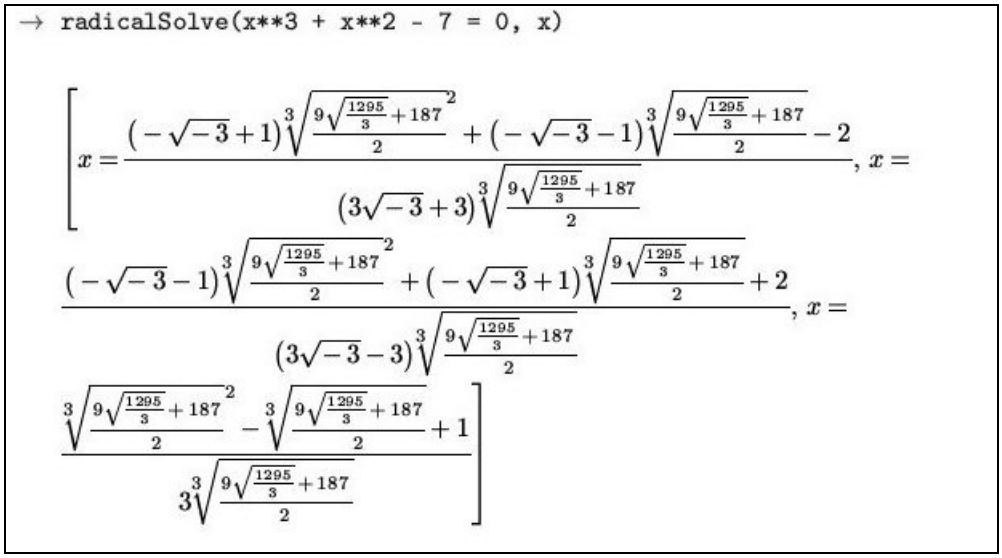
CARPENT Quentin
CONIL Christophe

AC20

Utilisation de logiciels libres pour la réalisation de TP MT26



Étude de la faisabilité

Responsable d'UV: M. Petitjean

Responsable de sujet: M. Bachard

Semestre de printemps 2004

Remerciements

Nous souhaitons tout d'abord remercier M. Petitjean et M. Bachard, pour nous avoir permis de réaliser cette UV et d'étudier un sujet nous intéressant particulièrement. Nous remercions aussi M. Lannuzel, pour sa disponibilité et pour les documents MT26 fournis. Bien que cela soit difficile, nous voudrions remercier également tous les anonymes qui ont apporté des réponses à nos différentes interrogations, sur les forums d'Axiom et de GNUplot, au cours des derniers mois.

Introduction

Le recours au calcul formel est une notion primordiale au cours des études d'un élève ingénieur. Actuellement, au sein de l'UTBM, Maple, logiciel « star » de calcul formel tiens cette place prédominante: Facilité d'emploi, aide complète, interface détaillée et intuitive, rien ne lui fait défaut: Et pourtant, MAPLE est un logiciel propriétaire, donc fermé. Il est payant, et très onéreux. (La version actuelle de Maple 9.5 coûte environ 995 € l'unité, pour une utilisation dans le domaine éducatif) Il est ainsi quasi-impossible pour un étudiant de se procurer une version « légale » du logiciel: On trouve bon nombre de copies en libre circulation.

MT26 est une UV mathématique ou l'on se sert de Maple pour résoudre une série de problèmes graphiques, et arithmétiques: Ces problèmes sont diversifiés (limites, sommes, matrices, études de fonctions, programmation..) et peuvent tous être résolus au cours du semestre avec une intervention minimale du professeur.

L'intitulé de notre AC20, « Utilisation de logiciels libres pour la réalisation de TPs MT26 » va nous conduire à chercher des solutions alternatives à Maple, à les tester, à définir leur facilité d'utilisation, et enfin à juger de leur capacité à devenir des logiciels d'enseignement pédagogique. Cette démarche pourra par la suite de permettre aux étudiants de découvrir la capacité qu'ont les logiciels libres à se rapprocher étonnamment des logiciels propriétaires, tout en profitant des avantages liés à ceux-ci.

Table des matières

Détermination des logiciels utilisés

Les contraintes imposées lors de la définition du sujet (portabilité, gratuité) nous ont amené à choisir 2 logiciels différents pour l'étude de la problématique: l'un étant spécifique au graphisme, et l'autre au calcul formel utilisé dans Maple.

Notre premier choix s'est ainsi porté sur GNUplot, logiciel de tracé très utilisé dans le monde du libre, tandis que la résolution des exercices de calcul et de programmation se fera avec Axiom (Axiom Computer Algebra System). Ce dernier est en effet un des logiciels libres de calcul formel les plus complets à l'heure actuelle.

Ces deux logiciels répondent aux contraintes imposées : Ceux-ci sont libres, et compilable sur de multiple systèmes d'exploitation, tels que Linux, MacOs, Solaris, Windows. Ils sont tous les deux au moins en licence GPL. Les versions binaires sont disponibles pour chacun des systèmes, excepté en ce qui concerne Windows, la compilation pouvant se révéler très longue et fastidieuse.

Nous utiliserons pour des raisons pratiques un PC, avec un système d'exploitation Linux Debian. L'interface graphique utilisée, KDE, ne posera aucun problèmes. Ce système nous permet d'installer, de configurer, et d'utiliser les 2 logiciels avec une grande facilité.

Présentation générale des logiciels

Axiom Computer Algebra System

Axiom est un logiciel utile pour la recherche et le développement d'algorithmes mathématiques. Il définit une hiérarchie de types mathématiquement correcte, fortement typée. Il a un langage de programmation et un compilateur intégrés, ce qui lui confère rapidité d'exécution et souplesse d'utilisation.

Axiom est développé depuis 1973 et fut autrefois vendu comme un produit commercial. Il est aujourd'hui publié comme logiciel libre.

Des efforts sont en cours pour étendre ce logiciel en :

- développant une meilleure interface utilisateur
- le rendant utilisable comme outil d'enseignement
- développant un protocole de serveur algébrique
- intégrant des fonctions mathématiques supplémentaires
- reconstruisant l'algèbre dans un style de programmation littéraire
- intégrant la programmation logique
- développant un journal d'Axiom avec comité de lecture.

GNUplot

GNUplot permet quand à lui de créer des graphes qui peuvent être intégrés à des documents.

Le logiciel reconnaît la plupart des fonctions mathématiques de base (trigonométrique, Bessel, etc.) et permet de tracer des courbes dans bon nombre de repères (logarithmiques, polaires, paramétriques, 3D). Il est également possible de créer des graphes à partir de fichiers de données. Cette notion est l'aspect mis en avant: En effet, GNUplot n'est absolument pas un logiciel de calcul numérique: Les études de fonction sont réduites à leur minimum.

Il s'agit d'un logiciel très léger (3 mo environ) et utilisable sur de multiples systèmes d'exploitation. Des tutoriels, fichiers d'aides, et version binaires sont disponibles pour tous systèmes.

Informations sur les logiciels

Axiom

Axiom est basé sur une licence BSD modifiée. Elle est courte (environ 1 page), et se révèle donc facile à lire. Il n'y a pas de clauses difficiles à comprendre comme dans d'autres licences. Cette licence accorde :

→ Le droit d'exécuter le programme quel que soit les intentions ou le but

→ Le droit d'étudier le code source du programme

→ Le droit de modifier le code source

→ Le droit de publier les modifications

→ Le droit d'intégrer une partie du code dans un logiciel non-libre, et ce sans contraintes. Ce dernier aspect, même s'il favorise l'interopérabilité des softwares, peut dans une certaine mesure nuire au libre.

La version actuelle est la version 4. C'est une version bêta, suffisamment stable pour en permettre une utilisation régulière. Quelques bugs subsistent toutefois.

Le développement d'Axiom est actuellement assuré par la collaboration de d'une centaine de programmeurs, essentiellement dirigés par une équipe de 7 personnes.

Tim Daly
David Mentré
Camm Maguire
Juergen Weiss
Bill Page
Dylan Thurston

La liste de tous les programmeurs est visible grâce à la commande)credits, dans axiom.

GNUplot

Le développement de GNUplot est à l'heure actuelle très ralenti : En effet, la version disponible (Version 4.0) semble satisfaire les développeurs ainsi que la majorité des utilisateurs. Les principaux développeurs à l'heure actuelle sont :

Alexander Mai
Alex C. Woo
Hans-Bernhard Broeker
Clark Gaylord
Johannes Zellner
Lars Hecking
Petr Mikulik
Per Persson
Ethan Merritt
James R. Van Zandt

La licence de GNUplot est une licence GPL. Cette licence accorde 4 libertés principales:

→ Le droit d'exécuter le programme quel que soit les intentions ou le but

→ Le droit d'étudier le code source du programme

→ Le droit de modifier le code source

→ Le droit de publier les modifications

Contrairement à la licence BSD d'Axiom, elle implique que toute utilisation du code dans une autre application force cette autre application à être elle aussi compatible avec la licence GPL.

Ergonomie: Prise en main

Axiom

Mode Console

La prise en main graphique d'Axiom peut s'avérer très déroutante au début pour un utilisateur habitué à Windows, et ce pour différentes raisons:

- Pas de menus apparent
- Peu attractif graphiquement (car en mode console)
- Pas d'interface intégrée au logiciel.
- Les chargements des librairies sont affichés entre le calcul et le résultat, ce qui fait que les résultats apparaissent parfois plusieurs pages après le calcul !

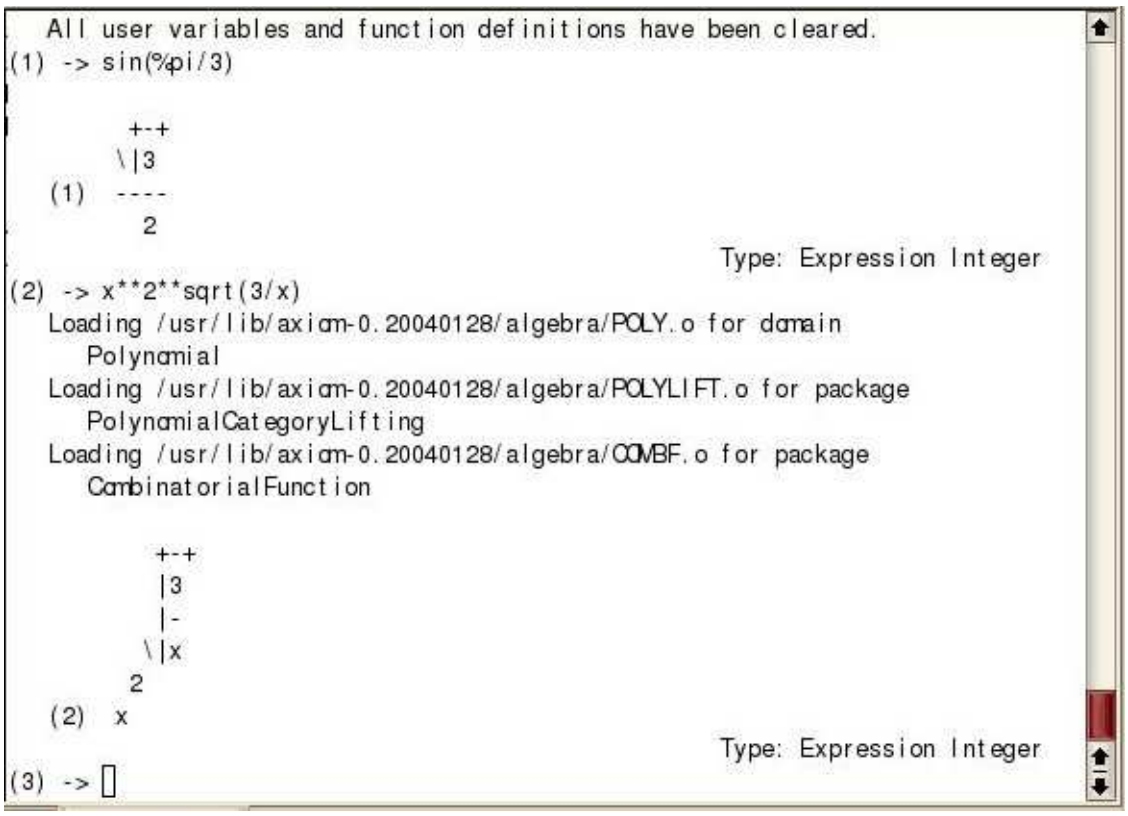


Figure 1. Exemple de calcul

L'utilisation du mode console est donc très peu attractif : les racines, matrices, et autres formules complexes apparaissent en mode ASCII, ce qui rend la lecture des résultats difficile, et exige parfois une certaine gymnastique visuelle afin de réussir à trouver le résultat correct.

