APL-Journal

A Programming Language

1-2/2023

APL-Germany e.V.

Doppelnummer

Nr. 1-2 2023

Jahrgang 42

ISSN - 1438-4531

RHOMBOS-VERLAG

Caspar David Friedrich
Ruine Oybin

Dieter Kilsch
Vedische Mathematik

Herbert Voß
Typesetting external program code and its output: hvextern

Kamila Szewczyk
Seemingly Impossible APL Programs

Martin Barghoorn
Zeit und Raum
Erkenntnis durch Bilder

Liebe Mitglieder von APL Germany,
liebe APL-Freunde,

heute erhalten Sie die neue Ausgabe des *APL-Journals*. Die Beiträge basieren hauptsächlich auf Vorträgen während der beiden Tagungen des Jahre 2023 in Bingen am Rhein und Berlin-Adlershof. Diese fanden in Präsenz statt, aber das Interesse an Online-Teilnahme aus dem In- und Ausland ist nach Corona geblieben.

Mein Artikel stellt einige Rechenmethoden der vedischen Mathematik vor, wie sie im „alten Indien" benutzt worden sein soll. Sie kennt das Dezimalsystem einschließlich der Ziffer Null. Dieses Zahlensystem gelangte früh nach Arabien und der Mathematiker und Kaufmannssohn Fibonacci (Sohn des Bonacci) brachte es Anfang des dreizehnten Jahrhunderts nach Italien. Wir sprechen in unserem Zahlensystem zwar von arabischen Ziffern, aber nach Ihrem Ursprung sind es indische.

Martin Barghoorn sucht in seinem Artikel nach einer Ordnung seiner Bilder in Raum und Zeit und regt vielleicht einige Leser an, nach seinem Muster Ordnung die vielen Bildverzeichnisse auf dem Rechner zu bringen. Er benutzt und ergänzt die in einer Bilddatei gespeicherten Metadaten zu Aufnahmeort und -zeit.

In ihrer Arbeit geht Kamila Szewczyk, Universität Saarland und Dyalog Ltd., auf die Verarbeitung unendlicher Mengen am Beispiel des Suchprogramms von Berger und seine Implementierung in Dyalg-APL ein. Ferner stellt sie ihre Implementierung der für die Berechnung von Potenzreihen benötigten algebraischen und analytischen Funktionen in Dyalog-APL vor.

Herbert Voss zeigt in seinen Artikel zu LaTeX einen Weg, wie LaTeX-Duplikate oder Fremdprogramme angestoßen werden und deren Kode und/oder Ergebnisse in das Dokument einfließen. Dadurch wird sichergestellt, dass das kompilierte Hauptdokument immer die neuesten Versionen der Fremdprogramme und deren Berechnungen enthält.

Ich wünsche Ihnen viel Freude und Anregungen beim Lesen der Artikel.

Unsere letzte Mitgliederversammlung fand am 24. April 2023 statt. Zu dem verschickten Protokoll gab es keine Einwände, es ist damit beschlossen. Die nächste Tagung mit Mitgliederversammlung findet am 29./30. April an der Johannes-Gutenberg-Universität in Mainz statt und ist bei Erscheinen der gedruckten Ausgabe des Journals schon vorüber. Das Protokoll der Mitgliederversammlung erhalten Sie wieder per E-Mail.

Ich wünsche Ihnen weiterhin viel Erfolg und gute Gesundheit und freue mich auf die Teilnahme vieler Mitglieder an den Tagungen.

Dieter Kilsch

Kontakt:

Prof. Dr. Dieter Kilsch
E-mail: d.kilsch@th-bingen.de

INHALT

Vedische Mathematik

Dieter Kilsch

28. Dezember 2023

Inhaltsverzeichnis

1 Einleitung

Vedische Mathematik beschreibt Formeln und Verfahren der alten indischen Mathematik. In dieser Zeit wurden viele Aufgaben im Kopf gelöst und hierzu wurden viele Formeln zur Vereinfachung der Berechnung erstellt.

Die indische Mathematik kannte bereits unser Dezimalsystem und damit ein Stellensystem mit einer Null. Wir bezeichnen unsere Ziffern heute als arabische Ziffern. Fibonacci, ein italienischer Mathematiker und Sohn eines Kaufmanns Bonacci, brachte Sie aus Nordafrika nach Europa. Araber hatten Sie Jahrhunderte vorher in Indien kennengelernt und eingeführt. Es sind also in Wirklichkeit indische Ziffern.

Die Inhalte stammen aus [1], einige Erklärungen in § 3 sind dort aber nicht enthalten. Die Nachweise zu den Verfahren zu finden, war die Motivation zu dieser Arbeit.

Eine für mich sehr überraschende Vorgehensweise ist die Berechnung der periodischen Dezimalbrüche, in [1] am Beispiel $1/19$ vorgestellt. Alle einzelnen Schritte werden später begründet. Die Periode endet mit 1. Anschließend wird jede vorhergehende wegen des Nenners 19 durch Multiplikation mit 2 aus der aktuellen berechnet und ein möglicher Übertrag berücksichtigt:

(1)	$\frac{1}{19} = 0,\overline{??????????????????1}$, Übertrag 0.
(2) $2 \cdot 1 = 2$	$\frac{1}{19} = 0,\overline{??????????????????21}$, Übertrag 0.
(3) $2 \cdot 2 = 4$	$\frac{1}{19} = 0,\overline{??????????????????421}$, Übertrag 0.
(4) $2 \cdot 4 = 8$	$\frac{1}{19} = 0,\overline{??????????????????8421}$, Übertrag 0.
(5) $2 \cdot 8 = 16$	$\frac{1}{19} = 0,\overline{??????????????????68421}$, Übertrag 1.
(6) $2 \cdot 6 + 1 = 13$	$\frac{1}{19} = 0,\overline{??????????????368421}$, Übertrag 1.
(7) $2 \cdot 3 + 1 = 7$	$\frac{1}{19} = 0,\overline{??????????????7368421}$, Übertrag 0.
(8) $2 \cdot 7 + 0 = 14$	$\frac{1}{19} = 0,\overline{??????????47368421}$, Übertrag 1.

(9) $2 \cdot 4 + 1 = 9$ $\frac{1}{19} = 0,\overline{????????947368421}$, Übertrag 0.

(10) $2 \cdot 9 + 0 = 18$ $\frac{1}{19} = 0,\overline{???????8947368421}$, Übertrag 1.

(11) $2 \cdot 8 + 1 = 17$ $\frac{1}{19} = 0,\overline{??????78947368421}$, Übertrag 1.

(12) $2 \cdot 7 + 1 = 15$ $\frac{1}{19} = 0,\overline{?????578947368421}$, Übertrag 1.

(13) $2 \cdot 5 + 1 = 11$ $\frac{1}{19} = 0,\overline{????1578947368421}$, Übertrag 1.

(14) $2 \cdot 1 + 1 = 3$ $\frac{1}{19} = 0,\overline{???31578947368421}$, Übertrag 0.

(15) $2 \cdot 3 + 0 = 6$ $\frac{1}{19} = 0,\overline{??631578947368421}$, Übertrag 0.

(16) $2 \cdot 6 + 0 = 12$ $\frac{1}{19} = 0,\overline{?2631578947368421}$, Übertrag 1.

(17) $2 \cdot 2 + 1 = 5$ $\frac{1}{19} = 0,\overline{?5263157894736842 1}$, Übertrag 0.

(18) $2 \cdot 5 + 0 = 10$ $\frac{1}{19} = 0,\overline{052631578947368421}$, Übertrag 1.

Die weitere Berechnung $2 \cdot 0 + 1 = 1$ liefert die Ziffer 1 ohne Rest, mit der wir angefangen haben. Damit schließt sich die Periode.

Umgekehrt kann man nach dem Start $10 : 19 = 0$ Rest 10 von links nach rechts durch 2 mit Rest dividieren. Ein Rest führt zum Übertrag als Zehnerstelle für die nächste Division.

(1) $10 : 19 = 0$ Rest 10 $\frac{1}{19} = 0,\overline{0??????????????????}$.

(2) $10 : 2 = 5$ Rest 0 $\frac{1}{19} = 0,\overline{05??????????????????}$.

(3) $5 : 2 = 2$ Rest 1 $\frac{1}{19} = 0,\overline{052??????????????????}$.

(4) $12 : 2 = 6$ Rest 0 $\frac{1}{19} = 0,\overline{0526??????????????????}$.

$\ddots$

(5) $13 : 2 = 6$ Rest 1 $\frac{1}{19} = 0,\overline{052631578947 36????}$.

(6) $16 : 2 = 8$ Rest 0 $\frac{1}{19} = 0,\overline{0526315789473 68???}$.

(7) $8 : 2 = 4$ Rest 0 $\frac{1}{19} = 0,\overline{052631578947368??}$.

(8) $4 : 2 = 2$ Rest 0 $\frac{1}{19} = 0,\overline{0526315789473684?}$.

(9) $2 : 2 = 1$ Rest 0 $\frac{1}{19} = 0,\overline{052631578947368421}$.

Die Periode erreicht ihre maximale Länge 18.

2 Multiplikationen

2.1 Standardverfahren der Multiplikation

Auch in der vedischen Mathematik gibt es ein Standardverfahren zur Multiplikation zweier Zahlen ohne besondere Eigenschaften. Als Beispiel vergleichen wir das europäische mit dem vedischen Vorgehen: $499 \cdot 287 = 143213$:

Europ.	**Vedisch**				
$499 \cdot 287$			4	9	9
998			2	8	7
3992	$4 \cdot 2$	$4 \cdot 8 + 2 \cdot 9$	$4 \cdot 7 + 9 \cdot 8 + 9 \cdot 2$	$9 \cdot 7 + 9 \cdot 8$	$9 \cdot 7$
3493	6	13	14	6	
143213	$1\ 4$	3	2	1	3

Im vedischen System werden alle Produkte, deren Ergebnisse zur selben Dezimalstelle beitragen, zusammengefasst berechnet:

(a) Zur 1er-Stelle trägt nur das Produkt der 1er-Stellen bei: $9 \cdot 7 = 63$: 3 wird in der untersten Zeile eingetragen, 6 in der 10er-Spalte als Übertrag.

(b) Zur 10er-Stelle tragen die beiden Produkte aus 1er- und 10er-Stelle und der Übertrag bei: $9 \cdot 7 + 9 \cdot 8 + 6 = 141$: 1 wird in der untersten Zeile eingetragen, der Übertrag 14 in der 100er-Spalte.

(c) Zur 100er-Stelle tragen die beiden Produkte aus 1er- und 100er-Stelle, das Produkt der 10er-Stelle und der Übertrag bei: $4 \cdot 7 + 9 \cdot 8 + 9 \cdot 2 + 14 = 132$: 2 wird in der untersten Zeile eingetragen, der Übertrag 13 in der 1000er-Spalte.

(d) Zur 1000er-Stelle tragen die beiden Produkte aus 10er- und 100er-Stelle und der Übertrag bei: $4 \cdot 8 + 2 \cdot 9 + 13 = 63$: 3 wird in der untersten Zeile eingetragen, der Übertrag 6 in der 10000er-Spalte.

(e) Zur 10000er-Stelle trägt das Produkt der beiden 100er-Stelle und der Übertrag bei: $4 \cdot 2 + 6 = 14$: 4 wird in der untersten Zeile eingetragen, der Übertrag 1 ebenfalls in der untersten Zeile in der 100000er-Spalte.

Das Ergebnis lautet als 143213.

Um die Ziffern betragsmäßig kleiner zu halten, führte die vedische Mathematik negative Ziffern ein und berechnete statt $499 \cdot 287$ mit den negativen Ziffern durch Überstrich gekennzeichnet $50\overline{1} \cdot 3\overline{13}$. Das könnte man natürlich auch im europäischen System:

Europ.	**Vedisch**					
$50\overline{1} \cdot 3\overline{13}$				5	0	$\overline{1}$
$150\overline{3}$				3	$\overline{1}$	$\overline{3}$
$\overline{5}01$		$5 \cdot 3$	$5 \cdot \overline{1} + 0 \cdot 3$	$5 \cdot \overline{3} + 0 \cdot \overline{1} + \overline{1} \cdot 3$	$0 \cdot \overline{3} + \overline{1} \cdot \overline{1}$	$\overline{1} \cdot \overline{3}$
$1\overline{5}03$			$\overline{1}$			
$15\overline{6}\overline{8}13$	1	5	$\overline{6}$	$\overline{8}$	1	3
143213	1	4	3	2	1	3

2.2 Multiplikation nahe Vielfachen und Teilern von Zehnerpotenzen

Verfahren 2.1 *Dieses Verfahren setzt voraus, dass die beiden Faktoren die Voraussetzung der Überschrift erfüllen. Dann kann man die Multiplikation nach der Regel*

$$\begin{aligned} A \cdot B &= (10^n \cdot x + a)(10^n \cdot x + b) = 10^n \cdot x(10^n \cdot x + a + b) + ab \\ &= 10^n \cdot x(A + b) + ab = 10^n \cdot x(B + a) + ab \,, \end{aligned}$$

vereinfachen.

