BESOINS PROFESSIONNELS DU PROFESSEUR DE MATHÉMATIQUES POUR ENSEIGNER L'ALGORITHMIQUE ET LA PROGRAMMATION

Nicolas ROS[1]
SFR - AEF, Université Toulouse 2 Jean Jaurès (INSPE)

Résumé. Dans le système de l'enseignement secondaire en France, les professeurs de mathématiques semblent rencontrer des difficultés à exploiter toutes les fonctionnalités des objets de l'algorithmique et la programmation. Certains mettent en avant leur objectif de familiariser les élèves à la programmation. D'autres engagent leurs élèves à programmer pour rendre « ludiques » leurs cours. Pour les professeurs, il semblerait que ces objets ne soient conçus que comme des moyens pour enseigner les mathématiques et faire rencontrer la programmation. Dans cet article, nous cherchons à repérer plus précisément les besoins professionnels du professeur de mathématiques pour enseigner l'algorithmique et la programmation. Pour ce faire, nous analysons un recueil de productions des élèves professeurs de mathématiques à l'INSPE Toulouse Occitanie-Pyrénées. Puis nous dégageons des besoins du professeur de mathématiques d'une part en algorithmique et programmation pour enseigner et d'autre part en didactique pour problématiser, motiver, concevoir et diriger l'enseignement de l'algorithmique et la programmation.

Mots-clés. Algorithmique et programmation, enseignement des mathématiques, besoin praxéologique, praxéologie mathématique « mixte », pensée algorithmique.

Abstract. In the French secondary school system, mathematics teachers seem to have difficulties in exploiting all the functionalities of algorithmic and programming objects. Some of them emphasize their objective of familiarizing students with programming. Others engage their students in programming to make their lessons "fun". For teachers, it seems that these objects are only conceived as means to teach mathematics and to introduce programming. In this article, we seek to identify more precisely the professional needs of the mathematics teacher to teach algorithmics and programming. To do so, we analyse a collection of productions of student teachers of mathematics at INSPE Toulouse Occitanie-Pyrénées. Then, we identify some needs of the mathematics teacher on the one hand in algorithmics and programming to teach and on the other hand in didactics to problematize, motivate, design and direct the teaching of algorithmics and programming.

Keywords. Algorithmics and programming, mathematics education, praxeological need, "mixed" mathematical praxeology, algorithmic thinking.

Introduction

À l'institut national supérieur du professorat et de l'éducation Toulouse Occitanie-Pyrénées (INSPE TOP), en première année du master MEEF[2] second degré, parcours mathématiques[3], les étudiants qui observent une utilisation d'outils logiciels en collège ou en lycée analysent difficilement les fonctions didactiques relatives à cette utilisation ou à ce qu'elle pourrait être. Certains de ces étudiants pensent :

- qu'un professeur gagne à faire programmer ses élèves pour dynamiser son cours ou le rendre plus « ludique » ;

[1] nicolas.ros@univ-tlse2.fr

[2] Le master Métiers de l'enseignement, de l'éducation et de la formation (MEEF) a été mis en place à la rentrée 2013, lors de la création des écoles supérieures du professorat et de l'éducation (ESPE) qui ont précédé les INSPE.

[3] Dans la suite, nous parlons de M1 MEEF mathématiques pour la 1re année et de M2 MEEF mathématiques pour la 2de année.

- qu'un élève doit apprendre à « manipuler » un outil de programmation en salle informatique avant de l'utiliser en résolution de problèmes.

Cependant, pour enseigner l'algorithmique et la programmation, qu'un professeur de mathématiques choisisse de faire programmer ses élèves en salle informatique ou non, il reste confronté à la même difficulté : définir les apprentissages qu'il vise pour ses élèves et choisir une organisation de son enseignement.

Dans cet article, nous questionnons les pratiques d'enseignement de l'algorithmique et la programmation, en collège et dans les voies générale et technologique du lycée, pour dégager quelques éléments de réponse à la question « Quelles sont les difficultés du professeur[4] de mathématiques à enseigner l'algorithmique et la programmation ? ». Nous avons choisi pour terrain de recherche le dispositif de formation à l'algorithmique et programmation mis en place en master MEEF mathématiques à l'INSPE TOP ; ce dispositif et son utilisation sont exposés dans la partie 2. Plus précisément, dans la suite, nous présentons : les éléments de la théorie anthropologique du didactique (TAD) nécessaires pour notre recherche et utilisés par l'ensemble des formateurs et élèves professeurs[5] observés ; une extension de ces éléments théoriques propre à l'étude de l'algorithmique et la programmation ; le dispositif de formation étudié, son écologie et le modèle de référence choisi ; l'analyse d'un corpus de quelques-unes des productions des élèves professeurs du terrain observé et les résultats qui en découlent pour cette recherche.

1. Problématique et question de recherche

1.1. Cadre théorique de référence

Parmi les nombreux gestes professionnels que le professeur de mathématiques accomplit, celui qui est sa raison d'être est *« mettre en place, dans une classe de collège ou de lycée, une certaine organisation de savoir « mathématique » »* (Chevallard, 2002a). Pour le réaliser, il effectue des analyses *a priori* et *a posteriori* d'« organisations de savoir ». Rappelons ce que nous entendons par « organisation de savoir ».

Praxéologies disciplinaires et didactiques

En TAD, *« un postulat fondateur [...] est que toute activité humaine peut se décomposer en une succession de tâches t_1, t_2, ..., t_n de certains types T_1, T_2, ..., T_n »* (Chevallard, 2017, p. 33). Pour mettre en mots un *type de tâches*, on a recourt à un verbe[6] et un déterminatif car un seul verbe, par exemple « appeler », désigne un genre de tâches (Chevallard, 1999) qui requiert un déterminatif : s'agit-il d'appeler une fonction ou le professeur ? Accomplir un type de tâches T dans une institution donnée[7] consiste à mettre en œuvre une certaine technique τ, justifiée par

[4] Dans cet article, nous écrivons « le professeur de mathématiques » — parfois abrégé « le professeur » — afin de désigner « la position institutionnelle de professeur de mathématiques ».

[5] Pour leur formation, les étudiants sont invités à s'emparer durant les deux ans du master MEEF mathématiques de quelques éléments de la TAD lors d'une formation conçue à partir des besoins de la profession de professeur de mathématiques. Dans la suite de l'article, nous écrivons « l'élève professeur » pour désigner « la position institutionnelle d'élève professeur » et « le formateur en mathématiques » — abrégé « le formateur » — afin d'indiquer « la position institutionnelle de formateur en mathématiques ».

[6] Parfois, nous utilisons plusieurs verbes, par exemple pour le type de tâches « écrire, mettre au point et exécuter un programme » ; selon les besoins, un tel type de tâches peut être analysé comme l'amalgame de types de tâches.

[7] En TAD, une institution I est un dispositif social qui permet — et impose — à ses sujets (personnes y occupant

une technologie θ qui permet de la penser, parfois aussi de la produire, et qui à son tour est justifiée par une théorie Θ. Toute technique ou technologie est destinée à être mise en mots afin d'en favoriser sa compréhension et sa diffusion. En général, on distingue les blocs [T / τ] correspondant au « savoir-faire » et [θ / Θ] correspondant au « savoir » en les appelant respectivement la *praxis* et le *logos*. Ainsi, toute activité humaine peut être modélisée par une praxéologie, notée [*T* / τ / θ / Θ], dont la (re)construction, par exemple en classe de mathématiques, constitue une organisation de savoir qu'on nomme organisation praxéologique. Lorsqu'une praxéologie est réalisée lors de l'étude propre aux mathématiques, on parle de praxéologie mathématique et d'organisation mathématique (abrégé OM) pour l'organisation praxéologique correspondante.

De fait, certaines organisations mathématiques ne relèvent pas exclusivement de l'étude des mathématiques « pures ». Chevallard (2002b) a défini une *organisation mathématique « mixte »* — correspondant à une *praxéologie mathématique « mixte »* — comme une organisation praxéologique certes mettant en jeu des objets mathématiques mais aussi quelques objets non mathématiques. Nous rencontrerons de telles praxéologies dans les parties 2 et 3, en particulier quand la mise en œuvre de la technique requiert l'usage d'objets informatiques.

Par ailleurs, mettre en place l'étude d'une organisation praxéologique particulière, c'est accomplir une activité qui peut être modélisée par une praxéologie qu'on nomme *praxéologie didactique* ; la mise en forme d'une telle praxéologie didactique constitue alors une *organisation didactique* (abrégé OD). Certes, en tant qu'étude, la (re)construction d'une praxéologie peut s'effectuer de diverses façons. Néanmoins, à certains moments d'une telle étude, on peut toujours identifier les mêmes *types de situations* qu'on appelle les *moments de l'étude* ou moments didactiques ; ces derniers caractérisent les gestes de l'étude récurrents au sein de chaque organisation didactique (Chevallard, 1999). La réalisation d'épisodes relevant de ces moments de l'étude peut notamment concerner le *moment de première rencontre* avec le type de tâches *T*, le moment d'exploration du type de tâches *T* et de l'émergence de la technique τ ou *moment exploratoire*, le moment de l'émergence de la technologie θ et de sa théorie Θ ou *moment technologico-théorique*, le *moment de travail* de la technique τ d'une organisation praxéologique [*T* / τ / θ / Θ], le *moment d'évaluation* d'une organisation praxéologique [*T* / τ / θ / Θ] et de sa maîtrise.

Le modèle praxéologique et le modèle des moments de l'étude nous permettront de réaliser des analyses en répondant aux questions « Comment est-il prévu que soient ou ont été/auraient pu être accomplies les tâches *t* relevant d'un certain type de tâches *T* ? » et « Comment est-il prévu que soit ou a été/aurait pu être mise en place l'étude de telle organisation praxéologique ? ».

Besoins praxéologiques

En TAD, lorsqu'on juge que le rapport à un objet *o* d'une instance observée — le formateur ou le professeur de mathématiques pour notre recherche — (Chevallard, 2021, p. 97) n'est pas conforme au rapport à l'objet *o* qu'on souhaiterait voir exister, Michèle Artaud (2021) considère qu'on évalue un *besoin* de l'instance observée. C'est le cas si on estime que cette dernière ne connaît pas l'objet *o* ou qu'elle réalise des actions en lien avec l'objet *o* d'une façon qu'on estime ne pas être adéquate. En TAD, le rapport de toute instance à un objet *o* étant la « résultante » de l'ensemble des praxéologies qui mettent en jeu l'objet *o* parmi celles connues

diverses positions offertes par *I*) la mise en jeu de manière de faire et de penser propres. Dans cet article, nous observons diverses institutions, en particulier une classe de mathématiques en collège ou en lycée et un dispositif de formation d'élèves professeurs en INSPE.

de l'instance (Chevallard, 2021, p. 97), le besoin de l'instance observée s'examine à partir de la notion de praxéologie : on évalue un besoin en une praxéologie (didactique, mathématique, etc.).

