Réflexions Méthodologiques en Calcul Formel

Nik Lygerōs, Olivier Marguin, Michel Mizony

Département de Mathématiques. Université Lyon I
69622 Villeurbanne cedex

Résumé : Nous montrons à partir de quelques exemples en Maple, tirés de notre enseignement ces deux dernières années en DEUG première et deuxième années à l'Université Lyon-I et en Classes Préparatoires (Math. Sup. et Math. Spé.) au Lycée du Parc et à celui de la Martinière-Monplaisir, comment le calcul formel peut s'intégrer dans l'enseignement des mathématiques et comment il peut changer notre rapport à celles-ci. Nous commençons par examiner quelques préjugés courants sur l'utilisation de Maple. Ensuite nous abordons différentes applications élémentaires mais riches en enseignements (nature du tracé d'une courbe et étude de la suite de Fibonacci) en insistant sur la nécessité d'une compréhension du fonctionnement interne d'un calcul formel («boîte noire» de la factorisation). Puis nous analysons quelques différences fondamentales qui existent sur le plan du lexique entre un langage de bas niveau et un calcul formel. Enfin nous mettons en évidence, grâce à l'exemple des matrices magiques, l'intérêt heuristique de l'approche formelle. En annexe, nous développons deux exemples plus complexes du point de vue technique, comme un problème de factorisation et la surinterprétation fractale des commentaires de Conway.

A) **Introduction** ou quelques préjugés sur Maple.

Si le calcul formel fait maintenant partie des programmes des classes préparatoires ainsi que de certaines filières de DEUG et de licence, il est loin d'avoir pris sa vraie place. A notre avis, cela vient de ce qu'il est mal connu, y compris parmi les enseignants. Nous commencerons ici par dénoncer trois idées reçues :

1) Le logiciel de calcul formel ne sait rien faire : dans ce cas, comment expliquer qu'en une dizaine de minutes, Maple est capable de résoudre toutes les questions d'une épreuve d'examen de mathématiques de 1 h 30, posée en DEUG première année en 1996 ?

2) Le logiciel sait tout faire : cette idée nous semble aussi dangereuse que la précédente, car elle conduit à vouloir utiliser le logiciel dans des domaines où il n'est pas bon, au prix de contorsions déraisonnables. A titre d'illustration, voici un exercice proposé en DEUG l'année dernière :

Enoncé : simplifier la fraction :

$$\frac{1+\cos a + i\,\sin a}{1+\cos b + i\,\sin b}\ .$$

Corrigé (pour Maple V version 3) : z := (1+cos(a)+I*sin(a))/(1+cos(b)+I*sin(b));
z := **convert**(z,tan); # on convertit en tangentes de l'arc moitié

z := **normal**(z); # on met sous forme normalisée
z := **subs**(a/2=x,b/2=y,z); # on pose x=a/2 et y=b/2
z := **simplify**(z,trig); # on simplifie
z := **convert**(z,exp); # on transforme en exponentielles
z := **combine**(z,exp); # on regroupe les exponentielles
z := **evalc**(z); # on met sous forme cartésienne
z := **subs**(x=a/2,y=b/2,z); # on revient aux variables a et b
z := **factor**(z); # on factorise

Beaucoup trop d'efforts pour obtenir finalement l'expression :

$$\frac{\cos\frac{a}{2}}{\cos\frac{b}{2}}\left(\cos\frac{a-b}{2}+i\,\sin\frac{a-b}{2}\right).$$

Nous avons mis au point le corrigé ci-dessus car les instructions usuelles permettant la mise sous forme cartésienne, ou sous forme trigonométrique, en utilisant ou non la multiplication par la quantité conjuguée ne donnaient pas de résultats satisfaisants. Nous considérons donc qu'il s'agit d'un type d'exercices à éviter car il ne présente pas d'intérêt dans l'apprentissage et l'utilisation du calcul formel.

3) L'usage du logiciel dispense de réfléchir : en réalité, c'est tout le contraire et l'on doit constamment s'interroger sur ce qu'on est en train de faire. A cet égard, l'exercice suivant est instructif :

Enoncé : exprimer $\tan 6t$ en fonction de $\tan t$.

Tentative de solution (en Maple V version 3) :

y := tan(6*t);
y1 := **expand**(y); # on exprime y en fonction de sin(t) et cos(t)
y2 := **convert**(y1,tan); # on transforme sin(t) et cos(t) en fonction de tan(t/2)

On obtient $\tan 6t$ en fonction de $\tan\frac{t}{2}$. Mais ensuite on a beau faire : impossible de faire apparaître $\tan t$.

