# LE CALCUL FORMEL
## Une introduction aux logiciels MAPLE et MATHEMATICA
## (1ère partie)

**Jean-Paul ROY**

*Relativement peu diffusé dans l'enseignement français, et plutôt réservé jusqu'à présent aux étudiants avancés, le* calcul formel *commence à pénétrer le premier cycle des universités. Certains professeurs des disciplines scientifiques expérimentent même dès le lycée cette nouvelle voie pédagogique. Le but du présent article est double : d'une part, jeter quelque lumière sur le calcul formel lui-même et sa démarche. Mais surtout, encourager les professeurs de lycée à* mettre les mains dans le cambouis *et à découvrir ce qu'une longue pratique formaliste a peut-être pu leur cacher : que les objets mathématiques sont bel et bien des objets de calcul, qu'on y rencontre de la programmation, qu'on y rencontre toute la programmation.*

### Du Calcul Numérique au Calcul Formel

Beaucoup d'entre nous se souviennent sans doute des temps bénis où le sentiment d'appartenir à une race de pionniers nous faisait enfourcher un super BASIC permettant de calculer les racines d'une équation du second degré avec 6 ou 8 décimales. La plupart des scientifiques (mais aussi hélas trop souvent leurs collègues *littéraires* étaient en effet formés à la programmation par le biais des mathématiques, plus précisément de leur volant numérique. L'un des buts du langage LSE était de donner à la chaîne de caractères le statut principal, afin de *casser* cette image trop mathématisante de la programmation.

Bien entendu, les professeurs de maths entichés d'informatique étaient bien un peu gênés lorsqu'on leur reprochait de ne travailler qu'avec des approximations pas toujours fiables, sans être capable de manipuler des quantités exactes comme $\sqrt{3}$ connues non par une quelconque représentation décimale, mais par des propriétés algébriques

(avoir un carré égal à 3). Ou encore de ne pouvoir écrire simplement l'algorithme le plus mécanique qui soit : dériver une fonction.

Il fallait en effet disposer de langages sachant manier des objets relativement sophistiqués. On rejetait la faute sur BASIC, en attendant des jours meilleurs. Ceux-ci arrivèrent lorsque nous quittâmes le stade autodidacte, et que l'Université nous montra fièrement Pascal et ses *pointeurs*, qui permirent enfin de dériver une formule. Le rêve attendu semblait en passe d'être réalisé, mais les espoirs furent vite déçus. Car enfin, on pouvait bien passer une fonction en paramètre (au moins dans les *vrais* Pascal) mais on ne pouvait pas écrire une fonction dont le résultat soit *une autre* fonction. Ce qui était la moindre des choses si l'on voulait d'autres amusements que l'inversion des matrices carrées avec la méthode de Gauss...

Mais l'Université avait plus d'un tour dans son sac. Certains d'entre nous y apprirent LISP, avec lequel la dérivation devenait un exercice facile, et qui permettait d'implémenter des systèmes symboliques à l'intérieur desquels la connaissance des propriétés des objets se laissait élégamment représentée sous forme de liste, sous la houlette de systèmes plus ou moins experts. En théorie, c'était gagné. Nous tenions LE langage idéal, pour lequel le numérique relevait plus ou moins du déluge, et qui permettait enfin de modéliser nos *vraies* mathématiques.

Hélas. Malgré les étonnantes capacités de LISP à traiter sans trop d'efforts des connaissances de nature qualitative, il manquait l'essentiel : une vaste bibliothèque d'algorithmes déjà programmés. Dans les salons LISP, des murmures évoquaient pourtant l'existence mythique de monstres logiciels dont les noms comme MACSYMA ou REDUCE sonnaient comme des arguments d'autorité. Les livres regorgeaient de références sur les étonnantes capacités de ces *systèmes de calcul formel* qui permettaient de *tout* faire : des entiers à 400 chiffres, des fractions exactes, des nombres complexes, des développements limités, des tracés de surfaces, mais aussi d'inverser des matrices, de dériver des champs de vecteurs ou d'utiliser une transformation de Laplace pour intégrer une équation différentielle. *Ils* savaient faire tout cela, sans calcul approché, de manière symbolique : il ne s'agit plus de calculer avec 15 chiffres significatifs, mais de garantir un résultat exact, ni plus ni moins [1]. Par exemple, obtenir le déterminant de la matrice :

