UTILISATION DES ORDINATEURS EN THEORIE DES NOMBRES (*)

par Jean-Louis NICOLAS
Université de Limoges (France)

INTRODUCTION

De tout temps les arithméticiens ont calculé des familles de nombres et ont construit des tables numériques. A partir de ces calculs, ils établissaient des conjectures qu'ils démontraient ou qu'ils transmettaient à leurs descendants. Depuis quelques années, la puissance des ordinateurs a permis d'augmenter considérablement ces calculs, et a entraîné aussi une amélioration spectaculaire des algorithmes, c'est-à-dire des méthodes de calcul (cf. [4]).

Nous nous proposons de montrer sur quelques exemples l'évolution des calculs arithmétiques, et de mettre en évidence la conjonction des développements de recherches en informatique et en mathématique. Ce sera particulièrement net dans les méthodes de factorisation, avec le crible quadratique dû à C. Pomerance qui utilise au mieux le calcul vectoriel des ordinateurs Cray, pour effectuer des opérations arithmétiques classiques, mais très astucieusement choisies.

Calcul de π

La formule de Machin [1706], illustration classique des propriétés de la fonction Arc tg:

$$\frac{\pi}{4} = 4 \operatorname{Arc\,tg} \frac{1}{5} - \operatorname{Arc\,tg} \frac{1}{239}$$

a été utilisée en 1949 par Reitweisner pour calculer 2000 décimales de π, puis par Genuys en 1958 pour en obtenir 10000.

En 1961, Shanks et Wrench (cf. [11]) lui ont préféré, pour calculer 100000 décimales, la formule:

$$\frac{\pi}{4} = 6 \operatorname{Arc\,tg} \frac{1}{8} + 2 \operatorname{Arc\,tg} \frac{1}{57} + \operatorname{Arc\,tg} \frac{1}{239}$$

mieux adaptée au calcul binaire.

(*) D'après l'exposé fait au Colloque Informatique, Mathématique et Logique (Bruxelles, 21-23 mai 1984).

BULLETIN DE LA SOCIÉTÉ MATHÉMATIQUE DE BELGIQUE, t. XXXVII, 1985.

On trouvera dans le livre 'π' (cf. [6] p. 199–206) un historique très complet du calcul des décimales de π. Un progrès théorique important a été accompli par Salamin (cf. [10]) qui a utilisé la convergence très rapide de la moyenne arithmético-géométrique. Si l'on part de deux nombres positifs a_0 et b_0, on définit successivement

$$a_{n+1} = \frac{1}{2}(a_n + b_n) \quad \text{et} \quad b_{n+1} = \sqrt{a_n b_n}$$

et il est facile de voir que les deux suites (a_n) et (b_n) sont adjacentes, et ont pour limite un nombre qui est par définition la moyenne arithmético-géométrique de a_0 et b_0: a.g.m. (a_0, b_0). Si l'on observe que

$$a_{n+1} - b_{n+1} = \frac{1}{2} \frac{(a_n - b_n)^2}{(\sqrt{a_n} + \sqrt{b_n})^2}$$

on constate que, si a_n et b_n ont en commun k chiffres décimaux, a_{n+1} et b_{n+1}ont en commun $2k$ chiffres décimaux environ.

La formule de Salamin est:

$$\frac{\pi}{4} = \frac{(\text{a.g.m.}(a_0, b_0))^2}{1 - \sum_{j=1}^{\infty} 2^{j+1}(a_j^2 - b_j^2)}$$

avec $a_0 = 1$, $b_0 = 1/\sqrt{2}$

Cette formule n'était pas commode à programmer, à cause des difficultés de la multiprécision, c'est-à-dire de la manipulation des grands nombres en ordinateurs, que nous traiterons au paragraphe suivant. Cependant Y. Tamura et Y. Kanada ont réussi à la programmer et à calculer 8 millions de décimales de π (cf. [8]).

Multiprécision

Les machines à calculer ou les ordinateurs utilisent en général des nombres d'au plus 10 ou 12 chiffres. Pour traiter des nombres plus grands, ou pour traiter des nombres réels avec beaucoup de décimales, il faut disposer d'un système de sous-programmes, appelé système de multiprécision. La principe est de couper les grands nombres en tranches de 4 ou 5 chiffres, et de faire les opérations sur ces tranches (sans oublier les retenues) comme on fait avec des tranches de un chiffre lorsqu'on fait une opération à la main.