Wir beginnen mit einem Beispiel mit $n = x = 1$:

Beispiel 2.2 (Zahlen in der Nähe von 10)

$$8 \cdot 13 = (10 - 2)(10 + 3) = 10 \cdot (8 + 3) + (-2) \cdot 3 = 10 \cdot 11 - 6 = 104 \,.$$

In die Tabelle füllen wir in die erste Spalte die beiden zu multiplizierenden Zahlen und in die zweite die Differenz zu 10: Die 10er-Stelle erhalten wir durch kreuzweise Addition: $7 - 2$ oder $8 - 1$, die 1er-Stelle durch Multiplikation der beiden positiven und negativen Zehnerdifferenzen.

8	-2
13	3
11	-6

$11 \cdot 10 - 6 = 104$

Im zweiten Beispiel betrachten wir $n = 2$:

Beispiel 2.3 (Zahlen in der Nähe von 100)

$$91 \cdot 112 = (100 - 9)(100 + 12) = 100(91 + 12) - 9 \cdot 12 = 10300 - 108 = 10192$$

In die Tabelle füllen wir in die erste Spalte die beiden zu multiplizierenden Zahlen und in die zweite die Differenz zu 100: Die 100er-Stelle erhalten wir durch kreuzweise Addition: $91 + 12$ oder $1128 - 9$, die Einerstelle durch Multiplikation der beiden positiven und negativen Hunderterdifferenzen. Wir erhalten $91 \cdot 112 = 10300 - 108 = 10192$.

91	-9
112	12
103	-108

Jetzt betrachten wir Beispiele mit $x \neq 1$, die Zahlen liegen also nicht in der Nähe einer 10er-Potenz.

Beispiel 2.4 (Zahlen in der Nähe von 50 mit Basis 100) Mit $n = 2$ und $x = \frac{1}{2}$, also der Basis 100 und dem Quotienten 2 erhalten wir mit 2.1:

$$48 \cdot 53 = \frac{1}{2} \cdot 100 \cdot (48 + 3) - 6 = \frac{1}{2} \cdot 5100 - 6 = 2550 - 6 = 2544 \, .$$

In der ersten Spalte (100er-Spalte) der Tabelle werden die beiden Zahlen notiert, in der zweiten die Differenz zu 50. Die dritte Zeile enthält das Ergebnis nach bekanntem Schema. Die vierte Zeile berücksichtigt den Divisor 2. $51 : 2 = 25$ Rest 1. Für diesen werden in der zweiten Spalte 50 addiert. Das Ergebnis lautet $58 \cdot 43 = 2500 + 44 = 2544$.

48	-2
53	3
51	-6
25	50-6

Alternativ kann auch mit der Basis 10 und dem Faktor 5 gearbeitet werden:

Beispiel 2.5 (Zahlen in der Nähe von 50 mit Basis 10) Mit $n = 1$ und $x = 5$, also der Basis 10 und dem Faktor 5 erhalten wir mit 2.1:

$$48 \cdot 53 = 5 \cdot 10 \cdot (48 + 3) - 6 = 5 \cdot 510 - 6 = 2550 - 6 = 2544 \, .$$

Für die vierte Zeile muss die Zahl in der ersten Spalte (10er-Spalte) mit 5 multipliziert werden. Das Ergebnis ist $2550 - 6 = 2544$.

48	-2
53	3
51	-6
255	-6

Beispiel 2.6 (Nebenrechnung bei größerem Abstand)

$32 \cdot 63$ berechnen wir um 50 mit Basis 10 und Faktor 5, die Nebenrechnung mit Basis 10 und Faktor 1.
Wir erhalten $32 \cdot 63 = 2250 - 234 = 2016$

		N.B.:		
32	-18		18	8
63	13		13	3
45	-234		21	24
225	-234		23	4

Beispiel 2.7 (Zahlen in der Nähe von 250)

$260 \cdot 244$ berechnen wir um 250 mit Basis 1000 und Divisor 4. Das Ergebnis lautet $260 \cdot 244 = 63000 + 500 - 60 = 63440$.

260	10
244	-6
254	-60
$63 + \frac{1}{2}$	-60

2.3 Weitere Sonderfälle

Verfahren und Beispiele 2.8 (Faktoren mit gleichem Zehneranteil)

$$\begin{aligned} A \cdot B &= (10x + a)(10x + b) = 100x^2 + 10x(a + b) + ab = 10x(10x + a + b) + ab \\ &= 10x(A + b) + ab = 10x(B + a) + ab \end{aligned}$$

Beispiele:

- $23 \cdot 28 = 20 \cdot (23 + 8) + 3 \cdot 8 = 20 \cdot 31 + 24 = 644$,
- $23 \cdot 38 = 20 \cdot (23 + 18) + 3 \cdot 18 = 20 \cdot 41 + 54 = 874$,

  aber auch
- $23 \cdot 38 = 30 \cdot (23 + 8) - 7 \cdot 8 = 30 \cdot 31 - 56 = 874$,
- $37 \cdot 44 = 40 \cdot (37 + 4) - 3 \cdot 4 = 40 \cdot 41 - 12 = 1628$.

Verfahren und Beispiele 2.9 (Gleicher Zehneranteil, Summe der Einerstelle gleich zehn)

$$(10x + a)(10x + b) = 100x^2 + 10x(a + b) + ab = 100x(x + 1) + ab$$

Beispiele:

- $93 \cdot 97 = 100 \cdot 9 \cdot 10 + 21 = 9021$
- $44 \cdot 46 = 100 \cdot 4 \cdot 5 + 24 = 2024$

Verfahren und Beispiele 2.10 (Gleicher 100er-Anteil, Summe der 2 letzten Stellen gleich 100)

$$(100x + a)(100x + b) = 10000x^2 + 100x(a + b) + ab = 10000x(x + 1) + ab$$

Beispiel:

- $123 \cdot 177 = 10000 \cdot 1 \cdot 2 + 23 \cdot 77 = 20000 + 1771 = 21771$

Verfahren und Beispiele 2.11 (Dritte binomische Formel)

$$(a + b)(a - b) = a^2 - b^2$$

Beispiele:

- $24 \cdot 26 = (25 - 1)(25 + 1) = 625 - 1 = 624$,
- $104 \cdot 96 = (100 + 4)(100 - 4) = 10000 - 16 = 9984$.

2.4 Quadratzahlen

Verfahren 2.12 (Zahlen in der Nähe eines Vielfachen einer Zehnerpotenz)

$$A^2 = (10^n \cdot x + a)^2 = x10^n(x10^n + a + a) + a^2 = x10^n(A + a) + a^2$$

Beispiel 2.13 ($x = 1 \ \wedge \ n = 1$)

- $13^2 = 10 \cdot (13 + 3) + 9 = 169.$

13	3
13	3
16	9

- $107^2 = 100(107 + 7) + 7^2 = 11449$
- $93^2 = 100(93 - 7) + 49 = 8649$

Beispiel 2.14 ($n > 1 \ \vee \ x > 1$)

Mit $x = 7$ und n=2, also der Basis 100 mit dem Faktor 7, erhalten wir $685^2 = (7 \cdot 10^2 - 12)^2 = 7 \cdot 67000 + 225$.

685	-15
685	-15
670	225

Verfahren und Beispiele 2.15 (Zahlen mit Einerstelle 5) (Vergleiche (2.9) und (2.12)!)

$$A^2 = (10x + 5)^2 = 100x^2 + 100x + 25 = 100x(x + 1) + 25$$

Beispiele:

- $15^2 = 100 \cdot 1 \cdot 2 + 25 = 225$
- $45^2 = 100 \cdot 4 \cdot 5 + 25 = 2025$
- $115^2 = 100 \cdot 11 \cdot 12 + 25 = 13225$
- $135^2 = 100 \cdot 13 \cdot 14 + 25 = 18225$.

2.5 Nicht-dezimale Maßeinheiten

Indien besaß ein nicht-dezimales Währungsystem (1 Rupie = 16 Annas; 1 Anna = 12 Pies)und benutzte speziell in der britischen Zeit auch nicht-dezimale Längen- und Flächenmaße (1 foot = 12 inches).

Beispiel 2.16 Wie groß ist die Fläche eines Rechtecks mit den Seitenlängen 4'7" und 8'9"?

Die Tabelle enthält in der dritten Zeile in der ersten Spalte mit $4 \cdot 8$ das Produkt der beiden Fuß-, in der letzten mit $7 \cdot 9$ der beiden Zoll-Angaben[1] und in der mittleren die Summe der beiden kreuzweisen Produkte.

Aus $92\,\text{ft.in.} = 7 \cdot 12 + 8\,\text{ft.in.} = 7\,\text{sq.ft.} + 8 \cdot 12\,\text{sq.in.}$ folgt die vierte und fünfte Zeile, die sechste aus $1\,\text{sq.ft.} = 144\,\text{sq.in.}$.
Damit folgt das Ergebnis: 4'7" · 8'9" = 40 sq.ft. 15 sq.in.

4		7
8		9
sq.ft.	ft.·in.	sq.in.
32	92	63
32	7·12 + 8	63
32+7		63+8·12
39		159
40		15

Natürlich hilft APL bei solchen Berechnungen:

```
      144 144⊤×/12⊥⍣¯1(4 7)(8 9)
40 15
```

Beispiel 2.17 Eine investierte Rupie erbringt 2 Rupien 5 Annas, welchen „Return of Invest" bringen 4 Rupien 9 Annas?

Bei der Berechnung verzichte ich auf die Umrechnung von Annas in Pies. Die Tabelle liefert nach obigem Vorgehen als Ergebnis: 4 Rp. 9 An. bringen einen Erlös von 10 Rp. $8\,{}^{13}/_{16}$ An.

2		5
4		9
Rupien	Annas	Annas
8	38	${}^{45}/_{16}$
8	2·16 + 6	${}^{45}/_{16}$
8+2		${}^{45}/_{16}$+ 6
10		8 + ${}^{13}/_{16}$

Mit APL erhält man:

```
      16 16 16⊤×/16⊥⍣¯1(2 5)(4 9)
10 8 13
```

3 Periodische Dezimalbrüche

3.1 Grundlagen

Satz 3.1 *Die letzte Ziffer der Periode des Dezimalbruchs* $\frac{1}{n} = 0.\overline{a_1 a_2 \ldots a_k}$ *multipliziert mit der letzten Ziffern von* n *ist* $9 \bmod 10$*:*

$$a_k \cdot n \equiv 9 \bmod 10\ .$$

Beweis:

$$\begin{aligned}
\underbrace{99\ldots99}_{\text{k Ziffern}} \cdot \frac{1}{n} &= \underbrace{99\ldots99}_{\text{k Ziffern}} \cdot 0.\overline{a_1 a_2 \ldots a_k} = \left(10^k - 1\right) \cdot 0.\overline{a_1 a_2 \ldots a_k} \\
&= a_1 a_2 \ldots a_k.\overline{a_1 a_2 \ldots a_k} - 0.\overline{a_1 a_2 \ldots a_k} \\
&= a_1 a_2 \ldots a_k \\
\underbrace{99\ldots99}_{\text{k Ziffern}} &= n \cdot a_1 a_2 \ldots a_k
\end{aligned}$$

◇

Satz 3.2 *Sind* n *und 10 teilerfremd, so gelten:*

(a) Bei der Berechnung des Dezimalbruchs von $\frac{1}{n}$ *bilden die Reste*

$$r_0 = 1;\ r_i = 10 r_{i-1} - a_i n$$

eine geometrische Folge modulo n *mit dem Faktor* 10*.*

(b) Die Periode des Dezimalbruchs von ${}^1/_n$ *beginnt mit der ersten Stelle hinter dem Komma.*

[1] 1 Fuß = 1 feet; 1 Zoll = 1 inch

Beweis:

(a) $1 \rightarrow 10 = a_1 \cdot n + r_1 \quad \text{mit} \quad 10 \equiv r_1 \bmod n$
$\rightarrow 10r_1 = a_2 \cdot n + r_2 \quad \text{mit} \quad 10r_1 \equiv r_2 \bmod n$
$\rightarrow \ldots$

(b) 10 ist eine Einheit in $\mathbb{Z}/n\mathbb{Z}$, also $10^{\varphi(n)} = 1$ mit $\varphi(n) = \left|\left(\mathbb{Z}/n\mathbb{Z}\right)^{\times}\right|$. Nach $\varphi(n)$-maligem Anwenden des Faktors 10 erhält man wieder den Rest 1 und die Periode beginnt von vorne. Die Periode ist damit ein Teiler von $\varphi(n)$. ◇

Beispiel 3.3 $\frac{1}{7} : 1 \rightarrow 3 \rightarrow 2 \rightarrow 6 \rightarrow 4 \rightarrow 5 \rightarrow \frac{1}{7} = 0.\overline{142857}$

Die Reste und Ziffern des Dezimalbruchs folgen aus

$$\begin{aligned} 10 \cdot 1 &= 10 = \mathbf{1} \cdot 7 + 3 \\ 10 \cdot 3 &= 30 = \mathbf{4} \cdot 7 + 2 \\ 10 \cdot 2 &= 20 = \mathbf{2} \cdot 7 + 6 \\ 10 \cdot 6 &= 60 = \mathbf{8} \cdot 7 + 4 \\ 10 \cdot 4 &= 40 = \mathbf{5} \cdot 7 + 5 \\ 10 \cdot 5 &= 50 = \mathbf{7} \cdot 7 + 1 \end{aligned}$$

Beachten Sie Satz 3.1: $7 \cdot 7 = 9 \bmod 10$.