$$\begin{pmatrix} a^2 & p \\ \frac{1}{p} & a \end{pmatrix}$$

sous la forme $a^3$-1, pour ensuite factoriser ce dernier en (a - 1) ($a^2$ + a + 1), sans avoir en aucune manière besoin de connaître les valeurs de a ou de p. Les inconnues y restent inconnues ; rien n'empêche plus tard de leur attribuer des valeurs, et continuer par du *gros calcul* numérique. Bien entendu, nous nous contentions encore une fois d'un théorème d'existence. Qui parmi nous, professeurs de lycée, aurait rêvé d'un tel système chez soi ou en classe ? Et d'ailleurs ces systèmes, qui sont-ils, d'où viennent-ils et à quoi servent-ils ?

**Le Calcul Formel**

A l'origine FORTRAN (FORmula TRANslator, 1955), le premier langage kamikaze (textuellement *vent divin* en japonais) dans lequel les formules se laissaient écrire sous la forme DELTA = b**2-4*a*c au lieu d'obliger le programmeur à détailler les instructions au niveau du langage-machine. Particulièrement adapté au calcul numérique, FORTRAN était mal à l'aise avec les problèmes de nature symbolique, pour lesquels les données se représentaient par des arbres, à la manière des arbres syntaxiques utilisés pour représenter la structure grammaticale d'une phrase et son découpage en groupe nominal, groupe verbal, etc. Les expressions mathématiques aussi se représentent par de tels arbres, en suivant simplement une autre grammaire. Quelques greffes furent tentées sur FORTRAN, à la suite des balbutiements de l'Intelligence Artificielle dans les années 1956-58. C'est précisément alors qu'il travaillait chez IBM pendant l'été 58 sur le problème de la dérivation automatique d'une fonction que John McCarthy mit la dernière touche à un langage symbolique à l'opposé de FORTRAN. Il le nomma LISP (LISt Processor), la liste étant simplement un moyen de visualiser un arbre de manière linéaire. Le tour de force de McCarthy fut de représenter les fonctions elles-mêmes comme des arbres, aboutissant ainsi à un langage *auto-référent*: langage et méta-langage coïncidaient enfin (ce qui après tout est bien le cas pour les langues naturelles). FORTRAN et LISP étaient quasiment figés à leur naissance et - dotés d'emblée d'une puissance maximale pour leurs besoins respectifs - ils n'évoluèrent finalement pas beaucoup, chacun suivant sa branche propre.

Malgré la force symbolique de LISP, ce dernier avouait de grandes faiblesses en calcul numérique. Sa lenteur dans ce domaine ne le

destinait pas à être le langage de base pour les premiers *systèmes de calcul formel* qui commençaient à émerger. Ceux-ci furent donc des mélanges de FORTRAN et d'assembleur. Après tout, n'importe quel langage *raisonnable* permet de programmer n'importe quoi ! Mais la tâche était rude car les langages étaient pauvres, et les attraits de LISP en firent vite un meilleur candidat. Vers 1970, d'excellents compilateurs numériques furent disponibles pour le LISP du Massachusetts Institute of Technology, et l'un des plus vastes systèmes généraliste ayant jamais existé naquit à Boston : MACSYMA. Ecrit en LISP, son énorme bibliothèque s'enrichît d'algorithmes produit par des utilisateurs, mathématiciens et physiciens. Lorsque l'on demande aux auteurs de MACSYMA un exemple frappant de ses capacités, ils répondent par un gros calcul d'astrophysique lié aux équations de Swarzschild d'un trou noir en relativité générale !...

