I - Avant de commencer

XCAS est un logiciel libre de calcul formel. En particulier, vous allez pouvoir le télécharger gratuitement en toute légalité sur

`http://www-fourier.ujf-grenoble.fr/~parisse/irem.html`

Nous verrons plus tard ce que veut dire calcul formel. Pour aujourd'hui, nous commencerons doucement puisque le but de notre séance est de créer un programme qui teste l'appartenance d'un point de coordonnées (X,Y) à une droite d'équation $y = ax + b$. On veut qu'il nous donne la réponse si on se contente de lui rentrer dans l'ordre les coefficients a et b puis les coordonnées X et Y.

II - Allumons l'ordinateur

Je pense que vous en êtes capables...
Vous avez cliqué sur l'icône correspondant à **XCAS**. Vous voyez normalement ceci, à la couleur près :

Si tel n'est pas le cas, allez dans le menu `Cfg->configuration du CAS->niveau` et choisissez `Universite` car vous êtes très fort(e).
Dans ce même onglet `Cfg`, vous pouvez modifier la taille de la police, les couleurs.
Nous voici prêts à programmer.

III - Que voulons-nous que l'ordinateur fasse ?

Nous voudrions donner uniquement comme renseignements à l'ordinateur les coefficients a et b de l'équation réduite de la droite et les coordonnées (X,Y) du point à tester. Pour que **XCAS** comprenne ce que l'on veut de lui, il faudra parler son langage.

IV - Procédure

Une procédure est une sorte de fonction informatique qui a un nom, un début et une fin et qui dépend d'un certain nombre de variables fournis par l'utilisateur. On peut par exemple appeler notre procédure `pointsurdroite` et son squelette ressemblera à ça :

```
pointsurdroite(a,b,X,Y):={   // c'est le titre du programme...
                             // On indique qu'on rentrera dans l'ordre a,b,X et Y
                             // il faudra rentrer des instructions ici
  }                          // marque la fin du programme
```

V - Fonction

Notre but est de vérifier si $2X + 1 = Y$. Il faut donc d'abord faire calculer $2X + 1$ par l'ordinateur. On introduit donc une fonction comme on a l'habitude de le faire en cours. Donnons un nom à la fonction affine associée à la droite : soit $f : x \mapsto 2x + 1$. La syntaxe est pratiquement la même pour **XCAS** :

```
f:=x->2*x+1
```

Alors, pour calculer par exemple l'image de 4 par `f` on rentre

```
f(4)
```

Ben oui, $2\times4+1=9$.

VI - SI...ALORS...SINON

Que faites-vous pour savoir si un point appartient à une droite donnée ?
SI ..... ALORS ..... SINON.....
XCAS parle presque la même langue :

```
si (la condition à tester)
alors action 1;
sinon action2;
fsi;      // pour dire que c'est fini...
```

Nous y sommes presque....

VII - Variables locales

La variable `f` que nous introduisons *à l'intérieur* du programme n'est pas rentrée par l'utilisateur : c'est l'ordinateur qui la crée suite à nos instructions. On n'en a plus besoin en dehors du programme et on va donc libérer la case mémoire où elle se trouvait. On dit que c'est une variable locale.
On l'introduit par :

```
local f;
```

VIII - Le programme...

Récapitulons :
– Une ligne pour introduire le programme

```
pointsurdroite(a,b,X,Y):={
```

– une ligne pour introduire la variable locale :

```
local f;
```

– une ligne pour dire qui est `f` :

```
f:=x->a*x+b
```

En effet, on rentre f de la manière la plus générale car on veut pouvoir traiter toutes les fonctions affines.
– le cœur du programme :

```
si (f(X)==Y) alors "le point appartient à la droite";
sinon "le point n'appartient pas à la droite";
fsi;
```

L'égalité sur **XCAS** est traitée par ==.
Pour faire comprendre qu'on rentre une phrase que l'ordinateur n'a pas besoin de lire mais qui est juste là pour aider l'utilisateur, on la met entre guillemets " ".
– on ferme la procédure :

```
};;
```

IX - Comment l'utiliser ?

Pour savoir si le point de coordonnées (2 ;5) appartient à la droite d'équation $y=3x+1$, on rentre

```
pointsurdroite(3,1,2,5)
```

Faites d'autres essais.

X - Un peu d'esthétique...

Au lieu de répondre impersonnellement « le point appartient à la droite », on aimerait obtenir la réponse « le point de coordonnées (2,5) appartient à la droite d'équation $y = 3x + 1$ ».
Ça devrait être possible car les coordonnées du point et les coefficients de l'équation sont stockés par **XCAS** sous les appellations a, b, X et Y.
Il suffit ensuite de savoir que **XCAS** « colle [a] » les morceaux de « phrases » à l'aide du symbole +.
Cela donne donc :

```
"Le point de coordonnées (" + X +" , "+ Y +") appartient à la droite d'équation y="+ a +"x +"+ b;
```

On ne met pas X entre guillemets car X doit être lu par **XCAS** et traduit par la valeur qu'on aura rentré en 3ème position.

