The (digital) life of Pi

Jonathan M. Borwein and Mason S. Macklem


Jonathan has become widely known as an advocate for experimental mathematics, a field in which the extensive computational power of modern day computers is used to discover mathematical theorems. The article below, joint with Mason S. Macklem, discusses many of the digital truths about π, together with the computational and algorithmic advances that have been made discovering these truths.

The desire to understand π, the challenge, and originally the need, to calculate ever more accurate values of π, the ratio of the circumference of a circle to its diameter, has challenged mathematicians—great and less great—for many many centuries. Recently, π has provided compelling examples of computational mathematics. It is also part of both mathematical culture and of the popular imagination.

Why computations of π? One historical motivation was very much in the spirit of modern experimental mathematics: to see if the decimal expansion of π repeats, which would mean that π is the ratio of two integers (i.e., rational), or to recognize π as algebraic—the root of a polynomial with integer coefficients—and later to look at digit distribution. The question of the rationality of π was settled in the late 1700s, when Lambert and Legendre proved (using continued fractions) that the constant is irrational. The question of whether π was algebraic was settled in 1882, when Lindemann proved that π is transcendental. But this was known as far back as Aristophanes in 414 BCE, and thus hardly justified further computations.

So what possible motivation lies behind modern computations of π? One motivation is the raw challenge of harnessing the stupendous power of modern computer systems. Programming such calculations are definitely not trivial, especially on large, distributed memory computer systems. There have been substantial practical spin-offs. For example, some new techniques for performing the fast Fourier transform (FFT), heavily used in modern science and engineering computing, had their roots in attempts to accelerate computations of π. And always the computations help in road-testing computers—often uncovering subtle hardware and software errors.

Pi in the digital age: With the substantial development of computer technology in the 1950s, π was computed to thousands and then millions of digits. These computations were greatly facilitated by the discovery soon after of advanced algorithms for the underlying high-precision arithmetic operations. For example, in 1965 it was found that the (then newly-discovered) fast Fourier transform (FFT) [5, 7] could be used to perform high-precision multiplications much more rapidly than conventional schemes.

Such methods (e.g., for ÷, √x see [5, 6, 7]) dramatically lowered the time required for computing π and other constants to high precision. We are now able to compute algebraic
values of algebraic functions essentially as fast as we can multiply, $O_B(M(N))$. In spite of
these advances, into the 1970s all computer evaluations of $\pi$ still employed classical formulae.

In 1973, Guilloud and Boyer used a formula of Euler for arccot, namely

$$x \sum_{n=0}^{\infty} \frac{(n!)^2 4^n}{(2n+1)! (x^2 + 1)^{n+1}} = \arctan \left( \frac{1}{x} \right),$$

to compute a million digits of $\pi$. Specifically, they used this formula to express $\pi/4 = 12 \arctan(1/18) + 8 \arctan(1/57) - 5 \arctan(1/239)$ in the efficient form

$$\pi = 864 \sum_{n=0}^{\infty} \frac{(n!)^2 4^n}{(2n+1)! 325^{n+1}} + 1824 \sum_{n=0}^{\infty} \frac{(n!)^2 4^n}{(2n+1)! 3250^{n+1}} - 20 \arctan \left( \frac{1}{239} \right),$$

where the terms of the second series are just decimal shifts of the first.

Truly new types of infinite series formulae, based on elliptic integral approximations, were
discovered by Srinivasa Ramanujan (1887–1920) around 1910, but were not well known (nor
fully proven) until quite recently when his writings were widely published. They are based
on elliptic functions and are described at length in [2, 5, 7]. One of these series is the
remarkable:

$$\frac{1}{\pi} = 2\sqrt{2} \sum_{k=0}^{\infty} \frac{(4k)! (1103 + 26390k)}{(k!)^4 396^k}.$$ (1)

Each term of this series produces an additional eight correct digits in the result. When
Gosper used this formula to compute 17 million digits of $\pi$ in 1985, and it agreed to many
millions of places with the prior estimates, this concluded the first proof of (1), as described
in [4]! Actually, Gosper first computed the simple continued fraction for $\pi$, hoping to
discover some new things in its expansion, but found none.

At about the same time, David and Gregory Chudnovsky found the following rational
variation of Ramanujan’s formula:

$$\frac{1}{\pi} = 12 \sum_{k=0}^{\infty} \frac{(-1)^k (6k)! (13591409 + 545140134k)}{(3k)! (k!)^3 640320^{3k+3/2}}.$$ (2)

Each term of this series produces an additional 14 correct digits. The Chudnovskys im-
plemented this formula using a clever scheme that enabled them to use the results of an
initial level of precision to extend the calculation to even higher precision. They used this
in several large calculations of $\pi$, culminating with a then-record computation of over four
billion decimal digits in 1994.

