Pluralités culturelles et universalité des mathématiques :
enjeux et perspectives pour leur enseignement
et leur apprentissage

espace mathématique francophone
Alger : 10-14 Octobre 2015

ÉTUDE D'UNE TRANSPOSITION DIDACTIQUE DE L'ALGORITHMIQUE AU LYCÉE : UNE PENSÉE ALGORITHMIQUE COMME UN VERSANT DE LA PENSÉE MATHÉMATIQUE

Nathalie BRIANT[*] – Alain BRONNER[**]

Résumé – Nous nous positionnons sur la place de la *pensée algorithmique* relativement à la *pensée mathématique*, en analysant deux points. Le premier consiste à considérer l'émergence de la pensée algorithmique lors de la résolution d'un problème mathématique en utilisant un environnement informatisé. Nous définissons la *double transposition* qui s'en suit, en caractérisant la démarche algorithmique. Le second consiste à montrer comment le détour par une *pensée algorithmique* a permis de développer une *pensée algébrique* et d'asseoir des concepts algébriques relatifs à la notion d'équation. Nous nous appuyons sur les résultats d'une ingénierie didactique expérimentée sur des élèves de 15 ans du lycée français.

Mots-clefs : pensée algorithmique, pensée algébrique, algorithme, équations, programmation

Abstract – We position ourselves about the place of *algorithmic thinking* in relation to *mathematical thinking*, analysing two points. The first is to consider the emergence of algorithmic thinking when solving a math problem using a computerized environment. We define *double transposition* that follows, featuring algorithmic approach. The second is to show how the detour through an *algorithmic thinking* has helped develop an *algebraic thinking* and sit algebraic concepts relating to the notion of equation. We build on the results of a didactic engineering tested on 15 years students of French high school.

Keywords: algorithmic thinking, algebraic thinking, algorithm, equations, programming

Préambule : Nous apportons ici une contribution au groupe de travail n°3, en nous centrant sur l'axe 2 intitulé « *L'activité du sujet au cœur du développement de la pensée mathématique* », et plus particulièrement sur le développement de la pensée algébrique des élèves, par le biais d'une pensée algorithmique. L'axe 1, « *La pertinence d'une différentiation de différents modes de pensées mathématiques* », est également convoqué puisque des caractéristiques propres à chacun de ces deux modes de pensée sont évoquées.

La réforme des lycées en France de 2009 s'est accompagnée d'un changement de programmes. Relativement à la classe de seconde (secondaire 2, 1ère année, 15–16 ans), une part d'algorithmique a été introduite dans le programme de mathématiques. L'objet de ce texte fait suite à nos travaux de thèse intitulée « *Étude didactique de la reprise de l'algèbre par l'introduction de l'algorithmique au niveau de la classe de seconde du lycée français* » (Briant 2013). Nous souhaitons développer trois hypothèses sur l'apport de l'algorithmique dans l'enseignement des mathématiques :

[*] LIRDEF. Université de Montpellier – France – nathalie.briant@fde.univ-montp2.fr
[**] LIRDEF. Université de Montpellier – France – alain.bronner@fde.univ-montp2.fr

Briant N., Bronner A. (2015) Étude d'une transposition didactique de l'algorithmique au lycée : une pensée algorithmique comme un versant de la pensée mathématique. In Theis L. (Ed.) *Pluralités culturelles et universalité des mathématiques : enjeux et perspectives pour leur enseignement et leur apprentissage* – Actes du colloque EMF2015 – GT3, pp. 231-246.

- concernant certains types de problèmes mathématiques, dont nous donnons quelques exemples plus loin, à toute forme de pensée mathématique nécessaire à leur résolution s'adjoint une pensée algorithmique particulière, dès que l'on cherche à les résoudre en utilisant les Technologies de l'Information et de la Communication pour l'Enseignement (TICE), et notamment des langages de programmation ;

- par le choix de certains types de tâches, le développement de la pensée algorithmique chez les élèves peut permettre le développement de leur pensée mathématique en rendant plus visible les objets utilisés et les étapes de la procédure de résolution ;

- le fait de dégager une démarche algorithmique sur le problème à résoudre amène à se situer au niveau d'un type de tâches, et non plus au niveau de la tâche elle-même (au sens de Chevallard 1999). C'est-à-dire qu'un point de vue plus général est abordé, au-delà des cas particulier de certaines variables, permettant ainsi un accès plus important aux concepts en jeu et une meilleure compréhension de ceux-ci.

Pour soutenir ces hypothèses, nous reprenons certains résultats de notre travail de thèse, montrant comment des élèves de seconde du lycée français (15-16 ans) ont lié *pensée algorithmique* et *pensée algébrique* et comment ce travail leur a permis d'asseoir leurs connaissances en algèbre. Le cadre didactique sous-jacent à cette étude est celui de la théorie anthropologique du didactique de Chevallard (1999).

I. L'AVENEMENT D'UNE PENSEE ALGORITHMIQUE POUR RESOUDRE CERTAINS TYPES DE PROBLEMES MATHEMATIQUES UTILISANT LES TICE

Dans cette première partie, nous nous proposons de développer ce que recouvre pour nous le terme de *pensée algorithmique*, pensée qui se développe au sein d'une *pensée mathématique*, mais sans se fondre complètement dans celle-ci, mais venant la compléter, amenant d'autres points de vue sur le problème à résoudre, d'autres techniques et d'autres technologies.

1. Un exemple de la nécessité de développer une pensée algorithmique pour résoudre un problème mathématique dans un environnement informatisé

La plupart des procédures de résolution de problèmes mathématiques ne se présentent pas directement sous la forme d'un algorithme. En général, la structure de cet algorithme reste à déterminer, à partir d'éléments de la résolution mathématique du problème posé dans un dispositif papier-crayon. Nous cherchons à montrer que la recherche d'un algorithme à partir des éléments de cette résolution n'est pas *transparente*[1], au sens de Artigue (1997), pour l'individu qui doit accomplir cette tâche.

[1] Artigue (1997) définit le concept de pseudo-transparence comme un *phénomène qui renvoie à des décalages dans les modes de représentation (interne et à l'interface) des objets*. Ce chercheur souligne que même si les phénomènes peuvent sembler minimes, ils perturbent le fonctionnement didactique. Pour exemplifier ce phénomène, prenons le cas de l'écriture des expressions algébriques sur support informatisé en considérant par exemple la tâche « entrer l'expression $\frac{x+4}{x-7}$ dans un logiciel ou une calculatrice ». Il est nécessaire d'adapter l'expression, de la *transposer* pour qu'elle soit lisible par une calculatrice actuelle de type collège (ou encore un tableur) sous la forme d'une écriture linéaire parenthésée du type $(x+4)/(x-7)$. La confrontation des deux environnements papier-crayon et informatique peut s'avérer bénéfique pour une compréhension approfondie de concepts, en exhibant des techniques ou des technologies différentes. Par exemple ici, la technique consistant à transformer une écriture spatiale en une linéarisation et un parenthésage des expressions permet d'asseoir les règles de priorité des opérations.

Commençons par présenter un exemple d'algorithme portant sur la simplification de la racine carrée d'un entier naturel noté *N*, sous la forme $a\sqrt{b}$ avec *a* et *b* entiers et *b* le plus petit possible.

