Rapport sur un enseignement de programmation

Claude Lobry

Juillet 2008

1 Avant-propos

Ce texte n'a qu'un rapport lointain avec l'"Épreuve Pratique au bac S". Il relate une expérience d'enseignement en licence de Sciences de la Vie à l'Université de Nice. Il s'agit de l'enseignement d'un langage de programmation. J'ai le sentiment que ce type d'activité aurait mieux sa place dans le cadre d'un enseignement de mathématiques au lycée et qu'il pourrait fournir la matière de nombreux travaux pratiques. Mais je n'ai aucune expérience dans ce domaine. C'est donc aux personnes compétentes de juger s'il y a des leçons pour le lycée à tirer de cette expérience. C'est pourquoi elle est relatée ici à la demande du responsable de la commission.

2 Introduction

Il s'agit d'un enseignement de 12 fois 2 h de "travaux pratiques" devant un PC. Les étudiants sont des étudiants de deuxième année de Sciences de la Vie de l'Université de Nice, c'est-à-dire des étudiants qui ont majoritairement fait un baccalauréat S mais qui étaient "mauvais" en mathématiques. Un enseignement de première année essaye bien de leur redonner du goût pour cette discipline mais l'horaire est insuffisant pour faire des miracles. Le titre officiel de l'unité de valeur dont je vais parler est :

Modélisation Mathématique et Informatique

mais, pudiquement, les emplois du temps se contentent d'afficher "Informatique" ce qui passe beaucoup plus facilement. Un "polycopié" est distribué le premier jour. Voici quelques extraits de son introduction qui indiquent quel est l'objectif poursuivi :

> *Le but de ce cours est de familiariser avec les principes de la modélisation de la* dynamique des systèmes biologiques*. Il s'agit de transformer des hypothèses et des connaissances biologiques (par exemple ce que nous savons sur la vitesse de croissance d'une population de bactéries) en un programme informatique qui fera presque instantanément des calculs qu'il ne serait pas possible de faire à la*

main. Certes, il existe des logiciels "clef en mains" pour lesquels il suffit de faire quelques "clics" de souris pour voir apparaître toutes sortes de courbes. Mais comment ces logiciels procèdent-ils ? C'est ce que nous cherchons à comprendre. Pour comprendre un "programme informatique" il faut connaître quelques rudiments de programmation. La façon la plus simple d'y arriver est d'apprendre à faire soi-même des petits programmes. C'est ce que nous allons faire dans un langage particulièrement adapté : Le langage PASCAL. Les deux premiers cours (environ 4 TD) seront consacrés aux premiers rudiments et les applications ne seront pas très intéressantes en tant que telles : nous ferons une petite calculatrice et un programme qui trace des graphes de fonctions, toutes choses que votre calculatrice du lycée faisait fort bien. À partir du troisième cours nous pourrons aborder des modèles simples de dynamique des populations. Nous étudierons :

- *La croissance exponentielle.*
- *La croissance logistique.*
- *Les modèles "proie-prédateur".*
- *Les modèles de compétition.*
- *Des modèles stochastiques.*
- *Des modèles spatialisés.*

Ce cours s'adresse à un public très hétérogène. Parmi vous il y a :

- *Ceux qui savent à peine ce qu'est un ordinateur et qui n'ont pas l'habitude de travailler avec un éditeur de texte, je pense très rares.*
- *Ceux qui sont familiers avec le système Windows et qui ont déjà travaillé avec un éditeur de texte (par exemple pour faire un rapport avec le traitement de texte Word), les plus nombreux, je pense.*
- *Quelques-uns qui connaissent déjà un langage de programmation.*

Pour les premiers, les deux ou trois premières séances de TD seront assez difficiles. Il faut un peu de temps pour s'habituer à manipuler rapidement quelques commandes. Il ne faut surtout pas se décourager. Cela vient finalement assez vite. Pour les seconds les choses devraient se passer sans problème. Enfin, ceux qui connaissent déjà un langage de programmation iront très vite. Tant mieux pour eux. (...)

Naturellement il y a loin entre l'ambition affichée dans ce polycopié et les résultats dont je vais faire le compte rendu dans ce rapport. Mais avant je dois m'expliquer un peu sur la philosophie de cet enseignement.

3 Réponse à quelques interrogations

Pourquoi enseigner la programmation en Sciences de la Vie ?

