Le numérique n'existe pas*

Jamil Alioui

19 janvier 2022

> À la fermeture matérielle des soudures, des rivets et des cachets de garantie, s'ajoute une fermeture plus essentielle et plus aliénante : l'objet n'est plus décodable, plus compréhensible comme résultat d'une opération de construction. On ne peut lire en lui l'opération constructrice. Il est étranger comme une langue étrangère.
>
> Simondon

Il est certain que le champ de recherche que l'on nomme habituellement « humanités numériques » se situe quelque part entre la technique et la culture. En première analyse, tout porte à croire que ces deux ensembles de problèmes – problèmes culturels et problèmes techniques – recouvrent respectivement les deux termes de l'expression : d'un côté les « humanités » renverraient à la culture comme ensemble de normes, de valeurs, d'œuvres, de pratiques, etc. ; de l'autre le « numérique » renverrait à la technique comme ensemble disparate de réalités matérielles et logicielles : réseaux, machines, codages, calculs, langages de programmation, programmes, interfaces, etc. Alors les « humanités numériques » seraient quelque chose comme l'étude de « la numérisation de la culture et [de] son traitement informatique » [1], pour reprendre une expression proposée par Raphaël Baroni et Claus Gunti.

J'aimerais proposer ici de considérer les choses un peu différemment et problématiser l'existence du numérique à partir d'une articulation technologique avec

*Conférence prononcée le 19 janvier 2022 à Besançon, dans le cadre du séminaire du laboratoire « Logiques de l'agir » de l'Université de Franche-Comté (axe 3, sur les « humanités numériques »), sur invitation de Sarah Carvallo et Fabien Ferri.

1. Raphaël Baroni et Claus Gunti, *Introduction à l'étude des cultures numériques*, Paris, Armand Colin, 2020, p. 15.

l'informatique dite analogique, informatique presque oubliée de nos jours bien qu'historiquement cruciale. Pourquoi proposer une telle articulation ? Pour deux raisons.

La première s'explique par la lecture de Gilbert Simondon, qui m'a occupé ces dernières années dans le cadre de ma thèse. En particulier, j'ai pu constater – et je ne suis pas le seul [1] – l'absence de pensée approfondie des ordinateurs et du numérique chez Simondon. L'étude des raisons pour lesquelles un philosophe de la technique aussi génial que lui n'a pas approfondi l'étude de l'informatique et en particulier celle de l'informatique que nous appelons aujourd'hui « numérique » a été jusqu'à me faire remanier la structure entière de ma recherche, alors qu'elle était initialement inspirée par la médiologie de Friedrich Kittler et cantonnée à l'étude technologique des machines à écrire. La recherche m'a amené à identifier la possibilité d'une réponse valable au problème de cette absence thématique, à condition de suivre la piste de l'analogie, un concept central et structurant chez Simondon, en particulier dans le cadre de la technologie.

La seconde raison est génétique : l'analogique offre un moyen d'esquisser une complication technologique de la relation entre le logiciel et le matériel, relation que l'on prend souvent comme point de départ pour penser le numérique. On considère en effet habituellement que programmer un ordinateur numérique signifie résoudre un problème en « faisant exécuter » par la machine une série d'instructions préalablement « rédigées » et « données » sous forme textuelle ou écrite. On trouve par exemple une telle perspective chez Clarisse Herrenschmidt, alors qu'elle définit le logiciel comme

1. Voir notamment : Ludovic Duhem, « Penser le numérique avec Simondon », in *NepH, Nouvelle revue de Philosophie* (2014), Numéro spécial « Philosophie du numérique » (cité en référence au document disponible en *open access* sur la plateforme *Academia.edu*) ; Ludovic Duhem, « La réticulation du monde. Simondon penseur des réseaux », in *Gilbert Simondon ou l'invention du futur*, sous la dir. de Vincent Bontems, Paris, Klincksieck, 2016, p. 227-239 ; Frédéric Pascal, « Gilbert Simondon et l'informatique I », in *Gilbert Simondon ou l'invention du futur*, sous la dir. de Vincent Bontems, Paris, Klincksieck, 2016, p. 241 ; *Ibid.*, p. 241 ; Jérémy Grosman, « Simondon et l'informatique II », in *Gilbert Simondon ou l'invention du futur*, sous la dir. de Vincent Bontems, Paris, Klincksieck, 2016, p. 247 ; Michael Kurtov, « Simondon et l'informatique III. L'évolution des langages de programmation à la lumière de l'allagmatique », in *Gilbert Simondon ou l'invention du futur*, sous la dir. de Vincent Bontems, Paris, Klincksieck, 2016, p. 255 ; Coline Ferrarato, « Penser le numérique avec Simondon ? », in *Implications philosophiques* (2019), sous la dir. de Jean-Hugues Barthélémy, titre du dossier : « Simondon 1958-2018 », url : http://www.implications-philosophiques.org/non-classe/penser-le-numerique-a-partir-de-simondon/ et Coline Ferrarato, *Philosophie prospective du logiciel*, Londres, ISTE Editions, 2019.

> ensemble, par définition écrit, de caractères, de chiffres, de mots et de règles, qui permettent d'assembler ces éléments pour transcrire un algorithme, écrire des programmes et donner des instructions à un ordinateur. [1]

De même, l'interprétation du numérique comme écriture à trois niveaux – « idéal », « applicatif » et « sémiotique » –, proposée il y a une décennie par l'École de Compiègne [2], admet, selon moi, un présupposé similaire. Disons qu'il en va de même pour toutes les réflexions qui considèrent la programmation comme une activité technique sans toutefois la saisir ou l'éclairer en relation problématique avec l'individuation de la machine. Or, en informatique analogique, programmer signifie justement reconfigurer et restructurer les organes internes de façon à faire de la machine elle-même un analogue manipulable du problème à résoudre. Ainsi, du point de vue analogique, il n'y a écriture que dans la mesure où la machine « concrétisée », en train de fonctionner, est elle-même un symbole. Du moins, il m'a paru que si Simondon, comme je le pense, a préféré l'analogique au détriment du numérique pour décrire l'ensemble de ce que, sur les traces de Jacques Lafitte [3], il nomme des « dispositifs à information », alors il s'agit de comprendre ce qu'un tel choix signifie exactement, tant d'un point de vue technologique que symbolique ou culturel.

Ma stratégie consiste à montrer – en cinq moments – que la programmation informatique, conçue comme une activité indépendante du fonctionnement réel de l'ordinateur, n'est pas en mesure d'être étudiée par une technologie ; elle ne peut être problématisée et connue que par une praxéologie, c'est-à-dire par une sociologie des usages, car elle n'est qu'un usage possible de l'ordinateur. L'hypothèse de départ, issue de ma lecture de Simondon, est que seule la programmation qui consiste à individuer l'ordinateur – c'est-à-dire à en « perpétuer l'invention » – peut être comprise comme une activité *stricto sensu* technique et ainsi être étudiée et connue par une technologie. (Le mot technologie est entendu ici comme discours scientifique sur les opérations techniques, comme l'expliquent Jacques Guillerme et Jan Sebestik [4].) Or, il ne suffit pas, en informatique numérique, de « descendre » au niveau de la programmation en « langage machine » pour retrou-

1. Clarisse Herrenschmidt, *Les trois écritures. Langue, nombre, code*, Paris, Gallimard, 2007, p. 405.

2. Stéphane Crozat, Bruno Bachimont, Isabelle Cailleau, Serge Bouchardon et Ludovic Gaillard, « Éléments pour une théorie opérationnelle de l'écriture numérique », in *Document numérique* 14 (2011), p. 9-33.