Toutefois une alternative existe à ce problème : L'utilisation supplémentaire du logiciel GNU TeXmacs et l'utilisation de son plugin intégré Axiom.

Intégration à TeXmacs

1) Présentation de TeXmacs

TeXmacs se définit comme un éditeur WYSIWYG (what you see is what you get) de textes scientifiques : ce que vous voyez à l'écran est identique à ce que vous obtenez à l'impression. TeXmacs permet l'édition de textes structurés et présente beaucoup de caractéristiques qui le rendent extrêmement attrayant :

- Interface conviviale et assez facile à maîtriser. Il faudra cependant s'habituer à la différence de "philosophie" qui l'anime, vis-à-vis des traitements de texte plus classiques comme ceux de MS-Office et OpenOffice, où la notion de document structuré est peu employée. Il faut noter que TeXmacs n'est pas compatible avec les formats de traitement de textes habituels (doc, rtf...)

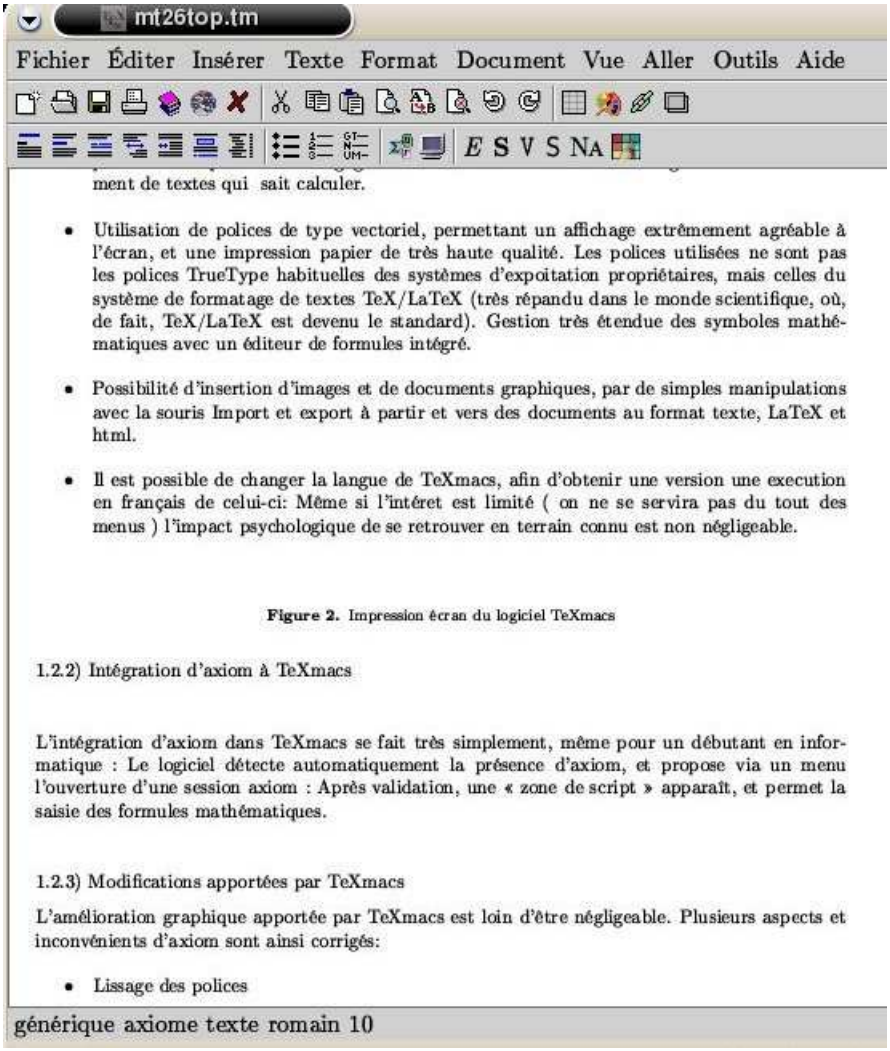
- Interfaçage avec des plusieurs logiciels libres de calcul numérique ou formel. TeXmacs 1.0 possède actuellement en standard une interface pour les logiciels de calcul de même que pour un interpréteur du langage Scheme. TeXmacs est donc un logiciel libre de traite-

ment de textes qui sait calculer.

- Utilisation de polices de type vectoriel, permettant un affichage extrêmement agréable à l'écran, et une impression papier de très haute qualité. Les polices utilisées ne sont pas les polices TrueType habituelles des systèmes d'exploitation propriétaires, mais celles du système de formatage de textes TeX/LaTeX (très répandu dans le monde scientifique, où, de fait, TeX/LaTeX est devenu le standard). Gestion très étendue des symboles mathématiques avec un éditeur de formules intégré.

- Possibilité d'insertion d'images et de documents graphiques, par de simples manipulations avec la souris Import et export à partir et vers des documents au format texte, LaTeX et html.

- Il est possible de changer la langue de TeXmacs, afin d'obtenir une version une exécution en français de celui-ci: Même si l'intérêt est limité (on ne se servira pas du tout des menus) l'impact psychologique de se retrouver en terrain connu est non négligeable.



ment de textes qui sait calculer.

- Utilisation de polices de type vectoriel, permettant un affichage extrêmement agréable à l'écran, et une impression papier de très haute qualité. Les polices utilisées ne sont pas les polices TrueType habituelles des systèmes d'expoitation propriétaires, mais celles du système de formatage de textes TeX/LaTeX (très répandu dans le monde scientifique, où, de fait, TeX/LaTeX est devenu le standard). Gestion très étendue des symboles mathématiques avec un éditeur de formules intégré.

- Possibilité d'insertion d'images et de documents graphiques, par de simples manipulations avec la souris Import et export à partir et vers des documents au format texte, LaTeX et html.

- Il est possible de changer la langue de TeXmacs, afin d'obtenir une version une execution en français de celui-ci: Même si l'intéret est limité (on ne se servira pas du tout des menus) l'impact psychologique de se retrouver en terrain connu est non négligeable.

Figure 2. Impression écran du logiciel TeXmacs

1.2.2) Intégration d'axiom à TeXmacs

L'intégration d'axiom dans TeXmacs se fait très simplement, même pour un débutant en informatique : Le logiciel détecte automatiquement la présence d'axiom, et propose via un menu l'ouverture d'une session axiom : Après validation, une « zone de script » apparaît, et permet la saisie des formules mathématiques.

1.2.3) Modifications apportées par TeXmacs

L'amélioration graphique apportée par TeXmacs est loin d'être négligeable. Plusieurs aspects et inconvénients d'axiom sont ainsi corrigés:

- Lissage des polices

Figure 2. Impression écran du logiciel TeXmacs

2) Intégration d'Axiom à TeXmacs

L'intégration d'Axiom dans TeXmacs se fait très simplement, même pour un débutant en informatique : Le logiciel détecte automatiquement la présence d'Axiom, et propose via un menu l'ouverture d'une session Axiom : Après validation, une « zone de script » apparaît, et permet la saisie des formules mathématiques.

3) Modifications apportées par TeXmacs

L'amélioration graphique apportée par TeXmacs est loin d'être négligeable. Plusieurs aspects et inconvénients d'Axiom sont ainsi corrigés:

- Lissage des polices
- Résultats des calculs très lisibles, structurés
- Coloration des calculs, résultats, types pour une différentiation plus aisée.
- Chargement des différentes librairies en arrière plan : pas d'affichage intempestif.

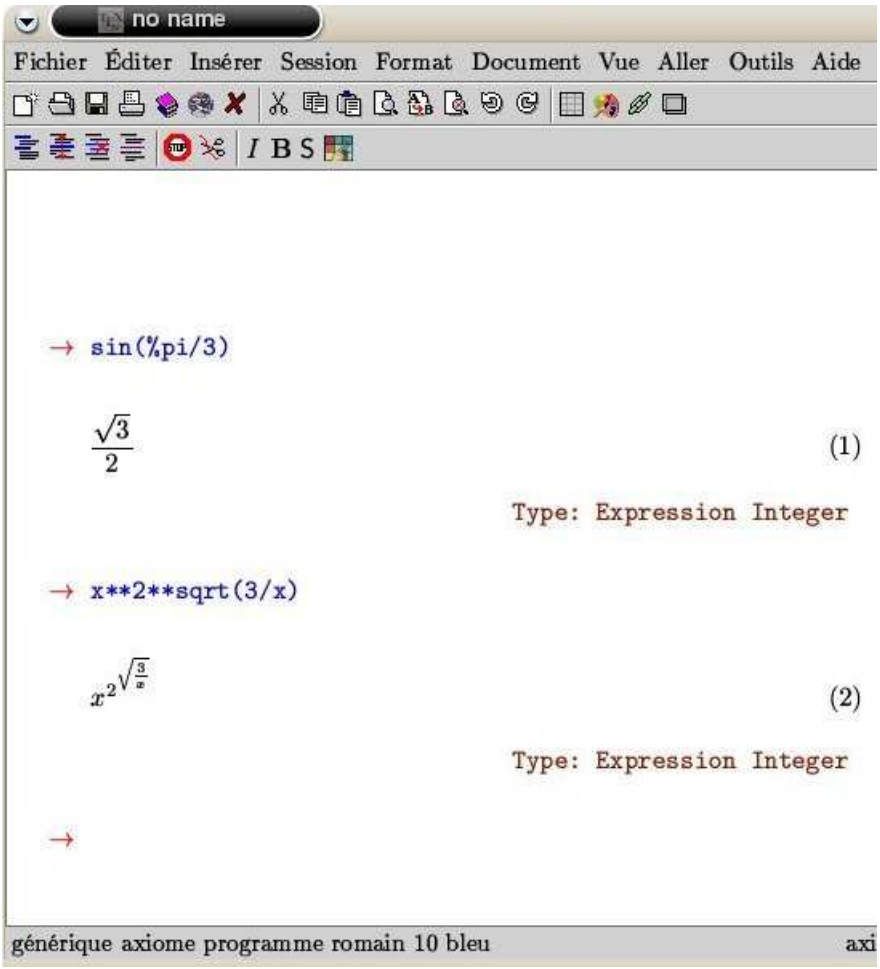


Figure 3. Utilisation d'une session Axiom dans TeXmacs

Le carré bleu extérieur symbolise la session Axiom, les parties bleu foncé le calcul, les parties marron le type du résultat, et le chiffre à droite du résultat, un historique de calcul (on saura ainsi dans quel ordre on a effectué les calculs)

GNUplot

Prise en main

La présentation graphique dépend du système d'exploitation : Sous Linux, tout menu est inexistant, alors que dans la version Windows, une série de menus permet de ne pas être « perdu » au lancement de l'application. Ainsi, la difficulté aux premiers abords dépend de la version utilisée.

L'utilisation de GNUplot, contrairement à Axiom, est impossible de façon pratique sans terminal graphique: Il est néanmoins possible d'exporter la courbe dans un fichier PostScript, latex, pdf, ou gif mais la visualisation de la courbe exige un environnement graphique: Il existe un composant supplémentaire, non fourni avec celui de base, dans le package GNUplot-x11. Ce package permet la création d'une fenêtre d'affichage, externe à la console. Celle ci ouvre ainsi une fenêtre de pré-visualisation à la demande d'affichage de la courbe.

