1 Historique

Les premières traces que l'on trouve de l'existence de Pi remontent aux environs de 2000 ans av.J.-C. chez les Babyloniens et les Egyptiens. On a trouvé en effet en 1936 sur une tablette babylonienne la valeur approchée de 3+1/7. Sur le papyrus de Rhind conservé au British Museum π est approximé par $(\frac{16}{9})^2$ soit environ 3,16...

C'est Archimède (-287,-212) qui fait le premier un véritable calcul mathématique pour déterminer une valeur approchée de π. Archimède approche la circonférence du cercle par les périmètres de polygones réguliers inscrits et exinscrits et donne un bon encadrement de π entre 3,1410 et 3,1428 en utilisant 96 côtés. En fait, le nombre d'Archimède, π , n'avait pas encore le statut de nombre : Archimède entendait là le rapport L/d du périmètre du cercle à son diamètre.

Archimède considère pour cela un cercle de rayon 1 et dessine à l'intérieur et à l'extérieur deux polygones réguliers à 3×2^n côtés. Notons a_n le demi-périmètre du polygone circonscrit et b_n celui du polygone inscrit. Par des considérations géométriques on montre que pour n=1, cas de l'hexagone on a $a_1 = 2\sqrt{3}$ $b_1 = 3$. Ensuite pour tout n : $\frac{1}{a_n} + \frac{1}{b_n} = \frac{2}{a_{n+1}}$ et $b_n \times a_{n+1} = b_{n+1})^2$. On peut en utilisant ces formules de récurrence encadrer π avec la précision souhaitée.
Cette méthode ne donne pas une suite qui converge rapidement vers π : on gagne mois d'un chiffre décimal à chaque étape (3 chiffres toutes les 5 étapes).

La naissance de l'analyse moderne (calcul intégral et différentiel) permet de nouvelles définitions de π qui ne sont pas issues de la géométrie. John Wallis (1616-1703) trouva la formule de produit infini suivante : $\pi = 2 \times \frac{2\times2}{1\times3} \times \frac{4\times4}{3\times5} \times \frac{6\times6}{5\times7}...$

$$\pi = lim_{n\to+\infty} \frac{2^{4n} \times n!^4}{n(2n)!^2}$$

Jolie formule mais guère utile pour calculer π, la convergence est particulièrement lente.
C' est à James Gregory (1638-1675) que l'on doit la formule de décomposition de l'arc tangente qui est est la base du calcul de π :

$arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7}... = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} x^{2k+1}$.

Leonhard Euler (1707-1783), un des plus grands mathématicien, nous a fourni de trés nombreuses formules.

Parmi celles-ci on a la belle formule :
$\pi = 2(1 + \frac{1}{3} + \frac{1\times2}{3\times5} + \frac{1\times2\times3}{3\times5\times7} + \frac{1\times2\times3\times4}{3\times5\times7\times9}...$
Nous démontrerons cette formule avant de l'utiliser pour calculer π avec l'algorithme du compte-gouttes.
On doit aussi à Euler d'avoir pérénisé la notation π qui est la première lettre du mot grec $\Pi\epsilon\rho\iota\mu\epsilon\theta\rho o\Theta$ signifiant périmètre.

La connaissance de π que donne l'analyse moderne conduit à des méthodes de calcul efficaces .

John Machin (1680-1752), grâce à sa formule est le premier à acccéder à la centième décimale .

Les principaux mérites de ses sucesseurs furent la patience et la persévérance. Ce sera le cas de William Shanks (1812-1882) , qui passa 20 années à calculer 707 décimales. Elles sont inscrites en spirale au Palais de la découverte. L'arrivée des ordinateurs aprés la deuxième guerre mondiale, provoqua une révolution chez les chasseurs de décimales . En 1973 Jean Guilloud et Martine Bouyer furent les premiers à atteindre un million de décimales.

2 démonstration de la formule d'Euler $\pi = 2\sum_{n=0}^{\infty} \frac{2^n(n!)^2}{(2n+1)!}$

C'est une astuce d'Euler qui permet un développement en série trés inhabituel de z=arctan(t) :

$$arctan(t) = \frac{t}{1+t^2}[\sum_{n=0}^{\infty} \frac{n!^2 2^{2n}}{(2n+1)!}(\frac{t^2}{1+t^2})^n]$$

pour t positif. Euler l'a utilisée pour calculer 20 décimales de π. en effet $arctan(\pi) = 20arctan(\frac{1}{7}) + 8arctan(\frac{3}{79})$.
Pour s'en convaincre calculons :$(7+i)^{20}(79+3i)^8$. En développant on obtient :
Pour $t = \frac{1}{7}$ $\frac{t^2}{1+t^2} = 0.02$ et $t = \frac{3}{79}$ $\frac{t^2}{1+t^2} = 0.00144$ ce qui est trés pratique pour les calculs.

2.1 faisons le changement de variable $x = \frac{t^2}{1+t^2}$ pour t positif x varie dans [0 ;1[

On a $t^2 - x(1+t^2) = 0$ puis $t^2(1-x) = x$ donc $t = \sqrt{(\frac{x}{1-x})}$.

Posons maintenant $y = \frac{1+t^2}{t} z = \frac{1+t^2}{t} arctan(t) = \frac{1+\frac{x}{1-x}}{\sqrt{\frac{x}{1-x}}} arctan(t)$
$y = \frac{1-x+x}{(1-x)\sqrt{\frac{x}{1-x}}} arctan(t) = \frac{1}{\sqrt{\frac{x(1-x^2)}{1-x}}} arctan(t) = \frac{1}{\sqrt{x-x^2}} arctan(t)$
Donc $y = \frac{1}{\sqrt{x-x^2}} arctan\sqrt{\frac{x}{1-x}}$
$y = \frac{1}{\sqrt{x-x^2}} z$.

2.2 Simplifions maintenant l'écriture de $z = arctan\sqrt{\frac{x}{1-x}}$

On a : $\frac{x}{1-x} = tan^2(z) = \frac{sin^2(z)}{cos^(z)} = \frac{sin^2(z)}{1-sin^2(z)}$. Par suite :
$(1-x)sin^2(z) = (1-sin^2(z))x$ puis : $sin^2(z)(1-x+x) = x$ donc $sin^2(z) = x$
, $sin(z) = \sqrt{x}$.
$z = arcsin(\sqrt{x})$ et finallement : $y = \frac{1}{\sqrt{x-x^2}} arcsin(\sqrt{x})$.

2.3 Cherchons maintenant le développement en série entière de y avec une équation différentielle.

Dérivons la formule x : $y = \frac{1}{\sqrt{x-x^2}} arcsin(\sqrt{x})$ par rapport à x.

Nous obtenons : $y' = \frac{2x-1}{2(x-x^2)\sqrt{x-x^2}} arcsin(x) + \frac{1}{\sqrt{x-x^2}} \frac{\frac{1}{2\sqrt{x}}}{\sqrt{1-x}}$
Soit : $y' = -\frac{1-2x}{2(x-x^2)} y + \frac{1}{2(x-x^2}$ donc :
$2y'(x-x^2) + (1-2x)y = 1$
Cherchons y sous la forme : $y = \sum_{n=0}^{\infty} a_n x^n$. $y' = \sum_{n=0}^{\infty} n a_n x^{n-1}$. Par suite les coefficients a_n vérifient :
$\sum_{n=1}^{\infty} 2n a_n x^{n-1}(x-x^2) + \sum_{n=0}^{\infty} a_n x^n (1-2x) = 1$
En regroupant les termes on obtient :
$a_0 + \sum_{n=1}^{\infty} x^n [(2nan + an) - 2(n-1)a_{n-1} - 2a_{n-1}] = 1$
$a_0 + \sum_{n=1}^{\infty} x^n (a_n(2n+1) - 2n a_{n-1}) = 1$
Identifions les termes des deux membres de l'équation :
$a_0 = 1$, $3a_1 - 2a_0 = 0$ ...$(2n+1)a_n - 2n(a_{n-1} = 0$. Nous obtenons : $a_0 = 1$, $a_1 = \frac{2}{3}$, $a_n = a_{n-1} \frac{2n}{2n+1}$
Par récurrence on trouve : $a_n = \frac{n!^2 2^{2n}}{(2n+1)!}$

Donc :

$$y = \sum_{n=0}^{\infty} \frac{n!^2 2^{2n}}{(2n+1)!} x^n$$

Cherchons le rayon d convergence de cette série entière. Pour cela nous allons utiliser la règle de d'Alembert : Si $\lim|\frac{a_{n+1}}{a_n}| = l$ alors le rayon R de convergence de la série $\sum_{n=0}^{\infty} a_n x^n$ est $R = \frac{1}{l}$.