Satz 3.4 *Für* $n^2 \equiv 9 \bmod 10$ *gilt* $a_{i-1} \equiv n \cdot r_i \bmod 10$.

Beweis:

$r_i = 10r_{i-1} - a_{i-1} \cdot n \quad \rightarrow \quad r_i\, n = 10r_{i-1}n - a_{i-1} \cdot n^2 \equiv -a_{i-1} \cdot (-1) \bmod 10$
$\rightarrow \quad r_i\, n \equiv a_{i-1} \bmod 10$ ◇

Beispiel 3.5 $\frac{1}{7} : \cancel{1 \rightarrow} 3 \rightarrow 2 \rightarrow 6 \rightarrow 4 \rightarrow 5 \rightarrow 1 \rightarrow \frac{1}{7} = 0.\overline{142857}$

Die Ziffern des Dezimalbruchs folgen aus

$$\begin{aligned} 7 \cdot 3 &= \mathbf{21} \\ 7 \cdot 2 &= \mathbf{14} \\ 7 \cdot 6 &= 4\mathbf{2} \\ 7 \cdot 4 &= \mathbf{28} \\ 7 \cdot 5 &= \mathbf{35} \\ 7 \cdot 1 &= \mathbf{7} \end{aligned}$$

3.2 Weitere Ergebnisse

Bemerkung 3.6

$$\begin{aligned} &\tfrac{1}{7} : 1 \rightarrow 3 \rightarrow 2 \rightarrow 6 \rightarrow 4 \rightarrow 5 \\ &\quad 1 \rightarrow 3 \rightarrow 2 \rightarrow -1 \rightarrow -3 \rightarrow -2 \,(\bmod 7) \end{aligned}$$

Die Reste erhält man durch Multiplikation modulo 7 von links nach rechts und durch Multiplikation mit 5 von rechts nach links, denn $10 \cdot 5 = 50 \equiv 1 \bmod 7$. *Die Reste der zweiten Hälfte zu den Resten der ersten Hälfte addiert ergibt jeweils 7.*

Diese Beobachtung kann verallgemeinert werden:

Satz 3.7 *Ist die Länge der Periode gerade, so ist die Summe der Reste der ersten Hälfte und der Reste der zweiten Hälfte gleich dem Nenner. Präziser bei größerem Nenner: Nimmt man in der zweiten Hälfte die Reste negativ, so sind die Summen null.*

BEWEIS: Mit der Periodenlänge l gilt $\left(10^{l/2}\right)^2 = 10^l = 1$ und damit $10^{l/2} \equiv \pm 1$. Bei $10^{l/2} \equiv 1$ wäre die Periodenlänge $\frac{l}{2}$. ◇

Satz 3.8 *Ist die Periodenlänge eines Dezimalbruchs gerade und gilt* $n^2 \equiv 9 \bmod 10$*, so ist die Summe der Ziffern der ersten Hälfte einer Periode der Dezimalbruchdarstellung und der Ziffern der zweiten Hälfte jeweils 9.*

BEWEIS: Die korrespondierende Ziffer zu k in der zweiten Hälfte ist nach Satz 3.7 $n-k$ $(\equiv -k)$. Damit folgt $kn + (n-k)n = n^2$ ◇

Ich nenne diese Eigenschaft eines Dezimalbruch *9-komplementär*. Sie gilt garantiert bei Nennern, die auf 3 und 7 enden, aber teilweise auch bei Nennern, die auf 1 oder 9 enden, siehe Beispiele ab 3.12.

Beispiel 3.9 (Siebtel)

		(1 | →)	3	→	2	→	6	→	4	→	5	→	1
$\frac{1}{7}$	=	0.		1		4		2		8		5		7
		(2 | →)	6	→	4	→	5	→	1	→	3	→	2
$\frac{2}{7}$	=	0.		2		8		5		7		1		4
$\frac{5}{7}$	=	0.		7		1		4		2		8		5

Die Multiplikation der letzten Ziffer der Periode des Dezimalbruchs von $1/7$ mit zwei legt die letzte Ziffer der Periode des Dezimalbruchs zu $2/7$ fest. Die Weiteren ergeben sich durch zyklisches Permutieren, da auch die Reste zyklisch permutiert werden, denn es gilt Satz 3.4. Das gilt analog auch bei $5/7$.

Beispiel 3.10 (Dreizehntel)

		(1 | →)	10	→	9	→	12	→	3	→	4	→	1	
$\frac{1}{13}$	=	0.		0		7		6		9		2		3	
		(2 | →)	7	→	5	→	11	→	6	→	8	→	2	
$\frac{2}{13}$	=	0.		1		5		3		8		4		6	
$\frac{5}{13}$	=	0.		3		8		4		6		1		5	(endet mit 5)
$\frac{11}{13}$	=	0.		8		4		6		1		5		3	(endet mit 3)

Auch hier legt die Multiplikation der letzten Ziffer der Periode des Dezimalbruchs von $1/13$ die letzte Ziffer der Periode des Dezimalbruchs zu $n/13$ fest. Allerdings muss diese nicht im Dezimalbruch von $1/13$ auftreten. Der Dezimalbruchs von $2/13$ enthält die fehlenden Ziffern. Die letzte Ziffer der Periode von $n/13$ kann also in $1/13$ oder $2/13$ auftreten und dies bestimmt die Reihenfolge der Ziffern der Periode von $n/13$.

3.3 Divisor endet auf neun

Satz 3.11 *$n = 10m - 1$ sei eine natürliche Zahl mit der letzten Ziffer neun.*

(a) Die Periode des Dezimalbruchs von $1/n$ endet mit 1.

(b) Die Ziffern der Periode $1/n = 0.\overline{a_1 a_2 \ldots a_{k-2} a_{k-1} 1}$ können von rechts mit 1 beginnend durch Multiplikation mit Übertrag mit m berechnet werden.

BEWEIS:

(a) folgt aus Satz 3.1.

(b)

- *$n = 9$: Bei $\frac{1}{9} = 0.\overline{1}$ liegt das Ergebnis mit dem ersten Schritt bereits fertig vor.*
- *$n > 10$. In diesem Fall ist $m > 1$ und die erste Stelle der Periode eine null. Mit $n = 10m - 1$ folgt aus $\frac{1}{n} = 0.\overline{0 a_2 a_3 \ldots a_{k-3} a_{k-2} a_{k-1} 1}$*

$$\begin{aligned} 1 &= \left(10m \cdot \frac{1}{n}\right) - \frac{1}{n} = m \cdot 0.\overline{a_2 a_3 \ldots a_{k-1} 10} - 0.\overline{0 a_2 a_3 \ldots a_{k-3} a_{k-2} a_{k-1} 1} \\ &= u_2.\overline{r_2 r_3 \ldots r_{k-3} r_{k-2} r_{k-1} r_k u_2} - 0.\overline{0 a_2 a_3 \ldots a_{k-3} a_{k-2} a_{k-1} 1} \end{aligned}$$

mit $m \cdot a_i + u_{i+1} = 10 \cdot u_i + r_i \equiv r_i \bmod 10$ $(i = 2, \ldots, k-1)$.
Aus dieser Gleichung folgen:

- $u_2 = 1$,
- $r_2 = 0$,
- $a_i = r_{i+1} = m \cdot a_{i+1} + u_{i+2} - 10 \cdot u_{i+1}$ *für* $i = 2, \ldots, k-2$.
- $a_{k-1} = r_k = m - 10 \cdot u_k$, *denn* $u_{k+1} = u_1 = 0$.

◇

Beispiel 3.12 Bei der Berechnung des Dezimalbruchs kann man anstelle der Multiplikation von rechts nach links auch von links nach rechts Dividieren! Die Mitte in den Dezimalbrüchen ist durch einen senkrechten Strich markiert.

(a) $\frac{1}{19} = 0.\overline{052631578|947368421}$.

(b) $\frac{1}{29} = 0.\overline{03448275862068|96551724137931}$.

(c) $\frac{1}{39} = 0.\overline{025|641}$ Dieser Dezimalbruch ist nicht 9- sondern 6-komplementär!

(d) $\frac{1}{49} = 0.\overline{020408163265306122448|979591836734693877551}$

(e) $\frac{1}{69} = 0.\overline{01449275362|31884057971}$ Nicht komplementär!

(f) $\frac{m}{99} = 0.\overline{0m}$ m-komplementär!

Die Beispiele zeigen, das auch die Dezimalbrüche von Brüchen $1/n$, deren Nenner n auf 9 enden, oft 9-komplementär sind, aber nicht immer! Satz 3.8 greift hier natürlich nicht. Eine klare Aussage zu treffen, welche dieser Brüche 9-komplementär sind, ist mir nicht gelungen.

Es folgen weitere abgeleitete Beispiele:

Beispiel 3.13 (Multiplikation eines bekannten Bruchs)

(a) 3.12 (c): $\frac{1}{13} = \frac{3}{39} = 3 \cdot 0.\overline{025641} = 0.\overline{076|923}$.
Die Ziffernfolge kann von rechts nach links durch Multiplizieren mit vier berechnet werden.

(b) 3.12 (d):
$\frac{1}{7} = \frac{7}{49} = (7 \cdot 0.\overline{020408163265306122448|979591836734693877551}) = 0.\overline{142|857}$

(c) 3.12 (e): $\frac{1}{23} = \frac{3}{69} = 0.\overline{04347826086|95652173913}$

(d) $\frac{1}{17} = \frac{7}{119} = 0.\overline{05882352|94117647}$

Alle Beispiele sind 9-komplementär.

Beispiel 3.14 (Division eines bekannten Bruchs)

(a) 3.12 (a): $\frac{1}{38} = \frac{1}{2} \cdot \frac{1}{19} = \frac{1}{2} \cdot = 0.0\overline{263157894|786842105}$

(b) 3.13 (b): $\frac{1}{21} = \frac{1}{3} \cdot \frac{1}{7} = \frac{1}{3} \cdot 0.\overline{142|857} = 0.\overline{047|619}$

Diese Beispiele sind nicht komplementär.

Beispiel 3.15 (Nenner endet auf eins)

(a) $\frac{1}{11} = 0.\overline{09}$ ist 9-komplementär.

(b) $\frac{1}{21} = 0.\overline{047619}$ ist nicht komplementär.

(c) $\frac{1}{31} = 0.\overline{032258064516129}$ hat ungerade Periodenlänge 15.

(d) $\frac{1}{41} = 0.\overline{02439}$ hat ungerade Periodenlänge 5.

(e) $\frac{1}{51} = 0.\overline{0196078431372549}$ ist nicht komplementär.

(f) $\frac{1}{61} = 0.\overline{016393442622950819672131147540983606557377049180327868852459}$
ist 9-komplementär.

```
r←a Divper b;y;i;aa
⍝  V1.1  01.11.2023  D.Kilsch
⍝ Berechnen der Ziffern einer Periode des Bruchs a/b für b teilerfremd zu 10
⍝ für einen Bruch, dessen Nenner teilerfremd zu 10 ist.
⍝ a, b  S    Zähler und Nenne des Bruchs
⍝ r     TV   Dezimalbruch bis zum Ende der ersten Periode
⍝       AV[1]  S  <0: Fehlernummer, dann:
⍝                  ¯401: Nenner nicht teilerfremd zu 10
⍝         [2]  AV TV  Fehlertext
⍝         [3]  TV     Aufrufsequenz
⍝═
⍝  lokale Variable
⍝ aa    S   Rest bei Division von zehnmal vorhergehendem Rest durch b
⍝ i     S   Schleifenzähler, Periodenlänge, halbe Periodenlänge
⍝       V   stellenweise Summe der Ziffern der erstenn und zweiten Hälfte
⍝ y     S   ganzer Anteil bei Division von zehnmal vorherg. Rest durch b
⍝═══════════════════════════════════════════════════════════════════════
→(0=2 5|b)/F1
r←(⍕y←⌊a÷b),'.'⋄i←0⋄aa←10×a-y×b
DO:
r←r,⍕y←⌊aa÷b
aa←10×aa-y×b
→(aa≠10×a)/DO
⍝═══  Untersuchung der Periodenart
→(0=2|i←¯2+⍴r)/ELSE1
'Periodenlänge ungerade: ',⍕i⋄→ENDE          ⍝ i: Periodenlänge
ELSE1:i←i÷2⋄i←(⍎¨i↑2↓r)+⍎¨(i+2)↓r             ⍝ i: halbe Periodenlänge
→(0=∧/i=↑i)/ELSE2
(⍕↑i),'-komplementärer Dezimalbruch'⋄→ENDE   ⍝ i: Ziffersummen 1.,2.Hälfte
ELSE2:'nicht komplementärer Dezimalbruch'
ENDE:→0
⍝═══════  Fehlermeldungen  ═══════════════  i neu belegt  ═══════
F1:i←'Nenner nicht teilerfremd zu 10'⋄r←¯401⋄→FE
FE:r←r('Divper: ',i)((⍕↑1↓⎕LC),' ')
```

Abb. 1: APL-Routine zur Berechnung der Periode

Literatur

[1] S. Bharati Krishna Tirtha. Vedic Mathematics. New Dehli: Motilal Banarsidass, 1992. ISBN: 978-81-208-0164-6.