3. Jacques Lafitte, *Réflexions sur la science des machines*, Paris, Vrin, 1932, 1972.

4. Jacques Guillerme et Jan Sebestik, « Les commencements de la technologie », in *Documents pour l'histoire des techniques* 14 (2007), url : http://dht.revues.org/1226.

ver une telle individuation technique. Dans la mesure où l'idée de « descendre » ou de « monter » implique déjà des degrés ou des niveaux d'abstraction – comme dirait Luciano Floridi [1] –, le langage machine entendu comme langage « au bas de l'échelle » est déjà, sinon une contradiction, du moins une « abstraction » [2] du fonctionnement réel de l'ordinateur. Une programmation correspondant réellement et actuellement à la poursuite de l'individuation technique de la machine – c'est-à-dire aussi, au sens de Simondon, une programmation capable de se « naturaliser » [3] – n'existe pour l'instant (et jusqu'à preuve du contraire) qu'en informatique analogique ; de là, à mon sens, la fertilité du rapprochement entre les deux perspectives.

Quel est le but de ma proposition ? En un mot : contredire l'idée selon laquelle le numérique serait une espèce d'extériorité ou d'événement qui, de façon irruptive, viendrait comme infléchir ou altérer un cours jusqu'alors « normal » de la culture. Car affirmer que le numérique est quelque chose qui arrive à la culture c'est substantialiser une certaine idée de la culture. Or, à une telle substantialisation correspond la privation des moyens de nommer et de problématiser l'information transindividuelle. Réduire la culture à des formes figées c'est rapporter l'actualité opératoire et régulatrice aux limbes de l'indétermination, c'est-à-dire en-deçà de toute régulation possible. De surcroit, il m'a paru possible – à partir de cette proposition technologique critique – de comprendre pourquoi Simondon ne s'est pas intéressé au « numérique » : sa technologie ne permet pas de concevoir une réalité technique digne de ce nom qui existe en contradiction avec la dimension « régulatrice par essence » [4] de la culture. Ainsi, Simondon, philosophe de la technique parce que philosophe de la culture, ne s'est pas intéressé au « numérique » parce que, du point de vue de sa technologie – analogique de part en part –, le « numérique » ne peut tout simplement pas exister.

En tous cas, montrer que le numérique n'est pas une altération transcendante de la culture mais un nouveau mode d'existence de cette dernière – et le montrer non pas à partir d'une sociologie des usages ou d'une praxéologie mais bien à partir d'une philosophie de la technique ou d'une technologie – me semble dégager une perspective originale sur ce phénomène que l'on appelle « transition numérique » en nous rappelant – comme le faisait déjà en son temps Simondon

1. Luciano Floridi, *The Philosophy of Information*, Oxford University Press, 2011.
2. Gilbert Simondon, *Du mode d'existence des objets techniques*, Paris, Aubier, 1958, 1969, 1989, 2001, 2012, p. 26.
3. *Ibid.*, p. 57.
4. *Ibid.*, p. 18.

à l'encontre des discours fatalistes – que subsiste, contre vents et marées, une différence cruciale et un choix à faire entre adaptation et régulation.

1. Le moment de l'interprétation

La distinction entre informatique analogique et informatique numérique trouve une partie de ses origines dans la caractérisation de ce que manipule l'ordinateur. Le cybernéticien français Louis Couffignal explique – dans un texte de 1952 – que les machines numériques, contrairement aux machines analogiques, « ne s'occupent que de nombres écrits avec des chiffres, c'est-à-dire de nombres rationnels » [1]. Il veut dire par là qu'une machine numérique opère sur des représentations de quantités, traitées en dernière instance sous la forme de nombres entiers. Une lecture de John von Neumann corrobore : « dans une machine digitale décimale, chaque nombre est représenté de la même façon que dans l'écriture ou l'imprimerie conventionnelles, c'est-à-dire comme une suite de chiffres décimaux » [2]. Le fait que von Neumann se réfère ici à des machines décimales qui n'existent plus aujourd'hui est moins important que la forme unidimensionnelle de suite ou de série que prennent ces représentations de quantités : cette forme est conventionnelle [3], nous dit-on. Comme le précise à juste titre Jean Lassègue lisant Turing, ces marques que manipule l'ordinateur numérique sont culturelles, c'est-à-dire qu'elles sont elles-mêmes des représentations : la machine opère, écrit-il, « sur des marques interprétées par l'être humain comme les signes des nombres » [4]. Toutefois – et ce point est important –, si tous les nombres sont des représentations de quantités, toutes les représentations de quantités ne sont pas nécessairement des nombres. Pour cette raison, il me semble indiqué de parler de signes de quantités plutôt que, comme Lassègue, de signes de nombres. Ceci dit, ce qui importe ici est que l'ordinateur numérique manipule des signes et non les quantités elles-mêmes. Le fonctionnement technique de l'ordinateur numérique n'est en lien qu'avec une formule préalable du problème, avec sa description ou sa représentation préexistante, en chiffres arabes par exemple ; ce qui signifie que l'ordinateur numérique n'est pas en lien immédiat avec le problème et que la relation d'information entre le problème à résoudre et la machine est inactuelle.

1. Louis Couffignal, *Les machines à penser*, Paris, Minuit, 1952, p. 59.
2. John von Neumann, *L'ordinateur et le cerveau*, trad. par Pascal Engel, Paris, Flammarion, 1996, p. 18.
3. Jean Lassègue, *Turing*, Paris, Les Belles Lettres, 2003, p. 96.
4. *Ibid.*, p. 26.

Cette inactualité concerne aussi la mobilisation et le stockage de l'information au sein de l'objet technique ordinateur. Comme nous l'explique le professeur Philippe Breton, une machine est considérée comme analogique lorsque « l'information a comme support un signal continu, une oscillation dans une ligne électrique par exemple » [1]. Que signifie ici « continu » ? Le qualificatif de continu, écrit Couffignal, « traduit bien le fait que l'on ne peut passer d'un état [physique] à un autre, sans traverser des états intermédiaires, et même une infinité d'états intermédiaires » [2]. Autrement dit, la continuité analogique est celle des états physiques possibles de la machine : la machine analogique peut revêtir un nombre infini d'états parce qu'elle manipule des quantités physiques plutôt que des représentations de ces dernières. En un certain sens, on pourrait aller jusqu'à dire que l'ordinateur analogique mesure plutôt qu'il ne calcule. Ses résultats sont ses états ; ils peuvent et même doivent être interprétés. Pour cette raison, si l'on ne peut pas dire que l'ordinateur analogique manipule des signes, on peut dire qu'il est lui-même le signe ou, pour parler comme Simondon, le symbole d'un problème à résoudre (le mot « symbole » étant plus adapté dans la mesure où le problème se continue réellement et actuellement dans la machine analogique). Mais si la machine est elle-même la représentation d'une quantité alors des affirmations comme celle de John von Neumann – selon laquelle

> Dans une machine analogique, chaque nombre est représenté par une quantité physique appropriée, dont la valeur mesurée, selon une unité définie au préalable, est égale au nombre en question. [3]

– de telles propositions introduisent une certaine confusion car elle décrivent l'informatique analogique selon une triade conceptuelle – quantité, nombre et machine – dans laquelle, en dernière analyse, le concept de nombre est contingent (la machine étant elle-même la représentation de la quantité).

Réciproquement, peuvent être dites « numériques » les machines convoyant l'information en unités élémentaires. Ces machines fonctionnent donc à partir de signaux à valence finie, autrement dit, les grandeurs y sont représentées avec un nombre limité d'éléments. Par exemple, lorsqu'on dit travailler avec un ordinateur « huit bits », on entend que le bus de données de la machine convoie des signaux composés de huit occurrences de deux types de marques ou, si l'on préfère, de huit marques pouvant chacune revêtir deux états différents. Concrètement, les câbles

1. Philippe Breton, *Une histoire de l'informatique*, Paris, Seuil, 1987, 1990, p. 146.
2. Couffignal, *Les machines à penser*, *op. cit.*, p. 59.
3. von Neumann, *L'ordinateur et le cerveau*, *op. cit.*, p. 15.

qui relient les organes les uns aux autres – par exemple la mémoire et l'unité arithmétique – sont au nombre de huit, et chacun d'eux conduit une différence de potentiel électrique soit nulle, soit positive, par exemple de cinq volts. Techniquement, ces deux états physiques sont obtenus par stabilisation d'un régime de causalité, par le moyen d'éléments techniques semi-conducteurs (en règle générale des transistors). Il ne s'agit donc jamais d'un potentiel strictement nul ou strictement égal à cinq volts, mais bien plutôt d'un potentiel inférieur à une certaine valeur, supérieur à une autre valeur, ou éventuellement compris entre deux valeurs. Toutefois, en considérant de tels états comme étant « binaires », nous effectuons une interprétation, dans la mesure où nous distinguons deux états matériels selon un seuil abstrait, conventionnel et, en ceci, extérieur à la machine. La connaissance de ces seuils implique d'ailleurs, très concrètement, le recours à une abondante documentation relative aux composants électroniques employés.