L'existence de l'écriture $\sqrt{N} = a\sqrt{b}$ étant assurée[2], il reste à trouver une procédure qui permette de déterminer les valeurs de *a* et *b*.

Une première possibilité est de décomposer *N* en facteurs premiers et de considérer la parité des exposants des nombres premiers en présence pour calculer *a* et *b* : un premier algorithme peut ainsi être réalisé. Cet algorithme est souvent utilisé en environnement papier-crayon (pour des nombres « pas trop grands ») et se présente sous la forme suivante :

Algorithme n°1

Effectuer la décomposition de *N* en facteurs premiers : $N = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$

Pour tout entier i compris entre 1 et *k* :

- Si α_i est pair, *a* contient le facteur $p_i^{\frac{\alpha_i}{2}}$;
- Si α_i est impair, *a* contient le facteur $p_i^{\frac{\alpha_i - 1}{2}}$ et *b* contient le facteur p_i.

Une seconde possibilité est d'utiliser l'égalité $N = a^2 b$ et d'effectuer l'algorithme suivant :

Algorithme n°2

Pour chaque entier *I* compris entre 1 et Ent($\sqrt{N}$) :

- Tester si la division euclidienne de *N* par I^2 donne un reste nul ;
  - Si c'est le cas, affecter à *a* la valeur de *I* ;
  - Si ce n'est pas le cas, *a* garde sa valeur.
- Passer à la valeur suivante de *I.*

Calculer la valeur de $b = N/a^2$

Ce second algorithme donne, en sortie de la structure répétitive, la plus grande valeur possible de *a*, puisque cette valeur est modifiée à chaque nouvelle valeur de *I* qui vérifie $N \equiv 0\ [I^2]$. La valeur de *b* en est alors déduite.

L'un ou l'autre de ces algorithmes répond au problème posé, cependant le second possède l'avantage sur le premier de ne pas nécessiter d'effectuer en préambule la décomposition du nombre *N* en facteurs premiers.

Si l'objectif est de programmer cet algorithme, une machine n'ayant pas -a priori- implanté en son sein un programme de décomposition en facteurs premiers d'un nombre entier, le second algorithme est, de ce fait, plus facilement transposable en un algorithme *informatisé*,

[2] Si *N* est un carré parfait, il existe *a* entier tel que $N = a^2$ et par suite : $\sqrt{N} = a\sqrt{1}$;
Si *N* n'est pas un carré parfait, il existe un plus grand carré parfait *a* qui divise *N* (au pire *a* = 1). Alors $N = a^2 b$ et $\sqrt{N} = a\sqrt{b}$. Si *a* est le plus grand possible, *b* sera le plus petit possible.

qui pourra alors être écrit dans un langage de programmation, compréhensible par une machine, comme celui réalisé ci-dessous sur le logiciel Algobox[3].

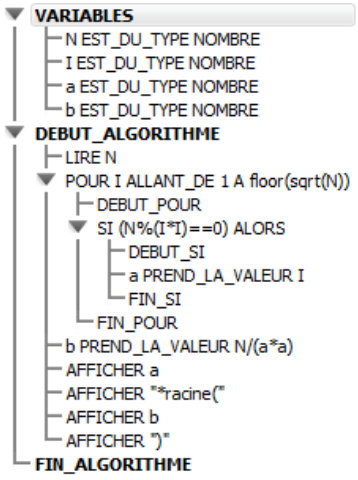
Soit *N* un entier naturel. Pour chaque entier *I* compris entre 1 et Ent($\sqrt{N}$) : - Tester si la division de *N* par I^2 donne un reste nul ; - Si c'est le cas, affecter à *a* la valeur de *I* ; - Si ce n'est pas le cas, passer à la valeur suivante de *I*. Calculer la valeur de $b : N/a^2$ Afficher racine(*N*) = *a**racine(*b*)		N = 120 ***Algorithme lancé*** 2*racine(30) ***Algorithme terminé*** N = 256 ***Algorithme lancé*** 16*racine(1) ***Algorithme terminé*** N = 1789 ***Algorithme lancé*** 1*racine(1789) ***Algorithme terminé***
Algorithme de simplification de $\sqrt{N}$ sous la forme $a\sqrt{b}$ où (a, b) ∈ N^2 et où b est le plus petit possible.	*Programme correspondant à l'algorithme ci-contre sous Algobox*	*Résultats obtenus par le programme pour trois valeurs particulières de N*

Figure 1 *- Exemple de simplification des racines carrées sur le logiciel Algobox*

Cet exemple montre que la recherche d'un algorithme *informatisé* s'appuie sur la résolution mathématique mais que celle-ci nécessite d'être *transposée* (au sens de Balacheff, 1994) pour tenir compte des actions qui sont élémentaires pour la machine. Ainsi ce premier exemple montre que bien que la résolution de ce type de tâches repose sur un algorithme (n°1) en environnement papier-crayon, mais que la recherche d'un autre algorithme (n°2) est souvent nécessaire afin résoudre le même type de tâches dans un environnement informatisé. Ainsi, une pensée algorithmique vient s'intégrer à la pensée mathématique initiale, non pas en se substituant à elle, mais en la complétant.

2. *De la résolution mathématique au programme informatique : une double transposition*

Le concept de transposition didactique (Chevallard 1985) a été repris par Balacheff (1994) et retravaillé en tenant compte des contraintes liées à l'apprentissage de savoirs en environnement informatique sous le nom de *transposition informatique.* Le savoir enseigné dans une situation classique d'enseignement est différent du savoir enseigné avec un ordinateur, ce qui peut se schématiser comme suit :

[3] Algobox est un logiciel de programmation, libre et gratuit, développé en 2009 par Pascal Brachet**,** professeur de mathématiques au lycée Bernard Palissy à Agen. L'auteur le définit lui-même comme un *logiciel pédagogique d'aide à la création et à l'exécution d'algorithmes*. Ce logiciel est bien adapté pour des lycéens puisque son principe est que le code de l'algorithme se construit pas à pas grâce à des instructions de base pré-écrites que l'on insère, comme des « briques » dans le corps du programme. L'activité est alors davantage centrée sur la réflexion du choix des instructions et de leurs articulations plutôt que sur la syntaxe de lignes de code.