Les modèles mathématiques simulés sur ordinateur sont de plus en plus utilisés, par exemple, pour comprendre les dynamiques complexes de l'expression du génome, de la physiologie des cellules, de l'interaction des populations...

Il existe toutes sortes de logiciels "clef en main" qui permettent, en apparence, de se débarrasser de l'investissement mathématique nécessaire pour construire les équations qui représentent le phénomène considéré. De façon plus ou moins évidente les représentations informelles du biologiste sont traduites en un programme informatique immédiatement exécutable, programme qui "produit" des "courbes" plus ou moins esthétiques, mais toujours difficiles à interpréter dans la mesure où l'on *ne sait pas exactement ce que fait le logiciel*. Ce côté "magique" me semble *contraire à l'esprit scientifique*. De la même manière qu'il n'est pas possible d'interpréter des images de microscopie électronique sans avoir la moindre idée de la façon dont fonctionne un microscope électronique, je pense qu'il faut avoir une idée de ce que fait un programme informatique de simulation pour interpréter ses résultats.

En résumé il ne s'agit pas de devenir expert en programmation mais simplement de comprendre "comment ça marche".

Pourquoi enseigner le langage PASCAL ?

Avouons-le, je suis assez ignorant en programmation et PASCAL est le seul langage que je maîtrise. C'est celui que j'utilise dans ma pratique de chercheur – je simule beaucoup de systèmes d'équations différentielles ordinaires – et je vois bien que mes jeunes collègues, qui ne jurent que par Matlab ou Scilab, me considèrent comme un vieil attardé. Je ne conteste pas qu'ils ont certainement raison et que pour un usage professionnel PASCAL n'est peut-être pas une bonne idée.

Toutefois, pour avoir essayé des langages plus modernes, il m'a semblé qu'ils n'ont pas les vertus pédagogiques de PASCAL, sur lesquelles je reviendrai et que je résume en : *PASCAL est un langage qui est un prolongement naturel simple du langage mathématique et qui éclaircit ce dernier.*

La version libre "Free Pascal" est facilement récupérable et fonctionne sans problème. Sa convivialié n'est pas extraordinaire, mais suffisante.

Cela dit je n'exclus pas que d'autres langages aient les même vertus.

Pourquoi un enseignement exclusivement en travaux pratiques ?

On aurait pu imaginer un mélange d'enseignement théorique donnant quelques principes de base de programmation, accompagné de quelques séances de TD et/ou TP. J'ai préféré du "tout TP". La raison en est la suivante : je ne sais plus faire des cours magistraux ! Je me souviens qu'il y a une vingtaine d'années je faisais des cours devant des amphithéâtres de Sciences de la Vie dans un silence absolu avec des étudiants attentifs et qui arrivaient à l'heure. Je ne sais plus obtenir ce résultat et, soit je suis "cool" et mes cours ont lieu dans une ambiance sonore qui rend toute concentration impossible, soit j'obtiens le silence à force d'exclusions répétées mais alors il ne reste qu'un effectif d'étudiants réduit à environ un dixième de la promotion. Dans les deux cas le résultat est catastrophique !

4 Les modalités de l'enseignement

Les objectifs

Je suis parti du principe que les étudiants n'étaient, pour la plupart, pas bons en mathématiques. Je leur supposais simplement la compréhension de ce qu'est une fonction et sa représentation graphique, sinon au sens où l'entendent les mathématiciens, au moins au sens pratique et opérationnel où l'entendent les biologistes lorsqu'ils tracent des courbes d'évolution de quelque chose. Concernant l'usage de l'ordinateur je leur supposais une capacité raisonnable à se débrouiller à l'intérieur de Windows.

Je souhaite qu'à l'issue de l'enseignement tous maîtrisent :

- Le concept d'affectation d'un nombre ou d'un symbole dans une mémoire, le " :=" de PASCAL.
- La structure d'un programme PASCAL avec les "procédures".
- La boucle "Repeat" (ou For i := 1 to n).
- Le test "If $\ll$ $\gg$ then ".
- La technique d'allumage d'un pixel de l'écran.

et soient capable d'écrire un programme qui réalise le tracé du graphe d'une fonction donnée, assez convivial pour que l'utilisateur puisse placer ses axes dans l'écran, afficher des unités, déterminer son domaine de définition.