Du moment que l'on admet de faire correspondre à deux classes de tensions électriques deux signes, par exemple « rond » et « carré » – ou « zéro » et « un » –, on est en mesure de considérer ces signes comme les lettres d'un alphabet élémentaire ; on peut dire ensuite que des mots de huit caractères d'un alphabet ne comprenant que deux lettres, « rond » et « carré » – ou « zéro » et « un » –, sont matérialisés, solidifiés ou encore « mécanisés » dans la machine. La relation entre une série de tensions électriques et un nombre entier implique ici une convention de lecture, qui est aussi un facteur externe engrammé au plus profond de la machine : l'état physique « 10010110 » peut tantôt signifier « 150 », tantôt signifier « 105 », selon qu'on adopte une lecture dite *big endian* ou une lecture dite *little endian* de la série de tensions électriques. D'accord sur le sens de la lecture, on peut enfin choisir d'attribuer aux combinaisons possibles – et si la machine a un bus de 8 bits cela fait 256 combinaisons possibles – autant de lettres et de symboles que l'on veut. Cette mise en correspondance de fonctionnements techniques et de significations issues de la culture arrive, on le voit, bien avant que l'on puisse même parler de langage de programmation. La technicité mise en œuvre pour individuer les états physiques de la machine numérique se trouve interprétée en ronds, carrés, zéros, un – c'est-à-dire selon des formes préexistantes et non en tant que fonctionnement actuel –, bien avant que la machine soit programmée. En informatique analogique, au contraire, la programmation a pour but de mettre la machine dans un état interprétable ; l'état physique est donc interprété au plus tôt en même temps que la programmation.

2. Imprécision et erreur

La finitude de la machine numérique ainsi que la rationalité des nombres qu'elle manipule ont pour corrélat que ces machines « ne peuvent atteindre les grandeurs physiques intervenant dans les problèmes qu'à travers des valeurs approchées de leur mesure » [1]. Von Neumann corrobore Couffignal : « la limitation principale des machines analogiques est leur degré de précision » [2]. Il affirme que les machines analogiques ne peuvent atteindre une précision que de 3 à 5 décimales alors que les machines numériques « peuvent atteindre n'importe quel degré de précision requis ». D'un point de vue matériel, selon von Neumann toujours, « il est beaucoup plus facile d'augmenter le degré de précision dans un régime numérique que dans un régime analogique ». Ce mythe est séduisant ; sa diffusion explique certainement, en partie du moins, pourquoi l'histoire a tant favorisé l'informatique numérique. On le retrouve notamment trente ans plus tard, en 1987, dans le texte d'un des directeurs du développement scientifique d'IBM-France, René Moreau, pour qui, je cite, le calcul numérique « permet en principe, d'obtenir la précision désirée » [3].

Toutefois cette précision n'est arbitraire qu'en apparence ; en réalité elle a un coût : son augmentation implique soit une miniaturisation des composants, soit une augmentation du nombre d'instructions successives dans le logiciel. Or, la miniaturisation ne peut aller au-delà de la granularité physique ; de même, l'augmentation de la taille du programme implique une augmentation de la mémoire ainsi que de la durée d'exécution, c'est-à-dire une augmentation des ressources énergétiques et matérielles, c'est-à-dire physiques. En bref, entretenir l'idée que le degré de précision numérique puisse être fixé arbitrairement implique une abstraction à l'endroit de limites physiques pourtant bien réelles de la machine. Au fond, cette idée implique un postulat de discontinuité entre le calculant et le calculé ; comme si le calculant n'existait pas dans le même monde que le calculé. Le spécialiste allemand des ordinateurs analogiques Bernd Ulmann est plus fonctionnaliste – et donc aussi plus raisonnable – lorsqu'il explique que les ordinateurs numériques possèdent un pouvoir que les ordinateurs analogiques n'ont pas : celui de convertir du temps en précision. [4] Historiquement, il semble que la prise

1. Couffignal, *Les machines à penser*, *op. cit.*, p. 59.
2. von Neumann, *L'ordinateur et le cerveau*, *op. cit.*, p. 34.
3. René Moreau, *Ainsi naquit l'informatique*, Paris, Dunod, 1987, p. 21.
4. Bernd Ulmann, *Analog Computing*, Munich, Oldenbourg Verlag, 2013, p. 4.

de conscience de la continuité du calculant et du calculé en numérique arrive *a posteriori* et corresponde à la fondation de la théorie dite de la complexité. Réciproquement, l'informatique analogique s'est construite *a priori* sur l'hypothèse d'une communauté et d'une continuité entre le calculant et le calculé, d'une continuité de la physique ou de la nature, en même temps objet et méthode de programmation.

On comprend mieux l'enjeu de la distinction entre manque de précision et présence d'erreurs si l'on s'arrête sur les réponses techniques recherchées et apportées. Le manque de précision – qui caractérise l'informatique analogique – fait signe vers le besoin d'une meilleure articulation de la science physique et de ses applications techniques en général : il exige une meilleure concrétisation (pour reprendre un terme simondonien), c'est-à-dire un progrès technique ; la présence d'erreurs – inéluctable en informatique numérique – réclame l'invention de procédures *ad hoc* de « gestion des erreurs », telles que celles auxquelles faisait déjà référence von Neumann, alors qu'il définissait « l'art du calcul par ordinateur » comme étant l'art de réduire les effets de l'amplification des erreurs arithmétiques « à un niveau négligeable » [1]. Il ne s'agit donc pas nécessairement, en numérique, de concrétiser davantage la machine.

Pourquoi est-il essentiel de ne pas confondre les deux dimensions, manque de précision et présence d'erreurs ? Lorsque René Moreau, comparant l'informatique numérique et l'analogique, affirme que « la lecture d'un nombre n'est jamais entachée d'erreurs [alors que] celle d'une indication fournie par une aiguille qui se déplace sur un cadran l'est toujours » [2], il va trop vite en besogne et omet de préciser que toute machine numérique tronque nécessairement – et d'une nécessité apodictique – les informations échantillonnées ; il ne précise pas non plus que l'indication fournie par l'aiguille n'est « entachée d'erreurs » que si le cadran est gradué, c'est-à-dire si une opération de numérisation a lieu. Réciproquement, le cadran d'un thermomètre où il serait simplement indiqué « chaud » et « froid » serait très imprécis, mais non « entaché d'erreurs ».

Le moment de la numérisation joue donc un rôle crucial dans la mise en relation du problème à résoudre et du fonctionnement de la machine : la numérisation est un moment d'interprétation mais elle est aussi le moment où se crée une distance infranchissable entre l'actualité du problème et son traitement informatique. Pour dire les choses autrement, en analogique, le fait d'être imprécis ne contredit

1. von Neumann, *L'ordinateur et le cerveau*, *op. cit.*, p. 36.
2. Moreau, *Ainsi naquit l'informatique*, *op. cit.*, p. 20.

pas la réalité d'une relation de connaissance avec le problème alors qu'en numérique, la fixation d'un seuil auquel l'erreur devient négligeable est une opération qui transcende le fonctionnement de la machine et qui est relative à une finalité externe, antérieure et inactuelle. Mettre sur un même plan le manque de précision analogique et les erreurs inhérentes au numérique, c'est faire preuve de pragmatisme et d'utilitarisme, mais c'est aussi renoncer à des moyens de problématiser l'actualité et la réalité de la relation d'information.