While the Ramanujan and Chudnovsky series are in practice considerably more efficient
than classical formulae, they share the property that the number of terms needed increases
linearly with the number of digits desired: if you want to compute twice as many digits of $\pi$,
you must evaluate twice as many terms of the series. Relatedly, the Ramanujan-type series

$$\frac{1}{\pi} = \sum_{n=0}^{\infty} \left( \frac{(2n)}{16^n} \right)^3 \frac{42n + 5}{16}. \quad (2)$$

allows one to compute the billionth binary digit of $1/\pi$, or the like, without computing the
first half of the series, thus foreshadowing some surprising recent results.

Reduced operational complexity algorithms: In 1976, Eugene Salamin and Richard
Brent independently discovered a reduced complexity algorithm for $\pi$. It is based on the
arithmetic-geometric mean iteration (AGM) and some other ideas due to Gauss and Legendre around 1800, although Gauss, nor many after him, never directly saw the connection to effectively computing π.

**Quadratic Algorithm (Salamin-Brent).** Set \(a_0 = 1, b_0 = 1/\sqrt{2}\) and \(s_0 = 1/2\). Calculate

\[
a_k = \frac{a_{k-1} + b_{k-1}}{2} \quad (A) \quad b_k = \sqrt{a_{k-1}b_{k-1}} \quad (G)
\]

\[
c_k = a_k^2 - b_k^2, \quad s_k = s_{k-1} - 2^k c_k \quad \text{and compute} \quad p_k = \frac{2a_k^2}{s_k}. \quad (4)
\]

Then \(p_k\) converges quadratically to \(π\). Each iteration of the algorithm doubles the correct digits. Successive iterations produce 1, 4, 9, 20, 42, 85, 173, 347 and 697 good decimal digits of \(π\), and takes \(\log N\) operations for \(N\) digits. Twenty-five iterations computes \(π\) to over 45 million decimal digit accuracy. A disadvantage is that each of these iterations must be performed to the precision of the final result.

In 1985, Jonathan and Peter Borwein discovered families of algorithms of this type. For example, here is a genuinely third-order iteration:

**Cubic Algorithm:** Set \(a_0 = 1/3\) and \(s_0 = (\sqrt{3} - 1)/2\). Iterate

\[
r_{k+1} = \frac{3}{1 + 2(1 - s_k^3)^{1/3}}, \quad s_{k+1} = \frac{r_{k+1} - 1}{2} \quad \text{and} \quad a_{k+1} = r_{k+1}a_k - 3^k(r_{k+1} - 1).
\]

Then \(1/a_k\) converges cubically to \(π\). Each iteration triples the number of correct digits.

**Quartic Algorithm:** Set \(a_0 = 6 - 4\sqrt{2}\) and \(y_0 = \sqrt{2} - 1\). Iterate

\[
y_{k+1} = \frac{1 - (1 - y_k^4)^{1/4}}{1 + (1 - y_k^4)^{1/4}} \quad \text{and} \quad a_{k+1} = a_k(1 + y_{k+1})^4 - 2^{2k+3}y_{k+1}(1 + y_{k+1} + y_{k+1}^2).
\]

Then \(1/a_k\) converges quartically to \(π\). (Note that only the power of 2 or 3 used in \(a_k\) depends on \(k\).) This quartic algorithm, with the Salamin–Brent scheme, was first used by Bailey in 1986 and was used repeatedly by Yasumasa Kanada in Tokyo in computations of \(π\) over the past 15 years or so, culminating in a 200 billion decimal digit computation in 1999 (see Figure 1). Only 35 years earlier in 1963, Dan Shanks—a very knowledgeable participant—was confident that computing a billion digits was forever impossible. Today it is easy on a modest laptop.

**Back to the future:** In December 2002, Kanada computed \(π\) to over \(1.24\) trillion decimal digits! His team first computed \(π\) in hexadecimal (base 16) to 1,030,700,000,000 places, using the following two arctangent relations:

\[
π = 48\tan^{-1}\frac{1}{49} + 128\tan^{-1}\frac{1}{57} + 20\tan^{-1}\frac{1}{239} + 48\tan^{-1}\frac{1}{110443}
\]

\[
π = 176\tan^{-1}\frac{1}{57} + 28\tan^{-1}\frac{1}{239} - 48\tan^{-1}\frac{1}{682} + 96\tan^{-1}\frac{1}{12943}
\]

The first formula was found in 1982 by K. Takano, a high school teacher and song writer. The second formula was found by F. C. W. Störmer in 1896. Kanada verified the results of these two computations agreed, and then converted the hex digit sequence to decimal. The resulting decimal expansion was checked by converting it back to hex. These conversions are themselves non-trivial, requiring massive computation.