Pour les meilleurs je souhaite une bonne maîtrise du schéma d'Euler d'intégration des systèmes différentiels du plan, la compréhension de phénomènes comme l'effet stroboscopique voire des artefacts liés à la "pixelisation".

Je souhaite également réconcilier les étudiants avec les mathématiques en leur dévoilant tout doucement que ce qu'ils croient être de l'informatique est en fait une utilisation intelligente de mathématiques du Lycée.

Le contrôle des connaissances

Pour motiver les étudiants j'utilise un procédé que certain jugeront peut-être démagogique. Ils ont la certitude, *a priori*, que s'ils font bien ce que je leur demande ils auront au moins la moyenne. La procédure est la suivante :

- Chacun doit réaliser trois programmes de difficulté croissante et me les présenter. Chaque programme reçoit une note sur vingt. Donc trois notes E_1, E_2 et E_3.
- Chaque étudiant est libre de perfectionner son programme et de me le représenter jusqu'à obtenir 20/20. Le travail collectif est encouragé. Pour être certain que l'étudiant ne se contentera pas de présenter le programme d'un collègue plus efficace il sait que je glisserai quelques "bogues" dans son source et qu'il devra corriger.
- Un contrôle final individuel avec des questions théoriques et pratiques dont la note E_f est sur soixante.
- La note finale est $\frac{E_1+E_2+E_3+E_f}{6}$.

Ajoutons que dès la troisième séance pratiquement tous les étudiants ont obtenu la note de 20 pour leur premier programme. Tous sont motivés pour obtenir 20 au deuxième programme, ce qu'ils obtiennent, les meilleurs ont 20 au troisième TP avant la fin des séances, les plus mauvais ont 10. Il ne reste aux plus mauvais qu'à obtenir 10/60 au contrôle final. Ce qu'ils obtiennent.

Le polycopié

Ce point est essentiel pour suivre ma démarche. Le polycopié que je distribue n'est pas un **cours**. Je veux dire par là qu'il n'a pas les qualités (précision, clarté, exhautivité...) que l'on attend d'un ouvrage qui, en principe, doit se suffire à lui-même et dont la rédaction demande beaucoup de soin. Ce n'est pas le cas de ce polycopié qui est conçu comme la description d'une succession de manœuvres à effectuer, manœuvres devant conduire à un certain résultat visible sur l'ordinateur. C'est un mélange de considérations théoriques et pratiques où chaque nouveau concept est testé le plus vite possible. Tout n'est pas nécessairement très clair dans le polycopié **mais je sais que je serai présent** pour répondre individuellement ou collectivement. En voici un extrait, au début.

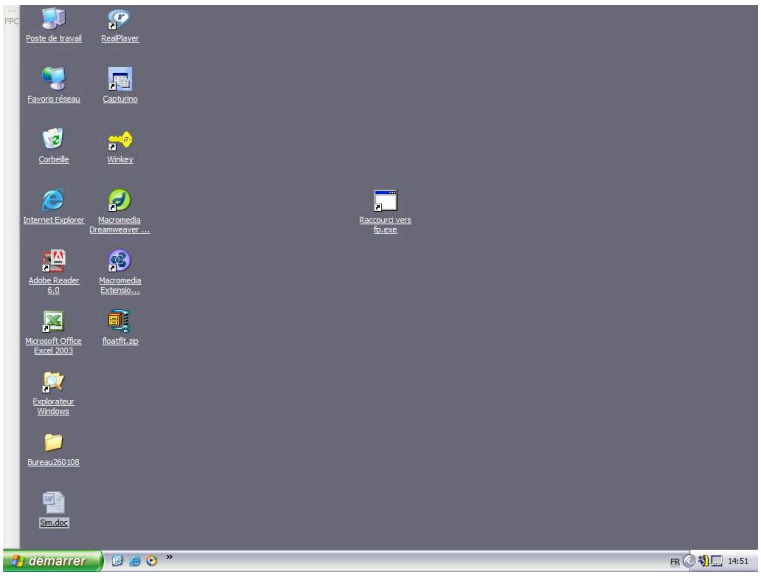

FIG. 1 – L'icône de "Raccourci vers fp.exe"