3. Analogies de structures et analogies d'opérations

Comment concevoir le rôle des nombres et des mathématiques en informatique analogique ? Dans quelle mesure ne sont-ils pas, eux aussi, une réalité qui s'interface entre le problème et la machine ?

Il est vrai que si, selon l'« allagmatique » de Simondon [1], on distingue entre des analogies de structures d'un côté et des isodynamismes ou analogies opératoires de l'autre, alors les mathématiques peuvent être entendues comme une représentation de dynamismes physiques. Ainsi, comme l'écrit Moreau, « il y a bien analogie entre le montage électrique et la fonction [décrivant une réalité problématique] » [2], mais cette analogie – il est essentiel de le préciser (Moreau ne le fait pas) – est une analogie d'opérations et non une analogie de structures. Employer les mathématiques en informatique analogique est donc un moyen technique de transposer ou de traduire les dynamismes problématiques d'un domaine structurel (par ex. l'hydraulique ou la démographie) à un autre (par ex. la mécanique ou l'électronique). Mais « moyen technique » ne signifie pas « nécessité apodictique » ; pour s'en assurer il suffit de remarquer qu'il existe de nombreuses réalités techniques convoyant une information quantitative sans « passer » ni par les mathématiques ni par les nombres, à l'instar des bouées, de certains baromètres ou de dispositifs de génération, d'enregistrement et de restitution d'information à finalité esthétique. En effet, lorsqu'on affirme que programmer un ordinateur analogique consiste à le structurer en vue de créer la simulation d'un problème, on ne fait qu'affirmer que la programmation transforme la machine en analogie opératoire. En informatique analogique c'est l'invention de l'isodynamisme qui est l'opération de programmation. Que les mathématiques aient une fonction dans

1. Gilbert Simondon, *L'individuation à la lumière des notions de forme et d'information*, Grenoble, Millon, 2005, 2013, 2017, p. 529.

2. Moreau, *Ainsi naquit l'informatique*, *op. cit.*, p. 19.

ce processus d'invention est contingent. Nous pouvons donc dire, avec Jean Lassègue, que programmer une machine analogique consiste à « [recréer] un analogue physique – reposant sur une mesure de longueurs continues – de la fonction mathématique à calculer » [1] à condition de bien préciser que la fonction mathématique n'est qu'un moyen technique de créer l'analogie opératoire ou l'isodynamisme. C'est en ce sens que l'ordinateur analogique programmé peut être considéré comme une représentation ou un symbole du problème à résoudre sans requérir *en droit* d'être une chose mathématique.

Réciproquement, l'informatique numérique peut, à juste titre, être dite « numérique » car elle implique nécessairement un premier moment d'interprétation auquel correspond en général l'attribution de chiffres à des classes de causalités circulaires. En ce sens, s'il y a des analogies en informatique numérique, il ne peut s'agir que d'analogies de structures. En informatique numérique les isodynamismes sont apodictiquement impossibles. Cette proposition prend tout son sens si l'on considère l'une des plus importantes difficultés rencontrées durant l'histoire de la « numérisation », celle du « temps réel ». D'un point de vue analogique, il y a *a priori* une isochronie entre la nature, l'ordinateur et l'opérateur ; mais d'un point de vue numérique, le temps réel est un problème insoluble et, comme l'explique parfaitement l'historien Jérôme Ramunni, seul « un temps suffisamment court pour que l'on puisse, par exemple, modifier la trajectoire d'un avion en temps utile » [2] est techniquement vraisemblable. Autrement dit, le temps réel numérique est bien une imitation du temps réel ; il est une rapidité d'exécution assez élevée pour faire croire à l'utilisateur qu'il y a isochronie entre l'imitation et l'imité. On retrouve ici la présence d'un seuil arbitraire car externe au fonctionnement de la machine et fondé sur des pratiques conventionnelles ou des habitudes de travail nécessairement inactuelles. La tension vers l'infini du nombre d'éléments traités par seconde ne permet pas à la machine numérique d'être synchronisée avec l'actualité ; le fossé est absolu.

Est-ce à dire que l'informatique numérique n'est pas opératoire ? Est-ce à dire qu'elle n'est pas dynamique ? À vrai dire, cela dépend de ce que l'on entend exactement par opératoire, dynamique, statique. Je propose une réponse à partir de la façon dont l'informatique numérique rapporte le problème à résoudre à une certaine conception de la subjectivité. John von Neumann, par exemple, identifie à plusieurs reprises et tout à fait spontanément la « résolution d'un problème » à

1. Lassègue, *Turing*, *op. cit.*, p. 26.
2. Jérôme Ramunni, *La physique du calcul. Histoire de l'ordinateur*, Paris, Hachette, 1989, p. 97.

« l'application de l'intention de l'utilisateur à la machine » [1]. Ainsi, il rapporte « le montage du problème – l'expression du problème à résoudre » à « l'intention de l'utilisateur » [2]. De même, une remarque assez générale de Donald Ervin Knuth et Luis Trabb Pardo, dans leur imposante recherche sur la préhistoire des langages de programmation, semble orientée par un présupposé identique :

> Au cours des siècles suivant l'époque de la civilisation grecque, et malgré un développement remarquable de l'écriture des relations fonctionnelles statiques, les mathématiciens n'ont jamais inventé de bonne notation pour les processus dynamiques. Lorsqu'une procédure impliquait des séquences de décisions non triviales, les méthodes disponibles pour une description précise restaient informelles et plutôt lourdes. [3]

Selon les auteurs, ce qui est entendu par « processus dynamique » – et dont l'écriture fait problème – ne correspond qu'à un processus « impliquant des séquences de décisions non triviales ». Ce texte, portant sur les langages de programmation et par conséquent limité à l'informatique numérique, confine spontanément la signification du terme « dynamisme » au champ de la décision, c'est-à-dire au domaine de l'esprit conscient, réfléchissant et agissant. Aussi étonnant que cela puisse paraître, dans cette perspective, les équations fonctionnelles et différentielles décrivant des mouvements de particules ou de planètes – c'est-à-dire des mouvements physiques –, sont des relations fonctionnelles statiques et triviales. Autrement dit, l'informatique numérique ne se conçoit comme dynamique et opératoire que dans la mesure où elle présuppose la liberté d'un esprit conscient, logique et imprévisible mais porteur d'intentions, « situé » dans un monde matériel déterministe et, en ce sens, statique. À ce cognitivisme caractéristique de l'informatique numérique semble correspondre une métaphysique kantienne, selon laquelle seules les représentations ont rang d'être, radicalement séparées d'une inatteignable nature.

4. Individualité et individuation techniques

Cette affinité entre l'ordinateur numérique et une certaine conception de la subjectivité ou, du moins, de la personnalité psychosomatique a une influence sociologique sur la relation entre l'utilisateur et la machine. En particulier, c'est sous

1. von Neumann, *L'ordinateur et le cerveau*, *op. cit.*, p. 22.
2. *Ibid.*, p. 26.
3. Donald Ervin Knuth et Luis Trabb Pardo, « The Early Development of Programming Languages », in *A History of Computing in the Twentieth Century*, sous la dir. de Nicholas Metropolis, Jack Howlett et Gian-Carlo Rota, Academic Press, 1980, p. 200, je traduis.

la modalité du service et de l'asservissement qu'une telle relation se développe ; elle prend bientôt la forme d'un système dit d'« exploitation ». Cette modalité relationnelle s'est manifestée dès les origines : d'un côté, comme le montrent parfaitement Knuth et Pardo, les informaticiens ont immédiatement entrevu le besoin d'« assistances à la programmation » alors inexistantes [1] (je vais revenir sur ce point) ; de l'autre, les exigences des utilisateurs mathématiciens, physiciens ou ingénieurs paraissaient raisonnables alors que, comme le rapporte Ramunni, ils « ne voulaient pas connaître les détails de la machine mais […] désiraient s'en servir facilement » [2]. Par là s'éclaire notamment le fait que la programmation des ordinateurs numériques ait immédiatement été conçue, selon le mot de von Neumann, comme une « organisation des ordres logiques » [3]. Toutefois, initialement, ce système d'exploitation était un organe physique comparable au panneau de connexion caractérisant les ordinateurs analogiques.

