Pi is in Log Space

Preliminary Version

Chee Yap[*]
Courant Institute, New York University

June 22, 2010

Abstract

We say that a real number α is in Log space if its n-th bit ($n \in \mathbb{Z}$) can be computed in $O(\log |n|)$ space. We show that $\pi = 3.14159\cdots$ is in Log space. This implies that π is in the complexity class SC. The latter result has been conjectured by Lipton, and widely assumed to be true from the work of Bailey, Borwein and Plouffe (BBP) and others. Our result extends to other constants such as $\log 2$ or π^2 that also possess two essential ingredients: they have BBP-like series and have bounded irrationality measures.

1 Introduction

Computing approximations of mathematical constants such as $\pi = 3.14159\cdots$ has been a quest from antiquity. With the advent of modern computing, and for about two decades from the 1970s to 1990s, the algorithms of choice for computing π to very high precision have been based on the arithmetic-geometric mean (AGM) from Salamin and Brent (1975). From the 1990s, a new class of algorithms were introduced by Rabinowitz and Wagon [7]. They are called "spigot algorithms" because such algorithms can start "dripping output digits" in the midst of the larger computation. The dripped digits are not re-used, so theoretically such algorithms may only use sublinear space, an important property if we want to compute ultra-billion digits. Gibbons [4] introduced a bit-streaming paradigm for spigot algorithms that avoided any prior commitment as to the ultimate number of desired digits. The most famous of the spigot algorithms is the BBP algorithm [1] which has the further property it can compute the n-th digit without computing the first $n-1$ digits. If one wants to compute all the first n digits of π, then AGM algorithms are asymptotically faster than any spigot algorithm, but the tradeoff is that AGM algorithms must use at least linear space. A spigot algorithm made possible the current record high precision for computing π (2.7 trillion digits by Bellard (2009) [9]).

[*] Visiting Oxford University Computing Lab, 2009-2010

The key property of spigot algorithms is that they use small space. Formally, it is claimed that BBP-like algorithms can compute the n-th bit of constants such as π in time polynomial in n, and in space poly-logarithmic in n. In complexity theory, algorithms with such complexity bounds are called SC **algorithms**. Decision problems with SC algorithms constitute the complexity class SC. Thus it is claimed that "π is in the class SC" [1, 2]. But some researchers have noticed that this claim is non-obvious and unproven. We refer to a blog by Lipton [6] who posed the status of π in SC as an open problem. In this paper, we will resolve this question for π and other constants.

The BBP algorithm [1] is based on the following remarkable series for π:

$$\pi = \sum_{k=0}^{\infty} \frac{1}{(16)^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right) = \sum_{k \geq 0} (16)^{-k} \varphi(k) \quad (1)$$

where $\varphi(k) = \frac{120k^2+151k+47}{512k^4+1024k^3+712k^2+194k+15}$ is a fixed rational function. See [2] for a simple proof. Why should such a series allow us to compute the n-th bit of π without obtaining the first n bits? Intuitively, it is the denominator of 16^k in the k-th term. But the issue is more subtle, and the original BBP paper acknowledged it as follows: "*There is always a possible ambiguity when computing a digit string base b in distinguishing a sequence of digits $a(b-1)(b-1)(b-1)$ from $(a+1)000$. In this particular case, we consider either representation as an acceptable computation. In practice this problem does not arise.*" Lipton rightly pointed out that such a claim is a fudge. The real issue (as we will see) is the number m of terms beyond the n-th term that must be examined in order to determine the n-th bit. If m cannot be bounded in terms of n, then the BBP algorithm (suitably adjusted to avoid any fudge) is only recursive with no complexity bounds, much less in SC.

2 Computing bits of Pi-like Constants

We will show that π is in the complexity class SC. In fact, we prove a stronger and more general result: any real constant α that (like π) possesses two general ingredients can be computed in logarithmic space. The first ingredient is the existence of a BBP-like series, analogous to the BBP series (1). The second ingredient is that α must possess a finite irrationality measure. The use of BBP-like series is widely expected ever since [1]. The use of irrationality measure was anticipated by Lipton [6] although it has never been formally exploited in this context. Such measures are well-known in Transcendental Number Theory [3].