Par suite, notre question se reformule en la question de recherche Q : « De quelles praxéologies professionnelles le professeur de mathématiques a-t-il besoin pour enseigner l'algorithmique et la programmation ? ». Dans la section suivante, nous détaillons les praxéologies professionnelles étudiées pour cette recherche. Au sujet des « praxéologies pour la profession », nous invitons le lecteur désireux de précisions à lire ce qu'ont écrit Cirade (2017) et Chevallard et Cirade (2010).

Activités algorithmique et de programmation

Pour notre recherche, en nous référant au document ressources *Algorithmique et programmation pour le lycée* (Ministère de l'éducation nationale [MEN], 2017), un *algorithme* est une procédure de résolution qui s'applique *« à une famille d'instances d'un problème et produit, en un nombre fini d'étapes constructives, effectives, non-ambigües et organisées, la réponse au problème pour toute instance de cette famille »* ; cette définition est essentiellement celle de Donald Knuth (1997, pp. 4-6). D'autre part, un *programme* (informatique) est une *« méthode opérationnelle »*[8] (Knuth, 1997, p. 5) c'est-à-dire une procédure écrite en langage de programmation qui a les caractéristiques d'un algorithme mais qui peut ne pas se terminer en un nombre fini d'étapes.

Pour le thème *Algorithmique et programmation*, le curriculum prescrit[9] en mathématiques pour le collège et le lycée accorde une place importante à la programmation. Cependant, l'algorithmique y a aussi une place réservée. Il est attendu que *« les élèves s'initient à la programmation »* et *« développent des méthodes de programmation »* mais aussi qu'ils acquièrent *« des méthodes qui construisent la pensée algorithmique »* puis qu'ils traitent au moyen de l'algorithmique *« les problèmes [...] en relation avec les autres parties du programme »*. Les attendus sont principalement : pour le cycle 3 (élèves de 9-12 ans), *« programmer les déplacements d'un robot ou ceux d'un personnage sur un écran en utilisant un logiciel de programmation »* ; pour le cycle 4 (élèves de 12-15 ans), *« écrire, mettre au point et exécuter un programme simple »* voire le modifier ou le compléter (les autres attendus sont relatifs à des savoirs informatiques : notions d'algorithme, de programme et de variable informatique à intégrer dans un programme ; séquences d'instructions ; notion en acte de *« bloc-utilisateur »* propre au logiciel *Scratch* ; notion en acte de programmations événementielle, parallèle et objet) ; pour le lycée (élèves de 15-18 ans), décrire un algorithme en langage naturel ou dans un langage de programmation, réaliser un algorithme à l'aide d'un programme simple écrit dans un langage de programmation textuel et interpréter (lire, comprendre) un algorithme, le modifier ou le compléter (de nombreux autres attendus concernent des savoirs informatiques : type des variables ; notion de fonction informatique ; *« affectation (notée ← en langage naturel) »* ; etc.) Le curriculum prescrit expose des *« exemples d'algorithmes »* — par exemple, pour *« déterminer par balayage un encadrement de $\sqrt{2}$ d'amplitude inférieure ou égale à 10^{-n} »* — ou suggère parfois le recours à l'algorithmique — par exemple, pour le thème *Espace et géométrie* du cycle 4, l'élève doit savoir *« mettre en œuvre ou écrire un protocole de construction d'une figure géométrique »*.

Enfin, en cycle 4, *« la maîtrise des langages informatique* [sic] *[...] est le moyen d'acquérir d'autres démarches d'investigation, d'autres modes de résolution de problèmes, de simulation*

[8] Traduction française de « computational method ». Knuth précise *« A procedure that has all of the characteristics of an algorithm except that it possibly lacks finiteness may be called a* computational method *»*.

[9] Pour plus de détails, voir le site : https://eduscol.education.fr/90/j-enseigne-au-cycle-4?menu_id=75

ou de modélisation » (Ministère de l'éducation nationale, de l'enseignement supérieur et de la recherche [MENESR], 2016) et en lycée, *« la mise en place d'activités relevant de l'algorithmique et la programmation ne doit pas faire perdre de vue leur objectif de formation mathématique »* (Ministère de l'éducation nationale et de la jeunesse, 2019) : pour le professeur de mathématiques, l'étude de l'algorithmique et la programmation a pour visée celle des mathématiques, sans nécessairement s'y restreindre.

Considérons une institution où on étudie — possiblement un instant — l'algorithmique et la programmation pour représenter et traiter de l'information en mathématiques ; par exemple, une classe de mathématiques en cycle 4 où on souhaite exécuter un programme pour déterminer les nombres premiers inférieurs ou égaux à 100. Pour le professeur de mathématiques enseignant dans cette institution, faire étudier l'algorithmique, c'est d'abord faire rencontrer une activité algorithmique[10], soit un dispositif d'étude en algorithmique. Quand une praxéologie est réalisée lors d'une telle étude, on parle de *praxéologie algorithmique* et d'*organisation algorithmique* (abrégé OA) pour l'organisation praxéologique correspondante. De même, lorsque le professeur fait rencontrer une activité de programmation c'est-à-dire un dispositif d'étude en programmation, il fait étudier la programmation. Quand une praxéologie est réalisée lors d'une telle étude, on parle de *praxéologie de programmation* et d'*organisation de programmation* pour l'organisation praxéologique correspondante. Certes, si un programme *prog* donné est l'écriture en langage de programmation d'un algorithme, la tâche t : « exécuter le programme *prog* » appartient au type de tâches « mettre en œuvre un algorithme » (dans un environnement de programmation). Mais si le programme $prog_{Syr}$ a pour rôle de calculer le temps de vol pour la suite de Syracuse[11], du fait de la conjecture de Collatz, ce programme n'est pas l'écriture en langage de programmation d'un algorithme et la tâche t' : « exécuter le programme $prog_{Syr}$ » n'appartient pas au type de tâches « mettre en œuvre un algorithme ». Ainsi, une praxéologie de programmation n'est pas une praxéologie algorithmique bien que certaines de ses tâches puissent relever de l'étude de l'algorithmique. D'autre part, une praxéologie algorithmique n'est pas non plus une praxéologie de programmation : la tâche algorithmique « écrire le protocole de construction d'un triangle équilatéral » n'appartient pas à un type de tâches de programmation lorsqu'elle doit être réalisée en mode débranché.

Néanmoins, dans une institution où une praxéologie algorithmique est étudiée, la pensée algorithmique est à l'œuvre. Modeste (2012) a circonscrit cette pensée *« non seulement comme une pensée mathématique mais aussi comme la pensée informatique »* ; elle s'exprime dans le registre de la langue naturelle, du pseudo-code (langage naturel en notant ← l'affectation) ou d'un langage de programmation. Plus précisément, la pensée algorithmique — en tant que « pensée (ou activité) en algorithmique » — est une subsomption de toutes les activités algorithmiques qui peuvent être caractérisée comme suit :

> *L'activité algorithmique ne se résume donc pas à l'apprentissage et la mise en œuvre d'algorithmes mais englobe aussi leur production, leur compréhension et leur étude. […] En mathématiques, l'activité algorithmique peut se définir comme une étape dans la résolution d'une certaine catégorie de problèmes dans laquelle on recherche un procédé systématique et effectif de résolution* (Modeste, 2012, p. 473).

En TAD, le rapport d'un individu (Chevallard, 2021, p. 97) à l'algorithmique est donc engendré par le complexe des praxéologies algorithmiques *« produire un algorithme »*, *« constituer un*

[10] Les activités humaines faisant étudier l'algorithmique pour résoudre des problèmes — y compris dans le cadre d'un contrôle ou d'une vérification, mais aussi pour simuler un phénomène ou en proposer un modèle.

[11] Pour plus de détails, par exemple, voir : https://fr.wikipedia.org/wiki/Conjecture_de_Syracuse

répertoire d'algorithmes » (en particulier, les *« exemples d'algorithmes »* mentionnés dans le curriculum prescrit pour le lycée), *« mettre en œuvre un algorithme »*, *« comprendre un algorithme »* et *« étudier un algorithme »*. Les tâches de ces praxéologies algorithmiques peuvent être analysées comme appartenant à des types de tâches de programmation quand l'algorithme étudié soit est fourni, soit doit être produit en langage de programmation ; nous rencontrerons une telle tâche algorithmique dans la partie 2.2. De plus, le rapport d'un individu à l'algorithmique et programmation est au moins engendré par le complexe de ces praxéologies algorithmiques et par le complexe des praxéologies de programmation.

Pour notre recherche, les praxéologies professionnelles étudiées sont :

- les praxéologies algorithmiques à enseigner ;
- les praxéologies didactiques pour mettre en place l'étude d'une praxéologie algorithmique ;
- les praxéologies algorithmiques pour enseigner, utiles au professeur pour concevoir et construire ces praxéologies didactiques.

Les besoins du professeur de mathématiques seront repérés en formation initiale pour enseigner l'algorithmique et programmation. Par suite, nous analyserons des praxéologies mathématiques « mixtes » et des tâches algorithmiques appartenant à des types de tâches algorithmiques ou de programmation.

1.2. État de la question dans la littérature

Les recherches antérieures ont dégagé au moins trois types de besoins du professeur de mathématiques pour enseigner l'algorithmique et la programmation : en ingrédients technologico-théoriques de praxéologies algorithmiques à enseigner, notamment révélés par Nguyen et Bessot (2003), par Briant (2013) puis par Strock et Artaud (2016) ; en praxéologies algorithmiques pour enseigner, particulièrement identifiés par Modeste et Ouvrier-Buffet (2011) ; en praxéologies didactiques pour l'enseignement de l'algorithmique et la programmation. Pour ces derniers besoins, Couderette (2016) a confirmé les hypothèses de Strock (2013) : il y aurait une *« difficulté pour la profession de mener deux axes de temps didactique « simultanément », l'un lié aux mathématiques et l'autre à l'algorithmique »*. Couderette a observé une professeure de mathématiques enseignant en classe de seconde : cette dernière manque de certains savoirs informatiques (affectation de variables, statut du signe « = », notions distinctes d'algorithme et de programme), évite le travail sur la notion de variable en mathématiques et en informatique et *« prend en charge l'avancée du savoir »*. Au sujet de l'organisation didactique mise en place par la professeure, Couderette (2016) écrit :

> *On peut alors comprendre que la place dévolue à l'algorithmique soit comprise comme une « mise en textes algorithmiques » de concepts mathématiques […] : l'enjeu est-il d'aborder des concepts mathématiques sous un autre angle, ou d'étudier des concepts algorithmiques ? [...] pour elle, il est important que les élèves « fassent des programmes qui leur servent » [...]. Cette manière d'aborder l'algorithmique en favorisant son versant informatique est elle aussi confortée par les documents officiels qui présentent presque tous les algorithmes codés dans un langage informatique* (Couderette, 2016, p. 289).