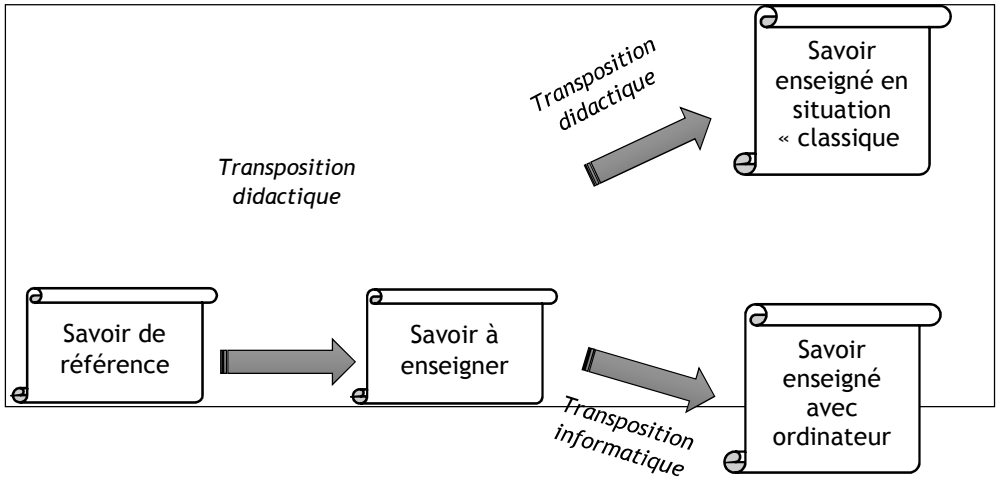
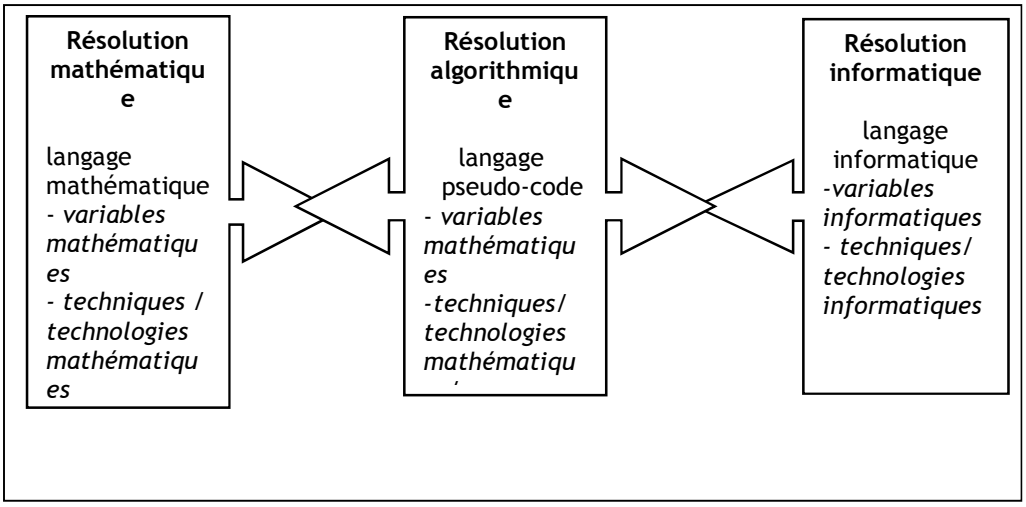


Figure 2 *- Transpositions didactique et informatique (Chevallard 1982, Balacheff 1994)*

Balacheff (1994) explique qu*'aux contraintes de la transposition didactique s'ajoutent, ou plutôt se combinent, celles de modélisation et d'implémentation informatiques*. Ce chercheur définit deux types de contraintes liées à la transposition informatique, les *contraintes de la modélisation computable* et les *contraintes logicielles et matérielles* des supports informatiques. Les premières portent sur la représentation et le traitement interne des savoirs dans la machine et les secondes sur la représentation et le traitement au niveau de l'interface, autrement dit ce qui est « visible » pour le sujet. Balacheff (ibid.) donne l'exemple du logiciel Cabri-Géomètre qui possède une représentation interne des objets géométriques issue de la géométrie analytique sur un modèle des nombres réels et une interface offrant une représentation de ces objets sous forme d'un pavage fini de pixels. Il précise que ces représentations ne sont pas transparentes : « *les systèmes de représentations ayant leurs propres caractéristiques, l'univers interne et l'interface combinent des effets générateurs et des phénomènes non intrinsèques aux entités représentées.* » (ibid., p.16)

Nos recherches sur l'intégration de l'algorithmique dans l'enseignement des mathématiques nous amènent à reprendre le concept de transposition informatique de Balacheff mais avec une adaptation, tenant compte de la singularité de l'algorithmique. En effet, lorsqu'une tâche de type « concevoir un programme pour résoudre un problème » est donnée, nous voyons émerger une *double transposition*, associée à des techniques différentes, justifiées par des technologies relevant du domaine mathématique, du domaine informatique, ou des deux conjointement.

Nous schématisons et explicitons ci-dessous cette double transposition.



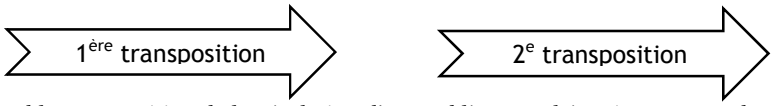


Figure 2 *- Double transposition de la résolution d'un problème mathématique en vue de sa programmation*

- *Premier cadre de la figure 2 : résolution mathématique*

Ce premier cadre symbolise la résolution du problème posé dans le cadre mathématique « habituel », c'est-à-dire en environnement classique papier-crayon. Cette résolution s'appuie sur des techniques et un environnement technologico-théorique mathématique : c'est ainsi que la résolution du problème est dite *mathématique* (par opposition ici à *informatique*).

Cette résolution peut avoir donné lieu à un premier algorithme, que nous nommons *algorithme mathématique*, comme l'algorithme n°1 de l'exemple de la simplification d'une racine carrée, mais notons que toutes les résolutions de problèmes mathématiques ne se présentent pas nécessairement sous la forme d'un algorithme.

- *Deuxième cadre de la figure 2 : résolution algorithmique*

La résolution mathématique achevée, une première transposition a lieu pour déterminer un algorithme *informatisé*, écrit en *pseudo-code*, notions que nous allons définir plus précisément.

Dans l'exemple de la simplification d'une racine carrée, nous avons vu que l'algorithme mathématique n°1, utilisé habituellement dans un environnement papier-crayon, nécessite la connaissance des nombres premiers, ce qui n'est pas généralement pas implanté de base dans un logiciel quelconque de programmation. La recherche d'autres algorithmes intervient alors dans cette phase, comme l'algorithme n°2 présenté plus haut, dont la structure tient compte des actions élémentaires réalisables par une machine et des logiciels de programmation intégrés : nous nommons ainsi *algorithme informatisé* ce type d'algorithme dont la fonction est de faciliter le passage à la deuxième étape qui sera développée plus loin.

Cette première transposition se fait à différents niveaux :

- au niveau du langage : nous passons d'un langage mathématique, c'est-à-dire *le langage utilisé usuellement par les écrits mathématiques* (Modeste 2012, p.62) à un langage en *pseudo-code* ressemblant à un langage de programmation, qui serait débarrassé de ses problèmes de syntaxe. Modeste (2012, p.24) le présente comme *un langage intermédiaire, inspiré des instructions des langages informatiques mais libéré de certaines contraintes et manipulant directement les objets mathématiques* ;

- au niveau des techniques et technologies utilisées toujours en considérant notre exemple, les techniques utilisées pour l'algorithme n°1 reposent sur la décomposition du nombre N en facteurs premiers et la parité des exposants des nombres premiers en présence, alors que pour l'algorithme n°2, elles reposent sur la recherche par divisions successives d'un plus grand élément dont le carré divise le nombre N ; les technologies-théories sous-jacentes s'en trouvent alors modifiées.

