Présentation

du

package

Euclide

de Dominique Rodriguez et Herbert Voß

François Hache – 2016

Sommaire

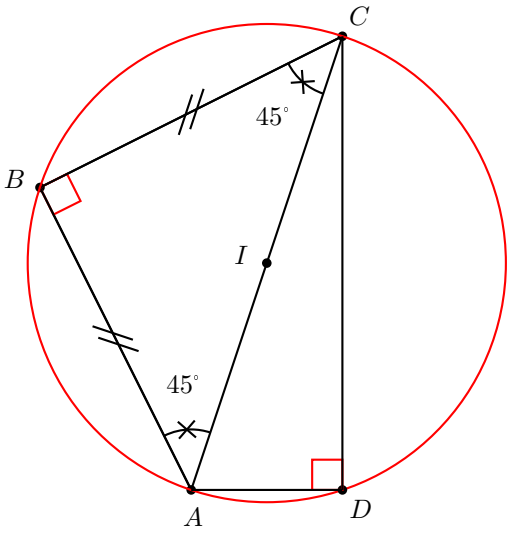
Première partie

Notions de base

En géométrie, on a souvent des figures à construire ; en analyse, on a souvent des courbes à tracer. Les outils de `PsTricks` sont déjà très performants mais les points sont représentés par leurs coordonnées. Ce serait plus pratique d'avoir un système qui permet de travailler avec des points représentés par leurs noms, comme on fait en géométrie.
C'est exactement ce que propose le package `euclide`.
L'auteur de ce package (qui date déjà d'une bonne dizaine d'années) est Dominique Rodriguez et je ne peux que le remercier pour ce magnifique travail !

1 Première figure

Voici une figure de géométrie relativement simple mais contenant déjà pas mal d'objets :



On y voit un triangle rectangle ADC, un triangle rectangle isocèle ABC de même hypoténuse $[AC]$, et leur cercle circonscrit de centre I, milieu de $[AC]$.
Les angles droits sont marqués, les angles à la base du triangle isocèle aussi ; de plus les côtés de même longueur sont codés.
On peut tout faire avec `PsTricks` mais les codages des côtés égaux et des angles ne sont pas commodes à faire (particulièrement le codage de l'angle $\widehat{ABC}$).
J'ai donc utilisé le package `euclide` dont le vrai nom est `pst-eucl` qui vient en complément de PsTricks (ce qui veut dire que toutes les instructions de `PsTricks` sont utilisables en plus des instructions de `euclide`).
On écrira donc dans le préambule :

```
\usepackage{pst-all}%      pour PsTricks
\usepackage{pst-eucl}%     pour euclide
```

Attention : je me suis rendu compte qu'il y avait un conflit entre le package `diagbox` (qui permet de tracer des traits obliques dans un tableau) et le package `pst-eucl` ; pour résoudre ce conflit, il faut charger le package `diagbox` avant le package `pst-eucl`.

1.1 Les points

Les points se définissent au moyen de l'instruction `\pstGeonode` ; on peut définir plusieurs points qui, par défaut, sont placés automatiquement sur le graphique avec leurs noms :

```
\pstGeonode(0,0){A}(-1,2){B}(1,3){C}(1,0){D}
```

Les noms des points sont écrits automatiquement en mode mathématique : A, B, C et D, ce qui veut dire qu'il ne faut pas mettre le signe $. Si on veut qu'ils ne soient pas écrits en mode mathématique, il suffit d'utiliser le booléen `PtNameMath` :

```
\psset{PtNameMath=false}
```

Les noms sont placés automatiquement horizontalement à droite du dessin du point ; on peut modifier cette position au moyen de la variable `PosAngle` qui donne, en degrés, l'angle fait par la position du point avec l'horizontale droite (qui correspond à 0) :

```
\pstGeonode[PosAngle=-45](1,0){D}
```

Quand il y a plusieurs points, les positions des points peuvent être définies par une liste :

```
\pstGeonode[PosAngle={-90,180,60,-45}](0,0){A}(-1,2){B}(1,3){C}(1,0){D}
```

1.2 Les triangles

Le package `euclide` est assez performant dans la gestion des triangles : on peut les tracer en utilisant une seule instruction qui définit à la volée les sommets par leurs coordonnées et en marquant leurs noms. Par exemple le triangle ABC sera défini par l'instruction :

```
\pstTriangle(0,0){A}(-1,2){B}(1,3){C}
```

Première remarque : les points sont définis par l'instruction `\pstTriangle`, ce n'est donc pas la peine de les avoir définis préalablement au moyen de `\pstGeonode`.
Le noms des points sont placés automatiquement et de façon intelligente, sur la bissectrice de l'angle au sommet. On peut naturellement modifier les positions des points au moyen des trois variables `PosAngleA`, `PosAngleB` et `PosAngleC` correspondant respectivement au premier sommet du triangle, au deuxième et au troisième.
On peut contrôler les dessins des trois sommets par les variables

`PointSymbolA`, `PointSymbolB` et `PointSymbolC`,

et même leurs noms par les variables `PointNameA`, `PointNameB` et `PointNameC` ; mais pourquoi changerait-on leurs noms ?
L'épaisseur du trait est gérée par la variable `linewidth`, tout comme dans `PsTricks` ; la valeur par défaut est `\pslinewidth` et on peut régler la largeur du trait en fonction de cette largeur :

```
linewidth=1.5\pslinewidth
```

On peut, naturellement, définir la largeur du trait en points ou en millimètres.

1.3 Marquage des angles

Angle droit

L'angle droit $\widehat{ABC}$ se marque au moyen de l'instruction `\pstRightAngle`. La marque de l'angle droit est un petit carré placé automatiquement dont le côté vaut `0,28 unit` ; comme je l'ai trouvée un peu trop grande, je l'ai réduite au moyen de la variable `RightAngleSize` :

```
\pstRightAngle[linecolor=red,RightAngleSize=0.2]{A}{B}{C}
```

Dans le marquage d'un angle droit, l'ordre des points ne compte pas : on obtient le même marquage en entrant `{A}{B}{C}` ou `{C}{B}{A}` ; il faut quand même que le sommet correspondant à l'angle droit soit au milieu !

Angle non droit

Un angle non droit se marque avec l'instruction `\pstMarkAngle`. Là encore, certaines variables permettent de tout contrôler comme `Mark`, si on veut coder l'angle, ou `LabelSep`, si on veut changer la distance entre le sommet et le label :

```
\pstMarkAngle[Mark=MarkCros,LabelSep=0.7]{C}{A}{B}{$45\degres$}
```

Les angles sont définis dans le sens trigonométrique donc l'angle `{C}{A}{B}` n'est pas le même que l'angle `{B}{A}{C}`.
C'est l'option `Mark=MarkCros` qui marque une petite croix dans l'angle ; les autres marques possibles sont `pstslash` (un trait), `pstslashh` (deux traits), `pstslashhh` (trois traits), `MarkHash`, `MarkHashh` et `MarkHashhh` (respectivement un, deux et trois traits épais), et enfin `MarkCross` (double croix).
Attention aux majuscules-minuscules dans les noms de ces variables !
Ces marques servent aussi à coder des segments (voir paragraphe 1.4).
L'étiquette (ici 45°) peut être n'importe quel texte ; elle est placée à une distance de `1 unit` du sommet de l'angle ; comme je trouve que c'est un peu loin, j'ai écrit `LabelSep=0.7`.
Si on ne veut pas d'étiquette, on entre une chaine vide `{}` en quatrième paramètre de l'angle.
Pour tracer un angle orienté, il suffit de rentrer `arrows=->` comme option.
Enfin on peut même agrandir ou diminuer le rayon de l'arc de cercle qui code l'angle au moyen de la variable `MarkAngleRadius` (valeur par défaut 0,4 `unit`).

1.4 Marquage des segments

Il reste à coder les segments égaux du triangle isocèle ABC ; c'est l'instruction `\pstSegmentMark` qui fait le travail, et c'est la variable `SegmentSymbol` qui détermine la marque de codage :

```
\pstSegmentMark[SegmentSymbol=MarkHashh]{A}{B}
```

Par défaut, le segment est tracé en même temps que sa marque et la marque par défaut est `MarkHashh`.
Si la marque est l'une des trois `MarkHash`, on peut modifier l'angle que fait le trait avec le segment grâce à la variable `MarkAngle`.
On peut même modifier la longueur de ces marques avec la variable `MarkHashLength`, et même l'écart qu'il y a entre deux traits avec la variable `MarkHashSep`.

1.5 Cercle et milieu

Première méthode

- Pour tracer le cercle de diamètre $[AC]$, on utilise l'instruction `\pstCircleAB` :

```
\pstCircleAB[linecolor=red]{A}{C}
```

- Une instruction permet de déterminer, tracer et nommer le milieu d'un segment ; c'est `\pstMiddleAB` qu'il faut utiliser :

```
\pstMiddleAB{A}{C}{I}
```

  On définit ainsi le milieu I du segment $[AC]$.

