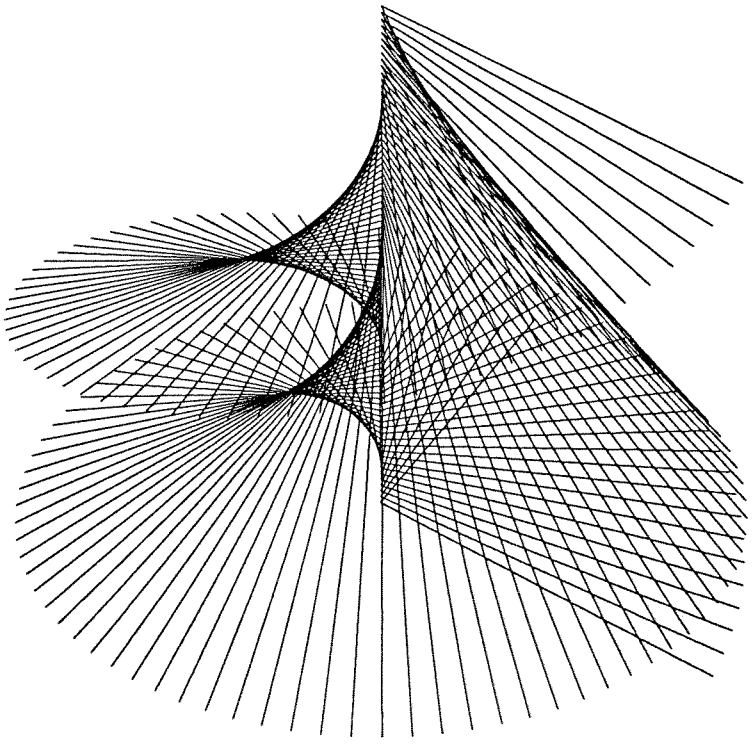
UNIVERSITE
LOUIS PASTEUR
STRASBOURG

1996

FAIRE DES MATHEMATIQUES AVEC UN LOGICIEL DE CALCUL FORMEL

par

Groupe Calcul formel de Mulhouse



I.R.E.M.
10, rue du Général Zimmer
67084 STRASBOURG CEDEX
Tél. : 88 41 64 40 (Bibliothèque)
Fax : 88 41 64 49

FAIRE DES MATHEMATIQUES

AVEC UN LOGICIEL DE CALCUL FORMEL

par

Groupe Calcul formel de Mulhouse

Groupe Calcul formel de Mulhouse

Les membres du groupe sont :

Lionel Darie (Collège de Soultz)
Marc Gilg (Université de Haute Alsace)
Jean Lefort (LEGT Colmar)
Abdenacer Makhlouf (Université de Haute Alsace),
Etienne Meyer (lycée Th. Deck Guebwiller)
Jean Perrin (lycée Louis Armand Mulhouse)
Michel Goze (Université de Haute Alsace)

Pour toute correspondance s'adresser à :

Abdenacer MAKHLOUF
Université de Haute Alsace
Laboratoire de Mathématiques
4, rue des frères Lumière
68093 Mulhouse cedex
émail : N.Makhlouf@univ-mulhouse.fr

Table des matières

Partie IV :

Des applications du calcul formel dans la recherche mathématique 139

Préface

Cette brochure est une contribution modeste de quelques enseignants de mathématiques qui, ayant compris très vite l'importance de l'outil informatique, se sont regroupés dans un groupe IREM pour intégrer cet outil dans leur enseignement par le biais des logiciels de calcul formel.
On présente ici nos périgrinations dans les différents domaines liés à la question.

Il faut tout d'abord apprendre à utiliser un tel logiciel. Naviguant entre Derive, Mathematica et Maple (les deux derniers étant très proches), on a testé les avantages et les inconvénients de chacun d'eux pour opter finalement pour Mathematica dont on présente l'essentiel d'un bon apprentissage. L'objectif étant de faire des mathématiques, on explore ainsi au moyen du calcul formel quelques notions de mathématiques.

Si l'ordinateur peut jouer le rôle d'une boite noire, il est important pour mieux l'exploiter de connaître les algorithmes utilisés. On en donne certains exemples.

Avec l'avénement des logiciels de calcul formel, l'ordinateur prend aussi de plus en plus de place dans le domaine de la recherche mathématique et c'est pourquoi on en présente quelques exemples dans le domaine de l'algèbre.

Groupe Calcul Formel Mulhouse

Le calcul formel

1.Introduction

L'aventure de l'ordinateur a commencé avec l'ère des calculatrices numériques de plus en plus sophistiquées pour répondre à la forte demande dans le domaine de la gestion où l'objectif était d'effectuer des calculs numériques fastidieux. Afin d'être plus performant, on a mis au point des formules et des méthodes algébriques pour simplifier la tâche de la machine et aller plus loin dans les calculs. On est passé du calcul ne comportant que des opérations arithmétiques, à un calcul numérique précédé d'un calcul algébrique; c'est ce qu'on a appelé le calcul scientifique (par exemple le calcul approché de Sin(Pi/25), Log(231)). La nouvelle étape consistait à automatiser le calcul algébrique, c'est ce qu'on a appelé le calcul formel ou symbolique (computer algebra).

Aujourd'hui, il est possible, avec un simple micro-ordinateur, de manipuler des symboles, des expressions algébriques ou analytiques : polynômes, fonctions trigonométriques, groupes, idéaux, tenseurs, systèmes différentiels et à partir de ces bases d'effectuer des résolutions d'équations, des intégrations formelles, des calculs de limites, des graphiques ....

Bien-sûr, des langages évolués tels Pascal ou C permettent déjà de travailler sur des chaînes de caractères donc de manipuler des symboles, mais c'est toujours à l'utilisateur de construire les procédures qui s'y appliquent : finalement, cela revient à concevoir son propre logiciel de calcul symbolique, et c'est une énorme tâche.

Les logiciels de calcul formel, eux, sont directement conçus pour représenter et manipuler des données non numériques de façon naturelle. Ils possèdent une bibliothèque de fonctions (commandes) pour effectuer les opérations algébriques de base. De plus, ils sont pourvus d'un langage de programmation permettant de définir des commandes élaborées ou ses propres procédures pour enrichir l'ensemble des commandes.

Une propriété commune aux logiciels récents de calcul formel est leur interactivité avec la possibilité d'écrire de petits programmes, faire des calcul, utiliser librement les données et les résultats et prendre des notes directement dans un même fichier.

1.1 Historique

Les premiers systèmes de calcul formel sont nés dès 1953 avec un logiciel de Kahrimanian et Nolan. LISP est apparu dans les années 60. Mais ces précurseurs étaient surtout des logiciels conçus par et pour des chercheurs. Ils fonctionnaient sur de gros systèmes, leur accès était peu commode et leur documentation médiocre: leur diffusion resta donc limitée à quelques laboratoires.

Les années soixante-dix ont pourtant vu éclore de grands systèmes comme Reduce (implanté sur 2000 sites dans le monde : 1/3 au Japon, 1/3 au USA, 1/3 en Europe), Macsyma ou Scratchpad, disponibles sur grosses unités centrales, dont l'ambition se bornait au calcul sans limitation de précision, au calcul sur des rationnels, au calcul algébrique sur des polynômes et à un peu de calcul analytique (dérivation, intégration). Gourmands en mémoire et en temps de calcul, il était impossible de les implanter sur les ordinateurs personnels d'alors.

A la fin des années 1980, l'explosion de la technologie va entraîner une diminution du prix des mémoires et rendre enfin possible l'installation de logiciels de calcul formel sur des micros. De nos jours, la montée en puissance régulière des ordinateurs de bureau, tant en mémoire qu'en rapidité (processeurs Pentium d'Intel, ou PowerPC de Motorolla), couplée à une forte baisse des tarifs, ouvre de nouvelles perspectives aux scientifiques de tous bords : le calcul formel devient accessible à tous, d'autant que les concepteurs de logiciels s'en donnent à coeur joie. Citons quelques logiciels de calcul symbolique disponibles aujourd'hui sur micro : Maple, Mathematica, Derive, MathPlus (version internationale de Theorist), Mathcad, Scientific WorkPlace... A ces moteurs de calcul peuvent aussi se greffer d'autres logiciels, d'autres modules qui en étendent encore les capacités.

■ 1.2 Mathematica : un système de calcul formel doté d'un langage de programmation évolué.

Mathematica est l'un de ces logiciels de calcul formel, encore jeune, puisque conçu en 1988 par une équipe américaine dirigée par Stephen Wolfram (Wolfram Research Inc.) basée à Champaign (Illinois) près de Chicago, et composée de plus de 120 personnes. Des collaborateurs dans le monde entier apportent également leur contribution à l'amélioration constante du logiciel.

Chaque logiciel de calcul formel possède ses qualités propres (et ses défauts, bien entendu) : ainsi Derive ne nécessite-t-il que peu de mémoire vive pour fonctionner et peut donc être implanté sur une grosse calculatrice (TI 92); Maple possède un éditeur de formules intégré... ; quant à Mathematica, c'est la jeunesse de ce produit qui fait en partie sa force, puisqu'il intègre un langage utilisant les idées les plus récentes en matière de programmation. Il permet en effet de s'exprimer presque comme en mathématiques et mêle avec bonheur les différents styles de programmation : style impératif de Pascal ou Fortran, fonctionnel de Lisp et déclaratif de Prolog. Cette richesse lui confère un pouvoir d'expression inégalé jusqu'à présent et autorise enfin un rapprochement entre l'activité mathématique et la programmation informatique. L'utilisateur n'est plus prisonnier d'un style. Il n'est plus contraint de programmer "à la Pascal" ; il choisit la manière de programmer la plus naturelle pour lui, ou celle qu'il juge la plus efficace.

Mais Mathematica reste un logiciel de calcul formel, permettant de combiner calcul symbolique et numérique, ainsi que graphiques 2D et 3D, dessins (importés de logiciels de dessins), texte (un traitement de texte est également fourni) et sons (générés par des fonctions Mathematica ou importés). La version qui suivra la version 2.2 sera vraiment révolutionnaire, puisqu'elle comportera un éditeur de formules scientifiques actives, fonctionnant à la fois pour les entrées et les sorties. L'utilisateur pourra alors définir ses propres symboles (associés, par exemple, à des opérations), qu'il conservera dans une palette. Cette version, déjà au stade de version bêta et testée par quelques utilisateurs chevronnés, va vraiment dans le sens d'une facilité d'utilisation optimale et préfigure ce que seront les grands systèmes de calcul formel de l'avenir. Plus que jamais, faire des mathématiques avec un ordinateur devient un acte naturel.

▒ 2. Réflexions sur le calcul formel

■ 2.1 Un domaine en pleine expansion

Dès le début, l'arrivée sur le marché de ces logiciels a suscité des critiques. On a d'abord tenté de faire au calcul formel le procès de son applicabilité, en grande partie parce que ni la technologie, ni la science des algorithmes de calcul symbolique n'étaient assez avancées pour fournir un logiciel réellement performant, et capable d'aider les chercheurs et ingénieurs dans leur travail journalier. Mais les travaux de ces vingt dernières années ont quasiment résolu le problème de la factorisation des polynômes, la thèse de M. Bronstein en 1987 a marqué une étape décisive dans le calcul des primitives d'une fonction, les années 80-90 ont vu la création d'algorithmes efficaces pour résoudre certains systèmes différentiels...La recherche en algorithmique symbolique a donc progressé à pas de géant, conjointement à la technologie. De nos jours, le calcul formel connaît une véritable explosion quant à ses domaines d'application : calculs en astronomie (calcul de l'orbite de la Lune, par exemple), chimie (prédiction du comportement réactif d'une molécule...), robotique, traitement du signal et codage, théorie du contrôle, biologie mathématique, théorie des groupes...On ne se pose plus aujourd'hui la question de l'applicabilité du calcul formel. En fait, il devient à la mode, et il faut dès maintenant s'y intéresser, car il sera de plus en plus difficile de prendre le train en marche, à l'image de ceux qui n'ont pas cru au développement de l'informatique et qui se retrouvent complètement dépassés aujourd'hui.

L'arrivée sur le marché de calculatrices utilisant des logiciels de calcul formel de plus en plus puissants demande même à ce que l'on revoie sérieusement notre enseignement des mathématiques, mais aussi son évaluation, puisque tous les élèves auront bientôt accès à ce moyen de calcul.

■ 2.2 Vers un rapprochement entre l'informatique et les mathématiques

L'informatique n'a jamais été perçue comme un outil naturel pour les mathématiques : par exemple, la nécessité d'introduire des cours d'informatique en classes préparatoires ne s'est imposée qu'en 1988, alors que ces classes sont un premier échelon pour la préparation au métier d'ingénieur, pourtant grand consommateur d'informatique et de mathématiques appliquées. C'est d'abord le langage informatique lui-même qui fait obstacle. L'une des grandes richesses des mathématiques est de pouvoir traduire différemment un problème suivant le concept dans lequel il est exprimé : on peut résoudre certains problèmes en utilisant l'arithmétique, mais aussi en les exprimant en termes de fonctions, ou encore de façon géométrique, ou algébrique...Cette souplesse d'expression est au contraire assez rare lorsque l'on construit un algorithme. En effet, il est toujours destiné à être concrétisé par un programme écrit dans un langage particulier, le plus souvent procédural. Dans ce cas, le problème sera le plus souvent exprimé en termes de suites récurrentes uniquement. En résolvant un problème par le biais de l'informatique, on risquait donc de tomber dans une certaine forme d'excès, visant à tout exprimer de façon itérative. Abandonnée des mathématiciens, qui ont bien du mal à en tirer pleinement parti, l'informatique a poursuivi seule son chemin.

Mais S. Wolfram, concepteur de Mathematica , nous met en garde contre cette séparation de l'informatique et des mathématiques : "One of the biggest mistakes of mathematics research (...) has been to let computer science get away". Il était donc nécessaire que naissent de nouveaux langages permettant au mathématicien de se détacher des contraintes imposées par la machine. Ces langages doivent se rapprocher au maximum du langage-même des mathématiques et autoriser une grande liberté d'expression pour ne pas scléroser l'imagination mais plutôt stimuler la créativité.C'est à l'heure actuelle, ce qu'offre déja Mathematica à ses utilisateurs. Nous allons donc assister à un rapprochement inévitable entre l'activité mathématique et la programmation informatique. Cela va imposer au mathématiciens de nouvelles compétences, tout en lui ouvrant de nouvelles voies d'investigations. Il sera, en fait, de plus en plus facile à un mathématicien de se mettre à l'informatique. On verra même bientôt l'informatique enseigné en mathématique, par le biais des logiciels de calcul formel...

■ 2.3 Pour une autre approche des mathématiques

L'un des grands avantages du calcul formel réside dans la possibilité de marier harmonieusement formules, graphiques, programmes, texte etc. Du coup, l'ordinateur est transformé en un véritable laboratoire d'expérimentation mathématique. L'étudiant, comme le chercheur, peut tester rapidemment ses intuitions, que ce soit en calcul exact ou approché. Tout cela ouvre de nouvelles voies pédagogiques, à l'heure où c'est à l'élève de construire au maximum son propre savoir.
Mais cette approche très pragmatique soulève de nouvelles difficultés, de nouvelles questions et mettra en exergue de nouvelles compétences. Il ne suffira plus d'être un virtuose du calcul. Il faudra savoir raisonner de façon plus algorithmique. Les grands problèmes d'examen de jadis seront alors beaucoup moins directifs et pourront faire appel à des calculs de plus en plus compliqués sans que l'utilisateur ait réellement à en souffrir. Il faudra aussi acquérir une certaine maîtrise de la programmation, tout en sachant contourner les faiblesses d'un logiciel qui ne saurait être parfait.

D'expérience, la pratique d'un logiciel de calcul formel ne dispense pas d'apprendre les mathématiques. Bien au contraire, pour utiliser de façon satisfaisante ce type d'outils, il faut connaître beaucoup de mathématiques ! Car si le logiciel fournit un résultat, il faut pouvoir en apprécier la crédibilité. et s'il refuse d'en produire un, il faut savoir pourquoi ; bien souvent, il faut aider le logiciel à choisir une méthode plus douce, plus rusée, moins directe...

D'autre part, le calcul formel permet de remettre au goût du jour de vieilles méthodes analytiques approchées, du temps où l'ordinateur était inconnu et où le calcul numérique était fui comme la peste, puisqu'impraticable. L'hégémonie du tout numérique touche à sa fin. Vont surgir çà et là des algorithmes "mixtes", mélangeant calcul symbolique et numérique pour plus d'efficacité. Mais qui peut dire si le calcul formel ne sera pas employé, dans un premier temps pour optimiser des méthodes encore essentiellement numériques, au lieu d'ouvrir la voie à des méthodes vraiment nouvelles ?

Quoiqu'il en soit, l'enseignement des mathématiques ne peut ignorer longtemps les nouvelles technologies et particulièrement les logiciels de calcul formel. Il faut évidemment adapter l'enseignement des mathématiques aux nouveaux outils dont vont disposer énormement d'élèves. Les avantages sont multiples. Les logiciels de calcul formel permettent de faire vite ce qu'on faisait lentement et de faire fiablement des choses qu'on ne pouvait faire qu'approximativement. On peut ainsi tenter des approches expérimentales. Les logiciels de calcul formel apportent beaucoup de réponses mais aussi génèrent beaucoup de questions. Les aspects numérique, algébrique et graphique intégrés dans un même logiciels permettent d'aborder un problème suivant différentes approches. Son interactivité aide à apprendre avec de la rétroaction : on essaye, on observe le résultat, on change les conditions intiales et observe les phénomènes réguliers.
L'apprentissage et la compréhension des mathématiques sera augmenté car la résolution d'un problème mathématique sur ordinateur nécessite la réécriture du problème. Cette tâche est toujours bénéfique puisqu'elle contribue efficacement à la compréhension profonde du problème.
La libération de l'élève des taches fastidieuses et des détours pénibles dus aux erreurs de calcul, lui permet de se concentrer sur la méthode et de rester ainsi plus près du raisonnement et du fil conducteur d'une démonstration.

Un autre avantage non sans intérêt est la possibilité d'élargir le champs des situations à étudier et de résoudre plus tôt des problèmes plus difficiles en évitant les blocages dûs à des difficultés techniques de calcul ou de résolution. De même que le contrôle d'un travail effectué sur ordinateur est beaucoup plus facile, il est plus transparent que sur une feuille, on peut suivre le cheminement et les différentes étapes que l'élève a suivi.

Afin de tirer profit de ces outils modernes, il est nécessaire que le professeur de mathématique s'y intèresse et l'intègre dans son enseignement en orientant les élèves vers un usage bénéfique. Ainsi la machine deviendra un partenaire de l'enseignant.

2.4 Un souhait pédagogique

En dehors de Derive pour lequel un effort de francisation a été fait (de façon incomplète), le professeur de lycée qui souhaite utiliser les logiciels de calcul formel se trouve à peu près dans la situation d'enseigner des mathématiques en utilisant un manuel de niveau licence écrit en anglais! Il parait donc indispensable de traduire au moins l'aide en ligne et si possible les commandes les plus courantes, tous les élèves n'étant pas anglicistes, surtout en Alsace. Il serait intéressant de disposer de modules adaptés à un niveau donné (2de, 1er, terminale?). Mais il rete le problème des "calculatrices" et des micro-ordinateurs. Ceci renvoie à l'équipement des salles de classe.
Tout ceci n'est qu'un souhait pédagogique. Un groupe IREM ne saurait à lui seul le faire aboutir, mais c'est son rôle que de le poser.

Partie I

INITIATION A MATHEMATICA

UN SURVOL DE MATHEMATICA

Le travail avec Mathematica se passe sous forme d'un dialogue interactif avec la machine.
In[numéro] est l'entrée qui consiste en un certain nombre d'instructions qu'on exécute en cliquant sur "l'étoile Mathematica" de la barre ou en appuyant sur la touche "Inser". Out[numéro] est la sortie qui évalue ou répond à la commande In[numéro], le résultat est stocké dans le tableau de valeur Out[...] au même indice. Par conséquent, on peut faire appel à cette valeur par la suite.
Mathematica dispose de 843 fonctions prédéfinies (version 2.2) qui permettent de résoudre des problèmes mathématiques dans les domaines numériques, symbolique et graphique. Mathematica est également un puissant langage de programmation permettant de créer ses propres fonctions, afin d'étendre les possibilités du logiciel selon les besoins spécifiques de l'utilisateur.

▩ Calcul numérique

Mathematica peut tout d'abord être utilisé comme une calculatrice scientifique disposant d'une excellente précision.

In[1]:=

11^400

Out[1]=

```
360640140275244358409809197227413432506014479571709426634264

855802219449716560461019123316080053385524493558697601426666

348963783176434920813875219211230512634500370892532904161700

906327187745943645060833414105691908434176506748938899800237

761799765945636927481116982426177429788773405571192647231900

458452364740988607922988137992026490228243977555744237380725

562015980241320969682738069176758878433380111117923236017840

01
```

In[2]:=

150! (*Factorielle*)(*Ceci est un commentaire*)

Out[2]=

```
57133839564458545904789328652610540031895535786011264182548375833179829124845398393126574488675311145377107878746854204162666250198684504466355949195922066574942592095735778929325357290444962472405416790722118445437122269675520000000000000000000000000000000
```

La fonction N[Pi,1000] calcule la valeur de Pi avec 1000 chiffres significatifs. expression //N donne une approximation numérique de l'expression, la précision par défaut est de 16 chiffres significatifs.

In[3]:=

N[Pi,300]

Out[3]=

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127 4
```

In[4]:=

E//N (*approximation numérique de e*)

Out[4]=

```
2.71828
```

Remarque : la machine affiche que 6 chiffres significatifs mais elle en a en mémoire 16.

In[5]:=

N[E,100]

Out[5]=

```
2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664 27
```

In[6]:=

Sin[%] (*calcul du sinus de la valeur précédente*)

Out[6]=

```
0.41078129050290869547600949201836059188830697039341534530457165880613518243765499587597861904543559 4
```

In[7]:=

Sin[Pi]

Out[7]=

0

Le calcul formel manipule des expressions sans perte de précision mais on peut avoir des approximations de ces expressions par

Expression //N ou N[Expression,précision]

La fonction Precision[nombre] retourne le nombre de chiffres significatifs et la fonction Accuracy[nombre] retourne la place de la virgule (le nombre de chiffres à droite de la virgule).

In[8]:=

a=0.234; b=N[E,21];

In[9]:=

{Precision[a], Precision[b], Accuracy[a], Accuracy[b]}

Out[9]=

{16, 21, 17, 21}

Le nombre de chiffres significatifs pour a n'est pas 3, la réponse correspond au nombre de chiffres significatifs avec lesquels Mathematica effectue les calculs.

▩ Aide

?Symbole : retourne les renseignements sur ce symbole tels que sa valeur ou sa syntaxe.

In[10]:=

?Precision

```
Precision[x] gives the number of digits of precision in the
   number x.
```

In[11]:=

?N

```
N[expr] gives the numerical value of expr. N[expr, n] does
   computations to n-digit precision.
```

In[12]:=

?a

```
Global`a

a = 0.234
```

In[13]:=

?Names

```
Names["string"] gives a list of the names of symbols which
   match the string. Names["string",
   SpellingCorrection->True] includes names which match afte
   spelling correction.
```

avec ?? on a en plus les attributs de la fonction

In[14]:=

??Names

```
Names["string"] gives a list of the names of symbols which
   match the string. Names["string",
   SpellingCorrection->True] includes names which match afte
   spelling correction.

Attributes[Names] = {Protected}

Options[Names] =
  {IgnoreCase -> False, SpellingCorrection -> False}
```

Names["chaine de caractères"] retourne la liste des fonctions qui commencent par ces caractères.*

In[15]:=

Names["V*"]

Out[15]=

```
{ValueForm, ValueList, ValueQ, ValueTable, Variables,

 VectorQ, VerifyConvergence, VerifySolutions, VerticalForm,

 ViewCenter, ViewPoint, ViewVertical}
```

In[16]:=

▩ Calcul symbolique

■ Affectations

Il ya des affectations instantanées(=) et des affectations différées (:=).
A un symbole on peut affecter une valeur ou un autre symbole.

In[16]:=

a=25

Out[16]=

```
25
```

In[17]:=

x=a; x

Out[17]=

```
25
```

In[18]:=

y:=a

In[19]:=

{x,y}

Out[19]=

```
{25, 25}
```

In[20]:=

a=11

Out[20]=

```
11
```

In[21]:=

{x,y}

Out[21]=

```
{25, 11}
```

Remarquez que la valeur de x n'a pas changé.
s=expression, le symbole s est lié à la forme évaluée de expression.
s:=expression, à chaque fois que s est utilisé on va évaluer expression.
La valeur d'un symbole peut être effacée à l'aide de la fonction Clear[symbole] ou par symbole=.

In[22]:=

Clear[x,y]

In[23]:=

?x

```
Global`x
```

■ Nombre complexes

On peut faire des opérations avec les nombres complexes:
x+Iy est le complexe x+iy.
Re[z] partie réelle de z.
Im[z] partie imaginaire.
Conjugate[z] le conjugué de z.
Abs[z] module de z.
Arg[z] argument de z.

In[24]:=

Sqrt[-1]

Out[24]=

```
I
```

■ Polynômes et fractions rationnelles

Les opérations de base sur les polynômes:
Expand[P] développe P.
FactorTerms[P] factorise les coefficients.
Factor[P] factorise P.
Collect[P,x] écrit P sous forme d'un polynôme en x.
Collect[P,{x,y}] écrit P sous forme d'un polynôme en x et y.
PolynomialQuotient[P,Q,x] retourne le quotient de la division de P par Q.
PolynomialRemainder[P,Q,x] retourne le reste de la division de P par Q.

In[25]:=

P=2(x-1)^2+6(x^5-x^4+x^3-x)

Out[25]=

```
                2                3    4    5
2 (-1 + x)  + 6 (-x + x  - x  + x )
```

In[26]:=

Expand[P]

Out[26]=

```
                    2       3       4       5
2 - 10 x + 2 x  + 6 x  - 6 x  + 6 x
```

In[27]:=

FactorTerms[P]

Out[27]=

```
                  2       3       4       5
2 (1 - 5 x + x  + 3 x  - 3 x  + 3 x )
```

In[28]:=

Factor[P]

Out[28]=

```
                              2       4
2 (1 - x) (1 - 4 x - 3 x  - 3 x )
```

In[29]:=

{"Quotient ",PolynomialQuotient[P,x-1,x]," ", "reste ",PolynomialRemainder[P,x-1,x]}

Out[29]=

```
                                 2       4
{Quotient , -2 + 8 x + 6 x  + 6 x ,  , reste , 0}
```

Les fonctions de base pour les fractions rationnelles:

Cancel[F] *Simplifie la fraction rationnelle F.*
Apart[F] *décompose F.*
Together[F] *réduit au même dénominateur.*
ExpandAll[F] *développe le numérateur et le dénominateur.*
ExpandDenominator[F] *développe le dénominateur.*
ExpandNumerator[F] *développe le numérateur.*
Numerator[F] *retourne le numérateur de F.*
Denominator[F] *retourne le dénominateur de F.*

In[30]:=

F=1/(x^2-1)(x^2+x+1)

Out[30]=

```
          2
1 + x + x
---------
        2
 -1 + x
```

Attention! Si c'est 1/produit que vous voulez il faut les mettre entre parenthèses.
FullForm[F] décrit les opérations.

In[31]:=

FullForm[F]

Out[31]//FullForm=

```
Times[Power[Plus[-1, Power[x, 2]], -1],

 Plus[1, x, Power[x, 2]]]
```

In[32]:=

F=1/((x^2-1)(x^2+x+1))

Out[32]=

```
                1
-----------------------------
        2              2
(-1 + x ) (1 + x + x )
```

In[33]:=

ExpandDenominator[F]

Out[33]=

```
          1
--------------------
            3    4
-1 - x + x  + x
```

In[34]:=

Apart[F]

Out[34]=

```
     1             1            -1 + x
---------- - --------- + --------------
6 (-1 + x)   2 (1 + x)                2
                          3 (1 + x + x )
```

■ Résolution des équations algébriques

Une équation s'écrit A==B.
Soit eq=A==B une equation où l'inconnue est x, pour la résoudre on a :
Solve[eq,x] *résout l'équation par rapport à x.*
Roots[eq,x] *fait la même chose qu'avant mais la présentation de la solution est différente.*
NSolve[eq,x] *résout numériquement l'équation.*
NRoots[eq,x] *idem*
Pour un système d'équations eq1,eq2,...,eqN où les inconnues sont x1,x2,...,xM, la résolution se
fait par: Solve[{eq1,...,eqN},{x1,...,xM}]. Idem avec Roots et NSolve.

In[35]:=

Solve[t^2-t+1==0,t]

Out[35]=

```
          1 - I Sqrt[3]          1 + I Sqrt[3]
{{t -> -------------}, {t -> -------------}}
                2                      2
```

In[36]:=

eq1=3x^3-14x^2+17x-6==0

Out[36]=

```
                    2      3
-6 + 17 x - 14 x  + 3 x  == 0
```

In[37]:=

Roots[eq1,x]

Out[37]=

```
                                  2
x == 3 || x == 1 || x == -
                                  3
```

Pour obtenir les solutions sous forme de liste on écrit :

In[38]:=

List@@Roots[eq1,x]

Out[38]=

```
                          2
{x == 3, x == 1, x == -}
                          3
```

ou comme une liste de régles

In[39]:=

List@@Roots[eq1,x]/.Equal->Rule

Out[39]=

```
                          2
{x -> 3, x -> 1, x -> -}
                          3
```

Si on veut un ensemble avec les valeurs solutions seulement :

In[40]:=

x/.(List/@Roots[eq1,x]/.{Or->List, Equal->Rule})

Out[40]=

```
       2
{3, 1, -}
       3
```

Ce qui peut avantageusement remplacé par :

In[41]:=

x/.{ToRules[Roots[eq1,x]]}

Out[41]=

```
       2
{3, 1, -}
       3
```

In[42]:=

eq2=x^3+4x^2+1==0

Out[42]=

```
          2    3
1 + 4 x  + x  == 0
```

In[43]:=

Roots[eq2,x]

Out[43]=

```
                                 1/3
          4                16 2
x == -(-) + ------------------------- +
          3                        1/3
               3 (-155 + 3 Sqrt[849])

                            1/3
     (-155 + 3 Sqrt[849])
--   ---------------------- ||
                  1/3
              3 2

                       1/3
           4      8 2    (1 - I Sqrt[3])
x == -(-) - ------------------------- -
           3                       1/3
                3 (-155 + 3 Sqrt[849])

                                              1/3
     (1 + I Sqrt[3]) (-155 + 3 Sqrt[849])
--   ------------------------------------ ||
                         1/3
                    6 2

                       1/3
           4      8 2    (1 + I Sqrt[3])
x == -(-) - ------------------------- -
           3                       1/3
                3 (-155 + 3 Sqrt[849])

                                              1/3
     (1 - I Sqrt[3]) (-155 + 3 Sqrt[849])
--   ------------------------------------
                         1/3
                    6 2
```

In[44]:=

Solve[{x(x+y)==0,x y==-1},{x,y}]

Out[44]=

```
{{x -> -1, y -> 1}, {x -> 1, y -> -1}}
```

In[45]:=

S={x^3-y^2==x*y,x+x y==1}

Out[45]=

```
   3    2
{x  - y  == x y, x + x y == 1}
```

In[46]:=

Solve[S,{x,y}]

Out[46]=

```
                                      2    4
{ToRules[Roots[y == 1 - 2 x + x  + x , y,

                                 2    3    5
   Using -> Roots[2 x - 2 x  + x  + x  == 1, x]]]}
```

La réponse veut dire qu'il n'a pas trouvé. Demander alors une résolution numérique.

In[47]:=

NSolve[S,{x,y}]

Out[47]=

```
{{y -> -1.35247 - 0.572239 I, x -> -0.780336 + 1.26687 I},

 {y -> -1.35247 + 0.572239 I, x -> -0.780336 - 1.26687 I},

 {y -> -0.328284 - 1.0839 I, x -> 0.413098 + 0.666588 I},

 {y -> -0.328284 + 1.0839 I, x -> 0.413098 - 0.666588 I},

 {y -> 0.361515, x -> 0.734476}}
```

Vecteurs et matrices

Array[V,n] construit un vecteur de n éléments V[1],...,V[n].
Array[M,{m,n}] construit une matrice M avec m lignes et n colonnes d'éléments M[i,j].
Table[V[i],{i,1,n}] génére un vecteur de n composantes, les V[i] étant une fonction de i.

Det[Matrice carrée]	*calcule le déterminant.*
Inverse[Matrice]	*calcule la matrice inverse.*
LinearSolve[M,V]	*résout le système linéaire MX=V.*

In[49]:=

A={{1,a,a^2},{a,1,a},{a^2,a,1}}

Out[49]=

```
          2                  2
{{1, a, a }, {a, 1, a}, {a , a, 1}}
```

In[50]:=

MatrixForm[A]

Out[50]//MatrixForm=

```
              2
1     a     a

a     1     a

 2
a     a     1
```

On peut également utiliser la notation PostFix et écrire A//MatrixForm.

In[51]:=

d=Det[A]

Out[51]=

```
          2     4
1 - 2 a  + a
```

In[52]:=

Factor[d]

Out[52]=

```
           2         2
(-1 + a)  (1 + a)
```

On génére la liste des 20 premiers nombres premiers.

In[53]:=

Table[Prime[i],{i,1,20}]

Out[53]=

```
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53,

 59, 61, 67, 71}
```

■ Fonctions

Il existe des fonctions prédéfinies, par exemple Sin, Exp, Timing, Names..
Toutes ces fonctions commencent par une majuscule et les arguments sont entre crochets.
On peut définir de nouvelles fonctions de la façon suivante: f[x_]:=Expression dépendant de x.
C'est une affectation différée qui se lit à f[quelque chose] est associée l'Expression.

In[54]:=

f[x_]:=x^2;

In[55]:=

f[7]

Out[55]=

```
49
```

In[56]:=

g[t_]:=Sqrt[1+t^2]*Log[t];

In[57]:=

g[4.5]

Out[57]=

```
6.93345
```

Les opérations de base sur les fonctions sont:
D[f[x],x] dérivation de f par rapport à x.
Integrate[f[x],x] primitive de f.
Integrate[f[x],{x,a,b}] intégrale définie de f entre a et b.
NIntegrate[f[x],{x,a,b}] valeur numérique de l'intégrale définie de f entre a et b.
Series[f[x],{x,x0,n}] développement limité à l'ordre n de f en x0.
Limit[f[x],x->x0] limite de f quand x tend vers x0.
Remarque: l'infini se dit Infinity.

In[58]:=

f[x_]:=Exp[Sin[x]]

In[59]:=

D[f[x],x]

Out[59]=

```
   Sin[x]
E       Cos[x]
```

In[60]:=

dl=Series[f[x],{x,0,10}]

Out[60]=

```
                2    4    5    6     7       8     9          10
               x    x    x    x     x    31 x     x     2951 x
1 + x + x + -- - -- - -- - --- + -- + ----- + ---- - -------- +
               2    8    15   240   90   5760    5670  3628800

      11
   O[x]
```

In[61]:=

Integrate[Log[x],x]

Out[61]=

```
-x + x Log[x]
```

In[62]:=

NIntegrate[f[x],{x,1,5}]

Out[62]=

```
5.55725
```

In[63]:=

Limit[Tan[x],x->Pi/2]

Out[63]=

```
-Infinity
```

In[64]:=

Limit[(Sin[Tan[x]]-Tan[Sin[x]])/x^7, x->0]

Out[64]=

```
  1
--(--)
  30
```

Remarque : Si Mathematica échoue, il faut lancer le package "Calculus'Limit'". Là il échoue rarement.

▩ Graphique

Les fonctions de dessin de base:

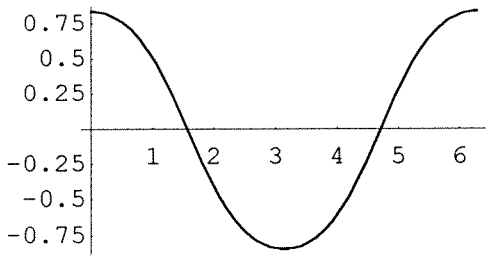
Plot[f[x],{x,xmin,xmax},options]
dessine le graphe de f(x) pour x allant de xmin à xmax.

ListPlot[list,options]
dessine la liste des points de list.

Plot3D[f[x,y],{x,xmin,xmax},{y,ymin,ymax}]
représentation de la surface z=f(x,y)

In[65]:=

Plot[Sin[Cos[x]],{x,0,2Pi}]

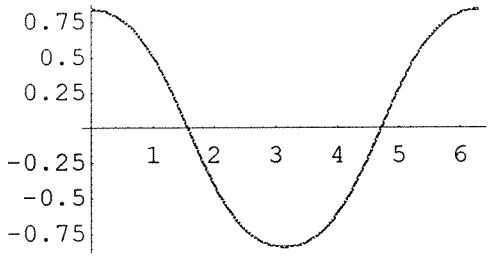


Out[65]=

-Graphics-

In[66]:=

Plot[Sin[Cos[x]],{x,0,2Pi},PlotStyle->RGBColor[1,0,0]]

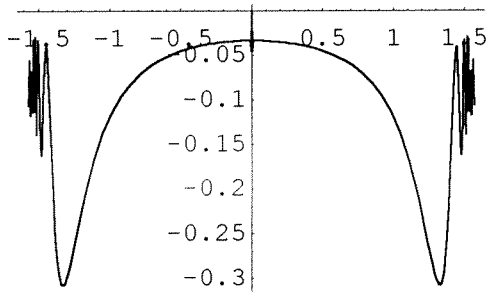


Out[66]=

-Graphics-

In[67]:=

Plot[(Sin[Tan[x]]-Tan[Sin[x]])/x^7,{x,-Pi/2,Pi/2}]

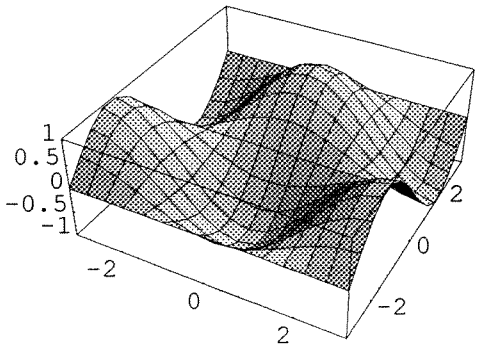


Out[67]=

-Graphics-

In[68]:=

Plot3D[Cos[x] Sin[y],{x,-Pi,Pi},{y,-Pi,Pi}]



Out[68]=

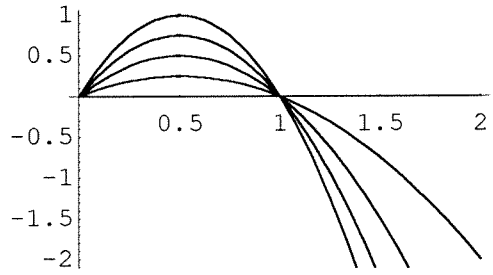
```
-SurfaceGraphics-
```

In[69]:=

f[c_,x_]:=c x(1-x);

In[70]:=

Plot[Evaluate[Table[f[c,x],{c,0,4}],{x,0,2}]]



Out[70]=

```
-Graphics-
```

LES FONCTIONS

En plus des 843 fonctions et commandes prédéfinies de base de Mathematica , on peut définir des fonctions qui viendront s'ajouter au langage et se comporteront comme les fonctions internes. Une fonction est un "programme" qui est exécuté à chaque appel de son nom avec éventuellement des valeurs fixées de ses arguments. Ces fonctions peuvent représenter des fonctions numériques et symboliques ainsi que des procédures et des opérateurs.

Mathematica s'enrichit de jour en jour par de nouveaux "packages" (ensemble de fonctions) spécifiques à un sujet donné. L'utilisateur peut créer ses propres "packages". Ils consistent en un programme avec une extension ".m" dont l'usage nécessite un chargement à partir de n'importe quel fichier de mathematica.

On donne dans la suite les différentes manières de définir une fonction avec des illustrations sur divers exemples.

▩ Définitions

f[x_]:=Expresion,
définit une fonction qui peut faire intervenir l'argument x. C'est en fait un programme qui peut être appelé avec l'argument x.
g[x_,y_,etc_]:=Expression,
est une fonction qui dépend de plusieurs arguments.

■ Fonctions d'une variable

In[1]:=

