informatique

remarques sur l'enseignement des mathématiques "appliquées" et les possibilités d'utilisation de machines "de poche" en BASIC

par Liviu Solomon,
Université de Poitiers

Le développement des moyens de calcul et leur utilisation dans l'industrie, les transports, les "services", etc, rend inévitable un déplacement des centres d'intérêt dans la recherche et l'enseignement des mathématiques. Il me semble que l'enseignement *théorique* doit tenir compte des possibilités, des nécessités, des problèmes posés par le calcul "réel", **qui se fait avec des machines.** Les établissements qui possèdent le *matériel* nécessaire deviennent de plus en plus nombreux, et l'enseignement des mathématiques dans un lycée disposant d'une dizaine de GOUPIL ou LOGABAX ne peut plus être le même que si de telles machines n'existaient pas.

Il serait impossible et inutile de créer dans chaque établissement, de toutes pièces, le "logiciel" adéquat. Mais il me semble que, suivant les niveaux, il est indispensable

a) de donner (dès le lycée !) des idées sur le calcul automatique, et sur (au moins) un langage de programmation ;

b) de ne pas utiliser les machines "en boîte noire" ;

c) de ne pas créer un état de *dépendance* par rapport aux machines, mais de rendre l'utilisateur (élève, étudiant — c'est-à-dire l'ingénieur ou le chirurgien de 2000-2010 !) capable de *maîtriser* la machine, de *s'en servir* de façon naturelle et pour des objectifs majeurs.

Si l'on voulait assurer, pour chaque élève ou étudiant, la possession et l'exploitation courante d'une machine rapide, avec beaucoup de mémoire, graphisme, etc, cela serait une dépense considérable pour un matériel utilisé pendant un temps très court, et pour une fraction infime de ses possibilités. La réalité est que 90 % des *problèmes calculatoires* qui peuvent se présenter au lycée, en classes préparatoires, à l'université, etc, peuvent, et *méritent* d'être illustrés sur "calculette" en langage BASIC. Je veux parler ici de machines qui ont une "mémoire morte" (ROM) de 24 à 64 K, et une mémoire vive (RAM) allant de 2 K à 18 K. Les machines d'aujourd'hui disposent d'un BASIC de bonne qualité, d'une vitesse raisonnable, d'une mémoire permettant d'aborder des problèmes non triviaux. L'élève ou l'étudiant peut utiliser la machine en Travaux dirigés ou en Travaux pratiques, et peut l'emporter *chez lui* ; il peut *essayer*, il peut *réfléchir* en se servant de la machine ; la "calculette" le rend *indépendant* de l'horaire de travail des techniciens du Centre de calcul de son établissement. Et s'il a une idée intéressante un samedi après-midi, il ne devra pas attendre la matinée du lundi pour l'essayer sur le terminal qui se trouve à l'école... Les machines individuelles, compactes, peu chères, sont *un outil de pensée et d'indépendance intellectuelle*. (Pour ceux qui attendent tout des ordinateurs sur les campus, je voudrais rappeler que l'abondance en moyens *matériels* de calcul n'existe que dans très peu de pays. Dans la moitié de l'Europe, une TI 57 se vend aujourd'hui encore à des prix dépassant le traitement mensuel d'un ingénieur, et une calculette en BASIC est inaccessible à l'universitaire moyen. Il ne me semble pas juste que l'on utilise sans parcimonie des moyens de calcul dont l'immense majorité de l'humanité ne peut même pas rêver.)

1. Quelques mots sur les très petites machines

On peut bien sûr penser à des machines qui se branchent sur la télévision ou sur un moniteur vidéo : SINCLAIR, THOMSON TO-7 ou MO-5, ATARI, APPLE, ou PC d'IBM (prix de ce dernier, suivant les configurations : de 28 000 à 48 000 francs...). Mais nous nous limitons ici à des machines de poche, pesant de 115 à 375 g, et ne disposant que d'un affichage spartiate, mais autonome. (Le lecteur qui a accès à un "vrai" ordinateur peut passer aux exemples de la fin. Mais ses élèves seraient peut-être intéressés justement par les petites machines...).

Parmi les constructeurs américains, il faut parler de HEWLETT-PACKARD. (TEXAS INSTRUMENTS n'offre actuellement que des

modèles en langage-machine.) Parmi les constructeurs japonais, nous retiendrons SHARP et CASIO (la X-07 de CANON est excellente, mais un peu lourde, et chère.)

Pour évaluer une machine, on doit connaître la taille de sa mémoire, la précision et la variété dans la représentation des nombres, la richesse en fonctions mathématiques, la rapidité de calcul, les possibilités (et le prix...) des périphériques, etc.

De façon globale, on peut dire que les CASIO et les SHARP sont remarquables, et sensiblement moins chères que les HEWLETT-PACKARD. Mais le rapport qualité/prix est en faveur des machines américaines.

Le *matériel bibliographique* fourni avec les SHARP et CASIO est discutable. Certains détails des programmes peuvent être exploités, mais en général ces programmes nous paraissent médiocres — ce qui est surprenant si on les compare aux possibilités du matériel. Tous ces livrets sont la preuve d'un protectionnisme outrancier : traduits en français *au Japon* (et visiblement sans participation d'un rédacteur de langue française...), *imprimés* au Japon, leur lecture est un exercice qui n'est ni facile, ni agréable...

Les livrets HP sont de bonne qualité, traduits avec soin en français ; mais les brochures "Mathématiques I-II-III" nous paraissent parfois éloignées de l'état des mathématiques d'aujourd'hui, et des performances des machines.

a) Nous allons examiner ensemble les CASIO et les SHARP, en décrivant d'abord ce que nous trouvons en commun sur les deux marques.

Les différentes variantes ont des unités de calcul travaillant sur des nombres avec des *exposants* entre -99 et $+99$, et des *mantisses* avec 12 chiffres significatifs. Mais les CASIO "mettent en mémoire" les 12 chiffres, tandis que les SHARP retiennent seulement 10 chiffres.

La richesse du BASIC correspond à une "mémoire morte" de *minimum* 24 K. (A titre de comparaison : le BASIC du MO-5 a une ROM de 16 K.) Pour ce qui est de la mémoire vive, les machines vont de 2 K jusqu'à environ 18 K, suivant le modèle, l'extension, etc.