Seconde méthode

- On trace d'abord le milieu I du segment $[AC]$ par l'instruction `\pstMiddleAB{A}{C}{I}`.
- On trace le cercle de centre I passant par le point A en utilisant l'instruction `\pstCircleOA` :

```
\pstCircleOA[linecolor=red]{I}{A}
```

1.6 Le code de la figure

Voici le code complet de la figure de la page 4 :

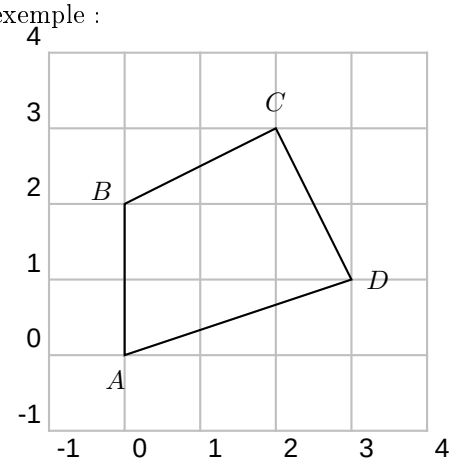
```
\def\xmin{-1.5} \def\xmax{2.5} \def\ymin{-0.5} \def\ymax{3.5}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
  \pstTriangle(0,0){A}(-1,2){B}(1,3){C}%            triangle ABC
  \pstRightAngle[linecolor=red,RightAngleSize=0.2]{A}{B}{C}% codage angle droit
  \pstSegmentMark[SegmentSymbol=MarkHashh]{A}{B}%   segment AB et codage
  \pstSegmentMark[SegmentSymbol=MarkHashh]{B}{C}%   segment BC et codage
  \pstMarkAngle[Mark=MarkCros,LabelSep=0.7]{C}{A}{B}{$45\degres$}% codage CAB
  \pstMarkAngle[Mark=MarkCros,LabelSep=0.7]{B}{C}{A}{$45\degres$}% codage BCA
  \pstCircleAB[linecolor=red]{A}{C}%                cercle de diamètre AC
```

```
  \pstGeonode[PosAngle=-45](1,0){D}%                point D
  \pstRightAngle[linecolor=red,RightAngleSize=0.2]{A}{D}{C}% codage angle droit
  \pstLineAB{A}{D}  \pstLineAB{D}{C}%               segments AD et DC
  \pstMiddleAB{A}{C}{I}%                            milieu I de AC
\end{pspicture}
```

2 D'autres outils de base

2.1 Les polygones

Pour tracer des polygones de plus de 3 sommets, c'est très simple : on définit les points et on demande de les relier pour en faire un polygone. C'est la variable `CurveType` qu'il faut utiliser. Voici un court exemple :




```
\psset{unit=1cm,PointSymbol=none}
\def\xmin{-1} \def\xmax{4}
\def\ymin{-1} \def\ymax{4}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\psgrid[subgriddiv=1,gridcolor=lightgray]
\pstGeonode
   [PosAngle={-110,150,90,0},CurveType=polygon]
   (0,0){A}(0,2){B}(2,3){C}(3,1){D}
\end{pspicture}
```

On remarque le `PointSymbol=none` dans `\psset` pour ne pas afficher les points.

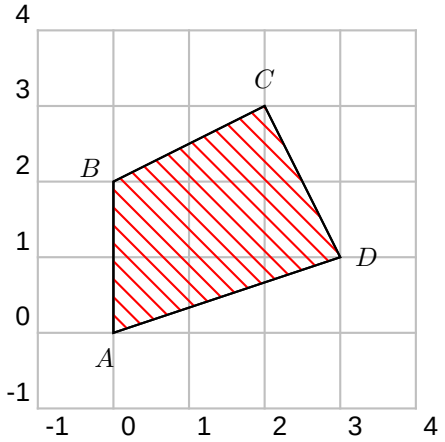
La construction est quand même rapide !
À part `polygon`, les autres options pour `CurveType` sont `polyline` (ligne brisée ouverte) et `curve` (courbe ouverte).
Comme on peut mélanger des instructions de `PsTricks` avec celles de `pst-eucl`, on va voir comment hachurer le polygone tracé précédemment. En `PsTricks`, c'est l'instruction `\pspolygon` qui permet de tracer un polygone dont les sommets sont définis par leurs coordonnées ; en utilisant conjointement `pst-all` (tous les packages de `PsTricks`) et `pst-eucl`, on peut tracer un polygone en utilisant ses sommets ; il suffit d'écrire :

```
\pspolygon(A)(B)(C)(D)
```

Attention ; les noms des points sont entourés de parenthèses (et pas d'accolades !).
On peut ainsi utiliser tous les outils de remplissage des polygones que permet `PsTricks` ; par exemple :

```
\pspolygon[fillstyle=vlines,hatchcolor=red](A)(B)(C)(D)
```

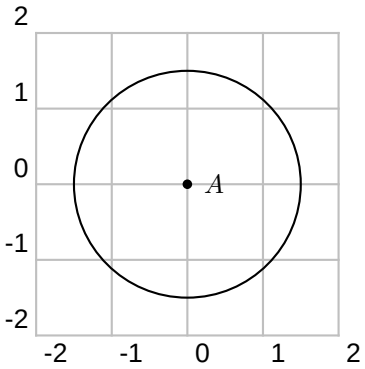
Ligne qui, ajoutée au code du polygone après la définition des points, donne :



2.2 D'autres cercles

On a vu comment tracer un cercle de diamètre donné dont on connaît les extrémités avec `pstCircleAB`, et on a vu comment tracer un cercle de centre un point donné et passant par un autre point avec `\pstCircleOA`.
Voici quatre exemples donnant d'autres façons de définir des cercles.

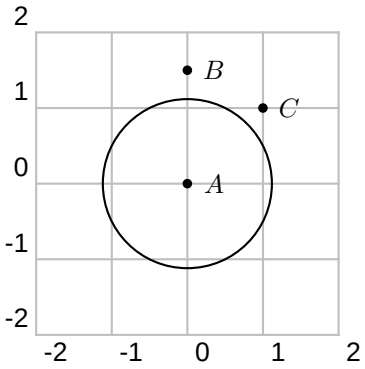
Cercle de centre A et de rayon k




```
\pstGeonode(0,0){A}
\pstCircleOA[Radius=\pstDistVal{1.5}]{A}{}
```

On définit le point A.
Comme le cercle est défini par son centre, c'est `\pstCircleOA` qu'il faut utiliser. Le rayon est défini par `Radius` et sa valeur est définie par `\pstDistVal`.
On laisse alors le second paramètre de `\pstCircleOA` vide.

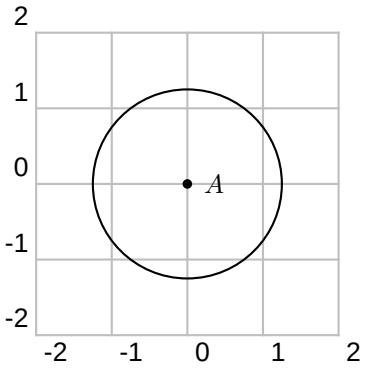
Cercle de centre A et de rayon BC




```
\pstGeonode(0,0){A}(0,1.5){B}(1,1){C}
\pstCircleOA[Radius=\pstDistAB{B}{C}]{A}{}
```

On définit les points A, B et C.
Le cercle est défini par son centre, donc on utilise `\pstCircleOA`. Le rayon est défini par `Radius` et sa valeur est la distance BC, définie par `\pstDistAB`.
Là aussi le second paramètre de `\pstCircleOA` sera laissé vide.

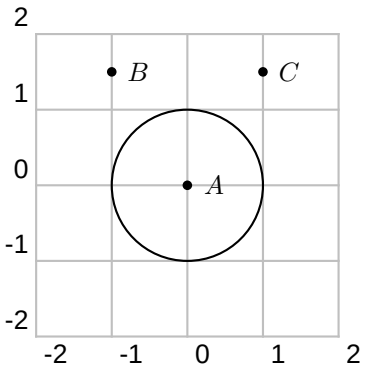
Cercle de centre A et de diamètre k




```
\pstGeonode(0,0){A}
\pstCircleOA[Diameter=\pstDistVal{2.5}]{A}{}
```

On définit le point A.
Comme le cercle est défini par son centre, c'est `\pstCircleOA` qu'il faut utiliser. Le diamètre est défini par `Diameter` et sa valeur est définie par `\pstDistVal`.
Le second paramètre de `\pstCircleOA` est encore vide.

