4
INFORMATIQUE

Voyage au cœur de votre calculatrice ou Coordinate Rotation Digital Computer and Co.

par Bernard KOKANOSKY et Jean-Louis LAMARD, professeurs de Spéciales à Amiens

1. Fonctions trigonométriques

La première idée pour calculer $\cos\theta$ ou $\sin\theta$ est d'utiliser leur définition même, c'est-à-dire les séries entières

$$\cos\theta = \sum_{n=0}^{+\infty} (-1)^n \frac{\theta^{2n}}{(2n)!} \quad \text{et} \quad \sin\theta = \sum_{n=0}^{\infty} (-1)^n \frac{\theta^{2n+1}}{(2n+1)!}$$

On peut évidemment se ramener au cas où $\theta \in [0, \frac{\pi}{4}]$ et la convergence est alors rapide (au pire, précision de 10^{-10} avec 13 termes). L'ennui de cette méthode est que les opérations intervenant ne sont pas des opérations "simples" pour un microprocesseur, lesquelles sont l'addition et le décalage (c'est-à-dire la multiplication par une puissance de 10).

C'est pourquoi les calculatrices utilisent un autre algorithme : le CORDIC.

1.1. Partie théorique

Il est clair qu'on peut se ramener, grâce aux formules usuelles, à la seule détermination de $\mathrm{tg}\theta$ pour $\theta \in [0, \frac{\pi}{4}]$.

L'idée consiste à mettre en mémoire morte (R.O.M.) une suite d'angles (θ_n) dont la tangente est simple ($\mathrm{tg}\theta_n = 10^{-n}$) et d'en déduire une nouvelle suite (α_n) tendant vers θ (donc $(\mathrm{tg}\alpha_n)$ tend vers $\mathrm{tg}\theta$) telle que, en outre, $\mathrm{tg}\alpha_n$ s'obtienne à partir de $\mathrm{tg}\alpha_{n-1}$ par des opérations "simples".

Soit donc la suite (θ_n) avec $\theta_n = \text{Arctg } 10^{-n}$. C'est clairement une suite de $[0, \frac{\pi}{4}]$, décroissante et tendant vers 0.

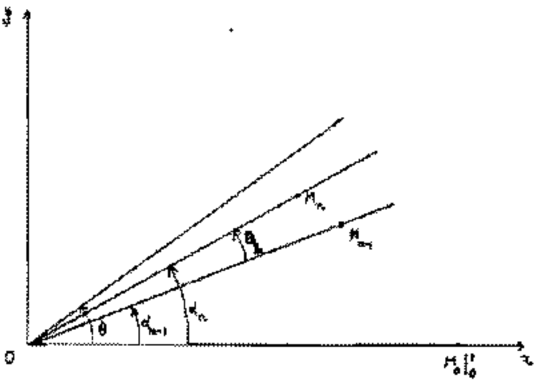
Définissons la suite (α_k) $(k \in \mathbb{N}^*)$ par :

$\alpha_1 = \theta_{i_1}$ θ_{i_1} étant le plus grand des $\theta_j \leqslant \theta$
(c'est-à-dire $\theta_{i_1} \leqslant \theta < \theta_{i_1 - 1}$)

$\alpha_2 = \alpha_1 + \theta_{i_2}$ θ_{i_2} étant le plus grand des $\theta_j \leqslant \theta - \alpha_1$
(c'est-à-dire $\alpha_1 + \theta_{i_2} \leqslant \theta < \alpha_1 + \theta_{i_2 - 1}$)

...

$\alpha_n = \alpha_{n-1} + \theta_{i_n}$ θ_{i_n} étant le plus grand des $\theta_j \leqslant \theta - \alpha_{n-1}$
c'est-à-dire $\alpha_{n-1} + \theta_{i_n} \leqslant \theta < \alpha_{n-1} + \theta_{i_n - 1}$)



Remarque : On notera que $(\theta_{i_n})_{n \in \mathbb{N}}$ n'est pas forcément une sous-suite de $(\theta_n)_{n \in \mathbb{N}}$ car $\theta_{i_n} = \theta_{i_{n+1}} = \dots = \theta_{i_{n+p-1}}$ avec $p = E\left(\dfrac{\theta - \alpha_{n-1}}{\theta_{i_n}}\right)$ avec bien sûr $\theta_{i_{n-1}} \neq \theta_{i_n}$. En d'autres termes c'est une suite "localement constante" dont tous les termes sont extraits de $\{\theta_n\}$ et non stationnaire.