We fix a real constant $\alpha > 0$ in this section. Let the binary expansion of α be given by

$$\alpha = \sum_{k=-m}^{\infty} \alpha_k 2^{-k}$$

where $\alpha_k \in \{0, 1\}$ and $\alpha_{-m} = 1$. Call α_k the **k-th bit** of α, and $msb(\alpha) := m$ is its **most significant bit position**. E.g., $msb(2^k) = -k$, $msb(\pi) = 1$,

$msb(0.1) = 4$. Moreover, if $k < msb(\alpha)$ then $\alpha_k = 0$. The **n-th (lower) approximation** of α is

$$\lfloor \alpha \rfloor_n := \sum_{k=-m}^{n} \alpha_k 2^{-k}.$$

The usual floor function $\lfloor \alpha \rfloor$ corresponds to $\lfloor \alpha \rfloor_0$. We may write $\alpha = (\alpha_{-m}\alpha_{-m+1}\cdots\alpha_0 \, . \, \alpha_1\alpha_2\cdots)_2$ for the binary notation for α. For instance, if $\alpha = \pi = 3.14159\cdots$ then

$$\pi = (11.001001\cdots)_2 = (\pi_{-1}\pi_0 \, . \, \pi_1\pi_2\pi_3\cdots)_2$$

For irrational α, its n-th bit is uniquely defined; otherwise, we achieve uniqueness by proscribing a suffix of all 1's in the binary notation. These definitions yield the identity:

$$2^m\alpha = (1.\alpha_{-m+1}\alpha_{-m+2}\cdots)_2 \qquad \text{where } \; m = msb(\alpha). \tag{2}$$

A constant $K > 0$ is called an **irrationality measure** of α if the inequality

$$0 < \left|\alpha - \frac{p}{q}\right| \leq \frac{1}{q^K} \tag{3}$$

holds for finitely many pairs (p, q) of positive integers. The infimum of all such K's, denoted $\mu(\alpha)$, is called *the* irrationality measure of α. Let $N(K) \geq 2$ denote a constant such that for all $q \geq N(K)$, the inequality (3) fails. For our proof, any finite K will do. In case $\alpha = \pi$, there is a long history of improving bounds for $\mu(\pi)$ by Hata and others [5]. The current record is $\mu(\pi) \leq 7.6804$ from Salikhov [8].

Besides the requirement $\mu(\alpha) < \infty$, the other ingredient is a suitable series for α. Let the constant α have a series of the form

$$\alpha = \sum_{k=0}^{\infty} t_k = S_n + R_n \tag{4}$$

where we have split the series into a finite sum $S_n := \sum_{k=0}^{n} t_k$, and a remainder series $R_n := \sum_{k=n+1}^{\infty} t_k$. We say the series (4) is **BBP-like** if each term t_k is a rational number of the form

$$t_k = 2^{-kc}\frac{p(k)}{q(k)} \tag{5}$$

where $p(k), q(k)$ are fixed polynomials with integer coefficients, and $c \geq 1$ is an integer. Clearly, the BBP series (1) is BBP-like.

We can now state the main result of this section:

Theorem 1. *For any positive real number α, if α has a finite irrationality measure K and a BBP-like series, then we can decide its n-th bit α_n in $O(\log |n|)$ space.*

Constants such as $\log 2$ and π^2 also have BBP-like series [1, 2]. They have bounded irrationality measures: $\mu(\log 2) \leq 3.575$ (Marcovecchio, 2009), $\mu(\pi^2) \leq 5.442$ (Rhin and Viola, 1996). Thus:

Corollary 2. *Let $\alpha \in \{\pi, \pi^2, \log 2\}$. We can decide the n-th bit of α in time in $O(\log |n|)$ space.*

¶1. Some Bounds. In the rest of this section, we will prove Theorem 1 through a series of lemmas. We fix an irrationality measure K and the BBP-like series (4)–(5) for α. Without loss of generality, we assume α is irrational; the only effect of this assumption is that the inequality $|\alpha - (p/q)| > 0$ in (3) becomes automatic.

The following is immediate:

Lemma 3. *In the BBP-like series (4)–(5), the remainder series R_n satisfies*

$$|R_n| < 2^{-nc+D \lg n} \tag{6}$$

for some constant $D > 0$ ($\lg = \log_2$).