This process is quite different from those of the previous quarter century. One reason is that reduced operational complexity algorithms require full-scale multiply, divide and square root operations. These in turn require large-scale FFT operations, which demand huge amounts of memory, and massive all-to-all communication between nodes of a large
parallel system. For this latest computation, even the very large system available in Tokyo
did not have sufficient memory and network bandwidth to perform these operations at
reasonable efficiency levels—at least not for trillion-digit computations. Utilizing arctans
again meant using many more arithmetic operations, but no system-scale FFTs, and it can
be implemented using $\times$, $\div$ by smallish integer values—additionally, hex is somewhat more
efficient!

Kanada and his team evaluated these two formulae using a scheme analogous to that
employed by Gosper and by the Chudnovskys in their series computations, in that they
were able to avoid explicitly storing the multiprecision numbers involved. This resulted in
a scheme that is roughly competitive in numerical efficiency with the Salamin-Brent and
Borwein quartic algorithms they had previously used, but with a significantly lower total
memory requirement. Kanada used a 1 Tbyte main memory system, as with the previous
computation, yet got six times as many digits. Hex and decimal evaluations included, it
ran 600 hours on a 64-node Hitachi, with the main segment of the program running at a
sustained rate of nearly 1 Tflop/sec.

The use of arctangents by Kanada and his team marks an unexpected return to these clas-
sical formulae, which were used throughout pre-digital calculations of $\pi$. Recent summaries
of the history of arctangent calculations of $\pi$ can be found in [8, 10].

**Why Pi?** Beyond practical considerations lies the abiding interest in the fundamental
question of the normality (digit randomness) of $\pi$. Kanada, for example, has performed
detailed statistical analysis of his results to see if there are any statistical abnormalities
that suggest $\pi$ is not normal, so far ‘no’. Indeed the first computer computation of $\pi$ and $e$
on ENIAC was so motivated by John von Neumann. The digits of $\pi$ have been studied
more than any other single constant, in part because of the widespread fascination with
and recognition of $\pi$. Kanada reports that the 10 decimal digits ending in position one
trillion are 6680122702, while the 10 hexadecimal digits ending in position one trillion are
3F89341CD5.

**How to compute the $N$-th digits of $\pi$.** One might be forgiven for thinking that es-
sentially everything of interest with regards to $\pi$ has been dealt with. Even insiders are
sometimes surprised by a new discovery. Prior to 1996, most folks thought if you want to
determine the $d$-th digit of $\pi$, you had to generate the (order of) the entire first $d$ digits.
This is not true, at least for hex (base 16) or binary (base 2) digits of $\pi$.

In 1996, Peter Borwein, Plouffe, and Bailey found an algorithm for computing individ-
ual hex digits of $\pi$. It (1) yields a modest-length hex or binary digit string for $\pi$, from
an arbitrary position, using no prior bits; (2) is implementable on any modern computer;
(3) requires no multiple precision software; (4) requires very little memory; and (5) has a
computational cost growing only slightly faster than the digit position. For example, the
millonth hexadecimal digit (four millonth binary digit) of $\pi$ can be found in four seconds
on a present generation Apple G5 workstation.

This new algorithm is not fundamentally faster than the best known schemes if used for computing all digits of $\pi$ up to some position, but its elegance and simplicity are of
considerable interest, and is easy to parallelize. It is based on the following at-the-time new
formula for $\pi$:

$$
\pi = \sum_{i=0}^{\infty} \frac{1}{16^i} \left( \frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6} \right)
$$  (5)
The (digital) life of Pi

