Der Zapfhahnalgorithmus nach Rabinowitz & Wagon

Seminar Computeralgebra: Pi
von zur Gathen & Nüsken

Florian Schoppmann

Fakultät für Elektrotechnik, Mathematik und Informatik
Universität Paderborn

13. Dezember 2004 – Wintersemester 2004/05

Motivation

Ludolph Van Ceulen

- *28.1.1540, Hildesheim, †31.12.1610 Leiden (Niederlande)
- Professor für Arithmetik, Vermessungskunde und Festungsbau an der Ingenieurs-Schule Leiden (ab 1600)
- Verbrachte einen Großteil seines Lebens damit, 35 Dezimalstellen von π mit geometrischer Methode zu berechnen
- In Deutschland bis ca. Erster Weltkrieg: „Ludolph'sche Zahl“

Hätte Van Ceulen einfacher rechnen und Jahre seines Lebens sparen können? Lässt sich ein einfacher Algorithmus formulieren?

Angaben übernommen von:

`http://www-history.mcs.st-andrews.ac.uk/Mathematicians/Van_Ceulen.html`

Ein Zapfhahnalgorithmus für π

- Veröffentlicht von Rabinowitz und Wagon 1995
- Basiert auf Reihendarstellung

$$\pi = \sum_{i=0}^{\infty} \frac{(i!)^2 2^{i+1}}{(2i+1)!} = \sum_{i=0}^{\infty} \frac{2 \cdot i!}{(2i+1)!!}$$

- Einfach implementierbar:
  - Verwendet nur Ganzzahl-Arithmetik
  - Keine externen Bibliotheken

    $\Rightarrow$ extrem kurze Implementierungen möglich:

C-Programm für 15000 Stellen von π

```
a[52514],b,c=52514,d,e,f=1e4,g,h;
main(){for(;b=c-=14;h=printf("%04d",e+d/f))
for(e=d%=f;g=--b*2;d/=g)d=d*b+f*(h?a[b]:f/5),a[b]=d%--g;}
```

Reihendarstellung von π – ein technisches Lemma

Lemma

Seien $i, j \in \mathbb{N}_0$. Dann gilt:

$$\int_0^1 x^i (1-x)^j \, dx = \frac{i! \cdot j!}{(i+j+1)!}$$

Beweis.

$$\begin{aligned}\int_0^1 x^i (1-x)^j \, dx &= \frac{1}{i+1} x^{i+1} (1-x)^j \Big|_0^1 - \int_0^1 \frac{-j}{i+1} x^{i+1} (1-x)^{j-1} \, dx \\ &= \frac{j}{i+1} \int_0^1 x^{i+1} (1-x)^{j-1} \, dx\end{aligned}$$

Induktiv und mit $\int_0^1 x^{i+j} \, dx = \frac{1}{i+j+1}$ folgt daraus das Lemma. $\square$

Reihendarstellung von π – ein technisches Lemma

Lemma

Seien $i, j \in \mathbb{N}_0$. Dann gilt:

$$\int_0^1 x^i (1-x)^j dx = \frac{i! \cdot j!}{(i+j+1)!}$$

Beweis.

$$\begin{aligned}\int_0^1 x^i (1-x)^j dx &= \frac{1}{i+1} x^{i+1} (1-x)^j \Big|_0^1 - \int_0^1 \frac{-j}{i+1} x^{i+1} (1-x)^{j-1} dx \\ &= \frac{j}{i+1} \int_0^1 x^{i+1} (1-x)^{j-1} dx\end{aligned}$$

Induktiv und mit $\int_0^1 x^{i+j} dx = \frac{1}{i+j+1}$ folgt daraus das Lemma. $\square$

Reihendarstellung von π

Satz

Die eingangs angegebene Reihendarstellung von π ist korrekt:

$$\pi = \sum_{i=0}^{\infty} \frac{2 \cdot i!}{(2i+1)!!}$$

Beweis.

