

La face cachée des nombres

Élise Janvresse, Thierry de la Rue
LMRS - Université de Rouen - CNRS

18 avril 2007

Il existe des nombres réels dont l'ordinateur ne peut s'approcher que par en-dessous ! L'objet de cette note est d'en présenter un, obtenu comme la limite croissante d'une suite produite par un programme informatique. Nous montrons qu'aucun programme ne peut produire une suite décroissante convergeant vers ce réel ! Ce sera aussi l'occasion d'aborder la notion de calculabilité, introduite par les mathématiciens avant même que les ordinateurs ne soient inventés.

Introduction

Comment un ordinateur peut-il produire un nombre réel, comme π ? Il ne peut bien sûr pas donner la valeur exacte en un temps fini car celle-ci comporte une infinité de décimales. Mais ce que l'on peut espérer, c'est que l'ordinateur imprime des approximations de plus en plus précises du nombre π , c'est-à-dire une suite de nombres décimaux convergeant vers π . Il existe d'ailleurs de nombreux algorithmes capables de le faire. Peut-on ainsi produire n'importe quel nombre réel par ordinateur ?

Qu'est-ce qu'un programme informatique ?

Un programme informatique n'est rien d'autre qu'un nombre entier. En effet, toutes les instructions traitées par un ordinateur sont traduites en langage binaire par des suites finies de 0 et de 1. Donc tout programme peut être vu comme un nombre entier écrit en base 2. Inversement, on peut considérer tout entier comme un programme : écrivons-le en base 2, et donnons la suite de 0 et de 1 à l'ordinateur : s'il est capable de l'interpréter comme une suite d'instructions, tout va bien ; sinon convenons que le programme associé ne fait simplement rien du tout.

Comme il n'y a qu'une quantité dénombrable de programmes possibles, et une quantité non dénombrable de nombres réels, la majorité d'entre eux restent inaccessibles par un programme informatique.

Notre but est d'explicitier un algorithme produisant un nombre réel accessible par en-dessous, mais pas par au-dessus. Plus précisément, nous allons construire un programme imprimant une suite croissante qui converge vers une limite telle qu'aucun programme n'est capable d'imprimer une suite décroissante convergeant vers cette dernière.

Construction

Il est facile de lister un par un tous les programmes possibles : cela revient à lister successivement tous les entiers $1, 2, 3, \dots$. Dans la suite, on notera par le même symbole n le nombre entier n et le programme informatique codé par le développement binaire de n .

Le programme que nous allons construire va faire tourner à tour de rôle tous les programmes possibles : il fait d'abord tourner le programme 1 pendant une seconde ; il passe alors au programme 2 pendant une seconde, puis reprend l'exécution du programme 1 pour encore 1 seconde ; il fait ensuite tourner le programme 3, puis retourne au programme 2, puis au programme 1 ; ensuite il débute le programme 4, puis revient au programme 3, puis au programme 2, puis au programme 1 ; etc. En fait, c'est un peu plus compliqué : sous certaines conditions, il arrivera que l'on décide d'éliminer définitivement certains programmes. On notera $k(n)$ le programme qui tourne à la n -ième étape : $(k(n))_{n \geq 1}$ est une sous-suite de $(1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, \dots)$.

Notre programme va imprimer une suite croissante $(a_n)_{n \geq 0}$ de décimaux, de la forme : $a_0 = 0$ et $a_n = a_{n-1} + x_n$ pour tout $n \geq 1$, avec

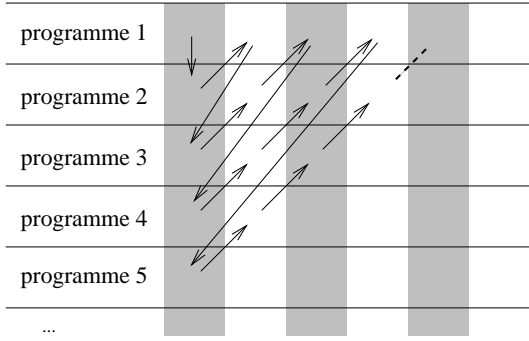


FIG. 1 – Ordre d'exécution des programmes

$x_n \in \{2^{-n}, 2^{-k(n)}\}$, où $k(n)$ est le programme qui tourne à l'étape n .

Voyons comment, à l'étape n , est choisi le nombre x_n . Pour connaître sa valeur, on regarde le dernier nombre imprimé par le programme $k(n)$. Si un tel nombre y existe et que $a_{n-1} \leq y \leq a_{n-1} + 2^{-k(n)}$, alors on pose $x_n = 2^{-k(n)}$ et on ne fera plus jamais tourner le programme $k(n)$. Sinon, on pose $x_n = 2^{-n}$.