```
f1[t_]:=t^3+Sqrt[t]-Sin[4t]
```

In[2]:=

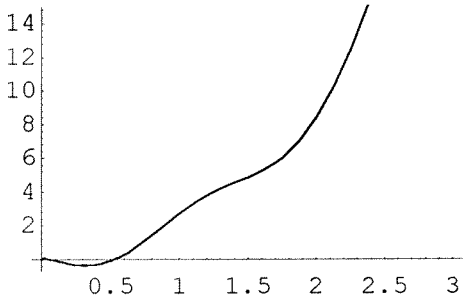
```
{f1[0],f1[a]}
```

Out[2]=

```
                    3
{0, Sqrt[a] + a  - Sin[4 a]}
```

In[3]:=

```
Plot[f1[x],{x,0,3}]
```



Out[3]=

```
-Graphics-
```

■ Fonction de plusieurs variables

In[4]:=

```
f2[x_,y_]:= Cos[x]+ Sin[x y]-1
```

In[5]:=

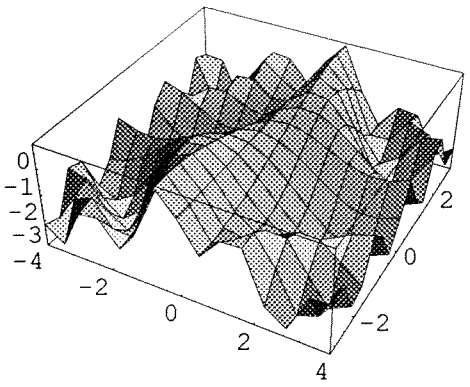
{f2[2,3],f2[4,a]}

Out[5]=

```
{-1 + Cos[2] + Sin[6], -1 + Cos[4] + Sin[4 a]}
```

In[6]:=

Plot3D[f2[x,y],{x,-4,4},{y,-3,3}]



Out[6]=

```
-SurfaceGraphics-
```

■ Fonctions avec points critiques

□ Exemple 1

In[7]:=

f3[0]:=0;

In[8]:=

f3[x_]:=x Log[x]

In[9]:=

{f3[3],N[f3[3]],f3[0]}

Out[9]=

```
{3 Log[3], 3.29584, 0}
```

In[10]:=

?f3

```
Global`f3

f3[0] := 0

f3[x_] := x*Log[x]
```

Ici la fonction est définie en 0 et en un x quelconque.

□ Exemple 2

In[11]:=

g[a]:=n a^(n-1);
g[x_]:=(x^n-a^n)/(x-a)

In[13]:=

g[x]

Out[13]=

$$\frac{-a^n + x^n}{-a + x}$$

In[14]:=

g[3]

Out[14]=

$$\frac{3^n - a^n}{3 - a}$$

In[15]:=

g[a]

Out[15]=

$$a^{-1+n}$$

Si on définit la même fonction sans la définir en a, on rencontre alors une indétermination en a, voir l'exemple suivant:

□ **Exemple 3**

In[16]:=

h[x_]:=(x^n-a^n)/(x-a)

In[17]:=

h[a]

```
                                    1
Power::infy: Infinite expression - encountered.
                                    0

Infinity::indet:
   Indeterminate expression 0 ComplexInfinity encountered.
```

Out[17]=

```
Indeterminate
```

□ **Exemple 4**

In[21]:=

k[0]:=Infinity;
k[x_]:=1/x

In[23]:=

{k[2],k[0]}

Out[23]=

$$\left\{\frac{1}{2}, \text{Infinity}\right\}$$

■ Définitions restreintes

□ **Exemple 1**

On définit une fonction uniquement sur les entiers par exemple.

In[24]:=

g1[x_Integer]:=x^2+1

In[25]:=

g1[2]

Out[25]=

```
5
```

In[26]:=

g1[Sqrt[2]]

Out[26]=

```
g1[Sqrt[2]]
```

In[27]:=

?g1

```
Global`g1

g1[x_Integer] := x^2 + 1
```

On peut augmenter une définition:

In[28]:=

g1[Sqrt[2]]:=bb;
g1[x_Rational]:=Ln[x]

In[30]:=

?g1

```
Global`g1

g1[2^(1/2)] := bb

g1[x_Integer] := x^2 + 1

g1[x_Rational] := Ln[x]
```

In[31]:=

{g1[1/2],g1[4],g1[Sqrt[2]],g1[Sin[1]]}

Out[31]=

```
      1
{Ln[-], 17, bb, g1[Sin[1]]}
      2
```

☐ **Exemple2**

La fonction est définie séparement suivant que x est supérieur ou inférieur à 1/2.

In[32]:=

g2[x_]:=Sqrt[2x+1]/;x>=-0.5;
g2[x_]:=Cos[2x+1]/;x<-0.5;

In[34]:=

{g2[5],g2[-10]}

Out[34]=

```
{Sqrt[11], Cos[19]}
```

In[35]:=

g2[aa]

Out[35]=

```
g2[aa]
```

Ici aa n'est pas précisé, alors on ne renvoit aucune réponse.

▩ Fonctions récursives

Une fonction peut être définie à l'aide d'un algorithme récursif. Si une telle définition est rencontrée, le calcul se fait en commençant par le plus bas niveau.
Notons que la limite maximale du niveau d'imbrication est 256 par défaut. Cette profondeur est fixée par la variable $RecursionLimit que l'on peut changer:
$RecursionLimit = nouvelle_valeur.

In[36]:=

```
c[n_,n_]:=1;c[n_,0]:=1;
c[n_,p_]:=c[n-1,p-1]+c[n-1,p]
```

In[38]:=

```
{c[7,6],c[11,4]}
```

Out[38]=

```
{7, 330}
```

▩ Fonctions à mémoire

La fonction garde en mémoire les valeurs calculées.
Pour cela on écrit: f[x_]:=f[x]=expression.

In[39]:=

```
c[n_,n_]:=1;c[n_,0]:=1;
c[n_,p_]:=c[n-1,p-1]+c[n-1,p]
```

In[41]:=

```
Timing[c[10,5]]
```

Out[41]=

```
{0.17 Second, 252}
```

In[42]:=

```
cc[n_,n_]:=1;cc[n_,0]:=1;
cc[n_,p_]:=cc[n,p]=cc[n-1,p-1]+cc[n-1,p]
```

In[44]:=

```
Timing[cc[7,6]]
```

Out[44]=

```
{0. Second, 7}
```

In[45]:=

```
Timing[cc[7,6]]
```

Out[45]=

```
{0. Second, 7}
```

In[46]:=

```
?cc
Global`cc

cc[2, 1] = 2

cc[3, 2] = 3

cc[4, 3] = 4

cc[5, 4] = 5

cc[6, 5] = 6

cc[7, 6] = 7

cc[n_, n_] := 1

cc[n_, 0] := 1

cc[n_, p_] := cc[n, p] = cc[n - 1, p - 1] + cc[n - 1, p]
```

Remarquons qu'on gagne de la vitesse mais au dépend de l'utilisation de plus de mémoire.
Il est conseillé de l'utiliser quand la fonction est définie de façon récursive.

■ Affectation instantannée et affectation différée

On donne un exemple où l'affectation instantannée est plus intéressante que l'affectation différée.

In[47]:=

```
R[x_]=Integrate[Sin[u]Exp[u+1],{u,0,x}];
P[x_]:=Integrate[Sin[u]Exp[u+1],{u,0,x}]
```

In[49]:=

```
Timing[R[5]]
```

Out[49]=

$$\{0.\ \text{Second},\ \frac{E}{2}-\frac{E^6\,(\cos[5]-\sin[5])}{2}\}$$

In[50]:=

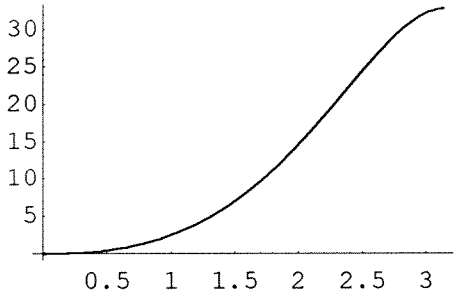
```
Timing[P[5]]
```

Out[50]=

$$\{0.83\ \text{Second},\ \frac{E}{2}-\frac{E^6\,(\cos[5]-\sin[5])}{2}\}$$

In[51]:=

Timing[Plot[R[x],{x,0,Pi}]]

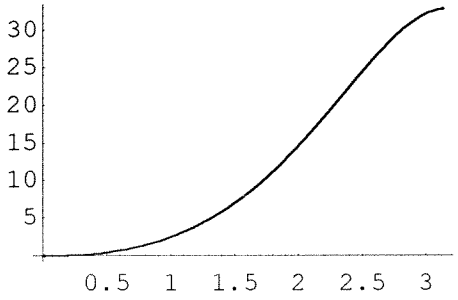


Out[51]=

```
{0.44 Second, -Graphics-}
```

In[52]:=

Timing[Plot[P[x],{x,0,Pi}]]



Out[52]=

```
{19.33 Second, -Graphics-}
```

In[53]:=

?R

```
Global`R

R[x_] = E/2 - (E^(1 + x)*(Cos[x] - Sin[x]))/2
```

In[54]:=

?P

```
Global`P

P[x_] := Integrate[Sin[u]*Exp[u + 1], {u, 0, x}]
```

Remarquons que dans le cas de l'affectation instantannée c'est la valeur de l'intégrale qui est associée à la fonction alors que dans le cas de l'affectation différée, on recalcule à chaque fois l'intégrale d'où la lenteur.

DEFINITION d'une FONCTION : APPLICATION à la DEFINITION de la FONCTION DERIVEE

Mathematica permet bien sûr de calculer le nombre dérivé d'une fonction donnée en un point donné.

In[1]:=

```
F[x_]:=x^2+a;
F'[3]
```

Out[2]=

```
6
```

Le point en lequel on demande le nombre dérivé n'est pas nécessairement donné par une valeur numérique.

In[3]:=

```
{F'[a],F'[x],F'[2x]}
```

Out[3]=

```
{2 a, 2 x, 4 x}
```

F' apparait donc bien comme la fonction dérivée de la fonction F.

▩ Cherchons à définir une fonction nommée G, égale à F'.

Voici une première idée qui semble bien naturelle :

In[4]:=

```
G[x_]:=F'[x]
```

Examinons notre tentative :

In[5]:=

```
{G[3],G[a],G[x],G[2x]}
```

Out[5]=

```
{6, 2 a, 2 x, 4 x}
```

Tout se passe comme prévu. La fonction G est effectivement définie comme égale à F'. DERIVE1 n'est autre que la fonction (post-fixée) ' , c'est-à-dire : Derivative[1].

▩ Cherchons à définir une fonction "DERIVE1"

qui donne comme résultat la fonction dérivée de la fonction donnée en argument.
Nous souhaitons obtenir, par exemple, à partir de G1: = DERIVE1[F], le résultat suivant :

{G1[3],G1[a],G1[x],G1[2x]} est évalué en {6 , 2 a , 2 x , 4 x}

In[6]:=

```
DERIVE1[F_]:=F'
G1:=DERIVE1[F]
{G1[3],G1[a],G1[x],G1[2x]}
```

Out[8]=

```
{6, 2 a, 2 x, 4 x}
```

Tout se passe bien comme prévu. La fonction DERIVE1 est effectivement définie comme la fonction qui dérive une fonction. Mais il arrive que les fonctions que l'on manipule n'ont pas de nom, qu'elles se présentent comme le résultat d'un calcul.

▩ Cherchons à définir une fonction "DERIVE2"

qui donne comme résultat une fonction de variable x, dérivée d'une expression expr par rapport à la variable x.
Nous souhaitons obtenir, par exemple, à partir de G2:= DERIVE2[x^2+a , x] , le résultat suivant :
{G2[3],G2[a],G2[x],G2[2x]} est évalué en {6 , 2 a , 2 x , 4 x}
La fonction prédéfinie ' de Mathematica (en fait F' est définie par Derivative[1][F]) n'est plus utilisable puisqu'on n'a pas de fonction en argument.
Il va falloir utiliser la fonction prédéfinie D de Mathematica qui à l'appel de D[F[x],x], donne F'[x].

In[9]:=

```
DERIVE2[expr_,x_]:=D[expr,x]
G2:=DERIVE2[x^2+a,x]
{G2[3],G2[a],G2[x],G2[2x]}
```

Out[11]=

```
{(2 x)[3], (2 x)[a], (2 x)[x], (2 x)[2 x]}
```

Cette fois-ci, c'est raté, et c'est normal puisque D[expr,x] n'est pas une fonction mais une expression.
Définissons donc, bien que cela ne réponde pas tout à fait à la question posée, G2[x] par D[expr,x] :

In[12]:=

```
G2[x_]:=D[x^2+a,x]
SetDelayed::write: Tag Times in (2 x)[x_] is Protected.
```

Out[12]=

```
$Failed
```

D'accord ... Les ennuis commencent. Reprenons :

In[13]:=

```
Clear[G2];
G2[x_]:=D[x^2+a,x]
{G2[3],G2[a],G2[x],G2[2x]}
General::ivar: 3 is not a valid variable.
General::ivar: 2 x is not a valid variable.
```

Out[15]=

$$\{D[9 + a, 3], 1 + 2\,a, 2\,x, D[a + 4\,x^2, 2\,x]\}$$

Voilà qui mérite réflexion !
G2[3] et G2[2x] n'ont pas été calculé car 3 et 2x ne sont pas des variables.
G2[truc] signifie : dérivée de truc^2+a par rapport à truc.

Il est bien compréhensible également que G2[x] donne le bon résultat.
Nous voilà en présence d'une fonction G2 qui ne donne le bon résultat que lorsqu'on l'applique à x.
Ce n'est pas fameux et c'est même dangereux car on risque de ne pas s'en apercevoir rapidement, l'application à x étant tellement courante.
Le problème vient du fait que dans la définition de G2[x], l'évaluation de D[expr,x] est diférée.
Essayons avec l'évaluation immédiate :

In[16]:=

```
Clear[G2];
G2[x_]=D[x^2+a,x];
{G2[3],G2[a],G2[x],G2[2x]}
```

Out[18]=

```
{6, 2 a, 2 x, 4 x}
```

C'est clair : pour une fois l'évaluation immédiate n'est pas seulement conseillée pour gagner du temps, mais elle est nécessaire.
(et il faudra en conséquence faire attention aux affectations éventuelles déjà existentes des lettres qui interviennent)
La solution que nous avons pour l'instant nécessite de connaître le nom de la variable par rapport à laquelle on dérive. Or celle-ci pourait bien être le résultat d'une évaluation, et dans ce cas ne pourait pas servir de variable pour définir G2.
D'où cet autre essai :

In[19]:=

```
Clear[G2];
G2[t_]:=D[x^2+a,x]/.x->t;
{G2[3],G2[a],G2[x],G2[2x]}
```

Out[21]=

```
{6, 2 a, 2 x, 4 x}
```

Cela se passe comme prévu. L'application de la règle x->t s'effectue après l'évaluation de D[expr,x].
Essayons sur un exemple un peu plus complexe.

In[22]:=

```
H[t_]:=D[x^2+a,First[{x,a}]]/.First[{x,a}]->t;
{H[3],H[a],H[x],H[2x]}
```

Out[23]=

```
{6, 2 a, 2 x, 4 x}
```

Mais nous n'avons pas bien réalisé l'objectif poursuivi : définir G2 comme fonction et DERIVE2 comme une fonction ayant pour arguments une expression expr et une variable x, et fournissant comme résultat une fonction : la dérivée de expr par rapport à x, comme fonction de x.
Pour y parvenir, il nous faut d'abord regarder de plus près ce qu'est une fonction dans MATHEMATICA et quel est le lien entre fonction et expression. Ce qu'il faut parvenir à faire, c'est extraire une fonction d'une expression.
MATHEMATICA possède une fonction qui réalise cela : c'est Function. Illustrons son utilisation.

In[24]:=

Function[x,x^2+a][t]

Out[24]=

```
       2
a + t
```

Nous venons d'appliquer la fonction x donne x^2+a à la valeur t de la variable.

Appelons FF la fonction de variable x, qui à x associe x^2+a

In[25]:=

FF:=Function[x,x^2+a]

Examinons si cela répond à notre attente :

In[26]:=

{FF[3],FF[a],FF[x],FF[2x]}

Out[26]=

```
                  2        2          2
{9 + a, a + a , a + x , a + 4 x }
```

Nous pouvons maintenant définir notre fonction DERIVE2

In[27]:=

Clear[DERIVE2]
DERIVE2[expr_,x_]:=Function[x,D[expr,x]]
Clear[G2]
G2:=DERIVE2[x^2+a,x]
{G2[3],G2[a],G2[x],G2[2x]}

```
General::ivar: 3 is not a valid variable.
General::ivar: 2 x is not a valid variable.
```

Out[31]=

```
                                                2
{D[9 + a, 3], 1 + 2 a, 2 x, D[a + 4 x , 2 x]}
```

Encore raté ... et pour cause ...

In[32]:=

Clear[DERIVE2]
DERIVE2[expr_,x_]:=Derivative[1][Function[x,expr]]
Clear[G2]
G2:=DERIVE2[x^2+a,x]
{G2[3],G2[a],G2[x],G2[2x]}

Out[36]=

```
{6, 2 a, 2 x, 4 x}
```

Cette fois-ci, l'objectif est atteint.

Essayons sur un autre exemple.

In[37]:=

H:=DERIVE2[a^2+x^3+a*x^2,a]
{H[1],H[x],H[a],H[2a],H[2x]}

Out[38]=

```
       2          2          2          2          2
{2 + x , 2 x + x , 2 a + x , 4 a + x , 4 x + x }
```

En guise d'exercice, on pourra essayer de prévoir les résultats aux séquences suivantes :

In[39]:=

```
a=3;f[x_]:=x^2;g[x_]:=D[f[x],x];
{g[x],g[t],g[a],g[3]}
f[x_]:=x^3
{g[x],g[t],g[a],g[3]}
General::ivar: 3 is not a valid variable.
General::ivar: 3 is not a valid variable.
```

Out[40]=

```
{2 x, 2 t, D[9, 3], D[9, 3]}
General::ivar: 3 is not a valid variable.
General::ivar: 3 is not a valid variable.
```

Out[42]=

```
      2      2
{3 x , 3 t , D[27, 3], D[27, 3]}
```

In[43]:=

```
a=3;f[x_]:=x^2;g[x_]:=Derivative[1][f][x];
{g[x],g[t],g[a],g[3]}
f[x_]:=x^3
{g[x],g[t],g[a],g[3]}
```

Out[44]=

```
{2 x, 2 t, 6, 6}
```

Out[46]=

```
      2      2
{3 x , 3 t , 27, 27}
```

In[47]:=

```
a=3;f[x_]:=x^2;g[x_]=D[f[x],x];
{g[x],g[t],g[a],g[3]}
f[x_]:=x^3
{g[x],g[t],g[a],g[3]}
f[x]:=x^2
{f[x],f[t],f[a],f[3]}
```

Out[48]=

```
{2 x, 2 t, 6, 6}
```

Out[50]=

```
{2 x, 2 t, 6, 6}
```

Out[52]=

```
    2    3
{x , t , 27, 27}
```

In[53]:=

```
a=3;f[x_]=x^2;g[x_]=D[f[x],x];
{g[x],g[t],g[a],g[3]}
f[x_]=x^3
{g[x],g[t],g[a],g[3]}
```

Out[54]=

```
{2 x, 2 t, 6, 6}
```

Out[55]=

```
 3
x
```

Out[56]=

```
{2 x, 2 t, 6, 6}
```

Les Listes

Introduction

Une liste dans Mathematica est un type de données très important, c'est l'objet le plus utilisé. Une liste peut décrire un ensemble, un tableau, un vecteur ou une matrice. Nous allons développer ces différents aspects.

Notions de base

Déclaration d'une liste

Une liste commence par une accolade '{' et se termine par une autre accolade '}', voici comment est déclarée la liste vide (une liste est vide si elle ne contient aucun élément) :

In[1]:=

```
vide = {};
```

Pour séparer les éléments contenus dans une liste, on utilise la virgule ',' :

In[2]:=

```
alphabet = {"a","b","c","d","e","f","g","h"};
lettre = {"a"};
```

Les éléments d'une liste peuvent être n'importe quel objet de mathematica, y compris des listes. Les éléments peuvent être d'un type différent :

In[4]:=

```
listes = { {1,2}, {3,4,5} };
mixte = {1, a, {l1, l2}, 5};
```

Il existe des fonctions de mathematica qui permettent de créer des listes.

Créér une liste avec la fonction Table[]

La fonction 'table[]' permet de créér une liste à partir d'une fonction du numéro de la case (voir Accés à un élément de la liste). Voici un exemple :

In[6]:=

```
Table[i, {i,6}]
```

Out[6]=

```
{1, 2, 3, 4, 5, 6}
```

Dans cet exemple nous pouvons étudier la syntaxe de la fonction Table : elle possède deux paramètres : le premier est la formule, le deuxième est une liste qui représente la variable et sa dernière valeur.

Si nous voulons créer une liste dont les éléments sont les carrés de 1 à 8 nous avons comme formule n^2 avec n prenant comme dernière valeur 8 :

In[7]:=

```
Table[n^2,{n,8}]
```

Out[7]=

```
{1, 4, 9, 16, 25, 36, 49, 64}
```

Il se peut que nous voulons uniquement les carrés de 5 à 8. Pour cela nous allons modifier le paramètre '{n,8}' en rajoutant la valeur minimale de n soit 5 :

In[8]:=

```
carré = Table[n^2,{n,5,8}]
```

Out[8]=

```
{25, 36, 49, 64}
```

Attention! Le numéro de la case ne correspond plus à la valeur de n pour le résultat de cette case :

In[9]:=

```
carré[[ 2]]
```

Out[9]=

```
36
```

Si nous inversons l'ordre de 5 et 8 dans la liste '{n,5,8}', le résultat sera la liste vide :

In[10]:=

```
Table[n^2,{n,8,5}]
```

Out[10]=

```
{}
```

Dans les exemples précédents, la différence entre deux valeurs successives du paramètre est 1, nous pouvons changer cela :

In[11]:=

```
Table[h,{h,0,2,0.5}]
```

Out[11]=

```
{0, 0.5, 1., 1.5, 2.}
```

Ici le pas est de 0.5.

*Nous pouvons utiliser "Table" pour créér une liste de polynomes définie par la suite Un = x^n + 3 * n, pour n de 0 à 5 :*

In[12]:=

```
Table[ X^j+3 j,{j,0,5}]
```

Out[12]=

```
                    2        3         4         5
{1, 3 + X, 6 + X , 9 + X , 12 + X , 15 + X }
```

La syntaxe de "table" est donc la suite :

Table[relation,{indice, début, fin, pas}]

□ Créer une liste avec la commande Array[]

La commande Array[] permet de créer une liste dont les éléments sont des variables indexées. L'instruction Array[elem, n] crée la liste { elem[1], elem[2], .. ,elem[n]}. Attention elem n'est pas le nom de la liste mais des éléments!

In[13]:=

```
Array[a,5]
```

Out[13]=

```
{a[1], a[2], a[3], a[4], a[5]}
```

Pour créer une liste de 2 listes de 3 éléments :

In[14]:=

```
Array[b,{2,3}]
```

Out[14]=

```
{{b[1, 1], b[1, 2], b[1, 3]}, {b[2, 1], b[2, 2], b[2, 3]}}
```

In[15]:=

■ Opérations de base

□ Accès à un élément de la liste :

Une liste peut être représentée comme un ensemble de cases numérotées de gauche à droite. L'élément de gauche est dans la case numéro 1. Pour accéder à l'élément placé en n on utilise la syntaxe : ' liste [[n]]'. Voici un exemple :

In[15]:=

```
alphabet[[ 4]]
```

Out[15]=

```
d
```

Le résultat a le même type que l'objet qui se trouve dans la case. Si c'est une liste la réponse sera de type liste:

In[16]:=

```
mixte[[ 3]]
```

Out[16]=

```
{l1, l2}
```

Si le numéro spécifié est supérieur au nombre d'éléments, il se produit une erreur :

In[17]:=

```
lettre[[ 2]]
Part::partw: Part 2 of {a} does not exist.
```

Out[17]=

```
{a}[[2]]
```

Pour obtenir le j ème élément à partir de la fin nous utilisons l'instruction liste[[-j]]:

In[18]:=

```
alphabet[[-6]]
```

Out[18]=

```
c
```

Si la liste est composée d'une autre liste, nous pouvons accéder aux éléments de celle-ci en utilisant l'opérateur [[ ]] deux fois :

In[19]:=

```
listes[[1]][[2]]
```

Out[19]=

```
2
```

Il est aussi possible d'utiliser la commande liste[[i, j]] :

In[20]:=

```
listes[[1,2]]
```

Out[20]=

```
2
```

Si nous avons plusieurs listes imbriquées, nous utilisons la commande [[i,j,k,...]] avec le nombre d'indice inférieur ou égal au nombre de listes imbriquées.

Pour créer une liste avec les éléments i et j de la liste enum nous utilisons la commande enum[[{i, j}]] :

In[21]:=

```
alphabet[[ {1,3,5} ]]
```

Out[21]=

```
{a, c, e}
```

□ **Accés à une liste avec la commande Part[]**

La commande Part[liste, i] a le même effet que liste[[i]] :

In[22]:=

```
Part[listes,1]
```

Out[22]=

```
{1, 2}
```

Si nous avons plusieurs listes imbriquées nous utilisons Part[liste, i, j,..] :

In[23]:=

```
Part[listes,1,2]
```

Out[23]=

```
2
```

La commande Part[] est intèressante si nous voulons créer une liste à partir d'élément d'une autre liste. Pour créer une liste avec les éléments i et j de la liste "enum" il faut utiliser la commande Part[enum, {i, j}] :

In[24]:=

```
Part[alphabet, {1,3,5}]
```

Out[24]=

```
{a, c, e}
```

□ **Changer un élément d'une liste**

Nous pouvons aussi utiliser la commande [[ ]] dans une affectation :

In[25]:=

```
mixte
mixte[[ 2]] = b;
mixte
```

Out[25]=

```
{1, a, {l1, l2}, 5}
```

Out[27]=

```
{1, b, {l1, l2}, 5}
```

□ **Longeur d'une liste**

La longeur d'une liste est le nombre d'éléments de la liste. La fonction 'Length[ ]' permet de connaître la longueur d'une liste :

In[28]:=

```
Length[ vide]
```

Out[28]=

```
0
```

In[29]:=

```
Length[listes]
```

Out[29]=

```
2
```

□ **Extraction d'une sous-liste : les commandes Take[] et Drop[]**

La commande Take[] permet de prendre une partie de la liste. Pour prendre les n premiers éléments de"liste" taper Take[liste, n], pour les m derniers éléments taper Take[liste, -m] :

In[30]:=

```
Take[alphabet,4]
```

Out[30]=

```
{a, b, c, d}
```

In[31]:=

```
Take[alphabet,-4]
```

Out[31]=

```
{e, f, g, h}
```

Pour choisir les éléments de m à n taper Take[liste, {m,n}] :

In[32]:=

```
Take[alphabet, {2,5}]
```

Out[32]=

```
{b, c, d, e}
```

La commande Drop[] s'utilise de la même façon, mais elle enlève la partie spécifiée et donne le reste de la liste :

In[33]:=

```
Drop[alphabet,2]
```

Out[33]=

```
{c, d, e, f, g, h}
```

In[34]:=

```
Drop[alphabet,-2]
```

Out[34]=

```
{a, b, c, d, e, f}
```

In[35]:=

```
Drop[alphabet, {3,6}]
```

Out[35]=

```
{a, b, g, h}
```

In[36]:=

■ Opérations qui modifient la longueur d'une liste

La longueur d'une liste n'est pas définitivement fixe. Nous pouvons ajouter ou suprimmer des éléments.

□ Ajouter des éléments dans une liste : les commandes Append[] et Prepend[]

La commande Append[liste, elem] permet d'ajouter un élément "elem" à la fin de liste :

In[36]:=

```
Append[alphabet,"i"]
```

Out[36]=

```
{a, b, c, d, e, f, g, h, i}
```

Cette commande crée une nouvelle liste de longueur 9, mais le contenue de 'alphabet' reste inchanger :

In[37]:=

```
alphabet
```

Out[37]=

```
{a, b, c, d, e, f, g, h}
```

Pour modifier 'alphabet' il faut taper l'instruction :

In[38]:=

```
alphabet = Append[alphabet,"i"]
```

Out[38]=

```
{a, b, c, d, e, f, g, h, i}
```

In[39]:=

```
alphabet
```

Out[39]=

```
{a, b, c, d, e, f, g, h, i}
```

La commande Prepend[] se comporte de la même façon que Append[], sauf qu'elle ajoute un élément au début de la liste :

In[40]:=

```
Prepend[mixte,{1,{2,3},4}]
```

Out[40]=

```
{{1, {2, 3}, 4}, 1, b, {11, 12}, 5}
```

Remarque : Ici l'élément ajouté est une liste.

□ Ajouter ou enlever un élément à une position déterminée : les commades Insert[] et Drop[]

Les commandes précédentes permettent d'ajouter un élément au début où à la fin d'une liste. La fonction Insert[liste, element, i] permet d'insérer un élément à la position i :

In[41]:=

```
Insert[alphabet,"#",3]
```

Out[41]=

```
{a, b, #, c, d, e, f, g, h, i}
```

Un signe - devant l'entier de position permet de donner l'indice à partir de la fin de la liste :

In[42]:=

```
Insert[alphabet,"#",-3]
```

Out[42]=

```
{a, b, c, d, e, f, g, #, h, i}
```

La commande Delete[liste, i] permet d'enlever l'élément qui est à la position i :

In[43]:=

```
Delete[alphabet,4]
```

Out[43]=

```
{a, b, c, e, f, g, h, i}
```

□ Réduction de listes imbriquées : la commande Flatten[]

La commande Flatten[liste] supprime les imbrications d'une liste :

In[44]:=

```
Flatten[{ {1,{a,c},2},{3,4} }]
```

Out[44]=

```
{1, a, c, 2, 3, 4}
```

Flatten[liste, n] permet de supprimer les n premiers niveaux d'imbrication :

In[45]:=

```
Flatten[{ {1,{a,c},2},{3,4}}, 1]
```

Out[45]=

```
{1, {a, c}, 2, 3, 4}
```

Il existe aussi une commande FlattenAt[liste,{i,j,..},] pour supprimer des niveaux spécifiques :

In[46]:=

```
FlattenAt[{{1,{a,{aa,bb},b,c},2,3},zz,yy},{1,2}]
```

Out[46]=

```
{{1, a, {aa, bb}, b, c, 2, 3}, zz, yy}
```

Ici nous avons supprimé les accolades à l'élément {1,2} qui est la liste {a,{aa,bb},b,c}.

□ Combinaison de plusieurs listes : les commandes Join[] et Union[]

La fonction Join[liste1, liste2,...] permet de concaténer des listes. Elle conserve les éléments qui sont doubles :

In[47]:=

```
Join[alphabet,mixte,{1,2,5}]
```

Out[47]=

```
{a, b, c, d, e, f, g, h, i, 1, b, {11, 12}, 5, 1, 2, 5}
```

La fonction Union[liste1, liste2, ...] donne la réunion des listes, elle enlève les éléments doubles et elle classe les éléments. Attention, l'ordre des éléments est modifié :

In[48]:=

```
Union[{b,a,d,u,v,j,c},alphabet]
```

Out[48]=

```
{a, b, c, d, e, f, g, h, i, a, b, c, d, j, u, v}
```

Dans l'exemple précédent, il semble que certaine lettre sont doubles. En faite la premiere lettre a est le charactère "a", elle est écrite entre guillemets (voir déclaration de la liste aplphabet) alors que la deuxiéme lettre a est la variable a. Pour obtenir le résultat annoncé il faut mettre les lettres entre guillemets pour les déclarer en tant que lettres et non en tant que variables :

In[49]:=

```
Union[{"b","a","d","u","v","j","c"},alphabet]
```

Out[49]=

```
{a, b, c, d, e, f, g, h, i, j, u, v}
```

In[50]:=

▩ Listes utilisées comme ensemble

Les fonctions Join[] et Union[] peuvent être interprèter comme la réunion de deux listes. Il exite d'autre fonctions mathematica qui sont en relation avec les outils de la théorie des ensembles.

□ L'intersection

La fonction Intersection[liste1,liste2,...] crée la liste d'intersection de plusieurs ensemble :

In[50]:=

```
Intersection[alphabet,{"a","b","a","x","y","z"}]
```

Out[50]=

```
{a, b}
```

Remarquer que les éléments doubles n'apparaissent pas deux fois :

In[51]:=

```
Intersection[{a,a},{a,a,b}]
```

Out[51]=

```
{a}
```

□ **L'ensemble complémentaire**

La fonction Complement[environnement,liste1,...] crée une liste avec les éléments de liste1, liste2,... qui ne sont pas dans evironement :

In[52]:=

```
Complement[alphabet,{"d","e","f"},{"a","b"}]
```

Out[52]=

```
{c, g, h, i}
```

Si dans liste1 il y a des éléments en dehors de environnement alors ils seront suprimmés :

In[53]:=

```
Complement[alphabet,{"a","x","y"}]
```

Out[53]=

```
{b, c, d, e, f, g, h, i}
```

□ **Les partitions et regroupements**

La fonction Partition[liste, n] permet de créer une partition de la liste avec des sous ensembles à n éléments :

In[54]:=

```
Partition[alphabet,3]
```

Out[54]=

```
{{a, b, c}, {d, e, f}, {g, h, i}}
```

Si n ne divise pas la longueur de la liste, alors la partition obtenue ne contient pas tous les éléments de la liste.

In[55]:=

```
Length[alphabet]
```

Out[55]=

```
9
```

In[56]:=

```
Partition[alphabet,2]
```

Out[56]=

```
{{a, b}, {c, d}, {e, f}, {g, h}}
```

La fonction Partition[liste, n, m] permet de regrouper les éléments par ensemble de n éléments avec un décalage de m éléments. Chaque sous ensemble ainsi créé a une intersection de n-m élément avec son prédécesseur et son successeur :

In[57]:=

```
Partition[alphabet,4,3]
```

Out[57]=

```
{{a, b, c, d}, {d, e, f, g}}
```

Si n < m alors nous obtenons une partition avec un <<trou>> de m-n éléments entre eux :

In[58]:=

```
Partition[alphabet,2,5]
```

Out[58]=

```
{{a, b}, {f, g}}
```

□ **Permutation cyclique d'éléments**

La fonction RotateLeft[liste] effectue une permutation circulaire vers la gauche des éléments de la liste :

In[59]:=

```
RotateLeft[alphabet]
```

Out[59]=

```
{b, c, d, e, f, g, h, i, a}
```

de même il existe RotateRight[liste] pour une permutation vers la droite :

In[60]:=

```
RotateRight[alphabet]
```

Out[60]=

```
{i, a, b, c, d, e, f, g, h}
```

La commande RotateLeft[liste, n] permet de faire une permutation de n éléments vers la gauche :

In[61]:=

```
RotateLeft[alphabet, 3]
```

Out[61]=

```
{d, e, f, g, h, i, a, b, c}
```

et vers la droite :

In[62]:=

```
RotateRight[alphabet, 4]
```

Out[62]=

```
{f, g, h, i, a, b, c, d, e}
```

□ Permutation d'éléments

La fonction Permutations[liste] crée une liste avec toutes les permutations possibles :

In[63]:=

```
Permutations[{1,2,3}]
```

Out[63]=

```
{{1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2},

 {3, 2, 1}}
```

La fonction Signature[liste] donne la signature de la permutation qui ordonne la liste dans un ordre standard :

In[64]:=

```
Signature[{a,b,c}]
```

Out[64]=

```
1
```

In[65]:=

```
Signature[{b,a,c}]
```

Out[65]=

```
-1
```

Pour tester si les éléments de la liste sont dans l'ordre poser la question OrderedQ[liste] :

In[66]:=

```
OrderedQ[alphabet]
```

Out[66]=

```
True
```

In[67]:=

```
OrderedQ[{3,5,4}]
```

Out[67]=

```
False
```

C'est l'odre croissant qui compte :

In[68]:=

```
OrderedQ[{5,4,3}]
```

Out[68]=

```
False
```

In[69]:=

▩ Utilisation des listes en algèbre linéaire

□ Définition d'une matrice comme liste multiple :

Une matrice est décrite dans mathematica comme une liste qui a autant d'éléments que la matrice a de lignes et où chaque élément est la liste des éléments d'une ligne. voici un exemple :

Une matrice à deux lignes et trois colonnes s'écrit de la façon suivante :

In[69]:=

```
M = { {a11, a12, a13}, {a21,a22,a23} }
```

Out[69]=

```
{{a11, a12, a13}, {a21, a22, a23}}
```

□ Affichage matriciel :

La commade MatrixForm[liste] permet un affichage sous forme de matrice classique :

In[70]:=

```
MatrixForm[M]
```

Out[70]//MatrixForm=

```
a11   a12   a13

a21   a22   a23
```

□ Produit matriciel

Le point "." est utilisé pour symboliser le produit matriciel :

In[71]:=

```
S= {{b11,b12},{b21,b22}};
S.M
```

Out[72]=

```
{{a11 b11 + a21 b12, a12 b11 + a22 b12, a13 b11 + a23 b12},

 {a11 b21 + a21 b22, a12 b21 + a22 b22, a13 b21 + a23 b22}}
```

□ La matrice transposée

Pour transposer la matrice, utiliser la fonction Transpose[liste] :

In[73]:=

```
MatrixForm[Transpose[M]]
```

Out[73]//MatrixForm=

```
a11   a21

a12   a22

a13   a23
```

☐ **La matrice inverse**

Pour inverser une matrice, utiliser la fonction Inverse[liste] :

In[74]:=

```
MatrixForm[Inverse[S]]
```

Out[74]//MatrixForm=

```
          b22                          b12
----------------------      -(----------------------)
-(b12 b21) + b11 b22          -(b12 b21) + b11 b22

           b21                          b11
--(----------------------)    ----------------------
  -(b12 b21) + b11 b22        -(b12 b21) + b11 b22
```

☐ **Le déterminant**

La fonction Det[liste] donne le déterminant de la matrice :

In[75]:=

```
Det[S]
```

Out[75]=

```
-(b12 b21) + b11 b22
```

☐ **La dimension d'une matrice**

La fonction Dimensions[liste] donne les dimensions de la matrice sous forme de liste :

In[76]:=

```
Dimensions[M]
```

Out[76]=

```
{2, 3}
```

☐ **Les mineurs d'une matrice**

*La fonction Minors[M, k] donne la liste des k*k mineurs de la matrice M :*

In[77]:=

```
Minors[M,2]
```

Out[77]=

```
{{-(a12 a21) + a11 a22, -(a13 a21) + a11 a23,

  -(a13 a22) + a12 a23}}
```

☐ **Matrice Pussance et Exponentielle**

La fonction MatrixPower[matrice,n] donne la puissance n-iéme de la matrice :

In[78]:=

MatrixPower[S,3]

Out[78]=