$$\sum_{i=0}^{\infty} \frac{(i!)^2 2^{i+1}}{(2i+1)!} = \sum_{i=0}^{\infty} 2^{i+1} \int_0^1 x^i (1-x)^i dx \quad \text{(nach Lemma)}$$

$$= 2 \int_0^1 \sum_{i=0}^{\infty} (2x)^i (1-x)^i dx = 2 \int_0^1 \frac{1}{1-2x(1-x)} dx$$

$$= 2 \arctan 2x - 1 \Big|_0^1 = \pi \qquad \square$$

Reihendarstellung von π

Satz

Die eingangs angegebene Reihendarstellung von π ist korrekt:

$$\pi = \sum_{i=0}^{\infty} \frac{2 \cdot i!}{(2i+1)!!}$$

Beweis.

$$\sum_{i=0}^{\infty} \frac{(i!)^2 2^{i+1}}{(2i+1)!} = \sum_{i=0}^{\infty} 2^{i+1} \int_0^1 x^i (1-x)^i dx \quad \text{(nach Lemma)}$$

$$= 2 \int_0^1 \sum_{i=0}^{\infty} (2x)^i (1-x)^i dx = 2 \int_0^1 \frac{1}{1-2x(1-x)} dx$$

$$= 2 \arctan 2x - 1 \Big|_0^1 = \pi \qquad \square$$

Darstellungen mit variabler Basis

Die letzte Formel auseinander gezogen:

$$\pi = \sum_{i=0}^{\infty} \frac{2 \cdot i!}{(2i+1)!!} = 2 + \frac{1}{3}\Bigl(2 + \frac{2}{5}\Bigl(2 + \frac{3}{7}\Bigl(2 + \ldots\Bigr)\Bigr)\Bigr)$$

Ist eine „einfache“ Umrechnung in Dezimaldarstellung möglich?
– Ja, aber es ist noch etwas Hintergrundwissen nötig...

Definition

1. *Darstellung bezüglich einer variablen Basis* $b := (b_n)_{n\in\mathbb{N}}$, (b_n) Folge positiver rationaler Zahlen
2. Kurzschreibweise $(a_0; a_1, a_2, \ldots)_b$
3. *fixe Basis*

Im Folgenden: Darstellungen zur Basis $c := (\frac{1}{3}, \frac{2}{5}, \frac{3}{7}, \ldots)$

Darstellungen mit variabler Basis

Die letzte Formel auseinander gezogen:

$$\pi = \sum_{i=0}^{\infty} \frac{2 \cdot i!}{(2i+1)!!} = 2 + \frac{1}{3}\Big(2 + \frac{2}{5}\Big(2 + \frac{3}{7}\Big(2 + \dots\Big)\Big)\Big)$$

Ist eine „einfache“ Umrechnung in Dezimaldarstellung möglich?
– Ja, aber es ist noch etwas Hintergrundwissen nötig...

Definition

1. *Darstellung bezüglich einer variablen Basis* $b := (b_n)_{n \in \mathbb{N}}$, (b_n) Folge positiver rationaler Zahlen
2. Kurzschreibweise $(a_0; a_1, a_2, \dots)_b$
3. *fixe Basis*

Im Folgenden: Darstellungen zur Basis $c := (\frac{1}{3}, \frac{2}{5}, \frac{3}{7}, \dots)$

Reguläre Darstellungen

Definition

Eine Darstellung $(a_0; a_1, a_2, a_3, \dots)_c$ heißt:

1. *regulär*, wenn $a_0 \in \mathbb{N}_0$ und für $i = 1, 2, 3, \dots$ gilt: $0 \leq a_i \leq 2i$.
   Dabei: a_i *maximale Ziffer* $:\Leftrightarrow a_i = 2i$
2. *abbrechend*, wenn $\exists N \in \mathbb{N} : \forall i \geq N : a_i = 0$

Bezeichnung *normalisiert* bewusst nicht verwendet, denn Darstellungen bezüglich der Basis c sind *nicht* eindeutig. Beispiel:

$$(0; 2, 0, 0, 0, \dots)_c = \frac{2}{3} = (0; 0, 2, 3, 4, \dots)_c$$

Ganzzahlteil einer regulären Darstellung

Satz

$(0; 2, 4, 6, \ldots)_c = 2$. *Demnach liegen reguläre Darstellungen der Form* $(0; a_1, a_2, a_3, \ldots)_c$ *im Interval* $[0, 2]$.

