LA PRISE EN COMPTE DES NOTIONS DE BOUCLE ET DE VARIABLE INFORMATIQUE DANS L'ENSEIGNEMENT DES MATHEMATIQUES AU LYCEE[1]

Chi Thanh NGUYEN et Annie BESSOT
Didactique des Mathématiques, Laboratoire LEIBNIZ
Université Joseph Fourier, Grenoble I

Résumé. Une étude de la notion d'algorithme dans les programmes et les manuels nous permet de questionner la prise en compte, dans l'enseignement secondaire, des deux notions de boucle et de variable, fondamentales pour la programmation des algorithmes. Des expérimentations, dans une classe de 2nde et une classe de 1ère scientifique, complétées par une enquête auprès d'enseignants de mathématiques, apportent des éléments de réponse aux questions posées.

La notion d'algorithme est d'une importance égale dans la constitution des sciences mathématiques et informatiques. Depuis les années 1970 jusqu'à aujourd'hui, on peut attester de la présence d'algorithmes et de leur programmation dans les programmes des mathématiques au lycée.

Quelle est alors la vie de ces objets dans l'enseignement des mathématiques au lycée ? Les connaissances mathématiques sous-jacentes facilitent-elles ou, au contraire, créent-elles des difficultés à l'émergence de certaines notions fondamentales de l'informatique liées à la programmation des algorithmes ?

I. De la notion d'algorithme à celles d'instruction et de variable

La notion d'algorithme, comme outil des mathématiques, est ancienne puisqu'elle intervient dès qu'il y a calcul numérique, l'algorithme dit d'Euclide étant un exemple d'une intervention comme élément de démonstration.

Hilbert (1862-1943) pose le premier le problème de l'existence d'un algorithme universel pour une classe de problèmes[2].

---

[1] Cet article s'appuie sur un mémoire de DEA Environnements Informatiques d'Apprentissage Humain et Didactique (Nguyen 2002) préparé sous la direction d'Annie Bessot et de Philippe Jorrand au sein de l'équipe DDM du laboratoire Leibniz. Les discussions, avec Alain Birebent au sein de l'équipe DDM, ont permis de reprendre avec profit certaines parties de l'article. Nous le remercions.

Les années 30, en consacrant l'existence de problèmes indécidables,[3], rendent nécessaire la recherche d'une définition de la notion d'algorithme.

Pour montrer qu'il n'existe pas d'algorithme, ou même l'énoncer rigoureusement, nous avons besoin d'une définition de ce qu'est un algorithme. (Matiiassévitch 2000)

Ce moment est celui de l'émergence dans la théorie mathématique de l'objet « algorithme ». Des logiciens et des mathématiciens comme Gödel, Church, Kleene, Post et Turing cherchèrent à formaliser la notion d'algorithme. Nous retiendrons la définition attachée à la machine de Turing :

Turing utilisa une machine hypothétique que nous appelons 'machine de Turing'. Turing définissait l'algorithme comme un ensemble d'instructions pour sa machine simple. (Goldschlager et *al.*, 1986)

En effet, les différentes définitions formelles de la notion d'algorithme sont équivalentes (théorie de la calculabilité) :

Tout ce qui peut être calculé peut l'être par une machine de Turing. (Goldschlager et *al.*, 1986)

Nous avons examiné une vingtaine d'ouvrages de mathématiques et d'informatique actuels et les définitions du mot « algorithme » qui y sont présentes (Nguyen 2002). Dans ces définitions, plusieurs caractéristiques sont mises en avant. Deux d'entre elles sont prépondérantes aussi bien en mathématiques qu'en informatique : « la séquencialité » et « l'effectivité ».

Séquencialité. Un algorithme peut être décrit par une suite d'actions ordonnées.
Effectivité. L'exécution d'un algorithme donne toujours lieu au résultat escompté.

Ces caractéristiques semblent préalables aux choix de l'écriture des algorithmes dans un langage de programmation compréhensible par une machine. Un tel langage nécessite une description discrète des opérations constituantes d'un algorithme, et une particularisation des objets sur lesquels porte le calcul. Il faut donc se donner les moyens pour décrire un tel procédé. N'Guyen (1996) distingue :

Plusieurs niveaux de description qui se différencient par leur degré d'abstraction ou de concrétisation. [...] l'un qualifié de bas niveau[4] et l'autre de haut niveau.

Dans ce qui suit, nous présentons deux notions liées à la description des algorithmes en langages de programmation « de haut niveau » et de style impératif[5].

---

[2] On se réfère ici au 10ème problème des 23 problèmes que Hilbert a présenté en 1900 au Second Congrès International des Mathématiciens. Ce problème peut s'énoncer comme suit « Existe-t-il un algorithme universel pour la résolution des équations diophantiennes ? ». Une équation diophantienne est une équation de la forme $D(x_1, x_2, \ldots, x_m) = 0$ où D est un polynôme à coefficients entiers, positifs ou négatifs.

[3] Un problème est réputé décidable lorsqu'il existe un algorithme dont l'entrée est une instance du problème et dont la sortie est une réponse au problème.

[4] « Ce bas niveau correspond à un haut niveau d'abstraction où concevoir un algorithme, c'est pourvoir une machine rudimentaire[...] d'un logiciel qui la fait fonctionner. Cette machine illustre le double concept de transformation (tête de lecture - écriture) et de mémorisation (état de l'unité de contrôle ou processeur) des données. » (N'Guyen1996)

Le style apparu historiquement le premier fut qualifié d'impératif parce qu'il ramène l'écriture d'un algorithme à une succession d'ordres impérieux intimés à la machine. (Ganascia 1998).

I.1. La programmation des algorithmes en style impératif

La programmation présentée dans les manuels scolaires et dans l'enseignement de mathématiques, que ce soit en Basic, Pascal ou dans un langage de calculatrices est du style impératif. C'est pour cette raison que nous restreignons notre étude aux algorithmes écrits dans le style impératif.

Ce style est fondé sur deux sortes de structures : les structures de contrôle (appel de fonction, branchement, itération) et les structures de données. Le contrôle permet d'assembler des instructions qui opèrent séquentiellement sur des données, en transformant l'état de la mémoire.

En programmation impérative, on décrit l'état d'une machine et en particulier de sa mémoire. Un programme est donc une suite d'opérations que l'on fait pour modifier la mémoire d'une machine en fonction de l'état de cette mémoire. (Veigneau 1999, p13)

Une instruction typique de ce style est l'initialisation d'une variable par une valeur, et les structures de contrôle typiques sont les itérations (en quelque sorte, les boucles). Les deux notions « instruction » et « variable » sont alors fondamentales. La notion de valeur cède le pas à celle de variable, la notion d'expression à celle d'instruction.

I.2. Instructions, variables

- Instructions

On ne peut pas écrire un algorithme sans utiliser une succession d'instructions. En général, on utilise les instructions suivantes (dites élémentaires) :

- Instructions d'affectation : Exp → Variable

C'est une instruction fondamentale portant sur les variables, elle consiste à changer la valeur de la variable. Cette instruction indique que la variable doit prendre une nouvelle valeur, celle de l'expression *Exp*. Elle se lit : « *Variable* prend la valeur de *Exp* ».

- Instructions conditionnelles[6] :
  - Si Condition ... Alors Action1 ... Sinon Action2 ... FinSi ;
  - Si Condition ... Alors aller à ... FinSi ;

- Instructions conditionnelles[7] :
  - Si Condition ... Alors Action1 ... Sinon Action2 ... FinSi ;
  - Si Condition ... Alors aller à ... FinSi ;

---

[5] Il existe d'autres styles. Par exemple, le style fonctionnel repose sur l'utilisation intensive des fonctions. Il est fondé sur la notion de calcul plutôt que sur la notion de modification de l'état de la mémoire de la machine. Dans ce style, une variable est un nom pour une valeur et non pour une case mémoire. On ne peut pas modifier une variable puisqu'une variable fait référence à une valeur, c'est-à-dire une constante.

[6] Dans le cadre de ce travail, nous ne parlons pas de l'instruction d'aiguillage.

[7] Dans le cadre de ce travail, nous ne parlons pas de l'instruction d'aiguillage.

- Instructions itératives :
  - Tant que Condition ... Faire Action ... FinTantque ;
  - Pour variable = départ jusqu'à arrivée Faire Action ... FinPour ;
  - Répéter Action Jusqu'à Condition FinRépéter ;

Dans la suite de notre article, nous considérons le quatre types de boucles suivantes : B1 (Tant que ... Faire), B2 (Si ... Aller à), B3 (Pour ... Faire) et B4 (Répéter ... Jusqu'à).

Les trois types d'instructions (affectation, conditionnelle, itération) déterminent trois formes de base de la construction des algorithmes : la séquence, le branchement et l'itération.