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Correct Digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miyoshi and Kanada</td>
<td>1981</td>
<td>2,000,036</td>
</tr>
<tr>
<td>Kanada-Yoshino-Tamura</td>
<td>1982</td>
<td>16,777,206</td>
</tr>
<tr>
<td>Gosper</td>
<td>1985</td>
<td>17,526,200</td>
</tr>
<tr>
<td>Bailey</td>
<td>Jan. 1986</td>
<td>29,360,111</td>
</tr>
<tr>
<td>Kanada and Tamura</td>
<td>Sep. 1986</td>
<td>33,554,414</td>
</tr>
<tr>
<td>Kanada et. al</td>
<td>Jan. 1987</td>
<td>134,217,700</td>
</tr>
<tr>
<td>Kanada and Tamura</td>
<td>Jan. 1988</td>
<td>201,326,551</td>
</tr>
<tr>
<td>Chudnovskys</td>
<td>May 1989</td>
<td>480,000,000</td>
</tr>
<tr>
<td>Kanada and Tamura</td>
<td>Jul. 1989</td>
<td>536,870,898</td>
</tr>
<tr>
<td>Kanada and Tamura</td>
<td>Nov. 1989</td>
<td>1,073,741,799</td>
</tr>
<tr>
<td>Chudnovskys</td>
<td>Aug. 1991</td>
<td>2,260,000,000</td>
</tr>
<tr>
<td>Chudnovskys</td>
<td>May 1994</td>
<td>4,044,000,000</td>
</tr>
<tr>
<td>Kanada and Takahashi</td>
<td>Oct. 1995</td>
<td>6,442,450,938</td>
</tr>
<tr>
<td>Kanada and Takahashi</td>
<td>Jul. 1997</td>
<td>51,539,600,000</td>
</tr>
<tr>
<td>Kanada and Takahashi</td>
<td>Sep. 1999</td>
<td>206,158,430,000</td>
</tr>
<tr>
<td>Kanada-Ushiro-Kuroda</td>
<td>Dec. 2002</td>
<td>1,241,100,000,000</td>
</tr>
</tbody>
</table>

Figure 1. Calculations of $\pi$

which was discovered using integer relation methods (see [7]), with a computer program that ran for several months and then produced the (equivalent) relation:

$$\pi = 4 F\left(1, \frac{1}{4}; \frac{5}{4}, -\frac{1}{4}\right) + 2 \tan^{-1}\left(\frac{1}{2}\right) - \log 5$$

where $F(1,1/4; 5/4, -1/4) = 0.955933837\ldots$ is a Gaussian hypergeometric function. Surprisingly, the proof of this result is neither long nor inaccessible, and indeed can fit on a single overhead slide!

The algorithm in action. In 1997, Fabrice Bellard of INRIA computed 152 binary digits of $\pi$ starting at the trillionth position. The computation took 12 days on 20 workstations working in parallel over the Internet. Bellard’s scheme is based on the following variant of (5):

$$\pi = 4 \sum_{k=0}^{\infty} \frac{(-1)^k}{4^k(2k+1)} - \frac{1}{64} \sum_{k=0}^{\infty} \frac{(-1)^k}{1024^k} \left( \frac{32}{4^k+1} + \frac{8}{4^k+2} + \frac{1}{4^k+3} \right),$$

which permits hex or binary digits of $\pi$ to be calculated roughly 43% faster than (5).

In 1998 Colin Percival, then a 17-year-old student at Simon Fraser University, utilized 25 machines to calculate first the five trillionth hexadecimal digit, and then the ten trillionth hex digit. In September 2000, he found the quadrillionth binary digit is 0, a computation that required 250 CPU-years, using 1734 machines in 56 countries.

A last comment for this section is that Kanada was able to confirm his 2002 computation in only 21 hours by computing a 20 hex digit string starting at the trillionth digit, and comparing this string to the hex string he had initially obtained in over 600 hours. Their agreement provided enormously strong confirmation.

Changing world views. In retrospect, we may wonder why in antiquity $\pi$ was not measured to an accuracy in excess of $22/7$? Perhaps it reflects not an inability to do so but
a very different mind set to a modern experimental—Baconian or Popperian—one. In the same vein, one reason that Gauss and Ramanujan did not further develop the ideas in their identities for \( \pi \) is that an iterative algorithm, as opposed to explicit results, was not as satisfactory for them (especially Ramanujan). Ramanujan much preferred formulae like

\[
\pi \approx \frac{3}{\sqrt{67}} \log (5280), \quad \frac{3}{\sqrt{163}} \log (640320) \approx \pi
\]
correct to 9 and 15 decimal places, both of which rely on deep number theory. Contrastingly, Ramanujan in his famous 1914 paper *Modular Equations and Approximations to Pi* [2, p.253] found

\[
\left( 9^2 + \frac{19^2}{22} \right)^{1/4} = 3.14159265258 \ldots
\]

“empirically, and it has no connection with the preceding theory.” Only the marked digit is wrong. Indeed, much life remains in this most central of numbers.

Acknowledgement

Thanks are due to many, especially Peter Borwein and David Bailey.

References


There are many Internet resources on Pi, a reliable selection is kept at http://www.expmath.info.

Department of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada, B3H1W5. E-mail: jborwein@cs.dal.ca

Department of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada, B3H1W5 E-mail: mason@cs.dal.ca

Received 8 May 2006.