> les points de contrôle étaient des objets physiques réels, et leurs connexions par câblage exprimaient la nature du problème. Dans le cas présent [nous sommes en 1957-58], les ordres sont des entités idéales, stockées dans la mémoire, et ce sont par conséquent les contenus de ce segment particulier de la mémoire qui expriment le problème. [4]

Cette « expression du problème » – comme dit von Neumann – qui deviendra « *software* » désigne donc bien, au départ, un organe physique, matériel, de la machine qui était un organe de contrôle. On voit ici comment l'idée d'une programmation comme chose écrite ou – si l'on préfère – comme langage trouve une origine techno-symbolique dans une certaine lisibilité de la réalité technique. Ainsi, si on ne peut pas vraiment contredire Dominique Pignon qui, commentant von Neumann [5], explique que la distinction entre *software* et *hardware* s'origine dans le présupposé d'une distinction entre « organisation logique » de l'ordinateur et « réalisation matérielle », il paraît tout de même crucial de faire remarquer que si l'on admet que la distinction technologique entre « organisation logique » et « réalisation matérielle » se rapporte à la distinction aristotélicienne entre forme et matière, alors une étonnante proximité s'installe entre l'idée de forme logique et celle d'injonction. En ce sens, Gilbert Simondon, alors qu'il s'interroge sur les raisons qui fondent le rapprochement entre individualité et matérialité, souligne que la forme logique peut être entendue comme une « intention » ou un « ordre

1. *Ibid.*, p. 198.
2. Ramunni, *La physique du calcul*, *op. cit.*, p. 162.
3. von Neumann, *L'ordinateur et le cerveau*, *op. cit.*, p. 70.
4. *Ibid.*, p. 29.
5. *Ibid.*, p. 27, note.

de fabrication », en particulier – je cite – « lorsque celui qui pense n'est pas celui qui travaille » [1]. En un mot, la distinction entre forme logique et réalisation matérielle, à l'origine de l'idée de logiciel et de système d'exploitation, substantialise un rapport de travail qui se caractérise par le fait que certaines personnes lisent et écrivent la technique alors que d'autres ne font que l'exécuter et l'utiliser.

Ainsi, une critique bien connue formulée du point de vue numérique consiste à reprocher aux ordinateurs analogiques d'être dépourvus de mémoire et d'être « limités » à une logique combinatoire ainsi qu'au traitement de données actuelles. Selon cette même critique, l'ordinateur numérique – contrairement à l'analogique – mettrait en œuvre une logique séquentielle et posséderait la capacité de traiter des données de façon cumulative, c'est-à-dire de reprendre des données déjà traitées. C'est techniquement vrai. Mais en quoi est-ce un avantage sur l'informatique analogique ? Selon quel référentiel ? Selon quelle exigence ? La question est parfaitement légitime dans la mesure où, si l'on adopte une perspective analogique, rien n'empêche de considérer l'ordinateur analogique comme étant lui-même une mémoire ou une écriture. Cette proposition corrobore parfaitement la conception de la programmation analogique comme symbolisation par structuration technique plutôt que par imitation de systèmes de représentation préexistants. Considérons à cet égard la façon dont Bernd Ulmann décrit l'ordinateur analogique :

> sa structure interne n'est pas fixée – en fait, un problème est résolu sur une telle machine en modifiant sa structure de manière appropriée pour générer un *modèle*, un *analogue* du problème. Cet analogue est ensuite utilisé pour *analyser* ou *simuler* le problème à résoudre. Ainsi, la structure d'un ordinateur analogique qui a été configuré pour résoudre un problème spécifique représente le problème lui-même tandis qu'un ordinateur numérique à programme enregistré conserve sa structure et seul son logiciel change. [2]

Selon Ulmann, l'individualité technique de la machine analogique se caractérise par une variabilité de la structure, une mobilité ou amovibilité de ses éléments, une « reconfigurabilité » qui n'est autre que la marge de manœuvre de l'opérateur pour programmer sa simulation. L'ordinateur analogique est donc un objet technique spécial qui n'acquiert la plénitude de son individualité (technique aussi bien que symbolique) que lorsqu'il est programmé. Autrement dit, l'ordinateur analogique sans programme n'est pas encore un individu technique ; programmer un ordinateur analogique, c'est l'individuer ; c'est, au sens de Simondon, en perpétuer, en continuer l'invention.

1. Simondon, *ILNFI*, *op. cit.*, p. 57.
2. Ulmann, *Analog Computing*, *op. cit.*, p. 2, je traduis.

Réciproquement, le développement de la programmation numérique est conditionné par une stabilité de la machine, c'est-à-dire par une individualité technique achevée. C'est du moins ce que corrobore l'histoire de l'EDSAC [1] tel que la rapporte Ramunni : cette machine de 1949 n'a présenté aucune innovation architecturale majeure par rapport à l'EDVAC [2], son prédécesseur. Son invention, qui n'impliquait pas une réécriture complète des programmes, dégage alors la possibilité d'une autonomisation de l'activité de programmation. [3] De façon générale, comme l'écrit Ramunni, « le développement du logiciel a été dû à la volonté des constructeurs de réduire le plus possible les différences techniques entre les machines de la première et de la deuxième génération » [4]. Le développement du *software* dépend ainsi d'une uniformisation et d'une standardisation du *hardware*, c'est-à-dire de l'établissement d'une forme individualisée conventionnelle de ce dernier. La conception technique de l'ordinateur numérique, bien qu'elle puisse mettre en œuvre une certaine modularité ou extensibilité, est ainsi toujours finalisée. Autrement dit, l'ordinateur numérique – avec ou sans son programme – est un objet technique individué : s'il n'est peut-être pas immédiatement utilisable, il est toujours déjà fonctionnel.

À ce stade on pourrait conclure que l'ordinateur analogique et l'ordinateur numérique se distinguent à partir de leur relation à l'écriture : le premier serait une écriture ou une mémoire alors que le second emploierait des écritures, posséderait une mémoire. Mais cette distinction est incomplète car, en réalité, l'ordinateur numérique – comme l'analogique lorsqu'il est programmé – est aussi lui-même une écriture puisque, jusqu'à preuve du contraire, l'ordinateur numérique est bien, selon le mot de Simondon, du « geste humain déposé, fixé, devenu stéréotypie et pouvoir de recommencement » [5]. Toutefois, cette écriture qu'il est, en tant qu'il est un objet technique inventé, n'existe pas tout à fait comme existe l'écriture imitée. La mémoire que possède l'ordinateur numérique n'est pas exactement opératoire et efficace comme est opératoire et efficace la mémoire qu'il est.