Ces trois formes sont suffisantes pour concevoir tout algorithme (Goldschlager et al., 1986).

- Variables

Dans le style impératif, une variable est un nom que l'on donne à une case de la mémoire. Les deux opérations principales que l'on peut effectuer sur une variable sont l'initialisation par lecture et l'initialisation par affectation. Elles servent à attribuer des valeurs initiales aux variables ou à remplacer une valeur précédemment écrite par une autre.

Construire un algorithme pour résoudre un problème, c'est souvent, trouver une méthode de calcul d'une solution à ce problème, puis écrire cette méthode sous forme d'une suite d'instructions relevant des trois types d'instructions élémentaires ci-dessus : initialisation (variable), condition ou itération (boucle).

Les deux notions de variable et boucle sont donc fondamentales dans la programmation du style impératif. Nous centrerons notre travail sur ces deux notions.

II. Analyses des programmes et des manuels

II.1 Évolution des programmes des années 1970 à 2000[8]

Les programmes des années 70 sont caractérisés par la réforme des mathématiques modernes. Les années 80 imposent une contre-réforme. Dans les années 90, une autre réforme se met en place influencée par les progrès de la technologie informatique que le nouveau programme 2000 va prolonger.

a. Les programmes des années 1970

L'accent est mis sur des exposés strictement axiomatiques, les problèmes de fondements, les structures de groupes, d'anneaux et de corps. (Lê Van 2001, p. 37)

Dans ces programmes, la notion d'algorithme est introduite implicitement sous forme d'organigrammes en relation avec d'autres savoirs comme fonctions, résolution exacte d'une équation ou d'un système linéaire, et avec la logique :

---

[8] Nous résumons ici les résultats de notre analyse (Nguyen 2002).

Il est demandé aux élèves de mettre en œuvre, sur chacun des exemples à leur portée, les idées générales concernant la conduite logique : d'un raisonnement (...) ; d'un algorithme (substitutions et combinaisons linéaires). (programmes 1970)

Un algorithme sert ainsi à décrire un procédé permettant de calculer ou de trouver des solutions exactes d'une équation et de discuter l'existence de ces solutions. Le rôle de cette notion pour le raisonnement est alors mise en relief. Mais les algorithmes mentionnés ne sont jamais itératifs : la notion de boucle avec condition d'arrêt est absente.

Des éléments d'arithmétique sont présents à cette époque avec un point de vue structurel. L'algorithme d'Euclide n'est pas mis en avant comme méthode de recherche d'un PGCD (Ravel 2000). De la même façon, la mise hors programme de la formule des accroissements finis, et l'absence de la notion de suites font que toutes les méthodes itératives d'approximation d'une équation ne peuvent pas exister. La vie de la notion d'algorithme semble alors fragile. Parmi les manuels étudiés de l'époque (Collections Cossart ; Riche ; Queysanne-Revuz ; Durrande), seul Durrande parle de la notion d'itération à propos de l'usage de la machine à calculer de bureau en Seconde.

b. Le programme des années 1980

Dans ce programme, l'importance accordée au thème des suites en relation avec l'étude de fonctions, la place des activités réservées aux élèves et l'introduction officielle des calculatrices dans l'enseignement des mathématiques s'accompagnent d'une place importante laissée à la notion d'algorithme. L'intention d'introduire l'enseignement de la programmation est officiellement explicitée. De plus :

Le domaine de suites numériques est considéré comme un premier terrain pour l'introduction d'éléments algorithmiques. (Lê Van 2001, p. 174)

L'apparition des calculatrices dans les établissements scolaires valorise les activités portant sur les algorithmes pour conjecturer et contrôler les résultats. La notion d'algorithme n'est pas abordée comme un objet de savoir à enseigner mais comme un outil idéal pour illustrer, d'abord l'ambition de développer les démarches scientifiques chez l'élève, ensuite l'utilisation des calculatrices. L'aspect algorithmique devient présent dans le programme en même temps que l'aspect structurel du programme précédent disparaît (cf. Lê Van, 2001 ; Ravel, 2002). La notion d'algorithme doit outiller, dans le domaine de l'analyse, les techniques des tâches « approximation d'un nombre réel » et « approximation d'une solution réelle d'une équation ». C'est la conception expérimentale de l'activité mathématique « conjecturer, contrôler les résultats, comparer les différentes méthodes » qui est alors mis en avant.

A cette époque, la plupart des calculatrices des élèves ne sont pas programmables. De ce fait, la validation de l'écriture des algorithmes ne peut relever de son implémentation dans la machine de l'élève, mais de la réalisation « étape par étape » à la main (ou à la calculatrice). D'ailleurs, bien que les algorithmes mentionnés soient souvent itératifs, les deux notions de boucle et de variable sont absentes du programme.

c. Le programme des années 1990

Ce programme reprend pour l'essentiel, les objectifs et la substance des programmes précédents concernant la notion d'algorithme et ses applications.

Il convient de mettre en valeur *les aspects algorithmiques* des problèmes étudiés (approximation d'un nombre à l'aide des suites, recherche des solutions approchées d'une équation numérique, calcul de valeur approchée d'une intégrale etc.). On explicitera ce type de démarche sur quelques exemples simples : construction et mise en forme d'algorithme, comparaison de leur performance pour le traitement d'un même problème (...) (programme 1990)

Un élément nouveau est l'introduction d'activités de programmation :

- En 2nde et 1ère, programmer le calcul de valeurs numériques d'une fonction sur des exemples simples.
- En Terminale, programmer le calcul des valeurs d'une fonction, le calcul des termes d'une suite récurrente.
- A partie de la 1ère, savoir programmer une instruction séquentielle ou conditionnelle.
- En Terminale, savoir programmer une instruction itérative comportant éventuellement un test d'arrêt.

(extraits des programmes 1990)

La notion de boucle est ainsi introduite officiellement en 1990 au niveau 1ère S et Terminale S. Mais aucune rubrique des programmes n'est réservée à l'enseignement de la programmation : comment cela peut-il se réaliser dans des manuels ? En réalité, il faut attendre la fin des années 1980 et le début des années 1990 pour voir l'apparition de la traduction de quelques-uns de ces algorithmes en langage Basic, Pascal ou calculatrices dans certains manuels[9]. Dans ces manuels, la responsabilité laissée à l'élève est la plupart du temps d'exécuter les programmes ou d'expliquer les algorithmes donnés. Programmer ou construire un algorithme n'est *a priori* pas du ressort de l'élève.

Concernant les algorithmes en arithmétique (programme 1998), le programme suggère de mettre en place des « notions élémentaires permettant l'élaboration d'algorithmes simples et fondamentaux ». Mais cette suggestion n'est pas suivie par les manuels (Ravel 2000).

d. Le programme des années 2000

Ce programme accorde toujours de l'importance à l'informatique, et souligne le caractère mathématique de certaines notions comme boucle et test :

Certaines notions informatiques élémentaires (boucle, test, récursivité, tri, cheminement dans des graphes, opérations sur des types logiques) font partie du champ des mathématiques et pourraient être objets d'enseignement dans cette discipline. (programme 2000, 1ère S)

En même temps il recommande l'usage de certains logiciels (tant sur ordinateurs que sur calculatrices).

- L'informatique, devenue aujourd'hui absolument incontournable, permet de rechercher et d'observer des lois expérimentales dans les deux champs naturels d'application interne des mathématiques : les nombres et les figures du plan et de l'espace.

---

[9] Par exemple dans *Mathématiques* Collection Daniel Fredon, Edition Colin 1990 ou *Math* Collection Terracher, Edition Hachette 1994.

- Il est ainsi nécessaire de familiariser le plus tôt possible les élèves avec certains logiciels.
- On exploitera les possibilités offertes par les tableurs, par les grapheurs et par les logiciels de géométrie.

L'aspect algorithmique d'une fonction est mentionné en classe de 2nde en relation avec l'idée intuitive d'instruction élémentaire :

Sur le tableur, explicitation des différentes étapes du calcul d'une formule en appliquant d'une colonne à l'autre une seule opération (+, −, ×, /, carré, √). Explicitation de l'enchaînement des fonctions conduisant de x à f(x). (programme 2000)

En classe de 1ère S le programme insiste :

Compte tenu de l'horaire imparti et des débats en cours, il n'est proposé aucun chapitre informatique. Néanmoins, *l'élève devra mettre en œuvre, notamment sur sa calculatrice, les notions de boucle et test.* (les italiques sont de nous)

En relation avec l'usage des logiciels, le programme souligne que :

Il est à noter aussi que l'informatique, sanctionnant immédiatement et visiblement les fautes de syntaxe, contribue à former à l'esprit de rigueur, notamment dans la manipulation des objets traités (nombre, variable, figures géométriques).