[2] D. J. Robinson. Abstract Algebra: An Introduction with Applications. 2^{nd} ed. De Gruyter Textbook. Berlin: Walter de Gruyter, 2015. ISBN: 978-3-11-034086-0.

Seemingly Impossible APL Programs

Kamila Szewczyk
Universität des Saarlandes
Dyalog Ltd.
kspalaiologos@gmail.com

1 Introduction

Envision a hypothetical black-box apparatus scanning an infinitely long punched tape containing a message. If it is considered appropriate, a green lamp turns on; otherwise, the tape is shredded. The objective of this thought experiment is to devise a way to algorithmically generate valid tapes given any black-box machine. Mathematically speaking, the tape is merely a function $C := \mathbb{N} \to \mathbb{B}$ (a Cantor space, conventionally denoted as $2^{\mathbb{N}}$ and assumed to be total throughout the paper). Consequently, the device is a total function $(\mathbb{N} \to \mathbb{B}) \to \mathbb{B}$. The thought experiment reduces to finding such p that given a $f : C \to \mathbb{B}$, $f(p) = 1$.

Generally speaking, it is usually impossible to establish decidable equality between functions $A \to B$ such that A is infinite[1]. However, there are many function types for which decidable equality can be established. For example, the set $C \to A$ admits decidable equality for all sets A that admit equality themselves.[2]

2 Topological observations

Finite parts of the function's output depend on finite parts of the function's input, hence it is continuous. Further, it has also been demonstrated that f is uniformly continuous[3]. Consider distinct sequences α and β such that subsequences α' and β' that consider the first m elements, where $\alpha' \equiv \beta'$ also implies equality under f, i.e. $f(\alpha') \equiv f(\beta')$. The paper considers minimum m, which is habitually called the *modulus of uniform continuity*[4].

3 Cantor Search program by U. Berger

Ulrich Berger (1990) suggests the following Cantor space search program[5]:

```
data Bit = Zero | One
type Cantor = [Bit]
find :: (Cantor -> Bool) -> Cantor
forsome, forevery :: (Cantor -> Bool) -> Bool
find p = if forsome(\a -> p(Zero : a))
   then Zero : find(\a -> p(Zero : a))
   else One : find(\a -> p(One : a))
forsome p = p(find p)
forevery p = not(forsome(\a -> not(p a)))
```

One way to make this program easier to understand is to rewrite the mutually recursive call in `find` as follows:

```
find p = if p(Zero : find(\a -> p(Zero : a))
   then Zero : find(\a -> p(Zero : a))
   else One : find(\a -> p(One : a))
```

The input argument $f : C \to \mathbb{B}$ to `forevery` or `forsome` has a certain modulus of uniform continuity m. For example, if $m = 0$, then f is a constant function.

Apply inductive reasoning with the base case of $m = 0$ where equality is trivially established by sampling the functions for the Cantor space $g(n) = 0$. When $m > 0$, an element is appended to the front of the currently considered Cantor space[1] g_0 such that $g(n) = a$ if $n = 0$ and $g(n) = g_0(n)$ otherwise. Since $\mathbb{B}$ is finite, it is possible and sufficient to permute $a \in \{0, 1\}$. If such a Cantor space makes the input function return a truthy value, the computation is over and a suitable Cantor space has been found. Otherwise, the process is repeated on both of the Cantor spaces obtained by permuting a, however, this operation decreases m in the recursive calls by one. As such, it is trivial to deduce that Berger's algorithm terminates and has exponential complexity.

4 An APL implementation

The author suggests the following APL implementation further verified with Dyalog APL/S-64 Version

[1] A *cons* operation, represented by a colon in Berger's code.

18.2.45405. `⎕io←0` and further `(⎕fr ⎕pp)←1287 34` is assumed for all code presented in the paper.

```
Cantor←{
  C←{r←⎕NS⍬ ⋄ r.f←⍵.f{⍵=0:⍵⍵ ⋄ ⍺⍺ ⍵-1}⍺ ⋄ r}
  F←{b←⍺⍺ P 0 ⋄ ⍺⍺ b:b ⋄ ⍺⍺ P 1}
  P←{r←⎕NS⍬ ⋄ r.P←P ⋄ r.C←C ⋄ r.F←F
     r.f←⍺⍺∘(⍵∘C){(⍺⍺ F ⍬).f ⍵} ⋄ ⍵ C r}
  A←{⍺⍺(~∘⍺⍺ F)⍵}
  (⍺⍺≡⍵⍵)A ⍵
 }
```

Where `A` is Berger's `forevery` quantifier which is passed `⍺⍺≡⍵⍵` argument (i.e. it determines equality between the dop's functional operands) and `C` is the *cons* operation for Cantor spaces. Finally, `F` is the find function which asks to permute g_0 and yields the result if found, while `P` handles the process of prepending and recursively calling back to `F`.

4.1 Example usage

Despite exponential complexity, it is practical to establish equality in moderately advanced cases, for example:

```
  x←{⊃∧/⍵.f¨1 3 5}
  (x Cantor {(⍵.f 1)∧(⍵.f 3)∧(⍵.f 6)})⍬
0
  (x Cantor {(⍵.f 1)∧(⍵.f 3)∧(⍵.f 5)})⍬
1
  ({3=+/⍵.f¨⍸⍳5} Cantor {3=+/⍵.f¨⍳5})⍬
1
  ({3=+/⍵.f¨⍸⍳5} Cantor {3=+/⍵.f¨⍳4})⍬
0
```

4.2 Potential improvements

Martín Escardo has previously presented a variety of improvements to Ulrich Berger's method[2].

5 Formal power series

A closely related concept to the Cantor space is a formal power series, which will be represented in the paper as a function $\mathbb{N} \to \mathbb{R}$.[2]

A formal power series is an infinite sum $\sum_{i=0}^{\infty} a_i x^i$ that is considered independently from any notion of convergence and can be manipulated with the usual algebraic operations on series. This concept can be thought of as the equivalent of `x⊥a¨⍳∞` in APL[7].

Consider the power series of $\cos x$ obtained through Taylor's theorem:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}$$

An APL representation of it is as follows. Consider the coefficients of the powers of `⍵`. Since formal power series is a dfn $\mathbb{N} \to \mathbb{R}$, i.e. a mapping from the term number to the coefficient:

```
cos←{
  2|⍵:0
  n←2|0.5×⍵
  n:÷-!⍵ ⋄ ÷!⍵
}
```

5.1 Elmentary operations

Many elementary operations follow from the rules of algebra:

```
neg←{-⍺⍺ ⍵}
add←{(⍺⍺+⍵⍵) ⍵}
sub←{(⍺⍺-⍵⍵) ⍵}
```

Further, two convenience operations `cons` and `const`, where `cons` takes a polynomial left argument of degree n, multiplies the formal power series by x^n and adds the polynomial to the result. `const` introduces a polynomial function.

```
const←{⍵<≢⍺:⍵⊃⍺⋄0}
cons←{⍵<≢⍺:⍵⊃⍺⋄⍺⍺ ⍵-≢⍺}
```

Multiplication of formal power series is defined by the Cauchy formula as follows:

$$\left(\sum_{i=0}^{\infty} a_i\right) \cdot \left(\sum_{j=0}^{\infty} b_j\right) = \sum_{k=0}^{\infty} c_k$$

Where the coefficients c_k are given by:

$$c_k = \sum_{l=0}^{k} a_l b_{k-l}$$

The Cauchy formula elegantly translates to APL:

```
mul←{(⍺⍺¨+.×⍵⍵¨∘⍸)⍳1+⍵}
```

5.2 Reciprocals, division and composition

A simple implementation of reciprocals follows from the observation that given two formal power series $P(x)$ and $Q(x)$, $P(x)/Q(x) = 1$ implies that $Q(x) = P(x)^{-1}$. This leads to an implicit system of equations where the product of free terms equals to 1, while the convolution of $n > 0$ unique terms must equal to zero,

[2] A formal power series is conventionally defined over a set more general than $\mathbb{R}$, however, this paper restricts them to $\mathbb{R}$ for practical reasons

arguing for the following recurrence relation for the coefficients of Q - b_n, given the coefficients of P - a_n:

$$b_0 = \frac{1}{a_0},$$
$$b_i = \frac{-1}{a_0} \sum_{j=0}^{i-1} b_j a_{j+1}.$$

The APL translation (assuming $a_0 \neq 0$) follows:

```
recip←{
  0=⍺⍺ 0:⎕SIGNAL 8
  ⍵=0:÷⍺⍺ 0
  z←1+⍳⍵
  (-÷⍺⍺ 0)×+/(⍺⍺¨z)×(⍺⍺ ∇∇)¨⍵-z
}
```

Arguing for the following trivial definition of `div`:

```
div←{(⍺⍺ mul (⍵⍵ recip)) ⍵}
```

For example:

```
  tan←sin div cos
  .5⌊⍎tan¨⍳20
0.546302489792367409317817547223669 6
  3○.5
0.546302489843790513255179465780285 5
```

McIlroy suggests the following definition of composition assuming a non-zero free term[6]:

```
jot←{
  0≠⍵⍵ 0:⎕SIGNAL 8
  ⍵=0:⍺⍺ 0
  ((⍵⍵ 1○+)mul((⍺⍺ 1○+)jot ⍵⍵))⍵-1
}
```

5.3 Calculus

Integration and derivatives are implemented via the well-known power rule:

```
derv←{(⍺⍺ ⍵+1)×⍵+1}
int←{⍺←0⋄⍵=0:⍺⋄(⍺⍺ ⍵-1)÷⍵}
```

The derivation of `derv` follows. Consider $(a_n x^n)'$. By the power rule, this is equivalent to $a_n \cdot n x^{n-1}$. The formula for the coefficient of the n-th term is thus $a_{n+1}(n+1)$. The derivation of `int` follows analogically, however, special care is taken when handling the constant of integration. `int` allows the caller to specify its value (by default assumed to be 0).

This allows for the demonstration of the well-known property $\int \sin x\, dx = -\cos x$:

```
  (¯1○(sin int)¨ ≡ (cos neg)¨)⍳10
1
```

The exponential function is defined per its well-known power series, while the formal power series for $\log(1+x)$ are derived via an integral:

```
exp←{÷!⍵}
log1p←{(¯1○* int) ⍵}
```

5.4 Recursive definitions

The exponential function can be defined in terms its own integral:

```
  my_exp←{1(my_exp int)⍵}
  (exp¨⍳5) ≡ (my_exp¨⍳5)
1
```

This definition makes clever use of the constant of integration: `int` will not evaluate the input function for `⍵=0`, establishing a base case. The `⍵>0` case makes use of inductive reasoning to define the function for `⍵` via prior definitions through `⍳⍵`.

Another example of this technique is the following mutually recursive definition of `sin` and `cos`:

```
my_sin←{(my_cos int) ⍵}
my_cos←{((1○const) sub (my_sin int))⍵}
```

5.5 Potential improvements

The domain of many functions can be extended to handle more common use cases. Highly recursive functions such as `recip` can be sped up significantly via memoisation, for example through `dfns.memo`.

References

[1] A. M. Turing *On Computable Numbers, with an Application to the Entscheidungsproblem.* 1936, J. of Math. 58: 230–265.

[2] M. Escardo *Infinite sets that admit fast exhaustive search.* 22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007), Wroclaw, Poland, 2007, pp. 443-452, doi: 10.1109/LICS.2007.25.

[3] Bridges, D., Richman, F. *Varieties of Constructive Mathematics (London Mathematical Society Lecture Note Series).* 1987, Cambridge University Press, doi:10.1017/CBO9780511565663

[4] J. Berger *Brouwer's fan theorem.* 2019, arXiv

[5] U. Berger. *Totale Objekte und Mengen in der Bereichtheorie.* PhD thesis, Munich LMU, 1990.

[6] M. D. McIlroy *The music of streams.* Information Processing Letters 77 (2001) 189-195.

[7] R. Hui, *Bring Something Beautiful.* Vector Vol. 24, No. 4

Typesetting external program code and its output: hvextern

Herbert Voß

Abstract

When writing a book with a mathematical, scientific or technical background, the output of programs is often inserted as text or an illustration; in many cases, also with complete or partial indication of the respective source code. As an author, you have the problem of keeping such external sample programs in sync with the current manuscript. If you keep the source code in the book manuscript itself, and create the external examples at the same time as typesetting the main document, you can be sure the code and output stay consistent.

1 Introduction

If you use LaTeX to write a book about LaTeX, you can easily insert the output of the examples directly in the main document. [2] This does not necessarily mean that the examples will also work as small individual documents. All examples in a larger book use the main document's preamble, which is not available to a reader of the book.