Beweis. Setze

$$S_n := \sum_{i=0}^{n} \frac{(2i)i!}{(2i+1)!!}$$

Zeige: $\lim_{n\to\infty} S_n = 2$. Induktion:

$$2 - S_n = \frac{2^{n+1}}{\binom{2n+1}{n}} = \frac{2(n+1)!}{(2n+1)!!} \tag{1}$$

Da der Ausdruck (1) offensichtlich für $n \to \infty$ gegen 0 strebt, folgt die Behauptung. □

Ganzzahlteil einer regulären Darstellung

Satz

$(0; 2, 4, 6, \ldots)_c = 2$. *Demnach liegen reguläre Darstellungen der Form* $(0; a_1, a_2, a_3, \ldots)_c$ *im Interval* $[0, 2]$.

Beweis. Setze

$$S_n := \sum_{i=0}^{n} \frac{(2i)i!}{(2i+1)!!}$$

Zeige: $\lim_{n\to\infty} S_n = 2$. Induktion:

$$2 - S_n = \frac{2^{n+1}}{\binom{2n+1}{n}} = \frac{2(n+1)!}{(2n+1)!!} \tag{1}$$

Da der Ausdruck (1) offensichtlich für $n \to \infty$ gegen 0 strebt, folgt die Behauptung. $\square$

Fehlerabschätzung der Partialreihe

Satz

Sei $\pi_m := (\underbrace{2; 2, 2, 2, \ldots, 2}_{m \text{ Ziffern}})_c$ *für ein* $m \in \mathbb{N}$. *Dann gilt:*

$$0 < \pi - \pi_m < \frac{8}{3} \cdot 10^{-3m/10} \tag{2}$$

Beweis.

$$\pi - \pi_m = \sum_{i=m}^{\infty} \frac{(i!)^2 2^{i+1}}{(2i+1)!} < \frac{(m!)^2 2^{m+2}}{(2m+1)!} \leq \frac{8}{3} \cdot 2^{-m} < \frac{8}{3} \cdot 10^{-3m/10} \quad \square$$

Folglich: Mit $m := \lceil \frac{10n}{3} \rceil$ gilt $\pi - \pi_m < \frac{8}{3} \cdot 10^{-n}$, denn

$$\pi - \pi_m \overset{(2)}{<} \frac{8}{3} \cdot 10^{-3\lceil \frac{10n}{3} \rceil / 10} \leq \frac{8}{3} \cdot 10^{-3\frac{10n}{3}/10} = \frac{8}{3} \cdot 10^{-n}$$

Fehlerabschätzung der Partialreihe

Satz

Sei $\pi_m := (\underbrace{2; 2, 2, 2, \ldots, 2}_{m\ \textit{Ziffern}})_c$ *für ein* $m \in \mathbb{N}$. *Dann gilt:*

$$0 < \pi - \pi_m < \frac{8}{3} \cdot 10^{-3m/10} \tag{2}$$

Beweis.

$$\pi - \pi_m = \sum_{i=m}^{\infty} \frac{(i!)^2 2^{i+1}}{(2i+1)!} < \frac{(m!)^2 2^{m+2}}{(2m+1)!} \leq \frac{8}{3} \cdot 2^{-m} < \frac{8}{3} \cdot 10^{-3m/10} \quad \square$$

Folglich: Mit $m := \lceil \frac{10n}{3} \rceil$ gilt $\pi - \pi_m < \frac{8}{3} \cdot 10^{-n}$, denn

$$\pi - \pi_m \overset{(2)}{<} \frac{8}{3} \cdot 10^{-3\lceil \frac{10n}{3} \rceil/10} \leq \frac{8}{3} \cdot 10^{-3\frac{10n}{3}/10} = \frac{8}{3} \cdot 10^{-n}$$

Ein hypothetischer Algorithmus zur Darstellungsumrechnung

Wenn jede reguläre abbrechende Darstellung der Form $(0; a_1, a_2, a_3, \ldots)_c$ stets < 1 *wäre*:

Umrechnung in die Dezimaldarstellung

Eingabe: $(a_0; a_1, a_2, a_3, \ldots, a_n, 0, \ldots)_c$.
Ausgabe: Gewünschte Stellen der Dezimaldarstellung.

