 endfor;
63 for i:=1 upto 7:
64   trace T[i] avecCrayon(1,black);
65 endfor;
66
67 trace LigneBrisee(U1,V2,W3,U4,V5,W6,U7) avecCrayon(1,Peru) ;
68 trace Segment(U1,U7) dashed evenly;
69 trace Segment(E1,E7) dashed evenly;
70 pointe A1; pointe B1; pointe C1;
71 pointe A2; pointe B3; pointe C4;
72 pointe A5; pointe B6; pointe C7;
73 gddLabel.lft(texttext("$A$"),A1);

```

```

74 gddLabel.lft(texttext("$B$"),B1);
75 gddLabel.bot(texttext("$C$"),C1);
76 gddLabel.top(texttext("$A'$"),A2);
77 gddLabel.bot(texttext("$B'$"),B3);
78 gddLabel.top(texttext("$C'$"),C4);
79 gddLabel.bot(texttext("$A''$"),A5);
80 gddLabel.top(texttext("$B''$"),B7);
81 gddLabel.bot(texttext("$C''$"),C7);
82
83 pointe U1; pointe V1; pointe W1;
84 label.lft(texttext("$U$"),Ptr(U1)) withcolor Gray;
85 label.bot(texttext("$V$"),Ptr(V1)) withcolor Gray;
86 label.lft(texttext("$W$"),Ptr(W1)) withcolor Gray;
87
88 label.lft(texttext("$U'$"),Ptr(U4)) withcolor Gray;
89 label.lrt(texttext("$V'$"),Ptr(V2)) withcolor Gray;
90 label.urt(texttext("$W'$"),Ptr(W3)) withcolor Gray;
91 label.urt(texttext("$U''$"),Ptr(U7)) withcolor Gray;
92 label.urt(texttext("$V''$"),Ptr(V5)) withcolor Gray;
93 label.lrt(texttext("$W''$"),Ptr(W7)) withcolor Gray;
94 pointe U4; pointe V2; pointe W3;
95 pointe U7; pointe V5; pointe W7;
96 pointe E1; pointe F1; pointe G1;
97 label.lrt(texttext("$E$"),Ptr(E1)) withcolor MediumSlateBlue;
98 label.lft(texttext("$F$"),Ptr(F1)) withcolor MediumSlateBlue;
99 label.bot(texttext("$G$"),Ptr(G1)) withcolor MediumSlateBlue;
100
101 label.ulft(texttext("$F'$"),Ptr(F7)) withcolor MediumSlateBlue;
102 pointe F7;
103 label.rt(texttext("$E''$"),Ptr(E7)) withcolor MediumSlateBlue;
104 pointe E7;
105 endfig;
106 end.

```



## 14 Historique

**21 octobre 2024** : corrections de Quark67 (merci à lui) sur la documentation et modifications suite à ses remarques. `DemieHyperbole` devient `DemiHyperbole` (sans compatibilité). Création de deux nouveaux types de pointage de `point` : croix et disque (voir commande `pointe`).

**13 octobre 2024** : publication de la version 1.0 sur le CTAN.

## Références

- [1] Hans HAGEN et al. *The luamplib package. Use LuaTeX's built-in MetaPost interpreter.* Version 2.34.5. 3 août 2024. URL : <https://ctan.org/pkg/luamplib>.
- [2] Jens-Uwe MORAWSKI. *The latexMP package. Interface for  $\TeX$ -based typesetting in MetaPost.* Version 1.2.1. 21 juin 2020. URL : <https://ctan.org/pkg/latexmp>.
- [3] Hans RADEMACHER et Otto TOEPLITZ. *The enjoyment of math. Translated from the German. With a new foreword by Alex Kontorovich.* English. Reprint. Princeton Sci. Libr. Princeton, NJ : Princeton University Press, 2023. ISBN : 978-0-691-24154-8 ; 978-0-691-24153-1. DOI : [10.1515/9780691241531](https://doi.org/10.1515/9780691241531).
- [4] Esger RENKEMA. *The minim-mp package. Low-level mplib integration for LuaTeX.* Version 2024/1.6. 6 avr. 2024. URL : <https://ctan.org/pkg/minim-mp>.
- [5] THE METAPOST TEAM et John HOBBY. *The metapost package. A development of Metafont for creating graphics.* 26 août 2021. URL : <https://ctan.org/pkg/metapost>.
- [6] Toby THURSTON. *The Drawing-with-Metapost package. How to draw technical diagrams with MetaPost.* 16 avr. 2023. URL : <https://ctan.org/pkg/drawing-with-metapost>.