Couderette précise que des difficultés (hybridité de l'algorithmique, affectation, variable, signe « = », validation d'un algorithme) mentionnées par Knuth sont abordées dans le curriculum prescrit mais que *« nulle indication sur une technique d'étude d'un algorithme »* n'est proposée ; le curriculum prescrit ne semble satisfaire les besoins du professeur ni en savoirs informatiques — transposés en ingrédients de praxéologies algorithmiques pour enseigner — ni

en ingrédients de praxéologies didactiques pour l'enseignement de l'algorithmique et la programmation. En particulier, la professeure de mathématiques observée par Couderette semble rencontrer des difficultés à diriger l'étude lors de l'enseignement concomitant d'un curriculum de mathématiques et d'un curriculum d'algorithmique. Afin d'enseigner l'algorithmique et la programmation, le professeur de mathématiques aurait donc besoin de s'équiper de praxéologies didactiques pour diriger l'étude lors de l'enseignement concomitant des mathématiques et de l'algorithmique et programmation. Quid de ses besoins en praxéologies didactiques pour concevoir l'enseignement de l'algorithmique et la programmation ? Il aurait également des besoins en praxéologies algorithmiques à enseigner et pour enseigner.

Nous proposons donc d'apporter des éléments de réponse aux deux questions suivantes qui sont dérivées de notre question Q de recherche :

> $Q_{did-A\&P}$: « De quelles praxéologies didactiques le professeur de mathématiques a-t-il besoin pour enseigner l'algorithmique et la programmation ? »
>
> $Q_{algo-A\&P}$: « De quelles praxéologies algorithmiques le professeur de mathématiques a-t-il besoin pour enseigner l'algorithmique et la programmation ? »

2. Terrain de la recherche

Notre terrain de recherche est le dispositif de formation à l'algorithmique et programmation mis en place en master MEEF mathématiques à l'INSPE TOP chaque année universitaire entre 2018 et 2021. Il est rattaché à des dispositifs, déployés sur les quatre semestres du master et intitulés *Enseigner les mathématiques* (ELM), dans lesquels l'étude de quelques éléments de didactique des mathématiques — dont ceux présentés dans la partie 1.1 — est proposée ; l'objectif principal y est d'équiper les élèves professeurs pour qu'ils puissent analyser le curriculum prescrit, des scénarios didactiques, des manuels, etc., puis concevoir et diriger une étude.

Ce dispositif concerne quatre groupes composés de 16 à 18 élèves professeurs ; quatre formateurs l'encadrent, chacun étant responsable d'un groupe (ils enseignent aussi dans le dispositif ELM et encadrent les travaux de recherche des élèves professeurs). Pour chaque activité qui y est étudiée, après soit un travail en autonomie (en mode débranché et dans un environnement de programmation), soit l'accompagnement d'un formateur, des échanges dirigés par le formateur mènent à un bilan, avant une synthèse.

En M1 MEEF mathématiques, le dispositif consiste en trois séances de 3 *h* d'étude d'activités mathématiques et algorithmiques ; une séance est associée à un outil de programmation (calculatrices ou émulateur *Numworks*, *Scratch 3* et *Pyzo*[12] *3*). À la fin de chaque séance, à partir du curriculum prescrit et des documents ressources *Algorithmique et programmation* (MENESR, 2016 ; MEN, 2017 ; MENJ, 2019), une synthèse dégage les types de tâches algorithmiques et les savoirs informatiques à enseigner (variable informatique qui *« est une étiquette collée sur une boîte »* et qu'on trace, *« affectation (notée ← en langage naturel) »*, etc.) En M2 MEEF mathématiques, le dispositif se compose d'un module ELM3-MilieuNum de 6 *h* — agencé en deux séances, S1 et S2, de 3 *h* — et d'un module ELM3-Num de 6 *h*. La séance S1 est dédiée à la programmation en langage *Scratch* : des activités algorithmiques sont étudiées — dont celle présentée en figure 1 — pour dégager des praxéologies algorithmiques à enseigner ; à partir du document ressources *Algorithmique et programmation* (MENESR, 2016), une synthèse résume les fonctionnalités du logiciel *Scratch* et l'étude possible avec ce logiciel d'activités

[12] *Pyzo* est un environnement de développement de programmation en *Python*.

mathématiques intégrant l'algorithmique et la programmation ou d'activités algorithmiques (algorithmique comme outil ou objet). La séance S2, dédiée à la programmation en langage *Python*, est organisée comme la séance S1, en se référant aux documents ressources *Algorithmique et programmation* (MEN, 2017 ; MENJ, 2019). Le contenu de ce module sera précisé dans la partie 2.2. Le module ELM3-Num du dispositif est également la première partie d'un dispositif de formation au *numérique* de 24 *h* encadré par les mêmes formateurs ; il permet aux élèves professeurs de travailler les compétences B2.1 *« Identifier les situations d'apprentissage propices à l'utilisation des Tice »* et B2.2 *« Concevoir des situations d'apprentissage et d'évaluation mettant en œuvre des logiciels généraux ou spécifiques à la discipline, au domaine et niveau d'enseignement »* du référentiel du certificat informatique et internet niveau 2 *« enseignant »* (C2i2e) (Ministère de l'enseignement supérieur et de la recherche, 2011).

Cet ancrage numérique du dispositif pourrait contraindre le formateur et le chercheur : l'étude de l'enseignement de l'algorithmique et programmation pourrait sembler interrompue par la requête d'une conception de *« situations d'apprentissage »* avec et par le *numérique* ; seul l'aspect outil de l'algorithmique pourrait sembler visé, les intitulés des compétences B2.1 et B2.2 imposant une *« utilisation des Tice »* et de *« logiciels »*. Est-il possible d'atténuer ces deux contraintes ?

2.1. Le numérique, un moyen d'être pour l'algorithmique et la programmation

Les formateurs préfèrent substituer « numérique » à « Tice ». Qu'entendent-ils par *numérique*[13] ? Une « dimension du réel social coextensive à la réalisation d'une activité humaine sollicitant la numérisation de l'information ». Ils jugent que l'algorithmique et programmation, à la croisée des mathématiques et de l'informatique, fait partie du numérique et donc que le travail des compétences B2.1 et B2.2 s'inscrit dans la continuité de l'étude de son enseignement ; principalement, leur argumentation est fondée sur les propos suivants du conseil scientifique de la société informatique de France (SIF) :

> *L'adjectif* « numérique » *qualifie toutes les activités qui s'appuient sur la numérisation de l'information […] Si on numérise l'information, c'est pour la possibilité de la traiter avec toutes les méthodes de l'informatique. En ce sens, l'informatique est au cœur du numérique. […] Certains limitent les sciences du numérique à une liste plus ou moins exhaustive de domaines comme la robotique, l'automatique, les télécoms, les mathématiques appliquées, le traitement du signal, les circuits électroniques* (Conseil scientifique de la SIF, 2014, p. 16).

En tant que chercheur, nous connaissons les tensions actuelles entre numérique et informatique (Berry, 2019). Nous nous contentons donc du choix des formateurs d'esquisser aux élèves professeurs cette argumentation et d'expliciter la possibilité pour eux d'identifier des « situations » propices à l'utilisation de l'algorithmique lors du travail de la compétence B2.1. Ce choix nous semble minimaliser la première contrainte.

Concernant la seconde contrainte, les formateurs choisissent de demander aux élèves professeurs de choisir trois « situations » propices à l'utilisation du *numérique* voire, en particulier, de l'algorithmique ; ils précisent que, pour chaque « situation », il y a deux cas : soit le *numérique*, voire l'algorithmique, est un outil pour étudier une activité mathématique, soit l'algorithmique est l'objet de l'étude d'une activité algorithmique. Nous estimons que cette consigne amoindrit la seconde contrainte puisqu'elle mentionne la dialectique outil/objet de l'algorithmique.

[13] Le conseil national des universités possède une « section 27 - Informatique » mais pas de « section *numérique* ». Ces formateurs connaissent les missions de la direction du numérique éducatif (DNE) du ministère de l'éducation nationale. Voir : https://www.education.gouv.fr/direction-du-numerique-pour-l-education-dne-9983

Dans le module ELM3-Num, les formateurs demandent à un élève professeur qui a produit un scénario didactique d'analyser :

- en fonction des cas, soit la praxéologie mathématique « mixte » enjeu de l'étude et le type de tâches enjeu principal de l'étude du point de vue du *numérique* ou de l'algorithmique, soit la praxéologie algorithmique enjeu principal de l'étude ;
- la problématicité ou non, ainsi que les raisons d'être de l'enjeu principale de l'étude puis la conformité ou non de celui-ci aux attendus du curriculum prescrit ;
- la fonction du type de tâches enjeu principal de l'étude du point de vue du *numérique* ou de l'algorithmique par rapport à la praxéologie enjeu principal de l'étude ;
- pour l'enjeu principal de l'étude, la praxéologie didactique prévue en termes de moments de l'étude.

Ces éléments sont étudiés par les élèves professeurs au sein du module ELM3-MilieuNum présenté dans la section suivante.

2.2. Contenu du dispositif ELM3-MilieuNum

Nous spécifions maintenant : le contenu du module ELM3-MilieuNum ; le travail de préparation des formateurs pour une des activités travaillée en séance S1 ; suite à nos commentaires, l'adaptation du travail préparatoire des formateurs prévu pour la séance S2. Ces différents matériaux nous permettront de constituer notre modèle de référence (Chevallard, 2017, p. 51) pour réaliser l'analyse de notre recherche dans la partie 3.

Pour le module ELM3-MilieuNum, les formateurs ont constitué un corpus d'activités (une activité algorithmique et 11 activités mathématiques avec programmation requise — cinq en langage *Scratch* et six en langage *Python*) pour que les élèves professeurs puissent :

- utiliser l'équipement logiciel de la profession de professeur de mathématiques qui est seulement en partie décrit par la noosphère[14] du métier (quid de la version d'un logiciel ou de l'environnement de développement d'un langage de programmation ?) ;
- identifier les praxéologies algorithmiques à enseigner, en mode débranché ou non ;
- a) identifier le type de tâches de l'enjeu principal de l'étude du point de vue mathématique et celui du point de vue algorithmique puis analyser les praxéologies mathématiques « mixtes » et les praxéologies algorithmiques associées ;

  b) évaluer les activités du corpus en termes de : présence ou non de tâches problématiques à étudier ; raisons d'être et conformité ou non aux attendus du curriculum prescrit pour l'enjeu principale de l'étude ;

  c) analyser la fonction des types de tâches algorithmiques par rapport à l'enjeu principal de l'étude ;

  d) analyser une praxéologie didactique possible pour l'enjeu de l'étude, en particulier avec les moments de l'étude ;

  e) proposer des modifications éventuelles de ces activités.

