Jonathan Borwein, Pi and the AGM

Richard P. Brent

Australian National University, Canberra
and CARMA, University of Newcastle

26 Sept 2017

In fond memory of Jon Borwein 1951–2016

$2^{10}e^{\pi}$ *days was not enough*

Abstract

We consider some of Jon Borwein's contributions to the high-precision computation of π and the elementary functions, with particular reference to the fascinating book *Pi and the AGM* (Wiley, 1987) by Jon and his brother Peter Borwein.

Here "AGM" is the *arithmetic-geometric mean*, first studied by Euler, Gauss and Legendre. Because the AGM has second-order convergence, it can be combined with fast multiplication algorithms to give fast algorithms for the *n*-bit computation of π, and more generally the elementary functions. These algorithms run in "almost linear" time $O(M(n)\log n)$, where $M(n)$ is the time for *n*-bit multiplication.

The talk will survey some of the results and algorithms, from the time of Archimedes to the present day, that were of interest to Jon. In several cases they were discovered or improved by him.

A message from Peter Borwein

Peter Borwein writes:

> *"I would've loved to attend. But unfortunately I have multiple sclerosis. It makes it impossible to travel. If you could pass on my regards and best wishes to everyone I would greatly appreciate it. Thank you"*

Why π?

Why was Jon interested in π?

Perhaps because it is *transcendental* but appears in many mathematical formulas. For example, here are some that I like:

$$\begin{aligned}
e^{i\pi} &= -1, && \text{(Euler)},\\
\frac{\pi}{4} &= \arctan(1) = 1 - \frac{1}{3} + \frac{1}{5} - \cdots && \text{(Gregory/Leibnitz)},\\
&= \arctan(\tfrac{1}{2}) + \arctan(\tfrac{1}{3}) && \text{(Euler)},\\
&= 4\arctan(\tfrac{1}{5}) - \arctan(\tfrac{1}{239}) && \text{(Machin)},\\
\sqrt{\pi} &= \Gamma(\tfrac{1}{2}), \text{ where } \Gamma(s) := \int_0^\infty x^{s-1}e^{-x}\,dx,\\
\zeta(2n) &= (-1)^{n+1}\frac{B_{2n}(2\pi)^{2n}}{2(2n)!} \quad \text{for non-negative integers } n.
\end{aligned}$$

The *Bernoulli numbers* $B_{2n} \in Q$ ($B_0 = 1$, $B_2 = \frac{1}{6}$, $B_4 = -\frac{1}{30}, \cdots$) have a simple exponential generating function $x/(e^x - 1) - x/2$.

Generalisations

$e^{i\pi} = -1$ is a special case of $e^{i\theta} = \cos\theta + i\sin\theta$.

$\sqrt{\pi} = \Gamma(\frac{1}{2})$ is a special case of $\sqrt{\pi} = 2^{2s-1}\frac{\Gamma(s)\Gamma(s+\frac{1}{2})}{\Gamma(2s)}$.

Euler's formula for $\zeta(2n)$ follows from the Hadamard product

$$\frac{\sin \pi z}{\pi z} = \prod_{n=1}^{\infty}\left(1 - \frac{z^2}{n^2}\right).$$

There are infinite families of arctan formulas for π whose proofs are based on the identity

$$\arctan u + \arctan v = \arctan\left(\frac{u+v}{1-uv}\right).$$

This may be more familiar in the form

$$\tan(\alpha+\beta) = \frac{\tan\alpha + \tan\beta}{1 - \tan\alpha\,\tan\beta}.$$

We'll later see examples involving elliptic integrals, theta functions, etc. Thus, a formula for π is usually just the tip of a large iceberg!

Why so many digits?

To compute the circumference of the planet Earth to an accuracy of 1mm from the formula $2\pi r$, we only need π to 11 decimal digits.

The circumference of the observable universe can be computed to within one Planck length (1.6×10^{-35} metres) if we know π to about 62 decimal digits, assuming that space-time is flat and we know the radius.

Hence, why would anyone ever be interested in computing π to more than 100 decimal digits?

Some answers

One possible answer: because e is too easy, and γ is too hard!

More seriously, as we saw already, π is just the tip of several icebergs, and we want to see what is underwater.

To find identities using the PSLQ algorithm (Ferguson and Bailey) or to numerically verify conjectured identities, we need to be able to compute the relevant terms to high precision.

Thus, we want to be able to compute many constants to high precision, for example $\zeta(3)$, γ, $\exp(\pi\sqrt{163})$, $\Gamma(p/q)$ for rational p/q, $\ldots$

To implement arbitrary-precision software such as MPFR, we need algorithms for the computation of elementary and special functions to arbitrary precision.

Another answer

As in mountain-climbing, “because it is there!”

Of course, a mountain has a finite height, so in principle we can get to the top. $\pi = 3.14159265\cdots$ has a nonterminating (and non-periodic) decimal (or binary) expansion, so we can never compute all of it.

For this reason, I prefer to work on algorithms for computing π than on programs to approximate it to a large (but finite) number of digits. I suspect that Jon Borwein had the same view.

The “game” is to find an algorithm with the least possible asymptotic time complexity (usually ignoring constant factors). It will not necessarily be the most practical algorithm for computing π, because of constant factors, implementation difficulties, memory requirements, etc.

Some means

To refresh your memory, the *arithmetic mean* of $a, b \in \mathbb{R}$ is

$$\mathrm{AM}(a,b) := \frac{a+b}{2}\,.$$

The *geometric mean* is

$$\mathrm{GM}(a,b) := \sqrt{ab},$$

and the *harmonic mean* (for $ab \neq 0$) is

$$\mathrm{HM}(a,b) := \mathrm{AM}(a^{-1}, b^{-1})^{-1} = \frac{2ab}{a+b}\,.$$

Assuming that a and b are positive, we have the inequalities

$$\mathrm{HM}(a,b) \leq \mathrm{GM}(a,b) \leq \mathrm{AM}(a,b).$$

Later we may have $a, b \in \mathbb{C}$. To resolve the ambiguity in the square root we assume that $\Re(GM(a,b)) \geq 0$, and $\Im(GM(a,b)) \geq 0$ if $\Re(GM(a,b)) = 0$.

The arithmetic-geometric mean

Given two positive reals a_0, b_0, we can iterate the arithmetic and geometric means by defining, for $n \geq 0$,

$$\begin{aligned} a_{n+1} &= \mathrm{AM}(a_n, b_n) \\ b_{n+1} &= \mathrm{GM}(a_n, b_n). \end{aligned}$$

The sequences (a_n) and (b_n) converge to a common limit called the *arithmetic-geometric mean* (AGM) of a_0 and b_0. We denote it by $\mathrm{AGM}(a_0, b_0)$.

The harmonic-geometric mean

We could define an iteration

$$\begin{aligned} a_{n+1} &= \mathrm{HM}(a_n, b_n) \\ b_{n+1} &= \mathrm{GM}(a_n, b_n). \end{aligned}$$

However, we see that

$$\begin{aligned} a_{n+1}^{-1} &= \mathrm{AM}(a_n^{-1}, b_n^{-1}) \\ b_{n+1}^{-1} &= \mathrm{GM}(a_n^{-1}, b_n^{-1}). \end{aligned}$$

Thus, the common limit is just $\mathrm{AGM}(a_0^{-1}, b_0^{-1})^{-1}$.

Replacing the arithmetic mean by the harmonic mean in the definition of the AGM does not give anything essentially new.

