π au compte-gouttes

par Sylvain BARON

En juin 1997, les éditions Pour la Science-Belin ont publié un livre de Jean-Paul Delahaye : « **Le fascinant nombre π** ». Le trouvant en librairie cet été, j'avais immédiatement le sentiment de tenir LE livre de référence actuel sur π, englobant et dépassant tous ceux que ma curiosité naturelle m'avait fait acquérir sur ce sujet au cours des vingt-cinq ans passés. Le signe π (prononcé pi) est la première lettre du mot grec signifiant périmètre. Elle fut adoptée par Euler pour représenter le nombre 3,1415926535897... que l'on peut définir simplement comme le rapport de la longueur de la circonférence d'un cercle à la longueur de son diamètre. On démontre de façon presque élémentaire (Euclide, 300 avant J.C.) que, dans un espace euclidien, ce nombre ne dépend pas du rayon du cercle, et, de façon beaucoup moins simple (Ferdinand von Lindemann, 1882 après J.C.), que ce nombre est transcendant, c'est à dire qu'il n'est racine d'aucun polynôme à coefficients entiers.

Le calcul de **π** a occupé les hommes depuis plus de 5000 ans et ce n'est pas le moindre mérite du livre de J.P. Delahaye que de retracer cette histoire et donner des tableaux synthétiques (dates, noms et méthodes) sur les progrès accomplis. Bien que, depuis plus de 100 ans, la démonstration de Lindemann ôte toute idée de trouver une périodicité quelconque dans la suite des décimales de **π**, on n'a jamais cessé de les calculer en nombres de plus en plus grands. Malgré de fréquentes lectures de textes variés de vulgarisations mathématiques (*Quadrature* par exemple), j'en étais resté personnellement au record du million de décimales de Jean Guilloud du Commissariat à l'Énergie Atomique en 1973. Il faut dire que je ne dois pas consulter assez régulièrement le *Guiness book of records*, car Jean Guilloud, avec qui j'avais souvent discuté des techniques de ses performances, m'avait fait remarqué qu'il y figurait. Ses records successifs ont tenu 15 ans (de 1966 à 1981). Il avait d'ailleurs calculé un peu plus de décimales qu'il n'en annonçait pour ne pas risquer d'être détrôné par quelqu'un qui publierait un million et une décimale. Quand on lui demandait combien il en avait en réserve, il répondait malicieusement : « suffisamment ». On sait aujourd'hui qu'il en avait calculé 1 001 250. En 1981, le japonais Kanada publie 2 000 036 décimales de **π** suivi par Guilloud qui revient avec 2 000 050 mais qui jette l'éponge car de nouvelles formules de calcul apparaissent. Aujourd'hui le record est celui de Kanada, en octobre 1995, qui dépasse six milliards de décimales (6 442 450 938) !

Nous n'entrerons pas ici dans la discussion sur l'intérêt de tels calculs et nous renvoyons au livre de Delahaye qui développe très bien ce point. Notons que ce livre nous indique aussi les 19 sites Internet qui parlent du nombre **π** de façon à permettre à tous les curieux de se tenir régulièrement à jour.

L'article que nous publions ci-dessous, transcrit la conférence que nous avons faite aux Journées portes-ouvertes du 6 novembre 1997. Article et conférence donnent une version APL d'**un autre sujet d'étonnement que nous a procuré le livre de Delahaye : on connaît aujourd'hui un algorithme qui donne les décimales de π une à une**. A chaque boucle, une nouvelle décimale est produite, une par une, comme tombent, une à une, les gouttes d'eau d'un compte-gouttes que l'on presse avec régularité. Entendons nous, il ne s'agit pas d'un algorithme qui pourrait donner un nombre infini de décimale en laissant tourner dans un coin une machine pendant des millénaires. Les progrès pour obtenir six milliards de décimales ne résident pas là. L'algorithme compte-gouttes doit savoir au départ combien de décimales on veut obtenir et son encombrement en mémoire comme son temps de calcul sont comparables aux algorithmes utilisées, à l'époque, par Jean Guilloud. Mais la grande nouveauté est qu'une fois le choix fait de calculer disons 500 décimales, on les obtient progressivement au lieu de les obtenir toutes ensemble à la fin des calculs complets de tous les termes d'une série, chaque terme étant calculé avec 500 décimales. Cet étonnant progrès utilise une série d'Euler qui s'y prête bien et on aurait pu le découvrir depuis plus de 200 ans. L'idée a été trouvée en 1968 pour calculer le nombre **e** et mise au point tout récemment, en 1988, pour calculer **π**. Cet algorithme compte-gouttes permet de plus à n'importe qui de calculer à la main plusieurs dizaines de décimales de **π** en quelques heures et plusieurs centaines en quelques jours. Le procédé reste tout à fait fascinant. Il aurait épargné beaucoup de peine à tous les calculateurs vertueux des siècles passés qui travaillaient alors sans calculatrices ni ordinateurs.