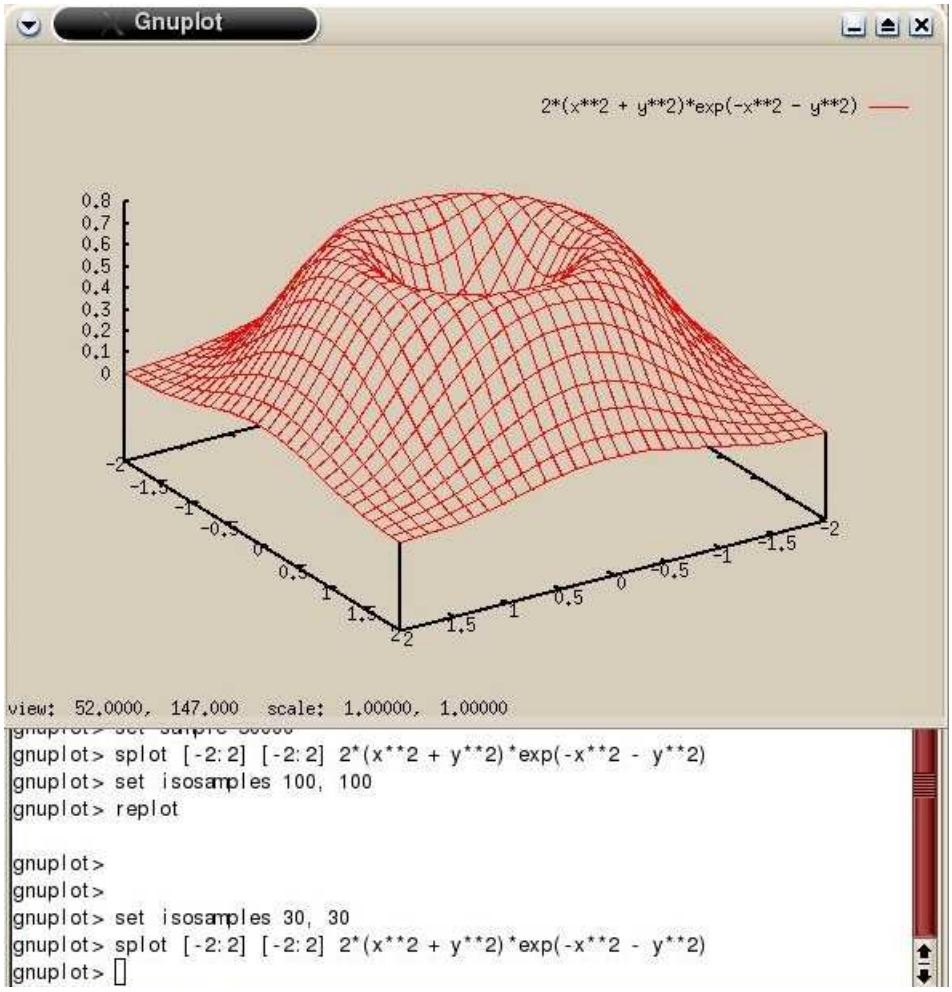


Figure 4. Exemple d'affichage d'une courbe (Environnement KDE)

Intégration dans TeXmacs

Tout comme Axiom, l'intégration d'une session GNUplot est possible dans TeXmacs. Néanmoins, l'avantage est bien plus limité,

- Il est impossible de faire évoluer, tourner les graphiques, une fois qu'ils ont été tracés.

- Il est impossible d'afficher les couleurs sur les graphiques, ce qui pose un problème lors d'affichage de plusieurs courbes en simultané. Ceci est du à l'utilisation du script TeXmacs : en effet celui-ci utilise un fichier d'exportation $\LaTeX$ et non PostScript. L'inconvénient est majeur car le format $\LaTeX$ exporté par GNUplot ne permet pas ou du moins pas aisément, une utilisation de la couleur.

- Il se révèle qu'entre chaque ligne tapée, toute variable, enregistrement, et paramètre enregistré se retrouve effacé, ce qui est très désavantageux: Il est impossible de déclarer une fonction, puis de l'afficher. Pour remédier à cela, on tapera toutes les commandes à la suite les unes des autres, tout en les séparant par des points virgules.

Pourtant, nous utiliserons cette méthode dans la résolution de la problématique car elle présente certains avantages :

- Possibilité d'intercaler une session Axiom avec une session GNUplot.

- Interaction simplifiées entre les 2 logiciels.

- Facilité de mise en page du présent rapport

Prise en main du logiciel

Axiom

Premiers pas

Axiom possède comme tout logiciel, certains avantages et certains inconvénients.

Le principal avantage est l'intuitivité du nom de la majorité des fonctions : Le nom de la fonction permet en général de comprendre l'utilité de la fonction en elle-même. Un système «d'aide» permet de découvrir les arguments à fournir à la fonction. Un système d'auto détection des arguments fournis à la fonction permet de choisir entre différentes utilisations de la fonction : On pourra ainsi utiliser integrate(f) tout comme integrate(f(x),x).

L'inconvénient majeur réside pourtant dans le système d'aide : il n'existe pas de regroupement par catégories des fonctions (comme dans Maple par exemple) De plus le fonctionnement de celles-ci n'est pas expliqué, et les tutoriels existant sont très limités dans leur champ d'application. Il n'existe pas de texte explicatif, comme dans Maple, pour chaque fonction.

Recherche de solutions

La recherche de solutions est souvent longue et fastidieuse pour Axiom : Les ressources sur internet sont maigres, et en général assez mal détaillées. Lorsque l'on bloque sur un problème, le fichier d'aide est totalement inutile. Les solutions aux problèmes sont donc souvent très difficiles à trouver. Il existe cependant un moyen de résoudre certains blocages : On peut faire un certain rapprochement avec Maple, beaucoup de commandes étant similaires à ce dernier.

GNUplot

Premiers pas

GNUplot a l'avantage de permettre une prise en main immédiate après lecture du fichier d'aide (et des tutoriels). Les fonctions sont simples et logiques et permettent d'obtenir des résultats très rapidement.

Recherche de solutions

On ne bloque jamais bien longtemps sur un problème. Les ressources sur internet sont abondantes et détaillées, le fichier d'aide est correct. On ne relève pas de difficultés majeures à l'utilisation générale de GNUplot.

Résolution des TPs MT26

Instructions Maple

Instructions d'algèbre

1) Soit le polynôme P(x)= x^3 - 3x + 2. En déterminer les racines réelles ou complexes et factoriser P. Déterminer le polynôme dérivé et les décompositions en éléments simples des fractions rationnelles Q1(x) = $\frac{1}{P(x)}$ et Q2(x)= $\frac{P'(x)}{P(x)}$.

→ `P:=x^3-3*x+2`

$$x^3 - 3x + 2 \tag{128}$$

Type: Polynomial Integer

On découvre ici la structure générale du calcul sous Axiom: En rouge, l'invite de de commande. en bleu, le calcul demandé. En noir en dessous le résultat. à droite, entre parenthèses, le « numéro » de calcul, suivi du type de résultat. Pour des raisons de place, on fera abstraction du type retourné par la fonction.

→ `)set messages type off`

On résout l'équation $x^3 - 3x + 2 = 0$

→ `solve(P=0,x)`

$$[x = 1,\ x = -2] \tag{129}$$

On factorise P

→ `factor(P)`

$$(x-1)^2(x+2) \tag{130}$$

On calcule sa dérivée

→ `differentiate(P)`

$$3x^2 - 3 \tag{131}$$

→ `Q1:=1/P`

$$\frac{1}{x^3 - 3x + 2} \tag{132}$$

→ `Q2:=differentiate(P)/P`

$$\frac{3}{x^3 - 3x + 2}x^2 - \frac{3}{x^3 - 3x + 2} \tag{133}$$

Pour décomposer la fonction en éléments simples, on utilise ici une syntaxe particulière : « ::UP(x, FRAC INT) ». La fonction partialFraction prend en effet en paramètres 2 arguments de type Univariate Polynomial (UP) or P est un polynôme Fraction Integer. On effectue ainsi le changement demandé : ::NOUVEAUTYPE(variable, ANCIENTYPE)

→ partialFraction(1,P::UP(x, FRAC INT))

$$\frac{-\frac{1}{9}x+\frac{4}{9}}{(x-1)^2}+\frac{\frac{1}{9}}{x+2} \tag{134}$$

→ partialFraction(differentiate(P),P::UP(x, FRAC INT))

$$\frac{2}{x-1}+\frac{1}{x+2} \tag{135}$$

2) Même exercice pour P1(x)=x^3-3x^2+2x, P2(x)=x^4+1 et P3(x)=$\sqrt{2}$(x^4+1). Étudier la différence entre les deux derniers cas et expliquer pourquoi.

→ P1:=x^3-3*x^2+2*x

$$x^3-3x^2+2x \tag{136}$$

→ P2:=x^4+1

$$x^4+1 \tag{137}$$

→ P3:=sqrt(2)*(x^4+1)

$$\sqrt{2}x^4+\sqrt{2} \tag{138}$$

→ factor(P1)

$$(x-2)(x-1)x \tag{139}$$

→ factor(P2)

$$x^4+1 \tag{140}$$

→ factor(P3)

$$\sqrt{2}\left(x^4+1\right) \tag{141}$$

→ partialFraction(1,P1::UP(x, FRAC INT))

$$\frac{\frac{1}{2}}{x-2}-\frac{1}{x-1}+\frac{\frac{1}{2}}{x} \tag{15}$$

Type: PartialFraction UnivariatePolynomial(x,Fraction Integer)

→ partialFraction(1,P2::UP(x, FRAC INT))

$$\frac{1}{x^4+1} \tag{16}$$

Type: PartialFraction UnivariatePolynomial(x,Fraction Integer)

→ partialFraction(1,P3::UP(x, FRAC AN))

UnivariatePolynomial(x,Fraction AlgebraicNumber) is not a valid type.

3) Résolution des systèmes:

$$\text{(S1)}\quad \begin{cases} 3x+4y-z=12 \\ x-y+2z=-3 \\ x+y+z=2 \end{cases}$$

On doit basculer tous les termes de l'égalité du même coté, afin d'avoir un systèmes d'équations, sans secondes membres. La résolution se fait pas l'intermédiaire d'une liste d'équations.

→ solve([3*x+4*y-z-12, x-y+2*z+3 , x+y+z-2],[x,y,z])

$$[[x=1,\ y=2,\ z=-1]] \tag{17}$$

$$\text{S(2)}\quad \begin{cases} 3x+4y-z=12 \\ x-y+2z=-3 \\ 4x+3y+z=9 \end{cases}$$

→ solve([3*x+4*y-z-12, x-y+2*z+3 , 4*x+3*y+z-9],[x,y,z])

$$[[x=-\%B,\ y=\%B+3,\ z=\%B]] \tag{18}$$

4) Briseurs de codes

a) les nombres suivants sont-ils premiers ? Noter le temps de calcul pour chacun d'eux.

Pour ne pas avoir à calculer le temps de calcul manuellement, on active l'option temps de calcul. Ce temps de calcul est mesuré par le moteur d'Axiom, et non celui de TeXmacs, ce qui résulte un décalage entre le temps annoncé et le temps mesuré. L'affichage sur de très nombreuses lignes est un facteur d'erreur.

→)set messages time on

→ prime?(58358298767740033)

$$false \tag{19}$$

Time: 0.01 (OT) = 0.01 sec

→ prime?(408506280913741802317)

$$true \tag{20}$$

Time: 0 sec

```
→ prime?(factorial(30)+1)
```

$$false \tag{21}$$

```
                                                              Time: 0 sec
```

```
→ prime?(131124424344439150898996787974484867)
```

$$false \tag{22}$$

```
                                                  Time: 0.01 (OT) = 0.01 sec
```

```
→ prime?(7719068319927551)
```

$$true \tag{23}$$

```
                                                             Time: 0 sec
```

```
→ prime?(834384970974248697794)
```

$$false \tag{24}$$

```
                                                              Time: 0 sec
```

```
→ factors(58358298767740033)
```

$$[[factor=563, exponent=1], [factor=3233509, exponent=1], [factor=3205\backslash 6799, exponent=1]] \tag{25}$$

```
                                  Time: 0.01 (IN) + 0.01 (OT) + 0.08 (GC) = 0.10 sec
```

```
→ factors(factorial(30)+1)
```

$$[[factor=31, exponent=1], [factor=12421, exponent=1], [factor=82561, exponent=1], [factor=1080941, exponent=1], [factor=7719068319927551, exponent=1]] \tag{26}$$

```
                                        Time: 0.03 (IN) + 0.01 (OT) + 0.12 (GC) = 0.16 sec
```

```
→ factors(131124424344439150898996787974484867)
```

$$[[factor=3, exponent=1], [factor=947, exponent=1], [factor=32165873, exponent=1], [factor=1434884743560008132693419, exponent=1]] \tag{27}$$

```
                                           Time: 0.03 (IN) + 0.12 (GC) = 0.15 sec
```

```
→ factors(834384970974248697794)
```

$$[[factor=2, exponent=1], [factor=7, exponent=2], [factor=1103, exponent=1], [factor=7719068319927551, exponent=1]] \tag{28}$$

```
                                           Time: 0.01 (IN) + 0.04 (GC) = 0.05 sec
```

```
→ )set messages time off
```

Note: Les réponses aux calculs sont quasi-instantanés.

