Rôle d'un logiciel dans la transposition didactique du concept d'algorithme : le cas du logiciel AlgoBox en France et des programmes du lycée entre 2009 et 2019

Antoine Meyer, Simon Modeste

Rôle d'un logiciel dans la transposition didactique du concept d'algorithme : le cas du logiciel *AlgoBox* en France et des programmes du lycée entre 2009 et 2019

Antoine Meyer
LIGM, Université Gustave Eiffel, CNRS, ESIEE Paris

Simon Modeste
IMAG, Université de Montpellier, CNRS

Résumé

Cet article présente une analyse épistémologique et didactique de phénomènes curriculaires relatifs à l'enseignement de l'algorithmique en France pendant la période 2009-2019 dans le cursus de mathématiques du lycée. Notre étude s'appuie sur l'analyse du logiciel AlgoBox*, proposé pour une initiation à l'algorithmique en cohérence avec ces programmes et très utilisé pendant la période. Ce logiciel présente des choix techniques marqués dont nous questionnons les conséquences. Nous montrons en quoi le « projet » incarné par* AlgoBox *se heurte à des difficultés d'ordres conceptuel et didactique.*

Mots-clés : algorithmique ; programmation ; lycée ; curriculum ; didactique.

Abstract

This article presents an epistemological and didactical analysis of curricular phenomena regarding the teaching of algorithmics in France in 2009-2019 in the mathematics curriculum *of high school. Our study is based on the analysis of* Algobox*, a software designed for an introduction to algorithmics in coherence with the* curriculum*, and much used during this period. This software makes marked technical choices, whose consequences are questioned here. We show how the "project" embodied by* AlgoBox *runs up against conceptual and didactical difficulties.*

Keywords: algorithmics; programming; high school; curriculum; didactics.

Introduction

L'histoire de l'enseignement de l'informatique au lycée en France est marquée par différentes périodes, montrant des rapports divers à l'informatique, avec des postures liées aux évolutions techniques de l'époque (Baron & Bruillard, 2011). Ainsi, l'option informatique des lycées mise en place dans les années 1980 se présente essentiellement comme un enseignement de programmation. Les années 1990 sont marquées par l'arrivée sur le marché de « progiciels », dont la prise en main et l'usage pédagogique viennent remplacer l'enseignement de la programmation. C'est au début des années 2000 que le débat sur l'enseignement des concepts de la science informatique, de la programmation et de l'algorithmique reprend de l'ampleur et aux alentours des années 2010 que l'on peut noter le retour d'éléments d'algorithmique et de programmation au lycée général (à partir de 2009), ainsi que de premiers enseignements autonomes d'informatique (à partir de 2012).

Cet article se focalise sur la période 2009-2019, délimitée par deux événements particulièrement marquants dans l'histoire de l'enseignement de l'informatique au secondaire en France[1]. Malgré quelques aménagements opérés au fil des années, cet enseignement nous semble relever d'un point de vue singulier sur la discipline en général, et sur l'algorithmique en particulier.

L'année 2009 marque l'entrée au lycée d'éléments d'algorithmique dans les programmes de mathématiques des filières générales du lycée (en seconde en 2009, puis en première en 2010 et en terminale en 2011), à l'occasion d'une réforme initiée par Xavier Darcos et poursuivie par Luc Châtel, alors ministres de l'Éducation Nationale. L'introduction de ces concepts auprès d'enseignants disposant de connaissances très hétérogènes en informatique a donné lieu à l'élaboration de contenus tout à fait spécifiques. Les attentes de ces nouveaux programmes ont engendré des phénomènes de transposition didactique[2] (Chevallard, 1985) du concept d'algorithme et de l'activité algorithmique, qui ont déjà fait l'objet d'études en didactique (Modeste, 2012 ; 2020).

En 2019, autre jalon important, une réforme profonde du lycée général, incluant notamment la suppression des filières S, L et ES du lycée général, a entériné l'apparition de l'informatique au lycée en tant que discipline scolaire à part entière : la spécialité « Numérique et Sciences Informatiques » (NSI) proposée en première et en terminale. Pour autant, les éléments d'algorithmique et de programmation déjà présents dans les programmes de mathématiques n'en furent pas retirés. C'est cependant à partir de cette date que

1 Travail réalisé avec le soutien financier de l'ANR (projet DEMaIn, référence <ANR-16-CE38-0006-01>). Cet article est accompagné de 3 annexes, accessibles en ligne sur le site de la revue qui apportent des précisions sur certains aspects de nos analyses [URL : https://chc.hypotheses.org/1670].

2 Nous présentons le concept de transposition didactique un peu plus loin.

le langage de programmation Python est officiellement prescrit comme langage de référence pour enseigner ces contenus, ce qui constitue une rupture par rapport aux années précédentes (où aucun langage spécifique n'était imposé).

La question du choix du premier langage de programmation à enseigner à des débutants est ancienne et complexe, et a donné lieu dans la littérature à d'innombrables publications, au point d'être qualifiée avec humour par Gal-Ezer et Harel de « *guerre culturelle* » (Gal-Ezer & Harel, 1998, p. 82, notre traduction). Un exemple d'étude récente portant sur un grand nombre d'élèves et comportant une revue de littérature sur le sujet peut être trouvé dans (Chen & *al*., 2019). Sans prétendre répondre à cette question, c'est à cet enjeu que nous allons nous intéresser, en étudiant l'influence sur les curriculums du logiciel *AlgoBox* et de son langage, conçus spécifiquement pour l'enseignement de l'algorithmique au lycée dans le contexte de la réforme de 2009.

Pour ce faire, nous avons étudié les contenus algorithmiques des curriculums de mathématiques de lycée de la période 2009-2019, des particularités du logiciel *Algobox*, et de la place qu'il a occupée dans l'écosystème curriculaire de la période.

Notre article s'organise en quatre parties :

- présentation du cadre conceptuel, de la méthodologie et du corpus ;
- présentation et analyse des curriculums et leurs évolutions entre 2009 et 2019 ;
- présentation et analyse du logiciel *AlgoBox* et de son langage, et ses particularités techniques ;
- analyse et discussion de la place et du rôle d'*AlgoBox* au sein du curriculum.

Afin d'analyser les contenus d'algorithmique (et plus généralement d'informatique) dans le curriculum prescrit de mathématiques au lycée, ainsi que les caractéristiques et effets du logiciel *AlgoBox*, nous prenons appui sur les concepts de transposition didactique et de transposition informatique. La présentation de ces outils théoriques nous amène à préciser des éléments d'épistémologie relatifs aux concepts étudiés, qui constitueront une référence pour l'analyse. Nous présentons ensuite la méthodologie de l'étude et le corpus mobilisé.

Cadrage théorique

Transposition didactique

Pour cette recherche, nous ferons principalement appel au concept de transposition didactique. Ce concept a été introduit en didactique des sciences par Chevallard (1985) et s'est fortement développé en didactique des mathématiques (Chevallard & Bosch, 2014), en particulier dans le cadre de la théorie anthropologique du didactique (Bosch & Gascón, 2014).

La transposition didactique est définie comme le phénomène de passage d'un savoir de référence (académique, professionnel…) dit savoir savant, au savoir à enseigner (transposition dite externe), puis du savoir à enseigner au savoir enseigné (transposition dite interne). Le savoir à enseigner est celui dont les institutions d'enseignement prescrivent l'apprentissage, de façon plus ou moins explicite (c'est le savoir défini par les programmes officiels dans le cas de l'enseignement obligatoire en France).

Le processus de transposition didactique est soumis à l'action de divers agents (scientifiques et sociétés savantes, inspecteurs, sociétés d'enseignants et syndicats, acteurs du monde professionnel, rédacteurs de manuels, producteurs de ressources…) qui constituent ce que la théorie anthropologique du didactique nomme la noosphère, et qui participe à sélectionner, découper, réorganiser, structurer, définir, apprêter le corpus du savoir à enseigner et le savoir enseigné.

En rendant compte de la distance qui sépare le savoir savant du savoir enseigné, le concept de transposition didactique permet d'exercer une vigilance épistémologique sur les phénomènes curriculaires :

> En effet, alors que l'école vit sur la fiction consistant à voir dans les objets d'enseignement des copies simplifiées mais fidèles des objets de la science, l'analyse épistémologique, en nous permettant de comprendre ce qui gouverne l'évolution de la connaissance scientifique, nous aide à prendre conscience de la distance qui sépare les économies des deux systèmes (Artigue, 1990, pp. 244-245).

Cette « économie » des deux systèmes a été théorisée dans une approche écologique du didactique (Artaud, 1998) qui permet d'expliciter les conditions et contraintes qui régissent la co-habitation des savoirs entre eux, et leur existence, apparition, disparition, évolution, dans les curriculums. Dans cet article, nous retiendrons pour nos interprétations l'idée d'un écosystème didactique dans lequel les savoirs cohabitent et évoluent selon des dynamiques curriculaires.

Transposition informatique

« *Aux contraintes de la transposition didactique s'ajoutent, ou plutôt se combinent, celles de modélisation et d'implémentation informatiques : contraintes de la modélisation computable, contraintes logicielles et matérielles des supports informatiques de réalisation* » (Balacheff, 1994, p. 364). Ainsi, l'informatisation des objets d'enseignement, au sein de logiciels n'est pas une simple traduction, mais produit la transformation de ces objets. Il en résulte des objets d'enseignement possédant certaines représentations, propriétés et permettant ou limitant certaines opérations.

Ce phénomène de transposition informatique nous semble pertinent pour notre problématique au sens où les logiciels et langages de programmation utilisés

ont une implication directe sur la perception des concepts disciplinaires sous-jacents (ici algorithme et programme).

Repères épistémologiques

Dans une préoccupation de vigilance épistémologique mentionnée précédemment, nous présentons un ensemble de points d'attention sur lesquels s'articuleront nos analyses des curriculums et du logiciel *AlgoBox*.

Nous nous appuyons sur des définitions et concepts issus directement du savoir savant, ou d'analyses épistémologiques antérieures, en particulier en ce qui concerne les concepts de problème algorithmique, d'algorithme et de programme, sur celui de spécification, de statut fonctionnel des variables, et enfin sur des considérations relatives à la représentation et à la sémantique des programmes.

Les éléments issus de ces considérations épistémologiques et de l'analyse du savoir savant peuvent être vus comme l'explicitation d'un modèle épistémologique de référence au sens de la théorie anthropologique du didactique (Bosch & Gascón, 2014), et serviront de référence pour nos analyses.

• *Algorithme et problème algorithmique*

Dans sa thèse, en appui sur plusieurs ouvrages de référence, Modeste (2012) propose une définition générale du concept d'algorithme reposant sur le concept important de « problème algorithmique », qu'il convient de distinguer du concept général de problème rencontré en mathématiques, en science ou en didactique.

> Un problème p est un couple (I, Q), où I est l'ensemble des instances (pouvant être décrit par plusieurs paramètres) du problème et Q une question portant sur ces instances (spécifiant les propriétés de la solution attendue). […]
>
> Un algorithme est une procédure de résolution de problème, s'appliquant à une famille d'instances du problème et produisant, en un nombre fini d'étapes constructives, effectives, non-ambigües et organisées, la réponse au problème pour toute instance de cette famille (Modeste, 2012, p. 25).