Another mean

Note that $\mathrm{AGM}(a_0, b_0) = \mathrm{AGM}(b_0, a_0)$ is symmetric in a_0, b_0. This is not true if we use a slightly different iteration

$$\begin{aligned} a_{n+1} &= \mathrm{AM}(a_n, b_n) \\ b_{n+1} &= \mathrm{GM}(a_{n+1}, b_n) \end{aligned}$$

which converges to a limit which we denote by $\mathrm{ARM}(a_0, b_0)$ ("AR" for "Archimedes", as we'll explain shortly).

The ARM is slightly easier to implement in a program than the AGM, as we can just drop the subscripts and iterate $\{a := \mathrm{AM}(a, b);\ b := \mathrm{GM}(a, b)\}$, avoiding the use of a temporary variable.

Archimedes

Archimedes (c.287–c.212 BC) gave perhaps the first iterative algorithm for computing π to arbitrary precision, and used the first few iterations to show that

$$3.1408 \approx 3\tfrac{10}{71} < \pi < 3\tfrac{1}{7} \approx 3.1429\,.$$

Many people believe that $\pi = 3\frac{1}{7}$. Archimedes knew better.

Digression: a more recent proof that $\pi < 3\frac{1}{7}$ is

$$0 < \int_0^1 \frac{x^4(1-x)^4}{1+x^2}\,dx = \frac{22}{7} - \pi.$$

To evaluate the integral, write the integrand as

$$x^6 - 4x^5 + 5x^4 - 4x^2 + 4 - \frac{4}{1+x^2}$$

and integrate term by term, using

$$\int_0^1 \frac{dx}{1+x^2} = \arctan(1) = \frac{\pi}{4}.$$

Inscribed and circumscribed polygons

Archimedes' key idea is to use the perimeters of inscribed and circumscribed polygons in a circle of radius $1/2$ to give lower and upper bounds on π. We start with hexagons and keep bisecting angles to get polygons with $6 \cdot 2^n$ sides.

Let A_n denote the perimeter of a circumscribed regular $6 \cdot 2^n$-gon, and B_n ditto for the inscribed regular $6 \cdot 2^n$-gon. Writing $\ell_n := 6 \cdot 2^n$, $\theta_n := \frac{\pi}{\ell_n}$, we see that

$$B_n = \ell_n \sin \theta_n < \pi < A_n = \ell_n \tan \theta_n.$$

The initial values are $\ell_0 = 6, \theta_0 = \pi/6, A_0 = 2\sqrt{3}, B_0 = 3$. Using "half-angle" formulas we can verify that

$$\begin{aligned} A_{n+1} &= \mathrm{HM}(A_n, B_n), \\ B_{n+1} &= \mathrm{GM}(A_{n+1}, B_n). \end{aligned}$$

Archimedes continued

Recall that

$$\begin{aligned} A_{n+1} &= \mathrm{HM}(A_n, B_n), \\ B_{n+1} &= \mathrm{GM}(A_{n+1}, B_n). \end{aligned}$$

To avoid the harmonic mean, define $a_n := 1/A_n,\ b_n := 1/B_n$. Then

$$\begin{aligned} a_{n+1} &= \mathrm{AM}(a_n, b_n), \\ b_{n+1} &= \mathrm{GM}(a_{n+1}, b_n). \end{aligned}$$

This is just an instance of the “Archimedes mean” ARM defined previously, so we see that

$$\mathrm{ARM}\left(\frac{\sqrt{3}}{6}, \frac{1}{3}\right) = \frac{1}{\pi}\,.$$

Similar methods give

$$ARM(\cos\theta, 1) = \frac{\sin\theta}{\theta}, \quad ARM(\cosh\theta, 1) = \frac{\sinh\theta}{\theta}\,.$$

Upper and lower bounds via Archimedes

Using Archimedes' method gives (correct digits in blue):

$$
\begin{aligned}
\text{iteration } 0: &\quad 3.0000000 < \pi < 3.4641017 \\
\text{iteration } 1: &\quad 3.1058285 < \pi < 3.2153904 \\
\text{iteration } 2: &\quad 3.1326286 < \pi < 3.1596600 \\
\text{iteration } 3: &\quad 3.1393502 < \pi < 3.1460863 \;\; < 3.1464 \\
\text{iteration } 4: &\quad 3.1410319 < \pi < 3.1427146 \;\; < 3.1435 \\
\text{iteration } 5: &\quad 3.1414524 < \pi < 3.1418731 \\
\text{iteration } 6: &\quad 3.1415576 < \pi < 3.1416628 \\
\text{iteration } 7: &\quad 3.1415838 < \pi < 3.1416102 \\
\text{iteration } 8: &\quad 3.1415904 < \pi < 3.1415971
\end{aligned}
$$

The bounds satisfy

$$A_n - B_n = \pi \left(\frac{\tan \theta_n - \sin \theta_n}{\theta_n} \right) < 2^{-2n-1}.$$

We get two bits of accuracy per iteration (linear convergence).

Implications of Archimedes' method

David Bailey has observed that there are at least *eight* recent papers in the "refereed" literature claiming that $\pi = (14 - \sqrt{2})/4 = 3.1464\cdots$, and another three claiming that $\pi = 17 - 8\sqrt{3} = 3.1435\cdots$.

These claims must be incorrect, due to Lindemann's 1882 theorem that π is transcendental, but we can give a more elementary disproof of the claims for anyone who does not understand Lindemann's proof.

Since $A_3 < 3.1464$ and $A_4 < 3.1435$, we see that four iterations of Archimedes' method suffice to disprove the claims.

Four iterations of Archimedes' method suffice to show that

$$3.1408 < 3\tfrac{10}{71} < \pi < 3\tfrac{1}{7} < 3.1429\,,$$

as (correctly) claimed by Archimedes.

What if Archimedes made a small change?

We've seen that the essential part of Archimedes' method is the iteration

$$\begin{aligned} a_{n+1} &= \mathrm{AM}(a_n, b_n), \\ b_{n+1} &= \mathrm{GM}(a_{n+1}, b_n). \end{aligned}$$

If Archimedes had written it this way, he might have considered making a small change and using the (more symmetric) iteration

$$\begin{aligned} a_{n+1} &= \mathrm{AM}(a_n, b_n), \\ b_{n+1} &= \mathrm{GM}(a_{n}, b_n). \end{aligned}$$

This is just the arithmetic-geometric mean!

What if $\cdots$ continued

Archimedes would have found that the new (AGM) iteration converges much faster than the old (ARM) iteration. To see this, suppose that $x_n := a_n/b_n = 1 + \varepsilon_n$. Then

$$x_{n+1} = \tfrac{1}{2}(a_n/b_n + 1)/\sqrt{a_n/b_n} = \tfrac{1}{2}(x_n^{1/2} + x_n^{-1/2}),$$

so

$$1 + \varepsilon_{n+1} = \tfrac{1}{2}((1+\varepsilon_n)^{1/2} + (1+\varepsilon_n)^{-1/2}) = 1 + \tfrac{1}{8}\varepsilon_n^2 + O(\varepsilon_n^3).$$

Thus $\varepsilon_{n+1} \approx \frac{1}{8}\varepsilon_n^2$ if $|\varepsilon_n|$ is small.
This is an example of *quadratic* convergence – the number of correct digits roughly doubles at each iteration. In contrast, the ARM has only *linear* convergence – the number of correct digits increases roughly linearly with each iteration.

The limit

Although the AGM iteration converges faster than the ARM iteration, it does not give the same limit. Thus, it's not immediately obvious that it is useful for computing π (or anything else of interest).

Gauss and Legendre solved the problem of expressing AGM(a, b) in terms of known functions. The answer may be written as

$$\frac{1}{\mathrm{AGM}(a,b)} = \frac{2}{\pi}\int_0^{\pi/2} \frac{d\theta}{\sqrt{a^2\cos^2\theta + b^2\sin^2\theta}}\,.$$

The right-hand-side is the product of a constant (whose precise value will be significant later) and a *complete elliptic integral*.