1: **Wiederhole**
2: Gebe den Ganzzahlteil aus (also a_0). Setze: $a_0 \leftarrow 0$.
3: Multipliziere mit 10 wie folgt: Multipliziere alle a_i mit 10 und regularisiere anschließend von recht nach links (Reduktion jeder Ziffer, Weiterreichung von Überträgen nach links).
4: Speichere regularisierte Darstellung wieder in $(a_0; a_1, a_2, a_3, \ldots)_c$.
5: **Bis** gewünschte Genauigkeit erreicht

Umrechnung in Dezimaldarstellung

Problem: Zu frühe Ausgabe des Ganzzahlteils
Mögliche Abhilfe:

- Nach Multiplikation mit 10 und Regularisierung: Ausgabe von Ziffern nur im Falle $(a_0 \mod 10) < 9$

  $\Rightarrow$ Im weiteren Schleifen-Durchäufen nur Änderung derzeitiger Einer-Ziffer des Ganzzahlteils möglich
- Ausgabe von $\lfloor a_0/10 \rfloor$, Anschließend Reduktion von a_0 modulo 10

Bei vielen aufeinander folgenden 9en könnte a_0 „groß“ werden.
Daher leicht andere Implementierung:

- Zehner-Stelle entscheidet, ob Ausgabe der Ziffern links von ihr
- Nach Subtraktion der ausgegebenen Ziffern: Speicherung der Ziffern vor der Einer-Stelle als *Vorziffern*
- Vorziffern stets speicherbar als Ziffer gefolgt von 9en

Der Zapfhahnalgorithmus für π

Zapfhahnalgorithmus

Eingabe: Gewünschte Anzahl Stellen n.

Ausgabe: n Stellen der Dezimaldarstellung von $\pi_{\lceil 10n/3 \rceil}$.

1: $A := (a_0; a_1, a_2, \ldots, a_{\lceil 10n/3 \rceil - 1})_c \leftarrow (2; 2, 2, \ldots, 2)_c$.
Vorziffern $\leftarrow 0$.

2: **Für** $i \leftarrow 1, 2, \ldots, n$ **erledige**

3: Multipliziere jedes Element von A mit 10 und regularisiere.

4: Reduziere a_0 modulo 10. q sei Quotient.

5: Wenn $q = 9$, Vorziffern $\leftarrow$ Vorziffern $\|$ q. Wenn $q = 10$, addiere 1 auf jede Vorziffer, gebe Vorziffern aus, Vorziffern $\leftarrow 0$. Andernfalls gebe alle Vorziffern aus, Vorziffern $\leftarrow q$.

6: **Ende von Für**

7: Wenn letzte Vorziffer $= 9$, addiere 1 auf jede Vorziffer. Andernfalls addiere 1 nur auf die letzte Vorziffer. Gebe alle Vorziffern aus.

Korrektheit

Lemma

1. *Der Zapfhahnalgorithmus ist korrekt. Insbesondere liefert er n Stellen von π mit einem Fehler $< 12\frac{2}{3} \cdot 10^{-n}$.*
2. *Er hat lineare Konvergenz sowie bezüglich der Eingabe n Laufzeit $\mathcal{O}(n^2)$ und Speicherplatz-Bedarf $\mathcal{O}(n)$.*

Beweis (nur 1): Der Algorithmus liefert Näherung $\widetilde{\pi}_m$ von π_m. π_m ist wiederum Näherung von π. Addiere maximale Fehler. $\square$

Weitere Eigenschaften des Algorithmus:

- Untere Schranke für Anzahl korrekt ausgegebener Ziffern bestimmbar
- Prinzipiell mit einer Tabellenkalkulation „implementierbar“ $\Rightarrow$ Live-Demo...

