Pourquoi et comment utiliser XCAS au lycée

I Introduction

XCAS existe depuis quelques années mais est très mal connu au lycée car sa puissance fait peur. **XCAS** permet en effet de traiter des problèmes très sophistiqués, mais, ce qu'on passe souvent sous silence, des problèmes très simples qui permettent par exemple d'initier des élèves de l'école primaire à la programmation !
Nous commencerons par aborder la géométrie dynamique au lycée car c'est un domaine où une comparaison avec les logiciels massivement utilisés par les professeurs du secondaire est possible, ce qui n'est pas le cas du calcul formel, de la programmation et surtout la possibilité d'associer les trois plus le tableur qui constitue une des grandes forces de **XCAS**.
Nous poursuivrons par un sujet de l'épreuve pratique de Terminale S qui défraie la chronique actuellement et effraie nombre de collègues.
Nous traiterons ensuite un problème que ni un logiciel de géométrie classique, ni un tableur ne peuvent traiter et qui est pourtant très intéressant mathématiquement.
Nous finirons par un exemple de TP donné en Terminale S et qui utilise un soupçon d'algorithmique.

II XCAS et la géomérie dynamique au lycée

Un argument souvent soulevé par ses détracteurs est que **XCAS** fonctionne principalement en ligne de commande. Notre expérience avec les élèves nous incite à dire qu'il ne s'agit pas d'un inconvénient mais plutôt d'un avantage pour eux : les élèves de nos classes sont en effet habitués à dialoguer via leur ordinateur sur les sites de messagerie instantanée. Le petit bémol qu'on pourrait apporter est qu'avec **XCAS**, il ne faut pas faire de fautes d'orthographe !
Les commandes sont de plus en français et adaptées aux besoins des élèves français car créées par des professeurs de l'Institut Joseph FOURIER de l'Universite Joseph FOURIER de Grenoble et testées grâce à l'IREM de Grenoble. Que dites-vous de :
`parallele, droite, segment, aire, longueur, angle, resoudre, graphe, cercle, circonscrit, simplifier, factoriser, fonction_derivee,symetrie, projection,...` ?
Le mode de construction d'une figure et de résolution du problème correspond de même à la démarche suivie par l'élève sur sa feuille de papier.

Pour illustrer notre propos, considérons par exemple le problème classique suivant donné en classe de Seconde :

On considère un triangle ABC rectangle en A tel que AC = 3 *et* AB = 4. *Soit M un point quelconque du segment [AC]. On construit le rectangle AMNP tel que N appartienne à [BC] et P à [AB].*
Étudiez l'aire du rectangle AMNP en fonction de la position de M.

Commençons par ouvrir une fenêtre de géométrie en tapant simultanément sur Alt et g puis définissons les points A et B dans un repère judicieusement choisi. On utilise `point(x,y)` (ou même `point(z)` pour les terminales) :

```
A:=point(0,0)    // plaçons A
B:=point(0,-4)   // plaçons B tel que AB=4 en laissant le 1er quadrant
    libre pour y tracer la courbe à la fin de la séance
```

Définissons ensuite le point C tel que le triangle ABC soit direct, rectangle en A et que l'on ait AC = $\frac{3}{4}$AB à l'aide de la commande `triangle_rectangle` :

```
triangle_rectangle(A,B,3/4,C)
```

Créons maintenant un réel quelconque de [0; 3] que l'on puisse faire varier à la souris à l'aide de la commande `assume(t=[valeur de départ,mini,maxi])` :

```
assume(t=[1,0,3])
```

Définissons maintenant le point M d'abscisse variable t :

```
M:=point(t,0)
```

Pour définir N, commençons par définir la perpendiculaire en M à la droite (AC). La syntaxe est tout à fait naturelle grâce à `perpendiculaire(Point,Droite)`. C'est bien ce que l'élève doit tracer sur sa feuille de papier pour tracer le rectangle :

```
d:=perpendiculaire(M,droite(A,C))
```

Définissons ensuite N comme l'intersection de d et (BC) grâce à `inter_unique` :

```
N:=inter_unique(d,droite(B,C))
```

Pour obtenir P, nous commençons par définir la parallèle à (AC) passant par N à l'aide de `parallele(Point,Droite)` :

```
D:=parallele(N,droite(A,C));
```

puis l'intersection de D et (AB) :

```
P:=inter_unique(D,droite(A,B))
```

Il ne reste plus qu'à définir le rectangle APNM grâce à la commande `polygone` :

```
R:=polygone(A,P,N,M)
```

et à dessiner le rectangle dépendant du *paramètre* t :

```
couleur(R,jaune+rempli)
```

En faisant varier t à la souris, le rectangle bouge .
Définissons maintenant l'expression de l'aire du rectangle dépendant de t. Appelons-la par exemple `Surf` :

```
Surf:=aire(R)
```

Donnons son expression simplifiée :

```
simplifier(Surf)
```

Réponse du logiciel

$$-\frac{4}{3}t^2+4t$$

Définissons le point X de la courbe représentative de la fonction associée d'abscisse t :

```
X:=point(t,Surf)
```

Traçons enfin la courbe représentative de la fonction associée à l'aide de la commande `graphe` :

```
graphe(Surf,t=0..3,couleur=bleu)
```

Voici une visualisation statique :