Elliptic integrals

The *complete elliptic integral of the first kind* is defined by

$$K(k) := \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}} = \int_0^1 \frac{dt}{\sqrt{(1-t^2)(1-k^2t^2)}},$$

and the *complete elliptic integral of the second kind* by

$$E(k) := \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2 \theta}\, d\theta = \int_0^1 \frac{\sqrt{1-k^2t^2}}{\sqrt{1-t^2}}\, dt.$$

The variable k is called the *modulus*, and $k' := \sqrt{1-k^2}$ is called the *complementary modulus*.

Some (confusing) notation

It is customary to define

$$K'(k) := K(\sqrt{1-k^2}) = K(k')$$

and

$$E'(k) := E(\sqrt{1-k^2}) = E(k'),$$

so in the context of elliptic integrals a prime ($'$) does *not* denote differentiation. Apologies for any confusion, but this is the convention that is used in the literature, including *Pi and the AGM*.

On the occasions when we need a derivative, we use operator notation $\mathrm{D}_k K(k) := dK(k)/dk$.

Pi and the AGM uses the "dot" notation $\dot{K}(k) := dK(k)/dk$, but this is confusing and hard to see, so we'll avoid it.

k and k' can in general be complex, but for the moment we'll assume that they are real and in the interval $(0, 1)$.

What’s in a name?

The arc-length L of an ellipse with semi-major axis a and semi-minor axis b is given by

$$L = 4\int_0^{\pi/2} \sqrt{a^2\cos^2\theta + b^2\sin^2\theta}\, d\theta = 4aE'(b/a).$$

Elliptic functions arise by inverting (incomplete) elliptic integrals.

Elliptic curves are named because of their connection with elliptic functions.

Connection with hypergeometric functions

In terms of the Gaussian hypergeometric function

$$F(a,b;c;z) := 1 + \frac{a \cdot b}{1! \cdot c}\, z + \frac{a(a+1) \cdot b(b+1)}{2! \cdot c(c+1)}\, z^2 + \cdots$$

we have

$$K(k) = \frac{\pi}{2}\, F\left(\tfrac{1}{2}, \tfrac{1}{2}; 1; k^2\right)$$

and

$$E(k) = \frac{\pi}{2}\, F\left(-\tfrac{1}{2}, \tfrac{1}{2}; 1; k^2\right).$$

We also have

$$K'(k) = \frac{2}{\pi} \log\left(\frac{4}{k}\right) K(k) - f(k),$$

where $f(k) = k^2/4 + O(k^4)$ is analytic in the disk $|k| < 1$.
Note: in this talk, log always denotes the natural logarithm.

The AGM and elliptic integrals

Substituting $(a,b) \mapsto (1,k)$ above, and recalling that $k^2 + (k')^2 = 1$, we have

$$\begin{aligned}\frac{1}{\mathrm{AGM}(1,k)} &= \frac{2}{\pi}\int_0^{\pi/2} \frac{d\theta}{\sqrt{\cos^2\theta + k^2\sin^2\theta}}\\ &= \frac{2}{\pi}\int_0^{\pi/2} \frac{d\theta}{\sqrt{1-(1-k^2)\sin^2\theta}}\\ &= \frac{2}{\pi}\int_0^{\pi/2} \frac{d\theta}{\sqrt{1-(k')^2\sin^2\theta}}\\ &= \frac{2}{\pi}K'(k),\end{aligned}$$

so

$$\mathrm{AGM}(1,k) = \frac{\pi}{2K'(k)}\,.$$

Computing both E' and K' via the AGM

We have seen that, if we start from $a_0 = 1$, $b_0 = k \in (0, 1)$ and apply the AGM iteration, then $K'(k)$ can be computed from

$$\lim_{n\to\infty} a_n = \frac{\pi}{2K'(k)} \,.$$

We also have

$$\frac{E'(k)}{K'(k)} = \frac{1+k^2}{2} - \sum_{n=0}^{\infty} 2^n \left(a_n - a_{n+1}\right)^2,$$

so $E'(k)$ can be computed at the same time as $K'(k)$.

Logarithms and the AGM

Recall that, for small k, we have

$$K'(k) = \frac{2}{\pi} \log\left(\frac{4}{k}\right) K(k) + O(k^2),$$

but

$$\frac{2}{\pi} K(k) = F\left(\tfrac{1}{2}, \tfrac{1}{2}; 1; k^2\right) = 1 + O(k^2).$$

Thus, assuming that $k \in (0, 1)$, we have

$$K'(k) = \left(1 + O(k^2)\right) \log\left(\frac{4}{k}\right).$$

An explicit bound on the $O(k^2)$ term is given in Thm. 7.2 of *Pi and the AGM*.

First attempt to compute π via the AGM

Choose $k := 2^{2-n}$ for some sufficiently large positive integer n. Then

$$\log\left(\frac{4}{k}\right) = n\log 2,$$

but

$$\frac{\pi}{2\,\mathrm{AGM}(1,k)} = K'(k) = \left(1 + O(k^2)\right)\log\left(\frac{4}{k}\right),$$

which gives

$$\frac{\pi}{\log 2} = 2n\,\mathrm{AGM}(1,k)\ \left(1 + O(4^{-n})\right).$$

Thus, we can compute $\pi/\log 2$ to $(2n + O(1))$-bit accuracy using an AGM computation. Similarly for $\pi/\log 3$, etc.

Historical notes

The algorithm for $\pi/\log 2$ was essentially given by Salamin in HAKMEM (1972), pg. 71, although presented as an algorithm for computing $\log(4/k)$, assuming that we know π.

On the same page Salamin gives an algorithm for computing π, taking $k = 4/e^n$ instead of our $k = 4/2^n$. With his choice $\pi \approx 2n\,\mathrm{AGM}(1, k)$. However, this assumes that we know e, so it is not a "standalone" algorithm for π via the AGM.

In 1975, Salamin (and independently the speaker) discovered an algorithm for computing π via the AGM *without* needing to know e or $\log 2$ to high precision. It is called the "Gauss-Legendre" or "Brent-Salamin" algorithm, and is about twice as fast as the algorithm given in HAKMEM (1972).

In 1984, Jon and Peter Borwein discovered another quadratically convergent algorithm for computing π, with about the same speed as the Gauss-Legendre algorithm. We'll describe the Gauss-Legendre and Borwein-Borwein algorithms shortly.

Legendre's relation

The Gauss-Legendre algorithm takes advantage of a nice identity known as *Legendre's relation*: for $0 < k < 1$,

$$E(k)K'(k) + E'(k)K(k) - K(k)K'(k) = \frac{\pi}{2} \cdot$$

For a proof, see *Pi and the AGM*, Sec. 1.6.

A special case, obtained by taking $k = k' = 1/\sqrt{2}$, is

$$\left(2E(1/\sqrt{2}) - K(1/\sqrt{2})\right) K(1/\sqrt{2}) = \frac{\pi}{2} \cdot$$

Aside: it can be shown (*Pi and the AGM*, Thm. 1.7) that

$$K(1/\sqrt{2}) = \frac{\Gamma^2\left(\frac{1}{4}\right)}{4\pi^{1/2}} \quad \text{and} \quad 2E(1/\sqrt{2}) - K(1/\sqrt{2}) = \frac{\Gamma^2\left(\frac{3}{4}\right)}{\pi^{1/2}} \cdot$$

A quadratically convergent algorithm for π

Using Legendre's relation and the formulas that we've given for E and K in terms of the AGM iteration, it is not difficult to derive the **Gauss-Legendre [Brent-Salamin]** algorithm.