Ici nous avons : $a_n \approx \frac{(\frac{n}{e})^{2n} 2\pi n 2^{2n}}{(\frac{2n+1}{e})^{2n+1}\sqrt{2\pi(2n+1)}}$ en utilisant la formukle de Stirling :
$n! \approx (\frac{n}{e})^n\sqrt{2\pi n}$.
Simplifions : $a_n \approx \frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{2n+1}}(\frac{2n}{2n+1})^{2n}\frac{n}{2n+1} \approx \frac{1}{4\sqrt{\pi}}\frac{1}{\sqrt{n}}$
Par suite $\frac{a_{n+1}}{a_n}| \approx \sqrt{\frac{n+1}{n}}$ donc $\lim|\frac{a_{n+1}}{a_n}| = 1$, donc $R = 1$ donc la série :

$$y = \sum_{n=0}^{\infty} \frac{n!^2 2^{2n}}{(2n+1)!} x^n$$

converge sur [0 ;1[
On a donc le développement suivant de y en remplaçant x par $\frac{t^2}{1+t^2}$
$y = \sum_{n=0}^{\infty} \frac{n!^2 2^{2n}}{(2n+1)!}(\frac{t^2}{1+t^2})^n$, puis en revenant à arctan(t) sachant que : $y = \frac{1+t^2}{t} arctan(t)$
$arctan(t) = \frac{t}{1+t^2} y = \frac{t}{1+t^2}\sum_{n=0}^{\infty} \frac{n!^2 2^{2n}}{(2n+1)!}(\frac{t^2}{1+t^2})^n$.
Cette série converge maintenant sur R tout entier.

2.4 Formule d'Euler.

Il suffit maintenant de prendre pour t la valeur 1 pour avoir la formule d'Euler.
$\frac{\pi}{4} = arctan(1) = \frac{1}{2}\sum_{n=0}^{\infty} \frac{n!^2 2^{2n}}{(2n+1)!}\frac{1}{2^n}$. D'où :

$$\pi = 2\sum_{n=0}^{\infty} \frac{2^n (n!)^2}{(2n+1)!}$$

3 algorithme du compte-gouttes

Rappelons l'écriture de π grâce à la formule d'Euler que nous venons de voir.

$$\pi = 2\sum_{n=0}^{\infty} \frac{2^n (n!)^2}{(2n+1)!}$$

$$\pi = 2\sum_{n=0}^{\infty} \frac{1 \times 2 \times 3 \times 4 ... \times n}{3 \times 5 \times 7 \times 9 ... \times (2n+1)}$$

aprés simplification.
ou encore $\pi = 2 + 2\frac{1}{3} + 2\frac{1}{3}\frac{2}{5} + 2\frac{1}{3}\frac{2}{5}\frac{3}{7} + 2\frac{1}{3}\frac{2}{5}\frac{3}{7}\frac{4}{9}...$ Ecrivons ceci en utilisant l'algorithme d'Horner qui utilise moins de multiplications.

$$\pi = 2 + \frac{1}{3}(2 + \frac{2}{5}(2 + (\frac{3}{7}(2 + \frac{4}{9}(2 + \frac{5}{11}(2 + \frac{6}{13}(.... + \frac{n}{2n+1}))))))$$

Cette écriture est à rapprocher de l'écriture décimale de π :
$\pi = 3,141592653589... = 3+\frac{1}{10}(1+\frac{1}{10}(4+\frac{1}{10}(1+\frac{1}{10}(5+\frac{1}{10}(9+\frac{1}{10}(2+\frac{1}{10}(...))))))$
Dans notre base dix le pas $\frac{1}{10}$ qui permet de passer aux différents digits de l'écriture est constant. Notons cette base $[\frac{1}{10}, \frac{1}{10}, \frac{1}{10}...]$ En comparant avec la série d'Euler nous dirons que dans la base à pas variable $[1, \frac{1}{3}, \frac{2}{5}, \frac{3}{7}, \frac{4}{9}...]$ π s'écrit : $[2, 2, 2, 2, 2, ...]$ Le problème est de construire un algorithme qui opère le changement de base. C'est l'objet de l'algorithme du compte-gouttes. Initialement c'est A. Sale en 1968 qui a eu l'idée de ce procédé et l'a utilisé pour calculer e.

3.1 nombre de décimales

La série $s_n = \sum_{k=0}^{k=n} 2+\frac{1}{3}(2+\frac{2}{5}(2+(\frac{3}{7}(2+\frac{4}{9}(2+\frac{5}{11}(2+\frac{6}{13}(....+\frac{k}{2k+1}))))))) = 2\sum_{k=0}^{k=n} \frac{1\times2\times3\times4...\times k}{1\times3\times5...\times(2k+1)}$ s'obtient en sommant les termes de la suite $u_n = \frac{1\times2\times3\times4...\times n}{1\times3\times5...\times(2n+1)}$.
Les termes u_n sont tous positifs et s_n converge vers π en croissant.
Le rapport de termes consécutifs $\frac{u_n}{u_{n-1}} = \frac{n}{2n+1}$ est inférieur à $\frac{1}{2}$. L'erreur commise en s'arrêtant au terme u_n est donc inférieure à : $\frac{u_n}{2} + \frac{u_n}{2^2} + \frac{u_n}{2^3}.... = \sum_{k=1}^{\infty} \frac{a_n}{2^k} = \frac{a_n}{2}\frac{1}{1-1/2} = a_n$ et $a_n \leq \frac{1}{2^n}$

Par conséquent pour obtenir k décimales de π c'est à dire avec une précision de 10^{-k} on doit aller jusqu'a l'indice n de u_n tel que :
$\frac{1}{2^n} \leq 10^{-k}$ soit $n \geq k\frac{ln(10)}{ln(2)}$ ou encore $n \leq k \times 3,32$. Par exemple pour obtenir 14 chifres pour π il faut aller jusqu'a : $n = 3,32 \times 14$ soit u_{47} , c'est à dire sommer 47 termes.
La convergence n'est pas trés rapide mais par contre l'algorithme est particulièrement simple. C'est là tout son intérêt.

3.2 changement de base

Avant d'expliquer l'algorithme du compte-goutte qui est un algorithme de changement de base spécial, décrivons l'algorithme du compte-goutte qui convertit un nombre écrit en base 3 en son écriture décimale. Ceci permettra de comprendre comment cela se passe pour π.
Soit un base B, c'està dire un nombre entier. Un nombre x ayant une partie entière inérieure à B s'ècrit dans la base B de la manière suivante :

$x = x_0 + \frac{x_1}{B} + \frac{x_2}{B^2} + \frac{x_3}{B^3} + ...$. x_i est un entier inférieur à B. C'est le ième caractère après la virgule de l'écriture de x dans la base B . Prenons x=3,1415 en base dix cela signifie bien évidemment : $x = 3 + \frac{1}{10} + \frac{4}{10^2} + \frac{1}{10^3} + \frac{5}{10^4}$.
Soit alors $x = 0,1024_3$ l'écriture en base 3 d'un nombre x. $x = 0 + \frac{1}{3} + \frac{0}{3^2} + \frac{2}{3^3} + \frac{1}{3^4}$
Utilisons la décomposition de Horner pour diminuer le nombre de multiplications. $x = \frac{1}{3}(1 + \frac{1}{3}(0 + \frac{1}{3}(2 + \frac{1}{3} \times 1)))$ Nous allons multiplier x par 10 pour faire sortir son premier chiffre après la virgule en base 10.
$10x = \frac{10}{3} + \frac{0}{3^2} + \frac{20}{3^3} + \frac{10}{3^4}$ soit avec Horner :
$10x = \frac{1}{3}(10 + \frac{1}{3}(0 + \frac{1}{3}(20 + \frac{1}{3}10)))$ et faisons la division euclidienne de 10 par 3 $10 = 3 \times 3 + 1$

$10x = \frac{1}{3}(10 + \frac{1}{3}(0 + \frac{1}{3}(20 + 3 + \frac{1}{3})))$ et faisons la division euclidienne de 23 par 3 $23 = 3 \times 7 + 2$

$10x = \frac{1}{3}(10 + \frac{1}{3}(0 + 7 + \frac{1}{3}(2 + \frac{1}{3}(1))))$ et faisons la division euclidienne de 7 par 3 $7 = 3 \times 2 + 1$

$10x = \frac{1}{3}(10 + 2 + \frac{1}{3}(1 + \frac{1}{3}(2 + \frac{1}{3}(1))))$ et faisons la division euclidienne de 12 par 3 $12 = 3 \times 4 + 0$

$10x = 4 + \frac{1}{3}(0 + \frac{1}{3}(1 + \frac{1}{3}(2 + \frac{1}{3}(1))))$.