Nous présentons ci-après une activité du corpus (*cf.* figure 1) utilisant l'exercice 2 du sujet de DNB[15] Métropole-La Réunion-Antilles-Guyane du 29 juin 2017) et le travail de préparation afférent des formateurs. La consigne est « Répondre aux questions de l'énoncé sans programmer

[14] La *noosphère* — la sphère « où l'on pense — d'une institution *I* désigne l'ensemble des institutions qui se vouent à « réfléchir » sur *I*, à critiquer, suggérer, impulser, entraver les changements qui affectent *I* ou pourraient l'affecter. La noosphère d'une institution donnée est indispensable à la vie de l'institution.

puis vérifier vos réponses en exécutant les programmes avec le logiciel *Scratch* ».

On donne le programme suivant qui permet de tracer plusieurs triangles équilatéraux de tailles différentes. Ce programme comporte une variable nommée « **côté** ». Les longueurs sont données en pixels.

On rappelle que l'instruction 

signifie que l'on se dirige vers la droite.

Numéros d'instruction	Script	Le bloc **triangle**
1	Quand est cliqué	définir triangle
2	effacer tout	stylo en position écriture
3	aller à x: −200 y: −100	répéter 3 fois
4	s'orienter à 90	avancer de côté
5	Mettre côté à 100	tourner de 120 degrés
6	répéter 5 fois	
7	triangle	relever le stylo
8	avancer de côté	
9	Ajouter à côté −20	

1. Quelles sont les coordonnées du point de départ du tracé ?
2. Combien de triangles sont dessinés par le script ?
3. a. Quelle est la longueur (en pixels) du côté du deuxième triangle tracé ?
   b. Tracer à main levée l'allure de la figure obtenue quand on exécute ce script.
4. On modifie le script initial pour obtenir la figure ci-contre. Indiquer le numéro d'une instruction du script **après laquelle** on peut placer l'instruction 

pour obtenir cette nouvelle figure.

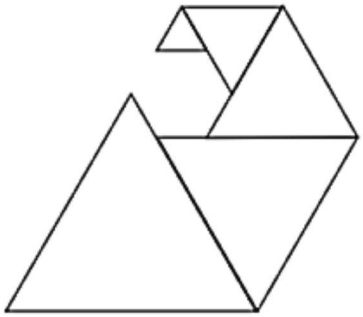


Figure 1 *: Extrait du corpus du module ELM3-MilieuNum.*

Pour les formateurs, seul l'environnement papier-crayon est disponible pour l'élève[16] (conditions de l'épreuve du DNB) et les trois premières questions ont pour fonction de lui permettre de comprendre le protocole de construction fourni par le programme. Par la suite, l'élève peut s'engager dans la part essentielle de son travail, en question 4 ; il doit y modifier le protocole de construction, soit en écrire un autre en modifiant celui fourni.

[15] Dans le système d'enseignement en France, le diplôme national du brevet (DNB) et le baccalauréat sont des évaluations nationales concernant respectivement les élèves de 14-15 ans et de 17-18 ans.

[16] Dans cet article, nous écrivons « l'élève » afin d'indiquer « la position institutionnelle d'élève ».

Enjeu de l'étude du point de vue mathématique et du point de vue algorithmique (a)

Pour cet exercice 2, les formateurs supposent donnée une classe de mathématiques en cycle 4 et analysent *a priori* ainsi une praxéologie mathématique « mixte » qui est enjeu principal de l'étude et une praxéologie algorithmique.

T_{math} : Écrire un protocole de construction d'une figure géométrique.

Les formateurs supposent que : les protocoles de construction fournis et attendus sont explicités sous forme de programmes ; chaque sous-figure élémentaire (en général, un cercle, un triangle — équilatéral ou rectangle et isocèle en un sommet —, un carré voire un rectangle) de la figure fournie est associée à une seule sous-figure élémentaire (de même nature géométrique) de la figure attendue et ces deux sous-figures élémentaires sont superposables.

τ_{math} : Décrire, interpréter et comprendre le protocole de construction fourni puis le mettre en œuvre dans l'environnement papier-crayon pour vérifier qu'il a correctement été interprété et compris. Repérer des sous-figures élémentaires et des invariants — géométriques, métriques et angulaires — de la figure fournie puis des schémas récurrents du protocole de construction fourni et les effets sur les sous-figures élémentaires repérées après chaque itération. Comparer la figure fournie et la figure attendue puis exprimer les éléments de comparaison en termes d'instructions. Modifier le protocole de construction fourni avec ces instructions pour que sa mise en œuvre produise la figure attendue. Mettre en œuvre dans l'environnement papier-crayon le protocole de construction obtenu pour vérifier qu'il réalise la figure attendue. Si ce n'est pas le cas, recommencer dès le début sinon le protocole de construction obtenu est le protocole de construction attendu.

θ_{math} :

$\theta_{math,\,algo}$: Un protocole de construction représente un algorithme. Un algorithme peut être écrit en langage naturel, par exemple sous forme de protocole de construction, et en langage de programmation, sous forme de programme. Tout environnement dans lequel est écrit ou exécuté un algorithme peut fournir des rétroactions qui permettent de tester et de corriger l'algorithme lorsqu'on le met en œuvre. Mettre au point un algorithme consiste à le tester (en le mettant en œuvre plusieurs fois, en testant successivement les instructions qui le composent, etc.) et à le corriger avant de le mettre en œuvre de nouveau pour vérifier qu'il produit ce qui est attendu. Dans un algorithme, un schéma récurrent permet d'itérer un certain nombre de fois une même séquence d'instructions.

$\theta_{math,\,géo}$: En général, les figures élémentaires sont des cercles, des triangles particuliers (équilatéraux, rectangles et isocèles en un sommet), des carrés voire des rectangles. On peut se référer à une définition et à des propriétés caractéristiques pour qualifier une figure, une transformation ou une relation entre figures : par exemple, deux triangles sont semblables si et seulement si les angles du premier sont égaux aux angles associés du second ou si et seulement si les longueurs des côtés du premier sont proportionnelles aux longueurs des côtés associés du second. Autant que possible, on se réfère à la définition d'une transformation — par exemple, un agrandissement d'une figure est l'image de cette figure, éventuellement déplacée, par une homothétie de rapport supérieur à 1 ; pour une réduction, le rapport (positif) est inférieur à 1 — et à des discours explicitant les effets d'une transformation sur une figure.

Pour la praxéologie algorithmique, enjeu principal de l'étude du point de vue de l'algorithmique, les formateurs l'analysent ainsi :

T_{algo} : Modifier un programme.

Pour les formateurs, le programme peut être modifié en y insérant une instruction donnée.

τ_{algo} : Écrire, interpréter et comprendre le programme fourni puis l'exécuter pour vérifier que le programme a correctement été interprété et compris. Comprendre l'instruction qui est donnée (les données à renseigner pour son exécution et son rôle). Repérer la ligne du programme où l'instruction donnée peut être insérée pour modifier le programme comme attendu puis y insérer l'instruction. Exécuter et mettre au point le programme (le tester et, si nécessaire, le corriger en changeant la ligne d'insertion de l'instruction ; recommencer jusqu'à ce que le programme produise ce qui est attendu).

θ_{algo} : Un programme peut être assimilé à un algorithme qui pourrait ne pas toujours se terminer en un nombre fini d'étapes et qui est écrit dans un langage de programmation. Tout environnement dans lequel est écrit et exécuté un programme fournit des rétroactions qui permettent de le tester et de le corriger. Mettre au point un programme consiste à le tester (en l'exécutant plusieurs fois, en testant successivement les différentes instructions, etc.) et à le corriger avant de l'exécuter de nouveau pour vérifier que ce qu'il produit est conforme à l'attendu.

Les formateurs précisent que « écrire, interpréter et comprendre un programme » et « exécuter un programme » sont aussi des types de tâches algorithmiques rencontrés mais qu'aucun ne constitue l'enjeu principal de l'étude du point de vue algorithmique.

Tâches problématiques, raisons d'être et conformité au curriculum prescrit (b)

Les formateurs estiment qu'aucune raison d'être ne motive l'étude. En outre, le type de tâches T_{math} est une « compétence associée » du thème *Espace et géométrie* du curriculum prescrit pour le cycle 4 et le type de tâches T_{algo} est indiqué dans le document ressources *Algorithmique et programmation* pour le cycle 4 (MENESR, 2016) : pour les formateurs, l'enjeu de l'étude pour cet exercice est donc conforme aux attendus du curriculum prescrit.

Fonction du type de tâches algorithmique par rapport à l'enjeu principal de l'étude (c)

Le type de tâches T_{algo} est celui auquel appartient la tâche « modifier le script initial » ; l'élève accomplit cette tâche pour mettre en œuvre la technique τ_{math} permettant d'accomplir le type de tâches T_{math} : l'algorithmique est donc vue comme un outil.

Praxéologie didactique possible pour l'enjeu de l'étude (d)

Les formateurs attendent que les élèves professeurs précisent le rapport supposé à l'algorithmique de l'élève d'une classe de mathématiques avant l'étude pour expliciter une praxéologie didactique possible : si l'élève en sait très peu, un *moment d'institutionnalisation* pourrait être visé, sinon des *moments de travail ou d'évaluation* pourraient être proposés.

Outre l'analyse du type de tâches enjeu principal de l'étude algorithmique, et une proposition de praxéologie didactique possible, les formateurs attendent des élèves professeurs qu'ils analysent : le type de tâches associé à la praxéologie mathématique « mixte » enjeu principal de l'étude ; suffisamment de ses éléments techniques pour pouvoir l'accomplir ; la plupart de ses éléments technologiques relevant de l'algorithmique ; les liens fonctionnels (raison d'être, outil) entre les diverses praxéologies. Les formateurs estiment que le travail effectué au sein du module

ELM3-MilieuNum permet aux élèves professeurs de produire un scénario didactique et une analyse comme attendu dans le cadre du module ELM3-Num.

Suite à notre observation de la séance S1, les formateurs nous ont demandé des commentaires. Nous en exposons quelques-uns : le programme de l'énoncé présenté est le seul du corpus qui présente une numérotation des différentes étapes de l'algorithme ; le *logos* de l'élève professeur pourrait s'enrichir d'ingrédients technologico-théoriques pour comprendre un programme : en particulier, une définition d'un algorithme ; un discours rappelant que, dans un algorithme, attribuer un nom explicite aux variables ainsi que commenter et spécifier des lignes permet d'en faciliter la lecture ; pour mettre en œuvre un algorithme, on gagne à exploiter une *table de traces*[17] (Knuth, 1997 ; Strock & Artaud, 2016) et, si l'algorithme produit une figure géométrique complexe, une *bande dessinée algorithmique* (suite exhaustive de sous-figures produites par l'exécution de séquences d'instructions de l'algorithme) — par exemple, pour modifier l'algorithme étudié dans cette activité, elle pourrait aider l'élève à « repérer la ligne du programme où l'instruction donnée peut être insérée » après chaque passage dans la boucle « répéter ». Suite à ces commentaires, les formateurs ont adapté leur travail préparatoire pour la séance S2 et y ont fait travailler ces éléments aux élèves professeurs.