L'expression « résolution de problème » est à entendre ici dans un sens plus concret et opératoire que dans le sens général de « démarche de résolution de problèmes » utilisé parfois dans le milieu de l'éducation pour faire référence à un certain type de compétences d'investigation, de raisonnement ou d'expérimentation attendu des élèves. Nous reviendrons sur cette polysémie du mot « problème » plus loin.

Nous insistons également sur les notions d'« instance » d'un problème (potentiellement définie par plusieurs paramètres), et sur celle de « résultat », composantes essentielles de la notion stricte d'algorithme. Dans une pratique experte, il est attendu que la présentation d'un algorithme s'accompagne de sa « spéci-

fication », qui consiste généralement en la description *a priori* du problème que l'algorithme prétend résoudre. Cette exigence s'étend également à la pratique de la programmation.

• *Programmes et machines*

Comme nous l'avons vu, un algorithme est un objet de nature abstraite dont la définition ne fait pas nécessairement référence à un quelconque dispositif physique de calcul[3]. Il est cependant entendu que la raison d'être d'un certain nombre d'algorithmes est d'être traduits (ou « implémentés ») en un « texte » appelé programme destiné à être exécuté par une telle « machine », par exemple par un ordinateur.

Nous définissons un programme comme une suite de symboles (un texte), agencée selon une grammaire ou syntaxe prédéfinie, dont l'interprétation obéit à une certaine sémantique tenant compte des capacités et contraintes de la machine (réelle ou virtuelle) à laquelle il est destiné. Dans Durand-Guerrier & *al.* (2019), où nous explorons l'articulation entre syntaxe et sémantique dans une perspective de didactique des mathématiques et de l'informatique, nous précisons cette distinction dans les termes suivants :

> Dans ce contexte [en informatique], « syntaxe » fait référence aux règles de composition d'un texte valide dans le langage de programmation choisi, et « sémantique » à l'effet attendu de chaque élément syntaxique de ce langage et des combinaisons de ces éléments sur la machine réelle (ou sur un modèle de la machine) (Durand-Guerrier & *al.*, 2019, p. 122, notre traduction).

On nommera dans ce texte « langage de programmation » un système cohérent formé d'un ensemble de symboles, d'une syntaxe et d'une sémantique (en référence à une certaine machine). Nous ne détaillerons pas plus ces définitions, qui font l'objet de discussions plus approfondies dans le chapitre cité.

Il nous paraît également important d'insister sur la différence entre algorithme et programme, comme le résume clairement cet extrait de (Froidevaux, Gaudel & Soria, 1990, p. 6) :

> Un algorithme doit être exprimé dans un langage de programmation pour être compris et exécuté par un ordinateur. Cependant, il faut bien comprendre qu'un algorithme est indépendant du langage de programmation utilisé. L'algorithme d'Euclide programmé en Pascal, en Ada, ou en Lisp, reste toujours l'algorithme d'Euclide.

• *Représentation des algorithmes et programmes*

Dans une perspective d'enseignement de l'algorithmique et/ou de la programmation, particulièrement à desti-

3 On peut néanmoins faire appel à une notion de « machine abstraite » afin de caractériser formellement le sens d'un algorithme, notamment pour pouvoir étudier sa complexité. Nous ne mobiliserons pas ce concept dans cet article.

nation de débutants, le choix d'un mode de représentation des algorithmes (textuelle ou graphique, plus ou moins formalisée) est une question importante[4].

Dans une étude épistémologique à visée didactique, Nguyen (2005) étudie les choix de langage ou de machine de référence dans des traités d'initiation à l'algorithmique et à la programmation, et identifie deux stratégies principales (vues comme des façons d'organiser le corps des savoirs dans un enseignement) :

> **Première stratégie : présence d'une machine de référence, réelle ou fictive**
>
> Ces machines fictives peuvent être virtuelles […], ou idéales […]. L'enseignement de l'algorithmique et de la programmation est alors lié à la compréhension de l'architecture de la machine et à la conception d'un langage orienté vers cette machine.
>
> **Deuxième stratégie : absence de machine de référence**
>
> Dans cette stratégie, l'enseignement de l'algorithmique passe par celui d'un langage de programmation représentatif d'une classe de langages existants […]. Cet enseignement suppose la connaissance préalable d'un langage de programmation (Nguyen, 2005, p. 58).

Pour caractériser les représentations des algorithmes, Modeste (2012, 2020) propose quant à lui trois paradigmes que sont la preuve algorithmique (les algorithmes restent implicites dans les constructions mathématiques et les preuves), l'algorithme mathématique (représentation des algorithmes dans un langage spécifique mélange de langage mathématique et de mots-clés liés aux structures de contrôle ou inspirés de langages de programmation, parfois appelé pseudo-langage) et algorithme informatique (représentation des algorithmes dans des langages informatiques, formels, ayant une syntaxe et une sémantique bien définies).

Ce que nous retenons des deux points de vue de Nguyen et Modeste est l'importance de prendre en compte les représentations des algorithmes pour comprendre les positionnements épistémologiques et la transposition didactique en jeu. La question des effets sur l'enseignement et l'apprentissage de l'algorithmique du choix d'un langage de programmation ou d'une représentation semi-formelle d'algorithmes telle que le « pseudo-code », constitue un thème de recherche à part entière qui dépasse le périmètre de ce travail.

• *Statuts fonctionnels des variables*

Le concept de variable est central dans l'apprentissage de la programmation et, dans une certaine mesure, de l'algorithmique. Samurçay (1985) étudie les conceptions et difficultés d'élèves au sujet des variables pendant la phase d'alphabétisation, comme elle appelle la période de première initiation à l'informatique. Elle

4 Voir par exemple Chen & *al*. (2019) au sujet des impacts différents des langages textuels et par blocs sur l'apprentissage.

identifie en particulier deux statuts fonctionnels de la variable :

> Dans les situations-problèmes que rencontre effectivement le sujet, les variables interviennent avec des statuts fonctionnels différents. Dans un premier temps on peut en distinguer deux :
>
> – les variables qui sont les données explicites du problème ;
>
> – les variables qui sont rendues nécessaires par la solution informatique (Samurçay, 1985, p. 146).

Nous reprenons à notre compte cette catégorisation, en particulier dans un contexte de programmation impérative, et la complétons par un troisième statut fonctionnel correspondant aux variables destinées à désigner le résultat final d'un algorithme (l'usage d'une ou plusieurs variables ayant ce statut n'est pas systématique mais apparaît dans l'écriture de nombreux algorithmes). Par commodité, nous désignerons dorénavant ces trois statuts sous les noms de « variables-paramètres », « variables de travail » et « variables-résultat ».

• *Résolution de problèmes et pensée informatique*

Wing (2006) définit la pensée informatique comme une manière possible pour les humains (et non les ordinateurs) de résoudre des problèmes, dans un sens du mot « problème » plus général que le strict sens algorithmique évoqué ci-dessus. Elle caractérise la pensée algorithmique comme constituée de plusieurs méthodes et formes de raisonnement, qui ont ensuite fait l'objet de débats. Dans une revue de littérature à ce sujet, Selby et Woollard (2013) tentent de repérer et nommer les caractéristiques de la pensée informatique qui font consensus à la fois du point de vue du savoir savant et au sein de la recherche sur l'enseignement de l'informatique. Ils en dégagent ainsi plusieurs composantes propres, invariantes, consensuelles et bien définies, que sont : la capacité à raisonner en termes de décompositions, à l'aide d'abstractions, de manière algorithmique, en termes d'évaluations et en termes de généralisations. Nous renvoyons au texte de Selby et Woollard pour une définition plus détaillée de ces composantes.

Dans ce travail, nommer et distinguer différentes composantes de la pensée informatique nous permettra de poser la question des types de raisonnements que les transpositions didactique et informatique à l'œuvre dans les programmes et dans *AlgoBox* permettent ou entravent.

Méthodologie et corpus

Pour approfondir les questions présentées dans la problématique, nous allons conjointement étudier les curriculums d'algorithmique de la période 2009-2019 et présenter une analyse technique et didactique du logiciel *AlgoBox*, l'objectif étant de montrer comment ont co-évolué le logiciel et les curriculums.

Dans un premier temps nous porterons un regard didactique sur l'histoire récente de l'enseignement de l'algorithmique dans les programmes de mathématiques, et de l'enseignement de l'informatique dans l'enseignement secondaire en France. Ceci permettra de contextualiser la situation de la période 2009-2019, de détailler les principaux événements institutionnels liés aux curriculums, et d'identifier les principales caractéristiques de la transposition didactique externe du concept d'algorithme à l'œuvre.

Dans un second temps, nous étudierons en détail le logiciel *AlgoBox*, en menant une analyse des choix techniques et didactiques qui l'ont porté, ainsi qu'une identification et analyse de ses versions et de leur chronologie. Ceci nous permettra de caractériser la transposition informatique du concept d'algorithme réalisée par le logiciel.

Enfin, nous tenterons de montrer la place et le rôle qu'*AlgoBox* a pu jouer dans le curriculum « réel » au cours de la période étudiée, en d'autres termes son impact possible sur la transposition didactique interne des concepts en jeu, en mettant en perspective les spécificités du logiciel et celles des programmes de mathématiques. Ces analyses seront menées selon les repères épistémologiques ci-dessus.

Pour développer cette étude nous nous sommes appuyés sur les documents institutionnels témoignant du savoir à enseigner (programmes, documents-ressources), et du savoir enseigné (manuels scolaires, sujets de baccalauréat), des documents attestant des échanges et débat au sein de la noosphère (rapports d'experts, consultation, réactions…), ainsi que sur le logiciel *AlgoBox* lui-même (site officiel, documentation, code source, historique des versions, recueils d'activités).

Regard didactique sur l'histoire récente de l'enseignement de l'informatique dans les programmes de mathématiques du lycée

Panorama de l'enseignement de l'algorithmique au lycée (2009-2019)

- *Éléments précurseurs*

La Commission de Réflexion sur l'Enseignement des Mathématiques (CREM), créée en 1999, confiée à la direction du mathématicien et académicien Jean-Pierre Kahane[5] et composée de chercheurs en mathématiques, informatique, didactique des mathématiques, et d'enseignants, formateurs et inspecteurs, produisit divers rapports entre 1999 et 2002 dont une compilation, le rapport

5 Duperret J.-C., « Histoire de la C.R.E.M. (1996-1999) », site *Educmath*, 2007 [URL : http://educmath.ens-lyon.fr/Educmath/ressources/etudes/crem/].

Kahane, fut réalisée en 2002 (Kahane, 2002). Parmi les recommandations issues des premières années de la commission figure le rapport « Informatique et enseignement des mathématiques » dans lequel est proposée l'introduction d'une part d'informatique dans l'enseignement des sciences mathématiques et dans la formation des maîtres.