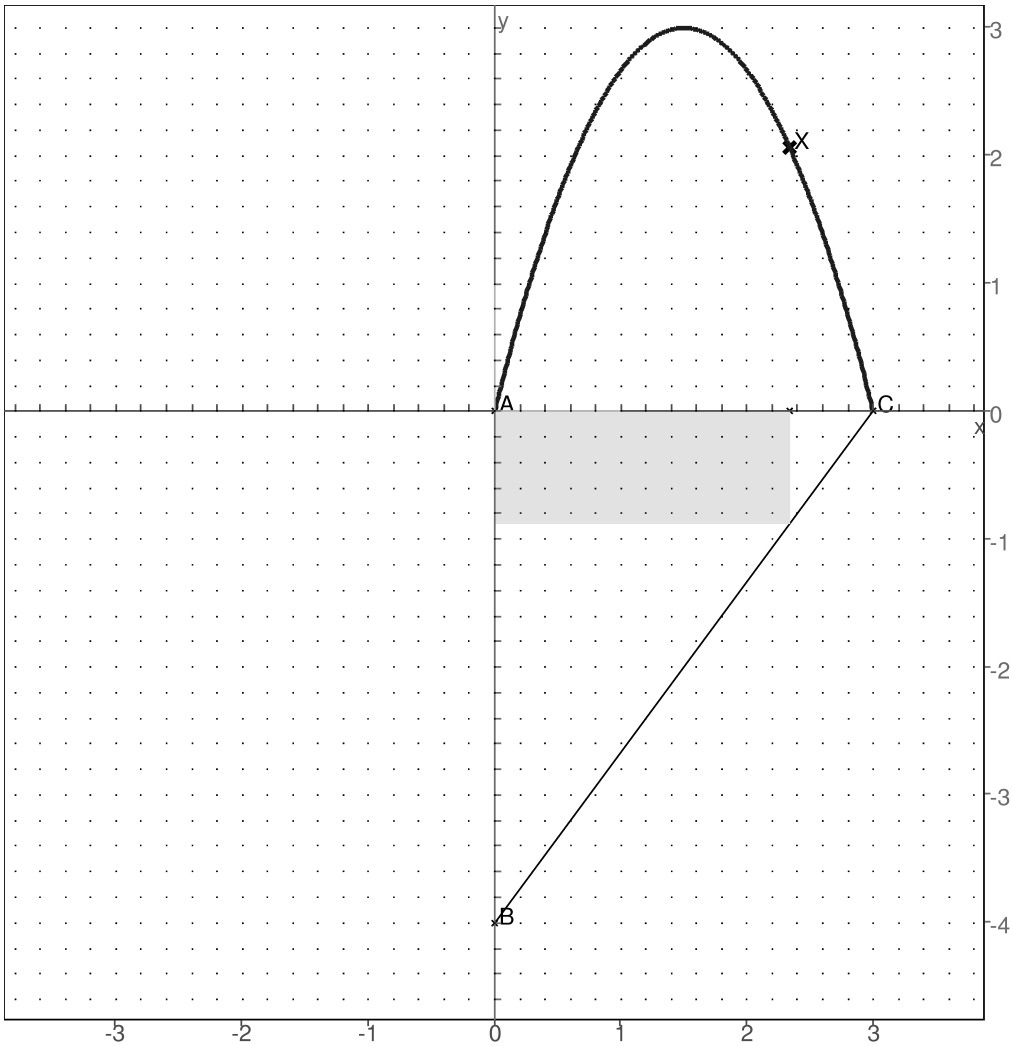
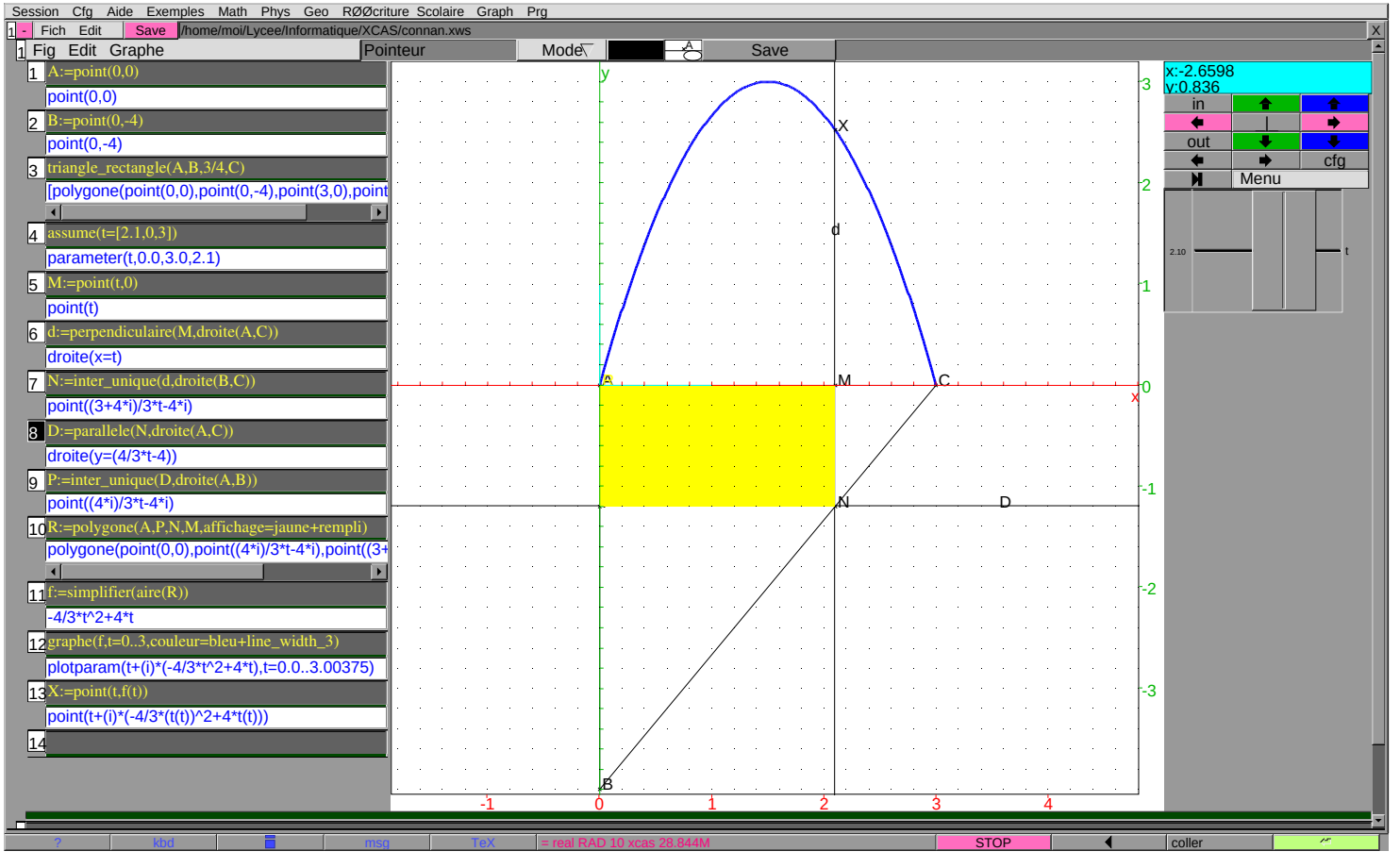