Cercle de centre A et de diamètre BC




```
\pstGeonode(0,0){A}(-1,1.5){B}(1,1.5){C}
\pstCircleOA[Diameter=\pstDistAB{B}{C}]{A}{}
```

On définit les points A, B et C.
Le cercle est défini par son centre, donc on utilise `\pstCircleOA`. Le diamètre est défini par `Diameter` et sa valeur est la distance BC, définie par `\pstDistAB`.
Et le second paramètre de `\pstCircleOA` sera encore laissé vide.

Deuxième partie

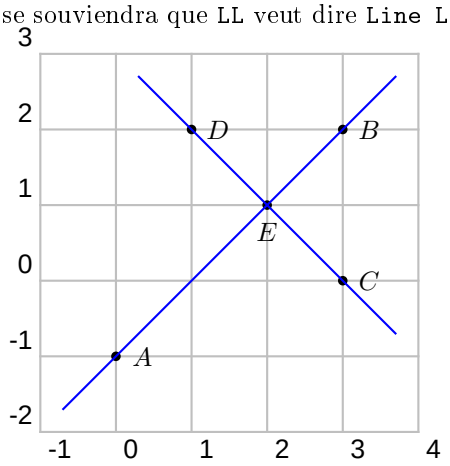
Les intersections

Le package `euclide` permet de déterminer des intersections de droites, de cercles et de courbes.

3 Intersection de droites

3.1 Figure simple

On commence par un exemple basique de détermination du point d'intersection de deux droites ; on définit quatre points A, B, C et D par `\pstGeonode`, puis on définit le point d'intersection des droites (AB) et (CD) par l'instruction `\pstInterLL`.
On se souviendra que `LL` veut dire `Line Line` pour désigner l'intersection de deux lignes.




```
\psset{unit=1cm}
\def\xmin {-1}  \def\xmax {4}
\def\ymin {-2}  \def\ymax {3}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
  \psgrid[subgriddiv=1,gridcolor=lightgray]
  \pstGeonode(0,-1){A}(3,2){B}(3,0){C}(1,2){D}
  \pstInterLL[PosAngle=-90]{A}{B}{C}{D}{E}
  \psset{linecolor=blue, nodesep=-1}
  \pstLineAB{A}{B} \pstLineAB{C}{D}
\end{pspicture}
```

L'instruction `\pstInterLL` a besoin de cinq paramètres : les deux premiers correspondent à la première droite, les deux suivants à la seconde droite, et le cinquième est le point d'intersection de ces deux droites. Attention, ce point d'intersection peut ne pas exister !
Là aussi, on peut positionner le nom du point d'intersection au moyen de la variable `PosAngle`.

Une petite nouveauté dans ce dessin, la variable `nodesep` que l'on avait déjà vue lors de la création d'arbres pondérés :

- quand elle est à 0 (valeur par défaut), la ligne tracée démarre et arrive précisément sur les points ;
- quand elle est strictement positive, la ligne démarre après le point de départ et arrive avant le point d'arrivée ;
- quand elle est négative, la ligne démarre avant le point de départ et se poursuit après le point d'arrivée. Cela permet de tracer des demi-droites, ou des droites passant par deux points que l'on connaît sans avoir à utiliser l'équation de la droite et `\psplot`.

On peut différencier le point de départ et le point d'arrivée avec les variables `nodesepA` et `nodesepB`.

Voici ce que cela donne, après que l'on a défini les points A et B par :

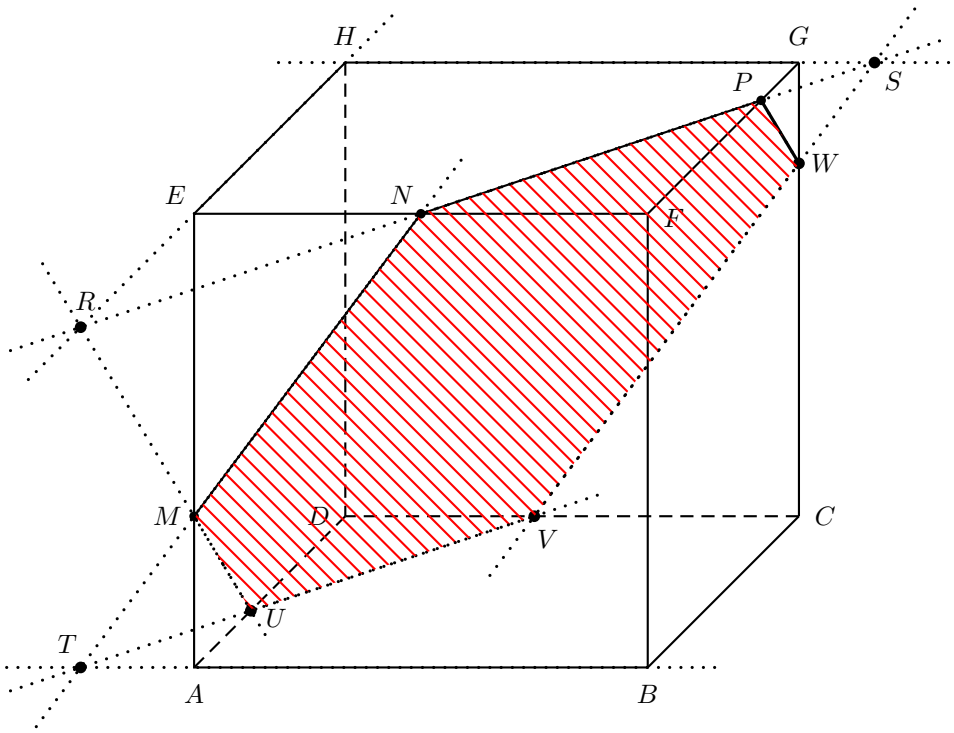
```
\pstGeonode[PosAngle={90,90}](2,0){A}(6,0){B}
```


```
\pstLineAB{A}{B}
\pstLineAB[nodesep=-1]{A}{B}
\pstLineAB[nodesepB=-1]{A}{B}
\pstLineAB[nodesep=1]{A}{B}
```

3.2 Section de cube

Pour illustrer de façon plus convaincante (si besoin est) l'intersection de deux droites, j'ai ressorti de mes cartons un vieil exercice de section de cube que l'on faisait autrefois en 1re S.
Dans le cube $ABCDEFGH$, on place les points M, N et P respectivement sur les segments $[AE]$, $[EF]$ et $[FG]$ comme sur la figure ci-dessous. On cherche la section du cube par le plan (MNP).

La méthode consiste à tracer d'autres points extérieurs au cube qui vont permettre de déterminer les intersections des arêtes du cube avec le plan (MNP). Avec `PsTricks`, il faut calculer les coordonnées des points d'intersection et tracer les bonnes droites ; avec `pst-eucl`, on peut directement définir les points comme intersections de droites puis les utiliser ensuite.



Voici le code de cette figure :

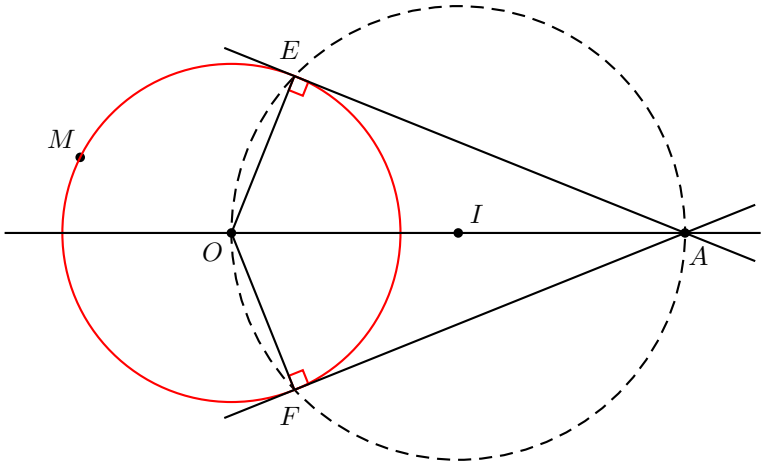
```
\psset{unit=1cm,PointSymbol=none}
\def\xmin{-2} \def\xmax{11} \def\ymin{0} \def\ymax{10}

\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)

%%% construction du cube

\pstGeonode[PosAngle={-90,-90,-10,135},CurveType=polygon]
           (1,1){A}(7,1){B}(7,7){F}(1,7){E}%%% ABFE
\pstGeonode[PosAngle={0,180,90,90}](9,3){C}(3,3){D}(3,9){H}(9,9){G}

\pstLineAB{B}{C} \pstLineAB{C}{G}
\pstLineAB{F}{G} \pstLineAB{G}{H}
\pstLineAB{H}{E}

%%% arêtes cachées

\pstLineAB[linestyle=dashed]{A}{D}
\pstLineAB[linestyle=dashed]{C}{D}
\pstLineAB[linestyle=dashed]{H}{D}

%%% placement des points M, N et P et tracé des segments MN et NP

\psset{PointSymbol=*}%%% on marque les nouveaux points
\pstGeonode[PosAngle={180,135,135},CurveType=polyline]
                    (1,3){M}(4,7){N}(8.5,8.5){P}

%%% Construction de la section (tracés en pointillés)

\psset{linestyle=dotted,linewidth=1.2pt,nodesep=-1}
\pstInterLL[PosAngle=80]{H}{E}{N}{P}{R}%    (HE) inter (NP) donne R
\pstLineAB{R}{H}
\pstInterLL[PosAngle=-45]{H}{G}{N}{P}{S}%   (HG) inter (NP) donne S
\pstLineAB{S}{H}
\pstLineAB{R}{S}
\pstInterLL[PosAngle=120]{M}{N}{A}{B}{T}%   (MN) inter (AB) donne T
\pstLineAB{T}{N}
\pstLineAB{T}{B}
\pstInterLL[PosAngle=-15]{R}{M}{A}{D}{U}%   (RM) inter (AD) donne U
\pstLineAB[nodesepB=-0.5]{R}{U}
\pstInterLL[PosAngle=-60]{T}{U}{D}{C}{V}%   (TU) inter (DC) donne V
\pstLineAB{T}{V}
\pstInterLL[PosAngle=0]{V}{S}{G}{C}{W}%     (VS) inter GC) donne W
\pstLineAB{V}{S}
\pstLineAB[linestyle=solid,nodesep=0]{P}{W}

%%%% remplissage de la section MNPWVU en hachuré et rouge

\pspolygon[fillstyle=vlines,hatchcolor=red](M)(N)(P)(W)(V)(U)

\end{pspicture}
```