Instructions d'analyse

1) Calculs de sommes

a) Calculer pour plusieurs valeurs de n, puis dans le cas général, la somme Sn= $\sum_{k=1}^{n} k^2$ et vérifier la relation Sn=$\frac{n(n+1)(2n+1)}{6}$

→ sum(k^2, $k = 1..5$)

$$55 \tag{30}$$

→ sum(k^2,k=1..n)

$$\frac{2n^3+3n^2+n}{6} \tag{31}$$

→ (n)*(n+1)*(2*n+1)

$$2n^3+3n^2+n \tag{32}$$

2) Calculer les limites suivantes

→ limit((sqrt(1+x)*sin(x)+log(1-x))/(x-sin(x)),x=0)

$$-\frac{15}{4} \tag{33}$$

→ limit((x^2*(1+1/x)^x-%e*x^3*log(1+1/x)),x=%plusInfinity)

$$\frac{e}{8} \tag{34}$$

3) Calcul d'intégrales et de primitives

→ integrate((x+1)/(x^2+1),x=0..1)

$$\frac{2\log(2)+\pi}{4} \tag{35}$$

→ integrate((x+1)/(x^2+1),x=0..%plusInfinity)

$$+\infty \tag{36}$$

→ integrate(1/(x^2+4),x=0..%plusInfinity)

$$\frac{\pi}{4} \tag{37}$$

→ integrate((x+1)/(x^2+1),x)

$$\frac{\log\left(x^2+1\right)+2\arctan\left(x\right)}{2} \tag{38}$$

4) Intégrer par partie, puis calculer.

```
→ integrate(x^2*sin(x),x)
```

$$2x\sin\left(x\right)+\left(-x^2+2\right)\cos\left(x\right) \tag{39}$$

```
→ integrate(x^2*sin(x),x=0..%pi)
```

$$\pi^2-4 \tag{40}$$

4) Effectuer un changement de variable

```
→ integrate(x/sqrt(4-9*x^2),x)
```

$$\frac{x^2}{\sqrt{-9x^2+4}-2} \tag{41}$$

Type:

```
→ subst(integrate(x/sqrt(4-9*x^2),x),[x=sqrt(t)])
```

$$\frac{t}{\sqrt{-9t+4}-2} \tag{42}$$

5) Développements limités

On active l'option streams en niveau 5: Cela signifie que les développements limités vont être réalisés à l'ordre 5. En théorie, le dernier argument de series sert à cela, mais le résultat est incorrect.

```
→ )set streams calculate 5
```

```
→ series(tan(x),x=0,5)
```

$$x+\frac{1}{3}x^3+\frac{2}{15}x^5+O\left(x^7\right) \tag{43}$$

```
→ series(log(tan(x)),x=0)
```

$$\log\left(x\right)+\frac{1}{3}x^2+\frac{7}{90}x^4+O\left(x^6\right) \tag{44}$$

6) Calcul différentiel

```
→ f:= operator('f);
```

```
→ x^2*D(f(x),x)+f(x)=0
```

$$x^2 f'(x) + f(x) = 0 \tag{46}$$

On utilise % pour rappeler le résultat du dernier calcul. Ici % correspond à $x^2 f'(x) + f(x) = 0$

→ `solve(%,f,x)`

$$\left[particular = 0,\ basis = \left[e^{\frac{1}{x}}\right]\right] \tag{47}$$

7) Calculer pour des valeurs simples de a, les sommes $\sum_{x=1}^{\infty} \frac{1}{x^a}$

→ `n:=5`

$$5 \tag{48}$$

Le calcul d'une somme de série lorsque celle-ci tend vers l'infini, et cela même si elle converge, est impossible dans Axiom. Les domaines infinis ne sont pas pris en charge.

→ `sum(1/x^n,x=1..%plusInfinity)`

```
There are 6 exposed and 2 unexposed library operations named sum
   having 2 argument(s) but none was determined to be applicable.
   Use HyperDoc Browse, or issue
                              )display op sum
   to learn more about the available operations. Perhaps
   package-calling the operation or using coercions on the arguments
   will allow you to apply the operation.


Cannot find a definition or applicable library operation named sum
   with argument type(s)
                        Fraction Polynomial Integer
                  SegmentBinding OrderedCompletion Integer


   Perhaps you should use "@" to indicate the required return type,
   or "$" to specify which version of the function you need.
```

Instructions d'algèbre linéaire

1) Réduction des matrices

a) Saisir la matrice M, en calculer le déterminant, puis l'inverse

→ `m:=matrix [[4,-1,-1],[-1,4,-1],[-1,-1,4]]`

$$\begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{bmatrix} \tag{49}$$

On réutilise % pour appeler le dernier résultat.

```
→ determinant(%)
```

$$50 \tag{50}$$

```
→ m^-1
```

$$\begin{bmatrix} \frac{3}{10} & \frac{1}{10} & \frac{1}{10} \\ \frac{1}{10} & \frac{3}{10} & \frac{1}{10} \\ \frac{1}{10} & \frac{1}{10} & \frac{3}{10} \end{bmatrix} \tag{51}$$

```
→ characteristicPolynomial(m, x)
```

$$-x^3+12x^2-45x+50 \tag{52}$$

```
→ solve(%=0,x)
```

$$[x=5,\ x=2] \tag{53}$$

```
→ eigenvectors(m)
```

$$\left[\left[eigval=2,\ eigmult=1,\ eigvec=\left[\begin{bmatrix}1\\1\\1\end{bmatrix}\right]\right],\ \left[eigval=5,\ eigmult=2,\ eigvec=\left[\begin{bmatrix}-1\\1\\0\end{bmatrix},\ \begin{bmatrix}-1\\0\\1\end{bmatrix}\right]\right]\right] \tag{54}$$

2) Même question pour une autre matrice

```
→ m1 := matrix [[4,-2,-2],[2,4,-2],[-2,-2,4]]
```

$$\begin{bmatrix} 4 & -2 & -2 \\ 2 & 4 & -2 \\ -2 & -2 & 4 \end{bmatrix} \tag{55}$$

```
→ determinant(m1)
```

$$48 \tag{56}$$

```
→ m1^-1
```

$$\begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{12} & \frac{1}{4} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{4} & \frac{5}{12} \end{bmatrix} \tag{57}$$

→ characteristicPolynomial(m1, x)

$$-x^3+12x^2-44x+48 \tag{58}$$

→ solve(%=0,x)

$$[x=6,\ x=4,\ x=2] \tag{59}$$

→ eigenvectors(m1)

$$\left[\left[eigval=2,\ eigmult=1,\ eigvec=\left[\left[\begin{matrix}1\\0\\1\end{matrix}\right]\right]\right],\ \left[eigval=4,\ eigmult=1,\ eigvec=\left[\left[\begin{matrix}1\\-1\\1\end{matrix}\right]\right]\right],\ \left[eigval=6,\ eigmult=1,\ eigvec=\left[\left[\begin{matrix}0\\-1\\1\end{matrix}\right]\right]\right]\right] \tag{60}$$

3) Matrice de Vandermonde

On vide les propriétés de x y z et t afin de réaliser le prochain exercice. Ainsi le type de matrice n'est pas erroné.

→)clear properties x y z t

→ mv := matrix
[[1,x,x**2,x**3],[1,y,y**2,y**3],[1,z,z**2,z**3],[1,t,t**2,t**3]]

$$\begin{bmatrix}1 & x & x^2 & x^3\\ 1 & y & y^2 & y^3\\ 1 & z & z^2 & z^3\\ 1 & t & t^2 & t^3\end{bmatrix} \tag{61}$$

On remarque que lorsqu'une fonction ne possède qu'un argument, on n'est pas obligé de mettre cet argument entre parenthèses accolées à la fonction.

→ determinant mv

$$\left((-x+t)y^2+\left(x^2-t^2\right)y-t\,x^2+t^2x\right)z^3+\left((x-t)y^3+\left(-x^3+t^3\right)y+t\,x^3-t^3x\right)z^2+\left(\left(-x^2+t^2\right)y^3+\left(x^3-t^3\right)y^2-t^2x^3+t^3x^2\right)z+\left(t\,x^2-t^2x\right)y^3+\left(-t\,x^3+t^3x\right)y^2+\left(t^2x^3-t^3x^2\right)y \tag{62}$$

→ factor(%)

$$-(x-t)(y-x)(y-t)(z-y)(z-x)(z-t) \tag{63}$$

Graphiques

- Partie Axiom : La partie traitée sera uniquement numérique. En effet, il semble qu'une implémentation graphique d'Axiom existe, mais elle nécessite la présence d'un « viewport manager », composant introuvable malgré toutes les recherches effectuées. Les commandes de tracés sont effectuées via la fonction draw (fonction, variables). Après questionnement d'un développeur Axiom, il semblerait que cette fonction n'ai pas été importée lors du portage d'Axiom. Le viewport manager était intégré dans la version commerciale d'Axiom. Une implémentation future dans la version libre est à venir.

Courbes dans le plan

1) Représentations graphiques

a) Définir la fonction f(x)=$\frac{x^2+4x-5}{x+1}$ et en construire la courbe représentative.

b) Décomposer la fraction en éléments simples et tracer l'asymptote sur le même graphique.

```
→ F:=(x^2+4*x-5)/(x+1)
```

$$\frac{x^2+4x-5}{x+1} \tag{64}$$

```
→ partialFraction((x^2+4*x-5)::UP(x, FRAC INT),(x+1)::UP(x, FRAC INT))
```

$$x+3-\frac{8}{x+1} \tag{65}$$

2) Soit G_n définie sur $\mathbf{R}$ par $G_n(y) = 1\text{-}e^{-\lambda y^n}$ si $y \geqslant 0$, 0 si $y < 0$ et sa dérivée $g_n(y)$.

a) Représenter graphiquement G_n et g_n pour plusieurs valeurs de n entier naturel positif.

Pour définir une fonction par morceaux, on fait appel aux fonctions de programmation d'Axiom, la fonction « piecewise » n'existant pas.

```
→ a(y) == if y < 0 then 0 else 1-exp(-y^n)
→ n:=2
```

$$2 \tag{67}$$

```
→ a(-3)

   Compiling function a with type Integer -> Expression Integer
```

$$0 \tag{68}$$

```
→ a(5)

   Compiling function a with type PositiveInteger -> Expression Integer
```

$$\frac{e^{25}-1}{e^{25}} \tag{69}$$

3) On considère la fonction f définie par $f(x) = x$ si $|x| < 1$ et $1/x$ sinon

a) Définir f et tracer sa courbe représentative
b) Étudier l'existence des intégrales généralisées suivantes

```
→ f(x) == if abs(x) < 1 then 1 else 1/x                    Type: Void
→ f(0.5)

   Compiling function f with type Float -> Float
```

$$1.0 \tag{2}$$

L'intégration de fonctions définies en programmation est impossible.

```
→ integrate(f(x), x=0..%plusInfinity)


      >> Error detected within library code:
      integrate: pole in path of integration
   protected-symbol-warn called with (NIL)
```

4) Définition des fonctions périodiques du polycopié:

```
→ h(x)==acos(cos(x))
```

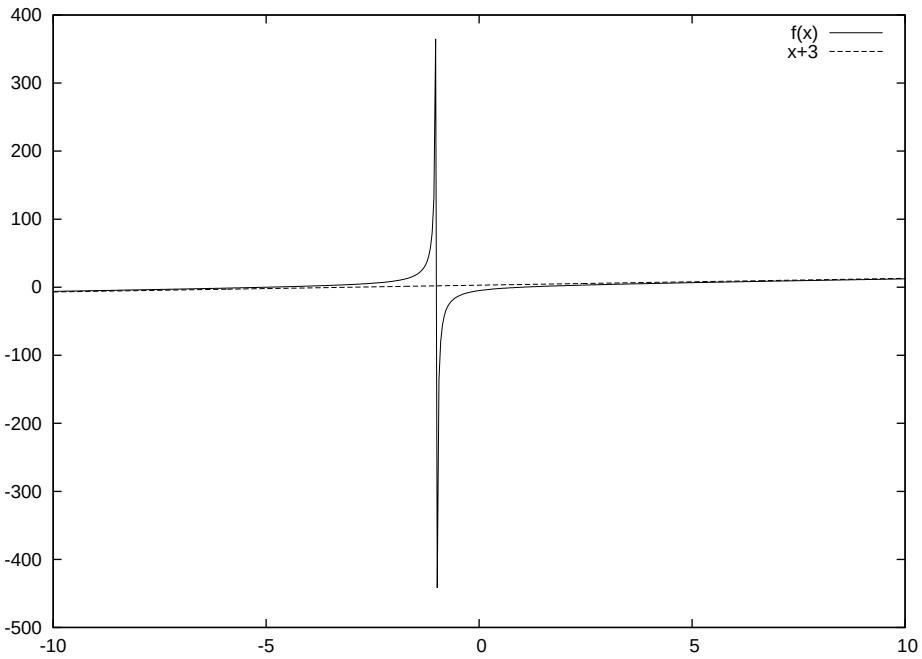
```
→ g(x)==atan(tan(x))
```

5 et 6: Les tracés étant impossibles à réaliser avec Axiom, cette partie sera traitée uniquement avec GNUplot.