Pour calculer n décimales il faut initialiser trois vecteurs à 3,32×n valeurs (arrondi à l'entier supérieur). Delahaye indique le premier programme compte-gouttes réalisé en C trouvé sur Internet pour un calcul de 2400 décimales de π. Le calcul est fait par paquet de 4 chiffres en base 10 000. Arrondi à l'entier supérieur on a $14 \cong 4 \times 3{,}32$ et, $600 \times 14 = 8400$ (pour $600 \times 4 = 2\,400$ décimales). Le programme record en C est le suivant en 158 caractères , où le *printf(%4d* sert à afficher les chiffres par groupe de 4 :

```
int a=10000,b,c=8400,d,e,f[8401],g ;main(){for( ;b-c ;)
f[b++]=a/5 ;for( ;d=0,g=c*2 ;c-=14,printf("%.4d",e+d/a),
e=d%a)for(b=c ;d+=f[b]*a,f[b]=d%--g,d/=g--,--b;d*=b) ;}
```

Là, nous devons nous incliner. APL ne fera pas moins. Cet algorithme impose deux boucles par essence. Elles sont irréductibles en APL. L'équivalent en APLWIN d'APL2000 (Ex-APL*PLUS de l'ex-STSC) s'écrit ainsi :

```
∇ r←decpi n;z;j;q;k
[1]   ⍝ Exécute n fois comptegoutte
[2]   init ⌈n×3.32
[3]   r←⍳j←0 ◇ q←d
[4]
[5]   x:→(n<j←j+1)⍴0
[6]   q←comptegoutte q
[7]   r←r,¯1↑q
[8]   k←1+⍴r
[9]   y:→(r[k←k-1]≠10)⍴x
[10]  r[k]←0 ◇ r[k-1]←1+r[k-1]
[11]  →y
      ∇

      ∇ init n
[1]   ⍝ initialise a b et d pour comptegoutte
[2]   ⎕pw←80
[3]   a←Φ0,⍳n
[4]   b←(¯1↓1+2×a),10
[5]   d←((⍴b)⍴2),0
[6]
      ∇

      ∇ r←comptegoutte d;i;f;rt;s
[1]   ⍝ Algorithme compte-gouttes (Delahaye,
  ⍝ page 99-101)
[2]   rt←(i←0)×f←10×d←¯1↓d
[3]
[4]   loop: →((⍴d)=i←i+1)⍴fin
[5]   rt[i+1]←a[i]×⌊(f[i]+rt[i])÷b[i]
[6]   →loop
[7]   fin:
[8]
[9]   s←f+rt
[10]  r←(b|s),⌊0.1×¯1↑s
      ∇

        decpi 12
3 1 4 1 5 9 2 6 5 3 5 8
```

Pour mieux comprendre le mécanisme, on peut **afficher les gouttes une à une** à l'aide de la fonction "**Affiche**" qui tient compte qu'une goutte qui vaut 10 vaut 0 mais modifie la goutte d'avant en lui ajoutant 1.

```
    ∇ r←decpi n;z;j;q;k
[1]   ⍝ Exécute n fois comptegoutte
[2]   init ⌈n×3.32
[3]   r←⍳j←0 ◇ q←d
[4]
[5]   x:→(n<j←j+1)⍴0
[6]   q←comptegoutte q
[7]   r←r,¯1↑q
[8]   Affiche r
[9]
[10]  ⍝ Ci-dessous, on corrige le cas
[11]  ⍝ où (10=¯1↑q) en 0 avec +1 sur
[12]  ⍝ la valeur qui précède.
[13]
[14]  k←1+⍴r
[15]
[16]  y:→(r[k←k-1]≠10)⍴x
[17]  r[k]←0 ◇ r[k-1]←1+r[k-1]
[18]  →y
    ∇
```

```
    ∇ r←Affiche pi;a;d;f;i;m
[1]   ⍝ affiche les décimales de pi
[2]   ⍝ quand elles sont sûres
[3]   ⍝ (pas de 9 suivi de 10)
[4]   ''
[5]   ''
[6]   ''
[7]   r←⍕pi ◇ →(2>⍴pi)⍴0
[8]   i←-1+(10=¯1↑pi)
[9]
[10]  r←(r≠' ')/r ◇ f←i↑r ◇ a←⍴r←i↓1↓r
[11]  a←50×⌈a÷50 ◇ m←55⍴(10⍴1),0
[12]  r←' ',' ',m\((((a÷50),50)⍴a↑r),[1]' '),[1]50↑f
[13]  r[1;1 2]←'3,'
    ∇
```

```
      p←decpi 15
3

3,
  1

3,1
  4
```

```
3,14
  1

3,141
  5

3,1415
  9

3,14159
  2

3,141592
  6

3,1415926
  5

3,14159265
  3

3,141592653
  5

3,1415926535
  8

3,1415926535 8
  9

3,1415926535 89
  7

3,1415926535 897
  9
```

