Algorithme CORDIC pour calculer le logarithme

Nicole Bopp

Résumé : À l'occasion d'un enseignement en L1, j'ai découvert l'algorithme CORDIC utilisé par les calculatrices pour obtenir les valeurs de certaines fonctions, en particulier la fonction logarithme. Pour comprendre l'intérêt de cet algorithme et surtout sa précision, je montrerai comment les questions que l'on se pose naturellement mènent à un théorème dont la démonstration exige un peu d'analyse élémentaire. J'espère avoir ainsi convaincu certains étudiants, affichant haut et fort leur désintérêt pour les mathématiques que nous leur enseignons, que ces mathématiques leur seront indispensables même s'ils projettent de faire des études d'informatique.

Mots-clés : Algorithme - Calculatrice - CORDIC - Logarithme

Introduction

Un enseignement appelé MTU (Méthodologie du Travail Universitaire) a été introduit en L1 (nouvelle appellation de la première année d'université) à l'Université Louis Pasteur en 2005/2006. L'objectif des concepteurs de cet enseignement était essentiellement une formation à la recherche de documents. Dans la filière mathématique et informatique de la licence, ce sont les mathématiciens qui ont pris en charge cet enseignement et leur objectif était plus précisément d'apprendre aux étudiants à lire un texte mathématique. C'est pourquoi, avec quelques collègues en charge de cet enseignement, nous avons choisi d'approfondir un thème mathématique déjà familier aux étudiants qui venaient de passer le baccalauréat.

J'ai choisi d'étudier avec mes étudiants les fonctions logarithmes. Les questions que nous avons approfondies ensemble étaient nombreuses : quelles en sont les différentes définitions ? pourquoi sont-elles équivalentes ? pourquoi les a-t-on inventées ? comment ont été calculées les premières tables de logarithmes ? comment calculer l'aire sous l'hyperbole ?

Pour répondre à ces questions les étudiants ont principalement recherché des informations sur internet (en tapant logarithme dans google) et recopié ce qu'ils avaient trouvé. Il a fallu faire un travail important pour comprendre les résultats trouvés et en écrire des démonstrations. J'étais assez contente car j'avais l'impression de leur avoir montré qu'on pouvait progresser en se posant de « bonnes » questions. Comme j'exprimais cette satisfaction à la fin d'une séance, l'un des étudiants me rétorque :« cela vous amuse peut-être de vous poser des questions mais nous pas du tout ». À quoi je lui réponds : « si vous voulez faire des études de mathématiques il faudra vous poser des questions ». Manque de chance, il ajoute « mais moi je veux faire des études d'informatique ». Voulant avoir le dernier mot je me lance : « Alors, vous voulez savoir comment votre calculatrice obtient les valeurs de la fonction logarithme. Eh bien, tapez CORDIC sur google et vous verrez ». Et c'est ainsi que j'ai vu arriver trois jours plus tard cet étudiant dans mon bureau avec un algorithme qu'il avait implanté sur sa calculatrice, qui tournait bien mais qu'il voulait

mieux comprendre.

Voilà ce qu'il avait trouvé sur le site Gérard Evrard ([1]).

Logarithme par l'algorithme CORDIC

```
Input X
 0->I
 1->Y
 Lbl 1
1+10^-I->Z
 Lbl 3
If XZ>10
 Goto 2
 XZ->X
Y-Log Z->Y
 Goto 3
 Lbl 2
Is(I,10)
 Goto 1
 Disp Y
```

Mode d'emploi : *Lancer le programme - Taper un nombre compris entre 1 et 10 - Le logarithme du nombre est affiché.*

Mathématiques à l'œuvre : *On utilise l'algorithme CORDIC. Il ne faut pas croire que la fonction Log, présente dans le programme, soit utilisée de façon dérobée, elle évite seulement le recours à une table. Pour être plus convaincant, on pourrait adapter le programme pour qu'il utilise xStat et yStat, sans aucune élévation à la puissance ni Log. Le principe est de multiplier le nombre de départ par des nombres de la forme Pi=(1+10^-i) afin d'atteindre 10 sans jamais le dépasser. On commence naturellement par la valeur de i la plus basse possible, soit 0 (ligne 2). Lorsque 10 est atteint, on a : 10=X*Produit(Pi) soit 1=log(X) + Somme(log(Pi)). ou log(X)=1-Somme(log(Pi)). Or une table précalculée donne les log(Pi), il suffit donc de les sommer dans la boucle à chaque fois qu'on multiplie par le Pi correspondant.*

En fait ce qui intriguait au premier chef cet étudiant était le sens de la commande `Is(I,10)`. Une rapide inspection du mode d'emploi de la calculatrice nous a appris que ce n'était rien d'autre qu'une boucle « `Rajouter 1 à la variable I tant que I est plus petit que 10` ». Ce qui l'étonnait plus c'est qu'il obtenait systématiquement une valeur légèrement supérieure à celle que donne la calculatrice quand on utilise la touche ln. Et, bien sûr, il voulait savoir comment cela fonctionnait.