Dans la lignée de ce rapport, les programmes de lycée mis en place entre 2004 et 2005 ont introduit, dans les classes de première et terminale littéraire uniquement, un domaine « algorithmique » (en tant que domaine transversal, tout comme le domaine « logique »). On peut considérer cette introduction comme une expérimentation, l'enseignement mathématique dans les classes littéraires étant soumis à moins de contraintes. La réforme initiée en 2009 a repris un certain nombre d'éléments de ces programmes, en étendant cette introduction à toutes les filières du lycée général et à la seconde.

• *Réforme des programmes du lycée de 2009–2012*

En mars 2009, en vue d'une réforme des programmes du lycée général, est mise en place une consultation pour des programmes de mathématiques de seconde contenant une part d'algorithmique (*op. cit.*, note 7). En juin 2009 paraît un document-ressource en algorithmique pour la classe de seconde et fin juillet 2009 paraissent les programmes définitifs de seconde, à appliquer à la rentrée de septembre 2009[6]. Nous n'analyserons pas en détail les réactions et évolutions qui ont eu lieu durant la période, très courte, de consultation[7].

L'introduction de l'algorithmique a été un point de débat important dans les communautés liées à l'enseignement des mathématiques (associations d'enseignants, sociétés savantes, Commission Française pour l'Enseignement des Mathématiques, réseaux des Instituts de Recherche sur l'Enseignement des Mathématiques…).

Dans le projet en consultation, la partie algorithmique constituait une section à part entière du programme et se basait fortement sur le calcul sur les nombres entiers[8], cette entrée « Calcul

6 Les programmes du Ministère de l'Éducation nationale (MEN) datant de 2009 sont disponibles sur le site *Eduscol.education.fr* : « Programme d'enseignement de mathématiques de la classe de seconde générale et technologique » (*BO* n° 30 du 23/07/2009) [URL : https://www.education.gouv.fr/bo/2009/30/mene0913405a.htm]. Voir aussi les documents-ressources, suggestions d'activité à destination des enseignants : « Ressources pour la classe de seconde – Algorithmique » [URL : https://cache.media.eduscol.education.fr/file/Programmes/17/8/Doc_ress_algo_v25_109178.pdf].

7 DGESCO, « Consultation. Projet de programme de mathématiques. Classe de seconde générale et technologique », mars 2009. [URL : https://web.archive.org/web/20090319032830/http://eduscol.education.fr/D0015/consult_Maths.htm]. On peut trouver une trace de cette consultation ainsi que des réactions qu'elle a pu provoquer sur le site du café pédagogique [URL : http://www.cafepedagogique.net/lemensuel/lenseignant/sciences/maths/Pages/2009/102_ALAUNE.aspx] ; [URL : http://www.cafepedagogique.net/lexpresso/Pages/2009/05/Nxprogrammesmathsconsultation.aspx].

8 « *Il s'agit de familiariser les élèves avec les grands principes d'organisation d'un algorithme : gestion des*

sur les nombres entiers » a été supprimée à l'issue de la consultation, et l'algorithmique a retrouvé un statut transversal similaire à celui des programmes de 2004-2005, ce qui constitue une modification importante du projet initial.

Notons également que les parties d'algorithmique des programmes de première et de terminale, parus respectivement en 2010 et 2011, sont essentiellement identiques à celles des programmes de seconde de 2009. Nous n'y reviendrons pas et ferons référence à « la réforme de 2009 » pour englober l'ensemble de ces programmes[9].

Les attendus en termes d'apprentissage sont relativement modestes. Les élèves sont entraînés à « *décrire certains algorithmes en langage naturel ou dans un langage symbolique* », à en réaliser certains *« à l'aide d'un tableur ou d'un petit programme* » et à « *interpréter des algorithmes plus complexes* ». Les capacités attendues en fin de cycle concernent, par exemple, la capacité à écrire des « *instructions élémentaires (affectation, calcul, entrée, sortie)* » et à programmer des structures de « *boucle et itérateur, instruction conditionnelle* », le tout « *dans le cadre d'une résolution de problèmes* ».

entrées-sorties, affectation d'une valeur et mise en forme d'un calcul, en opérant essentiellement sur des nombres entiers » (p. 9).

9 De même, pour une partie des citations des programmes, lorsque nous ne précisons pas de page, c'est que les extraits sont identiques dans tous les programmes de la période, de la seconde à la terminale.

Une analyse détaillée de ces programmes, dont nous proposons une relecture synthétique dans la section suivante, est disponible dans (Modeste, 2012). Elle fait apparaître en particulier un certain amalgame entre les concepts d'algorithme et de programme, sur lequel nous reviendrons à plusieurs reprises dans cet article.

• *Spécialité « Informatique et sciences du numérique » (ISN)*

À la rentrée 2012, en même temps que se met en place le nouveau programme de terminale, est créé un enseignement de spécialité de Terminale Scientifique intitulé Informatique et Sciences du Numérique[10], que l'on peut considérer comme une introduction de l'informatique en tant que discipline scolaire au lycée. Cette introduction a lieu dans un contexte où les communautés de recherche et professionnelle en informatique militent pour la mise en place d'un tel enseignement. En 2013, paraît un rapport de l'Académie des Sciences (2013) appelant à une généralisation de l'enseignement de l'informatique.

• *Réforme du collège de 2016*

Une réforme des programmes de collège est entrée en vigueur à la rentrée 2016. L'informatique comme discipline

10 Programmes MEN : « Enseignement de spécialité d'informatique et sciences du numérique de la série scientifique – classe terminale » (*BO* spécial n° 8 du 13 octobre 2011) [URL : https://www.education.gouv.fr/bo/11/Special8/MENE1119484A.htm].

scolaire y apparaît explicitement pour la première fois, son enseignement, au collège, est dispensé à la fois en mathématiques et en technologie. Au cycle 4, les contenus relevant du domaine informatique apparaissent essentiellement dans les thèmes « L'informatique et la programmation » en technologie, et « Algorithmique et programmation » en mathématiques[11].

En mathématiques, ce nouveau programme met explicitement l'accent sur l'activité de programmation proprement dite, vu comme un thème à part entière du programme, non nécessairement lié aux autres thèmes mathématiques, et en distinguant les termes algorithme et programme. Comme cela apparaît clairement dans les recommandations officielles (voir en particulier celles de 2016[12]), il est proposé d'initier en priorité les élèves à la programmation à l'aide d'un langage de programmation événementiel par blocs, au premier rang desquels le logiciel *Scratch*[13] fait figure de langage de référence, ce qui constitue une autre différence importante avec les programmes du lycée.

- *Aménagement des programmes du lycée de 2017*

En mai 2017, suite à la mise en œuvre simultanée de la réforme sur les trois années du cycle 4, est paru un aménagement des programmes de mathématiques de seconde, pour une mise en œuvre dès la rentrée 2017 et une stabilisation prévue en 2019[14]. Tout comme en cycle 4 et en contraste avec les programmes précédents du lycée, il y est explicitement fait mention de programmation, et le domaine Algorithmique et programmation gagne le statut de thème à part entière. Le document-ressource accompagnant cet aménagement des programmes précise cependant : « *À la différence du programme de mathématiques du cycle 4 du collège, il s'agit donc d'adosser explicitement les activités de la partie algorithmique et programmation aux mathématiques*[15] », ce que confirment les exemples traités, allant bien au-delà du niveau de la classe de seconde, orientés essentiellement vers l'analyse numérique et la simulation en probabilités et statistiques.

11 Documents-ressources MEN 2015 : « Programme d'enseignement du cycle des approfondissements (cycle 4) » (*BO* spécial n° 11 du 26 novembre 2015) [URL : https://www.education.gouv.fr/bo/15/Special11/MENE1526483Aannexe3.htm].

12 Documents-ressources MEN 2016 : « Algorithmique et programmation » [URL : https://cache.media.eduscol.education.fr/file/Algorithmique_et_programmation/67/9/RA16_C4_MATH_algorithmique_et_programmation_N.D_551679.pdf].

13 Site du logiciel *Scratch* [URL : https://scratch.mit.edu/].

14 Documents-ressources MEN 2017 : « Ressources pour le lycée – Mathématiques – Algorithmique et programmation » [URL : https://cache.media.eduscol.education.fr/file/Mathematiques/73/3/Algorithmique_et_programmation_787733.pdf]. Par contraste avec les langages de programmation par blocs utilisés au cycle 4. Voir le programme MEN 2017 : « Seconde générale et technologique – Aménagements des programmes d'enseignement de mathématiques et de physique-chimie », p. 10 (*BO* n° 18 du 4 mai 2017) [URL : https://www.education.gouv.fr/bo/17/Hebdo18/MENE1712512C.htm].

15 *Ibid*., p. 1.

Le programme s'appuie sur « *deux idées essentielles : la notion de fonction d'une part, et la programmation comme production d'un texte dans un langage informatique d'autre part*[16] ». À l'opposé de l'approche non prescriptive des programmes de 2009-2012, il introduit également des consignes précises concernant le choix d'un langage de programmation :

> Un langage de programmation simple d'usage est nécessaire pour l'écriture des programmes. Le choix du langage se fera parmi les langages interprétés, concis, largement répandus, et pouvant fonctionner dans une diversité d'environnements (*ibid.*, p. 10).

Cette description désigne sans le nommer le langage Python, déjà très répandu dans les communautés éducatives de nombreux pays (y compris dans l'enseignement supérieur français) et utilisé de manière explicite (et exclusive) dans le document-ressource[17].

• *Aménagements transitoires pour le bac 2018*

En 2017, en réponse à la situation transitoire que produit l'aménagement des programmes, un document diffusé par les canaux officiels propose une modification de l'écriture des algorithmes au baccalauréat[18]. Ce document sans en-tête ni présentation, propose la réécriture de 5 algorithmes donnés au bac 2017 selon de nouvelles conventions, explicitant pour chaque cas les nouvelles attentes. On peut y lire une volonté de structurer le langage de description des algorithmes. Parmi les changements notables on note une « *simplification de la syntaxe* » (expression utilisée à de nombreuses reprises) qui se traduit par :

- La suppression de la déclaration de variable, remplacée par une précision dans l'énoncé des « *hypothèses faites sur les variables* » et le fait de « *renoncer aux entrées sorties* » (les instructions « Saisir » et « Afficher » disparaissent). Ainsi, par exemple, S étant une variable saisie et λ une variable affichée dans le bloc « Sortie », on ne pose plus la question « Quelle valeur affiche cet algorithme si on saisit la valeur $S = 0{,}8$? » mais « Si la variable S contient la valeur 0,8 avant l'exécution de cet algorithme, que contient la variable λ à la fin de son exécution ? » ;

- La suppression des rubriques « entrée », « variables », « initialisation », « traitement » et « sortie » ;

- Le remplacement des instructions « prend la valeur » ou « affecter à » par le symbole « ← » ;

16 MEN, « Seconde générale et technologique », 2017, *op. cit.*

17 MEN, « Ressources pour le lycée », 2017, *op. cit.*

18 « Évolution de l'écriture des algorithmes au baccalauréat ». Ce document n'a pas de statut très clair, mais on en retrouve de nombreuses instances sur des sites académiques tels que le site disciplinaire de mathématiques de l'académie de Créteil – mis en ligne le 18/10/2017 [URL : https://maths.ac-creteil.fr/spip.php?article232].