Avant d'écrire un système de multiprécision, un bon apprentissage est de faire des opérations sur de grands nombres, avec une simple calculette:

EXEMPLE 1. — Ajouter deux nombres de 10 chiffres avec une calculette

	12	3456	7890
	54	3210	9876
		1	7666
	6666		
66			
66	6667	7666	

EXEMPLE 2. — Multiplier un nombre de 12 chiffres par un nombre de 7 chiffres.

a_2	a_1	a_0			7251	4368	9087
	b_1	b_0				652	3854
		a_0b_0				3502	1298
	a_1b_0				1683	4272	
a_2b_0				2794	5354		
	a_0b_1				592	4724	
a_1b_1				284	7936		
a_2b_1			472	7652			
			473	9731	5566	2498	1298

La division est plus difficile. Il suffit, pour s'en convaincre, d'essayer de faire à la main la division de 80 145 par 89, et d'analyser les étapes successives.

Revenons au calcul de π: la multiplication de deux nombres réels ayant $5 \cdot 10^6$ décimales nécessitera, par le procédé ci-dessus, 10^{12} multiplications élémentaires de deux nombres de 5 chiffres. A raison de 10^6 opérations par seconde, cela fait 10^6 secondes, soit plus de 10 jours. Les plus gros ordinateurs actuels effectuent 10^8 opérations par seconde, cela fait encore plus d'une heure de calcul. On a découvert des méthodes de calcul en multiprécision, qui sont plus rapides. Elles sont cependant très techniques, et difficiles à programmer. On trouvera la description de ces méthodes dans l'excellent livre de D. Knuth (cf. [3]).

Nombres de Mersenne

Ce sont les nombres de la forme $2^p - 1$, où p est premier, et qui sont premiers. On en connaît actuellement 29, correspondant aux valeurs de l'exposant $p = 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279,$

2203, 2281, 3217, 4253, 4423, 9689, 9941, 11213, 19937, 21701, 23209, 44497, 86243, 132049.

Un test spécifique, le test de Lucas-Lehmer, permet de décider si $2^p - 1$ est premier ou pas: 'Soit p premier $\neq 2$, $L_0 = 4$. On construit par récurrence:

$$L_{n+1} \equiv L_n^2 - 2 \bmod (2^p - 1)$$

Alors, $2^p - 1$ est premier si et seulement si $L_{p-2} = 0$'.

Le nombre $2^{132049} - 1$ est le plus grand nombre premier actuellement connu. Il a 39751 chiffres décimaux.

Tests de primalité

Pour des nombres ordinaires, qui n'ont pas la forme des nombres de Mersenne, il est beaucoup plus difficile de dire s'ils sont premiers ou non. Les meilleures méthodes actuellement connues sont basées sur le théorème de Fermat: Si p est premier, et si a est premier avec p, $a^{p-1} \equiv 1 \bmod p$. On en déduit le test suivant:

TEST: Soit N, a premier avec N

$a^{N-1} \not\equiv 1 \bmod N \Rightarrow N$ est composé

$a^{N-1} \equiv 1 \bmod N \Rightarrow N$ est probablement premier.

Ce test est très rapide. Le calcul du p.g.c.d. de a et de N, ainsi que le calcul de $a^{N-1} \bmod N$ se calcule en un nombre d'opérations $O(\log N)$, proportionnel au nombre de chiffres de N. Il permet d'affirmer très vite qu'un nombre est composé. Il ne permet pas de garantir qu'un nombre est premier. Le théorème de Wilson:

$$(N-1)! \equiv -1 \bmod N \Leftrightarrow N \text{ est premier}$$

n'est pas utilisable, car on ne sait pas calculer $(N-1)! \bmod N$ très vite.

Le test de Fermat ci-dessus a été amélioré. Un meilleur test est basé sur le résultat: Si N est premier, et a premier avec N, $a^{(N-1)/2} \equiv \pm 1 \bmod N$. Récemment un nouveau test a été mis au point par Adleman, Rumely et Pomerance, et amélioré par Cohen et Lenstra (cf. [1], [2], [5]).

Le nombre théorique d'opérations à effectuer est $O(\log N)^{c \log\log \log N}$. Il garantit qu'un nombre est premier, et pratiquement on peut l'appliquer à des nombres de 100 à 200 chiffres en un temps de l'ordre de quelques minutes d'ordinateur.

Cryptographie à clé publique

Une des raisons de l'étude intensive actuelle des tests de primalité et des méthodes de factorisation est la publication d'une méthode de cryptographie (cf. [9]). Le principe de ce codage est le suivant:

1) Choisir deux nombres premiers p et q (de 50 chiffres environ)
2) Calculer $n = pq$
3) Calculer $\phi(n) = (p-1)(q-1)$
4) Choisir d premier avec $\phi(n)$
5) Calculer e tel que $ed \equiv 1 \bmod \phi(n)$

Les nombres p, q, $\phi(n)$, d sont gardés secrets, n et e sont publiés dans un annuaire.