- Partie GNUplot: La partie traitée sera uniquement numérique, GNUplot n'étant pas un logiciel de calcul formel, mais uniquement de tracé. On utilisera les résultats trouvés précédemment pour le tracé des courbes.

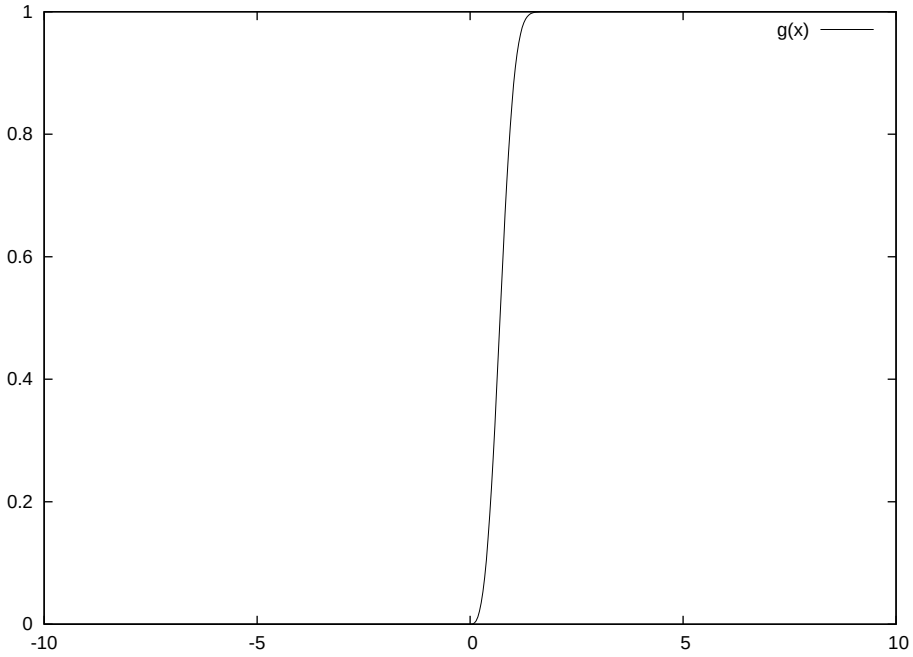
La commande set sample 500 sert à modifier la précision du tracé. 500 est un bon compromis entre rapidité d'exécution et qualité de tracé. On définit ensuite la fonction f(x), puis on trace sur le même graphique f(x), et x+3

```
GNUplot] set sample 500;f(x)=(x**2+4*x-5)/(x+1);plot f(x),x+3
```

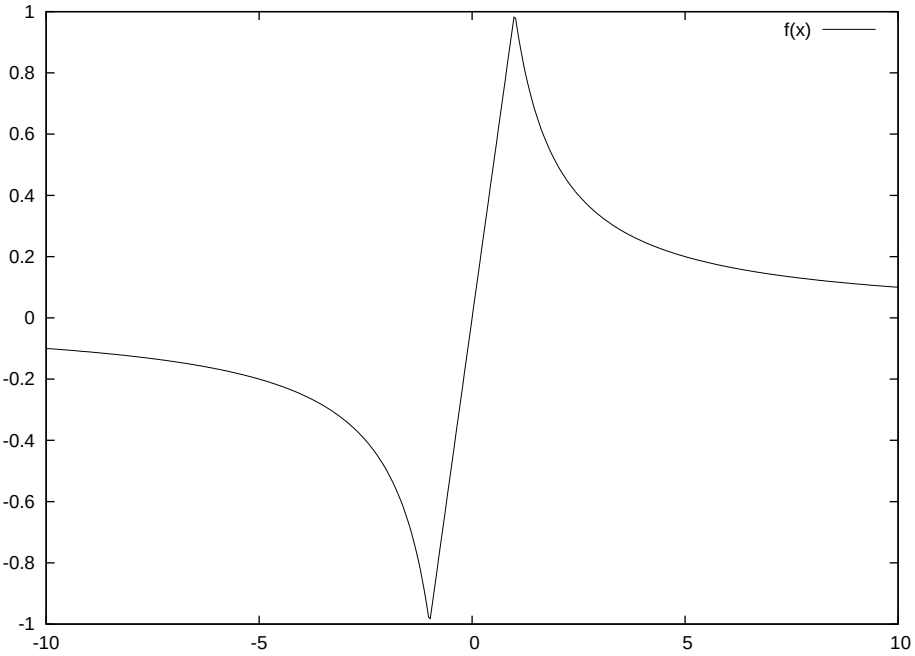


On définit la fonction par morceaux de la façon qui suit :
fonction= test?résultat:test?résultat:...:dernier_cas

```
GNUplot]  set sample 500;g(x) = 0>x ? 0 :1-exp(-2*x**n);n=3;plot g(x)
```

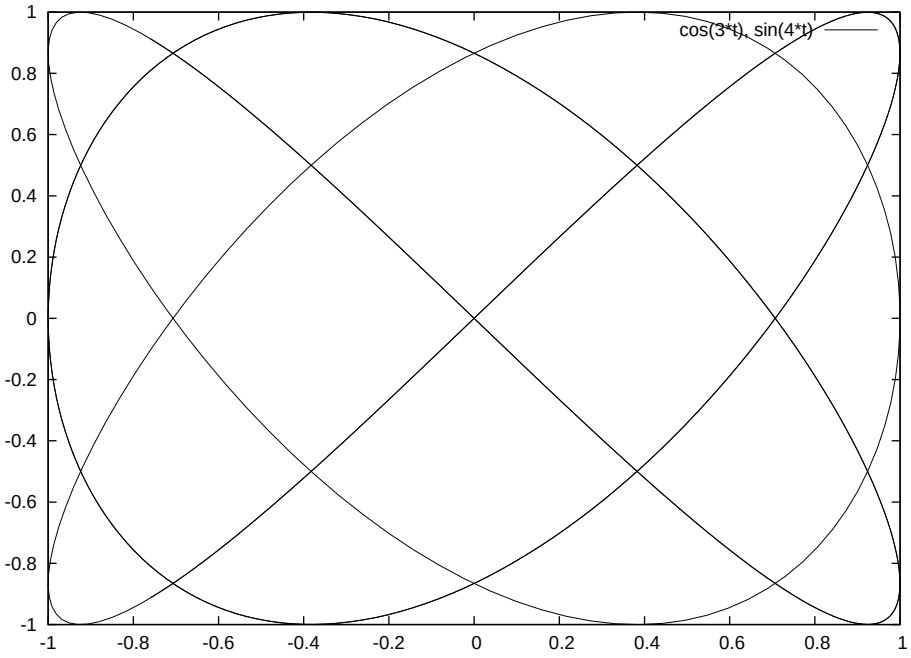



```
GNUplot] set sample 500;f(x) = 1>abs(x) ? x : 1/x; plot f(x)
```

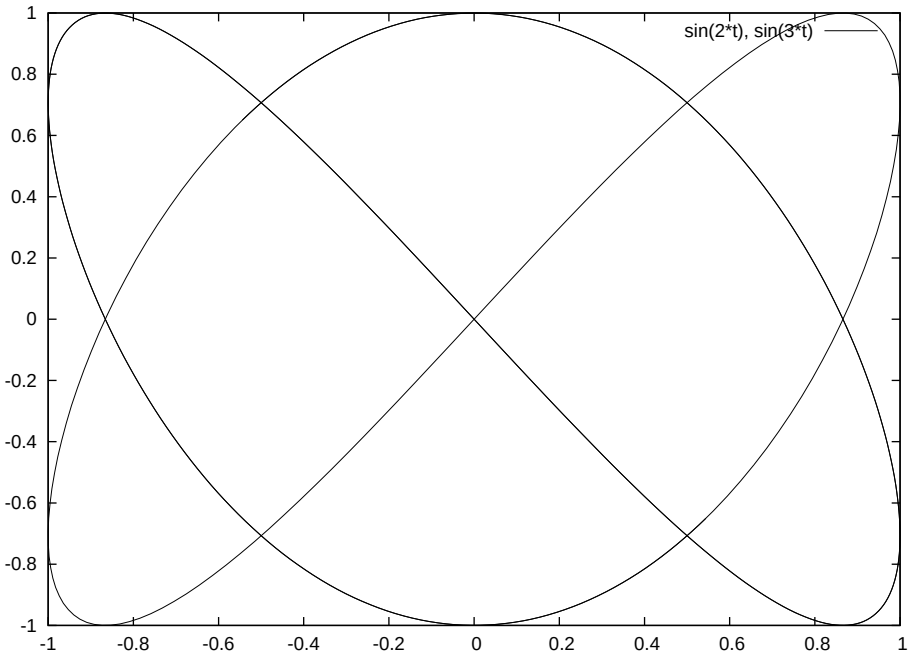


On active le mode paramétrique par la commande set parametric. La variable par défaut est t.