Set $a_0 = 1$, $b_0 = 1/\sqrt{2}$, $s_0 = \frac{1}{4}$ and iterate (for $n = 0, 1, \ldots$)

$$a_{n+1} = \frac{a_n + b_n}{2}\,, \;\; b_{n+1} = \sqrt{a_n b_n}, \;\; s_{n+1} = s_n - 2^n\,(a_n - a_{n+1})^2.$$

Then we get upper and lower bounds on π:

$$\frac{a_n^2}{s_n} > \pi > \frac{a_{n+1}^2}{s_n}\,,$$

and both bounds converge quadratically to π. The lower bound is more accurate, so the algorithm is often stated with just the lower bound a_{n+1}^2/s_n.

Gauss and Legendre

Gauss c. 1828

Louis Legendre (1752-1797) politician
unrelated to the mathematician

Adrien-Marie Legendre
Mathematician (1752-1833)

Legendre

How fast does it converge?

n	a_{n+1}^2/s_n		a_n^2/s_n
0 :	2.914213562373095048801689	$< \pi <$	4.000000000000000000000000
1 :	3.140579250522168248311331	$< \pi <$	3.187672642712108627201930
2 :	3.141592646213542282149344	$< \pi <$	3.141680293297653293918070
3 :	3.141592653589793238279513	$< \pi <$	3.141592653895446496002915
4 :	3.141592653589793238462643	$< \pi <$	3.141592653589793238466361

Compare Archimedes:

0 :	3.0000000	$< \pi <$	3.4641017
1 :	3.1058285	$< \pi <$	3.2153904
2 :	3.1326286	$< \pi <$	3.1596600
3 :	3.1393502	$< \pi <$	3.1460863
4 :	3.1410319	$< \pi <$	3.1427146
		$\cdots$	
37 :	3.141592653589793238462636	$< \pi <$	3.141592653589793238462659

Jacobi theta functions

To estimate the speed of convergence and, more precisely, to obtain upper and lower bounds on the error after n iterations, we consider the parameterisation of the AGM in terms of *Jacobi theta functions*.

> *When I was a student, abelian functions were, as an effect of the Jacobian tradition, considered the uncontested summit of mathematics, and each of us was ambitious to make progress in this field. And now? The younger generation hardly knows abelian functions.*
>
> Felix Klein
>
> *Development of Mathematics in the 19th Century*, 1928.

We are all "younger generation" now.

Theta functions and the AGM

We need the basic theta functions of one variable defined by

$$\theta_3(q) := \sum_{n\in\mathbb{Z}} q^{n^2}, \quad \theta_4(q) := \sum_{n\in\mathbb{Z}} (-1)^n q^{n^2}, \quad |q| < 1.$$

It is not difficult to show that

$$\frac{\theta_3^2(q) + \theta_4^2(q)}{2} = \theta_3^2(q^2) \text{ and } \sqrt{\theta_3^2(q)\theta_4^2(q)} = \theta_4^2(q^2),$$

which shows that the AGM variables (a_n, b_n) can, if scaled suitably, be parameterised by $(\theta_3^2(q^{2^n}), \theta_4^2(q^{2^n}))$.

Theta functions and the AGM

If $1 = a_0 > b_0 = \theta_4^2(q)/\theta_3^2(q) > 0$, where $q \in (0,1)$, then the variables a_n, b_n appearing in the AGM iteration satisfy

$$a_n = \frac{\theta_3^2(q^{2^n})}{\theta_3^2(q)}, \quad b_n = \frac{\theta_4^2(q^{2^n})}{\theta_3^2(q)}.$$

We can write q (which is called the *nome*) explicitly in terms of the elliptic integral K with $k' = b_0/a_0$, in fact

$$q = \exp(-\pi K'(k)/K(k)).$$

This is due to Gauss/Jacobi.

A useful special case is $k = k` = 1/\sqrt{2}$. Then $K' = K$ and

$$q = e^{-\pi} = 0.0432139\ldots$$

Theta functions and the AGM

Recall that in the Gauss-Legendre algorithm we have $a_0 = 1$, $b_0 = 1/\sqrt{2}$, $s_0 = \frac{1}{4}$ and, for $n \ge 0$,

$$a_{n+1} = \frac{a_n + b_n}{2}, \;\; b_{n+1} = \sqrt{a_n b_n}, \;\; s_{n+1} = s_n - 2^n \left(a_n - a_{n+1}\right)^2.$$

Take $q = e^{-\pi}$, and write

$$a_\infty := \lim_{n\to\infty} a_n = \theta_3^{-2}(q) = 2\pi^{3/2}/\Gamma^2(\tfrac{1}{4}) \approx 0.8472,$$
$$s_\infty := \lim_{n\to\infty} s_n = \theta_3^{-4}(q)/\pi = 4\pi^2/\Gamma^4(\tfrac{1}{4}) \approx 0.2285\,.$$

It is curious that the algorithm computes π as the ratio of a_∞^2 and s_∞, both of which appear more “complicated” than π.

As on the previous slide, $a_n = \theta_3^2(q^{2^n})/\theta_3^2(q)$, and thus

$$s_n - s_\infty = \theta_3^{-4}(q) \sum_{m=n}^{\infty} 2^m \left(\theta_3^2(q^{2^m}) - \theta_3^2(q^{2^{m+1}})\right)^2.$$

Remark

The expression for $s_n - s_\infty$ can be "simplified" if we use the theta function

$$\theta_2(q) := \sum_{n\in\mathbb{Z}} q^{(n+1/2)^2}.$$

Jacobi's identity

$$\theta_3^4(q) = \theta_2^4(q) + \theta_4^4(q)$$

connects θ_2, θ_3 and θ_4. Using it, we see that

$$\theta_3^2(q) - \theta_3^2(q^2) = \frac{\theta_2^4(q)}{4\theta_3^2(q^2)} \,,$$

so

$$s_n - s_\infty = \theta_3^{-4}(q) \sum_{m=n}^{\infty} 2^{m-4} \frac{\theta_2^8(q^{2^m})}{\theta_3^4(q^{2^{m+1}})} \,.$$

Theta functions and the AGM

Write $a_n/a_\infty = 1 + \delta_n$ and $s_n/s_\infty = 1 + \varepsilon_n$. Then

$$\delta_n = \theta_3^2(q^{2^n}) - 1 \sim 4q^{2^n} \quad \text{as} \quad n \to \infty,$$

and

$$\varepsilon_n = \frac{\pi}{16} \sum_{m=n}^{\infty} 2^m \, \theta_2^8(q^{2^m}) \, \theta_3^{-4}(q^{2^{m+1}}) \sim 2^{n+4} \pi q^{2^{n+1}}.$$

Upper and lower bounds

Writing

$$\frac{a_n^2/a_\infty^2}{s_n/s_\infty} = \frac{a_n^2}{\pi s_n} = \frac{(1+\delta_n)^2}{1+\varepsilon_n},$$

it is straightforward to obtain an upper bound on π:

$$0 < a_n^2/s_n - \pi < U(n) := 8\pi q^{2^n}.$$

Convergence is quadratic: if $e_n := a_n^2/s_n - \pi$, then

$$\lim_{n\to\infty} e_{n+1}/e_n^2 = \tfrac{1}{8\pi}\,.$$

Replacing a_n by a_{n+1} and δ_n by δ_{n+1}, we obtain a lower bound (after $n+1$ square roots)

$$0 < \pi - \frac{a_{n+1}^2}{s_n} < L(n) := (2^{n+4}\pi^2 - 8\pi)q^{2^{n+1}}.$$

Equivalently, after $n > 0$ square roots we have

$$0 < \pi - \frac{a_n^2}{s_{n-1}} < (2^{n+3}\pi^2 - 8\pi)q^{2^n}.$$

Remark

Pi and the AGM, and also Salamin (1976), give a slightly weaker lower bound

$$\pi - \frac{a_{n+1}^2}{s_n} < \frac{2^{n+4}\pi^2 q^{2^{n+1}}}{a_\infty^2}.$$

Compare our

$$\pi - \frac{a_{n+1}^2}{s_n} < (2^{n+4}\pi^2 - 8\pi)q^{2^{n+1}}.$$

Note that $a_\infty^{-2} \approx 1.3932$.
The factor $(2^{n+4}\pi^2 - 8\pi)$ is best possible, since

$$\pi - \frac{a_{n+1}^2}{s_n} = (2^{n+4}\pi^2 - 8\pi)q^{2^{n+1}} - O(2^n q^{2^{n+2}}).$$

