EMP Educação Matemática Pesquisa
ISSN 1983-3156
A2
http://dx.doi.org/10.23925/1983-3156.2019v21i4p185-200

Du logos des organisations algorithmiques dans l'enseignement secondaire

JEAN-MARCEL STROCK[1]
MICHELE ARTAUD[2]

Abstract. In France, Algorithmics is registered into domains to be taught in high school officially since 2009 and it has appeared in the college program which will come into effect in 2016. The examination of praxeologies to teach shows that these suffer from a technologico-theoretical deficit, which is not without consequences on their ecology within the didactic systems. Our communication will highlight this deficit on the type of tasks, "To run an algorithm", which will be considered as a generic example. It will give ingredients likely to complete logos of algorithmic praxeology to teach around this type of tasks, on the basis of a study carried out in Master's degree (Strock, 2013).

Résumé. En France, l'algorithmique est inscrite dans les domaines à enseigner au lycée officiellement depuis 2009 et elle figure au programme du collège qui entrera en vigueur en 2016. L'examen des praxéologies à enseigner montre que celles-ci souffrent d'un déficit technologico-théorique qui n'est pas sans conséquences sur leur écologie au sein des systèmes didactiques. Notre communication mettra en évidence ce déficit sur un type de tâches, Faire fonctionner un algorithme, qui sera considéré comme exemple générique. Elle donnera des ingrédients de nature à compléter le logos de la praxéologie algorithmique à enseigner autour de ce type de tâches en s'appuyant sur une étude menée en master (Strock, 2013).

1. Praxéologie algorithmique à enseigner

1.1. L'algorithmique dans les programmes du lycée

L'algorithmique a été introduite comme un secteur des programmes du lycée avec le programme de seconde mis en place à la rentrée 2009. On voit d'abord que l'étude de ce secteur est clairement affirmée comme devant répondre aux besoins des domaines mathématiques figurant dans les programmes :

> L'algorithmique a une place naturelle dans tous les champs des mathématiques et les problèmes posés doivent être en relation avec les autres parties du programme (fonctions, géométrie, statistique et probabilités, logique) mais aussi avec les autres disciplines ou la vie courante. À l'occasion de l'écriture d'algorithmes et de petits programmes, il convient de donner aux élèves de bonnes habitudes de rigueur et de les entraîner aux pratiques systématiques de vérification et de contrôle. (MEN[3], 2009a, p. 9)

Cela peut institutionnellement légitimer la mise en œuvre d'une pédagogie de l'enquête (Chevallard, 2008) relativement à l'algorithmique. Le fait que ce secteur n'a pas véritablement un programme par année renforce cette légitimation : les textes officiels fixent des éléments qui doivent être acquis à la fin de la classe terminale et que nous reproduisons ci-après :

[1] Laboratoire ADEF (EA 4671), Aix-Marseille Univ., France– jean-marcel.STROCK@univ-amu.fr
[2] Laboratoire ADEF (EA 4671), Aix-Marseille Univ., France– michele.artaud@univ-amu.fr
[3] L'acronyme MEN signifiera Ministère de l'Éducation nationale.

Instructions élémentaires (affectation, calcul, entrée, sortie). Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables : • d'écrire une formule permettant un calcul ; • d'écrire un programme calculant et donnant la valeur d'une fonction ; ainsi que les instructions d'entrées et sorties nécessaires au traitement.
Boucle et itérateur, instruction conditionnelle Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables : • de programmer un calcul itératif, le nombre d'itérations étant donné ; • de programmer une instruction conditionnelle, un calcul itératif, avec une fin de boucle conditionnelle. (MEN, 2009a, p. 10)

Cette dernière remarque doit cependant être pondérée par le fait que certains ingrédients apparaissent à l'intérieur des domaines. Pour la classe de seconde par exemple, on trouve mentionné à l'égard du domaine « Fonctions » :

Domaine	Thème	Commentaires
Fonctions	Étude qualitative de fonctions	◇ Même si les logiciels traceurs de courbes permettent d'obtenir rapidement la représentation graphique d'une fonction définie par une formule algébrique, il est intéressant, notamment pour les fonctions définies par morceaux, de faire écrire aux élèves un algorithme de tracé de courbe. (MEN, 2009a, p. 3)
	Équations	◇ Encadrer une racine d'une équation grâce à un algorithme de dichotomie. (MEN, 2009a, p. 4)

Tableau 1. Activités algorithmiques au programme de seconde pour le domaine « Fonctions ».

L'algorithmique fait l'objet d'un document « Ressources pour la classe de seconde – Algorithmique » rédigé en 2009 et disponible sur le site Éduscol (MEN, 2009b). Ce document précise en quelques 33 pages un certain nombre d'éléments des attentes institutionnelles à l'égard de l'algorithmique que nous avons vu exposés brièvement dans le programme. Nous présenterons ici les ingrédients qui nous semblent les plus significatifs à l'égard de la délimitation d'une praxéologie algorithmique à enseigner, de ses raisons d'être et de sa fonctionnalisation.