L'utilisation des logiciels[10] ne rentre-t-elle pas en conflit avec l'ambition de former l'esprit algorithmique ? On notera que les méthodes d'approximation d'une solution, qui était un habitat privilégié de la notion d'algorithme dans les programmes précédents, mobilise maintenant à la fois la calculatrice et le tableur.

- Calcul des termes d'une suite sur calculatrice ou tableur (programme 2000)

Le terme tableur est ambigu : s'agit-il du tableur de la calculatrice ou du logiciel tableur (comme Excel) ?

Il est important pour notre propos d'établir une distinction claire entre les deux acceptions. Nous utiliserons respectivement « *tableur-calculatrice* » et « *tableur-logiciel* » quand il nous semblera pertinent de séparer ces deux significations. Certaines calculatrices actuelles fournissent des valeurs d'une fonction ou d'une suite numérique grâce à des commandes pré-établies sans obligation de construction d'un programme à partir d'un algorithme : nous nommerons tableur-calculatrice un tel tableau et les gestes instrumentés qui permettent de le produire (ceux que l'opérateur doit accomplir pour entrer la valeur initiale, le pas et la fonction). Un tableur-logiciel (comme Excel) demande que l'on formule des procédures de calcul : il relève d'une problématique de la programmation en style fonctionnel dans laquelle les notions de boucle et de variables (comme case d'une mémoire) ne sont pas pertinentes.

En bref, le programme 2000 souligne, dès la 1ère, la mise en œuvre des notions de boucle et de variable sur les calculatrices (ce qui n'était pas le cas des programmes précédents). On peut s'interroger sur l'influence de l'usage des calculatrices contemporaines et des logiciels sur cette mise en œuvre.

---

[10] Les réponses d'enseignants à une enquête menée dans Nguyen (2002) montrent qu'ils respectent les directives du programme en accordant une place prépondérante à l'utilisation des logiciels, en particulier au tableur.

II.2 Regard sur trois collections de manuels (2nde et 1ère S) [11] du programme 2000

a. La partie identifiable au *cours*

L'analyse des programmes 2000 montre qu'en 2nde et en 1èreS une niche possible de la notion d'algorithme serait de permettre une étude des notions de programmation comme boucle avec condition d'arrêt et variable. Comment cela est-il réalisé dans les manuels examinés ?

Dans la partie strictement cours des manuels examinés, tous les algorithmes sont « montrés » (souvent sous la forme d'organigramme) dans des problèmes résolus ou dans les travaux dirigés : en Seconde, dans le chapitre des fonctions, ces activités (pour décomposer une fonction donnée en opérations élémentaires ou calculer les valeurs numériques d'une fonction) s'intitulent « *calculogramme* » (Déclic), « *montage des fonctions* » (Belin) ou « *boîte noire* » (Bréal). Elles pourraient permettre d'introduire la comparaison des algorithmes. Par exemple, pour un calcul des valeurs de $x^2 + 3x + 2$, différents schémas (ou différents algorithmes) sont « économiquement » dissemblables en termes de coût ou de nombre d'opérations. Cette piste n'est jamais suivie. Il s'agit simplement de *montrer* un algorithme dans la partie cours que l'élève devra reprendre dans la partie exercices. De plus, comme le nombre des méthodes de résolution approchée a été réduit et leur présentation reportée en Terminale, nous pouvons conclure que la comparaison des algorithmes a pratiquement disparu de l'enseignement des mathématiques en Seconde et en Première.

Les opérations concernant les variables dans les programmes informatiques des manuels examinés sont en général écrites sans mentionner que la notion de variable renvoie à des mémoires d'affectation. Cependant Belin et Déclic accompagnent les programmes de commentaires concernant les instructions. Nous avons essayé de repérer les diverses explications portant sur l'affectation (indissociable du symbole $\rightarrow$) à l'extérieur et à l'intérieur d'une boucle. Dans un même manuel, par exemple Déclic, les fonctions du symbole $\rightarrow$ peuvent être *le calcul, le stockage ou l'initialisation.* Belin ajoute à ces usages l'égalité de 2 nombres pour expliquer le symbole $\rightarrow$: « *If C = 1 ; 0* $\rightarrow$ *A : Si C = 1, A = 0* ».

Dans ce manuel, deux types d'opérations à l'intérieur d'une boucle, sont induites par le symbole $\rightarrow$:

- La mise à jour : $Int(rand \times 2) + S \rightarrow S$; $N + 1 \rightarrow N$; $I + 1 \rightarrow I$
- L'initialisation : $F \rightarrow L2(A)$; $L1(A) + 1 \rightarrow L1(A)$, *etc.*

La mise à jour porte essentiellement sur la variable « compteur » et obéit au modèle canonique « s'initialise à zéro et s'incrémente de 1 ».

Les tests d'arrêt ne sont pas présentés : les programmes s'arrêtent quand le résultat est obtenu.

---

[11] Les ouvrages analysés sont : *Mathématique* Collection Bréal, *Déclic Maths* Collection Hachette et *Math* Collection Belin. Nous les nommons dans la suite de cet article Bréal, Déclic et Belin.

b. Analyse des exercices

L'analyse des exercices des manuels[12] nous a permis de repérer 8 types de tâches portant sur les algorithmes, les trois derniers types concernant plus particulièrement la notion de variable :

T1 : Ecrire un algorithme pour un problème donné.
T2 : Exécuter un programme donné (principalement sur la calculatrice et « à la main », l'ordinateur pouvant être évoqué).
T3 : Trouver le résultat d'un algorithme donné pour différentes instances d'un type de problèmes.
T4 : Commenter un programme ou un algorithme.
T5 : Traduire un algorithme donné dans un langage de programmation.
T6 : Contrôler des valeurs particulières d'une variable.
T7 : Mettre à jour une variable.
T8 : Modifier le test d'arrêt d'une boucle.

Type de tâches	T1		T2		T3		T4		T5		T6		T7		T8	
Manuels	2°	1S	2°	1S	2°	1S	2°	1S	2°	1S	2°	1S	2°	1S	2°	1S
Belin	0	1	2	5	22	4	0	0	0	0	0	1	0	1	0	1
Déclic	2	0	2	5	15	0	1	0	0	0	0	1	0	1	0	0
Bréal	0	0	4	1	5	1	5	2	3	1	0	0	0	0	0	0

Tableau 1. Nombre d'exercices de type Ti selon les manuels et selon les niveaux

Comme le montre le tableau 1, ces tâches existent souvent à 1 ou 2 exemplaires dans certains manuels. Tout particulièrement, les exercices sur la notion de variable informatique sont quasi inexistants.

Seuls les exercices de type 3 (en seconde) et dans une moindre mesure du type 2 sont bien représentés, tout particulièrement au niveau seconde dans Belin (22) et Déclic (15). Les types T4 (7) et T5 (4) apparaissent comme spécifiques au manuel Bréal : ce manuel offre ainsi la possibilité de travailler sur des algorithmes donnés par les questions posées : « *que fait l'algorithme ?* » ou « *traduire l'algorithme dans un langage de calculatrice et exécuter l'algorithme* ».

En conclusion

Si on considère les manuels comme un premier modèle des pratiques enseignantes, notre analyse permet de formuler certaines clauses du contrat didactique relatif à la notion d'algorithme au niveau seconde et première S :

L'écriture d'un programme n'est pas à la charge de l'élève (absence de savoir sur la construction des algorithmes, rareté des exercices T1). Par contre un élève de seconde ou de 1ère S doit savoir exécuter un programme donné sur sa calculatrice ou l'expliquer (importance des exercices T2 et T3). Par conséquent, dans les manuels étudiés, *l'écriture d'une boucle ou d'un test d'arrêt n'est pas à la charge de l'élève.*

[12] L'analyse détaillée en terme de praxéologies (Chevallard 1991) se trouve dans Nguyen 2002.

Dans le domaine de l'analyse (2nde et 1ère), la notion d'algorithme (dans les 3 manuels examinés) est associée à deux tâches: décomposer une fonction donnée en opérations élémentaires et calculer les valeurs numériques d'une fonction. Or, pour calculer les valeurs numériques d'une fonction, les trois manuels font appel aux tableurs (calculatrice et logiciel). Cet usage décharge l'élève de la répétition des calculs et masque l'intervalle sur lequel on fait ce calcul, et qui, cependant, conditionne son début et sa fin. Cet usage ne rend-il pas difficile la formulation de la répétition des calculs par l'écriture d'une boucle ? Ne s'oppose-t-il pas à la nécessité de la *formulation de l'initialisation et de la condition d'arrêt de cette action répétée* ?