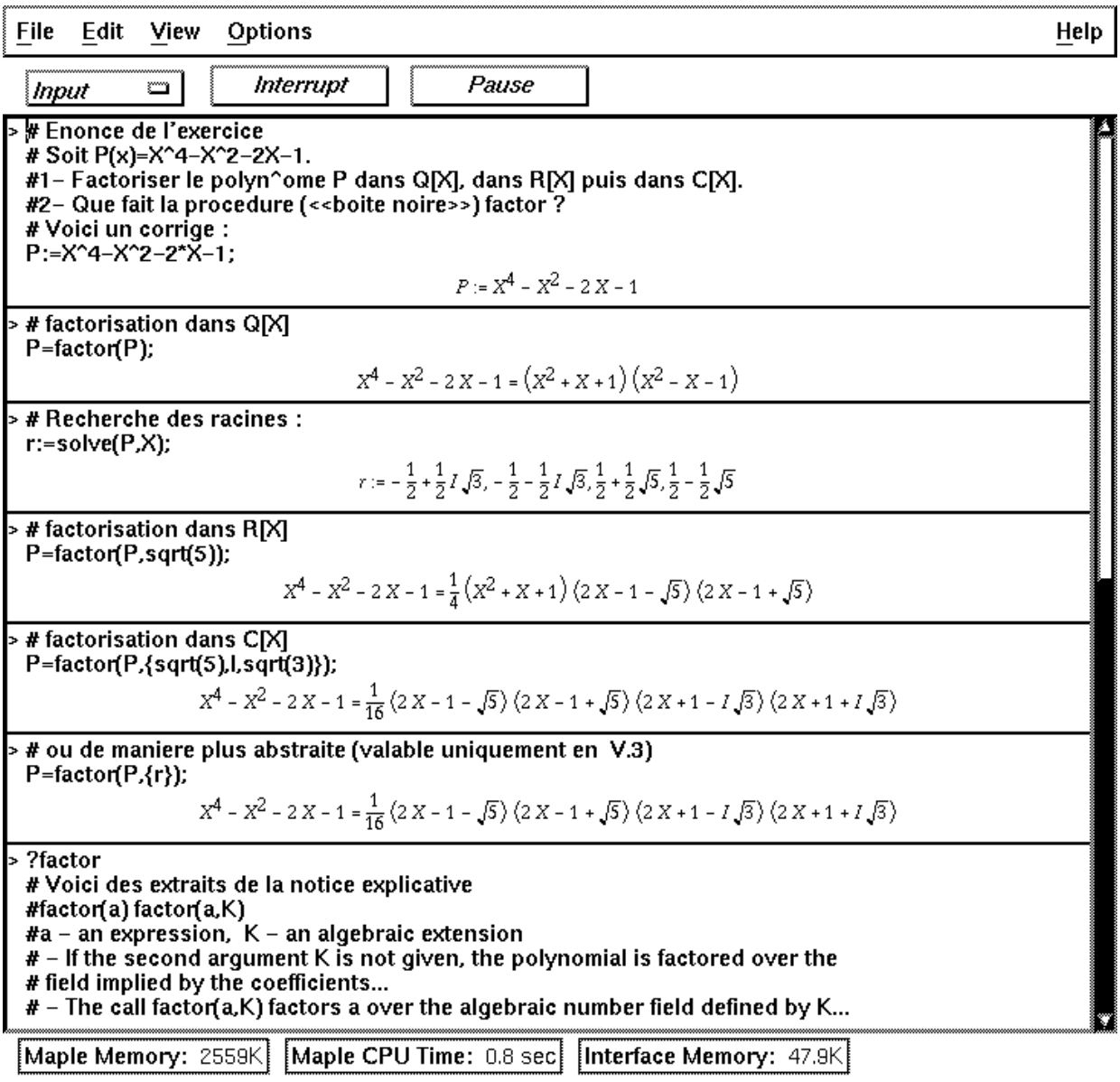
Quand nous avons posé cet exercice l'an dernier en DEUG, la plupart des étudiants s'obstinèrent en vain.

Or si l'on prend du recul sur notre tentative de solution, on s'aperçoit qu'on est tout simplement allé trop loin : on a exprimé $\tan 6t$ en fonction de $\tan\frac{t}{2}$, ce qui revient à exprimer $\tan 12t$ en fonction de $\tan t$. Il aurait fallu partir de $\tan 3t$. D'où :

Corrigé (pour Maple V version 3) :

y := tan(3*u);
y1 := **expand**(y); # on exprime y en fonction de sin(u) et cos(u)
y2 := **convert**(y1,tan); # on transforme sin(u) et cos(u) en fonction de tan(u/2)
u := 2*t; y2; # on fait le changement de variable : u = 2t

B) **La procédure factor** ou la «boîte noire» de la factorisation d'un polynôme.
Voici une session en Maple V.3. sur la factorisation :



- La première surprise est de constater trois décompositions distinctes de ce polynôme. Le polynôme très simple à factoriser à la main permet une vérification rapide de la justesse du résultat affiché. Il est évident que d'autres exercices de factorisation ne se prêtent pas à des vérifications immédiates; l'on propose alors d'autres démarches de vérifications.

- Le recours à l'aide par le menu Help (en haut à droite de la session de travail) ou par ?nom de la procédure est en général effectué, mais essentiellement dans le but de bien utiliser la procédure, en particulier pour entrer les paramètres de la procédure. À ce propos, il est important de signaler que Maple est une immense bibliothèque de définitions mathématiques accessibles par l'aide en ligne. Rares sont les utilisateurs qui s'intéressent aux passages de l'aide concernant le contenu mathématique sous-tendu par la procédure. Parfois, dans notre pratique de recherche, il nous arrive parfois d'y avoir recours pour vérifier les définitions

d'objets mathématiques ou les différentes conventions adoptées, par exemple la définition de la transformation de Fourier ou celle de la fonction de Bessel.

- Il s'agit alors de comprendre ce qui est dit de théorique sur la procédure et de voir des concepts et théorèmes mathématiques. Pour cet exemple, il faut revenir à la définition du corps de nombres engendré par les coefficients d'un polynôme et au concept d'extension de corps. La «boîte noire» de la factorisation commence à devenir moins magique, elle devient «boîte grise».
- La deuxième surprise est de réaliser que la théorie des corps de nombres est riche et utile en calcul formel.

Il est toujours possible de faire afficher à l'écran le contenu interne de la procédure **factor** grâce aux instructions : **interface**(**verboseproc** = **2**); et **print**(**factor**);