XI - Tracé

On peut aussi demander d'afficher le point de coordonnées (X,Y) et la droite d'équation $y = ax + b$. Pour cela, on utilise `point` et `graphe` :

```
D:=graphe(f(t),t);
A:=point(X,Y);
```

Pour régler l'affichage du point en plus gros pour bien le voir en rouge :

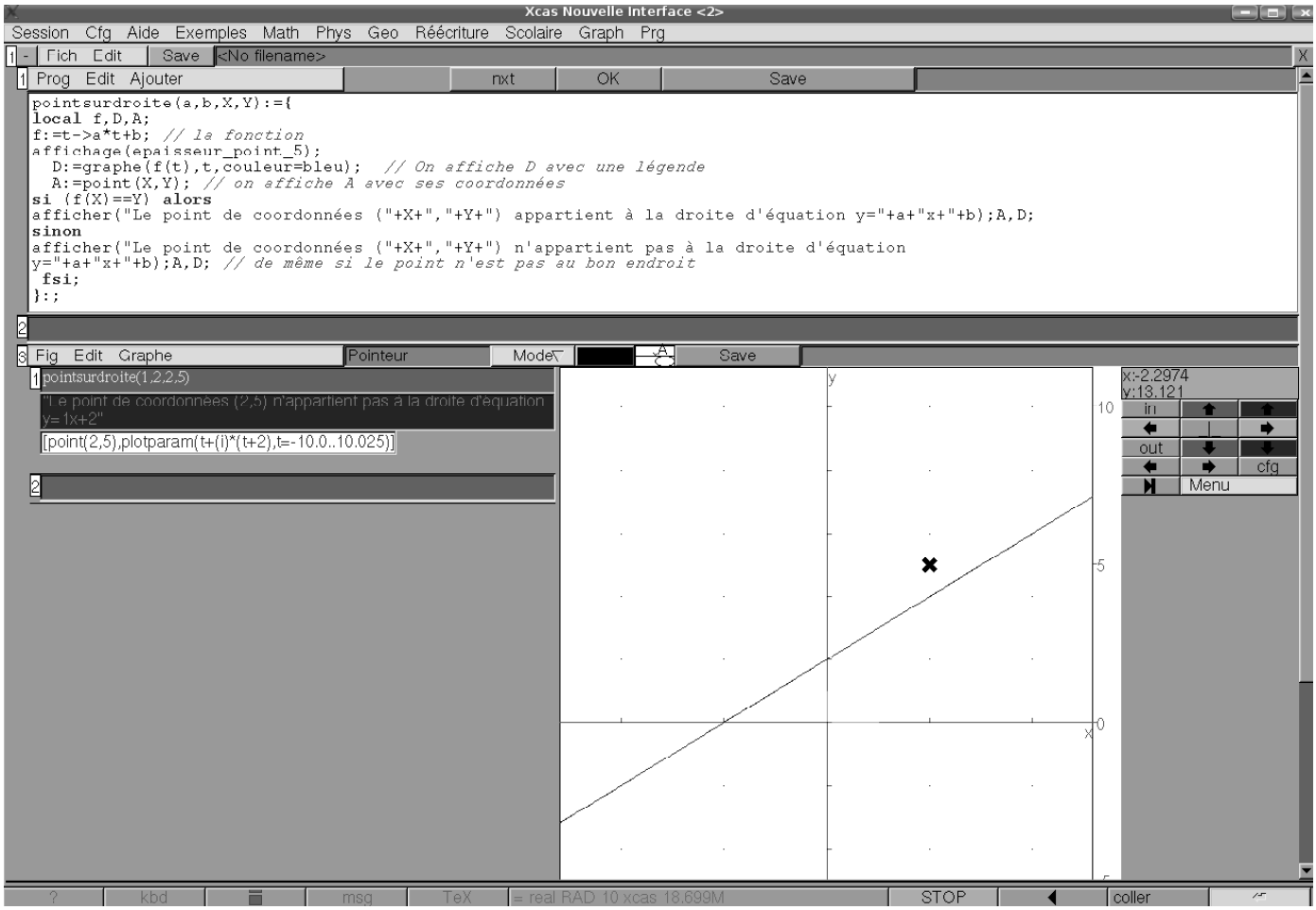
```
affichage(epaisseur_point_5+rouge);
```

Le programme devient alors :

```
pointsurdroite(a,b,X,Y):={
local f,D,A;
f:=t->a*t+b; // la fonction
affichage(epaisseur_point_5); // le point épai
  D:=graphe(f(t),t,couleur=bleu);  // On affiche D
  A:=point(X,Y); // on affiche A
si (f(X)==Y) alors
afficher("Le point de coordonnées ("+X+","+Y+") appartient à la droite d'équation y="+a+"x+"+b);A,D;
sinon
afficher("Le point de coordonnées ("+X+","+Y+") n'appartient pas à la droite d'équation
y="+a+"x+"+b);A,D; // de même si le point n'est pas au bon endroit
 fsi;
}:;
```

On a rajouté `afficher( )` car nous demandons à **XCAS** de retourner à la fois la phrase et un graphique : `afficher( )` permet en fait d'afficher un résultat intermédiaire.
Le résultat est affiché en-dessous :

[a] On dit concatène



XII - Sans parachute...

À vous d'imaginer une procédure `AppartientCourbe(expression,X,Y)` qui teste si un point de coordonnées (X,Y) appartient à la courbe d'équation y=expression.