La résolution algorithmique du problème donné s'appuie sur la transposition à venir qui permettra de transformer *l'algorithme informatisé* en un programme, dont les contraintes sont propres au logiciel et à la machine choisis pour l'implanter.

- *Troisième cadre de la figure 2 : résolution informatique*

Une seconde transposition aboutit à l'écriture du programme avec un logiciel adéquat.

Elle se fait elle aussi à différents niveaux :

- au niveau du langage : il s'agit de passer du langage en pseudo-code de l'algorithme à un langage informatique, c'est-à-dire un langage de programmation. À titre d'exemple, en reprenant la simplification de la racine carrée (cf. figure 1), la propriété « I^2 divise N », *élémentaire* pour un individu, nécessite une reformulation pour donner un équivalent qui soit compréhensible par une machine, selon sa structure interne et dans son langage. Le test choisi sous la forme « $N \ \% \ I^2 = 0$ »[4] a été choisi mais toute autre formulation utilisant une succession d'opérations élémentaires pour le logiciel et préprogrammées dans celui-ci, conviendrait (comme par exemple *« Ent (N/I²) = N/I² »*).

- au niveau des variables : les variables mathématiques utilisées dans les algorithmes vont céder la place aux variables informatiques dans le programme informatique. Remarquons que ces variables font partie de la technologie des praxéologies informatiques mais leur importance dans la programmation nous les fait préciser ici ;

- au niveau des techniques et technologies utilisées : aux techniques et technologies-théories propres à la résolution du problème en environnement papier-crayon viennent s'adjoindre celles liées aux principes de programmation informatique (notion de variables informatiques, affectation de variables, lecture/écriture, structures alternatives, structures répétitives).

Au centre de cette seconde transposition, apparaît la nécessité de comprendre ce qui est élémentaire pour une machine et ce qui ne l'est pas, ce qui va impliquer de comprendre les bases du fonctionnement d'un ordinateur, au niveau des instructions que celui-ci peut exécuter. Nous sommes au cœur de la *transposition informatique* (Balacheff 1994) et d'*instrumentation*[5] (Rabardel 1995) avec des problèmes de *pseudo-transparence*, au sens d'Artigue[6] (2005).

En conclusion de cette première partie, les tâches du type « concevoir un programme pour résoudre un problème mathématique » se décomposent en deux phases :

- la création *d'algorithmes informatisés* ou bien l'adaptation *d'algorithmes mathématiques* à partir de la résolution mathématique du problème en environnement « classique » papier-crayon ;

- l'adaptation de l'algorithme informatisé retenu en un programme, implanté sur un système informatique donné.

Bien entendu, ces phases ne sont pas forcément séquentielles ni chronologiques, de nombreux aller-et-retour pouvant survenir. Avec l'expérience, certains experts peuvent élaborer plus directement un programme mais on peut faire l'hypothèse que, face à des problèmes complexes, ils ont aussi besoin de passer par un algorithme informatisé et du pseudo-code.

Pour résoudre un tel type de tâches, le sujet va devoir apprendre à se décentrer de sa posture d'individu pour se placer dans la position de tenir compte de ce que sait faire la machine, s'il veut que son algorithme soit transférable en un programme. C'est en ce sens qu'émerge une *pensée algorithmique*, non entièrement « superposable » à la *pensée*

[4] La syntaxe « *a % b* » renvoie le reste de la division euclidienne de *a* par *b* sous le logiciel Algobox.

[5] L'*instrument* constitue pour Rabardel (1995) *une entité mixte qui tient à la fois du sujet et de l'artefact* : l'instrument est alors l'unité entre un *artefact* et l'organisation d'actions possibles, appelées les *schèmes d'utilisation*, qui constituent un ensemble structuré d'invariants correspondant à des catégories d'opérations réalisables à l'aide de l'artefact considéré. Tout au long de ce processus de conception, de création, de modification et d'utilisation d'un l'outil, le sujet évolue aussi personnellement en même temps qu'il s'approprie cet outil, et cette évolution concerne à la fois son comportement et sa connaissance. L'*instrumentation* (et l'instrumentalisation) est ce qui consiste à faire d'un artefact un instrument.

[6] Cf. note n°1 de bas de page.

mathématique mais intimement liée à elle. Pour la définir, nous choisissons cette définition de Modeste (2012), lui-même inspiré par Hart[7] :

> La pensée algorithmique serait alors une façon d'aborder un problème en essayant de systématiser sa résolution, de se questionner sur la façon dont des algorithmes pourraient ou non le résoudre. (p.47)

Nous retrouvons ainsi le troisième point indiqué dans l'introduction, sur le fait qu'une pensée algorithmique amène à une *généralisation* d'une tâche donnée, en s'intéressant au type de tâches sous-jacent, c'est-à-dire en ne considérant pas le problème pour des valeurs numériques données, mais en envisagent sa résolution de manière plus générale.

II. LIENS ENTRE PENSEE ALGORITHMIQUE ET PENSEE ALGEBRIQUE DANS LE CADRE D'UNE INGENIERIE DIDACTIQUE EN CLASSE DE SECONDE

Nous présentons ici une partie de l'ingénierie didactique, développée dans le cadre de notre thèse (Briant 2013). L'objectif est double ici. Il s'agit d'une part d'illustrer le modèle développé dans la première partie (Cf. section I.2 et figure 2) et d'autre part de montrer comment le détour par une pensée algorithmique a permis de développer et d'asseoir des concepts algébriques gravitant autour de la notion d'équation.

La situation que nous décrivons ci-après a été expérimentée auprès d'élèves de trois classes de seconde du lycée français (secondaire 2, 1ère année, 15–16 ans) et menée par les enseignants de ces classes. La situation offre un travail sur la *modélisation* des équations, ce qui consiste en la détermination d'une équation *paramétrée* qui couvre tous les cas de figure d'une liste d'équations donnée. Par exemple, pour les deux équations $1{,}5x - 4 = 412x + \frac{2}{3}$ et $x - 4 = 0$, un *modèle* qui convient est $ax + b = cx + d$, alors que $x + a = 0$ ne convient que pour la seconde (*a, b, c, d* nombres réels fixés). Cette modélisation est permise par l'introduction de l'algorithmique et de l'outil informatique, comme Balacheff (1994) l'exprime :

> L'expression computationnelle des objets d'enseignement pour leur inscription dans un dispositif informatique dédié à l'apprentissage n'est pas le résultat d'un simple processus de traduction d'un système de représentation vers un autre, mais celui d'un véritable processus de modélisation et donc de théorisation des objets d'enseignement et de leurs conditions d'existence. (p.10)

Balacheff précise que l'introduction de l'outil informatique va remettre en question l'écologie des savoirs enseignés, dans le sens où elle conduit à l'explicitation de contenus d'enseignement jusque-là non-dits, voire à la création de nouveaux objets d'enseignements (ibid.). C'est bien ce qui se produit dans le cadre de notre expérimentation où le détour par l'algorithmique permet de considérer les équations comme *objet*[8] de l'algèbre au sens de Douady (1986) sur lesquels on amène les élèves à s'interroger.

