

$$1 \text{ ly} = 9.4607 \times 10^{15} \text{ m}$$

What is in a quadrillion?  
A tenth of a light year  
is a quadrillion meters.

# 2

## Was ist die billiardste Dezimalstelle von $\pi$ ?

*Es kann keinen praktischen Nutzen haben zu wissen, dass Pi irrational ist, aber wenn wir es wissen können, dann wäre es sicherlich unerträglich, es nicht zu wissen.*

Edward Titchmarsh (1899–1963)

Möglicherweise werden wir niemals die Antwort auf die oben gestellte Frage finden. Auch wenn es einige effiziente Algorithmen zur Berechnung von  $\pi$  gibt, erfordern sie alle die Berechnung der Dezimalziffern der Reihe nach (3,14159 usw.), und selbst mit den leistungsfähigsten Computern, die es gibt, würde es einfach zu lange dauern, die billiardste Stelle zu erreichen. Eine Billiarde sind  $10^{15}$ . Im Moment (Mai 2010) sind die ersten knapp 2,7 Billionen Ziffern von  $\pi$  bekannt (berechnet durch Fabrice Bellard Ende 2009 in 131 Tagen auf einem normalen Desktop-PC). Und falls Sie es schon immer wissen wollten: Die zehn Dezimalziffern bis zur billionsten Stelle sind 6680122702. Wir werden später (in Kapitel 7) auf die Bemühungen zurückkommen, immer mehr Dezimalstellen von  $\pi$  zu berechnen, und einige der dabei benutzten Methoden vorstellen, aber im Moment wollen wir eine etwas andere Frage betrachten: Was ist die billiardste Stelle in der *Binärentwicklung* von  $\pi$ ?

Vermutlich denken Sie, das sei genauso hoffnungslos, und doch ist das nicht der Fall. Wir können Ihnen die Antwort einfach mitteilen: Die billiardste Binärziffer von  $\pi$  ist eine 0.<sup>1</sup> Wenn wir schon dabei sind, verraten wir Ihnen noch ein Geheimnis: Sie könnten der erste Mensch sein, der die billiardste Stelle in der *hexadezimalen* Darstellung von  $\pi$  berechnet; die Methode, die wir gleich beschreiben werden, funktioniert für beide Basen, 2 und 16, aber noch hat sie niemand benutzt, um die billiardste Hexadezimalziffer zu finden. Die Ziffern der Zahl  $\pi$  wären um einiges leichter zu handhaben, wenn nur die Evolution uns Menschen mit zwei oder sechzehn Fingern versorgt hätte!

---

<sup>1</sup> Am Ende dieses Kapitels finden Sie einige Details zu dieser Berechnung.

Der Schlüssel zu dieser bemerkenswerten (wenn auch anscheinend völlig nutzlosen) Information ist die folgende Formel, die 1995 von Peter Borwein (dem Bruder Ihres ersten Autors), David Bailey und Simon Plouffe entdeckt wurde und nach ihnen als BBP-Formel bezeichnet wird [Bailey et al. 97]:

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left( \frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right). \quad (2.1)$$

(Es liegt im Wesen eines guten Teils der heutigen Experimentellen Mathematik, dass sie Formeln benötigt, die ziemlich viel Platz beanspruchen. Die obige Formel liefert hierfür nur einen ersten Vorgeschmack. Allerdings kommen in den meisten unserer Beispiele, so wie auch hier, nur sehr einfache mathematische Objekte vor.)

Mit Formel (2.1) können Sie direkt die binären oder hexadezimalen Ziffern von  $\pi$  beginnend mit der  $n$ -ten Stelle berechnen, ohne dass Sie dazu die vorhergehenden  $n - 1$  Stellen benötigen. Alles, was Sie für diese Berechnung benötigen, ist ein einfacher Algorithmus, der auf normaler 64- oder 128-Bit-Arithmetik läuft. Wir werden noch darstellen, wie diese Rechnung ausgeführt wird, aber hauptsächlich interessieren wir uns dafür, wie die BBP-Formel entdeckt wurde.