It usually makes more sense to create examples as separate documents or programs that are independent of the main document. To do this, the complete source code is written from the main document into an external file, which is then processed using a specified program and the result is integrated back into the main document as a PDF, PNG, text, or whatever form is appropriate. From the entire source code of the example, you can use markers to place the essential lines of code in the main document before or next to the example output.

> The output of the following examples was generated "on-the-fly" when compiling this *TUGboat* article. Any change in the example code therefore automatically led to updated output during the next compilation process.

To begin, first we'll show a short example: *TUGboat* is normally compiled with pdfLaTeX, so there are problems with an example that absolutely requires the use of XeLaTeX. The example must be created externally and the output integrated as a PDF. It makes more sense to do this from the main document and include the output directly, as shown here:

美好的一天.

First published in *Die TEXnische Komödie* 2/2022, pp. 30–60. Translation by the author.
This article is from: TUGboat, Volume 43 (2022), No. 3

Herbert Voß

doi.org/10.47397/tb/43-3/tb135voss-extern

This is made possible by the `hvextern` package, which defines only one environment and one command. [5]

The corresponding code for the above inline example is:

Inline example

```
[...] as shown here:
\begin{externalDocument}[
  compiler=xelatex, inline, runs=2, force,
  grfOptions={height=8pt}, crop, cropmargin=0,
  cleanup, docType=latex]{voss}
\documentclass{ctexart}% needs xelatex
\pagestyle{empty}
\begin{document}
```

美好的一天

```
\end{document}
\end{externalDocument}
This is made [...]
```

Any change in this example will automatically be kept in sync — during the next translation process, the output of the main document will be updated, and with it the new code of the example will be run, and thus also the new output will be inserted.

In the example above, only the output was included without showing the source code. Depending on the application, it may be desirable to display portions or all of the source code; this is described on the following pages.

Currently the `hvextern` package supports external documents for METAPOST, TeX, ConTeXt, LaTeX, LuaTeX, LuaLaTeX, XeTeX, XeLaTeX, Lua, Perl, Java, Python, and shell scripts.

2 Syntax

The package, which has no special options, is loaded as usual: `\usepackage{hvextern}`. The package defines only one environment, `{externalDocument}`, and one command, `\runExtCmd`:

Syntax

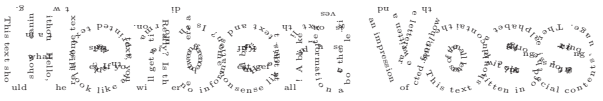
```
\begin{externalDocument}[⟨options⟩]
        {⟨output filename without extension⟩}
... source code ...
\end{externalDocument}

\runExtCmd[⟨options⟩]
    {⟨command⟩}
    {⟨output filename without extension⟩}
```

The main document must be compiled with the `-shell-escape` option (or with two dashes, as usual), otherwise no external commands will be run and thus the correct output will not be shown.

lualatex invocation

```
lualatex --shell-escape ⟨latexfile⟩
```

Let's show another example: The following code for character manipulation must be compiled using the program sequence `latex→dvips→ps2pdf`, because it does not work with other TEX engines. With the environment `externalDocument`, however, you can write the complete code in an external file, specify the necessary process, and embed the result as a PDF. The only important thing is that when creating the graphic, the standard output of the page number is suppressed and any white space is cut off using the crop option. Of course, this does not apply if a complete page is to be included (see page 20).



The code of the above example looks like:

dvips example

```
\begin{externalDocument}[compiler=latex,crop,
  force=false, cleanup={log,aux,ps,dvi},
  grfOptions={width=\linewidth}]{voss}
\documentclass{article}
\usepackage[american]{babel}
\pagestyle{empty}
\usepackage{pst-text,blindtext}
\begin{document}
\DeclareFixedFont{\SF}{T1}{phv}{b}{n}{2cm}
\pstextpath(0,-1ex){\pscharpath*[
  linestyle=none]{\SF Herbert Voss}}{%
\tiny \blindtext}
\end{document}
\end{externalDocument}
```

The meaning of each option:

`compiler=latex` Use LaTeX to compile. The rest of the invocation, including other programs, is determined by the internal definition of `\hv@extern@runLATEX`.

`crop` Crop the whitespace with `pdfcrop`.

`force` Recreate the output even if it already exists.

`cleanup={log,aux,ps,dvi}` Delete the specified auxiliary files at the end.

`grfOptions={width=\linewidth}` Scale output to the current linewidth.

`voss` Filename that is extended internally by a consecutive number.

The external filename, extended by a consecutive number, can be printed into the margin by setting the keyword `showFilename`. In general it is printed in the outer margin, or in twocolumn mode in the outer column. If the example is set in twocolumn mode but inside a starred floating environment over both columns, then use the keyword `outerFN` (see Figure 1). Then `hvextern` doesn't test for twocolumn mode.

A vertical shift of the filename is possible by specifying a length for `shiftFN`, e.g., `shiftFN=5ex`.

Essentially, it doesn't matter which programming language is used, as long as minimum communication between the main document and the external program is guaranteed: this consists only of the requirement that the external document must provide its output with the same file name with which it was called. However, even this can be a problem in some programming languages, as shown below with some examples.

By default, source code and output are displayed one above the other, so that a page break in the source code is not a problem. The following example creates and runs a Python program, and then includes the output as a PNG format file. The header of the `externalDocument` environment is:

Options for Python

```
\begin{externalDocument}[compiler=python3,
  code, ext=py, docType=py, usefancyvrb,
  grfOptions={width=\linewidth}]{voss}
... Python code ...
\end{externalDocument}
```

It is only in rare cases that you will want to output the complete source code. Therefore, areas can be defined using so-called markers, which then delimit the output. The markers are written to the external file as normal comments, the only reason why they are programming language dependent; the comment character is not uniform. For Python the markers are:

Python marker lines

```
\hv@extern@ExampleType{py}
  {\NumChar StartVisibleMain}
  {\NumChar StopVisibleMain}
  {\NumChar StartVisiblePreamble}
  {\NumChar StopVisiblePreamble}
```

and for plain TEX:

TEX marker lines

```
\hv@extern@ExampleType{tex}
  {\perCent StartBody}
  {\string\bye}
  {\perCent StartVisiblePreamble}
  {\perCent StopVisiblePreamble}
```

`\perCent` and `\NumChar` are the TEX and Python comment characters % and #, which must be escaped for LaTeX. Internally, the category of the character is changed so that it is available as a normal character using the `\perCent` or `\NumChar` command.

After calling the Python program, it must be ensured that the file name is determined in order to provide the output with the same main file name and a different file extension. In Python this is possible with the following code:

Python: get filename

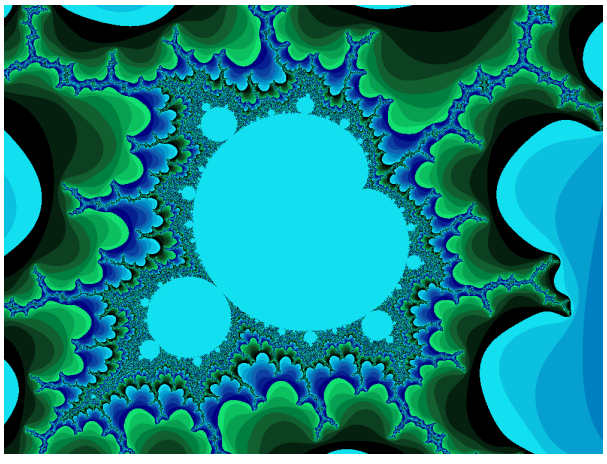
```
fileName = os.path.basename(
   os.path.splitext(__file__)[0])
```

Depending on the output format, this file name is extended by `.pdf`, `.png`, or `.txt`, so that the output can be easily inserted into the LaTeX main document. In addition, the markers are now used so that the output of parts of the Python source code can be done, requested with the `code` keyword. (Output grayscaled for printed *TUGboat*.)

voss-3.py

```
from PIL import Image
import subprocess
# drawing area (xa < xb and ya < yb)
xa = -0.1716
xb = -0.1433
ya = 1.022
yb = 1.044
maxIt = 1024  # iterations
imgx = 1000   # image size
imgy = 750
image = Image.new("RGB", (imgx, imgy))
```

```
for y in range(imgy):
    cy = y * (yb - ya) / (imgy - 1)  + ya
    for x in range(imgx):
        cx = x * (xb - xa) / (imgx - 1) + xa
        c = complex(cx, cy)
        z = 0
        for i in range(maxIt):
            if abs(z) > 2.0: break
            z = z * z + c
        r = i % 4 * 6
        g = i % 8 * 32
        b = i % 16 * 16
        image.putpixel((x, y), b*65536 + g*256 + r)
```



In a purely formal way, the output of the source code can be defined by analogy to LaTeX as a preamble (general definitions) and program body (application), whereby two slightly different background colors are used for differentiation. The markers can be used anywhere in the document. The above example was created with the following LaTeX code:

Complete example code

```
\begin{externalDocument}[compiler=python3, force=false,
  %showFilename,
  runs=1, code, ext=py, docType=py,
  usefancyvrb, grfOptions={width=\linewidth}]{python}
import os
#StartVisiblePreamble
from PIL import Image
import subprocess
# drawing area (xa < xb and ya < yb)
xa = -0.1716; xb = -0.1433
ya = 1.022;   yb = 1.044
maxIt = 1024  # iterations
imgx = 1000   # image size
imgy = 750
image = Image.new("RGB", (imgx, imgy))
#StopVisiblePreamble

#StartVisibleMain
for y in range(imgy):
    cy = y * (yb - ya) / (imgy - 1)  + ya
    for x in range(imgx):
        cx = x * (xb - xa) / (imgx - 1) + xa
        c = complex(cx, cy)
        z = 0
        for i in range(maxIt):
            if abs(z) > 2.0: break
            z = z * z + c
        r = i % 4 * 6
        g = i % 8 * 32
        b = i % 16 * 16
        image.putpixel((x, y), b*65536 + g*256 + r)
#StopVisibleMain
# now get the filename created by the latex
imageName = os.path.basename(
  os.path.splitext(__file__)[0])
image.save(imageName+".png", "PNG")
\end{externalDocument}
```

By specifying a width for the output of the source code, code and result can be arranged side by side, as shown in Figure 1.

3 Using markers in the source code

The markers identify the areas of the source code that are to be output in the (LaTeX) main document. For an external document with TeX or LaTeX code, the use of the markers are shown in the following examples:

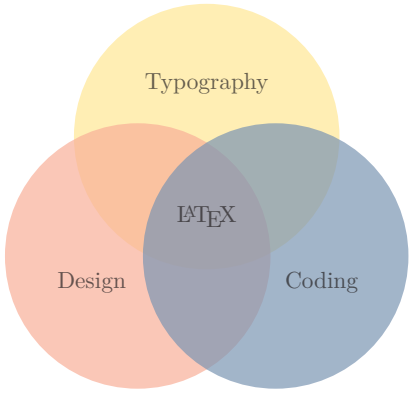
LaTeX marker lines

```
[...]
%StartVisiblePreamble
[... listed preamble code ...]
%StopVisiblePreamble
[...]
\begin{document}
[... listed body code ...]
\end{document}
```

Everything between the `%StartVisiblePreamble` and `%StopVisiblePreamble` lines is printed with the background color `BGpreamble` (default `black!12`). All of the lines between `\begin{document}` and `\end{document}`, on the other hand, are considered as the text body and printed with the background color `BGbody` (default `black!8`).

```
\usepackage{tikz}
\usepackage[hks,pantone,xcolor]{xespotcolor}
```

```
\SetPageColorSpace{HKS}
\definecolor{HYellow}{spotcolor}{HKS05N,0.5}
\definecolor{HRed}{spotcolor}{HKS14N,0.5}
\definecolor{HBlue}{spotcolor}{HKS38N,0.5}
\begin{tikzpicture}[fill opacity=0.7]
 \fill[HYellow]( 90:1.2) circle (2);
 \fill[HRed]   (210:1.2) circle (2);
 \fill[HBlue]  (330:1.2) circle (2);
 \node at ( 90:2)  {Typography};
 \node at ( 210:2) {Design};
 \node at ( 330:2) {Coding};
 \node {\LaTeX};
\end{tikzpicture}
```



voss-4.tex

Figure 1: Example for side-by-side code and output inside a `figure*` environment in twocolumn mode.

For TEX we use:

TEX marker lines

```
[...]
%StartVisiblePreamble
[... listed preamble code ...]
%StopVisiblePreamble
[...]
%StartBody
[...]
\bye
```

Now everything between `%StartBody` and `\bye` is the printed text body.

The markers are defined by the internal macro `\hv@extern@ExampleType`. This macro expects five parameters, for example for Java:

Java marker lines

```
\hv@extern@ExampleType{java}
  {//StartVisibleMain}
  {//StopVisibleMain}
  {//StartVisiblePreamble}
  {//StopVisiblePreamble}
```

The comment starter for Java is `//`, for Lua `--`, and for Perl `#`. The latter must be escaped by using `\NumChar`, as already shown in an example above. In general, the option `docType` defines the type of the source code (the comment starter), and it must have one of these values:

```
context java latex lua mp pl py tex sh
```

As you can see, only `tex` is allowed for `docType`, not `latex`, `pdflatex`, etc. This is because the comment starter is uniformly `%` for all TEX variants.