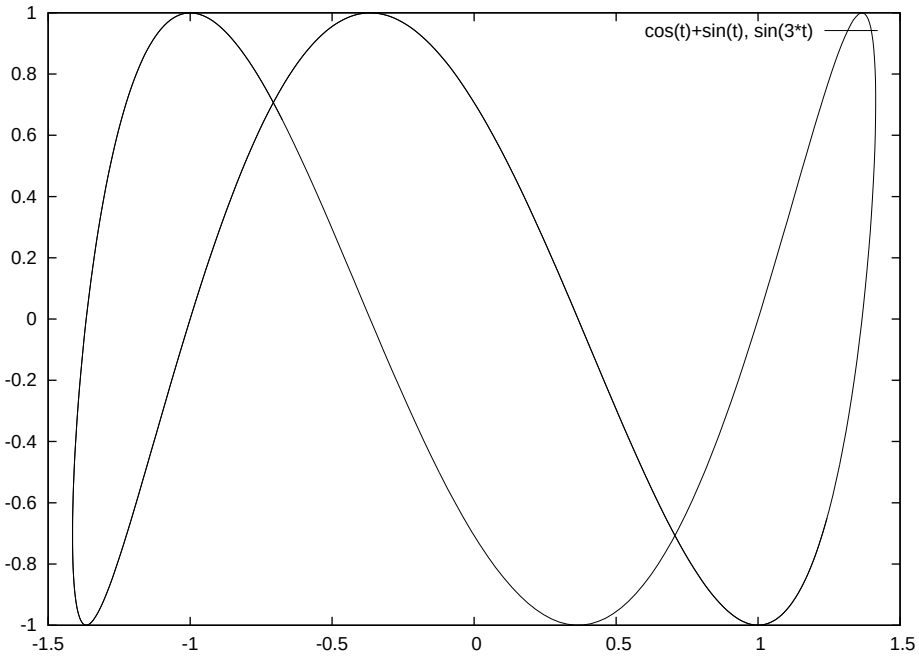
```
GNUplot] set sample 500;set parametric;plot cos(3*t),sin(4*t)
```



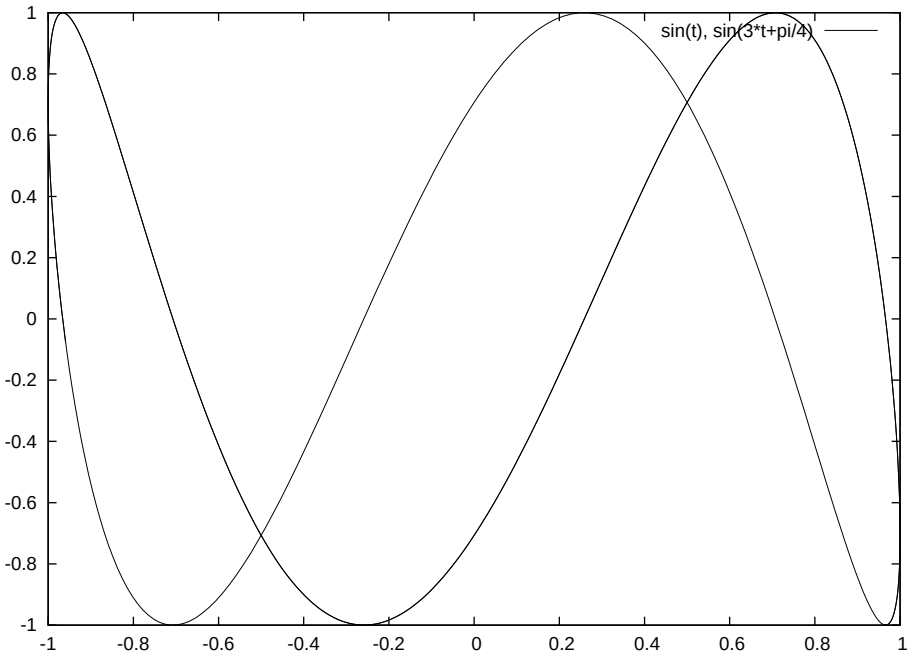

```
GNUplot] set parametric;set sample 500;plot sin(2*t),sin(3*t)
```



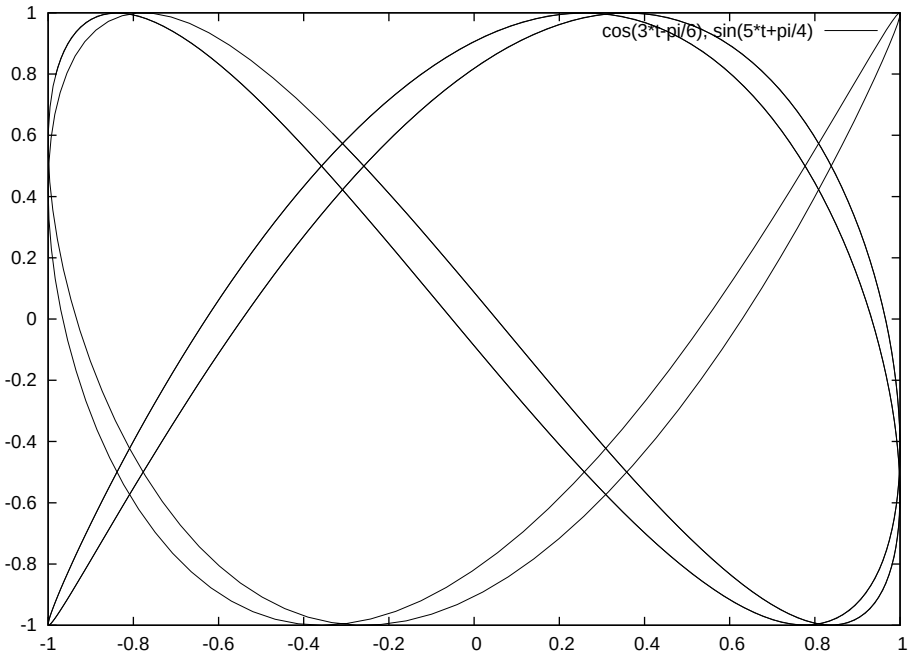
```
GNUplot] set parametric;set sample 500;plot cos(t)+sin(t),sin(3*t)
```




```
GNUplot] set parametric;set sample 500;plot sin(t),sin(3*t+pi/4)
```

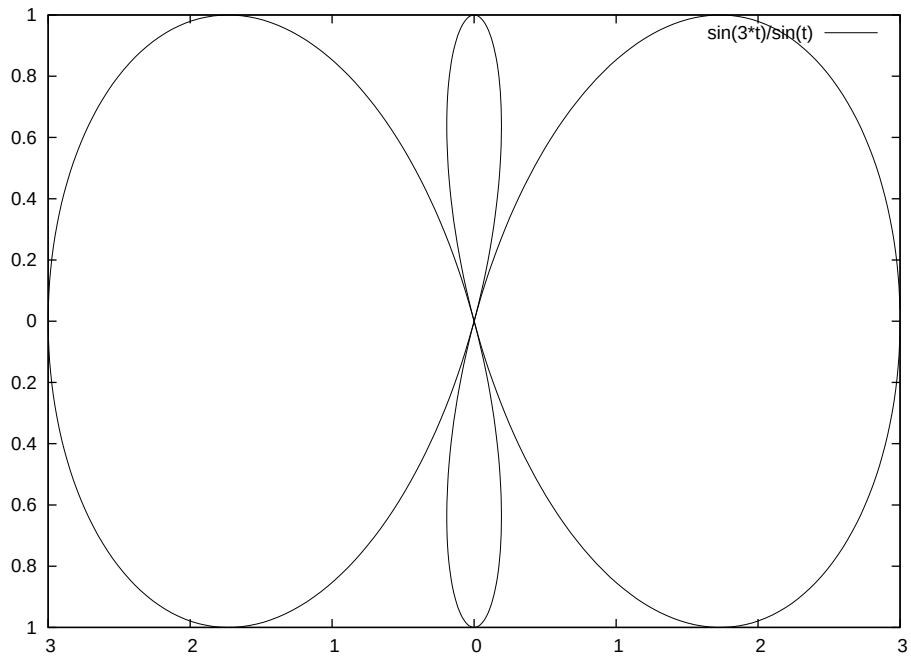


```
GNUplot] set parametric;set sample 500;plot cos(3*t-pi/6),sin(5*t+pi/4)
```

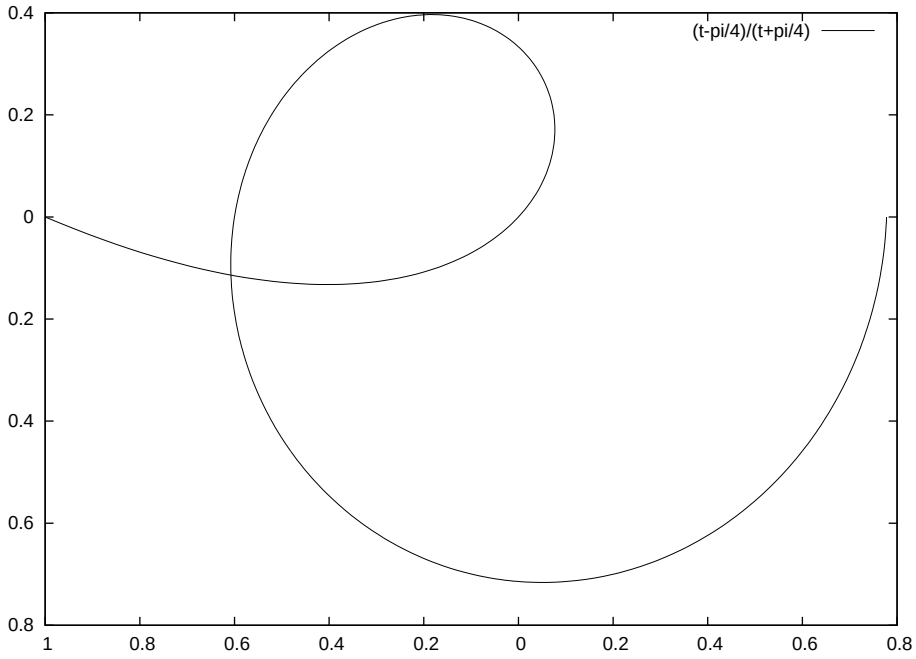


On active le mode polaire. Par défaut la variable est t.

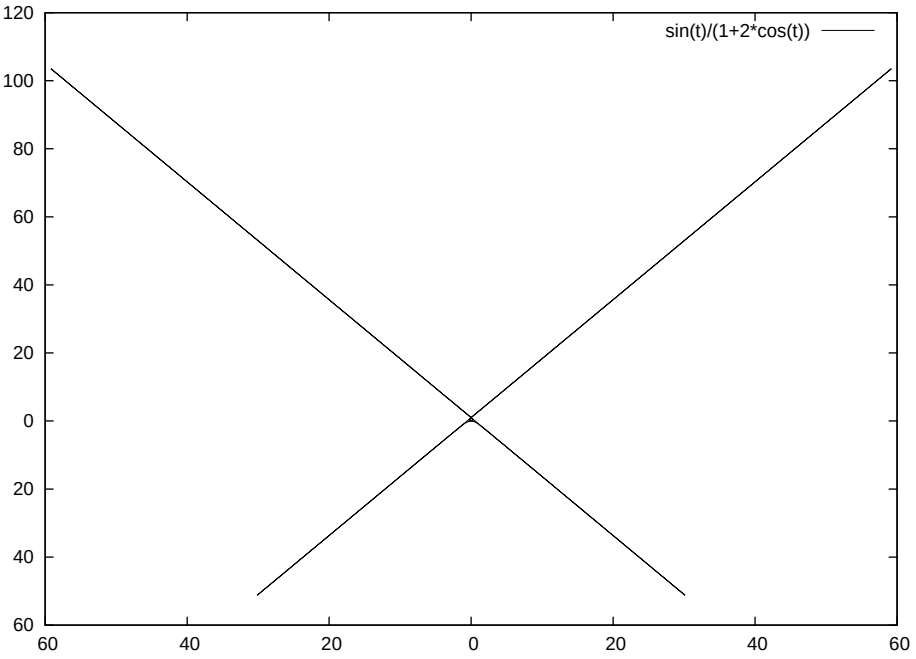
```
GNUplot] set polar;set sample 500;plot sin(3*t)/sin(t)
```



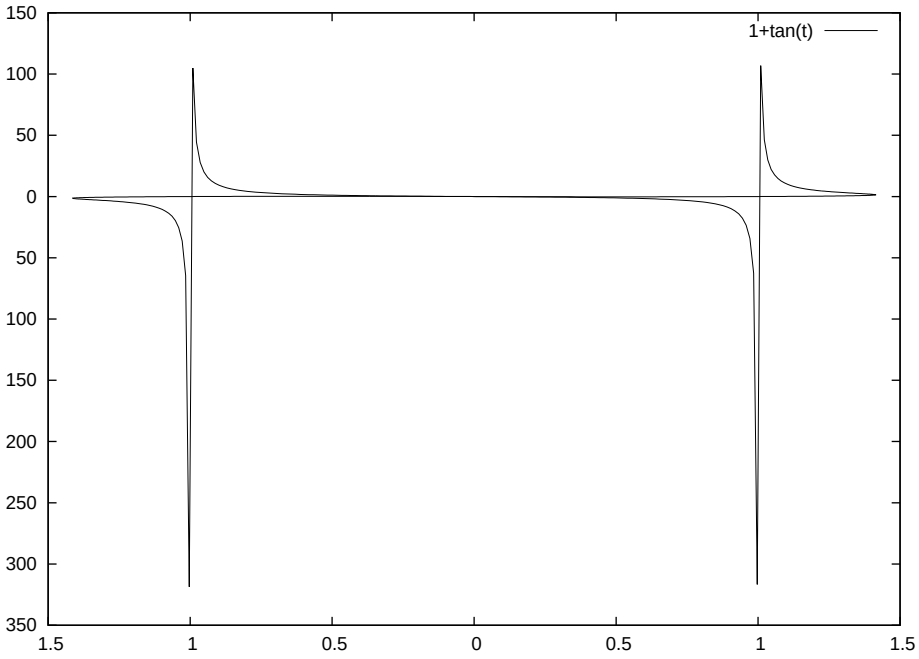
```
GNUplot] set polar;set sample 500;plot (t-pi/4)/(t+pi/4)
```




```
GNUplot] set polar;set sample 500;plot sin(t)/(1+2*cos(t))
```