D'autres systèmes généraux virent le jour, largement diffusés comme REDUCE (bâti au-dessus de Standard-Lisp) ou à diffusion limitée comme SCRATCHPAD/AXIOM (IBM). Des chercheurs américains fondèrent même leur petite société (Soft Warehouse) pour commercialiser un LISP dont le but était de permettre l'écriture d'un *mini-Macsyma* fonctionnant sur les micro-processeurs Z80 puis i8086. Pari gagné, ce fut en 1979 la naissance du MU-MATH de David Stoutmeyer, qui incorporait un LISP sans parenthèses rebaptisé MU-SIMP et adapté au traitement d'objets mathématiques, ainsi qu'un grand nombre de fichiers tout prêts contenant des utilitaires de dérivation, intégration, trigonométrie, etc. Hélas, malgré un succès d'estime, le produit n'a jamais connu une utilisation massive. Peut-être à cause du LISP, peu enseigné comme *langage général* (à tort, mais c'est une autre histoire), à cause aussi de réglages manuels que l'utilisateur devait faire pour contrôler la direction des simplifications à apporter à une expression (faut-il ou non factoriser ? seulement le dénominateur ? etc.). Enfin sans doute parce que l'ergonomie n'était pas vraiment le souci majeur de l'époque. Récemment, la même société a mis sur le marché un autre excellent produit (DERIVE) entièrement piloté par menus, ayant à peu près les mêmes capacités. Mais l'absence de véritable langage de programmation en limite l'utilisation, même s'il devrait recevoir un acceuil très favorable en tant que calculette mathématique de haut niveau (il est actuellement sous licence et distribué par Nathan Logiciels).

A côté de ces systèmes généraux, il existe beaucoup de systèmes spécialisés, la plupart du temps dans le domaine public faute d'un marché suffisant. Ils sont ciblés sur l'algèbre (CAYLEY), sur la géométrie

algébrique (MACAULAY), sur la relativité (SHEEP), sur la physique des hautes énergies (FORM), etc. Ils répondent à un besoin précis et sont très efficaces, mais nécessitent la plupart du temps des connaissances pointues dans leur domaine.

Si nous allons nous intéresser à MAPLE et MATHEMATICA, c'est que parmi les *« grands »* systèmes généraux de calcul formel, ce sont à la fois les plus récents, les plus ergonomiques, et ils sont dorénavant disponibles sur nos micro-ordinateurs (compatibles-PC, Macintosh), avec des prix éducation et même des versions étudiant ! Ils sont écrits en C pour plus de rapidité et de compacité, mais simulent les mécanismes essentiels de LISP, dont ils gardent plus ou moins la marque, en restant ouverts à un style de programmation de type Pascal ou C.

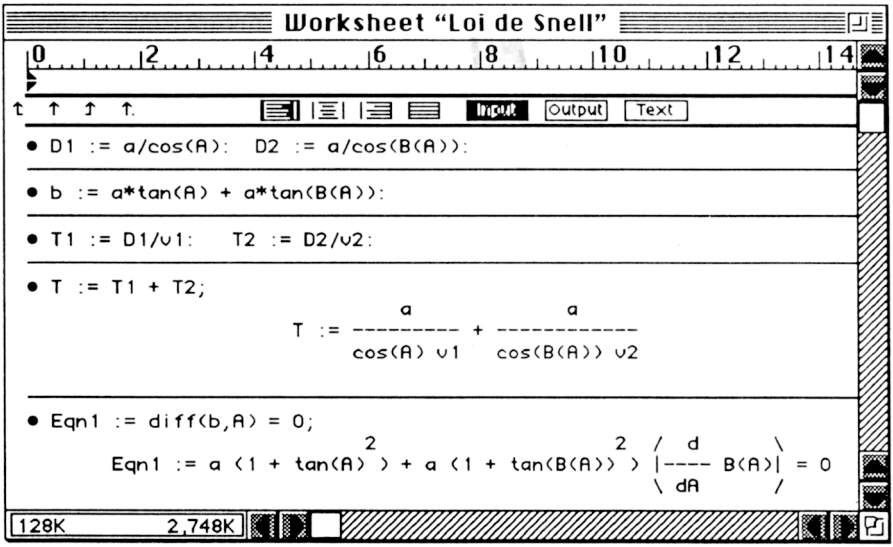
Les exemples développés dans chaque système se programment aussi dans l'autre. Au niveau du lycée ou des classes préparatoires, les différences sont minimes et les choix ergonomiques personnels peuvent faire la différence. Au niveau de la recherche mathématique très avancée, MAPLE est pour l'instant en tête.

**Le Système MATHEMATICA 2.0**

Steven Wolfram a une formation de physicien théoricien. Il commence à s'intéresser à la physique des hautes énergies, puis (notamment à Princeton) aux automates cellulaires de Von Neuman, et écrivit sur le sujet un ouvrage de référence. Puis il se tourna vers le calcul formel, et avec l'aide d'une poignée d'amis et des appuis universitaires, il prit le parti de créer - à travers sa propre société *Wolfram Research, Inc.* - un système complet de calcul formel [2] qu'il baptisa *Mathematica* (paraît-il sous l'influence de Steve Jobs, qui le distribua en série avec sa fameuse machine NeXT, aux côtés d'Objective-C et de Common-Lisp).