La représentation des nombres est le plus souvent limitée au mode "réel", mais sur certaines machines on dispose d'une "demi-précision" (5 chiffres significatifs au lieu de 10 ou 12, et moins de mémoire dépensée).

Toutes ces machines acceptent un nombre raisonnable de "niveaux" de parenthèses et de dépendance fonctionnelle. Elles possèdent les instructions fondamentales du BASIC.

Tous les modèles permettent l'écriture de lignes *multi-instructions*. Le schéma "TANT QUE (R) FAIRE" est présent sous la forme de

l'instruction de boucle FOR-TO-STEP-NEXT, ou encore peut être réalisé à l'aide de IF-THEN-ELSE.

Cette dernière instruction est parfois présente seulement sous la forme appauvrie IF-THEN.

Notons que sur les SHARP, l'instruction de boucle FOR-NEXT travaille seulement avec des valeurs *entières* du pas (ceci peut compliquer certains programmes : par exemple, dans le calcul d'intégrales). Après IF-THEN, les SHARP n'acceptent que le renvoi à une adresse, ou une instruction d'affectation ; les CASIO sont plus souples, permettant d'inclure de petites "procédures", par exemple, une boucle FOR-NEXT.

Toutes les machines disposent des instructions GOSUB et RETURN, qu'il convient d'appeler "renvois avec retour" ou "sous-routines". (De "vrais" sous-programmes existent seulement sur les machines HP, CANON, etc. ; cf. plus loin.)

On ne peut pas définir ici de *fonctions* (à l'aide de DEF FN et FN END), ce qui nous fait recourir à des sous-routines : c'est une confusion des genres, préjudiciable à la *logique* de la programmation. (Un sous-programme exécute une certaine *tâche* à l'intérieur d'un, ou de plusieurs programmes appelants ; une fonction calcule une *valeur*.)

L'éventail des *fonctions mathématiques* est raisonnablement large. Certaines machines (comme la FX-750 de CASIO) sont riches en fonctions *statistiques* ; d'autres (comme la PC-140I de SHARP) ont de nombreuses "touches calculatrice" ; d'autres (comme la PB-700 de CASIO) ont un écran de 4 lignes avec des possibilités de graphismes ; etc.

La *rapidité de calcul* semble satisfaisante. La PB-100 de CASIO (aujourd'hui dépassée) exécutait la boucle vide FOR-NEXT 1000 fois en 7 secondes. (Le Personal Computer d'IBM, malgré l'encombrement, le prix, la réclame, est exactement *cinq fois* plus rapide, pas plus !) La PC-1245 de SHARP est relativement lente, la PC-1350 est rapide.

Il est très important d'avoir la possibilité de *créer des variables à indices* : les instructions correspondantes (DIM, READ, DATA, RESTORE) existent même sur la moins chère des SHARP (la PC-1245), sur PB-700, FX-750 de CASIO, etc. Ceci permet d'utiliser des notations "naturelles" (par exemple : B(I,J) pour le terme matriciel b_{ij}). Mais aucune de ces machines n'offre de programmes incorporés de *traitement de matrices* (résolution de systèmes, inversion de matrices, ou simplement la somme MAT A = B + C, etc.).

Si les SHARP ont le mérite d'avoir *toutes* les instructions DIM, DATA, etc., elles ont *toutes* le défaut de *détruire* les variables à indices lors de l'exécution de RUN ! (Il faut faire GOTO, ce qui est incorrect du point de vue du BASIC.)

Sur toutes les SHARP on dispose des opérateurs booléens (AND, OR, NOT).

Toutes ces machines ont les 26 mémoires "fixes" A, B, ..., Z, et la possibilité d'utiliser la mémoire "vive" en partie pour des variables, en partie pour des lignes-programme. Sur les SHARP, la mémoire-programme est "d'un seul tenant" ; on peut avoir plusieurs programmes en mémoire, mais ils doivent être bien délimités (si un programme se termine en 160, il doit être terminé par END, et un autre peut commencer, disons, en 200). Sur les CASIO, la mémoire-programme peut être partagée en 10 "zones" indépendantes ; on peut avoir une ligne 30 dans la zone 0, une ligne 30 dans la zone 1, etc. On peut *effacer* une zone indépendamment d'une autre, on peut *appeler* une zone comme sous-programme dans une autre (mais on ne peut pas appeler une *partie* d'une zone).

b) Examinons brièvement la machine HP-71.

Notons tout de suite que le BASIC de HP est proche du standard IEEE (américain), tandis que les constructeurs japonais (tous !) ont adopté le standard MS-X (américain lui aussi) : l'accord n'est pas pour demain...

La qualité du BASIC de la HP-71 correspond à une ROM de 64 K, avec 240 mots-clé disponibles. La mémoire vive est de 18 K. On peut encore ajouter 4 modules (en RAM ou en ROM). Le module MAT-PAC permet le traitement des matrices (jusqu'à l'inversion de matrices 45 x 45, qui prend 6 minutes !!...), la recherche des zéros d'une fonction numérique, l'intégration, etc. (On reprend ici le "contenu" des touches SOLVE et INTEGRATE de machines HP antérieures.)

La HP-71 permet de définir des entiers, des réels (avec des mantisses de 12 chiffres ; mais l'unité de calcul travaille avec des mantisses de 15 chiffres), et des nombres en "semi-précision" (SHORT), avec des mantisses à 5 chiffres. Avec le module MAT-PAC, on peut calculer aussi avec des nombres *complexes*, en pleine précision, ou en semi-précision. Les opérations matricielles s'étendent elles aussi aux matrices à composantes complexes.

MAT-PAC est écrit en langage-machine et "tourne" à une vitesse bien plus grande qu'un programme BASIC sur la même machine. (Un programme de recherche des zéros de polynômes travaille dans un éventail des exposants entre ± 50000...)

Il est possible de définir des *fonctions* ; une fonction peut avoir jusqu'à 14 variables, mais on ne peut malheureusement pas définir de fonctions de X ( ), *tableau* unidimensionnel à *n* composantes.