The `compiler` option defines the base program to be run, and the entire invocation, which may involve additional programs. The following `compiler` values are currently supported:

```
context java latex lua lualatex luatex
mpost pdflatex perl python3 sh tex texlua
xelatex xetex
```

The configurations for Lua, Perl, Java, shell, and Python all have the same structure; they only differ in the comment character to be used. For Lua, we have

Lua marker lines

```
\hv@extern@ExampleType{lua}
  {--StartVisibleMain}
  {--StopVisibleMain}
  {--StartVisiblePreamble}
  {--StopVisiblePreamble}
```

Sometimes, both `docType` and `compiler` are the same, for example when using Lua: `docType=lua` and `compiler=lua`. Indeed, for Lua files that also have the `.lua` extension, the value `lua` must be assigned three times:

Options for Lua code

```
ext=lua, compiler=lua, docType=lua,
```

The following Lua example writes plain text to standard output, so we pass the `redirect` option to the `externalDocument` environment; the output is then redirected into a file of the same main name but with the extension `.txt`. This is read verbatim from within the main LaTeX document and can therefore contain any characters.

```
io.write("1. "..type("Hello world").."  ")
print("2. "..type(10.4*3))
io.write("3. "..type(print).."  ")
io.write("4. "..type(type).."  ")
print("5. "..type(true))
io.write("6. "..type(nil).."  ")
print("7. "..type(type(X)))
```

voss-5.lua

```
io.write("8. "..type(a).."  ")

a = 10
io.write("9. "..type(a).."  ")
a = "a string!!"
io.write("10. "..type(a).."  ")
a = print
print("11. "..type(a))
```

```
1. string
2. number
3. function
4. function
5. boolean
6. nil
7. string
8. nil
9. number
10. string
11. function
```

The same applies to the following example with Java code:

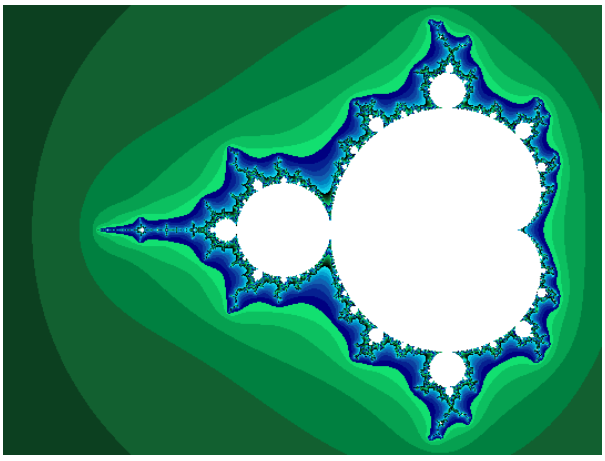
Options for Java code

```
ext=java, compiler=java, docType=java,
```

voss-6.java

```
  public static int iterZahl(
    final double cx,
    final double cy,
    int maxIt,
    final double radius){
  // count the number of iterations
    int zaehler = 0;
    double zx = 0.0, zy = 0.0, tmp;
    do {
      tmp = zx*zx - zy*zy + cx;
      zy = 2*zx*zy + cy;
      zx = tmp;
      zaehler++;
  // run as long as the kength of the vector
  // is smaller than the radius
    } while (zx*zx+zy*zy<=radius && zaehler<maxIt);
    return zaehler;
  }
```

```
    double xa = -2.5, xe = 0.7, ya = -1.2, ye = 1.2;
    double dx = (xe-xa)/(imageWidth-1),
           dy = (ye-ya)/(imageHeight-1);
    double cx, cy; int R, G, B;
    double radius = 10.0;  int maxIt = 1024;
    cx = xa;
    for (int sp = 0; sp < imageWidth; sp++) {
      // from top to bottom:
      cy = ye;
      for (int ze = 0; ze < imageHeight; ze++) {
        int zIter = iterZahl(cx,cy,maxIt,radius);
        if (zIter == maxIt) {
          g.setColor(Color.WHITE);
          g.drawLine(sp, ze, sp, ze);
        } else {
          R = zIter % 4 * 6 ;
          G = zIter % 8 * 32;
          B = zIter % 16 * 16;
          g.setColor(new Color(R,G,B));
          g.drawLine(sp, ze, sp, ze);
        }
        cy = cy - dy;
      } // for ze
      cx = cx + dx;
    } // for sp
```



4 Options

4.1 Program(s) and number of runs

In general, any selected `compiler` program should be found in your search path, with `pdflatex` being the default. However, in rare cases it may be necessary to specify a path for the program, which is done by assigning to `progpath`. A `/` must appear at the end, for example '`progpath=./bin/`'.

Here is the code defining the options `progpath`, `compiler`, `runs`, and `runsequence`. The full list of `compiler` values was given on previous page. (The definitions are omitted.)

Compiler options

```
\define@key{hv}{progpath}[]{...}
\define@choicekey*+{hv}{compiler}[\val\nr]{%
    context,...,xetex}
    [pdflatex]{...}
\define@key{hv}{runs}[1]{...}
\define@key{hv}{runsequence}[]{...}
```

Instead of using `compiler`, `biber` and `xindex`, an explicit run sequence can also be specified via the `runsequence` parameter. A comma-separated list is expected. The input filename is added to each program being run. For example, this sequence generates the bibliography and (with additional options) index, besides the main document:

Invocation (runsequence) example

```
runsequence={lualatex,biber,xindex -l de -c DIN2,
  makeglossaries,lualatex,lualatex},
cleanup={log, aux, toc, bbl, blg,
  run.xml, bcf, idx, ilg},
pages={1,2,3,4,5,6,7,8,9},
```

The example also prints pages 1–9 of the created external document, which also has a glossary and a list of symbols and acronyms.

voss-7.tex

```
\documentclass[paper=a5,parskip=half-,DIV=12,
     bibliography=totoc,
     listof=totoc,fontsize=12pt]{scrreprt}
\usepackage[ngerman]{babel}
\usepackage{libertinus-otf,hvindex}
\usepackage{biblatex,makeidx}\makeindex
\addbibresource{biblatex-examples.bib}
\usepackage[abbreviations,symbols,postdot,
    stylemods,style=index]{glossaries-extra}
\makeglossaries
\title{Umlaute} \author{Friedrich Schiller}
```

```
\maketitle \tableofcontents
\chapter{Introduction}   \section{Words}
\Index{Österreich} \Index{Öresund}
\Index{Ober} \Index{Ostern} \Index{Oberin}
\Index{Österreich} \Index{Öresund}
\Index{Ödem} \Index{Oligarch} \Index{Oder}
\Index{Goldmann}  \Blindtext[3]
\section{Glossary}
First use: \gls{cafe}, \gls{html}, \gls{pi}.
Next use: \gls{cafe}, \gls{html}, \gls{pi}.
\printindex    \printglossaries
\nocite{*}\raggedright\printbibliography
```

4.2 Graphics options

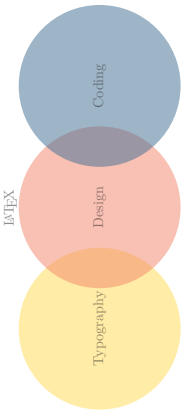
Graphics options

```
\define@key{hv}{grfOptions}[]{...}
```

The value for `grfOptions` is passed to the well-known `\includegraphics` macro, for example `{angle=90, width=\linewidth}`, as shown in the following example.

```
\usepackage{tikz}
% needs xelatex:
\usepackage[hks,pantone,
  xcolor]{xespotcolor}
```

```
\SetPageColorSpace{HKS}
\definecolor{HYellow}{spotcolor}
  {HKS05N,0.8}
\definecolor{HRed}{spotcolor}
  {HKS14N,0.8}
\definecolor{HBlue}{spotcolor}
  {HKS38N,0.8}
\begin{tikzpicture}
  [fill opacity=0.5]
\fill[HYellow](0,0)circle(2);
\fill[HRed]   (3,0)circle(2);
\fill[HBlue]  (6,0)circle(2);
 \node at (0,0){Typography};
 \node at (3,0){Design};
 \node at (6,0){Coding};
 \node at (3,2.2){\LaTeX};
\end{tikzpicture}
```



voss-8.tex

Since source code and output are arranged next to each other here, the specification `\linewidth` refers to the current width of the minipage. Thus the output has the maximum possible size.

4.3 Listing options

Listings options

```
\define@key{hv}{lstOptions}[]{...}
```

The value assigned is passed to either the macro `\lstinputlisting` or, if `usefancyvrb` is specified, to the macro `\VerbatimInput` from the `fancyvrb` package. It should be noted that the options for the respective packages have different meanings and names, so it is not so easy to switch between `listings` and `fancyvrb`.

The following example uses the `listings` package, which is the default and therefore does not require any parameter setting. A slightly exotic list of options is given, and we omit the graphical output (which is seen in the next example), purely for the demonstration:

Listings options example

```
lstOptions={basicstyle=\sffamily\slshape\scriptsize,
            columns=fullflexible,showoutput=false},
```

voss-9.tex

```
\documentclass[landscape]{article}
\usepackage[margin=1cm]{geometry}
\usepackage{pst-calendar}
```

```
\psscalebox{0.5}{%
  \psCalDodecaeder[
    Year=2022,style=june]%
}
\hspace{4cm}
\psscalebox{0.5}{%
  \psCalDodecaeder[
    Year=2022,style=july]%
}
```

4.4 Background color for the code

Different colors for the background and the frame can be selected. They can be modified via the following parameters, which show the defaults in brackets. (The actual definitions are omitted.) `BG` is the abbreviation for "background" and `BO` for "border":

Color options

```
\define@key{hv}{BGpreamble}[black12]{...}
\define@key{hv}{BGbody}[black8]{...}
\define@key{hv}{BOpreamble}[black12]{...}
\define@key{hv}{BObody}[black8]{...}
```

The parameter values are passed to a `tcolorbox` environment (of the package with the same name), and evaluated there. [3] Because the background and frame have the same color, the frame remains "invisible" by default. This changes with different values, for example:

Differing frame and background colors

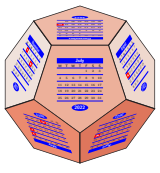
```
  BGpreamble=red!10, BOpreamble=red,
  BGbody=blue!8, BObody=blue,
```

Typically, you should use subtle colors so that the output does not fade into the background compared to the code.

voss-10.tex

```
\usepackage{pst-calendar}
```

```
\psscalebox{0.3}{%
  \psCalDodecaeder[
    Year=2022,style=july]%
}
```



We'll return to the default gray colors now.

4.5 Type of source code

The current version of `hvextern` supports source code in METAPOST, plain TEX, LATEX, ConTEXt, Python, Lua, shell, and Perl. Each language's definition contains the source code markers already mentioned, and the program invocation sequence if special treatment is necessary. For example, source code in LATEX requires special treatment if the program used is `latex`; the corresponding definition contains the following:

Marker and run setting for dvips

```
\hv@extern@ExampleType{latex}
  % for _all_ LaTeX engines
  {\string\begin\string{document\string}}
  {\string\end\string{document\string}}
  {\perCent StartVisiblePreamble}
  {\perCent StopVisiblePreamble}

% only for the sequence latex->dvips->ps2pdf
\def\hv@extern@runLATEX#1#2#3#4{%
  %path/compiler/file/extension
  \ShellEscape{#1#2\space #3#4}%
  \ShellEscape{#1dvips\space #3.dvi}%
  \ShellEscape{#1ps2pdf\space
     -dAutoRotatePages=/None\space
     -dALLOWPSTRANSPARENCY\space#3.ps}%
}
```

The macro must have the following structure:

Macro implementing the run sequence

```
\def\hv@extern@run<NAME>#1#2#3#4{%
  %path/compiler/file/extension
  ...}
```

The definition for TEX is similar. The type of source code and the program used can be different for TEX, LATEX and ConTEXt, for example type `latex` but program `lualatex`.

4.6 Output of one or more full pages

In the event that only a subset of pages are to be output, this can be controlled via the `pages` parameter, as we saw in a previous example. It expects a comma-separated list of the pages to be printed. The individual pages can be framed with the `frame` parameter in order to achieve a clearer presentation.