Erweiterungen des Zapfhahnalgorithmus

Laufzeitverbesserungen um einen konstanten Faktor n:

- In jedem Schleifendurchlauf Multiplikation mit 10^n (und nicht mehr nur mit 10)
- Berechnung der Stellen links der Einer-Stelle jeweils im (10^n)er-System
- Änderung mehr als einer Vorziffer nur, wenn $n+1$ aufeinander folgende Nullen an einer durch n teilbaren Position der Dezimaldarstellung von π_m (notwendige Bedingung)

Im Fall $n = 4$ (Multiplikation mit 10000):

- Änderung mehr als einer Vorziffer nur, wenn 5 aufeinander folgende Nullen an einer durch 4 teilbaren Position
- Abwarten auf immer nur eine neue Vorziffer für 50.000 Stellen ausreichend, da selbst für 4 Nullen erst bei Position 54.936 zum ersten Mal der Fall (in der Dezimaldarstellung von π)

Zapfhahnalgorithmus für die Euler'sche Zahl

Gibt es andere Basis, so dass der hypothetische Algorithmus funktionieren würde?

– Ja, zumindest für Euler'sche Zahl. Zapfhahnalgorithmus seit 1968 bekannt (Sale):

$$e = (2; 1, 1, 1, \ldots)_b \text{ mit Basis } b := (\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots)$$

Lemma

$(0; 1, 2, 3, \ldots)_b = 1$. *Demnach liegen reguläre und abbrechende Darstellungen der Form* $(0; a_1, a_2, a_3, \ldots)_b$ *im Interval* $[0, 1)$.

Beweis.

$$(0; 1, 2, 3, \ldots)_b = \sum_{i=2}^{\infty} \frac{i-1}{i!} = \frac{2-1}{2!} + \frac{3-1}{3!} + \ldots$$

$$= 1 - \frac{1}{2!} + \frac{1}{2!} - \frac{1}{3!} + \frac{1}{3!} - \cdots = 1 \qquad \square$$

Zapfhahnalgorithmus für die Euler'sche Zahl

Gibt es andere Basis, so dass der hypothetische Algorithmus funktionieren würde?
– Ja, zumindest für Euler'sche Zahl. Zapfhahnalgorithmus seit 1968 bekannt (Sale):

$$e = (2; 1, 1, 1, \dots)_b \text{ mit Basis } b := (\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots)$$

Lemma

$(0; 1, 2, 3, \dots)_b = 1$. *Demnach liegen reguläre und abbrechende Darstellungen der Form* $(0; a_1, a_2, a_3, \dots)_b$ *im Interval* $[0, 1)$.

Beweis.

$$(0; 1, 2, 3, \dots)_b = \sum_{i=2}^{\infty} \frac{i-1}{i!} = \frac{2-1}{2!} + \frac{3-1}{3!} + \dots$$

$$= 1 - \frac{1}{2!} + \frac{1}{2!} - \frac{1}{3!} + \frac{1}{3!} - \dots = 1 \qquad \square$$

Zapfhahnalgorithmus für die Euler'sche Zahl

Gibt es andere Basis, so dass der hypothetische Algorithmus funktionieren würde?
– Ja, zumindest für Euler'sche Zahl. Zapfhahnalgorithmus seit 1968 bekannt (Sale):

$$e = (2; 1, 1, 1, \dots)_b \text{ mit Basis } b := (\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots)$$

Lemma

$(0; 1, 2, 3, \dots)_b = 1$. *Demnach liegen reguläre und abbrechende Darstellungen der Form* $(0; a_1, a_2, a_3, \dots)_b$ *im Interval* $[0, 1)$.

Beweis.

$$(0; 1, 2, 3, \dots)_b = \sum_{i=2}^{\infty} \frac{i-1}{i!} = \frac{2-1}{2!} + \frac{3-1}{3!} + \dots$$

$$= 1 - \frac{1}{2!} + \frac{1}{2!} - \frac{1}{3!} + \frac{1}{3!} - \dots = 1 \qquad \square$$

Anderer Reihendarstellungen von π

Gibt es auch für π „bessere“ Basen?