La rareté des explications et l'inexistence des exercices, concernant la notion de variable dans un programme informatique, ne peuvent être sans conséquence sur la conception de cette notion. De quels moyens dispose l'élève pour différencier la notion de variable informatique (qui, rappelons-le, désigne une case de la mémoire) et la notion pré-construite (et déjà présente au collège) de variable en mathématique ? Si on se réfère à Samurçay, la prise en compte de la répétition pourrait permettre cette différenciation :

C'est surtout dans cette dernière activité que le concept de variable informatique apparaît différent de celui de variable mathématique (symbole représentant un élément non spécifié ou inconnu d'un ensemble), et que l'opération d'affectation (relation asymétrique) diffère de la relation d'égalité (évidemment symétrique). (Samurçay 1985, p. 145)

III. La notion de boucle comme formulation de la répétition d'une action chez des élèves de 2nde et de 1èreS

Au niveau de deux classes du lycée (2nde et 1ère S), nous avons conçu et mis en place une situation expérimentale dont l'enjeu est la formulation de la répétition d'une action (écriture d'une boucle) dans un programme. Après la présentation des choix cruciaux de cette situation (nommée CALCULATOR), nous analysons globalement les réponses des élèves (2nde et 1ère S) puis le processus interactif d'un binôme de 2nde (L et J) qui aboutit à l'écriture d'une boucle.

III.1. Le texte de l'expérimentation en 2nde et 1ère S

Exercice[13] (Sans calculatrice)

Soit la fonction : $x \rightarrow f(x) = x^2 + 1$.

L'élève TOURNESOL veut calculer toutes les images des réels x appartenant à l'intervalle [-2, 3] avec un pas donné. Comme le tableur de sa calculatrice est en panne, il doit faire le calcul lui-même.

[13] Cet exercice est inspiré de l'exercice 24 (page 126) du chapitre *Fonction numérique* dans le manuel « *Mathématiques Seconde* » (Edition Bréal 2000).

a. Il décide de dresser un tableau de valeurs en utilisant un pas de 0,2 :

x	- 2	- 1,8	- 1,6	.
f(x)	5	4,24	3,56	.

Si l'on dit que - 2 est la première valeur de x ; - 1,8 est la deuxième valeur de x et ainsi de suite, pouvez-vous aider TOURNESOL en calculant les valeurs de x et de f(x) pour :

La sixième valeur de x :

La onzième valeur de x :

La treizième valeur de x :

b. TOURNESOL se propose maintenant de réduire le pas en le prenant égal à 0,03.

Il commence les calculs :

$x = -2$; puis $f(-2) = (-2)^2 + 1 = 5$

$x = -1,97$; puis $f(-1,97) = (-1,97)^2 + 1 = 3,8809 + 1 = 4,8809$; ...

TOURNESOL s'aperçoit qu'il y a beaucoup de calculs à faire. Il a l'idée de s'adresser à son camarade CALCULATOR pour faire les calculs.

TOURNESOL *décide d'écrire un message à* CALCULATOR *contenant un petit programme de calcul à répéter.*

Ce programme doit être le plus court possible.

Ce programme doit permettre, avec le pas égal à 0,03 de fournir en retour à TOURNESOL toutes les images (et seulement ces images) des réels x appartenant à l'intervalle [-2, 3].

Quel message à *CALCULATOR* écririez-vous à la place de TOURNESOL ?

III.2. Choix et raisons de ces choix

Le problème général suivant génère les parties a et b de l'exercice :

Soit f une fonction réelle. Calculer f (a + k.p) avec un pas donné p et pour a + k.

$p \in [a, b]$, k entier.

Dans l'exercice proposé, nous avons choisi :

• f : x -> $f(x) = x^2 + 1$: fonction usuelle à partir de la seconde, associée à un schéma de calcul simple des images.

• l'intervalle [-2 ; 3] :

- de bornes entières « petites », l'une négative, l'autre positive en accord avec ce qui se fait habituellement ;

- non symétrique par rapport à 0 : dans le cas contraire, la parité de la fonction f pourrait permettre de réduire le nombre de calcul à exécuter et donc la répétition.

• l'indisponibilité de la calculatrice pour bloquer la solution institutionnelle à ce type de problème qu'est le tableur-calculatrice.

• de ne pas donner l'écriture $x_k = a + k.p$ pour ne pas fournir la solution au problème de la formulation.

• de demander, en partie a, des valeurs non successives de x_n pour permettre le fonctionnement de deux modèle d'action au moins (ils seront détaillés dans l'analyse qui suit).

Le pas p est une variable didactique du problème : les valeurs possibles déterminent des situations différentes.

i) Situation d'action

Cette situation a pour but d'assurer la compréhension du problème. L'observable de cette compréhensionest l'action (individuelle) de calcul des élèves.

• Si p entier ($0 < p \leq 5$), il est possible de produire une liste exhaustive de toutes les valeurs de x (stratégie « liste »).

• $p = 0,2$ (partie a)

Dans ce cas, le nombre des valeurs x à calculer (26) rend encore possible la stratégie « liste », mais cette stratégie devient plus coûteuse. Le coût de la stratégie « liste » peut permettre l'émergence de stratégies plus économiques, prenant en compte la répétition. C'est une situation d'action : on peut répondre à la question sans avoir à expliciter la répétition. Un modèle implicite de cette répétition peut être :

$x_{k+1} = x_k + p$ modèle de récurrence M(R)
$x_{k+1} = a + k.p$ modèle fonctionnel M(F)

On peut donc prévoir *a priori* deux stratégies de calcul possibles associées à ces deux modèles.

M(R). Calculer les valeurs de x en ajoutant la valeur p à la valeur initiale $x = a$ (modèle $x_n = x_{n-1} + p$), calculer f(a) puis successivement $x_n^2 + 1$ jusqu'à $x_{26} = b$.

M(F). Calculer les valeurs de x suivant la formule $x_k = a + (k-1).p$.

Calculer $x_k^2 + 1$.

L'observable de M(R) peut être un tableau de valeurs complété (à partir de celui qui est donné) où l'on passe d'une colonne à l'autre en ajoutant successivement p pour obtenir x_6, x_{11}, et x_{13}. Celui de M(F) est la présence d'un schéma de calcul donnant indépendamment du tableau x_6, x_{11}, et x_{13} et interprétable par $x_k = a + (k-1).p$.

ii) Situation 2

• $p = 0,03$

Les conséquences du choix de cette valeur pour p sont les suivantes :

le nombre de calculs répétitifs (167) bloque la stratégie « liste », mais aussi la *construction d'un tableau.*

La division $(a - b)/p$ n'est pas un nombre entier : aucun x_k ne prendra la valeur 3 ce *qui complexifie la formulation de l'arrêt des calculs.*

• Les autres choix de la partie b

Les élèves travaillent en binôme. Les interactions verbales et écrites peuvent favoriser l'évolution des procédures de résolution du problème et permettent l'observation de cette évolution.

La question posée dans cette partie rompt certaines clauses du contrat didactique de la notion d'algorithme, puisque dans cette partie l'élève doit prendre en charge l'écriture d'un programme à l'opérateur fictif CALCULATOR. Cet opérateur est capable de faire les calculs, aussi nombreux soient-ils. Cette situation est une situation de communication *évoquée*[14] avec dissymétrie entre l'émetteur et le récepteur, ce qui lui donne certaines caractéristiques d'une situation adidactique de formulation. L'écriture d'un message rend nécessaire l'explicitation des modèles implicites de l'action (Brousseau 1998).

Cependant l'absence de rétroactions pour la validation laisse prévoir que des interventions seront incontournables pour la compréhension de la tâche. Nous y reviendrons. Mais auparavant, quelles peuvent être les réponses possibles dans cette situation ?

• *En ce qui concerne la répétition*

- Absence de formulation de la répétition, noté M0
- Formulation de la répétition, noté M.

Autres notations :

sans condition d'arrêt noté M^-; avec condition d'arrêt noté M^+

$M^-(R)$ (processus récurrent). Présence de la formule $x_{n+1} = x_n + p$ (1) sans formulation d'une condition d'arrêt / $M^+(R)$, idem avec formulation d'une condition d'arrêt.

$M^-(F)$ (processus fonctionnel). Présence de la formule $x_{k+1} = a + k.p$ (2) / $M^+(F)$, idem avec formulation d'une condition d'arrêt

Nous rattachons *a priori* les réponses possibles de formulation à quatre types de boucles :

$$\mathbf{M^+(R).\ x_{k+1} = x_k + p}$$

Programme 1 (B1)	**Programme 2 (B2)**	**Programme 3 (B3)**	**Programme 4 (B4)**
Début	*10. Début*	*Début*	*Début*
Entrer f, a, b, p	*20. Entrer f, a, b, p*	*Entrer f, a, b, p*	*Entrer f, a, b, p*
a → x	*30. a → x*	*a → x*	*a → x*
Tant que x ≤ b faire	*40. Calculer f(x)*	*E[(b – a)/p] → m*	*Répéter*
Calculer f(x)	*50. x + p → x*	*Pour n = 0 jusqu'à m faire*	*Calculer f(x)*
x + p → x	*60. Si x ≤ b alors aller à 40*	*Calculer f(x)*	*x + p → x*
Fin tant que	*70. Fin*	*x + p → x*	*Jusqu'à x > b*
Fin		*Fin*	*Fin*

Tableau 2. Quatre boucles fondées sur $x_{n+1} = x_n + p$

[14] Cette classe de situation a déjà été utilisée par Samurcay et Rouchier (1985).