Remark that in the case of (1) for π, we have the sharper bound

$$0 < R_n < \frac{1}{15} 16^{-n} < 2^{-4n-3}.$$

A standard notation for the fractional part of α is $\{\alpha\} = \alpha - \lfloor \alpha \rfloor$. We generalize this notation: for any $n \in \mathbb{Z}$,

$$\{\alpha\}_n := \{2^n \alpha\}. \tag{7}$$

Clearly, $\{\alpha\}_n \in [0, 1)$ and $\{\alpha\}_0 = \{\alpha\}$. Here is the connection between $\{\alpha\}_n$ and $\lfloor \alpha \rfloor_n$:

$$\begin{aligned} \{\alpha\}_n &= 2^n \alpha - \lfloor 2^n \alpha \rfloor \\ &= 2^n (\alpha - \lfloor \alpha \rfloor_n) \\ &= (0.\alpha_{n+1} \alpha_{n+2} \cdots)_2. \end{aligned}$$

This last equality shows:

$$\alpha_n = 1 \quad \Longleftrightarrow \quad \{\alpha\}_{n-1} \geq 1/2. \tag{8}$$

Lemma 4. *Let $2^n \geq N(K)$ and $\epsilon_n := 2^{-(K-1)n-2}$.*
(a) $\alpha_n = 1$ if and only if $\{\alpha\}_{n-1} \in (\frac{1}{2} + 2\epsilon_n, 1 - 2\epsilon_n)$.
(b) $\alpha_n = 0$ if and only if $\{\alpha\}_{n-1} \in (2\epsilon_n, \frac{1}{2} - 2\epsilon_n)$.

Proof. By definition of K, for all $p \in \mathbb{Z}$, $|\alpha - p2^{-n}| > 2^{-Kn}$, i.e.,

$$\left| 2^{n-1} \alpha - \frac{p}{2} \right| \quad > \quad 2^{-(K-1)n-1} = 2\epsilon_n. \tag{9}$$

This is equivalent to *(a)* and *(b)*. **Q.E.D.**

In the next two lemmas, we choose m be a sufficiently large value as a function of n.

LEMMA 5. *Assume n and ϵ_n as in the previous lemma. Choose m large enough so that $mc - D\lg m \geq Kn + 1$.*
(a') $\alpha_n = 1$ if and only if $\{S_m\}_{n-1} \in (\frac{1}{2} + \epsilon_n, 1 - \epsilon_n)$.
(b') $\alpha_n = 0$ if and only if $\{S_m\}_{n-1} \in (\epsilon_n, \frac{1}{2} - \epsilon_n)$.

Proof. By the previous lemma, $\alpha_n = 1$ iff $\{\alpha\}_{n-1} \in (\frac{1}{2} + 2\epsilon_n, 1 - 2\epsilon_n)$. Writing $\alpha = S_m + R_m$, we see that

$$\{\alpha\}_{n-1} = \left\{2^{n-1}S_m + 2^{n-1}R_m\right\} = \{S_m\}_{n-1} + 2^{n-1}R_m$$

holds, provided $2^{n-1}|R_m| \leq 2\epsilon$. This proviso follows from Lemma 3:

$$2^{n-1}|R_m| \leq 2^{n-1}2^{-mc+D\lg m} \leq 2^{-(K-1)n-2} = \epsilon_n$$

by our choice of m. Then *(a'),(b')* are equivalent to *(a),(b)*, in Lemma 4. **Q.E.D.**

Let $t^i_{k,n}$ denote the i-th bit of $\left\{2^{n-1}t_k\right\}$:

$$\left\{2^{n-1}t_k\right\} = (0.t^1_{k,n}t^2_{k,n}t^3_{k,n}\cdots)_2.$$

We use the first m bits to define the number

$$\widetilde{t}^m_{k,n} := (0.t^1_{k,n}t^2_{k,n}t^3_{k,n}\cdots t^m_{k,n})_2 \tag{10}$$

Thus, $\widetilde{t}^m_{n,k}$ is the m-th approximation of $\left\{2^{n-1}t_k\right\}$.

Consider the sum

$$S^n_m := \sum_{k=0}^{m} \widetilde{t}^m_k. \tag{11}$$

Note that $\{S^n_m\}$ is an approximation of $\{S_m\}_{n-1}$.

LEMMA 6. *Let $2^n \geq N(8)$ and choose $m \geq 2$ such that $m - \lg(m+1) \geq (K-1)n + 2$, and assume $D \geq 2$.*
(a") $\alpha_n = 1$ if and only if $\{S^n_m\} \geq \frac{1}{2}$.
(b") $\alpha_n = 0$ if and only if $\{S^n_m\} < \frac{1}{2}$.

