Calcul des fonctions usuelles par la calculatrice

Longtemps les calculs numériques furent une affaire de spécialistes : les abacistes. Ceux-ci calculaient en chiffres romains sur des tables dédiées à cet effet à l'aide de petits cailloux (on dit en latin *calculi*). Afin de garder leur pouvoir, ils retardèrent l'introduction en occident des chiffres arabes et de la numération décimale de position. Cette numération se prêtent bien mieux aux calculs numériques cela permit à Pascal, alors âgé de 19 ans, de mettre au point en 1642 la première machine capable d'additionner et de soustraire : la Pascaline. Depuis les techniques n'ont fait que progresser et les premières machines à calculer électroniques sont apparues au début des années 1970. Elles rendirent obsolètes les règles à calculs et autres tables de logarithme qui étaient jusqu'alors utilisées pour évaluer les fonctions trigonométriques, exponentielles et logarithmiques. L'objet de cet article est de présenter les démarches algorithmiques qui permettent à nos calculatrices modernes de déterminer les valeurs prises pas les fonctions usuelles.

Représentation des nombres

Dans les calculatrices scientifiques modernes, les nombres à virgules sont représentés à l'aide d'une mantisse formée de 13 chiffres en base 10, chacun étant codé sur 4 bits. Cette mantisse est suivie d'un exposant correspondant à la puissance de dix multiplicatrice de la mantisse. La calculatrice va donc travailler avec 13 chiffres significatifs, mais à cause des petites erreurs qui s'accumulent lors des calculs, on peut considérer que les dernières décimales sont peu sûres et c'est pourquoi la calculatrice n'affichera que les 10 premières. A contrario, le logiciel de calcul formel Maple affiche toutes les décimales de ses calculs, il faut donc savoir que les dernières sont parfois fausses.

Avec cette représentation électronique en base 10 des nombres manipulés, les opérations d'addition et de soustraction sont assez faciles à implémenter au niveau des circuits électroniques, elles seront donc rapidement exécutées par le processeur de la calculatrice. Les multiplications sont en revanche coûteuses en temps de calculs et les divisions le sont encore plus. Nous chercherons donc à en réduire le nombre. C'est pour cette raison que les évaluations des fonctions numériques à partir de développement en série entière ne conduisent pas à des algorithmes efficaces : il y aurait trop de multiplications complexes. Cependant, soulignons que les multiplications et divisions par 10 sont rapides à réaliser car elles n'opèrent qu'au niveau de l'exposant. Notons aussi que les multiplications et divisions par 2, ou plus généralement par un petit entier, sont elles aussi assez rapides.

La calculatrice va nous donner des valeurs numériques approchées des nombres que nous souhaitons calculer. Il est donc intéressant de préciser le vocabulaire relatif à la notion de valeur approchée. On dit qu'un réel x est une valeur approchée d'un réel a avec une erreur *absolue* inférieure $\varepsilon \geq 0$ lorsqu'on a $|a-x| \leq \varepsilon$. Cette notion de valeur approchée n'est pas satisfaisante lorsqu'on cherche à connaître les 13 premières décimales d'un nombre. En effet il importe peu de connaître a à la précision 10^{-13}, si le nombre a est de l'ordre de plusieurs milliards. En revanche, il n'est pas suffisant de connaître a à la précision 10^{-13}, si le nombre a est lui-même de l'ordre de 10^{-13}. Pour ces raisons, nous introduisons la notion d'erreur relative : on dit que qu'un réel x est une valeur approchée d'un réel non nul a avec une erreur relative inférieure à ε lorsqu'on a $\dfrac{|a-x|}{|a|} \leq \varepsilon$. Si l'on connaît a avec une erreur relative de l'ordre de 10^{-14} alors on peut affirmer que les 13 premières décimales de a sont correctes. Soulignons au passage que cette notion d'erreur relative est globalement compatible avec la multiplication et le passage à l'inverse mais ne l'est pas avec l'addition et la soustraction.

Il ne nous reste maintenant plus qu'à voir les algorithmes permettant d'évaluer les différentes fonctions usuelles.