On peut définir des sous-routines, mais aussi de véritables *sous-programmes* identifiés par des *noms* et la liste de leurs *paramètres formels* ; ils peuvent se trouver "physiquement" n'importe où (par exemple : "à l'intérieur" d'un autre programme), et ils seront appelés (avec SUB CALL, suivi par la liste des *paramètres effectifs*) dans le programme en cours d'exécution. Les programmes gagnent ainsi en *modularité*, on peut mieux mettre en évidence les blocs logiques qui les composent.

La mémoire vive est organisée en *fichiers*, dont en premier lieu les *fichiers BASIC* contenant les programmes (ceux-ci sont indépendants les uns des autres, et leur nombre n'est limité que par la taille de la mémoire disponible, et la taille des programmes eux-mêmes) ; il y a ensuite les fichiers de *données* (DATA), des fichiers LEX (extension du langage), BIN (binaire), etc.

La fonction VAL présente une particularité nouvelle : si une chaîne de caractères forme une expression mathématique susceptible d'être *calculée*, VAL calcule cette valeur (ce qui n'est pas du BASIC standard).

Notons aussi la possibilité de construire des programmes *récursifs*, ce qui n'est possible ni dans le BASIC standard, ni en FORTRAN (mais qui fait penser à des langages plus récents : PASCAL, LOGO, LSE).

Les possibilités de la machine ne sont pas limitées au BASIC : un module en langage FORTH est disponible, et le constructeur annonce en FORTH une vitesse de 5 à 10 fois plus grande qu'en BASIC.

La machine (comme la HP-75) est chère, et sûrement inaccessible à l'étudiant. Avec les périphériques, son prix est de l'ordre du prix d'un APPLE. Ses possibilités sont *considérables*, fruit du travail d'une équipe puissante d'universitaires américains (groupés à Corvallis, dans l'Oregon). Mais les progrès techniques sont tellement rapides, qu'on peut se retrouver dans un an en possession d'une machine à la fois admirable et dépassée, et dont on n'aura pu exploiter que la millième partie des possibilités...

*****

Les machines que nous venons de décrire sont en BASIC, et "de poche", ce qui tend à les faire placer dans la catégorie des jouets peu sérieux.

On affirme souvent que le BASIC est un mauvais langage, qu'il donne de mauvaises habitudes ; ou encore qu'il est tellement facile, qu'on peut l'apprendre en trois jours.

Il me semble qu'on peut construire des programmes atroces dans n'importe quel langage, et qu'on peut mal utiliser un CRAY-ONE... Par contre, avec un apprentissage correct sur des concepts mathématiques bien formulés, et en tenant compte des phénomènes numériques nouveaux (instabilité, mauvais conditionnement, etc, etc.), le travail sur machine en BASIC peut être de grande utilité.

Ce n'est pas un langage de la recherche et de l'industrie, et ce n'est pas avec des programmes en BASIC que l'on va commander une navette spatiale. Mais ce langage, bien que "non structuré", permet la transcription précise des algorithmes, la mise en évidence des parties séquentielles, des points de branchement, des segments répétitifs, des tests d'arrêt. Dans *l'enseignement*, les problèmes sont bien délimités ("scolairement délimi-

tés", si l'on veut — mais sans délimitation et analyse, il n'y a ni enseignement, ni synthèse). Là où l'on examine *séparément* les concepts, les théorèmes, les méthodes des mathématiques, de la mécanique, de la physique — un langage relativement simple, largement répandu dans le monde, bénéficiant de l'appui de machines très variées et d'une littérature riche (et non toujours sans intérêt...) peut rendre service.

(On doit signaler l'apparition *très récente* du "TRUE BASIC", langage *structuré, compilé,* avec de riches possibilités *graphiques,* influencé par le PASCAL. IBM s'engage dans cette voie ; le "TRUE BASIC" a donc une formidable machine industrielle et de promotion à sa disposition).

2. Et que faire avec les très petites machines ?

Que ce soit sur machine de poche ou sur ordinateur, je ne suis pas sûr que l'on fait bien comprendre aux élèves de lycée (et aux étudiants) les questions *difficiles* et *importantes* liées à la résolution de la toute simple équation algébrique du second degré. Car il y a encore de nombreux programmes *d'ordinateur* où l'on transcrit simplement la formule scolaire — sans un mot sur la perte de précision par soustraction de nombres proches, les risques de dépassement, l'instabilité et l'amplification de l'erreur au voisinage du point critique correspondant au discriminant *nul*. Et on peut encore voir des programmes avec la "sortie" : PAS DE RACINE, si le discriminant est négatif. Et lire des articles où l'on parle du "comportement lunatique (!) de l'équation du second degré, qui a parfois deux racines, parfois une seule, et parfois n'en a pas du tout."... (La même formule scolaire apparaît d'ailleurs dans un programme de recherche des zéros des polynômes du fascicule "Mathématiques 1" de HEWLETT-PACKARD. Le MAT-PAC, lui, est bien performant, mais couvert par le secret...)

L'algorithme "scolaire" de recherche du PGCD passe parfois par la décomposition des deux nombres en facteurs, ce qui revient à remplacer un travail facile par un autre, plus compliqué... Ou alors on propose aux étudiants (aux élèves ?) l'algorithme par soustractions successives — sans que l'on se méfie des cas où l'un des nombres serait *très grand*, et l'autre petit. (Par exemple : PGCD de 777777777777 et 12.) Si l'algorithme par soustractions était justifié du temps où les machines n'avaient pas de multiplication rapide, pourquoi ne pas enseigner aujourd'hui l'algorithme exposé déjà par EUCLIDE ? Et qui conduit d'ailleurs, par l'intermédiaire de l'identité de BEZOUT, à un véritable foisonnement de résultats *théoriques* de grand intérêt (voir par exemple **LIPSON**, Addison-Wesley, 1981).

Est-ce que les élèves et les étudiants ont bien compris la formule des combinaisons C_n^p, et la *différence* entre une formule de définition, et le mode de calcul adéquat ? On peut le montrer ici, en expliquant comment transformer la formule qui semble "appeler" trois fois un sous-programme de factorielle, en un calcul avec une seule boucle, et pas de factorielle.