$M^+(F).\ x_k = a + n.k$

Programme 1' (B1)	**Programme 2' (B2)**	**Programme 3' (B3)**	**Programme 4' (B4)**
Début	*10. Début*	*Début*	*Début*
Entrer f, a, b, p	*20. Entrer f, a, b, p*	*Entrer f, a, b, p*	*Entrer f, a, b, p*
0 → n	*30. 0 → n*	*0 → n*	*0 → n*
Tant que np+a ≤ b faire	*40. Calculer f(x)*	*E[(b – a)/p] → m*	*Répéter*
Calculer f(np+a)	*50. n + 1 → n*	*Pour n = 0 jusqu'à m faire*	*Calculer f(np+a)*
n + 1 → n	*60. Si np+a ≤ b alors*	*Calculer f(np+a)*	*n + 1 → n*
Fin tant que	*aller à 40*	*Fin*	*Jusqu'à np+a > b*
Fin	*70. Fin*		*Fin*

Tableau 3. Quatre boucles fondées sur $x_{k+1} = a + n.k$

La condition d'arrêt peut être placée avant l'action (B1, B1', B3, B3, B3') ou après (B2, B2', B4, B4').

La condition d'arrêt peut être attachée :

i) à un compteur et donc au nombre de calculs à répéter (B3, B3'),

ii) à l'appartenance de x_k à l'intervalle [a ; b][15] (autres boucles)

III.3. Résultats de l'expérimentation

L'expérimentation a été réalisée en mai 2002 dans deux lycées de Grenoble[16] : en classe de Seconde (33 élèves) du lycée Grésivaudan et en classe de Première, option technique (25 élèves) du lycée Vaucanson (programme 1ère S).

a. Situation 1 (question a)

Le tableau 4 résume les réponses des élèves dans les deux classes observées.

Procédures / Classes	Résultat seulement	Prolongement tableau M(R)	Formule $x_n = a + np$ M(F)	Total
2nde	13	18	2	33
1ère	13	6	6	25
Total	26	24	8	58

Tableau 4. Répartition des réponses

Ceux qui écrivent seulement le résultat n'ont laissé aucune trace de leur procédure : nous ne pouvons donc pas les interpréter. C'est la réponse dominante en 1ère (13 sur 25)

Par contre en 2nde, les procédures M(R) dominent (18 sur 33).

La proportion d'élèves ayant utilisé une formule arithmétique M(F) en 1ère est plus important qu'en 2nde. Ce fait peut s'expliquer par l'enseignement des suites en 1ère: les

---

[15] Dans le cas particulier où f(b) est le maximum (resp. minimum) de la fonction f sur [a, b], la condition d'arrêt peut aussi être attachée à l'image de l'intervalle [a ; b] par la fonction. C'est le cas de f : x -> f(x) = x^2+1, pour laquelle f(3) = 10 est le maximum de la fonction f sur [-2 ; 3].

[16] Nous remercions vivement les enseignants qui nous ont accueillis : Françoise Bayarri, Marion Dieudonné et Olivier Tavan.

six élèves de 1ère font tous la même erreur, ils remplacent n par 6 pour calculer la 6e valeur de x, les indices n d'une suite débutant habituellement à n = 0.

Aucun élève, même en 1ère, n'éprouve le besoin d'expliciter une formule du type « modèle récurrent », sur laquelle se fonde implicitement le prolongement du tableau.

Avant la situation 2, les enseignants dans les deux classes ont donné la correction suivante au tableau, en évitant d'induire une formule quelconque :

sixième valeur de x = -1 f(-1) = 2,
onzième valeur de x = 0 f(-1) = 1
treizième valeur de x = 0,4 f(-1) = 1,16

b. Situation 2 (question b). Résultats globaux

Après une première lecture de l'énoncé, les élèves se sont interrogés sur ce que CALCULATOR pouvait faire. L'explication de l'enseignant a été la suivante :

c'est quelqu'un qui peut faire plein de calculs, très vite et très juste mais il faut lui dire ce qu'il doit faire.

Le tableau 5 donne le type de messages produits par les élèves.

Procédures / Classes	Pas de réponse	Mo	M^-	M^+	Total
2nde	4	3	3	6	16
1ère	0	0	7	5	12
Total	4	3	10	11	28

Tableau 5. Répartition des messages selon la formulation de la répétition

En 1ère tous les messages (12 binômes) prennent en compte la répétition, alors qu'en 2nde, seulement 9 binômes (sur 16) le font. On peut penser que l'explicitation de l'un des éléments de l'invariant de boucle (M^- ou M^+), en l'occurrence les formules $x_{k+1} = x_k + p$ et $x_{k+1} = a + k.p$, est facilitée par les connaissances sur la notion de suites.

Par contre, la notion de suite ne semble pas favoriser la prise en compte de la condition d'arrêt puisque la proportion d'élèves qui l'écrivent est pratiquement le même en 2nde et en 1ère.

De plus, parmi les 21 binômes écrivant des formules (M^- ou M^+), 13 n'écrivent pas l'autre invariant de la boucle $f(x) = x^2 + 1$: ce calcul est totalement confié à l'exécuteur.

Examinons le contenu des messages

- En ce qui concerne le modèle explicite de la répétition

Formulation répétition / Classes	M(F) $x_{n+1} = a + np$	M(R) $x_{n+1} = x_n + p$	Total
2nde	5	4	9
1ère	9	3	12
Total	14	7	21

Tableau 6. Répartition des messages selon le type de formulation de la répétition

Le nombre modeste de binômes (même en 1ère) formulant la récurrence M(R), par rapport à ceux qui écrivent une suite arithmétique M(F) (7 contre 14) n'est-il pas l'indice du passage du remplissage d'un tableau à l'écriture du modèle mathématique sous-jacent à cette action ? Notons qu'en programmation, on recourt fréquemment aux suites récurrentes : d'une part, elles offrent l'intérêt d'économiser de la place dans la mémoire, d'autre part elles établissent un lien étroit entre la formule $u_{n+1} = f(u_n)$ et l'affectation $f(u) \rightarrow u$.

- Nature des boucles et de la condition d'arrêt

Comme nous l'avons déjà dit, peu de binômes (11 sur 28) écrivent une condition d'arrêt : cette écriture repose sur la prise en compte d'un intervalle réel auquel appartient soit la variable indépendante x, soit la variable dépendante f(x). Or le tableau de valeurs, et *a fortiori* celui des tableurs, prend en charge ce travail.

Le tableau 7 permet de comparer la position de la condition d'arrêt dans les formulations :

Condition / Classes	Avant B1, B3	Après B2, B4	Total
2nde	2	4	6
1ère	1	4	5
Total	3	8	11

Tableau 7. Répartition des messages selon la place de la condition d'arrêt

Le schéma action/test (condition d'arrêt après) est majoritaire par rapport au schéma test/action (condition d'arrêt avant) : le second schéma nécessite d'anticiper, avant l'action, des effets de la succession des actions.

Ce résultat conforte celui d'études précédentes :

La boucle *tant que ... faire* est plus difficile à gérer que la boucle *répéter ... jusqu'à*. En effet, la boucle *tant que ... faire* nécessite l'anticipation de la condition d'arrêt de la boucle avant la construction des instructions du corps de la boucle. Cette anticipation ne correspond pas aux usages spontanés d'une répétition. (Rouchier et Samurçay 1984 citée par Mejias 1985, pp 17-18).

Les débutants rencontrent de très nombreuses difficultés dans la manipulation de la boucle *pour* n = ... *jusqu'à ... faire*.(Soloway 1982 cité par Mejias 1985).

- Variables

Variable / Classe	n	x	Total
2nde	3	3	6
1ère	3	2	5
Total	6	5	11

Tableau 8. Répartition des élèves selon la présence de x ou de n dans la boucle

L'écriture d'une condition d'arrêt « compteur » et celle de la formule $x_{n+1}=a+np$ vont de pair ... à condition de formuler n ! Ainsi deux binômes (un de chaque niveau) formulent un calcul dont le modèle implicite est bien $x_{n+1}=a+np$: en l'absence de formulation de n, ils écrivent une condition d'arrêt sur x ou f(x). Nous présenterons en détail ci-après l'exemple du travail de formulation de l'un de ces binômes, J et L (2^{nde}).