Diese Geschichte begann mit der bekannten klassischen Formel

$$\log 2 = \sum_{k=1}^{\infty} \frac{1}{k2^k}. \quad (2.2)$$

Um 1994 erkannten Peter Borwein und Simon Plouffe, damals beide an der Simon Fraser University in Kanada, dass man diese Formel nutzen kann, um einzelne Binärziffern von  $\log 2$  zu berechnen. Angenommen, Sie wollen ein paar der Binärziffern ab der Position  $d + 1$  angeben. Das ist äquivalent dazu, den Wert  $\{2^d \log 2\}$  zu berechnen, wobei  $\{\dots\}$  den gebrochenen Anteil bezeichnet<sup>2</sup>. Aus (2.2) ergibt sich:

$$\begin{aligned} \{2^d \log 2\} &= \left\{ \left\{ \sum_{k=1}^d \frac{2^{d-k}}{k} \right\} + \left\{ \sum_{k=d+1}^{\infty} \frac{2^{d-k}}{k} \right\} \right\} \\ &= \left\{ \left\{ \sum_{k=1}^d \frac{2^{d-k} \bmod k}{k} \right\} + \left\{ \sum_{k=d+1}^{\infty} \frac{2^{d-k}}{k} \right\} \right\}. \end{aligned} \quad (2.3)$$

(Gehen Sie diese Formel langsam Schritt für Schritt durch. Hier passiert nichts Tiefsinniges, nur die Schreibweise sieht etwas kompliziert aus. Wenn wir den gebrochenen Anteil einer Summe berechnen, können wir die ganz-

<sup>2</sup> Mit der am Ende von Kapitel 1 eingeführten Schreibweise gilt also:  $\{x\} = x - \lfloor x \rfloor$ .

zahligen Anteile einzelner Summanden jederzeit vernachlässigen, und das „mod  $k$ “ im Zähler des ersten Terms können wir einfügen, weil wir nur am gebrochenen Anteil des Quotienten nach der Division durch  $k$  interessiert sind.)

Nun lassen Sie uns sehen, wie wir die Formel (2.3) benutzen können, um Binärziffern von  $\log 2$  ab der Position  $d + 1$  zu berechnen. Zunächst gibt es eine höchst effiziente Methode, um die Zähler  $2^{d-k} \bmod k$  in der ersten Summe zu berechnen. Die naive Vorgehensweise wäre, die 2 einfach  $(d - k)$ -mal mit sich selbst zu multiplizieren und jedes dabei entstehende ganzzahlige Vielfache von  $k$  gleich wieder abzuziehen. Dabei würde man also niemals eine Zahl speichern müssen, die größer als  $k$  wäre, doch wenn  $d - k$  sehr groß ist (wie es der Fall wäre, wenn  $d$  eine Billiarde ist), würde es eine sehr große Anzahl an Schritten erfordern – tatsächlich eine zu große Anzahl. Doch wenn Sie die Potenz  $2^{d-k}$  durch wiederholtes Quadrieren berechnen, können Sie die Rechnung gewaltig reduzieren. Zum Beispiel ist

$$2^{50} = (((((2^2)^2)^2)^2)^2)^2 \times (((2^2)^2)^2)^2 \times 2^2,$$

was nur fünf statt fünfzig Verdopplungen (sowie zwei Multiplikationen) erfordert. Wenn Sie die Rechnung bezüglich eines relativ kleinen Moduls, etwa mod 10, durchführen und alle dabei auftauchenden Vielfachen von 10 sofort weglassen, dann könnten Sie sie sogar im Kopf anstellen. Nach ein wenig unkomplizierter Programmierarbeit erhalten Sie schnell ein Computerprogramm, das derartige Berechnungen mit großer Effizienz durchführt.<sup>3</sup>

Damit haben Sie eine effiziente Methode, um die erste Summe in (2.3) zu berechnen. Da nur wenige Binärziffern ab der Position  $d + 1$  berechnet werden sollen, kann die zweite (unendliche) Summe nach einigen Termen abgebrochen werden. (Beachten Sie, dass die einzelnen Terme in der zweiten Summe schnell klein werden.) Und insgesamt erhalten Sie damit die gesuchten Binärziffern.