Fig. 1.1 *Le rectangle et la courbe représentative de Surf*


Session Cfg Aide Exemples Math Phys Geo RØØcriture Scolaire Graph Prg
Fich Edit Save /home/moi/Lycee/informatique/XCAS/connan.xws
Fig Edit Graphe
Pointeur
Mode
Save
1 A:=point(0,0)
point(0,0)
2 B:=point(0,-4)
point(0,-4)
3 triangle_rectangle(A,B,3/4,C)
[polygone(point(0,0),point(0,-4),point(3,0),point
4 assume(t=[2.1,0,3])
parameter(t,0.0,3.0,2.1)
5 M:=point(t,0)
point(t)
6 d:=perpendiculaire(M,droite(A,C))
droite(x=t)
7 N:=inter_unique(d,droite(B,C))
point((3+4*i)/3*t-4*i)
8 D:=parallele(N,droite(A,C))
droite(y=(4/3*t-4))
9 P:=inter_unique(D,droite(A,B))
point((4*i)/3*t-4*i)
10 R:=polygone(A,P,N,M,affichage=jaune+rempli)
polygone(point(0,0),point((4*i)/3*t-4*i),point((3+
11 f:=simplifier(aire(R))
-4/3*t^2+4*t
12 graphe(f,t=0..3,couleur=bleu+line_width_3)
plotparam(t+(i)*(-4/3*t^2+4*t),t=0.0..3.00375)
13 X:=point(t,f(t))
point(t+(i)*(-4/3*(t(t))^2+4*t(t)))
14
x:-2.6598
y:0.836
in
out
cfg
Menu
2.10
t
y
x
A
B
C
d
M
N
D
X
-1
0
1
2
3
4
-3
-2
-1
0
1
2
3
? kbd msg TeX = real RAD 10 xcas 28.844M STOP coller

III XCAS et les suites

Il existe principalement deux types de sujets proposés pour l'épreuve expérimentale du Bac S : les problèmes que peuvent traiter les logiciels de géométrie dynamique que nous venons d'étudier et l'étude de suites à l'aide du deuxième pilier de l'Éducation Nationale, le tableur. Nous allons donc nous intéresser à ce deuxième type de sujet. Voici par exemple le sujet 5 de la banque d'exercices de 2007 :

Étude de la suite $u_{n+1} = a \cdot u_n + b$ pour a et b donnés : convergence et valeur de la limite.

On supposera par la suite que $a \neq 1$. En effet, dans le cas contraire, $u_{n+1} = u_n + b$: la suite est donc arithmétique et ne mérite pas une étude poussée...
On construit une procédure donnant u_n en fonction de a, b et le premier terme u_0. On traite le cas $n = 0$ pour initialiser. Ensuite il suffit de rentrer l'expression de u_n en fonction du terme précédent :

```
u(n,a,b,uo):={
si n==0 alors return uo;
sinon a*u(n-1,a,b,uo)+b // C'est-à-dire u(n)=a*u(n-1)+b
}:;
```

Par exemple, pour $a = 0{,}7$, $b = 1$ et $u_0 = 1$, on obtient :

```
u(100,.7,1,1);u(500,.7,1,1) // le 100-ème et le 500-ème terme
```

Réponse du logiciel

```
3.333333333,3.333333333
```

Et pour $a = -1/2$, $b = 1$ et $u_0 = 1$, on obtient :

```
u(100,-0.5,1,1);u(500,-0.5,1,1)
```

Réponse du logiciel

```
0.6666666667,0.6666666667
```

On peut préférer fixer a,b et uo hors de la procédure et définir u de manière plus standard ainsi :

```
u(n):={
si n==0 alors return uo;
sinon a*u(n-1)+b
}:;
```

et alors :

```
a:=0.7
b:=1
uo:=1
u(100);
```

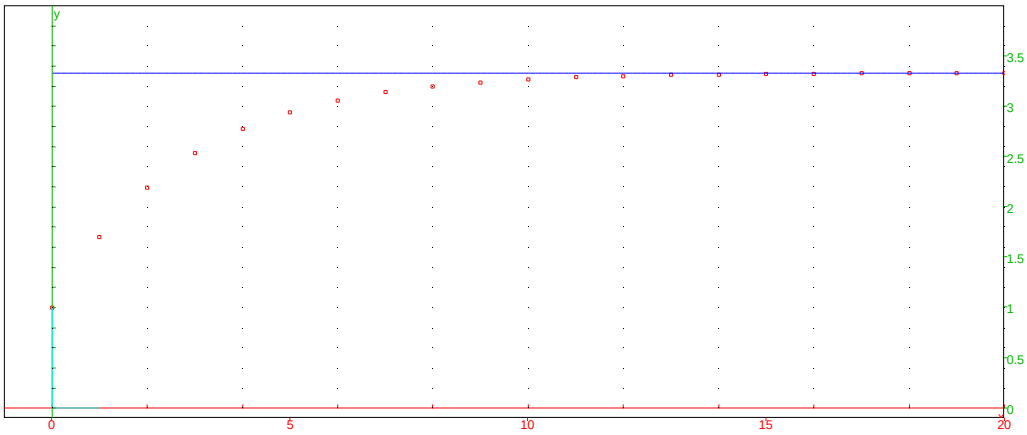
On a l'impression d'avoir de sérieux candidats pour la valeur de la limite. Pour confirmer cette intuition, représentons graphiquement les premiers termes de la suite. On crée une séquence S de coordonnées, vide au départ, et à laquelle on ajoute les points coordonnées (k, u_k) pour k variant de 0 à n :

```
AffSuite(n,a,b,uo):={
local S,k;
S:=NULL;  // S est vide au départ : il n'y a pas encore de points
affichage(point_carre+rouge); // pour l'esthétique...
pour k de 0 jusque n pas 1 faire
   S:=S,point(k,u(k,a,b,uo)); // on rajoute le point de coordonnées $(k,
       u_k)$ à notre liste
fpour
}
```

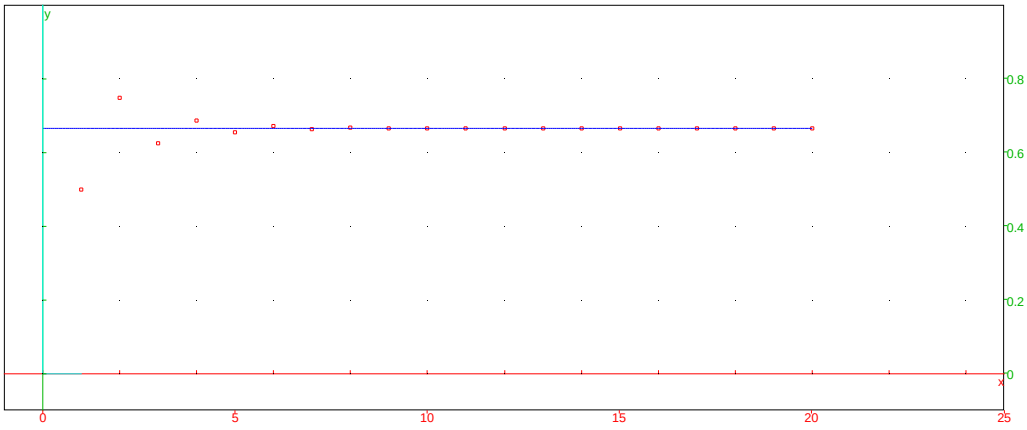
On affiche les valeurs de la suite ainsi que la limite présumée issue de notre calcul :

```
AffSuite(20,-.5,1,1),graphe(0.66666667,x=0..20,couleur=bleu)
```

et on obtient pour $a = 0,7$, $b = 1$ et $u_0 = 1$:



et pour $a = -1/2$, $b = 1$ et $u_0 = 1$:



Notre intuition est confortée.