Comment envoyer un message? On lit n et e dans l'annuaire. On met le message sous la forme d'un (ou plusieurs) nombres $M \leqslant n$. On calcule

$$C \equiv M^e \bmod n.$$

On envoie C, qui est le message codé.

Comment décoder un message? Seul l'auteur du code, qui connaît d, peut le faire en calculant

$$M \equiv C^d \bmod n.$$

Cette formule s'établit en utilisant le théorème d'Euler, qui généralise le théorème de Fermat: Si a et n sont premiers entre eux,

$$a^{\phi(n)} \equiv 1 \bmod n.$$

Pour percer ce code, il faut savoir trouver p et q, c'est-à-dire factoriser n. La meilleure méthode actuelle de factorisation, que nous allons exposer au paragraphe suivant, permet de factoriser des nombres de 70 chiffres au maximum. En choisissant n avec une centaine de chiffres, il est impossible à un ennemi de le factoriser. La méthode est basée sur le fait qu'il est relativement facile de construire de grands nombres premiers, et qu'il est impossible à l'heure actuelle de factoriser des grands nombres.

Le crible quadratique de C. Pomerance (cf. [7])

On veut factoriser N. Le paramètre théorique qui servira à la mesure de la vitesse de l'algorithme est:

$$L = \exp(\sqrt{\log N \log\log N}).$$

En pratique, on choisira des bornes voisines des bornes théoriques de façon à minimiser le temps sur des exemples.

1) On considère la forme quadratique

$$Q(A) = ([\sqrt{N}] + A)^2 - N$$

où $[t]$ est la partie entière de t.

Pour les valeurs de $A = 0, \pm 1, \ldots, \pm A \max \sim L^{\sqrt{9/8}}$ on observe que $Q(A) \sim 2A\sqrt{N}$, et n'a guère plus de la moitié des chiffres de N.

2) On choisit une 'base' de nombres premiers, par exemple tous les nombres premiers $\leqslant B \sim L^{1\sqrt{8}}$:

$$p_1 = 2, p_2 = 3, \ldots, p_k \leqslant B .$$

3) On recherche une famille $A_1, A_2, \ldots, A_{k+1}$ dans l'intervalle ($-A$ max, A max) telle que $Q(A_i)$ se factorise complètement sur la base de nombres premiers. On verra plus tard comment s'effectue la recherche des A_i. Comme tous les facteurs de $Q(A_i)$ appartiennent à la base, on peut écrire:

$$Q(A_i) = p_1^{a_{i,1}} \cdot p_2^{a_{i,2}} \ldots p_k^{a_{i,k}} \quad \text{avec} \quad a_{i,j} \geqslant 0 .$$

4) Soit $K = \mathbb{Z}/2\mathbb{Z}$ le corps à deux éléments. Pour chaque i, on note $\vec{v}_i$ le vecteur de K^k dont les coordonnées sont:

$$\vec{v}_i = (a_{i,1} \bmod 2, a_{i,2} \bmod 2, \ldots, a_{i,k} \bmod 2).$$

La famille de vecteurs $\vec{v}_1, \ldots, \vec{v}_{k+1}$ est linéairement dépendante. Il existe une combinaison linéaire du type:

$$\vec{v}_{i_1} + \vec{v}_{i_2} + \ldots + \vec{v}_{i_r} = 0.$$

On obient une telle combinaison par la méthode du pivot de Gauss en un temps $O(L^{\sqrt{9/8}})$. On observe que ces calculs sur le corps K sont très bien adaptés aux ordinateurs.

5) On pose:

$$x = p_1^{(a_{i_1,1} + a_{i_2,1} + \ldots + a_{i_r,1})/2} \ldots p_k^{(a_{i_1,k} + \ldots + a_{i_r,k})/2}$$

On a:

$$Q(A_{i_1}) Q(A_{i_2}) \ldots Q(A_{i_r}) \equiv x^2 \bmod N$$

On pose:

$$y = ([\sqrt{N}] + A_{i_1}) \ldots ([\sqrt{N}] + A_{i_r})$$

On a: $y^2 \equiv x^2 \bmod N$

et p.g.c.d. $(y - x, N)$ est un facteur de N en général non trivial.

Comment trouver les A tels que les facteurs de $Q(A)$ soient petits? Si p divise $Q(A)$ il divise aussi $Q(A + p), Q(A + 2p)\ldots$ d'où la méthode de crible.