Il se proposait d'exposer cet algorithme et ce qu'il en avait compris à la séance suivante car un exposé oral était exigé pour obtenir une note de contrôle continu. Voilà pourquoi j'ai tenté de comprendre les mathématiques qui justifient l'efficacité de cet algorithme et trouvé que sa simplicité était remarquable.

1. Le principe de l'algorithme pour le calcul de $\ln x$

L'algorithme CORDIC[1], inventé[2] en 1959 pour calculer les valeurs des fonctions trigonométriques, a permis aux premiers calculateurs de poche (HP 35 en 1972) d'être rapides malgré la taille réduite de leur mémoire. C'est en effet un algorithme économe en calculs comme nous le verrons plus loin. Nous allons le décrire pour le calcul du logarithme népérien d'un nombre décimal x strictement compris entre 1 et 10.

1.1. Pourquoi se restreindre aux nombres strictement compris entre 1 et 10 ?

Si on connaît $\ln 10$, on peut ramener le calcul de $\ln X$, où X est un nombre strictement positif, à celui d'un nombre compris entre 1 et 10 en utilisant le résultat élémentaire suivant.

Si X un nombre strictement positif, il existe un entier $n \in \mathbb{Z}$ tel que $1 \leqslant 10^n X < 10$.

La caractérisation fonctionnelle du logarithme implique que

$$\ln X = \ln(10^n X) - n \ln 10 \ .$$

Si $10^n X = 1$ c'est terminé car $\ln 1 = 0$. Sinon on est ramené au calcul du logarithme de $10^n X$, nombre compris strictement entre 1 et 10.

Dans la pratique on travaille avec des nombres décimaux car ce sont les seuls que la calculatrice utilise. Le nombre n dont l'existence est donnée par le lemme peut même s'obtenir sans peine à la main. Si on voulait regarder d'un peu plus près le fonctionnement de la calculatrice, il faudrait considérer le fait qu'elle utilise l'écriture en base 2 des nombres, mais nous allons en rester à ce qui est apparent, c'est-à-dire à leur expression en base 10.

On considère dans la suite un nombre décimal x appartenant à l'intervalle $]1, 10[$.

1.2. Traduire l'algorithme

L'algorithme décrit dans l'introduction utilise des labels, ce qui le rend difficile à comprendre. Un examen rapide permet de le réécrire en utilisant les boucles usuelles, ce qui facilitera son analyse.

```
On se donne x compris entre 1 et 10.
On affecte la valeur ln 10 à la variable y.

Pour i allant de 0 jusqu'à 10 (ou jusqu'à N), faire
      1 + 10^{-i} → z ;
      Tant que xz ⩽ 10 faire
            xz → x ;
            y − ln(z) → y ;

Le résultat cherché est y.
```

[1] CORDIC est l'acronyme de COordinate Rotation DIgital Computing.
[2] par Jacques Volder, ingénieur dans une firme aéronautique.

On remarque tout d'abord que la valeur 10 intervient de deux manières différentes. Elle apparaît d'une part dans la longueur de la boucle « `Pour` i `allant de 0 jusqu'à` 10 ». Ceci est arbitraire et pour plus de clarté nous avons remplacé 10 par N à cet endroit. D'autre part 10 intervient dans les valeurs affectées à la variable z, à savoir $1 + 10^{-i}$, et nous expliquerons plus loin pourquoi ce choix simplifie les calculs.

Ce qui frappe tout d'abord c'est que le programme utilise la fonction logarithme de certains nombres. Mais on voit rapidement qu'il utilise seulement les logarithmes des nombres de la forme $1 + 10^{-i}$. Il repose donc sur le préalable suivant.

1.3. Préalable

On utilise un petit nombre de valeurs de la fonction logarithme népérien[3] qui sont stockées dans la mémoire de la calculatrice. Plus précisément les valeurs que l'on pourra utiliser sont de la forme suivante :

z	$\ln z$
10	2.302585092994012
2	0.693147180559945
1.1	0.095310179804325
$\vdots$	$\vdots$
$1.000\cdots1$	

J' ai recopié les quelques valeurs données dans ce tableau dans l'article [4] de J. Laporte. Le nombre de décimales indiquées est choisi en fonction de la précision de la calculatrice et donc, de la précision que l'on espère obtenir pour le calcul de $\ln x$.