Wenn  $d < 10^7$  ist, kann die gesamte Rechnung mit der üblichen 64-Bit-Arithmetik ausgeführt werden, die bei den meisten Computern Standard ist. Mit einer 128-Bit-Gleitkommaarithmetik können Sie bequem  $d \leq 10^{15}$  handhaben. Für  $d$  jenseits von  $10^{15}$  benötigen Sie spezielle arithmetische Routinen, aber damit würden Sie die milliardste Stelle überschreiten, die ja unser Ausgangspunkt war.

Nun mag das zwar eine nette Beobachtung gewesen sein, aber bis jetzt nichts auch nur entfernt Experimentelles. Doch sobald sie diese Entdeckung für  $\log 2$  gemacht hatten, fragten sich Borwein und Plouffe, ob man nicht einen ähnlichen Zaubertrick auch für  $\pi$  ausführen könnte. Das mag viel-

<sup>3</sup> Derartige Klammergymnastik ist typisch für moderne Verfahren. Oft führt cleveres Umsortieren einer Berechnung zu einer gewaltigen Arbeitersparnis.

leicht für die Menschheit genauso nutzlos sein, aber wenn man den Status von  $\pi$  in der Mathematik bedenkt und die Faszination, die schon die alten Griechen für die Berechnung von  $\pi$  verspürten, dann wäre das auf jeden Fall interessant!<sup>4</sup>

Offensichtlich kann dieselbe Methode auf jede Konstante  $\alpha$  angewendet werden, die mittels einer Formel der Bauart

$$\alpha = \sum_{k=0}^{\infty} \frac{p(k)}{b^k q(k)}$$

dargestellt werden kann (mit einer natürlichen Zahl  $b > 1$  und Polynomen  $p$  und  $q$  mit ganzzahligen Koeffizienten, wobei  $q$  keine Nullstellen auf den natürlichen Zahlen hat). Auch wenn verschiedene Reihenentwicklungen von  $\pi$  bekannt sind, förderte eine Durchsicht der verfügbaren Literatur nichts von der obigen Form zutage. Also taten sich Borwein und Plouffe mit David Bailey zusammen, einem Mathematiker mit Expertise im wissenschaftlichen Rechnen, der damals am NASA Ames Research Center in Kalifornien arbeitete,<sup>5</sup> um festzustellen, ob  $\pi$  sich als Linearkombination von anderen Konstanten der gesuchten Form schreiben lässt.

Bailey hatte ein Computerprogramm entwickelt, das solche Linearkombinationen finden kann und auf dem *PSLQ-Algorithmus* basiert, der von dem amerikanischen Mathematiker und Bildhauer Helaman Ferguson entwickelt worden war. Der Name „PSLQ“ stammt von der Vorgehensweise des Algorithmus, der im Laufe der Rechnung einen Vektor von Partialsummen gewisser Quadrate und eine orthogonale Dreiecksmatrix benutzt (auf Englisch: *partial sum of squares* und *lower triangular orthogonal matrix* oder *LQ matrix*).