Nous obtenons $10x = 4 + r$ 4 plus un reste r. $r = (0,0121)_3$ ou encore $r = \frac{0}{3} + \frac{1}{9} + \frac{2}{27} + \frac{3}{81}$ $r < 1$. Donc la partie entière de 10x est x. Comme le reste est inférieur à 1 le premier chiffre après la virgule de x est 4. Il suffit de recommencer le processus précédent avec le reste r obtenu pour convertir ainsi x en base 10.
Pour que l'on obtienne ainsi l'écriture décimale de x il suffit de démontrer que le reste obtenu à chaque itération est bien nul.
Démontrons donc que $r = \frac{2}{3} + \frac{2}{3^2} + \frac{2}{3^3}...$ est toujours inférieur à 1, car c'est le maximum que l'on puisse obtenir. On reconnait la somme des termes d'une suite géométrique . D'où : $r < \frac{2}{3}\frac{1}{1-1/3}$. On a bien $r < 1$.
On obtiendra donc bien à chaque étape le digit de x recherché.
On peut faire les calculs à la main dans le tableau ci-dessous.
Voici la première étape.

A		1	1	1	1
B		3	3	3	3
x	0	1	0	2	1
×10		10	0	20	10
retenue	4	2	7	3	0
somme		12	7	23	10
reste	0	0	1	2	1

Le procédé est le suivant :
1) Multiplier chaque caractère de l'écriture de x par la 10 et ajouter le résultat à la retenue. (La première retenue à droite est 0)
2) Partant de la droite faire la division euclidienne de la somme obtenue par la base 3. Le reste est mis dans la colonne considérée et dans la ligne reste.
3) Le quotient est multiplié par le 1 de la ligne A puis il est mis dans la ligne retenue de la colonne précédente.
4) On recommence en se déplaçant d'un rang à gauche jusqu'à atteindre la première colonne. La retenue sortante est le digit à récupérer.
5) On recommence un nouveau tableau en remplaçant x par le reste obtenu.

3.3 programme de conversion

voici le programme pour calculatrice ti92.
Prgm
ClrIO
$10 \rightarrow a$
$5 \rightarrow c$
$0 \rightarrow d$
$newList(5) \rightarrow f$
$3 \rightarrow g$
$0 \rightarrow f[1] : 1 \rightarrow f[2] : 0 \rightarrow f[3] : 2 \rightarrow f[4] : 1 \rightarrow f[5]$
For $j, 4, 1, -1$
For $b, c, 2, -1$
$f[b] * a + d \rightarrow d$
$mod(d, g) \rightarrow f[b]$
$int(d/g) \rightarrow d$
EndFor
Disp d
$0 \rightarrow d$
EndFor
EndPrgm
Voici maintenant le même en C.

```
#include <stdio.h>
long a=10,b,c=4,i=1,d=0,f[5],g=3;
 void main()
{
f[0]=0;f[1]=1;f[2]=0;f[3]=2;f[4]=1;
while (i<8)
{ b=4;
while (b>0)
{ d+=f[b]*a;f[b]=d%g;d/=g ;
```

```
b--;}
printf("%ld",d);
d=0;
i++;
}
 }

Nous obtenons alors le résultat:\
C:\PI>base1
4197530
```

Vérifions en calculant à la calculatrice $\frac{1}{3} + \frac{2}{27} + \frac{1}{81} = 0,41975308642$

3.4 description de l'algorithme pour π

Voici le tableau représentant les calculs pour les premiers digits .

A	π	r	1	2	3	4	5	6	7	8	9	10	11
B	=		3	5	7	9	11	13	15	17	19	21	23
initiali		2	2	2	2	2	2	2	2	2	2	2	2
$\times 10$		20	20	20	20	20	20	20	20	20	20	20	20
retenue		10	12	12	12	10	12	7	8	9	0	0	0
somme	3	30	32	32	32	30	32	27	28	29	20	20	20
reste		0	2	2	4	3	10	1	13	12	1	20	20
$\times 10$		0	20	20	40	30	100	10	130	120	10	200	200
retenue		13	20	33	40	65	48	98	88	72	150	132	96
somme	1	13	40	53	80	95	148	108	218	192	160	332	296
reste		3	1	3	3	5	5	4	8	5	8	17	20
$\times 10$		30	10	30	30	50	50	40	80	50	80	170	200
retenue		11	24	30	40	40	42	63	64	90	120	88	0
somme	4	41	34	60	70	90	92	103	144	140	200	258	200
reste		1	1	0	0	0	4	12	9	4	10	6	16
$\times 10$		10	10	0	0	0	40	120	90	40	100	60	160
retenue		4	2	9	24	55	84	63	48	72	60	66	0
somme	1	14	12	9	24	55	124	183	138	112	160	126	160
reste		4	0	4	3	1	3	1	3	10	8	0	22

Expliquons l'algorithme. On reconnait aux deux premières lignes les numérateurs et dénominateurs des pas de la base à pas variable. A la troisième ligne , initialisation, on place l'écriture de π ans cette base. On la multiplie par 10 pour obtenir 10π dans la base à pas variable , mais dans cette base le digit correspondant ne doit pas dépasser le dénominateur : (le nombre écrit dans la ligne de B juste au dessus) . C'est la même situation que dans la base dix ou les chiffres sont de 0 à 9. Pour obtenir l'écriture normalisée dans la base à pas variable , nous devons donc faire la division euclidienne par le nombre

correspondant de la ligne de B , ce qui donnera un quotient q qu'il faudra multiplier par le nombre de la ligne de A placé dans la colonne précédente et un reste que l'on conserve en le mettant dans la ligne reste. Expliquons en détail cette situation qui se produit pour la première fois (avec une retenue nonnulle) dans la colonne située à droite des caractères gras.
$\pi = ...(20 + \frac{8}{17}(20 + \frac{9}{19}(20...$ on a par la division euclidienne de 20 par 19 : $20 = 1 \times 19 + 1$ q=1, r=1. On doit donc effectuer : $\frac{9}{19}(1 \times 19 + 1)$. Le reste 1 est mis dans la ligne du reste juste en dessous , le quorient q=1 doit lui être multiplié par le 9 du $\frac{9}{19}$. Il joue le rôle d'une retenue. On le met donc dans la ligne retenue, située dans la colonne précédente . C'est le 9 en caractère gras ,la première retenue non nulle, située dans la ligne retenue du premier sous tableau , en partant de la droite et en se déplaçant vers la gauche bien entendu.
On procède donc de la droite vers la gauche de la manière suivante :(on prend comme exemple la colonne avec les caratères gras)
1) On remplit la ligne $\times 10$ en multipliant la ligne précédente par 10.
2) On place dans la ligne somme la somme de la ligne $\times 10$ avec la ligne retenue soit : 20+9=29
3) On effectue la division euclidienne de somme par le nombre situé dans la ligne B et la même colonne soit :$29 = 17 \times 1 + 12$
4) On place le reste 12 dans la ligne" reste" et dans la même colonne. Ensuite , on multiplie le quotient 1 par le nombre précédent de la ligne A et on met le résultat dans la ligne "retenue" précédente : $1 \times 8 = 8$
5) On recommence le processus dans la nouvelle colonne et on progresse de la droite vers la gauche.
On obtient 30 comme dernière somme.
On obtient donc à la dernière ligne le développement de 10π dans la base à pas variable : $10\pi = 30 + \frac{1}{3}(2 + \frac{2}{5}(3 + \frac{3}{7}...)))$.
Autrement dit $10\pi - 30 = \frac{1}{3}(2 + \frac{2}{5}(3 + \frac{3}{7}...)))$. Majorons cette écriture pour montrer que 3 est le début du développement de π.

Vu l'écriture normalisée dans la base à pas variable les digits de l'écriture sont inférieurs au dénominateur du pas. Donc on a :
$10\pi - 30 \leq 2\frac{1}{2} + 4(\frac{1}{2})^2 + 6(\frac{1}{2})^3 + ... + 2n(\frac{1}{2})^n$ en majorant chaque terme du quotient par $\frac{1}{2}$. Soit en mettant 2 en facteur :
$10\pi - 30 \leq 2(1 + \frac{1}{2} + 2(\frac{1}{2}) + 3(\frac{1}{2})^2 + ...n(\frac{1}{2})^{n-1}$. Or la somme $\sum_{k=1}^{\infty} kx^{k-1}$ s'obtient en dérivant :
$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$. On obtient donc la majoration pour $x < 1$: $\sum_{k=1}^{\infty} kx^{k-1} \leq \frac{1}{(1-x)^2}$ donc $10\pi - 30 \leq 4$
Par suite $\pi - 3 < 0,4$, 3 est bien le premier chiffre de l'écriture décimale de π.