```
                  2
{{b11 (b11  + b12 b21) + b21 (b11 b12 + b12 b22),

                  2
  b12 (b11  + b12 b21) + b22 (b11 b12 + b12 b22)},

                                                 2
 {b11 (b11 b21 + b21 b22) + b21 (b12 b21 + b22 ),

                                                 2
  b12 (b11 b21 + b21 b22) + b22 (b12 b21 + b22 )}}
```

La fonction MatrixExp[matrice] donne l'exponentiel de la matrice :

In[79]:=

MatrixForm[MatrixExp[{{0,1},{1,0}}]]

Out[79]//MatrixForm=

$$\begin{pmatrix} \frac{1}{2\,E} + \frac{E}{2} & \frac{-1}{2\,E} + \frac{E}{2} \\ \frac{-1}{2\,E} + \frac{E}{2} & \frac{1}{2\,E} + \frac{E}{2} \end{pmatrix}$$

Représentations graphiques

1. Introduction

Mathematica dispose de fonctions graphiques sophistiquées, autorisant aussi bien les reprèsentations 2D que 3D, cartèsiennes ou paramétriques, en échelle logarithmique, semi-logarithmique, par lignes de niveau, par niveaux de gris (graphiques de densité). On peut aussi réaliser des graphiques de valeurs discrètes : on obtient un ensemble de points, que l'on peut relier entre eux par des segments de droite.

De nombreuses options sont disponibles : longueur de segments, titres, domaine de représentation, couleurs, pointillés, taille des points, quadrillage, cadre, graduation des axes etc.

Une suite de graphiques peut même être incorporée dans un tableau, par exemple en taille réduite de façon à être visualisée d'un seul coup d'oeil. Il est même très facile de réaliser des graphiques animés, en 2D ou 3D. Des programmes externes, tels que MathLive permettent un rendu réaliste des objets 3D (texture lisse) et une manipulation avec la souris en temps réel de ces objets (rotations, translations, homothéties).

Même si la programmation des graphiques en Mathematica peut apparaitre pénible au débutant, elle s'avère d'une telle richesse et d'une telle puissance qu'il est nécessaire de prendre quelques heures pour bien l'étudier. Les possibilités offertes sont immenses et n'ont, en fait, de réelles limites que celles de notre propre imagination. C'est ainsi que ce que l'on pense être impossible à réaliser par le logiciel n'est souvent que l'expression de notre ignorance du langage, et une étude approfondie de la littérature spécialisée sur le graphisme avec Mathematica aura tôt fait de nous rassurer... voire de nous émerveiller.

Enfin, Mathematica contient des primitives graphiques 2D et 3D telles que : point, segment, rectangle, cercle, polygone, parallélépipède, ligne polygonale, segment de texte...

Un package externe, Descartes2D distribué par la société MOI S.A., utilise ces primitives graphiques ainsi que d'autres ressources de Mathematica afin de proposer tout un ensemble de fonctions nouvelles pour faire de la géométrie cartésienne : grâce à ce module additionnel, il est donc possible de démontrer une propriété géométrique, y compris sur les coniques, en passant par la géométrie analytique. L'avantage réside en la totale compatibilité de ces nouvelles fonctions avec le reste du logiciel : à tout moment, l'utilisateur peut créer ses propres fonctions incorporant les nouvelles primitives du package.

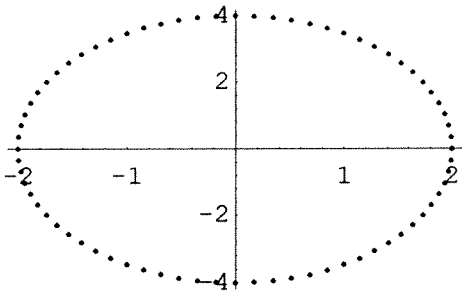
2. Courbes en 2 dimensions

Premiers exemples

Exemples de base :

In[1]:=

dessin0=ListPlot[Table[{2Cos[i*Degree],4Sin[i*3.14/180]},{i,0,360,5}]]

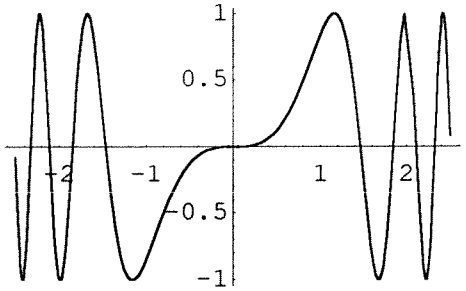


Out[1]=

-Graphics-

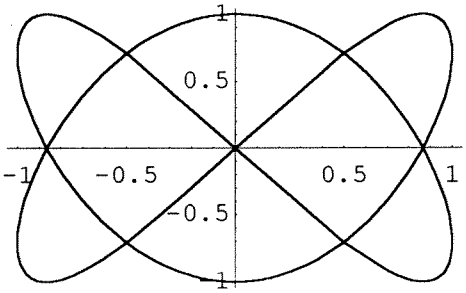
In[2]:=

dessin1=Plot[Sin[x^3],{x,-2.5,2.5}];



In[3]:=

dessin2=ParametricPlot[{Sin[2t],Cos[3t]},{t,0,2Pi}]

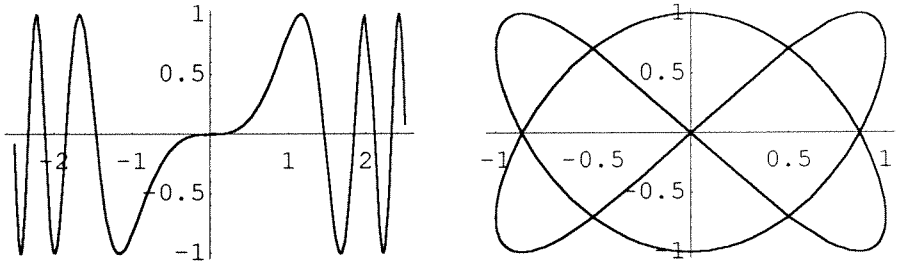


Out[3]=

-Graphics-

In[4]:=

Show[GraphicsArray[{dessin1,dessin2}]]



Out[4]=

-GraphicsArray-

Les fonctions de base de dessin en 2D sont :

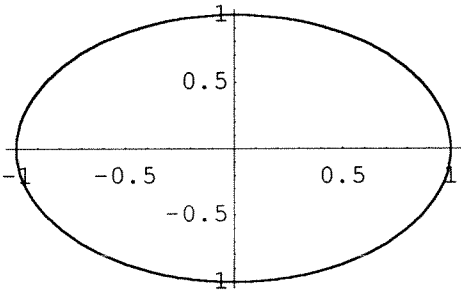
ListPlot[liste_de_points] ,
Plot[f[x], {x,xmin,xmax}] et
ParametricPlot[{x[t], y[t]}, {t, tmin, tmax}].

Pour représenter en ligne une série de graphiques, on utilise la fonction GraphicsArray[] associée à la commande de visualisation Show.

■ Exemple 2

In[5]:=

ParametricPlot[{Sin[t],Cos[t]},{t,0,2Pi}]



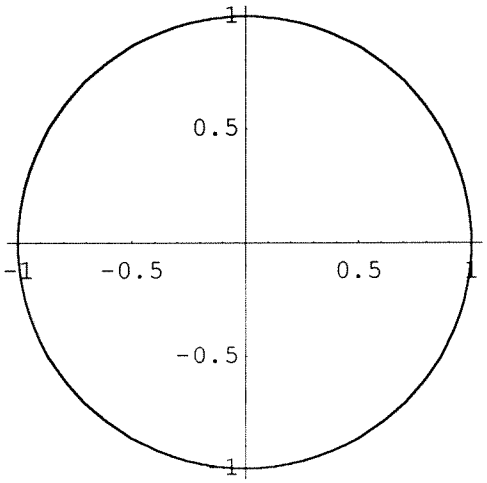
Out[5]=

```
-Graphics-
```

On remarque que la représentation graphique obtenue n'est pas un cercle. En effet, le repère choisi par défaut n'est pas orthonormé : le rapport entre les modules des vecteurs unitaires est le nombre d'or. Ce rapport peut être fixé par l'utilisateur à l'aide de l'option "AspectRatio".

In[6]:=

ParametricPlot[{Sin[t],Cos[t]},{t,0,2Pi},AspectRatio->1]



Out[6]=

```
-Graphics-
```

De façon générale les fonctions graphiques s'écrivent :

ListPlot[liste_de_points, options_graphiques] ,
Plot[f[x], {x,xmin,xmax}, options_graphiques] et
ParametricPlot[{x[t], y[t]}, {t, tmin, tmax}, options_graphiques].

Options graphiques 2D

Il existe une grande liste d'options graphiques, les plus importantes sont :

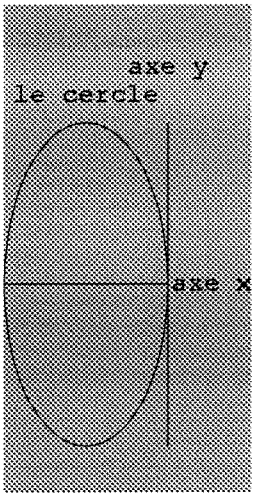
PlotJoined ->True : joint deux points successifs d'une liste par un segment.
AspectRatio->(echelle relative hauteur/largeur)
Axes->True : trace les axes.
Axes->False : le graphique ne comporte pas les axes.
AxesOrigin->{x0,y0} : les axes représentés sont les droites x=x0, y=y0.
AxesLabel->{"axe des abscisses", "axe des ordonnées"} : représente les noms des axes.
Frame->True : encadre le graphique.
PlotLabel->"dessin" : représente le nom du graphique.
PlotRange->{{xmin,xmax},{ymin,ymax}} : limites du graphique.
PlotRegion->{{mx,sx},{my,sy}} : tracé homothétique
PlotRegion->{{0,1},{0,1}} (le tracé utilise tout l'écran)
(utile pour faire des zooms).
Ticks->True : marquage des graduations des axes.
Ticks->False : graduations non marquées.
PlotStyle->{Thickness[valeur]
PointSize[valeur],
Dashing[{r1,r2,...}]
RGBColor[rouge,vert,bleu]
}

In[7]:=

```
(* liste des options choisies pour les graphiques *)
opts = Sequence[
        AspectRatio->2,
        Axes->True,
        AxesOrigin->{1,0},
        Background->GrayLevel[3/4],
        AxesLabel->{"axe x", "axe y"} ,
        PlotLabel->"le cercle",
        PlotRange->{{-1,1},{-1,1}},
        Ticks->None,
        PlotStyle->{ RGBColor[1,0,0],
                     Thickness[0.05],
                   }];
```

In[10]:=

ParametricPlot[{Sin[t],Cos[t]},{t,0,2Pi},Evaluate[opts]]



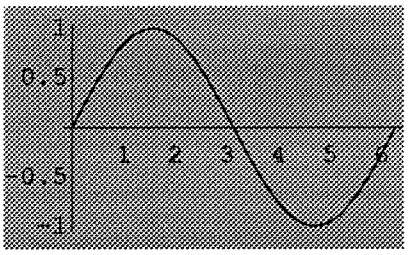
Out[10]=

-Graphics-

Voici quelques exemples pour illustrer l'option "PlotRegion"

In[11]:=

i=1;j=1;
Plot[Sin[x],{x,0,2Pi},PlotRegion->{{0,i},{0,j}},
Background->GrayLevel[2/3]]

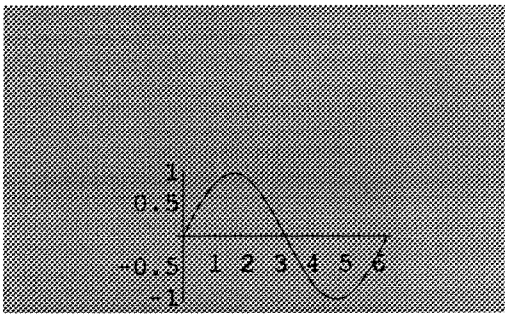


Out[12]=

-Graphics-

In[13]:=

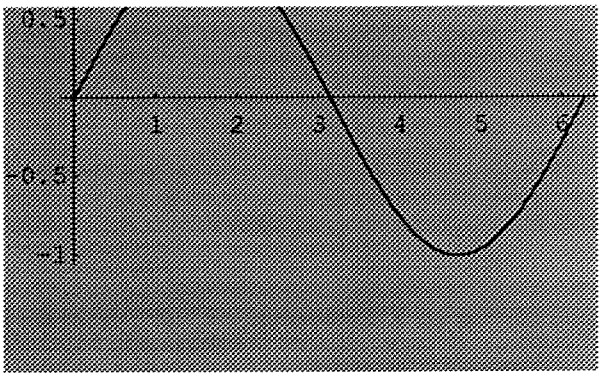
i=1;j=1/2;
Plot[Sin[x],{x,0,2Pi},PlotRegion->{{0,i},{0,1/2}},
Background->GrayLevel[2/3]]



Out[14]=

-Graphics-

```
i=1; j=1.5;
Plot[Sin[x],{x,0,2Pi},PlotRegion->{{0,i},{0,j}},
Background->GrayLevel[2/3]]
```



Out[16]=

```
-Graphics-
```

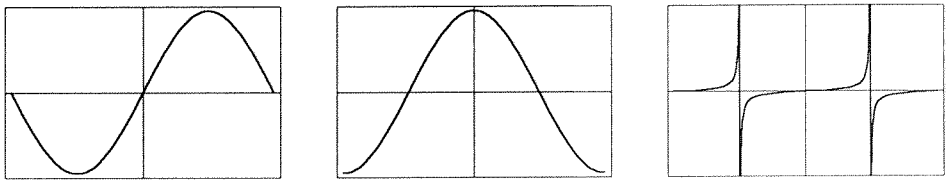
Voilà un ensemble de graphiques, programmé de façon très rationnelle, grâce à des manipulations de listes et à l'utilisation d'une fonction "pure" qui s'applique à une liste de "noms de fonctions" (ex : cosinus, sinus...).

In[17]:=

```
opts=Sequence[Axes->True, Ticks->None, Frame->True,
        FrameTicks->None,DisplayFunction->Identity];
```

In[18]:=

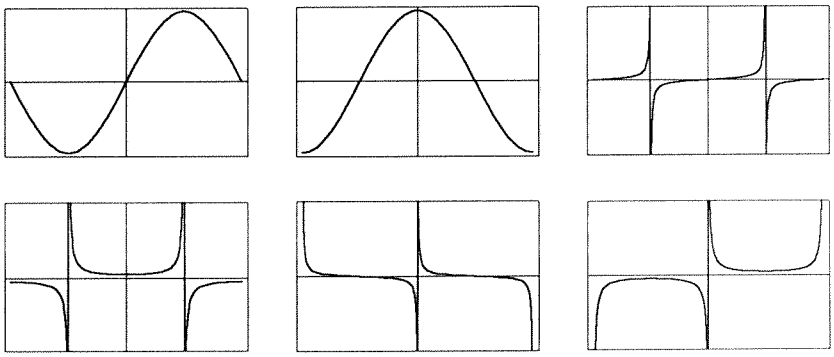
```
(* liste de graphiques *)
gr={Plot[Sin[x],{x,-Pi,Pi},Evaluate[opts]], Plot[Cos[x],{x,-Pi,Pi},Evaluate[opts]],
        Plot[Tan[x],{x,-Pi,Pi},Evaluate[opts]]};

(* représentation en tableau des différents graphiques *)
Show[GraphicsArray[gr, GraphicsSpacing->{.2,.3}]];
```



L'écriture de gr est particulièrement pénible (répétition de Plot[ ]). L'utilisation des fonctions pures (voir exemple suivant) est beaucoup plus pratique.

In[23]:=

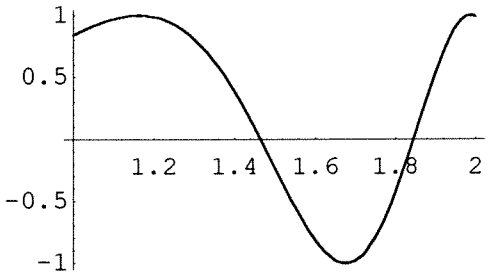
```
(* liste de graphiques *)
gr=Plot[#[x],{x,-Pi,Pi},Evaluate[opts]]& /@
                                    {Sin,Cos,Tan,Sec,Cot,Csc};

(* groupement par trois des graphiques pour obtenir deux lignes *)
gr = Partition[gr,3];

(* représentation en tableau des différents graphiques *)
Show[GraphicsArray[gr, GraphicsSpacing->{.2,.3}]];
```



■ Restriction de l'intervalle des abscisses

In[29]:=

```
Plot[Sin[x^3],{x,1,2}];
```

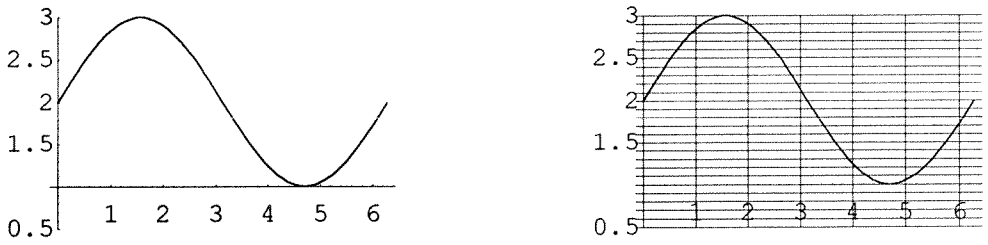


■ Choix de l'intervalle des ordonnées

Par l'option PlotRange ->{y1, y2}. Seul gr0 constitue ici le graphique fondamental. Les autres lignes de commandes montrent comment réaliser un quadrillage personnalisé, sans modifier la ligne de commande du graphique initial.

In[30]:=

```
gr0 = Plot[Sin[x]+2,{x,0, 2Pi}, PlotRange->{.5, 3},
DisplayFunction->Identity];
majeures = {1/2,3/2,2,5/2,3};
mineures = {#,{GrayLevel[.7]}}& /@ Complement[Range[1/2,3,1/10],
                                              majeures,{0}];

gr1 = Show[gr0, GridLines->{Automatic, Join[majeures, mineures]},
                                    DisplayFunction->Identity];

Show[GraphicsArray[{gr0,gr1}, GraphicsSpacing->.5,
          DisplayFunction:>$DisplayFunction]];
```

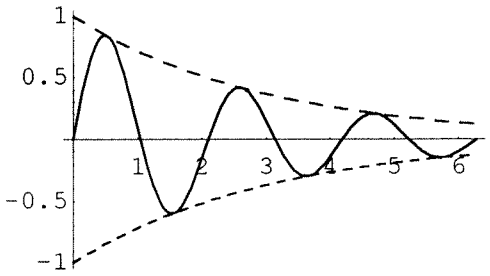


■ Superposition de plusieurs courbes

Pour les distinguer les unes des autres, on peut utiliser des effets particuliers (pointillés, épaisseur de trait, couleurs...).

In[37]:=

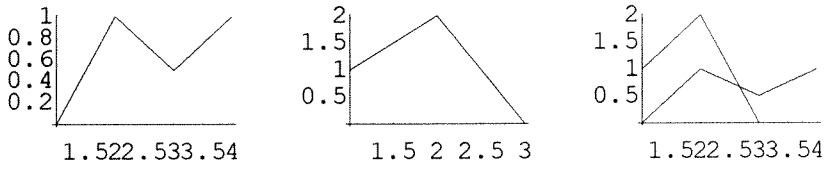
```
Plot[{Exp[-x/3] Sin[3x], Exp[-x/3],-Exp[-x/3]},{x,0,2Pi},
          PlotStyle->{Dashing[{}],Dashing[{0.03,0.03}],
                Dashing[{0.02,0.02}]}];
```



On peut aussi utiliser la fonction Show . Voici un exemple tirant parti de la commande ListPlot pour dessiner une liste de points. Grâce à l'option PlotJoined->True, ces points sont reliés par une ligne polygonale.

In[38]:=

```
graph1 = ListPlot[{0,1,0.5,1},PlotStyle->RGBColor[1,0,0],
                                PlotJoined->True, DisplayFunction->Identity];
graph2 = ListPlot[{1,2,0},PlotStyle->RGBColor[0,1,0],
                                PlotJoined->True, DisplayFunction->Identity];
graph3 = Show[graph1, graph2, DisplayFunction->Identity];
Show[GraphicsArray[{graph1, graph2,graph3}],
                                DisplayFunction:>$DisplayFunction];
```



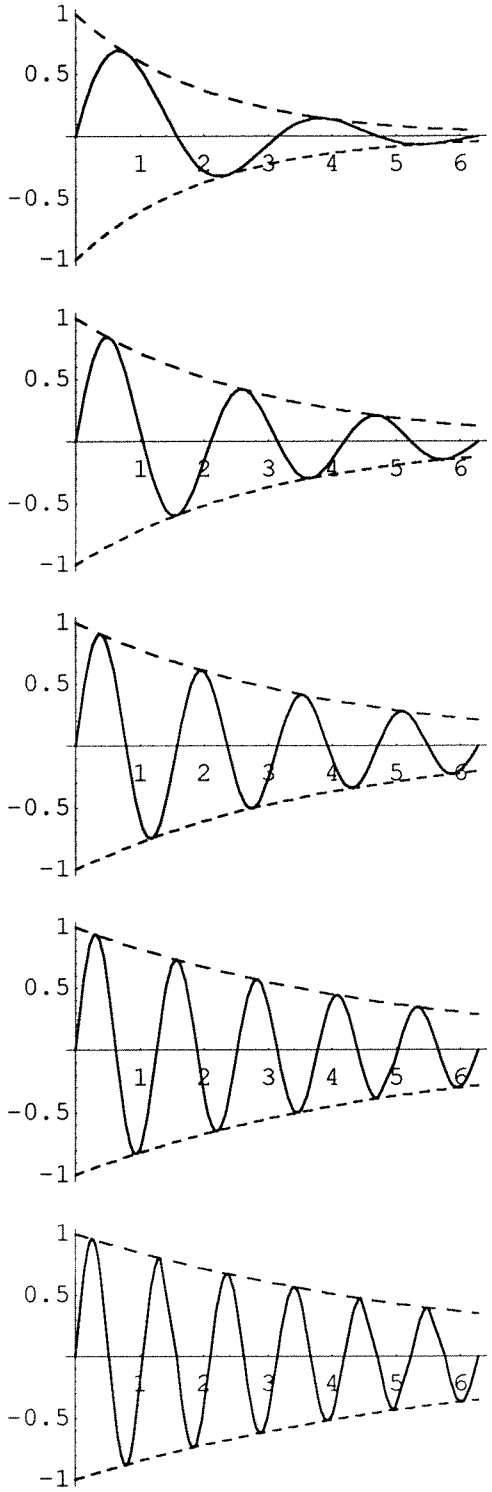
■ Animations

Pour réaliser un "dessin animé", il suffit de dessiner une suite de figures (commande Do) que l'on groupera dans une cellule unique en double-cliquant sur le grand crochet (à droite) qui réunit les graphiques. L'animation est produite par la sélection de Animate Selected Graphics du menu Graph. Le contrôle de la vitesse et du sens de défilement de l'image est assuré par les boutons en bas à gauche.

Voici un exemple d'une animation graphique s'appuyant sur l'exemple précédent :

In[42]:=

```
Do[Plot[{Exp[-x/p] Sin[p x], Exp[-x/p],-Exp[-x/p]},{x,0,2Pi},
        PlotStyle->{Dashing[{}],Dashing[{0.03,0.03}],
                   Dashing[{0.02,0.02}]}];,{p,2,6}];
```



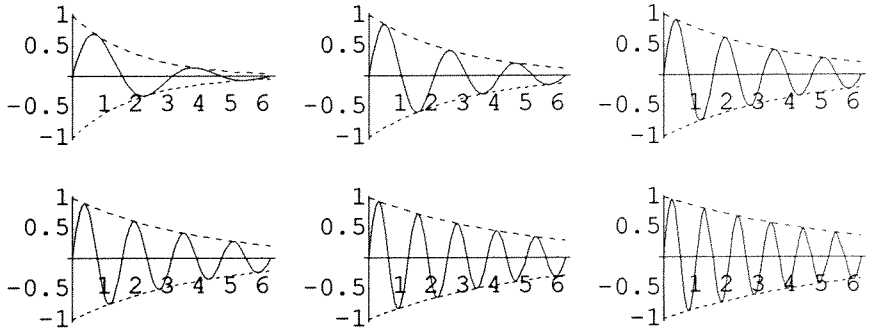
Pour représenter les différents graphiques sur le papier, on peut les placer dans un tableau :

In[43]:=

```
Clear[gr]
```

In[44]:=

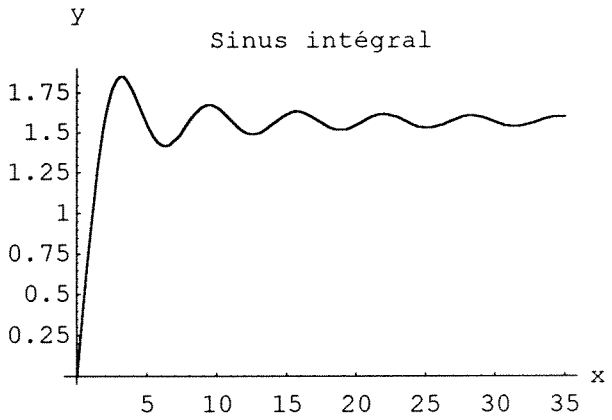
```
gr[p_] := Plot[{Exp[-x/p] Sin[p x], Exp[-x/p],-Exp[-x/p]},{x,0,2Pi},
            PlotStyle->{Dashing[{}],Dashing[{0.03,0.03}],
                        Dashing[{0.02,0.02}]},DisplayFunction->Identity];

Show[GraphicsArray[{Table[gr[p],{p,2,4}],Table[gr[p],{p,4,6}]}],
                        DisplayFunction->$DisplayFunction];
```



■ Titre d'un graphique et noms des axes

In[48]:=

```
f[x_]=Integrate[Sin[u]/u,{u,0,x}];
Plot[f[x],{x,0,35},AxesLabel->{"x","y"},
            PlotLabel->"Sinus intégral",PlotRange->All];
```



Le graphique ci-dessus donne envie de conjecturer que la fonction "Sinus intégral" a pour limite un nombre voisin de 1.5. Vérifions cela en calculant l'intégrale généralisée suivante :

In[50]:=

```
Integrate[Sin[u]/u,{u,0,Infinity}]
```

Out[50]=

$$\frac{\text{Pi}}{2}$$

In[51]:=

```
N[%]
```

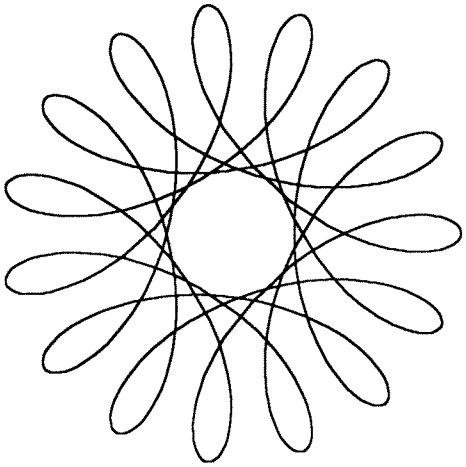
Out[51]=

```
1.5708
```

■ Courbes paramétrées

In[1]:=

```
ParametricPlot[{4 Cos[-11t/4]+7 Cos[t], 4 Sin[-11t/4]+7 Sin[t]},
        {t,0,8Pi}, AspectRatio->Automatic, Axes->None];
```



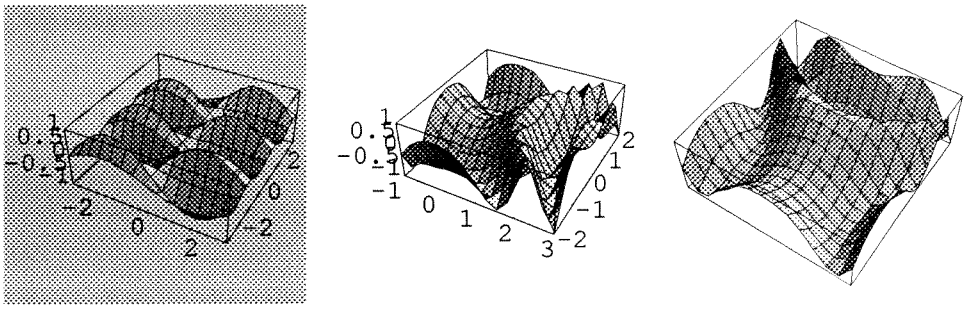
In[2]:=

▒ 3. Courbes en 3 dimensions

■ Premiers exemples

In[2]:=

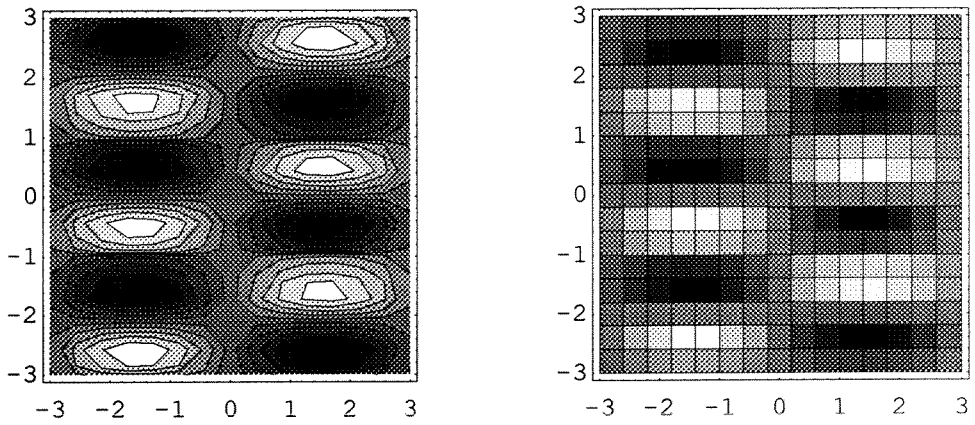
```
gr1 = Plot3D[Sin[x] Sin[3y], {x,-3,3},{y,-3,3},
        Background->GrayLevel[0.8], DisplayFunction->Identity];
gr2 = Plot3D[Sin[x^2 - y^2],{x,-1,3},{y,-2,2},
                DisplayFunction->Identity];
gr3 = Plot3D[Sin[x^2 - y^2],{x,-1,3},{y,-2,2}, Ticks->None,
                ViewPoint->{-1.679, -1.117, 2.717},
                DisplayFunction->Identity];
Show[GraphicsArray[{gr1,gr2,gr3}],DisplayFunction->$DisplayFunction];
```



■ Lignes de niveau, graphiques de densité

In[6]:=

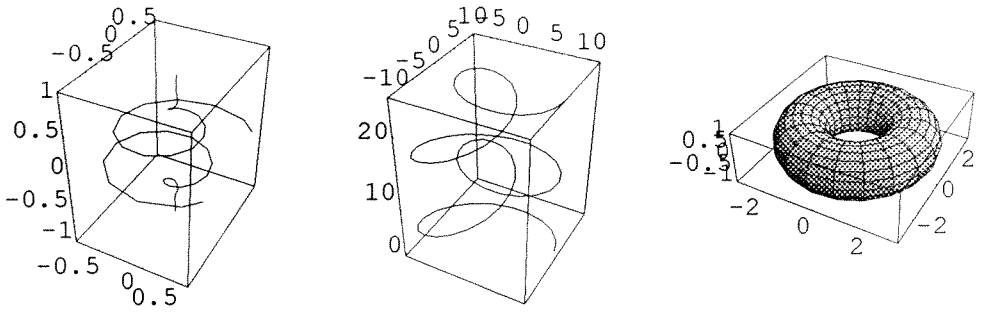
```
gr1 = ContourPlot[Sin[x] Sin[3y], {x,-3,3},{y,-3,3},
                  DisplayFunction->Identity];
gr2 = DensityPlot[Sin[x] Sin[3y], {x,-3,3},{y,-3,3},
                  DisplayFunction->Identity];
Show[GraphicsArray[{gr1,gr2}, GraphicsSpacing->.3]];
```



■ Courbes et surfaces paramétrées

In[10]:=

```
gr1 = ParametricPlot3D[{Cos[2Pi t] Exp[-t^2], Sin[2Pi t] Exp[-t^2],  t/3},
                       {t,-3,3}, DisplayFunction->Identity];
gr2 = ParametricPlot3D[{4Cos[-t/2]+7Cos[t], 4Sin[-t/2]+7Sin[t],t},
                       {t,0,8Pi}, DisplayFunction->Identity];
gr3 = ParametricPlot3D[{Cos[s](2+Cos[t]), Sin[s](2+Cos[t]),Sin[t]},
            {s,0,2Pi},{t,0,2Pi}, DisplayFunction->Identity];

Show[GraphicsArray[{gr1,gr2,gr3}],DisplayFunction->$DisplayFunction];
```



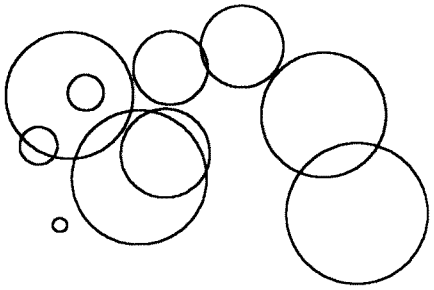
4.Objets graphiques élémentaires

Exemples en 2D

Quelques bulles de savon :

In[16]:=

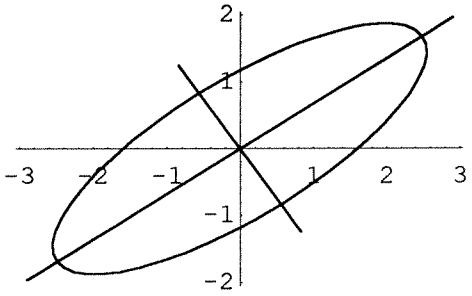
```
bulles = Table[Circle[{5 Random[],3 Random[]},Random[]],{10}];
gr1 = Show[Graphics[bulles], AspectRatio->Automatic];
```



Une ellipse point par point, avec tracé des axes :

In[91]:=

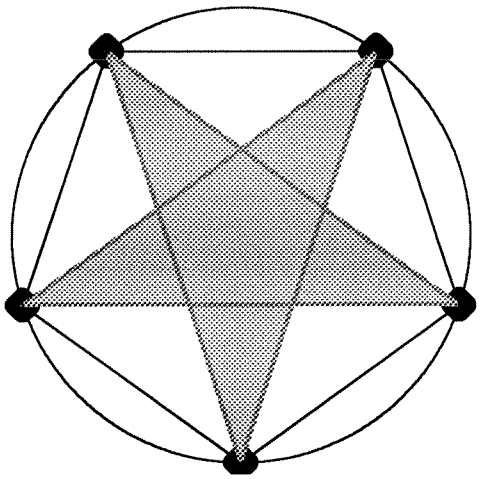
```
ellipse[t_] := {3 Cos[t], Sin[t]};
theta = 34 Degree;
            (* matrice de rotation *)
matrice = {{Cos[theta], -Sin[theta]},{Sin[theta],Cos[theta]}};
rotationEllipse = Table[matrice.ellipse[t],{t,0,2Pi,Pi/24}];
a = matrice.{3.5,0};
b = matrice.{0,1.5};
gr2 = Show[Graphics[{Line[rotationEllipse],
            Line[{a,-a}],
            Line[{b,-b}]},Axes->Automatic]];
```



Une figure polygonale :

In[99]:=

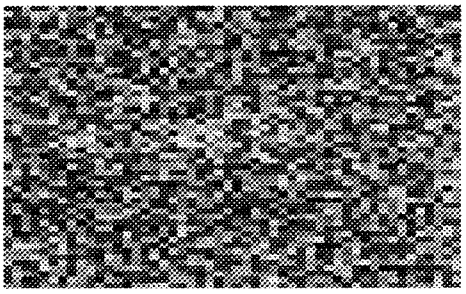
```
primRac5Unite = Exp[2Pi I / 5];
pts = {Re[#],Im[#]}& /@ (-I N[primRac5Unite^Range[5]]);
pts2 = Join[pts,pts][[ 2 Range[5] ]];

LimnPolygon[points:{__},linestyle___] :=
        {Polygon[points],Flatten[{linestyle,
         Line[Append[points,First[points]] ]}]};
        gr3 = Show[Graphics[{
                        PointSize[.05], Point/@pts,
                        Line[Append[pts,First[pts]] ],
                        { GrayLevel[.8], LimnPolygon[pts2,GrayLevel[.6],
                                        Thickness[.01]] },
                        Circle[{0,0},1]
                        }],AspectRatio->1];
```



La fonction Raster convertit une liste de nombres en une ligne composée de petits carrés juxtaposés. La couleur (ou le niveau de gris) de chaque carré est fonction du nombre coorespondant dans la liste. Cette commande est utilisée pour convertir des images digitalisées importées en graphiques Mathematica .

In[33]:=

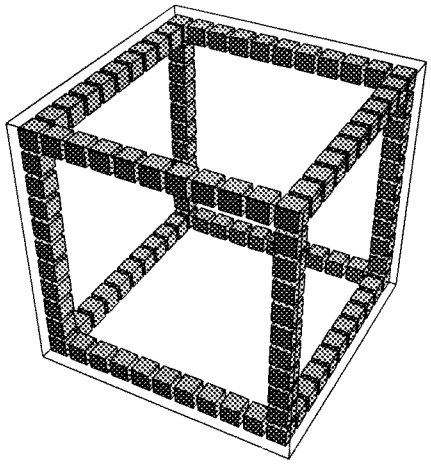
```
m = Table[Table[Random[],{i,1,50}],{j,1,50}];
Show[Graphics[Raster[m,ColorFunction->Hue]]];
```



■ Exemples en 3D

In[35]:=

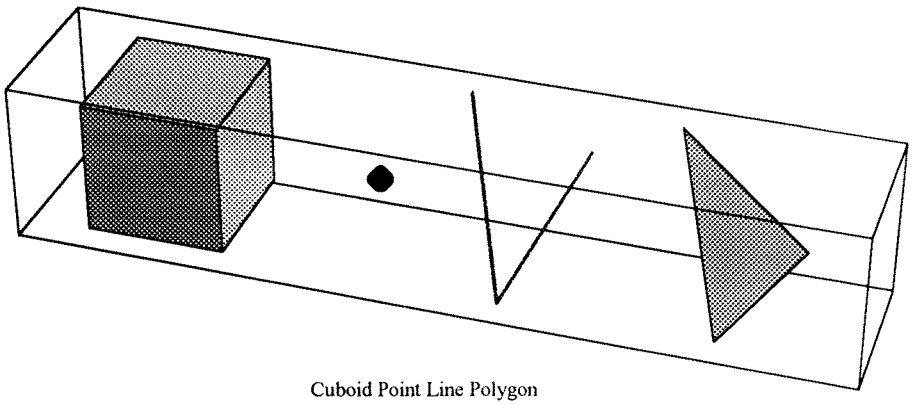
```
ClearAll[f,g,h]
g[l_,d_] := Cuboid[l,l+d]
h[m_,l_] := g[ Insert[l,#,m],-0.08]& /@ Range[0,1,.1]
f[l_] := h[#,l]& /@ {1,2,3}
Show[Graphics3D[f /@ {{0,0},{0,1},{1,0},{1,1}} ]];
```



Autre exemple plus élémentaire :

In[40]:=

```
Show[Graphics3D[{
        Cuboid[{0,0,0},{1,1,1}],
        PointSize[.02], Point[{2,.5,.5}],
        RGBColor[0,0,1], Line[{{2.5,1,1},{3,0,0},{3.5,.5,1}}],
        Polygon[{{4,1,1},{4.5,0,0},{5,.5,.5}}]
        },
        PlotRange->{{-.5,5.5},{-.1,1.1},{-.1,1.1}},BoxRatios->Automatic,
        ViewPoint->{0.885, -2.703, 1.833}, Ticks->None,
                Axes->{True,False,False}, DefaultFont->{"Times-Roman",8},
                AxesLabel->
        {"Cuboid Point Line Polygon",
                None, None}
]];
```



Cuboid Point Line Polygon

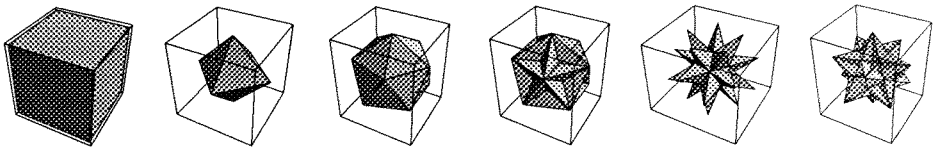
5. Utilisation des packages standards

In[41]:=