Ainsi $n \longmapsto i_n$ qui est évidemment croissante au sens large ne peut rester constante à partir d'un certain rang. Donc $\lim\limits_{n \to +\infty} i_n = +\infty$.

Or par construction on a l'encadrement $\theta - \varepsilon_n < \alpha_n \leqslant \theta$ avec $\varepsilon_n = \theta_{i_{n+1} - 1}$ et compte tenu de ce qui précède : $\varepsilon_n \to 0$.

La fonction tangente étant croissante sur $[0, \frac{\pi}{4}]$, il vient :

$$\operatorname{tg}(\theta - \varepsilon_n) < \operatorname{tg} \alpha_n \leqslant \operatorname{tg} \theta$$

ce qui fournit avec les accroissements finis :

$$\operatorname{tg}\theta - \frac{\varepsilon_n}{\cos^2 \xi_n} < \operatorname{tg}\alpha_n \leqslant \operatorname{tg}\theta \quad \text{avec} \quad \xi_n \in]\theta - \varepsilon_n, \theta[\subset [0, \frac{\pi}{4}]$$

Ainsi : $\operatorname{tg}\theta - 2\varepsilon_n < \operatorname{tg}\alpha_n \leqslant \operatorname{tg}\theta$ (c'est la raison du choix de $[0, \frac{\pi}{4}]$ et non de $[0, \frac{\pi}{2}]$, sinon $\frac{1}{\cos^2 \xi_n}$ ne serait pas maîtrisé).

Conclusion : $\operatorname{tg}\alpha_n \longrightarrow \operatorname{tg}\theta$.

1.2. Algorithme de calcul des tg α_n

L'intérêt du CORDIC réside dans la simplicité du passage de $\operatorname{tg}\alpha_{n-1}$ à $\operatorname{tg}\alpha_n$.

Posons en effet $k_n = \operatorname{tg}\theta_{i_n}$ $(= 10^{-i_n})$. On a alors :

$$\operatorname{tg}\alpha_n = \operatorname{tg}(\alpha_{n-1} + \theta_{i_n}) = \frac{\operatorname{tg}\alpha_{n-1} + k_n}{1 - k_n \operatorname{tg}\alpha_{n-1}}$$

Ainsi si $M_{n-1}\left|\begin{matrix} x_{n-1} \\ y_{n-1}\end{matrix}\right.$ est un point tel que $\frac{y_{n-1}}{x_{n-1}} = \operatorname{tg}\alpha_{n-1}$, le point

$M_n\left|\begin{matrix} x_n \\ y_n\end{matrix}\right.$ avec $x_n = x_{n-1} - k_n y_{n-1}$ et $y_n = k_n x_{n-1} + y_{n-1}$ vérifiera $\frac{y_n}{x_n} = \operatorname{tg}\alpha_n$.

On passe donc de M_{n-1} à M_n par la transformation de matrice : $\begin{pmatrix} 1 & -k_n \\ k_n & 1 \end{pmatrix}$, ce qui justifie la terminologie CORDIC.

Par une itération "évoluante", une suite de $\mathbf{R}^2$ est ainsi définie :

$$M_n\left|\begin{matrix} x_n \\ y_n\end{matrix}\right. \quad \text{avec} \quad \frac{y_n}{x_n} = \operatorname{tg}\alpha_n$$

pourvu que $M_1\left|\begin{matrix} x_1 \\ y_1\end{matrix}\right.$ vérifie $\frac{y_1}{x_1} = \operatorname{tg}\alpha_1 = \operatorname{tg}\theta_{i_1} = k_1$

On constate alors qu'en partant de $M_0\left|\begin{matrix} 1 \\ 0\end{matrix}\right.$, l'"itération" donne bien M_1. Ce qui permet d'avoir toujours le même point de départ M_0 quel que soit θ .