On poursuit ce principe au tableau suivant. L'écriture de 10π dans la base à pas variable est remplaceé par celle $10(\pi - 3)$ dans la nouvelle ligne

$\times 10$.
Le dernier r obtenu est 13 qui nous donne le premier chiffre de l'écriture décimale de π. Remarquons que notre majoration précédente suffit ici à le démontrer car le développement peut être majoré par 3+4 . Le problème se pose lorsque l'on a l'unité de r qui dépasse 6.
Nous verrons par la suite qu'effectiement, parfois on n'obtient pas le digit cherché. Des retenues ultérieures vont le modifier.

Nous allons faire un premier programme qui correspond point par point à ce calcul . Nous verrons alors qu'il se pose parfois le problème suivant :
La "décimale" qui sort i en bout de colonne peut dépasser 10 et donc modifier la décimale précédente. Nous verrons ensuite comment résoudre cette question.

3.5 programme compte-goutte imparfait

Voici le programme correspondant exactement à ce que l'on a fait à la main au dessus.(pour la ti92).
Prgm
Local u,v
ClrIO
$10 \rightarrow$ a
$200 \rightarrow c$
$0 \rightarrow u : 10 \rightarrow v$
$1 \rightarrow b$
$0 \rightarrow e$
$newList(c) \rightarrow f$
For b,1,c
$2 \rightarrow f[b]$
EndFor
While $c > 0$
$2 * c \rightarrow g : 0 \rightarrow d : c \rightarrow b$
$Whileb > 0$
$d + f[b] * a \rightarrow d$
$g - 1 \rightarrow g$
$mod(d, g) \rightarrow f[b]$
$int(d/g) \rightarrow d : g - 1 \rightarrow g : b - 1 \rightarrow b$
$Ifb \neq 0$ Then
$d * b \rightarrow d$
EndIf
EndWhile
$c - 4 \rightarrow c$
If $u + v = 10$ then

```
Output  0,0,string(e+int(d/a))&"."\\
Else\\
Output  u,v,string(e+int(d/a))\\
```
EndIf
$v + 10 \rightarrow v$
$If v > 150 Then$
$10 \rightarrow v$
$10 + u \rightarrow u$
EndIf
$mod(d, a) \rightarrow e$
EndWhile
EndPrgm

On voit bien à la troisième ligne le digit 10 télescopé par le digit suivant. On aurait du obtenir :3,14159265358979323884626433832795 02...

3.6 théorème

Le nombre x dont l'écriture dans la base $c = (1; \frac{1}{3}, \frac{2}{5}, \frac{3}{7}, \frac{4}{9}, ..., \frac{n}{2n+1}, ...)$ est $x = (0; 2, 4, 6, 8, .., 2n, ...)$ est 2.
Par conséquent tout nombre y ayant une représentation régulière de la forme $y = (0; a_1, a_2, a_3, ...)$ dans la base c est entre 0 et 2 : $0 \leq y < 2$. On entend par régulière une représentation de y dans laquelle pout tout i a_i est un entier inférieur au dénominateur du pas :$2i + 1$.
FDémonstration.
Montrons que $x = \sum_{i=1}^{\infty} \frac{(2i)i!}{(2i+1)!!} = 2$. Dans cette écriture $(2i + 1)!! = 1 \times 3 \times 5 \times 7 \times ... \times (2i + 1)$.
Une écriture commode de $(2i + 1)!!$ est : $(\frac{(2i+1)!}{i! \times 2^i}$
Par exemple $1 \times 3 \times 5 \times 7 \times \times 9 = \frac{1 \times 3 \times 5 \times 7 \times 9 \times 2 \times 4 \times 8}{2 \times 1 \times 2 \times 2 \times 2 \times 3 \times 2 \times 4} = \frac{9!}{4! \times 2^4}$.
Notons u_n la somme partielle $u_n = \sum_{i=1}^{i=n} \frac{(2i)i!}{(2i+1)!!} = 2 \times \frac{1}{3} + 4 \times \frac{2 \times 1}{5 \times 3} + ... + 2n \frac{n!}{(2n+1)!!}$
u_n est une suite croissante. Montrons quelle est majorée par 2 et qu'elle converge vers 2. Pour cela calculons $2 - u_n = r_n$.
$r_1 = 2 - \frac{2}{3} = \frac{4}{3}$; $r_2 = \frac{4}{3} - \frac{4 \times 2}{3 \times 5} = \frac{4}{5}$; $r_3 = \frac{4}{5} - 6 \times \frac{3 \times 2 \times 1}{7 \times 5 \times 3} = \frac{16}{35}$; $r_4 = \frac{16}{63}$
Autrement écrit on a : $r_1 = 2 \times \frac{2!}{3!!}$; $r_2 = 2 \times \frac{3!}{5!!}$; $r_3 = 2 \times \frac{4!}{7!!}$; $r_4 = 2 \times \frac{5!}{9!!}$.
Soit donc l'hypothèse de récurrence : $r_n = 2 \frac{(n+1)!}{(2n+1)!!}$.
Supposons cette hypothèse vraie à l'ordre n-1 : $r_{n-1} = 2 \frac{n!}{(2n-1)!!}$ alors on a :
$r_n = r_{n-1} - \frac{(2n)n!}{(2n+1)!!}$ (on enlève le terme suivant de la série u_n).
$r_n = \frac{2 \times n!}{(2n-1)!!} - \frac{(2n)n!}{(2n+1)!!}$
$r_n = \frac{2 \times n!(2n+1) - (2n)n!}{(2n+1)!!} = \frac{n!(4n+2-2n)}{(2n+1)!!} = \frac{n!(2n+2)}{(2n+1)!!} = 2 \frac{n!(n+1)}{(2n+1)!!} = \frac{2 \times (n+1)!}{(2n+1)!!}$.
Il s'en suit que l'hypothèse de récurrence est vraie à l'ordre n. Elle donc vraie pour tout n. Donc r_n est positif et donc 2 majore u_n.
Calculons la limite de r_n en utilisant la formule de STIRLING :$n! \approx (\frac{n}{e})^n \sqrt{2\pi n}$
$r_n = \frac{2(n+1)!}{(2n+1)!!} = \frac{2(n+1)!n!2^n}{(2n+1)!}$

Par suite : $r_n \approx \frac{2(\frac{n+1}{e})^{n+1} \sqrt{2\pi(n+1)} (\frac{n}{e})^n \sqrt{2\pi n}}{(\frac{2n+1}{e})^{2n+1} \sqrt{2\pi(2n+1)}}$

$r_n \approx \frac{\sqrt{4\pi}(n+1)^{n+1} n^n \sqrt{(n+1)n}}{(2n+1)^{2n+1} \sqrt{2n+1}}$.

$r_n \approx 2\sqrt{\pi} \frac{n^{n+1} n^n}{(2n)^{2n+1}} \frac{n}{(2n)^{0,5}}$
$r_n \approx \sqrt{2\pi} \frac{n^{2n+1} n^{0,5}}{2^{2n+1} n^{2n+1}} \approx \sqrt{2\pi} \frac{n^{0,5}}{2^{2n+1}}$

Par conséquent $\lim r_n = 0$. C'est ce qu'il fallait démontrer.

3.7 conséquence du théorème

La représentation d'un nombre dans la base $c = (1; \frac{1}{3}, \frac{2}{5}, \frac{3}{7}, \frac{4}{9}, ..., \frac{n}{2n+1}, ...)$ n'est pas unique. Par exemple :
$2 - \frac{2}{3} = \frac{4}{3}$ est représenté par : $(0; 2, 4, 6, 8, .., 2n, ..) - (0; 2, 0, 0, 0...)$. Donc $\frac{4}{3}$ est représenté par $(0; 0, 4, 6, 8, .., 2n, ..)$.
Par suite $\frac{2}{3}$ est représenté par $(0; 0, 2, 3, 4, ..., n, ..)$, mais aussi par $0; 2, 0, 0, 0, ...)$.
Une conséquence de tout ceci est que la partie entière de $y = (a_0; a_1, a_2, ...)$ est soit a_0 soit $a_0 + 1$ selon que le nombre $(0; a_1, a_2, ...)$

est dans [0 ;1[ou dans [1 ;2[.
Par suite dans notre algorithme du compte-gouttes le digit a_0 obtenu en fin de boucle n'est pas obligatoirement celui de π. On va donc conserver ce digit en mémoire pour l'itération suivante et l'appeler prédigit afin de pouvoir l'augmenter.