On trouve d'abord un certain nombre de types de tâches : trois types de tâches apparaissent en lien avec l'évaluation : « **analyser** le fonctionnement ou le but d'un algorithme existant ; **modifier** un algorithme existant pour obtenir un résultat précis ; **créer** un algorithme en réponse à un problème donné » (MEN, 2009b, p. 5), tandis que les sept suivantes sont données comme susceptibles d'être identifiées et travaillées :

– comprendre et analyser un algorithme préexistant ;

– modifier un algorithme pour obtenir un résultat particulier ;

– analyser la situation : identifier les données d'entrée, de sortie, le traitement... ;

– mettre au point une solution algorithmique : comment écrire un algorithme en « langage courant » en respectant un code, identifier les boucles, les tests, des opérations d'écriture, d'affichage... ;

– valider la solution algorithmique par des traces d'exécution et des jeux d'essais simples ;

– adapter l'algorithme aux contraintes du langage de programmation : identifier si nécessaire la nature des variables... ;

– valider un programme simple. (MEN, 2009b, p. 3)

On voit donc se dégager un réseau de types de tâches que nous modéliserons comme suit [4] :

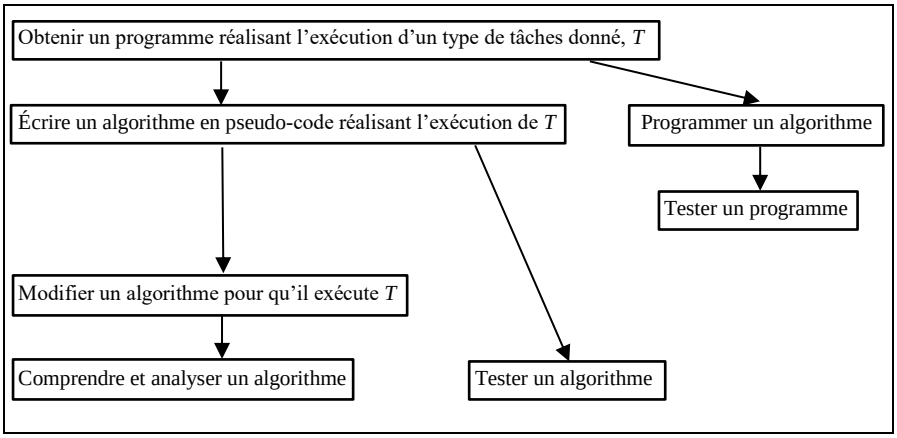

Figure 1. Réseau de types de tâches.

Obtenir un programme réalisant l'exécution d'un type de tâches donné *T* peut demander l'écriture d'un algorithme réalisant *T*, le test de l'algorithme obtenu, la programmation de cet algorithme et le test du programme obtenu. Pour écrire un algorithme réalisant *T*, on peut se procurer un algorithme réalisant un type de tâches *T'* (qui peut être égal à *T*) et éventuellement le modifier pour qu'il exécute *T*, ce qui suppose de comprendre et d'analyser un algorithme, puis le tester. On notera que l'établissement d'un programme apparaît comme une raison d'être de l'établissement d'un algorithme.

On trouve ensuite, sous le titre « Une initiation à l'algorithmique » (*Ibid.*, p. 6) des ingrédients de l'environnement technologico-théorique de la praxéologie à mettre en place. Le premier est la définition d'un algorithme. Les auteurs reprennent une définition de l'*Encyclopaedia Universalis*, « 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 » (Hebenstreit, 2013), définition précédée du constat que, dans « un premier temps rédiger un algorithme consiste à décrire les différentes étapes de calcul pour résoudre un problème algébrique, numérique ou décisionnel » (MEN, 2009b, p. 6). Ils précisent ensuite ce qu'ils entendent par « règles à appliquer » : « un traitement fait sur des données imposé par une suite "d'instructions" visant à transformer ces données pour arriver au résultat visé » (*Ibid.*). Avant de préciser les instructions, qui sont considérées comme les « briques de base des algorithmes dont l'assemblage dans un ordre précis conduit au résultat attendu » (*Ibid.*), les auteurs du document proposent de structurer un algorithme en trois parties : « la préparation du traitement » qui comporte « le repérage des données » et également

4. La flèche entre deux types de tâches exprime que le second est susceptible d'être un sous-type de tâches du premier.

« l'entrée des données » ; le traitement qui comportent « les instructions à donner pour une exécution automatique » (*Ibid*.), en précisant qu'on ne traitera que d'algorithmes séquentiels ; et enfin « la sortie des résultats ». On notera que la première partie de cette structure se verra, sans autre commentaire, présentée en deux morceaux, variables et entrées, dans les algorithmes proposés dans la deuxième partie.

L'environnement technologico-théorique est complété par des instructions, qui sont celles qui sont citées par le programme, et pour lesquelles un choix de pseudo-code est effectué. Les voici présentées de manière synthétique :

Affectation de données dans des variables : *identificateur* **prend la valeur** *valeur* ;
Lecture (ou entrée) des données : **Saisir** *identificateur* ;
Écriture ou sortie des données : **Afficher** *identificateur* ; **Afficher** « message » ;
La « structure alternative » :

Si *condition* **alors**
    Traitement 1
Sinon
    Traitement 2

Les structures répétitives :

Pour I de 1 jusqu'à N faire
    Traitement 1

Tant que *condition* **faire**
    Traitement 1

Répète
    Traitement 1
    jusqu'à *condition*

(*Ibid*., pp. 7-9)

On notera que la présentation des instructions s'accompagne d'un certain nombre de définitions comme par exemple :

> […] la « condition », c'est une expression logique prenant l'une des deux valeurs VRAI ou FAUX, qui peut être simple ou complexe. […] Une « condition complexe » est une combinaison logique de conditions simples. (*Ibid*., p. 8)

Le document de la collection *Ressources pour la classe* recèle en outre des éléments pour la fonctionnalisation du secteur algorithmique. On y trouve par exemple :