- Le remplacement de mots en français par des symboles mathématiques (par exemple « supérieur à 1 » devient « ≥1 »).

On peut noter que les mots-clés « Fin Pour » et « Fin Tant que » sont conservés, et que, malgré tout, il n'y a pas complète uniformité de langage dans les algorithmes réécrits.

- *Réforme du lycée de 2019-2020*

Suite à une consultation initiée à la fin de l'année 2017 et pendant une partie de 2018, de nouveaux programmes de seconde et première du lycée général et technologique ont été publiés en janvier 2019, puis pour la terminale en juillet 2019[19]. Cette réforme a amené à la disparition des séries S, ES et L du lycée général, au profit d'un choix de matières dites « de spécialité » par les élèves : trois matières d'un volume de quatre heures par semaine en première, et deux d'un volume de six heures en terminale.

Le thème « Algorithmique et programmation » de seconde conserve essentiellement le même contenu que celui de 2017. Il précise les contenus visés (variables, instructions et structures de contrôle élémentaires, fonctions) et prescrit Python comme langage de référence. Le programme mentionne explicitement les « types de variables » à savoir utiliser : booléens, entiers, flottants et chaînes de caractères. Les listes sont introduites en première générale, pour les élèves ayant choisi la spécialité Mathématiques. Aucune notion nouvelle n'est introduite en terminale.

Deux matières d'informatique proprement dite ont été créées simultanément, un enseignement obligatoire « Sciences numériques et technologie » (SNT) en seconde, et une spécialité « Numérique et sciences informatiques » (NSI) en première et terminale, élargissant considérablement la portée de l'ISN de 2012. L'analyse des programmes de ces deux enseignements, dont l'un relève plutôt de la culture numérique et l'autre de science informatique, ne sera pas détaillée ici.

Analyse de la transposition didactique

Cette section propose une analyse de la transposition didactique à l'œuvre dans le curriculum de mathématiques de lycée au cours de la période étudiée, en appui sur les travaux précédents du domaine, notamment (Modeste, 2012 ; 2020), et en appui sur les éléments épistémologiques de la deuxième partie.

19 Programmes MEN 2019 : « Programme d'enseignement de mathématiques de la classe de seconde générale et technologique » (*BO* spécial n° 1 du 22 janvier 2019) [URL : https://www.education.gouv.fr/bo/19/Special1/MENE1901631A.htm] ; « Programme d'enseignement de spécialité de mathématiques de la classe de première de la voie générale » (*BO* spécial n° 1 du 22 janvier 2019) [URL : https://www.education.gouv.fr/bo/19/Special1/MENE1901632A.htm] ; « Programme d'enseignement de spécialité de mathématiques de la classe terminale de la voie générale » (*BO* spécial n° 8 du 25 juillet 2019) [URL : https://www.education.gouv.fr/bo/19/Special8/MENE1921246A.htm].

- *Présentation et définition du concept d'algorithme*

Le programme de seconde de 2009 ne définit pas formellement le concept d'algorithme, mais en donne quelques exemples supposés avoir été rencontrés par les élèves au collège (« *algorithmes opératoires, algorithme des différences, algorithme d'Euclide, algorithmes de construction en géométrie* »), ainsi que quelques aspects généraux : « *Il s'agit de familiariser les élèves avec les grands principes d'organisation d'un algorithme : gestion des entrées-sorties, affectation d'une valeur et mise en forme d'un calcul*[20] ».

Dans ce passage, les concepts d'entrées-sorties et d'affectation sont identifiés comme constitutifs, et même caractéristiques, de celui d'algorithme. Ces concepts, qui relèvent plutôt d'une *certaine* pratique de la programmation (en particulier de style dit *impératif*), ne sont pas forcément essentiels à l'activité ou au raisonnement algorithmique eux-mêmes. Il est difficile de déterminer quelles références sont à l'origine de cette conception de l'algorithmique, même si l'on en trouve des traces dans d'autres écrits francophones des années 1980 (voir par exemple Samurçay, 1985).

Le document-ressource lié[21] donne une description légèrement plus détaillée du concept d'algorithme tel qu'il est envisagé en classe de seconde. Il s'appuie en particulier sur une définition attribuée à l'*Encyclopedia Universalis* :

> On définit parfois les algorithmes de la manière suivante : « *un algorithme est une suite finie de règles à appliquer dans un ordre déterminé à un nombre fini de données pour arriver, en un nombre fini d'étapes, à un certain résultat et cela indépendamment des données.* » Le résultat doit donc s'obtenir en un temps fini[22].

Cette définition initiale est ensuite détaillée dans le document, en portant une attention particulière aux « *algorithme*[s] *simple*[s] », c'est-à-dire « *ne comportant pas de fonctions ou sous-programmes*[23] ». Selon les auteurs du document, la conception d'un tel algorithme obéit à trois grandes étapes : 1) préparation du traitement (incluant le repérage, l'agencement et éventuellement l'entrée des données, et même le repérage de données intermédiaires), 2) traitement proprement dit, et 3) sortie des résultats.

Dans le programme de mathématiques de cycle 4 mis en application à partir de la rentrée 2016, un accent est mis sur certains aspects alors absents des programmes de lycée. Par exemple, les tout premiers attendus de fin de cycle mentionnés sur le thème Algorithmique et programmation sont « *décomposer un problème en sous-problèmes afin de structurer un programme* », et « *reconnaître des schémas* ». Comme évoqué à la section précédente, ces deux

20 MEN, « Programme d'enseignement de mathématiques de la classe de seconde générale et technologique », 2009, *op. cit.*

21 MEN, « Ressources pour la classe de seconde – Algorithmique », 2009, *op. cit.*

22 *Ibid.,* p. 6.

23 *Ibidem.*

compétences sont identifiées par la littérature comme caractéristiques de la pensée informatique, et sont essentiellement absentes des textes encadrant les programmes de lycée, de par la conception de l'algorithme qu'ils véhiculent.

Suite à l'aménagement des programmes de 2017, on constate une très nette évolution dans la façon dont le concept d'algorithme est abordé. On trouve par exemple dans le document-ressource la définition suivante[24] :

> Un algorithme est une procédure de résolution de problème, abstraction faite des caractéristiques spécifiques qu'il peut revêtir. Par exemple, un algorithme de tri ne résout pas le problème du tri d'un jeu particulier de données mais a pour objectif de trier n'importe quel jeu de données : le problème du tri s'applique à différentes instances, c'est-à-dire à différents jeux de données.
>
> Un algorithme s'applique donc 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[25].

Il est intéressant de comparer cette définition à celle proposée par Modeste (2012) et citée dans la section précédente. Il est significatif que ce document officiel reprenne à son compte l'idée fondamentale de problème algorithmique, et fasse apparaître très clairement la notion d'instance d'un problème et celle de résultat. L'objet algorithme prend ici un sens mathématique précis, contrairement à la définition informelle qui prévalait depuis 2009.

• *Algorithmique et programmation*

Dans les curriculums mis en place en 2009-2012, au sujet de l'« entrée des données », il est précisé qu'elle « *peut se manifester par la saisie de caractères ou de nombres sur le clavier, ou la lecture de la position du pointeur de la souris, ou encore par la lecture d'un fichier contenant ces nombres ou caractères*[26] ». La description de l'étape de « *sortie des résultats* » nous paraît également significative :

> Les résultats obtenus peuvent être affichés sur l'écran, ou imprimés sur papier, ou bien encore conservés dans un fichier. Si on n'en fait rien, ils « restent » en mémoire jusqu'à la prochaine exécution ou sont perdus. À l'occasion, la sortie pourrait être graphique (afficher ou déplacer le pointeur de la souris ou des objets sur l'écran) ou sonore… voire sur Internet[27].

On constate que les auteurs font appel à un champ lexical relevant d'une réalité technologique concrète ainsi qu'à un utilisateur interagissant par le biais d'un dispositif technique. Ceci semble favoriser une vision du concept d'algorithme en tant que texte d'une procédure exécutable sur une certaine « machine ».

24 MEN, « Ressources pour le lycée », 2017, *op. cit.*

25 *Ibid*., p. 3.

26 MEN, « Programme d'enseignement de mathématiques de la classe de seconde générale et technologique », 2009, *op. cit.*

27 *Ibidem*.

Ceci nous semble davantage correspondre au concept de programme informatique qu'à celui d'algorithme. Modeste (2012) décrit ce phénomène consistant à faire écrire hors d'un dispositif d'exécution des algorithmes contenant des instructions fortement liées à une machine, et nomme ces produits de la transposition didactique « programmes-papier », considérés comme caractéristiques d'un écrasement des notions d'algorithme et de programme.

- *Problèmes et algorithmes*

On peut noter dans le discours des documents officiels de 2009-2012 que les compétences associées à la résolution de problème ne sont pas spécifiquement de nature algorithmique. En réalité, le terme « problème » utilisé est à entendre dans un sens général, et non dans le sens particulier de *problème algorithmique*, qui reste lui non-défini. L'algorithme est donc entendu non pas (ou pas uniquement) comme la solution d'un problème algorithmique, mais comme outil mobilisable dans l'étude expérimentale d'un problème mathématique ou scientifique. Ceci correspond plutôt au rôle de la programmation dans une démarche expérimentale ou d'investigation, au sens où c'est bien l'exécution d'instructions et de calculs (impossibles à la main) par une machine programmée qui permet de générer des exemples ou des contre-exemples, de simuler des modèles ou de tester des conjectures, et non l'algorithme en lui-même en tant que méthode. Le rapport du CREM (Kahane, 2002) contrairement aux programmes, identifiait bien cette distinction.

De plus, la notion d'instance ou de paramètres d'un problème n'apparaît qu'en filigrane (par exemple de manière implicite dans la description de l'étape de préparation des données en vue du traitement) comme l'ensemble des données initialement disponibles ou saisies par « l'utilisateur ». De même, la notion formelle de résultat d'un algorithme peut être vue comme identifiée à la suite intégrale des effets, « sorties » ou autres manifestations observables effectuées par l'algorithme (ou plutôt par le programme qui l'implémente).

À ce titre, le document « Évolutions de l'écriture des algorithmes au bac » de 2017 (*supra,* note 16) propose des évolutions assez caractéristiques : tout en semblant s'éloigner d'un langage de programmation par une simplification de la syntaxe et la suppression des instructions « saisir » et « afficher », elles suppriment la structuration en entrées-sortie qui caractérisent l'algorithme, et proposent ainsi un point de vue où, suite à un certain nombre de commandes, les variables d'un environnement sont modifiées, sans distinctions entre elles (effets de bord). Ceci montre que subsiste une difficulté à distinguer le concept d'algorithme de celui de programme dans le discours de l'institution.

- *Présentation et représentation des algorithmes*

Le niveau de description des algorithmes proposé dans les programmes de lycée de la réforme de 2009-2012 est « *une formalisation en langage naturel propre à donner lieu à traduction sur*

une calculatrice ou à l'aide d'un logiciel ». La question des niveaux de description n'est ici pas complètement claire. Le programme annonce tout d'abord proposer une « *formalisation en langage naturel* » (expression potentiellement contradictoire) avant d'évoquer la possibilité d'un « *langage symbolique* », sans plus de précision. Par ailleurs, la notion de langage de programmation est évoquée par le biais des calculatrices et logiciels.