3.8 problème de la retenue

Nous avons vu que la partie entière de $(a_0; a_1, a_2, a_3, ...)$ était a_0 ou a_0+1 selon que le nombre $(0; a_1, a_2, ...)$ est dans [0 ;1[ou dans [1 ;2[. Le digit a_0 sortant en fin d'itération n'est pas forcément le bon digit de π. Il faudra peut être l'augmenter de 1. Nous l'appelons prédigit et le gardons temporairement en mémoire. Si à l'itération suivante nous obtenons une retenue de 1 nous l'augmentetrons de 1, mais s'il est 9 son augmentation de 1 va augmenter le digit précédent. Par suite les prédigits égaux à 9 doivent être gardés avec le prédigit qui les précèdent. Cela oblige à faire une queue de prédigits avec les 9 qui se suivent précédés d'un autre prédigit afin de pouvoir être occasionnellement augmentés. Si lors de la prochaine itération on obtient ni un 9 ni une retenue on pourra dire que ce sont les digits véritables de π.

3.9 algorithme du compte-gouttes

1) Initialisations : Soit A=(2 ;2,2,2...) un tableau de taille int(3,4n)=p , n étants le nombre de décimales de π que l'on désire.q=retenue =0.
2) Faire n fois la procédure suivante :
2,1) pour i=p jusqu'à 1 par pas de -1 faire :
$x = 10 \times A(i) + q$; A(i)=reste de la division euclidienne de x par (2i+1) ; q= quotient de la division euclidienne de x par (2i+1) ;
$q = q \times i$ c'est la nouvelle retenue mise un cran à gauche.
2,2) On sort ici de la boucle qui nous donne le prédigit.
Réduire la valeur A(1) du tableau modulo 10 (c'est la dernière valeur du

tableau que l'on vient d'obtenir)
q=int(A(1)/10) : q est le nouveau prédigit ; A(1)=reste de (A(1)/10) gardé dans le tableau.
2,3) traitement des prédigits :
Si q n'est ni 9 ni 10 l'ancien prédigit est bien le digit de π. Il est affiché.
Si q=9 mettre q dans la queue des prédigits.
Si q=10 le nouveau prédigit est 0. On augmente tous les anciens prédigits de 1, ils deviennent alors les véritable digits de π et sont affichés.

3.10 programme

code en Pascal

```
const n=1000 ;
len=10*n div 3 ;
var i,j,k,q,x,nines,predigit :integer ;
a :array[1..len] of longint ;
begin
for j :=1 to len do a[j]=2 ;
nines :=0 ;predigit :=0 ;
for j=1 to n do
begin q :=0 ;
for i := len down to 1 do
begin
x :=10*a[i]+q*i ;
a[i] :=x mod(2*i-1) ;
q :=x div(2*i-1) ;
end ;
a[1] :=q mod 10 ; q :=q div 10 ;
if q=9 then nines :=nines +1 else if q=10 then
begin write(predigit+1) ;
for k :=1 to nines do write(0) ;
predigit :=0 ; nines :=0
end
else begin
write(predigit) ; predigit :=q ;
if nines <> 0 then
begin
for k :=1 to nines do write (9) ;
nines :=0
end
end
end ;
```

writeln(predigit) ;
end

3.11 programme simplifié

Nous allons calculer les caractères de π en base 10000. Un caractère sera un entier de 4 chiffres allant de0 à 9 puisque c'est un entier inférieur à 10000. Comme dans les 1000 premières décimales pe π on ne rencontre pas la suite "0000" qui pourrait venir de la queue des prédigits "9999" augmentée de 1, on peut simplifier l'algorithme en ne gardant qu'un caratère (mais en base 10000) en attente.
C'est le rôle de la mémoire tampon e.
Les digits sortiront 4 par 4 et la mémoire tampon sera constituée de 4 digits. Nous avons vu que pour obtenir n décimales on doit prendre $n \times 3,32$ digits pour π dans la base variable. En prenant la base 10000 au lieu de la base 10 , cela revient à regrouper ces digits par 4. Autrement dit le programme affiche p blocks de 4 décimales tels que :
$p \times 4 \times 3.32 = dimensiondutableau$. $4 \times 3,32 \approx 14$. On décrémente donc la variable c qui calcule le nombre de sous tableaux de 14 à chaque sortie des 4 digits.

Programme compte-gouttes de calcul de pi pout ti92 pour 54 digits. On doit utiliser 54 fois 3,32 soit 180 caractères dans la base variable pour π afin d'en recueillir 54 en base 10.
Prgm
ClrIO
$newList(106) \rightarrow f$
$10000 \rightarrow a : 180 \rightarrow c : 1 \rightarrow b : 0 \rightarrow e$
For b,1,c
$int(a/5) \rightarrow f[b]$
EndFor
While $c > 0$
$2 * c \rightarrow g : 0 \rightarrow d : c \rightarrow b$
While $b > 0$
$d + f[b] * a \rightarrow d : g - 1 \rightarrow g : mod(d, g) \rightarrow f[b]$
$int(d/g) \rightarrow d : g - 1 \rightarrow g : b - 1 \rightarrow b$
$Ifb \neq 0Then$
$d * b \rightarrow d$
EndIf
EndWhile
$c - 14 \rightarrow c : string(e + int(d/a)) \rightarrow x : dim(x) \rightarrow l$
if $u + v = 4$ then

```
output 0,0,  left(x1)&".&right(x,3)
else
output u,v,left("0000",4-l)&x
```

endif
$v + 25 \rightarrow v$
if $v > 150$ then
$0 \rightarrow v$
$10 + u \rightarrow u$
endif
$mod(d, a) \rightarrow e$
EndWhile
endprgm
Voici les résultats obtenus

0003141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117067982148086513282306647093844609550582231725359408128481117450284102701938521105559644622948954930381964428810975665933446128475648233786783165271201909145648566923460348610454326648213393607260249141273724587006606315588174881520920962829254091715364367892590360011330530548820466521384146951941511609433057270365759591953092186117381932611793105118548074462379962749567351885752724891227938183011949129833673362440656643086021394946395224737190702179860943702770539217176293176752384674818467669405132000568127145263560827785771342757789609173637178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813629774771309960518707211349999998372978049951059731732816096318595024459455346908302642522308253344685035261931188171010003137838752886587533208381420617177669147303598253490428755468731159562863882353787593751957781857780532171226806613001927876611195909216420198938095257201065485863278865936153381827968230301952035301852968995773622599413891249721775283479131515574857242454150695950829533116861727855889075098381754637464939319255060400927701671139009848824012858361603563707660104710181942955596198946767837449448255379774726847104047534646208046684259069491293313677028989152104752162056966024058038150193511253382430035587640247496473263914199272604269922796782354781636009341721641219924586315030286182974555706749838505494588586926995690927210797509302955321165344987202755960236480665499119881834797753566369807426542527862551818417574672890977772793800081647060016145249192173217214772350141441973568548161361157352552133475741849468438523323907394143334547762416 86

3.12 calcul de e avec l'algorithme du compte-goutte

On peut utiliser le même principe pour calculer d'autres nombres remarquables . Soit par exemple le nombre e tel que ln(e)=1.
e s'écrit (2,1,1,1,1...) dans la base à pas variable ($\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, $\frac{1}{5}$...).
Le tableau ci-dessous nous montre comment le calculer.

A	e	1r	1	1	1	1	1	1	1	1	1
B		1	2	3	4	5	6	7	8	9	10
initiali		2	1	1	1	1	1	1	1	1	1
$\times 10$		20	10	10	10	10	10	10	10	10	10
retenue		7	4	3	2	2	2	2	8	9	10
somme	27	27	14	13	12	12	12	18	19	11	10
reste		7	0	1	0	2	0	4	3	2	1

Voici le programme et le résultat obtenu par la calculatrice ti92.