## Index

- [\\_E](#), 51
- [Abscisse](#), 8
- [Addition](#), 8
- [AdditionAbscisses](#), 8
- [AdditionOrdonnee](#), 8
- [AdditionVecteur](#), 12
- [AireTriangle](#), 34
- [Arc](#), 39
- [arccos](#), 51
- [arcsin](#), 51
- [arctan](#), 51
- [AsymptoteHyperbole](#), 32
- [avecCrayon](#), 5
- [AxeDeSimilitude](#), 22
- [AxeRadical](#), 21
- [Axes](#), 47
- [AxesBords](#), 49
  
- [Barycentre](#), 9
- [Bissectrice](#), 10
  
- [CadreRepere](#), 50
- [Centre](#), 19, 24, 31
- [CentreRadical](#), 22
- [Cercle](#), 17
- [CercleCirconscrit](#), 34
- [CercleCP](#), 17
- [CercleD](#), 18
- [CercleEuler](#), 35
- [CercleExinscrit](#), 35
- [CercleInscrit](#), 34
- [CerclePrincipale](#), 32
- [CercleTroisPoints](#), 18
- [ch](#), 51
- [ChampVecteurs](#), 54
- [ChampVecteursDD](#), 55
- [Chemin](#), 38
- [colorie](#), 4, 5
- [cos](#), 51
- [Courbe](#), 53
- [CourbeDat](#), 38
- [CourbeEnPolaires](#), 53
- [CoVertex](#), 24
  
- [Debut](#), 47
- [DemiGrandAxe](#), 24
- [DemiHyperbole](#), 32
  
- [DemiPetitAxe](#), 24
- [Directrice](#), 28, 32
- [DistancePointDroite](#), 16
- [Droite](#), 15
- [DroitePerpendiculaire](#), 16
  
- [Ellipse](#), 23
- [EllipseF](#), 23
- [EquationDroite](#), 15
- [EtiquetteChemin](#), 45
- [Excentricite](#), 25, 27, 31
- [exp](#), 51
  
- [Fenetre](#), 6
- [fermeture](#), 5
- [Fin](#), 47
- [fleche](#), 4
- [Foyer](#), 24
  
- [gddCouleurPoint](#), 6
- [gddEnPlace](#), 3
- [gddExtensionDroite](#), 15
- [gddLabel](#), 45
- [gddPointType](#), 6
- [gddTaillePoint](#), 5
- [gddTraceArcDeCercle](#), 38
- [gddTraceObjet](#), 5
- [gddU](#), 3
- [gddW](#), 3
- [gddXlabel](#), 47
- [gddYlabel](#), 47
- [Graduations](#), 49
- [GraduationsBords](#), 49
- [GrilleRepere](#), 50
  
- [Homothetie](#), 40
- [HyperboleFD](#), 31
  
- [Inclinaison](#), 25, 28, 31
- [IntersectionCercles](#), 19
- [IntersectionDroiteCercle](#), 19
- [IntersectionDroites](#), 16
- [Inversion](#), 42
- [IsoBarycentre](#), 9
  
- [LigneBrisee](#), 38
- [ln](#), 51
- [Longueur](#), 8

LongueurSegment, 14

Marque, 44  
marque, 45  
MarqueTrait, 44  
Milieu, 9

NombreCotesPolygone, 37  
Norme, 13

Ordonnee, 8  
OrdonneeRelativePointDroite, 16  
Orthocentre, 34

PairImp, 10  
PairTOPoint, 10  
ParaboleFD, 27  
Pi, 51  
Point, 7  
PointDansRepere, 7  
PointDe, 11  
pointe, 5  
PointImp, 11  
PointPolygone, 37  
PointTOPair, 10  
Polygone, 36  
PolygoneRegulier, 36  
ProduitScalaire, 13  
ProjectionPointSurDroite, 15

Rayon, 18  
Repere, 46

RepereMinMax, 48  
ReportSurDroite, 16  
Representation, 52  
Rotation, 9  
RotationCentre, 9

ScalaireVecteur, 12  
Segment, 14  
SegmentTOVecteur, 14  
sh, 51  
SigneOrtho, 43  
sin, 51  
Sommet, 28, 31  
SoustractionVecteur, 12  
SymetrieAxiale, 41  
SymetrieCentrale, 41

tan, 51  
TangenteCommuneExterieur, 20  
TangenteCommuneInterieur, 20  
TangenteEllipse, 25  
TangenteExterieurEllipse, 26  
th, 51  
trace, 4  
Triangle, 33

Unites, 50

Vecteur, 11  
VecteurP, 12  
Vertex, 24