---

<sup>4</sup> Natürlich hängt es davon ab, was genau man unter „nützlich“ versteht, wenn ein bestimmtes mathematisches Resultat als „nutzlos“ erklärt wird – und selbst dann ist das ein Werturteil, das sich im späteren Rückblick als falsch erweisen kann. Wenn es einer großen Anzahl an Leuten Vergnügen bereitet oder sie dazu anregt, über das Resultat nachzudenken, dann kann man es sicher als „nützlich“ bezeichnen – und in diesem Sinne ist das Borwein-Plouffe-Resultat genauso „nützlich“ wie Literatur oder Kunst. In der Mathematik gibt es oft eine versteckte Nützlichkeit, indem sich von Methoden, die entwickelt wurden, um ein „nutzloses“ Resultat zu erhalten, später herausstellt, dass sie andere Anwendungen in wesentlich handfesteren Situationen haben. Und tatsächlich hat sich dieser Algorithmus schon dadurch als nützlich erwiesen, dass er wegen seines geringen Speicherplatzbedarfes in zumindest einen Fortran-Compiler integriert worden ist. Auch in Bellards und anderen  $\pi$ -Rekorden ist er benutzt worden, um das Ergebnis durch direkte Berechnung einiger Hexadezimalziffern um die höchsten berechneten Stellen herum zu verifizieren. Das dauert dann nur Stunden anstatt der Monate, die sonst nötig wären, um sämtliche Stellen nochmals neu zu berechnen. (Details können Sie in dem Buch *Mathematics by Experiment* [Borwein und Bailey 08] finden.) Wie viele andere Mathematiker auch beschäftigen sich Ihre beiden jetzigen Autoren mit der Mathematik nicht wegen ihrer „Nützlichkeit“, aber uns ist klar, dass die Frage der Nützlichkeit für viele von Interesse ist.

<sup>5</sup> Er ist jetzt am Lawrence Berkeley Laboratory in Kalifornien tätig.

Der PSLQ-Algorithmus ist eine Methode zum Auffinden von ganzzahligen linearen Abhängigkeiten (kurz: von Integerrelationen<sup>6</sup>). Ohne in die Details gehen zu wollen, tun solche Algorithmen Folgendes: Zunächst sei festgestellt, dass die Aufgabe, eine reelle Konstante  $a$  als rationale Linearkombination

$$a = q_1 a_1 + q_2 a_2 + \dots + q_n a_n$$

aus gewissen anderen reellen Konstanten  $a_1, a_2, \dots, a_n$  mit rationalen Koeffizienten  $q_1, \dots, q_n$  zu schreiben, äquivalent ist zu der Aufgabe, ganze Zahlen  $\lambda_0, \lambda_1, \dots, \lambda_n$  zu finden mit  $\lambda_0 \neq 0$  und

$$\lambda_0 a + \lambda_1 a_1 + \dots + \lambda_n a_n = 0.$$

Falls beliebige reelle Zahlen  $a_0, a_1, \dots, a_n$  und eine Detektionsschwelle  $\varepsilon > 0$  vorgegeben sind, dann findet ein Integerrelationsalgorithmus entweder Koeffizienten  $\lambda_0, \lambda_1, \dots, \lambda_n$  mit

$$|\lambda_0 a_0 + \lambda_1 a_1 + \dots + \lambda_n a_n| < \varepsilon$$

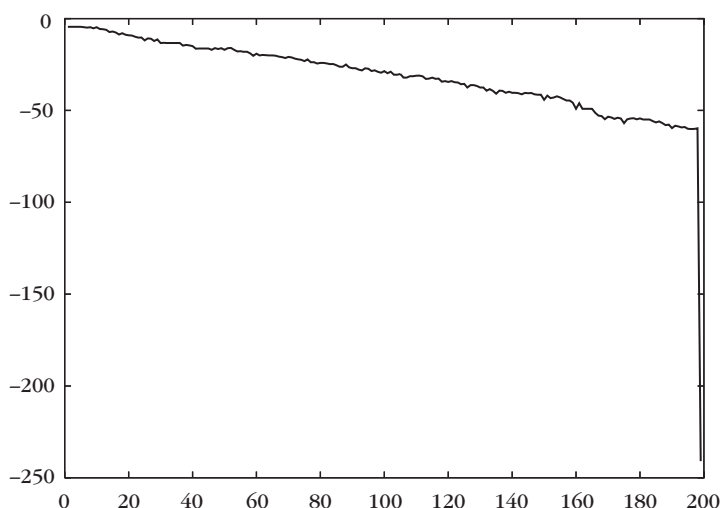
oder teilt uns mit, dass innerhalb einer Kugel um den Ursprung mit einem Radius, den das Programm ebenfalls ausgibt, kein solcher Ausdruck existiert.