1.3. Partie pratique

Pour des raisons matérielles évidentes(!) la machine ne connaît que les N_0 premiers termes de la suite θ_n :

θ_1 = Arctg 0,1 $\simeq$ 0,099 668 652 4
θ_2 = Arctg 10^{-2} $\simeq$ 0,009 999 666 7
θ_3 = Arctg 10^{-3} $\simeq$ 0,000 999 999 6
θ_4 = Arctg 10^{-4} $\simeq$ 10^{-4}

.
.
.

θ_{N_0} = Arctg $10^{-N_0} \simeq 10^{-N_0}$.

La machine s'arrête lorsqu'il n'est plus possible d'ajouter θ_{N_0} au dernier α_n obtenu (qu'on notera α_N) sans dépasser θ .

Ainsi $\operatorname{tg} \alpha_N = \dfrac{y_N}{x_N}$ vérifie :

$$(\operatorname{tg} \theta) - 2 \times 10^{-N_0} < \operatorname{tg} \alpha_N \leqslant \operatorname{tg} \theta \, .$$

(En général N_0 = 14 et le nombre "d'itérations" N est de l'ordre de 40. La convergence n'est pas très "rapide" mathématiquement, c'est-à-dire vis-à-vis de N, mais, les opérations étant "simples", le temps de calcul est très bref).

Exemple : Début des calculs pour tg 0,35 .

n	θ_{i_n}	k_n	α_n	x_n	y_n	$\operatorname{tg} \alpha_n = \dfrac{y_n}{x_n}$
0				1	0	0
1	0,099 668 653	0,1	0,099 668 653	1	0,1	0,1
2	0,099 668 653	0,1	0,199 337 306	0,99	0,2	0,202 020 202
3	0,099 668 653	0,1	0,299 005 959	0,97	0,299	0,308 247 423
4	0,009 999 667	10^{-2}	0,309 005 626	0,967 010	0,308 7	0,319 231 445
5	0,009 999 667	10^{-2}	0,319 005 293	0,963 923	0,318 370 1	0,330 285 822
...						
12	0,000 1	10^{-4}	0,349 404 294	0,953 944 105	0,347 572 928	0,364 353 557
13	0,000 1	10^{-4}	0,349 504 294	0,953 909 347	0,347 668 322	0,364 466 837

La valeur "exacte" est : 0,365 028 ...
On a donc au rang 13 une précision de l'ordre de 6×10^{-4}.

1.4. Remarque : C.O.R.D.I.C. = pseudo-division

On notera la parenté de cet algorithme avec celui de la division de $a \in \mathbf{N}$ par $b \in \mathbf{N}^*$.

Soit $(b_n)_{n \in \mathbf{N}}$ avec $b_n = b \times 10^{-n}$. On construit la suite (a_n) avec :

$a_1 = a - b_{i_1}$ $\quad b_{i_1}$ étant le plus grand des $b_j \leqslant a$

$a_2 = a_1 - b_{i_2}$ $\quad b_{i_2}$ étant le plus grand des $b_j \leqslant a_1$

etc.

Exemple : Division de 14 par 3 : On retranche $b_{i_1}=3$, $b_{i_2}=3$, $b_{i_3}=3$, $b_{i_4}=3$, $b_{i_5}=0,3$ (ce qui "à la main" se traduit par l'abaissement d'un zéro au dividende), etc.

2. Fonction logarithme

2.1. Séries

Il existe de nombreuses méthodes de calcul de Log X basées sur un développement en série entière. Signalons en particulier le procédé suivant :

En posant $x = \dfrac{X-1}{X+1}$, il vient

$$\text{Log } X = \text{Log } \frac{1+x}{1-x} = 2 \text{ Argth } x = 2 \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1}$$

(car $|x| < 1$ pour tout $X \in \mathbf{R}^{*+}$) .