Les programmes explicitent le fait qu'aucun langage n'est imposé, ce qui semble aller à l'encontre de l'idée de formalisation. Se pose également la question de l'évaluation des apprentissages des élèves, notamment dans le contexte du baccalauréat : comment par exemple évaluer la capacité des élèves à « *interpréter des algorithmes* » s'il n'existe pas de langage commun à tous les élèves pour les écrire, ni de sémantique pour en définir l'interprétation attendue ?

Le choix d'un pseudo-langage relativement normalisé dans les documents institutionnels ne garantit en aucun cas la cohérence de la présentation des algorithmes[28]. En l'absence d'une description claire de la sémantique des conventions utilisées (particulièrement en ce qui concerne le statut des variables), l'usage d'un tel niveau de description est même susceptible d'introduire un grand nombre d'implicites potentiellement sources de difficultés.

Sur le plan de la représentation des algorithmes le document « Évolutions de l'écriture des algorithmes au bac » de 2017 est aussi éclairant sur la difficulté des curriculums à se positionner. On y trouve :

- des modifications syntaxiques sans changements majeurs quant à la sémantique (comme l'introduction de « ← ») ;
- la conservation de convention des instructions « Fin… » sans utilité sémantique en présence d'indentation ;
- des changements présentés comme simplement syntaxiques alors que leur portée sémantique est d'envergure, comme la suppression de la déclaration des entrées-sorties (avec mention des variables et des hypothèses les concernant renvoyées aux énoncés), ce qui revient à une indifférenciation *a priori* des statuts fonctionnels des variables de l'algorithme.

Conclusion de la section

Entre les périodes 2009-2012 et 2017-2019, les orientations des programmes d'enseignement et leur transposition didactique des concepts liés à l'algorithmique et la programmation ont connu des changements rapides et radicaux. Après un amalgame entre algorithmes et programmes dans les textes

28 L'annexe A, sur le site de la revue, présente une analyse détaillée du pseudo-langage utilisé dans le document-ressource accompagnant les programmes de seconde de 2009 (*cf.* ressources 2009 du MEN).

de la période 2009-2012 et dans les ressources et manuels (Modeste, 2012), les aménagements curriculaires de 2017 ont choisi comme référence une définition de l'algorithme prenant davantage en compte la notion de « problème algorithmique », et en particulier sa dimension fonctionnelle : à toute instance du problème qu'il résout, l'algorithme associe un résultat correspondant.

Après une relative invisibilité, volontaire ou fortuite, de la programmation et de certains aspects techniques qui y sont liés (syntaxe, environnement matériel, etc.), celle-ci fait désormais partie des contenus visés et constitue un aspect explicite d'un thème à part entière des programmes.

Après une absence de normalisation de la présentation des algorithmes et un recours à une diversité de langages et outils de programmation *ad hoc*, les enseignements de mathématiques de lycée s'appuient aujourd'hui sur un langage de référence, Python, de notoriété internationale et dont l'usage dépasse le contexte de l'enseignement.

Les évolutions du discours institutionnel sur le savoir à enseigner montrent une prise en compte progressive de certains enjeux épistémologiques et didactique relatifs à l'algorithmique, cependant on note qu'à la fin de la période étudiée subsistent encore un certain nombre de points de tension épistémologiques.

Nous verrons dans la suite de l'article en quoi le logiciel *Algobox* est un témoin de premier plan de ces changements successifs, depuis son apparition en 2009 à la suite de la réforme, sa large adoption par les manuels et les enseignants dans les années qui ont suivi, jusqu'à sa quasi-disparition à partir de 2017 au profit du langage Python.

Analyse du logiciel *AlgoBox*

Après cette analyse de la transposition didactique en jeu au lycée, nous nous intéressons au logiciel *AlgoBox* conçu et largement utilisé lors de la période 2009-2019. Après une rapide présentation générale de l'interface du logiciel et de son utilisation et un historique de son évolution, nous décrivons en détail quelques-uns de ses choix de conception (apparents ou revendiqués), ce qui nous permettra d'en tirer une analyse de la transposition informatique du concept d'algorithme à l'œuvre dans ce logiciel.

Présentation générale

Le logiciel libre *AlgoBox* a été conçu et développé à partir de 2014, de manière essentiellement autonome, par Pascal Brachet, professeur de mathématiques au lycée. Il est également l'auteur de nombreuses fiches pédagogiques distribuées librement en ligne, et de plusieurs autres logiciels libres, dont l'éditeur LaTeX Texmaker[29].

29 Toutes ces ressources sont accessibles sur le site de Pascal Brachet [URL : https://www.xm1math.net].

Aperçu d'utilisation

Le principe du logiciel *AlgoBox* est de permettre la création du texte d'un algorithme, rédigé dans une syntaxe évoquant un pseudo-langage algorithmique de type impératif très proche des recommandations du document-ressource[30].

Le logiciel est constitué d'un environnement de travail (interface graphique contenant menus, boutons, zones de rétroaction textuelles et graphiques) et d'un langage textuel permettant l'écriture d'algorithmes, que nous surnommerons par commodité « langage *AlgoBox* ». En termes d'apparence et d'ergonomie, l'interface est proche de celle d'un environnement de développement intégré classique (*Integrated Development Environment* ou IDE), dont elle reprend certaines conventions : coloration syntaxique, repliement de portées, etc.

Le logiciel propose deux modes de rédaction d'algorithmes. Le premier, que nous qualifierons de « mode classique » (*cf.* figure 1), fonctionne à l'aide d'une série de boutons. Chaque appui ouvre une fenêtre de dialogue permettant de spécifier les divers paramètres d'une nouvelle instruction, qui est ensuite insérée à la ligne courante. L'utilisateur ne saisit donc pas intégralement le texte de l'instruction. Un bouton est grisé lorsque la ligne courante n'autorise pas l'insertion d'une instruction de ce type.

Le second mode de rédaction ou « mode éditeur », permet de saisir directement le texte de l'algorithme, et propose les outils classiques d'un environnement de programmation moderne (complétion automatique de mots-clés, aide à la composition). Des boutons permettent de vérifier la conformité syntaxique de l'algorithme ou de basculer vers le mode classique.

Une fois un algorithme composé, il est possible de l'exécuter, c'est-à-dire d'en observer concrètement les effets sur la machine, en cliquant sur le bouton « Tester Algorithme ». Ceci provoque l'ouverture d'une nouvelle fenêtre (*cf.* figure 2) où est repris le texte de l'algorithme, et où apparaissent progressivement les diverses invites de saisie et affichages provoqués par son exécution. Des outils permettent de contrôler le processus, par exemple en procédant à une exécution pas à pas permettant de visualiser les valeurs successives des variables.

Choix de design et philosophie

AlgoBox se présente[31] comme un logiciel « *gratuit, libre, multi-plateforme et facile d'utilisation* » et « *100 % conforme aux programmes du lycée et du collège* » pour l'initiation à l'algorithmique dans le contexte des programmes de mathématiques de collège et de lycée. Il met

30 MEN, « Programme d'enseignement de mathématiques… », 2009, *op. cit.*

31 Toutes les citations de cette section sont issues du site *AlgoBox* [URL : https://www.xm1math.net/algobox/].

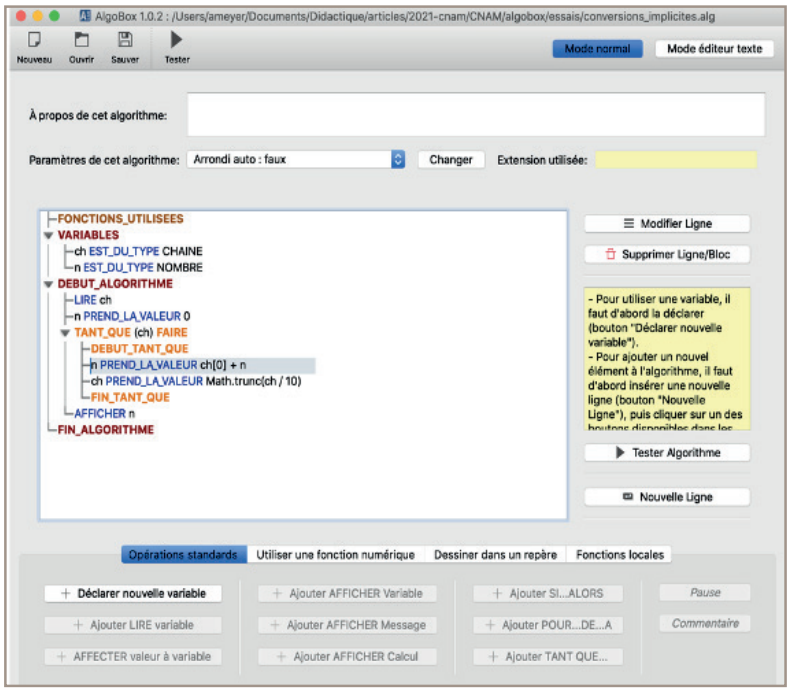

Figure 1 – Fenêtre principale d'*AlgoBox* (mode classique)

Figure 2 – Fenêtre d'exécution

en avant un certain nombre de caractéristiques, dont les suivantes :

- « *Un apprentissage centré sur la logique, pas sur les problèmes de syntaxe* » ;
- « *Une interface en français, moderne et ergonomique* » ;
- « *La syntaxe la plus proche de celle des exercices au Bac* » ;
- « *Une approche réaliste, pédagogique et non élitiste de l'algorithmique* ».

À la lecture de ces arguments, il apparaît que la vocation principale de ce logiciel est de répondre aux besoins introduits par la réforme de 2009. Il est également clair que la philosophie de ce logiciel est de permettre aux enseignants de mathématiques et à leurs élèves de travailler sur les notions algorithmiques en faisant l'économie de l'apprentissage d'une syntaxe nouvelle. Il est ici sous-entendu que la syntaxe du langage *AlgoBox* « va de soi », du fait d'être si « *proche du langage naturel algorithmique* » (à supposer qu'un tel langage existe et soit unique).

Comme on le détaillera par la suite, on peut aussi supposer à la lecture des divers documents relatifs à ce logiciel que les algorithmes composés sous *AlgoBox* se veulent idéaux, débarrassés des contingences et tracasseries associées (à tort ou à raison) à l'activité de programmation informatique : nécessité de familiarisation avec des procédures techniques inhabituelles, prise en compte des contraintes ou limites inhérentes à une architecture matérielle, à une modélisation et représentation informatique des données, etc. Une pratique trop technique ou technologique de la programmation apparaîtrait ainsi comme « *élitiste* ». *AlgoBox* se veut donc un logiciel « *démocratique* ».

La lecture des recueils d'activités publiés par l'auteur lui-même pour les classes de première et terminale permet de nuancer ces revendications d'« évidence » : une annexe fournie, d'une dizaine de pages, y présente les « *équivalences entre pseudo-codes* », « problèmes de syntaxe » les plus fréquents et astuces de fonctionnement d'*AlgoBox*[32].