1.4. La boucle principale ou un algorithme simpliste

Il s'agit de la boucle décrite par la commande « `Tant que` ».

Fixons un entier $i \in \{0, \ldots, N\}$ et multiplions x par $(1 + 10^{-i})$ jusqu'à ce qu'on dépasse 10. Chaque fois qu'un produit par $(1 + 10^{-i})$ est effectué, soustrayons $\ln(1 + 10^{-i})$ à $\ln 10$. Que peut-on dire du nombre y obtenu ainsi ?

Pour plus de clarté, nous allons l'écrire en remplaçant 10^{-i} par un réel $\alpha > 0$. Comme la suite $\Big((1+\alpha)^n\Big)_{n\in\mathbb{N}}$ est strictement croissante et tend vers $+\infty$, on établit sans peine le

Lemme 1. —*Si x est un réel appartenant à l'intervalle $[1, 10]$, il existe un unique $n \in \mathbb{N}$ tel que*

$$x(1+\alpha)^n \leqslant 10 < x(1+\alpha)^{n+1} \ .$$

La croissance de la fonction logarithme implique que

$$\ln x + n\ln(1+\alpha) \leqslant \ln 10 < \ln x + (n+1)\ln(1+\alpha) \ ,$$

[3]On pourra trouver dans ([3], page 31) une rapide description des méthodes utilisées au XVII$^{\text{e}}$ siècle, en particulier par Briggs, pour établir les tables de logarithme.

ce qui démontre la

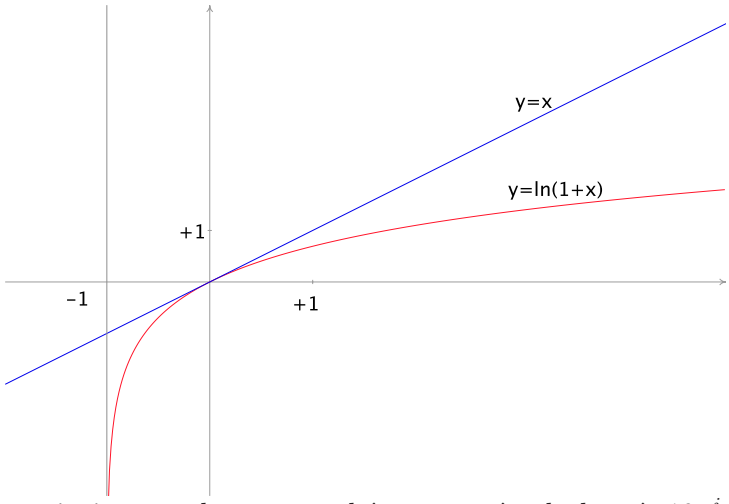
Proposition 2. *Il existe un entier $n \in \mathbb{N}$ tel que le nombre $y_n = \ln 10 - n\ln(1+\alpha)$ est une valeur approchée par excès de $\ln x$ à $\ln(1+\alpha)$ près. Plus précisément on a*

$$\ln x \in]y_n - \ln(1+\alpha), y_n] \ .$$

On peut dire plus grossièrement que y_n est une valeur approchée à α près en utilisant l'inégalité (1) qui est bien connue. Elle est illustrée par la figure[4] ci-dessous.

Pour tout $\alpha \geqslant -1$ on a

(1) $\qquad \ln(1+\alpha) \leqslant \alpha \ .$



En prenant $\alpha = 10^{-i}$, on obtient ainsi une valeur approchée par excès de $\ln x$ à 10^{-i} près, ce qui est assez satisfaisant et demande une seule boucle. On peut remarquer que les seuls produits à effectuer sont des produits par $(1+10^{-i})$ ce qui revient à faire un décalage de virgule puis une addition, qui sont deux opérations très simples. Les seules autres opérations à effectuer sont des soustractions. Ceci explique que ce choix pour α est particulièrement judicieux.

Mais l'algorithme CORDIC ne se contente pas de cette seule boucle car le nombre de pas à effectuer, qui est égal à $n+1$ où n est l'entier dont l'existence est énoncée à la proposition 2, peut être très grand.

Par exemple pour $x = 3$ et $i = 12$ le nombre de pas m est tel que

$$3(1+10^{-12})^m > 10 \quad \text{c'est-à-dire tel que} \quad m > \frac{\ln \frac{10}{3}}{\ln(1+10^{-12})} > 10^{12} \ ,$$

car $\ln \frac{10}{3} > 1$ et $\ln(1+10^{-12}) \leqslant 10^{-12}$.