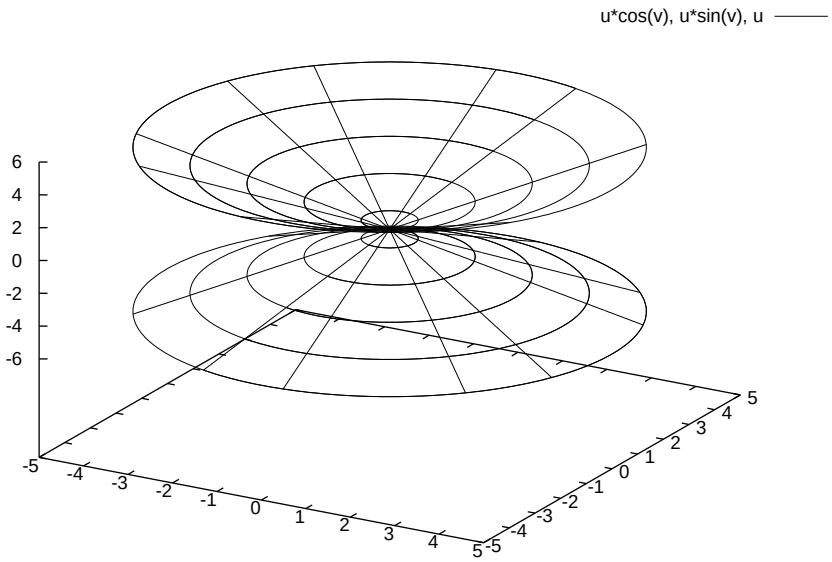
```
GNUplot] set polar;set sample 500; plot 1+tan(t)
```



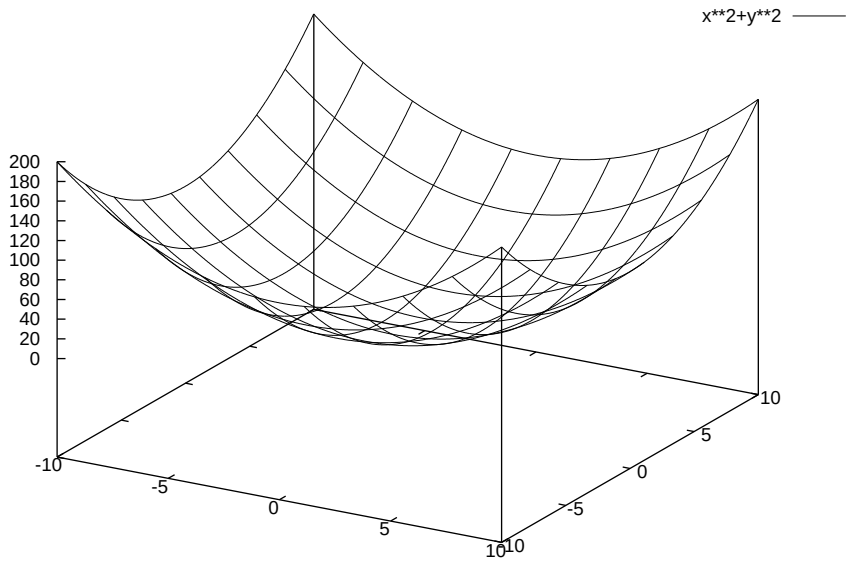
Surfaces dans l'espace

En mode paramétrique, les variables de surface sont u et v.

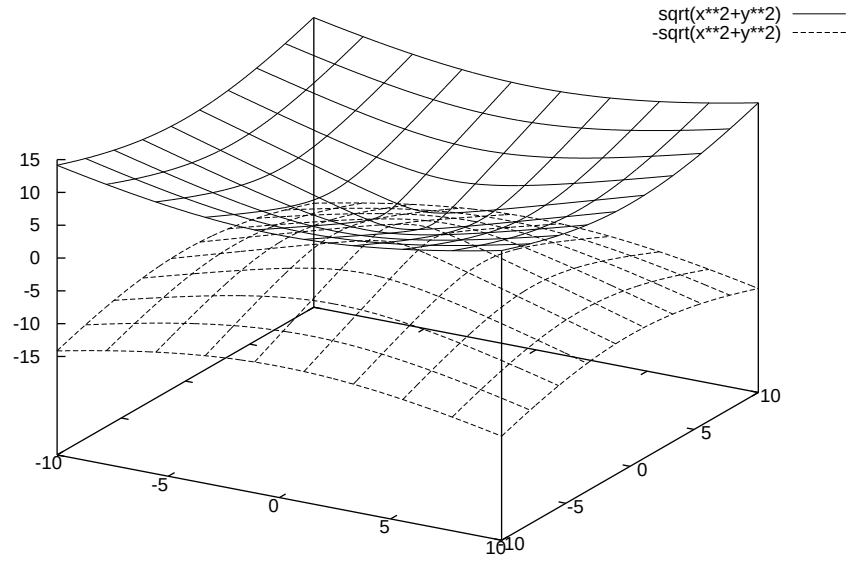
```
GNUplot] set sample 500;set parametric;splot u*cos(v),u*sin(v),u
```



```
GNUplot] set sample 500;splot [-10:10] [-10:10] x**2+y**2
```




```
GNUplot] set sample 500;splot sqrt(x**2+y**2),-sqrt(x**2+y**2)
```



Programmation

Suites

1) Peut on faire confiance à la calculatrice ?

Soit la suite $(un)_n$ définie par $u_0 = \frac{11}{2}$, $u_1 = \frac{61}{11}$, u_{n+1}=111-$\frac{1130}{u_n}$+$\frac{3000}{u_n \cdot u_{n-1}}$

a) Calculer les premiers termes. Quelle est la limite apparente ?

Il est malheureusement impossible (ou du moins très peu pratique) de taper directement les lignes de programmation dans Axiom. En effet, si l'on ne passe pas par l'intermédiaire d'un fichier texte, il est impossible de donner une indentation correcte au programme. On créera donc tout d'abord un fichier (ici nommé exo1.input) dont la structure est celle donnée ci-dessous, puis o chargera ce fichier dans Axiom par la commande)read . Un debbuger plus ou moins efficace se charge de détecter les erreurs. on commentera les lignes par des #, même si le commentaire n'est pas inclut dans Axiom.

Contenu de exo1.input et structure générale des programmes à écrire dans un fichier texte.

```
p : Integer -> Float
# On défini une fonction p comme prenant un entier en parametre, et retournant un
# réel.
p n ==
# On donne le nom du paramêtre: n
   if n = 0 then
         # On remarque l'absence de end, et autres crochets. Cela est du au fait
         # qu'axiom fait attention à l'indentation pour définir les structures
         # imbriquées
         11/2
         # On retourne 11/2
      else
         if n = 1 then
               61/11
            else
               111- 1130/( p (n-1)) + 3000/( p (n-1) * p (n-2))
               # Un résultat seul sur une ligne signifie la fin du programme,
               # et le retour de la valeur par la fonction
```

```
→ )read exo1.input


   p : Integer -> Float



                                                                  Type: Void p n ==

      if n = 0 then

            11/2

         else

            if n = 1 then

                  61/11
```

```
        else
           111- 1130/( p (n-1)) + 3000/( p (n-1) * p (n-2))
```

```
→ p 10
```

```
   Compiling function p with type Integer -> Float
   Compiling function p as a recurrence relation.
```

$$5.86095152603794319 \tag{76}$$

```
→ p 15
```

$$5.94398604992555088 \tag{77}$$

```
                                                              Type: Float
```

```
→ p 20
```

$$98.676371816945483885 \tag{78}$$

```
                                                              Type: Float
```

c) Calculer, pour diverses valeurs de n, $w_n = \frac{6^{n+1}+5^{n+1}}{6^n+5^n}$

```
→ w(n) == (6^(n+1)+5^(n+1))/(6^n +5^n)
```

```
                                                              Type: Void
```

```
→ w(10)
```

```
   Compiling function w with type PositiveInteger -> Fraction Integer
```

$$\frac{411625181}{70231801} \tag{80}$$

```
→ w(10) :: Float
```

$$5.8609515225161319728 \tag{81}$$

```
→ w(15) :: Float
```

$$5.9390504854611181512 \tag{83}$$

Remarque importante: Axiom fait des erreurs d'arrondi. On remarque une différence significative entre les valeurs des 2 fonctions en 15. On peut néanmoins assez facilement corriger cette erreur en modifiant le type retourné par la fonction en Fraction Integer au lieu de Float.

```
→ w(100) :: Float
```

$$5.9999999879253266734 \tag{82}$$

2) Étude suivant a de la suite définie par $u_0 = \frac{1}{2}$ $u_n = 4a\,u_n\,(1 - u_n)$

```
→ )read exo2.input

   u : Integer -> Float


                                                          Type: Void u n ==
      a := 0.2
      if n = 0 then
            1/2
         else
            4*a*u (n-1)*(1 - u (n-1))

→ u(10)

      Function definition for a is being overwritten.
      Compiling function u with type Integer -> Float
   0.013634905645531645253                                                (85)

→ u(1003)


      >> System error:
      Invocation history stack overflow.
   protected-symbol-warn called with (NIL)
```

u(1003) n'est pas calculé. Ceci est du à une protection de la mémoire, du programme. L'exécution est empêchée.

Sommes de séries

1) Soit la suite $(u_n)_n$ avec $u_n = \frac{1}{n}$. On considère la série de terme général u_n.

a) Calculer, pour différentes valeurs de n $v_n = \sum u_k$, $w_n = \frac{v_n}{\ln(n)}$, $x_n = v_n - \ln(n)$

```
→ )read exo4.input

   v : Integer -> Float
```

```
                                                          Type: Void v n ==
    if n = 1 then
           1
       else
           1/n +v (n-1)

                                                  Type: Void w : Integer -> Float

    Compiled code for w has been cleared.
                                                          Type: Void w n ==
      v n/log(n)

    1 old definition(s) deleted for function or rule w
                                                  Type: Void x : Integer -> Float

                                                          Type: Void x n ==
      v n - log(n)

                                                               Type: Void

→ v 7

    Compiling function v with type Integer -> Float
    Compiling function v as a recurrence relation.
  2.5928571428571428572                                                  (92)
                                                             Type: Float

→ w 3

    Compiling function w with type Integer -> Float
  1.6687719154825352216                                                  (93)
                                                             Type: Float

→ x 13

    Compiling function x with type Integer -> Float
```

0.6151843976722183977 (94)

Type: Float

b) Déterminer la valeur de n_0 telle que pour $n > n_0$ $v_n > A$ pour A = 5,6,8 et 10. Faire un programme qui affiche les résultats intermédiaires, et un autre qui ne donne que n. Comparer les temps de calcul.