> […] L'enseignement de l'algorithmique ne relève pas, à ce niveau, de cours spécifiques ; au contraire, l'introduction de chaque nouvel élément (variable, boucle, itération, etc.) devrait apparaître lors de la résolution de problèmes pour lesquels les démarches habituelles sont malcommodes ou peu performantes : par exemple dans le cas de répétition d'une tâche, ou dans le cas d'un traitement trop long pour être envisagé « à la main ».
>
> […] Il serait souhaitable d'intégrer l'écriture d'algorithmes dans tous les domaines du programme.
>
> […] Enfin, il faut avant tout éviter de confronter les élèves à des difficultés trop importantes ; en effet, la classe de seconde est une classe de détermination et il ne s'agit pas d'y former des programmeurs mais de faire en sorte que les mathématiques et l'algorithmique soient au service d'activités de résolution de problèmes pour les sciences. […] (*Ibid*., p. 4)

Ce même document présente également des exemples de dispositifs de classe permettant la mise en œuvre de l'algorithmique, ainsi qu'une liste d'algorithmes assez bien fournie et variée pour les différents domaines tels que la géométrie, les fonctions, les probabilités… (*Ibid.*, pp. 10-30) Mais l'on peut regretter cependant la volonté de ne pas expliciter de manière plus détaillée comment fonctionnaliser l'algorithmique au sein des différents domaines. On peut y lire :

> […] Les exemples suivants ne sont décrits ni dans le détail, ni dans leur contenu ni dans leur mise en œuvre dans la classe. Ils peuvent servir de base à la mise en place d'activités de groupe

permettant des premières réflexions autour de la notion d'algorithme dans le cadre de situations axées sur la communication. En ce sens, après un temps d'investigation les élèves seront chargés de produire un « texte » (écrit ou oral) compréhensible et directement utilisable par leurs camarades par exemple. […] (*Ibid*., p. 10)

1.2. Une praxéologie autour du type de tâches *« Faire fonctionner un algorithme »*

Les sujets du baccalauréat ont enregistré cette officialisation de l'étude de l'algorithmique au lycée et on trouve donc depuis quelques années un travail algorithmique dans chaque sujet proposé. Nous en examinerons ici un exemple.

Un travail algorithmique proposé au baccalauréat

On considère l'algorithme suivant, extrait d'un exercice de baccalauréat sorti en Métropole, en 2013, dans la série scientifique (MEN, 2013, p. 4) :

Variables :	a, b et m sont des nombres réels.
Initialisation :	Affecter à a la valeur 0.
	Affecter à b la valeur 1.
Traitement :	Tant que $b - a > 0{,}1$
	Affecter à m la valeur $\frac{1}{2}(a+b)$
	Si $f(m) < 1$ alors Affecter à a la valeur m.
	Sinon Affecter à b la valeur m.
	Fin de Tant que.
Sortie :	Afficher a.
	Afficher b.

L'un des objectifs de l'exercice est de déterminer un encadrement des deux solutions de l'équation $f(x) = 1$, la solution α sur]0 ; 1], puis la solution β sur]1 ; +∞[, où f est la fonction définie sur]0 ; +∞[par $f(x) = \frac{2+2\ln(x)}{x}$. On met d'abord en évidence dans l'exercice que la fonction f est continue et strictement croissante sur]0 ; 1] et 1 est une valeur strictement comprise entre $\lim\limits_{x\to 0^+} f(x)$ et $f(1)$; donc par application d'un corollaire au théorème des valeurs intermédiaires, l'équation $f(x) = 1$ admet une unique solution α sur]0 ; 1]. Par ailleurs, on admet que l'équation $f(x) = 1$ possède une unique solution β sur]1 ; +∞[et on demande de trouver un entier naturel n vérifiant $n < \beta < n + 1$. Aucune indication de technique n'est donnée aux candidats. Ceci laisse supposer qu'ils emploieront, dans une grande majorité des cas, la méthode par balayage. À l'aide de la calculatrice, on observe par balayage que $f(5) > 1$ et $f(6) < 1$. De plus, la continuité de f sur [5 ; 6] permet de conclure à l'existence d'au moins une solution à l'équation $f(x) = 1$ et puisque l'on a admis que l'équation $f(x) = 1$ possède une unique solution β sur]1 ; +∞[, cette solution est nécessairement comprise entre 5 et 6. L'entier naturel cherché est donc 5. C'est en ce point qu'intervient l'algorithmique, à travers la suite de questions suivantes (MEN, 2013, p. 4) :

a. Faire tourner cet algorithme en complétant le tableau ci-dessous que l'on recopiera sur la copie.

	Étape 1	Étape 2	Étape 3	Étape 4	Étape 5
a	0				
b	1				
$b - a$					
m					

b. Que représentent les valeurs affichées par cet algorithme ?

c. Modifier l'algorithme ci-dessus pour qu'il affiche les deux bornes d'un encadrement de β d'amplitude 10^{-1}.

On le voit, l'algorithme proposé apparaît ici insuffisamment fonctionnalisé par l'étude mathématique effectuée dans l'exercice. Cet algorithme renvoie en effet les deux bornes obtenues pour encadrer le nombre α par dichotomie avec une amplitude au plus égale à 0,1, ce qui entre en conformité avec le programme de mathématiques de terminale scientifique qui préconise ce type d'activités :

> Des activités algorithmiques sont réalisées dans le cadre de la recherche de solutions de l'équation $f(x) = k$. (MEN, 2011, p. 5)

Mais il aurait été meilleur du point de vue de la dialectique des médias et des milieux de faire d'abord étudier l'algorithme, vérifier le résultat pour une amplitude de 0,1 et de lui faire produire un encadrement d'amplitude 10^{-3} ou 10^{-4}.

Trois types de tâches algorithmiques apparaissent dans l'énoncé :

T_F : « Faire fonctionner un algorithme donné en pseudo-code » ;

T_I : « Interpréter les valeurs affichées à la sortie d'un algorithme donné en pseudo-code » ;

T_M : « Modifier un algorithme de recherche d'une valeur approchée à ε près de la solution sur un intervalle I d'une équation du type $f(x) = k$ de façon à ce que la valeur approchée affichée soit celle de la solution de $f(x) = k$ sur un intervalle J disjoint de I non borné ».