La fonction racine carrée

Commençons par étudier le fonctionnement de la touche $\sqrt{.}$. Nous souhaitons évaluer la racine carrée d'un réel a strictement positif. Celle-ci étant solution de l'équation $x^2 - a = 0$, l'application de la méthode des tangentes de Newton nous invite à considérer une suite (x_n) vérifiant la relation de récurrence

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right).$$

On montre sans peine, qu'en partant d'un $x_0 > 0$, la suite obtenue converge vers $\sqrt{a}$. Il suffit alors à la machine de calculer les termes successifs de cette suite jusqu'à obtention de deux termes consécutifs égaux, termes qui constitueront alors une approximation satisfaisante de $\sqrt{a}$. Notons que la convergence de la suite (x_n) est quadratique car on peut montrer :

$$x_{n+1} - \sqrt{a} \sim \frac{1}{2\sqrt{a}}\left(x_n - \sqrt{a}\right)^2$$

Ainsi, plus x_n est proche de $\sqrt{a}$, plus x_{n+1} s'en rapproche ; la convergence est donc rapide.

Cette méthode est connue depuis l'antiquité, on l'appelle méthode de Héron ou algorithme de Babylone. Elle est simple mais a pour inconvénient de nécessiter une division complexe à chaque étape. Afin de réduire le nombre de divisions, nous allons procéder de manière plus détournée.

Au lieu de calculer $\sqrt{a}$, cette fois-ci nous allons évaluer $1/\sqrt{a}$ puis conclure par un passage à l'inverse. Le nombre $1/\sqrt{a}$ est solution de l'équation $\frac{1}{x^2}-a=0$. Par la méthode des tangentes de Newton, nous sommes amener à considérer une suite (x_n) définie par :

$$x_{n+1}=\frac{3x_n-ax_n^3}{2}$$

On montre que, pour $x_0\in\left]0,1/\sqrt{a}\right]$, la suite croît vers $1/\sqrt{a}$ et cette convergence est, elle aussi, quadratique. Cette fois-ci, il y a 3 multiplications complexes à chaque étape et un passage à l'inverse à la fin de l'algorithme, c'est plus efficace.

Ci-dessous est représentée la progression du nombre de décimales correctes lors du calcul de $\sqrt{3}$, par la formule d'itération $x_{n+1}=\frac{3x_n-ax_n^3}{2}$.

L'accélération finale dans l'obtention de ces décimales illustre le caractère quadratique de la convergence.

n	Approximation
0	3,0000000000
1	2,250000000000
2	**1**.869230769230
3	**1.7**45809828502
4	**1.732**211772409
5	**1.73205083**0003
6	**1.732050807569**

La fonction exponentielle

Nous allons chercher ici à évaluer $\exp(a)$ pour $a\in\mathbb{R}$ mais nous nous limiterons au cas où a est positif. En effet si le réel a est négatif, il suffit pour conclure de considérer $-a$ puis de réaliser un passage à l'inverse au terme des calculs.

Pour calculer $\exp(a)$ avec $a\geq 0$, nous allons décomposer a en une somme de réels dont on connaît les exponentiels. Pour cela nous exploitons une liste de valeurs précalculées figurant dans les mémoires internes de la calculatrice. Ces valeurs sont celles des réels $\ln(1+10^{-k})$ pour $k\in\{0,1,\ldots,7\}$

Pour décomposer a, nous commençons par poser $a_0=a$, puis nous cherchons parmi les valeurs de la liste précédente, la plus grande qui soit inférieure à a_0. Notons la x_1 et reprenons le processus précédent à partir du complément $a_1=a_0-x_1$. On continue ainsi de suite, jusqu'à ce que ce processus s'arrête, c'est-à-dire jusqu'à l'obtention d'un réel positif $\varepsilon=a_n$ inférieur à $\ln(1+10^{-7})$ (quantité elle-même inférieure à 10^{-7}). Nous avons alors la relation

$$a=x_1+\cdots+x_n+\varepsilon$$

d'où nous tirons

$$\mathrm{e}^a=\mathrm{e}^{x_1}\cdots\mathrm{e}^{x_n}\mathrm{e}^{\varepsilon}.$$

Les quantités e^{x_k} sont connues car de la forme $1+10^{-i}$ et la quantité e^{ε} s'évalue par l'égalité $\mathrm{e}^{\varepsilon}\simeq 1+\varepsilon$ ce qui fournit au final :

$$\mathrm{e}^a\simeq\mathrm{e}^{x_1}\cdots\mathrm{e}^{x_n}(1+\varepsilon)$$

avec une erreur relative de l'ordre de 10^{-14}.