Or, par division ou multiplication par des puissances de 2, il est toujours possible de se ramener à $X \in [\frac{1}{2}, 2]$ (la machine devra alors "connaître" Log 2), ce qui entraîne $|x| \leqslant \frac{1}{3}$. De ce fait la convergence de la série sera rapide :

$$|R_n| = |2 \sum_{n+1}^{\infty} \frac{x^{2p+1}}{2p+1}| < \frac{2}{3(2n+3)} \sum_{n+1}^{\infty} \frac{1}{9^p} = \frac{1}{12(2n+3)9^n}$$

Mais, comme pour le développement du sinus dans $[0, \frac{\pi}{4}]$, si la convergence est rapide, les opérations ne sont pas des opérations "simples".

2.2. Partie théorique

Par multiplication par une puissance de 10, on se ramène au calcul de Log X avec $X \in [1, 10]$ (la machine "connaîtra" Log 10).

L'idée consiste à mettre en mémoire morte les logarithmes d'une suite (a_n) et d'en déduire une nouvelle suite (A_n) convergeant vers Log X, telle que, en outre, A_n s'obtienne à partir de A_{n-1} par des opérations "simples" (en l'occurrence ici une simple addition !).

Soit donc la suite (a_n) avec $a_n = 1 + 10^{-n}$ dont la machine connaîtra les logarithmes. Définissons une nouvelle suite (X_n) par :

$X_1 = a_{i_1}X$ $\quad a_{i_1}$ étant le plus grand des a_j tels que $a_jX \leqslant 10$

c'est-à-dire $a_{i_1}X \leqslant 10 < a_{i_1-1}X$

$X_2 = a_{i_2} X_1$ $\quad a_{i_2}$ étant le plus grand des a_j tels que $a_j X_1 \leqslant 10$
c'est-à-dire $a_{i_2} X_1 \leqslant 10 < a_{i_2 - 1} X_1$

...

$X_n = a_{i_n} X_{n-1}$ $\quad a_{i_n}$ étant le plus grand des a_j tels que $a_j X_{n-1} \leqslant 10$
c'est-à-dire $a_{i_n} X_{n-1} \leqslant 10 < a_{i_n - 1} X_{n-1}$

Comme dans le cas de la tangente, la suite $(a_{i_n})_{n \in \mathbb{N}^*}$ n'est pas forcément une sous-suite de (a_n), mais elle ne peut rester stationnaire à partir d'un certain rang et tend donc en décroissant vers 1.

Comme : $\dfrac{10}{a_{i_{n+1} - 1}} < X_n \leqslant \dfrac{10}{a_{i_{n+1}}}$ (1), la suite (X_n) tend vers 10.

Par ailleurs $X_n = a_{i_n} a_{i_{n-1}} \ldots a_{i_1} X$; donc en posant :

$$A_n = \text{Log}\, 10 - (\text{Log}\, a_{i_1} + \ldots + \text{Log}\, a_{i_n})$$

on obtient une suite (A_n) tendant vers $\text{Log}\, X$ et telle que $A_n = A_{n-1} - \text{Log}\, a_{i_n}$.

2.3. Partie pratique

La machine connaît les N_0 premiers termes de la suite $(\text{Log}\, a_n)$.

Elle s'arrête lorsqu'il n'est plus possible de multiplier par a_{N_0} le dernier X_n obtenu sans dépasser 10, c'est-à-dire pour X_N tel que $a_{N_0} X_N > 10$ (2).

Montrons que, quitte à ajouter un terme de correction à A_N, on peut se contenter de $N_0 = 4$, ce qui prouve que cet algorithme est remarquablement "économique" (il suffit que la machine connaisse 5 logarithmes ou plutôt 6 avec Log 10 !). C'est là que réside son intérêt avec bien sûr l'utilisation d'opérations simples.

Nous avons
$$A_N = \text{Log}\, 10 - \text{Log}\, \frac{X_N}{X}$$

d'où
$$A_N - \text{Log}\, X = - \text{Log}\, \frac{X_N}{10}$$

soit, classiquement :

$$A_N - \text{Log}\, X = - \text{Log}(1 - \varepsilon_N) \text{ avec } \varepsilon_N = 1 - \frac{X_N}{10}$$

La formule de Taylor-Lagrange nous donne :

$$A_N - \text{Log}\, X = \varepsilon_N + \frac{\varepsilon_N^2}{2} + \frac{\varepsilon_N^3}{3} \, \frac{1}{(1 - c_N)^3} \qquad \text{avec} \qquad c_N \in \left]0, \varepsilon_N\right[$$

Or, clairement, $X_n \leqslant 10$ $(\forall n \in \mathbb{N})$ et en outre $X_N > \frac{10}{a_{N_0}}$ (ces 2 inégalités découlant respectivement de (1) et (2)).