Prgm
ClrIO
Local e,u,v
$0 \rightarrow u : 0 \rightarrow v$
$10000 \rightarrow a$
$160 \rightarrow c$
$1 \rightarrow b$
$0 \rightarrow e$
$newList(c) \rightarrow f$
For b,2,c
$1 \rightarrow f[b]$
EndFor
$2 \rightarrow f[1]$
While $c > 0$
$0 \rightarrow d : c \rightarrow b : c \rightarrow g$
While $b > 0$
$d + f[b] * a \rightarrow d$
$mod(d, b) \rightarrow f[b]$
$int(d/b) \rightarrow d$
$b - 1 \rightarrow b$
EndWhile
$c - 14 \rightarrow c : string(e + int(d/a)) \rightarrow x : dim(x) \rightarrow l$
If u+v=0 Then

```
Output  0,10,left(x,1)&"."\\
Else\\
Output  u,v,left("0000",4-l)&x\\
EndIf\\
$v+25\rightarrow v$\\
```

If $v > 150$ Then
$10 \rightarrow v$
$10 + u \rightarrow u$
EndIf
$mod(d, a) \rightarrow e$
EndWhile

EndPrgm

```
Calcul de e en C avec l'algorithme du compte-gouttes
#include <stdio.h>
long a=10000,b,c=1000,d,e,f[1000];
int main()
{
while (b<c)
{
f[b]=1000;b++;
}
f[1]=2000;
while (c>0)
{d=0;b=c;
while (b>0)
{d=d+f[b]*a;f[b]=d%b;d=d/b;b=b-1; }
 c-=10;printf("%.4ld",e+d/a);e=d%a;
}
 }
```

```
Résultats obtenusă:
C:\PI>e
27182818284590452353602874713526624977572470936999595749669676277240766394571382178525166427427466391932003059921817413596629043572900334295260513232862794349076323382988075319525101901157383418793070215408914993488476146066808226480016847741185374234544243710753907774499206955170276183313845830007520449338265602976067371132007093287091274437470472306969772
```

4 Calcul de π par la formule de MACHIN

Nous allons maintenant calculer 1000 décimales de π grâce à la formule de Machin (1680-1752) :

$\pi = 4(4arctan(1/5) - arctan(1/239))$ avec $arctan(\frac{1}{p}) = \sum_{k=0}^{\infty}(\frac{(-1)^k}{(2k+1)p^{2k+1}})$

4.1 démonstration

Calculons : $z = \frac{(5+i)^4}{239+i}$.

$z = \frac{476+480i}{239+i} = \frac{(476+480i)(239-i)}{239^2+1} = \frac{114244+114244i}{57122} = 2 + 2i = 2\sqrt{2}e^{i\frac{\pi}{4}}$.

Donc $Arg(z) = \frac{\pi}{4}$, par suite $\frac{\pi}{4} = 4Arg(5+i) - Arg(239+i) = 4Arctan(\frac{1}{5}) - Arctan(\frac{1}{239})$.

4.2 Calcul de arctan(x) par son développement en série

C'est James Gregory (1638-1675) qui découvrit la formule : $arctan(x) = \sum_{k=0}^{\infty}\frac{(-1)^k x^{2k+1}}{2k+1}$ pour x dans [-1 ;1].

Démontrons cette formule et majorons l'erreur commise en s'arrêtant au rang N dans le calcul de arctan(x) :$\sum_{k=0}^{N}(\frac{(-1)^k x^{2k+1}}{(2k+1)})$

$\sum_{n=0}^{N}(-t^2)^n = \frac{1-(-t^2)^{N+1}}{1+t^2}$ d'aprés la formule de la somme des termes d'une suite géométrique.

Donc $\frac{1}{1+t^2} = (\sum_{n=0}^{N}(-t^2)^n) + \frac{(-t^2)^{N+1}}{1+t^2} = \sum_{n=0}^{N}(-1)^n t^{2n} + (-1)^{N+1}\frac{t^{2N+2}}{1+t^2}$.

Intégrons pour obtenir arctan(x).

$arctan(x) = \int_0^x \frac{1}{1+t^2}dt = \int_0^x \sum_{n=0}^{N}(-1)^n t^{2n} + \int_0^x (-1)^{N+1}\frac{t^{2N+2}}{1+t^2}$

Notons $E_N(x) = \int_0^x (-1)^{N+1}\frac{t^{2N+2}}{1+t^2}$, alors on a :

$arctan(x) = \sum_{n=0}^{N}(-1)^n\frac{x^{2n+1}}{2n+1} + E_N(x)$

Majorons $E_N(x)$. On a : $\frac{1}{1+t^2} \leq 1$.

Donc : $|E_N(x)| \leq \int_0^x t^{2n+2}dt$. Par suite : $|E_N(x)| \leq \frac{x^{2N+3}}{2N+3}$.

Ceci implique que :

$$\lim_{N\to+\infty} E_N(x) = 0$$

pour x dans]-1 ;1[. d'où $arctan(x) = \sum_{k=0}^{\infty}\frac{(-1)^k x^{2k+1}}{2k+1}$ et l'erreur commise en s'arrétant au rang N est inférieure à : $\frac{x^{2N+3}}{2N+3}$.

4.3 calcul du nombre N pour obtenir une precision donnée

Nous avons deux causes d'erreurs dans le calcul de π :

La première est due à l'arrêt du calcul de arctan(x) à l'ordre N . Elle est

inférieure à $\frac{x^{2N+3}}{2N+3}$.
L'autre est due aux arrondis dans les calculs. Chaque nombre que l'on somme n'étant connu qu'avec la précision 10^{-n} n étant le nombre de chiffres aprés la virgule que l'on se donne pour les calculs.
Par exemple pour obtenir 100 chiffres de π soit une précision de 10^{-100} avec la formule de Machin $\pi = 4(4arctan(1/5) - arctan(1/239))$ nous devons d'abord calculer N.
L'erreur dans le calcul de $arctan(1/5)$ en s'arrétant à l'ordre N est inférieure à $\frac{1}{5^{2N+3}(2N+3)}$, celle dans le calcul de $arctan(1/239)$ en s'arrétant au même ordre N est trés inférieure : $\frac{1}{239^{2N+3}(2N+3)}$
Au total l'erreur en calculant π en s'arrétant à l'ordre N avec la formule de Machin est donc majorée par :
$E_N = \frac{4}{2N+3}(\frac{4}{5^{2N+3}} + \frac{1}{239^{2N+3}}) \leq \frac{4}{2N+3}\frac{5}{5^{2N+3}}$ majoration grossière .
Soit $E_N \leq \frac{20}{(2N+3)5^{2N+3}}$
En prenant N=70 on obtient $E_{70} \leq 10^{-101}$

Remarquons que $\frac{E_{N+1}}{E_N} = \frac{1}{25}\frac{2n+5}{2n+3} \approx 0,04$. L'erreur est divisée par 25 à chaque nouvelle itération. On gagne environ 1,4 chiffre pour toute nouvelle itération.

Calculons de même comment obtenir 1000 chiffres pour π après la virgule. Cherchons d'abord à avoir $E_N \leq 10^{-1000}$ soit : $\frac{20}{(2N+3)5^{2N+3}} \leq 10^{-1000}$.
Pour N=714 nous obtenons $ln(E_{714}) = ln(\frac{20}{5^{1427}1427}) = ln20 - 1427ln5 - ln1427 \approx -1000.68$.
$\frac{lnE_{714}}{ln10} \approx -1000$ donc l'erreur commise en tronquant à N=714 est inférieure à 10^{-1000}.
(Pour N=715 nous obtenons $log(E_{715}) \approx -1002$.) Nous avons 1427 nombres à ajouter ou soustraire pour obtenir π . Il suffit de prendre 1004 décimales dans les calculs pour obtenir π avec 1000décimales. En effet l'erreur au total est alors inférieure à : $10^{-1002} + 1427 \times 10^{-1004} \leq 10^{-1002}$. (715 itérations par arctangente).

4.4 calculs en multiprécision

Les "grands réels " (nous devrions dire les grands décimaux) sont des nombres décimaux. Dans notre cas $x = \pi$ avec 1000 chiffres après la virgule . Ils sont représentés par : $x = x[0] + \frac{x[1]}{B} + \frac{x[1]}{B^2} + \frac{x[3]}{B^3} + ... + \frac{x[n]}{B^n}$. B est la base utilisée et $0 \leq x[i] < B$
Pour notre écriture décimale B=10 et $\pi = 3 + \frac{1}{10} + \frac{4}{10^2} + \frac{1}{10^3}....$
Pour éviter des calculs compliqués de changement de base nous allons prendre $B = 10000 = 10^4$. On pourra alors écrire des nombres avec quelques milliers de décimales (de 1000 à 9999) avec une conversion simple en base 10. Si on veut plus de décimales on choisit une base plus grande par exemple

10^5.
Comment représenter alors un grand décimal x ayant une partie entière de un digit (on pense à π ou à e) et 1000 digits après la virgule.
On se donne un tableau x :

x[0]	x[1]	x[2]	x[3]	...	....	...	...	....	...	...	...	...	....	x[n]