Numerical values of upper and lower bounds

n	$a_n^2/s_n - \pi$	$\pi - a_{n+1}^2/s_n$	$\frac{a_n^2/s_n - \pi}{U(n)}$	$\frac{\pi - a_{n+1}^2/s_n}{L(n)}$
0	8.58e-1	2.27e-1	0.790369040	0.916996189
1	4.61e-2	1.01e-3	0.981804947	0.999656206
2	8.76e-5	7.38e-9	0.999922813	0.999999998
3	3.06e-10	1.83e-19	0.999999999	1.000000000
4	3.72e-21	5.47e-41	1.000000000	1.000000000
5	5.50e-43	2.41e-84	1.000000000	1.000000000
6	1.20e-86	2.31e-171	1.000000000	1.000000000
7	5.76e-174	1.06e-345	1.000000000	1.000000000
8	1.32e-348	1.11e-694	1.000000000	1.000000000

$U(n) := 8\pi \exp(-2^n\pi)$ and $L(n) := (2^{n+4}\pi^2 - 8\pi)\exp(-2^{n+1}\pi)$ are the bounds given above. It can be seen that they are very accurate, as expected from our analysis. In fact, they are sharp enough to suggest an AGM algorithm (though not the best one) for computing $e^{\pm\pi}$.

A family of algorithms for π

Recall that the Gauss-Legendre algorithm yields an approximation a_n^2/s_n [or a_{n+1}^2/s_n] to $\pi = a_\infty^2/s_\infty$.

Using the expressions for a_n and s_n in terms of theta functions, we see that

$$\pi = \frac{a_n^2\, \theta_3^{-4}(q^{2^n})}{s_n - \theta_3^{-4}(q)\sum_{m=n}^{\infty} 2^{m-4}\, \theta_2^8(q^{2^m})\theta_3^{-4}(q^{2^{m+1}})}$$

[or similarly with the numerator replaced by $a_{n+1}^2\theta_3^{-4}(q^{2^{n+1}})$].
This gives a family of algorithms for π (two for each $n \geq 0$).

The expression for π is essentially of the form

$$\pi = \frac{a_n^2 - O(q^{2^n})}{s_n - O(2^n q^{2^{n+1}})} \quad \left[\text{or } \frac{a_{n+1}^2 - O(q^{2^{n+1}})}{s_n - O(2^n q^{2^{n+1}})}\right]$$

and shows precisely how the algorithms approximate π and why they provide upper [lower] bounds.

A drawback of these algorithms is that they require a sufficiently accurate approximation to $q = e^{-\pi}$.

The Borwein^2 quadratic AGM algorithm for π

In *Pi and the AGM*, Jon and Peter Borwein present a *different* quadratically convergent algorithm for π based on the AGM. (It is Algorithm 2.1 in Chapter 2, and was first published in 1984.)

Instead of using Legendre's relation, the Borwein-Borwein algorithm uses the identity

$$K(k)\, \mathrm{D}_k K(k)\Big|_{k=1/\sqrt{2}} = \frac{\pi}{\sqrt{2}}\,,$$

where D_k denotes differentiation with respect to k.

Using the connection between $K(k')$ and the AGM, we obtain

$$\pi = 2^{3/2} \left.\frac{(\mathrm{AGM}(1,k'))^3}{\mathrm{D}_k\, \mathrm{AGM}(1,k')}\right|_{k=1/\sqrt{2}} .$$

An algorithm for approximating the derivative in this formula can be obtained by differentiating the AGM iteration symbolically. Details are given in *Pi and the AGM*.

The Borwein^2 quadratic AGM algorithm for π

The Borwein-Borwein algorithm (Alg. 2.1 of *Pi and the AGM*):

$$x_0 := \sqrt{2};\ \ y_1 := 2^{1/4};\ \ \underline{\pi}_0 := \sqrt{2};\ \ \overline{\pi}_0 := 2 + \sqrt{2};$$
$$\text{for } n \geq 0,\ x_{n+1} := \tfrac{1}{2}(x_n^{1/2} + x_n^{-1/2});$$
$$\text{for } n \geq 1,\ y_{n+1} := \frac{y_n\, x_n^{1/2} + x_n^{-1/2}}{y_n + 1};$$
$$\text{for } n \geq 1,\ \underline{\pi}_n := \frac{2\,\overline{\pi}_{n-1}}{y_n + 1},\ \ \overline{\pi}_n := \underline{\pi}_n \left(\frac{x_n + 1}{2}\right).$$

Then $\overline{\pi}_n$ decreases monotonically to π, and $\underline{\pi}_n$ increases monotonically to π. (The algorithm given in *Pi and the AGM* defines $\overline{\pi}_n := \overline{\pi}_{n-1}(x_n + 1)/(y_n + 1)$ and omits $\underline{\pi}_n$.)

The AGM iteration is present in *Legendre form*: if $a_0 := 1$, $b_0 := k' = 1/\sqrt{2}$, and we perform the AGM iteration, then $x_n = a_n/b_n$ and, for $n \geq 1$, $y_n = \mathrm{D}_k b_n / \mathrm{D}_k a_n$.

How fast does Borwein-Borwein converge?

n	$\underline{\pi}_n$		$\overline{\pi}_n$
0 :	1.414213562373095048801689	$< \pi <$	3.414213562373095048801689
1 :	3.119132528827772757303373	$< \pi <$	3.142606753941622600790720
2 :	3.141548837729436193482357	$< \pi <$	3.141592660966044230497752
3 :	3.141592653436966609787790	$< \pi <$	3.141592653589793238645774
4 :	3.141592653589793238460785	$< \pi <$	3.141592653589793238462643

Compare **Gauss-Legendre:**

n	a_{n+1}^2/s_n		a_n^2/s_n
0 :	2.914213562373095048801689	$< \pi <$	4.000000000000000000000000
1 :	3.140579250522168248311331	$< \pi <$	3.187672642712108627201930
2 :	3.141592646213542282149344	$< \pi <$	3.141680293297653293918070
3 :	3.141592653589793238279513	$< \pi <$	3.141592653895446496002915
4 :	3.141592653589793238462643	$< \pi <$	3.141592653589793238466361

Borwein-Borwein gives better upper bounds, but worse lower bounds, for the same value of *n* (i.e. same number of sqrts).

Bounding the error using theta functions

As for the Gauss-Legendre algorithm, we can express the error after *n* iterations of the Borwein-Borwein algorithm using theta functions, and deduce the asymptotic behaviour of the error.