1.5. Le principe de l'algorithme

Pour limiter le nombre de pas, on se rapproche le plus possible de 10 en multipliant x d'abord par des puissances de 2, puis par des puissances de 1,1, puis de 1,01 *etc.* Si on désire une précision de 10^{-N}, on termine en faisant le produit par des puissances de $(1+10^{-N})$. On espère que le nombre de pas de l'algorithme ainsi mis en place ne sera pas trop grand. C'est ce que nous allons démontrer dans le paragraphe suivant.

[4]Cette figure a été tracée à l'aide du logiciel libre Edugraph, distribué sous licence GPL et dû à Joël Amblard.

2. Majorations du nombre de pas

2.1. Etape 0

En prenant $\alpha = 10^0 = 1$, on déduit du lemme 1 l'existence d'un entier n_0 tel que

$$x(1+10^0)^{n_0} \leqslant 10 < x(1+10^0)^{n_0+1} \ .$$

Comme x est supérieur à 1, $2^4 x$ est supérieur à 16 et par conséquent n_0 est inférieur ou égal à 3. Posons

$$x_0 = x(1+10^0)^{n_0} \ ,$$

et remarquons que, par construction, on a $x_0 \leqslant 10 < 2x_0$.

2.2. Etape 1

En prenant $\alpha = 10^{-1}$, on déduit du lemme 1 l'existence d'un entier n_1 tel que

$$x_0(1+10^{-1})^{n_1} \leqslant 10 < x_0(1+10^{-1})^{n_1+1} \ .$$

Montrons que *l'entier n_1 est inférieur ou égal à 10.*

Par construction, le nombre x_0 vérifie l'inégalité

$$x_0 \leqslant 10 < 2x_0 \ .$$

Puisque $x_0(1+10^{-1})^{n_1}$ est inférieur ou égal à 10, on en déduit par l'absurde que

$$(1+10^{-1})^{n_1} < 2 \ ,$$

ce qui est équivalent à

$$n_1 < \frac{\ln 2}{\ln(1+\frac{1}{10})} \ .$$

Pour conclure, il reste à minorer $\ln(1+10^{-1})$. Pour cela on utilise l'inégalité (2) suivante.

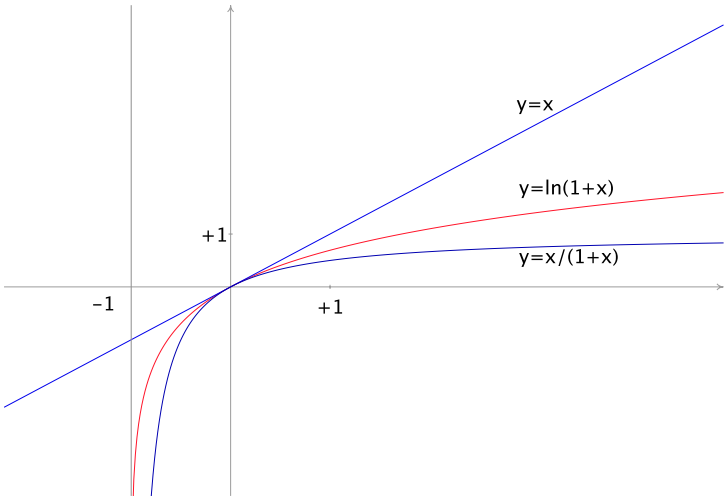
Pour tout $\alpha \geqslant -1$ on a

$$\frac{\alpha}{1+\alpha} \leqslant \ln(1+\alpha) \ . \tag{2}$$

Cette inégalité se démontre aisément en étudiant les variations de la fonction

$$\alpha \longmapsto \ln(1+\alpha) - \frac{\alpha}{1+\alpha} \ .$$

Elle est illustrée par la figure ci-contre.



En majorant $\ln 2$ par 1, on en conclut que $n_1 < \dfrac{1+\frac{1}{10}}{\frac{1}{10}} = 11$,

d'où le résultat.

2.3. Le théorème justifiant l'algorithme

En itérant le processus on comprend que l'algorithme CORDIC pour le calcul des valeurs du logarithme repose sur le théorème suivant.