Très tôt, Wolfram décida que le graphisme ne serait pas un gadget, mais une composante essentielle du système, davantage tourné vers les ingénieurs que vers les mathématiciens purs (marché déjà pris par MAPLE). D'une part en proposant le standard Postscript comme sortie possible des images mathématiques (pour les imprimantes à laser, pour les transporter d'une machine à l'autre sous forme de fichiers ascii, ou pour les enrichir avec des logiciels graphiques), mais surtout en développant les *note books*, carnets électroniques permettant de mélanger du texte, des formules mathématiques, des graphiques, des animations, du son, et architecturés sous la forme de paragraphes et de

cellules que l'utilisateur peut ouvrir ou fermer à son gré. Certains livres sur Mathematica ne sont d'ailleurs que d'immenses notebooks, la version 2.0 offrant d'excellentes capacités d'édition et de mise en page de documents scientifiques, évitant ainsi d'avoir à transiter par des traitements de texte ou autres outils de PAO classique.




Hélas, le système est gourmand en mémoire, et il est illusoire de vouloir le faire fonctionner sur un ordinateur ne disposant pas d'un minimum de 8Mo de mémoire, d'un disque dur, et d'un processeur rapide (680x0, 80386). Mais une telle machine, un peu *snob* il y a deux ans, se trouve à l'heure actuelle à moins de 10 000 francs. Encore deux autres années et elles seront courantes sur les campus et dans les lycées. Ce qui explique sans doute la décision de Wolfram Research de ne *pas* porter le logiciel sur le processeur Intel 80286, la machine étant frappée d'obsolescence (sur 80386 deux versions coexistent, l'une sous DOS et l'autre sous Windows-3, les notebooks nécessitant cette dernière).

Plusieurs universités ont commencé à produire des notebooks pour illustrer leurs propres cours de mathématiques [3] ou de physique (mécanique, traitement du signal, physique quantique, relativité générale, etc). Ceci est d'autant plus intéressant que Wolfram Research distribue gratuitement MathReader, programme permettant de visualiser un notebook (par exemple des notes de cours, des exercices, des travaux d'élèves). Fantastique outil pédagogique en puissance, associé à un rétro-projecteur d'écran. On peut rêver à ce que la technique permet

réellement de faire en 1992, comment elle peut changer en profondeur la manière dont nous dispensons nos enseignements, surtout comment elle peut modifier la relation que les élèves entretiennent avec les *objets théoriques*, permettant d'attaquer l'échec scolaire sur un autre front.

Avec la version 2.0, les disciplines concernées s'élargissent : les géographes ne sont pas oubliés puisque Mathematica contient maintenant une base de données cartographique extensible (en provenance de la CIA...), et avec une mémoire suffisante (8Mo) il est possible de visualiser en couleur le globe terrestre sous différentes projections (azimuthale, de Mercator, centrée sur un pôle, etc), mais aussi des pays en particulier (ce qui s'avère assez peu précis, chaque pays étant représenté par un polygone, que l'on peut néammoins raffiner). Les chimistes pourront vérifier que la configuration électronique du Néon est bien $1s^2\ 2s^2\ 2p^6$ et voudront peut-être (Dieu les aide) *programmer* le principe d'exclusion de Pauli pour retrouver la classification périodique des Elements ! Le musicien électro-acoustique essayera diverses modulations de fréquences de la forme `Sin[a t + b Sin[c t]]` avec la fonction Play qui *joue* une fonction sur un intervalle. Un ouvrage d'économie vient de paraître, qui programme ses modèles en Mathematica [4]. Citons l'ouvrage de Crandall [5] sur les sciences en général : mathématiques, astronomie, physique, biologie, chimie, ainsi que celui de Wagon [6] axé sur les mathématiques, qui implémente le graphisme-tortue de manière très originale. Une vingtaine d'ouvrages sont en effet déjà disponibles, notamment chez l'éditeur américain Addison-Wesley. En français, un seul titre jusqu'à présent, celui de J-C. Culioli, de l'Ecole des Mines [7], qui devra être mis à jour. Un précieux petit guide de référence (en anglais) [8] a été réalisé par la firme *Variable Symbols*, qui distribue aussi le logiciel hypertexte *Mathematica Help* pour Windows-3 ou Macintosh.