Consider the AGM iteration with $a_0 = 1, b_0 = k' = (1 - k^2)^{1/2}$. Then a_n and b_n are functions of k, and we denote the derivative of a_n with respect to k by $\mathrm{D}_k a_n$. In *Pi and the AGM* it is shown that, for $n \geq 1$,

$$\overline{\pi}_{n-1} = \left(2^{3/2} b_n^2 a_n / \mathrm{D}_k a_n\right) \big|_{k=1/\sqrt{2}} .$$

As before, for $q = e^{-\pi}$,

$$a_n = \frac{\theta_3^2(q^{2^n})}{\theta_3^2(q)} , \text{ and } b_n = \frac{\theta_4^2(q^{2^n})}{\theta_3^2(q)} .$$

Thus, we have to differentiate the expression for a_n with respect to k, where $k = (1 - b_0^2)^{1/2} = \theta_2^2(q)/\theta_3^2(q)$.

Bounding the error

With some care, and simplifying the result in the same manner as for the Gauss-Legendre algorithm, differentiation gives

$$\mathrm{D}_k a_n = \sum_{m=1}^{n} \mathrm{D}_q \left(\frac{\theta_2^4(q^{2^{m-1}})}{4\theta_3^2(q)\theta_3^2(q^{2^m})} \right) \Big/ \mathrm{D}_q \left(\frac{\theta_2^2(q)}{\theta_3^2(q)} \right) \Bigg|_{q=e^{-\pi}} .$$

We denote the limit as $n \to \infty$ by $\mathrm{D}_k a_\infty$. In fact,

$$\mathrm{D}_k a_\infty = \frac{2^{3/2} a_\infty^3}{\pi} = 0.547486\ldots$$

Now we can write

$$\mathrm{D}_k a_\infty = \mathrm{D}_k a_n + \sum_{m=n+1}^{\infty} \mathrm{D}_q \left(\frac{\theta_2^4(q^{2^{m-1}})}{4\theta_3^2(q)\theta_3^2(q^{2^m})} \right) \Big/ \mathrm{D}_q \left(\frac{\theta_2^2(q)}{\theta_3^2(q)} \right) \Bigg|_{q=e^{-\pi}} .$$

Bounding the error

The denominator is a constant which can be evaluated as

$$c := \mathrm{D}_q \left(\frac{\theta_2^2(q)}{\theta_3^2(q)} \right) \Bigg|_{q=e^{-\pi}} = \frac{e^{\pi}\Gamma^4(\frac{1}{4})}{16\pi^3\sqrt{2}} = 5.699228\ldots$$

Putting these pieces together gives an upper bound (for $n \geq 1$)

$$0 < \overline{\pi}_n - \pi < 2^{n+4}\pi^2 q^{2^{n+1}},$$

and a lower bound

$$0 < \pi - \underline{\pi}_n < 4\pi q^{2^n},$$

where $q = e^{-\pi}$.

These can be compared with the *lower bound* $2^{n+4}\pi^2 q^{2^{n+1}}$ and *upper bound* $8\pi q^{2^n}$ for the Gauss-Legendre algorithm.

Numerical values of upper and lower bounds

n	$\overline{\pi} - \pi$	ratio to bound	$\pi - \underline{\pi}$	ratio to bound
1	1.01e-3	0.9896487063	2.25e-2	0.9570949132
2	7.38e-9	0.9948470082	4.38e-5	0.9998316841
3	1.83e-19	0.9974691480	1.53e-10	0.9999999988
4	5.47e-41	0.9987456847	1.86e-21	1.0000000000
5	2.41e-84	0.9993755837	2.75e-43	1.0000000000
6	2.31e-171	0.9996884727	6.01e-87	1.0000000000
7	1.06e-345	0.9998444059	2.88e-174	1.0000000000
8	1.11e-694	0.9999222453	6.59e-349	1.0000000000

It can be seen that the bounds are very accurate (as expected from the exact expressions for the errors in terms of theta functions). The upper bound overestimates the error by a factor of $1 + O(2^{-n})$.

A fourth-order algorithm for π

The Borwein brothers did not stop at quadratic (second-order) algorithms for π. In Chapter 5 of *Pi and the AGM* they gave algorithms of orders 3, 4, 5 and 7. Here is a nice iteration of order 4, derived using a modular identity of order 4.

$$\begin{aligned} y_0 &:= \sqrt{2} - 1; \quad a_0 := 2y_0^2; \\ y_{n+1} &:= \frac{1 - (1 - y_n^4)^{1/4}}{1 + (1 - y_n^4)^{1/4}}\,; \\ a_{n+1} &:= a_n(1 + y_{n+1})^4 - 2^{2n+3} y_{n+1}(1 + y_{n+1} + y_{n+1}^2). \end{aligned}$$

Then $\pi_n := 1/a_n$ converges quartically to π, i.e. the number of correct digits is multiplied by (approximately) 4 each iteration!

An error bound is

$$0 < \pi - \pi_n < 4\pi^2\, 4^{n+1} \exp(-2\pi\, 4^n).$$

Convergence of the quartic algorithm

The table shows the error $\pi - \pi_n$ after n iterations of the Borwein quartic algorithm, and the ratio

$$\frac{\pi - \pi_n}{4\pi^2\, 4^{n+1} \exp(-2\pi\, 4^n)}$$

of the error to the upper bound given on the previous slide.

n	$\pi - \pi_n$	$(\pi - \pi_n)/\text{bound}$
0	2.273790912e-1	0.7710517124
1	7.376250956e-9	0.9602112619
2	5.472109145e-41	0.9900528160
3	2.308580715e-171	0.9975132040
4	1.110954934e-694	0.9993783010
5	9.244416653e-2790	0.9998445753
6	6.913088685e-11172	0.9999611438
7	3.376546688e-44702	0.9999902860
8	3.002256862e-178825	0.9999975715

A cubic algorithm for π

Here is an iteration of order 3:

$$\begin{aligned} a_0 &:= 1/3; \quad s_0 := (\sqrt{3}-1)/2; \\ r_{n+1} &:= 3/(1+2(1-s_n^3)^{1/3}); \\ s_{n+1} &:= (r_{n+1}-1)/2; \\ a_{n+1} &:= r_{n+1}^2 a_n - 3^n(r_{n+1}^2 - 1). \end{aligned}$$

Then $\pi_n := 1/a_n$ converges cubically to π.

The proof uses modular equations (*Pi and the AGM*, Chs. 4–5).

An empirical error bound (a slightly weaker result has been proved) is

$$0 < \pi - \pi_n < 4\pi^2\, 3^{n+1} \exp(-2\pi\, 3^n).$$

Compare the bound for the quartic algorithm:

$$0 < \pi - \pi_n < 4\pi^2\, 4^{n+1} \exp(-2\pi\, 4^n).$$

An observation

After $2n$ iterations of the Gauss-Legendre algorithm we have an (accurate) error bound

$$0 < \pi - a_{2n+1}^2/s_{2n} < 4\pi^2\, 4^{n+1} \exp(-2\pi\, 4^n).$$

This is the same as the (accurate) error bound

$$0 < \pi - \pi_n < 4\pi^2\, 4^{n+1} \exp(-2\pi\, 4^n);$$

for the Borwein quartic algorithm!

On closer inspection we find that the two algorithms (Gauss-Legendre "doubled" and Borwein quartic) are *equivalent*, in the sense that they give *exactly* the same sequence of approximations to π. This is perhaps implicit in *Pi and the AGM*, but I have not seen it stated explicitly.