Nous reproduisons ci-dessous le corrigé des questions algorithmiques proposé par l'association des professeurs de mathématiques de l'enseignement public (APMEP, 2013, p. 3) :

4. a. On obtient :

	Étape 1	Étape 2	Étape 3	Étape 4	Étape 5
a	0	0	0,25	0,375	0,4375
b	1	0,5	0,5	0,5	0,5
$b - a$	1	0,5	0,25	0,125	0,0625
m	0,5	0,25	0,375	0,4375	
$f(m)$	≈ 1,23	≈ 3,09	≈ 0,10	≈ 0,79	≈ 1,03

Le tableau a été complété par la ligne « $f(m) \approx$ » pour montrer les affectations à a ou à b.

Le tableau précédent sera probablement considéré comme correct, mais si on interprète la question très rigoureusement, d'un point de vue algorithmique, on doit supposer que l'étape 1 est l'initialisation, et les étapes de 2 à 5 correspondant aux itérations de 1 à 4. Dans ce cas, pour l'étape 1 n'a pas (*sic*) de valeur de m, et la valeur $b - a$ va servir à savoir si l'itération suivante va être utile ou non. Dans ce cas, on va écrire dans la colonne les valeurs en mémoire à la fin de l'itération de la boucle « Tant que », ce qui donne le tableau suivant :

	Étape 1	Étape 2	Étape 3	Étape 4	Étape 5
a	0	0	0,25	0,375	0,4375
b	1	0,5	0,5	0,5	0,5
$b - a$	1	0,5	0,25	0,125	0,0625
m		0,5	0,25	0,375	0,4375

b. Cet algorithme renvoie les deux bornes obtenues pour encadrer le nombre a par dichotomie, avec une amplitude au plus égale à 0,1.

c. Pour que l'algorithme donne un encadrement de β avec la même précision. Il faut modifier l'initialisation, en mettant :

Affecter à a la valeur 5.

Affecter à b la valeur 6.

Puis, dans le traitement, modifier le test « Si » pour qu'il soit : « Si $f(m) > 1$ », afin de prendre en compte la décroissance de f sur l'intervalle [5 ; 6].

(Une autre possibilité serait d'affecter 6 à a et 5 à b, et de modifier le « Tant que » pour avoir « Tant que $a - b > 0{,}1$ » et alors a serait la borne haute de l'encadrement, et b la borne basse).

La résolution de la question 4. a. par l'APMEP fait apparaître une difficulté située au niveau de la technique à mettre en œuvre pour accomplir la tâche t_F appartenant à T_F si bien que le corrigé propose deux réponses différentes pour cette même question. La raison principale est à chercher au niveau du tableau proposé par l'énoncé qui ne suit pas de près la structure de l'algorithme : on notera par exemple que $b - a$ figure dans ce tableau alors que $f(m)$ n'y figure pas bien que ces deux valeurs permettent de décider si des conditions sont vérifiées ou non.

L'étape 1 est identifiée dans le corrigé comme l'initialisation et à ce titre, les deux seules variables initialisées dans la partie initialisation sont a et b prenant respectivement pour valeurs initiales 0 et 1. Certes, la question 4. a. est ambiguë mais la réponse présentée comme rigoureuse l'est aussi. On peut souligner que $b - a$ n'est pas une variable et que l'algorithme ne comporte que trois variables a, b et m.

Ce traitement met ainsi en exergue un défaut au niveau des ingrédients technologiques de la praxéologie organisée autour du type de tâches T_F, et notamment une définition sans doute ambiguë de la notion d'étape.

Il est à noter que le corrigé de l'APMEP fournit la réponse à la question 4. b. sans justification : on aurait pu attendre par exemple une justification du fait qu'on a bien affaire à la méthode de dichotomie. On peut regretter l'absence de traces d'une technique qui permettrait de la produire ou encore d'ingrédients technologico-théoriques, relevant tous deux de l'algorithmique.

Il en va un peu autrement pour la troisième question : le changement de condition dans le test est là justifié par le changement de monotonie de la fonction, quand on modifie l'intervalle d'étude. Le corrigé de l'APMEP fournit cette fois-ci la réponse à la question 4. c. ainsi que des ingrédients algorithmiques d'une technique qui permettrait de la produire. Cependant, on ne voit aucune trace d'ingrédients algorithmiques technologico-théoriques.

Mise en évidence d'éléments d'une praxéologie relative à T_F

Considérons maintenant le cours de mathématiques pour la classe de seconde du centre national d'enseignement à distance (CNED, 2013). L'algorithmique est introduite dans la première séquence, après qu'une étude dans le domaine des fonctions a été menée. La section qui lui est consacrée débute par une partie « A Historique » dans laquelle la notion d'algorithme est définie, et se poursuit par une partie « B Vocabulaire – Tableau de fonctionnement » qui comprend un certain nombre de définitions, la dernière étant celle du « tableau de

fonctionnement d'un algorithme » qui « décrit, à chaque étape de l'algorithme, le contenu des variables (une colonne par variable) » (CNED, 2013, p. 39).