```
Needs["Graphics`Polyhedra`"];
```

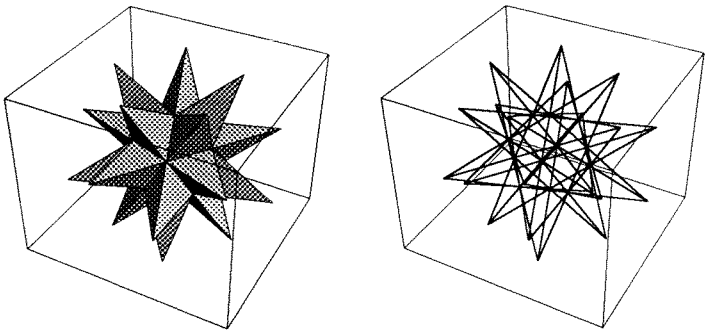
In[42]:=

```
Show[GraphicsArray[Polyhedron/@
{Hexahedron,Octahedron,Icosahedron,GreatDodecahedron,
GreatStellatedDodecahedron, GreatIcosahedron}]];
```



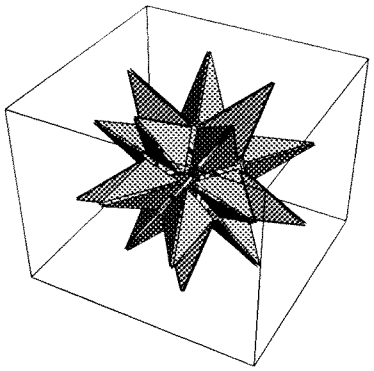
In[43]:=

```
stella = Show[Polyhedron[GreatStellatedDodecahedron],
                                        DisplayFunction->Identity];
stellaFil = Show[stella/.Polygon->Line, DisplayFunction->Identity];
Show[GraphicsArray[{stella,stellaFil}],
        DisplayFunction->$DisplayFunction];
```



In[46]:=

```
Show[stella/.Polygon[ pts_] :>
     Module[{cent = Apply[ Plus, pts]/4.},
        Polygon[Map[(#-cent)0.8 + cent&,pts]]],
DisplayFunction->$DisplayFunction ]
```



Out[46]=

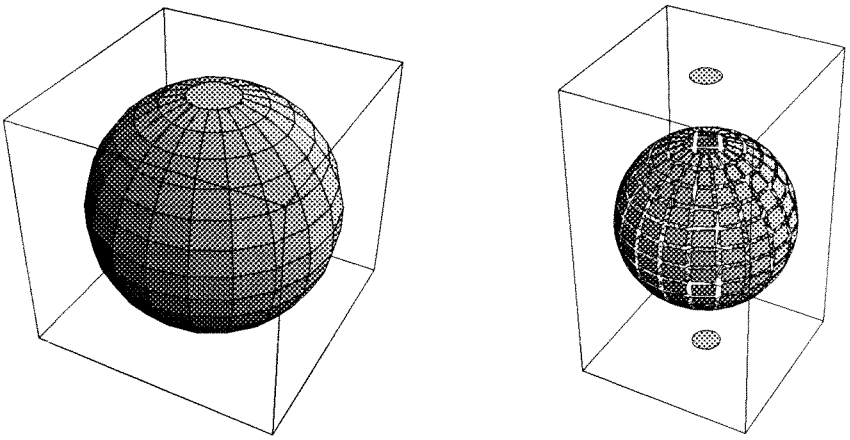
```
-Graphics3D-
```

In[47]:=

```
Needs["Graphics`Shapes`"];
```

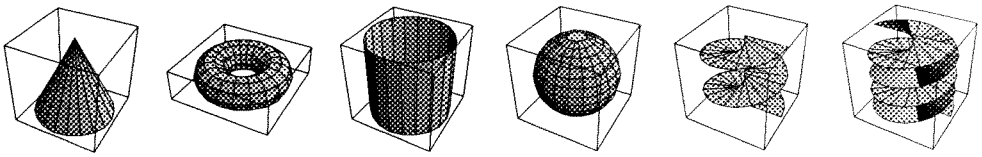
In[48]:=

```
sphr = Show[Graphics3D[Sphere[]],DisplayFunction->Identity];
sphr2 = Show[sphr /. Polygon[ pts_] :>
      Module[{cent = Apply[ Plus, pts]/4.},
         Polygon[Map[(#-cent)0.8 + cent&,pts]]],
         DisplayFunction->Identity];
Show[GraphicsArray[{sphr,sphr2}],
                        DisplayFunction->$DisplayFunction];
```



In[51]:=

```
Show[GraphicsArray[Graphics3D /@
            {Cone[], Torus[], Cylinder[], Sphere[], Helix[], DoubleHelix[]}
            ]];
```



In[52]:=

```
Needs["Graphics`Graphics3D`"];
```

In[53]:=

```
pts = Table[{t,Cos[t]^2,Sin[2t]},{t,0,5Pi,Pi/15}];
gr0 = ScatterPlot3D[pts,DisplayFunction->Identity];
```

In[55]:=

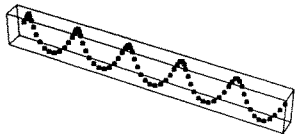
```
Needs["Graphics`ImplicitPlot`"];
```

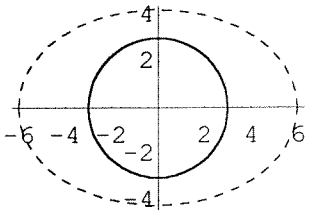
In[56]:=

```
gr = ImplicitPlot[{x^2 + 2 y^2 == 36, x^2 + y^2 == 9},{x,-6,6},
            PlotStyle->{Dashing[{0.03}], Thickness[.007]},
                        DisplayFunction->Identity];
```

In[57]:=

```
Show[GraphicsArray[{gr0,gr}],DisplayFunction:>$DisplayFunction,
                        GraphicsSpacing->1];
```



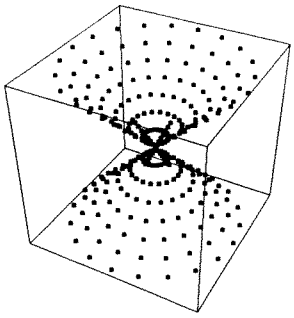


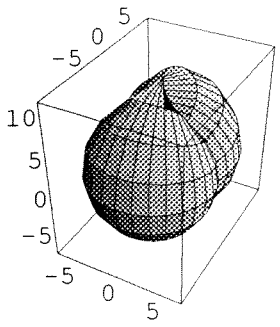
In[58]:=

```
Needs["Graphics`ParametricPlot3D`"];
```

In[107]:=

```
gr1 = PointParametricPlot3D[{v Cos[u],v Sin[u],v},
            {u,0,2Pi,Pi/10},{v,-2,2},DisplayFunction->Identity];
gr2 = SphericalPlot3D[(2+Sin[phi])(3+Cos[theta]),
            {theta,0,2Pi},{phi,0,Pi,Pi/18},DisplayFunction->Identity];
Show[GraphicsArray[{gr1,gr2}],DisplayFunction:>$DisplayFunction,
                        GraphicsSpacing->1];
```

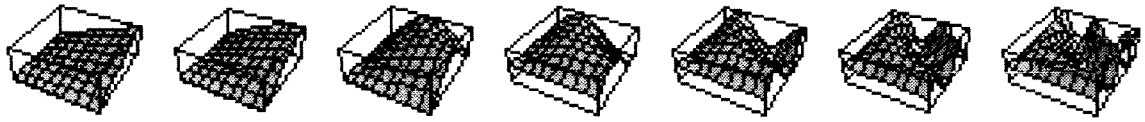




Partie II

DES EXEMPLES POUR L'ENSEIGNEMENT

Des exemples d'applications pour les CPGE et le DEUG.



Exemple 1

(d'après QCM Mathématiques - Dunod)

Soit la matrice 3 x 3 maMatrice :

```
(maMatrice={{0,1,0},{0,0,1},{1,0,0}})//MatrixForm
```

```
0   1   0
0   0   1
1   0   0
```

Question : calculez la puissance 1992 de cette matrice.
Réponse : cette matrice est une matrice de permutation qui échange les vecteurs e1, e2, e3 selon une permutation circulaire : e1-> e3 -> e2 -> e1 .

Permutons les colonnes de maMatrice selon la permutation circulaire (vers la droite) décrite ci-dessus:

```
(maMatricePermutee1=Transpose[
RotateRight[Transpose[maMatrice]]])//MatrixForm
```

```
0   0   1
1   0   0
0   1   0
```

Le résultat sera exactement le même si maMatrice est multipliée par elle-même, puisque c'est une matrice décrivant la permutation :

```
MatrixPower[maMatrice,2]//MatrixForm
```

```
0   0   1
1   0   0
0   1   0
```

Effectuons à nouveau une permutation vers la droite :

```
(maMatricePermutee2=Transpose[
RotateRight[Transpose[maMatricePermutee1]]])//MatrixForm
```

```
1   0   0
0   1   0
0   0   1
```

Cela revient à élever au cube maMatrice :

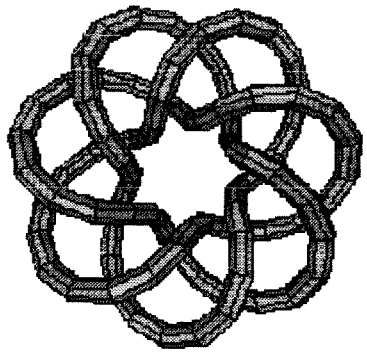
```
MatrixPower[maMatrice,3]//MatrixForm
```

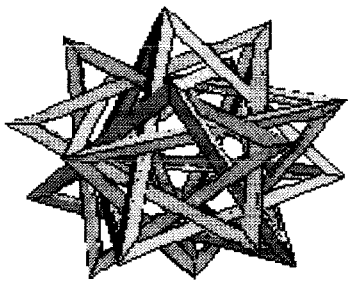
```
1   0   0
0   1   0
0   0   1
```

On obtient la matrice identité, ce qui est très intéressant puisque les puissances multiples de 3 de cette matrice seront égales à la matrice identité.
En particulier la puissance 1992. *Mathematica* fournit d'ailleurs un résultat immédiat :

```
MatrixPower[maMatrice,1992]//MatrixForm
```

```
1   0   0
0   1   0
0   0   1
```





Exemple 2

Calcul des déterminants d'ordre trois et quatre de matrices à éléments formels.

Ordre 3 :

Soit m une matrice 3 x 3 : *Mathematica* donne immédiatement le déterminant de cette matrice.

```
(m={{a,b,c},{d,e,f},{g,h,i}})//MatrixForm
```

```
a   b   c
d   e   f
g   h   i
```

```
Det[m]
```

```
-(c e g) + b f g + c d h - a f h - b d i + a e i
```

Ordre 4 :

On donne les matrices carrées d'ordre 4 : mat1, mat2 et mat3.
mat3 est-elle inversible ?

```
mat1={{a1,a1,a1,a1},{a2,a2,a2,a2},{a3,a3,a3,a3},{a4,a4,a4,a4}}//MatrixForm
```

a1	a1	a1	a1
a2	a2	a2	a2
a3	a3	a3	a3
a4	a4	a4	a4

```
mat2={{b1,b2,b3,b4},{b1,b2,b3,b4},{b1,b2,b3,b4},{b1,b2,b3,b4}}//MatrixForm
```

b1	b2	b3	b4
b1	b2	b3	b4
b1	b2	b3	b4
b1	b2	b3	b4

```
(mat3=mat1-mat2)//MatrixForm
```

a1 - b1	a1 - b2	a1 - b3	a1 - b4
a2 - b1	a2 - b2	a2 - b3	a2 - b4
a3 - b1	a3 - b2	a3 - b3	a3 - b4
a4 - b1	a4 - b2	a4 - b3	a4 - b4

```
Det[mat3]
```

```
0
```

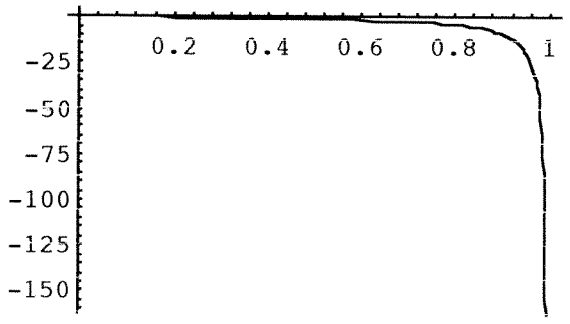
Le déterminant de mat3 est nul, donc elle n'est pas inversible.

La démonstration est simple : il suffit d'utiliser la propriété d'invariance du déterminant par l'ajout à l'une des lignes d'une combinaison linéaire des autres lignes ; on fait alors apparaître deux lignes multiple l'une de l'autre : les vecteurs-ligne associés sont liés, ce qui entraîne la nullité du déterminant.

Exemple 3

primitive de 1/ln(x), sur] 0 ; 1 [? primitive de 1/ x ln(x) ? sur quel intervalle ?

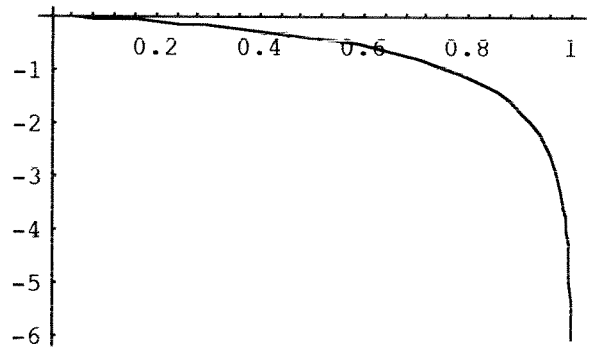
```
Plot[1/Log[x],{x,0,1}];
```




```
Integrate[1/Log[x],x]
```

```
LogIntegral[x]
```

```
Plot[LogIntegral[x],{x,0,1}];
```

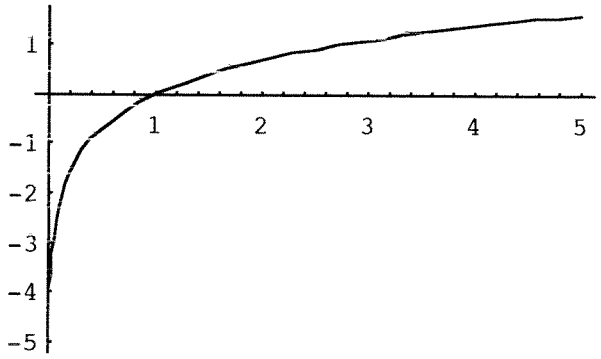


Primitive de 1 / x lnx ? sur quel intervalle ?
Voici la réponse brutale de *Mathematica* :

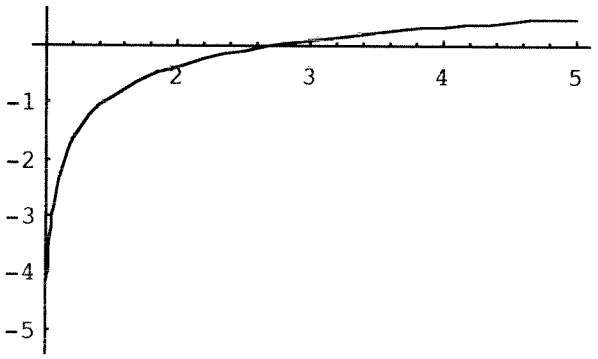
```
Integrate[1/(x Log[x]),x]

Log[Log[x]]
```

Visiblement, cette fonction "primitive" est définie dès que lnx > 0 . Donc sur] 1 ; +∞ [. Ce qu'illustrent bien les diverses représentations graphiques ci-dessous.

```
Plot[Log[x],{x,0,5}];
```

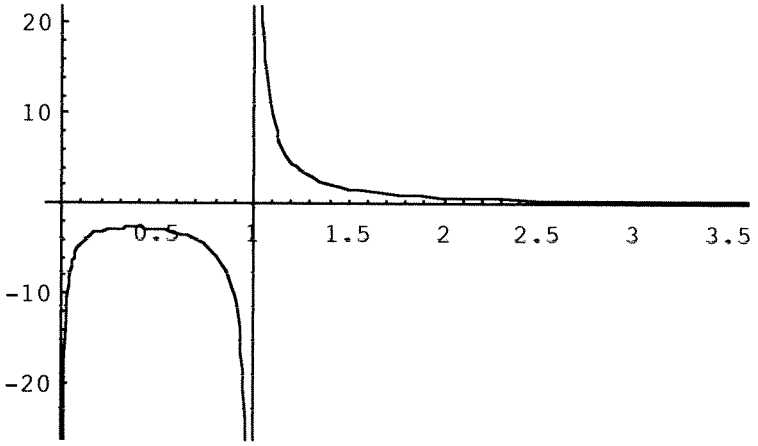



```
Plot[Log[Log[x]],{x,1,5}];
```



Cela tendrait à montrer que la fonction 1/ x lnx n'admet de primitive que sur] 1 ; +∞[. Or, considérons la représentation graphique de 1 / x lnx sur] 0 ; 1 [U] 1 ; +∞ [:

```
Plot[1/(x Log[x]),{x,0,5}];
```



Cette fonction est continue sur l'intervalle] 0 ; 1[, donc elle admet une primitive sur cet intervalle, qui **ne peut pas** être ln(lnx). La formule classique de la primitive de 1/ x lnx est ln | lnx |. *Mathematica* ne donne pas ln | lnx | comme primitive de 1 / x lnx. Donc, **méfiance**.

Même si le logiciel fournit de nombreux résultats, il est encore nécessaire de savoir un peu de mathématiques !

■ *Exemple 4*

Matrices 3 x 3 <u>orthogonales circulantes</u>.

```
(circul={{a,b,c},{c,a,b},{b,c,a}})//MatrixForm
```

$$\begin{matrix} a & b & c \\ c & a & b \\ b & c & a \end{matrix}$$

```
Transpose[circul].circul//MatrixForm
```

$$\begin{matrix} a^2 + b^2 + c^2 & a\,b + a\,c + b\,c & a\,b + a\,c + b\,c \\ a\,b + a\,c + b\,c & a^2 + b^2 + c^2 & a\,b + a\,c + b\,c \\ a\,b + a\,c + b\,c & a\,b + a\,c + b\,c & a^2 + b^2 + c^2 \end{matrix}$$

La matrice "circul" étant orthogonale, on a ${}^t\text{circul}.\text{circul} = \text{Id}_3$. Ce qui implique $a^2 + b^2 + c^2 = 1$ et $ab + ac + bc = 0$. Or $(a + b + c)2 = a^2 + b^2 + c^2 + 2\,(ab + ac + bc)$.

```
Expand[(a+b+c)^2]
```

$$a^2 + 2\,a\,b + b^2 + 2\,a\,c + 2\,b\,c + c^2$$

On voit donc que $(a+b+c)^2 = 1$. Donc **<u>a+b+c = ±1</u>**.

Exemple 5

Plusieurs façons de programmer une factorielle ; entraînement à la programmation procédurale et non procédurale...

```
ClearAll["Global`*"]
Remove["Global`*"]
```

1) Programmation procédurale type Pascal

```
factorielle[1][n_]:=Module[{i=0,factorielle=1},
                           While[i<n, i=i+1;factorielle=factorielle
                           factorielle]

factorielle[1][10]
```

```
3628800
```

2) Version procédurale type C

```
factorielle[2][n_]:=Module[{i=0,factorielle=1},
                           While[i<n, ++i;factorielle*=i];factoriel

factorielle[2][10]
```

```
3628800
```

3) Version récursive avec "empilement-dépilement"

```
factorielle[3][0]=1;
factorielle[3][n_]:=n factorielle[3][n-1];

factorielle[3][10]
```

```
3628800
```

4) Version récursive "sans dépilement"

```
fact[1,k_]:= k;
fact[n_,k_]:=fact[n-1,n k];
factorielle[4][n_]:=fact[n,1];

factorielle[4][10]
```

```
3628800
```

5) Utilisation d'une fonction Mathematica : Product

```
factorielle[5][0]=1;
factorielle[5][n_]:=Product[i,{i,1,n}];

factorielle[5][10]
```

```
3628800
```

■ *6) Utilisation de l'opérateur multiplicatif sur la liste des n premiers entiers (non nuls). Fonctions "Times" et "Range[n]"*

```
factorielle[6][n_]:= Apply[Times,Range[n]];

Range[10]

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

factorielle[6][10]

3628800
```

■ *7) Version compilée*

```
factorielle[7]=Compile[{{n,_Integer}},Module[{i=0,factorielle=1},
                              While[i<n, ++i;factorielle*=i];factoriel

CompiledFunction[{n}, Module[{i = 0, factorielle = 1},
   While[i < n, ++i; factorielle *= i]; factorielle],
  -CompiledCode-]

factorielle[7][10]

3628800
```

■ *8) Primitive "!"de Mathematica : la plus rapide à taper...*

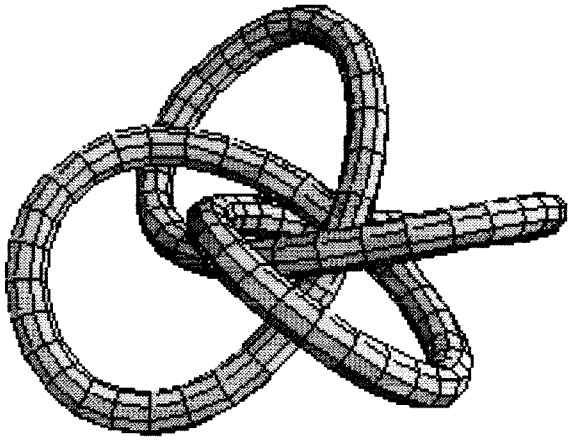
```
factorielle[8][n_]:=n!

factorielle[8][10]

3628800
```

Vérification et tests de rapidité

```
tableDesFactorielles=Table[factorielle[i],{i,1,8}];
Through[tableDesFactorielles[15]]

{1307674368000, 1307674368000, 1307674368000,
  1307674368000, 1307674368000, 1307674368000,
  1307674368000, 1307674368000}

Equal[Through[tableDesFactorielles[15]]]

True
```

```
Table[{i,First[N[Timing[factorielle[i][200]],15]]},{i,1,8}]//TableForm
```

```
1   0.149999999999636 Second
2   0.16666666666697 Second
3   0.116666666665878 Second
4   0.133333333334122 Second
5   0.0166666666664241 Second
6   0. Second
7   0.149999999999636 Second
8   0. Second
```



■ *Exemple 6*

Suite de Fibonacci : Différentes programmations.

■ *Itérative, avec notation matricielle [d'après R. Amalberti]*

```
f[x_]:= {{1,1},{1,0}}.x;
fib[1][n_]:=Part[Nest[f,{1,0}, n],2];
```

```
fib[1][200]
```

```
280571172992510140037611932413038677189525
```

En utilisant une fonction pure, on gagne une ligne, et quelques centièmes de seconde :

```
fib[2][n_]:=Nest[({{1,1},{1,0}}.#)&,{1,0}, n][[2]];
```

```
fib[2][200]
```

```
280571172992510140037611932413038677189525
```

```
Timing[fib[1][200]]
Timing[fib[2][200]]
```

```
{0.533333 Second, 280571172992510140037611932413038677189\
   525}
```

```
{0.483333 Second, 280571172992510140037611932413038677189\
   525}
```

■ *Récursive*

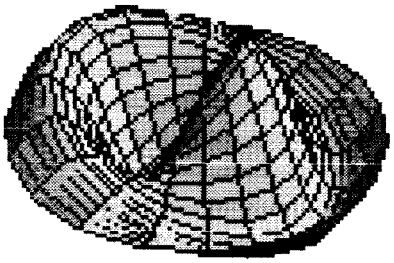
```
fib[3][0]:={1,0};
fib[3][n_]:={{1,1},{1,0}}.fib[3][n-1];
```

```
fib[3][200][[1,1,1]]
```

```
280571172992510140037611932413038677189525
```

```
Timing[fib[3][200][[1,1,1]]]
```

```
{0.2 Second, 280571172992510140037611932413038677189525}
```



Valeurs propres et vecteurs propres

On illustre la continuité et la nature des valeurs propres par rapport à un paramètre dans une matrice.

In[1]:=

```
A={{4,-2},{3,-5}};
MatrixForm[A]
```

Out[2]//MatrixForm=

```
4    -2

3    -5
```

On détermine l'ensemble des valeurs propres de la matice A à l'aide de la fonction de Mathematica : Eigenvalues[A]. Des vecteurs propres sont obtenus à l'aide de la fonction : Eigenvectors[A].

In[3]:=

```
Eigenvalues[A]
```

Out[3]=

```
  -1 - Sqrt[57]  -1 + Sqrt[57]
{-------------, -------------}
        2              2
```

In[4]:=

```
Eigenvectors[A]
```

Out[4]=

```
    9 - Sqrt[57]          9 + Sqrt[57]
{{-------------, 1}, {-------------, 1}}
        6                     6
```

In[5]:=

```
B={{4,-2},{3,5}};
MatrixForm[B]
```

Out[6]//MatrixForm=

```
4    -2

3    5
```

In[7]:=

```
Eigenvalues[B]
```

Out[7]=

```
  9 - I Sqrt[23]  9 + I Sqrt[23]
{--------------, --------------}
        2               2
```

Les valeurs propres sont en fait données dans le corps des complexes.

Etudions la continuité des valeurs propres par rapport à un paramètre dans une matrice 2x2 donnée.

In[8]:=

M={{4,-2},{3,p-5}};
MatrixForm[M]

Out[9]//MatrixForm=

```
4          -2

3          -5 + p
```

In[10]:=

vp=Eigenvalues[M]

Out[10]=

$$\left\{\frac{-1+p-\sqrt{(1-p)^2-4(-14+4p)}}{2},\ \frac{-1+p+\sqrt{(1-p)^2-4(-14+4p)}}{2}\right\}$$

On examine la partie imaginaire de la première valeur propre.

In[11]:=

vp1=First[vp]

Out[11]=

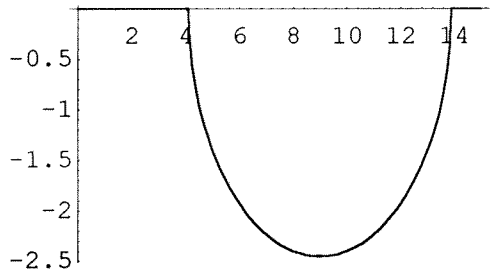
$$\frac{-1+p-\sqrt{(1-p)^2-4(-14+4p)}}{2}$$

In[12]:=

ff[x_]:=vp1/.p->x

In[13]:=

Plot[Im[ff[x]],{x,0,15}]



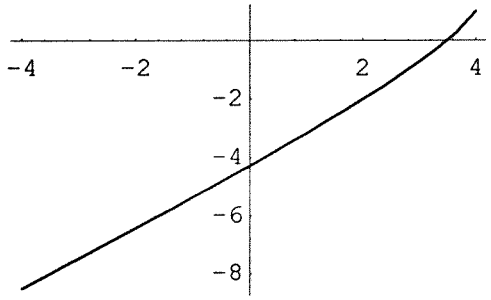
Out[13]=

```
-Graphics-
```

Les valeurs propres sont imaginaires d'après le dessin de p=4 à p=14 (à peu près).
On déterminera les valeurs précises plus loin.

In[14]:=

Plot[ff[x],{x,-4,4}]



Out[14]=

```
-Graphics-
```

D'après le dessin, les valeurs propres sont continues par rapport à p pour p<4.

Déterminons à partir du polyôme caractéristique les valeurs exactes du saut entre valeurs propres réelles et valeurs propres complexes.

In[15]:=

P[x_]:=Det[M-x IdentityMatrix[2]]

In[16]:=

P[x]

Out[16]=

```
                          2
-14 + 4 p + x - p x + x
```

In[17]:=

Collect[P[x],x]

Out[17]=

```
                            2
-14 + 4 p + (1 - p) x + x
```

In[18]:=

Delta[P_]:=Coefficient[P,x]^2-4 Coefficient[P,x,2]* Coefficient[P,x,0]

In[19]:=

Delta[P[x]]

Out[19]=

```
          2
(1 - p)  - 4 (-14 + 4 p)
```

In[20]:=

DeltaC[x_]:=Delta[P[x]]/.p->x

In[21]:=

DeltaC[x]

Out[21]=

```
          2
(1 - x)  - 4 (-14 + 4 x)
```

In[22]:=

S=Solve[DeltaC[x]==0,x]

Out[22]=

```
          18 - 4 Sqrt[6]          18 + 4 Sqrt[6]
{{x -> --------------}, {x -> --------------}}
                2                       2
```

In[23]:=

N[S]

Out[23]=

```
{{x -> 4.10102}, {x -> 13.899}}
```

On a les valeurs exactes (out[20]) de p pour le bord de l'intervalle où les valeurs propres sont complexes. Il s'agit des valeurs de p où la valeur propre est double.

Examinons la matrice dans ces cas.

In[24]:=

p1=x/.S[[1]]

Out[24]=

```
18 - 4 Sqrt[6]
--------------
       2
```

In[25]:=

MatrixForm[m=M/.p->p1]

Out[25]//MatrixForm=

```
4                          -2

                                18 - 4 Sqrt[6]
--                         -5 + --------------
3                                      2
```

In[26]:=

Eigenvalues[m]

Out[26]=

```
 8 - 2 Sqrt[6]  8 - 2 Sqrt[6]
{-------------, -------------}
       2              2
```

In[27]:=

Eigenvectors[m]

Out[27]=

```
        2
{{Sqrt[-], 1}, {0, 0}}
        3
```

In[28]:=

? Eigensystem

```
Eigensystem[m] gives a list {values, vectors} of the
   eigenvalues and eigenvectors of the square matrix m.
```

Eigensytem[] donne la liste des valeurs propres et des vecteurs propres d'une matrice carré.

In[29]:=

Eigensystem[m]

Out[29]=

```
                                                    2
{{4 - Sqrt[6], 4 - Sqrt[6]}, {{Sqrt[-], 1}, {0, 0}}}
                                                    3
```

Il y a une valeur propre double avec un espace propre de dimension 1.

Revenons à la matrice M et examinons pour quelles valeurs de p les deux vecteurs propres sont identiques.

In[30]:=

V=Eigenvectors[M]

Out[30]=

```
                                    2
    9 - p - Sqrt[57 - 18 p + p ]
{{{----------------------------, 1},
                  6

                                    2
    9 - p + Sqrt[57 - 18 p + p ]
 {{----------------------------, 1}}
                  6
```

In[31]:=

eq= V[[1]][[1]]==V[[2]][[1]]

Out[31]=

```
                                  2                                    2
  9 - p - Sqrt[57 - 18 p + p ]      9 - p + Sqrt[57 - 18 p + p ]
  ----------------------------- == -----------------------------
                 6                                  6
```

In[32]:=

Solve[eq,p]

Out[32]=

```
          18 - 4 Sqrt[6]           18 + 4 Sqrt[6]
{{p -> --------------}, {p -> --------------}}
                2                        2
```

Conclusion : Il s'agit des valeurs pour lesquelles la valeur propre est double.

In[33]:=

N[{(18-4 Sqrt[6])/2,(18+4Sqrt[6])/2}]

Out[33]=

```
{4.10102, 13.899}
```

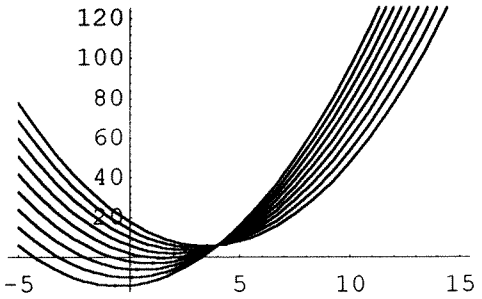
Représentation graphique des polynômes caractéristiques suivant p:

In[34]:=

T=Table[P[x]/.p->i,{i,0,8}];

In[35]:=

```
Plot[Evaluate[T],{x,-5,15}]
```



Out[35]=

```
-Graphics-
```

Jordanisation des matrices

On se propose d'illustrer l'algorithme de jordanisation d'une matrice carrée sur une matrice 5x5.

▩ 1ère étape : on rentre la matrice.

In[1]:=

A={{-5,-4,0,6,-3},{7,6,0,-6,3},{0,0,-1,1,0},{0,0,0,-1,0},{5,5,0,-4,2}}

Out[1]=

```
{{-5, -4, 0, 6, -3}, {7, 6, 0, -6, 3}, {0, 0, -1, 1, 0},

 {0, 0, 0, -1, 0}, {5, 5, 0, -4, 2}}
```

In[2]:=

MatrixForm[%]

Out[2]//MatrixForm=

```
-5   -4   0    6    -3

7    6    0    -6   3

0    0    -1   1    0

0    0    0    -1   0

5    5    0    -4   2
```

▩ 2 éme étape : détermination des valeurs propres

on calcule les valeurs propres à l'aide de la fonction de Mathematica "Eigenvalues".

In[3]:=

Eigenvalues[A]

Out[3]=

```
{-1, -1, -1, 2, 2}
```

On peut vérifier à l'aide de la factorisation du polynôme caractéristique

□ Polynôme caractéristique

In[4]:=

d=Det[A-t IdentityMatrix[5]]

Out[4]=

```
                2      3    4    5
-4 - 8 t - t  + 5 t  + t  - t
```

In[5]:=

Factor[d]

Out[5]=

```
              2         3
-((-2 + t)  (1 + t) )
```

Il y a deux valeurs propres : 2 avec une multiplicité 2 et -1 avec une multiplicité 3.

▩ 3 éme étape : Espaces caractéristiques

■ Espace caractéristique associé à la valeur propre -1.

On cherche les noyaux des opérateurs (A+ Id)^p avec p=1,2,3.

In[6]:=

MatrixForm[n1=A+1 IdentityMatrix[5]]

Out[6]//MatrixForm=

```
-4    -4    0    6    -3

7     7     0    -6   3

0     0     0    1    0

0     0     0    0    0

5     5     0    -4   3
```

In[7]:=

MatrixForm[n2=n1.n1]

Out[7]//MatrixForm=

```
-27   -27   0    12   -9

36    36    0    -12  9

0     0     0    0    0

0     0     0    0    0

30    30    0    -12  9
```

In[8]:=

MatrixForm[n3=n2.n1]

Out[8]//MatrixForm=

```
-126   -126   0    36   -27

153    153    0    -36  27

0      0      0    0    0

0      0      0    0    0

135    135    0    -36  27
```

In[9]:=

v={a,b,c,e,f};

In[10]:=

Solve[n3.v==0,{a,b,c,e,f}]

Out[10]=

```
                    3 f
{{a -> -b, e -> ---}}
                    4
```

In[11]:=

v/.%

Out[11]=

```
                  3  f
{{-b, b, c, ---, f}}
                  4
```

In[12]:=

v/.Solve[n2.v==0]

Out[12]=

```
                  3  f
{{-b, b, c, ---, f}}
                  4
```

In[13]:=

v/.Solve[n1.v==0]

Out[13]=

```
{{-b, b, c, 0, 0}}
```

In[14]:=

v3={0,0,0,3,4};
v2=n1.v3

Out[15]=

```
{6, -6, 3, 0, 0}
```

In[16]:=

v1={0,0,1,0,0};

■ espace caractéristique associé à la valeur propre 2

On cherche les noyaux des opérateurs (A-2 Id)^p avec p=1,2.

In[17]:=

MatrixForm[m1=A-2 IdentityMatrix[5]]

Out[17]//MatrixForm=

```
-7    -4    0    6    -3

7     4     0    -6   3

0     0     -3   1    0

0     0     0    -3   0

5     5     0    -4   0
```

In[18]:=

MatrixForm[m2=m1.m1]

Out[18]//MatrixForm=

```
6      -3     0      -24    9

-6     3      0      24     -9

0      0      9      -6     0

0      0      0      9      0

0      0      0      12     0
```

In[19]:=

w={p,q,r,s,t};

In[20]:=

w/.Solve[m2.w==0,{p,q,r,s,t}]

Out[20]=

```
      q    3 t
{{- - - ---, q, 0, 0, t}}
      2    2
```

In[21]:=

w/.Solve[m1.w==0,{p,q,r,s,t}]

Out[21]=

```
{{-q, q, 0, 0, q}}
```

In[22]:=

w2={-3,0,0,0,2};
w1=m1.w2

Out[23]=

```
{15, -15, 0, 0, -15}
```

▩ 4 éme étape : la matrice de Jordan

In[24]:=

P={v1,v2,v3,w1,w2}

Out[24]=