Verification

Here k is the number of square roots, and "$\pi -$ approximation" is the error in the approximation given by the Gauss-Legendre algorithm after $k-1$ iterations, or by the Borweins' quartic algorithm after $(k-1)/2$ iterations. The error is the same for both algorithms (computed to 1000 decimal digits).

k	$\pi -$ approximation
1	2.2737909121669818966095465906980480562749752399816e-1
3	7.3762509563132989512968071098827321760295030264154e-9
5	5.4721091456899418327485331789641785565936917028248e-41
7	2.3085807149343902668213207343869568303303472423996e-171
9	1.1109549335576998257002904117322306941479378545140e-694

Verification continued

For example, the first line follows from

$$\begin{aligned} a_1^2/s_0 = \pi_0 &= \tfrac{3}{2} + \sqrt{2} \\ &\approx \pi - 0.227 \end{aligned}$$

and the second line follows from

$$\begin{aligned} a_3^2/s_2 = \pi_1 &= \frac{(2^{-2} + 2^{-5/2} + 2^{-5/4} + \sqrt{2^{-5/4} + 2^{-7/4}})^2}{2^{3/4} + 2^{1/4} - 2^{-1/2} - \frac{5}{4}} \\ &\approx \pi - 7.376... \times 10^{-9}. \end{aligned}$$

Sketch of proof

As noted on page 171 of *Pi and the AGM*, the Borwein quartic iteration is equivalent to doubling a certain quadratic iteration (5.2 on page 170 with the parameter $r = 4$) that may be written as

$$\alpha_0 := 6 - 4\sqrt{2}; \quad k_0 := 3 - 2\sqrt{2};$$
$$k_n' = \sqrt{1 - k_n^2}; \quad k_{n+1} := \frac{1 - k_n'}{1 + k_n'};$$
$$\alpha_{n+1} := (1 + k_{n+1})^2 \alpha_n - 2^{n+2} k_{n+1}.$$

Then $\alpha_n \to 1/\pi$ with quadratic convergence.

Thus, it is sufficient to show that this iteration 5.2 is equivalent to the Gauss-Legendre algorithm. This can be done directly by induction on *n*. (Note that after one iteration of Gauss-Legendre we have $k = (1 - b_1^2/a_1^2)^{1/2} = 3 - 2\sqrt{2}$, so the subscripts are displaced by one relative to iteration 5.2.)

Some other fast algorithms for π

Let $(x)_n := x(x+1)\cdots(x+n-1)$ denote the *ascending factorial*. In Chapter 5 of *Pi and the AGM*, Jon and Peter Borwein discuss *Ramanujan-Sato* series such as

$$\frac{1}{\pi} = 2^{3/2} \sum_{n=0}^{\infty} \frac{(\frac{1}{4})_n (\frac{1}{2})_n (\frac{3}{4})_n}{(n!)^3} \, \frac{(1103 + 26390n)}{99^{4n+2}} \,.$$

This is linearly convergent, but adds nearly eight decimal digits per term, since $99^4 \approx 10^8$.

A more extreme example is the Chudnovsky series

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

which adds about 14 decimal digits per term.

Remarks on complexity

We consider the *bit complexity* of an algorithm. This is the (worst case) number of single-bit operations required to complete the algorithm. For a fuller discussion, see Chapter 6 of *Pi and the AGM*. We are interested in asymptotic results, so are usually willing to ignore constant factors.

If all operations are performed to (approximately) the same precision, then it makes sense to count *operations* such as multiplications, divisions and square roots. Algorithms based on the AGM fall into this category.

If the precision of the operations varies widely, then bit-complexity is a more sensible measure of complexity. An example is Newton's method, which is self-correcting, so can be started with low precision. Another example is summing a series with rational terms, such as

$$e = \sum_{k=0}^{\infty} \frac{1}{k!} \, .$$

Complexity of multiplication

The complexity of multiplying two n-bit numbers to obtain a $2n$-bit product is denoted by $M(n)$. The classical algorithm shows that $M(n) = O(n^2)$, but various asymptotically faster algorithms exist. They are associated with the names [Gauss, Kronecker, Cooley, Tukey,] Karatsuba, Toom, Cook, Schönhage, Strassen, Fürer, Harvey, van der Hoeven, and Lecerf, in roughly chronological order.

The best result so far [Harvey et al, 2016] is

$$M(n) = O\left(n \log n \, K^{\log^* n}\right)$$

with $K = 8$. Here the *iterated logarithm* function $\log^* n$ is defined by

$$\log^* n := \begin{cases} 0 & \text{if } n \le 1; \\ 1 + \log^*(\log n) & \text{if } n > 1. \end{cases}$$

It is unbounded but grows *extremely* slowly, e.g. slower than $\log \log \cdots \log n$ [any fixed number of logs], as $n \to \infty$.

Complexity of division, root extraction, etc

We'll follow *Pi and the AGM* and assume that $M(n)$ is nondecreasing and satisfies the regularity condition

$$2M(n) \leq M(2n) \leq 4M(n).$$

Newton's method can be used to compute reciprocals and square roots with bit-complexity

$$O\left(M(n) + M(\lceil n/2 \rceil) + M\left(\left\lceil n/2^2 \right\rceil\right) + \cdots + M(1)\right) = O(M(n)).$$

Using this technique, it can be shown that the bit complexities of squaring, multiplication, reciprocation, division, and root extraction are asymptotically the same, up to small constant factors. All these operations have complexity of order $M(n)$.

Complexity of summing certain series

To compute π to n digits (binary or decimal) by Machin's arctan formula

$$\pi = 16 \arctan(1/5) - 4 \arctan(1/239),$$

or to compute $1/\pi$ by the Chudnovsky series

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

we have to sum of order n terms.

Using *divide and conquer* (also called *binary splitting* or *FEE*) this can be done with bit-complexity

$$O(M(n) \log^2 n).$$

Complexity of computing π via the AGM

Suppose we compute π to n-digit accuracy using one of the quadratically convergent AGM algorithms. This requires $O(\log n)$ iterations, each of which has complexity $O(M(n))$. Thus, the overall complexity is

$$O(M(n)\log n).$$

This is (theoretically) better than series summation methods, which have complexity of order $M(n)\log^2 n$.

In practice, constant factors are important, and a method with complexity of order $M(n)\log^2 n$ may be faster than a method with complexity $O(M(n)\log n)$ unless n is sufficiently large. This is one reason for the recent popularity of the Chudnovsky series for high-precision computation of π, even though the AGM methods are theoretically (i.e. asymptotically) more efficient.

Computing $\log x$ via the AGM

We already mentioned Salamin's algorithm for computing $\log x$ for sufficiently large $x = 4/k$ (i.e. sufficiently small k) using

$$K'(k) = \left(1 + O(k^2)\right) \log\left(\frac{4}{k}\right).$$

We can evaluate $K'(k)/\pi$ using the AGM with $(a_0, b_0) = (1, k)$, and hence approximate $\log(4/k)$, assuming that π is precomputed.

To compute $\log x$ to n-bit accuracy requires about $2\log_2(n)$ AGM iterations, or $3\log_2(n)$ iterations if we count the computation of π.

If x is not sufficiently large, first multiply it by a suitable power of two, say 2^p, and later subtract $p \log 2$. This assumes that $\log 2$ is precomputed, and that the precision is increased to compensate for cancellation. If it would be faster, use the Taylor series, e.g. when $|x - 1| < 2^{-n/\log n}$.

An exact formula for $\log x$

The error term in the expression

$$\log(4/k) = (1 + O(k^2))K'(k)$$

can be written explicitly using hypergeometric series. We give an alternative using theta functions, for which the series converge faster. The ingredients are various identities that we have seen already:

$$\begin{aligned}
\log(1/q) &= \pi K'(k)/K(k), \\
k &= \theta_2^2(q)/\theta_3^2(q), \\
K(k) &= (\pi/2)\,\theta_3^2(q), \\
K'(k) &= (\pi/2)/\mathrm{AGM}(1,k).
\end{aligned}$$

Putting these pieces together gives Sasaki and Kanada's nice formula

$$\log(1/q) = \frac{\pi}{\mathrm{AGM}(\theta_2^2(q), \theta_3^2(q))}.$$

Improved AGM algorithm for log

Replacing q by q^4 to avoid fractional powers of q in the expansion of $\theta_2(q)$, we obtain

$$\log x = \frac{\pi/4}{\mathrm{AGM}(\theta_2^2(q^4), \theta_3^2(q^4))},$$

where $q = 1/x$ and we assume that $x > 1$.