Cours n° 1. *Comment ça marche*

Le tout premier programme

Le tout premier programme que vous allez étudier (et réaliser) est écrit ci-dessous. Jetez juste un coup d'œil, ne cherchez pas à comprendre tout, tout de suite :

- **program Calculatrice ;**
- **var**
- **x, y, z : real ;**
- **begin**
- **write ('Entrez votre premier nombre x = ') ;**
- **readln(x) ;**
- **write ('Entrez votre premier nombre y = ') ;**
- **readln(y) ;**
- **z := x*y ;**
- **write('le produit de x par y est : ') ;**
- **write(z :6 :6) ;**
- **readln ;**
- **end.**

Vous le voyez, ce texte n'est pas très parlant ! Toutefois on peut faire tout de suite quelques constatations : il est composé de 13 lignes qui sont presque toutes suivies d'un "point-virgule". Ces lignes semblent être des *instructions* ; par exemple "write(z :6 :6)" qui pourrait vouloir dire "Écrivez ceci", mais quoi ? z : 6 : 6 ? La première chose à bien comprendre est que le "programme" est un texte qui ne s'adresse pas à vous mais à l'ordinateur. Nous allons apprendre comment l'ordinateur lit ce texte.

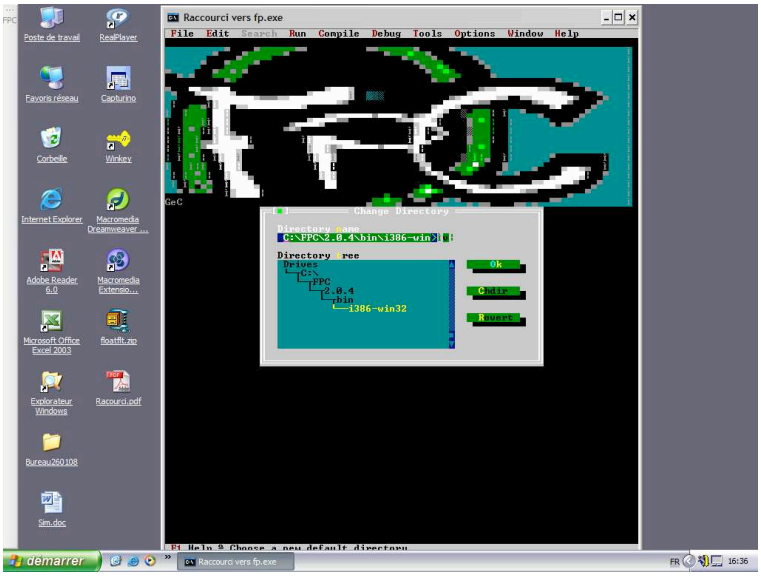

FIG. 2 – L'écran de l'éditeur FCP quand on a cliqué sur "Change Dir" du menu "File"

Édition du texte

Pour commencer il faut savoir qu'un ordinateur, en lui-même, n'est pas capable de lire ce texte. Ce texte ne peut être lu et compris que par un programme déjà installé sur l'ordinateur qui dans notre cas s'appelle Free Pascal Compilator (FPC) et dont l'icône qui permet de l'utiliser s'appelle **"Raccourci vers fp.exe"** et se trouve, en principe [1], sur le bureau. C'est l'icône que j'ai détachée au centre de la figure 1. Si vous cliquez sur cette icône vous voyez apparaître un "bazar" très compliqué qu'il ne faut surtout pas chercher à comprendre tout de suite. C'est *l'éditeur de texte* de FPC ; c'est un système de commandes qui permet de travailler sur le programme et que nous devons apprendre à utiliser. Comme dans la plupart des logiciels il y a des "menus" qui affichent diverses fonctions.

Allez dans **File**, cherchez **Change dir** [2]

et, si tout va bien [3], vous verrez apparaître (figure 2) l'arbre du dossier courant : (voir "Help").

Double-Cliquer sur "FPC" ; Double-Cliquer sur : PASCAL-08 ; OK ; retourner à **File** ; Cliquer sur **0pen** ; Ouvrir "Calculatrice.p".

et vous voyez apparaître (figure 3) exactement le même programme que celui que vous venez de lire sur cette feuille mais sous une forme plus agréable à regarder. Sur la figure 4 vous avez un agrandissement plus lisible mais il est préférable que vous regardiez

[1] Si ce n'est pas le cas se reporter à la feuille "Help".
[2] Si la souris n'agit pas sur la fenêtre consultez "Help".
[3] Si ce n'est pas le cas consulter "Help".