4 Intersection de cercles

C'est l'instruction `\pstInterCC` qui permet de déterminer l'intersection de deux cercles ; on se souviendra que `CC` veut dire `Circle Circle`.
Chacun des cercles doit être défini par son centre et un point. L'instruction `\pstInterCC` aura besoin de six paramètres : le centre et un point du premier cercle, le centre et un point du second cercle, et les noms des deux points d'intersection des deux cercles (s'ils existent).

Pour illustrer les intersections de cercles, on va déterminer les deux tangentes à un cercle issues d'un point extérieur au cercle.

Soient O et M deux points. On appelle $\mathscr{C}$ le cercle de centre O passant par M. Soit A un point extérieur au cercle $\mathscr{C}$. On veut tracer les deux tangentes au cercle $\mathscr{C}$ passant par A ; elles coupent le cercle en E et F.



On place d'abord les trois points O, M et A au moyen de `\pstGeonode`, puis le cercle de centre O passant par M avec `\pstCircleOA`. Après avoir déterminé le milieu I de $[OA]$ avec `pstMiddleAB`, on trace le cercle de centre I passant par A (en mode tirets).
On cherche ensuite les deux points d'intersection des deux cercles par la commande `\pstInterCC`, puis on complète la figure en traçant les tangentes.

Voici le code de la figure :

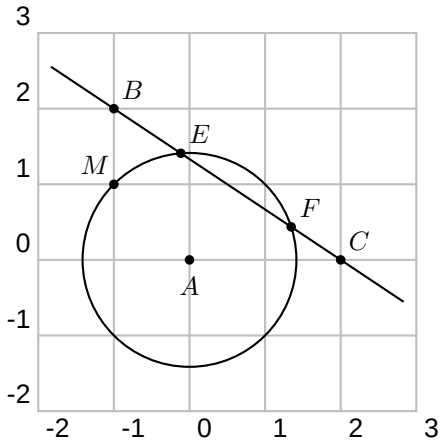
```
\psset{unit=1cm}
\def\xmin{-4} \def\xmax{7} \def\ymin{-4} \def\ymax{4}

\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)

%%% placement des points O, M et A
\pstGeonode[PosAngle={-135,-60,135}](0,0){O}(6,0){A}(-2,1){M}

%%% tracés des cercles
\pstCircleOA[linecolor=red]{O}{M}
\pstMiddleAB[PosAngle=45]{O}{A}{I}
\pstCircleOA[linestyle=dashed]{I}{A}

%%% intersection des deux cercles
```

```
\pstInterCC[PosAngleA=100,PosAngleB=-100,PointSymbol=none]{O}{M}{I}{A}{E}{F}

\pstLineAB[nodesepA=-3,nodesepB=-1]{O}{A}

%%% codages angles droits
\pstRightAngle[linecolor=red,RightAngleSize=0.2]{O}{E}{A}
\pstRightAngle[linecolor=red,RightAngleSize=0.2]{O}{F}{A}

%%% tangentes
\pstLineAB[nodesep=-1]{A}{E}
\pstLineAB[nodesep=-1]{A}{F}

%%% rayons
\pstLineAB{O}{E}
\pstLineAB{O}{F}

\end{pspicture}
```

5 Intersection droite – cercle

Cercle de centre A passant par M

Voici un premier exemple dans lequel on voit l'utilisation de `\pstInterLC` (`L` pour `Line` et `C` pour `Circle`) :

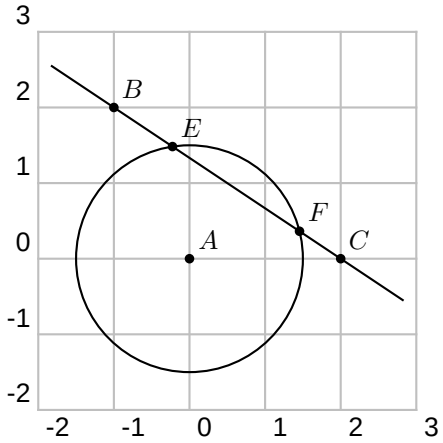



```
\psset{unit=1cm}
\def\xmin{-2}   \def\xmax{3}
\def\ymin{-2}   \def\ymax{3}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
  \psgrid[subgriddiv=1,gridcolor=lightgray]
  \pstGeonode[PosAngle={-90,45,45,135}]
             (0,0){A}(-1,2){B}(2,0){C}(-1,1){M}
  \pstCircleOA{A}{M}
  \pstLineAB[nodesep=-1]{B}{C}
  \pstInterLC[PosAngle=45]{B}{C}{A}{M}{E}{F}
\end{pspicture}
```

L'instruction `\pstInterLC` a besoin de six paramètres : deux points qui définissent la droite, le centre et le point du cercle, et les deux points d'intersection.
On retiendra que dans l'instruction `\pstInterLC`, le `L` est avant le `C`, donc on met d'abord les paramètres de la droite puis ceux du cercle.

Cercle de centre A et de rayon k

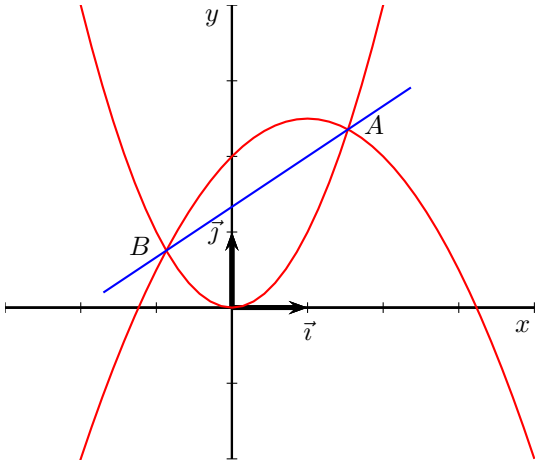
On peut tracer des cercles de différentes façons ;voici un autre exemple d'intersection . On remarque que le quatrième paramètre de `\pstInterLC` est vide puisque le cercle est défini par son centre et son rayon qui vaut ici 1,5.




```
\psset{unit=1cm,PosAngle=45}
\def\xmin{-2}   \def\xmax{3}
\def\ymin{-2}   \def\ymax{3}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
  \psgrid[subgriddiv=1,gridcolor=lightgray]
  \pstGeonode(0,0){A}(-1,2){B}(2,0){C}
  \pstCircleOA[Radius=\pstDistVal{1.5}]{A}{}
  \pstLineAB[nodesep=-1]{B}{C}
  \pstInterLC[Radius=\pstDistVal{1.5}]
             {B}{C}{A}{}{E}{F}
\end{pspicture}
```

6 Intersection de courbes

Avec `euclide` on peut déterminer les intersections de deux courbes représentant des fonctions ; c'est `\pstInterFF` qu'il faut utiliser (`FF` pour `Function Function`).
Le mode d'emploi de `pst-eucl` précise que l'intersection est déterminée par la méthode de Newton ; il faudra donc donner un point de départ à la recherche, tout en comprenant que cette recherche peut ne pas être fructueuse !
Il faut autant d'instructions `\pstInterFF` que de points d'intersection.