```
{{0, 0, 1, 0, 0}, {6, -6, 3, 0, 0}, {0, 0, 0, 3, 4},

 {15, -15, 0, 0, -15}, {-3, 0, 0, 0, 2}}
```

In[25]:=

PP=Transpose[P]

Out[25]=

```
{{0, 6, 0, 15, -3}, {0, -6, 0, -15, 0}, {1, 3, 0, 0, 0},

 {0, 0, 3, 0, 0}, {0, 0, 4, -15, 2}}
```

In[26]:=

AA=Inverse[PP].A.PP

Out[26]=

```
{{-1, 0, 0, 0, 0}, {0, -1, 1, 0, 0}, {0, 0, -1, 0, 0},

 {0, 0, 0, 2, 1}, {0, 0, 0, 0, 2}}
```

In[27]:=

MatrixForm[AA]

Out[27]//MatrixForm=

$$\begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

DES TRAVAUX PRATIQUES SUR LES POLYNÔMES ORTHOGONAUX.

Utilisation de Dérive et/ou de Maple.

Jean Lefort

On trouvera ci-après le détail d'un T.P. sur les polynômes orthogonaux réalisé dans une classe de math-spé T.A. en introduction à l'étude des séries de Fourier. Ce T.P. a été donné plusieurs années consécutives. Il est à peu près réalisable en deux heures.
L'établissement possédant une licence d'exploitation du logiciel Dérive (version 2.57) les élèves sont entraînés à l'utilisation de cet instrument. Cependant pour être sûr que l'obstacle à la compréhension n'est pas du à la méconnaissance du logiciel je distribue aux étudiants les pages 5 et 6 intitulées "TP à l'aide de DÉRIVE sur les polynômes orthogonaux" qui leur rappellent les commandes à utiliser et me permet éventuellement d'intervenir globalement et non individuellement en donnant en référence le numéro de la ligne. Si certains n'ont pas suivi rigoureusement le plan indiqué, il leur est facile de mettre en regard le numéro de ligne de leur écran.
Le texte des pages 2, 3 et 4 est un résumé des explications que je donne. Étant donné la configuration des lieux, une salle d'ordinateurs très exiguë mais contiguë à la salle de cours, je passe alternativement d'une salle à l'autre selon qu'il s'agit de faire un travail sur papier ou au tableau ou bien s'il s'agit d'utiliser les machines. En l'absence d'imprimante et de façon que les étudiants aient néanmoins une trace des représentations graphiques, je distribue vers la fin du T.P. (après exécution des lignes 11, 28 et 29), la page 7 que j'ai obtenu à l'aide du logiciel MAPLE.
Dans un but de diffusion de ce T.P., j'ai ajouté, pour les collègues qui peuvent disposer dans leur établissement de suffisamment de copie de MAPLE, les pages 8, 9 et 10 qui permettent de traiter le même sujet à partir de ce logiciel dont l'intérêt évident est d'inclure une programmation structurée.

Une comparaison rapide des deux logiciels montre la supériorité de DERIVE en ce qui concerne l'accessibilité : pas besoin d'apprendre une liste de commande en anglais (1/3 de mes étudiants n'ont jamais fait d'anglais) et en deux heures de temps on sait faire l'essentiel. Par contre MAPLE possède une programmation structurée très efficace et très courte : une seule commande dont la plus part des parties sont optionnelles permet de traiter presque tous les cas. En ce qui concerne le traitement de texte, ni l'un ni l'autre ne sont vraiment satisfaisant bien que l'avantage soit incontestablement à MAPLE. Mais c'est essentiellement un critère économique qui fait choisir un logiciel plutôt que l'autre : DERIVE revient à 3000 F en licence sur site tandis que MAPLE revient à 3000 F par poste ! De plus MAPLE ne tourne pas sur des appareils trop anciens et l'on sait que l'Éducation Nationale est très malthusienne en ce qui concerne la mise à jour du matériel (toujours une question de coût).

LES POLYNÔMES ORTHOGONAUX
TRAVAUX PRATIQUES

Préliminaires :

Le but de ce TP est de montrer comment obtenir de "bonnes" approximations sous forme polynomiale de certaines fonctions. Le problème étant de savoir ce que l'on appelle "bonnes" approximations ? Si nous prenons la fonction *sinus* nous connaissons les approximations polynomiales que sont les parties régulières des développements limités. Mais on sait que les développements limités sont surtout utilisés pour le calcul des limites, ce qui sous-entend que les approximations obtenues ne sont bonnes qu'au voisinage de 0. On pourrait penser que comme la série entière associée a un rayon de convergence infinie les approximations ainsi obtenues sont "bonnes" sur n'importe quel intervalle. Nous allons voir que nous pouvons faire beaucoup mieux (en un sens que nous préciserons) avec des polynômes de même degré. Une des raisons qui fait que l'on peut améliorer la précision de l'approximation est que la série entière ne converge pas uniformément sur $\mathbf{R}$. En pratique, plus on s'éloigne de l'origine moins bonne est la précision obtenue en remplaçant le *sinus* par son développement limité.

1ère étape :

Démontrer que $\langle f(x) / g(x) \rangle = \int_{-1}^{+1} f(x).g(x)\,dx$ est un produit scalaire sur l'espace vectoriel des fonctions C^0 sur $[-1 ; +1]$.
Remarquons que la norme associée à ce produit scalaire est la norme quadratique. Pour une telle norme, une fonction sera d'autant plus voisine de la fonction nulle que la quantité $\int_{-1}^{+1} f^2(x)dx$ sera petite. On fait donc une moyenne sur l'intervalle $[-1 ; +1]$ de l'écart quadratique à 0 .
Nous allons construire une famille orthogonale de polynômes pour ce produit scalaire. On cherche alors la projection de la fonction définie par $\sin(\pi x)$ sur une partie de cette famille. On peut alors se poser la question de savoir si, quand la partie augmente indéfiniment, le projeté tend ou non vers la fonction donnée et si oui de quelle façon.

2ème étape :

Nous cherchons une famille orthogonale de polynômes pour ce produit scalaire. Le procédé d'orthonormalisation de Schmidt étant assez fastidieux, nous nous limiterons à une famille orthogonale (que nous pourrons toujours normaliser par la suite). Nous allons seulement vérifier que les $P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}\left[(x^2-1)^n\right]$ forment une famille orthonormale. (La famille des polynômes de Legendre). Démontrer les résultats suivants :
a) $P_n(x)$ est un polynôme de degré n et de même parité que n .

Calcul à l'aide de DÉRIVE pour trouver $P_n(x)$ pour $n \leq 5$.
Représentation graphique des P_n pour $n \leq 5$.

On admettra que $\forall x \in [-1 ; +1]$ $\|P_n(x)\| \leq 1$ et que $P_n(1) = 1$.

b) Calcul de $\int_{-1}^{+1} P_n(x).P_m(x).dx$. On suppose n ≤ m , on dérive P_n n fois et on intègre P_m m fois (intégration par parties). On trouve 0 si $n \neq m$ et $\frac{(-1)^n}{(2^n n!)^2}\int_{-1}^{+1}(2n)!(x^2-1)^n dx = \|P_n\|^2$ si $n = m$.

3ème étape :

On se propose de calculer $I_m = \int_{-1}^{+1}(x^2-1)^m dx$.

$$I_m = \int_{-1}^{+1}(x^2-1)^{m-1}(x^2-1)dx = \int_{-1}^{+1}x[(x^2-1)^{m-1}.x]dx - I_{m-1} = \left[\frac{1}{2m}x(x^2-1)^m\right]_{-1}^{+1} - \frac{1}{2m}I_m - I_{m-1}$$

On obtient finalement $I_m = (-2m / (2m+1))\, I_{m-1}$ et comme $I_0 = 2$ on a $I_m = (-1)^m \frac{2(2^m m!)^2}{(2m+1)!}$ ce qui conduit à $\|P_n\|^2 = 2 / (2n + 1)$.

4ème étape :

On se propose de calculer la projection de $(\sin \pi x)$ sur la famille $\{P_i\}$ i variant de 1 à r :

$$p_r(\sin \pi x) = \sum_{n=0}^{r}\left[\frac{1}{\|P_n\|^2}\int_{-1}^{+1}P_n(x).\sin \pi x\, dx\right]P_n(x) .$$

Calcul à l'aide de DÉRIVE pour trouver :
$<P_n(x) / \sin \pi x>$ pour $n \leq 5$;
$p_r(\sin px)$ pour $r \leq 5$;
comparer graphiquement les différentes projections avec $\sin \pi x$.

5ème étape :

Étude de la convergence de $p_r(\sin \pi x)$ vers $\sin \pi x$ quand r tend vers l'infini. Les graphiques obtenus semblent indiquer qu'il y a convergence. De quelle nature ?

a) Majorons tout d'abord le coefficient de $P_n(x)$ dans l'expression de $p_r(\sin px)$:

$\int_{-1}^{+1}P_n(x)\sin \pi x dx = \int_{-1}^{+1}\frac{1}{2^n n!}\frac{d^n}{dx^n}\left[(x^2-1)^n\right]\sin \pi x dx$ on intègre n fois la dérivée $n^{ième}$ et on dérive n fois le sinus dans des intégrations par parties successives. On obtient : $\frac{\pi^n}{2^n n!}\int_{-1}^{+1}(x^2-1)^n t_n(x)dx$ où $t_n(x) = (-1)^n \sin(\pi x + n\frac{\pi}{2})$. L'expression intégrale admet la majoration évidente en module : $\frac{\pi^n}{2^n n!}I_n = \frac{2^{n+1}\pi^n n!}{(2n+1)!}$. Comme $|P_n(x)| \leq 1$ sur $[-1 ; +1]$, on voit que la valeur absolue du terme général de $p_r(\sin \pi x)$ est majorée par le terme général d'une série numérique convergente puisque son terme général $\frac{1}{\|P_n\|^2}\frac{2^{n+1}\pi^n n!}{(2n+1)!} \sim 2e\left(\frac{\pi e}{2n+1}\right)^n$ en utilisant la formule de Stirling. Il y a donc convergence normale vers $\sin(\pi x)$.

b) On peut écrire $p_r(\sin \pi x) = \sum_{i=1}^{r} a_i \frac{P_i(x)}{\|P_i\|}$ avec $a_i = <\frac{P_i(x)}{\|P_i\|} / \sin \pi x >$.

Étude avec DÉRIVE de Σa_i^2 pour $r = 1$; 3 et 5 .

On devine la convergence vers 1 qui est assez naturelle puisque $\|\sin(px)\| = 1$.

Conclusion.

Si on en a le temps, il est possible d'ajouter le calcul de la norme (ou du carré de la norme) du développement limité à l'ordre 5 de la fonction $\sin(\pi x)$. Cela est immédiat, et la comparaison avec les tracés effectués dans l'introduction, permettent de faire comprendre aux élèves l'intérêt des polynômes orthogonaux de Legendre.

Tracer à l'aide de DERIVE les courbes des fonctions
$\sin(\pi x)$; $\pi x - \pi^3 x^3/6$; $\pi x - \pi^3 x^3/6 + \pi^5 x^5/5!$
Calculer le carré de la norme de $\pi x - \pi^3 x^3/6 + \pi^5 x^5/5!$

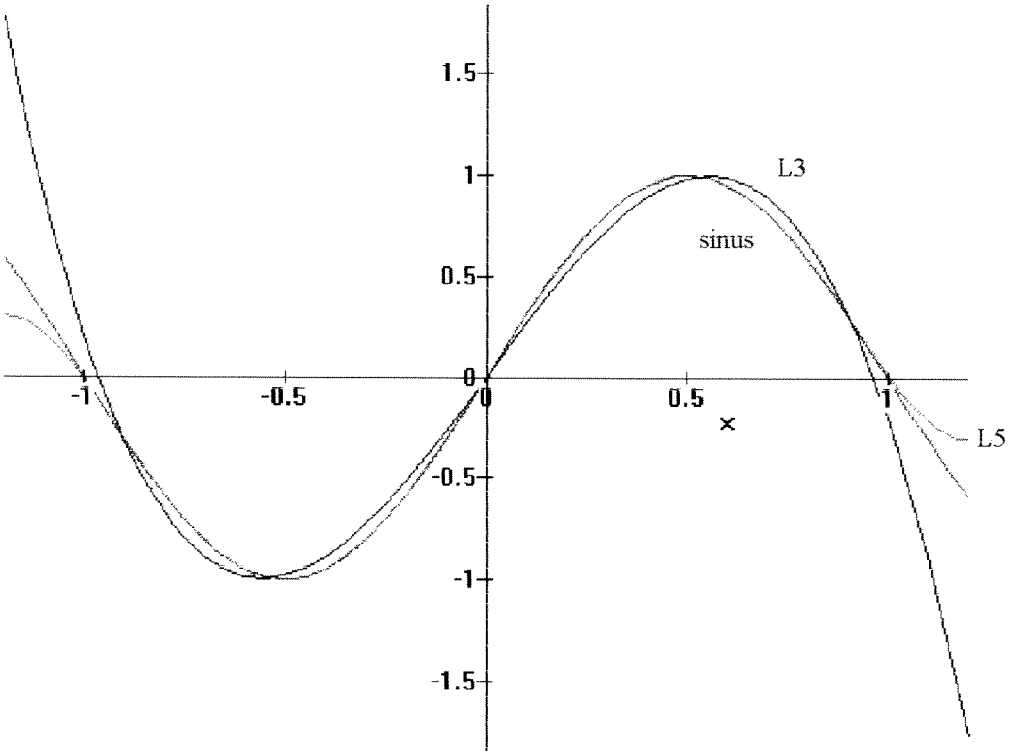
Les étudiants disposent des deux pages préparées suivantes. Cela leur permet de noter les résultats au fur et à mesure et, pour ceux qui sont moins bien familiarisés avec le logiciel DERIVE, de mieux suivre les diverses étapes du calcul sur machine ; pour les autres, cela permet au professeur de faire référence aux diverses formules par un numéro.
Pour les courbes, la visualisation ne pose pas de problème, mais faute d'imprimante correcte dans l'établissement, il a fallu se contenter de les observer. J'ai ensuite distribué des photocopies de ces courbes.
Remarque : Nous avons travailler avec $\sin(\pi x)$ pour avoir une période complète tout en utilisant le produit scalaire de Legendre et ses polynômes. On peut faire le même travail avec $\sin(x)$ et un produit scalaire modifié en intégrant sur $[-\pi\,;+\pi]$. Les difficultés sont analogues mais pas toujours placées aux mêmes endroits.

commande	affichage	n°
A		**1**
C - D - ordre n		**2**
A	n := 2	**3**
E #2		**4**
A	n := 3	**5**
E #2		**6**
A	n := 4	**7**
E #2		**8**
A	n := 5	**9**
E #2		**10**
A	n := n	**11**
Graphiques de #4, #6, #8 et #10 utiliser le Zoom pour avoir $-1 \leq x \leq 1$ On peut superposer ou non Supprimer les graphes à la fin.		
A #2 sin πx		**12**
C - I - de −1 à +1		**13**
A	n := 1	**14**
E #13		**15**
A	n := 3	**16**
E #13		**17**
A	n := 5	**18**
E #13		**19**
A	n := n	**20**

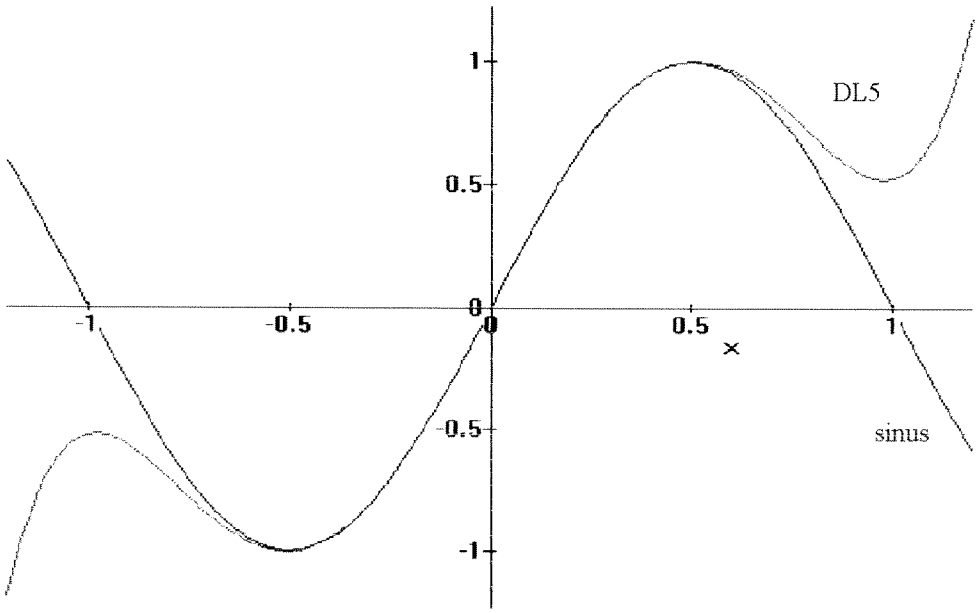
A (3/2) #15 x		**21**
X		**22**
A #21 + (7/2) #17 #6		**23**
E		**24**
X		**25**
A #23 + (11/2) #19 #10		**26**
E		**27**
X		**28**
Graphiques #23 et #26		
A	$\sin(\pi x)$	**29**
Graphique #29 superposé aux précédents		

A (3/2) #15 ^2		**30**
X		**31**
A #30 + (7/2) #17 ^2		**32**
X		**33**
A #32 + (11/2) #19 ^2		**34**
X		**35**

Courbes de la fonction sinus et de ses approximations de Legendre d'ordre 3 et 5 .



Courbes de la fonction sinus et de son développement limité à l'ordre 5.



Calcul des polynômes de Legendre.
(Jean Lefort)

On se propose de calculer les polynômes de Legendre successifs à partir de la formule Pn(x)= 1/2^n/n!*Qn(x) où Qn(x) est la dérivée n-ième de (x^2-1)^n .
On utilise pour cela l'opérateur D appliquée à la fonction f := (x,n) -> 1/2^n/n!*(x^2-1)^n .

```
> f:=(x,n) -> 1/2^n/n!*(x^2-1)^n ;
```

$$f := (x, n) \rightarrow \frac{\left(x^2 - 1\right)^n}{2^n\, n!}$$

Nous allons dériver n fois par rapport à la première variable.

```
> p:= (D[1]@@n)(f);
```

$$p := D_{[1]}^{(n)}(f)$$

```
> n:=2: normal(value(p)(x,n));
```

$$\frac{3}{2}x^2 - \frac{1}{2}$$

Si nous voulons avoir une liste des polynômes de Legendre pour diverses valeurs de n , il vaut mieux écrire une boucle :

```
> n:='n':

> for n from 1 to 5 do l(n):= normal(value(p)(x,n)) od;
```

$$l(1) := x$$

$$l(2) := \frac{3}{2}x^2 - \frac{1}{2}$$

$$l(3) := \frac{5}{2}x^3 - \frac{3}{2}x$$

$$l(4) := \frac{35}{8}x^4 - \frac{15}{4}x^2 + \frac{3}{8}$$

$$l(5) := \frac{63}{8}x^5 - \frac{35}{4}x^3 + \frac{15}{8}x$$

Nous pouvons ainsi récupérer ces différents polynômes en les appelants l(n). Utilisons les pour calculer leur norme au sens du produit scalaire de Legendre.

```
> for n from 1 to 5 do N:= int((l(n))^2,x=-1..1) od;
```

$$N := \frac{2}{3}$$

$$N := \frac{2}{5}$$

$$N := \frac{2}{7}$$

$$N := \frac{2}{9}$$

$$N := \frac{2}{11}$$

Puis projetons sin(Pi*x) sur cette famille de polynôme.

```
> for n from 1 to 5 do s(n) := 1/int((l(n))^2,x=-1..1)*int(l(n)*sin(Pi*x),x=-1..1)*l(n) od;
```

$$s(1) := 3\,\frac{x}{\pi}$$

$$s(2) := 0$$

$$s(3) := 7\,\frac{\left(-15+\pi^2\right)\left(\frac{5}{2}x^3-\frac{3}{2}x\right)}{\pi^3}$$

$$s(4) := 0$$

$$s(5) := 11\,\frac{\left(-105\,\pi^2+945+\pi^4\right)\left(\frac{63}{8}x^5-\frac{35}{4}x^3+\frac{15}{8}x\right)}{\pi^5}$$

L'approximation d'ordre 1 est donnée par s(1), celle d'ordre 3 par s(1) + s(3) et celle d'ordre 5 par s(1) + s(3) + s(5).
Les expression ne sont pas très agréable, nous les simplifions et donnons une valeur approchée :

```
> value(s(1)+s(3));evalf(");
```

$$3\,\frac{x}{\pi}+7\,\frac{\left(-15+\pi^2\right)\left(\frac{5}{2}x^3-\frac{3}{2}x\right)}{\pi^3}$$

$$2.692292524\,x-2.895604777\,x^3$$

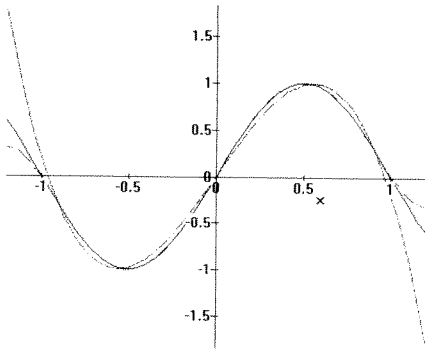
```
> value(s(1)+s(3)+s(5));evalf(");
```

$$3\,\frac{x}{\pi}+7\,\frac{\left(-15+\pi^2\right)\left(\frac{5}{2}x^3-\frac{3}{2}x\right)}{\pi^3}+11\,\frac{\left(-105\,\pi^2+945+\pi^4\right)\left(\frac{63}{8}x^5-\frac{35}{4}x^3+\frac{15}{8}x\right)}{\pi^5}$$

$$3.103460436\,x-4.814388364\,x^3+1.726905227\,x^5$$

Représentons graphiquement ces deux dernières expressions et comparons les à sin(Pi*x). L'approximation du premier degré (peu intéressante) n'a pas été représentée.

```
> plot({sin(Pi*x),s(1)+s(3),s(1)+s(3)+s(5)},x=-1.2..1.2);
```



On remarque l'excellente précision de la dernière approximation sur une période.
Il nous reste à calculer la norme, ou plus exactement le carré de la norme des différentes fonctions qui interviennent :

```
> int((sin(Pi*x))^2,x=-1..1); #norme de la fonction "sin(Pi*x)"
```

1

> **int((s(1))^2,x=-1..1);evalf(''); #norme de l'approximation d'ordre 1**

$$6\,\frac{1}{\pi^2}$$

.6079271016

> **int((s(1)+s(3))^2,x=-1..1); evalf(''); #norme de l'approximation d'ordre 3**

$$10\,\frac{315+2\,\pi^4-42\,\pi^2}{\pi^6}$$

.9912197658

> **int((s(1)+s(3)+s(5))^2,x=-1..1); evalf(''); #norme de l'approximation d'ordre 5**

$$42\,\frac{-103950\,\pi^2+\pi^8-120\,\pi^6+467775+6840\,\pi^4}{\pi^{10}}$$

.9999630362

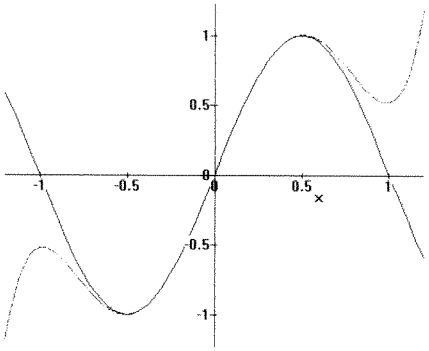
Comparons à la norme du développement limité d'ordre 5 ainsi qu'à la représentation graphique :

> **int((Pi*x-(Pi*x)^3/3!+(Pi*x)^5/5!)^2,x=-1..1);evalf('');**

$$-\frac{2}{15}\,\pi^4+\frac{1}{79200}\,\pi^{10}-\frac{1}{1620}\,\pi^8+\frac{4}{315}\,\pi^6+\frac{2}{3}\,\pi^2$$

1.125281130

> **plot({Pi*x-(Pi*x)^3/3!+(Pi*x)^5/5!,sin(Pi*x)},x=-1.2..1.2);**



DÉRIVE – MAPLE : 1 à 0

Jean Lefort

Chaque logiciel de calcul formel a ses avantages et ses inconvénients. Le calcul intégral et le calcul trigonométrique sont des exercices redoutables pour le calcul formel. Ce travail est un peu un pied de nez, mais il est aussi un moyen de montrer à des étudiants qu'il vaut mieux savoir où il faut chercher pour obtenir rapidement le bon résultat. Et pour savoir où chercher, il vaut mieux connaître ses formules !

Le problème posé tourne autour de la fonction $f(x) = \text{Arc}\sin\left(\frac{2x}{1+x^2}\right)$ et du calcul de $\int_0^{\sqrt{3}} \text{Arc}\sin\left(\frac{2x}{1+x^2}\right)dx$.

Nous allons commencer par faire une étude à la main puis voir ce que nous pouvons faire d'une part avec MAPLE (version V-2) puis avec DERIVE (version 2-57).

§ 1 - Étude du problème

Il y a plusieurs façon d'étudier la fonction $f(x) = \text{Arc}\sin\left(\frac{2x}{1+x^2}\right)$. Soit en reconnaissant dans l'argument de l'arcsinus une formule classique en trigonométrie, $\sin(2\theta) = \frac{2t}{1+t^2}$ où $t = \tan(\theta)$, ce qui permet d'écrire $f(x) =$ Arcsin(sin(2θ)), mais cela ne fait pas 2θ comme on le sait bien, mais 2θ ou $-2\theta \pm \pi$ selon l'intervalle où se trouve $\theta = \text{Arctan}(x)$. Utilisons plutôt la dérivation qui permet de raisonner directement sur x .

$f'(x) = \frac{1}{\sqrt{1-\frac{4x^2}{(1+x^2)^2}}} \frac{2(1+x^2)-4x^2}{(1+x^2)^2} = \frac{2(1-x^2)}{\sqrt{(1-x^2)^2}(1+x^2)}$ ce qui donne deux expressions selon que $|x| \le 1$ ou non. Dans le premier cas il vient $f'(x) = \frac{2}{1+x^2}$ et dans le deuxième $f'(x) = \frac{-2}{1+x^2}$. En intégrant pour revenir à la fonction f on ajuste la constante en comparant les valeurs pour $x = 0$, $x = 1$ ou $x = -1$. Il vient alors :

$$f(x) = \begin{cases} 2\text{Arc}\tan(x) & \text{pour } -1 \le x \le 1 \\ -2\text{Arc}\tan(x) + \pi & \text{pour } 1 \le x \\ -2\text{Arc}\tan(x) - \pi & \text{pour } x \le -1 \end{cases}$$

Si nous voulons maintenant chercher une primitive de f , celle qui s'annule en 0 , par exemple, il nous faudra distinguer trois cas et éventuellement découper l'intégrale par la relation de Chasles au point 1 ou -1 . Remarquons d'abord que $\int 2\ \text{Arctan}(x)dx = 2x\ \text{Arctan}(x) - \ln(1+x^2)$. Cela permet d'écrire :

$$\int_0^x f(t)dt = 2x\ \text{Arctan}(x) - \ln(1+x^2) \qquad \text{si } -1 \le x \le 1$$

$$\int_0^x f(t)dt = \int_0^1 f(t)dt + \int_1^x f(t)dt = -2x\ \text{Arctan}(x) + \ln(1+x^2) + \pi x - 2\ln(2) \qquad \text{si } 1 \le x$$

$$\int_0^x f(t)dt = \int_0^{-1} f(t)dt + \int_{-1}^x f(t)dt = -2x\ \text{Arctan}(x) + \ln(1+x^2) - \pi x - 2\ln(2) \qquad \text{si } x \le -1$$

Nous vérifions bien que la primitive ainsi obtenue est une fonction paire (c'est aussi une façon d'obtenir directement la dernière formule) et en particulier pour $x = \sqrt{3}$, la deuxième formule donne : $\int_0^{\sqrt{3}} \text{Arc}\sin\left(\frac{2x}{1+x^2}\right)dx = \frac{\pi\sqrt{3}}{3}$ dont une valeur approchée est 1, 8138 .

Il est également possible de calculer l'intégrale directement en utilisant une intégration par parties. Il est clair qu'il faut alors dériver l'arcsinus et que les difficultés de calcul sont du même ordre.

§ 2 - Étude avec MAPLE

Introduisons la fonction f :

> f(x):=arcsin(2*x/(1+x^2));

$$f(x) := \arcsin\left(2\frac{x}{1+x^2}\right)$$

Aucun essai de simplification ne donne quelque chose de nouveau, ainsi :

> simplify(f(x),trig);

$$f(x) := \arcsin\left(2\frac{x}{1+x^2}\right)$$

Cherchons la dérivée et simplifions la (notez au passage l'influence de la majuscule sur l'ordre **Diff / diff**) :

> Diff(f(x),x)=diff(f(x),x);

$$\frac{\partial}{\partial x}\arcsin\left(2\frac{x}{1+x^2}\right) = \frac{2\dfrac{1}{1+x^2} - 4\dfrac{x^2}{\left(1+x^2\right)^2}}{\sqrt{1-4\dfrac{x^2}{\left(1+x^2\right)^2}}}$$

> simplify(");

$$\frac{\partial}{\partial x}\arcsin\left(2\frac{x}{1+x^2}\right) = -2\frac{1}{1+x^2}$$

C'est là que nous nous rendons compte de l'erreur. MAPLE simplifie abusivement $\sqrt{(z^2)}$ en z . Or ici z vaut $(x^2 - 1)$ d'où l'apparition du signe moins ! On pourrait cependant penser que cette erreur dans le calcul de la dérivée n'influe pas sur le calcul de l'intégral de f . En fait si, car pour calculer la primitive il faudra bien passer par une simplification de radicaux au travers d'une intégration par parties.

> Int(f(x),x=0..sqrt(3))=simplify(int(f(x),x=0..sqrt(3)));

$$\int_0^{\sqrt{3}} \arcsin\left(2\frac{x}{1+x^2}\right) dx = \frac{1}{3}\pi\sqrt{3} + 2\ln(2)$$

Vérifions le en demandant la primitive de la fonction f.

> simplify(int(f(x),x));

$$x\arcsin\left(2\frac{x}{1+x^2}\right) + \ln\left(1+x^2\right)$$

Ce résultat ne correspond à rien de ce qui a été calculé à la main. D'après ce que nous avons vu, le logiciel pourrait simplifier le premier terme en $-2x$ Arctan(x) ce qui donnerait quelque chose qui ressemblerait vaguement au cas $|x| > 1$. Autant dire que nous ne sommes guère avancés ! Toutefois nous pouvons obtenir une valeur approchée, mais là encore attention à ce qu'on demande :

> evalf(int(f(x),x=0..sqrt(3)));

3.200093726

> evalf(Int(f(x),x=0..sqrt(3)));

1.813799364

Seule la deuxième formule est juste. Que s'est-il passé ? Dans le premier cas (avec le "i" minuscule) l'intégrale est d'abord calculée puis évaluée et c'est pour ça que le résultat est faux, tandis que dans le deuxième l'évaluation se fait directement par une méthode de calcul approché (genre simson ou mieux). Il est intéressant de noter cette différence que nous avions déjà utilisé ci-dessus : **Int(f(x),x=0..sqrt(3))=simplify(int(f(x),x=0..sqrt(3)));**

Finalement MAPLE ne nous donne pas grand chose si ce n'est la valeur apporchée de l'intégrale cherchée, mais ceci, de bonnes calculatrices programmables la donnent aussi. C'est finalement un peu maigre.

§ 3 - Étude avec DERIVE

Voici ce que donne la feuille de calcul DERIVE à laquelle nous avons incorporé des commentaires :

Introduisons la fonction f avec une commande **A**uteur :

1 $\quad \mathrm{ASIN}\left[\frac{2x}{1+x^2}\right]$

Utilisons la commande **S**implifie :

2 $\quad \mathrm{SIGN}(x^2-1)\ \mathrm{ATAN}\left[\frac{2x}{x^2-1}\right]$

Ce résultat est exact, mais ce n'est pas celui auquel nous nous attendions. Dérivons le par rapport à x à l'aide des commandes **C**alcul, **D**érive puis **S**implifions le résultat :

3 $\quad \frac{d}{dx}\left[\mathrm{SIGN}(x^2-1)\ \mathrm{ATAN}\left[\frac{2x}{x^2-1}\right]\right]$

4 $\quad -\frac{2\ \mathrm{SIGN}(x^2-1)}{x^2+1}$

En intégrant cette dernière expression entre 0 et x avec **C**alcul, **I**ntègre puis dév**E**loppe nous obtenons une forme équivalente à la fonction f initiale :

5 $\quad \int_0^x -\frac{2\ \mathrm{SIGN}(x^2-1)}{x^2+1}\,dx$

6 $\quad -2\ \mathrm{ATAN}(x)\ \mathrm{SIGN}(x^2-1)+\frac{\pi\ \mathrm{SIGN}(x^2-1)}{2}+\pi\ \mathrm{SIGN}(x+1)-\frac{\pi}{2}$

Ce résultat peut paraître compliqué mais il est exact et c'est sans doute la façon la plus simple d'exprimer la fonction en mettant en évidence les points de discontinuité de la dérivée tout en utilisant qu'une seule formule ! Comparons alors l'intégrale entre 0 et x de f sous la forme initiale puis sous cette dernière forme. Dans les deux cas on utilise les commandes **C**alcul, **I**ntègre puis dév**E**loppe, appliquées d'abord à #1 puis à #6 .

7 $\quad \int_0^x \mathrm{ASIN}\left[\frac{2x}{1+x^2}\right]dx$

8 $\quad x\ \mathrm{SIGN}(x^2-1)\ \mathrm{ATAN}\left[\frac{2x}{x^2-1}\right]+\mathrm{LN}(x+1)\ \mathrm{SIGN}(x^2-1)$

9 $\quad \int_0^x\left[-2\ \mathrm{ATAN}(x)\ \mathrm{SIGN}(x^2-1)+\frac{\pi\ \mathrm{SIGN}(x^2-1)}{2}+\pi\ \mathrm{SIGN}(x+1)-\frac{\pi}{2}\right]dx$

10 $\quad -2x\ \mathrm{ATAN}(x)\ \mathrm{SIGN}(x^2-1)+\mathrm{SIGN}(x^2-1)\ \mathrm{LN}\left[\frac{x^2+1}{2}\right]+\frac{\pi x\ \mathrm{SIGN}(x^2-1)}{2}+\pi x\ \mathrm{SIGN}(x+1)$

$\quad -\mathrm{LN}(2)-\frac{\pi\ x}{2}$

Seule la dernière forme (#10) donne le résultat juste comme nous pouvons le vérifier en remplaçant x par $\sqrt{3}$ à l'aide de la commande **A**uteur et en demandant la **S**implification d'abord de #8 :

11 $\quad x := \sqrt{3}$

12 $\quad 2\ \mathrm{LN}(2)+\frac{\sqrt{3}\ \pi}{3}$

Et nous obtenons un résultat faux, tandis qu'en demandant la **S**implification de #10 nous avons le résultat juste :

13 $\quad \frac{\sqrt{3}\ \pi}{3}$

Nous aurions pu obtenir directement une valeur approchée correcte en prenant l'intégrale de la fonction f (ligne #7) et en utilisant la commande appro**X** .

14 $\quad 1,81380$

Nous pouvons peut-être expliqué l'ensemble de ces résultats en disant que DERIVE considère la quantité $\mathrm{SIGN}(x^2-1)$ comme une constante, même dans l'intégration.

Equations différentielles

Les équations différentielles sont probablement l'outil mathématique le plus utilisépar les autres sciences. Beaucoup de problèmes ont une modélisation mathématique par un système différentiel, leurs résolutions formelles ou approchées est donc d'un grand intérêt. Le logiciel Mathematica permet de résourdre tous les cas que l'on sait faire à la main et utilise souvent des changements de variables pour se ramener à un problème résoluble classiquement. I l utilise également les algorithmes les plus récents pour effectuer des résolutions numériques.

▩ Résolution formelle

Elle se fait au moyen de la fonction : DSolve[].
DSolve[équation, y (fonction), x (variable)].
DSolve[équations, {fonctions}, {variables}].
La solution est soit donnée comme une fonction pure ou comme une régle.

In[1]:=

Sol=DSolve[{y'[x]+y[x]==0},y,x]

Out[1]=

```
                          C[1]
-{-{y -> Function[x, ----]}}
                           x
                          E
```

In[2]:=

Sol2=DSolve[{y'[x]+y[x]==0},y[x],x]

Out[2]=

```
              C[1]
-{-{y[x] -> ----}}
               x
              E
```

■ Représentation des trajectoires solutions

In[3]:=

L[t_]:=y[t]/.Sol

In[4]:=

r=C[1]->m

Out[4]=

```
C[1] -> m
```

In[5]:=

K[x_]:=L[t]/.r

In[6]:=

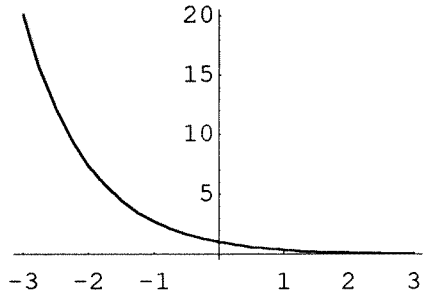
K[t]

Out[6]=

```
  m
-{--}
   t
  E
```

In[7]:=

```
m:=1;
Plot[K[t],{t,-3,3}]
```



Out[8]=

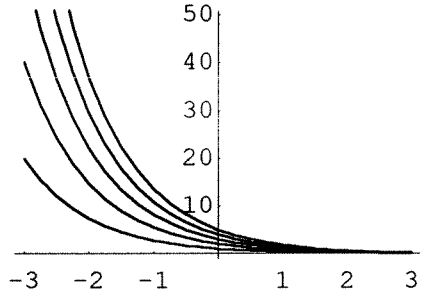
```
-Graphics-
```

In[9]:=

```
m=.;
```

In[10]:=

```
Plot[Evaluate[Table[K[t],{m,5}]],{t,-3,3}]
```



Out[10]=

```
-Graphics-
```

In[11]:=

```
DSolve[y''[x]-4 y'[x]+5 y[x]==
        Exp[2x](Sin[x]+2 Cos[x]), y[x], x]
```

Out[11]=

```
                  2 x
              -(E    x Cos[x])      2 x
{{y[x] -> ----------------- + E    C[2] Cos[x] +
                    2

      2 x                2 x
     E    x Sin[x] - E    C[1] Sin[x]}}
```

■ Problème avec condition initiale

In[12]:=

```
DSolve[{y''[x]-4 y'[x]+5 y[x]==Exp[2x](Sin[x]+2 Cos[x]),
        y[0]==0, y'[0]==1}, y[x], x]
```

Out[12]=

```
                  2 x                    2 x
              -(E    x Cos[x])    3 E    Sin[x]      2 x
{{y[x] -> ----------------- + ------------- + E    x Sin[x]}}
                    2                    2
```

■ Le package DSolve

Il permet d'élargir le champs des équations différentielles résolubles avec Mathematica. Considérons l'exemple suivant, sans le chargement du package Mathematica ne donne aucune réponse, avec le package DSolve on obtient une solution.

In[13]:=

DSolve[x y''[x]+(1-x^2) y'[x]-x y[x]==0,y[x],x]

Out[13]=

```
                                  2
DSolve[-(x y[x]) + (1 - x ) y'[x] + x y''[x] == 0, y[x], x]
```

In[14]:=

<<Calculus`DSolve`

In[15]:=

DSolve[x y''[x]+(1-x^2) y'[x]-x y[x]==0,y[x],x]

Out[15]=

```
                2                  2
               x /4               x
--           E     BesselI[0, --] C[1]
                                 4
{{y[x] -> ------------------------- +
                    Sqrt[Pi]

        2                  2
       x /4              -x
--   E     BesselK[0, ---] C[2]
                         4
--   --------------------------}}
                Sqrt[Pi]
```

In[16]:=

DSolve[{y''[x]+Sin[x]^2 y'[x]+y[x]==Cos[x]^2,
y[0]==1, y'[0]==0},y,x]

Out[16]=

```
                            2                                 2
DSolve[{y[x] + Sin[x]  y'[x] + y''[x] == Cos[x] ,

  y[0] == 1, y'[0] == 0}, y, x]
```

▩ Résolution numérique

Dans le cas où on ne trouve pas de solution formelle, on effectue la résolution approchée à l'aide de la fonction NDSolve[].
NDSolve[équation, y (fonction), {x (variable), min, max}]

In[17]:=

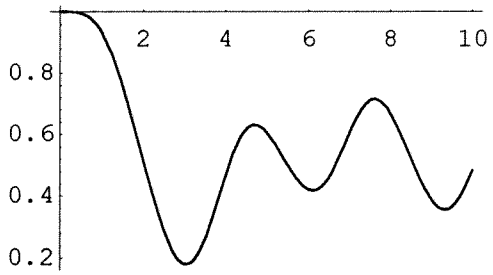
A=NDSolve[{y''[x]+Sin[x]^2 y'[x]+y[x]==Cos[x]^2,
y[0]==1, y'[0]==0},y,{x,0,10}]

Out[17]=

```
{{y -> InterpolatingFunction[{0., 10.}, <>]}}
```

In[18]:=

Plot[Evaluate[y[x]/.A],{x,0,10}]



Out[18]=

```
-Graphics-
```

■ Dessin de plusieurs courbes

In[19]:=

eq=Join[Table[y[i]'[x]==y[i][x],{i,1,3}],
Table[y[i][0]==i, {i,1,3}]]

Out[19]=

```
{(y[1])'[x] == y[1][x], (y[2])'[x] == y[2][x],

 (y[3])'[x] == y[3][x], y[1][0] == 1, y[2][0] == 2,

 y[3][0] == 3}
```

In[20]:=

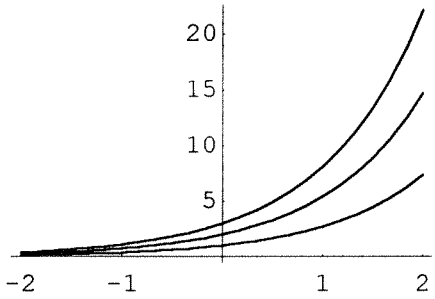
NDSolve[eq, Table[y[i],{i,3}], {x,-2,2}]

Out[20]=

```
{{y[1] -> InterpolatingFunction[{-2., 2.}, <>],

  y[2] -> InterpolatingFunction[{-2., 2.}, <>],

  y[3] -> InterpolatingFunction[{-2., 2.}, <>]}}
```

In[23]:=

Plot[Evaluate[Table[y[i][x],{i,3}]/.%20],{x,-2,2}]



Out[23]=

```
-Graphics-
```

▩ Systèmes d'équations différentielles ordinaires

In[26]:=

```
DSolve[{u'[x]==18 u[x]-16 v[x]-13 w[x],
        v'[x]==-u[x]+29 v[x]+5 w[x],
        w'[x]==-6 u[x]-2 v[x]+19 w[x]},
        {u[x],v[x],w[x]},x]
```

Out[26]=