C'est cette idée qui est à l'origine de l'algorithme présenté en annexe. Notons que cette algorithme peut gagner en efficacité en commençant par rechercher plus directement le plus grand multiple de $\ln 2$ inférieur à a.

La fonction logarithme népérien

L'idée sous-jacente au calcul de $\exp(a)$ revenait en fait à écrire :

$$a=n_0\ln 2+n_1\ln(1+10^{-1})+\cdots+n_7\ln(1+10^{-7})+\varepsilon$$

avec $n_k\in\mathbb{N}$ et $\varepsilon\in[0,10^{-7}]$ afin de pouvoir conclure :

$$\mathrm{e}^a\simeq 2^{n_0}(1+10^{-1})^{n_1}\cdots(1+10^{-7})^{n_7}(1+\varepsilon)$$

avec une erreur relative de l'ordre de 10^{-14}.

De manière inverse, pour calculer $\ln a$ avec $a>1$, on peut chercher à écrire

$$a=2^{n_0}(1+10^{-1})^{n_1}\cdots(1+10^{-7})^{n_7}(1+\varepsilon)$$

avec $n_k\in\mathbb{N}$ et $\varepsilon\in[0,10^{-7}]$ et exploiter la relation $\ln(1+\varepsilon)\simeq\varepsilon$ afin de pourvoir conclure :

$$\ln a\simeq n_0\ln 2+n_1\ln(1+10^{-1})+\cdots+n_7\ln(1+10^{-7})+\varepsilon$$

avec une erreur absolue inférieure à 10^{-14}. Si nous souhaitons plutôt disposer d'une erreur relative de l'ordre de 10^{-14}, il faudra exploiter l'approximation plus fine $\ln(1+\varepsilon) \simeq \varepsilon - \frac{1}{2}\varepsilon^2$.

La décomposition de a en un produit se fait aisément en commençant par diviser a par 2 tant que le résultat est supérieur à 1, puis en divisant par $1+10^{-1}$ etc. L'algorithme correspondant à cette démarche est présenté en annexe.

Les fonctions trigonométriques

Pour évaluer les fonctions cos, sin et tan sur un angle θ la démarche que nous allons suivre présente plusieurs similarités avec celle vue lors de l'étude de la fonction exponentielle. Cette démarche a été développée par J.E Volder en 1959 et est appelée algorithme CORDIC (pour Cordiante Rotation Digital Computer).

Pour calculer $\cos\theta, \sin\theta$ et $\tan\theta$, nous commençons par exploiter les formules de trigonométrie usuelles de sorte de ramener le problème au cas où $\theta \in [0, \pi/4]$. Ensuite nous décomposons θ en une somme d'angles dont les cosinus, sinus et tangentes sont connus. Pour cela nous allons encore une fois exploiter une liste de valeurs précalculées figurant en mémoire, les valeurs des angles $\arctan(10^{-k})$ pour $k \in \{0,\ldots,4\}$. Les cosinus, sinus et tangentes de ces angles sont aisément calculables.

Pour décomposer θ, nous posons $\theta_0 = \theta$ puis nous cherchons, parmi les angles précalculées, le plus grand qui soit inférieur à θ_0, notons le α_1. On considère alors le complément $\theta_1 = \theta_0 - \alpha_1$ et on reprend ce processus à partir de θ_1 et ce jusqu'à obtention d'un θ_n inférieur à $\arctan(10^{-4})$. Ainsi nous parvenons à la relation $\theta = \alpha + \varepsilon$ avec $\alpha = \alpha_1 + \cdots + \alpha_n$ et $\varepsilon = \theta_n$ réel positif inférieur à $\arctan(10^{-4})$ donc inférieur à 10^{-4}.