Enfin, nous n'avons pas mentionné aux formateurs le fait suivant. Concernant l'enjeu principal de l'étude du point de vue algorithmique, en analysant le type de tâches « modifier un programme », les formateurs interprètent la tâche « modifier le protocole de construction fourni » du point de vue de la programmation. Du point de vue algorithmique, il aurait été préférable d'analyser le type de tâches « modifier un algorithme ». Ici, l'algorithme étant représenté par un programme, la tâche algorithmique est analysée comme appartenant à un type de tâches de programmation, comme nous l'avons mentionné dans la partie 1.1.

3. Des éléments de réponse à la question de recherche

3.1. Élaboration du corpus de la recherche

Dans le module ELM3-Num, les élèves professeurs commencent à produire un scénario et une analyse comme attendu par les formateurs (*cf.* partie 2.1). Après la séance S2, lors de deux séances de 3 *h* (les trois séances étant espacées d'un mois), un élève professeur poursuit — et termine un mois après la seconde séance — son travail de conception et d'analyse, en échangeant avec les élèves professeurs de son groupe et son formateur. Il est attendu qu'un élève professeur mette en place dans une des classes en responsabilité ce scénario didactique ; l'élève professeur est donc aussi accompagné par son tuteur de terrain, professeur de mathématiques.

Parmi les scénarios didactiques produits par les élèves professeurs entre 2018 et 2021, nous n'observons que ceux concernant l'enseignement de l'algorithmique et la programmation. Dans la suite, nous désignons par *production A&P* et *analyse A&P* respectivement un scénario didactique et une analyse attendue des formateurs pour ce scénario.

[17] La *table de traces* (*algorithm trace table* en anglais, l'expression « tableau de fonctionnement » est aussi employée) d'un algorithme est un tableau qui, pour chaque instruction présente dans la structure entrée/traitement/sortie d'un algorithme, affiche les valeurs des variables, des prémisses relatives à une instruction conditionnelle ou à un test d'arrêt relatif à une instruction itérative du type « Tant que … Faire … » ou « Répéter … jusqu'à … », ou des chaînes de caractères à afficher.

Nous avons préparé un recueil de 107 *productions A&P* en collectant : en 2018-2019, 62 productions dont 34 *productions A&P* (le logiciel *Scratch* est présent dans 22 productions, la programmation en langage *Python* dans 10 et le logiciel *Algobox* dans 2) ; en 2019-2020, 61 productions dont 28 p*roductions A&P* (le logiciel *Scratch* est présent dans 16 productions — dont une qui exploite aussi la plateforme *Heure de code*[18] —, la programmation en langage *Python* dans 12) ; en 2020-2021, 69 productions dont 45 p*roductions A&P* (le logiciel *Scratch* est présent dans 29 productions et la programmation en langage *Python* dans 16). Toutes les praxéologies enjeu de l'étude présentées dans ces 107 *productions A&P* sont des praxéologies mathématiques « mixtes ». À partir de ce recueil, nous avons constitué un corpus de neuf de ces *productions A&P*. D'abord, nous avons trié les 107 scénarios : neuf scénarios spécifiques à notre recherche, pour lesquels il faut écrire un algorithme en langage courant (pas de production), réaliser des activités en mode débranché (quatre productions) ou comprendre un algorithme (cinq productions) et les 98 autres. Ensuite, nous avons classé ces 98 *productions A&P* restantes par thème puis par sujet du curriculum prescrit. Pour un sujet, nous avons retenu les *productions A&P* dont l'*analyse A&P* était plutôt conforme aux attendus des formateurs (voir partie 2.2) même si elle était en partie erronée ; la plupart des *productions A&P* écartées présentent une activité mathématique et une description de son travail en classe pour l'analyse. Chaque année, les quatre formateurs du dispositif évaluant les *productions A&P*, nous leur avons soumis ce premier tri pour aboutir à une sélection consensuelle. Enfin, aussi en considérant l'évaluation des formateurs, nous avons sélectionné huit *productions A&P* selon deux critères : l'*analyse A&P* n'est pas trop défaillante par rapport aux attendus ; l'activité du scénario proposé est connue du professeur de mathématiques (le curriculum prescrit, les documents ressources ou les manuels scolaires la présentent). Pour constituer notre corpus, nous avons voulu aussi conserver un des neufs scénarios spécifiques — dans la suite, désigné *production A&P* n° 8 — pour lequel il faut comprendre un programme car l'*analyse A&P* nous semble acceptable par rapport aux attendus.

Un formateur et un tuteur de terrain ayant accompagné un élève professeur dans son travail, nous analysons ce recueil et ce corpus pour réaliser notre recherche. Nous pensons que l'analyse des besoins de l'élève professeur peut contribuer à comprendre les besoins du professeur. Par ailleurs, nous savons que, dans une formation, les *« difficultés auxquelles un individu particulier [...] déclare être confronté constituent en fait un révélateur des difficultés auxquelles est confronté le collectif des élèves professeurs et, par-delà, la profession »* (Cirade, 2017, p. 732).

3.2. Analyse du corpus de la recherche

Description des analyses A&P du corpus

De notre point de vue de chercheur, nous commençons par présenter une synthèse des neuf *analyses A&P* du corpus dans le tableau 1.

Pour ces *productions A&P* du corpus, l'exploitation de l'algorithmique et la programmation intervient lors de l'émergence ou la mise en œuvre d'une technique τ_{math}. Cette exploitation est estimée peu pertinente si l'élève dispose d'une autre technique qui est jugée équivalente à la technique τ_{math} (Chevallard, 1999, pp. 259-261) et dont la mise en œuvre ne sollicite pas l'algorithmique et la programmation.

[18] Pour plus de détails, voir le site : https://code.org/learn

n°		T_{math} enjeu principal	T_{algo} enjeu principal	Mise en mots de τ_{algo}	Algorithmique : outil	Algorithmique : objet	Pertinence	Moments de l'étude (OD)
1	T^le S	Résoudre une équation du type $f(x)=k$.	Réaliser un algorithme à l'aide d'un programme.	Non	Oui	Non	Oui	Moment exploratoire
2	4e	Déterminer la valeur en entrée d'un programme de calcul connaissant la valeur en sortie.	Écrire, mettre au point et exécuter un programme.	Non	Oui	Oui	Non	Moment de travail
3	4e	Construire une frise.	Écrire, mettre au point et exécuter un programme.	Oui	Oui	Non	Non	Moment de travail
4	2de	Déterminer si un nombre est parfait ou non.	Écrire, mettre au point et exécuter un programme.	Oui	Oui	Non	Oui	Moment exploratoire
5	4e	Déterminer la valeur en sortie d'un programme de calcul connaissant la valeur en entrée.	Compléter et exécuter un programme.	Oui	Oui	Non	Non	Moment de travail
6	T^le S	Déterminer un seuil pour une suite.	Compléter et exécuter un programme.	Oui	Oui	Non	Oui	Moment de travail
7	2de	Déterminer un encadrement de $\sqrt{2}$ d'amplitude donnée.	Compléter, modifier, mettre au point et exécuter un programme.	Oui	Oui	Non	Oui	Moment exploratoire
8	2de	Déterminer les extremums d'une fonction sur un intervalle.	Comprendre, interpréter et exécuter un programme.	Non	Oui	Non	Oui	Moment exploratoire
9	2de	Déterminer un encadrement de $\sqrt{3}$ (à 10^{-5} près).	Compléter un algorithme. Exécuter un programme.	Non	Oui	Non	Oui	Moment exploratoire

Tableau 1 *: Description synthétique des neuf* analyses A&P.

En vue de leur analyse ultérieure, nous détaillons dans les figures 2 à 5 les *analyses A&P* n° 1, n° 2 et n° 4.

TVI – Algorithme de dichotomie

Soit la fonction f définie sur l'intervalle I = [-4 ; 4] par $f(x) = x^3 - 6x + 1$.

Partie I

1. Justifier que f est dérivable sur I et calculer $f'(x)$ pour x appartenant à I.
2. Tracer le tableau de variations de f.
3. Montrer que l'équation $f(x) = 0$ admet une unique solution α sur l'intervalle [0 ; 1].

Partie II : un peu de programmation

1. Visionner la vidéo https://www.youtube.com/watch?v=V7mlMCSrq1U
2. Déterminer cette solution grâce à un algorithme de dichotomie.
3. Programmer grâce au logiciel Python.

n°1 – Support de la situation didactique

Figure 2 *: Situation n° 1 - Support de la situation didactique.*

Algorithmique :
Créer une fonction sur Python : (définir la fonction mathématique à étudier ; créer une fonction reproduisant la méthode de dichotomie, avec pour entrées un intervalle et une valeur seuil,
Calculs itérés grâce aux boucles « Tant que » et « Si ».
Utilisation d'une valeur seuil pour encadrer la solution.

•**La plus-value des TICE** : La quantité de calculs et les décision binaires (SI/SINON) est plus rapide à traiter par informatique qu'à la main.

OD : La cas technique :
Le moment exploratoire se déroule à la Partie 2. En effet, l'élève , en connaissant l'algorithme de dichotomie en pseudo-code, doit le traduire en langage Python par de nombreuse phases de test pour vérifier que chaque instruction fonctionne correctement. L'élève confronte son programme à différentes valeurs d'entrées et valide ou non la cohérence des résultats retournés.

Figure 3 *: Situation n° 1* - Analyse A&P.

La plus value des TICE : Technique

- Dans cette activité, les résultats auraient pu être trouvés grâce à des calculs effectués à la main. Cependant le logiciel Scratch présente ici deux avantages non négligeables : il permet un gain de temps puisque lorsque le programme a été créé, il suffit de changer le nombre de départ sans réfléchir à toutes les étapes de calculs ; et une fiabilité au niveau de la fiabilité puisque, contrairement aux élèves, les erreurs de calculs ne seront pas un frein à al résolution des problèmes.
- De plus, le logiciel Scratch est un logiciel qui suscite l'intérêt des élèves puisqu'ils considèrent ici la programmation comme un « jeu ». Les fonctionnalités proposées permettent à l'élève de créer et de se familiariser petit à petit avec les principes de la programmation.

Figure 4 *: Situation n° 2* - Analyse A&P*, un extrait.*

Question : 8128 est-il un nombre parfait ?

Figure 5 *: Situation n° 4 - Support de la situation didactique.*

On est ici dans un cas technique.

Tâche t : déterminer si le nombre 8128 est un nombre parfait.

Types de tâches T :

- **mathématique : déterminer si un nombre entier naturel N est un nombre parfait.**
- **numérique : écrire un programme en Python.**

Technique :

[...]