Avant d'éclairer la différence, il faut noter que l'on peut dès lors rapprocher l'ordinateur numérique non programmé, c'est-à-dire dépourvu de mémoire, des ordinateurs analogiques programmés : tous les deux, en effet, sont des réalités techniques inventées, actuellement fonctionnelles et, en ce sens, des écritures. La

1. *Electronic Delay Storage Automatic Calculator.*
2. *Electronic Discrete Variable Automatic Computer.*
3. Ramunni, *La physique du calcul*, *op. cit.*, p. 99.
4. *Ibid.*, p. 149.
5. Simondon, *MEOT*, *op. cit.*, p. 191.

description de l'ordinateur numérique fait donc recours à deux écritures : une première qui est l'ordinateur numérique lui-même, dans sa structure intime dont j'ai montré qu'elle était nécessairement individuée et fonctionnelle – il partage par là des qualités avec l'ordinateur analogique programmé – ; et une seconde écriture (ou une seconde mémoire), qui est celle dans laquelle on rédige le programme pour mettre en correspondance la machine et le problème. Du côté analogique, l'écriture qu'est la structuration des dynamismes au fil de la programmation est bien la seule et unique écriture mise en œuvre. Certes, on pourrait considérer l'allure des panneaux de câblage ou l'aspect des différents organes d'affichage comme des formes d'écriture secondaire ou des interfaces comme celles des logiciels numériques. Cependant dans tous ces cas, si l'ordinateur analogique a une signification c'est toujours et uniquement dans la mesure où il représente la nature en la mettant en œuvre. Par exemple, l'ordinateur analogique n'est pas muni d'un « écran » mais d'un oscilloscope, dont la technicité – en l'occurrence : la déviation d'un spot d'électrons le long d'un tube cathodique et la production de photons visibles par un humain sur une vitre phosphorescente – participe intrinsèquement à la compréhension intuitive de sa fonction symbolique dans la mesure où le mouvement des électrons, et donc des photons visibles, est isodynamique à la chose calculée ou à la propriété de la chose mesurée. Réciproquement, les écrans LED que nous connaissons majoritairement aujourd'hui comme affichages polyvalents des ordinateurs numériques peuvent afficher littéralement n'importe quoi. La connaissance de la nature n'est d'aucun secours pour interpréter de telles interfaces. Pour produire de la signification, la programmation de logiciels utilisant de tels écrans mime des traditions humaines, des habitudes ; elle reprend par exemple l'apparence du papier dans une machine à écrire, celle d'une table de mixage audio ou de montage vidéo, elle prend l'aspect d'un calendrier, d'un bureau, d'un magasin. Elle peut même, d'ailleurs, faire semblant d'être un oscilloscope. L'historien des sciences George Basalla a proposé un terme pour désigner l'adjonction d'éléments structurels conçus de façon mimétique, techniquement inutiles dans le nouvel objet mais essentiels à la réalité imitée : le « skeuomorphisme » [1].

Voilà où se situe la différence entre l'écriture qu'est l'ordinateur numérique et les écritures qu'il manipule : ces dernières sont autant d'importations issues de la culture, d'imitations employées pour donner une impression de familiarité à l'opération de mise en relation du problème et de la technique. L'ordinateur

1. George BASALLA, *The Evolution of Technology*, Cambridge University Press, 1988, p. 107.

analogique, quant à lui, est dépourvu de skeuomorphes ; ses entrées et ses sorties sont « simplement » et immédiatement techniques, c'est pourquoi il peut être dit représentation de la physique ou symbolisation de la nature.

5. Universalisme des injonctions et utilitarisme

Les précédentes considérations permettent de mettre en question une perspective communément admise au sujet de l'informatique, selon laquelle – pour reprendre les mots de Moreau – une « machine numérique peut être universelle, [alors qu']une machine analogique ne peut l'être » [1]. Cette universalité-là n'est toutefois conçue que dans le sillage de l'universalité telle que Turing l'a définie, en 1936 [2]. Cet universalisme logicien se caractérise en particulier par une universalité des injonctions ou, pour parler comme von Neumann, des « structures d'action » [3]. Qu'est-ce à dire ? Simplement : que les machines numériques sont des machines auxquelles on doit pouvoir tout demander ; que c'est en ce sens qu'elles sont universelles. Se référant à l'expérience de pensée dite « de la chambre chinoise », proposée par le philosophe John Searle, Marcelo Dascal écrit à juste titre :

> En vérité, tout ce qui nous intéresse dans la communication avec l'ordinateur [numérique], c'est qu'il exécute ce que nous souhaitons, qu'il fonctionne conformément à nos instructions, nos demandes et nos paroles. Peu nous importe de savoir si derrière le rideau se cachent des robots stupides, qui déplacent des signes chinois sans leur accorder de signification ou des courants électriques qui font passer des signaux d'un endroit à l'autre. La seule chose qui compte est que le résultat soit « juste ». [4]

La « chambre chinoise » est – pour ainsi dire – une excellente analogie du numérique. On y voit assez bien en quoi la réponse de l'informatique numérique n'a pas besoin d'être isodynamique au problème à résoudre, pourvu qu'elle corrobore les résultats des opérations effectuées traditionnellement pour le résoudre de façon convaincante. Cette imitation remplit l'espace qui sépare le sens des injonctions logiques de celui de la technicité de la machine qui les exécute par des éléments imités toujours plus nombreux – imitant même parfois leur propres résultats, à l'instar de l'aspect de certains jeux vidéos qui reproduisent des géométries et des

1. Moreau, *Ainsi naquit l'informatique*, *op. cit.*, p. 20.

2. « Théorie des nombres calculables, suivie d'une application au problème de la décision » (éd. orig. 1937) in Alan Turing et Jean-Yves Girard, *La machine de Turing*, trad. par Julien Basch et Patrice Blanchard, Paris, Seuil, 1995, p. 66.

3. von Neumann, *L'ordinateur et le cerveau*, *op. cit.*, p. 21.

4. Marcelo Dascal, « Culture numérique. Enjeux pragmatiques et philosophiques », in *Diogène* 211 (2005), p. 40.

textures de très faible résolutions, ou de l'intérêt esthétique que certains artistes manifestent pour les *glitches* –, tant et si bien qu'on se demande finalement – et je soumets ici une vraie question ouverte – si l'informatique numérique a un jour été capable de symboliser autrement qu'en imitant ou en échouant.

Ceci dit, si cette imitation fait problème, c'est parce que – selon une logique que Simondon a parfaitement analysée dans son étude des ensembles techniques – elle cause un effet en retour sur les utilisateurs. Autrement dit, l'aliénation de la technique par l'humain engendre une aliénation technique de la culture. Dascal aborde ce problème du point de vue linguistique ; il critique ainsi le manque d'outils informatiques capables d'intégrer la dimension pragmatique de la communication linguistique réelle. Ainsi écrit-il à juste titre :

> Le principal problème des programmes qui traitent des textes en langues naturelles n'est pas l'absence de connaissances syntaxiques ou sémantiques (bien que, dans ce dernier domaine, il reste encore beaucoup à faire) mais l'absence presque totale d'outils pragmatiques capables de guider leur activité. Cela conduit, entre autres, les programmes existants à nous limiter à des formats fixes. Prenez, par exemple, les boîtes de dialogue des logiciels courants. Ces logiciels (ou plus exactement leurs concepteurs) veulent que notre dialogue avec eux reste purement sémantique, sans recours à la pragmatique. [1]

L'explication de Dascal – malgré son ancrage dans une perspective d'utilisateur – souligne bien cette aliénation en retour. Par extension, l'universalisme de l'injonction – en tant que limitation du langage à son pouvoir de contrôle et de délégation – prolonge et creuse la fracture sociale entre inventeurs et utilisateurs, fracture dans laquelle il trouve son origine.

Cette nécessité de pouvoir tout réclamer à l'ordinateur caractérise selon moi la fonction fondamentale du numérique. Elle s'oppose à un universalisme physicien qui se fonde sur l'ensemble des dynamismes physiques techniquement « implémentables ». Du point de vue physicien, l'ensemble des opérations possibles par une machine universelle au sens de Turing n'est alors qu'un sous-ensemble des opérations possibles dans l'univers. En effet, l'universalité logicienne de Turing ne s'attribue qu'à une machine à calculer capable d'émuler n'importe quelle autre machine à calculer. Par conséquent, elle ne s'attribue qu'à une machine et, en ce sens, à une réalité conçue en discontinuité radicale avec la réalité naturelle. Par là s'explique que du point de vue des injonctions – du point de vue logicien –, toute machine analogique ne puisse être que spécifique. Ainsi, lorsque Moreau affirme

1. *Ibid.*, p. 41.

qu'« on ne peut réaliser un montage analogique qui soit suffisamment général pour calculer n'importe quelle fonction » [1], il ne fait ni plus ni moins que reprocher à l'ordinateur analogique de ne pas être en mesure de répondre à n'importe quel ordre logique.