Ce logiciel fournit cependant plus qu'une simple interface de rédaction « *pas à pas* […] *hiérarchique et structurée* » d'algorithmes. La grande force revendiquée par ce logiciel est de permettre à la fois la composition d'algorithmes et la capacité à les faire fonctionner (les « *tester* » ou les exécuter) comme on ferait fonctionner des programmes. Il s'agit donc d'une véritable tentative de transposition informatique du concept d'algorithme. Nous montrerons que cette ambition se heurte à des difficultés d'ordres épistémologique et didactique.

32 Voir les brochures rédigées par P. Brachet : « Algorithmique en classe de première avec *AlgoBox* » [URL : https://www.xm1math.net/algobox/algobook.html] ; « Algorithmique en classe de terminale avec *AlgoBox* » [URL : https://www.xm1math.net/algobox/algobook.html].

Afin de caractériser la conception de l'algorithmique qui sous-tend le logiciel *AlgoBox*, le reste de cette section est consacré à un rappel chronologique de l'apparition de ses différentes fonctionnalités en regard avec l'évolution des programmes, et à une étude plus précise des caractéristiques techniques d'*AlgoBox*.

Historique des versions

La première version d'*AlgoBox* (version 0.1) a été rendue publique au début du mois de juillet 2009[33], quelques mois après la publication des projets de nouveaux programmes de mathématiques de seconde et la consultation qui l'a accompagnée[34]. Elle a également suivi de peu la publication du premier document d'accompagnement et des programmes définitifs[35]. Cette version fournissait déjà la plupart des fonctionnalités de base du logiciel, mais elle a depuis connu un certain nombre d'évolutions.

En quelques mois, *AlgoBox* a connu plusieurs mises à jour : sortie des versions 0.2 et 0.3 l'été 2009, puis 0.4 en octobre 2009. Le rythme s'est ensuite stabilisé à une nouvelle version par an environ, jusqu'à la version 0.9 en 2014. Enfin, après une pause de trois ans, la version 1.0 d'*AlgoBox* est apparue à l'été 2017, elle n'a connu que des mises à jour techniques mineures depuis[36].

Nous soulignons ici quelques évolutions représentatives des liens entre les évolutions curriculaires et les versions d'*AlgoBox* : l'apparition en 2010 (version 0.5) et 2011 (version 0.6) dans le logiciel de nouvelles fonctions mathématiques prédéfinies (calculs statistiques sur les listes en 2010, lois et distributions en 2011) coïncidant avec l'introduction de l'algorithmique dans les programmes de première et de terminale, incluant ces notions de statistiques et de probabilités ; et l'apparition des fonctions dites « locales » en 2017 (version 1.0) de façon concomitante à l'adaptation des programmes de seconde suite à la réforme du collège, et l'introduction de la notion de fonction informatique.

Caractéristiques techniques

Nous présentons ici une synthèse des caractéristiques techniques d'*AlgoBox* qui témoignent de choix spécifiques

33 Tous ces éléments factuels sont issus de la page « Changelog » du site *AlgoBox* [URL : https://www.xm1math.net/algobox/changelog.html], ainsi que des archives de code source disponibles sur la page [URL : https://www.xm1math.net/algobox/download.html].

34 DGESCO, « Consultation. Projet de programme de mathématiques. Classe de seconde générale et technologique », 2009, *op. cit.*

35 MEN, « Programme d'enseignement de mathématiques… » et « Ressources pour la classe de seconde – Algorithmique », 2019, *op. cit.*

36 Nous détaillons dans l'annexe B, sur le site de la revue, une sélection de modifications apportées au logiciel, en nous focalisant sur les évolutions liées à la syntaxe et à la sémantique du langage, et en les mettant en regard certains éléments de l'évolution des curriculums, lorsque cela nous semble éclairer celle du logiciel.

ou d'une certaine prise de distance avec le savoir savant[37].

Le langage *AlgoBox* se veut lisible et compréhensible par tout élève, sans apprentissage particulier de sa syntaxe. Il emploie donc essentiellement des mots-clés en français. Par exemple, le terme « prend la valeur », choisi pour l'affectation, est conforme à l'expression utilisée par le document d'accompagnement de 2009[38]. Dans l'ensemble, le langage *AlgoBox* est effectivement très proche de ce pseudo-langage officiel.

Une fois un algorithme décrit, il est possible de l'exécuter afin d'en observer le fonctionnement. Ceci est rendu possible par la traduction ligne à ligne de l'algorithme en un programme JavaScript, puis son exécution au sein d'une page web affichée dans la fenêtre d'exécution (*cf.* figure 2). Le langage *AlgoBox* s'apparente donc à une surcouche fine de JavaScript plutôt qu'à un langage de programmation à part entière.

Un algorithme est constitué dans *AlgoBox* de plusieurs rubriques : un bloc *FONCTIONS_UTILISEES* (depuis la version 1.0), un bloc *VARIABLES*, et le corps de l'algorithme lui-même, délimité par les mots-clés *DEBUT_ALGORITHME* et *FIN_ALGORITHME*. De même, tous les blocs (de boucles, de conditionnelles) sont délimités par des marqueurs explicites *DEBUT_…* et *FIN_…* . Dans le mode de travail « classique », *AlgoBox* représente l'imbrication des blocs et instructions sous une forme arborescente et indentée, proche de la présentation des algorithmes dans le document-ressource de 2009[39], comme on le voit en figure 1. Sur le plan technique, cette forme arborescente pourrait se passer de délimiteurs explicites, ce qui permettrait d'alléger la syntaxe.

Depuis sa version 1.0, *AlgoBox* permet à l'utilisateur de définir des fonctions dites « *locales* », qui peuvent ensuite être utilisées dans l'algorithme principal. Même si cette fonctionnalité permet effectivement de manipuler des fonctions personnalisées (principalement numériques), elle reste à un stade de développement très incomplet.

Dans la description d'un algorithme, le bloc VARIABLES recense toutes les variables de l'algorithme et définit leur type. Une telle contrainte n'existe pas dans le langage-cible JavaScript, dans lequel une variable peut être introduite en tout point du programme, et sans en spécifier le type (JavaScript obéissant à un typage dynamique). Cette différence de philosophie entraîne un certain nombre d'effets indésirables.

37 Ces caractéristiques sont analysées plus en détail dans l'annexe C, sur le site de la revue (Caractéristiques techniques détaillées).

38 MEN, « Programme d'enseignement de mathématiques… », 2009, *op. cit.*

39 MEN, « Ressources pour la classe de seconde – Algorithmique », 2009, *op. cit.*

AlgoBox permet en principe de manipuler trois types de valeurs : nombres, listes de nombres et chaînes de caractères. Ceci implique en particulier qu'il n'existe pas de type « nombre entier », ce qui peut sembler étonnant dans ce contexte.

Ainsi, *AlgoBox* présente certaines caractéristiques potentiellement problématiques. Malgré une réelle proximité avec l'esprit du pseudo-langage employé dans le document-ressource[40], le langage *AlgoBox* est entaché d'une certaine hétérogénéité (de langue, de style et de sémantique). Sa syntaxe n'est pas entièrement circonscrite, certaines instructions pouvant faire appel à du code JavaScript arbitraire. Un certain nombre d'effets indésirables relatifs à une insuffisance du contrôle de type sont également visibles. Nous y voyons les conséquences non maîtrisées d'une traduction imparfaite des algorithmes vers Javascript.

Selon nous, l'absence de spécification formelle de la syntaxe, le non-emploi de techniques de compilation appropriées pour la traduction des algorithmes, et le choix de déléguer l'analyse et l'exécution du code à un interpréteur JavaScript obéissant à ses propres contraintes techniques, produisent certains comportements à première vue incompréhensibles. Ces aspects d'*AlgoBox* sont susceptibles d'entraîner des difficultés conséquentes chez certains utilisateurs, particulièrement les débutants auxquels s'adresse en priorité ce logiciel.

Ces observations nous confortent dans l'idée que le langage et le mécanisme d'exécution d'*Algobox* souffrent d'une importante imprécision et d'un caractère hétérogène, voire imprévisible, et remet en question le projet d'*Algobox* de se présenter comme libéré des difficultés techniques supposées d'un « véritable » langage de programmation.

À un moment de l'histoire de l'enseignement de l'informatique et à une place du cursus où ces activités algorithmiques et de programmation étaient entièrement nouvelles pour un grand nombre d'élèves et d'enseignants, il est légitime de s'interroger sur l'impact qu'un tel outil a pu avoir sur les conceptions de ses utilisateurs (élèves ou enseignants). Dans ce sens, nous allons nous intéresser à la transposition informatique du concept algorithme réalisée par le logiciel *AlgoBox*.

Analyse de la transposition informatique

Le spectre thématique visé par *AlgoBox* est très nettement ancré dans le programme de mathématiques du lycée, comme en témoigne la palette de ses fonctionnalités (fonctions prédéfinies, repère graphique, etc.) ainsi que toute la documentation qui l'accompagne. Comme nous l'avons indiqué précédemment, *AlgoBox* ne se présente pas comme un environnement d'apprentissage de la programmation mais comme un logiciel « *d'aide à l'élaboration et à l'exécution d'algorithmes dans l'esprit*

40 *Ibidem.*

des nouveaux programmes de mathématiques du secondaire », proposant « *un apprentissage centré sur la logique, pas sur les problèmes de syntaxe* ». Cette déclaration d'intention nous semble relever d'un projet de transposition informatique du concept d'algorithme (et donc, dans une certaine mesure, d'une transposition didactique), que nous nous proposons d'analyser.

- *Absence de repérage du statut fonctionnel des variables*

Il n'existe aucune distinction *a priori* dans *AlgoBox* entre les différents statuts fonctionnels des variables évoqués au paragraphe « Repères épistémologiques ». Toutes les variables sont déclarées dans la rubrique VARIABLES. Il est donc impossible de déterminer lesquelles sont utilisées pour représenter « l'entrée », ou instance courante du problème (*variables-paramètres*), lesquelles, si elles existent, servent à construire le résultat final (*variables-résultat*), et lesquelles sont de simples variables de travail. En réalité, aucun mécanisme autre que les instructions de saisie et d'affichage ne permet à un algorithme *AlgoBox* de spécifier des paramètres de l'algorithme ou de renvoyer un résultat.

Une reconstruction *a posteriori* de ces catégories pourrait donc consister à considérer les variables qui font l'objet d'une saisie comme des variables-paramètres, celles qui font l'objet d'un affichage comme des variables-résultats, et les autres comme des variables de travail. Malheureusement ce critère se révèle insuffisant, notamment lorsque des affichages ou des saisies sont effectués pour de simples raisons « cosmétiques » ou d'ergonomie (affichage et choix dans un menu, invitations à « arrêter ou continuer », etc.), qui ne répondent pas à la *spécification* du problème algorithmique posé.

Cette impossibilité d'identification claire des paramètres et résultats entretient une confusion entre la spécification d'un algorithme et l'interface utilisateur d'un « logiciel ». Ceci nous semble problématique, du fait que la distinction entre les différents statuts fonctionnels des variables et la compréhension des différents rôles joués par des saisies et affichages sont des points notoirement délicats de l'enseignement de la programmation aux débutants.