Sur les 6 binômes écrivant un compteur « n =... », deux seulement (un en 2^{nde} et un en $1^{ère}$) calculent correctement la valeur de n (166), les autres font un calcul erroné, un binôme écrivant même n = 500/3. L'incrémentation de n, variable - compteur, « institutionnalisée » par les manuels, n'est entreprise par aucun binôme. Le passage de l'incrémentation de cette variable - compteur à l'affectation n → n + 1 devient alors impossible.

Comme nous l'avons annoncé dans l'analyse *a priori*, l'initialisation par affectation et la mise à jour des variables dans la boucle, sont des tâches difficiles. Un seul binôme ($1^{ère}$) y parvient, mais sans condition d'arrêt :

$x \rightarrow -2$; Repeat $x = 3$; $x + 0{,}03 \rightarrow x$; Displ $x^2 + 1$; Pause ; End.

c. Les difficultés de la formulation dans l'écriture des messages (Situation 2, question a)

Nous allons nous attarder sur le processus d'écriture de messages à CALCULATOR de deux élèves de seconde : J et L. Pour cela nous présentons :

- dans une première colonne de larges extraits du protocole d'observation de ce binôme (reconstruit à partir des notes de l'observateur[17] et de l'enregistrement audio) ;
- dans une deuxième colonne des commentaires sur les interactions entre les deux élèves, le contenu des messages successifs et le rôle joué par les interventions de l'observateur.

Nous serons particulièrement attentifs aux difficultés rencontrées dans la formulation de la répétition (en référence aux modèles M(F) et M(R)) et aux conditions de l'évolution de cette formulation dans le processus d'écriture d'un message à CALCULATOR.

[17] L'observateur est Laetitia Ravel.

- **CALCULATOR comme élève fictif performant**

Extrait de protocole	*Commentaires*
78. J : On dit qu'on dresse un tableau de valeurs en utilisant un pas de 0,03. On dit que la 1ère valeur c'est ça. On dit que ça pour toutes les valeurs. 79. L : Toutes les images de x appartenant à l'intervalle [-2 ;3] … en utilisant un pas de 0,03. Et amuse-toi à faire les calculs.	Tout d'abord, J et R écrivent un tableau de valeurs que CALCULATOR n'a plus qu'à « s'amuser » à remplir. L'ironie du verbe montre que J et L savent qu'il y a beaucoup de calculs : ils produisent les 4 premières valeurs.

Message 1

Soit x ∈ [-2, 3]					
CALCULATOR, on te propose de trouver toutes les images de x ∈ [-2, 3] en utilisant un pas de 0,03 sachant que f(x)-> x^2+1					
x	-2	-1,97	-1,94	-1,91	 3
f(x)					10

La procédure est celle qu'ils ont utilisée pour le problème initial (pas 0,2).

- **CALCULATOR comme robot qui ne sait que calculer avec les formules qu'on lui donne**

94. J : Je sais pas mais parce que c'est le temps, on nous donne 25 minutes.
95. L : Oui, ça paraît à la fois trop simple mais …
96. J : Parce que si, si on nous donnait 5 minutes, on penserait plus que ce soit ça mais comme c'est 25 minutes… […]
100. J. Non, c'est pas ça […]
103. L : On a trop de temps pour faire ça […]
107. Ob (Observatrice) : C'est le problème du temps hein ?
108. J : Oui
109. L. Trop de temps pour faire ça.
110. Ob : Ben, en fait, CALCULATOR, est-ce que tu crois qu'il comprend le mot « pas » ?
111. J : Voilà !
112. Ob : Tu imagines qu'il est bête et méchant, il ne fait que calculer.
113. L : Alors, on a mis un exemple après … On a mis un exemple. [il montre le tableau de valeurs avec x : - 2 ; -1,97 ; - 1,94 ; - 1,91 et 3]
114. Ob : Oui, mais est-ce qu'il est capable de compléter la suite ? Tu sais lui, il calcule avec les formules que tu lui donnes.
115. L : On prend … si on prend x à chaque fois x + 0,03. Et chaque fois on lui dit faut faire x + 0,03 et … à chaque x qu'il trouve il doit faire + 0,03.
116. J : Oui, c'est plus intéressant.

Par effet de contrat (le temps donné par l'enseignant doit être utile au travail des élèves), J et L doutent de leur solution.

L'intervention de l'observatrice (à la demande des élèves) sur « ce que comprend », « ce que peut faire » CALCULATOR permet un retour sur le premier message.

L et J vont tenter de formuler les implicites de calcul pris en charge par le tableau.

Message 2

Il faut trouver toutes les images de x ∈ [-2, 3] à l'aide du procédé suivant : il doit calculer les images de x tels que f(x)=x^2+1. En partant du x de départ, il doit additionner 0,03 à ce x, il continue d'additionner 0,03 au nouveau résultat de la 1ère opération

Dans ce message, L et J formulent une boucle M(F) sans condition d'arrêt et *sans différencier la variable x des valeurs successives.*
L'initialisation de x est réglée par « En partant du x du départ… ».

- **J et L découvrent qu'ils « n'arriveront jamais à 3 » avec un pas de 0,03.**

127. J : (à l'observateur) Eh, oui, il y une petite erreur madame [...] dans leur calcul. Parce que si y'a des intervalles de 0,03 ils n'arriveront jamais à 3 ... ? C'est un intervalle ... enfin comme ça fait 5. 130. L : Oui, oui c'est clair 131. J : Si tu fais 5 par 0,03 c'est pas bon. Déjà il arrivera pas à trouver un chiffre rond. 132. L : Il arrive pas directement à 3 133. J : Oui, il n'arrive pas à 3 134. L : Mais lui tu sais il s'en fout. Tu lui demandes de trouver juste les images avec un pas comme ça. 135. J : Oui mais un pas de 0,03, mais comme l'intervalle c'est –2 à 3 il arrive pas à 3, donc c'est pas bon.	Le choix du problème – le résultat de la division de la mesure de l'intervalle par le pas est non entier - est contesté par J : c'est une rupture du contrat de la discrétisation opérée habituellement dans un tableau de valeurs. L (134) accepte ce nouveau contrat pour CALCULATOR.

- **La difficulté à formuler la succession des valeurs de x**

167. L : En partant du x du départ il additionne 0,03 à ce x 168. J : Mais, au départ, ça peut être ... ? 169. L : A mon avis, c'était – 2, au début. 170. J : Oui 171. L : Puis il continue d'additionner 0,03 au nouveau résultat ... du 1er x et ainsi de suite. // [18] [...] 174. J : Oui, on doit trouver un machin. [...] 178. L : On fait x plus ça, on trouve x' après on fait x' moins ça ... plus ça on trouve x'' et après on continue. 179. J : Oui 180. L : On a ,là ,on fait, f(x) puis f(x') // 181. J : (Il s'adresse à Ob.) : Madame --- 182. L : Ouais, ouais c'est ça ... t'es d'accord ? 183. J : f(x), f(x') non f(x).	J et L inventent une notation sur 3 termes pour marquer la succession des valeurs prises par la variable x : « x, x', x'' ». Ils écrivent la deuxième et la troisième valeur en fonction de la 1ere notée x (comme la variable) selon le modèle M(F), mais reviennent à M(R) pour x''' ! .Il n'y a pas d'initialisation de la variable. La 1ére valeur est notée x.

Message 3

x+0,03=x'	$f(x) = x^2+1$
x+0,06=x''	$f(x') = x'^2+1$
x''+0,03=x'''	$f(x'') = x''^2+1$

- **Une découverte : la formulation de M(F)**

186. L.: On peut faire ça, après ... ça fait ... plus 0,06. Ah, ouais, voilà, voilà. 187. J.: Exact, exact... Comme tu faisais x 188. L : Tu effaces pas, tu écris en dessous, il vaut mieux que tu laisses tout. [...] 190. L : En fait, c'est ça. 191. J : Tu as x plus 0,03 égale à ...	

[18] Le symbole // signifie *inaudible*.

192. L : On trouve, on trouve la 2e valeur.
193. J : Voilà, regarde ... Ouais mais là, il faudrait trouver un nombre là.
194. L : Attends. // alors... de quoi ? ---
195. J : C'est pas tranquille là. Oui, mais ... oui, mais ... le résultat que tu trouves après il faut le mettre sous le calcul. //
196. L : Quand tu envoies un message à CALCULATOR, il se débrouille avec les calculs.

Message 3

1ère valeur de x = -2
2ième valeur de x = -2 + 0,03
3ème valeur de x = -2 + 0,06
4ème valeur de x = -2 + 0,09

Les 4 premières valeurs de x sont écrites en fonction de la valeur initiale de x, soit –2, et du calcul effectif des valeurs à ajouter à cette valeur initiale (M(F)). Dans ce message, les valeurs de la variable et la variable sont séparées.