Complexité d'un algorithme

Cette manière de procéder peut sembler naturelle et proche d'une démarche « papier-crayon », mais n'est pas optimale informatiquement. On peut, pour la culture des élèves, évoquer le problème crucial de la complexité d'un algorithme.
En effet, dans `AffSuite`, on recalcule `u(k,a,b,uo)` à chaque tour de boucle : pour aller jusqu'au rang n, on effectue donc $1+2+3+\cdots+n=\frac{n(n+1)}{2}$ calculs.
On dit que la complexité de l'algorithme est de l'ordre de n^2.
Voici une manière plus « informatique » et efficace de procéder :

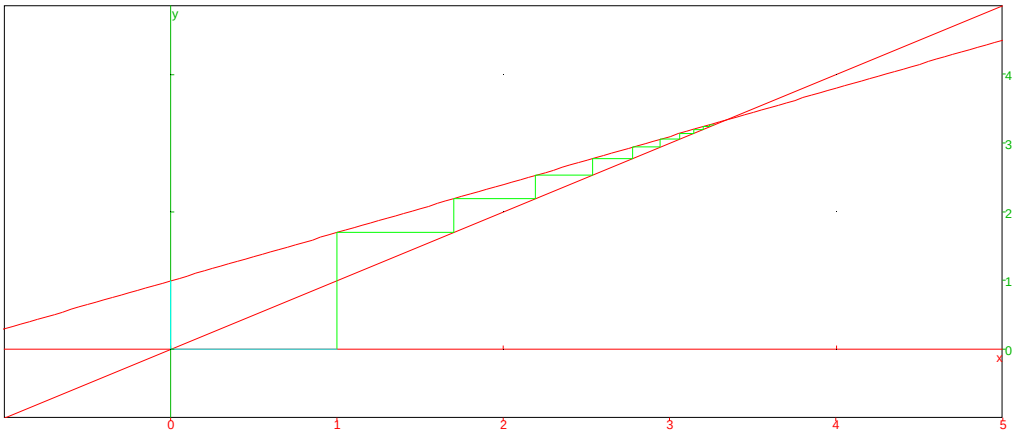
```
AffSuiteBis(n,a,b,u0):={
local S,k,uk;
affichage(point_carre+rouge);
S:=point(0,u0);  // premier terme
k:=u0;
pour k de 1 jusque n pas 1 faire
  uk:=a*uk+b;
  S:=S,point(k,uk); // on rajoute le point de coordonnées (k,u_k)
       à notre liste
fpour
}
```

On effectue n calculs pour avoir `AffSuiteBis(n,a,b,uo)` : la complexité est maintenant de l'ordre de n.
Le problème de la complexité est ici accessoire car dans les deux cas l'ordinateur effectue ces calculs simples très rapidement. Cependant, quand les algorithmes sont plus sophistiqués, cela peut permettre de sauver beaucoup de temps. On peut même évoquer le système RSA qui assure la sécurité des transactions sur internet ou bancaire et dont la sécurité est basée sur la complexité informatique du cassage du code théoriquement possible mais hors du temps humain pour l'instant.

Nous pouvons aussi remarquer que $u_{n+1}=f(u_n)$, avec $f(x)=ax+b$.
Nous pouvons représenter la suite directement avec la fonction **XCAS** `plotseq((f(x),u(0),n)` :

```
plotseq(0.7*x+1,1,10)
```

qui affiche un graphe que les élèves de terminales savent interpréter :



Le candidat pour la limite semble être l'intersection des droites d'équations $y = ax+b$ et $y = x$.
On obtient la solution avec `resoudre(équation,inconnue)` :

```
resoudre((7/10)*x+1=x,x)
```

Réponse du logiciel

$$\frac{10}{3}$$

Il ne reste plus qu'à le démontrer...
Soit p la solution de $ax+b=x$. Elle existe bien car on a supposé $a \neq 1$.
Alors, en effectuant la différence membre à membre de $u_{n+1} = au_n + b$ et $p = ap + b$, on obtient :

$$u_{n+1} - p = a(u_n - p)$$

Donc la suite $(u_n - p)$ est géométrique de raison a et de premier terme $u_0 - p$.
On obtient donc

$$u_n = (u_0 - p)a^n + p$$

Vérifions :

```
expu(n,a,b,uo):={
local p,c;
p:=resoudre(a*x+b=x)[0]; // la solution de l'équation
c:=simplifier(uo-p);  // on simplifie l'écriture de u0-p
return(c*a^n+p)  // on renvoie (uo-p)*a^n+p
}:;
```

Par exemple, toujours avec $a = 0,7$, $b = 1$ et $u_0 = 1$:

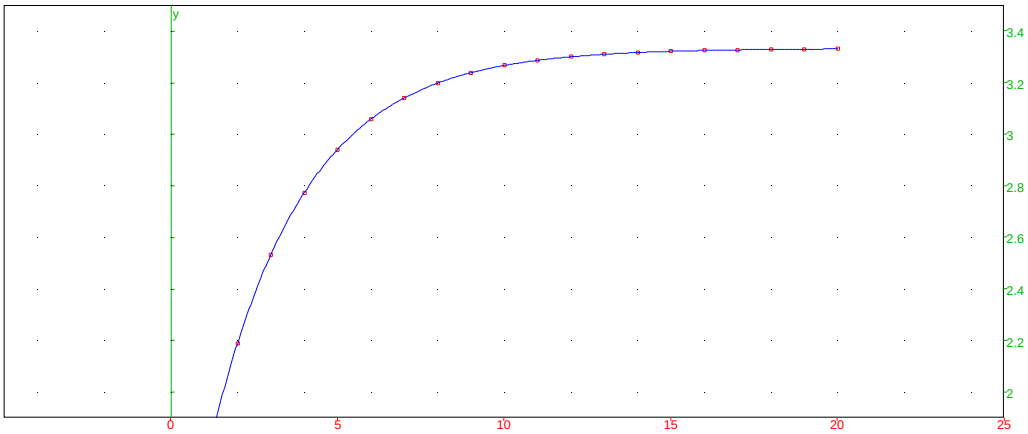
```
expu(n,7/10,1,1)
```

Réponse du logiciel

$$\frac{-7\cdot\left(\frac{7}{10}\right)^{n}}{3}+\frac{10}{3}$$

Comparons maintenant les graphes :

```
AffSuite(20,7/10,1,1),graphe(expu(x,7/10,1,1),x=0..20,couleur=bleu)
```