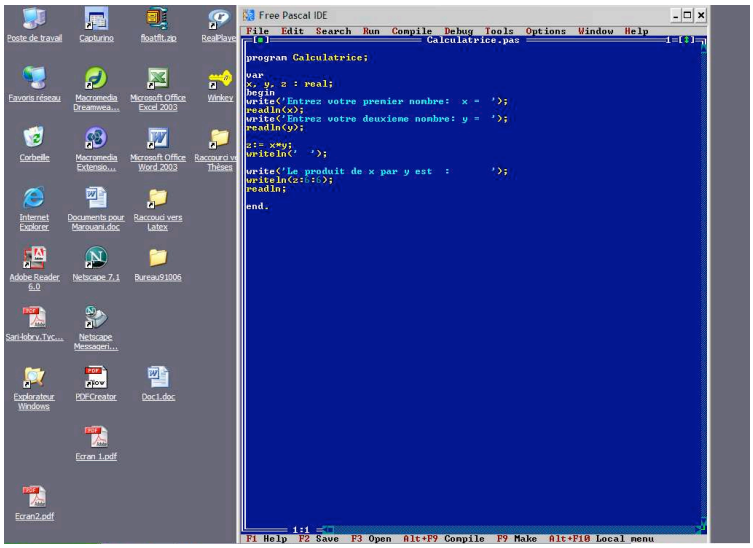

FIG. 3 – Le programme tel qu'il apparaît dans l'éditeur

sur l'écran en suivant sur la feuille. Cette fenêtre s'appelle la **fenêtre d'édition**. Maintenant vous pouvez avec la souris déplacer le curseur et taper du texte [4], modifier du texte, etc. (Vous avez les traditionnels **copier-coller** dans le menu **edit**). Apprenez à modifier le texte puis remettez-le sous sa forme initiale.

ATTENTION SI VOUS NE REVENEZ PAS EXACTEMENT AU TEXTE INITIAL LA SUITE NE MARCHERA PAS !

Exécution du texte

Comme on l'a deviné le programme est une suite d'instructions à exécuter et c'est le logiciel FPC dans lequel nous nous trouvons qui va le faire. Il suffit de le lui demander ; pour cela :

Aller dans le menu **Run**.
Faire **Run** dans le menu.

La fenêtre d'édition disparaît et apparaît une nouvelle fenêtre que nous appellerons "fenêtre de dialogue". On y lit des choses pas très claires puis, à la dernière ligne :

Entrez votre premier nombre x = ?

[4] Les flèches "droite", "gauche", "monter", "descendre" sont plus pratiques que la souris pour bien positionner le curseur.

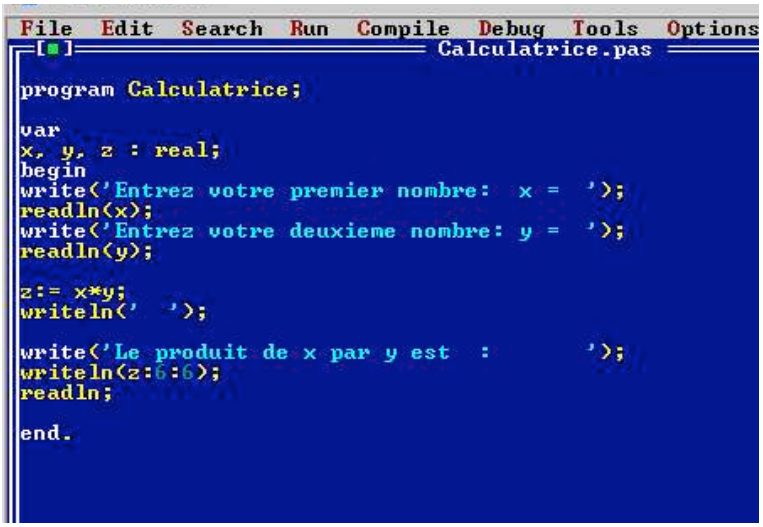

FIG. 4 – Le programme Calculatrice

Cette fois, il s'agit manifestement d'une instruction qui s'adresse à vous qui êtes devant l'écran. Obéissez et rentrez, par exemple, le nombre 4 (n'oubliez pas d'appuyer sur la touche "enter", mais maintenant tout le monde sait ça !). Vous voyez apparaître une deuxième demande : entrer un second nombre. Vous le faites (2 par exemple) et vous voyez apparaître le produit de x par y. Quand vous en avez assez vous faites "enter" et tout se ferme pour vous retrouver dans la fenêtre d'édition. De ce qui précède il faut retenir le point important suivant. Quand vous êtes devant l'écran de l'ordinateur deux situations peuvent se produire :