As in Salamin's algorithm, we have to ensure that x is sufficiently large, but now there is a trade-off between increasing x or taking more terms in the series defining the theta functions. For example, if $x > 2^{n/36}$, we can use

$$\theta_2(q^4) = 2(q + q^9 + q^{25} + O(q^{49})),$$
$$\theta_3(q^4) = 1 + 2(q^4 + q^{16} + O(q^{36})).$$

This saves about four AGM iterations, compared to Salamin's algorithm.

The complex AGM

So far we have assumed that the initial values a_0, b_0 in the AGM iteration are real and positive. There is no difficulty in extending the results that we have used to complex a_0, b_0, provided that they are nonzero and a_0/b_0 is not both real and negative. For simplicity, we'll assume that $a_0, b_0 \in \mathcal{H} = \{z \mid \Re(z) > 0\}$.

In the AGM iteration (and in the definition of the geometric mean) there is an ambiguity of sign. We always choose the square root with positive real part. Thus the iterates a_n, b_n are uniquely defined and remain in the right half-plane $\mathcal{H}$.

In the Sasaki-Kanada algorithm we no longer replace q by q^4 if this would give starting values $(\theta_2^2(q^4), \theta_3^2(q^4))$ for the AGM outside of $\mathcal{H}$.

Computing other elementary functions via the AGM

Recall that, for $z \in \mathbb{C}\backslash\{0\}$, $\log(z) = \log(|z|) + i\arg(z)$, provided we use the principal values of the logarithms.

Thus, if $x \in \mathbb{R}$, we can use the complex AGM to compute

$$\arctan(x) = \Im(\log(1 + ix)).$$

$\arcsin(x), \arccos(x)$ etc can be computed via arctan using elementary trigonometric identities such as $\arccos(x) = \arctan(\sqrt{1 - x^2}/x)$.

Since we can compute $\log, \arctan, \arccos, \arcsin$, we can compute $\exp, \tan, \cos, \sin$ (in suitably restricted domains) using Newton's method. Also, of course, $\exp(ix) = \cos(x) + i\sin(x)$ gives another way of computing the trigonometric functions.

Similarly for the hyperbolic functions $\cosh, \sinh, \tanh$ and their inverse functions.

Avoiding complex arithmetic

Although computing the elementary functions via the complex AGM is conceptually straightforward, it introduces the overhead of complex arithmetic. It is possible to avoid complex arithmetic by the use of Landen transformations (which transform incomplete elliptic integrals). See exercise 7.3.2 of *Pi and the AGM* for an outline of this approach.

Whichever approach is used, the bit-complexity of computing n-bit approximations to any of the elementary functions $(\log, \exp, \arctan, \sin, \cos, \tan$, etc) in a given compact set $A \subset \mathbb{C}$ that excludes singularities of the relevant function is $O(M(n) \log n)$. Here "n-bit approximation" means with absolute error bounded by 2^{-n}. We could require relative error bounded by 2^{-n}, but the proof would depend on a Diophantine approximation result such as Mahler's $|\pi - p/q| > 1/q^{42}$.

Computing non-elementary functions via the AGM

Certain non-elementary functions can be computed with complexity $O(M(n) \log n)$ via the AGM. For example, we can compute the elliptic integrals $K(k)$ and $E(k)$, and the Jacobi theta functions $\theta_2(q), \theta_3(q), \theta_4(q)$.

Functions that appear *not* to be in this class of “easily” computable functions include the Gamma function $\Gamma(z)$ and the Riemann zeta function $\zeta(s)$.

Lower bounds

It is plausible to conjecture that $\log(x)$ and the other elementary functions can not be computed with bit-complexity $O(M(n))$ (or anything smaller than $M(n)\log n$). However, as usual in complexity theory, nontrivial lower bounds are difficult to prove.

It is observed in *Pi and the AGM* (page 227) that all algebraic numbers can be computed with bit-complexity $O(M(n))$, so a proof that this is not possible for π would imply that π is transcendental (which we know to be true).

Conclusion

I hope that I have given you some idea of the mathematics contained in the book *Pi and the AGM*. In the time available I have only been able to cover a small fraction of the gems that can be discovered there. If you would like to know more, there is no substitute for reading the book yourself.

It is not an “easy read”, but it is a book that you can put under your pillow, like Dirichlet is said to have done with his copy of *Disquisitiones Arithmeticae*.

Although the research covered in *Pi and the AGM* is only a small fraction of Jon’s legacy, it is the part that overlaps most closely with my own research, which is why I decided to talk about it today.

Jon and Peter Borwein in happier times

References

D. H. Bailey, The computation of π to $29,360,000$ decimal digits using Borweins' quartically convergent algorithm, *Math. Comp.* **50** (1988), 283–296.

D. H. Bailey, Pi and the collapse of peer review, `http://mathscholar.org/pi-and-the-collapse-of-peer-review`, 20 July 2017.

D. H. Bailey and J. M. Borwein, *Pi: The Next Generation*, Springer, 2016.

M. Beeler, R. W. Gosper and R. Schroeppel, *HAKMEM*, AI Memo 239, MIT AI Lab, Feb. 1972. (Entry by E. Salamin.)

J. M. Borwein, The life of pi: from Archimedes to Eniac and beyond, prepared for Berggren Festschrift, 19 June 2012, `https://www.carma.newcastle.edu.au/jon/pi-2012.pdf`

J. M. Borwein and P. B. Borwein, The arithmetic-geometric mean and fast computation of elementary functions, *SIAM Review* **26** (1984), 351–365.

References cont.

J. M. Borwein and P. B. Borwein, *Pi and the AGM*, Monographies et Études de la Société Mathématique du Canada, John Wiley & Sons, Toronto, 1987.

J. M. Borwein, P. B. Borwein and D. H. Bailey, Ramanujan, modular equations, and approximations to pi or how to compute one billion digits of pi, *Amer. Math. Monthly* **96** (1989), 201-219.

J. M. Borwein and P. B. Borwein, A cubic counterpart of Jacobi's identity and the AGM, *Trans. Amer. Math. Soc.*, **323** (1991), 691–701.

R. P. Brent, Multiple-precision zero-finding methods and the complexity of elementary function evaluation, in *Analytic Computational Complexity* (edited by J. F. Traub), Academic Press, New York, 1975, 151–176.

R. P. Brent, Fast multiple-precision evaluation of elementary functions, *J. ACM* **23** (1976), 242–251.

References cont.

R. P. Brent and P. Zimmermann, *Modern Computer Arithmetic*, Cambridge University Press, 2010.

D. V. Chudnovsky and G. V. Chudnovsky, The computation of classical constants, *Proc. Nat. Acad. Sci. USA* **88**(21), 8178–8182.

J. Guillera, New proofs of Borwein-type algorithms for Pi, *Integral Transforms and Special Functions* **27** (2016), 775–782.

D. Harvey, J. van der Hoeven and G. Lecerf, Even faster integer multiplication, *J. Complexity* **36** (2016), 1–30.

E. Salamin, Computation of π using arithmetic-geometric mean, *Math. Comp.* **30** (1976), 565–570.

T. Sasaki and Y. Kanada, Practically fast multiple-precision evaluation of log(x), *J. Inf. Process.* **5** (1982), 247–250.