Les problèmes de **tri et classement** sont importants, et donnent lieu à des développements mathématiques fascinants. (Voir **D. KNUTH**, chez Addison-Wesley, vol. 3 — c'est sûrement l'ouvrage de référence.) J'ai eu l'occasion de voir des étudiants se servir *sur ordinateur* de la méthode des bulles (qui a un temps de calcul "en n^2"). Mais on peut enseigner cette méthode (qui est instructive et nécessaire) *sur calculette*, et faire éventuellement sur calculette aussi le tri de SHELL-METZNER, et même le QUICKSORT de HOARE (il y a une bonne variante BASIC dans l'article de **Y. LECLERC**, dans "LIST", octobre 1984). Quitte à passer *ensuite* sur ordinateur.

Les problèmes de classement peuvent être *illustrés* à l'aide du générateur de nombres aléatoires de la machine. (Et le générateur peut nous fournir aussi une suite *d'entiers* aléatoires, à "injecter" dans l'algorithme d'EUCLIDE : un bon générateur doit donner comme *rapport* entre le nombre de couples d'entiers premiers entre eux, et le nombre total de couples testés, la valeur $6/\pi^2$, conformément à un théorème de DIRICHLET, TCHEBYTCHEFF, CESARO.)

Il est instructif de *tester* les générateurs de nombres aléatoires des machines sur lesquelles on travaille. Rappelons (voir **FORSYTHE, MALCOLM et MOLER**, Prentice-Hall, 1977) que le générateur des machines IBM 360 avait la fâcheuse "propriété" que trois valeurs consécutives étaient liées par la relation

$$x_{k+2} = 6\,x_{k+1} - 9\,x_k\,.$$

De nombreux tests ont été proposés, et on peut enseigner, essayer quelques-uns *sur calculette*. (Après un nombre d'essais avec des échantillons allant jusqu'à 30000 nombres, et en utilisant une forme simplifiée du "test des 1000 boîtes" de KNUTH, il me semble que le générateur des machines SHARP est médiocre ; celui des HP est très bon. Celui des CASIO semble être le meilleur !) Est-ce que le générateur de nombres aléatoires de *chaque* ordinateur a été soigneusement testé ?

*****

On peut multiplier les exemples. Il existe aujourd'hui de nombreux ouvrages de "mathématiques numériques", où l'exposé théorique est entrecoupé d'*exemples*, parfois en FORTRAN (**CONTE et DeBOOR**, McGraw Hill, 1980 ; **DAHLQUIST et BJORK**, Prentice-Hall, 1974 ; **RICE**, McGraw Hill, 1983), parfois en ALGOL (**STOER et BULIRSCH**, Springer, 1980), parfois en BASIC (**ENGEL**, éd. Cédic, 1979), parfois sur calculette en langage-machine (**HENRICI**, Wiley, 1982). Le livre de **GEYMONAT** (Levrotto & Bella, 1981), ou le travail entrepris par l'équipe qui publie sous le nom de **Léonhard EPISTEMON** (éd. Cédic) semblent correspondre au même état d'esprit, au même désir d'œuvrer pour un enseignement plus vivant, plus efficace, non moins strict.

La table des matières de *chacun* de ces ouvrages peut être lue comme un "guide pour l'utilisateur d'une calculatrice en BASIC". Mais **que peut-on faire sur une calculette en BASIC avec 2 K de mémoire ?** Voici une liste (non limitative ; mais tout ce qui est sur cette liste **peut être fait** sur SHARP 1245, et une bonne partie a été ultérieurement "copiée" ou convenablement adaptée sur un SOLAR 16-40).

Arithmétique : PGCD, identité de BEZOUT, décomposition en facteurs premiers, polynômes, changements de bases de numération.

Calcul vectoriel : produit vectoriel, scalaire, mixte ; coordonnées polaires et sphériques ; les angles d'EULER.

Algèbre linéaire (I : méthodes directes) : décomposition $\mathbf{B} = \mathbf{LR}$ d'une matrice carrée — sans pivot, mais aussi avec pivot ; résolution de systèmes linéaires $\mathbf{Bx} = \mathbf{c}$, avec étude du mauvais conditionnement ; décomposition de CHOLESKY $\mathbf{A} = \mathbf{CC}^t$ pour une matrice symétrique et définie positive **A** ; résolution de systèmes $\mathbf{Ax} = \mathbf{c}$; inversion de matrices ; systèmes à matrice tridiagonale ; etc.

Simulation de processus infinis : suites, séries ; présentation des méthodes d'accélération de la convergence d'après AITKEN et RICHARDSON ; le procédé de RICHARDSON est présent dans de nombreux programmes d'ordinateur (et aussi dans la touche INTEGRATE des machines HP), sous le nom d'un auteur plus tardif - ROMBERG.

Interpolation : LAGRANGE, NEWTON ; par fractions continues, par splines cubiques — ce qui peut préparer les esprits en vue de l'étude des éléments finis.

Algèbre linéaire (II : méthodes itératives) : la méthode attribuée à GAUSS-SEIDEL ; la méthode du gradient conjugué ; la méthode de Von MISES et GEIRINGER (puissances) pour la valeur propre maximale, puis de WIELANDT (itération inverse) pour la recherche des valeurs propres et des vecteurs propres ; la recherche simultanée de *toutes* les valeurs propres (si réelles ; si possible...) d'après la méthode **LR** de RUTISHAUSER ; le *conditionnement* du problème des valeurs propres.

Dérivation numérique : dérivées premières, dérivées secondes ; dérivées partielles ; amélioration de la précision à l'aide du procédé de RICHARDSON.

Résolution d'équations pour une fonction numérique : méthode de balayage, interpolation linéaire, parabolique, méthode de NEWTON-RAPHSON, méthode du point fixe, procédé d'accélération de STEFFENSEN ; le cas des équations polynômes ; la méthode de NEWTON, de BAIRSTOW, la méthode **QD** de RUTISHAUSER pour la détermination simultanée de tous les zéros d'un polynôme.

Intégration numérique : trapèzes, point médian, mais surtout SIMPSON, GAUSS ; intégrales impropres ; intégrales multiples.

Equations différentielles ordinaires : méthodes de RUNGE-KUTTA, d'ADAMS, d'ADAMS-MOULTON ; autres méthodes à pas liés.