– Ja, Gosper (1974):

$$\pi = 3 + \frac{1}{60}\left(8 + \frac{2 \cdot 3}{3 \cdot 7 \cdot 8}\left(\dots\left(5i - 2 + \frac{i(2i-1)}{3(3i+1)(3i+2)}\left(\dots\right)\right)\right)\right)$$

- $\pi = (3; 8, 13, 18, \dots)_d$ mit $d := \left(\frac{1}{60}, \frac{6}{168}, \frac{15}{330}, \dots\right)$
- Darstellung mit maximalen Ziffern $(0; a_1, a_2, a_3, \dots)_d \leq 1{,}092\dots$
- Rabinowitz und Wagon: „within 1% of spigot-perfection“

Anderer Reihendarstellungen von π

Gibt es auch für π „bessere“ Basen?
– Ja, Gosper (1974):

$$\pi = 3 + \frac{1}{60}\left(8 + \frac{2 \cdot 3}{3 \cdot 7 \cdot 8}\left(\ldots\left(5i - 2 + \frac{i(2i-1)}{3(3i+1)(3i+2)}\left(\ldots\right)\right)\right)\right)$$

- $\pi = (3; 8, 13, 18, \ldots)_d$ mit $d := (\frac{1}{60}, \frac{6}{168}, \frac{15}{330}, \ldots)$
- Darstellung mit maximalen Ziffern $(0; a_1, a_2, a_3, \ldots)_d \leq 1{,}092\ldots$
- Rabinowitz und Wagon: „within 1% of spigot-perfection“

Ein Streaming-Algorithmus zur Berechnung von π

Gibt es einen Zapfhahnalgorithmus, bei dem keine vorherige Festlegung auf die Anzahl zu berechnender Stellen notwendig ist?

– Ja, Gibbons (2004):

- Bislang vorherige Festlegung der zu berechnenden Stellen notwendig aufgrund der Art der Regularisierung
- Neu: Streaming-Algorithmus zum Regularisieren bezüglich einer unendlichen variablen Basis

  $\Rightarrow$ n Stellen von π bereits im ersten Anlauf sofort mit Wahrscheinlichkeit 100% exakt berechenbar (bislang: Letzte Ziffern unter Umständen inkorrekt, da nur Genauigkeit $12\frac{2}{3} \cdot 10^{-n}$ garantierbar)
- Jedoch: Ganzzahl-Arithmetik mit unbeschränkter Genauigkeit notwendig

Ein Streaming-Algorithmus zur Berechnung von π

Gibt es einen Zapfhahnalgorithmus, bei dem keine vorherige Festlegung auf die Anzahl zu berechnender Stellen notwendig ist?
– Ja, Gibbons (2004):

- Bislang vorherige Festlegung der zu berechnenden Stellen notwendig aufgrund der Art der Regularisierung
- Neu: Streaming-Algorithmus zum Regularisieren bezüglich einer unendlichen variablen Basis

  $\Rightarrow$ n Stellen von π bereits im ersten Anlauf sofort mit Wahrscheinlichkeit 100% exakt berechenbar (bislang: Letzte Ziffern unter Umständen inkorrekt, da nur Genauigkeit $12\frac{2}{3} \cdot 10^{-n}$ garantierbar)
- Jedoch: Ganzzahl-Arithmetik mit unbeschränkter Genauigkeit notwendig

Zusammenfassung

- Zapfhahnalgorithmus: Ausgabe bereits während des Ablaufs
- Sehr einfache Grundidee (siehe Algorithmus für e). Erschwernis bei π: (Noch) keine optimale „Zapfhahn“-Basis bekannt
- Dennoch: Zapfhahnalgorithmus leicht implementierbar, nur Ganzzahl-Arithmetik $\Rightarrow$ Van Ceulen hätte 35 Dezimalstellen mühelos auch auf Papier in relativ kurzer Zeit ermitteln können.

Zum Weiterlesen...

- Jörg Arndt & Christoph Haenel (2000).
  π *Algorithmen, Computer, Arithmetik*.
  Springer-Verlag, Berlin, Heidelberg, New York, 2nd edition.
- Jeremy Gibbons (2004).
  Unbounded Spigot Algorithms for the Digits of π.
  `http://web.comlab.ox.ac.uk/oucl/work/jeremy.gibbons/publications/spigot.pdf`.
- Stanley Rabinowitz & Stan Wagon (1995).
  A Spigot Algorithm for the Digits of π.
  American Mathematical Monthly **102**(3), 195–203.
  `http://links.jstor.org/sici?sici=0002-9890%28199503%29102%3A3%3C195%3AASAFTD%3E2.0.CO%3B2-H`.