Chaque x[i] représente le ième caractère de l'écriture de x dans la base B. Ce ième caractère est un entier inférieur à B=10000. Son écriture décimale est formée de 4 digits que nous appellerons ième caractère de x dans son écriture en base B. Comme x a 1000 chiffres aprés la virgule, que les x[i] s'obtiennent en prenant à la queue leu- leu les décimales de x le tableau considéré doit avoir $\frac{1000}{4} + 1 = 251$ cases. Le plus un est pour le 3 de la partie entière. D'une manière générale la taille du tableau est : $1 + \frac{nombrededigitsdex}{log(B)}$ avec $log(B) = \frac{ln(B)}{ln(10)} = logarithmedcimaldeB.$
Log(B) donne le nombre de chiffres du caractère x[i] lorsqu'on l'écrit dans la base dix.
Le changement d'écriture de la base B à la base 10 va consister à prendre tous ces nombres en les écrivant par groupe de 4 pour ne pas omettre les zéros.On rajoute un point après l'écriture de x[0] pour la virgule

x[0]	x[1]	x[2]	x[3]	x[4]
3	1415	9265	3589	7932

4.5 addittion en multiprécision

Il s'agit d'ajouter deux grands décimaux de taille n, c'est à dire représentés dans la base B=10000 (par exemple) par des tableaux x et y de taille n. Par exemple avec nos conventions ci dessus ils seront de taille 251 pour avoir 1000 décimales après la virgule. La somme sera stockée dans le tableau x.
Soient donc nos deux tableaux x et y , dans lesquels x[i] et y[i] sont les ièmes caractères de x et y dans leur écriture en base B. Ce sont donc des entiers inférieurs à B, donc dans notre cas B=10000 , leur écriture décimale a 4 digits.

x[0]	x[1]	x[2]	x[3]	...	x[n-1]
y[0]	y[1]	y[2]	y[3]	..	y[n-1]

On procède comme à l'école primaire de la droite vers la gauche en ajoutant à chaque fois la retenue.
début
retenue=0
Pour i=n-1 jusqu'à 0 par pas de -1 faire :
x[i]=x[i]+y[i]+retenue
si $x[i] < B$ alors retenue=0 sinon retenue=1 , x[i]=x[i]-B
fin si
fin pour

fin

4.6 soustraction en multiprécision

x doit être plus grand que y. x et y sont représentés par des tableaux de taille n. Le résultat de x-y est mis dans le tableau x.

x[0]	x[1]	x[2]	x[3]	...	x[n-1]
y[0]	y[1]	y[2]	y[3]	..	y[n-1]

On procède comme à l'école primaire de la droite vers la gauche . Si $x[i] < y[i]$ on doit enlever une unité au caractère précédent x[i-1] et rajouter B à x[i]. Ceci donne l'algorithme suivant :
début
Pour i= n-1 jusqu'à 0 par pas de -1 faire :
x[i]=x[i]-y[i]
Si $x[i] < 0$ alors
Si i=0 erreur
sinon x[i]=x[i]+B , x[i-1]=x[i-1]-1
fin pour
fin

4.7 multiplication en multiprécision

Il s'agit de multiplier un grand décimal x de taille n par un petit entier q , c'est à dire inférieur à B. On n'a pas besoin pour utiliser la formule d Machin d'une multiplication par un nombre plus grand. x est représenté par un taleau x (ou pointé en C par *x) de taille n, le résultat est mis dans x.

x[0]	x[1]	x[2]	x[3]	...	x[n-1]
					q

On procède de la droite vers la gauche, comme à l'école primaire. Quand le résultat xi de $x[i] \times q$ dépasse la base pour trouver la retenue on calcule la partie entière de$\frac{xi}{B}$.

début
retenue=0
Pour i=n-1 jusqu'à 0 par pas de -1 faire :
$xi = x[i] \times q$
xi=xi+retenue
Si $xi > B$ alors faire retenue = partie entière($\frac{xi}{B}$), $xi = xi - B \times retenue$
sinon retenue = 0
x[i]=xi fin pour
fin

4.8 division en multiprécision

Il s'agit de diviser un grand décimal x de taille n (pointé par *x en C) par un petit entier q c'est à dire inférieur à la base B . On n'a pas besoin de plus pour utiliser la formule de Machin. Soit donc le tableau de taille n représentant x et l'entier q, le résultat donne un grand décimal de taille n que l'on appelle y, stocké dans un tableau pointé en C par *y.

x[0]	x[1]	x[2]	x[3]	...	x[n-1]	q
y[0]	y[1]	y[2]	y[3]	...	y[n-1]	

On procède comme à l'école primaire de la gauche vers la droite. La retenue est le reste de la division euclidienne de x[i] par q.
début
retenue =0
Pour i=0 jusqu'à n-1 par pas de 1 faire :
$xi = x[i] + retenue \times B$
q=partie entière($\frac{xi}{q}$
$retenue = xi - q \times d$
y[i]=q
fin pour
fin

4.9 calcul de arctan(1/p)=arccot(p)

$arccot(p) = \sum_{k=0}^{\infty}(\frac{(-1)^k}{(2k+1)p^{2k+1}})$

p est un petit entier puisqu'il vaudra 5 ou 239. Le résultat est un grand décimal de taille n. On utilise deux buffers de taille n pour effectuer les calculs.
Le premier uk sert à stocker $\frac{1}{p^{2i+1}}$. On passe d'un uk donnant le grand décimal $\frac{1}{p^{2i+1}}$ au suivant en le divisant deux fois de suite par p . Pour cela on utilise la fonction définie par l'algorithme de la division vu dans le paragraphe au-dessus.
Le deuxième buffer vk calcule le ième terme de la suite $(\frac{(-1)^i}{(2i+1)p^{2i+1}})$. Pour cela on divise uk par 2i+1 que l'on obtient en ajoutant 2 à chaque itération à la mémoire k.
Cette boucle s'arrête lorsque uk est nul , c'est à dire lorsque toutes ses cases le sont.

4.10 test d'arrêt

A la ième boucle de arccot(5) uk est égal à $\frac{(-1)^i}{5^{2i+1}(2i+1)}$. uk est donc nul lorsque ses mille et une cases le sont. Ceci s'obtiendra lorsque :

$\frac{(1}{5^{2i+1}(2i+1)} = 10^{-1000}$, soit en prenant le logarithme en base 10 : $log(5^{2i+1}(2i+1)) = 1000$ ou encore $\frac{(2i+1)ln(5)+ln(2i+1)}{ln(10)} = 1000$. Pour i=713 on obtient environ 1000. Il faut donc 713 itérations pour que arccot(5) soit nul. Il en faudra moins pour arccot(239).

Arccot(5) est alors connnu à 10^{-1000} prés, arccot(239) aussi. Comme nous avons ajouté 2×713 nombres connu chacun à 10^{-1000} prés, le résultat final est connu à $2 \times 10^{-1000} + 2 \times 713 \times 10^{-1000}$ prés soit environ 10^{-996}. Autrement dit les 4 derniers chiffres ne sont pas certains.