Natürlich wird man nur selten genau eine 0 für den Wert der Summe erhalten, und selbst wenn, kann man nicht sicher sein, ob das nicht nur ein Artefakt der jeweils verwendeten Computerarithmetik ist, die nur mit einer festen Anzahl an Binärstellen rechnet. Doch wenn Sie die Detektionsschwelle  $\varepsilon$  hinreichend klein wählen, können Sie so viel Vertrauen in das Ergebnis erhalten, wie Sie nur wollen. Für die Zwecke der Experimentellen Mathematik ist das normalerweise ausreichend, um feststellen zu können: „Das Ergebnis ist experimentell gleich 0.“

Wird dadurch tatsächlich bewiesen, dass die Linearkombination gleich 0 ist (oder dass es auch nur eine Linearkombination gleich 0 *gibt*)? Natürlich nicht. Und doch werden viele (die meisten?) Mathematiker, wenn man ihnen zwei geschlossene Ausdrücke zeigt, die auf beispielsweise 100 (oder 500 oder 1000 usw.) Stellen dieselbe Dezimalentwicklung haben, mit einigem Vertrauen die Schlussfolgerung ziehen, dass beide Ausdrücke gleich sind. Dieses Vertrauen wird in den meisten Fällen zweifellos so groß sein, dass der Mathematiker oder die Mathematikerin eine beträchtliche Menge an Zeit und Mühe auf den Versuch verwenden wird, einen strengen Beweis für die Gleichheit zu finden.<sup>7</sup>

<sup>6</sup> „Integerrelation“ ist eigentlich kein deutsches Wort, sondern eine Verballhornung des englischen *integer relation*, also „Beziehung mit ganzen Zahlen“. „Integerrelation“ ist aber gerade in zusammengesetzten Begriffen einfacher zu verwenden als „ganzzahlige lineare Abhängigkeit“.

<sup>7</sup> Jedoch werden Sie in Kapitel 10 einige Beispiele finden, in denen die Intuition in die Irre führt.



**Abb. 2.1** Verlauf von  $\log_{10}(\min_i |(A_k^{-1}x)_i|)$  als Funktion der Anzahl der Iterationen  $k$  in einem typischen Lauf von PSLQ, wobei  $x$  der Eingabevektor ist.

Das Folgende sollte ein Anwender über den PSLQ-Algorithmus wissen (wieder ohne alle Details).

Bezeichne den Eingabevektor  $(a_1, \dots, a_n)$  als  $x$ . Die Idee ist, eine Folge von Matrizen  $A_k$  zu konstruieren, so dass die Komponenten des Vektors  $y_k = A_k^{-1}x$  von Schritt zu Schritt immer kleiner werden. Dabei wird außerdem darauf geachtet, dass in jedem Schritt die größte und die kleinste Komponente von  $y_k$  sich um höchstens zwei oder drei Größenordnungen unterscheiden.

Wenn der Algorithmus eine Relation entdeckt, dann fällt die kleinste Komponente von  $y_k$  plötzlich bis zur Größenordnung der verwendeten Rechengenauigkeit ab (also auf ungefähr  $10^{-p}$ , falls die Rechnung mit  $p$  Stellen Genauigkeit durchgeführt wird). Die gesuchte Integerrelation wird dann durch die entsprechende Spalte der Matrix  $A_k^{-1}$  gegeben. Abbildung 2.1 zeigt dieses Verhalten für einen typischen Durchlauf von PSLQ.