Toutes ces questions (et beaucoup d'autres) peuvent donner lieu à des programmes de 5 à 30 lignes de BASIC : les petites machines **pourraient changer nos enseignements**, pourraient donner plus d'initiative, de **plaisir de travailler** à nos élèves. Pourquoi les réunir par groupes de 12 ou 20 dans les mêmes salles, pourquoi ne pas leur donner le désir et le plaisir de travailler, de lire, de réfléchir **seuls** ?

3. Exemples

Tous les programmes sont écrits ici pour la PC-1245 : il n'y a pas de ELSE ; PRINT *arrête* l'affichage ; END est indispensable dès qu'il y a plus *d'un* programme dans la machine.

Il semble utile, *instructif*, de mettre sur une seule ligne les instructions *qui forment une unité logique*.

1° L'équation algébrique du second degré

Il s'agit d'un exercice accessible (peut-être) aux élèves de seconde (de troisième ?). (Si le discriminant est **négatif**, on pourrait trouver, dans l'attente de l'introduction des nombres complexes, une formulation "provisoire" ; il me semble qu'on ne doit jamais contredire le Théorème fondamental de l'Algèbre : "Toute équation polynôme de degré n possède n racines.")

Le programme facilite la compréhension des **choix**, des **tests**. Il "fait mieux" que la formule scolaire en ce qui concerne les risques de dépassement et de perte de précision lors du calcul de la racine réelle "petite" en valeur absolue — mais l'instabilité au voisinage du discriminant **nul** subsiste.

```
10 INPUT "A=";A,"B=";B,"C=";C:IF A = 0 THEN 100
20 IF B = 0 THEN 80
30 Y = -B/A/2:D = 1 + 2*C/B/Y
40 IF D = 0 THEN PRINT "X1=X2=":PRINT Y:GOTO 10
50 IF ABS D<1 E-9 THEN PRINT "DISCRIMINANT=":PRINT D
60 IF D > 0 THEN LET Y = Y + Y * SQR D : Z = C/A/Y:
   PRINT "X1=":PRINT Y:PRINT "X2=":PRINT Z:
   GOTO 10
70 Z = ABS Y * SQR -D:PRINT "RE X=":PRINT Y:PRINT
   "IM X=":PRINT Z:GOTO 10
80 Y = -C/A:Z = SQR ABS Y:IF Y < 0 THEN PRINT "RE
   X=0":PRINT "IM X=":PRINT Z:GOTO 10
```

```
 90 PRINT "X1=": PRINT Z: PRINT "X2=": PRINT -Z:
    GOTO 10
100 IF B = 0 THEN PRINT "A=B=C=0??": GOTO 10
110 PRINT "X=": PRINT -C/B: PRINT "EQUA.LINEAIRE!":
    GOTO 10
```

2° Le PGCD, les couples de nombres premiers entre eux

L'exemple est destiné aux lycéens : ils seront surpris de "voir" se former une bonne approximation du quotient $\frac{6}{\pi^2}$.

```
10  RANDOM: M = 0: INPUT "N="; N
20  FOR I = 1 TO N: A = INT (RND 1 Exp 9):
    B = INT (RND 1 Exp 9)
30  A = A-B * INT (A/B): IF A = 0 THEN 50
40  B = B-A * INT (B/A): IF B < > 0 THEN 30
50  IF A+B = 1 THEN LET M = M+1
60  NEXT I: PRINT "M/N="; M/N
```

3° Le tri de SHELL-METZNER

Le programme est un "décalque" du programme proposé par **MILLER** (éd. Sybex, 1981) : écrit pour des machines comme APPLE, TRS80, Z80, etc, il se traduit sans difficulté sur SHARP 1245 (ou sur SOLAR, etc). Nous aurions pu aussi bien choisir le "Tri rapide" de HOARE — il est cependant plus long, et plus difficile à analyser.

A la ligne 20 nous avons introduit les n premiers nombres naturels *par ordre décroissant*, le programme va les ranger *par ordre croissant*. On peut aussi classer des nombres mis sur une ligne DATA ; ou générer des *nombres aléatoires* que le programme va classer ; ou encore transformer les lignes qui suivent en programme de *classement de chaînes de caractères.*

```
10  INPUT "N="; N: DIM F(N): C = 0: E = 0
20  FOR I = 1 TO N: F(I) = N + 1 - I: NEXT I
30  I = INT (I/2): K = N - I
40  FOR J = 1 TO K: L = J
50  M = I + L : C = C+1 : IF F(L) < F(M) THEN 70
60  E = E+1 : H = F(L) : F(L) = F(M) : F(M) = H : L = L - I :
    IF L > 0 THEN 50
70  NEXT J : IF I > 1 THEN 30
```

```
80 PRINT "COMPARAISONS=" : PRINT C : PRINT
   "ECHANGES=" : PRINT E
90 FOR I = 1 TO N : PRINT I; ":"; F(I) : NEXT I
```

C'est tout ! Le "corps" du programme a été mis en évidence. En 30, la *première* valeur de I est N : en toute rigueur, il faut *l'écrire*, car une variable de boucle n'a pas de valeur définie à la *sortie* de la boucle. Mais nous nous sommes permis d'utiliser le fait que *sur les SHARP*, le contenu de I sera exactement N.

Une fonction SWAP serait bien utile.

Il est instructif de mesurer les *temps* de calcul, de compter les *comparaisons* et les *échanges*. On trouve :

n	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160
comp.	27	80	138	216	263	357	490	548	607	668	746	885	987	1181	1187	1332
éch.	13	36	69	92	105	168	201	224	237	260	293	396	329	472	425	528
temps	8	22	40	55	67	92	122	140	152	170	190	230	240	302	298	340

La méthode est *très supérieure* à la méthode des bulles. (Et le **QUICKSORT** est encore meilleur). Dans un "calcul réel", il faut *enlever* les compteurs, car ils "coûtent" du temps : si $n = 150$, il y a 1612 additions à faire... (Sur des machines plus puissantes, on utilisera la fonction TIME).

Le temps, le nombre des échanges et des comparaisons, augmentent "en gros" avec n^p (p voisin de 3/2) suivant une loi assez compliquée, et qui dépend des partitionnements successifs (et des *valeurs à ranger*, bien sûr). On peut observer cette "croissance irrégulière" en examinant les valeurs pour $n = 120$ à 150.