Il est hors de question de *faire le tour* de *Mathematica* en quelques pages (arithmétique, algèbre, calcul différentiel et intégral, statistiques, algèbre linéaire, etc.). Voici deux courts exemples, l'un tournant autour de la récurrence pour mettre en relief l'idée de calcul exact, et l'autre pour illustrer les possibilités graphiques du système.

Soit la suite $(u_n)$ définie par une relation de récurrence de la forme $3u_n - 7u_{n-1} + 2u_{n-2} = 0$, avec les conditions initiales $u_0 = 0$ et $u_1 = 1/3$. Un certains nombre de commandes peu courantes ne sont pas chargées d'emblée dans le noyau du système, mais résident dans des *packages* sur le disque dur, lisibles car écrits dans le langage de Mathematica. Parmi ces commandes se trouve précisément `RSolve` dont la tâche est de

résoudre des équations de récurrence (celles au moins pour lesquelles on dispose de stratégies de résolution). Dans ce qui suit, `In[k]` dénote la k-ème demande de calcul de l'utilisateur, et `Out[k]` la réponse de Mathematica. Lorsqu'une demande est terminée par un point-virgule, on ne veut pas voir son résultat (seul son effet importe) :

```
In[1]  : <<RSolve.m        (* chargement du package *)
In[2]  : equation = {3u[n]-7u[n-1]+2u[n-2]==0,u[0]==0,u[1]==1/3};
In[3]  : RSolve[equation,u[n],n] (* d'inconnue la fonction n-> u[n] *)
Out[3] :                  1 2 + n
                       -9 (-)         2 + n
                          3          2
                {{u[n] -> ---------- + --------}}
                             5           20
(* Une seule solution donc, mais mal présentée. Le mot magique dans tous
ces logiciels est "simplify" ! La variable % représente le dernier
résultat *)
In[4] : sol = Simplify[%[[1]]] (* simplifions cette unique solution *)
Out[4] :                 1 n    n
                      -(-)  + 2
                         3
                {u[n] -> ----------}
                            5
(* extraction de la formule, dans laquelle on "fais" n=50 *)
In[5]  : s50 = sol[[1,2]] /. n->50    (* calcul exact *)
Out[5] : 16165625549295281212862792009130725867 5
         ------------------------------------
                  71789798769185258877024 9
(* On pourrait en demander une approximation avec virgule. Mais
intéressons-nous plutôt au nombre de chiffres du numérateur. Pour cela, on
convertit le numérateur en chaîne de caractères, puis en liste de
caractères, dont on calcule la longueur. En composant toutes ces
fonctions, cela donne : *)
In[6]  : Length[Characters[ToString[Numerator[%]]]]
Out[6] : 39
```

*Mathematica* encourage un style d'écriture fonctionnel, comme LISP. Bien qu'il laisse tous les styles ouverts y-compris celui du langage C, les emboîtements fonctionnels s'avèrent souvent plus rapides que les instructions impératives classiques. Voici par exemple deux moyens bien distincts de construire le polynome $1 + x + x^2 + ... + x^{100}$ :

```
(* le style C *)
In[7]  : p = 0;
In[8]  : For[i=0,i<101,i++,p+=x^i];
```

```
        (* résultat en 16.9 secondes sur Mac IIci *)

(* le style LISP ou APL *)
In[9]  : p = Apply[Plus,x^Range[0,100]];
        (* résultat en 0.4 seconde *)
```

Il ne faut pas généraliser, mais en retenir seulement que la notion d'efficacité d'un style ne veut rien dire en soi, et dépend souvent des choix d'implémentation du langage sous-jacent. Personnellement, le fait que le style fonctionnel soit efficace en *Mathematica* me satisfait beaucoup, car c'est lui qui est le plus proche de la pensée mathématique, qui n'a jamais historiquement eu besoin des concepts d'affectation et de boucle. Les élèves n'ont aucune obligation de s'éloigner de la mathématique usuelle pour apprendre la programmation, la composition de fonctions et la récurrence suffisant la plupart du temps.