Cependant, les machines analogiques peuvent tout de même être dites universelles dans la mesure où elles héritent leurs fonctionnements des dynamismes de l'univers ; elles sont dynamiquement universelles car isodynamiques – ou dynamiquement analogues – à l'univers. Il s'en suit que, du point de vue analogique, une critique peut être formulée en retour, à l'endroit de l'universalité de Turing : cet universalisme logicien – au moins jusqu'à l'émergence de la théorie de la complexité – n'a que très peu, voire pas du tout tenu compte de son coût physique et matériel. Par là s'explique que la discontinuité du calculant et du calculé – inhérente à la perspective logicienne qui préside à la conception de la machine – ait notamment pour corrélat un manque de conscience écologique, comme le soulignent Aldo Frigerio, Alessandro Giordani et Luca Mari, dans un intéressant article [2] sur la distinction entre analogique et numérique.

On pourrait croire alors que la référence psychosomatique et l'universalisme des injonctions « à la Turing » allaient naturellement et rapidement mener l'informatique numérique à l'instauration des langages de programmation ; or il n'en est rien. Le récit de Grace Murray Hopper, l'inventrice du premier compilateur informatique, illustre le climat de résistance dans lequel les premières commodités de programmation sont apparues :

> Durant les années d'émergence des premiers langages de programmation, nous entendions très souvent dire que la seule possibilité de programmer un ordinateur était de le faire en octal [c'est-à-dire uniquement avec les nombres auxquels correspondent les opérations élémentaires]. [3]

De même, en 1953, l'idée a été émise d'un logiciel spécifique qui devait permettre aux programmes mathématiques d'être écrits en notation mathématique et aux programmes liés à de la gestion d'information d'être écrits en anglais ; or cette

1. Moreau, *Ainsi naquit l'informatique*, *op. cit.*, p. 20.

2. Frigerio, Giordani et Mari pensent la différence entre analogique et numérique à partir de la nature de l'encodage conçu comme une fonction ; selon leur critère, l'analogique correspond à un « isomorphisme intensif », c'est-à-dire à une règle de conversion, et le numérique correspond à une « description extensive », c'est-à-dire à une liste de correspondances. Aldo Frigerio, Alessandro Giordani et Luca Mari, « On Representing Information : A Characterization of the Analog/Digital Distinction », in *Dialectica* 67.4 (2013), p. 455-483.

3. Grace Murray Hopper, « Keynote address », in *History of programming languages. Proceedings of the ACM SIGPLAN History of Programming Languages (HOPL) conference (juin 1978)*, sous la dir. de Richard L. Wexelblat, t. 1, Academic Press, 1981, p. 7, je traduis.

idée a été fort mal reçue. Hopper explique à ce sujet, non sans une certaine ironie, que les décideurs de jadis étaient absolument convaincus que « les ordinateurs ne pourraient jamais comprendre des mots en anglais » [1].

Tout porte à croire que si l'idée de commodité de programmation a jadis paru si excentrique, c'est parce qu'il a d'abord semblé qu'elle contredisait l'exigence d'universalité des injonctions. Cette proposition s'appuie en premier lieu sur le récit de Hopper, qui montre bien que les programmeuses d'alors – qui, elles, avaient les mains dans le cambouis – avaient identifié très tôt l'inutilité d'un formalisme absolument universel et général pour informer la machine : « pour tous les problèmes que nous avons résolus, nous avons constaté que nous n'avions pas besoin d'une généralité complète » [2]. Et en effet, l'instauration des premiers langages a très fortement cadré les usages possibles de la machine numérique et donc spécialisé cette dernière : le Fortran, par exemple, a fait de l'ordinateur numérique une machine scientifique et mathématique, le Cobol en a fait une machine d'administration et le Lisp une virtuose de la structuration de données. Ainsi, la relation entre langage informatique et langage humain ne pouvait être que tendue : le langage humain reflétant, comme l'explique Dascal, une situation relationnelle impliquant « plusieurs registres activés en même temps », résolue par le choix du « cadre interprétatif qui convient en fonction de circonstances changeantes et selon des milliers de signes contextuels (comme le ton, le regard ou les événements dans l'entourage) que nous percevons en sus du langage lui-même » [3] ; le langage informatique reflétant une situation de travail spécialisé, normalisé et délégué. Il a certainement paru évident, aux investisseurs, qu'un langage informatique ne rivaliserait jamais avec l'universalité et la puissance des langages humains ; pourquoi alors les développer ? Autant continuer à travailler en octal ! Et cette idée n'est pas complètement saugrenue si l'on se rappelle que les recherches d'un « langage universel » ou d'un « compilateur de compilateurs », pour reprendre le mot de Ramunni [4], ont rencontré de nombreux échecs.

N'empêche que, ces machines, il fallait bien les utiliser et les rentabiliser sans devoir réinventer la roue à chaque problème. Le projet universaliste de formalisme général des injonctions a donc petit à petit été abandonné au profit du développe-

1. *Ibid.*, p. 16.
2. *Ibid.*, p. 8.
3. Dascal, « Culture numérique », *op. cit.*, p. 43. Dascal joue ici « notre remarquable capacité pragmatique » contre « la compétence sémantique que Searle exigeait des somnambules qui tournaient en rond dans la chambre chinoise ».
4. Ramunni, *La physique du calcul*, *op. cit.*, p. 165.

ment du logiciel, en réponse à une exigence toujours plus explicitement utilitariste. Ce développement révélait alors le potentiel économique d'une spécialisation de la machine numérique. Selon Ramunni, sa mission était de « masquer pour le programmeur ou l'utilisateur médiocrement averti les complications structurelles de la machine, la nature complexe de son architecture. » [1] On découvrait ainsi qu'une machine à laquelle on peut tout demander – aussi fonctionnelle soit-elle – ne devient utile que lorsqu'on lui demande quelque chose de particulier. Autrement dit, les stéréotypes des « automates parfaits au service d'une humanité paresseuse et comblée » [2] – pour reprendre un mot de Simondon – dévoilaient leur essence mythologique.

Bref : l'universalisme logicien des injonctions et l'utilitarisme sont fondamentalement liés. Or – comme l'explique Simondon –, la relation utilitaire à la technique est nécessairement aliénante. Ceci s'explique notamment par le fait que l'émission d'un ordre en vue d'un résultat utile ne requiert pas la connaissance en détail des savoir-faire humains ou des fonctionnements en jeu. Universalisme logicien et utilitarisme ont donc aussi en commun une connaissance des techniques potentiellement très lacunaire et abstraite ; la partie technique du problème étant spontanément déléguée à qui de droit et asservie à des exigences abstraites. Cette dynamique de délégation creuse la distinction entre la conception et la réalisation dans le processus d'invention et rend possible un désintérêt toujours croissant pour les répercussions concrètes et réelles, dans le milieu, des opérations et fonctionnements actuels. (On retrouve ici l'absence de conscience écologique propre à l'informatique numérique.) L'aspect technique du travail est confiné dans une « boîte noire » et, aux yeux de celui qui donne l'ordre, celui qui produit est « libre d'obéir », selon le mot bien senti de Johann Chapoutot [3]. En un mot, l'opération à effectuer ne doit pas dévoiler sa technicité. Sa forme idéale est atteinte lorsqu'elle s'exécute comme par magie, c'est-à-dire quand l'ordre logique initial s'énonce comme une incantation ou s'effectue comme un geste.