- **Fenêtre éditeur de FPC** : Fenêtre à fond bleu. Vous êtes en situation de *programmeur* et vous pouvez modifier le programme que FPC est chagé d'exécuter.
- **Fenêtre de dialogue** : Fenêtre à fond noir. Vous êtes en situation d'*utilisateur* du programme et vous obéissez aux instructions que vous donne la machine.

Le programme "Calculatrice" est donc un programme qui permet à un utilisateur de faire de façon agréable et conviviale le produit de deux nombres.

L'usage du polycopié

Ces quelques premières pages sont donc destinées à faire comprendre ce qu'est le "source" qui se lit dans l'éditeur de texte du compilateur FPC dans une fenêtre dite d'édition où *l'étudiant est en posture de "programmeur"* alors que le programme qui s'exécute ouvre une fenêtre de dialogue où *l'étudiant se trouve en position d'utilisateur* du logiciel créé par le "programmeur".

L'idée est que tout ceci doit se passer lors de la première séance pour que tout de suite on puisse voir un programme s'exécuter. Dans la pratique ça marche plus ou moins bien, surtout pour ceux qui ont des difficultés avec la manipulation des fichiers à l'intérieur de l'éditeur.

J'observe les étudiants et je constate qu'ils rencontrent une difficulté liée a une erreur pédagogique grossière (voire une erreur pure et simple dans le polycopié) ou à une difficulté réelle. Je laisse les étudiants chercher, communiquer entre eux, "se prendre la tête". Après une demi-heure de cafouillage je demande le silence. Je fais alors un "cours" de dix à quinze minutes, dans un silence religieux, où j'explique avec les mains : édition, fenêtre de dialogue, double casquette "programmeur"/"utilisateur", etc. Tout le monde comprend, sauf deux ou trois irréductibles à qui il me reste tout le temps pour faire une leçon particulière.

Je n'insiste pas plus. Mon but n'est pas ici d'analyser le détail de ce cours, ses astuces pédagogiques, etc., mais simplement d'expliquer comment le polycopié intervient comme document pour susciter le dialogue, dialogue entre les étudiants, dialogue entre les étudiants et le prof.

L'expérience m'a montré que les imperfections du polycopié ne sont pas une gêne. Au contraire, elles justifient la présence du prof, facilitent le dialogue avec les étudiants. En revanche il faut que le contenu du polycopié soit bien présent à l'esprit du prof pour qu'il puisse répondre très vite à l'étudiant. J'en ai fait l'expérience la deuxième année où j'ai repris le polycopié tel quel. Je ne me souvenais plus très bien de mes intentions, je ne pouvais plus répondre aussi spontanément. Bref, c'était moins bien.

C'est pourquoi il me semble indispensable, pour que ça marche bien, que le prof prépare son propre polycopié. Cet effort est largement compensé par une présence très reposante devant les étudiants.

5 PASCAL et les mathématiques

Je pense que beaucoup de notions de PASCAL sont des notions qui aident à une meilleure compréhension des mathématiques. Je donne quelques exemples.

Affectation des variables et fonctions

Les mémoires qui contiennent des variables doivent être déclarées de type "entier", "réel" (avec 8 décimales), "double" (double précision), "string" (chaîne de caractère).

Si la mémoire **i** est déclarée "entier", la mémoire **x** est déclarée "réel" et si quelque part dans le programme on a :

$$\mathbf{i := sin(x)}$$

il y a peu de chance pour que le programme marche parce que la fonction **sin** ne renvoit que très rarement un entier.

Les procédures “fonction” :

function f (x :real) :real ;
begin ;
f := $x^3 - 3 * x + 7$;
end ;

explicitent bien un domaine de définition (la variable est réelle), les valeurs prises, et on encadre d'un **begin-end** l'algorithme de production de **f**. [5]

Les sommes

La réalisation de :

$$\sum_{i=1}^{4000} \frac{1}{1+i^2}$$

sous la forme :
For i := 1 to 4000 do
begin ;
u : $= \frac{1}{1+i^2}$;
S := S+u ;
end ;
facilite l'accès au symbole $\sum$.