La présentation de ces définitions est suivie d'exemples ; nous reproduisons ci-après celui correspondant au « tableau de fonctionnement d'un algorithme » :

> Écrivons l'algorithme de multiplication de deux entiers :
>
> | |
> |---|
> | ENTRER *a* et *b*
DANS *c*, METTRE *a* × *b*
AFFICHER *c* |
>
> Faisons fonctionner cet algorithme sur un exemple :
>
> prenons *a* = 7 et *b* = 5.
>
> À chacune des trois étapes de l'algorithme, précisons le contenu des variables *a*, *b* et *c*.
>
> Rassemblons le tout dans un tableau de fonctionnement :

	a	*b*	*c*
Entrée	7	5	
Traitement	7	5	35
Sortie			35

(CNED, 2013, pp. 40-41)

Cet exemple est suivi de deux autres qui permettent de voir émerger une technique relative au type de tâches T_F que l'on peut mettre en forme de la façon suivante :

τ_F : On repère la liste des noms des variables d'entrée (*a*, *b*, *c* …) de l'algorithme A considéré, on les compte et on note *n* leur nombre. On dresse alors un tableau contenant *n* + 1 colonnes, en ne faisant figurer la ligne sortie qu'à la fin de l'exécution du traitement[5] (*cf.* tableau 1 ci-après) :
Sur la ligne titrée Variable, on inscrit dans chaque cellule le nom d'une variable de la liste, jusqu'à épuisement de la liste (on peut éventuellement les ranger par ordre alphabétique). Exécuter l'algorithme et au fur et à mesure de son exécution, ajouter *d*…
Dans la plage Traitement, tracer une ligne par instruction d'affectation et stocker dans la cellule correspondante la nouvelle valeur affectée à la variable considérée, en respectant l'ordre d'apparition des instructions d'affectation de la partie Traitement.
Il ne reste plus qu'à reporter les valeurs finales des variables de sortie dans la ligne titrée Sortie.

Variable	*a*	*b*	*c*	…
Entrée				
Traitement				
Sortie				

Tableau 1. Faire fonctionner un algorithme.

Nous la mettrons en œuvre ci-après pour accomplir la tâche : Faire fonctionner l'algorithme de dichotomie pour approcher la solution inférieure à 1 de l'équation $f(x) = 0$ figurant dans l'exercice du baccalauréat analysé précédemment.

[5] La technique τ_F ne précise pas le nombre de lignes nécessaires dans la plage Traitement parce que le tableau est constitué au fur et à mesure que s'exécute le traitement.

Les variables d'entrée de l'algorithme sont *a*, *b* et *m* ; elles sont au nombre de trois. On dresse ainsi un tableau comprenant quatre colonnes (*cf.* tableau 2 ci-après). Dans la partie Entrée, on reporte les valeurs initiales de *a*, *b*, soit respectivement 0, 1. Dans la partie Traitement, on calcule l'expression $b - a$ pour les valeurs contenues dans les variables *a* et *b*. On regarde si la condition « $b - a$ est strictement supérieur à 0,1 » est vérifiée on non. Elle l'est en effet étant donné que $b - a$ vaut tout d'abord 1. On entre pour la première fois dans la boucle « Tant que », on calcule la demi-somme de *a* et *b* que l'on place dans la variable *m*. On calcule l'image de *m* par la fonction *f* et comme elle est strictement supérieure à 1 – sa valeur approchée à 10^{-2} près par excès valant 1,23 – on affecte à *b* la valeur contenue dans *m* à savoir 0,5. On calcule l'expression $b - a$ pour $a = 0$ et $b = 0{,}5$, ce qui donne 0,5 et ce résultat étant strictement supérieur à 0,1 on entre pour la deuxième fois dans la boucle, on affecte à *m* la demi-somme de *a* et *b*, c'est-à-dire 0,25. On calcule l'image de *m* par *f* dont une valeur approchée à 10^{-2} près par excès est −3,09 qui est cette fois-ci strictement inférieure à 1. Par conséquent, on affecte à *a* la valeur contenue dans *m*. On calcule l'expression $b - a$ pour $a = 0{,}25$ et $b = 0{,}5$, ce qui donne 0,25. La condition d'entrée dans la boucle est encore vérifiée, on peut y entrer une troisième fois, on calcule la demi-somme de *a* et *b*, ce qui donne 0,375 que l'on affecte à *m*. On calcule l'image de *m* par *f* dont une valeur approchée à 10^{-2} près par excès est 0,11 est bien strictement inférieure à 1. Il en résulte que l'on affecte à *a* la valeur contenue dans *m*, à savoir 0,375. On calcule $b - a$ pour $a = 0{,}375$ et $b = 0{,}5$, ce qui donne 0,125 qui reste strictement supérieur à 0,1. Cela permet de rentrer une quatrième fois dans la boucle, d'affecter à *m* la demi-somme de *a* et *b* soit 0,437 5 dont on calcule l'image par *f* et qui donne 0,80 à 10^{-2} près par excès. Comme $f(m)$ est strictement inférieur à 1, on affecte à *a* la valeur contenue dans *m*. On calcule $b - a$ pour $a = 0{,}437\ 5$ et $b = 0{,}5$, ce qui donne 0,0625. Cette fois-ci, $b - a$ est strictement inférieur à 0,1 ce qui implique que l'on sorte de la boucle. Dans la partie Sortie, on reporte les contenus respectifs de *a* et *b* à afficher.

Variable	*a*	*b*	*m*
Entrée	0	1	
Traitement			(0 + 1)/2 = 0,5
		0,5	
			(0 + 0,5)/2 = 0,25
	0,25		
			(0,25 + 0,5)/2 = 0,375
	0,375		
			(0,375 + 0,5)/2 = 0,437 5
	0,437 5		
Sortie	0,437 5	0,5	

Tableau 2. Faire fonctionner l'algorithme de dichotomie.

On peut souligner ici, comme nous l'avons déjà noté, que le fait que les instructions autres que l'affectation ne soient pas présentes nuit à la lisibilité du tableau comme trace de l'exécution de l'algorithme.