- **Pour le type de tâches numérique : (on considère que les diviseurs positifs de N sont à déterminer)**
  Ecrire une instruction qui permet à l'utilisateur de choisir un nombre (instruction Input()) et de stocker cette valeur dans une variable N de type entier (int()).
  Initialiser la variable Somme à 0 (la variable Somme aura pour valeur, à la fin du programme, la somme des diviseurs de N).
  Etablir une boucle qui va calculer Somme, la somme des diviseurs de N : pour chaque nombre entier compris entre 1 et N (boucle « for Nombre in range (n,m,p) : »), calculer le reste de la division euclidienne de N par Nombre (opération %). Si ce reste est nul, le nombre considéré est un diviseur de N, alors l'ajouter à Somme (instruction conditionnelle « if condition : »). A la sortie de la boucle, Somme contient la valeur de la somme des diviseurs de N.
  Calculer la moitié de Somme. Si le résultat est égal à N, afficher « N est un nombre parfait » (fonction print()) sinon afficher « N n'est pas premier » (instruction conditionnelle « if condition : // else : »).

Figure 6 *: Situation n° 4* - Analyse A&P.

Analyse des analyses A&P : le point de vue de la question $Q_{algo-A\&P}$

L'analyse des *analyses A&P* du corpus met en exergue des besoins du professeur en ingrédients technologiques pour enseigner : ce que sont et ce que permettent de faire des commentaires et des spécifications d'un programme, une variable avec un nom explicite, l'affectation de variables, l'indentation, les mouvements relatifs et absolus ainsi que l'orientation d'un lutin propres au logiciel *Scratch*, une fonction, etc., n'est jamais explicité dans la technologie algorithmique.

En constituant le recueil, nous dégageons trois types de tâches dont l'étude n'est presque jamais proposée et qui sont associés à des praxéologies algorithmiques à enseigner :

- $T_{algo,1}$: « Écrire un algorithme en langage courant » (jamais observé au travers du recueil) ;
- $T_{algo,2}$: « Réaliser des activités algorithmiques débranchées » (analysé dans 4 des 107 *productions A&P*) ;
- $T_{algo,3}$: « Comprendre un algorithme ou un programme » (la réalisation embryonnaire de quelques rares spécimens n'est observée que pour l'accomplissement de tâches comme, par exemple, déterminer un seuil pour la suite géométrique (u_n) de raison 0,9965 et de premier terme 0,01. En l'occurrence, l'élève professeur écrit : « il faut identifier les sous-structures du programme, les différentes variables et leur rôle. Ici, n désigne le nombre d'heures et u la valeur de u_n »).

Une seule de ces activités ne se prolonge pas par une activité de programmation et engage l'élève à mettre en ordre dans l'environnement papier-crayon des blocs d'instructions pour tracer un carré). Les types de tâches $T_{algo,1}$, $T_{algo,2}$ et $T_{algo,3}$ sont mentionnés dans le curriculum prescrit mais les praxéologies algorithmiques associées semblent délaissées ou oubliées par le professeur.

Concernant le type de tâches $T_{algo,1}$ à partir du corpus, nous observons que le *logos* de l'élève professeur est confus lorsqu'il s'agit de qualifier l'écriture d'un algorithme. Pourtant, les élèves professeurs ont été formés à identifier les différentes écritures d'un algorithme, en particulier de l'affectation (« = » ou ←). Par exemple, dans l'*analyse A&P* n° 1, l'élève professeur évoque l'« algorithme de dichotomie en pseudo-code » de la vidéo de Yvan Monka (2017). Or, comme le dit et l'écrit Yvan Monka, il étudie l'écriture de l'algorithme en « langage naturel » puis dans quelques langages de programmation de calculatrices. Tout se passe comme si cet élève professeur ne distinguait pas langage naturel et pseudo-code. La *praxis* de l'élève professeur pour produire des écritures d'un algorithme en langage courant pourrait aussi être lacunaire. Par exemple, dans l'*analyse A&P* n° 3, l'élève professeur écrit que « Faire construire la frise à la main est possible, mais l'intérêt est très limité. […] On aurait pu faire cette frise avec *GeoGebra* […]. J'ai choisi de le faire sur *Scratch* car *GeoGebra* utilise le mot « vecteur », qui n'est pas au programme ». Pour cet élève professeur, qui ne mentionne pas le « protocole de construction » de *GeoGebra* — et qui semble donc aussi mésestimer la nature algorithmique d'une technique pour construire la frise —, une frise est certes définie mais aussi constructible par glissement spatial. Or, construirait-on deux cercles extérieurement tangents et de même rayon uniquement par translation ? Enfin, comme l'illustre l'*analyse A&P* n° 4, la mise en mots d'une technique pour « écrire un programme » pourrait esquisser l'écriture d'un algorithme en langage courant, quitte à instancier pour certaines instructions au moyen d'un langage de programmation particulier. Nous relevons que la *production A&P* n° 4 est la seule du recueil qui expose une mise en mots aboutie d'une technique. Il semble donc que le professeur ait besoin d'une praxéologie algorithmique associée au type de tâches « écrire un algorithme en langage courant ».

À propos du type de tâches $T_{algo,2}$, en général, dans les 107 *productions A&P*, un algorithme est toujours écrit en langage de programmation. Et c'est davantage la programmation que l'algorithmique qui est mise en avant dans les *analyses A&P* du corpus : comme en témoigne l'*analyse A&P* n° 2, ce sont les « fonctionnalités proposées [du logiciel *Scratch* qui] permettent à l'élève […] de se familiariser petit à petit avec les principes de la programmation ». Cette emphase sur la programmation au détriment de l'algorithmique en mode débranché pourrait provenir du fait que, très souvent, lors de son analyse de l'OM prévue pour l'étude, l'élève professeur compare implicitement les techniques pour réaliser des « calculs effectués à la main » dans l'environnement papier-crayon et pour réaliser ces calculs — plus rapidement (*cf.* figure 3) et avec plus de fiabilité (*cf.* figure 4) — dans un environnement de programmation ; cette comparaison semble aussi valoir pour les autres types de tâches mathématiques mentionnés dans le corpus (y compris « déterminer la valeur en entrée d'un programme de calcul connaissant la valeur en sortie » (n° 2) puisque l'élève professeur engage l'élève à « remonter un programme de calcul » sans utiliser le calcul littéral). Une autre origine de cette emphase apparaît au travers de l'analyse de l'OA prévue pour l'étude, dans la *production A&P* n° 7. En effet, pour déterminer un encadrement de $\sqrt{2}$ à 10^{-3} près, une technique algorithmique par balayage est utilisée en mode débranché pour obtenir un encadrement à 10^{0} près puis à 10^{-1} près. Soudain, l'élève professeur interrompt la mise en œuvre de cette technique et promeut une technique de programmation comme si son émergence était prioritaire : pour cet élève professeur, la recherche en mode débranché des deux encadrements de $\sqrt{2}$ semble suffire à combler le manque de travail de l'activité en mode débranché et à préparer le travail essentiel de programmation. Certes, le curriculum prescrit met l'accent sur la conception d'*« un algorithme, mais aussi sur la phase de programmation effective »* (MENESR, 2016), en les distinguant cependant. Par ailleurs, les élèves professeurs sont formés à produire un algorithme en mode débranché, par exemple pour écrire l'algorithme d'Euclide, pour rechercher le zéro d'une fonction par dichotomie ou balayage et surtout pour construire une figure géométrique. Il semble donc que le professeur gagnerait à se munir de la praxéologie algorithmique associée au type de tâches « réaliser des activités algorithmiques débranchées » pour enseigner l'algorithmique et la programmation.

Concernant le type de tâches $T_{algo,3}$, dans les *analyses A&P* du corpus, hormis pour l'*analyse A&P* n° 4, nous observons que l'élève professeur développe peu (*cf.* figure 3, par exemple) ou pas du tout (*cf.* figure 4, par exemple) l'analyse de l'OA prévue à l'étude. Ce phénomène est confirmé par l'analyse du recueil : un élève professeur résume cela en cantonnant son analyse de la technique algorithmique à « La technique est très difficile à expliciter ». Quant à la technologie algorithmique, elle est soit inexistante (*cf.* figure 4, par exemple), soit confondue avec la technologie de l'OM enjeu de l'étude, soit réduite à la mention d'objets informatiques (*cf.* figure 6, par exemple) ; par catachrèse du mot « logiciel » et confusion avec la notion d'algorithme, un élève professeur analyse la technologie algorithmique comme suit : « Le logiciel *Scratch* est un logiciel de programmation qui permet d'entrer une succession d'instructions dans le but de résoudre un problème ». Faire allusion à un objet informatique ne permet pas de le connaître, lui et ses fonctions, pour comprendre ce qui peut en être fait lors d'une étude. Par ailleurs, aucun commentaire ou aucune spécification d'un programme, et pas plus pour l'usage d'une *table de traces*, ne sont évoqués au travers du recueil alors que les formateurs les ont fait travailler en séance S2. Si en acte, le professeur peut comprendre un certain algorithme ou programme, il semble qu'il aurait besoin de s'équiper davantage de la praxéologie algorithmique associée au type de tâches « comprendre un algorithme ou un programme » pour enseigner l'algorithmique et la programmation.

Par ailleurs, le professeur ne semble pas non plus être équipé de la praxéologie associée au type

de tâches $T_{algo,4}$: « étudier un algorithme ». En effet, peut-être en interprétant l'extrait *« algorithmes d'approximation numérique d'un extremum (balayage, dichotomie) »* du curriculum prescrit pour la classe de seconde, l'élève professeur producteur de l'*analyse A&P* n° 1 écrit : « Je mettrai également sur l'ENT un prolongement du TP introduisant la méthode d'encadrement par dichotomie. Cela permettra notamment de comparer les deux méthodes, pour les élèves intéressés ». Or, en classe de seconde, si un élève est engagé à comparer les algorithmes par balayage et par dichotomie, il est en droit de (se) demander lequel des deux algorithmes il doit privilégier s'il veut faire un choix. Le professeur gagnerait à savoir comparer des complexités d'algorithmes[19] pour répondre à la question d'un tel élève et comprendre qu'il n'est pas si facile de « comparer les deux méthodes ». Il pourrait donc être utile à un professeur de savoir que la *complexité*[20] de la recherche par balayage (séquentielle) est linéaire en fonction d'une taille fixée des données quand celle de la recherche dichotomique est logarithmique. En outre, de nombreuses *productions A&P* présentent la détermination d'un seuil pour une suite grâce à un programme comportant une boucle non bornée (*production A&P* n° 6, par exemple). Or aucune de ces productions ne mentionne l'importance de justifier que le programme se termine (le variant de boucle étant décroissant du fait de la monotonie de la suite étudiée). Il s'ensuit que la praxéologie algorithmique associée au type de tâches « étudier un algorithme » pourrait aussi être utile au professeur pour enseigner l'algorithmique et la programmation, en particulier pour construire des praxéologies didactiques.

Analyse des analyses A&P : le point de vue de la question $Q_{did-A\&P}$

De notre analyse du corpus, nous dégageons deux types de tâches :

- $T_{did,1}$: Concevoir l'étude de l'algorithmique et la programmation lors de l'étude d'une praxéologie « mixte » mathématiques-informatique ;
- $T_{did,2}$: Diriger l'étude de l'algorithmique et la programmation lors de l'étude d'une praxéologie « mixte » mathématiques-informatique.