D'où $\varepsilon_N \in \left]0;1-\frac{1}{a_{N_0}}\right[$ et a fortiori c_N est dans le même intervalle.

Ainsi $\frac{\varepsilon_N^3}{3}\ \frac{1}{(1-c_N)^3} < \frac{1}{3}\left(1-\frac{1}{a_{N_0}}\right)^3 \frac{1}{\left(\frac{1}{a_{N_0}}\right)^3} = \frac{1}{3}10^{-3N_0}$

$$(\text{car } a_{N_0} = 1+10^{-N_0})$$

Ainsi pour $N_0 = 4$:

$$A_N - \varepsilon_N - \frac{1}{2}\varepsilon_N^2 \in \left]\operatorname{Log}X\ ,\ \operatorname{Log}X + \frac{1}{3}\ 10^{-12}\right[$$

(On notera l'existence d'une seule opération non simple pour le calcul de $\frac{\varepsilon_N^2}{2}$)

Exemple :

La machine connaît
$$\begin{aligned} \operatorname{Log}10 &= 2{,}302\ 585\ 093 \\ \operatorname{Log}\ 2 &= 0{,}693\ 147\ 181 \\ \operatorname{Log}1{,}1 &= 0{,}095\ 310\ 179 \\ \operatorname{Log}1{,}01 &= 0{,}009\ 950\ 331 \\ \operatorname{Log}1{,}001 &= 0{,}000\ 999\ 500 \\ \operatorname{Log}1{,}0001 &= 0{,}000\ 099\ 995 \end{aligned}$$

Soit à calculer $\operatorname{Log}44{,}501 = \operatorname{Log}10 + \operatorname{Log}\ 4{,}450\ 1 = \operatorname{Log}10 + \operatorname{Log}X$

n	a_{i_n}	X_n	$\operatorname{Log}a_{i_n}$	A_n
0		4,450 1		2,302 585 093 (= Log 10)
1	2	8,900 2	0,693 147 181	1,609 437 912
2	1,1	9,790 22	0,095 310 179	1,514 127 733
3	1,01	9,888 122 2	0,009 950 331	1,504 177 402
4	1,01	9,987 003 422	0,009 950 331	1,494 227 071
5	1,001	9,996 990 425	0,000 999 500	1,493 227 571
6	1,000 1	9,997 990 124	0,000 099 995	1,493 127 576
7	1,000 1	9,998 989 923	0,000 099 995	1,493 027 581
8	1,000 1	9,999 989 822	0,000 099 995	1,492 927 586

Terme d'ajustement : $-\varepsilon_8 - \frac{1}{2}\varepsilon_8^2$ avec $\varepsilon_8 = 1 - \frac{X_8}{10} = 0{,}000\ 001\ 018$

d'où $-\varepsilon_8 - \frac{1}{2}\varepsilon_8^2 \simeq -0{,}000\ 001\ 018.$

Donc $\text{Log}\,4{,}450\,1 \simeq 1{,}492\,927\,586 - 0{,}000\,001\,018 = 1{,}492\,926\,568$

d'où $\text{Log}\,44{,}501 = 3{,}795\,511\,661$

ce qui est la valeur fournie à l'affichage par la touche LOG.

3/Fonction exponentielle

3.1. Partie théorique

Là encore, l'algorithme utilisé par les calculatrices n'est pas basé sur la série entière $e^x = \sum_0^\infty \frac{x^n}{n\,!}$

En pratique, on se ramène au calcul de e^x avec $x>0$ puis à celui de e^X avec $0 \leqslant X < \text{Log}\,10$ ($x = p\,\text{Log}\,10 + X$, d'où $e^x = 10^p e^X$: opération "simple").

On considère *la même suite (a_n)* que précédemment et on forme la suite (Y_n) :

$Y_1 = X - \text{Log}\,a_{i_1}$ $\quad a_{i_1}$ étant le plus grand des a_j tels que $\text{Log}\,a_j \leqslant X$

...