Il est vrai que les exemples permettant de la faire émerger dans le cours du CNED ne comportent que des instructions d'affectation. Si l'on se reporte à la séquence 6, toujours relative au domaine des fonctions, un TP travaille précisément sur la méthode de dichotomie (CNED, 2013, pp. 30-31). On y trouve un tableau de fonctionnement que nous reproduisons ci-après tout comme l'algorithme qu'il met en fonction.

Initialisation : Dans a mettre 2
Dans b mettre 5
Traitement : Tant que $b - a > 0{,}1$ faire
  Si $\left(\frac{a+b}{2}\right)^2 > 10$ alors
    Dans b mettre $\frac{a+b}{2}$
  Sinon
    Dans a mettre $\frac{a+b}{2}$
  Fin du Si
Fin du Tant que
Sortie : Afficher $\frac{a+b}{2}$

a	b	$\frac{a+b}{2}$	$\left(\frac{a+b}{2}\right)^2$	$\left(\frac{a+b}{2}\right)^2 > 10$	$b - a$	$b - a > 0{,}1$

(CNED, 2013, pp. 30-31)

On notera que l'auteur omet en titre de lignes : variable, entrée, traitement, sortie ; mais il met en colonne, à la suite des variables, les éléments pertinents à calculer pour tester la condition d'entrée dans la boucle (l'élément $b - a$) et celle relative à l'instruction conditionnelle (les éléments $\frac{a+b}{2}$ et $\left(\frac{a+b}{2}\right)^2$), mais aussi les conditions elles-mêmes ($b - a > 0{,}1$ et $\left(\frac{a+b}{2}\right)^2 > 10$). Ainsi, ce dernier tableau a un contenu qui excède largement celui donné dans la définition d'un tel tableau, donnée en page 40 de la séquence 1 :

« Le tableau de fonctionnement d'un algorithme décrit, à chaque étape de l'algorithme, le contenu des variables (une colonne par variable). »

2. Développer l'environnement technologico-théorique

On voit ainsi apparaître un « tableau de fonctionnement amélioré », objet qui ne figure pas dans le programme ni dans le document de la collection *Ressources pour la classe* que nous avons analysés ; le rapport institutionnel à cet objet pour l'institution « lycée français » semble fragmenté. Une enquête à propos de cet objet fait apparaître un certain nombre d'éléments qui sont absents de ce rapport institutionnel tout en étant de nature à améliorer la robustesse et la fiabilité des praxéologies algorithmiques. On met par exemple en lumière ainsi que le type de tâches T_F est considéré par les grands noms de l'algorithmique comme un type de tâches essentiel. Voici par exemple ce qu'en dit Donald Knuth : « An algorithm must be seen to be believed, and the best way to learn what an algorithm is all about is to try it » (Knuth, 1968, p. 4). Si D. Knuth n'utilise pas un tableau pour mettre en fonction l'algorithme d'Euclide qu'il donne en exemple, les tableaux apparaissent aujourd'hui comme un ingrédient incontournable d'une praxéologie algorithmique autour de T_F, et dont la dénomination en anglais s'avère être *trace tables*.

Une enquête à partir du moteur de recherche *Google*, *via* la requête « algorithm trace table », nous permet d'obtenir un premier document éclairant, extrait de *Wikipedia*, sous le titre « Trace Table ». On peut y lire :

> A trace table is a technique used to test algorithms, in order to make sure that no logical errors occur whilst the algorithm is being processed. The table usually takes the form of a multi-column, multi-row table ; with each column showing a variable, and each row showing each number input into the algorithm and the subsequent values of the variables.
>
> Trace tables are typically used in schools and colleges when teaching students how to program, they can be an essential tool in teaching student how a certain algorithm works and the systematic process that is occurring when the algorithm is executed.
>
> They can also be useful for debugging applications, using a trace table can help a programmer easily detect what error is occurring, and why it may be occurring. (Wikipedia, 2017-cor)

Puis, une seconde requête « desk check » nous conduit vers un document intéressant dans lequel Tim Whitford met en lumière une technique pour élaborer un « desk check » qui comporte une colonne par condition, point sur lequel l'auteur insiste :

> Desk checking is a manual (non-computerized) technique for checking the logic of an algorithm. The person performing the desk check effectively acts as the computer, using pen and paper to record results. The desk checker carefully follows the algorithm being careful to rigidly adhere to the logic specified. The desk check can expose problems with the algorithm.
>
> Desk checks are useful to check an algorithm (before coding) thereby confirming that the algorithm works as expected and saves time possibly writing a program that doesn't do what was intended. Another benefit of a desk check is that it confirms to the programmer/designer that the algorithm performs as intended.
>
> A desk check is normally done as a table with columns for :
>
> Pseudocode line number column (Pseudocode doesn't normally have lines numbers, but these are necessary in a desk check to specify the lines(s) being executed)

> One column per variable used. The columns should be in alphabetical order on variable name with the variable name at the top of the column. As the algorithm is executed, the new values of the variables are put in the appropriate column. Show working for calculations. If variable names consist of a number of words it is permissiable to put a space between each word in the name so that name fits better in the column by wrapping the next line. E.g. the variable column heading discount Price could be used rather than the actual variable name discountPrice.
>
> A condition column. The result of the condition will be true (T) or false (F). As the algorithm is executed, conditions are evaluated and the details are recorded in the column. Show working when evaluating the conditions. This is used whenever a condition is evaluated – IF WHILE or FOR statements all have explicit or implicit conditions.
>
> An Input/Output columns is used to show what is input by the user and displayed by the program. Show inputs with : the variable name, a "?" and the value input e.g. price? 200. Show outputs with the variable name. an =, and the value displayed (or just the value display) e.g. discountPrice = 180.
>
> Normally portrait page layout would be used for the desk check, however if the table is too wide, landscape layout may be used. (Whitford, 2010)

On y trouve également quelques explications supplémentaires : par exemple la nécessité de numéroter les lignes de l'algorithme pour savoir quelle ligne est exécutée ; l'intérêt de cette technique est qu'elle permet de vérifier l'algorithme, tout en gagnant du temps.