Pour faire une goutte il **faut** boucler dans la fonction comptegoutte. Pour avoir n gouttes (n décimales) il **faut** boucler n fois sur la fonction comptegoutte. C'est ainsi. L'arithmétique avec n décimales a ses lois. Si on accepte de se contenter de 16 décimales ou moins, plus de boucles : l'interpréteur APL s'en occupe et l'algorithme compte-gouttes, **qui n'est qu'un simple changement de base** comme l'explique Delahaye, se fait directement en APL avec un seul encode !

Si on pose `k←⍳50` et `v←⌽k÷1+2×k` , l'algorithme compte-gouttes n'est qu'un prodigieux mécanisme pour effectuer pas à pas, sur des nombres immenses, le calcul suivant:

```
      v⊥(⍴v)⍴2
3.1415926535897928
```

En effet, la formule précédente n'est que la traduction de la formule d'Euler suivante:

$\pi = 2(1 + 1/3 + 1\times2/3\times5 + 1\times2\times3/3\times5\times7 + 1\times2\times3\times4/3\times5\times7\times9 + ...)$

Cette série d'Euler peut s'écrire :

$\pi = (2 + 1/3(2 + 2/5(2 + 3/7(2 + 4/9(2 + 5/11 ...)))))$

et de même en base 10, π s'écrit :

π =(3 + 1/10(1 + 1/10(4 + 1/10(1 + 1/10(5 + 1/10 ...)))))

Bref, en base 10, π s'écrit : 3 1 4 1 5 9 ... et, en base 1/3 1/5 1/7 1/9 1/11 ..., π s'écrit : 2 2 2 2 2 ...

Cette remarque jointe à l'algorithme inventé pour effectuer tout changement de base pas à pas permet d'obtenir π décimale après décimale. Delahaye indique la méthode manuelle pour monter un tableau de chiffres que l'on fait progresser ligne à ligne et son explication nous a permis de réaliser le modèle APL. Inversement les lecteurs de notre revue pourront lire l'APL pour tracer pas à pas le mécanisme et comprendre la méthode sous-jacente. Mais cela ne dispense personne que cette question intéresse d'acquérir le merveilleux livre de Jean-Paul Delahaye pour y apprendre l'incroyable histoire de frères Chudnovsky dont nous avons parlé dans notre conférence ou pour connaître les démonstrations de centaines de formules qui donnent π dont certaines en APL se traduisent ainsi:

```
      c←+/×\1,÷2+÷⍳50
      2×c
3.1415926535897932


      k←⍳n←120000
      (4÷n)×(1-(k÷n)⋆2)⋆0.5
3.141575959


    ∇ r←x Viete y
[1]   r←x+y⋆0.5
    ∇
    ×/2÷1,1↓Viete\0 ,27⍴2
3.1415926535897952


Leibnitz

    k←0,⍳10000
8×+/÷(1+4×k)×(3+4×k)
3.1415426585893244


Euler

    k←⍳100000
    (6×+/÷k⋆2)⋆0.5
3.1415831043264412


  Découverte de Simon Plouffe (septembre 1995)

a←8×i←0,⍳50
    +/((4÷a+1)-(2÷a+4)+(1÷a+5)+(1÷a+6))×16⋆-i
3.1415926535897932


Arctangente par la formule de John Machin
```

```
      k←0,⍳50
      j←-1+2×k
      c←(4×¯1★k)÷-j
      +/c×(4×5★j)-(239★j)
3.1415926535897936
```