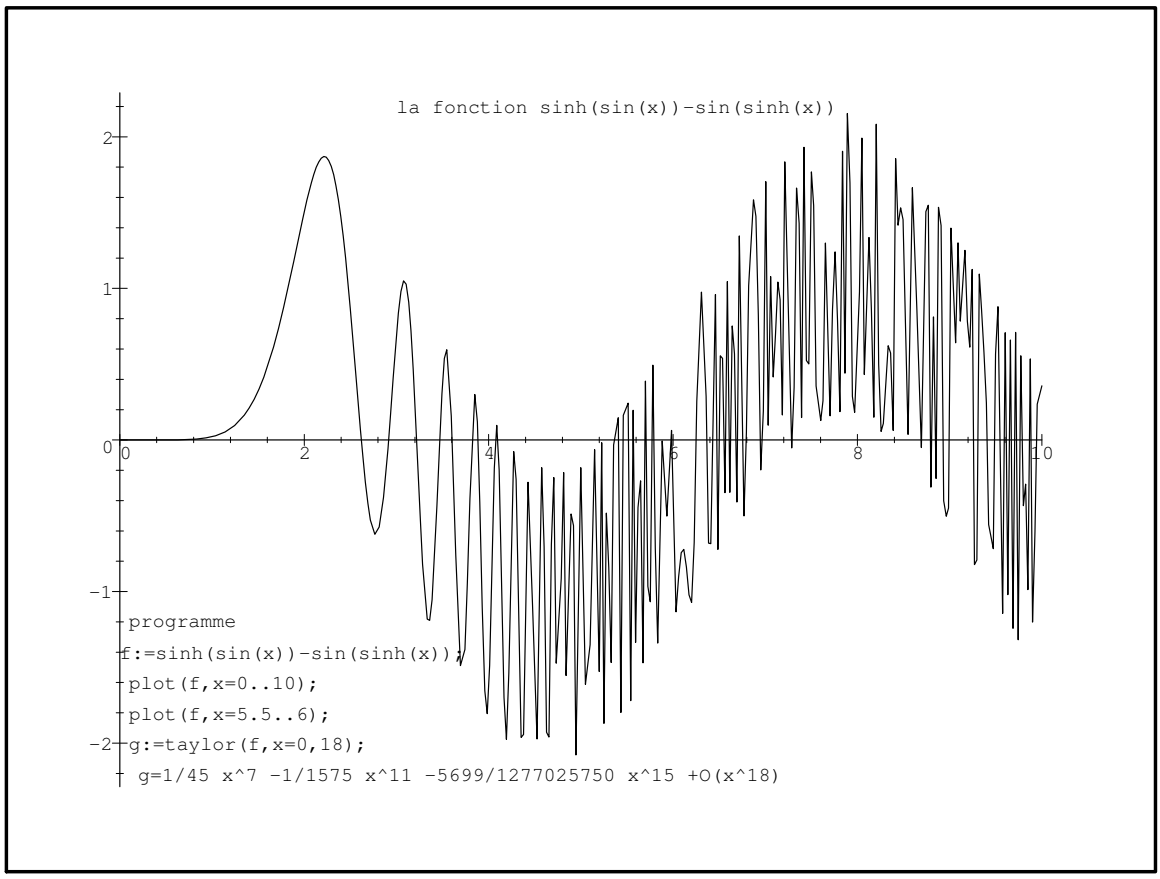
Peut-on aller plus loin, en se posant la question de factoriser sur **R** un polynôme à coefficients rationnels ou réels ? A priori, il n'y a pas de procédure spécifique en Maple, même dans les «bibliothèques annexes» (Faire?**share** pour connaître ces bibliothèques).

Notons dans un premier temps que la décomposition sur une extension de corps fonctionne bien et que pour obtenir la décomposition sur **C**, il suffit que la procédure **solve** fonctionne.

Partant de là, nous avons essayé d'écrire une procédure nous donnant directement une décomposition sur **R** d'un polynôme, en utilisant les procédures existantes, en particulier **solve** et **factor**, procédures qui ont des limites dans leur utilisation. (voir Annexe)

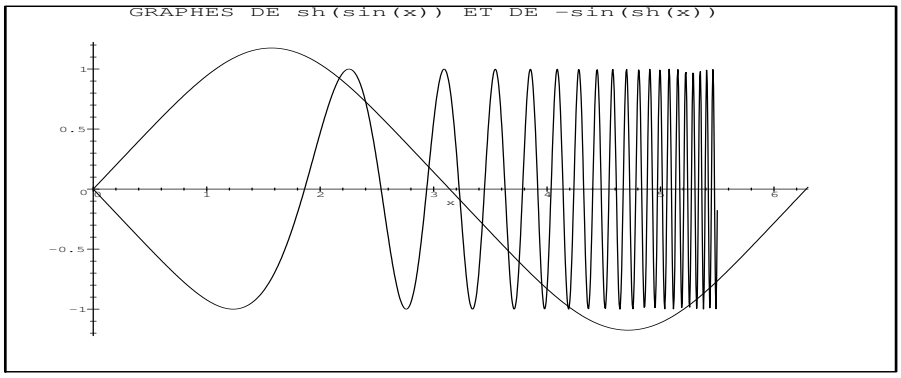
C) **Le tracé d'une courbe** ou la nécessité de la vérification.

Un exercice classique, lors de l'apprentissage des développements limités, consiste à faire calculer celui de la fonction $x \to \sinh(\sin(x)) - \sin(\sinh(x))$ en zéro, sans préciser l'ordre. Cet exercice, fastidieux à la main, est instantané en utilisant la procédure **taylor**(f,x=a,n). Mais, outre les vérifications formelles permettant de s'assurer que le premier terme du développement est bien en x^7, une vérification graphique par le tracé de la fonction est instructive. Le graphe semble très accidenté. Les questions qui se posent sont d'une part de comprendre ce graphe et d'autre part de se persuader que le logiciel répond assez correctement.



La fonction étant indéfiniment dérivable, une première vérification consiste à faire un grossissement de zones du tracé (par exemple la zone $x \in [5.5, 6]$). La nature lisse de la fonction est alors manifeste.

Mais les vérifications les plus importantes sont théoriques : le graphe est borné, l'établir en montrant que la fonction est bien bornée par $\sinh(1) + 1$. Le graphe présente une pseudo-période longue, montrer que celle-ci provient du terme $\sinh(\sin(x))$, alors que la pseudo-période courte provient du deuxième terme de la fonction. Etc...



D) **fibonacci** ou de la connaissance à l'efficacité.

La suite qui génère les nombres de Fibonacci se définit de la manière suivante : $F_0 = F_1 = 1$ et $F_{n+2} = F_{n+1} + F_n$ pour $n \geq 2$. Cette définition relue avec l'esprit de Maple permet d'écrire la procédure récursive suivante :

firec:=**proc**(n:integer)
options remember;
if n=0 **or** n=1 **then** 1
else firec(n-1)+firec(n-2)
fi:
end:

et firec(n) donne le nième nombre de Fibonacci. Seulement à l'aide de cette méthode les calculs deviennent très longs pour des grandes valeurs de n. Pour aller plus loin nous allons utiliser une approche matricielle en exploitant l'égalité : $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{pmatrix}$ pour écrire la procédure suivante :

fimat:=**proc**(n:integer)
local M,F:
with(linalg);
M:=**matrix**([[1,1],[1,0]]):
F:=**evalm**(M^n):
F[1,1];
end:

La *procédure* fimat(n) est bien plus performante que la précédente, et c'est d'ailleurs la même méthode qui a été utilisée pour la fonction de Maple **fibonacci**(n) que l'on trouve dans le *package* **combinat**.

