Computation of the n'th digit of π in any base in $O(n^2)$

Fabrice Bellard

January 10, 1997

This article is an *alpha* version. Please send any comments to
`Fabrice.Bellard@enst.fr`

Simon Plouffe explained in [1] a new algorithm to compute the n'th digit of π and some other mathematical constants in any base with very little memory. Its running time is $O(n^3 \log(n)^3)$. We present here an improvement of this algorithm whose running time is $O(n^2)$ while its memory requirements stay $O(1)$, which makes it practical to compute the millionth digit of π for example.

1 First result

We want to compute the n'th digit in base B of s, where

$$s = \frac{b}{A} \text{ with } A = \prod_{i=1}^{m} a_i \text{ and } gcd(a_i, a_j) = 1 \text{ for } i \neq j \ .$$

Let $b_i = b \bmod a_i$, $c_i = (\prod_{j \neq i} a_j)^{-1} \bmod a_i$ and $\alpha_i = b_i c_i \bmod a_i$. We have

$$b = (\sum_{i=1}^{m} \alpha_i \prod_{j \neq i} a_j) \bmod A$$

with the Chinese remainder theorem. Hence

$$s \bmod 1 = \sum_{i=1}^{m} \frac{\alpha_i}{a_i} \bmod 1.$$

The $(n+k)$'th digits of s, where $k \geq 1$, are the digits in base B of

$$D_n = sB^n \bmod 1 \ .$$

With the last formula we have

$$D_n = \sum_{i=1}^{m} \frac{\alpha_i B^n \bmod a_i}{a_i} \bmod 1.$$

As shown in [1], this equality is interesting because each term can be computed separatly, with a total memory of $O(1)$ if we already know b and $(a_i)_{i=1..m}$.

2 Second result

We want now to compute the n'th digit in base B of S_N, where

$$S_N = \sum_{k=1}^{N} \frac{b_k}{A_k}$$

with

$$A_k = \prod_{i=1}^{m} a_i^{\nu_{i,k}} \ ,\ 0 \leq \nu_{i,k} \leq \nu_i^{max} \text{ and } gcd(a_i, a_j) = 1 \text{ for } i \neq j \ .$$

With the first result we have

$$S_N = \sum_{k=1}^{m} \frac{\alpha_i}{a_i^{\nu_i^{max}}} \ ,$$

where

$$\alpha_i = (\sum_{k=1}^{N} \alpha_{i,k}) \bmod a_i^{\nu_i^{max}}$$

with

$$\alpha_{i,k} = (b_k (\prod_{j \neq i} a_j^{\nu_{j,k}})^{-1} \bmod a_i^{\nu_{i,k}}).a_i^{\nu_i^{max} - \nu_{i,k}} \text{ if } \nu_{i,k} > 0$$

and

$$\alpha_{i,k} = 0 \text{ if } \nu_{i,k} = 0.$$

The key observation is that we can use for all $\alpha_{i,k}$ the same modulo, hence

$$\alpha_{i,k} = b_k (\prod_{j \neq i} a_j^{\nu_{j,k}})^{-1} a_i^{\nu_i^{max} - \nu_{i,k}} \bmod a_i^{\nu_i^{max}} \ .$$

This can be rewritten

$$\alpha_i = \sum_{k=1}^{N} b_k (\frac{A_k}{a_i^{\nu_{i,k}}})^{-1} a_i^{\nu_i^{max} - \nu_{i,k}} \bmod a_i^{\nu_i^{max}} \ .$$

To have the digits after the n'th one, in base B, we compute

$$D_n = \sum_{i=1}^{m} \frac{\alpha_i B^n \bmod a_i^{\nu_i^{max}}}{a_i^{\nu_i^{max}}} \bmod 1.$$

The running time is $O(m.N.log(M))$ and the memory requirements stay $O(1)$ if we suppose that:

- b_k, A_k, a_i, $\nu_{i,k}$ and ν_i^{max} are given.
- $a_i^{\nu_i^{max}} \leq M$ for every i. We suppose that M fits in a computer word so that each operation takes a time of $O(1)$ except the modulo inversion which takes a time of $O(\log(M))$.
- ν_i^{max} is small so an exponentiation by it takes a time of $O(1)$.

3 Application

Given

$$\pi + 3 = \sum_{k=1}^{+\infty} \frac{k 2^k}{\binom{2k}{k}} ,$$

we can use the result of section 2 because if p is a prime number, we notice that

$$\nu(p, \binom{2n}{n}) \leq \lfloor \log(2n)/\log(p) \rfloor$$

where $\nu(p, n)$ is the multiplicity of p in n. It comes from the relation

$$\nu(p, \binom{2n}{n}) = \sum_{i=1}^{+\infty} \lfloor \frac{2n}{p^i} \rfloor - 2 \sum_{i=1}^{+\infty} \lfloor \frac{n}{p^i} \rfloor .$$

Hence, if we want the n'th digit of π in base B, we may use the following algorithm:

- $N \leftarrow \lfloor (n + \epsilon) \log_2(B) \rfloor$ where ϵ is a small integer to ensure we have the precision needed ; $sum \leftarrow 0$.
- For each prime number a with $2 < a < 2N$, do:
  - $\nu^{max} \leftarrow \lfloor \log(2N)/\log(a) \rfloor$; $m \leftarrow a^{\nu^{max}}$.
  - $\alpha \leftarrow 0$; $\nu \leftarrow 0$; $b \leftarrow 1$; $A \leftarrow 1$.
  - for k in $1..2N$ do:
    * $b \leftarrow \frac{k}{a^{\nu(a,k)}}.b \bmod m$; $A \leftarrow \frac{(2k-1)}{a^{\nu(a,2k-1)}}.A \bmod m$; $\nu \leftarrow \nu - \nu(a, k) + \nu(a, 2k - 1)$.
    * if $\nu > 0$ do: $\alpha \leftarrow \alpha + k.b.A^{-1}.a^{\nu^{max} - \nu} \bmod m$.
  - $\alpha \leftarrow \alpha.B^{n-1} \bmod m$; $sum \leftarrow sum + \frac{\alpha}{m} \bmod 1$.
- If we suppose that $\pi = (d_0.d_1 d_2 d_3 ...)_B$, then, if we neglect rounding errors, $sum = (0.d_n d_{n+1} d_{n+2} .. d_{n+q-1} ...)_B$. The number q of correct digits depends on ϵ.

The running time is $O(n^2)$ because there are $O(\frac{n}{\log(n)})$ prime numbers between 2 and $2n$. The memory requirements are, as expected, in $O(1)$.

4 Conclusion

We have presented an algorithm to compute the n'th digit in any base B of π whose running time is $O(n^2)$. It has the same running time as other classical methods for computing π (e.g. arctangent formulas), but it uses little memory, it is very simple and does not need high precision computations. It is still slower than the BBP algorithm [2], but it works in any base. As described in [1], the same algorithm may be used to compute other numbers such as $\zeta(3)$, π^2, π^3, π^4 and $\pi\sqrt{3}$.

References

[1] Simon Plouffe, *On the computation of the n'th decimal digit of various transcendental numbers*, November 1996.

[2] David H. Bailey, Peter B. Borwein and Simon Plouffe, *On the Rapid Computation of Various Polylogarithmic Constants*, April 1997 in Mathematics of Computation.