Um mit einiger Zuversicht sagen zu können, dass das, was der Algorithmus zurückgibt, eine tatsächliche Integerrelation ist (absolute Sicherheit ist natürlich nicht möglich), könnte man die Detektionsschwelle  $\varepsilon$  beispielsweise auf  $10^{-100}$  setzen. Die in der Rechnung verwendete Rechengenauigkeit muss dann noch um ein paar Größenordnungen kleiner sein, um sicherzustellen, dass die unvermeidlichen Rundungsfehler das Resultat nicht wesentlich beeinflussen. (Für einen Integerrelationsalgorithmus muss man fast immer eine hochgenaue Arithmetik einsetzen.)

Um nun auf Peter Borweins und Simon Plouffes Suche nach einer Formel zurückzukommen, mit der sie beliebige Binärziffern von  $\pi$  berechnen könn-

ten, erinnern wir daran, dass, wie sie bereits wussten, ihre log 2-Methode für jede Konstante funktionieren würde, die sich schreiben ließ als

$$\alpha = \sum_{k=0}^{\infty} \frac{p(k)}{q(k)2^k}.$$

Hier bezeichnen, wie zuvor,  $p$  und  $q$  ganzzahlige Polynome mit  $\deg p < \deg q$  und  $q$  ohne Nullstellen auf den natürlichen Zahlen. Eine erste Durchsicht der Literatur ergab ungefähr 25 Konstanten dieses Typs. Doch  $\pi$  gehörte nicht dazu. Borwein und Plouffe war jedoch klar, dass sie beliebige Binärziffern für jede Zahl berechnen konnten, die als Linearkombination solcher Konstanten ausgedrückt werden kann. Konnten sie eine derartige Linearkombination für  $\pi$  finden?

Als wir das letzte Mal bei den beiden Forschern waren, hatten sie sich gerade an David Bailey und seine Implementation des PSLQ-Algorithmus in hochgenauer Gleitkommaarithmetik gewandt. Konnte Baileys Programm eine Integerrelation für den Vektor  $(a_1, a_2, \dots, a_n)$  finden, wobei  $a_1 = \pi$  ist und  $a_2, \dots, a_n$  die aus der Literatur entnommenen Konstanten von der benötigten Bauart sind, alle ausgewertet mit einer Genauigkeit von einigen hundert Stellen?

Zunächst konnte es das nicht. Doch was, wenn sie noch einige andere Konstanten des benötigten Typs finden könnten?

Und so begann die Suche. Es war nicht direkt ein Umhertappen im Dunkeln, aber doch fast. Und die Suche dauerte eine ganze Weile, wobei jedes Mal, wenn eine neue Konstante in der Literatur gefunden wurde, der Algorithmus neu gestartet wurde.

Doch schließlich, nachdem etwa zwei Monate lang gerechnet worden war, stieß Baileys Programm auf die Goldader. Es fand die Formel

$$\pi = 4 \cdot {}_2F_1 \left( \begin{matrix} \frac{1}{4}, 1 \\ \frac{5}{4} \end{matrix} \middle| -\frac{1}{4} \right) + 2 \arctan \left( \frac{1}{2} \right) - \log 5,$$

wobei der erste Term auf der rechten Seite nach dem Faktor 4 der Wert einer hypergeometrischen Funktion und ungefähr gleich  $0,955933837 \dots$  ist (siehe auch Gleichung (4.2) weiter unten). Als die Forscher diesen Ausdruck in Summenschreibweise gebracht hatten, war die seitdem berühmt gewordene BBP-Formel gefunden. Die Suche war beendet.