Et voilà...Nous pouvons même nous payer le luxe de vérifier que expu vérifie la formule de récurrence initiale *dans le cas général,* c'est-à-dire pour a, b, u_0 quelconques !
Ce confort n'est bien sûr possible qu'avec un logiciel de calcul formel...

```
expu(n+1,a,b,uo)-(a*expu(n,a,b,uo)+b)
```

Réponse du logiciel

$$(a\cdot uo+b-uo)\cdot\frac{1}{a-1}\cdot(a)^{n+1}-\left(\frac{b}{a-1}\right)-\left(a\cdot((a\cdot uo+b-uo)\cdot\frac{1}{a-1}\cdot(a)^{n}-\left(\frac{b}{a-1}\right))\right)-b$$

Oups...

```
simplifier(ans()) // simplifie le résultat précédent
```

Réponse du logiciel

0

C'est mieux ! Nous pouvons aller jusqu'à calculer la limite de la suite dans le cas général :

```
limite(expu(n,a,b,uo),n,+infinity)
```

Mais **XCAS** ne sait que répondre...il faut bien sûr distinguer les cas :

```
assume(a>0) and assume(a<1); // 0<a<1
limite(expu(n,a,b,uo),n,+infinity)
```

Réponse du logiciel

$$\frac{-b}{a-1}$$

et dans le cas $a > 1$:

```
assume(a>1);
limite(expu(n,a,b,uo),n,+infinity)
```

Réponse du logiciel

```
infinity
```

Ainsi, **XCAS** nous permet non seulement d'*observer* le résultat mais surtout de le **démontrer** grâce à ses facultés de calcul formel. Nous allons illustrer plus précisément ce point dans le paragraphe suivant.

IV Ce que ni le tableur, ni un logiciel de géométrie ne peuvent faire

a. Preuve d'un théorème

Les coniques ne sont plus au programme de terminale mais sont encore sources d'activités. Nous allons ici nous occuper d'un théorème concernant le tracé de la tangente à une conique et le *prouver* à l'aide de **XCAS** ce qui n'est possible qu'avec un logiciel de calcul formel.
Le théorème initial est le suivant :

Construction de la tangente à une conique

Soit Γ une conique de foyer F et de directrice associée D. La tangente à Γ en tout point M qui n'appartient pas à l'axe focal coupe D en un point T tel que l'on ait $\overrightarrow{FM} \cdot \overrightarrow{FT} = 0$

Nous allons adapter l'énoncé au cas d'une demi-ellipse définie à l'aide d'une fonction :

Soient a et b deux réels strictement positifs tels que $a > b$ et f la fonction définie sur $[-a; a]$ *par*

$$f(x) = b\sqrt{1 - \frac{x^2}{a^2}}$$

Notons Γ *la courbe représentative de* f *dans un repère orthonormé* $(O; \vec{u}, \vec{v})$.
Soit F le point de coordonnées $(\sqrt{a^2 - b^2}; 0)$ *et D la droite d'équation* $x = \frac{a^2}{\sqrt{a^2-b^2}}$.
Montrez que la tangente à Γ *en tout point M d'ordonnée non nulle coupe D en un point T tel que l'on ait* $\overrightarrow{FM} \cdot \overrightarrow{FT} = 0$

Nous pouvons bien sûr commencer par illustrer des cas particuliers en prenant un point mobile sur Γ et en observant que la propriété reste vérifiée.
Nous allons sauter cette étape, notre propos étant de montrer comment un logiciel de calcul formel permet non seulement d'illustrer mais aussi de *prouver* un théorème général.
Nous commençons par définir la fonction f :

```
f:=x->b*sqrt(1-(x^2/a^2));;
```

Nous allons ensuite considérer un point quelconque de Γ d'absisse t, avec $t \in]-a; a[$:

```
assume(a>0):;
assume(t>-a) and assume(t<a):;
M:=point(t,f(t)):;
```

Pour définir la tangente d à Γ en M, nous allons introduire la fonction dérivée de f que nous noterons fp. Nous utiliserons les commandes **XCAS** `abscisse` et `ordonnee` qui renvoient l'abscisse et l'ordonnée d'un point bien sûr, et la commande `droite` qui permet de définir une droite de multiples façons (par une équation, deux points, comme intersection de deux plans, une représentation paramétrique,...). Nous la définirons ici par l'équation classique $y = f'(x_M) \times (x - x_M) + f(x_M)$:

```
fp:=fonction_derivee(f):;
d:=droite(y=fp(abscisse(M))*(x-abscisse(M))+ordonnee(M)):;
```

Mode « muet »

Vous noterez que chaque ligne de commande se termine par `:;` ce qui indique à **XCAS** qu'il doit évaluer cette ligne mais sans renvoyer de résultat. En effet, aucun tracé ne peut être effectué pour l'instant car aucune valeur numérique n'est affectée aux paramètres a, b et t.

Il reste à définir le point F et la droite D :

```
F:=point(sqrt(a^2-b^2),0):;
D:=droite(x=a^2/sqrt(a^2-b^2)):;
```

Le point T est l'intersection des droites d et D.
On utilise la commande `inter_unique(ensemble1,ensemble2)` :

```
T:=inter_unique(d,D):;
```

Il ne reste plus qu'à calculer le produit scalaire $\overrightarrow{FM}\cdot\overrightarrow{FT}$ à l'aide de `scalar_product`.

```
scalar_product(vecteur(M,F),vecteur(T,F))
```

Calcul vectoriel

Un vecteur pour **XCAS** est une liste de coordonnées entre deux crochets tout comme l'est un point. L'opération M – F est donc licite pour **XCAS**. Pour ne pas semer le trouble dans les esprits des élèves du lycée, on peut remplacer M – F par `coordonnees(M)-coordonnees(F)`, ou bien, pour éviter de calculer avec des systèmes de coordonnées, on peut préférer travailler sur les affixes en rentrant

```
scalar_product(affixe(M)-affixe(F),affixe(T)-affixe(F))
```

On peut également demander à Bernard PARISSE de créer une commande `vecteur(M1,M2)` synonyme de M2-M1 pour l'ordinateur mais moins choquante pour les élèves. C'est ce que j'ai fait...et obtenu une heure plus tard ! Encore un point fort de **XCAS**...