Cependant nous désirons mieux comprendre cette suite, et pour cela nous allons considérer sa fonction génératrice Φ. Un simple calcul dans les séries formelles permet de l'expliciter. En effet :

$\Phi(t) = \sum_{n=0}^{+\infty} F_n t^n = 1 + t + \sum_{n=2}^{+\infty}(F_{n-1} + F_{n-2})t^n = 1 + t\sum_{n=1}^{+\infty} F_{n-1}t^{n-1} + t^2\sum_{n=2}^{+\infty} F_{n-2}t^{n-2}$
$\Phi(t) = 1 + t\Phi(t) + t^2\Phi(t)$ et donc $\Phi(t) = -1/(t^2 + t - 1)$.

En Maple V.4 ceci se fait aisément par la série d'instructions suivante :

with(share); accès à la *sharelibrary*.
readshare(gfun,analysis): lecture du *package* gfun dans le répertoire analysis.
with(gfun); appel du *package* gfun.
guessgf([1,1,2,3,5],t); application à la liste des premiers nombres de Fibonacci de la procédure **guessgf** qui «devine» la fonction génératrice d'une liste donnée. Cela peut aussi se faire de la manière suivante :
rsolve(F(n+2)=F(n+1)+F(n),F(0..1)=1,F,'genfunc'(x));

Ensuite pour obtenir le nième nombre de Fibonacci, il suffit de lire le nième coefficient d'un développement de Taylor à l'ordre $n+1$, grâce à : **coeff**(**taylor**(Phi,t,n+1),t,n);

Cette méthode est un peu plus lente que la précédente mais elle permet d'aller plus loin lorsque l'on désire connaître, non pas le nombre de Fibonacci, mais le nombre de ses chiffres.

En effet en décomposant en éléments simples Φ et en désignant par $-\alpha$ et $-\beta$ les racines de $1-t-t^2=0$, l'on obtient assez facilement la formule: $F_n=(\beta^{n+1}-\alpha^{n+1})/(\beta-\alpha)$ où $\alpha=(1-\sqrt{5})/2$ et $\beta=(1+\sqrt{5})/2$. Enfin l'évaluation en flottants de F_n est extrêmement rapide, et ce, même pour des valeurs de n inaccessibles par les autres méthodes, et donne le nombre exact de ses chiffres.

E) **Lexique** ou les mille et un mots

Une des plus grandes différences conceptuelles entre un langage de calcul formel et un langage de bas niveau, c'est le lexique. Plus exactement la taille du lexique. Un langage est souvent constitué d'un petit nombre d'instructions à partir desquelles l'utilisateur implémente ses algorithmes. Tandis qu'un langage de calcul formel peut facilement contenir un millier de mots. Cette différence génère rapidement un problème d'ordre culturel. Un bon programmeur d'un langage de bas niveau connaît de nombreux algorithmes alors qu'un bon programmeur de calcul formel connaît avant tout un grand nombre de fonctions. La culture se déplace donc dans le sens d'un enrichissement du vocabulaire du programmeur. Ainsi une bonne approche du calcul formel, lorsque l'on désire résoudre un problème donné, n'est pas de programmer à partir de quelques instructions mais de se demander quelles fonctions du logiciel peuvent être utiles. Une conséquence directe de cette différence, c'est la taille des programmes. En effet un programme de calcul formel est généralement beaucoup plus court que celui d'un langage de bas niveau qui effectue la même tâche. Le calcul formel conduit à une programmation sémantiquement plus dense et qui est plus proche du langage mathématique.

Cette richesse de vocabulaire procure à l'utilisateur une grande liberté de manœuvre et lui permet de développer son propre style de programmation. En règle générale, la solution d'un problème s'écrit en trois étapes :

(i) mise en équation du problème,

(ii) résolution,

(iii) exploitation des résultats.

L'étape (ii) fait généralement appel à une fonction toute faite du logiciel, sorte de *boîte noire* sur laquelle on n'a aucun contrôle (par exemple **solve** ou **simplify**). En revanche, les étapes (i) et (iii), qui servent respectivement à préparer les données et à extraire les résultats, requièrent une certaine habileté de la part du programmeur: elles consistent en des réécritures d'expressions, souvent à l'aide des opérateurs **subs**, **seq** et **convert**.

Illustrons ceci en établissant grâce à Maple, pour un n donné, une formule d'approximation d'ordre $2n$ de la dérivée:

$$f'(x_0)=\sum_{k=1}^{n} a_k \frac{f(x_0+kh)-f(x_0-kh)}{h}+\mathrm{o}(h^{2n-1})$$

valable pour toute fonction numérique f supposée $2n$ fois continument dérivable au voisinage du point x_0. Par exemple, pour $n=5$, Maple trouve:

$$a_1=\frac{5}{6}\ ,\ a_2=-\frac{5}{21}\ ,\ a_3=\frac{5}{84}\ ,\ a_4=-\frac{5}{504}\ ,\ a_5=\frac{1}{1260}$$

```
# mise en équations du problème :
n := 5;
expr := (f(x0+k*h)-f(x0-k*h))/h;
y := convert(taylor(expr,h=0,2*n),polynom);
equations :=seq(expr=y,k=1..n);
variables :=seq((D@@(2*k-1))(f)(x0),k=1..n);
# résolution :
s := solve({equations},{variables});
# exploitation des résultats :
derivee := subs(s,D(f)(x0));
'D(f)(x0)' = combine(derivee) + o(h^(2*n-1)); # formule cherchée
```

F) **alias** ou un outil de démonstration.

En Maple la structure syntaxique de l'**alias** est la suivante :
alias(nom=expression);
où l'expression appartient au langage de Maple et n'est pas une constante numérique. Et la désactivation de l'**alias** se fait simplement par : **alias**(nom=nom);

Cette *instruction* permet d'écrire de manière abrégée, des expressions moins évidentes, syntaxiquement parlant. Par exemple, grâce à elle, l'on peut noter simplement la matrice identité (indépendamment de la dimension) par Id, en faisant : **alias**(Id=& *());

L'activation de l'**alias** affiche à l'écran tous les **alias** précédents. Ainsi I (le nombre imaginaire en Maple) qui est un **alias** préexistant apparait lors de la première création d'un **alias**.

Présentée de cette manière l'instruction **alias** semble n'être qu'une *affectation* (que l'on obtient par **:=**, pourtant nous allons montrer qu'elle est bien plus intéressante. Il ne s'agit pas d'une **macro** car contrairement à celle-ci les abréviations de l'**alias** n'affectent pas les *expressions* ou les *procédures* mais uniquement leur affichage.

