Macros de géométrie plane avec `Asymptote`

1 Préambule.

1.1 Les dimensions.

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

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

la constante `mainlevee` de type `bool` à la valeur `true` permet d'avoir un dessin «à main levée».

2 Les points

2.1 Tracé

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

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

2.2 Construction

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

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

▶ `point_angle_dist(pair O,pair A, real a, real d)` place le point situé à d cm de O et faisant un angle d avec (OA).

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

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

2.3 Points particuliers

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

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

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

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

3 Mesure et codage

3.1 Cotation

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

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

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

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

3.2 Codage des longeurs et des angles

▶ `code(int trait,pair[] K)`code une série de segments dont les extrémités sont contenues dans une matrice de type «pair[]». Le paramètre "trait" précise le codage :

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

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

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

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

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

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

4 Quadrillages

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

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

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

5 triangle et quadrilatères

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

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

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

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

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

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

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

6 Droites et segments

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

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

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

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

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

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

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

7 Les cercles

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

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

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

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

8 Repérage

8.1 Axes gradués

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

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

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

`crochet` fonctionne de la même façon.

▶ `void graduation(pair origine, pair unite, real debut, real fin, string originetxt="`O`", string unitetxt="1", real intermediaire=0,pen sty=currentpen)` trace un axe gradué, les valeurs `début` et `fin` sont exprimés en fonction du vecteur unité, `intermediaire` est une fraction du vecteur unité.

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

8.2 Repères

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

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

`origine` contient les coordonnées de l'origine.

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

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

`originetxt` contient le nom de l'origine.

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

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

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

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

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

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

8.3 Les points

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

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

9 Les courbes et fonctions

9.1 Courbes

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

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

9.2 Fonctions

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

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

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