Parallèlement à la décomposition de θ, nous calculons $\cos\alpha$ et $\sin\alpha$ en construisant deux suites $(x_p)_{0\le p\le n}$ et $(y_p)_{0\le p\le n}$ telles que $x_p = \cos(\alpha_1 + \cdots + \alpha_p)$ et $y_p = \sin(\alpha_1 + \cdots + \alpha_p)$. En exploitant les formules de développement de $\cos(a+b)$ et $\sin(a+b)$, on vérifie que ces suites sont définies par

$$\begin{cases} x_0 = 1 \\ y_0 = 0 \end{cases} \text{ et } \begin{cases} x_p = x_{p-1}\cos\alpha_p - y_{p-1}\sin\alpha_p \\ y_p = x_{p-1}\sin\alpha_p + y_{p-1}\cos\alpha_p \end{cases}.$$

Au terme de cette construction, on dispose de $x_n = \cos\alpha$, $y_n = \sin\alpha$ et $y_n/x_n = \tan\alpha$.

Cette démarche pourrait sembler satisfaisante mais elle induit trop de multiplications complexes. De plus, pour être efficace, elle nécessiterait la mise en mémoire des valeurs des cosinus et sinus des α_p c'est-à-dire des angles $\arctan 10^{-k}$.

Optimisons

Pour gagner en efficacité nous allons réorienter le problème et non plus calculer directement le cosinus et sinus de $\theta \in [0, \pi/4]$ mais plutôt en calculer la tangente. Notons que $\cos\theta$ et $\sin\theta$ se déduisent de $\tan\theta$ par les relations

$$\cos\theta = \frac{1}{\sqrt{1+\tan^2\theta}} \text{ et } \sin\theta = \frac{\tan\theta}{\sqrt{1+\tan^2\theta}}$$

Commençons par réécrire le système définissant les suites (x_p) et (y_p) :

$$\begin{cases} x_p = \cos\alpha_p (x_{p-1} - y_{p-1}\tan\alpha_p) \\ y_p = \cos\alpha_p (x_{p-1}\tan\alpha_p + y_{p-1}) \end{cases}.$$

Comme cette fois-ci ce n'est que le rapport y_p/x_p qui nous intéresse, il est possible de considérer les suites $(X_p)_{0\le p\le n}$ et $(Y_p)_{0\le p\le n}$ définies par :

$$\begin{cases} x_0 = 1 \\ y_0 = 0 \end{cases} \text{ et } \begin{cases} X_p = X_{p-1} - Y_{p-1}\tan\alpha_p \\ Y_p = X_{p-1}\tan\alpha_p + Y_{p-1} \end{cases}.$$

On observe aisément qu'on a pour tout $p \in \{0,\ldots,n\}$

$$\frac{Y_p}{X_p} = \frac{y_p}{x_p}$$

Notons que cette fois-ci, les calculs définissant les suites (X_p) et (Y_p) sont beaucoup plus rapides. Certes il existe toujours deux multiplications, mais celles-ci sont des multiplications par $\tan\alpha_p$ qui n'est rien d'autre qu'une puissance négative de 10, ces multiplications sont donc rapides.

Au terme de ce calcul, on obtient $Y_n/X_n = \tan(\alpha)$ et il ne reste plus qu'à en déduire la tangente de $\theta = \alpha + \varepsilon$. Pour cela nous exploitons la formule de développement de $\tan(a+b)$ et l'approximation $\tan\varepsilon \simeq \varepsilon + \frac{1}{3}\varepsilon^3$ afin d'obtenir

$$\tan\theta \simeq \frac{3Y_n + (3\varepsilon + \varepsilon^3)X_n}{3X_n - (3\varepsilon + \varepsilon^3)Y_n}$$

avec une erreur relative de l'ordre de 10^{-14}. L'algorithme correspondant à cette démarche est présenté en annexe.