L'**alias** prend toute sa puissance lorsqu'il est utilisé avec **RootOf**. En premier lieu cette utilisation permet de mener des calculs algébriques de façon élégante. Considérons un exemple dû à C. Gomez. Il s'agit de montrer l'égalité suivante :
$(\phi^5+1)/(\phi^2-1) = \phi^8/(\phi^3+1) - 3/2$ où ϕ est le nombre d'or.

Pour effectuer cela en Maple, il suffit de déclarer ϕ comme racine de l'équation x^2-x-1, grâce à : **alias**(phi=**RootOf**(x^2-x-1,x));
et de faire une *évaluation algébrique*, avec **evala**, de la différence des deux termes.

En second lieu, l'utilisation combinée de **alias** et **RootOf** permet d'effectuer de véritables démonstrations. Illustrons ceci par un exemple.

Montrer que :

$$\prod_{k=0}^{16}(z+\alpha^k) = \prod_{k=0}^{16}(1+z\alpha^k)$$

où α est une racine 17ème de l'unité.

Démonstration: -Pour $\alpha = 1$ l'égalité est une identité.

-Pour $\alpha \neq 1$, l'on déclare α comme racine de : $\sum_{k=0}^{16} x^k = 0$, en écrivant :

alias(alpha=**RootOf**(**sum**(x^m,m=0..16),x)); Ensuite l'on évalue algébriquement la différence des deux produits par :

evala(**product**((z+alpha^k),k=0..16)-**product**((1+z*alpha^k),k=0..16)); afin de montrer qu'elle est nulle. □

Comme le polynôme qui définit α est irréductible, α représente de manière *générique* toutes les racines complexes (ici 16). Ainsi Maple *démontre* d'un seul coup les 16 propositions. Une manière de procéder difficile, pour ne pas dire impossible, à mettre en place avec un langage de bas niveau (plus près du processeur).

G) **proc** ou de la modularité à la globalité

En Maple les *procédures* les plus élémentaires sont les *opérateurs fonctionnels*. Ainsi la notation flèche $->$ crée une fonction de zéro ou plusieurs variables, et la fonction **unapply** permet d'obtenir une fonction à partir d'une expression algébrique.

Les *procédures* générales sont régies par la syntaxe suivante :

nom:=**proc**(x1:type1,x2:type2,...)
local l1,l2,...:
global g1,g2,...:
options op1,op2,...;
corps
end:

Du point de vue didactique il est intéressant de remarquer que si l'on remplace le **:** qui se trouve après le **end** par un **;** alors Maple affichera à l'écran sa «réécriture» de la procédure sous une forme normalisée. Par ailleurs cette syntaxe respecte bien la règle d'or «maplelienne» à savoir qu'une phrase en Maple se termine par **:** ou **;**, puisque la fin de la phrase *procédure* se situe après le **end**.

Pour notre part, nous considérons une *procédure* en Maple comme une «cellule informatique» qui doit être indépendante de son environnement. Aussi, si la déclaration d'une procédure nécessite des fonctions qui sont hors du noyau alors les accès bibliothèque et les appels de *package* devront se faire dans le corps de la procédure. Cela permet, entre autres, de rendre la procédure plus stable en cas d'utilisation de la fonction **restart** pour réinitialiser.

Comme les tables en Maple sont des tables de hachage (i.e. mode de gestion de la mémoire), le temps de recherche dans la table créée par l'option **remember** est indépendant du nombre d'entrées stockées. Ainsi il est recommandé de l'utiliser même si elle consomme beaucoup de mémoire.

Le fait que le corps de la procédure soit écrit de la même manière qu'en intéractif nous incite à développer la stratégie suivante lors de l'implémentation d'une procédure. Si la procédure requiert plusieurs lignes non triviales de programmation en Maple, alors il est plus judicieux de commencer à écrire le corps de la procédure et ce, sans se préoccuper de la syntaxe générale plutôt que de faire le contraire. En effet cette méthode permet de tester de façon interactive son algorithme, ligne par ligne, ce qui facilite grandemment la recherche d'erreurs. Alors que la méthode «naturelle» ne le permettra pas puisque Maple n'affichera aucun calcul tant qu'il n'aura pas «vu» le mot **end** ! Et même à ce moment-là, il ne sera pas aisé de lire pour quelles valeurs des variables locales, par exemple, la procédure créée ne fonctionne pas. Par contre une fois que le corps du programme fonctionne sur plusieurs valeurs l'on peut alors le parer de la syntaxe de la procédure.

La structure même de la procédure en Maple amène naturellement l'utilisateur à la notion de programmation modulaire. Ainsi, comme dans un langage de bas niveau, grâce aux procédures l'on peut programmer en Maple en créant des modules qui s'appellent les uns les autres afin de résoudre un problème globalement difficile.

H) **Calcul formel** ou une heuristique à définir.

En ce qui concerne la place qu'occupe le calcul formel au sein de l'enseignement universitaire, nous pensons qu'il reste un gros effort à faire pour l'intégrer aux autres disciplines. Pour ne parler que des Mathématiques, les atouts sont de taille : le calcul formel donne les moyens de résoudre des problèmes réels, nécessairement compliqués, et non pas seulement des questions pipées car prévues pour que "ça marche". Plus encore, il apporte un puissant moyen d'expérimentation.

Nous illustrerons ce dernier point grâce aux matrices magiques : on dit qu'une matrice carrée réelle est *magique* lorsque les sommes des coefficients de chaque ligne, de chaque colonne et de chacune des deux diagonales sont égales. On se convainc facilement que l'ensemble E_n des matrices magiques d'ordre n est un sous-espace vectoriel de $M_n(\mathbf{R})$. Mais quelle est sa dimension, et peut-on en donner une base? Comme Maple ne sait pas faire le calcul pour un n indéterminé, on choisit des valeurs particulières de n.