Voici un troisième document, qui complète quelque peu le propos du précédent (Learning Systems, 1997). Il explicite d'abord l'intérêt de constituer une « table de traces » d'un algorithme, qui reprend l'essentiel de ce que nous avons déjà rencontré. L'auteur donne ensuite un exemple :

> Here is a trace table which shows the states of our vowel counting algorithm, first the algorithm again :
>
> 1. Set *curr* to 1
> 2. Set *last* to number of letters on the page
> 3. Set *count* to 0
> 4. Read *letter* at *curr*
> 5. If *letter is vowel* then increment *count*
> 6. Increment *curr*
> 7. If *curr* <= *last* go to step 4
>
> Note that I have abbreviated the variable names, this is so that I can keep the columns in the trace table to a reasonable width. Each variable is shown in italic in the algorithm.
>
> Next the data : *A page*
>
> The algorithm has six variables – the numeric variables *curr*, *last*, *count*, the character variable *letter* and the logical values *letter is vowel* and *curr* <= *last* so the table will need at least six columns. Sometimes there is also a column which indicates a step number or which contains the statement being executed.

qu'il commente ainsi :

> The trace table reveals some interesting things about algorithms. For instance it stresses that a variable doesn't change value until it is reevaluated by a statement which operates on the variable. You can see in line two that the variable letter contains 'A' but the state of the condition letteris vowel is unknown (?). Line three in the trace table shows the execution of statement five of the algorithm – If letter is vowel then increment count – and now the variable letter is vowel in the trace table takes on a value. Another thing shown by the trace table is the amount of work done by algorithms which use repetition. Large repetitious algorithms with large amounts of data will lead to enormous trace tables. The size of the table can be minimised by only showing the start and stop states of a loop and then showing one iteration of the loop in a separate table. Yet another thing revealed by the trace table is the initialisation of variables - setting variables to a known state. Initialising variables is a good defensive algorithm design technique and is strongly recommended. Remember that your algorithm will end up as a computer program and you should not assume that the particular programming language will take care of the initialisation of variables. It is good practice to assume that it will not. (*Ibid*.)

On décèle ici une des raisons pour laquelle se « focaliser » sur les variables et leur état : mettre en évidence l'initialisation des variables permet d'éviter des problèmes d'implémentation de l'algorithme ou encore mettre en évidence les étapes de l'algorithme qui font changer (ou pas) les valeurs de certaines variables. On voit également que l'on peut constituer des sous-tables pour éviter que la table ne soit « énorme » et, surtout, l'attention portée à la mise en forme d'une technique et au développement de son environnement justificatif.

3. Conclusion

En s'appuyant sur l'environnement technologico-théorique mis au jour par l'enquête menée, la mise en forme de la technique, τ_F, peut être développée de la façon suivante :

> On repère les noms des variables d'entrée (*a*, *b*, *c*…) de l'algorithme A considéré ; on les compte et on note *m* leur nombre.
>
> Si l'algorithme A comporte des instructions conditionnelles ou itératives, on identifie la condition de chaque instruction conditionnelle, la condition d'arrêt de chaque boucle « Tant que » ou « Répéter … jusqu'à … » ; on les dénombre et on note *n* leur nombre.
>
> Si l'algorithme A comporte des contenus de variables ou des chaînes de caractères à afficher ; on les compte et on note *p* leur nombre.
>
> On numérote chaque instruction de l'algorithme ; on les compte et on note *s* leur nombre.
>
> On construit un tableau à $m + n + p + 1$ colonnes comportant une colonne par variable, une colonne par condition, une colonne par chaîne de caractères à afficher, une instruction par ligne (*cf.* tableau 3 ci-après). On peut réduire le format du tableau en organisant séparément des sous-tableaux sur le même principe, pour chacune des conditions relatives aux instructions conditionnelles ou itératives.

On peut éventuellement transposer le tableau de traces ci-dessous, les lignes devenant des colonnes, et les colonnes, des lignes, si la situation l'impose. On exécute l'algorithme « à la main » en simulant l'exécution par un ordinateur : on part de la première instruction, on remplit une ligne par instruction en complétant dans la colonne appropriée la valeur des variables, des tests ou conditions, et des affichages.

Instructions	**Variables**				**Conditions d'une instruction conditionnelle ou test d'arrêt d'une instruction itérative du type Tant que ou Répéter … jusqu'à …**			**Contenu des variables ou chaînes de caractères à afficher**		
n°	*a*	*b*	*c*	…	C_1	…	C_n	T_1	…	T_p
1										
…										
q										
q + 1										
…										
r										
r + 1										
…										
s										

Tableau 3. Table de traces d'un algorithme.

En appliquant la technique τ_F à l'algorithme de dichotomie de l'exercice de baccalauréat présenté à la page 7 de notre article, on obtient la table de traces suivante (*cf.* tableau 4 ci-après).