Et pour aller encore plus vite…

Pour rendre cet algorithme encore plus efficace, il convient de travailler avec des angles de la forme $\arctan(2^{-k})$ avec $k \in \mathbb{N}$. A l'aide de l'inégalité :

$$\arctan(2a) \leq 2\arctan a$$

valable pour $a \geq 0$, on peut justifier qu'il est possible de décomposer n'importe quel $\theta \in [0, \pi/4]$ sous la forme

$$\theta = \sum_{k=1}^{+\infty} \varepsilon_k \arctan(2^{-k})$$

avec $\varepsilon_k = 1$ ou -1. La suite (ε_k) se construit par récurrence en posant ε_p de signe opposé à la différence $\theta - \sum_{k=1}^{p-1} \varepsilon_k \arctan(2^{-k})$. Les suites (X_p) et (Y_p) sont alors définies de la manière suivante :

$$\begin{cases} X_0 = K \\ Y_0 = 0 \end{cases} \text{ et } \begin{cases} X_p = X_{p-1} - \varepsilon_p 2^{-p} Y_{p-1} \\ Y_p = \varepsilon_p 2^{-p} X_{p-1} + Y_{p-1} \end{cases}$$

avec K constante précalculée égale à $\prod_{k=1}^{+\infty} \cos(\arctan(2^{-k}))$, constante voisine de $0{,}85879$. Avec cette démarche, nous constatons que les suites (X_p) et (Y_p) convergent respectivement vers $\cos\theta$ et $\sin\theta$, et nous disposons d'une méthode permettant de calculer efficacement les fonctions trigonométriques sur des nombres représentés en binaire.

Annexe : Algorithmes d'évaluation des fonctions usuelles

Racine carrée

Argument $a > 0$.

$x = 0$, $y = \min(1, 1/a)$

Tant que ($x < y$) faire { $x = y$, $y = (3x - ax^3)/2$ }

Répondre $1/y$.

La suite des x ainsi construits étant croissante, le test d'arrêt $x < y$ est satisfaisant. On préférera celui-ci au teste $x \neq y$ car, à cause de l'imprécision numérique, des phénomènes cycliques peuvent apparaître pour x voisin de $1/\sqrt{a}$.

Exponentielle

L_k correspond à la valeur de $\ln(1+10^{-k})$ figurant en mémoire

Argument $a \in \mathbb{R}$.

Début

$x = |a|$, $y = 1$, $k = 0$.

Tant que ($k \leq 7$) faire

{ Tant que ($L_k \leq x$) faire { $x = x - L_k$, $y = y + y10^{-k}$ }

$k = k + 1$ }

$y = y(1 + x)$

Si ($a \geq 0$) alors {Répondre y } Sinon {Répondre $1/y$ }

Fin

Logarithme népérien

L_k correspond à la valeur de $\ln(1+10^{-k})$ figurant en mémoire

Argument $a > 0$.

Début

Si ($a \geq 1$) alors { $x = a - 1$ } Sinon { $x = 1/a - 1$ }

$y = 0$, $k = 0$.

Tant que ($k \leq 7$) faire

{ Tant que $(10^{-7} \leq x)$ faire { $x = (x - 10^{-k})/(1 + 10^{-k})$, $y = y + L_k$ }

$k = k + 1$ }

$y = y + x - \frac{1}{2}x^2$.

Si ($a \geq 1$) alors {Répondre y }Sinon {Répondre $-y$ }

Fin

Tangente

A_k correspond à la valeur de $\arctan(10^{-k})$ figurant en mémoire

Argument $a \in [0, \pi/4]$.

Début

$t = a$, $x = 1$, $y = 0$, $k = 1$.

Tant que ($k \leq 3$) faire

{ Tant que ($L_k \leq t$) faire { $t = t - L_k$, $\begin{cases} x = x - 10^{-k} y \\ y = 10^{-k} x + y \end{cases}$ }

$k = k + 1$ }

Répondre $\dfrac{3y + (3t + t^3)x}{3x - (3t + t^3)y}$

Fin.