Solution (en Maple V version 3) :

```
with(linalg):
n := 3;  # on "expérimente" avec n = 3
x := vector(n^2);
M := matrix(n,n,x);
s := trace(M);  # on exprime que M est magique :
equations := seq(sum(M[i,'j'],'j'=1..n) = s, i=1..n),
seq(sum(M['i',j],'i'=1..n) = s, j=1..n),
sum(M['i',n+1-'i'],'i'=1..n) = s;
variables := seq(x[i], i=1..n^2);
A := genmatrix([equations], [variables]);  # génère la matrice du système
K := kernel(A, 'd'): d;  # donne la dimension de E_n
for v in K do matrix(n,n,v) od;  # affiche une base de E_n
```

Maple donne une dimension égale à 3 et pour base :

$$\begin{pmatrix} 2 & 2 & -1 \\ -2 & 1 & 4 \\ 3 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 1 \\ 2 & 0 & -2 \\ -1 & 0 & 1 \end{pmatrix}$$

Pour $n = 4, 5, 6, 7, 8, 9, 10$ il donne respectivement les dimensions 8, 15, 24, 35, 48, 63, 80. A partir de ces observations, on peut conjecturer que la dimension de E_n est égale à $(n-1)^2-1$, formule qu'on peut ensuite établir pour tout $n \geq 3$, par un raisonnement mathématique.

Mais nous pouvons aller un peu plus loin en exploitant ces resultats pour obtenir tous les carrés magiques d'ordre 3. Un carré magique d'ordre n est une matrice magique $n * n$ dont les coefficients sont les entiers de 1 à n^2. Nommons a, b, c les trois matrices qui forment la base du noyau obtenue.

a := **matrix**([[2, 2, −1], [−2, 1, 4], [3, 0, 0]]); b := **matrix**([[0, −1, 1], [1, 0, −1], [−1, 1, 0]]); c := **matrix**([[−1, 0, 1], [2, 0, −2], [−1, 0, 1]]);

À partir de cela nous pouvons écrire la forme générale d'un carré magique potentiel : CM := evalm($\phi * a + \chi * b + \psi * c$); De cette manière nous obtenons à l'affichage :

$$\begin{pmatrix} 2\phi - \psi & 2\phi - \chi & -\phi + \chi + \psi \\ -2\phi + \chi + 2\psi & \phi & 4\phi - \chi - 2\psi \\ 3\phi - \chi - \psi & \chi & \psi \end{pmatrix}$$

Il est facile de voir que ϕ est égal à 5 et que χ et ψ sont compris entre 1 et 9. Ainsi nous avons la matrice suivante :

$$\begin{pmatrix} 10 - \psi & 10 - \chi & -5 + \chi + \psi \\ -10 + \chi + 2\psi & 5 & 20 - \chi - 2\psi \\ 15 - \chi - \psi & \chi & \psi \end{pmatrix}$$

Maintenant il suffit d'éliminer les matrices à coefficients négatifs ou nuls grâce au test suivant :

```
test:=proc(N)
local i,j:
for i to n do
for j to n do if N[i,j] <= 0 then RETURN(false) fi; od:
od:
true
end:
```

Et de générer tous les carrés d'ordre 3 grâce au programme suivant dont le principe utilise des inégalités entre certains coefficients qui suffisent pour obtenir la consécutivité des coefficients !

```
ens:={seq(g,g=1..n^2)} minus {5}:
e:=0:
for χ in ens do
for ψ in ens minus {χ} do
if test(CM) and χ <> ψ
and χ+ψ > 5 and χ+ψ <> 10
and χ+2*ψ <> 15 and χ + 3 * ψ <> 20
and 2*χ + ψ <> 15 and 2*χ + 3 * ψ <> 25 then
e:=e+1: C[e]:=evalm(5*a+χ*b+ψ*c):
fi;
od:
od:
seq(evalm(C[d]),d=1..e);
```

Voici les 8 carrés magiques d'ordre 3 obtenus instantanément par Maple :

```
> read(magique);
Warning: new definition for   norm
Warning: new definition for   trace
```

$$\left[\begin{matrix} 4 & 9 & 2 \\ 3 & 5 & 7 \\ 8 & 1 & 6 \end{matrix}\right], \left[\begin{matrix} 2 & 9 & 4 \\ 7 & 5 & 3 \\ 6 & 1 & 8 \end{matrix}\right], \left[\begin{matrix} 6 & 7 & 2 \\ 1 & 5 & 9 \\ 8 & 3 & 4 \end{matrix}\right], \left[\begin{matrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{matrix}\right]$$

$$\left[\begin{matrix} 8 & 3 & 4 \\ 1 & 5 & 9 \\ 6 & 7 & 2 \end{matrix}\right], \left[\begin{matrix} 4 & 3 & 8 \\ 9 & 5 & 1 \\ 2 & 7 & 6 \end{matrix}\right], \left[\begin{matrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{matrix}\right], \left[\begin{matrix} 6 & 1 & 8 \\ 7 & 5 & 3 \\ 2 & 9 & 4 \end{matrix}\right]$$

Nous espérons que les quelques exemples traités ci-dessus montreront au lecteur que l'utilisation du calcul formel renouvellera profondément notre rapport à l'enseignement des mathématiques. Bien sûr, ces exemples sont des produits manufacturés qui masquent les nombreux allers-retours entre réflexion théorique et mise en pratique. Comme le calcul formel permet de par son langage une grande interaction entre la programmation et les mathématiques, il remet à l'honneur l'aspect experimental de celles-ci. Mais le calcul formel change aussi nos pratiques de recherche. Pour le lecteur intéressé, nous renvoyons à l'article de M. Mizony.

Références

Noam Chomsky : *Structures syntaxiques.*, éditions du Seuil, points 98, 1er trimestre 1979.

Jean-Paul Delahaye : *Les commentaires du mathématicien.*, Pour la Science, numéro 219, janvier 1996.

Umberto Eco : *Interprétation et surinterprétaion.*, éditions PUF, janvier 1996.

Philippe Fortin et Roland Pomès : *Premiers pas en Maple. Introduction à l'utilisation du calcul formel en mathématiques, physique et chimie.*, séries «Méthodes et Applications», éditions Vuibert, novembre 1994.

Claude Gomez, Bruno Salvy et Paul Zimmermann : *Calcul formel : Mode d'emploi.*, éditions Masson, février 1995.

Albert Lévine : *Exercices en Maple.* éditions ellipses, avril 1995.

Michel Mizony : *Le calcul formel dans ma pratique d'enseignant et de chercheur.*, Actes de l'université d'été : «Développer la recherche scientifique à travers l'étude de situations mathématiques», IREM de LYON, juillet 1996.

Paul Zimmermann : *Factorisation de polynômes : théorie et pratique.* Conférence Aleph & Géode, 29 avril 1997.