4° La résolution des systèmes d'équations linéaires par la méthode de GAUSS : décomposition B = LR avec pivot partiel

Il s'agit de l'une des questions les plus importantes dans les applications. D'après des statistiques fragmentaires, les problèmes d'Algèbre linéaire représentent 30 % à 50 % des *appels de programmes* (statistiques de la bibliothèque N.A.G., Oxford).

Ce qui est **essentiel** dans le programme principal, c'est l'appel de la sous-routine 150 (recherche du pivot, puis élimination de GAUSS ; si l'on préfère renoncer à la recherche du pivot, il suffit d'appeler GOSUB 200) ; ensuite, l'inversion de la matrice de permutation (en 90, 100) ; enfin la résolution des systèmes triangulaires (la sous-routine 220). Sur ordina-

teur, 150 et 220 seraient écrits comme *sous-programmes*, car ils seront appelés à l'intérieur de programmes autres que la résolution des systèmes.

Il va de soi que ce thème peut être beaucoup développé : calcul des résidus, amélioration itérative, etc.

En ne comptant pas les Remarques, le programme nécessite 0,75 K. Pour la matrice **B**, la matrice de permutation **P** et son inverse **Q**, enfin le vecteur du deuxième membre **D**, il nous faut $n^2 + 3n$ emplacements. (I, J, E, **P**, **Q** seront déclarés *entiers*, si possible : INTEGER, ou I%, J%, etc.). Sur la HP-71, on ferait OPTION BASE 1, pour ne pas "gaspiller" les emplacements d'indice 0. Sur les machines plus faibles, on peut "décaler les indices" (écrire I et utiliser 0 dans la machine, écrire 2 et utiliser 1, etc) : la complication qui en résulte est minime, *mais nous ne la recommandons pas* : il y a des "économies" à ne pas faire... Des systèmes avec $n = 7$ peuvent être examinés *sur la plus petite des machines*, pourquoi introduire une complication qui nous mènerait à $n = 8$?

```
 10 PRINT "SYSTEMES LINEAIRES" : PRINT "METHODE DE
    GAUSS" : PRINT "(PIVOT PARTIEL)"
 20 INPUT "N="; N: DIM B(N,N), P(N), Q(N), D(N): D = 1: E = 0
 30 FOR I = 1 TO N : PRINT "LIGNE:" ; I : P(I) = I
 40 FOR J = 1 TO N : PRINT "B(" ; I ; "," ; J ;")=" : INPUT
    B(I,J) : NEXT J : NEXT I
→50 FOR K = 1 TO N-1 : GOSUB 150 : PRINT "PIVOT=" :
    PRINT P : NEXT K : PRINT "DERNIER=" : PRINT B(N,N)
 60 FOR I = 1 TO N : PRINT "LIGNE:" ; P(I) : D = D * B(I,I) :
    REM : LECTURE DECOMPOSITION B = LR
 70 FOR J = 1 TO N : PRINT "LR(" ; P(I) ; "," J ; ")=" : PRINT
    B(I,J) : NEXT J : NEXT I
 80 PRINT "DETERMINANT=" : PRINT D : PRINT
    "ECHANGES=" ; E
 90 FOR I = 1 TO N : FOR J = 1 TO N : IF P(J) =
    I THEN LET Q(I) = J
100 NEXT J : NEXT I : REM : INVERSION MATRICE
    PERMUTATION
110 REM : RESOLUTION SYSTEMES TRIANGULAIRES
120 FOR I = 1 TO N : PRINT "D(";I; ")=" : INPUT D(Q(I)) :
    NEXT I
→130 GOSUB 220
140 FOR I = 1 TO N : PRINT "X(";I;")=" : PRINT D(I) : NEXT I :
    END

150 P = ABS B(K,K) : Q = P : REM : RECHERCHE PIVOT
160 FOR I = K+1 TO N : IF ABS B(I,K) > P THEN LET P =
    ABS B(I,K) : L = I
```

```
170 NEXT I : IF P = Q THEN 200
180 FOR J = 1 TO N : H = B(K,J) : B(K,J) = B(L,J) : B(L,J) =
    H : NEXT J
190 H = P(K) : P(K) = P(L) : P(L) = H : D = -D : E = E+1
200 P = B(K,K) : FOR I = K+1 TO N : B(I,K) = B(I,K)/P
210 FOR J = K+1 TO N : B(I,J) = B(I,J) - B(I,K) * B(K,J) :
    NEXT J : NEXT I : RETURN
220 FOR I = 2 TO N : FOR J = 1 TO I-1 : GOSUB 260 : NEXT J :
    NEXT I
230 FOR I = N TO 1 STEP -1 : IF I = N THEN 250
240 FOR J = I+1 TO N : GOSUB 260 : NEXT J
250 D(I) = D(I)/B(I,I) : NEXT I : RETURN
260 D(I) = D(I) - B(I,J) * D(J) : RETURN
```

Sur une très petite machine, il est préférable d'introduire les "modules" de 150 et de 220 "à leur place" dans le programme principal. (L'inversion de la matrice de permutation peut être incluse dans le sous-programme de décomposition). Mais sur HP-71 (SOLAR, APPLE, etc) ce sont bien deux *sous-programmes* (de nom, par exemple, DECOMP (A(,),B(),C(),D,E,F) et TRIANG(A(,),B(),C); les paramètres *formels* apparaissant ici n'ont pas les mêmes *noms*, mais sont du même *type* et sont *en correspondance* avec les paramètres *effectifs* lors de l'appel).

Et même sur une CASIO, on peut les écrire dans une certaine "zone"; écrire dans une autre zone la résolution des systèmes linéaires; dans une autre encore, un programme d'inversion des matrices; dans une autre, par exemple, le programme de la méthode **LR** de RUTISHAUSER pour la détermination simultanée de toutes les valeurs propres (**attention**: la décomposition doit se faire ici *sans échange de lignes*). On verrait alors comment les programmes s'organisent "par modules"; et il est au moins aussi important de "penser de manière structurée", que d'apprendre un langage de programmation structuré...