197. J.: OK, ouais mais il sait pas lire CALCULATOR. //

198. L : Comment on peut expliquer à chaque fois, on multiplie euh ... tu vois, comment expliquer ça ? ... Comment tu expliques ça là, comment dire ?
199. J : Mmm, ben justement c'est ça qui ...
200. L : Faut mettre ... sous forme d'opération ? Comment on le met ?
201. J : Parce que à la limite, si vraiment tu veux pas te casser le tête, tu fais vraiment le tableau mais bon, y'a beaucoup chiffres à mettre. Donc c'est à toi de ---- [...]
205. L : Comment on dit à chaque fois tu sais on fait deux fois 0,03, trois fois ...?
206. J : Oui, après qu'il passe 0,03 et 0,06 puis 0,0...
207. L : Ouais, attends, là ça fait quoi là, ça fait ...
208. J : Après il ... là, il additionne à chaque fois 0,03
209. L : Donc en fait là, là c'est - 2 plus une fois 0,03, deux fois 0,03, trois fois ... c'est pour chaque position. /
210. L : Voilà. [...]
213. J : Chaque fois, y c'est un, deux, trois ... Oui regarde, en multipliant par 0,03 par y et on dit y c'est ...
214. L : Ah ça change tout le temps y ? !
215. J : C'est un ..., non, on dit c'est un entier de ... qui va de ... de 1 à ... 100 ... 100 et quelque.
216. L : Regarde J ce que j'ai mis. // Et après on lui dit et ainsi de suite.

Le rôle du tableau est clairement explicité par J (201) !

Le retour réflexif, permis par le message 3, associé à la prise en compte des capacités limitées de CALCULATOR conduit à la formulation du lien entre la « position » de la valeur et l'entier qui multiplie le pas (209).

J tente de désigner l'entier (qui multiplie le pas) par y (213) et une condition d'arrêt -compteur (215) : « c'est un entier de 100 et quelque ».

Message 4

Trouver les images de x, x ∈ [-2, 3]
pour la 2ième valeur il doit additionner 0,03 au x de départ
pour la 3ème valeur il doit additionner 2(0,03) au x de départ
pour la 4ème valeur il doit additionner 3(0,03) au x de départ et ainsi de suite

Si ce message explicite bien le lien entre la « position » de la valeur et l'entier qui multiplie le pas selon le modèle M(F), il manque encore l'initialisation de la variable et la condition d'arrêt de la boucle.

- **Le problème de la condition d'arrêt est posé : un débat fructueux entre J et L**

217. J : Fais voir. // Tu veux chercher quoi ? 218. L : Au x du départ. Non mais là, là je me suis planté. Ca, c'est n'importe quoi. 219. J. : Ainsi de suite jusqu'à ce que f(x) soit … qui soit inférieur … oui, inférieur ou égal à, supérieur ou égal à 10. On va le faire au propre. 220. L : Oui, - 2… bon, on dit c'est ça là. 221. J : Comme tu dis, ainsi de suite jusqu'à ce que … 222. L : Mais c'est pas possible 223. J : Jusqu'à ce que f(x) soit supérieur à 10, car il sera plus dans l'intervalle de 2 à 3, donc …parce que comme il peut pas tomber sur 3 et bah, ce que … 224. L : Ce que f(x) égal 10 225. J : Non, supérieur ou égale à 10, non supérieur à … 226. L : Oui, là c'est égal car si on prend l'image de 3. Ca fera, ça fera 10. 227. J : Ouais, supérieur à 10. 228. L : Non égal. 229. J : Non, parce qu'il y a pas le droit d'être égal 230. L : Oui, il a le droit de … 231. J : Mais il ne sera pas égal à 10 232 . L : Mais l'image … 233. J.: Non elle pourra pas être égale à 10, parce que … 234. L : Oui, ça fera jamais le bon intervalle 235. J : Oui, ça trouvera jamais un nombre entier. 236. L : Non jusque x … 237. J : Non jusqu'à ce que f(x) il soit supérieur … supérieur à 10. 238. L : Sinon on dit jusqu'à ce que x soit le plus proche de 3, vu qu'il sera jamais égal à 3 ? 239 J : Non, parce que … non, jusqu'à ce que f(x) soit supérieur à … […] 242. L : Non, ça pourra pas être supérieur à 10 parce qu'après si jamais c'est supérieur à 10… 243. J : Ben justement ça rentre plus dans l'intervalle. Donc, voilà … Ainsi de suite […] 247. .J : Mais il faut que f(x) soit inférieur à 10, inférieur ou égale à 10. 248. L. : égale à 10. Parce que si on dit inférieur, il ça va s'arrêter tout de suite après 249. J : Ouais, non faut être supérieur ou égale à 10. Parce que si il est supérieur … après … 250. L : Mais non, parce que il s'arrête quand ça sera supérieur à 10, il suffit qu'il prenne l'image de … je sais pas combien ? Ah, mais non, c'est pas pareil si on dit ça et ça. […] 257 .L.: Non, mais quand il sera supérieur, bah, l'image elle sera supérieure donc ça marchera pas 258. J : Mais non, ben justement il s'arrêtera. 259. L : Ouais, mais non … jusqu'à ce que x soit le plus	Deux propositions en débat : - *Celle de J* P1 (223) : « Jusqu'à ce que f(x) soit supérieur à 10 » Arguments de J (223) : « il ne sera plus dans [2 ;3] et il ne peut pas tomber sur 3 » - *Celle de L* P2 (224) : « (jusqu'à) ce que f(x) égal 10 » Argument de L (226) : « f(3)=10 » Deux réfutations par J de la proposition P2 - (229) : « il n'a pas le droit d'être égal … à 10. » - (235) « ça trouvera jamais un nombre entier » - L raisonne sur la variable x et ses valeurs et formule une autre proposition : P3 (238) : « jusqu'à ce que x soit le plus proche de 3, vu qu'il sera jamais égal à 3 » - J maintient sa proposition P1 L est à court d'argument - J formule une nouvelle proposition P4 (247) : « (x) inférieur ou égale à 10 » - Réfutation immédiate par L de P4 qu'il utilise comme preuve pour P2 (248) : « égale à 10, parce que si on dit inférieur, il va s'arrêter tout de suite » - J revient alors à P1 (249) - L continue à raisonner sur x (250) : « il suffit qu'il prenne l'image de … » En voulant argumenter contre P1 (250), L se convainc ! - (258) J exprime clairement l'arrêt

proche de 3.
[...]
272. J : Hein alors, mais ça veut rien dire pour la 2e valeur
273. L : Ben pour trouver la 2ère valeur.
274. J. : Mais dans ce cas, c'est quoi la 1ère valeur ?
275. L : Mais on a dit que c'était - 2. Ben, tu mets la 1ère valeur de x : – 2.
276. J : Au quoi tu as mis ? Au x au départ ?
277. I : Oui.

- (259) L revient à P3, puis se rallie à P1

Le problème de l'initialisation de x est posé par J (274).

Ci-après le message final du binôme J et L

Trouver les images de x, x ∈ [-2, 3] 1ère valeur de x = -2 pour la 2ième valeur il doit additionner 0,03 au x de départ pour la 3ème valeur il doit additionner 2(0,03) au x de départ pour la 4ème valeur il doit additionner 3(0,03) au x de départ et ainsi de suite jusqu'à f(x) ≥ 10, sachant que f(x) = x^2+1

Ce message final est l'aboutissement d'un long processus au cours duquel les élèves différencient la désignation de la variable de celle de ses valeurs successives et explicitent :
- la relation entre la « position » de la valeur et l'entier qui multiplie le pas (M(F)) ;
- la condition d'arrêt (selon le schéma action/test de la boucle B4) ;
- l'initialisation de la variable.

Comme nous l'avions anticipé dans l'analyse *a priori* de la situation CALCULATOR, les conditions initiales se sont avérées insuffisantes pour provoquer ce processus : l'observateur a du intervenir pour modifier la représentation des élèves du récepteur du message. CALCULATOR, d'élève fictif, est devenu un robot fictif capable seulement de faire les calculs qu'on lui indique.

Les deux élèves abandonnent alors le tableau de valeurs, solution première et institutionnelle : leurs interactions les conduisent à prendre conscience des implicites du tableau de valeurs[19]) à savoir la succession et le calcul des valeurs de la variable, l'intervalle de définition et conjointement l'intervalle image.

Un bilan de l'expérimentation

L'expérimentation a montré les difficultés des élèves :
- dans la formulation de l'invariant des calculs, formule de récurrence M(R) ou fonctionnelle M(F), qui demande de se libérer du tableau (réponse institutionnelle au problème du calcul des valeurs d'une fonction sur un intervalle) ;
- dans l'écriture d'instructions à CALCULATOR pour initier et arrêter un calcul effectif.

[19] Nous pensons actuellement que ces implicites peuvent être encore plus profondément cachés par l'usage du tableur-calculatrice. Les tableurs-logiciels permettent-ils une explicitation de certains de ces implicites ?