Comment cette recherche de magie fonctionne-t-elle ? Quelle réalité technique se cache derrière l'universalisme des injonctions ? Je propose de répondre à cette question technologique par une description de la façon dont les opérations de substitution de signes sont déléguées. La première étape consiste à re-

1. *Ibid.*, p. 157.
2. Simondon, *MEOT*, *op. cit.*, p. 16.
3. Johann Chapoutot, *Libres d'obéir. Le management, du nazisme à aujourd'hui*, Paris, Gallimard, 2020.

marquer que certaines séquences de codes machine s'enchaînent de façon très redondante et répétitive. Du moment qu'une séquence est répétée au moins une fois, on en infère un type. Le principe de substitution consiste alors simplement à écrire une fois la séquence, à la placer à une adresse précise dans la mémoire, puis à contrôler l'exécution du code de façon à passer par ce segment autant de fois que nécessaire. Pour se faciliter la tâche, on nomme ces emplacements selon ce que le code effectue ; ce qui permet, dans un second temps, d'y faire des appels, c'est-à-dire d'y rediriger l'exécution. C'est à ce moment-là que, pour parler comme Aristote, la cause efficiente est identifiée à la cause finale : si ces mots d'appel commodes provoquent bien une réalité technique – en l'occurrence une succession de fonctionnements élémentaires du processeur –, d'un point de vue linguistique ils ne désignent cependant que des usages possibles ou des effets de ces fonctionnements. La différence est de perspective : les fonctionnements opératoires et la technicité élémentaire sont en principe toujours compris de façon analogique, c'est-à-dire en regard des dynamismes physiques de la nature – à l'instar par exemple des voltages et des seuils de tensions qui, s'ils encodent des bits, n'en demeurent pas moins des phénomènes physiques – ; les usages possibles et les effets, cependant, sont supputés ou évalués dans une perspective nécessairement restreinte vis-à-vis de la précédente, ils sont circonscrits à certains champs, déterminés par certaines habitudes et stéréotypies qu'ils ne peuvent que reproduire. Cette opposition entre fonctionnements et usages – ou, si l'on préfère, entre causalité et finalité – consacre la distinction conventionnelle entre programme « principal » et programmes « secondaires » : malgré une identité technologique évidente entre les deux sortes de programmes, on convient de se référer aux programmes secondaires comme éléments d'une « bibliothèque » au service d'un programme principal ; ce qui montre une nouvelle fois, en passant, comment l'informatique numérique se fonde sur l'instauration de relations de service.

De ces commodités et de leur systématisation émergent les langages de programmation informatique, comme systèmes d'appels et de contrôle de l'exécution, puis comme systèmes de systèmes d'appels, puis comme systèmes de systèmes de systèmes d'appels ; etc. Concrètement, pour chaque langage de programmation et pour chaque type de machine, un « compilateur » doit être programmé, dont la tâche est de transformer une écriture commode en code exécutable par la machine. Autrement dit, le compilateur porte la relation entre causalité et finalité : il relie une culture de surface (car imitant des habitudes) à une technicité des profondeurs, désormais activement protégée par la miniaturisation et même parfois par

une complexification exagérée dont le but est de compromettre la lisibilité du circuit. De cette relation entre surface et profondeurs, Simondon aurait certainement affirmé qu'elle est artificielle et abstraite, le compilateur ayant, pour les habitudes humaines, un rôle analogue à celui de la serre chaude pour la plante modifiée [1] : celui d'en contrôler le milieu afin d'en contrôler l'existence. Le nombre de compilateurs à développer et à maintenir à jour est égal au nombre de langages existants multiplié par le nombre d'architectures matérielles existantes sur lesquelles les programmes doivent pouvoir s'exécuter. (En informatique commerciale, on parle parfois d'« écosystème ».) Si l'on admet que ces réalités techniques sont censées, au départ, être des commodités, alors, tout bien considéré – c'est-à-dire globalement –, le gain est discutable.

Les langages de programmation constituent donc un moyen de spécialiser, de contextualiser, de rendre utile une machine informatique numérique qui, sans eux, demeurerait trop universelle, trop abstraite, trop difficile à prendre en main, trop éloignée des besoins et des problèmes réels des gens. Le langage *Plankalkül* de Konrad Zuse, par exemple, génétiquement conçu sans relation à une machine – comme l'expliquent Knuth et Pardo [2] –, est longtemps demeuré une espèce de vue de l'esprit. Carla Petrocelli explique notamment qu'il était alors absolument impossible de compiler un programme écrit avec *Plankalkül* [3] sur les ordinateurs de l'époque.

On voit bien par là que ce qu'on appelle « langage de programmation » n'est, au fond, ni une écriture ni un formalisme logique. Il naît plutôt comme l'institution d'une relation entre une machine et un problème spécifique, au prix d'un travail considérable de mise en œuvre des compilateurs. Ce travail contribue à soutenir très artificiellement une continuité précaire entre des contextes culturels et la technicité de la machine numérique, technicité dont la valeur symbolique intrinsèque – c'est-à-dire considérée en deçà de son pouvoir caractéristique de convoyer par imitation la valeur symbolique de choses préexistantes – reste à mon sens indéterminée. Tout porte à croire, en effet, que l'informatique numérique, au prix d'efforts

1. Simondon, *MEOT*, *op. cit.*, p. 57.

2. Knuth et Pardo, « The Early Development of Programming Languages », *op. cit.*, p. 208, les auteurs de cette préhistoire des langages de programmation distinguent entre « what was possible to implement » et « what was possible to write » ; le projet de Zuse étant limité à la seconde problématique.

3. Carla Petrocelli, « Konrad Zuse and his *Plankalkül* : the hope to emerge from the sleep of *sleeping beauty* », in *International Journal of Humanities and Arts Computing* 13.1-2 (2019), p. 258.

considérables et de répercussions imprévisibles, cherche à reproduire en l'imitant la continuité originaire qui est au principe de l'informatique analogique.

Conclusion

Sous quel mode existe donc le numérique ? En fin de compte, cela dépend de la définition que l'on accepte de la culture. Si, par « culture », on désigne des objets achevés, des pratiques documentées, des normes institutionnalisées, des traditions respectées, bref : si par culture on entend une matière, alors le numérique est un intéressant moyen de reproduction. Il facilite la besogne inhérente à certains travaux et donne – à qui peut en profiter – une agréable impression de faire des économies tout en allant beaucoup plus vite et en étant plus précis. Il nous permet de nous divertir, de continuer d'écrire, de faire de la musique, des images, des mathématiques et des affaires sans même devoir réfléchir au sens de ces formes d'expression et de production héritées du passé. Il reproduit pour nous une familiarité peut-être rassurante à certains égards, mais il reproduit aussi en même temps la division et la délégation du travail, ainsi que la société aliénée qui va avec.

Si, par contre on désigne par « culture » un ensemble de processus d'information en droit constitutifs de la relation entre subjectivité et objectivité ou, pour le dire autrement, si l'on conçoit la culture comme une dynamique de régulation (ou d'auto-régulation) ou – selon Simondon – comme une réalité transindividuelle, alors le numérique est un problème, car il désigne le recours à une machine qui brutalise l'idée de culture en la rapportant à un ensemble de références patrimoniales statiques ; c'est-à-dire à une matière. Il ne suffit pas, comme dirait Yves Citton, d'« apprendre à utiliser » [1] ; il faut expliquer comment le numérique – au détriment d'une idée de culture comme condition de toute expérience possible – récapitule une certaine représentation ou une certaine expérience de la culture : celle de ses inventeurs. Seules les machines analogiques, parce qu'elles sollicitent nécessairement l'invention et l'interprétation d'un opérateur, sont en mesure d'être des machines culturelles, c'est-à-dire connaissables, modulatrices, régulatrices, informationnelles et techniquement concrètes. Le numérique, au contraire, ne se concrétise pas. D'années en années, il s'artificialise toujours davantage.

1. Yves Citton, *Médiarchie*, Paris, Seuil, 2017, p. 365.

Ainsi, aux yeux du technologue, le numérique apparaît comme une méthode de délégation de l'interprétation ; il a pour corrélat nécessaire – et d'une nécessité apodictique – le désamorçage systématique de toute possibilité de thématiser la culture comme horizon transcendantal. C'est peut-être pour cette raison que, aux yeux de Simondon – éminent penseur des relations entre analogie et information, entre culture et technicité –, cette réalité que nous voudrions nommer « numérique » n'existe tout simplement pas.