Concernant le type de tâches $T_{did,1}$, nous observons que les énoncés des *productions A&P* du corpus ne font pas toujours travailler à l'élève une tâche problématique et que des raisons d'être de l'enjeu de l'étude sont souvent absentes. Par exemple, l'énoncé de la *production A&P* n° 8 mentionne une fonction f définie sur $[-3\,;1]$ par $f(x)=0{,}9x^3+3{,}8x2+1{,}2x-2$ dont il faut « déterminer des valeurs approchées des coordonnées du maximum et du minimum » sur l'intervalle $[-3\,;1]$. Pour ce faire, il s'agit d'exécuter un programme de recherche par balayage qui est fourni en langage *Python*. Or la tâche peut être aisément accomplie sans programmer, en utilisant une représentation graphique donnée par un traceur[21]. Formulée ainsi, la tâche à réaliser n'est donc pas problématique pour l'élève (un prolongement évoqué du scénario rend la tâche plus problématique en interdisant de zoomer sur l'écran). Par ailleurs, quelle est la raison d'être de cette l'étude ? Il n'y en a pas. Pour la *production A&P* n° 1 (*cf.* figure 2), la recherche des zéros de la fonction étudiée est problématique mais il n'y a pas de raison d'être à cette recherche-

[19] Par exemple, le professeur peut utiliser la fonction « time.clock() » qui renvoie le temps d'exécution (CPU) en secondes.

[20] Il s'agit de *complexité informatique*. L'objet de l'analyse de la complexité est de quantifier les ressources (temps d'exécution et place mémoire) nécessaires à l'exécution d'un algorithme ; l'objectif visé est de comparer les différents algorithmes qui sont des procédures de résolution d'un même problème. Plusieurs façons sont utilisées pour quantifier les ressources (meilleur des cas, pire des cas, en moyenne). Dans l'article, nous calculons des complexités en temps, dans le pire des cas, sur l'ensemble des données de taille fixée.

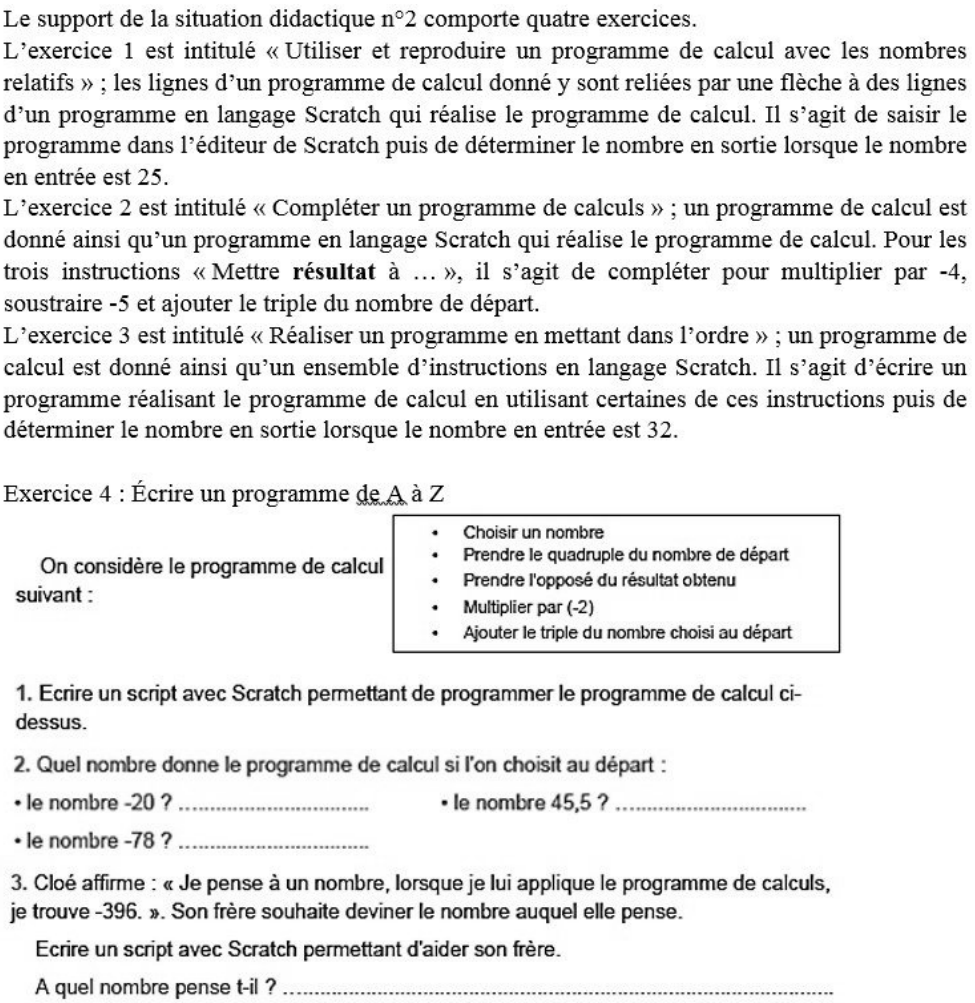
[21] Par exemple, outre une lecture graphique, l'« inspecteur de fonction » du logiciel *GeoGebra* fournit cela.

là. Enfin, pour la *production A&P* n° 7, si la tâche à réaliser est problématique puisqu'il s'agit de déterminer un encadrement de $\sqrt{2}$ à 10^{-n} près (n est un nombre entier naturel quelconque), ce n'est pas le cas pour la *production A&P* n° 9 car, avec une calculatrice, il est aisé de déterminer un encadrement de $\sqrt{3}$ à 10^{-5} près ; les raisons d'être de la recherche de ces encadrements sont là aussi absentes. En l'occurrence, pour concevoir l'enseignement de l'algorithmique et la programmation, certains ingrédients techniques de praxéologies didactiques de conception d'une étude semblent négligés par le professeur : s'assurer de la problématicité — pour l'élève — des tâches algorithmiques et mathématiques enjeux de l'étude ; en expliciter des raisons d'être sachant que l'initiation à la programmation mentionnée par le prescrit n'en est pas une.

Poursuivons cette analyse. D'une part, nous jugeons pertinente l'exploitation de l'algorithmique et la programmation pour la *production A&P* n° 4 (voir figure 5). En effet, l'efficacité et la fiabilité d'une technique en mode débranché de détermination de diviseurs d'un nombre entier donné sont moindres que celles de la technique algorithmique mentionnée. D'autre part, l'élève professeur planifie la réalisation d'un épisode du moment exploratoire — pour la technique τ_{math} — lors de la détermination de la perfection successivement des nombres 6, 8128 et 33550336, un outil de programmation étant disponible pour l'étude. Mais pour d'autres *productions A&P* du corpus, nous estimons peu essentielle l'exploitation proposée de l'algorithmique et la programmation. Par exemple, pour la *production A&P* n° 3, l'élève professeur contraint l'élève à exploiter le logiciel *Scratch* pour lui permettre de voir « le mouvement de déplacement et la répétition à l'identique du motif de base ». Mais, par exemple en faisant produire à l'élève une *bande dessinée algorithmique* dans l'environnement papier-crayon, l'élève ne pourrait-il pas reconnaître les « glissements » du motif de base de la frise étudiée ?

Cette exploitation de l'algorithmique et la programmation est même jugée non pertinente pour l'exercice 4 de la *production A&P* n° 2 (*cf.* figure 7). Comme la *production A&P* n° 3, cette production met en lumière un phénomène didactique que nous avons souvent observé dans le recueil : lors de la conception de l'enseignement de l'algorithmique et la programmation, l'élève professeur choisit de faire utiliser à l'élève un environnement de programmation sans questionner explicitement ce choix ; en l'occurrence, il sélectionne le logiciel *Scratch*. Or, par exemple, il pourrait faire utiliser des outils numériques plus familiers de l'élève comme une calculatrice ou un tableur si de nombreuses exécutions sont à réaliser. Lorsque ce choix d'un environnement de programmation est réalisé avant d'avoir conçu une étude, il contraint la conception de l'étude ($T_{did,1}$). À propos de la *production A&P* n° 2, l'élève professeur doit prévoir un travail préalable autour du logiciel *Scratch* (*cf.* figure 7 où les exercices 1, 2 et 3 ont été synthétisés) et forcer l'émergence d'une technique particulière pour répondre à l'interrogation du frère de Chloé. Mais l'élève pourrait rechercher le nombre à deviner en employant une technique par tâtonnements ou une technique algébrique après avoir modélisé le programme de calcul par une expression littérale pour résoudre une équation du premier degré. Pour éviter l'émergence d'autres techniques que celle qu'il souhaite, l'élève professeur est obligé de contraindre l'élève à « écrire un script avec *Scratch* permettant d'aider son frère », sans qu'il puisse prendre d'initiative. Nous parvenons à la même évaluation concernant la *production A&P* n° 5 (étude du terme d'une suite à motif croissant (« pattern » géométrique) avec le logiciel *Scratch*) ou de nombreuses autres productions. Au travers de ces productions, l'enseignement de l'algorithmique et de la programmation semble être bâti autour du choix initial par l'élève professeur d'un outil logiciel de programmation. Ce mode de conception de l'enseignement contraint aussi la direction d'une étude ($T_{did,2}$) : pour que l'élève exploite tel outil logiciel, l'élève professeur doit régir l'étude dès ses origines. Car il n'est pas prévu par l'élève professeur

que l'élève puisse poser des questions comme : « Quelle(s) technique(s) mettre en œuvre ? », « Pourquoi exécuter une technique sollicitant la pensée algorithmique plutôt qu'une autre ? « Dans quel environnement écrire et exécuter un algorithme ? », etc. Ainsi, ce sont l'étude prévue, la réalisation des moments de celle-ci, les rôles de l'élève et de l'élève professeur qui sont contraints par une telle constitution originelle du milieu d'une étude. Il semble donc que, pour concevoir l'enseignement de l'algorithmique et la programmation, le professeur gagnerait à s'équiper de praxéologies didactiques pour constituer un milieu initial de l'étude — lorsqu'il doit contenir un environnement de programmation — et pour diriger une étude sollicitant une programmation.



Le support de la situation didactique n°2 comporte quatre exercices.
L'exercice 1 est intitulé « Utiliser et reproduire un programme de calcul avec les nombres relatifs » ; les lignes d'un programme de calcul donné y sont reliées par une flèche à des lignes d'un programme en langage Scratch qui réalise le programme de calcul. Il s'agit de saisir le programme dans l'éditeur de Scratch puis de déterminer le nombre en sortie lorsque le nombre en entrée est 25.
L'exercice 2 est intitulé « Compléter un programme de calculs » ; un programme de calcul est donné ainsi qu'un programme en langage Scratch qui réalise le programme de calcul. Pour les trois instructions « Mettre **résultat** à … », il s'agit de compléter pour multiplier par -4, soustraire -5 et ajouter le triple du nombre de départ.
L'exercice 3 est intitulé « Réaliser un programme en mettant dans l'ordre » ; un programme de calcul est donné ainsi qu'un ensemble d'instructions en langage Scratch. Il s'agit d'écrire un programme réalisant le programme de calcul en utilisant certaines de ces instructions puis de déterminer le nombre en sortie lorsque le nombre en entrée est 32.