Dans cet exercice, on met plusieurs fonctions les unes après les autres. Le système est exactement le même. On écrit simplement une fonction après l'autre.

```
→ )read exo5.input

   calcul : Integer -> Float


                                                    Type: Void calcul A ==
     n := 1
     while v n < A repeat
       n := n+1
     n


                                          Type: Void calculaff : Integer -> Float


                                                 Type: Void calculaff A ==
     n := 1
     while v n < A repeat
      output(n)
      n := n+1
     n

→ )set messages time on

→ calcul 5

      Compiling function calcul with type Integer -> Float
   83.0                                                                  (99)
                                       Time: 0.01 (EV) + 0.06 (GC) = 0.07 sec

→ calcul 6

   227.0                                                                 (100)
                                            Time: 0.01 (EV) = 0.01 sec
```

```
→ calcul 8
```

1674.0 (101)

```
Type: Float                                    Time: 0.11 (EV) + 0.11
(GC) = 0.22 sec
```

Même calcul avec affichage intermédiaire en console: 0.35 secondes. en mode X : 2 secondes

```
→ calcul 10
```

12367.0 (102)

```
                                  Time: 0.74 (EV) + 0.85 (GC) = 1.59 sec
```

Même calcul avec affichage intermédiaire en console : 2.5 sec. En mode X : plantage de TeXmacs

2) Ordre des termes d'une série. Soit la série S1= $\sum \frac{(-1)^n}{n}$

a) Vérifier la semi-convergence

Cet exercice est impossible à réaliser. Comme vu précédemment, les fonctions sommes ne supportent pas les bornes infinies.

b) On construit une nouvelle série S2. Chaque terme positif est suivi de 2 termes négatifs. En calculer les sommes partielles d"ordre 30, 90, puis 300 et les comparer.

```
→ )read exo6.input
```

```
S2 : Integer -> Fraction Integer


Type: Void
Time: 0 sec

S2 n ==
   if n = 1 then
         1
      else
         if even?(n) = true then
               -(1/(2*n-2)+1/(2*n))+S2 (n-1)
            else
               1/n+S2 (n-1)


Type: Void
Time: 0 sec
```

```
→ S2 30
```

```
   Compiling function S2 with type Integer -> Fraction Integer
```

$$\frac{225175759291}{665454160800} \tag{105}$$

```
                                                    Time: 0.01 (OT) = 0.01 sec
```

```
→ S2 90 :: Float
```

$$0.34381124364859670606 \tag{106}$$

```
                                                    Time: 0.06 (GC) = 0.06 sec
```

```
→ S2 300 :: Float
```

$$0.34574164582781233234 \tag{107}$$

```
                                                    Time: 0.02 (EV) = 0.02 sec
```

```
→ )set messages time off
```

Arithmétique

Faire afficher directement, ou dans un tableau, les n premiers nombres premiers, puis faire afficher uniquement le n-ième nombre premier.

```
→ )read exo7.input
```

```
   prime : Integer -> Integer
```

```
                                                    Type: Void prime n ==
      i := 0
      x := 0
      while not (i = n) repeat
        while not prime?(x) repeat
           x := x+1
        print(x)
        i := i+1
        x := x+1
      x := x-1
      x
```

```
→ prime 7

      Function definition for x is being overwritten.

      Compiling function prime with type Integer -> Integer

      2

      3

      5

      7

      11

      13

      17

   17                                                                (110)

                                                          Type: PositiveInteger
```

Algèbre

Construire une matrice carrée n*n tel que chaque élément de la matrice soit égal à la somme du numéro de ligne et de colonne. Indiquer si la matrice est diagonalisable, en donner les valeurs propres et les vecteurs propres.

```
→ )read exo8.input

   mat : Integer -> Matrix Integer


                                                    Type: Void mat n ==

      matri := createGenericMatrix(n)

      for i in 1..(n::INT) repeat

         for j in 1..(n::INT) repeat

            matri(i,j) := i+j-1

      matri

→ mat 5

      Your expression cannot be fully compiled because it contains an

         integer expression (for n ) whose sign cannot be determined (in

         general) and so must be specified by you. Perhaps you can try

         substituting something like

                                    (n :: PI)

                                       or

                                   (n :: NNI)
```

```
   into your expression for n .
AXIOM will attempt to step through and interpret the code.
Compiling function mat with type Integer -> Matrix Integer
Compiling function G1888 with type Integer -> Boolean
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix} \tag{113}$$

Type: Matrix Integer

→ `determinant( mat 6 )`

$$0 \tag{114}$$

Détermination des vecteurs, et des valeurs propres.

→ `eigenvectors( mat 6 )`

$$\left[\left[eigval = 0, eigmult = 4, eigvec = \left[\begin{bmatrix} 1 \\ -2 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ -3 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ -4 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ -5 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right] \right], \right.$$

$$\left. \left[eigval = (\%O \mid \%O^2 - 36\%O - 105), eigmult = 1, eigvec = \right. \right.$$

$$\left. \left. \left[\begin{bmatrix} \frac{\%O - 29}{22} \\ \frac{2\%O - 47}{55} \\ \frac{3\%O - 43}{110} \\ \frac{\%O + 4}{55} \\ \frac{\%O + 59}{110} \\ 1 \end{bmatrix} \right] \right] \right] \tag{115}$$

```
Type: List Record(eigval: Union(Fraction Polynomial
Integer,SuchThat(Symbol,Polynomial Integer)),eigmult:
NonNegativeInteger,eigvec: List Matrix Fraction Polynomial Integer)
```

Procédures

1) Générer un triangle de pascal

Le programme génère des erreurs inhabituelles de « conversion ». La compilation effective du programme est impossible. On utilisera une version non compilée.

```
→ l :=12
```

$$12 \tag{116}$$

```
→ matripas := createGenericMatrix((l::PI));matripas(1,1) := 1;matripas(2,1) :=
  1;matripas(2,2) := 1;for i in 3..(l::PI) repeat {matripas(i,i) :=
  1,matripas(i,1) := 1,for j in 2..((i::PI)-1) repeat matripas(i,j) :=
  matripas(( i - 1 ),j) + matripas(( i - 1),( j - 1 ))};matripas
```

$$\begin{bmatrix}
1 & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} & x_{1,6} & x_{1,7} & x_{1,8} & x_{1,9} & x_{1,10} & x_{1,11} & x_{1,12} \\
1 & 1 & x_{2,3} & x_{2,4} & x_{2,5} & x_{2,6} & x_{2,7} & x_{2,8} & x_{2,9} & x_{2,10} & x_{2,11} & x_{2,12} \\
1 & 2 & 1 & x_{3,4} & x_{3,5} & x_{3,6} & x_{3,7} & x_{3,8} & x_{3,9} & x_{3,10} & x_{3,11} & x_{3,12} \\
1 & 3 & 3 & 1 & x_{4,5} & x_{4,6} & x_{4,7} & x_{4,8} & x_{4,9} & x_{4,10} & x_{4,11} & x_{4,12} \\
1 & 4 & 6 & 4 & 1 & x_{5,6} & x_{5,7} & x_{5,8} & x_{5,9} & x_{5,10} & x_{5,11} & x_{5,12} \\
1 & 5 & 10 & 10 & 5 & 1 & x_{6,7} & x_{6,8} & x_{6,9} & x_{6,10} & x_{6,11} & x_{6,12} \\
1 & 6 & 15 & 20 & 15 & 6 & 1 & x_{7,8} & x_{7,9} & x_{7,10} & x_{7,11} & x_{7,12} \\
1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 & x_{8,9} & x_{8,10} & x_{8,11} & x_{8,12} \\
1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1 & x_{9,10} & x_{9,11} & x_{9,12} \\
1 & 9 & 36 & 84 & 126 & 126 & 84 & 36 & 9 & 1 & x_{10,11} & x_{10,12} \\
1 & 10 & 45 & 120 & 210 & 252 & 210 & 120 & 45 & 10 & 1 & x_{11,12} \\
1 & 11 & 55 & 165 & 330 & 462 & 462 & 330 & 165 & 55 & 11 & 1
\end{bmatrix} \tag{118}$$

```
                                        Type: Matrix Polynomial Integer
```

```
→ )clear properties x
```

```
   Compiled code for x has been cleared.
```

Les 2 derniers exemples ne sont réalisables ni avec Axiom (pas de visualisation graphique) ni avec GNUplot (impossibilité de définir la courbe)

Calculs du programme MT26

Les exercices ici réalisés sont des exercices « en trop », réalisés afin de pouvoir visualiser d'autres exemples, en vue d'une éventuelle modification du polycopié. Ces calculs étaient demandés dans un polycopié précédent. On laissera ici l'affichage des types, afin de découvrir l'influence des paramètres sur la modification des types.

```
→ integrate((log(x)/((x-1)*sqrt(x*(1-x)))),x=0..1)
```

$$potentialPole \tag{119}$$

Quand l'intégration est impossible, avec une erreur en potentialPole, il suffit de rajouter l'option « noPole » en dernier argument de la fonction d'intégration.

```
→ integrate((log(x)/((x-1)*sqrt(x*(1-x)))),x=0..1, "noPole")
```

$$2\pi \tag{120}$$

```
→ integrate(log(x)/sqrt(x),x=0..1/2)
```

$$potentialPole \tag{121}$$

```
                                        Type: Union(pole: potentialPole,...)
```

```
→ integrate(log(x)/sqrt(x),x=0..1/2, "noPole")
```

$$\frac{-\log(4)-4}{\sqrt{2}} \tag{122}$$

```
                        Type: Union(f1: OrderedCompletion Expression Integer,...)
```

```
→ g(x) == integrate(t^(x-1)*exp(-t),t= 1..%plusInfinity, "noPole")
                                        Type: Union(pole: potentialPole,...)
   1 old definition(s) deleted for function or rule g
                                                                  Type: Void
→ g(3)
   Compiling function g with type PositiveInteger -> Union(f1:
      OrderedCompletion Expression Integer,f2: List OrderedCompletion
      Expression Integer,fail: failed,pole: potentialPole)
```

$$\frac{5}{e} \tag{124}$$

```
                        Type: Union(f1: OrderedCompletion Expression Integer,...)
→ g(7.5)
   Compiling function g with type Float -> Union(Result,"failed")
"failed"                                                                (125)
                                                  Type: Union("failed",...)
→ g(8)
```

$$\frac{13700}{e} \tag{126}$$

```
                        Type: Union(f1: OrderedCompletion Expression Integer,...)
```

Fin de la résolution des exercices

Conclusion

Grâce à ces exercices, et à leur résolution, on peut se faire une idée précise quand à l'application et à l'utilisation des logiciels Axiom et GNUplot en TD de MT26.

Étudions le cas de GNUplot : Celui-ci, est certes très simple d'emploi et est fourni avec une documentation détaillée, mais se révèle malheureusement bien trop incomplet, et ne satisfera que 10% environ des questions du TP d'MT26. Son champ d'application est beaucoup trop restreint par rapport à la demande faite, et de ce fait, est inutilisable en tant que logiciel de calcul formel. Néanmoins, une utilisation en TP de physique est tout à fait envisageable, pour le tracé de nuage de points, de comparaison avec une courbe, etc.. L'avantage qu'il présente face à un logiciel comme Excel, est, mis à part sa gratuité, sa capacité à faire évoluer le graphique en temps réel (rotations, zooms, etc..).

Dans le cas d'Axiom, il est nécessaire de faire la part des choses : La mentalité, le système d'aide, et les ressources disponibles creusent certes un écart important avec Maple, mais cette différence s'avère surtout déroutante au début, pendant la prise en main du logiciel: Les fonctions et la rapidité d'exécution de celui-ci en font un concurrent sérieux pour Maple. Néanmoins, Axiom n'est pas prêt, pour l'instant, à devenir un logiciel d'enseignement pédagogique. Cette opinion sera sans doute à revoir d'ici quelques semestres mais les résolutions d'exercices se font à l'heure actuelle bien souvent par tâtonnements: La spécificité des types empêche une résolution logique de certains exercices (en particulier pour la décomposition en éléments simples, vrai casse-tête, ou la compilation impossible de certains calculs sans raisons apparentes) Les efforts fait pour améliorer l'interface, développer l'intuitivité des fonctions, et ajouter les éléments manquants vont pourtant permettre d'ici peu une utilisation pratique de ce logiciel, même à un niveau éducatif. Dès le portage du ViewPort Manager, Axiom sera une alternative possible à Maple. Dans tous les cas, l'utilisation d'Axiom se fera obligatoirement à travers une interface TeXmacs, le mode console étant beaucoup trop déroutant pour un non-initié à Linux.

Cette étude complète nous aura donc permis de nous former à une certaine méthodologie de travail, la documentation étant à l'heure actuelle assez faible, en ce qui concerne axiom. De plus nous nous sommes habitués au fonctionnement de TeXmacs, à l'utilisation de structures bien définies, à un environnement totalement nouveau. Cette UV a été particulièrement formatrice en ce qui concerne la pratique de l'informatique avec des logiciels libres.

Annexes

Annexe 1 : Licence BSD d'axiom

Annexe 2: Fiche explicative du fonctionnement de GNUplot

Annexe 3: Manuel d'utilisation d'Axiom

Annexe 4: Bibliographie