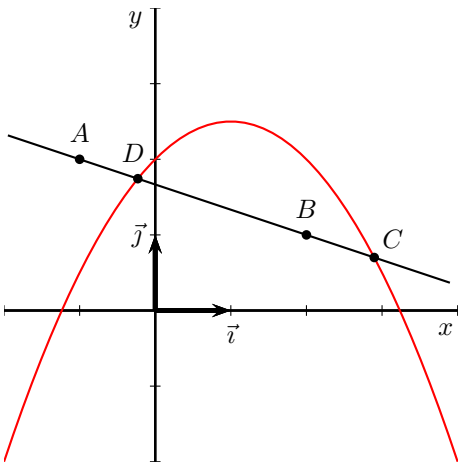
On va déterminer les points d'intersection A et B des courbes représentant les fonctions f et g définies par $f(x) = x^2$ et $g(x) = \dfrac{-x^2+2x+4}{2}$. En traçant les courbes représentant ces fonctions, on constate qu'elles ont deux points d'intersection dont les abscisses sont proches de -1 et de 1,5 ; ce sont donc ces deux valeurs qu'il faudra donner à `\pstInterFF` pour obtenir les points d'intersection. L'instruction `\pstInterFF` a besoin de quatre paramètres : le code de la première fonction, le code de la seconde, une valeur qui correspond au point de départ de la recherche par la méthode de Newton, et le nom du point d'intersection (s'il est trouvé).

Voici le code de la figure :

```
\psset{unit=1cm}
\def\xmin{-3} \def\xmax{4} \def\ymin{-2} \def\ymax{4}

\begin{pspicture*}(\xmin,\ymin)(\xmax,\ymax)

%%% axes et vecteurs
\psaxes[ticksize=-2pt 2pt, labels=none]
       (0,0)(\xmin,\ymin)(\xmax,\ymax)[$x$,-120][$y$,210]
\psaxes[linewidth=1.8pt]{->}(0,0)(1,1)[$\vec{\imath}$,d][$\vec{\jmath}$,180]

%%% définitions des deux fonctions
\def\f{x x mul}
\def\g{x -1 x mul 2 add mul 4 add 2 div}

%%% tracés des deux fonctions
\psplot[linecolor=red]{\xmin}{\xmax}{\f}
\psplot[linecolor=red]{\xmin}{\xmax}{\g}

\psset{PointSymbol=none}

%%% premier point d'intersection A en partant de l'abscisse 1,5
\pstInterFF[PosAngle=10]{\f}{\g}{1.5}{A}

%%% second point d'intersection B en partant de l'abscisse -1
\epstInterFF[PosAngle=170]{\f}{\g}{-1}{B}

%%% droite (AB)
\pstLineAB[nodesep=-1,linecolor=blue]{A}{B}

\end{pspicture*}
```

7 Intersection courbe – droite

Un exemple d'intersection courbe – droite créée par `\pstInterFL`.

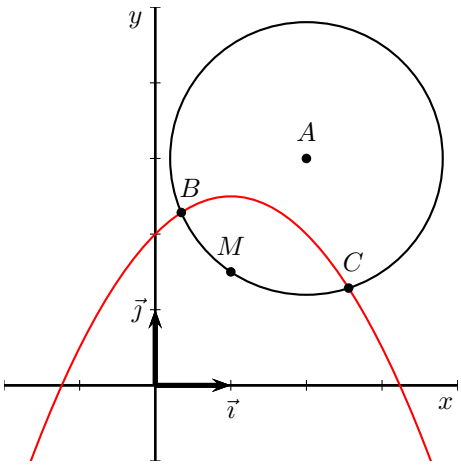



```
\psset{unit=1cm}
\def\xmin{-2}   \def\xmax{4}
\def\ymin{-2}   \def\ymax{4}
\begin{pspicture*}(\xmin,\ymin)(\xmax,\ymax)
  \psaxes[ticksize=-2pt 2pt,labels=none](0,0)
    (\xmin,\ymin)(\xmax,\ymax)[$x$,-120][$y$,210]
  \psaxes[linewidth=1.8pt]{->}(0,0)(1,1)
    [$\vec{\imath}$,d][$\vec{\jmath}$,180]
  \def\f{x -1 x mul 2 add mul 4 add 2 div}
  \psplot[linecolor=red]{\xmin}{\xmax}{\f}
  \pstGeonode[PosAngle=90](-1,2){A}(2,1){B}
  \pstLineAB[nodesepA=-1,nodesepB=-2]{A}{B}
  \pstInterFL[PosAngle=45]{\f}{A}{B}{2}{C}
  \pstInterFL[PosAngle=100]{\f}{A}{B}{0}{D}
\end{pspicture*}
```

Il faut autant d'instructions `\pstInterFL` que de points d'intersection. Cette instruction a besoin de cinq paramètres : le code de la fonction, les deux points définissant la droite, la valeur de départ de la recherche par la méthode de Newton, et le nom du point.

8 Intersection courbe – cercle

On termine cette série d'intersections par l'intersection d'une courbe et d'un ceercle ; vous avez sans doute deviné que l'instruction va s'appeler `pstInterFC` ! Et comme le `F` est avant le `C`, on mettra le code de la fonction avant les éléments de définition du cercle.
On définit et on trace la courbe représentant la fonction f qui à x associe $\dfrac{-x^2+2x+4}{2}$. On place les points $A\,(2\,;\,3)$ et $M\,(1\,;\,1{,}5)$. On trace le cercle de centre A passant par M, puis on cherche les points d'intersection de la courbe et du cercle.




```
\psset{unit=1cm}
\def\xmin{-2}   \def\xmax{4}
\def\ymin{-1}   \def\ymax{5}
\begin{pspicture*}(\xmin,\ymin)(\xmax,\ymax)
  \psaxes[ticksize=-2pt 2pt, labels=none](0,0)
         (\xmin,\ymin)(\xmax,\ymax)[$x$,-120][$y$,210]
  \psaxes[linewidth=1.8pt]{->}(0,0)(1,1)
         [$\vec{\imath}$,d][$\vec{\jmath}$,180]
  \def\f{x -1 x mul 2 add mul 4 add 2 div}
  \psplot[linecolor=red]{\xmin}{\xmax}{\f}
  \pstGeonode[PosAngle=90](2,3){A}(1,1.5){M}
  \pstCircleOA{A}{M}
  \pstInterFC[PosAngle=70]{\f}{A}{M}{0}{B}
  \pstInterFC[PosAngle=80]{\f}{A}{M}{2}{C}
\end{pspicture*}
```

La méthode de recherche est encore la méthode de Newton pour laquelle il faut entrer une valeur de départ. L'instruction `\pstInterFC` a besoin de cinq paramètres : le code de la fonction, le centre du cercle, un point du cercle, la valeur de départ de la recherche, et le nom du point d'intersection.
Il faudra autant d'instructions `\pstInterFC` que de points d'intersection.

Attention ! Pour déterminer l'intersection d'une courbe et d'un cercle, il est impératif que le cercle soit défini par son centre et un point ; on ne pourra pas faire appel à `Radius` ni à `Diameter` dans ce cas.

Troisième partie
D'autres objets

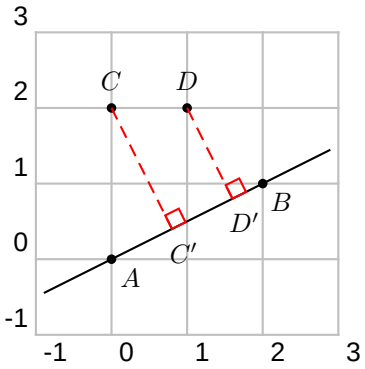
On peut faire encore plein de choses avec le package `euclide`.

9 Orthogonalité

9.1 Projection orthogonale

Avec le package `euclide`, on peut déterminer sans problème (et sans calcul !) le projeté orthogonal d'un point sur une droite, et donc tracer la droite perpendiculaire à une droite passant par un point extérieur à cette droite. C'est l'instruction `\pstProjection` qui fait ça parfaitement.
Cette commande nécessite trois paramètres, plus un optionnel que l'on met en fin de commande (ce qui est rare !). Les deux premiers paramètres sont deux points de la droite sur laquelle on va projeter. Le troisième paramètre est la liste des points dont il faut chercher les projetés.
Si on écrit `\pstProjection{A}{B}{C,D}`, on va chercher les projetés orthogonaux des points C et D sur la droite (AB) ; les projetés seront alors automatiquement nommés C' et D'.
Le paramètre optionnel de la commande sert à renommer les projetés si C' et D' ne nous conviennent pas : si on veut appeler E et F respectivement les projetés de C et D sur la droite (AB), on écrira `\pstProjection{A}{B}{C,D}[E,F]`. Le paramètre optionnel s'écrit avec des crochets.