- *Confusion algorithme-programme*

Un autre axe d'analyse de la transposition informatique à l'œuvre dans *AlgoBox* concerne la distinction (ou la confusion) entre les concepts d'algorithme et de programme. Le logiciel, dans son ambition de libérer l'élève débutant des difficultés liées à l'apprentissage de la syntaxe d'un langage de programmation, présente donc le paradoxe de vouloir proposer un langage à la fois « évident » et exécutable sur machine. Or, il nous semble possible que cette ambition soit d'emblée vouée à l'échec, car comme nous l'avons vu, les

concepts d'algorithme et de programme sont distincts et possèdent leurs caractéristiques propres.

Comme nous l'avons rappelé dans la section consacrée à l'analyse des programmes, et conformément à l'étude approfondie menée dans (Modeste, 2012, chap. 6), les programmes de la période 2009-2012 font apparaître un important amalgame entre ces deux concepts. Cette étude y décèle aussi une tentative de résumer l'activité algorithmique à un pur travail sur le langage, détaché d'une forme d'activité intellectuelle spécifique (il suffirait en quelque sorte de savoir lire ou écrire avec rigueur pour que « tout fonctionne »).

AlgoBox propose à ses utilisateurs de rédiger des algorithmes, et non des programmes. Ces algorithmes font cependant explicitement référence à un utilisateur par le biais des instructions de saisie et d'affichage, et donc, par extension, à une machine, même si ces notions relèvent du concept de programme. Le logiciel permet ensuite d'exécuter ces algorithmes, sans traduction visible vers un langage de programmation identifié comme tel, et en tentant de « gommer » les éventuelles difficultés techniques ou syntaxiques inhérentes à la programmation (avec plus ou moins de succès, comme on l'a vu).

Il apparaît donc qu'*AlgoBox* vise une transposition informatique conforme au concept d'algorithme tel qu'il transparaît dans les programmes de 2009-2012. Ce faisant, il tente de concrétiser l'« écrasement conceptuel » entre algorithmes et programmes que nous avons décrit, escamotant le concept de programme informatique à l'intérieur même d'un langage de programmation.

- ***Concept d'algorithme et pensée informatique***

Comme discuté ci-dessus, il n'existe pas dans *AlgoBox* de mécanisme permettant d'identifier les paramètres ni le résultat d'un algorithme. Par conséquent, il est impossible d'y concevoir un algorithme faisant appel à un autre algorithme réalisé précédemment, autrement qu'en le recopiant intégralement. Pour l'enseignant, cela signifie que tout problème doit être résolu « en une fois », et que sa solution ne pourra plus resservir (sauf à être reprise intégralement et adaptée).

Pour des raisons semblables, l'absence dans la transposition informatique opérée par *AlgoBox* des concepts de paramètres et de résultat fait obstacle au développement d'une approche *généralisatrice*, qui consiste à résoudre un problème spécifique en l'élargissant à un problème plus général, par exemple en introduisant de nouveaux paramètres, ou tout simplement à résoudre un problème plus général en modifiant un algorithme déjà produit pour un problème particulier.

Un tel travail est certes envisageable, par exemple en remplaçant l'affectation d'une constante à une variable par une

saisie. Cependant, nous défendons l'idée qu'une formalisation plus rigoureuse du concept de paramètre est indispensable pour aborder de manière satisfaisante ce type de démarche. L'impact possible d'un tel manque de conceptualisation est également illustré par ce que Modeste (2012) appelle « algorithmes-instanciés » et qu'il identifie comme symptomatiques de la période 2009-2012.

Comme on le voit, les caractéristiques d'*AlgoBox* ne permettent pas, ou pas pleinement, de mettre en œuvre certains types de démarches qui, comme nous l'avons déjà mentionné dans la partie « Repères épistémologiques », sont couramment considérés comme constitutifs de la pensée informatique.

- *Conclusion sur la transposition informatique*

Selon tous les critères évoqués dans cette section, liés au statut des variables, à la syntaxe et à la sémantique du langage, à la confusion entre algorithmes et programmes, au concept de problème (embarquant ceux de paramètre et de résultat) et à une certaine définition académique de la pensée informatique, on peut donc affirmer que la transposition informatique à l'œuvre dans *AlgoBox* obéit à un parti pris relativement éloigné du savoir savant.

Étant donné la large adoption de cet outil à partir de 2009, ces choix didactiques et épistémologiques (délibérés ou non) sont donc susceptibles d'avoir eu un impact sur les apprentissages et sur le type de méthodes et de raisonnements accessibles aux enseignants et aux élèves qui l'ont utilisé. Dans la dernière section, nous questionnerons la place et l'influence d'*AlgoBox* vis-à-vis de ce curriculum.

Place et influence d'*AlgoBox* au sein du curriculum

Dans cette section, nous souhaitons mettre en lumière les phénomènes qui lient le déploiement du logiciel *AlgoBox* et celui du curriculum de mathématiques de lycée entre 2009 et 2019 (tant du point de vue du savoir à enseigner que du savoir enseigné).

Un premier pas vers l'algorithmique en mathématiques

En 2009, l'enseignement de l'informatique est essentiellement absent de l'enseignement secondaire public général en France, à l'exception d'une part d'algorithmique dans les mathématiques de la filière littéraire, malgré des efforts de plusieurs organismes et groupements d'intérêts (sociétés savantes ou syndicats professionnels, Académie des sciences, etc.) de le promouvoir.

En lien avec les recommandations de la « commission Kahane » (Kahane, 2002), la réforme du lycée de 2009-

2012 généralise l'introduction de l'algorithmique dans les mathématiques du lycée général. Elle présente la démarche algorithmique comme étant « *depuis les origines, une composante essentielle de l'activité mathématique*[41] », et par conséquent fortement ancrée dans cette discipline. Elle en confie la prise en charge aux enseignants de mathématiques, sans déployer de véritable formation spécifique.

Ceci, ainsi que les conditions et le rythme de la réforme, associés à l'absence de langage imposé et à la consigne de transversalité (interdisant de fait la tenue de séances spécifiques) décourage l'adoption par les enseignants de langages de programmation « complets », dont la mise en place en classe est souvent perçue comme inaccessible. Ceci est renforcé par la perception par certains enseignants du caractère exogène de ces contenus, souvent absents de leur formation initiale.

L'apparition d'*AlgoBox* à la rentrée 2009 vient ainsi combler un besoin : un outil facile à prendre en main, présenté comme parfaitement adapté et focalisé sur le contenu du programme, ne nécessitant pas l'apprentissage d'une syntaxe particulière, permettant de réaliser les activités algorithmiques attendues des programmes, notamment dans une démarche de résolution de problème.

Une transposition informatique fidèle aux instructions officielles

Comme nous l'avons détaillé dans la section précédente, une analyse soigneuse de la transposition informatique du concept d'algorithme à l'œuvre dans *AlgoBox* révèle une grande conformité avec les consignes et préconisations des programmes. Cette fidélité est pleinement revendiquée par l'auteur d'*Algobox*. Notre analyse a aussi révélé un certain nombre de particularités dont il est légitime de questionner les effets.

En refusant en un sens de prendre en charge les aspects spécifiques et potentiellement complexes de l'informatique et de la programmation tels que l'apprentissage d'une syntaxe rigoureuse, les enjeux de représentation des données et en particulier des nombres, et plus généralement de modélisation informatique des objets manipulés, le logiciel semble entretenir l'idée (ou l'illusion) qu'il « suffit de savoir lire » pour savoir programmer.

Paradoxalement, et en contradiction avec sa propre philosophie, en imposant un mode de fonctionnement spécifique (types d'objets restreints, présentation normalisée, vocabulaire fixé), il est dans le même temps susceptible d'aller à l'encontre des avantages offerts par une expression plus libre des algorithmes telle qu'on la rencontre couramment dans nombre de cursus universitaires en informatique et en mathématiques, et même dans le document-ressource[42].

41 MEN, « Programme d'enseignement de mathématiques... », 2019, *op. cit.*

42 MEN, « Ressources pour la classe de seconde – Algorithmique », 2019, *op. cit.*

Enfin, en ne permettant pas (ou difficilement) l'émergence d'un raisonnement par décomposition ou par généralisation, ni le réemploi d'algorithmes précédemment étudié, il fait obstacle à la mise en œuvre de modes de raisonnement pourtant très répandus en informatique comme en mathématiques, en particulier dans le contexte d'une « *démarche de résolution de problème* » fréquemment citée comme objectif d'apprentissage dans les programmes.

Ainsi, *AlgoBox* produit une transposition informatique du concept d'algorithme fidèle aux programmes de 2009-2012. Par nécessité, le logiciel formalise certains éléments implicites des instructions officielles, révélant ainsi un certain nombre de phénomènes de transposition didactique qui semblent n'avoir pas été anticipés par les auteurs des programmes, et dont nous avons montré un certain nombre de limites ou d'aspects problématiques.

Un projet voué à l'échec ?

Les sections précédentes ont montré comment les programmes de mathématiques du lycée de 2009-2012 en algorithmique sont porteurs d'un projet (qui reste implicite, et peut-être non-intentionnel) : celui d'un langage d'algorithme dont la syntaxe et la sémantique seraient transparentes, et dont l'implémentation en machine devrait éviter tout aspect technique. Le logiciel peut alors être vu comme une tentative de réalisation de ce projet.

Selon nous, ce projet est intrinsèquement porteur de contradictions qui condamnent toute tentative de réalisation à se heurter à des difficultés d'ordres épistémologique, technique et didactique.

Sur le plan épistémologique, l'écrasement des notions d'algorithme et de programme que nous avons mis en lumière vient se heurter à la nature fondamentalement différente de ces deux concepts :

- le programme, comme texte formel avec des règles syntaxiques rigoureuses et doté d'une sémantique liée à une machine (réelle ou virtuelle) ;

- l'algorithme, comme expression dans un langage partiellement formalisé d'une méthode de résolution de problème, indépendamment de certaines contraintes techniques.

Il en découle un langage, dans le cas qui nous occupe le langage *AlgoBox*, dont on pourrait dire qu'il vise à rendre les algorithmes-papier (au sens de Modeste, 2012) exécutables. On peut dire qu'il cherche à invisibiliser les aspects techniques et informatiques (notamment en éludant la question du modèle de calcul ou de la machine, ou par ses tergiversations quant à la manipulation des nombres). Nous avons montré un certain nombre de problèmes techniques du logiciel et de son langage qui nous semblent révélateurs des limites de cette tentative d'invisibilisation.

Enfin, on peut voir derrière le discours de simplicité et d'accessibilité d'*AlgoBox*, une intention pédagogique résumée ainsi sur son site : « *un logiciel éducatif basé sur une logique pédagogique (apprentissage de l'algorithmique par* ***structures logiques*** *à travers un langage textuel proche du* ***langage naturel algorithmique****)* ». De ceci découle une tentative d'évitement des deux stratégies d'initiation à l'algorithmique identifiées par Nguyen (2005) : présence d'une machine de référence, ou appui sur un langage représentatif d'une classe de langages de programmation. Sur le plan didactique, cette position ne semble pas tenable.