Aussi bien la formulation d'une condition d'arrêt que l'initialisation des variables sont problématiques (l'initialisation est envisagée par le seul binôme J et L).

IV. La perception des enseignants des difficultés des élèves concernant les notions de boucle et de variable

- Nous avons interrogé des enseignants sur la situation CALCULATOR.

Proposeriez-vous cet exercice à vos élèves (soit tel quel, soit en les modifiant) ? Quelles modifications éventuelles y apporteriez-vous et dans quel but ? Quelles difficultés pourraient rencontrer les élèves ?

Quatre enseignants (sur les vingt ayant répondu) mentionnent les difficultés liées au caractère inhabituel d'un un tel exercice :

Ils n'ont aucun repère, aucun modèle auquel ils se rattachent. Comment gèreront-ils cette grande liberté ?

- L'analyse des manuels a montré que l'écriture d'un programme n'est jamais à la charge de l'élève, mais toujours à celle de l'enseignant. Le nombre faible d'enseignants (3 enseignants) qui mentionnent l'écriture du message comme une difficulté semblent indiquer une certaine ignorance vis-à-vis de ce travail de formulation et peut-être même une absence de telles activités dans leur enseignement (en accord avec ce que montrent les manuels examinés).

- Dans le questionnaire, nous présentions aux enseignants les 4 boucles (B1, B2, B3, B4) du tableau 2 (par. III.2,) concernant le problème CALCULATOR et leur demandions :

Quel est pour vous le programme qui serait le plus adapté à l'enseignement des mathématiques ? Précisez pourquoi (écrivez au verso si besoin) :

Confrontons les choix des enseignants à la nature des boucles présentes dans les messages des élèves lors de l'expérimentation :

Boucles	B1	B2	B3	B4
Enseignants	10	10	0	4
Elèves	3	0	0	8

Tableau 9. Choix des boucles par les enseignants et boucles présentes dans les messages à CALCULATOR

Cette confrontation montre des similitudes. Les élèves comme les enseignants ignorent B3, les enseignants donnant une raison vraisemblable : « *la notion de partie entière est hors du programme* ».

Elle montre surtout de nombreuses oppositions.

Chez les élèves, la boucle B4 est largement majoritaire, alors que chez les enseignants elle n'est jamais choisie seule : les 4 enseignants la choisissent conjointement à B1. Comme nous l'avons déjà signalé, des études précédentes (Laborde et al. 1985, Rogalski 1988, Samurçay 1985) ont montré la difficulté du schéma test / action, associée à B1.

Les 5 enseignants ayant une formation en informatique choisissent B1, et on peut avancer que cette formation joue un rôle important dans ce choix. En effet, en informatique, la boucle B1 a supplanté la boucle B2 : le branchement *aller à* de la

boucle B2, présente dès les premiers langages de programmations, a pratiquement disparu des langages structurés d'aujourd'hui.

Une proposition assez radicale consiste, au début des années 70, au rejet de l'usage de branchement de type *« aller à »* dans la conception des programmes (Dijkstra 1968), l'expression de la boucle prenant la forme standard [...] suivante : *tant que c faire a.* (Laborde et *al.* 1985, p 225)

Le choix de B1 ou B4, par les enseignants n'est jamais justifié par la disponibilité de la formule x = a + np. Or, pour beaucoup d'élèves c'est l'écriture d'une telle formule qui marque leur prise de conscience d'un invariant dans la répétition.

La boucle B2[20], absente dans les productions des élèves, apparaît comme la référence chez les enseignants de mathématiques. Les arguments des enseignants pour la préférer aux autres sont qu'elle est « facile à comprendre » (3 enseignants), plus « visuelle » par rapport aux autres (2) et proche de « l'idée naïve du pas » chez l'élève (2).

Conclusion

La rupture de contrat opérée par la situation CALCULATOR a permis aux élèves, au travers de l'écriture d'une boucle (en référence au style impératif de la programmation) un retour réflexif sur certaines notions mathématiques : variable et valeurs d'une variable, succession de valeurs d'une variable et suite, intervalle de définition d'une fonction et intervalle image.

Cependant, l'expérimentation a montré les limites de cette situation, qui ne fait pas rentrer *naturellement* les élèves dans une problématique de la programmation. En effet CALCULATOR n'est pas une *machine* qui fait effectivement les calculs demandés, valide ou invalide un message codifié. Des interventions (de l'enseignant ou des observateurs) ont été nécessaires pour que CALCULATOR ne soit pas assimilé à un élève (comme le suggérait d'ailleurs l'énoncé).

Comment transformer cette situation pour qu'elle intègre des conditions permettant de donner du sens à l'activité de programmation des algorithmes dans la résolution d'un problème (*a priori* décidable)[21] ? L'une de ces conditions est que CALCULATOR devienne (pour les élèves) un *dispositif d'exécution différée* :

L'explicitation des procédures par le sujet est rendue nécessaire par le fait que l'exécution de la procédure se fait et doit se faire dans un temps distinct de celui de la construction du programme : l'exécution est différée (Samurçay, Rouchier 1985)

Le travail, présenté sans cet article, montre aussi la nécessité de questionner l'utilisation actuel des tableurs dans l'enseignement. Le tableur – calculatrice, ou le tableur – logiciel, permet l'évitement de la prise en charge de la répétition des calculs et du traitement des variables, en particulier dans le domaine de l'analyse. Ceci n'est-il pas en contradiction avec la volonté affichée par les programmes d'initier les élèves à l'algorithmique et au raisonnement qui lui est propre ?

---

[20] Certains organigrammes des manuels scolaires illustrant des structures itératives se réfèrent encore au branchement archaïque *aller à* de la boucle B2.

[21] La caractérisation de ces conditions et la transformation du recepteur des messages des élèves, CALCULATOR , en une « machine » fait l'objet du travail de recherche actuel de Nguyen C.T. dans le cadre d'une thèse en cotutelle entre la France et le Viêt-nam.

Bibliographie

BROUSSEAU G. (1998) *Théorie des situations didactiques*. Éd. La Pensée Sauvage, Grenoble.

CHEVALLARD Y. (1991) Concepts fondamentaux de la didactique : perspectives apportées par une approche anthropologique, *Recherches en Didactique des Mathématiques,* vol. 12.1, éd. La Pensée Sauvage, Grenoble. 73-112.

GANASCIA J-G. (1998) *Dictionnaire de l'informatique et des sciences de l'information*. Flammarion, Paris.

GOLDSCHLAGER L., LISTER A. (1986) *Informatique et algorithmique*. Inter-Edition, Paris.

LABORDE C., BALACHEF N., MEJIAS B. (1985) Genèse du concept d'itération : une approche expérimentale. *Enfance* n° 2-3. 223–239.

LE VAN T. (2001) *Etude didactique de lien entre fonctions et équations dans l'enseignement des mathématiques au lycée en France et au Viêt-nam*. Thèse. Université Joseph Fourier, Grenoble I

MATIIASSEVITCH Y. (2000) Le dixième problème de Hilbert : que peut-on faire avec les équations diophantiennes ? in *De la recherche de la vérité*. Les éditions du Kangourou.

MEJIAS B. (1985) *Difficultés conceptuelles dans l'écriture d'algorithmes itératifs chez des élèves de collège*. Thèse. Université Scientifique et Médicale de Grenoble.

NGUYEN C. T. (2001) *La notion d'algorithme dans l'enseignement des mathématiques au lycée. Comment l'émergence des notions de boucles et de variables s'articule à des connaissances en mathématiques ?* Mémoire DEA EIAHD, Université Joseph Fourier, Grenoble I.

N'GUYEN H. X. (1996) Problèmes, algorithmes et théorèmes. *Séminaire Didatech* n°174, Laboratoire Leibniz, Grenoble. 55-67.

RAVEL L. (2000) *Des programmes ... à la classe en passant par les manuels : Etude de la réintroduction de l'Arithmétique en Terminale S spécialité*. Mémoire, DEA EIAHD, Université Joseph Fourier, Grenoble I.

RAVEL L. (2002) Arithmétique en terminale S spécialité : quel(s) enseignement(s) ? *Repères-IREM*, n° 49. 93-116.

ROGALSKI J. (1988) Les représentations mentales du dispositif informatique dans l'alphabétisation. *Actes du premier colloque franco-allemand de didactique.* 235 – 245

SAMURÇAY R. (1985) Signification et fonctionnement du concept de variable informatique chez des élèves débutants. *Educational Studies in Mathematic.* n° 16. 143 – 161.

SAMURÇAY R. , ROUCHIER A. (1985) De « faire » à « faire faire ». *Enfance* n° 2-3, 241-254.

VEIGNEAU S. (1999) *Approches impérative et fonctionnelle de l'algorithmique.* Edition Springer, Paris.