Voyons un petit exemple :

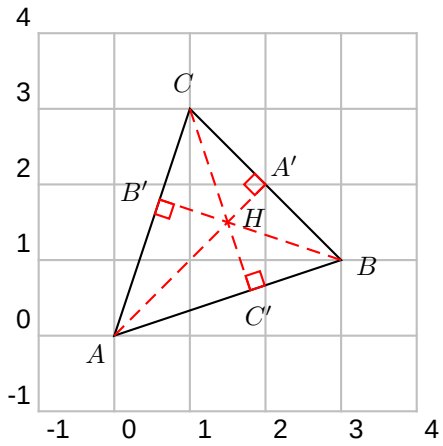



```
\psset{unit=1cm}
\def\xmin{-1}   \def\xmax{3}
\def\ymin{-1}   \def\ymax{3}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
  \psgrid[subgriddiv=1,gridcolor=lightgray]
  \pstGeonode[PosAngle={-45,-45,90,90}]
             (0,0){A}(2,1){B}(0,2){C}(1,2){D}
  \pstLineAB[nodesep=-1]{A}{B}
  \pstProjection[CodeFig=true,CodeFigColor=red,
     PointSymbol=none,RightAngleSize=0.2]{A}{B}{C,D}
\end{pspicture}
```

Après avoir défini les points A, B, C et D, on trace la droite (AB).
Regardons de plus près les options de l'instruction `\pstProjection`.
Le booléen `CodFig=true` trace automatiquement le segment en mode traits entre le point et son projeté, et marque l'angle droit. On comprend bien ainsi en quoi consiste le projeté orthogonal !
On comprend vite que `CodeFigColor` spécifie la couleur du codage de la figure.
Quant à `RightAngleSize` et `PointSymbol`, ils ont été vus dans une précédente partie.

9.2 Hauteurs et orthocentre

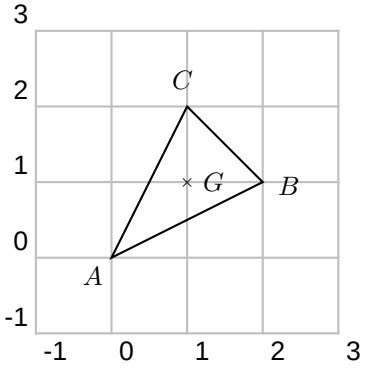
On peut maintenant tracer les trois hauteurs d'un triangle ainsi que son orthocentre :




```
\psset{unit=1cm,PointSymbol=none}
\def\xmin{-1}   \def\xmax{4}
\def\ymin{-1}   \def\ymax{4}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
  \psgrid[subgriddiv=1,gridcolor=lightgray]
  \pstTriangle(0,0){A}(3,1){B}(1,3){C}
  \psset{CodeFig=true,CodeFigColor=red,
         RightAngleSize=0.2}
  \pstProjection{A}{B}{C}%       définit C'
  \pstProjection{B}{C}{A}%       définit A'
  \pstProjection{C}{A}{B}%       définit B'
  \pstInterLL[PosAngle=10]{A}{A'}{B}{B'}{H}
\end{pspicture}
```

10 Centre de gravité

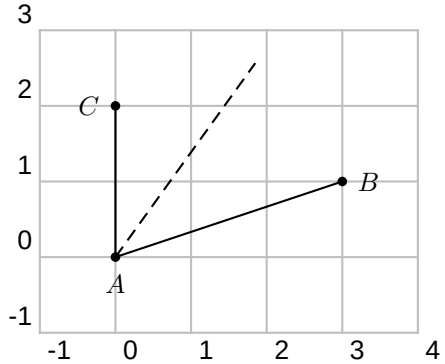
On a déjà vu comment déterminer le milieu d'un segment avec `\pstMiddleAB` ; on peut ainsi tracer les médianes d'un triangle et déterminer son centre de gravité par intersection de deux médianes. Mais il y a une instruction qui fait ça directement : c'est `\pstCGravABC`.
Quatre paramètres pour cette instruction : les trois premiers sont les trois sommets du triangle, et le quatrième est le nom du centre de gravité.
Le booléen `CodeFig` n'a aucun effet dans cette construction.




```
\psset{unit=1cm}
\def\xmin{-1}   \def\xmax{3}
\def\ymin{-1}   \def\ymax{3}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
  \psgrid[subgriddiv=1,gridcolor=lightgray]
  \pstTriangle[PointSymbol=none]
              (0,0){A}(2,1){B}(1,2){C}
  \pstCGravABC[PointSymbol=x]{A}{B}{C}{G}
\end{pspicture}
```

11 Bissectrices

Tracer la bissectrice d'un angle est un jeu d'enfant avec l'instruction `\pstBissectBAC` :

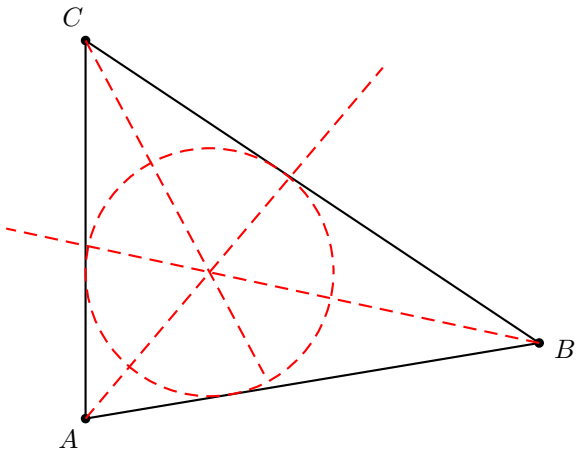



```
\psset{unit=1cm}
\def\xmin{-1}   \def\xmax{4}
\def\ymin{-1}   \def\ymax{4}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
  \psgrid[subgriddiv=1,gridcolor=lightgray]
  \pstGeonode[CurveType=polyline,PosAngle={0,-90,180}]
             (3,1){B}(0,0){A}(0,2){C}
  \pstBissectBAC[PointSymbol=none,PointName=none,
                 linestyle=dashed]{B}{A}{C}{A'}
\end{pspicture}
```

L'instruction `\pstBissectBAC` a besoin de quatre paramètres : les trois sommets de l'angle dont on veut la bissectrice, un quatrième point qui est obtenu par rotation (et qui permet de tracer la bissectrice).

On peut ne pas faire afficher ce quatrième point comme dans l'exemple en entrant comme option : `PointName=none`. Si on a donné à `PointName` la valeur `none`, on peut refaire afficher le nom des points en entrant `PointName=default`.
Les angles sont pris dans le sens trigonométrique donc si l'on entre `\pstBissectBAC{B}{A}{C}{A'}`, on n'aura pas le même dessin qu'avec `\pstBissectBAC{C}{A}{B}{A'}`. À essayer !

Vous avez donc en main tous les outils pour tracer les trois bissectrices des angles d'un triangle, et le cercle inscrit dans ce triangle :

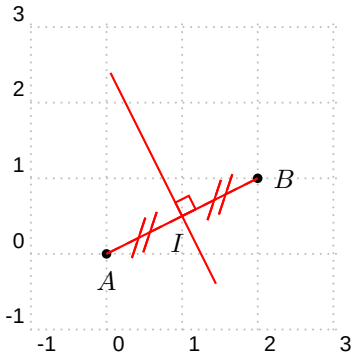


Enfin, pour tracer les cercles exinscrits au triangle, on utilisera l'instruction `\pstOutBissectBAC` qui trace une bissectrice extérieure au triangle.