Output page selection

```
\define@key{hv}{pages}[1]{...}
\define@key{hv}{pagesep}[1em]{%
     \hv@extern@pagesep=#1}
\define@boolkey{hv}[hv@extern@]{frame}[true]{}
```

It is up to the user to use the `grfOptions` parameter to ensure that the pages for the output are scaled as needed. This example outputs the first three pages of a document:

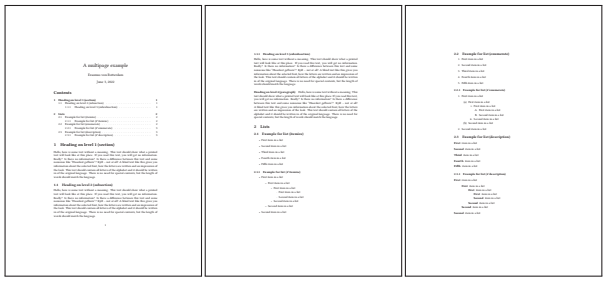
Page selection example

```
pages={1,2,3}, grfOptions={width=0.3\linewidth},
pagesep=1pt,
frame, compiler=lualatex, runs=2, % for the TOC
```

voss-11.tex

```
\usepackage[american]{babel}
\usepackage{libertinus}
\usepackage{blindtext}
```

```
\title{A multipage example}
\author{Erasmus von Rotterdam}
\maketitle
\tableofcontents
\blinddocument
```



4.7 Output as a float

As a rule, the output is in the running text, which can be undesirable if the text width is relatively small. Larger free spaces can then arise on one side, which is always unfavorable. In such cases you should use the `float` option, in which case, as usual, a caption can be specified using the `caption` parameter and a cross-reference label can be specified u sing `l abel`. The floating type is by default `f igure` and the placement can be set by the optional argument `floatsetting`. It is preset to `!htb`.

voss-12.tex

```
\usepackage{pst-coxeterp}
```

```
\begin{pspicture}(-1,-1)(1,1)
  \Simplex[dimension=2]\end{pspicture}
\begin{pspicture}(-1,-1)(1,1)
  \Simplex[dimension=3]\end{pspicture}
\begin{pspicture}(-1,-1)(1,1)
  \Simplex[dimension=5]\end{pspicture}
\begin{pspicture}(-1,-1)(1,1)
  \Simplex[dimension=7]\end{pspicture}
```

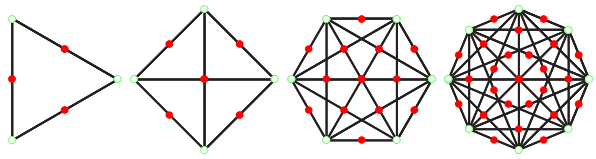


Figure 2: An example for Coxeter images.

Float options

```
\define@boolkey{hv}[hv@extern@]{float}[true]{}
\define@key{hv}{caption}[]{...}
\define@key{hv}{label}[]{...}
```

Figure 2 shows an example as a floating object. It was created with the following parameters:

Float example

```
[...]
 float,
 caption={An example for Coxeter images.},
 label=img:cox,
[...]
```

The specification `float` refers only to the output; otherwise, a previous listing could not have a page break and the typesetting of the text would be more difficult. On the other hand, it may well be that other text appears between the code and the output of an example. Then manual intervention is necessary, for example by using the command `\captionof` from the package `caption`, which allows a caption without floating space.

5 Cropping white space

When displaying the output of examples, usually only the area that contains a graphic or text is of interest, and we want to remove any surrounding white space. For documents that consist of only one page, the document class `standalone` can generally be used, which automatically removes all white space. If you have more than one page or want to use another special document class, `hvextern` provides the `crop` option:

Crop options

```
\define@boolkey{hv}[hv@extern@]{crop}[true]{}
\define@key{hv}{cropmargin}[2]{...}% in pt
```

`crop` can also be applied to documents with multiple pages. In this case, however, you should make sure that the pages have headers and footers so that the white space that is cut off always has the same size. Otherwise the pages end up with different heights, as shown in the example below, which is usually undesirable. Among other things, the following parameters were set:

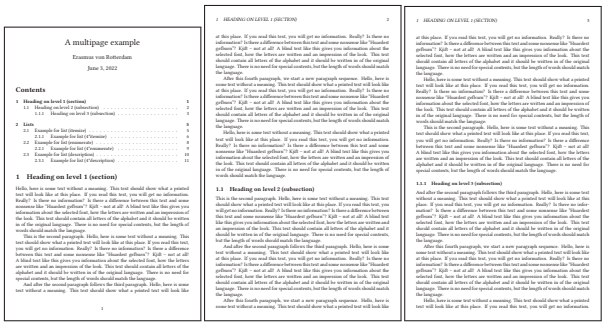
Crop example

```
pages={1,2,3}, grfOptions={width=0.3\linewidth},
frames, pagesep=1pt, crop, cropmargin=5,%is 5pt
compiler=lualatex, runs=2, % for the TOC
```

voss-13.tex

```
\usepackage[american]{babel}
\usepackage{libertinus}
\usepackage{blindtext}
\pagestyle{headings}
```

```
\title{A multipage example}
\author{Erasmus von Rotterdam}
\maketitle
\tableofcontents
\Blinddocument
```



5.1 Source code and output side by side

Normally the source code is printed first and then the output. This order cannot be changed with the current version of `hvextern`. For side-by-side output, the `mpwidth` parameter determines the width of a left minipage and is always evaluated if it is greater than 0 pt. A second minipage is then reserved for output for the remainder of the line, except for the value of `mpsep`. Both minipages are aligned to the value of `mpvalign`, the top edge by default.

Side-by-side options
```
\define@key{hv}{mpwidth}[0pt]{...}
\define@key{hv}{mpsep}[1em]{...}
```

The default distance between the two minipages is 1 em, as shown.

5.2 Horizontal alignment of the output

The code is always left-aligned, whereas the output can use various known alignments via the `align` option. The use of the `ragged2e` package does not have any advantages here.

Horizontal alignment
```
\define@key{hv}{align}[\centering]{...}
```

The default with `align=\centering`:

```
\rule{0.5\linewidth}{3mm}
```

Left-justified with `align=\raggedright`:

```
\rule{0.5\linewidth}{3mm}
```

Right-justified with `align=\raggedleft`:

```
\rule{0.5\linewidth}{3mm}
```

Side by side, default with `align=\centering`:

```
\rule{0.2\linewidth}{3mm}
```

Side by side, with `align=\raggedright`:

```
\rule{0.2\linewidth}{3mm}
```

Side by side, with `align=\raggedleft`:

```
\rule{0.2\linewidth}{3mm}
```

5.3 Inline output, rather than displayed

The output can be printed within a line in the so-called inline mode. This can make sense if you don't have certain characters available in your document's font, but they can be generated externally and then input as PDF. Here, the corresponding source code should not be shown, so `code=false` is automatically set with `inline`.

Inline option
```
\define@boolkey{hv}[hv@extern@]{inline}[true]{...}
```

An example has already been shown on page 280.

5.4 Handling plain text output

For LaTeX documents, the output is generally PDF, but when using a programming language such as Perl, the output would more typically be plain text. This must be redirected or written to a file so that it can be inserted verbatim into the main document. This can be controlled with the parameters `includegraphic` and `redirect`. The output is typeset with `listings` or `fancyvrb`, and options for the typesetting environment set with `textoptions`.

With `includegraphic=false` it is up to the user to ensure that every output within the external document is written to a text file; `hvextern` looks for a file with the right name. This is done automatically with `redirect`, but then all program output ends up in the external text file.

Plain text output options
```
\define@boolkey{hv}[hv@extern@]{redirect}[true]{}
\define@boolkey{hv}{includegraphic}[true]{}
\define@key{hv}{textOptions}[]{...}
```

The text file must have the same main file name as the external file, but with the extension `.txt`. As we've seen, each program can determine by itself what name it was called with, so it is easily possible to determine the correct name for the text output file. For a Perl program, this could be achieved with the following code, for example:

Perl: get filename
```
my $filename = $0; # the current filename
$filename =~ s/\.pl//; # without extension .pl
$filename = "${filename}.txt"; # for the output
open(my $fh, '>', $filename);
```

However, in the next example, the optional keyword `redirect` is given, so determining the filename in the code is not needed. The example is set with:

Example output redirect
```
compiler=perl, includegraphic=false, docType=pl,
ext=pl, usefancyvrb, runs=1, code, redirect,
tcbox=false, force, lstOptions={fontsize=\small,
  fontfamily=tt, frame=lines}
```

```
my $number = 1;
my $start = 1;
my $end    = 9;
my $found = 0;
```

```
print "Searching for Kaprekar constants\n";
while ($number < 8) {
 print "${number}-digits: ";
 foreach ($start...$end){
  @chars = split(//,$_);
  $Min = join("",sort(@chars));
  $Max = reverse($Min);
  $Dif=$Max-$Min;
  if($_ eq $Dif) {
```

voss-20.pl

```
   $found = 1;
   print $_,",  ";
  }
 }
 if (!$found) { print "---\n"; }
 else         { print "\n"; }
 $found = 0;
 $number++;
 $start = $start*10;
 $end   = $end*10;
}
```

```
Searching for Kaprekar constants
1-digits: ---
2-digits: ---
3-digits: 495,
4-digits: 6174,
5-digits: ---
6-digits: 549945,  631764,
7-digits: ---
```

(Just for the sake of completeness: A Kaprekar constant is a number A with $\max(A) - \min(A) = A$, where max and min are the sorted/reverse-sorted digits of the number, e.g., $A = 495 = 954 - 459$.)

Our next example is in Lua, and also produces text output; but instead of using `redirect`, the code outputs to the appropriate file. This filename can be determined as follows:

Lua: get filename

```
-- get full filename
local filename = arg[0]
-- delete extension
local shortFN = str:match("(.+)%..+")
-- open external file
outFile = io.open(shortFN..".txt","w+")
```

```
function nextrow(t)
  -- fill table
  local ret = {}
  t[0], t[#t+1] = 0, 0
  for i = 1, #t do
    ret[i]=t[i-1]+t[i]
  end
  return ret
end

function triangle(n)
 t = {1}
 for i = 1, n do
  m = (n - i)
  for j = 1,m do
    outFile:write("  ")
  end
  for k = 1,i do
   outFile:write(
    string.format("%4s",t[k]))
  end
  outFile:write("\n")
  t = nextrow(t)
 end
end
```

voss-21.lua

```
triangle(10)
```

```
                   1
                 1   1
               1   2   1
             1   3   3   1
           1   4   6   4   1
         1   5  10  10   5   1
       1   6  15  20  15   6   1
     1   7  21  35  35  21   7   1
   1   8  28  56  70  56  28   8   1
 1   9  36  84 126 126  84  36   9   1
```

5.5 Generating the bibliography and index

The current version of `hvextern` has predefined support for constructing the bibliography with Biber and an index with `xindex`, via the following parameters:

Index and bibliography options

```
\define@boolkey{hv}[hv@extern@]{biber}[true]{}
\define@boolkey{hv}{xindex}[true]{}
\define@key{hv}{xindexOptions}[]{...}
```

In principle, the external run of Biber does not require any further parameters, whereas the `xindex` program requires information about the language, the style file, etc., for example. The next example is generated with the following parameters:

xindex example

```
\begin{externalDocument}[
compiler=lualatex,runs=2,pages=2,crop,
xindex,xindexOptions={-l DE --config DIN2},
docType=latex,cleanup={log,aux,ilg,idx},...]
```

To use other bibliography or index programs, you can use the `runsequence` option; see the example on p. 284.

```
\usepackage{makeidx}
\makeindex
\usepackage{hvindex}
```

voss-22.tex

```
\Index{Österreich} \Index{Öresund}
\Index{Ödipus} \Index{Öchsle}
\Index{Ostern} \Index{Ober} \Index{Oberin}
\Index{Österreich} \Index{Öresund}
\Index{Ödem} \Index{Oligarch} \Index{Oder}
\Index{Ostern} \Index{Ober} \Index{Oberin}
\Index{Obstler} \Index{Öl} \Index{ölen}
\Index{Oder|seealso{Fluss}} \Index{Göbel}
\Index{oder} \index{Fluss!Oder}
\Index{Goethe} \Index{Göthe} \Index{Götz}
\Index{Goldmann}
\printindex
```

Index

6 Verbatim modes: `listings` and `fancyvrb`

6.1 Using `listings`

By default the command `\lstinputlisting` from the package `listings` is used for printing the source code. We saw an example of setting `hvextern`'s `lstOptions` option for it earlier.

6.2 Package `fancyvrb`

There are no fundamental objections to the `listings` package, but sometimes it makes more sense to use `\VerbatimInput` from the `fancyvrb` package, especially when displaying non-ASCII Unicode characters. Most of the examples in this article use `fancyvrb`, by passing the `usefancyvrb` option.

6.3 Vertical space

Vertical space can be controlled by four keywords for the stretchable vertical space:

`aboveskip` The vertical space before the environment `externalDocument` or the command `\runExtCmd` (default `\medskipamount`).

`belowpreambleskip` The vertical space between the preamble and body (default `\smallskipamount`). If the preamble is missing, then there will be only `aboveskip`.

`belowbodyskip` The vertical space between body and output (default `\smallskipamount`).

`belowskip` The vertical space after the environment `externalDocument` or the command `\runExtCmd` (default `\medskipamount`).

7 Supported METAPOST and TEX engines

Here we show a few examples using the common TEX-world programs. (LATEX is omitted here since most of the examples throughout this article use LATEX.)