Ainsi :

```
scalar_product(vecteur(F,M),vecteur(F,T))
```

Réponse du logiciel

$$\left(t-\sqrt{a^2-b^2}\right)\left(\frac{(-a^4+a^2t^2)\sqrt{a^2-b^2}}{-a^4+a^2t^2+(a^2-t^2)b^2}-\sqrt{a^2-(b^2)}\right)+$$
$$\frac{b\sqrt{1-\frac{t^2}{a^2}}\left(a\sqrt{a^2-t^2}\cdot b^3-a^3\sqrt{a^2-t^2}\cdot b+at\sqrt{a^2-t^2}\cdot b\cdot\sqrt{a^2-b^2}\right)}{-a^4+a^2t^2+\left(a^2-t^2\right)\cdot b^2}$$

Ouh la la...Voici un calcul qui a de quoi déprimer n'importe quel élève de lycée. Heureusement, nous pouvons faire faire « le sale boulot » à **XCAS** grâce à la commande `simplifier`

et se concentrer sur le problème géométrique sans être gêné par les difficultés de calcul algébrique.
Cette fois-ci :

```
simplifier(scalar_product(vecteur(M,F),vecteur(T,F))
```

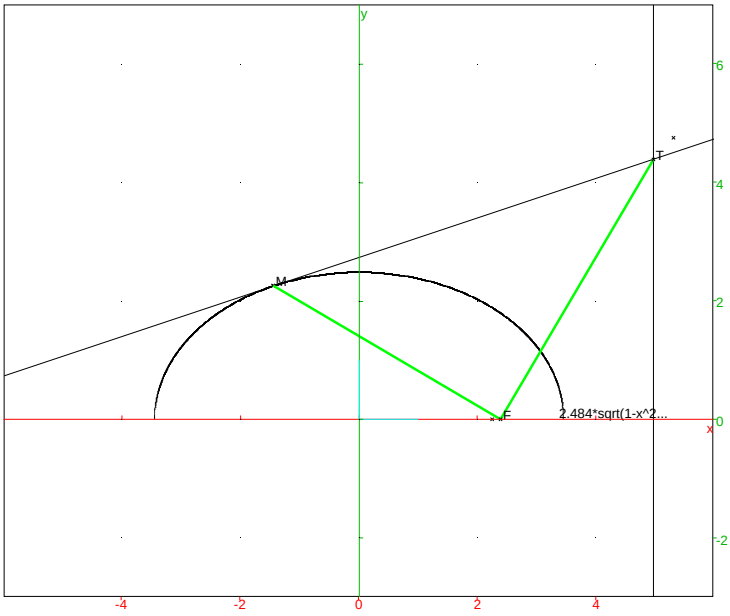
renvoie bien 0, ce qui prouve notre théorème car les calculs ont été effectués dans le cas général.

b. Illustration graphique

Pour observer ce phénomène en géométrie dynamique, il suffit d'ouvrir une fenêtre de géométrie en tapant Alt - g puis en définissant a, b et t comme étant des paramètres que nous ferons varier à la souris :

```
a:=element(0..5)
b:=element(0..a)
t:=element(-a..a)
```

le reste des instructions restant inchangé. On obtient :



c. Bilan de cette activité

Nous voici donc en mesure de tracer n'importe quelle tangente à une ellipse à l'aide d'une équerre puisqu'il suffit de tracer la perpendiculaire à (MF) en M puis l'intersection T de cette droite avec une directrice : la tangente est la droite (TM).
Il s'agit d'un problème géométrique. L'ordinateur nous permet de prouver ce théorème en se dispensant de passer trop de temps dans des calculs algébriques compliqués : nous restons

dynamiques mathématiquement sans être perturbés par l'aquisition d'une virtuosité algébrique ou informatique. Les instructions données correspondent à la démarche mathématique. Nous avons fait des mathématiques et très peu d'informatique en somme. Nous nous sommes concentrés sur le raisonnement et l'ordinateur nous a permis de ne pas nous perdre dans les calculs.
Le travail avec un calculateur formel est donc la continuation logique du travail avec une calculatrice usuelle, mais en offrant beaucoup plus de posibilités bien sûr.

V Un soupçon d'algorithmique

L'initiation aux rudiments d'algorithmique peut commencer très tôt puisque je la pratique en primaire. Il est donc raisonnable de faire s'exercer des élèves de Terminale S sur ces notions primordiales pour des scientifiques.
Nous présentons ici le texte d'un TP donné à des élèves de Terminale S après une première séance de découverte de la fonction exponentielle et alors qu'ils avaient manipulé les outils de calcul de **XCAS** un mois plus tôt.

a. Approximation affine

On s'intéresse aux fonctions f dérivables sur $\mathbb{R}$ satisfaisant la condition

$$f'(x) = k \cdot f(x) \qquad (1)$$

pour tout $x \in \mathbb{R}$, avec k un réel arbitrairement fixé.
On va tenter, dans cette section, d'obtenir une approximation de l'allure de la courbe représentative d'une solution de (1).
On va pour cela considérer que pour h « suffisamment petit » et pour tout réel a, $f'(a) \simeq \frac{f(a+h)-f(a)}{h}$.
Montrez alors que $f(a+h) \simeq (1+kh)f(a)$.

b. Subdivision

Pour obtenir ce tracé, nous allons choisir un réel k, un segment $[a; b]$, nous allons fixer l'image de a par f pour avoir une solution unique de (1) dans un premier temps.
Nous allons subdiviser le segment $[a; b]$ en N segments de même longueur : quelle sera la longueur de chaque petit segment ?

c. Tracé d'une ligne brisée

Ouvrons une fenêtre de géométrie en tapant simultanément sur Alt et g.
On définit un point dont on connaît les coordonnées avec la commande `point` :

```
A:=point(1,2);
B:=point(3,-1);
C:=point(4,5);
```

On relie ces points par des segments de droite grâce à la commande `polygone_ouvert(liste de points)` :

```
polygone_ouvert(A,B,C)
```

d. Tracé en boucle

Oublions pour un temps notre problème et voyons comment nous pourrions tracer « point par point » sur $[-3\,;3]$ la courbe représentative d'une fonction g vérifiant :

$$\frac{g(b)-g(a)}{b-a}=2$$

pour tous réels distincts a et b de $[-3\,;3]$ et $f(-3)=-1$.
On peut par exemple subdiviser le segment $[-3\,;3]$ en segments de longueurs 0,25 et réfléchir au moyen d'obtenir les images par f de chacune des extrémités des segments de la subdivision.
Il suffit de penser que $\frac{g(x+0{,}25)-g(x)}{x+0{,}25-x}=2$, c'est-à-dire $g(x+0{,}25)=\ldots\ldots+g(x)$.
On peut donc calculer $g(x+0{,}25)$ si l'on connaît $g(x)$.
On va donc partir du point de coordonnées $(-3\,;-1)$ et obtenir de proche en proche les coordonnées de plusieurs points de la courbes en faisant des petits sauts de 0,25 et en s'arrêtant à 3 :

```
S:=NULL; // on crée une suite de points  vide au départ
X:=-3;  // au départ X vaut -3
Y:=-1;  // au départ Y vaut -1
tantque X<=3  faire  // tant que s'écrit tantque en XCAS
   S:=S,point(X,Y);  // on  rajoute le point de coordonnées (X,Y) à notre
        liste
   X:=X+0.25; // on avance de 0,25 à chaque tour de boucle
   Y:=0.5+Y; // on sait que g(X+0,25)=0,5+g(X)
ftantque:;  // f comme fin de la boucle
polygone_ouvert(S); // on relie les points de la liste S à la règle
```

et on découvre sans surprise qu'il s'agit d'un segment de droite.