12 Cercle circonscrit

12.1 Médiatrice d'un segment

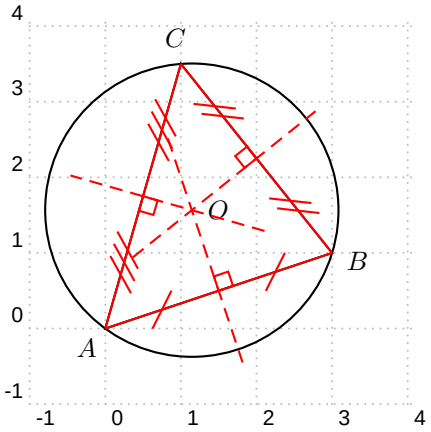
On peut tracer la médiatrice d'un segment au moyen de l'instruction `\pstMediatorAB`.
Voyons comment cette commande trace la médiatrice d'un segment `[`AB`]` : elle détermine le milieu du segment (dont il faut donner le nom, par exemple I) et un deuxième point (dont il faut également donner le nom, par exemple M_I) qui est l'image du deuxième point du segment (ici B) par la rotation de centre I et d'angle 90°. Enfin cette macro-commande trace le segment `[`IM_I`]`.
L'instruction `\pstMediatorAB` aura donc besoin de quatre paramètres : les extrémités du segment dont on veut la médiatrice, le nom du milieu du segment, et le nom du point de construction de la médiatrice. Cette instruction crée donc deux points I et M_I dont les attributs sont contrôlés respectivement par `PointNameA` et `PointSymbolA`, `PointNameB` et `PointSymbolB`. Comme le point M_I est un point de construction, il n'a pas vocation à apparaître dans la figure ; on écrira donc : `PointNameB=none` et `PointSymbolB=none` pour avoir une figure « propre ».
Le booléen `CodeFig` peut être activé dans cette construction de médiatrice.
Au passage, on remarquera que la grille est tracée au moyen du booléen `showgrid` qui est passé en option dans `\pspicture` (et qui veut dire `showgrid=true`).
Savez-vous comment on dit « médiatrice » en anglais ? C'est « perpendicular bisector ». Et quand on voit les similitudes entre les constructions (à la règle et au compas) d'une bissectrice d'un angle et d'une médiatrice d'un segment, on comprend pleinement le bien-fondé de cette appellation !




```
\psset{unit=1cm,CodeFig=true,CodeFigColor=red}
\def\xmin{-1}   \def\xmax{3}
\def\ymin{-1}   \def\ymax{3}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
  \psgrid[subgriddiv=1,gridcolor=lightgray]
  \pstGeonode[PosAngle={-90,0}](0,0){A}(2,1){B}
  \psset{PointNameB=none,PointSymbolB=none,
         PointSymbolA=none,PosAngle=-100}
  \pstMediatorAB[nodesep=-1,linecolor=red,
    RightAngleSize=0.2]{A}{B}{I}{M_I}
\end{pspicture}
```

12.2 Cercle circonscrit

En utilisant deux des trois médiatrices des côtés d'un triangle, on peut maintenant déterminer le centre du cercle circonscrit au triangle, et tracer ce cercle. On va utiliser l'instruction `\pstCircleABC`. Quatre paramètres sont nécessaires : les trois sommets du triangle et le centre du cercle circonscrit.



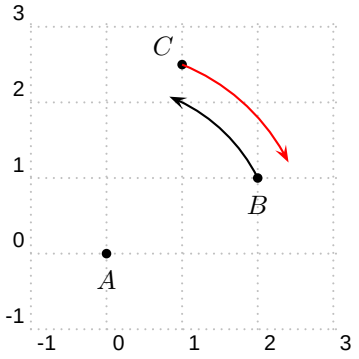

```
\psset{unit=1cm,PointSymbol=none}
\def\xmin{-1}   \def\xmax{4}
\def\ymin{-1}   \def\ymax{4}
\begin{pspicture}[showgrid]
       (\xmin,\ymin)(\xmax,\ymax)
  \pstTriangle(0,0){A}(3,1){B}(1,3.5){C}
  \psset{CodeFig=true,CodeFigColor=red,
         RightAngleSize=0.2}
  \pstCircleABC{A}{B}{C}{O}
\end{pspicture}
\end{pspicture}
```

Pour tracer les trois médiatrices des côtés sans tracer le cercle, on utilisera le booléen `DrawCirABC` :

```
\pstCircleABC[DrawCirABC=false]{A}{B}{C}{O}
```

13 Arc de cercle

On peut, avec `PsTricks` tracer un arc, avec ou sans flèche, mais il faut donner la mesure de l'angle en degrés. Avec `euclide`, on peut également tracer un arc en utilisant trois points ; l'instruction `\pstArcOAB` nécessite trois points $\{A\}$, $\{B\}$, $\{C\}$, et trace l'arc de cercle correspondant à l'angle $\left(\overrightarrow{AB},\overrightarrow{AC}\right)$ sur le cercle de centre A passant par B :




```
\psset{unit=1cm,arrowsize=3pt 2}
\def\xmin{-1}   \def\xmax{3}
\def\ymin{-1}   \def\ymax{3}
\begin{pspicture}[showgrid](\xmin,\ymin)(\xmax,\ymax)
  \pstGeonode[PosAngle={-90,-90,135}]
             (0,0){A}(2,1){B}(1,2.5){C}
  \pstArcOAB[arrows=->]{A}{B}{C}
  \pstArcnOAB[arrows=->,linecolor=red]{A}{C}{B}
\end{pspicture}
```

Pour avoir un arc dans l'autre sens, on utilise `\pstArcnOAB`.

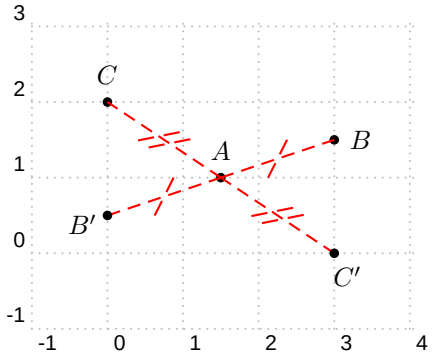
Quatrième partie

Transformations

Dernière partie de la présentation de l'excellent package `euclide`, consacrée aux transformations.

14 Symétrie centrale

Un centre A, deux points B et C, et leurs images B' et C' par la symétrie de centre A, composent la figure ci-dessous. L'instruction `\pstSymO` a besoin de deux paramètres : le centre de la symétrie, la liste des points dont il faut chercher les symétriques. Il faut avoir défini ces points préalablement.




```
\psset{unit=1cm,CodeFig=true,CodeFigColor=red}
\def\xmin{-1}   \def\xmax{4}
\def\ymin{-1}   \def\ymax{3}
\begin{pspicture}[showgrid]
      (\xmin,\ymin)(\xmax,\ymax)
  \pstGeonode[PosAngle={90,0,90}]
          (1.5,1){A}(3,1.5){B}(0,2){C}
  \pstSymO[PosAngle={200,-60}]{A}{B,C}
\end{pspicture}
```

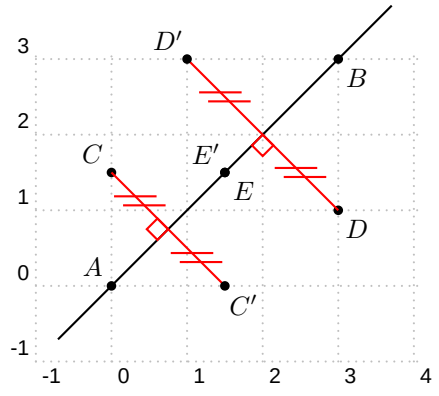
Tout comme pour la projection orthogonale (voir page 18), les images de B et de C sont automatiquement appelées B' et C' ; si on veut les appeler respectivement D et E on écrira :

```
\pstSymO[PosAngle={200,-60}]{A}{B,C}[D,E]
```

En activant le booléen `CodeFig`, le codage de la figure est rajouté ; la couleur de ce codage est gérée par `CodeFigColor`.

15 Symétrie orthogonale

On définit une symétrie orthogonale par rapport à une droite très simplement en utilisant l'instruction `\pstOrtSym`. Cette instruction a besoin de trois paramètres : les deux premiers désignent la droite qui sert d'axe de symétrie, et le troisième est la liste des points dont on veut les images.
Là aussi, on peut activer le codage de la figure par le booléen `CodeFig`. Comme dans les autres exemples, le codage est en mode `tirets` (`dashed`) ; si on veut changer ce style, on utilisera la variable `CodeFigStyle` comme dans l'exemple ci-dessous.
Si on cherche l'image d'un point situé sur l'axe (AB) (comme le point E de la figure), il vaut mieux désactiver le booléen `CodeFig` en le mettant à `false`.
Enfin comme les angles droits sont automatiquement codés quand `CodeFig` est à `true`, on réduira la taille du codage au moyen de `RightAngleSize`.



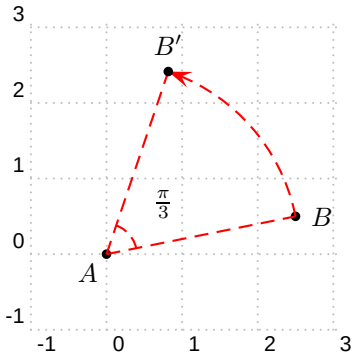

```
\psset{unit=1cm,CodeFig=true,CodeFigColor=red,
       CodeFigStyle=solid}
\def\xmin{-1}   \def\xmax{4}
\def\ymin{-1}   \def\ymax{3}
\begin{pspicture}[showgrid](\xmin,\ymin)(\xmax,\ymax)
  \pstGeonode[PosAngle={135,-45,135,-45}]
       (0,0){A}(3,3){B}(0,1.5){C}(3,1){D}(1.5,1.5){E}
  \pstLineAB[nodesep=-1]{A}{B}
  \pstOrtSym[RightAngleSize=0.2,PosAngle={-45,135}]
          {A}{B}{C,D}
  \pstOrtSym[CodeFig=false,PosAngle=135]{A}{B}{E}
\end{pspicture}
```

16 Rotation

Pas de difficulté non plus avec la rotation.
L'angle de la rotation est défini par `RotAngle`.