$Y_n = Y_{n-1} - \text{Log}\,a_{i_n}$ $\quad a_{i_n}$ étant le plus grand des a_j tels que $\text{Log}\,a_j \leqslant Y_{n-1}$

soit : $\text{Log}\,a_{i_n} \leqslant Y_{n-1} < \text{Log}\,a_{i_n - 1}$

D'où $a_{i_{n+1}} \leqslant e^{Y_n} < a_{i_{n+1}-1}$, ce qui prouve que $Y_n \to 0$.

Or, clairement, $$e^{Y_n} = \frac{e^X}{a_{i_1} a_{i_2} \dots a_{i_n}},$$

donc $B_n = a_{i_1} a_{i_2} \dots a_{i_n}$ tend vers e^X

(et $B_{n+1} = B_n \times a_{i_{n+1}} = B_n + 10^{-(i_{n+1})} B_n$: opération "simple").

3.2. Partie pratique

La machine s'arrête lorsqu'il n'est plus possible de retrancher $\text{Log}\,a_{N_0}$ au dernier Y_n obtenu (noté Y_N) sans obtenir un résultat négatif (donc $0 < Y_N < \text{Log}\,a_{N_0}$).

Montrons que, comme pour le Log, avec un terme correctif on peut se contenter de $N_0 = 4$. En effet :

$$e^X - B_N = B_N(e^{Y_N} - 1) = B_N\left(Y_N + \frac{Y_N^2}{2}\right) + \underbrace{B_N \frac{Y_N^3}{6} e^{c_N}}_{>0}$$

avec $c_N \in]0, Y_N[$

Or $\bullet\ Y_N^3 < (\mathrm{Log}\, a_{N_0})^3$

$\bullet\ e^{c_N} < e^{Y_N} < a_{N_0}$

$\bullet$ De $Y_N \geqslant 0$, on tire $e^{Y_N} \geqslant 1$; donc $B_N = \dfrac{e^X}{e^{Y_N}} \leqslant e^X < 10$,

puisque $X \in [0, \mathrm{Log}\ 10[$.

Ainsi :
$$0 < B_N \frac{Y_N^3}{6} e^{c_N} < \frac{5}{3}\, a_{N_0} \left(\mathrm{Log}(a_{N_0})\right)^3 < 1{,}7 \times 10^{-12}$$

avec $N_0 = 4$

Finalement :
$$B_N \left(1 + Y_N + \frac{Y_N^2}{2}\right) \in]e^X - 1{,}7 \times 10^{-12}, e^X[$$

Exemple : $e^{0,212}$

n	Log a_{i_n}	Y_n	a_{i_n}	B_n
0		0,212		1
1	0,095 310 179	0,116 689 820	1,1	1,1
2	0,095 310 179	0,021 379 640	1,1	1,21
3	0,009 950 331	0,011 429 310	1,01	1,222 1
4	0,009 950 331	0,001 478 979	1,01	1,234 321
5	0,000 999 500	0,000 479 478	1,001	1,235 555 321
6	0,000 099 995	0,000 379 483	1,000 1	1,235 678 877
7	0,000 099 995	0,000 279 488	1,000 1	1,235 802 445
8	0,000 099 995	0,000 179 493	1,000 1	1,235 926 025
9	0,000 099 995	0,000 079 498	1,000 1	1,236 049 618

Terme d'ajustement : $\left(B_9 \quad Y_9 + \dfrac{Y_9^2}{2}\right) = 0{,}000\,098\,267$

D'où $e^{0,212} = 1{,}236\,049\,618 + 0{,}000\,098\,267 = 1{,}236\,147\,885$
ce qui est la valeur fournie à l'affichage par la touche EXP.

Remarque : Dans le tableau, nous n'avons indiqué que 9 chiffres, mais en fait les calculs ont été effectués avec 11 chiffres.

Bibliographie

J.E. Volder "The Cordic Trigonometric Computing Technique"

L'Ordinateur Individuel n° 24 (février 1981)

E.G. Kogbetzianz "Mathematical Methods for digital computers" 1960 Wiley, Ruston, Wilf).