La notion de *règle de calcul* n'est pas oubliée. Pour certains, c'est même elle qui fait la grande force du langage. Comme en mathématique (un peu comme en PROLOG mais de manière plus limitée), on peut utiliser des règles, par exemple (en supposant qu'elles n'existent pas), les règles de calcul de primitives dont l'ensemble définit la fonction `Integrale` :

```
Integrale[f_ + g_,x_] := Integrale[f,x] + Integrale[g,x]
Integrale[a_ f_,x_]   := a Integrale[f,x] /; FreeQ[a,x]
Integrale[x_^n_.,x_] :=
    x^(n+1)/(n+1) /; FreeQ[n,x] && n != -1
Integrale[Cos[a_. x_ + b_.],x_] :=
    -a Sin[a x + b] /; FreeQ[{a,b},x]
```

Les deux premières règles expriment la linéarité de l'intégration, les deux autres traitent de schémas particuliers. Notez dans certaines règles un "`/;`" qui signifie la présence de *conditions d'application* de la règle : pour intégrer `x^n`, on écarte les cas où `x` apparaîtrait dans `n`, ainsi que le cas spécial `n=-1` (logarithme). Le point figurant à côté de certaines variables est un peu plus technique, nous n'entrerons pas ici dans les détails (cf. [2] p. 239). A partir de ce *programme* de 4 règles, vous pourrez déjà demander par exemple :

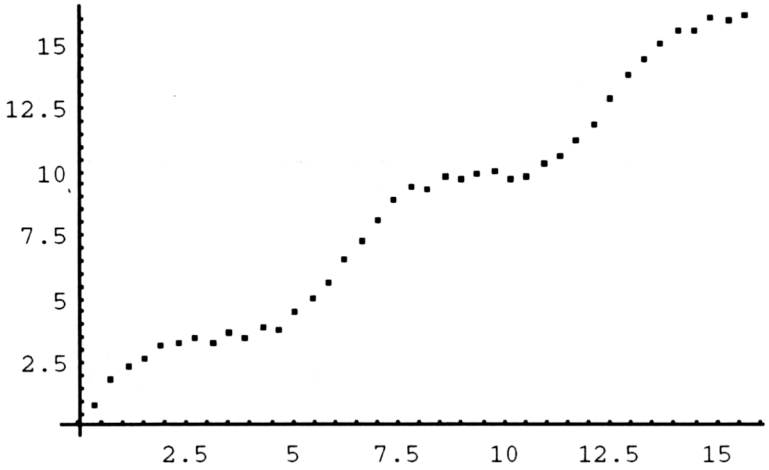
```
In[53]  : Integrale[2k/t^5 - 3/t + 4Cos[theta - t],t]
Out[53] :  -k                  1
          ------ - 3 Integrale[-, t] - 4 Sin[t - theta]
             4                 t
          2 t
```

Notez que le système avoue son échec sur la primitive de 1/t, mais ne provoque aucune erreur. L'élève, s'en apercevant, rajoutera simplement la règle manquante, sans avoir à mettre de *rustines* dans le code déjà écrit. Le découpage d'un programme en règles disjointes remplace très avantageusement les IF...THEN...ELSE... traditionnels, grâce à la capacité de **reconnaissance de formes** (pattern matching) qu'offre *Mathematica* : ce dernier trouvera lui-même la bonne règle à appliquer.

*Mathematica* est nettement moins fort que *Maple* sur le plan des algorithmes de calcul formel. Le mathématicien professionnel travaillera plutôt avec ce dernier. Mais la force de *Mathematica* tient avant tout à son très beau langage de programmation, et à ses capacités graphiques. Pour un travail élémentaire, l'enseignant (ou l'élève) peut facilement reconstruire les algorithmes vus en classe, au lieu de laisser l'ordinateur faire *bêtement* le calcul. Le fait de pouvoir *expliciter ses stratégies de calcul,* de les enseigner en quelque sorte à la machine, de les voir à l'oeuvre sur des exemples trop lourds pour les appliquer à la main, voilà une activité qui peut changer notablement l'art et la manière d'apprendre et de *faire* des mathématiques.