L'objectif annoncé aux élèves est la création de programmes permettant de résoudre « automatiquement » les équations de la liste ci-dessous (cf. figure 3) :

---

[7] Référence donnée par Modeste (2012) : Hart E. W. (1998) Algorithmic Problem Solving in Discrete Mathematics. In Morrow L. J. & Kenney M. J. (Eds.) *The teaching and learning of Algorithm in school mathematics, 1998 NCTM Yearbook* (pp. 251-267). Reston, VA : National Council of Teachers of Mathematics.

[8] Citons Douady (1986) afin d'expliciter les concepts d'objet et d'outil : « […] un concept est ***outil*** lorsque nous focalisons notre intérêt sur l'usage qui en est fait pour résoudre un problème. Un même outil peut être adapté à plusieurs problèmes, plusieurs outils peuvent être adaptés à un même problème. Par ***objet***, nous entendons l'objet culturel ayant sa place dans un édifice plus large qui est le savoir savant à un moment donné, reconnu socialement. »

1. Réaliser un algorithme sur le logiciel Algobox permettant de résoudre les trois premières équations ci-dessous, **sans les transformer au préalable**.

2. Signaler par une * les équations similaires. Faire fonctionner l'algorithme pour ces équations.

3. Comment peut-on résoudre les équations restantes avec un autre algorithme ? Le construire et résoudre les autres équations à l'aide de ce nouvel algorithme.

*Équation 1 :	$x + 3 = 0$	Équation 7 :	$\frac{7}{2}x + 3 = \frac{2}{3}$
*Équation 2 :	$2x - 3 = 4$	Équation 8 :	$\frac{7}{2}x + \frac{2}{5} = \frac{8}{3} + \frac{1}{2}x$
*Équation 3 :	$3 - 2x = -2$	Équation 9 :	$3 = 2x + 1$
Équation 4 :	$2 + x = 5x$	Équation 10 :	$3x + 2 = 5 + 3x$
Équation 5 :	$2x + 3 = 3x + 1$	Équation 11 :	$1{,}8x - 3 = 2{,}5x + 7{,}4$
Équation 6 :	$8 - x = \sqrt{2}$		

Figure 3 *- Fiche élève de la situation expérimentée*

1. Contexte de l'expérimentation menée

Au niveau des programmes institutionnels français, les différentes capacités requises pour résoudre algébriquement les équations présentées en figure 3 se situent en classe de quatrième du collège (secondaire 1, 3e année, 13–14 ans). Le programme de ce niveau de classe stipule de « mettre en équation et résoudre un problème conduisant à une équation du premier degré à une inconnue » ainsi que de « réduire une expression littérale à une variable, du type 3x – (4x – 2) » (MEN 2008). Pour la classe de troisième du collège (secondaire 1, 4e année, 14–15 ans), il est précisé de « *compléter les bases du calcul littéral et d'en conforter le sens, notamment par le recours à des équations ou des inéquations du premier degré pour résoudre des problèmes* » (Ibid.). Pour la classe de seconde du lycée (secondaire 2, 1ère année, 15–16 ans), le programme enjoint de savoir « *résoudre une équation se ramenant au premier degré* » (MEN 2009).

Ainsi le public ciblé pour l'expérimentation menée, constitué d'élèves de classes de seconde, a-t-il déjà reçu un enseignement concernant la résolution algébrique de ce type d'équations. En revanche, le travail de modélisation – explicité précédemment – pour obtenir la forme générique de ces équations, ne fait pas partie expressément des programmes institutionnels français, et les élèves testés ne l'ont pas rencontré au cours de leur scolarité. Il s'agit de retravailler des connaissances déjà rencontrées au collège, mais en alliant *ancien et nouveau,* en proposant ce travail dans le cadre de l'algorithmique. Les élèves sont amenés à manipuler les inconnues, les paramètres et les techniques de résolution des équations du premier degré. Si le but pour l'élève est la réalisation d'un programme qui « tourne » sur une machine, l'objectif d'enseignement est une reprise du concept d'équation et des objets qui gravitent autour de ce concept.

Pour situer plus globalement la situation proposée aux élèves dont la fiche de travail est présentée en figure 3, décrivons brièvement la séquence dans laquelle elle s'inscrit. Celle-ci composée de trois situations[9] déclinées comme suit :

[9] Les situations n°1 et n°3 ne sont pas explicitées davantage dans ce document. Elles sont données ici afin de situer l'expérimentation menée. La situation n°1 a fait l'objet d'un travail de groupes de 4 à 5 élèves qui se sont accordés sur un classement possible des équations données, travail ensuite restitué au groupe classe par un rapporteur puis discuté par l'ensemble de la classe. La situation n°3 a été menée en salle informatique, avec la

- la situation n°1 permet un travail sur la *catégorisation* d'équations de degré 1 et 2, ce qui consiste en un classement par les élèves d'une vingtaine d'équations polynomiales de degré 1 ou 2, selon des critères à déterminer. Le choix de ces équations permet d'identifier les conceptions des élèves relatives à la notion d'équation et de renforcer, ou le cas échéant de faire émerger la connaissance que la forme et la nature d'une équation influent sur sa technique de résolution ;

- les situations n°2 et n°3 sont construites de façon similaire et offrent chacune deux types de tâches : le premier sur la *modélisation* des équations, ce qui consiste en la détermination d'une équation *paramétrée* qui couvre tous les cas de figure d'une liste d'équations donnée ; le second sur la détermination des techniques de résolution de types d'équations reconnues, par le biais de l'algorithmique et de la programmation. La différence entre les deux situations est le degré des équations considérées : pour la situation n°2, les équations sont polynomiales de degré 1 alors que pour la situation n°3, elles sont de degré 2.

C'est la situation n°2 qui est développée dans cet article. Elle a été conduite en février 2011 dans trois classes de seconde d'un même lycée. Les trois professeurs expérimentateurs ont, au moment de l'expérimentation, déjà initié leurs élèves à la structure de base d'un algorithme et à une prise en main du logiciel Algobox, débutant ainsi l'*instrumentation*, au sens de Balacheff (1994), de ce logiciel. Le travail se déroule en binômes, au cours d'une séance d'une heure, les élèves disposant d'un poste informatique pour deux et d'une liste d'équations à résoudre, résolution à effectuer en élaborant un ou des algorithmes qui sont ensuite programmés.