e. Procédure

Cela fonctionne pour ce cas particulier mais ça devrait aussi « marcher » si l'on change les valeurs du pas, du coefficient directeur, des bornes de l'intervalle de définition, de l'image de la borne inférieure de cet intervalle.
On a donc envie de créer un fonction informatique, une *procédure*, qui reprend cette méthode mais dans le cas général, avec des coefficients quelconques qu'on donnera à l'ordinateur. On crée ainsi une sorte de fonction de plusieurs variables.
Notons donc h le pas, m le coefficient directeur, $[a\,;b]$ l'intervalle de définition et yo l'image de a.
Donnons également un nom à notre procédure, par exemple `TraceAffine`.
On va ouvrir une fenêtre de programmation en tapant Alt p :

```
TraceAffine(h,m,a,b,yo):={ // on précise quels sont les variables de
    notre procédure
S:=NULL; // on crée toujours une suite de points  vide au départ
X:=a;  // au départ X vaut a cette fois
Y:=yo;  // au départ Y vaut yo
tantque X<=b  faire  // on s'arrête à X=b
   S:=S,point(X,Y);  // on  rajoute le point de coordonnées (X,Y) à notre
        liste
   X:=X+h; // on avance de h à chaque tour de boucle
   Y:=h*m+Y; // on sait que g(X+h)=h*m+g(X)
ftantque:;  // f comme fin de la boucle
polygone_ouvert(S); // on relie les points de la liste S à la règle
}:;  // on termine la procédure en fermant l'accolade
```

Pour valider, on clique sur OK .
Pour exécuter cette procédure, on place le curseur sur une ligne de commande.
Par exemple, pour retrouver le cas précédent, on rentre :

```
TraceAffine(0.25,2,-3,3,-1)
```

f. À vous de jouer

Vous êtes maintenant armés pour adapter ce que nous venons de voir à la méthode d'Euler sachant que cette fois $f(X+h) = (1+kh)f(X)$ comme nous l'avons vu au premier paragraphe.
Construisez donc une procédure `Euler(h,k,a,b,yo)`...

g. Estimation de l'erreur

Nous avons vu en cours que la solution de l'équation différentielle $y' = y$ avec $y(0) = 1$ s'appelle la fonction exponentielle qui se note exp pour **XCAS**.
Nous avons utilisé une approximation pour obtenir son tracé par la méthode d'Euler. Il est bien sûr primordial d'avoir un ordre de grandeur de l'erreur commise.
Nous avons écrit que $\exp(a+h) \simeq (1+h) \times \exp(a)$.
Créons alors une fonction `Exp(h,x)` qui donne une approximation de e^x pour un h donné.
Il suffit d'utiliser la procédure précédente en ôtant la partie concernant le tracé et en prenant $a = 0$, $k = 1$, $b = x$:

```
Exp(h,x):={
X:=0;
Y:=1;
tantque X<=x  faire
   X:=X+h;
   Y:=(1+h)*Y;
ftantque:;
}:;
```

Par exemple, on peut comparer `Exp(h,1)` et exp(1) pour des valeurs successives de h entre 0,1 et 0,0001, puis en prenant d'autres valeurs de x.

Étudiez le rapport $\frac{Exp(0.01,x)-\exp(x)}{\exp(x)}$ pour différentes valeurs de x.
Que pensez-vous de l'approximation ?

On peut visualiser l'erreur commise en créant une fonction `erreur` :

```
erreur(x):=(1-Exp(0.01,x)/evalf(exp(x)))*100
```

Puis en créant la suite des points de coordonnées `(x,erreur(x))` pour x variant de 0 à 100 avec un pas de 1.
La commande `seq` comme séquence[a] permet de créer cette suite de valeurs :

```
seq(point(x,erreur(x)),x=0..100)
```

Des commentaires ?

VI Moralité...

On reproche souvent à l'utilisation de l'informatique au lycée d'exister pour elle-même et de semer la confusion dans les esprits des élèves en les incitant à confondre observation et preuve.
On peut également remarquer que sur les 97 propositions de sujets pour l'épreuve expérimentale du Bac S 2008, les domaines traités couvrent une très petite partie du programme du Bac : essentiellement les suites, la géométrie, voire un peu de probabilités.
Ces reproches sont souvent justifiés compte tenu des limitations des logiciels promus officiellement : le tableur et les logiciels de géométrie dynamique, par essence, ne permettent que d'observer des données numériques ou géométriques et ne peuvent élaborer une preuve générale, ce que peut faire un logiciel de calcul formel. Ils ne peuvent non plus aborder le calcul différentiel, intégrale, algébrique que de manière numérique ou expérimentale et donc passent à côté de la plus grande partie du programme de terminale S.
D'un autre côté, on reproche aux logiciels de calcul formel d'être trop compliqués à utiliser au lycée. Nous venons de voir que c'est injustifié et qu'au contraire travailler en ligne de commande et ainsi dialoguer naturellement avec l'ordinateur peut en fait être beaucoup plus simple que de naviguer dans des menus, sous-menus et autres sous-sous-menus d'icônes à cliquer.
Enfin, de manière plus précise, le logiciel **XCAS** semble être particulièrement adapté au lycée car il permet à la fois de faire du calcul formel, de la géométrie dynamique, du tableur et est largement francisé.
Il resterait à parler de l'initiation à la programmation qui devrait faire partie du bagage minimum fourni aux élèves des sections scientifiques (ce qui est le cas dans de nombreux pays européens) et qui peut être également abordée grâce à **XCAS** dès la seconde comme j'en ai fait l'expérience depuis plusieurs années dans mes classes...et même dès l'école primaire grâce à la *Tortue* de **XCAS**, mais ceci est un autre problème qui sera abordé dans un prochain article...