Proof. We note that our assumption $m - \lg(m+1) \geq (K-1)n + 2$ implies the corresponding condition

$$mc - D\lg m \geq Kn + 1$$

in Lemma 5 (use the fact that $c \geq 1, D \geq 2, m \geq 2$). Hence Lemma 5 implies that $\{S_m\}_{n-1} > \frac{1}{2} + \epsilon_n$ or $\{S_m\}_{n-1} < \frac{1}{2} - \epsilon_n$. Consider the sum

$$T^n_m := \sum_{k=0}^{m} \left\{2^{n-1}t_k\right\}. \tag{12}$$

Clearly, $\{S_m\}_{n-1} = \{T^n_m\}$, and hence $\{T^n_m\} > \frac{1}{2} + \epsilon_n$ or $\{T^n_m\} < \frac{1}{2} - \epsilon_n$. The sum S^n_m in (11) is an approximation of T^n_m. The difference $T^n_m - S^n_m$, is non-negative and upper bounded by $2^{-m}(m+1) = 2^{-m+\lg(m+1)}$. By our assumption on m, this is at most $2^{-(K-1)n-2} = \epsilon_n$. Hence $\{S^n_m\} > \frac{1}{2}$ or $\{S^n_m\} < \frac{1}{2}$, corresponding to $\alpha_n = 1$ or $\alpha_n = 0$ (resp.). In particular, $\{S^n_m\}$ cannot be equal to $\frac{1}{2}$. Thus $\{S^n_m\} \geq \frac{1}{2}$ iff $\{S^n_m\} > \frac{1}{2}$. This concludes our proof. **Q.E.D.**

Note that we could also have written condition *(b")* as "$\alpha = 0$ iff $\{S^n_m\} \leq \frac{1}{2}$". Although it is not wrong, it would appear confusing in combination with *(a")*. In our application, it will be slightly easier to use *(a")* because we only need to check that first bit of $\{S^n_m\}$ is 1 to conclude that $\{S^n_m\} \geq \frac{1}{2}$.

¶2. The Algorithm. Our problem of computing α_n is now reduced to deciding the predicate

$$\{S^n_m\} \geq \frac{1}{2}. \tag{13}$$

Conceptually, we organize the data in the sum S^n_m in the following $(m+1) \times m$ bit-array A^n_m:

$$A^n_m = \begin{bmatrix} t^1_{0,n} & t^2_{0,n} & t^3_{0,n} & \cdots & t^m_{0,n} \\ t^1_{1,n} & t^2_{1,n} & t^3_{1,n} & \cdots & t^m_{1,n} \\ \vdots & & & & \\ t^1_{m,n} & t^2_{m,n} & t^3_{m,n} & \cdots & t^m_{m,n} \end{bmatrix} \tag{14}$$

The k-th row represents the binary number $\widetilde{t}^m_{k,n} = (0.t^1_{k,n}t^2_{k,n}\cdots t^m_{k,n})_2$, and so S^n_m is just the sum of the values in the $m+1$ rows. We say this is conceptual because we cannot afford to store this matrix. Instead, we generate each entry in Log space:

Lemma 7. *We can generate $t^i_{k,n}$ (the i-th bit of $\left\{2^{n-1}t_k\right\}$) in $O(\log N)$ space where $N = n+k+i$.*

Proof. We have $t_k = 2^{-kc}p(k)/q(k)$, and it is easy to compute $p(k)$ and $q(k)$ in $O(\log k)$ space. Let c_i be the i-th bit of $p(k)/q(k)$.

CLAIM: We can compute c_i in $O(\log(k+i))$ space. To see this, note that $c_i = 1$ iff $\left\{2^{i-1}p(k)/q(k)\right\} \geq \frac{1}{2}$. We consider two cases.
(a) $i \geq 1$: Note that we can compute $P_i := 2^{i-1}|p(k)| \,\mathbf{mod}\, |q(k)|$ in $O(\log(k+i))$ space. Then $c_i = 1$ iff $2P_i \geq |q(k)|$.
(b) $i \leq 0$: Note that $2^{-i} > |p(k)|$ implies $2^{i-1}p(k)/q(k) < 1/2$ and so $c_i = 0$. This can be checked in $O(\log k)$ space. If $2^{-i} \leq |p(k)|$, then $-i \leq C\log k$ for some $C > 0$. Then $\left\{2^{i-1}p(k)/q(k)\right\} = P_i/Q_i$ where $Q_i = 2^{-i+1}|q(k)|$ and $P_i = |p(k)| \,\mathbf{mod}\, Q_i$. Both Q_i, P_i can be computed $O(\log k)$ space. Then $c_i = 1$ iff $2P_i \geq Q_i$. This establishes our claim.