2. *Analyse a priori*

Une analyse a priori de cette situation permet de montrer la double transposition nécessaire pour passer de la résolution mathématique à la conception du programme. Si nous contextualisons les trois étapes de la figure 2 à la situation présentée, nous obtenons le schéma suivant pour la résolution de l'équation $ax + b = cx + d$ $(a \neq c)$:

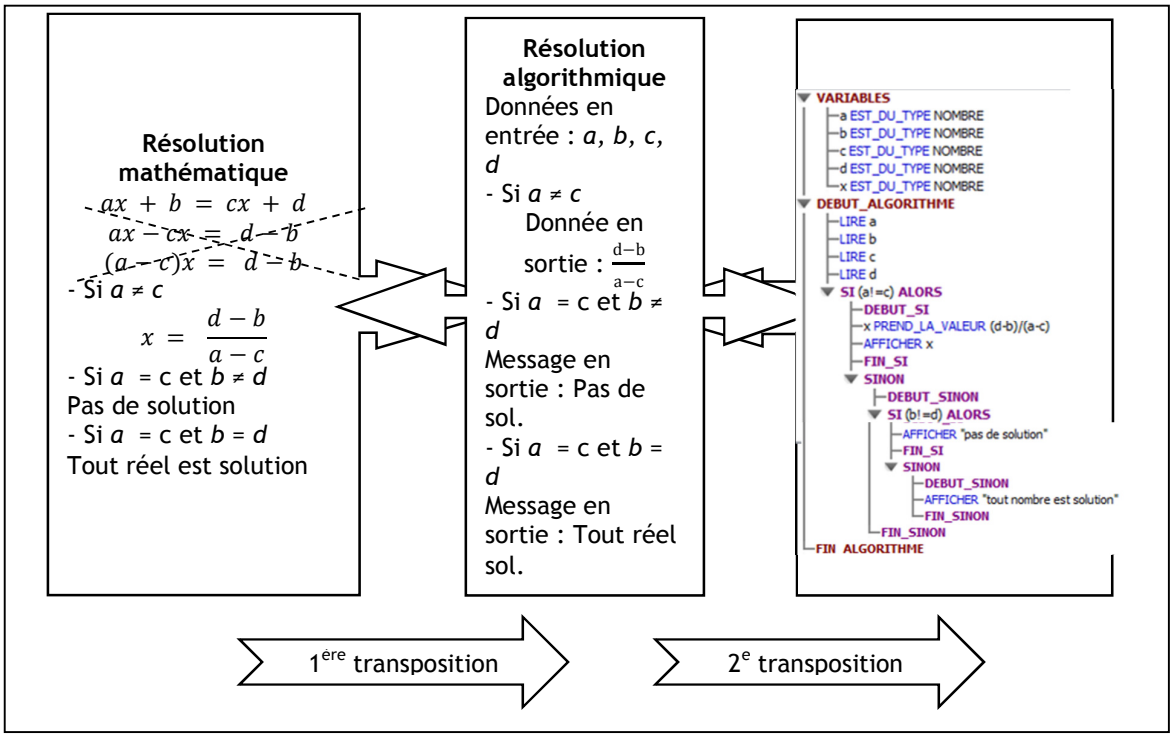


Figure 4 *- Double transposition de la résolution d'une équation du 1er degré en vue de sa programmation*

Cette double transposition a les caractéristiques suivantes :

- la *première transposition* nécessaire à la conception d'un algorithme à partir de la résolution « mathématique » de l'équation du premier degré implique la disparition d'étapes intermédiaires de la résolution en environnement papier-crayon (indiquées en pointillés sur la figure 4). Il existe ici une *pseudo-transparence* (Artigue 1997) des environnements papier-crayon et algorithmique et une *non-congruence* de ces deux environnements, au sens de Duval (1995). Il y a un réel travail de transposition à effectuer pour passer du résultat mathématique : « L'équation $ax + b = cx + d$ admet pour solution $\frac{d-b}{a-c}$, si $a \neq c$ » à la formulation algorithmique associée. D'autre part, la condition « si $a \neq c$ » arrive en fin de la résolution « à la main », sous forme d'une discussion selon la nullité ou non de $a - c$, alors qu'elle est nécessaire tout au début de l'algorithme. Il y a toute une *reconstruction* nécessaire à envisager pour concevoir l'algorithme à partir de la résolution de l'équation, une autre pensée dans laquelle il faut entrer, une *pensée algorithmique*.

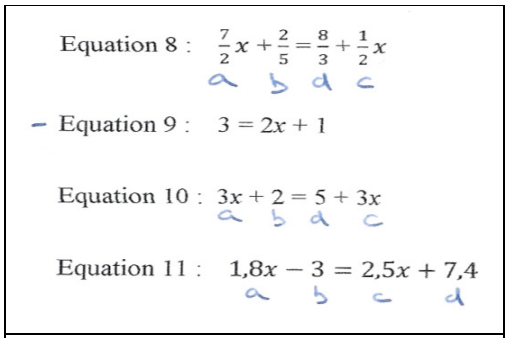
- la *seconde transposition* pour la conception du programme informatique à partir d'un algorithme nécessite non seulement la compréhension d'un nouveau langage, mais également une adaptation de l'algorithme aux contraintes du logiciel utilisé. Par exemple, le logiciel Algobox ne permet pas l'affichage direct d'une expression à calculer. Le logiciel n'accepte pas l'instruction « *afficher (d – b)/(a – c)* », ce qui oblige à passer par une variable intermédiaire. Ceci induit ici encore une non-congruence entre l'algorithme conçu et le programme informatique.

3. Analyse a posteriori

Bilan des résultats des élèves

Outre la mise en place elle-même de cette situation conçue pour permettre la prise en compte de l'aspect *objet* des équations au sens de Douady (1986), nous avons relevé des indicateurs montrant l'émergence de cette prise en compte par les élèves et leur entrée dans une *pensée algébrique.* Nous partageons la conception d'une pensée algébrique au sens de Radford (2006 et 2008), non nécessairement liée à l'utilisation de symbolismes algébriques et de règles formelles, mais caractérisée par deux grands principes : la possibilité de nommer des quantités indéterminées ou inconnues (dans des registres variés comme la langue, des schémas ou du symbolisme divers) et de raisonner sur ces quantités comme si elles étaient connues. Les indicateurs renvoient à des aspects différents des pensées algorithmique et algébrique comme le montrent les exemples suivants :

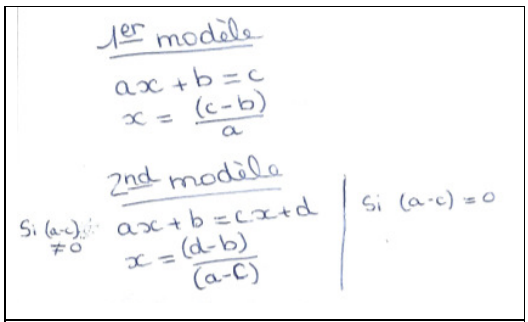
- l'utilisation en actes de *paramètres* nécessaires pour écrire sous une appellation unique différentes équations données dans le cadre d'une démarche algorithmique avec les bons choix des variables informatiques ; ce qui montre qu'une structure générale algébrique est bien dégagée par l'élève (cf. figure 5).



Production de l'élève Pierre

Figure 5 *- Élève considérant des équations du premier degré sous une forme générique*

- l'utilisation du terme *modèle* d'une équation lors de la recherche d'une forme générique d'équation correspondant à une liste donnée, montrant ainsi que l'élève est entré dans une posture spécifique d'une pensée algorithmique où il s'agit de dégager des procédures générales adaptées à des données (cf. figure 6).