Annexe

Session sur une factorisation dans $\mathbf{R}$

```
> restart:interface(labelling=false);
                          # Un programme de factorisation sur R
  decomp:=proc(P:polynom,x)
  local X,r,rr,i,ii,testR,testc,Pr,Q,Qs,d,B;
  X:=x:r:=[solve(P,X)];
  r:=[seq(expand(simplify(combine(convert(r[i],exp),exp))),i=1..nops(r))];
                                                  # utile pour decomposer x^7+1
  rr:=[seq(evalc(r[i]),i=1..nops(r))];rr:=normalracine(rr);
                                                  # utile pour decomposer x^8+1
  testR:=1;testc:=1;
  for i to nops(r) do
  if type(op(i,r),function) then testR:=0;
                       # il y a un RootOf, la decomposition ne sera pas complete
  Q[i]:=subs(_Z=X,op(r[i]))
  elif type(op(i,r),complex)  and not(type(op(1,r[i]),function)) then
    if simplify(evalc(rr[i])-evalc(conjugate(rr[i])))=0 then
  Q[i]:=X-simplify(rr[i])                                        # racine reelle
   else
    Q[i]:=sqrt(combine(X^2-expand(simplify(2*evalc(Re(rr[i]))))*X+
  expand(simplify(abs(rr[i])^2)),trig)) fi            # racines complexes conjuguees
  else Q[i]:=1;testc:=0 fi                #la decomposition sera formelle ou partielle
  od;
   Qs:=[seq(Q[i],i=1..nops(r))];
  Pr:=product(Qs[ii],ii=1..nops(Qs));
                    # Pr n'est pas forcement sous la forme d'un polynome, d'ou ...
  B:=1:d:=nops(Pr);if d>1 and type(Pr,`*`) then
   for i to d  do if type(op(i,Pr),polynom) then
  B:=B*op(i,Pr) fi od fi;Pr:=B;                          #... recours a la recursivite
  if degree(Pr)=degree(P) then Pr:=normal(lcoeff(P)*Pr)
  else if testR*testc=0 then Pr:=Pr*quo(P,Pr,X) else
   Pr:=Pr*decomp(quo(P,Pr,X),X) fi; fi;
  subs(x=X,Pr);                                    # c'est la decomposition obtenue
  end:

        # tri des racines du polynome pour eviter des formes algebriques egales
  normalracine:=proc(L:list)
  local i,j,d,aa;
  d:=nops(L);if d=0 then [NULL] else
  for i to d-1 do  aa[i]:=L[i];
  for j from i+1 to d do
  if type(L[i],complex) and type(L[j],complex) then if
   abs(evalf(Re(L[i]-L[j])))<10^(-8) and abs(evalf(Im(L[i]+L[j])))<10^(-8)
   then aa[i]:=conjugate(L[j]) fi;fi;
  od;od;aa[d]:=L[d];
  [seq(aa[i],i=1..d)] fi; end:
```

Ce programme repose essentiellement sur la procédure **solve**. Voici quelques résultats obtenus:

> decomp(X^8+a*X^4+a-1,X);

$$\left(X^2-\sqrt{2}\,X+1\right)\left(X^2+\sqrt{2}\,X+1\right)\left(a-1+X^4\right)$$

> for i from 5 to 8 do X^i+1=decomp(1+X^i,X) od;

$$X^5+1=\frac{1}{8}\left(4\,X^2-2\,X-X\sqrt{5+\sqrt{5}}\,\sqrt{5-\sqrt{5}}+4\right)\left(2\,X^2-X+X\sqrt{5}+2\right)(X+1)$$

$$X^6+1=\left(X^2-\sqrt{3}\,X+1\right)(X^2+1)\left(X^2+\sqrt{3}\,X+1\right)$$

$$X^7+1=\left(X^2-2\cos\left(\frac{1}{7}\pi\right)X+1\right)\left(X^2-2\cos\left(\frac{3}{7}\pi\right)X+1\right)\left(X^2+2\cos\left(\frac{2}{7}\pi\right)X+1\right)(X+1)$$

$$X^8+1=\frac{1}{4}\left(2\,X^2-X\sqrt{2}\,\sqrt{2+\sqrt{2}}-X\sqrt{2}\,\sqrt{2-\sqrt{2}}+2\right)\left(X^2-\sqrt{2-\sqrt{2}}\,X+1\right)$$
$$\left(2\,X^2+X\sqrt{2}\,\sqrt{2+\sqrt{2}}-X\sqrt{2}\,\sqrt{2-\sqrt{2}}+2\right)\left(X^2+\sqrt{2+\sqrt{2}}\,X+1\right)$$

> decomp(3*u^3+2*u^2-3*u-1,u);

$$\frac{1}{243}\left(9\,u-2\sqrt{31}\,\cos\left(\frac{1}{3}\arctan\left(\frac{9}{65}\sqrt{1419}\right)\right)+2\right)$$
$$\left(9\,u+\sqrt{31}\,\cos\left(\frac{1}{3}\arctan\left(\frac{9}{65}\sqrt{3}\,\sqrt{473}\right)\right)+2+\sqrt{31}\,\sin\left(\frac{1}{3}\arctan\left(\frac{9}{65}\sqrt{3}\,\sqrt{473}\right)\right)\sqrt{3}\right)$$
$$\left(9\,u+\sqrt{31}\,\cos\left(\frac{1}{3}\arctan\left(\frac{9}{65}\sqrt{3}\,\sqrt{473}\right)\right)+2-\sqrt{31}\,\sin\left(\frac{1}{3}\arctan\left(\frac{9}{65}\sqrt{3}\,\sqrt{473}\right)\right)\sqrt{3}\right)$$

> decomp(X^5+X+1,X);

$$-\frac{1}{18}(X^2+X+1)\left(6\,X\left(100+12\sqrt{69}\right)^{1/3}+\left(100+12\sqrt{69}\right)^{2/3}+4-2\left(100+12\sqrt{69}\right)^{1/3}\right)\left(\right.$$
$$-3\,X^2\left(100+12\sqrt{3}\,\sqrt{23}\right)^{2/3}+50\,X+6\,X\sqrt{3}\,\sqrt{23}+2\,X\left(100+12\sqrt{3}\,\sqrt{23}\right)^{1/3}$$
$$+2\,X\left(100+12\sqrt{3}\,\sqrt{23}\right)^{2/3}-9\left(100+12\sqrt{3}\,\sqrt{23}\right)^{1/3}-\left(100+12\sqrt{3}\,\sqrt{23}\right)^{1/3}\sqrt{3}\,\sqrt{23}$$
$$\left.-18-2\sqrt{3}\,\sqrt{23}\right)\Big/\left(\left(100+12\sqrt{69}\right)^{1/3}\left(100+12\sqrt{3}\,\sqrt{23}\right)^{2/3}\right)$$

Commentaires de Conway et surinterprétation fractale.