Exercice 4 : Écrire un programme de A à Z

On considère le programme de calcul suivant :

- Choisir un nombre
- Prendre le quadruple du nombre de départ
- Prendre l'opposé du résultat obtenu
- Multiplier par (-2)
- Ajouter le triple du nombre choisi au départ

1. Ecrire un script avec Scratch permettant de programmer le programme de calcul ci-dessus.

2. Quel nombre donne le programme de calcul si l'on choisit au départ :

• le nombre -20 ? ..... • le nombre 45,5 ? .....

• le nombre -78 ? .....

3. Cloé affirme : « Je pense à un nombre, lorsque je lui applique le programme de calculs, je trouve -396. ». Son frère souhaite deviner le nombre auquel elle pense.

Ecrire un script avec Scratch permettant d'aider son frère.

A quel nombre pense t-il ? .....

***Figure** 7 : Situation n°2 - Support de la situation didactique, une synthèse.*

Enfin, souvent, l'algorithmique et la programmation sont étudiées quand la classe en ressent le besoin, lors de l'étude mathématique. Mais pour la *production A&P* n° 2 (*cf.* figure 7), elles sont étudiées avant que l'étude mathématique qui en tirera profit n'ait commencé. En effet, en amont du travail par la classe de la question 3 de l'exercice 4, l'élève professeur met en place un parcours d'étude de l'algorithmique et la programmation — il s'agit successivement d'exécuter, de compléter puis d'écrire un programme en langage *Scratch* ; l'élève est ainsi engagé à s'équiper en praxéologies algorithmiques et de programmation supposées être nécessaires pour réaliser la tâche mathématique qu'il aura à réaliser. Cette production, à l'instar d'autres *productions A&P* du recueil, témoigne de la conception d'une étude isolée de l'algorithmique et

la programmation, souvent sous forme de « travaux pratiques » en salle informatique, parfois spiralée sur l'année scolaire et quelquefois différenciée. L'objectif visé par l'élève professeur semble être que l'élève atteigne un niveau minimal d'acquisition de certaines praxéologies algorithmiques et de programmation. En général, le travail par l'élève de telles praxéologies n'est que peu accompagné par le professeur : sur les 107 *productions A&P* du recueil, seule la *production A&P* n° 4 témoigne de « coups de pouce » spécifiques — comportant des mises en garde quant au fonctionnement du logiciel *Scratch*, des instructions à insérer dans le programme attendu, des questions comme « Tester le programme avec le nombre 30. Il manque un nombre dans la liste des diviseurs. Lequel ? », etc. Il semble donc que le professeur soit contraint de développer un dispositif dédié à l'étude de l'algorithmique et la programmation pour que l'élève puisse s'engager ultérieurement dans l'étude mathématique qui s'en nourrira. Le professeur de mathématiques rencontrerait donc des difficultés pour concevoir et aussi pour diriger une étude sollicitant l'algorithme et la programmation du fait d'un besoin en praxéologies didactiques.

Conclusion

Dans le contexte de l'étude de praxéologies « mixtes » mathématiques-informatique, notre recherche a pu dégager deux types de besoins du professeur de mathématiques pour enseigner l'algorithmique et la programmation : des besoins en praxéologies algorithmiques pour enseigner et en praxéologies didactiques. Il semble que le professeur de mathématiques gagnerait à s'équiper des praxéologies associées aux types de tâches :

$T_{algo,1}$: Écrire un algorithme en langage courant ;

$T_{algo,2}$: Réaliser des activités algorithmiques débranchées ;

$T_{algo,3}$: Comprendre un algorithme ou un programme ;

$T_{algo,4}$: Étudier un algorithme.

Nous avons montré que, pour le professeur, les objets algorithmiques sont davantage perçus comme des outils pour une étude mathématique que comme des objets d'étude ; pour lui, il s'agirait de familiariser l'élève à la programmation. En outre, quand le professeur serait insuffisamment équipé en savoirs informatiques, son enseignement de l'algorithmique et la programmation pourrait en pâtir. De plus amples recherches pourraient avantageusement repérer l'équipement en savoirs informatiques jugé utile au professeur de mathématiques pour enseigner l'algorithmique et la programmation. Par ailleurs, cette recherche met en exergue les besoins du professeur en praxéologies didactiques relatives aux types de tâches $T_{did,1}$: « Concevoir l'étude de l'algorithmique et la programmation lors de l'étude d'une praxéologie « mixte » mathématiques-informatique » et $T_{did,2}$: « Diriger l'étude de l'algorithmique et la programmation lors de l'étude d'une praxéologie « mixte » mathématiques-informatique ». Nous avons identifié des contraintes pesant sur la conception et la direction de l'étude ; elles émanent essentiellement de la difficulté du professeur à : problématiser et motiver une étude qui sollicite l'algorithmique et la programmation ; concevoir un rôle de l'élève suffisamment ample et un milieu pour l'étude lorsqu'un environnement de programmation est prévu pour celle-ci. En 2013, Strock avait repéré des difficultés du professeur à gérer le temps de l'étude lors de l'enseignement concomitant d'un programme de mathématiques et d'un programme d'algorithmique. De futures recherches pourraient repérer d'autres besoins du professeur — peut-être ceux relatifs à la planification de l'avancement du temps didactique — et proposer des moyens de satisfaire les besoins repérés.

Références bibliographiques

Artaud, M. (2021). Des grandeurs et de leur mesure : besoins praxéologiques de la position de professeur et leur satisfaction. Dans H. Chaachoua, A. Bessot, B. Barquero, L. Coulange, G. Cirade, P. Job, A.-C. Mathé, A. Pressiat, M. Schneider & F. Vandebrouk (éds.), *Actes de la XX^e école d'été de didactique des mathématiques* (vol. 1, pp. 197-226). Autrans : La pensée sauvage.

Berry, G. (2019). L'éducation à l'informatique. Cours au Collège de France. https://www.college-de-france.fr/agenda/cours/ou-va-informatique/enseigner-informatique

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 en Didactique des mathématiques, Université Montpellier 2].

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

Chevallard, Y. (2002a). Organiser l'étude. 1. Structures & fonctions. Dans J.-L. Dorier, M. Artaud, R. Berthelot & R. Floris (éds.), *Actes de la XI^e école d'été de didactique des mathématiques* (pp. 3-32). Grenoble : La pensée sauvage.

Chevallard, Y. (2002b). Organiser l'étude. 3. Ecologie & régulation. Dans J.-L. Dorier, M. Artaud, R. Berthelot & R. Floris (éds.), *Actes de la XI^e école d'été de didactique des mathématiques* (pp. 41-56). Grenoble : La pensée sauvage.

Chevallard, Y. & Cirade, G. (2010). Les ressources manquantes comme problème professionnel. Dans G Gueudet & L Trouche (éds.), *Le travail documentaire des professeurs en mathématiques* (pp. 41-55). Rennes : PUR & Lyon : INRP.

Chevallard, Y. (2017). La TAD et son devenir : rappels, reprises, avancées. Dans G. Cirade, M. Artaud, M. Bosch, J.-P. Bourgade, Y. Chevallard, C. Ladage & T. Sierra (éds.), *Évolutions contemporaines du rapport aux mathématiques et aux autres savoirs à l'école et dans la société* (pp. 27-65). https://citad4.sciencesconf.org.

Chevallard, Y. (2021). La question curriculaire à la lumière de la TAD : défigement praxéologique et questionnement du monde. Dans H. Chaachoua & A. Bessot (éds.), *Actes de la XX^e école d'été de didactique des mathématiques* (vol. 1, pp. 93-111). Autrans : La pensée sauvage.

Cirade, G. (2017). Éléments d'écologie des problèmes de la profession. Dans G. Cirade, M. Artaud, M. Bosch, J.-P. Bourgade, Y. Chevallard, C. Ladage & T. Sierra (éds.), Évolutions contemporaines du rapport aux mathématiques et aux autres savoirs à l'école et dans la société (pp. 785-802). https://citad4.sciencesconf.org.

Conseil scientifique de la SIF (2014). L'informatique : la science au cœur du numérique. *1024 - Bulletin de la société informatique de France, 2*, 13-20.

Couderette, M. (2016). Enseignement de l'algorithmique en classe de seconde : une introduction curriculaire problématique. *Annales de didactique et de sciences cognitives, 21*, 267-296.

Knuth, D. (1997). *The Art of Computer Programming. Fundamental Algorithms (3e éd., vol. 1)*. États-Unis : Addison-Wesley.

Modeste, S. (2012). La pensée algorithmique : apports d'un point de vue extérieur aux mathématiques. Dans J.L. Dorier & S. Coutat (éds.), *Enseignement Des Mathématiques et Contrat Social : Enjeux et Défis pour le 21e Siècle - Actes du Colloque EMF 2012* (GT3, pp. 467-479).

Modeste, S. & Ouvrier-Buffet, C. (2011). The appearance of algorithms in curricula. A new opportunity to deal with proof? *CERME 7* . Rzeszów, Poland.

Monka, Y. (2017). *Comprendre le principe de dichotomie*. https://www.youtube.com/watch?v=V7mlMCSrq1U

Nguyen, C. T. & Bessot, A. (2003). La prise en compte des notions de boucle et de variable. *Petit x, 62*, 7-32.

Strock, J.-M. (2013). *Pédagogie de l'enquête sur l'algorithmique dans l'enseignement des mathématiques au secondaire : une étude exploratoire de praxéologies nécessaires*. Mémoire de Master. Université Aix-Marseille 1.

Strock, J.-M. & Artaud, M. (2016). Du logos des organisations algorithmiques dans l'enseignement secondaire. *Educação Matemática Pesquisa, Revista do Programa de Estudos Pós-Graduados em Educação Matemática, 4*(21), 185-200.

Ministère de l'enseignement supérieur et de la recherche (2011). Certificat informatique et internet de l'enseignement supérieur. *Bulletin officiel n° 5 du 3 février 2011.*

Ministère de l'éducation nationale, de l'enseignement supérieur et de la recherche. (2016). *Ressources d'accompagnement du programme de mathématiques (cycle 4). Algorithmique et programmation*. https://cache.media.eduscol.education.fr/file/Algorithmique_et_programmation/67/9/RA16_C4_MATH_algorithmique_et_programmation_N.D_551679.pdf

Ministère de l'éducation nationale. (2017). *Ressources pour le lycée. Mathématiques. Algorithmique et programmation*. https://cache.media.eduscol.education.fr/file/Mathematiques/73/3/Algorithmique_et_programmation_787733.pdf

Ministère de l'éducation nationale et de la jeunesse. (2019). *Ressources pour les classes de seconde et de première de la voie générale et technologique. Algorithmique et programmation*. https://eduscol.education.fr/document/24592/download