Wir sollten darauf hinweisen, dass das Resultat, mit dem wir dieses Kapitel motivierten, also die Berechnung der milliardsten Binärstelle von  $\pi$ , selbst mit der BBP-Formel eine eindrucksvolle Leistung ist. Diese spezielle Berechnung wurde im September 2000 durchgeführt, organisiert von Colin Percival, der damals ein Student im Grundstudium an der Simon Fraser University in Kanada war. Die Rechnung verschlang 250 CPU-Jahre, verteilt auf 1734 Rechner in 56 Ländern. Die milliardste Hexadezimalstelle von  $\pi$  zu berechnen, wird noch wesentlich mühsamer sein, da sie den Binärziffern um die Stelle  $4 \times 10^{15}$  entspricht. Doch Fortschritte bei der Rechenleistung seit



dem Projekt von Percival sollten diese Berechnung mit ziemlicher Sicherheit inzwischen möglich gemacht haben.

Wir beenden dieses Kapitel noch mit einer interessanten Beobachtung von Bailey und Richard Crandall. Betrachten Sie die Iteration

$$x_0 = 0;$$

$$x_n = \left\{ 16x_{n-1} + \frac{120n^2 - 89n + 16}{512n^4 - 1024n^3 + 712n^2 - 206n + 21} \right\}.$$

Der Bruch in dieser Formel stammt aus der BBP-Formel (2.1), wenn Sie einfach die vier Brüche in (2.1) auf den Hauptnenner bringen und dann den Index um 1 verschieben.

Nun definiere  $y_n$  durch

$$y_n = \lfloor 16x_n \rfloor.$$

Bailey und Crandall wurden durch eine Analyse der statistischen Verteilung der Hexadezimalziffern von  $\pi$  darauf gebracht, sich diese Zahlen  $y_n$  genauer anzusehen. Denn wenn Sie das Einheitsintervall in sechzehn gleich große Teilintervalle aufteilen und diese mit  $0, 1, \dots, 15$  durchnummerieren, dann ist  $y_n$  die Nummer des Teilintervalls, in dem  $x_n$  liegt.

Jetzt kommt das Überraschende: Die Folge  $(y_n)$  wurde berechnet, und es stellte sich heraus, dass die erste Million ihrer Werte *genau* mit der ersten Million der Hexadezimalziffern von  $\pi - 3$  übereinstimmt. (Das ist eine ziemlich aufwändige Rechnung, die ungefähr  $n^2$  Rechenschritte benötigt und nicht besonders gut parallelisierbar ist.) Dies veranlasste Bailey and Crandall zu der Vermutung, dass die Folge  $(y_n)$  genau mit den hexadezimalen Ziffern von  $\pi - 3$  übereinstimmt.

Bis jetzt ist diese Vermutung noch nicht endgültig bewiesen worden. Aber bewiesen oder nicht, ist sie denn wahr? Oder ist die Gleichheit der ersten Million Werte ein irreführender Zufall, ein Beispiel für die stets präsente Gefahr in der Experimentellen Mathematik – zwar selten, aber durchaus real –, dass solche Regelmäßigkeiten manchmal in Hunderten, Millionen, Milliarden (und mitunter noch *weitaus* mehr) Fällen auftreten, bevor sie dann doch noch versagen? Wie wir schon angedeutet haben, werden wir auf dieses beunruhigende, aber auch faszinierende Phänomen der irreführenden Evidenz in Kapitel 10 zurückkommen.

## Untersuchungen

1. *BBP-Formeln.* Diese gibt es für eine große Anzahl an Konstanten in verschiedenen Zahlenbasen. Zum Beispiel können Sie BBP-Formeln finden für

- (a) den Logarithmus jeder Primzahl kleiner als 23,
- (b)  $\pi^2$  zur Basis 2 und zur Basis 3,

(c) die Catalansche Konstante

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

zur Basis 2.

Was  $\pi$  zur Basis 10 betrifft, so ist bereits schlüssig gezeigt worden, dass es eine derartige Formel nicht gibt.