```
                    11 x     22 x       33 x
                  7 E        E        3 E
{{u[x] -> (------- + ----- + -------) C[1] +
                    11       11       11

           11 x       22 x       33 x
         7 E        2 E        9 E
--   (------- + ------- - -------) C[2] +
           11         11         11

           11 x     22 x       33 x
         7 E        E        6 E
--   (------- - ----- - -------) C[3],
           11         11       11

                   11 x       22 x       33 x
                 -E         3 E        2 E
-- v[x] -> (------ + ------- - -------) C[1] +
                   11         11         11

           11 x       22 x       33 x
         -E         6 E        6 E
--   (------ + ------- + -------) C[2] +
           11         11         11

           11 x       22 x       33 x
         -E         3 E        4 E
--   (------ - ------- + -------) C[3],
           11         11         11

                   11 x       22 x     33 x
                 5 E        4 E        E
-- w[x] -> (------- - ------- - -----) C[1] +
                   11         11       11

           11 x       22 x       33 x
         5 E        8 E        3 E
--   (------- - ------- + -------) C[2] +
           11         11         11

           11 x       22 x       33 x
         5 E        4 E        2 E
--   (------- + ------- + -------) C[3]}}
           11         11         11
```

In[27]:=

```
DSolve[{y1'[x]==y1[x](-y2[x]^2 + y3[x]),
        y2'[x]==y3[x]^2 -(y2[x]^2 y3[x]),
        y3'[x]==-6 y2[x] y3[x] (y3[x] -y2[x]^2)},
        {y1[x],y2[x],y3[x]},x]
```

Out[27]=

```
                                  2
{y1'[x] == y1[x] (-y2[x]  + y3[x]),

                                 2
 {y3[x] -> C[2] - 3 y2[x] ,

                        Sqrt[3] y2[x]                3/2
-- {-(Sqrt[3] ArcTanh[-------------]) + 3 x C[2]     -
                         Sqrt[C[2]]

                                          2
     3 Sqrt[C[2]] Integrate[y2[x][x] , x] -

                                 2
     9 x Sqrt[C[2]] y2[x]  == C[3]}}}
```

▩ Champs de vecteurs

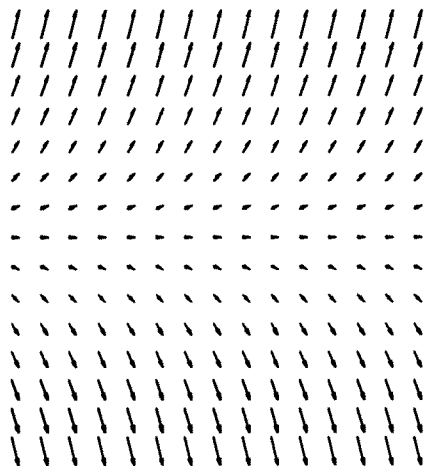
On représente l'équation différentielle : y'=f(x,y) par le champs de vecteurs {x°=1, y°=f(x,y) }; les dérivées ° sont par rapport à t.
L'équation différentielle autonome (le temps n'apparait pas) du second ordre : x"+a(x)x'+b(x)=0 correspond au champs de vecteurs : {x'=y, y'=c-a(x)y-b(x)x} .
En particulier, les équations linéaires du second ordre: x"+ax'+b=0 où a,b sont des constantes se ramènent aux champs de vecteurs : {x'=y, y'=c-ay-bx} .
La représentation d'un champs de vecteurs dans le plan se fait à l'aide de la commande PlotVectorField[] qui nécessite le package "Graphics'PlotField"
PlotVectorField[{1ere composante, 2eme composante}
{1ere coordonnée, min, max²},{2eme coord, min, max}].

In[28]:=

```
<<Graphics`PlotField`
```

In[29]:=

PlotVectorField[{1,y},{x,-4,4},{y,-4,4}]



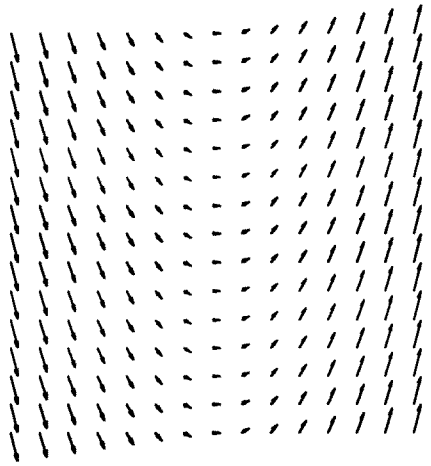
Out[29]=

-Graphics-

Les trajectoires sont des exponentielles.

In[30]:=

PlotVectorField[{1,x},{x,-4,4},{y,-4,4}]

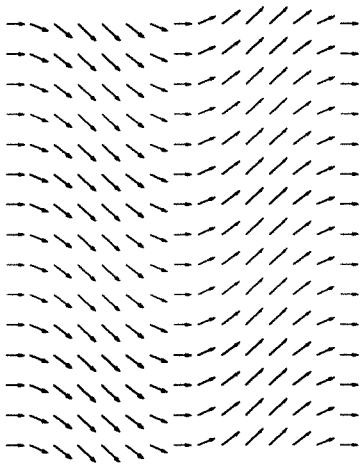


Out[30]=

-Graphics-

In[31]:=

PlotVectorField[{1,Sin[x]},{x,-Pi,Pi},{y,-4,4}]



Out[31]=

```
-Graphics-
```

Equations aux dérivées partielles

Il y a dans la version 2.2 un package (PDSolve1) qui permet de résoudre certaines équations aux dérivées partielles d'ordre un.

In[32]:=

<<Calculus`PDSolve1`

In[33]:=

DSolve[-x y z+z^2 D[f[x,y,z],z]+ y^2 D[f[x,y,z],y]+
x^2 D[f[x,y,z],x]==0,f[x,y,z],{x,y,z}]

Out[33]=

```
{{f[x, y, z] ->

                 2       x                      2     x
              x y  z Log[-]                x y z  Log[-]
                       y                          z
  --  -(-----------------------) + ------------------------ +
                    2                                    2
        -(x y) + y  + x z - y z      -(x y) + x z + y z - z

            1   1  1   1
       C[1][- - -, - - -]}}
            x   y  x   z
```

INTRODUCTION AU CALCUL DES LIMITES

Jean Perrin , groupe Calcul Formel Mulhouse

L'objectif est de préparer les élèves de seconde et de première au calcul des limites de fonctions avec DERIVE et leur calculatrice.

Méthode de travail:

Dans un premier temps, avec des élèves n'ayant aucun notion des limites, je les fais chercher des "simplifications" d'expressions à l'aide de leurs calculatrices (voir exemple ci joint), et pour certains d'entres eux, à tour de rôle (problème de nombre de machines), sur les ordinateurs avec DERIVE.

Le but est qu'ils trouvent que certaines expressions prennent un aspect plus simple "pour X assez grand ou X assez petit ".
Ils découvrent également avec ces méthodes "inductives, ou expérimentales" les cas indéterminés et donnent un sens intuitif aux concepts Limite, Infini, Infiniment petit, Asymptote et ceci avant leur introduction un peu plus formelle et leur usage dans le cadre d'exercices "classiques" de l'enseignement secondaire.

Un certain nombre de problèmes, entre autres axiomatiques, sont posés par les élèves curieux et rigoureux. Ce besoin de rigueur permet au professeur de définir correctement les concepts . D'habitude la démarche est inverse et paradoxalement ici, la demande de rigueur provient des élèves.

Solution axiomatique..et pratique!

Les fondements logiques d'une telle pratique se trouvent dans l'Analyse-non-standard (IST et ZFE sont des solutions théoriques): lire
Lutz,Makhlouf,Meyer interpretation de l'Analyse en terme d'ordre de grandeur : fondement pour une nouvelle pedagogie (Article dans L'Ouvert et Brochure APMEP)
Toni levi Figures d'infini
Deledicq et Diener analyse non standard
Diener et Reeb analyse non standard
Université de Poitier infini

Remarques

1) L'évaluation des solutions utilisant ces techniques pose un probème
à la fois aux élèves et aux collègues corrigeant le bac; il reste du chemin à parcourir...

2) Par certains aspects, le fini est plus compliqué et plus intéressant que l'infini : il permet l'expérimentation, il illustre le phénomène des ordres de grandeur, il montre certaines impossibilités, par exemple comparer 10^10^10^10...et 7^17^13^7....^ce qui met en évidence les formes indéterminées.

3) L'utilisation spécifique de DERIVE possède les avantages suivants :
Premièrement, les calculs exacts et approchés sont très performants
Les résultats sont visibles sur un écran d'ordinateur et imprimables (50chiffres..)

On peut faire des simulations avec des formules, et donc des hypothèses formelles
L'outil graphique n'est pas à négliger car il permet de visualiser (avec des zooms) ces phénomènes.

En pratique, je procède en plusieurs temps:

- *expérimentation avec machines et ordinateurs (voir suite)*
- *résolution d'exercices réels*
- *mise en forme pour les "dinosaures"*
- *pour les élèves intéressés et concernés compléments théoriques*

Exemples d'exercices donnés à des élèves pour l'apprentissage des limites

CONSIGNES

Donnez des exemples
Expliquez et justifiez vos réponses
La réponse est une expression plus simple

A) Supposons que x devient très grand, cherchez...

1) $2+3/x$
2) $(x^2+2)/(x+3)$
3) $(x+4)/(x^2-5)$
4) $(x+7)/(x-5)$
5) x^2-x

B) x devient très petit, cherchez

1) x^2-x
2) $4/x$
3) $(1+x)^2$
4) $5-x$
5) $10/(2-x)$

Autre exemple d'usage de DERIVE et des infiniment petits.

Etude de la fonction (particulièrement en zéro) :
f(x)=x*(10^10^x+1000)/(10^10^x-10) ou alors g(x)=x^2+0.001*ln Isin(1000Pix)I

Conclusion

La réaction des élèves est particulièrement positive(spécifiquement des redoublants..),ils semblent se passionner pour les problèmes de grands nombres,d'approximation,des incongruités de leur machine,surtout quand ils voient pourquoi,il y en a qui "comprennent enfin pourquoi on ne peut pas diviser par zéro"....

Pour les limites, les approximations (physique), les dérivées les suites, les séries (convergences, séries de Fourrier), développements limités, développements en séries entières développements de Taylor, DERIVE est un outil très utile à condition d'en disposer... Pour ma part je conseille vivement à mes collègues l'usage de ce logiciel pour leur enseignement...

1: "A) vers l'infini"

2: "A1)"

3: f1 10

4: 2.3

5: f1 100

6: 2.03

7: f1 1000

8: 2.003

9: f1 10000000000

10: 2.00000

55: "B) vers les petits.."

56: h1 0.1

57: -0.09

58: h1 0.001

59: $-9.99\ 10^{-4}$

60: $h1\ 10^{-10}$

61: $-9.99999\ 10^{-11}$

62: h2 0.1

63: 40

64: h2 0.001

65: 4000

66: $h2\ 10^{-9}$

67: $4\ 10^{9}$

68: h3 0.1

69: 1.21

70: h3 0.001

71: 1.00200

72: $h3\ 10^{-10}$

73: 1.00000

74: h4 0.1

75: 4.9

76: h4 0.001

77: 4.999

78: h4 0.00000001

79: 4.99999

80: h5 0.1

81: $\frac{100}{19}$

82: h5 0.001

83: $\frac{10000}{1999}$

84: h5 0.000000001

85: $\frac{10000000000}{1999999999}$

Une session Mathematica pour Calculer les limites des fractions rationnelles à l'aide des ordres de grandeur

Le but de cette session sous le logiciel Mathematica est l'application de l'Analyse approximative au calcul des limites et des dérivées des fractions rationnelles. On se place dans le cadre de la théorie ZFE présentée dans l'ouvrage: "L'interprétation de l'analyse en termes d'ordres de grandeur: fondement pour une nouvelle pédagogie" (Lutz - Makhlouf - Meyer) (Brochure APMEP).

On introduit dans ce cadre le prédicat bien déterminé d'où découle les nombres réels très petits et très grands avec un statut bien défini. La définition et le calcul de limites consistent, comme nous le suggère l'intuition, en un calcul sur les ordres de grandeur. L'analyse approximative sous jacente se pratique à l'aide de simplications formelles au moyen de régles d'approximations naturelles.

La limite en zéro d'une fonction consiste seulement en un calcul sur les ordres de grandeurs des nombres. En effet, il s'agit de simplifier formellement une expression algébrique par rapport à un ensemble de régles d'approximation à travers des manipulations algébriques. On définit les régles régissant les quantités très petites, très grandes et bien déterminées ainsi qu'une procédure calculant la limite en zéro à l'aide de ces régles. Mathematica dispose d'une fonction limite et dérivée pour vérifier nos résultats.

Les limites en un point quelconque et à l'infini s'y ramènent à l'aide d'un changement de variable, de même que le calcul du nombre dérivé et de la fonction dérivée.

▩ Outils de Mathematica utilisés

□ Les régles

On rappelle les fonctions de Mathematica utilisées dans la suite:
Une régle est une liste de substitutions. Il y a deux types: {a->b} ou {a:>b}
a->b : b est évalué puis a est remplacé par b,
a:>b : remplace a par b puis évalue b.
On applique une régle R à une expression Expr de la façon suivante:
Replace[Expr,R] (qu'on peut aussi écrire Expr/.R) si on veut appliquer la régle une seule fois. Si on veut l'appliquer plusieurs fois, ReplaceRepeated[Expr,R] (ou encore Expr//.R).
Expr/.R : applique une fois la règle R dans Expr
Expr//.R/;Test : applique la règle dans Expr si le Test est vérifié et tant qu'il y a des modifications.
Dans les expressions suivantes,

x_. +y_ : l'argument x a par défaut la valeur 0.
x_. y_ : l'argument x a par défaut la valeur 1.
a_^x_. : l'argument x a par défaut la valeur 1.

Les simplifications algébriques

Soit P une expression algébrique,
Expand[P] : developpe P,
ExpandAll[P] : developpe toutes les parties de l'expression P,
Factor[P] : factorise P,
Simplify[P] : simplifie P,
PowerExpand[P] : developpe les puissances dans P, par exemple Sqrt[x^2] devient x,
Together[P] : réduit au même dénominateur,

Autres fonctions

On définit une nouvelle fonction f de la manière suivante:
f[x_]:=expression dépendant de l'argument muet x.
Limit[f[x],x->a], calcule la limite de f quand x tend vers a.
Timing[Instruction] donne le temps de calcul.
Pour créer une nouvelle procédure " lim" ayant pour argument une fonction f, on utilise la procédure "module" :
lim[f_]:=Module[{variables locales}, instructions].

Calcul des limites en zéro

On définit d'abord les régles d'approximations.

Régles additives entre les très petits

In[1]:=

```
RP1={     m_. tp^(a_.)+n_. tp^(b_.):>n tp^b/;a>b};
```

In[3]:=

```
f[x_]=(1+x+x^2)/(-5x^(1/3)+3x^4)
```

Out[3]=

$$\frac{1 + x + x^2}{-5\,x^{1/3} + 3\,x^4}$$

In[4]:=

```
a=f[tp]//.RP1
```

Out[4]=

$$\frac{-(1 + tp)}{5\,tp^{1/3}}$$

In[5]:=

```
RP2={     a_+b_. tp^c_.:>a /;b=!=tg&&c=!=tg&&c>0};
```

In[6]:=

```
a=a//.RP2
```

Out[6]=

$$\frac{-1}{5\ tp^{1/3}}$$

■ régles entre très petit et très grand

In[7]:=

```
RP3={     (a_)/tp:>tg/;a=!=tp&&a>0,
          (a_)/tp:>-tg/;a=!=tp&&a<0,
          (a_) tp:>0/;a=!=tg,
          tp^(a_):>tg/;a<0&&a=!=tp&&a=!=tg,
          tp^(a_):>tp/;a>0&&a=!=tp&&a=!=tg
  };
```

In[9]:=

```
a=a//.RP3
```

Out[9]=

$$\frac{-tg}{5}$$

■ Régles additives pour les très grands

In[10]:=

```
RP4={     a_.+tg:>tg/;a=!=-tg,
          a_. tg:>-tg/;a=!=tp&&N[a]<0,
          a_. tg:>tg/;a=!=tp&&N[a]>0,
          a_ /tg:>tp/;N[a]>0,
          a_ /tg:>-tp/;N[a]<0
  };
```

In[11]:=

```
a=a//.RP4
```

Out[11]=

```
-tg
```

■ Régles pour conclure

In[12]:=

```
RP5={     b_. tp^a_. :>0,
          b_. tg^a_. :>infinie/;N[b]>0&&N[a]>0,
          b_. tg^a_. :>-infinie/;N[b]<0&&N[a]>0,
          b_. tg^a_. :>0/;N[b]<0
          };
```

In[13]:=

a//.RP5

Out[13]=

```
-infinie
```

Pour vérifier utilisons la fonction de Mathematica.

In[14]:=

Limit[f[x],x->0]

Out[14]=

```
-Infinity
```

Définissons une procédure lim0[ ] qui calcule la limite en 0 de f[x]:

In[15]:=

lim0[f_]:=Module[{},Together[ExpandAll[f[tp]]]//.RP1//.RP2//.RP3//.RP4//.RP5]

□ **Exemples**

In[17]:=

lim0[f]

Out[17]=

```
-infinie
```

In[18]:=

f2[x_]=7+6/x^2+x^(-2)

Out[18]=

```
      7
-7 + --
      2
     x
```

In[19]:=

lim0[f2]

Out[19]=

```
infinie
```

In[20]:=

f3[x_]=(x^4+x^3)/(x^2+x+1)

Out[20]=

```
   3    4
  x  + x
-----------
           2
1 + x + x
```

In[21]:=

lim0[f3]

Out[21]=

```
0
```

In[22]:=

f4[x_]=(3 x^3+5 x^2)/(4 x^4-4 x^3)^2

Out[22]=

$$\frac{5\,x^2+3\,x^3}{(-4\,x^3+4\,x^4)^2}$$

In[23]:=

lim0[f4]

Out[23]=

infinie

In[24]:=

Limit[f4[x],x->0]

Out[24]=

Infinity

Ecrivons une procédure qui fait les deux calculs.

In[25]:=

```
lim[f_]:=Module[{},
            {"ZFE>",lim0[f],
      "MATHEMATICA>",Limit[f[x],x->0]}
                  ]
```

In[26]:=

lim[f4]

Out[26]=

{ZFE>, infinie, MATHEMATICA>, Infinity}

In[27]:=

f5[x_]=(3 x^6-x^2+x^(-1))/(-3 x^(-1)+5 x^3+1)

Out[27]=

$$\frac{\frac{1}{x}+3\,x^6-x^2}{1-\frac{3}{x}+5\,x^3}$$

In[28]:=

lim[f5]

Out[28]=

{ZFE>, -(1/3), MATHEMATICA>, -(1/3)}

In[29]:=

f6[x_]=1/(1+(x^(-1))/(2+x^(-2)))

Out[29]=

```
          1
--------------------
            1
1 + -----------
              -2
     (2 + x   ) x
```

In[30]:=

lim[f6]

Out[30]=

```
{ZFE>, 1, MATHEMATICA>, 1}
```

In[31]:=

f7[x_]=(1/(1+x^(-1)))-(1/(2+x^(-1)))

Out[31]=

```
   1         1
------- - -------
    1         1
1 + -   2 + -
    x         x
```

In[32]:=

lim[f7]

Out[32]=

```
{ZFE>, 0, MATHEMATICA>, 0}
```

▩ Calcul de limite en a

On se ramène à l'aide d'un changement de variable à un calcul de limite en 0. Ensuite on utilise la suite des manipulations déja définie.

In[33]:=

limite[f_,a_]:=Module[{},
{"ZFE>",Together[ExpandAll[f[tp+a]]]//.RP1//.RP2//.RP3//.RP4//.RP5,
" MATHEMATICA>",Limit[f[x],x->a]}]

□ Exemples

In[36]:=

f8[x_]=(x^2+3x-1)/(x^2-1)

Out[36]=

```
                2
-1 + 3 x + x
-------------
            2
   -1 + x
```

In[37]:=

limite[f8,1]

Out[37]=

```
{ZFE>, infinie,  MATHEMATICA>, Infinity}
```

In[38]:=

f9[t_]=(t^2+3t-1)/(3t^3+2t)

Out[38]=

$$\frac{-1+3t+t^2}{2t+3t^3}$$

In[39]:=

limite[f9,2]

Out[39]=

```
          9                      9
{ZFE>, --,  MATHEMATICA>, --}
          28                     28
```

In[40]:=

limite[f9,0]

Out[40]=

```
{ZFE>, -infinie,  MATHEMATICA>, -Infinity}
```

In[41]:=

f10[x_]=((1+m x)^3-1)/x

Out[41]=

$$\frac{-1+(1+m\,x)^3}{x}$$

In[42]:=

limite[f10,0]

Out[42]=

```
{ZFE>, 3 m,  MATHEMATICA>, 3 m}
```

In[43]:=

f11[x_]=1/(x-1)-2/(x^2-x)

Out[43]=

$$\frac{1}{-1+x}-\frac{2}{-x+x^2}$$

In[44]:=

limite[f11,1]

Out[44]=

```
{ZFE>, -infinie,  MATHEMATICA>, -Infinity}
```

▩ Limites à l'infinie

On fait le changement de variable t->1/tp et on utilise les mêmes transformations. LimGp[ ] calcule la limite en plus l'infini et LimGn[ ] en moins l'infini.

In[45]:=

limGp[f_]:=Module[{},

```
        {"ZFE>",Together[ExpandAll[f[1/tp]]]//.RP1//.RP2//.RP3//.RP4//.RP5,
        " MATHEMATICA>",Limit[f[x],x->Infinity]}]
```

In[46]:=

```
limGn[f_]:=Module[{},
        {"ZFE>",Together[ExpandAll[f[-1/tp]]]//.RP1//.RP2//.RP3//.RP4//.RP5,
        " MATHEMATICA>",Limit[f[x],x->-Infinity]}
                ]
```

```
General::spell1:
   Possible spelling error: new symbol name "limGn"
     is similar to existing symbol "limGp".
```

In[48]:=

```
f12[x_]=1/x
```

Out[48]=

```
1
-
x
```

In[49]:=

```
limGp[f12]
```

Out[49]=

```
{ZFE>, 0,  MATHEMATICA>, 0}
```

In[50]:=

```
f13[x_]=(1+x+x^2)/(1+3x-x^3)
```

Out[50]=

```
             2
    1 + x + x
  ------------
              3
  1 + 3 x - x
```

In[51]:=

```
limGp[f13]
```

Out[51]=

```
{ZFE>, 0,  MATHEMATICA>, 0}
```

In[52]:=

```
f14[x_]=(1+x+x^6)/(1+3x-x^3)
```

Out[52]=

```
             6
    1 + x + x
  ------------
              3
  1 + 3 x - x
```

In[53]:=

```
limGp[f14]
```

Out[53]=

```
{ZFE>, -infinie,  MATHEMATICA>, -Infinity}
```

In[54]:=

limGn[f14]

Out[54]=

```
{ZFE>, infinie,  MATHEMATICA>, Infinity}
```

In[55]:=

f15[x_]=(1+x+m x^3)/(1+3x-2x^3)

Out[55]=

$$\frac{1 + x + m\,x^3}{1 + 3\,x - 2\,x^3}$$

In[56]:=

limGp[f15]

Out[56]=

$$\left\{ZFE>, -\frac{m}{2}, \text{MATHEMATICA>}, -\frac{m}{2}\right\}$$

▩ Calcul du nombre et de la fonction dérivée

De la même manière la dérivée d'une fonction f en un point a revient à simplifier l'expression (f(a+tp)-f(a))/tp par rapport aux règles d'approximations.
Pour déterminer la fonction dérivée, il suffit de laisser a comme paramètre.

In[57]:=

der[f_,a_]:=Module[{},
Together[ExpandAll[(f[a+tp]-f[a])/tp]]//.RP1//.RP2//.RP3//.RP4//.RP5]

In[58]:=

g1[t_]=2 t^2+t+1

Out[58]=

$$1 + t + 2\,t^2$$

In[59]:=

der[g1,0]

Out[59]=

```
1
```

In[60]:=

der[g1,t]

Out[60]=

$$1 + 4\,t$$

In[61]:=

g2[x_]=((x+3)(2x-1))/(x^2+3x-2)

Out[61]=

$$\frac{(3 + x)\,(-1 + 2\,x)}{-2 + 3\,x + x^2}$$

In[62]:=

der[g2,t]

Out[62]=

$$\frac{-1 - 2t + t^2}{(-2 + 3t + t^2)^{2\,2}}$$

In[63]:=

D[g2[t],t] (*vérification avec la fonction de Mathematica*)

Out[63]=

$$\frac{(3 + t)(-1 + 2t)(3 + 2t)}{(-2 + 3t + t^2)^{2\,2}} + \frac{2(3 + t)}{-2 + 3t + t^2} + \frac{-1 + 2t}{-2 + 3t + t^2}$$

In[64]:=

Together[%]

Out[64]=

$$\frac{-1 - 2t + t^2}{(-2 + 3t + t^2)^{2\,2}}$$

OUF! c'est la même chose.

Partie III

ALGORITHMES

LA FACTORISATION SANS CARRE

La factorisation des polynômes est extrêmement utile dans de nombreuses questions et a donné lieu à de nombreuses théories et techniques de factorisation peu élémentaires. Nous n'allons pas aborder ici ce problème dans sa généralité mais examiner une possibilité offerte par le logiciel DERIVE sous la dénomination "factorisation sans carré".

De quoi s'agit-il ?

L'exécution de la commande "factorisation sans carré" du logiciel DERIVE fournit les résultats suivants : (en troisième colonne, on a mis la factorisation "complète" dans R, également fournit par le logiciel à l'aide de la commande "factorisation radical")

donnée	résultat	factorisation
$x^2 + 2x + 1$	$(x + 1)^2$	$(x + 1)^2$
$x^4 - 2x^3 - 3x^2 + 4x + 4$	$(x^2 - x - 2)^2$	$(x - 2)^2(x + 1)^2$
$x^4 + 3x^3 - 3x^2 - 11x - 6$	$(x + 1)^2(x^2 + x - 6)$	$(x - 2)(x + 3)(x + 1)^2$
$x^{10} + 4x^6 + 2x^5 + 4x^2 + 4x + 1$	$(x^5 + 2x + 1)^2$	$(x^5 + 2x + 1)^2$
$x^7 + 2x^6 - 3x^5 - 6x^4 + 3x^3 + 6x^2 - x - 2$	$(x + 2)(x^2 - 1)^3$	$(x + 2)(x - 1)^3(x + 1)^3$
$x^{12} - 12x^{11} + 25x^{10} + 205x^9 - 872x^8$ $- 593x^7 + 6706x^6 - 3980x^5 - 14792x^4$ $+ 11520x^3 + 4864x^2 + 15360x - 18432$	$(x - 4)^4(x^2 + x - 6)^2$ $(x^4 + 2x^3 - x - 2)$	$(x - 1)(x + 2)(x^2 + x + 1)$ $(x + 3)2(x - 2)^2(x - 4)^4$

Ce dernier exemple n'est-il pas impressionnant ? Il a bien sûr été construit à l'envers, comme les autres exemples d'ailleurs, et servira à expliquer le fonctionnement de l'algorithme de "factorisation sans carré".

Cette factorisation, comme l'illustrent les exemples ci-dessus, regroupe en un même polynôme (non factorisé) les racines de même ordre.
Elle permet donc d'obtenir un polynôme R qui a exactement les mêmes racines que le polynôme P de départ ; les racines de R sont toutes des racines simples et ce polynôme R est déjà partiellement factorisé dans la mesure où chacun de ses facteurs ainsi obtenus regroupe les racines de même ordre de P. La recherche des racines de P se ramène donc à la recherche des racines de polynômes plus simples, dont toutes les racines sont d'ordre 1.

Si P désigne le dernier polynôme du tableau ci-dessus, "la factorisation sans carré" permet de conclure que P admet, sur l'ensemble des nombres complexes :

- une racine d'ordre 4, la racine de $(x - 4)$
- deux racines d'ordre 2, les racines de $(x^2 + x - 6)$
- quatre racines d'ordre 1, les racines de $(x^4 + 2x^3 - x - 2)$

Le polynôme R dont il est question ci-dessus est $(x - 4)(x^2 + x - 6)(x^4 + 2x^3 - x - 2)$.

Ceci est déjà un résultat remarquable, d'autant plus remarquable que, comme nous allons le voir, la "factorisation sans carré" n'utilise que des connaissances et des techniques très simples.

Précisons, à l'aide d'une formulation générale, le résultat fournit par cette factorisation.

Soit P un polynôme à coefficients réels. Il se factorise complètement sur C et les racines non réelles sont deux à deux conjuguées et de même ordre. On peut donc écrire (à un coefficient multiplicatif près) : $P(x) = \prod_i (x - x_i)^{n_i} = \prod_j^J P_j^j$ avec P_j polynôme à coefficients réels et dont toutes les racines sont simples. Par ailleurs, les polynômes P_j sont premiers entre eux et J est l'ordre maximal des racines de P.

La "factorisation sans carré" consiste à fournir cette factorisation à l'aide des polynômes P_j , donnés sous forme développée.

Comment ça marche ?

Voici l'algorithme permettant d'obtenir cette factorisation :

- Première étape :

On pose $Q_1 = P$ et pour $1 \leq n \leq J$, $Q_{n+1} = PGCD(Q_n, P^{(n)})$ avec $P^{(n)}$: dérivée d'ordre n de P.

- Deuxième étape :

On pose $P_J = Q_J$, $P_{J-1} = \frac{Q_{J-1}}{P_J^2}$, $P_{J-2} = \frac{Q_{J-2}}{P_J^3 P_{J-1}^2}$, jusqu'à $P_1 = \frac{Q_1}{P_J^J P_{J-1}^{J-1} \cdots P_2^2}$.

On aura alors, puisque $P = Q_1$: $P = P_1 P_2^2 P_3^3 \cdots P_J^J$.

Il reste à s'assurer que tous les P_j sont des polynômes et qu'ils sont premiers entre eux. (Nous examinerons ces points sur l'exemple "générique" plutôt que d'en proposer une démonstration dont la difficulté majeure réside dans les notations.)

Examinons les outils nécessaires à l'exécution de cet algorithme :

Il y a trois outils utilisés :

- la dérivation d'un polynôme
- la recherche du PGCD de deux polynômes (qui se ramène à l'outil suivant)
- la division de deux polynômes

Chacun de ces outils est une simple manipulation formelle sur des polynômes (qui peuvent se représenter par des suites finies de coefficients).

L'algorithme de "factorisation sans carré" est donc uniquement constitué de manipulations formelles élémentaires.

Examinons ceci sur un exemple :

Reprenons le dernier l'exemple du tableau de présentation des résultats de la "factorisation sans carré" :

$$P(x) = x^{12} - 12x^{11} + 25x^{10} + 205x^9 - 872x^8 - 593x^7 + 6706x^6 - 3980x^5 - 1479x^4 + 11520x^3 + 4864x^2 + 15360x - 18432$$

C'est sous cette forme développée que le polynôme est fourni au logiciel, mais pour l'explication et la justification du fonctionnement de l'algorithme, nous allons prendre la forme factorisée de P (non "connue" du logiciel !)

$P(x) = (x - 4)^4(x^2 + x - 6)^2(x^4 + 2x^3 - x - 2)$

En posant

$F_1 = x^4 + 2x^3 - x - 2$

$F_2 = x^2 + x - 6$

$F_3 = 1$

$F_4 = x - 4$

on a : $P = F_1^1F_2^2F_3^3F_4^4$ (A partir de là, la démonstration est quasiment générale.)

- Première étape

$Q_1 = P$	$= F_1^1F_2^2F_3^3F_4^4$	$P^{(1)} = R_1F_2^1F_3^2F_4^3$	avec R_1 : polynôme premier avec P
Q_2	$= F_2^1F_3^2F_4^3$	$P^{(2)} = R_2F_3^1F_4^2$	avec R_2 : polynôme premier avec P
Q_3	$= F_3^1F_4^2$	$P^{(3)} = R_3F_4^1$	avec R_3 : polynôme premier avec P
Q_4	$= F_4^1$	$P^{(4)} = R_4$	avec R_4 : polynôme premier avec P
Q_5	$= 1$		

- Deuxième étape

$P_4 = Q_4 = F_4$

$P_3 = \dfrac{Q_3}{P_4^2} = \dfrac{F_3^1F_4^2}{F_4^2} = F_3$

$P_2 = \dfrac{Q_2}{P_4^3P_3^2} = \dfrac{F_2^1F_3^2F_4^3}{F_3^2F_4^3} = F_2$

$P_1 = \dfrac{Q_1}{P_4^4P_3^3P_2^2} = \dfrac{F_1^1F_2^2F_3^3F_4^4}{F_2^2F_3^3F_4^4} = F_1$

Les polynômes P_j fournis par l'algorithme sont bien les polynômes F_j attendus.

Remarque : le polynôme R = F1F2F3F4 qui a exactement les mêmes racines que P, mais toutes à l'ordre 1, peut s'obtenir plus simplement (mais sous forme entièrement developpée) par la formule $R = \dfrac{P}{PGCD(P,P')}$.

Remarque :
la fonction Mathematica qui effectue la factorisation sans carré est FactorSquareFree[ ].

LE THEOREME DE STURM

Soit P un polynôme à coefficients réels, sans racines multiples. On a donc PGCD(P,P') = 1.
(Prendre P/PGCD(P,P') si tel n'est pas le cas au départ.)
Soient a et b (a < b) deux réels.

Le théorème de Sturm donne le nombres de racines réelles comprises entre a et b.

Première étape :

On construit une suite de polynômes P_n de la façon suvante :
$P_0 = P$
$P_1 = P'$
P_2 = opposé du reste de la division de P_0 par P_1 : $P_0 = Q_1.P_1 - P_2$
P_3 = opposé du reste de la division de P_1 par P_2 : $P_1 = Q_2.P_2 - P_3$
...
On poursuit ainsi jusqu'à l'obtention d'un reste nul (ce qui se produit nécessairement, puisque la suite des degrés des polyômes P_n est strictement décroissante).

P_r = opposé du reste de la division de P_{r-2} par P_{r-1} : $P_{r-2} = Q_{r-1}.P_{r-1} - P_r$
avec degré de $P_r = 0$ et P_r non nul car PGCD(P,P') = 1.

Deuxième étape :

On pose : W(x) = nombre de changements de signe dans la suite $P_0(x)$, $P_1(x)$, ...$P_r(x)$.

Remarque :

Si x n'est pas une racine de P, $P_0(x)$ ne sera pas nul.
De toute façon, $P_r(x)$ n'est pas nul car P_r est un polynôme constant non nul.
Si l'un des $P_i(x)$ est nul (pour $0 < i < r$), on comptera un seul changement de signe de $P_{i-1}(x)$ à $P_{i+1}(x)$.

On calcule alors W(a) et W(b) et le résultat fourni par le théorème de Sturm est :

Le nombre de racines réelles comprises entre a et b est W(a) - W(b).

Examen sur un exemple :

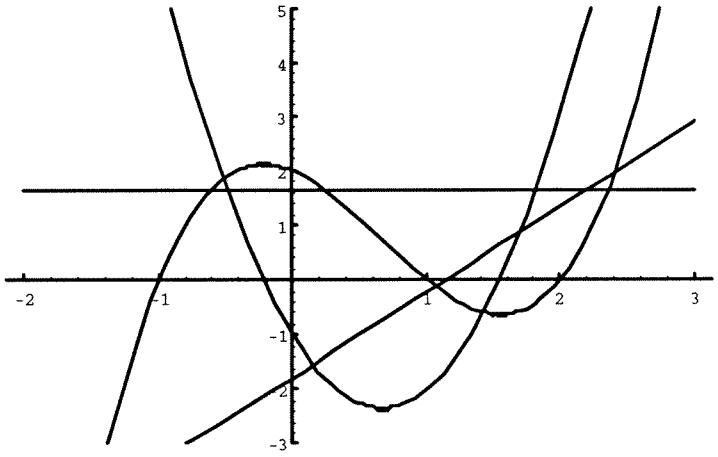
Pour traiter les exemples, nous utilisons le logiciel MATHEMATICA (en annexe, sont données les fonctions crées pour illustrer ce théorème de Sturm).

Prenons P = (x+1)(x-1)(x-2). Le nombre de racines de P entre -2 et 3 est égale à 3. Ce résultat est bien donné par la fonction nbracine : *nbracine[p,-2,3]* fournit **3**.

Examinons la suite *stp* des polynômes P_i : *stp=Expand[suitesturm[p]]* fournit comme résultat

$$\{2 - x - 2x^2 + x^3, -1 - 4x + 3x^2, -\frac{16}{9} + \frac{14x}{9}, \frac{81}{49}\}$$

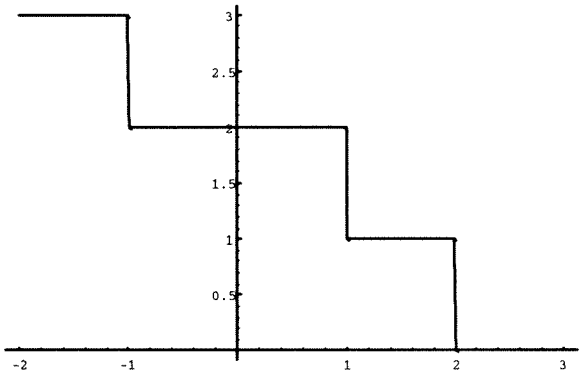
Dessinons ces polynômes sur [-2;3] : *Plot[Evaluate[stp],{x,-2,3},PlotRange->{-3,5}]*



Appelons ls(x) la liste des signes de $P_i(x)$.
On obtient : ls(-2) = {-1,1,-1,1} et ls(3) = {1,1,1,1} (obtenue à l'aide de la fonction *listesigne*)
On a donc W(-2) = 3 et W(3) = 0 ce qui donne bien W(-2) - W(3) = 3.

Essayons de mieux comprendre pourquoi.

Examinons pour cela la fonction W(x). *Plot[nbchgsgn[stp],{x,-2,3}]*



W(x) est une fonction décroissante, en escalier, changeant de valeur précisément aux racines de P. (avec un saut de 1 à chaque fois)

Les fonctions P_i, elles sont continues et changent donc de signe en leurs racines. Comment se fait-il que W(x) change de valeur aux racines de P et seulement aux racines de P ?
C'est ce que nous allons démontrer maintenant.

Démonstration :

La fonction W(x) est à valeur dans l'ensemble des entiers naturels.

L'ensemble des racines de tous les polynômes P_i est fini.
Soit $(x_0, x_1, x_2, \ldots x_p)$ la liste ordonnée (en ordre croissant) de toutes ces racines.

Sur chaque intervalle $]x_j,x_{j+1}[$ la fonction W est constante. (Puisque sur ces intervalles, les polynômes P_i , qui sont continus, sont de signe constant non nul)
Soit j un indice pour lequel la valeur de W sur $]x_{j-1},x_j[$ est différente de celle sur $]x_j,x_{j+1}[$. Il existe alors nécessairement un (ou plusieurs) indice k pour lequel $P_k(x)$ s'annule en changeant de signe en x_j.

Soit ls1 la liste des signes de $P_i(x)$ sur $]x_{j-1},x_j[$ et ls2 la liste des signes de $P_i(x)$ sur $]x_j,x_{j+1}[$.
Soit w1(n,m) le nombre de changements de signe dans la liste ls1, de l'indice n à l'indice m inclus. La valeur de W sur $]x_{j-1},x_j[$ est w1(0,r).
Soit w2(n,m) le nombre de changements de signe dans la liste ls2, de l'indice n à l'indice m inclus. La valeur de W sur $]x_j,x_{j+1}[$ est w2(0,r).

Ce que nous souhaitons établir est w1(0,r) = w2(0,r) lorsque x_j n'est pas une racine de P et que w1(0,r) = w2(0,r) + 1 lorsque x_j est une racine de P.

Premier cas : supposons que x_j soit une racine de $P = P_0$.

Puisque $P_1 = P'$ et que P et P ' n'ont pas de racine commune, on se trouve dans l'une des deux situations suivantes :

x	x_{j-1}		x_j		x_{j+1}
P '			+		
P		-	0	+	

x	x_{j-1}		x_j		x_{j+1}
P '			-		
P		+	0	-	

On a donc w1(0,1) = 1 et w2(0,1) = 0 et il ne reste plus qu'à montrer que w1(1,r) = w2(1,r).

S'il n'y a plus de changement entre les signes de ls1 et de ls2 à partir de i = 1, c'est le cas.
Si non, soit k le plus indice pour lequel cela se produit.
On sait que k n'est pas égal à 1 (voir les tableaux ci-dessus) ni égal à r (puisque P_r est un polynôme constant non nul).
Il s'agit de montrer que w1(k-1,r) = w2(k-1,r).

On a $P_k(x_j) = 0$.
Comme $P_{k-1} = Q_k.P_k - P_{k+1}$.(formule valable pour k de 1 à r-1) on a $P_{k-1}(x_j) = -P_{k+1}(x_j)$.

Or, pour tout indice i (de 0 à r-1), les polynômes P_i et P_{i+1} n'ont pas de racine commune.
En effet, si s est une racine de P_i et de P_{i+1}, elle est également racine de P_{i-1} à cause de la formule $P_{i-1} = Q_i.P_i - P_{i+1}$, et finalement racine de P_1 et de P_0 , c'est-à-dire de P ' et de P. Or ceci n'est pas possible puisque PGCD(P,P') = 1.

P_{k-1} et P_{k+1} sont donc de signes constants et opposés sur $]x_{j-1},x_{j+1}[$ et l'on se trouve dans l'un des quatre cas suivants :

x	x_{j-1}		x_j		x_{j+1}
P_{k+1}			+		
P_k		-	0	+	
P_{k-1}			-		

x	x_{j-1}		x_j		x_{j+1}
P_{k+1}			-		
P_k		+	0	-	
P_{k-1}			+		

x	x_{j-1}		x_j		x_{j+1}
P_{k+1}			+		
P_k		+	0	-	
P_{k-1}			-		

x	x_{j-1}		x_j		x_{j+1}
P_{k+1}			-		
P_k		-	0	+	
P_{k-1}			+		

Dans ces quatre cas, on a w1(k-1,k+1) = 1 et w2(k-1,k+1) = 1. Il ne reste donc plus qu'à établir que w1(k+1,r) = w2(k+1,r) ce qui s'obtient par récurrence à l'aide du raisonnement précédent. On a finalement w1(0,r) = w2(0,r) + 1.

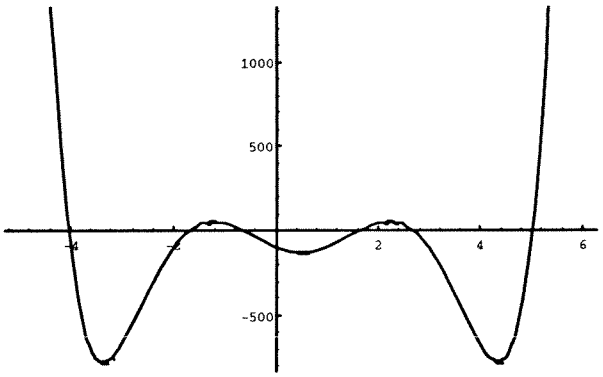
Deuxième cas : supposons que x_j ne soit pas une racine de $P = P_0$.

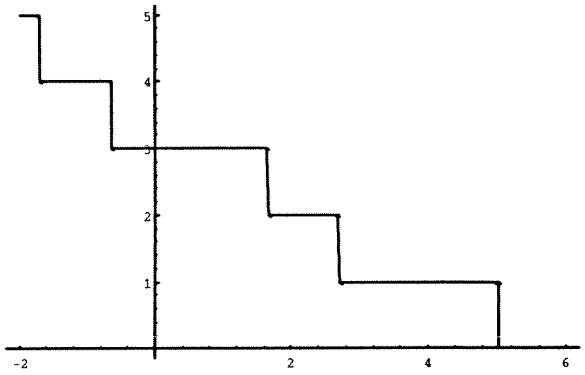
Alors le raisonnement ci-dessus s'applique dès le départ et montre que w1(0,r) = w2(0,r).

Illustration d'un deuxième exemple :

Soit P = -100 + (-5 + x) (-3 + x) (-1 + x) x (2 + x) (4 + x)

Voici sa représentation graphique (à gauche) et celle de W (à droite).





Voici quelques résultats fournis par la fonction *nbracine* :

nbracine[p,-2,6] = 5 ; nbracine[p,0,2] = 1 ; nbracine[p,3,4] = 0.

Bien sûr, ce théorème ne donne les racines, mais il donne le nombre de racines sur un intervalle donné.

Est-ce que cet algorithme permet d'isoler les racines ? (ce qui en permettra une détermination approchée par dichotomie parexemple) En théorie, la réponse est oui. Qu'en est-il en pratique ? (lorsque deux racines sont très proches par exemple) Cette question est laissée à votre réflexion ; de toute façon, elle ne relève plus du domaine du calcul formel !

Que se passe-t-il si P admet des racines multiples ?

Faisons l'essai avec $P = (x+2)^2(x-3)^3(x^2+x+1)$
Eh bien ! nbracine (p,-3,4) par exemple donne comme réponse 2. (avec au passage un message d'erreur sans importance !)

En fait le résultat est juste :

Pour tout polynôme, W(a) - W(b) est égal au nombre de racines réelles de P sur [a ; b].

(sans tenir compte de l'ordre de multiplicité)

La démonstration ci-dessus est-elle à adapter ? à changer complètement ?

Annexe :

les fonctions crées avec MATHEMATICA et utilisées dans le traitement des exemples