Théorème 3. —*Soit x un réel strictement compris entre 1 et 10. Il existe alors une suite d'entiers $(n_i)_{i\in\mathbb{N}}$ tels que*

(i) $0 \leqslant n_i \leqslant 10$;

(ii) Pour tout entier positif N,

$$x\left(\prod_{i=0}^{N}(1+10^{-i})^{n_i}\right) \leqslant 10 < x\left(\prod_{i=0}^{N}(1+10^{-i})^{n_i}\right)(1+10^{-N}) \ .$$

Démonstration. —Le théorème se démontre par récurrence sur N en utilisant exclusivement les outils déjà mis en place dans la détermination de n_0 et n_1 (étapes 0 et 1). On suppose donc connus des entiers $n_0, n_1, \ldots, n_{N-1}$ vérifiant (i) et (ii) et on pose

$$x_{N-1} = x\prod_{i=0}^{N-1}(1+10^{-i})^{n_i} \ .$$

On a alors par (ii)

$$x_{N-1} \leqslant 10 < x_{N-1}(1+10^{-(N-1)}) \ .$$

On déduit, toujours du lemme 1, l'existence d'un entier n_N tel que

$$x_{N-1}(1+10^{-N})^{n_N} \leqslant 10 < x_{N-1}(1+10^{-N})^{n_N+1} \ .$$

Il reste donc à majorer n_N. Un raisonnement par l'absurde analogue à celui de l'étape 1 montre que

$$(1+10^{-N})^{n_N} < 1+10^{-(N-1)} \quad \text{ce qui est équivalent à} \quad n_N < \frac{\ln(1+10^{-(N-1)})}{\ln(1+10^{-N})} \ .$$

L'inégalité (1) implique que $\ln(1+10^{-(N-1)}) \leqslant 10\times 10^{-N}$,

et l'inégalité (2) que $\ln(1+10^{-N}) \geqslant \dfrac{10^{-N}}{1+10^{-N}}$.

On en conclut que

$$n_N < 10(1+10^{-N}) < 11 \ ,$$

ce qui démontre le théorème. □

2.4. Conséquences du théorème

Posons, en utilisant les notations du théorème, $y_N = \ln 10 - \sum_{i=0}^{N} n_i \ln(1+10^{-i})$. En utilisant à nouveau l'inégalité (1), on déduit de (ii) que

$$y_N - 10^{-N} < \ln x \leqslant y_N \ .$$

Nous avons donc démontré la

Proposition 4. —*Pour tout entier N, il existe une suite d'entiers positifs $n_i \leqslant 10$ tels que y_N soit une valeur approchée par excès de $\ln x$ à 10^{-N} près.*

Dans le cas où $N = 10$, y_{10} est exactement le nombre y que l'algorithme, décrit en **1.2.**, permet de déterminer. Les entiers $n_i + 1$ donnent le nombre de pas de chaque boucle `Tant que` et nous venons de démontrer qu'il y a au plus 11 pas. Si on cherche par exemple une valeur approchée de $\ln x$ à 10^{-12} près, il faudra au plus 130 pas pour obtenir le résultat où chaque pas consiste en une soustraction, un test de comparaison avec 10 et une multiplication par $(1 + 10^{-i})$, ce qui est très rapide.

Ceci explique le succès de cet algorithme car il a été facile de l'implanter sur une calculatrice ayant peu de mémoire.

Conclusion

A ma grande surprise le principe utilisé par les calculatrices pour obtenir les valeurs de la fonction logarithme ne repose pas du tout sur l'exploitation du développement en série de $\ln(1 + x)$ ou d'une de ses variantes plus performantes comme $\ln \frac{1-x}{1+x}$, mais simplement sur le théorème 3 dont la démonstration ne demande que la connaissance des inégalités (1) et (2).

Le calcul des valeurs des fonctions trigonométriques repose sur les mêmes principes que ceux que nous venons de voir. On peut en trouver une description rapide dans l'article [4] de J. Laporte et une étude plus complète dans l'article [2] de C. Devalland. Évidemment on peut consulter l'article original [5] de Jacques Volder, dont on trouve une reproduction sur le site [4].

Bibliographie

[1] G. Evrard (2000/2001), *TI81, Logarithme par l'algorithme CORDIC*, en ligne à l'adresse `http://gerard.evrard.free.fr/Pages/Logitheque/TI81-4.html`

[2] C. Devalland (2004) *L'algorithme CORDIC*, en ligne à l'adresse `http://cdeval.free.fr/article.php3?id_article=50`

[3] E. Hairer & G. Wanner (2001), L'analyse au fil de l'histoire, *Springer*.

[4] J. Laporte (1981), *Le secret des algorithmes*, en ligne à l'adresse `http://www.jacques-laporte.org/LeSecretDesAlgorithmes.htm`

[5] J. Volder (1959) *The CORDIC Trigonometric Computing Technique*, IRE Trans. Electronic Computing, **EC - 8** , 330–334.

Nicole Bopp
IRMA et IREM
Université Louis Pasteur, Strasbourg
`bopp@math.u-strasbg.fr`