2. Für  $e$  ist zu keiner Basis eine BBP-Formel bekannt. Man vermutet, dass es keine gibt. Können Sie eine „natürliche“ langsame Reihe für  $e$  finden? (Eine Reihe muss nicht schnell konvergieren, um nützlich zu sein. Denn wie wir gesehen haben, ist es die Existenz geeigneter langsam konvergenter Reihen, die dazu führt, dass einzelne Ziffern herausgepickt werden können.) Wie Richter Potter Stewart werden Sie eine „natürliche“ Reihe erkennen, wenn Sie sie sehen.
3. *Tröpfel-Algorithmen für  $\pi$  und  $e$ .* Ein *Tröpfel-Algorithmus* für eine Konstante ist ein Algorithmus, der deren Ziffern eine nach der anderen produziert („Tröpfchen für Tröpfchen“).

Das ist besonders einfach für  $e$ , weil Überträge kein großes Problem sind. Der folgende Algorithmus, der von Stanley Rabinowitz und Stan Wagon stammt, erzeugt die Ziffern von  $e$ :

Setze zur Initialisierung alle Einträge des Vektors  $A$  der Länge  $n+1$  zu 1. Dann wiederhole die folgenden Schritte  $(n-1)$ -mal:

- (a) Multipliziere jeden Eintrag in  $A$  mit 10.
- (b) Reduziere rechts beginnend den  $i$ -ten Eintrag von  $A$  modulo  $i+1$  und addiere den Quotienten der Division zum nächsten Eintrag links. Der letzte auf diese Weise produzierte Quotient ist die nächste Ziffer von  $e$ .

Dieser Algorithmus basiert auf der folgenden Formel, die eine einfache Umformulierung der schnell konvergierenden Reihe  $e = \sum_{n=0}^{\infty} 1/n!$  ist:

$$e = 1 + \frac{1}{1} \left( 1 + \frac{1}{2} \left( 1 + \frac{1}{3} \left( 1 + \frac{1}{4} \left( 1 + \frac{1}{5} (1 + \dots) \right) \right) \right) \right).$$

Implementieren Sie nun einen entsprechenden Tröpfel-Algorithmus für  $\pi$ , indem Sie die folgende Formel zeigen und dann in ähnlicher Weise umsetzen:

$$\pi = 2 + \frac{1}{3} \left( 2 + \frac{2}{5} \left( 2 + \frac{3}{7} \left( 2 + \dots \left( 2 + \frac{k}{2k+1} (2 + \dots) \right) \right) \right) \right).$$

Der letzte Term kann durch  $2 + 4k/(2k+1)$  mit  $k = n \log_2 10$  approximiert werden, um  $n$  Ziffern von  $\pi$  „Tröpfchen für Tröpfchen“ zu erhalten.

Wenn Sie den Algorithmus laufen lassen wollen, ohne die Anzahl der Ziffern zuvor festzulegen, müssen Sie noch etwas mehr Sorgfalt aufwenden.

Wir können die mutmaßliche hexadezimale Iteration für  $\pi$ , mit der wir das Kapitel beendeten, als spektakulären, wenn auch unbewiesenen Tröpfel-Algorithmus für  $\pi$  zur Basis 16 ansehen. Wenn Sie heuristisch voraussetzen, dass die hexadezimalen Reste von  $\pi$ , also  $\{16^n \pi\}$ , sich wie unabhängige, gleichverteilte Zufallsgrößen in  $[0, 1]$  verhalten, dann kann gezeigt werden, dass die Wahrscheinlichkeit, dass noch eine Abweichung auftritt, kleiner als 1 zu  $10^8$  ist.



<http://www.springer.com/978-3-8274-2661-1>

Experimentelle Mathematik

Eine beispielorientierte Einführung

Borwein, J.; Devlin, K.

2011, XII, 158 S. 40 Abb., Softcover

ISBN: 978-3-8274-2661-1