Notons que la suite $(a_n)_{n \geq 0}$ est croissante et bornée par 2. En effet,

$$a_n \leq a_0 + \sum_{k \geq 1} 2^{-k} + \sum_{j=1}^n 2^{-j} \leq 2.$$

Cette suite converge donc vers une limite ℓ .

Preuve par l'absurde

Il reste à s'assurer qu'aucun programme n'est capable d'imprimer une suite décroissante convergente vers ℓ .

Supposons qu'un tel programme existe. Il correspond à un entier k . Notons $(b_i)_{i \geq 0}$ la suite qu'il imprime.

Comme $(a_n)_{n \geq 0}$ et $(b_i)_{i \geq 0}$ convergent vers ℓ , il existe deux entiers I et N tels que pour tout $i \geq I$, $b_i - \ell \leq 2^{-k}/2$ et pour tout $n \geq N$, $\ell - a_n \leq 2^{-k}/2$.

Lors de la construction de $(a_n)_{n \geq 0}$, il y a deux cas possibles :

- ou bien le programme k est éliminé à une certaine étape n de la construction pendant laquelle on fait tourner le programme k . Cela signifie que le dernier nombre b_i imprimé par k est compris entre a_{n-1} et $a_{n-1} + 2^{-k}$. Mais alors $x_n = 2^{-k}$ et donc $b_i \leq a_n = a_{n-1} + 2^{-k}$. Comme $a_n < \ell$, cela contredit la décroissance de $(b_i)_{i \geq 0}$ vers ℓ .

- ou bien le programme k n'est jamais éliminé. Une fois que le programme k a imprimé b_I ,

intéressons nous au prochain temps $n \geq N + 1$ où l'on fait à nouveau tourner k : Comme à cet instant n le dernier nombre b_i imprimé par k est tel que $i \geq I$, on obtient que

$$b_i - a_{n-1} \leq (b_i - \ell) + (\ell - a_{n-1}) \leq 2^{-k}.$$

et donc k est éliminé à cette étape, ce qui contredit notre hypothèse.

On arrive de toute façon à une contradiction, donc le programme k n'existe pas.

À quoi ressemble ℓ ?

Puisque ℓ est la limite de la suite produite par notre programme, nous pourrions croire qu'il est possible de connaître ℓ avec une précision arbitraire. Or il n'en est rien ! Lors de l'exécution de notre programme, on ne peut jamais savoir si on est proche ou non de la limite ℓ . Plus précisément, il est impossible que pour tout $i \geq 1$, il existe une étape n_i à laquelle on sait que les i premières décimales de a_{n_i} sont celles de ℓ . En réalité, aucun programme n'est capable de calculer pour tout i les i premières décimales de ℓ . En effet, si c'était possible, on pourrait facilement écrire un programme produisant une suite décroissante de décimaux convergente vers ℓ . Il suffirait de reprendre les i premières décimales de ℓ auxquelles on ajouterait 10^{-i} . La suite $(b_i)_{i \geq 1}$ ainsi obtenue serait décroissante et convergente vers ℓ . Mais nous avons montré qu'une telle suite ne pouvait pas être produite par un programme informatique !

Définissable, mais pas calculable

Un nombre réel x est dit *calculable* s'il existe un programme capable d'imprimer la suite des décimales de x , autrement dit qui nous permet de connaître x avec une précision arbitraire. Par exemple, tout nombre algébrique est calculable, π et e le sont aussi. Nous venons de voir que le nombre ℓ est un exemple de nombre qui n'est pas *calculable*. Toutefois, ℓ fait partie du club très fermé des nombres *définissables*, c'est-à-dire ceux pour lesquels on peut énoncer dans le langage des mathématiques usuelles une propriété qui n'est vérifiée que par eux. Ils ne sont pas si nombreux : puisqu'il n'y a qu'une quantité dénombrable de propriétés possibles (elles doivent être exprimées par une suite finie de symboles), il n'en existe qu'une quantité dénombrable. Aussi, la grande majorité des

réels ne sont pas définissables. La notion de nombre définissable a été développée par Émile Borel dans son dernier livre « Les nombres inaccessibles »[1]. Le nombre ℓ est un exemple de nombre *définissable*, mais pas *calculable*. Il en existe bien d'autres ; le plus célèbre d'entre eux est la constante Ω de Chaitin [2] (que nous espérons présenter dans un prochain article).

La suite produite par notre programme est un exemple de *suite de Specker* (voir par exemple [3]), du nom de celui qui en 1949 a exhibé le premier exemple de suite croissante de nombres calculables dont la limite n'est pas calculable.

Remerciements

Nous remercions Steve Kalikow, de l'université de Memphis, qui nous a présenté cette construction.

Références

- [1] Émile Borel, *Les nombres inaccessibles.*, Gauthier-Villars, Paris, 1952.
- [2] Gregory Chaitin, *Meta math ! : The quest for omega*, 2005.
- [3] Klaus Weihrauch, *Computable analysis*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2000, An introduction.