[a] C'est-à-dire suite

VII Annexe : quelques conseils avant de commencer

Si vous êtes un peu perdu pour commencer, une aide multiple est variée est accessible un peu partout :

- vous pouvez commencer par consulter l'aide fournie avec votre installation en allant dans le menu `Aide`. Par exemple, pour les soucis d'interface, vous allez dans `Aide->interface`. Tout est en français et au format html;
- vous pouvez posez une question sur le forum dédié à **XCAS** : `Aide->Internet->Forum`;
- vous pouvez me poser une question directement par courrier électronique...

Quelques conseils néanmoins pour éviter les pièges habituels (il ne s'agit que de quelques petites trucs usuels. N'hésitez pas à poser des questions si des problèmes subsistent) :

- lors de l'installation, on vous demande quel niveau vous souhaitez : choisissez `Universite`. Si vous avez déjà installé **XCAS**, allez dans `Cfg->configuration générale->niveau->université` puis cliquez sur `sauver`. Ainsi, à chaque démarrage, vous aurez une fenêtre normale. Vous pouvez également configurer la taille de la police, les couleurs;
- il est souvent utile de passer à la ligne sans valider l'instruction : on appuie simultanément sur Shift ⇑ et Enter ;
- vérifiez si vous n'avez pas oublié une lettre, mis un accent, une mauvaise ponctuation;
- si l'on veut supprimer une ligne, on clique sur son numéro qui apparaît alors sur fond noir puis on va sur `Edit->supprimer les niveaux selectionnes`;
- lorsque vous voulez rentrer une procédure, il vaut mieux ouvrir une fenêtre de programmation en tapant simultanément sur Alt et p . Ce n'est pas obligatoire, mais cela permet de faire tester son code par **XCAS** sans mettre en péril le reste de la session. On clique sur OK pour valider son programme. De plus, vous profitez ainsi de la coloration syntaxique de l'éditeur qui met en valeur les mots-clés : vous trouverez à la page suivante un aperçu de ce que cela donne pour un des programmes de l'étude de la méthode d'Euler vue à la section précédente. On remarquera sur ces captures d'écran que l'on peut commenter son code, c'est-à-dire donner quelques explications qui ne seront pas évaluées par **XCAS** mais utiles au lecteur éventuel : les commentaires apparaissent en vert à la suite de `//`. On peut aussi commenter dans une céllule à part en tapant simultanément sur Alt et c :

Session Cfg Aide Exemples Math Phys Geo RØØcriture Scolaire Graph Prg

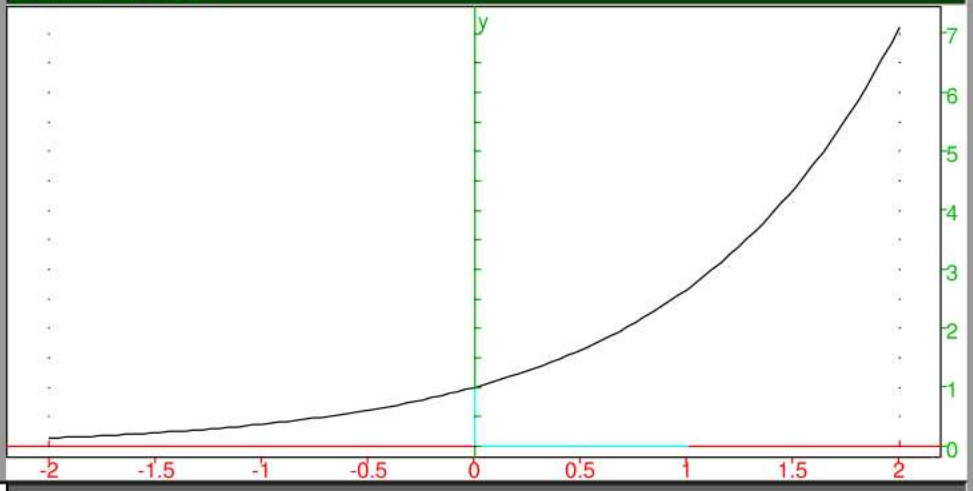
1 - Fich Edit Save /home/moi/Lycee/Informatique/XCAS/methodeEuler.xws X

1 Prog Edit Ajouter nxt OK Save

```
EulerExpo(alpha,N,a,b,yo):= {
S:=NULL; // la suite des points est vide au depart
X:=a;  // au depart X vaut a
Y:=yo;  // au depart Y vaut yo
h:=(b-a)/N  // nous divisons [a,b] en N seqments de lonqueur h
pour k de 0 jusque N pas 1 faire
X:=a+k*h; // on avance de h a chaque tour de boucle
Y:=(1+alpha*h)*Y;  // f(X+h)=(1+alpha*h)f(X)
P:=point(X,Y);  // on cree le nouveau point
S:=S,P;  // on le rajoute a notre liste
fpour  // fin de la boucle
polygone_ouvert(S); // on relie les points de la liste S  la reqle
}:;
```

2 EulerExpo(1,100,-2,2,exp(-2))

Evaluation time: 2.24



3

? kbd msg TeX = real RAD 10 xcas 18.059M STOP coller


Session Cfg Aide Exemples Math Phys Geo RØØcriture Scolaire Graph Prg
Fich Edit Save /home/moi/Lycee/Informatique/XCAS/EpreuveTS/sujet5.xws
On construit une procØdure donnant u(n) en fonction de a, b et le premier terme u(0). On traite le
l'expression de u(n) en fonction du terme prØcØdent.
Prog Edit Ajouter nxt OK Save
u(n,a,b,uo):= {
if(n==0) return uo;
a*u(n-1,a,b,uo)+b
};;
// Parsing u
// Warning: u declared as global variable(s) compiling u
u: recursive definition
Done
Menu
Par exemple, pour a=0,7, b=1 et u(0)=1, on obtient
u(100,1,1,1);u(500,1,1,1)
(101, 501)
Menu
Pour a=-1/2, b=1 et u(0)=1, on obtient
u(100,-.5,1,1);u(500,-.5,1,1)
(0.6666666667, 0.6666666667)
Menu
On a l'impression d'avoir de sØrieux candidats pour la valeur de la limite. Pour confirmer cette intu
termes de la suite. On crØe une sØquence S de coordonnØes, vide au dØpart, et laquelle on aj
de 0 n.
Prog Edit Ajouter nxt OK Save
AffSuite(n,a,b,uo):= {
local S,k;
S:=NULL;
affichage(point_carre+rouge);
for(k:=0;k<=n;k++) {
S:=S,point(k,u(k,a,b,uo));
}
// Parsing AffSuite
// Warning: u declared as global variable(s) compiling AffSuite
Done
Menu
On affiche les valeurs de la suite ainsi que la limite prØsumØe issue de notre calcul
AffSuite(20,1,1,1),plot(0.66666667,x=0..20,couleur=blue)
y
? kbd msg TeX = real RAD 10 xcas 18.75M STOP coller