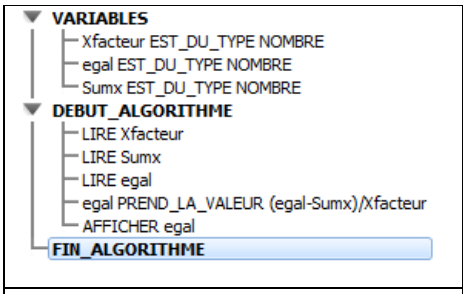


Production de l'élève Victorien

Figure 6 *- Production d'élève considérant la modélisation des équations*

C'est clairement le passage par l'algorithmique qui provoque l'évolution du rapport aux objets de l'algèbre, ce mode de pensée invitant les élèves à faire un pas de côté par rapport à leur considération habituelle de ces équations et des techniques qu'ils ont automatisées. Ces exemples montrent que le travail engagé va bien au-delà de la simple résolution d'équations du premier degré : les objets de l'algèbre sont réinterrogés, revisités, allant jusqu'à l'étude *de nouveaux objets* comme souligné par Balacheff, et en particulier l'étude du concept de *paramètre*. Chevallard (1989) précisait déjà l'intérêt didactique de l'introduction du concept de paramètre dans l'enseignement de l'algèbre élémentaire, en nommant ce concept *la force de l'algèbre*, et en affirmant que s'interroger sur les paramètres d'un système est une façon d'entrer dans la *modélisation algébrique*. Le détour par l'algorithmique permet d'accéder à ce concept de paramètre, en en réalisant une certaine *matérialisation,* au sein d'un programme informatique.

En effet, les élèves se sont questionnés sur la façon de communiquer les équations du type $ax + b = c$ à l'ordinateur. La différentiation du rôle des lettres permet de relier les notions de paramètre et d'inconnue à la structure de l'algorithme à concevoir : les paramètres sont des données d'entrée (les *connues* de Descartes) alors que l'inconnue est une donnée de sortie de l'algorithme. La structure de l'équation aide à comprendre la structure de l'algorithme, et réciproquement. L'exemple de l'élève Aliaume est significatif de ce changement de ces nouvelles connaissances (cf. figure 7).



Production de l'élève Aliaume

Figure 7 *- Programme de résolution de l'équation* $ax + b = c$

Pour déterminer une forme générique pour une liste d'équations de la forme $ax + b = c$, l'élève Aliaume utilise des noms pour les paramètres, évocateurs de leur statut. Il nomme l'équation :

(Xfacteur)x + (Sumx) = (egal).

Le coefficient de l'inconnue est nommé « *Xfacteur* », pour rappeler le rôle que joue ce paramètre que l'on multiplie par *x*. Le second terme du membre de gauche s'appelle « *Sumx* », ce qui lui confère le statut d'un nombre que l'on ajoute au terme en *x*. Enfin le paramètre de droite est appelé « *egal* », évoquant l'annonce du résultat d'une opération. Ces notations sont plus qu'un simple choix de vocabulaire et dénotent bien le sens attribué à ces objets. Elles permettent à l'élève de retrouver facilement de quel paramètre il s'agit, lorsqu'il manipule une équation avec des coefficients déterminés. C'est bien ici encore le détour par une *pensée algorithmique* qui autorise cette démarche.

Les limites de l'expérimentation

Néanmoins, la cohérence globale de la situation a été perdue de vue par certains élèves qui ne possédaient pas les premiers concepts de base de l'algorithmique, ni n'étaient suffisamment instrumentés pour l'utilisation du logiciel Algobox. Les observations menées dans les différentes classes ont montré que le nombre de manipulations liées à des questions de communication avec l'environnement informatique vient parasiter les rétroactions qui auraient été intéressantes d'un point de vue mathématique. Laborde (2003) indique que l'acquisition d'une genèse instrumentale est lente et complexe et que les premiers schèmes mis en place ne sont pas les plus efficaces. De plus, les schèmes impliquent une certaine connaissance mathématique :

> The use of technology in mathematics by students was analyzed from an instrumental perspective […]. The mentioned research papers provide examples that show that the instrumental genesis takes place over a long period of time and that the first schemes constructed by the learner are not the most effective. What also appears is that the schemes involve some mathematical knowledge. A double question arises from these observations: What are the relevant tasks allowing both the development of instrumentation schemes and of mathematical knowledge? [10]

L'équilibre entre les deux pôles, développer les schèmes et asseoir des connaissances mathématiques, est délicat à trouver, ce qu'a confirmé l'expérimentation conduite.

[10] L'utilisation de la technologie en mathématiques par les élèves a été analysée dans une perspective instrumentale […]. Les travaux de recherche mentionnés sont des exemples qui montrent que la genèse instrumentale se déroule sur une longue période de temps et que les premiers schèmes construits par l'apprenant ne sont pas les plus efficaces. Ce qui apparaît aussi, c'est que les schèmes impliquent une certaine connaissance mathématique. Une double question se pose à partir de ces observations : Quelles sont les tâches pertinentes permettant à la fois le développement de systèmes d'instrumentation et de la connaissance ?

III. CONCLUSION

En conclusion, nous revenons d'une part, sur la question de la pensée algorithmique pour matérialiser des objets de l'algèbre et d'autre part sur celle de l'existence d'une pensée algorithmique, qui s'adjoint à la pensée mathématique.

Pour le premier point, le détour par l'algorithmique, évoqué plus haut lors de la mise en œuvre de la situation exposée, est à rapprocher d'un *micro-monde* dont Capponi et Laborde (1995) donnent la définition suivante :

> Un micro-monde est une création d'un monde de réalités artificielles fournissant un modèle (au sens des logiciens) d'une théorie. Ce monde comporte des objets sur lesquels on peut agir grâce à des actions, on peut aussi créer de nouveaux objets. Une fois créés, les objets ont un comportement régulé par la théorie sous-jacente au modèle. Même si l'utilisateur du micro-monde peut agir sur ces objets, ces derniers présentent donc une certaine autonomie, de la même manière qu'on ne peut faire n'importe quoi avec un objet matériel. (p. 265).

Capponi et Laborde utilisent cette définition du micro-monde pour caractériser le logiciel de géométrie dynamique Cabri-géomètre. Nous effectuons ici le parallèle avec le logiciel de programmation Algobox qui se comporte comme tel, puisqu'il est possible de créer des représentations d'un objet théorique et d'agir sur ses représentations. La comparaison ne peut cependant être totale, la conception de Cabri-géomètre est *dédiée* à la manipulation d'objets géométriques et à l'exploration de leurs propriétés, alors qu'un logiciel quelconque de programmation n'est pas spécialement dédié au domaine algébrique. Néanmoins, ce parallèle permet de comprendre comment s'est créée une nouvelle représentation des objets de l'algèbre, dans le cadre de l'algorithmique et de la programmation. Laborde (2003) utilise le terme de *médiation* de l'outil informatique pour définir le rapport dialectique qui existe entre l'action et la signification mathématique. L'accès aux objets mathématiques abstraits, non *ostensifs*[11], se fait par l'intermédiaire de registres sémiotiques (Duval 1993) où l'utilisation de signes en donne des *ostensifs* (Chevallard & Bosch 1999). Ainsi l'algorithmique et la programmation forment un micro-monde pour le domaine algébrique, où il est possible d'explorer et d'expérimenter sur des objets de l'algèbre, comme s'ils étaient des objets *matériels*. Ceci renforce une entrée dans la pensée algébrique, comme montré dans les résultats précédents.

Pour le second point de cette conclusion, la situation exposée propose un type de problème mathématique dont est recherchée une résolution algorithmique pour toutes les instances du problème posé. La résolution algorithmique et son écriture dans un environnement informatique nécessitent une *transposition*. Nous pensons que la complexité de cette transposition vient non seulement de la *non-congruence* entre les environnements papier-crayon et informatisé, comme nos analyses le montrent, mais également d'une *pensée algorithmique* qui n'est pas contenue entièrement dans la *pensée mathématique*.