Pour conclure ce trop rapide tour d'horizon de *Mathematica*, voici un exemple graphique 2D où l'aspect numérique n'est pas absent. Supposons que nous ayons une série statistique de points `{x,y}` vérifiant `y = x + Sin[x] + 0.5 Random[]`, où Random retourne un réel aléatoire entre 0 et 1. Calculons une telle série avec des x variant de 0 à 5p espacés de p/8 :

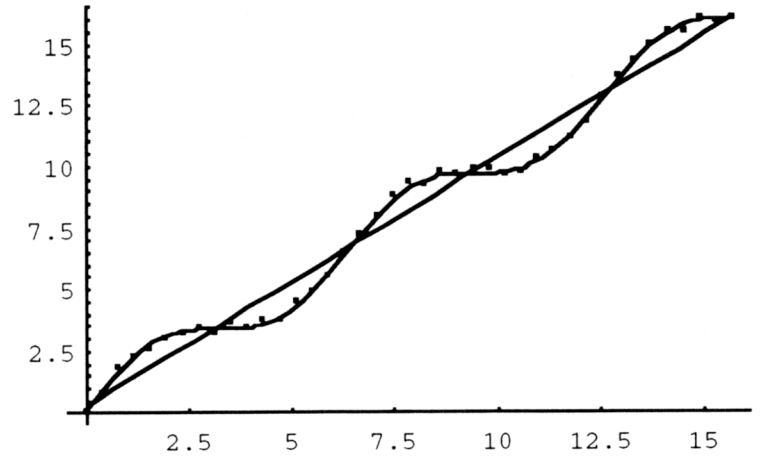
```
In[54]  : f[x_] := x + Sin[x] + 0.5 Random[];
In[55]  : points = N[Table[{x,f[x]},{x,0,5Pi,Pi/8}]];(* N=Numérique *)
In[56]  : graphPoints = ListPlot[points]
Out[56] :
```




Nous pouvons chercher une droite ou une sinusoïde oblique qui approche au mieux cette série de données (au sens des moindres carrés), en utilisant la primitive statistique `Fit` :

```
In[57]  : tendance1 := Fit[points,{1,x},x] (* dans la base {1,x} *)
Out[57] : 0.299414 + 1.0066 x
In[58]  : tendance2 := Fit[points,{1,x,Cos[x],Sin[x]},x]
Out[58] : 0.179663 + 1.00577 x - 0.0564276 Cos[x] + 1.02975 Sin[x]
```

On constate (si nous ne le savions pas à priori !) que la composante en sinus est dominante par-rapport à celle en cosinus. En superposant les trois graphiques, il vient :



- Suite de cet article dans le prochain Bulletin :
  le système MAPLE V.

Jean-Paul ROY
Département d'Informatique
Faculté des Sciences de Nice

## BIBLIOGRAPHIE

[1] : *Calcul Formel,* par J.Davenport, Y.Siret & E.Tournier. Masson, 1986 (très technique).

[2] : *Mathematica,* by S.Wolfram. Addison-Wesley, 1991 (2nd edition). En cours de traduction.

[3] : *Calculus and Mathematica,* by Brown, Porta & Uhl. Addison-Wesley, 1991 (avec disquette).

[4] : *Microeconomic Analysis,* by H. Varian, W. W. Norton and Company, Third Edition, 1992.

[5] : *Mathematica for the Sciences,* by R. Crandall. Addison-Wesley, 1991.

[6] : *Mathematica in Action,* by S. Wagon. Freeman & Co. 1991.

[7] : *introduction à Mathematica,* par J-C. Culioli, Ellipses, 1991.

[8] : *The Mathematica Quick Reference v2,* by N. Blachman. Addison-Wesley, 1992.

Ce livre, ainsi que "*Mathematica Help*" est disponible chez Ritme Informatique (cf. distributeurs ci-dessous).

**Distributeurs en France** (versions complètes, tarifs éducation).

Il existe une version étudiant quasi complète aux alentours de 1 000 F H.T.

MAPLE : SIMULOG, 1 rue James Joule - 78182 St QUENTIN YVELINES Cedex (tél : 30.12.27.00). Prix de base : 5000 FHT (IBM-PC), 3200 FHT (Macintosh).

MATHEMATICA : RITME Informatique, 34 Bld Haussmann - 75009 Paris (tél : 42.46.00.42). Prix de base : 4165 FHT (IBM-PC et Macintosh).

**Sociétés-mères** :

MAPLE : Waterloo Maple Software, 160 Columbia St. West, Waterloo, Ontario, CANADA N2L 3L3.

MATHEMATICA : Wolfram Research Inc., 100 Trade Center Drive, Champaign, IL 61820, USA.