Les représentations dans des axes cartésiens

- Un pixel de l'écran se repère par sa colonne (un entier en commençant par la gauche, comme en maths) et une ligne (un entier en commençant par le haut, pas comme en maths !).
- Tracer des axes, c'est tracer un “repère-axes” dans le “repère-écran”. Afficher à l'écran un point de coordonnées (x, y) réelles dans des axes particuliers est un exercice qui oblige à fabriquer une procédure

  Point(x,y)

  assez complexe qu'il faut maîtriser sous peine de voir ses axes à l'envers et le graphe de sa fonction symétrique de celui que l'on cherche à tracer...

[5] Ici on se souviendra des excès du “bourbakisme” qui voulait, qu'en mathématiques, tout soit “ensemble”. Ainsi une fonction était une “partie du produit cartésien” de X (espace de la variable) par Y (espace de définition des valeurs de la fonction).

Les graphes

Tracer le graphe ce sera écrire une procédure du genre :
x := Xmin ;
repeat
y := f(x) ;
Point (x,u) ;
x := x+h ;
until x > Xmax ;
dans laquelle le pas de discrétisation (on dit aussi d'échantillonnage) **h** va jouer un rôle important.

- S'il est trop grand la courbe est un pointillé.
- Les points sont plus rapprochés quand la pente est faible.
- Avec la dérivée on peut prédire les valeurs de **h** à partir desquelles le trait est continu.

Plus intéressant, avec une fonction du genre **Sin(1000*x)** et un **h** petit mais pas trop on peut faire apparaître le graphe continu de la fonction **Sin(3*x)**. On peut même prévoir ce qui va se passer en fonction de **h** (la stroboscopie).

L'exponentielle

- La croissance exponentielle est très rapide. Pourquoi se fait-on "jeter" de l'ordinateur quand on programme :
  For i := 1 to 10000 do
  begin ;
  u : = 2*u
  end ; ?
- On visualise sur l'écran la vitesse de convergence des approximations :
  x :=0 ;
  y := 1 ;
  repeat
  Point(x,y) ;
  x := x+h ;
  y := y+h*k*y ;
  until x > 10 ;
  vers le graphe de la fonction $x \to e^{k\,t}$.

On pourrait multiplier les exemples. Il y a certainement beaucoup à dire sur les mathématiques, la pédagogie des mathématiques et la compréhension de l'informatique. Un sujet d'actualité !

6 Conclusion

Je craignais des problèmes liées à la difficulté de naviguer dans Windows (que je connais mal) et dans l'éditeur de Free-PASCAL. Craintes infondées. Ces jeunes sont nés en même temps que les PC. Ils maîtrisent très bien une technique d'essai-erreur qui leur permet de résoudre les problèmes, sans forcément comprendre ce qui se passe, mais certainement de façon plus efficace que moi avec mon esprit cartésien. Ils peuvent donc surmonter de nombreux petits problèmes liés au système d'exploitation bien plus rapidement que moi. C'est valorisant pour eux et il s'établit entre nous un rapport de confiance : nous sommes là pour nous aider à comprendre. Naturellement, à l'intérieur d'un programme, quand il ne suffit plus de "faire marcher", quand il faut comprendre pour pouvoir créer à son tour, je reprends la main.

C'est formateur car ils se rendent compte que la procédure essai-erreur est efficace *jusqu'à un certain point* où il faut se résoudre à *théoriser*, à se servir de son intelligence. C'est un début de prise de conscience de ce qu'est une théorisation.

Quand à l'impact de cet enseignement sur la compréhension des mathématiques je ne veux pas conclure mais je peux livrer une anecdote dont je jure la véracité. À la fin de l'enseignement, à une étudiante qui réussit très bien je demande :

« Au lycée vous étiez bonne en mathématiques ?
– Pas du tout, j'étais nulle !
– Pourtant ce que vous faites là ce sont des mathématiques et vous les faites fort bien...
– Ah non, m'sieur, ça c'est pas des mathématiques ! Ça c'est logique. »

À chacun d'en tirer la morale !