Ainsi, les programmes de 2009-2012 et le logiciel, dans une illusion de transparence (Chevallard, 1985), cherchent à proposer une copie simplifiée, mais fidèle du concept algorithme (pour reprendre la formule d'Artigue, 1990), et se heurtent alors à des problèmes de transposition didactique et informatique.

Les aménagements des programmes survenus en 2017, et plus encore la réforme du lycée de 2019-2020, distinguent plus nettement les notions d'algorithme et de programme et désignent Python comme langage de référence pour la programmation. Ces changements ont donné un coup d'arrêt à l'utilisation d'*AlgoBox*. En témoignent le document-ressource[43], dont les exemples sont exclusivement rédigés en Python, l'absence de mise à jour significative d'*AlgoBox* depuis 2017, son retrait de la liste des logiciels fournis au CAPES de mathématiques dès la session 2018, et la disparition des mentions de ce logiciel dans les manuels scolaires et canaux de diffusion habituels (dont le site *éduscol*).

Ce tournant s'est vu renforcé par l'apparition dès 2019 d'une matière de tronc commun en classe de seconde (SNT) et d'une spécialité du cycle terminal (NSI) pleinement consacrées à l'informatique et mettant un fort accent sur l'apprentissage de la programmation, de la création d'un nouveau corps de professeurs pour enseigner ces matières (CAPES NSI).

Une influence forte sur le savoir enseigné

Selon l'auteur d'*AlgoBox*, « *la quasi-totalité des manuels scolaires de mathématiques (de la seconde à la terminale)* » citent ou utilisent ce logiciel. Nous avons pu observer qu'une grande majorité des ressources et la quasi-totalité des manuels de la période 2009-2017 présentent effectivement leurs algorithmes avec son langage (éventuellement parmi d'autres langages).

L'une des raisons de cette diffusion large et rapide dans les manuels, les ressources et donc les classes est certainement la forte proximité d'*AlgoBox* avec le contenu des programmes. Les contraintes qui se sont exercées sur la mise en œuvre

43 MEN, « Ressources pour le lycée – Mathématiques – Algorithmique et programmation », 2017, *op. cit.*

de ces programmes, mentionnées plus haut (faible formation des enseignants, nature et urgence de la mise en place de la réforme…), ont aussi probablement joué un rôle.

En effet, la transposition didactique du concept algorithme proposée par les programmes de 2009-2012 est partagée par les manuels (pour des exemples d'analyses voir Modeste, 2012). On retrouve aussi dans les ressources et manuels, souvent en cohabitation avec *AlgoBox*, les langages des calculatrices Casio et TI, et les logiciels *Xcas*, *Scratch* et *Scilab*. La proximité entre le type de langage utilisé dans les ressources institutionnelles pour décrire les algorithmes et le langage apparaît alors encore plus nettement. Il est donc très probable que le logiciel ait été utilisé très massivement comment nous le pensons.

Quelques indices laissent penser qu'*AlgoBox* a contribué à modeler et diffuser un langage de description des algorithmes, inspiré de celui du document-ressource de 2009. Le premier est la présence récurrente dans les sujets du baccalauréat de 2012 à 2019 d'algorithmes basés sur le langage d'algorithmes conforme aux attentes des documents officiels, mais comprenant des corps de blocs délimités par les mots-clés « FinSi », « FinPour » et « FinTantque », caractéristique du logiciel. Ces mots-clés sont conservés dans le document « Évolution de l'écriture des algorithmes au Baccalauréat » déjà mentionné (*cf.* note 18).

Le deuxième indice est la présence, dans certains manuels ultérieurs à la réforme de 2019, de présentations des algorithmes qui conservent certains éléments de forme et des mots-clés des langages d'*AlgoBox* et du document-ressource de 2009, alors que le langage Python est dorénavant imposé. Il est probable que chez les enseignants comme chez les rédacteurs de manuels, la période 2009-2019 a contribué à construire des conceptions de la notion d'algorithme et de sa présentation, qui continueront d'influencer le savoir enseigné.

Conclusions

Forquin (2008) identifie un certain nombre de questionnements sociologiques concernant l'objet curriculum : en tant qu'opérateur de sélection et de transmission culturelles, en tant que véhicule possible d'une « *culture d'école* » à part entière, et enfin en tant qu'objet à dimension politique. Sur ce dernier point, il écrit :

> La question importante est ici celle de savoir qui dispose du pouvoir de contrôle sur l'élaboration (et la promulgation) des programmes et des plans d'étude, et comment ces phénomènes de contrôle se traduisent aux différentes étapes de ce qu'on pourrait appeler « *la chaîne de production curriculaire* » (Forquin, 2008, p. 9).

Il questionne ainsi ce que la théorie anthropologique du didactique appelle la noosphère, et s'intéresse plus préci-

sément aux phénomènes qui entrent en jeu dans la chaîne de production curriculaire. La transposition didactique s'intéresse plus particulièrement aux produits de ces phénomènes.

En étudiant la transposition didactique du concept algorithme dans les instructions officielles et les ressources d'enseignement de la période 2009-2019, ainsi que dans *AlgoBox*, nous avons mis en lumière de tels phénomènes curriculaires, qui illustrent comment le contrôle dont parle Forquin peut relever d'influences complexes, pouvant dépasser les intentions de ceux qui élaborent les curriculums.

En 2009 paraissent de nouveaux programmes introduisant l'algorithmique dans l'enseignement des mathématiques. Bien que de nombreux logiciels pour l'initiation à l'algorithmique ou la programmation soient déjà disponibles, le logiciel s'impose en proposant un langage parfaitement adapté aux attentes institutionnelles et un logiciel offrant une prise en main facile, dans un contexte où les enseignants manquent de repères. Le logiciel incarne la conception de l'algorithmique portée par les programmes (dont un amalgame algorithme-programme), et son langage a colonisé un certain nombre de ressources pédagogiques, de manuels scolaires et jusqu'aux sujets du baccalauréat.

Une inflexion du discours se produit dans les années 2017-2020, certainement liée aux limites du projet de 2009, au développement de l'informatique comme discipline scolaire, et à diverses prises de position et analyses dans la noosphère. Les programmes de 2019-2020 nous semblent marquer un retour à une situation plus proche d'autres curriculum*s* internationaux, comme en témoigne l'adoption du langage Python.

Il nous a semblé important d'analyser et documenter la période 2009-2019, qui peut être vue comme une sorte de bulle, à la fois une exception française dans le choix d'introduire une part d'algorithmique dans les mathématiques, et dans la transposition didactique qui en découle.

Cette analyse apporte aussi des éléments pour réfléchir à des questions liées à l'enseignement de l'informatique et à son développement comme discipline scolaire : l'impact des choix de logiciels et langages informatiques sur les apprentissages et sur la chaîne de production curriculaire ; la difficulté de conception de logiciels pédagogiques *ad hoc* adaptés à des choix curriculaires précis ; les effets de transposition informatique, y compris pour des concepts issus de l'informatique. En particulier, les choix curriculaires actuels désignent Python comme langage de programmation de référence pour l'enseignement de l'algorithmique et de la programmation au lycée. Ce choix d'un langage de programmation issu du milieu professionnel ne fait pas pour autant disparaître les enjeux de transpositions informatique et didactique.

Bibliographie

Académie des Sciences (2013). *L'enseignement de l'informatique en France. Il est urgent de ne plus attendre.* Rapport n° 513. [URL : http://www.academie-sciences.fr/activite/rapport/rads_0513.pdf].

Artaud M. (1998). « Introduction à l'approche écologique du didactique – L'écologie des organisations mathématiques et didactiques ». *In* ARDM (dir.). *Actes de la IX^e École d'été de didactique des mathématiques.* Caen : IUFM de l'académie de Caen, pp. 101-139.

Artigue M. (1990). « Épistémologie et Didactique ». *Recherches en Didactique des Mathématiques,* 10 (2.3), pp. 241-285.

Balacheff N. (1993). « La transposition informatique, un nouveau problème pour la didactique ». *In* M. Artigue, N. Balacheff, R. Gras, C. Laborde & P. Tavignot (dir.). *Colloque « Vingt ans de didactique des mathématiques en France », 15-17 juin 1993.* Grenoble : La Pensée Sauvage, pp. 364-370.

Baron G.-L. & Bruillard É. (2011). « L'informatique et son enseignement dans l'enseignement scolaire général français : Enjeux de pouvoirs et de savoir ». *In* J. Lebaume, A. Hasni & I. Harlé (dir.). *Recherches et expertises pour l'enseignement scientifique.* Bruxelles : De Boeck, pp. 79-90.

Bosch M. & Gascón J. (2014). « Introduction to the Anthropological Theory of the Didactic (ATD) ». *In* A. Bikner-Ahsbahs & S. Prediger (eds.). *Networking of Theories as a Research Practice in Mathematics Education.* Springer International Publishing, pp. 67-83.

Chen C., Haduong P., Brennan K., Sonnert G. & Sadler P. (2019). « The effects of first programming language on college students' computing attitude and achievement: A comparison of graphical and textual languages ». *Computer Science Education*, 29(1), pp. 23-48.

Chevallard Y. (1985). *La transposition didactique – du savoir savant au savoir enseigné.* Grenoble : La Pensée Sauvage.

Chevallard Y. & Bosch M. (2014). « Didactic Transposition in Mathematics Education ». *In* S. Lerman (ed.). *Encyclopedia of Mathematics Education.* Springer Netherlands, pp. 170-174.

Durand-Guerrier V., Meyer A. & Modeste S. (2019). « Didactical Issues at the Interface of Mathematics and Computer Science ». *In* G. Hanna, D. A. Reid & M. de Villiers (eds.). *Proof Technology in Mathematics Research and Teaching.* Springer International Publishing, pp. 115-138.

Forquin J.-C. (2008). *Sociologie du Curriculum.* Presses Universitaires de Rennes (PUR).

Froidevaux C., Gaudel M.-C. & Soria M. (1990). *Types de données et algorithmes.* Paris : McGraw-Hill.

Gal-Ezer J. & Harel D. (1998). « What (Else) Should CS Educators Know? ». *Communications of the ACM*, 41(9), pp. 77-84.

Kahane J.-P. (2002). *Enseignement des sciences mathématiques : Commission de réflexion sur l'enseignement des mathématiques : Rapport au ministre de l'Éducation nationale.* Paris : Odile Jacob/CNDP.

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

Modeste S. (2020). « Prendre en compte l'épistémologie de l'algorithme. Quels apports d'un modèle de conceptions ? Quelle transposition didactique ? ». *Recherches en Didactique des Mathématiques,* 40(3), pp. 363-404.

Nguyen C. T. (2005). « Étude didactique

de l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement mathématique secondaire à l'aide de la calculatrice ». Thèse en didactique des mathématiques, Université de Grenoble.

Samurçay R. (1985). « Signification et fonctionnement du concept de variable informatique chez des élèves débutants ». *Educational Studies in Mathematics*, 16(2), pp. 143-161.

Selby C. & Woollard J. (2013). *Computational thinking: The developing definition* [Rapport de recherche]. University of Southampton, UK [URL : https://eprints.soton.ac.uk/356481/].

Wing J. M. (2006). « Computational thinking ». *Communications of the ACM,* 49(3), pp. 33-35.