On initialise un tableau $T(A)$ en simple précision pour $-A$ max $\leqslant A \leqslant A$ max, en posant:

$$T(A) = \operatorname{Log} Q(A)$$

Pour chacun des nombres premiers $p = p_1, \ldots, p_k$ de la base, et pour leurs puissances p^α, on soustrait $\log p$ de $T(A)$ chaque fois que $Q(A)$ est multiple de p^α. Les A pour lesquels $Q(A)$ est multiple de p^α appartiennent

à deux progressions arithmétiques de raison p^α qu'il est facile de déterminer.

A la fin du criblage, les A recherchés, tels que $Q(A)$ n'ait que $p_1, \ldots, p_k$ comme facteurs premiers vérifient théoriquement $T(A) = 0$; pratiquement, à cause des erreurs d'arrondi, $T(A)$ sera petit. Les autres A auront pour $Q(A)$ un facteur premier $\geqslant B$, et donc $T(A) \geqslant \log B$, ce qui est un nombre relativement grand. On distinguera donc sans peine (même avec un calcul en simple précision à l'ordinateur) les A tels que $Q(A)$ se factorise dans la base, et pour ceux-là on détermine par division successive par p_1, $p_2, \ldots, p_k$ la factorisation complète de $Q(A)$. Pour que l'algorithme fonctionne, il faut pouvoir fabriquer suffisamment de tels A. La fonction ψ de De Bruijn

$$\psi(x, y) = \sum_{\substack{n \leqslant x \\ p \mid n \Rightarrow p \leqslant y}} 1$$

qui compte le nombre d'entiers $\leqslant x$ dont tous les facteurs premiers sont $\leqslant y$, vérifie la relation

$$\psi(x, y) = x \exp((-u \log u)(1 + o(1)))$$

lorsque $u = \log x / \log y$ tend vers l'infini en restant inférieur à $(\log x)^{1-\varepsilon}$.

On peut interpréter ce résultat en estimant que la probabilité qu'un nombre entier voisin de x ait tous ses facteurs premiers $\leqslant y$ est environ $e^{-u \log u}$. On peut donc conjecturer que la probabilité que $Q((A)$ se factorise sur la base est environ $e^{-u \log u}$, avec $u = \dfrac{\log Q(A)}{\log B} \sim \dfrac{1}{2} \dfrac{\log N}{\log B}$. On en déduit une estimation heuristique de l'algorithme en $O(L^{\sqrt{9/8}})$.

Cet algorithme a permis de décomposer le nombre $(10^{71} - 1)/9$ qui s'écrit en base 10 avec 71 chiffres 1 en 10 heures de Cray XMP, qui est le plus gros ordinateur actuel.

REFERENCES

[1] ADLEMAN, L. M., POMERANCE, C., RUMELY, R. S., 1983. On distinguishing prime numbers from composite numbers. – *Ann. Math.* 117, p. 173–206.

[2] COHEN, H. and LENSTRA, H. W. Jr. Primality testing and Jacobi sums. *Math. of Comp.*, 42 (1984), p. 297–330.

[3] KNUTH, D. E., 1981. *The art of computer programming*, vol. 2, semi numerical algorithms. – 2nd ed, Addison Wesley, Reading, Mass.

[4] NICOLAS, J. L., 1980. Utilisation des ordinateurs en théorie des nombres. – *Gazette de la S.M.F.* n° 13, pp. 77–84.

[5] NICOLAS, J. L. Tests de primalité. *Expositiones Mathematicae*, 2 (1984), p. 223–234.

[6] *π. Supplément au Petit Archimède* n° 64–65. Paris 1980.
[7] POMERANCE, C. The quadratic sieve factoring algorithm. Proceedings of Eurocrypt 84, Advances on cryptology, *Lecture Notes in Computer Science*, n° 209, Springer-Verlag, p. 169–182.
[8] *Pour la Science*, mars 1983, n° 65, la culture des π, p. 9.
[9] RIVEST, R. L., SHAMIR, A., ADLEMAN, L., 1978. A method for obtaining digital signatures and public key crypto-systems. *Com. A.C.M.*. vol. 21, pp. 120–126.
[10] SALAMIN, E., 1976. Computation of π Using Arithmetic-Geometric Mean. – *Mathematics of Computation* n° 30, pp. 565–570.
[11] SHANKS, D. and WRENCH, J. W., 1962. 'Calculations of π to 10000 decimals'. – *Mathematics of Computation* n° 16, pp. 76–99.