Cette idée est développée par Modeste (2012) qui précise qu'une *activité mathématique est centrée sur la résolution de problèmes* et que lorsque la pensée algorithmique est considérée *en tant que pensée mathématique parmi d'autres,* elle est alors *une approche particulière de certains problèmes mathématiques* (p. 47). Cependant, Modeste montre, en s'appuyant sur les travaux de Knuth[12], que l'essor de la science informatique a fait évoluer la pensée

[11] D'après Chevallard et Bosch (1999), les objets *ostensifs* sont ceux qui possèdent une forme matérielle (comme un crayon) ou accessible aux sens humains comme les gestes, les mots, les schémas, les écritures, etc. ; les objets *non ostensifs* sont les idées, les notions, les concepts, reconnus par une institution, *qui ne peuvent qu'être évoqués ou invoqués par la manipulation adéquate de certains objets ostensifs associés (un mot, une phrase, un graphisme, une écriture, un geste ou tout un long discours).*

[12] Référence citée par Modeste :

algorithmique, et qu'il faut tenir compte de cette évolution pour reconsidérer cette forme de pensée. Pour ce chercheur, la différence essentielle réside dans le fait que la pensée algorithmique, vue comme pensée mathématique, *ne questionne pas l'efficacité des algorithmes qu'elle produit,* et *qu'elle n'utilise pas la notion de variable informatique*, ce qui renvoie à la notion de *complexité* d'un algorithme et à l'opération d'*affectation* (cf. §.4.3). Nous rejoignons Modeste quant à la nécessité de considérer une pensée algorithmique qui ne soit pas complètement incluse dans la pensée mathématique.

> Il nous semble donc que, pour aborder la pensée algorithmique, il soit indispensable d'appréhender simultanément les deux points de vue, intra-mathématique mais aussi extra-mathématique. (Modeste, 2012, p.53)

Il nous apparaît en effet, après avoir mené cette expérimentation, qu'ajouter un point de vue informatique (donc extra-mathématique) permet de prendre en considération une dimension qui n'existe pas en environnement mathématique usuel en papier-crayon, et qui induit un mode de pensée différent. L'existence de cette pensée spécifique nous semble être un élément didactique important qui permet de mieux comprendre la transposition qui se produit lors de la recherche d'algorithmes pour résoudre un problème.

REFERENCES

Artigue M. (1997) Le logiciel Dérive comme révélateur de phénomènes didactiques liés à l'utilisation d'environnements informatiques pour l'apprentissage. *Educational Studies in Mathetatics* 33(2), 133-169.

Artigue M. (2005) L'intelligence du calcul. Dans *Actes de l'Université d'été de Saint-Flour, France : le calcul sous toutes ses formes.*

Artigue M., Haspekian M. (2007) L'intégration de technologies professionnelles à l'enseignement dans une perspective instrumentale : le cas des tableurs. In Baron, Guin, Trouche (Eds.) *Environnements informatisés et ressources numériques pour l'apprentissage (*pp. 37-63). Paris : Hermès.

Balacheff N. (1994) Didactique et intelligence artificielle. *Recherches en didactique des mathématiques, 14*(1), 9-42.

Bosch M., Chevallard, Y. (1999) La sensibilité de l'activité mathématique aux ostensifs : objet d'étude et problématique. *Recherches en didactique des mathématiques* 19(1), 77-123. http://yves.chevallard.free.fr/spip/spip/IMG/pdf/Sensibilite_aux_ostensifs.pdf

Briant N. (2013) *Étude didactique de la reprise de l'algèbre par l'introduction de l'algorithmique au niveau de la classe de seconde du lycée français.* (Thèse de doctorat, université Montpellier 2).

Capponi B., Laborde C. (1995) *Cabri-classe, apprendre la géométrie avec un logiciel.* Archimède : Grenoble.

Chevallard Y. (1985) *La transposition didactique*. Grenoble : La pensée sauvage.

Chevallard Y. (1989) Le passage de l'arithmétique à l'algèbre dans l'enseignement des mathématiques au collège - Deuxième partie. Perspectives curriculaires : la notion de modélisation. *Petit x* n°19, 43-75.

Chevallard Y. (1999) L'analyse des pratiques enseignantes en théorie anthropologique du didactique. *Recherches en Didactique des Mathématiques, 19*(2), 221-266.

Duval R. (1993) Registres de représentation sémiotique et fonctionnement cognitif de la pensée. *Annales de didactique et de sciences cognitives. 5*. 37-65. IREM de Strasbourg

---

Knuth, D. E. (1985) Algorithmic thinking and mathematical thinking. The American Mathematical Monthly, 92 (1), 170-181. (Rééd. avec corrections (1996) Algorithms in Modern Mathematics and Computer Science. In Selected Papers on Computer Science, 87-114)

Duval R. (1995) S*émiosis et pensée humaine : Registres sémiotiques et apprentissages intellectuels.* Berne : Peter Lang.

Douady R. (1986) Jeux de cadres et Dialectique outil-objet. *Recherches en didactique des mathématiques*, *7*(2), 5-31.

Haspekian M. (2005) *Intégration d'outils informatiques dans l'enseignement des mathématiques. Étude du cas des tableurs.* (Thèse de doctorat, université Paris 7).

Laborde C. (2003) *The design of curriculum with technology : Lessons from projects based on dynamic geometry environments.* Reaction to A. Cuoco & P. Goldenberg's presentation "CAS and curriculum : Real improvement or déjà vu all over again ?" CAME Symposium, Reims.

MEN (2008) Ministère de l'éducation nationale. Programmes du collège. Programmes de l'enseignement de mathématiques. BO spécial n° 6 du 28 août 2008. Paris.

MEN (2009) MINISTÈRE DE L'ÉDUCATION NATIONALE. PROGRAMME DE MATHÉMATIQUES DE LA CLASSE DE SECONDE GÉNÉRALE ET TECHNOLOGIQUE. BULLETIN OFFICIEL N° 30 DU 23 JUILLET 2009.

Modeste S. (2012) *Enseigner l'algorithme pour quoi ? Quelles nouvelles questions pour les mathématiques ? Quels apports pour l'apprentissage de la preuve ?* (Thèse de doctorat, Université de Grenoble).

Rabardel P. (1995) *Les hommes et les technologies. Approche cognitive des instruments contemporains*. Paris : Armand Colin.

Radford L. (2006) Algebraic Thinking and the Generalization of Patterns: A Semiotic Perspective. In S. Alatorre, J. L. Cortina, M. Sáiz, A. Méndez (Eds.), *Proceedings of the 28th Conference of the International Group for the Psychology of Mathematics Education, North American Chapter (*pp. 2-21). Mérida: Universidad Pedagógica Nacional.

Radford L. (2008) Iconicity and Contraction: A Semiotic Investigation of Forms of Algebraic Generalizations of Patterns In *Different Contexts*. *ZDM – The International Journal on Mathematics Education*. DOI 10.1007/s11858-007-0061-0.