	Variables					**Conditions**		**Affichages**	
n°	***a***	***b***	***b* − *a***	***m***	***f*(*m*)**	***b* − *a* > 0,1**	***f*(*m*) < 1**	***a***	***b***
1	0								
2		1							
3			1						
4						Vrai			
5				0,5					
6					≈ 1,23				
7							Faux		
8		0,5							
9			0,5						
10						Vrai			
11				0,25					
12					≈ −3,09				
13							Vrai		
14	0,25								
15			0,25						
16						Vrai			
17				0,375					
18					≈ 0,10				
19							Vrai		
20	0,375								
21			0,125						
22						Vrai			
23				0,4375					
24					≈ 0,79				
25							Vrai		
26	0,4375								
27			0,0625						
28						Faux			

29								0,4375	
30									0,5

Tableau 4. Exemple de mise en œuvre d'une table de traces.

On peut voir alors à travers cette mise en œuvre les atouts que présente une telle technique. Elle permet non seulement de suivre au plus près les valeurs prises par les variables au fur et à mesure de l'exécution de l'algorithme, mais encore de détecter avec plus d'aisance d'éventuelles erreurs de logique, par exemple au niveau de la condition d'une boucle « Tant que » ou d'une instruction conditionnelle, contrairement à la technique employée dans le corrigé proposé par l'APMEP.

Le travail mené, illustré ici sur un type de tâches considéré comme paradigmatique, met ainsi en évidence un sous-développement des praxéologies à enseigner dans lequel on peut voir un effet de plusieurs conditions et contraintes. Parmi celles-ci, le fait que les objets informatiques aient à être fonctionnalisés dans le travail mathématique (Chevallard, 1992) conduit la noosphère à pousser en avant l'algorithmique comme secteur des mathématiques (Strock, 2013, p. 11) ce qui gêne sans doute la transposition des ingrédients technologico-théoriques propres à la construction de techniques fiables et robustes.

On peut penser que la difficulté à faire émerger un *logos* purement algorithmique pourrait alors provenir de l'asservissement de l'algorithmique aux mathématiques. De plus, la quasi-impossibilité de mettre en œuvre une réelle pédagogie de l'enquête dans un cadre curriculaire aussi strict que celui du lycée semble se propager *ipso facto* de l'enseignement des mathématiques à l'enseignement de l'algorithmique. Dialectiquement, les difficultés liées à la mise en œuvre d'une pédagogie de l'enquête dans l'enseignement de l'algorithmique conduit à atrophier le *logos* algorithmique.

Références

APMEP (2013). *Corrigé du sujet de mathématiques, du baccalauréat de la série scientifique, donné en Métropole, en 2013*. http://www.apmep.asso.fr/IMG/pdf/MetropoleS_correction_20_juin_2013.pdf

Artaud, M. (2007). La TAD comme théorie pour la formation des professeurs. Structures et fonctions. In L. Ruiz-Higueras, A. Estepa & F. J. García (Éds), *Sociedad, escuela y matemáticas. Aportaciones de la teoría antropológica de lo didáctico (TAD)* (pp. 241-259). Jaen, Espagne : Publicaciones de la Universidad de Jaén.

Chevallard, Y. (1992). Intégration et viabilité des objets informatiques dans l'enseignement des mathématiques. In Cornu, B. (ed), *L'ordinateur pour enseigner les mathématiques*, pp. 183-204. Paris : Éditions PUF. http://yves.chevallard.free.fr/spip/spip/IMG/pdf/Integration_et_viabilite_des_objets_informatiques.pdf

Chevallard, Y. (2008). Didactique de l'enquête codisciplinaire et des parcours d'étude et de recherche. In *Colloque international « Efficacité et Équité en Éducation »*, Université Rennes 2, Campus Villejean, 19-21 novembre 2008. http://yves.chevallard.free.fr/spip/spip/IMG/pdf/YC_-_Sem_nat_DDM_-_23_mars_2007.pdf

CNED (2013). Ressources Mathématiques – Seconde. In *Académie en ligne.* http://www.academie-en-ligne.fr/Lycee/Ressources.aspx?PREFIXE=AL7MA20

Knuth, D. E. (1968). *The Art of Computer Programming*. *Volume 1 : Fundamental Algorithms.* Chapter 1 - Basic concepts, p. 4. Reading, Massachusetts : Addison-Wesley, Second Printing, 1969.

Hebenstreit, J (2013). Informatique – Principes. In *Encyclopædia Universalis*. http://www.universalis.fr/encyclopedie/informatique-principes/4-algorithmes-et-decidabilite/

Learning Systems (1997). Step-Form algorithms - the simplest form of algorithm and: How to use Trace Tables. In *3GL Program Design*. http://users.evtek.fi/~jaanah/IntroC/DBeech/3gl_step.htm

Ministère de l'Éducation nationale (2009a). Programme de mathématiques de la classe de seconde. In *Bulletin officiel n°30 du 23 juillet 2009*. http://media.education.gouv.fr/file/30/52/3/programme_mathematiques_seconde_65523.pdf

Ministère de l'Éducation nationale (2009b). *Ressources pour la classe de seconde - Algorithmique*. Éduscol. http://media.eduscol.education.fr/file/Programmes/17/8/Doc_ress_algo_v25_109178.pdf

Ministère de l'Éducation nationale (2011). Programme de mathématiques de la classe de terminale scientifique. In *Bulletin officiel spécial n°8 du 13 octobre 2011*. http://media.education.gouv.fr/file/special_8_men/98/4/mathematiques_S_195984.pdf

Ministère de l'Éducation nationale (2013). *Sujet de mathématiques, du baccalauréat de la série scientifique, donné en Métropole, en 2013*. http://www.maths-france.fr/Terminale/TerminaleS/ProblemesBac/2013/2013-france-metropolitaine-specifique-enonce.pdf

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

Whitford, T. (2010). *Desk Check Guide*. http://ironbark.xtelco.com.au/subjects/PE/PEWeb/other_resources/desk_check_guide.html

Wikipedia (2017). *Trace table*. http://en.wikipedia.org/wiki/Trace_table