Applications numériques

Nous suggérons d'utiliser la matrice **T** symétrique, définie positive, de composantes entières ne dépassant pas 10 (le point de départ est ici la matrice de WILSON ou la matrice de RUTISHAUSER); ensuite $\mathbf{N} = \mathbf{T}\,\mathrm{diag}\,(3,2,1)\,\mathbf{T}^{-1}$; c'est-à-dire:

$$\mathbf{T} = \begin{bmatrix} 10 & 9 & 1 \\ 9 & 10 & 5 \\ 1 & 5 & 9 \end{bmatrix}, \qquad \mathbf{N} = \begin{bmatrix} 617 & -719 & 331 \\ 410 & -477 & 220 \\ -250 & 293 & -134 \end{bmatrix}, \qquad (\det \mathbf{T} = 1,\ \det \mathbf{N} = 6)$$

Nous proposons au lecteur de résoudre les systèmes $\mathbf{Tx} = (-5,4,18)$, puis $\mathbf{Nx} = (3048, 2024, -1238)$. (Notations abusives...)

Il pourra passer ensuite à des deuxièmes membres "légèrement perturbés", en écrivant par exemple 4.1 au lieu de 4, respectivement 2024.1 au lieu de 2024. Non, il n'y a pas d'erreur : les deux matrices sont *très mal conditionnées*, et ceci n'a aucun rapport avec la "petitesse" du déterminant.

La matrice **N** peut servir pour illustrer le *mauvais conditionnement du problème des valeurs et des vecteurs propres* (l'exemple est analogue à un exemple de **DAVIS** et **MOLER**, mais présente des propriétés de "sensitivité" beaucoup plus accentuées). (Le conditionnement de **N** ne dépend ni de *son* déterminant, ni de *ses* valeurs propres (1, 2, 3), mais des valeurs propres de $\mathbf{M} = \mathbf{N}^t\mathbf{N}$. Cela vaut la peine de les calculer...)

5° Les équations-polynômes, méthode de BAIRSTOW

Les équations-polynômes à *coefficients réels* sont résolues par mise en évidence de facteurs *linéaires* ("fragment Newton" du programme) ou quadratiques ("fragment Bairstow"). Nous nous limitons ici à une succession de calculs de Bairstow : formation de facteurs $x^2 + Sx + P$, *déflations* successives, amélioration des facteurs à l'aide de *l'équation initiale*, enfin résolution des équations quadratiques "améliorées".

La méthode peut être étudiée dans de nombreux ouvrages, par ex. dans le traité classique de **E. DURAND**, vol. I (éd. Masson, 1960).

La méthode est à convergence *quadratique* au voisinage des zéros, et elle *échoue* au voisinage des zéros multiples (ou même de zéros "très proches"). (Le programme du MAT-PAC, basé sur la méthode de LAGUERRE qui est à convergence *cubique*, et utilisant le calcul avec des nombres complexes, est de très loin meilleur).

Si le polynôme est de degré *impair*, il suffit d'introduire une "racine artificielle" $x = 0$, en le multipliant par x.

Les *coefficients* sont en DATA (on aurait pu les introduire au clavier, ou encore *choisir* entre les deux variantes, avec ON K GOTO ou GOSUB).

En 30, nous avons choisi une initialisation peu prétentieuse ($S = 0$, $P = 0$). Dans les "vrais" programmes, on utilise des estimations classiques de CAUCHY, LAGUERRE, FEJER, etc.

La sous-routine 90 "charge" le tableau des coefficients K(J). (Sur certaines machines, on ne peut pas utiliser la même lettre comme nom de tableau, et nom de variable "scalaire" : c'est le cas de la HP-71...) La sous-routine 100 fait le calcul de BAIRSTOW : sur une machine avec ELSE, on devrait l'utiliser par exemple en 150, 160. Les lignes 110, 120 seront aisément comprises par le lecteur persévérant...

Les déflations se font en 170, les couples de zéros sont calculés en 190 (en évitant une partie des pièges de la "formule scolaire"). La *somme* et le *produit* de tous les zéros sont formés en S(0) et P(0). En H et L nous avons mis des compteurs des itérations ; en T, une *tolérance* : elle ne doit pas être excessivement fine, sous peine de voir la machine "se planter" dans une boucle sans fin...

Sur une machine "à grand écran", on peut faire PRINT U, V : ceci permet "d'observer" le processus de convergence. Sur PC-1245, PRINT arrête le calcul, et il est préférable de l'éviter. Avec BEEP, on se rend compte que la machine est en train de "faire quelque chose"...

```
10 PRINT "EQUATION-POLYNÔME" : PRINT "(DEGRE PAIR," : PRINT "RACINES SIMPLES)" : PRINT "METHODE BAIRSTOW"
20 INPUT "DEGRE=" ; N : M = N : DIM K(N), S(N/2), P(N/2) : INPUT "TOLERANCE=" ; T : GOSUB 90
30 H = 0 : L = 0 : FOR I = 1 TO N/2 : S = 0 : P = 0
40 GOSUB 100 : GOSUB 170 : PRINT "DEGRE COURANT=" ; N : NEXT I
50 N = M : PRINT "DEGRE INITIAL=" ; M : PRINT "ITERATIONS=" ; H+L : INPUT "TOLERANCE FINE=" ; T : P(0) = 1 : S(0) = 0
60 RESTORE : GOSUB 90 : FOR I = 1 TO N/2 : S = S(I) : P = P(I) : GOSUB 100 : P(0) = P(0) * P : S(0) = S(0) + S
70 GOSUB 190 : NEXT I
80 PRINT "SOMME RACINES=" : PRINT -S(0) : PRINT "PRODUIT RACINES=" : PRINT P(0) : PRINT "ITERATIONS=" ; H+L : END

90 FOR J = 0 TO N : READ K(J) : NEXT J : RETURN

100 BEEP 1 : L = L+1 : IF L > = 2*M THEN LET H = H+L : L = 0 : PRINT "ITERATIONS:" ; H
110 B = 0 : C = 0 : E = 0 : F = 0 : FOR J = 0 TO N-1 : GOSUB 180
120 D = E : E = F : F = C - S*E - P*D : NEXT J : J = N : GOSUB 180
130 A = E*E - D*F
140 U = (B*E - C*D)/A : V = (C*E - B*F)/A : S = S+U : P = P+V
150 IF ABS U OR ABS V > T THEN 100
160 S(I) = S : P(I) = P : RETURN
```

```
170 B = 0 : C = 0 : FOR J = 0 TO N : GOSUB 180 : K(J) = C :
    NEXT J : N = N-2 : RETURN

180 A = B : B = C : C = K(J) - S*B - P*A : RETURN

190 PRINT "DEUX RACINES:" : IF S = 0 THEN 240
200 Y = -S/2 : D = 1 - 4 * P/S/S
210 IF ABS D < T THEN PRINT "DISCRIMINANT=" : PRINT D
220 IF D > 0 THEN LET Y = Y + Y * SQR D : Z = P/Y : PRINT
    "X1=" : PRINT Y : PRINT "X2=" : PRINT Z : RETURN
230 Z = ABS Y * SQR -D : PRINT "RE X=" : PRINT Y : PRINT
    "IM X=" : PRINT Z : RETURN
240 Y = SQR ABS P : IF P < 0 THEN PRINT "X1=" : PRINT Y :
    PRINT "X2=" : PRINT -Y: RETURN
250 PRINT "RE X=0" : PRINT "IM X=" : PRINT Y : RETURN

260 DATA 7, 9.5, -4.6, -9.55, -.9589, 1.46135, -.67799,
    -1.19706
270 DATA -.12207968
280 DATA 1.30485472, .91050492, -.23093984, -.2950856
290 DATA -.00958136, .02415288, .00161712, -.00036288
```