```
suitesturm[p_]:=Module[{der,avder,ss,suivant,nss},
        der=D[p,x];avder=p;ss={p,der};
        suivant=-PolynomialRemainder[avder,der,x];
        While[Exponent[der,x]>0,
                (avder=der;der=suivant;
                suivant=-PolynomialRemainder[avder,der,x];
                AppendTo[ss,der])];
        ss]

nbchgsgn[l_]:=Module[{n,l1},
        l1=Map[Sign,l];n=0;
        Do[n=n+Abs[l1[[i]]-l1[[i+1]]]/2,{i,Length[l1]-1}];
        n]

nbracine[p_,a_,b_]:=Module[{sp,wa,wb},
        sp:=suitesturm[p];
        wa:=nbchgsgn[sp/.x->a];wb:=nbchgsgn[sp/.x->b];
        wa-wb]

listesigne[p_,l_,a_]:=Map[Sign,l/.x->a]
```

Utilisation des règles : Application à la détermination de l'ordre de grandeur d'une expression

Le but de cet article est d'illustrer la manipulation des règles dans Mathematica.
La fonction og que nous allons définir ici, prend comme argument une expression expr, une variable x, une valeur finie ou infinie a et un signe s. Elle donne comme résultat l'ordre de grandeur de expr lorsque x tend vers a.
a = tg et s =1 signifie que x tend vers plus l'infini.
a = tp et s = -1 signifie que x tend vers 0 par valeurs négatives.

De façon plus précise, og donne comme résultat un triplet :
le premier élément est l'expression,
le deuxième élément est l'ordre de grandeur cherché s'il peut être trouvé, "?" si non,
le troisième élément donne l'indication de signe sur l'ordre de grandeur cherché s'il peut être trouvé, sinon il signale le problème.
Dans le cas où l'ordre de grandeur ne peut être trouvé, un commentaire signale où se trouve la difficulté.

Dans la version présentée ici, tous les problèmes ne sont pas résolus.
Il faudrait en particulier accompagner ces règles de règles heuristiques de transformations des expressions pour lever les indéterminations.

In[1]:=

```
signe[0]=0;
signe[x_]:=1/;Positive[x];
signe[x_]:=-1/;Negative[x];
(*compléter avec règle des signes*)
signe[x_]:="?";
regleatome:=(x_/;AtomQ[x]):>{x,x,signe[x]};
reglecopieplus=Plus->zfeplus;
reglecopietimes=Times->zfetimes;
reglecopiepower=Power->zfepower;
reglecopie={reglecopieplus,
            reglecopietimes,
            reglecopiepower};
```

In[11]:=

```
SetAttributes[zfeplus,{Flat,OneIdentity,Orderless}];
SetAttributes[zfetimes,{Flat,OneIdentity,Orderless}];
```

In[13]:=

```
zfeplus[{0,0,0},x_]=x;
zfeplus[{v1_,"?",c1_},{v2_,"?",c2_}]={v1+v2,"?",c1<>" et "<>c2};
```

```
zfeplus[{v1_,tg,1},{v2_,tg,1}]={v1+v2,tg,1};
zfeplus[{v1_,tg,-1},{v2_,tg,-1}]={v1+v2,tg,-1};
zfeplus[{v1_,tg,_},{v2_,tg,_}]:=
      Module[{},
      Print["Problème avec : (",
      v1,") + (",v2,") : tg+tg"];
      {v1+v2,"?","tg+tg"}];
zfeplus[{v1_,tp,1},{v2_,tp,1}]={v1+v2,tp,1};
zfeplus[{v1_,tp,-1},{v2_,tp,-1}]={v1+v2,tp,-1};
zfeplus[{v1_,tp,_},{v2_,tp,_}]={v1+v2,tp,"?"};
zfeplus[{v1_,tg,s_},{v2_,_,_}]={v1+v2,tg,s};
zfeplus[{v1_,tp,_},{v2_,y_,s_}]={v1+v2,y,s};
zfeplus[{v1_,x_,_},{v2_,y_,_}]:={v1+v2,tp,"?"}/;x+y==0;
zfeplus[{v1_,x_,_},{v2_,y_,_}]:={v1+v2,x+y,signe[x+y]};
```

In[26]:=

```
zfetimes[{0,0,0},x_]={0,0,0};
zfetimes[{v1_,"?",c1_},{v2_,"?",c2_}]={v1*v2,"?",c1<>" et "<>c2};
zfetimes[{v1_,"?",c_},{v2_,_,_}]={v1*v2,"?",c};
zfetimes[{v1_,tg,_},{v2_,tp,_}]:=
      Module[{},
      Print["Problème avec : (",
      v1,") * (",v2,") : tg*tp"];
      {v1*v2,"?","tg*tp"}];
zfetimes[{v1_,tg,"?"},{v2_,_,_}]={v1*v2,tg,"?"};
zfetimes[{v1_,tg,s1_},{v2_,tg,s2_}]={v1*v2,tg,s1*s2};
zfetimes[{v1_,tp,"?"},{v2_,_,_}]={v1*v2,tp,"?"};
zfetimes[{v1_,tp,s1_},{v2_,tp,s2_}]={v1*v2,tp,s1*s2};
zfetimes[{v1_,tg,_},{v2_,_,"?"}]={v1*v2,tg,"?"};
zfetimes[{v1_,tp,_},{v2_,_,"?"}]={v1*v2,tp,"?"};
zfetimes[{v1_,tg,s1_},{v2_,_,s2_}]={v1*v2,tg,s1*s2};
zfetimes[{v1_,tp,s1_},{v2_,_,s2_}]={v1*v2,tp,s1*s2};
zfetimes[{v1_,x_,_},{v2_,y_,_}]:={v1*v2,x*y,signe[x*y]};
```

In[39]:=

```
zfepower[{v_,_,_},{r_,x_!NumberQ[x],_}]={v^r,"?","non traité"};
zfepower[{v_,0,0},{r_,0,0}]={v^r,"?","0^0"};
zfepower[{v_,0,0},{r_,_,_}]={v^r,0,0};
zfepower[{v_,_,_},{r_,0,0}]={v^r,1,1};
zfepower[{v_,"?",c1_},{r_,"?",c2_}]={v^r,"?",c1<>" et "<>c2};
zfepower[{v_,_,_},{r_,"?",c_}]={v^r,"?",c};
zfepower[{v_,"?",c_},{r_,_,_}]={v^r,"?",c};
zfepower[{v_,tg,_},{r_,n_/;EvenQ[n],1}]={v^r,tg,1};
zfepower[{v_,tp,_},{r_,n_/;EvenQ[n],1}]={v^r,tp,1};
zfepower[{v_,tg,s_},{r_,n_/;OddQ[n],1}]={v^r,tg,s};
zfepower[{v_,tp,s_},{r_,n_/;OddQ[n],1}]={v^r,tp,s};
zfepower[{v_,tg,_},{r_,p_/;EvenQ[p],-1}]={v^r,tp,1};
zfepower[{v_,tp,_},{r_,p_/;EvenQ[p],-1}]={v^r,tg,1};
```

```
zfepower[{v_,tg,s_},{r_,x_/;OddQ[x],-1}]={v^r,tp,s};
zfepower[{v_,tp,s_},{r_,x_/;OddQ[x],-1}]={v^r,tg,s};
zfepower[{v_,tg,1},{r_,_,1}]={v^r,tg,1};
zfepower[{v_,tg,1},{r_,_,-1}]={v^r,tp,1};
zfepower[{v_,tg,1},{r_,_,_}]:=
      Module[{},
      Print["Problème avec :(",v,") ^ (",r,") : tg^r"];
      {v^r,"?","tg^r"}];
zfepower[{v_,tg,_},{r_,_,_}]:=
      Module[{},
      Print["Problème avec :(",v,") ^ (",r,") : pb ^"];
      {v^r,"?","pb ^"}];
zfepower[{v_,tp,1},{r_,_,1}]={v^r,tp,1};
zfepower[{v_,tp,1},{r_,_,-1}]={v^r,tg,1};
zfepower[{v_,tp,1},{r_,_,_}]:=
      Module[{},
      Print["Problème avec :(",v,") ^ (",r,") : tp^r"];
      {v^r,"?","tp^r"}];
zfepower[{v_,tp,_},{r_,_,_}]:=
      Module[{},
      Print["Problème avec :(",v,") ^ (",r,") : pb ^"];
      {v^r,"?","pb ^"}];
zfepower[{v_,x_,_},{r_,y_,_}]:={v^r,x^y,signe[x^y]};
```

In[63]:=

```
og[e_,var_,val_,sg_]:=
      e/.Flatten[{var->{var,val,sg},reglecopie,regleatome}]
```

Voici les fonctions utilisées pour l'illustration.

In[64]:=

```
f[x_]:=x^2-x
```

In[65]:=

```
g[x_]:=((x+1)^2-(x+1)^3)/x
```

In[66]:=

```
h[x_]:=(x^2-x)/(x^(1/2))
```

Voici ce que donne l'application de la fonction og.

In[67]:=

```
og[f[x],x,tg,-1]
```

Out[67]=

```
          2
{-x + x , tg, 1}
```

In[68]:=

og[f[x],x,tg,1]

```
                                2
Problème avec : (-x) + (x ) : tg+tg
```

Out[68]=

```
          2
{-x + x , ?, tg+tg}
```

In[69]:=

og[f[x],x,tp,1]

Out[69]=

```
          2
{-x + x , tp, ?}
```

In[70]:=

og[f[x],x,1,1]

Out[70]=

```
          2
{-x + x , tp, ?}
```

In[71]:=

og[f[x],x,-2,-1]

Out[71]=

```
          2
{-x + x , 6, 1}
```

In[72]:=

og[g[x],x,tg,1]

```
                              2              3
Problème avec : ((1 + x) ) + (-(1 + x) ) : tg+tg
```

Out[72]=

```
           2            3
  (1 + x)  - (1 + x)
{------------------------, ?, tg+tg}
            x
```

In[73]:=

og[g[x],x,tg,-1]

```
                             2           3     1
Problème avec : ((1 + x)  - (1 + x) ) * (-) : tg*tp
                                         x
```

Out[73]=

```
           2            3
  (1 + x)  - (1 + x)
{------------------------, ?, tg*tp}
            x
```

In[74]:=

og[g[x],x,tp,1]

```
                    1            2          3
Problème avec : (-) * ((1 + x)  - (1 + x) ) : tg*tp
                    x
```

Out[74]=

$$\left\{\frac{(1+x)^2-(1+x)^3}{x},\ ?,\ tg*tp\right\}$$

In[75]:=

og[g[x],x,2,1]

Out[75]=

$$\left\{\frac{(1+x)^2-(1+x)^3}{x},\ -9,\ -1\right\}$$

In[76]:=

og[h[x],x,tg,1]

```
                                2
Problème avec : (-x) + (x ) : tg+tg
```

Out[76]=

$$\left\{\frac{-x+x^2}{\text{Sqrt}[x]},\ ?,\ tg+tg\right\}$$

In[77]:=

og[h[x],x,tp,1]

```
                      1              2
Problème avec : (-------) * (-x + x ) : tg*tp
                  Sqrt[x]
```

Out[77]=

$$\left\{\frac{-x+x^2}{\text{Sqrt}[x]},\ ?,\ tg*tp\right\}$$

In[78]:=

og[h[x],x,1,1]

Out[78]=

$$\left\{\frac{-x+x^2}{\text{Sqrt}[x]},\ tp,\ ?\right\}$$

Partie IV

APPLICATIONS DU CALCUL FORMEL DANS LA RECHERCHE MATHEMATIQUE

Groupe des permutations

On présente une application de Mathematica aux groupes de permutations. L'objectif étant de dresser la table des caractères du groupe S(3) et de montrer la résolubilité des groupes S(n) avec n<5 en donnant la suite des groupes dérivés.
On écrit les différentes fonctions :
- Les permutations de S(n) sont représentées par un vecteur de longueur n .
- T[v,w,n] effectue la composition de deux permuations v et w de S(n).
- transpo[n] génère les transpositions de S(n).
- compo[G] compose les éléments d'une liste G.
- elmgr[n] génère tous les éléments de S(n).
- Inv[v] donne l'inverse de v.
- Cla[v,G] donne les éléments de la classe de v dans G.
- GQ[G] renvoi la liste des représentants des classes de conjugaison G.
- mat[v,n] renvoi la représentation matricielle de v de S(n).
- eltmatC[n,G] renvoi les représentations matricielles d'une liste G de permutations de S(n).
- RS[n,G] donne la représentation standard.
- caract[n] donne la table des carctères de S(n).
- Les groupes dérivés sont donnés par "suitedriv[G]" qui nécessite les fonctions "dérivé[G]" et "commutateurs[G]".

Le programme

In[1]:=

```
T[v_,w_,n_]:=Table[v[[w[[i]]]],{i,1,n}]
```

In[2]:=

```
transpo[n_]:=Module[{id,i,j,G},
      id=Table[i,{i,1,n}];
      G={id};
      For[i=1,i<n,i++,
      For[j=i+1,j<=n,j++,w=id;w[[i]]=id[[j]];
      w[[j]]=id[[i]];
            G=Append[G,w]]];
      G]
```

In[3]:=

```
Compo[G_]:=Module[{i,j,GG,Gr},GG={};Gr=G;
      For[i=2,i<=Length[Gr],i++,
      For[j=2,j<=Length[Gr],j++,
      GG=Append[GG,T[Gr[[i]],Gr[[j]],Length[Gr[[1]]]] ]
      ]];
      Gr=Union[Gr,GG];
      Gr]
```

In[4]:=

```
elmgr[n_]:=Module[{G},G=transpo[n];
      While[Length[G]<n!,G=Compo[G]];
            G]
```

In[5]:=

```
Inv[w_]:=Module[{v,i},
```

```
        v=Table[i,{i,Length[w]}];
        For[i=1,i<=Length[w],i++,
        v[[w[[i]] ]]=i];v];
```

In[6]:=

```
Cla[v_,G_]:=Module[{i,w,n,g},
        n=Length[G[[1]] ];
        g={};
        For[i=1,i<=Length[G],i++,
       w=T[Inv[G[[i]] ],v,n];
        w=T[w,G[[i]],n];
       g=Append[g,w];
        ];
        Union[g]]
```

In[7]:=

```
GQ[G_]:=Module[{q,c,f},
        q={};f={};c=G;
        While[c=!={},
        q=Union[q,Cla[c[[1]],G ]];
        f=Union[f,{c[[1]]} ];
        c=Complement[G,q];];f]
```

▩ Représentation matricielle

In[8]:=

```
mat[v_,n_]:=Module[{i,j,mm,m},mm=Array[m,{n,n}];
        For[i=1,i<=n,i++,For[j=1,j<=n,j++,m[i,j]=0]];
        For[i=1,i<=n,i++,m[v[[i]],i]=1];
        mm
              ]
```

□ représentation réguliére

In[9]:=

```
eltmatC[n_,G_]:=Module[{i,MG,L},MG={};L=Length[G];
        For[i=1,i<=L,i++,MG=Append[MG,mat[G[[i]],n]]];
        MG
                ]
```

□ Représentation standard

In[10]:=

```
RS[n_,MG_]:=Module[{i,j,L,V,P,PP},L=Length[MG];V={};
        P={};
        For[i=1,i<=n,i++,
        PP={1};
        For[j=2,j<=n,j++,PP=Append[PP,0]];
        PP[[i]]=1;
        P=Append[P,PP]; ];
        PP=Inverse[P];
        For[i=1,i<=L,i++,
              V=Append[V,PP.MG[[i]].P];
              V[[i]]=Delete[V[[i]],1];
              For[j=1,j<n,j++,
                    V[[i,j]]=Delete[V[[i,j]],1]; ];
              ];
```

```
V];
```

In[13]:=

```
caract[n_]:=Module[{Gq,ma,Rs,lgq,T,G,t},
    G={};Gq={};T={};ma={};Rs={};
        G=elmgr[n];
    Gq=GQ[G];
    lgq=Length[Gq];
    T=Array[t,{lgq+2,lgq}];
    For[i=1,i<=lgq+2,i++,
        For[j=1,j<=lgq,j++,t[i,j]=0 ]];
    For[i=1,i<=lgq,i++,t[2,i]=Gq[[i]] ];
    For[i=1,i<=lgq,i++,t[1,i]=Length[Cla[t[2,i],G] ]];
    For[i=1,i<=lgq,i++,t[3,i]=1];
    ma=eltmatC[n,Gq];
    For[i=1,i<=lgq,i++,t[4,i]=Det[ma[[i]] ]];
    Rs=RS[n,ma];
    For[i=1,i<=lgq,i++,
    t[5,i]=Sum[Rs[[i]] [[m,m]],{m,n-1}] ];
    Print[MatrixForm[T] ] ];
```

□ groupes dérivés

In[15]:=

```
commutateurs[G_]:=Module[{i,j,GG,g}, GG={G[[1]]};
        For[i=2,i<=Length[G],i++,
        For[j=2,j<=Length[G],j++,
        n=Length[G[[1]]];
        g=T[T[Inv[G[[i]]],Inv[G[[j]]],n],T[G[[i]],G[[j]],n],n];
        GG=Append[GG,g]
        ]];
        GG=Union[GG];GG]
```

In[16]:=

```
compose[G_]:=Module[{i,j,GG,Gr},GG={};Gr=G;
    n=Length[Gr[[1]]];
    For[i=2,i<=Length[Gr],i++,
    For[j=2,j<=Length[Gr],j++,
    GG=Append[GG,T[Gr[[i]],Gr[[j]],n ]]
    ]];
    Gr=Union[Gr,GG];
    Gr]
```

In[17]:=

```
dérivé[G_]:=Module[{GG},GG=commutateurs[G];compose[GG]];
```

In[18]:=

```
suitederiv[G_]:=Module[{GG,i,r}, GG=G;
        i=0; r=2^(Int[Length[G]/2]+1);
        While[Length[GG]>1 || i>=r, i=i+1; GG=dérivé[GG];
        Print["groupe_dérivé(",i,")=",GG]]]
```

Applications

▩ génération des groupes

In[19]:=

G=elmgr[3]

Out[19]=

{{1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1}}

In[20]:=

MatrixForm[MG=eltmatC[3,G]]

Out[20]//MatrixForm=

1 0 0
0 1 0
0 0 1

1 0 0
0 0 1
0 1 0

0 1 0
1 0 0
0 0 1

0 1 0
0 0 1
1 0 0

0 0 1
1 0 0
0 1 0

0 0 1
0 1 0
1 0 0

▩ Table des caractères de S(3)

In[21]:=

R=GQ[G]

Out[21]=

{{1, 2, 3}, {1, 3, 2}, {2, 3, 1}}

In[22]:=

RS[3,MG]

Out[22]=

{{{1, 0}, {0, 1}}, {{0, 1}, {1, 0}}, {{-1, 0}, {-1, 1}}, {{0, -1}, {1, -1}},
{{-1, 1}, {-1, 0}}, {{1, -1}, {0, -1}}}

In[23]:=

MatrixForm[%]

Out[23]//MatrixForm=

```
1   0
0   1

0   1
1   0

-1  -1
0   1

0   1
-1  -1

-1  -1
1   0

1   0
-1  -1
```

In[24]:=

caract[3]

1 3 2

{1, 2, 3} {1, 3, 2} {2, 3, 1}

1 1 1

1 -1 1

2 0 -1

▩ Résolubilité des S(n) n<5

■ n=3

In[25]:=

G=elmgr[3]

Out[25]=

{{1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1}}

In[26]:=

G1=commutateurs[G]

Out[26]=

{{1, 2, 3}, {2, 3, 1}, {3, 1, 2}}

In[27]:=

Compo[G1]

Out[27]=

{{1, 2, 3}, {2, 3, 1}, {3, 1, 2}}

In[28]:=

dérivé[G]

Out[28]=

{{1, 2, 3}, {2, 3, 1}, {3, 1, 2}}

In[29]:=

suitederiv[G]

groupe_dérivé(1)={{1, 2, 3}, {2, 3, 1}, {3, 1, 2}}
groupe_dérivé(2)={{1, 2, 3}}

■ **n=4**

In[30]:=

G4=elmgr[4]

Out[30]=

{{1, 2, 3, 4}, {1, 2, 4, 3}, {1, 3, 2, 4}, {1, 3, 4, 2}, {1, 4, 2, 3},
{1, 4, 3, 2}, {2, 1, 3, 4}, {2, 1, 4, 3}, {2, 3, 1, 4}, {2, 3, 4, 1},
{2, 4, 1, 3}, {2, 4, 3, 1}, {3, 1, 2, 4}, {3, 1, 4, 2}, {3, 2, 1, 4},
{3, 2, 4, 1}, {3, 4, 1, 2}, {3, 4, 2, 1}, {4, 1, 2, 3}, {4, 1, 3, 2},
{4, 2, 1, 3}, {4, 2, 3, 1}, {4, 3, 1, 2}, {4, 3, 2, 1}}

In[31]:=

suitederiv[G4]

groupe_dérivé(1)={{1, 2, 3, 4}, {1, 3, 4, 2}, {1, 4, 2, 3}, {2, 1, 4, 3},
{2, 3, 1, 4}, {2, 4, 3, 1}, {3, 1, 2, 4}, {3, 2, 4, 1}, {3, 4, 1, 2},
{4, 1, 3, 2}, {4, 2, 1, 3}, {4, 3, 2, 1}}
groupe_dérivé(2)={{1, 2, 3, 4}, {2, 1, 4, 3}, {3, 4, 1, 2}, {4, 3, 2, 1}}
groupe_dérivé(3)={{1, 2, 3, 4}}

MATHEMATICA ET LOIS D'ALGEBRES

Michel GOZE
Faculté des Sciences et Techniques
Laboratoire de Mathématiques
32 rue du Grillenbreit
68000 COLMAR

Je ne vais pas présenter ici le logiciel MATHEMATICA , son domaine d'applications est trop vaste et un résumé de quelques lignes comporte nécessairement plus d'oublis que d'informations. Je ne veux qu'exposer le rôle de ce logiciel dans la résolution d'un problème d'algèbre concernant la détermination des lois de multiplication dans un espace vectoriel, problème que l'on ne peut résoudre sans recours à une mécanique capable d'écrire les énoncés (ici des équations polynomiales) et de résoudre les conditions posées.

I. MATHEMATICA ET L'ALGEBRE LINEAIRE

Tout le calcul algébrique que l'on enseigne aux étudiants des deux premières annnées de l'université peut être éffectué en utilisant ce logiciel , à condition de savoir traduire dans le langage de MATHEMATICA les énoncés des problèmes (une connaissance plus qu'élémentaire de l'anglais est suffisante). Les réponses sont présentées sous forme finale : un polynôme est écrit sous forme de polynôme, une matrice sous forme matricielle, une fraction rationnelle sous forme de fraction rationnelle. C'est ainsi que les problèmes de réduction de matrices (recherche des valeurs et vecteurs propres d'une matrice carrée, diagonalisation et mise sous forme de Jordan d'une matrice) rentre dans la compétence de ce logiciel. Ceci est d'ailleurs bien pratique pour préparer un énoncé de problème d'examen : par exemple on se fixe comme sujet la mise sous forme de Jordan d'une matrice carrée nilpotente ; on choisit une matrice nilpotente déjà réduite avec des invariants de similitude bien particuliers de manière à tester l'étudiant sur ses connaissances dans ce domaine. On prend une matrice inversible (on écrit une matrice et on teste immédiatement le déterminant). Le logiciel calcule l'inverse de la matrice de passage et, via la formule de changement de bases, la matrice à réduire. Ceci ne prend que quelques secondes pour avoir cette matrice, la matrice de passage, son inverse et la matrice réduite. Les risques d'erreur dans les énoncés de tels problèmes d'examen sont quasiment nuls. Mais cela n'est pas sans quelques dangers ; par ce procédé on occulte totalement la partie que l'on demande aux étudiants de soigner : les calculs détaillés provenant de l'algorithme de réduction de la matrice et c'est peut-être dans cette partie que se cachent de nombreuses difficultés calculatoires.

II. LOIS D'ALGEBRE SUR C^N

Considérons l'espace vectoriel complexe C^n (on peut aussi prendre R^n sur le corps R). On veut définir sur cet espace une multiplication et éventuellement déterminer toutes

les lois de multiplication possibles. Qu'est-ce qu'une loi de multiplication ? Comme les propriétés de distributivité doivent au moins être conservées (sinon adieu les simplifications), on peut définir une loi de multiplication par une application bilinéaire sur C^n à valeurs sur C^n (le produit de 2 vecteurs est un vecteur). Des classes particulières de lois de multiplication dominent les mathématiques:

- les lois associatives : $X.(Y.Z) = (X.Y).Z$ où X, Y et Z sont des vecteurs de C^n.
  L'espace vectoriel est dans ce cas muni d'une structure de loi d'algèbre associative.
- les lois de Jacobi : $X.(Y.Z) + Y.(Z.X) + Z.(X.Y) = 0$. Les algèbres corespondantes sont dans ce cas appelées algèbres de Lie.
- les lois de Jordan: (j'ai oublié la définition).

Le problème de déterminer toutes les lois d'algèbres de Lie (ou de Jordan ou associatives) se présentent naturellement. Bien entendu on regroupe les lois isomorphes afin que la classification ne dépende pas d'une base éventuellement fixée dans C^n. Deux lois d'algèbres, notées • et ♥, sont isomorphes s'il existe un isomorphisme f de C^n vérifiant:

$$X \bullet Y = f^{-1}(f(X) \heartsuit f(Y)).$$

Ce problème de classification est très difficile à aborder. On peut s'en rendre compte aisément en faisant des essais dès les petites dimensions : classer par exemple les algèbres de Lie de dimension 3 n'est pas aisé et c'est pourtant le cas le plus simple (en dimension 2 il n'y a qu'une seule algèbre de Lie à un isomorphisme près dont la loi n'est pas nulle). Actuellement seules sont établies les classifications des algèbres de Lie de dimension inférieure ou égale à 5. Et ce "maigre résultat" n'est pas dû à un manque d'intérêt. Afin de contourner les difficultés, on réduit la classe des algèbres à étudier en ne s'intéressant qu'à des algèbres vérifiant une propriété particulière. C'est ainsi que l'on fut conduit à s'intéresser aux algèbres de Lie simples, c'est-à-dire aux algèbres de Lie ne contenant aucun idéal propre. Leur classification fut établie au début de ce siècle par Elie Cartan (inspiré profondément par les travaux de Killing). Une deuxième classe d'algèbres de Lie apparaît naturellement, lorsqu'on examine le comportement des opérateurs linéaires adjoints, intimement liés à la structure de l'algèbre :

$$adX : Y \rightarrow X \_ Y \qquad \text{(la multiplication de Jacobi est ici notée \_)}$$

Si tous ces opérateurs sont nilpotents (une puissance est nulle), l'algèbre correspondante est dite nilpotente. Peut-on classer les algèbres de Lie nilpotentes ? Ce problème se ramène à classer simultanément les opérateurs adXi où (Xi) est une base de C^n.

Sachant que la réduction simultanée de deux opérateurs n'est pas facile à exprimer, on ne peut être étonné en apprenant que la classification des algèbres de Lie nilpotentes n'est établie que pour les dimensions inférieures ou égales à 7 (un résultat récent semble prouver que la classification en dimension quelconque est impossible). Ce résultat, quelque peu déprimant pour les classificateurs, amène, non pas à abandonner purement et simplement ce problème, mais à le contourner pour l'aborder sur un plan plus topologique. Revenons à la définition d'une loi de multiplication de Lie. Cette loi est supportée par une application bilinéaire alternée, elle est parfaitement définie par ses composantes, appelées constantes de structure, relatives à une base fixée de C^n:

$$\left[x_i, x_j\right] = \sum_{k=1}^{n} C_{ij}^{k} x_k$$

Les identités de Jacobi se traduisent par des équations polynomiales portant sur ces constantes de structure:

$$\sum_{l=1}^{n} C_{ij}^{l} C_{kl}^{s} + C_{jk}^{l} C_{il}^{s} + C_{ki}^{l} C_{jl}^{s} = 0,\ i \le j \le k,\quad 1 \le l \le n$$

Ainsi l'ensemble des algèbres de Lie sur C^n peut être présenté comme un sous-ensemble de l'espace vectoriel paramétré par ces constantes de structure et défini par des équations polynomiales. C'est donc une variété algébrique.

III. DETERMINATION DES COMPOSANTES ALGEBRIQUES A L'AIDE DE MATHEMATICA.

Une variété algébrique est en général constituée d'un nombre fini de morceaux (appelées composantes) qui sont eux aussi des ensembles algébriques (l'exemple le plus simple est donné par une équation polynomiale d'une variable et sa décomposition en un nombre fini de facteurs). L'ensemble des algèbres de Lie qui est muni d'une structure de variété algébrique est donc une réunion finie de composantes algébriques. La question est donc de savoir estimer (à défaut de compter) le nombre de ces morceaux pour une dimension donnée. Et très rapidement la complexité des équations polynomiales de Jacobi, le nombre élevé de variables (les constantes de structure) vont décourager tous les traqueurs de composantes. L'ensemble des algèbres de Lie complexes de dimension 2 est un plan de dimension 2. En effet une loi d'algèbre de Lie sur C^2 est définie par:

$$\left[e_1, e_2\right] = a e_1 + b e_2,\ a,b \in C$$

et l'ensemble des algèbres de Lie de dimension 2 est paramètré par a et b. En dimension 3, les choses se compliquent un peu : l'ensemble des algèbres de Lie est défini par un système de trois équations polynomiales homogènes de degré 2 portant sur 9 variables. Mais encore dans ce cas, on arrive très facilement à déterminer le nombre de composantes (il y en a deux). En fait tout se passe à peu près bien jusqu'en dimension 7 (à peu près bien est ici un doux euphémisme, car, pour la dimension 7, il aura fallu près de quinze ans pour venir à bout du résultat : la variété algébrique correspondante est donnée par un système de 245 équations polynomiales portant sur 147 variables, mais le résultat a été établi en évitant la résolution de ce système). Au-delà de la dimension 7, même si on restreint notre étude aux seules algèbres de Lie nilpotentes, ce problème est insaisissable. Il est nécéssaire d'inventer une technique de recherche originale. C'est ici qu'un logiciel de calcul formel nous est d'un grand secours. Reprenons l'étude de la variété des algèbres de Lie nilpotentes. On a pu démontrer que pour les dimensions inférieures ou égales à 6, ces variétés étaient irréductibles et qu'en dimension 7, elle était réunion de 2 composantes irréductibles. Que se passe-t-il en dimension supérieure ? Comment se comportent les composantes par rapport aux invariants classiques des algèbres nilpotentes ? Existe-t-il des composantes ne comportant que des algèbres de Lie nilpotentes dont les opérateurs adjoints correspondants n'ont que de petits blocs de Jordan ? Sur ces questions il est difficile d'avoir quelque opinion en n'observant que les petites dimensions. En dimension 8 la variété explose et de nouvelles composantes

apparaissent. Mais dans cette dimension il n'est pas permis de commencer un calcul à la main, voici pourquoi : les conditions de Jacobi sont impressionnantes (229 inconnues et 448 équations). Mais les difficultés majeures proviennent des conditions de nilpotence : voir en annexe quatre de conditions parmi toutes.

Est-il seulement raisonnable d'imager de recopier sans erreur ces équations polynomiales ? Et de les résoudre ? ...

En utilisant MATHEMATICA, on a pu montrer que la variété des nilpotents de dimension 8 était formée de 8 composantes.

Etude des algèbres associatives à l'aide du calcul formel

Abdenacer MAKHLOUF
Université de Haute Alsace
Laboratoire de Mathématiques
4, rue des frères Lumière F-68093 Mulhouse
é-mail : N.Makhlouf@univ-mulhouse.fr

On se propose d'étudier l'ensemble des lois d'algèbres associatives sur l'espace vectoriel C^n. L'étude des composantes irréductibles de cette variété algébrique est basée sur la notion de perturbation, ce qui conduit directement à la mise en place de procédures de calcul formel. En effet, la manipulation des objets infinitésimaux formels s'effectue naturellement sous le logiciel Mathematica. D'autres invariants algébriques concernant les algèbres associatives sont calculables à l'aide de ce logiciel, par exemple la dimension du deuxième groupe de cohomologie de Hochschild ou la détermination des produits scalaires invariants.

I. La variété des lois d'algèbres associatives

Rappelons qu'une loi d'algèbre associative est donnée par une application bilinéaire μ sur C^n :

$$\mu : C^n \times C^n \to C^n$$

vérifiant

$$\forall X,Y,Z \in C^n \quad \mu(\mu(X,Y),Z) - \mu(X,\mu(Y,Z)) = 0.$$

La loi d'algèbre est unitaire si

$$\exists E \in C^n : \forall X \in C^n \quad \mu(X,E) = \mu(E,X) = X.$$

La loi d'algèbre est dite nilpotente si

$$\exists p \in N : \forall X,Y \in C^n \quad (\mu(X,Y))^p = 0.$$

En fixant une base $\{e_i\}$ de C^n, on identifie μ à ses constantes de structure $\left(C_{ij}^k\right)$:

$$\mu(e_i,e_j) = \sum_{k=1}^{n} C_{ij}^k e_k .$$

Les constantes de structure des lois d'algèbre associatives vérifient le système d'équations polynomiales quadratiques suivant :

$$\sum_{l=1}^{n} C_{ij}^l C_{lk}^s - C_{il}^s C_{jk}^l = 0 \quad 1 \le i,j,k,s \le n.$$

Si de plus les lois d'algèbre sont unitaires avec e_1 comme élément neutre, on a

$$C_{1i}^j = C_{i1}^j = \delta_{ij} \quad 1 \le i,j \le n.$$

Ainsi, l'ensemble des lois d'algèbres associatives unitaires, noté Alg_n, est une variété algébrique plongée dans C^{n^3}. Cette variété quadratique est non régulière et non réduite en général. Elle est fibrée par l'action du groupe linéaire $GL(n,C)$: deux lois d'algèbre μ_1 et μ_2 sont isomorphes s'il existe f de $GL(n,C)$ tel que

$$\forall X,Y \in C^n \quad \mu_2(X,Y) = f^{-1}\mu_1(f(X),f(Y))$$

L'orbite d'une loi d'algèbre μ_0 , notée $\vartheta(\mu_0)$, est l'ensemble des lois qui lui sont isomorphes.

I.1. Les classifications

Une première piste pour l'étude de la variété algébrique Alg_n, pour une dimension n fixée, consiste à établir la classification des lois à isomorphisme près (c'est à dire, déterminer toutes les orbites). Ce travail est réalisable pour les petites dimensions mais très vite le nombre d'orbites devient très grand. P. Gabriel a établi la classification jusqu'en dimension 4 [Ga]. Ces classifications étaient connues en partie par R.S. Peirce (1870) et B.G. Scorza (1938). La classification en dimension 5 a été effectué par G. Mazzola [Maz]. On trouve 59 classes d'isomorphie en dimension 5. Ceci préjuge du nombre de classes pour des dimensions plus grandes. Ainsi pour comprendre cette variété fort complexe on est conduit à étudier ses composantes irréductibles.

I.2. Les composantes irréductibles

Une variété algébrique est dite *réductible* si on peut la décomposer en une réunion de deux variétés algébriques. Par exemple, la variété algébrique plongée dans C^2 définie par l'équation $xy = 0$ est la réunion de deux variétés irréductibles engendrées par $x = 0$ et $y = 0$. Toute variété algébrique engendrée par un nombre fini d'équations algébriques est formée d'un nombre fini de composantes irréductibles. Par conséquent, Alg_n est constituée d'un nombre fini de composantes irréductibles. Les lois d'algèbre à orbites ouvertes pour la topologie de Zariski, appelée *algèbres rigides,* présentent un intérêt particulier. En effet, l'adhérence de Zariski de l'orbite d'une loi d'algèbre rigide est une composante irréductible de Alg_n. Comme le nombre de composantes irréductibles est fini, le nombre de classes de lois d'algèbre rigides est aussi fini.

L'étude des composantes irréductibles ou des lois d'algèbre rigides se fait à l'aide d'une approche infinitésimale basée sur la notion de *perturbation des lois d'algèbre* initié par M. Goze dans le cadre de l'Analyse Non Standard [Go]. La force de cette approche réside dans le fait que l'étude des points infiniment proches dans la variété Alg_n revient à examiner formellement un point. Ce langage se prête très bien à une programmation sous un logiciel de calcul formel.

Définition

Soit μ_0 une loi d'algèbre standard de Alg_n. Une loi d'algèbre μ de Alg_n est une perturbation de μ_0 (on note $\mu \approx \mu_0$) si

$$\forall X, Y \text{ standard} \in C^n \quad \mu(X,Y) \approx \mu_0(X,Y) \text{ (lire infiniment proche)}$$

Si la base est fixée et standard, ceci revient à dire que les constantes de structure de μ et celles de μ_0 sont infiniment proches. D'où la définition équivalente de la rigidité.

Définition

Une loi d'algèbre standard μ_0 de Alg_n est rigide si et seulement si toute perturbation μ de μ_0 est dans l'orbite de μ_0 $(\mu \in \vartheta(\mu_0))$.

Voir [M.G]

Une loi d'algèbre rigide détermine une composante irréductible. Les lois d'algèbres qui sont dans cette composante sont soit dans l'orbite de cette algèbre soit admette une perturbation qui est est dans l'orbite.
Plus généralement, dans ce même formalisme on a que deux lois sont dans une même composante irréductible si une perturbation de l'une est dans l'orbite de l'autre. Ces

critères exprimés dans le langage des infiniment petits sont par leur aspect formel directement applicable dans un langage de calcul formel.

I.3. Détermination des composantes irréductibles de Alg_n

Un algorithme de détermination de composantes irréductibles de Alg_n est illustré sous le logiciel Mathematica pour la dimension 3. Il utilise un ensemble de procédures regroupées dans un "package" "alas" La démarche est basée sur le résultat suivant :

Proposition

Soit μ_0 une loi d'algèbre standard de Alg_n et *μ une perturbation de μ_0. Si $X_0 \in C^n$ est un idempotent pour μ_0, c'est à dire $\mu_0(X_0, X_0) = X_0$, alors il existe X, $X \approx X_0$, tel que X soit idempotent pour μ.*

voir [G.M] et [Mak]

Conséquence

Le nombre d'idempotents indépendants ne diminue pas par perturbation.

Ce dernier résultats conduit à un algorithme pour déterminer les composantes irréductibles de Alg_n pour une dimension fixée. En effet, le nombre d'idempotent p est compris entre 1 et n. Si on fixe p, une algèbre avec p idemptents indépendant ne peut se perturber que sur une algèbre ayant p idempotents indépendant ou plus. Donc, cette algèbre est soit dans une composante déterminée par une loi d'algèbre avec p idempotents ou plus soit détermine une nouvelle composante.

II. Quelques invariants algébriques

Une autre application du calcul formel dans le domaine des algèbres associatives a pour objectif de l'utiliser pour calculer certains invariants pour une algèbre donnée. Un premier invariant directement lié à l'étude des algèbres rigides est la dimension du deuxième groupe de cohomologie de Hochschild. La deuxième application est de savoir si on peut munir une loi d'algèbre associative d'un produit scalaire invariant.

II.1. Deuxième groupe de cohomologie de Hochschild

Soit μ_0 une loi d'algèbre de Alg_n.

On définit l'espace des cobords par :

$$B^2(\mu_0,\mu_0) = \{\varphi : C^n \times C^n \to C^n \ / \ \varphi = \delta_1 f,\ f \in GL(n,C)\}$$

avec $\delta_1 f(X,Y) = \mu_0(f(X),Y) - f(\mu_0(X,Y)) + \mu_0(X,f(Y)) \quad X,Y \in C^n$

et l'espace des cocycles par

$Z^2(\mu_0,\mu_0) = \{\varphi : C^n \times C^n \to C^n \ / \ \delta_2\varphi = 0\}$

avec $\delta_2\varphi(X,Y,Z) = \mu_0(\varphi(X,Y),Z) - \mu_0(X,\varphi(Y,Z)) + \varphi(\mu_0(X,Y),Z) - \varphi(X,\mu_0(Y,Z)) \quad X,Y,Z \in C^n$

Le deuxième groupe de cohomologie de Hochschild est :

$$H^2(\mu_0,\mu_0) = \frac{Z^2(\mu_0,\mu_0)}{B^2(\mu_0,\mu_0)}$$

Ce groupe est intimement lié aux perturbations. Les cocycles qui ne sont pas des cobords donnent d'eventuelles directions dans lesquelles on peut perturber. D'après le théorème de Gerstenhaber Si $H^2(\mu_0,\mu_0) = \{0\}$ alors la loi d'algèbre μ_0 est rigide.

La dimension de l'espace des cobords d'une loi d'algèbre correspond à la dimension de son orbite. Il correspond géométriquement au plan tangent de Zariski à l'orbite.

II.2. Algèbres métrisables

Un couple (μ, f) est une loi d'algèbre associative métrisable si μ est une loi d'algèbre associative et f une forme bilinéaire symétrique non dégénérée sur C^n vérifiant

$$\forall X,Y,Z \in C^n \quad f(\mu(X,Y),Z) = f(X,\mu(Y,Z)) \text{ (condition d'invariance)}$$

Le problème consiste à examiner pour une loi d'algèbre associative donnée toutes les formes bilinéaires symétriques invariantes. S'il en existe qui soit non dégénérée alors la loi d'algèbre est métrisable.

III. Le logiciel

Il s'agit d'un ensemble de procédures, réalisé avec la collaboration de G. Valeiras, écrites sous le logiciel Mathematica regroupées dans un "package" appelé "alas". Ces procédures permettent après chargement du "package" "alas" d'effectuer des calculs liés à l'étude de la variété des algèbres associatives directement dans un fichier Mathematica.

Le "package "alas" comporte les procédures suivantes :

La loi d'algèbre de dimension n se définit, dans une base $\{x[1],...,x[n]\}$ par ses constantes de structure. On rentre les différentes valeurs : $mu[x[i],x[j]] \quad i,j = 1,\cdots,n$.

La procédure *assoc*[] donne l'ensemble des relations d'associativité.

La procédure *ChangeBase*[...] permet d'écrire la loi d'algèbre dans une nouvelle base.

La procédure *mu*[*base*] sert à afficher la loi d'algèbre dans la base considérée.

Les procédures *DimZ2*[] (resp. *DimB2*[]) calcule la dimension de l'espace des cocycles (resp. des cobords) de la cohomologie de Hochschild.

La procédure *metris*[] donne les conditions de métrisabilité de l'algèbre et la procédure *matf*[] affiche la matrice des formes bilinéaires symétriques invariantes.

Références

[Ga] Gabriel P. *Finite représentation type is open,* Lecture Notes in Maths N°488, (1974).

[Go] Goze M. *Perturbations of Lie algebras structures,* NATO Adv. Sci. INSt. Serie C 297 (1988).

[G.M] Goze M. et Makhlouf A., *On the rigid complex algebras*, Communications in Algebra N°18, (1990).

[Maz] Mazzola G. *The algebraic and geometric classification of associative algebras of dimension five,* Manuscripta math 27, (1979).

[M.G] Makhlouf A. et Goze M. *Classification of rigid associative algebras in low dimensions,* Actes colloque Colmar , collec. Travaux en cours Hermann (1995).

[Mak] Makhlouf A. *Irreducible components of nilpotent associative algebras,* revista Matematica de la Universidad Complutense Madrid, Vol 6, 1 (1993).

[W] Wolfram S. *Mathematica : a system for doing mathematics by computer* Addison-Wesley

Etude de Alg(3) sous Mathematica

Composantes irréductibles de Alg(3)

- *On se propose de déterminer les composantes irréductibles de la variété des lois d'algèbre associatives de dimension 3. On utilise l'approche non standard basée sur l'étude de l'évolution par perturbation du nombre d'idempotents dans la loi d'algèbre. On considère ici les trois cas où le nombre d'idempotents est 3, 2 ou 1. Les lois d'algèbre avec trois idempotents sont isomorphes à l'algèbre commutative semi-simple qui détermine une composante irréductible.*
  On examine d'abord le cas où le nombre d'idempotents est 2 ensuite 1.

lois avec deux idempotents

- *On charge le programme.*