À présent nous allons traîter un exemple arithmétique qui ne provient pas, à l'instar des matrices magiques, d'un cours d'informatique ad hoc pour montrer les performances d'un logiciel.

Il s'agit d'une suite de nombres entiers décrite pour la première fois par M. Hilgemeir en 1986 dans son article : *Die Gleichniszahlen-Reihe*. Mais son étude «exhaustive» est due à J.H. Conway et a été publiée en 1987 dans l'article : *The Weird and Wonderful Chemistery of Audioactive Decay*. Voici sa définition : 1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, 31131211131221, ... Plus explicitement nous partons du chiffre 1 et nous en faisons un commentaire en écrivant qu'il y a «un un», et ainsi de suite. Il est clair que cette suite ne comporte qu'un seul nombre à savoir «1» et que tous les autres «nombres» ne sont que des commentaires de celui-ci. Il est facile de voir qu'un commentaire (de Conway) ne contient pas le chiffre 4 (et donc aussi les suivants). Ainsi un commentaire a un nombre pair de chiffres.

Conway a montré qu'il existe, en moyenne, un rapport constant entre le nombre des chiffres du commentaire et celui des chiffres du commenté. Cette constante, obtenue par O. Atkin, est la solution d'une équation de degré 71 et sa valeur approchée est égale à 1,303577...

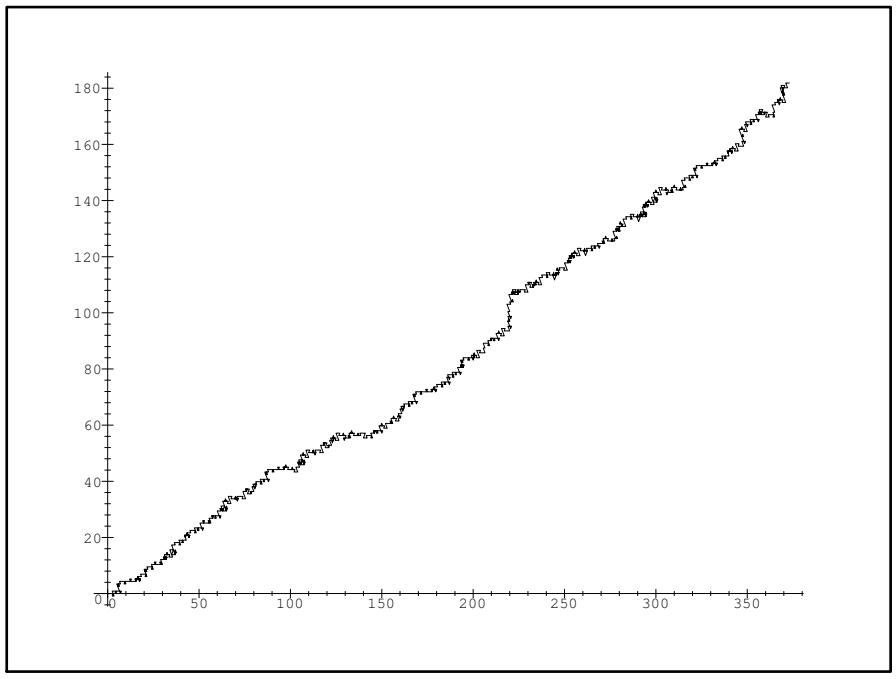
Pour étudier les commentaires de «un», Conway les a chimiquement interprétés. Plus précisément il a introduit la notion d'élément de la façon suivante. Un commentaire c pourra être coupé en deux commentaires a et b si le nième commentaire de c est la concaténation des nièmes commentaires de a et b. Du point de vue de cette théorie de la coupure, les nombres qui ne peuvent pas être décomposés sont appelés les «éléments». Le théorème chimique de Conway affirme qu'à partir d'une certaine étape la suite des commentaires de «un» ne comporte que 92 éléments (auxquels il a donnés les noms des éléments chimiques du tableau de Mendeleïev !). Quant au théorème arithmétique, il énonce les propriétés statistiques de l'évolution des éléments dans la suite de Conway. Si la suite ne commence pas par «un» alors un certain nombre d'autres éléments apparaissent mais avec une densité qui tend vers zéro. Ces éléments Conway les a surnommés, comme cela était prévisible, transuraniens.

Pour notre part nous avons décidé de surinterpréter (pour employer la terminologie de U. Eco) les commentaires de «un» du point de vue fractal. Nous sommes partis du fait qu'un commentaire n'est constitué que des chiffres 1, 2 et 3. Nous avons donc plongé le problème dans le plan complexe de la manière suivante.

Nous partons de l'origine et nous lisons un commentaire donné. Si nous lisons un 1 alors nous nous déplaçons d'une unité, si nous lisons 2 nous nous déplaçons dans la direction de j et si nous lisons 3 dans la direction de $\bar{\jmath}$.

À partir de cela il est facile d'implémenter, en Maple, un algorithme qui surinterprète de manière fractale le nième commentaire de «un». Dans ce cadre fractal, il est alors naturel d'introduire la notion de dimension fractale. Il existe de nombreuses possibilités de dimensions, mais une des plus naturelles pour une courbe est celle qui est défini comme le rapport des logarithmes de la longueur curviligne sur la longueur euclidienne.

Voici le fractal que nous avons obtenu pour le 25ème commentaire de «un».



À partir du fractal obtenu, il est facile de voir que la direction de celui-ci indique clairement que le commentaire considéré comporte beaucoup plus de «un» que de «deux», et plus de «deux» que de «trois». Cela illustre bien les résultats obtenus par Conway.

Nota Bene : Les lecteurs intéressés par les programmes ou plus généralement par des commentaires sur l'utilisation de Maple peuvent contacter les auteurs aux adresses suivantes :

lygeros ou marguin ou mizony@desargues.univ-lyon1.fr