Now the desired ith bit in this lemma, $t^i_{k,n}$, is just $c_{i+kc-n+1}$ since $2^{n-1}t_k = 2^{n-1+kc}p(k)/q(k)$. Using the claim, $t^i_{k,n}$ can be generated in $O(\log(k+i+(n-1+kc))) = O(\log(n+k+i))$ space. **Q.E.D.**

¶3. Proof of Theorem 1. It remains to decide (13) in Log space. Let the sum of the j-th column of the matrix A_m^n be s_j. We compute S_m^n by the obvious algorithm: starting from $j = m$ down to $j = 1$, we compute s_j, and then add the carry c_j. Initially, $c_m = 0$. In the j-th step, we can compute the j-th bit of S_m^n as $(s_j + c_j) \mathbf{mod}\, 2$. But we are not interested in this bit, as we only want to generate the carry c_{j-1} for the next column using the formula $c_{j-1} = \lfloor (s_j + c_j)/2 \rfloor$. When we reach $j = 1$, the predicate (13) is decided with the equivalence: $S_m^n \geq \frac{1}{2}$ iff $(s_1 + c_1) \mathbf{mod}\, 2 = 1$ (i.e., $s_1 + c_1$ is odd).

We bound the storage used by this algorithm. The space to store the s_j's is $O(\log n)$ since $0 \leq s_j \leq m + 1$. The same bound holds for the c_j's, since a geometric series yields the bound $0 \leq c_j < 2(m + 1)$. Since we did not store the matrix A_m^n, we must account for the space used to generate each entry: by Lemma 7, the space to generate $t_{k,n}^i$ is $O(\log(n + k + i))$. Since $k, i \leq m$, this space is $O(\log(n + m))$. But the constraint on m in Lemma 6 allows a choice $m = O(n)$. To be concrete, we may choose $m = \max\{7, 2(K - 1)n + 4\}$. This completes the proof.

3 Remarks

- Our Log-space algorithm for π is unlike the original BPP algorithm. Therefore the "*SC* status" of the original BBP-like algorithms remains open.

- We have assumed base 2 in the above development. If π has a BBP-like series in base d, then again our result extends to showing that π in base d is in *DL*. In the most interesting case of $d = 10$, extensive search for BBP-like series has been unsuccessful. Hence it is not known if π in base d belongs to *SC*.

- There are highly non-constructive aspects in the concept of irrationality measure (3). Although we have an explicit K, we do not know bounds on the "finitely number" $M = M(K)$ of pairs p, q that are exceptions to the inequality (3). To bound M, we only need to bound the number of distinct q's since there can be at most one value of p for each $q \geq 2$. But even if we know M, we still do not know $N = N(K)$, which bounds the exceptional values of q. These non-constructive features do not detract from our application of irrationality measures in showing Theorem 1. But it does mean that some implicit constants in the big-Oh notation for the complexity of our algorithm for π are unknown.

4 Acknowledgment

I would like to thank Joel Ouaknine and Ben Worrell for helpful discussions, and especially for bringing Lipton's blog and the question of π to my attention.

References

[1] D. Bailey, P. Borwein, and S. Plouffe. On the rapid computation of various polylogarithmic constants. *Mathematics of Computation*, 66(218):903–913, 1997.

[2] D. J. Broadhurst. Polylogarithmic ladders, hypergeometric series and the ten millionth digits of $\zeta(3)$ and $\zeta(5)$. *arXiv:math.CA/9803067*, March 1998.

[3] N. Fel'dman and Y. V. Nesterenko. *Number Theory IV: Transcendental Numbers*, volume 44 of *Encyclopaedia of Mathematical Sciences*. Springer-Verlag, Berlin, 1998. Translated from Russian by N. Koblitz.

[4] J. Gibbons. An unbounded spigot algorithm for the digits of pi. *Amer. Math. Monthly*, 113(4):318–328, 2006.

[5] M. Hata. Improvement in the irrationality measures of π and π^2. *Proc. Japan. Acad. Ser. A Math. Sci.*, 68:283–286, 1992.

[6] R. J. Lipton. Cook's Class containts Pi, March 15 2009. Blog at `http://rjlipton.wordpress.com/2009/03/15/cooks-class-contains-pi/`.

[7] S. Rabinowitz and S. Wagon. A spigot algorithm for the digits of pi. *Amer. Math. Monthly*, 102:195–203, 1995.

[8] V. K. Salikhov. On the irrationality measure of pi. *Usp. Mat. Nauk*, 63:163–164, 2008. English transl. in Russ. Math. Surv 63, 570-572, 2008.

[9] Wikipedia. Numerical approximations of π, 2010. `http://en.wikipedia.org/wiki/Numerical_approximations_of_pi`.