```
<<d:\nacer\alas\alas.m
```

- *On fixe la dimension et la base*

```
n=3;
SetBase[x];
```

- *La forme générale des lois d'algèbres unitaires de dimension 3 est :*

```
For[i=1,i<=n,i++,mu[x[1],x[i]]=x[i];mu[x[i],x[1]]=x[i]]
mu[x[2],x[2]]=x[2];
mu[x[2],x[3]]=Sum[a[i] x[i],{i,n}];
mu[x[3],x[2]]=Sum[b[i] x[i],{i,n}];
mu[x[3],x[3]]=0;
Print[REL=assoc[]];
```

```
   2     2
{b  , -a  , b  b , -(a  a ), -a  + a  a , -a  - a  a , a  + a  a ,
  2     2    1  2     1  2     1    1  3    1    2  3   1    2  3

          2
 -a  + a  , a  b  - a  b , -(a  b ) + a  b , b  - b  b , -b  - b  b ,
   3    3    2  1    1  2     3  1     1  3   1    1  3    1    2  3

                    2
 b  + b  b , b  - b  , a  - b  - a  b  + a  b ,
  1    2  3   3    3    1    1    3  2    2  3

 -a  - a  + b  + b  - a  b  + a  b }
   1    2    1    2    3  2    2  3
```

```
b[2]:=0; a[2]:=0; a[1]:=0; b[1]:=0;
Factor[Union[REL]]
```

```
{0, (-1 + a ) a , (1 - b ) b }
           3   3         3   3
```

Il y a en principe quatre cas à examiner mais ils se ramènent aux deux cas suivants:

■ loi d'algèbre (2.1)

a[3]:=1; b[3]:=0;

mu[base]

```
x  * x  = x
 1    1    1
x  * x  = x
 1    2    2
x  * x  = x
 1    3    3
x  * x  = x
 2    1    2
x  * x  = x
 2    2    2
x  * x  = x
 2    3    3
x  * x  = x
 3    1    3
```

■ loi d'algèbre (2.2)

a[3]:=0;b[3]:=0;

mu[base]

```
x  * x  = x
 1    1    1
x  * x  = x
 1    2    2
x  * x  = x
 1    3    3
x  * x  = x
 2    1    2
x  * x  = x
 2    2    2
x  * x  = x
 3    1    3
```

□ Perturbations de la loi d'algèbre (2.2)

mu[x[3],x[3]]:=eps x[3];

mu[base]

```
x  * x  = x
 1    1    1
x  * x  = x
 1    2    2
x  * x  = x
 1    3    3
x  * x  = x
 2    1    2
x  * x  = x
 2    2    2
x  * x  = x
 3    1    3
x  * x  = eps x
 3    3         3
```

assoc[]

```
{}
```

● *On constate que la perturbation est associative.*

```
ChangeBase[{z[1]==x[1],
      z[2]==x[2],
      z[3]==(1/eps) x[3]},x,z];
SetBase[z]
{z , z , z }
  1   2   3

mu[base]
z  * z  = z
 1    1    1
z  * z  = z
 1    2    2
z  * z  = z
 1    3    3
z  * z  = z
 2    1    2
z  * z  = z
 2    2    2
z  * z  = z
 3    1    3
z  * z  = z
 3    3    3
```

● *La perturbation de l'algèbre (2.2) est isomorphe à l'algèbre semi-simple avec 3 idempotents.*

■ **Conclusion.**

● *La loi d'algèbre (2.2) appartient à la composante formée par l'algèbre semi-simple et la loi d'algèbre (2.1) qui ne se perturbe pas sur l'algèbre à 3 idempotents définit une deuxième composante.*

▩ Lois avec un seul idempotent

```
SetBase[x];
For[i=1,i<=n,i++,mu[x[1],x[i]]=x[i];mu[x[i],x[1]]=x[i]]
mu[x[2],x[2]]=c[1]x[1]+c[3]x[3];
mu[x[2],x[3]]=Sum[a[i] x[i],{i,n}];
mu[x[3],x[2]]=Sum[b[i] x[i],{i,n}];
mu[x[3],x[3]]=0;
REL=asoc[];
b[2]:=0; a[2]:=0; b[1]:=0; a[1]:=0;
Factor[Union[REL]]
    2       2
{0, a  - c , -b  + c , (a  - b ) c }
    3   1   3   1   3   3   3
```

■ loi d'algèbre (1.1)

```
c[1]:=b[3]^2; a[3]:=b[3];
```

mu[base]

```
x  * x  = x
 1    1    1
x  * x  = x
 1    2    2
x  * x  = x
 1    3    3
x  * x  = x
 2    1    2
           2
x  * x  = b   x  + c  x
 2    2    3   1    3  3
x  * x  = b  x
 2    3    3  3
x  * x  = x
 3    1    3
x  * x  = b  x
 3    2    3  3
```

□ **Isomorphisme.**

ChangeBase[{z[1]==x[1],
z[2]==q x[1]+x[2]+p x[3],
z[3]==x[3]},x,z];

mu[base]

```
z  * z  = z
 1    1    1
z  * z  = z
 1    2    2
z  * z  = z
 1    3    3
z  * z  = z
 2    1    2
              2     2
z  * z  = (-q  + b  ) z  + 2 q z  + (2 p b  + c ) z
 2    2           3    1        2          3    3   3
z  * z  = (q + b ) z
 2    3          3   3
z  * z  = z
 3    1    3
z  * z  = (q + b ) z
 3    2          3   3
```

● *Si b[3] est distinct de 0. En posant q=-b[3] et p=-c[3]/2b[3], on obtient une algèbre avec deux idempotents, ce qui ne convient pas.*
Si b[3]=0. On peut supposer c[3]=1 sinon c'est la loi d'algèbre nulle.

b[3]:=0;c[3]:=1;

mu[base]

```
x  * x  = x
 1    1    1
x  * x  = x
 1    2    2
x  * x  = x
 1    3    3
x  * x  = x
 2    1    2
x  * x  = x
 2    2    3
x  * x  = x
 3    1    3
```

□ **Perturbations de (1.1)**

mu[x[3],x[3]]:=eps^2 x[3];
mu[x[2],x[3]]:=eps x[3]; mu[x[3],x[2]]:=eps x[3];

assoc[]

{}

ChangeBase[{z[1]==x[1],
z[2]==1/eps x[2],
z[3]==1/eps^2 x[3]},x,z];

mu[base]

```
z  * z  = z
 1    1    1
z  * z  = z
 1    2    2
z  * z  = z
 1    3    3
z  * z  = z
 2    1    2
z  * z  = z
 2    2    3
z  * z  = z
 2    3    3
z  * z  = z
 3    1    3
z  * z  = z
 3    2    3
z  * z  = z
 3    3    3
```

La loi d'algèbre obtenue possède deux idempotents.

■ loi d'algèbre (1.2)

c[1]:=b[3]^2; a[3]:=-b[3]; c[3]:=0;

mu[base]

```
x  * x  = x
 1    1    1
x  * x  = x
 1    2    2
x  * x  = x
 1    3    3
x  * x  = x
 2    1    2
           2
x  * x  = b   x
 2    2    3   1
x  * x  = -(b  x )
 2    3      3  3
x  * x  = x
 3    1    3
x  * x  = b  x
 3    2    3  3
```

□ **Isomorphisme.**

```
ChangeBase[{z[1]==x[1],
      z[2]==q x[1]+x[2],
      z[3]==x[3]},x,z];
```

mu[base]

```
z  * z  = z
 1    1    1
z  * z  = z
 1    2    2
z  * z  = z
 1    3    3
z  * z  = z
 2    1    2
             2      2
z  * z  = (-q  + b  ) z  + 2 q z
 2    2            3    1        2
z  * z  = (q - b ) z
 2    3          3   3
z  * z  = z
 3    1    3
z  * z  = (q + b ) z
 3    2          3   3
```

● *Si b[3] est différent de 0 et q=b[3] l'algèbre est une loi d'algèbre avec 2 idempotents. Donc elle ne convient pas.*
Si b[3]=0, la loi d'algèbre est l'algèbre unitaire nulle. Elle se perturbe facilement en une loi d'algèbre avec deux idempotents. Par exemple en posant mu[x[2],x[2]]=eps x[2].

■ **Conclusion.**

● *Les deux lois d'algèbre se perturbent sur des lois avec deux idempotents. Par conséquent, elles sont dans des composantes déja définies.*

Conclusion générale

- *La variété Alg(3) est formée de deux composantes irréductibles, la première est donnée par la loi d'algèbre semi-simple ayant trois idempotents et la deuxième composante est déterminée par la loi d'algèbre (2.1). toutes les autres lois sont dans les adhérences des orbites de ces deux lois d'algèbre.*
  Notons que ces deux lois d'algèbres sont rigides.

Calcul de quelques invariants

loi d'algèbre (2.2)

```
For[i=1,i<=n,i++,mu[x[1],x[i]]=x[i];mu[x[i],x[1]]=x[i]];
mu[x[2],x[2]]=x[2];
mu[x[2],x[3]]= x[3];
mu[x[3],x[2]]=0;
mu[x[3],x[3]]=0;
```

Calcul de la dimension du 2ème groupe de cohomologie

DimB2[x]

```
7
```

DimZ2[x]

```
7
```

- *La dimension du deuxieme groupe de cohomologie de Hochschild est nulle. Par conséquent, cette algèbre est rigide.*

Vérification de la métrisabilité

metris[]

```
{f[1, 2] - f[2, 2], -f[1, 2] + f[2, 2], f[1, 3] - f[2, 3],
-f[2, 3], f[2, 3], -f[1, 3] + f[2, 3], -f[3, 3], f[3, 3]}
```

f[1,2]:=f[2,2]; f[2,3]:=0; f[1,3]:=0; f[3,3]:=0;

metris[]

```
{}
```

matf[]

```
f       f
 1,1     2,2    0

f       f
 2,2     2,2    0


0       0       0
```

- *La loi d'algèbre 2.1 n'est donc pas métrisable.*

loi d'algèbre (2.1)

```
For[i=1,i<=n,i++,mu[x[1],x[i]]=x[i];mu[x[i],x[1]]=x[i]];
mu[x[2],x[2]]=x[2];
mu[x[2],x[3]]= 0;
mu[x[3],x[2]]=0;
```

mu[x[3],x[3]]=0;

■ **Calcul de la dimension du 2ème groupe de cohomologie**

DimB2[x]

8

DimZ2[x]

9

● *La dimension du deuxieme groupe de cohomologie de Hochschild est 1.*

■ **Vérification de la métrisabilité**

metris[]

```
{f     - f    , -f     + f    , -f    , f    , -f    , f    }
  1,2     2,2     1,2     2,2     2,3    2,3     3,3    3,3
```

f[1,2]:=f[2,2]; f[2,3]:=0; f[3,3]:=0;

metris[]

{}

matf[]

```
f       f       f
 1,1     2,2     1,3

f       f
 2,2     2,2    0

f
 1,3    0       0
```

● *La loi d'algèbre 2.2 est donc métrisable.*

▩ **loi d'algèbre (1.1)**

For[i=1,i<=n,i++,mu[x[1],x[i]]=x[i];mu[x[i],x[1]]=x[i]];
mu[x[2],x[2]]=x[3];
mu[x[2],x[3]]= 0;
mu[x[3],x[2]]=0;
mu[x[3],x[3]]=0;

■ **Calcul de la dimension du 2ème groupe de cohomologie**

DimB2[x]

7

DimZ2[x]

9

● *La dimension du deuxieme groupe de cohomologie de Hochschild est 2.*

■ **Vérification de la métrisabilité**

metris[]

```
{f     - f    , -f     + f    , -f    , f    , -f    , f    }
  1,3     2,2     1,3     2,2     2,3    2,3     3,3    3,3
```

f[1,3]:=f[2,2]; f[2,3]:=0; f[3,3]:=0;

metris[]

{}

matf[]

```
f       f       f
 1,1     1,2     2,2

f       f
 1,2     2,2     0

f
 2,2     0       0
```

La loi d'algèbre 1.1 est donc métrisable.

loi d'algèbre (1.2)

For[i=1,i<=n,i++,mu[x[1],x[i]]=x[i];mu[x[i],x[1]]=x[i]];
mu[x[2],x[2]]=0;
mu[x[2],x[3]]= 0;
mu[x[3],x[2]]=0;
mu[x[3],x[3]]=0;

Calcul de la dimension du 2ème groupe de cohomologie

DimB2[x]

```
5
```

DimZ2[x]

```
11
```

La dimension du deuxieme groupe de cohomologie de Hochschild est 6.

Vérification de la métrisabilité

metris[]

```
{-f    , f    , -f    , f    , -f    , f    }
   2,2    2,2     2,3    2,3     3,3    3,3
```

f[2,2]:=0; f[2,3]:=0; f[3,3]:=0;

metris[]

```
{}
```

matf[]

```
f       f       f
 1,1     1,2     1,3

f
 1,2     0       0

f
 1,3     0       0
```

La loi d'algèbre 1.2 n'est donc pas métrisable.

ANNEXE : le programme

❁ *le programme permet d'écrire les relations d'associativité à l'aide de la procédure "assoc[]", et d'effectuer un changement de base à l'aide de "ChangeBase". La loi d'algèbre se définit, dans une base {x[1],...,x[n]}, par ses constantes de structures qui sont les valeurs de mu[x[i],x[j]].*

```
BeginPackage["ALAS`"]

SetBase[x_Symbol] := base = Array[x, {n}];

(*  présentation de la loi  *)
mu[0, x_] := 0;
mu[x_, 0] := 0;
mu[x_+y_, z_] := mu[x, z] + mu[y, z]
mu[z_, x_+y_] := mu[z, x] + mu[z, y]
mu[a_ x_, y_] := a mu[x,y]
mu[x_, a_ y_] := a mu[x,y]
mu[v_, 1] := v; (* v à la puissance 1 *)
mu[v_, k_Integer?((#>1)&)] := (* v à la puissace k *)
   Module[{i,p},
      p=1;
      For[i=1, i<=k, i++,
         p = mu[v, p]];
      p
   ]


mu[b_List:base, lr_List:{}] :=
   Module[{i, j, muij},
      For[i = 1, i <= n, i++,
         For[j = 1, j <= n, j++,
            muij = Simplify[mu[b[[i]], b[[j]]] /. lr];
            If[(!NumberQ[#]||#!=0)&[muij],
               Print[b[[i]], " * ", b[[j]], " = ", Collect[Expand[muij], b]]
            ]
         ]
      ]
   ];

(*  conditions d'associativité  *)
assoc[i_Integer, j_Integer, k_Integer] :=
   Module[{prod1, prod2, lrel},
      prod1 = mu[mu[base[[i]], base[[j]]], base[[k]]];
      prod2 = mu[base[[i]], mu[base[[j]], base[[k]]]];
      lrel = Expand[Flatten[CoefficientList[prod2 - prod1, base]]];
      ordena[Union[Select[lrel, (!NumberQ[#]||#!=0)&]]]
   ]
assoc[] :=
```

```
  Module[{lrel,i,j,k},
      lrel = {};
      For[i=1, i<=n, i++,
        For[j=1, j<=n, j++,
          For[k=1, k<=n, k++,
            lrel = ordena[Union[lrel, assoc[i,j,k]]];
          ]
        ]
      ];
      lrel
  ]

(*  conditions de nilpotence  *)
nil[k_Integer] :=
  Module[{lrel,i},
      lrel={};
      For[i=1, i<=n, i++, lrel=Union[lrel, nil[base[[i]], k]]];
      lrel
  ]
nil[i_Integer, k_Integer] := nil[base[[i]], k];
nil[v_, k_Integer] :=
  Module[{lrel},
      lrel = Expand[Flatten[CoefficientList[mu[v, k], base]]];
      ordena[Union[Select[lrel, (!NumberQ[#]||#!=0)&]]]
  ]

(*  présentation ordonnée de la liste des conditions  *)
ordena[l_List] := Sort[l, menor]
menor[x_, y_] := size[x] <= size[y]
size[x_]:=
  Module[{},
    Switch[Head[x],
      Plus, Return[1000*Length[x]],
      Times, Return[100*Length[x]],
      Power, Return[10*Length[x]]
    ];
    0
  ]
var[le_] := Reverse[Union[Flatten[Variables[le]]]]

(*  Changement de base : écriture de la loi dans une nouvelle base  *)
ChangeBase[cb_List, x_:x, y_:y, le_:{}] :=
  Module[{lx, ly, i, j, ecb, temp},
      lx = Table[x[i], {i, 1, n}];
      ly = Table[y[i], {i, 1, n}];
      ecb=cb;
      If[Length[cb] < n,
        y[x] = First[Solve[ecb, ly]];
        For[i = 1, i <= n, i++,
          temp = ReplaceAll[y[i], y[x]] ;
```

```
            If[temp == y[i], ecb = Union[{y[i]==x[i]}, ecb]]
        ]
      ];
      ecb = Union[cb, le];
      x[y] = First[Solve[ecb, lx]];
      y[x] = First[Solve[ecb, ly]];
      For[i = 1, i <= n, i++,
        For[j = 1, j <= n, j++,
          temp = mu[y[i] /. y[x], y[j] /. y[x], x];
          mu[y[i], y[j]] = Collect[Expand[temp /. x[y]], ly];
        ]
      ]
  ]


Coord[i_Integer, v_, b_:x] := v /. Table[b[j]->If[j==i, 1, 0], {j,1,n}]
Coord[v_, b_:x] := Table[Coord[i,v,b], {i,1,n}]
Coord[0, b_:x] := Table[0, {i,1,n}]

Desp[ec_]:=
Module[{ecx, lv, v, rs},
      ecx=Expand[ec];
      lv=Reverse[Flatten[Variables[ecx]]];
      If[Length[lv]>0,
            v=lv[[1]];
            rs=Flatten[Solve[ecx==0, v]][[1]];
            Set[Evaluate[rs[[1]]], rs[[2]]]
      ]
]

(* Calcul la dimension de l'espace des 2-cobords de Hochschild *)
DimB2[b_:x]:=
Module[{A, a, i, j, f, Df, baseB},
  baseB=Array[b, {n}];
  A=Array[a, {n,n}];
  f[v_]:=Coord[v,b] . A . baseB;
  Df[i_Integer, j_Integer] := mu[f[b[i]], b[j]] +
                    mu[b[i], f[b[j]]] -
                      f[mu[b[i], b[j]]];
 For[i=1, i<=n, i++,
    For[j=1, j<=n, j++,
      Map[Desp, Coord[Df[i,j]]]
    ]
  ];
  n^2-Length[Flatten[Variables[A]]]
]

(* Calcul la dimension de l'espace des 2-cocycles de Hochschild *)
DimZ2[b_:x]:=
Module[{A, TA, a, i, j, k, f, Df, baseB},
```

```
   A=Array[a, {n,n,n}];
   baseB=Array[b, {n}];
   TA=Transpose[A, {1,3,2}];
   f[u_, v_]:=Coord[u,b] . TA . Coord[v,b] . baseB;
   Df[i_Integer, j_Integer, k_Integer] :=
      mu[f[b[i], b[j]], b[k]] -
      mu[b[i], f[b[j],b[k]]] +
      f[mu[b[i], b[j]], b[k]] -
      f[b[i], mu[b[j], b[k]]] ;
   For[i=1, i<=n, i++,
      For[j=1, j<=n, j++,
         For[k=1, k<=n, k++,
           Map[Desp, Coord[Df[i,j,k]]]
         ]
       ]
   ];
   Length[Flatten[Variables[A]]]
];

(* matrice des dérivations : étude des algèbres caractéristiquement nilpotentes*)
CarNil[a_Symbol, b_:x]:=
Module[{A, i, j, f, Df, baseB, sol},
   Clear[a];
   Format[a[i_,j_]] := Subscripted[a[i,j]];
   baseB=Array[b, {n}];
   A=Array[a, {n,n}];
   f[v_]:=Coord[v,b] . A . baseB;
   Df[i_Integer, j_Integer] := mu[f[b[i]], b[j]] +
                               mu[b[i], f[b[j]]] -
                                  f[mu[b[i], b[j]]];
   For[i=1, i<=n, i++,
      For[j=1, j<=n, j++,
         Map[Desp, Coord[Df[i,j]]]
       ]
   ];
   A = MatrixPower[A,n];
   sol={};
   For[i=1, i<=n, i++,
      For[j=1, j<=n, j++,
         If [!NumberQ[A[[i,j]]] || A[[i,j]]!=0,
              AppendTo[sol,Factor[Expand[A[[i,j]]]]]]
       ]
   ];
   ordena[sol]
]

(* Définition d'une forme bilinéaire symétrique invariante sur une loi d'algèbre *)
fb[0, x_] := 0;
fb[x_, 0] := 0;
fb[x_+y_, z_] := fb[x, z] + fb[y, z]
```

```
fb[z_, x_+y_] := fb[z, x] + fb[z, y]
fb[a_ x_, y_] := a fb[x,y]
fb[x_, a_ y_] := a fb[x,y]

initbil[]:=Module[{i,j},
        Array[f,{n,n}];
        For[i=1,i<=n,i++,
        For[j=i,j<=n,j++,
                fb[x[i],x[j]]=f[i,j];
                f[j,i]=f[i,j];
                fb[x[j],x[i]]=f[j,i]]];
        ]

(*  conditions d'invariance  *)
metris[i_Integer, j_Integer, k_Integer] :=
   Module[{prod1, prod2, lrel},
        prod1 = fb[mu[base[[i]], base[[j]]], base[[k]]];
        prod2 = fb[base[[i]], mu[base[[j]], base[[k]]]];
        lrel = Expand[Flatten[CoefficientList[prod2 - prod1, base]]];
        ordena[Union[Select[lrel, (!NumberQ[#]||#!=0)&]]]
        (* lrel = Expand[prod2 - prod1]; lrel*)
   ]

(* Conditions de métrisabilité *)
metris[] :=
   Module[{lrel,i,j,k},
        initbil[];
        lrel = {};
        For[i=1, i<=n, i++,
           For[j=1, j<=n, j++,
              For[k=1, k<=n, k++,
                 lrel = Union[lrel, metris[i,j,k]];
            ]
          ]
       ];
        lrel
   ]

(* affichage de la matrice de la forme bilinéaire symetrique invariante *)
matf[]:=Module[{i,j}, Print[MatrixForm[Table[f[i,j],{i,n},{j,n}]]]]

End[]
EndPackage[]
```

Desingularisation et parmétrisation des courbes algébriques

- *On applique l'algorithme de Goze pour desingulariser les courbes algébriques planes dans C^2 d'equations f(x,y)=0 où f(x,y) est un polynôme de C[x,y] . voir les articles de Michel Goze "étude locale des courbes algébriques planes", Astérisque(1983) et Robert Lutz et Michel Goze "courbes algébriques" Actes des journées SMF "Mathématiques finitaires et Analyse non standard" luminy 1985, publications Paris7.*
- *Le programme repose sur une suite de procédures qui permettent d'effectuer complétement la désingularisation et la parmétrisation à l'aide de la fonction finale "desingul". La procédure "homogène" extrait la partie homogène de plus bas degré du polynôme définissant la courbe et "deg" donne son degré . A partir du polynôme homogène de plus bas degré, la procédure "tangente" donne les différentes directions tangentielles, ce qui correspond au premier vecteur dans la décomposition pour chaque direction. La transformation du polynôme correspondante s'effectue à l'aide de "transfo". La procédure "désingularise" réalise complétement la désingularisation de façon récursive et donne la paramétrisation de chaque branche.*

Le programme

In[1]:=

```
homogene[f_,var_]:=Module[{u,m,mm,dx,dy,i,j,n},
      u=0;
      dx=Exponent[f,var[[1]]]+1;dy=Exponent[f,var[[2]]]+1;n=0;
      m=CoefficientList[f,var];
      For[i=1,i<=dx,i++,If[m[[i]]=={},m[[i]]=Table[0,{dy}]]];
      While[u==0,
      For[i=1,i<=dx,i++,
       For[j=1,j<=dy,j++,
       If[i+j-2==n,u=u+m[[i]][[j]]var[[1]]^(i-1) var[[2]]^(j-1)]]];
       n=n+1];
      u]
```

In[2]:=

```
deg[f_,var_]:=Exponent[f/.{var[[1]]->lbda var[[1]],
      var[[2]]->lbda var[[2]]},lbda ];
```

In[3]:=

```
tangente[f_,var_]:=Module[{u,m,s},
      m={};
      u=homogene[f,var];
      Off[Solve::svars];s=Solve[u==0,var];On[Solve::svars];
      Do[m=Append[m,var/.s[[i]]],{i,1,Length[s]}];
      m=m/.{var[[1]]->1,var[[2]]->1}]
```

In[4]:=

```
transfo[f_,var1_,var2_]:=Module[{u,r,i,j,lu,c,t,d,m,mat,uu},
      u=Union[tangente[f,var1]];
      r={};t={};lu=Length[u];mat={};
      d=deg[homogene[f,var1],var1];
      Do[uu=N[Abs[u[[i,1]]]];c=If[uu==0,1,0];
      mat=Append[mat,m={{u[[i,1]],c},
```

```
                    {u[[i,2]],If[c==0,1,0]}}];
        r=Append[r,{
        var1[[1]]->m[[1,1]]var2[[1]]+m[[1,2]]var2[[1]] var2[[2]],
        var1[[2]]->m[[2,1]]var2[[1]]+m[[2,2]]var2[[1]] var2[[2]]}],
        {i,1,lu}];
        Do[t=Append[t,Expand[(f/.r[[j]])/var2[[1]]^d]],{j,1,lu}];
          {t,mat}];
```

In[5]:=

```
désingularise[ls_,var_]:=Module[{lsm,h,lv,r,V,LV,i,j,to},
        lsm=Length[ls[[1]]];lv={};
        Do[
        If[(deg[h=homogene[ls[[1,i]],var],var]==1),
        Off[Solve::svars];
        r=Solve[ls[[1,i]]==0,var];
        On[Solve::svars];
        V=var/.r[[1]]/.{var[[1]]->t,var[[2]]->t};
        If[Det[ls[[2,i]]]=!=0,
            V=ls[[2,i]].{V[[1]],V[[1]] V[[2]]}];
        lv=Append[lv,V],
        to=transfo[ls[[1,i]],var,var];
        LV=désingularise[to,var];
        If[Det[ls[[2,i]]]=!=0,
        Do[LV[[j]]=ls[[2,i]].{LV[[j,1]],LV[[j,1]] LV[[j,2]]}
          ,{j,1,Length[LV]}]];
        lv=Append[lv,LV]],{i,1,lsm}];lv=ExpandAll[lv];lv]
```

In[6]:=

```
desingul[f_,var_]:=désingularise[{{f},{{{0,0},{0,0}}}},var]
```

▩ Application des différentes procédures

In[7]:=

```
f=4x^5-7x^3 y^2+x^2 y^3+3x y^4-y^5+9x^7+10y^9+x^2 y^8;
```

In[8]:=

```
homogene[f,{x,y}]
```

Out[8]=

```
     5       3  2     2  3         4     5
4 x  - 7 x  y  + x  y  + 3 x y  - y
```

In[9]:=

```
deg[homogene[f,{x,y}],{x,y}]
```

Out[9]=

```
5
```

In[10]:=

```
tangente[f,{x,y}]
```

Out[10]=

```
                        1       1
{{-1, 1}, {-1, 1}, {-, 1}, {-, 1}, {1, 1}}
                        2       2
```

In[11]:=

g=x^3-y^2

Out[11]=

```
 3    2
x  - y
```

In[12]:=

transfo[g,{x,y},{x,y}]

Out[12]=

```
          2
{{x - y }, {{{1, 0}, {0, 1}}}}
```

▩ Exemples de désingularisation

❁ *On illustre les procédures par trois exemples: le fameux cusp, une courbe régulière et une courbe singulière avec deux branches*

In[13]:=

desingul[x^3-y^2,{x,y}]

Out[13]=

```
      2   3
{{{t , t }}}
```

In[14]:=

desingul[x-y^2,{x,y}]

Out[14]=

```
    2
{{t , t}}
```

In[15]:=

desingul[x^3+x^2-y^2,{x,y}]

Out[15]=

```
            2             2    3          2           2    3
{{{2 t + t , -2 t - 3 t  - t }, {2 t + t , 2 t + 3 t  + t }}}
```

Titre : **Faire des mathématiques avec un logiciel de calcul formel.**

Auteurs : Groupe calcul formel de Mulhouse composé de :
Lionel DARIE - Marc GILG - Jean LEFORT - Abdenacer MAKHLOUF -
Etienne MEYER - Jean PERRIN et Michel GOZE.

Mots-clés : Calcul formel - Mathématique - Logiciel Mathematica -
Logiciel Derive - Enseignement - Algebre - Calcul de limites - Algorithme.

Resumé : Cette brochure est une contribution modeste de quelques enseignants de mathématiques qui essayent d'integrer dans leur enseignement l'outil informatique par le biais des logiciels de calcul formel. L'essentiel d'un bon apprentissage du logiciel Mathematica est proposé dans la première partie de cette brochure.
Mathematica ou tout autre outil semblable tel Mapple ou Derive permet d'explorer à l'aide de ses différents registres (graphique, calcul approché, calcul formel) des notions de mathématiques. La deuxieme partie est consacrée aux applications et à l'etude au moyen du calcul formel de certaines notions (algèbre, équations différentielles, calcul de limites...).
Certains algorithmes utilisés par les logiciels de calcul formel sont rappelés dans la partie 3 pour éviter de faire jouer à l'ordinateur le seul role de boite noire. Avec l'avenement des logiciels de calcul formel, l'ordinateur prend aussi de plus en plus de place dans le domaine de la recherche mathématique, ceci est illustré dans la dernière partie par quelques exemples d'applications dans le domaine de l'algèbre.

Sommaire :

Public concerné : enseignants du secondaire et universitaires et toute personne voulant s'initier à un logiciel de calcul formel.

Editeur : IREM de Strasbourg (**Brochure S. 168**).

ISBN : 2-911446-04-6.

Prix de vente : 65 F (+ port).