4.11 Programme

Voici le programme en C++ :

```
** Formule:
**
**    Pi/4 =  4*arctan(1/5)-arctan(1/239)             (Machin)
**      avec arctan(x) =  x - x^3/3 + x^5/5 - ...
**
**
**    Un réel en multiprécision est défini dans la base B par
**      X = x(0) + x(1)/B^1 + ... + x(n-1)/B^(n-1)
**     où 0<=x(i)<B
**
*/

#include <stdio.h>

#include <malloc.h>
#include <math.h>
long B=10000; /* base de travail */
long LB=4;    /* Log10(base)  */

/*
** Set the big real x to the small integer Integer
*/
void SetToInteger (long n, long *x, long Integer) {
  long i;
  for (i=1; i<n; i++) x[i] = 0;
```

```
  x[0] = Integer;
}
/*
** le grand réel x est_il égal à 0 ?
*/
long IsZero (long n, long *x) {
  long i;
  for (i=0; i<n; i++)
    if (x[i]) return 0;
return 1;
}
/*
** Addition de grands réels : x += y
**  comme à l'école avec repport de retenue
*/
void Add (long n, long *x, long *y) {
  long carry=0, i;
  for (i=n-1; i>=0; i--) {
    x[i] += y[i]+carry;
    if (x[i]<B) carry = 0;
    else {
      carry = 1;
      x[i] -= B;
    }
  }
}
/*
** soustraction de grands réels : x -= y
**  comme à l'école avec repport de retenue
**  x doit être plus grand que y
*/
void Sub (long n, long *x, long *y) {
  long i;
  for (i=n-1; i>=0; i--) {
    x[i] -= y[i];
if (x[i]<0) {
  if (i) {
        x[i] += B;
        x[i-1]--;
      }
}
  }
}
/*
```

```
** Multiplication du grand réel x par l'entier q
** x = x*q.
**  comme à l'école avec repport de retenue
*/
void Mul (long n, long *x, long q) {
  long carry=0, xi, i;
  for (i=n-1; i>=0; i--) {
    xi  = x[i]*q;
    xi += carry;
    if (xi>=B) {
      carry = xi/B;
      xi -= (carry*B);
    }
    else
      carry = 0;
    x[i] = xi;
}
}
/*
** Division du grand réel x par l'entier d
**le résultat est  y=x/d.
**  comme à l'école aves repport de retenue
**  d est inférieur à  MaxDiv*MaxDiv.
*/
void Div #include <stdio.h>
long a=10000,b,c=8400,d,e,f[8401],g;
main(){for(;b-c;)f[b++]=2;
for(;d=0,g=c*2;c-=14,printf("%.4ld",e+d/a),e=d%a)
for(b=c;d+=f[b]*a,f[b]=d%--g,d/=g--,--b;d*=b);}

Voici le programme en C++ :\\
\begin {verbatim}
/*
** Formule:
**
**    Pi/4 =  4*arctan(1/5)-arctan(1/239)                    (Machin)
**      avec arctan(x) =  x - x^3/3 + x^5/5 - ...
**
**
**    Un réel en multiprécision est défini dans la base B par
**      X = x(0) + x(1)/B^1 + ... + x(n-1)/B^(n-1)
**     où 0<=x(i)<B
**
*/
```

```
#include <stdio.h>
#include <malloc.h>
#include <math.h>
long B=10000; /* base de travail */
long LB=4;    /* Log10(base)  */
/*
** Set the big real x to the small integer Integer
*/
void SetToInteger (long n, long *x, long Integer) {
  long i;
  for (i=1; i<n; i++) x[i] = 0;
  x[0] = Integer;
}
/*
** le grand réel x est_il égal à 0 ?
*/
long IsZero (long n, long *x) {
  long i;
  for (i=0; i<n; i++)
    if (x[i]) return 0;
return 1;
}
/*
** Addition de grands réels : x += y
**  comme à l'école avec repport de retenue
*/
void Add (long n, long *x, long *y) {
  long carry=0, i;
  for (i=n-1; i>=0; i--) {
    x[i] += y[i]+carry;
    if (x[i]<B) carry = 0;
    else {
      carry = 1;
      x[i] -= B;
    }
  }
}
/*
** soustraction de grands réels : x -= y
**  comme à l'école avec repport de retenue
**  x doit être plus grand que y
*/
void Sub (long n, long *x, long *y) {
  long i;
```

```
  for (i=n-1; i>=0; i--) {
    x[i] -= y[i];
if (x[i]<0) {
  if (i) {
        x[i] += B;
        x[i-1]--;
      }
}
  }
}
/*
** Multiplication du grand réel x par l'entier q
** x = x*q.
**  comme à l'école avec repport de retenue
*/
void Mul (long n, long *x, long q) {
  long carry=0, xi, i;
  for (i=n-1; i>=0; i--) {
    xi  = x[i]*q;
    xi += carry;
    if (xi>=B) {
      carry = xi/B;
      xi -= (carry*B);
    }
    else
      carry = 0;
    x[i] = xi;
}
}
/*
** Division du grand réel x par l'entier d
**le résultat est  y=x/d.
**  comme à l'école aves repport de retenue
**  d est inférieur à  MaxDiv*MaxDiv.
*/
void Div (long n, long *x, long d, long *y) {
  long carry=0, xi, q, i;
  for (i=0; i<n; i++) {
    xi    = x[i]+carry*B;
    q     = xi/d;
    carry = xi-q*d;
    y[i]  = q;
  }
}
```

```
/*
** donne l' arc cotangente de l'entier p = arctan (1/p)
**  le résultat est  x (size n)
**  buf1 and buf2 sont 2 buffers de taille n
*/
void arccot (long p, long n, long *x, long *buf1, long *buf2) {
  long p2=p*p, k=3, sign=0;
  long *uk=buf1, *vk=buf2;
  SetToInteger (n, x, 0);
  SetToInteger (n, uk, 1); /* uk = 1/p */
  Div (n, uk, p, uk);
  Add (n, x, uk);           /* x  = uk */

  while (!IsZero(n, uk)) {

      Div (n, uk, p, uk);   /* Two steps for large p (see division) */
      Div (n, uk, p, uk);

    /* uk = u(k-1)/(p^2) */

    Div (n, uk, k, vk);       /* vk = uk/k  */
    if (sign) Add (n, x, vk); /* x = x+vk   */
    else Sub (n, x, vk);      /* x = x-vk   */
    k+=2;
    sign = 1-sign;
  }
}
/*
** Impression du grand réel x
*/
void Print (long n, long *x) {
  long i;
  printf ("%d.", x[0]);
  for (i=1; i<n; i++) {
    printf ("%.4d", x[i]);
    if (i%25==0) printf ("%8d\n", i*4);
  }
  printf ("\n");
}
/*
** Calcul de  Pi avec la formule de Machin
*/
void main ()
 {
```

```
  long NbDigits=1000;

  long size=1+NbDigits/LB;
  long *Pi      = (long *)malloc(size*sizeof(long));
  long *arctan  = (long *)malloc(size*sizeof(long));
  long *buffer1 = (long *)malloc(size*sizeof(long));
  long *buffer2 = (long *)malloc(size*sizeof(long));
  /*
 On réserve des tableaux de 251 cases qui contiennent  chacune un entier inférieur à B s
 donc de 4 digits.
  ** Formule utilisée
  **
  **   Pi/4 = 4*arctan(1/5)-arctan(1/239) (Machin)
  */
    SetToInteger(size,Pi,0);
    arccot (5, size, arctan, buffer1, buffer2);
    Mul (size, arctan, 4);
    Add(size,Pi,arctan) ;
     arccot (239, size, arctan, buffer1, buffer2);
      Sub(size,Pi,arctan);
  Mul (size, Pi, 4);
  Print (size, Pi);  /* Print out of Pi */
  free (Pi);
  free (arctan);
  free (buffer1);
  free (buffer2);
}
```

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706799
82148086513282306647093844609550582231725359408128481117450284102701938521105559644622948954930381964
42881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372
45870066063155881748815209209628292540917153643678925903600113305305488204665213841469519415116094
33057270365759591953092186117381932611793105118548074462379962749567351885752724891227938183011949129
83367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200
05681271452635608277857713427577896091736371787214684409012249534301465495853710507922796892589235420
19956112129021960864034418159813629774771309960518707211349999998372978049951
```

```
05973173281609631859      800\\
50244594553469083026425223082533446850352619311881710100031378387528865875332083
81420617177669147303      900\\
59825349042875546873115956286388235378759375195778185778053217122680661300192787
66111959092164202156    1000\\
```

5 conclusion

Pour obtenir plus de chiffres jusqu'à 10000 on peut utiliser cet algorithme. On peut l'améliorer en prenant une série qui converge plus vite vers π. On peut prendre par exemple la formule de Gauss :$\frac{\pi}{4} = 12 \times arctan(\frac{1}{18} + 8 \times arctan(\frac{1}{57}) - 5 \times arctan(\frac{1}{239})$.
Cela réduira le temps de calcul qui devient vite important.
Nous avons dans notre algorithme utilisé la multiplication et la division par un entier petit, c'est à dire plus petit que la base. Sa limitation est là. Si l'on veut faire ces opérations sur des grands décimaux cela augmente énormément la complexité des calculs . La rapidité et la puissance des ordinateurs ne sont pas suffisantes pour permettre de calculer π avec des millions de décimales avec les deux algorithmes que nous venons de voir.
De nouveaux algorithmes sont nécessaires. La transformée de Fourier rapide permet de diminuer le temps de calcul dans la multiplication des grans nombres. D'auters formules pour calculer π ont une convergence bien meilleure. Simon Plouffe en 1995 a établit que $\pi = \sum_{k=0}^{\infty} \frac{1}{16^k}(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6})$. Cette formule est remarquable car elle permet de calculer le 40 milliardième digit de π en base deux sans calculer les précédents. La formule $\pi = \frac{9801}{\sqrt{8}} \sum_{n=0}^{\infty}(\frac{(4n)!(1103+26390n)}{(n!)^4 396^{4n}})^{-1}$, découverte en 1910 par le génial mathématicien calculateur prodige Ramanujan, permit de calculer un milliard de décimales.
Plus que la puissance des nouveaux ordinateurs, c'est donc les dévelopements récents des mathématiques qui ont permit d'atteindre une telle précision dans le calcul de π.

6 bibliographie

Le fascinant nombre π. Jean-Paul Delahaye. Belin
Terre et espace. Hors série nř 5. Tangente. Editions Archimède
Autour du nombre π, Pierre Eymard et Jean-Pierre Lafon, Hermann
www.multimania.com/bgourevith/
http :/perso-info.enst-bretagne.fr/ brouty/Maths/pi.htm
http ://numbers.computation.free.fr/Constants/Algorithms/representation.html
pour la science numéros199 de mai1994 et 215 deseptembre 1995

American Mathematical Monthly, March 1995, a spigot algoritm for pi, Stanley Rabinovitz et Stan Vagon.