16.1 Angle défini par une valeur

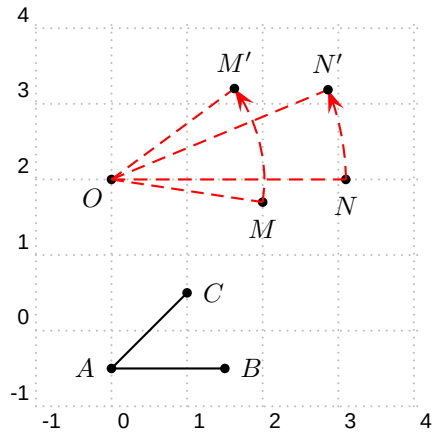
On donne une valeur (en degrés) à l'angle de la rotation : `RotAngle=60`. On passe cette affectation dans `\psset` ou en option dans l'instruction `\pstRotation`. Une fois l'angle défini, cette instruction a besoin de deux paramètres : le centre de la rotation, et la liste des points dont on veut les images.
Le booléen `CodeFig` permet le codage de la figure en traçant un arc de cercle fléché du point vers son image. On modifie la taille des flèches au moyen de la variable `arrowscale`.
On peut faire afficher la mesure de l'angle de la rotation grâce à la variable `TransformLabel`.




```
\psset{unit=1cm,CodeFig=true,CodeFigColor=red}
\def\xmin{-1}   \def\xmax{3}
\def\ymin{-1}   \def\ymax{3}
\begin{pspicture}[showgrid](\xmin,\ymin)(\xmax,\ymax)
  \pstGeonode[PosAngle={-135,0}](0,0){A}(2.5,0.5){B}
  \pstRotation[RotAngle=60,arrowscale=2,
      PosAngle=90,TransformLabel=\frac{\pi}{3}]{A}{B}
\end{pspicture}
```

16.2 Angle défini par trois points

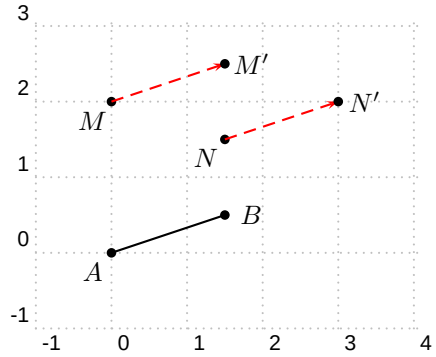
L'angle de la rotation peut être défini au moyen de trois points de la figure ; c'est alors l'instruction `\pstAngleAOB` qu'il faut utiliser.
Si on utilise plusieurs fois l'angle (comme dans l'exemple ci-dessous), on peut le stocker dans une variable par l'instruction `\def`.
On peut également multiplier l'angle précédemment défini par un coefficient `AngleCoef` ; seule contrainte : il faut définir le coefficient avant de définir l'angle par `RotAngle`.
Dans l'exemple qui suit, le point M a pour image M' dans la rotation de centre O et d'angle $\left(\overrightarrow{AB},\overrightarrow{AC}\right)$; le point N a pour image N' dans la rotation de centre O et d'angle $\frac{1}{2}\left(\overrightarrow{AB},\overrightarrow{AC}\right)$.




```
\psset{unit=1cm,CodeFig=true,CodeFigColor=red}
\def\xmin{-1}   \def\xmax{4}
\def\ymin{-1}   \def\ymax{4}
\begin{pspicture}[showgrid](\xmin,\ymin)(\xmax,\ymax)
  \pstGeonode[PosAngle={0,180,0},CurveType=polyline]
      (1.5,-0.5){B}(0,-0.5){A}(1,0.5){C}
  \def\angle{\pstAngleAOB{B}{A}{C}}
  \pstGeonode[PosAngle={-135,-90,-90}]
      (0,2){O}(2,2){M}(3.1,2.4){N}
  \psset{PosAngle=90,arrowscale=2}
  \pstRotation[RotAngle=\angle]{O}{M}
  \pstRotation[AngleCoef=0.5,RotAngle=\angle]{O}{N}
\end{pspicture}
```

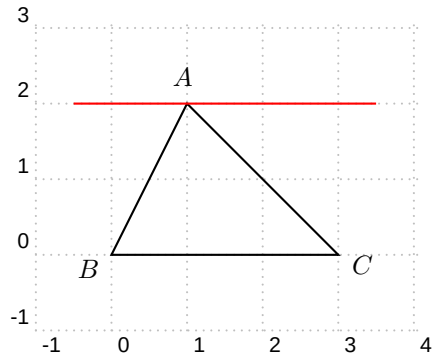
17 Translation

Deux points permettent de définir une translation ; ce sont les deux premiers paramètres qu'il faudra donner à `\pstTranslation`, le troisième étant la liste des points dont on veut les images.




```
\psset{unit=1cm,CodeFig=true,CodeFigColor=red}
\def\xmin{-1}   \def\xmax{4}
\def\ymin{-1}   \def\ymax{3}
\begin{pspicture}[showgrid](\xmin,\ymin)(\xmax,\ymax)
  \pstGeonode[PosAngle={-135,0,-135,-135}]
          (0,0){A}(1.5,0.5){B}(0,2){M}(1.5,1.5){N}
  \pstLineAB{A}{B}
  \pstTranslation{A}{B}{M,N}
\end{pspicture}
```

L'option `CodeFig=true` trace les segments $[MM']$ et $[NN']$.
On peut utiliser une translation pour tracer une droite passant par un point donné et parallèle à une droite donnée, comme dans l'exemple ci-dessous.
Soit ABC un triangle. On veut tracer la droite Δ passant par A et parallèle à (BC).

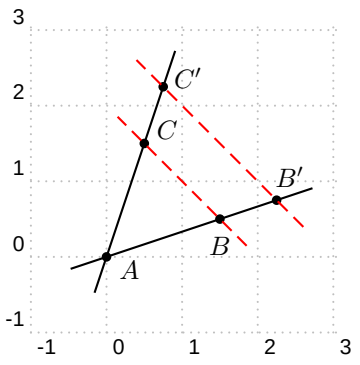



```
\psset{unit=1cm,PointSymbol=none}
\def\xmin{-1}   \def\xmax{4}
\def\ymin{-1}   \def\ymax{3}
\begin{pspicture}[showgrid](\xmin,\ymin)(\xmax,\ymax)
  \pstTriangle(0,0){B}(3,0){C}(1,2){A}
  \pstTranslation[PointName=none]{B}{C}{A}
  \pstLineAB[nodesepA=-1.5,nodesepB=0.5,
                linecolor=red]{A}{A'}
\end{pspicture}
```

Dans cet exemple, le point A' n'est pas dessiné dans la figure (`PointName=none`) mais on peut l'utiliser pour tracer la parallèle à (BC) passant par A.

18 Homothétie

Un centre et un rapport permettent de définir une homothétie. Le rapport sera défini par `HomCoef` et sera passé en option (obligatoire!) dans l'instruction `\pstHomO`; cette instruction nécessite deux paramètres : le centre de l'homothétie, et la liste des points dont on veut les images. Par défaut, l'image du point B sera appelée B' sauf si on entre comme option en fin de traitement un autre nom, comme dans les autres transformations étudiées dans cette partie.




```
\psset{unit=1cm}
\def\xmin{-1}   \def\xmax{3}
\def\ymin{-1}   \def\ymax{3}
\begin{pspicture}[showgrid](\xmin,\ymin)(\xmax,\ymax)
  \pstGeonode[PosAngle={-30,-90,30}]
             (0,0){A}(2,0.5){B}(0.5,1.5){C}
  \pstHomO[HomCoef=1.5,PosAngle={60,20}]{A}{B,C}
  \psset{linecolor=red,linestyle=dashed,nodesep=-0.5}
  \pstLineAB{B}{C} \pstLineAB{B'}{C'}
  \psset{linestyle=solid,linecolor=black}
  \pstLineAB{A}{B'}  \pstLineAB{A}{C'}
\end{pspicture}
```

L'option `CodeFig=true` est sans effet dans le cas d'une homothétie.

19 Références

Rappelons le nom de l'auteur de ce package fort efficace : Dominique Rodriguez.
Puis il a été aidé dans son développement par Herbert Voß.

Le premier mode d'emploi de ce package était en français et on peut le trouver en cliquant ici; il date de 2005.
La dernière version, de 2015, est en anglais et on la trouve ici.

Vous retrouverez dans ces documents tout ce dont je vous ai parlé au cours de ces quatre parties consacrées au package `euclide`, et même un peu plus.

Index

R

S

T

V