Applications numériques

On peut commencer par considérer l'équation $x^2+1=0$ (DATA 1, 0, 1, degré 2), puis $x^2-1=0$ (DATA 1, 0, −1, degré 2).

On prendra ensuite $x^3-2x^2-x+2=0$ (DATA 1, −2, −1, 2, 0, degré 4).

Pour le polynôme $7x^6+6x^5+5x^4+4x^3+3x^2+2x+1$,
on trouve trois couples de zéros complexes conjugués, de modules voisins de 0.7, *d'où une convergence relativement lente.*

Le polynôme dont les coefficients ont été écrits sur les lignes 260 à 290 est de degré 16 ; il a été construit en multipliant le polynôme de degré 6 précédent, par le polynôme de degré 10 ayant les zéros 0.1, −0.2, 0.3, −0.4, 0.5, −0.6, 0.7, −0.8, 0.9, −1.

Sur la machine considérée, on obtient les valeurs
−.2000000231, .1000000002, −.3999981753, .2999999173, .6999995487,
−.6000102947, .9000001608, −.7999929991, −1.000000940, .5000004679,

$$-.63411\ \ 06765 \pm .28765\ \ 37272\ i,$$
$$.41068\ \ 42033 \pm .63988\ \ 94043\ i,$$
$$-.20514\ \ 37584 \pm .68379\ \ 69494\ i.$$

Il y a en général 4 à 6 chiffres significatifs corrects. Suivant l'initialisation choisie, il y a 85 à 110 itérations, dont les 4/5-ièmes dans la première partie du programme.

En éliminant les *commentaires*, on peut étudier sur la PC-1245 des équations-polynômes de degré 25. On peut s'attendre à des difficultés en présence de zéros proches. Nous avons isolé sur la ligne 270 un coefficient : en *changeant son signe*, certains des zéros *réels* du polynôme de degré 16 précédent vont engendrer des couples de zéros *complexes conjugués* ; il y aura des couples de zéros de modules proches, *et la machine utilisée ici ne pourra pas les mettre en évidence* (avec ce programme...)

6° Equations différentielles, la méthode de RUNGE-KUTTA

Considérons le problème de CAUCHY pour un système (non autonome) de deux équations

$$\dot{u} = f(t,u,v), \quad \dot{v} = g(t,u,v), \quad u(0) = u_0, \quad v(0) = v_0 .$$

On passe aisément au cas d'un système de plusieurs équations, ou à des équations d'ordre supérieur. Le cas de l'équation unique est facile.

Sur une machine ne disposant pas de la définition de *fonctions*, nous devons recourir à des sous-routines.

Le programme est écrit ici pour un *exemple* : l'équation du *pendule circulaire*, qui conduit au *système*.

$$\dot{u} = v \ , \quad \dot{v} = -(g/l) \sin u.$$

La qualité du résultat peut être estimée à l'aide de *l'intégrale de l'énergie*, dont la valeur sera mise dans une mémoire, soit W :

$$v^2/2 - (g/l) \cos u = C^{ste}.$$

Au tout début, il faut introduire en mémoire la constante g/l.

Vers la fin (*entre* la ligne d'incrémentation de la variable indépendante, et la ligne de RETURN) on écrit les deuxièmes membres des équations, multipliées par le *demi-pas*, noté classiquement η (= "éta"), et mis ici en E.

Le programme a été construit en collaboration avec **M. HOCQUEMILLER.**

```
10 INPUT "G/L=" ; G , "T0=" ; T , "U0=" ; U , "V0=" ; V
20 W = V*V/2 - G * COS U
30 INPUT "H/2=" ; E, "ITER=" ; N
40 FOR I = 1 TO N : K = 0 : L = 0 : A = U : B = V
50 GOSUB 110 : GOSUB 100 : GOSUB 120
60 GOSUB 110 : GOSUB 120 : U = U + P : V = V + Q :
100 GOSUB
 70 U = A + K/3 : V = B + L/3 : NEXT I
80 PRINT "T=" ; T : PRINT "U=" : PRINT U : PRINT "V=" :
   PRINT V
90 PRINT "ERR.REL. INT.PREM.=" : PRINT (V*V/2 - G *
   COS U)/W - 1 : GOTO 30
100 T = T + E
110 P = V * E : Q = - G * E * SIN U : U = A + P : V = B + Q
120 K = K + P : L = L + Q : RETURN
```

Dans le cas d'un **système** de *n* équations, on peut utiliser le même schéma, avec des **matrices** uni-lignes de noms B, K, P, U (A doit être *évité* sur la PC-1245...). Une fois n'est pas coutume — les *sous-routines* sont ici bien pratiques. (Sur ordinateur, on se sert des instructions matricielles, par exemple MAT U = A + P, MAT K = K + P, etc).