7.1 METAPOST example

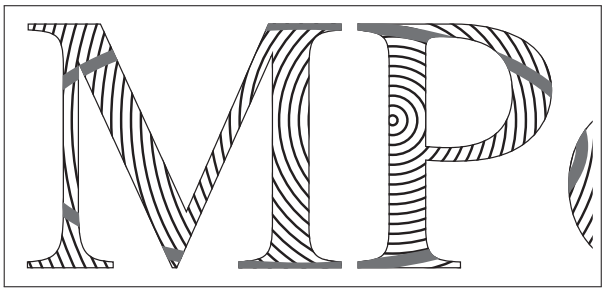
MetaPost example

```
\begin{externalDocument}[
  compiler=mpost,docType=mp,...]
```

```
defaultfont:="ptmr8r";
warningcheck:=0;
```

```
draw fullcircle shifted (0.5,0.6) xscaled 8cm
  yscaled 3.5cm withpen pencircle scaled 5bp
  withcolor 0.33; % gray bands
special("/Times-Roman findfont 150 "
& " scalefont setfont "
& " 0 10 moveto (MPost) false charpath 2 "
& " clip stroke gsave 150 70 translate "
& " 2 4 600 {dup 0 moveto 0 exch 0 exch"
& " 0 360 arc stroke} for grestore ");
```

voss-23.mp



Here is the definition of the command sequence for running METAPOST, just in case you want to modify something:

MetaPost run sequence

```
\def\hv@extern@runMP#1#2#3#4{%
  % path / compiler / file / extension
  \ShellEscape{#1#2\space -tex=tex\space #3#4}%
  \ShellEscape{#1tex\space "\string\input\space
    epsf\string\relax\string\nopagenumbers
    \string\epsfbox{#3.1}\string\bye"}%
  \ShellEscape{#1dvips\space -j\space -E\space
    -o\space #3.eps\space epsf.dvi}%
  \ShellEscape{#1epstopdf\space #3.eps}%
}
```

7.2 Plain TEX example

Plain TEX example

```
\begin{externalDocument}[
  compiler=tex,docType=tex,...]
```

voss-24.tex

```
\footline={\footsc the electronic journal
 of combinatorics {\footbf 16} (2009),
 \#R00\hfil\footrm\folio}
```

```
\font\bigrm=cmr12 at 14pt
\centerline{\bigrm An elementary proof
  of the reconstruction conjecture}

\bigskip\bigskip
\centerline{D. Remifa\footnote*{Thanks to the
  editors of this journal!}}
\smallskip
\centerline{Department of Inconsequential Studies}
\centerline{Solatido College, North Kentucky, USA}
\centerline{\tt remifa@dis.solatido.edu}
\bigskip
\centerline{\footrm
Submitted: Jan 1, 2009; Accepted: Jan 2, 2009;
  Published: Jan 3, 2009}
\centerline{\footrm Mathematics Subject
  Classifications: 05C88, 05C89}
\bigskip\bigskip
\centerline{\bf Abstract}
\smallskip
{\narrower\noindent
The reconstruction conjecture states that the
multiset of unlabeled vertex-deleted subgraphs
of a graph determines the graph, provided it
has at least 3 vertices.  A version of the problem
was first stated by Stanis\l aw Ulam. In this
paper, we show that the conjecture can be proved
by elementary methods.  It is only necessary to
integrate the Lenkle potential of the Broglington
manifold over the quantum supervacillatory measure
in order to reduce the set of possible
counterexamples to a small number (less than a
trillion).  A simple computer program that
implements Pipletti's classification theorem for
torsion-free Aramaic groups with symplectic socles
can then finish the remaining cases.}

\bigskip
\beginsection 1. Introduction.

This is the start of the introduction.
```


An elementary proof of the reconstruction conjecture

D. Remifa*
Department of Inconsequential Studies
Solatido College, North Kentucky, USA
remifa@dis.solatido.edu

Submitted: Jan 1, 2009; Accepted: Jan 2, 2009; Published: Jan 3, 2009
Mathematics Subject Classifications: 05C88, 05C89

Abstract

The reconstruction conjecture states that the multiset of unlabeled vertex-deleted subgraphs of a graph determines the graph, provided it has at least 3 vertices. A version of the problem was first stated by Stanislaw Ulam. In this paper, we show that the conjecture can be proved by elementary methods. It is only necessary to integrate the Lenkle potential of the Broglington manifold over the quantum supervacillatory measure in order to reduce the set of possible counterexamples to a small number (less than a trillion). A simple computer program that implements Pipletti's classification theorem for torsion-free Aramaic groups with symplectic socles can then finish the remaining cases.

1. Introduction.
This is the start of the introduction.

* Thanks to the editors of this journal!

THE ELECTRONIC JOURNAL OF COMBINATORICS 16 (2009), #R00 1

7.3 ConTEXt example (mkIV)

This example is run with ConTEXt from TEX Live 2022, but it should also work with the new LMTX.

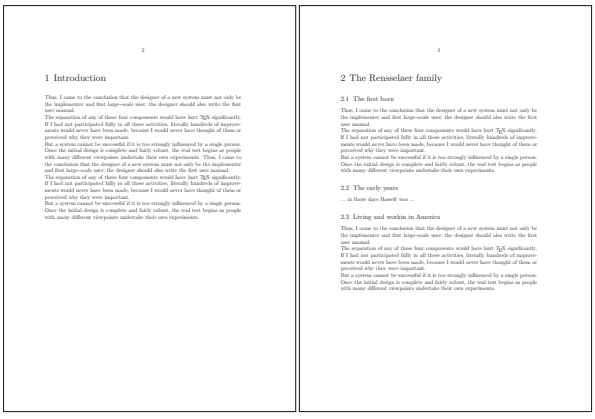
ConTEXt example

```
\begin{externalDocument}[pages={3,4},
compiler=context,docType=context,runs=2,...]
```

voss-25.tex

```
\definehead
  [myhead]
  [section]
\setuphead
  [myhead]
  [numberstyle=bold,
   textstyle=bold,
   before=\hairline\blank,
   after=\nowhitespace\hairline]
```

```
\startstandardmakeup
\midaligned{From Hasselt to America}
\midaligned{by}
\midaligned{J. Jonker and C. van Marle}
\stopstandardmakeup
\placecombinedlist[content]
\chapter{Introduction}
\input knuth \input knuth
\chapter[rensselaer]{The Rensselaer family}
\section{The first born}
\input knuth
\section{The early years}
... in those days Hasselt was ...
\section{Living and workin in America}
\input knuth
\chapter[lansing]{The Lansing family}
\input knuth
... the Lansing family was also ...
\chapter[cuyler]{The Cuyler family}
\input knuth
... much later Tydeman Cuyler ...
\myhead[headlines]{And the end}
foo
```


2

1 Introduction

Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large-scale user; the designer should also write the first user manual.
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.
But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments. Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large-scale user; the designer should also write the first user manual.
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.
But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

3

2 The Rensselaer family

2.1 The first born

Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large-scale user; the designer should also write the first user manual.
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.
But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

2.2 The early years

... in those days Hasselt was ...

2.3 Living and workin in America

Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large-scale user; the designer should also write the first user manual.
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.
But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

8 Running arbitrary external commands

To typeset listing of the current directory of this document we can use the macro `\runExtCmd` with the optional argument `redirect`. We filter the output with additional commands.

```
\runExtCmd[redirect]
  {ls -laB | awk '{print $6,$7,$8,$9}' }
  {voss}
```

to produce the directory listing:

```
Nov 3 18:49 .
Nov 1 23:39 ..
Jun 3 17:36 .dict.pws
Nov 3 18:49 Exa-extern
Jun 3 18:04 Makefile
Nov 3 18:49 firstpage.tex
Nov 3 18:49 lastpage.tex
Nov 3 18:49 tb135voss-extern.aux
Nov 3 18:49 tb135voss-extern.bbl
Jun 3 17:36 tb135voss-extern.bib
Nov 3 18:49 tb135voss-extern.blg
Nov 3 18:49 tb135voss-extern.log
Oct 31 16:12 tb135voss-extern.ltx
Nov 3 18:49 tb135voss-extern.out
Nov 3 18:49 tb135voss-extern.pdf
```

The general behaviour is similar to the environment, `externalDocument`: the output is saved in an intermediate file, in this case `voss-`⟨*num*⟩`.txt` and then read by `\VerbatimInput`. The options `code` and `showFilename` are off by default.

9 Other options

Most of the options which `hvextern` provides for `externalDocument` and `\runExtCommand` have been discussed. Here is a brief summary of some that have not been seen, or mentioned only in passing.

`force` With `force=false` an existing output file is used, thus reducing compilation time. This option should only be used in exceptional cases, because with it, a change in the main document in the source code of the example does not lead to updated example output.

`moveToExampleDir` Move all generated example files, both source and output, to the directory specified by `ExamplesDir`. This can ease document development and maintenance when there are many examples. The directory itself must be created by the user.

`ExampleDir` The directory to which example files are moved if requested.

`cleanup` Auxiliary files from an (LA)TEX or other run can be deleted to improve the overview in a directory. By default, these are the `.aux` and `.log` files: `cleanup={aux,log}`.

`framesep` Value for `\fbox` if keyword `frame` is used.

`mpsep` Distance between code and output (default 1 em).

`pagesep` Distance between pages for multipage output (default 1 em).

`shiftFN` Length to shift marginal filename; positive values shift up.

`inline` False by default; if true, include the generated output in the paragraph, not as a display.

`showoutput` True by default; if false, omit the generated output.

`code` True by default (unless `inline=true`); if false, omit the source code listing.

`tcbox` If `false`, do not use any box commands from the `tcolorbox` package (for debugging and in case of bugs).

`eps` Convert the generated PDF file to EPS (historical reasons).

10 Caveats

Due to issues with `tcolorbox`, you can expect problems if a page break appears in the code part immediately before the first code line is printed. In such a case put a `\newpage` or `\pagebreak` just *before* the `externalDocument` environment. If you do not need `tcolorbox` features, you can disable its use with the option `tcbox=false`.

References

[1] C. Heinz, J. Hoffmann, B. Moses. The listings package, version 1.8d, 2020-03-24. `ctan.org/pkg/listings`

[2] R. Niepraschk. The showexpl package, version 0.3s. `ctan.org/pkg/showexpl`

[3] T.F. Sturm. The tcolorbox package, version 5.0.2, 2022-01-07. `ctan.org/pkg/tcolorbox`

[4] T. Van Zandt, H. Voß, et al. The fancyvrb package, version 4.2. `ctan.org/pkg/fancyvrb`

[5] H. Voß. The hvextern package, version 0.28. `ctan.org/pkg/hvextern`

◇ Herbert Voß
Wasgenstraße 21
14129 Berlin, Germany
herbert (at) dante dot de
https://hvoss.org/

Zeit und Raum
Erkenntnis durch Bilder

Barghoorn@zedat.fu-berlin.de

Kant und Friedrich

Wir befinden uns momentan im Kant-Jahr. Der berühmte deutsche Philosoph wurde am 22. April vor dreihundert Jahren geboren. Das regt diverse Medien an, Kants Werke aus den Tiefen der Regale und Universitätsfolianten hervorzuholen und dem Publikum die Frage zu stellen, wie sich seine Lehren und die Erkenntnisse seiner philosophischen Nachfolger in unserer heutigen Zeit nutzen lassen. Außerdem jährt sich 2024 zum 250. Mal der Geburtstag des gleichfalls berühmten Malers der Romantik, Caspar David Friedrich, dessen Bilder ebenfalls eine nachhaltige Wirkung bis heute haben. Beide nennt man auch *Leitfiguren der Moderne*. Das hat mich zu dem Projekt inspiriert, die klassischen philosophischen Konzepte wissenschaftlich-philosophischen Realitätsbeschreibung auf digital gespeicherte Bilder anzuwenden. Kurzgefasst lautet meine Aufgabenstellung, mit welchen Metainformationen sind Bilder mindestens auszustatten, um die Realität nachhaltig zu beschreiben.

Die Anwendung des Konzeptes von Zeit und Raum aus der klassischen Philosophie, definiert von Immanuel Kant und Arthur Schopenhauer, kann in den Kontext zur modernen Fotografie und der Verwaltung digitaler Bilder gesetzt werden. Dieser Artikel zielt darauf ab, die Verbindung zwischen abstrakten philosophischen und physikalischen Ideen und der Eigenschaft und Organisation digitaler Bilder herzustellen, indem man sie praktisch anwendet und damit ihre allgemeine Gültigkeit bestätigt.

Zeit und Raum als Instrumente der Wahrnehmung und Erkenntnis

Raum und Zeit sind für Immanuel Kant die einzigen reinen Formen sinnlicher Anschauung **a priori**, also **vor** jeder Erfahrung sowie von notwendiger Allgemeingültigkeit. Raum und Zeit sind damit keine empirischen Begriffe, sie sind auch nicht substanziell, dennoch existieren sie.

Arthur Schopenhauer, als Nachfolger Kants, verlagert Zeit und den Raum in das Subjekt hinein. Zeit und Raum, welche die erste Klasse der systematischen Ordnung der Schopenhauer'schen Vorstellungen ausmachen, sind beide notwendige Bedingungen einer empirischen Vorstellung.

Die Zeit ist ein Element unseres Bewusstseins und zugleich eine Form der empirischen Vorstellung. Die Berücksichtigung der *Zeit* eröffnet die Möglichkeit, dass wir die Dinge *nacheinander* ordnen können (Reihung).

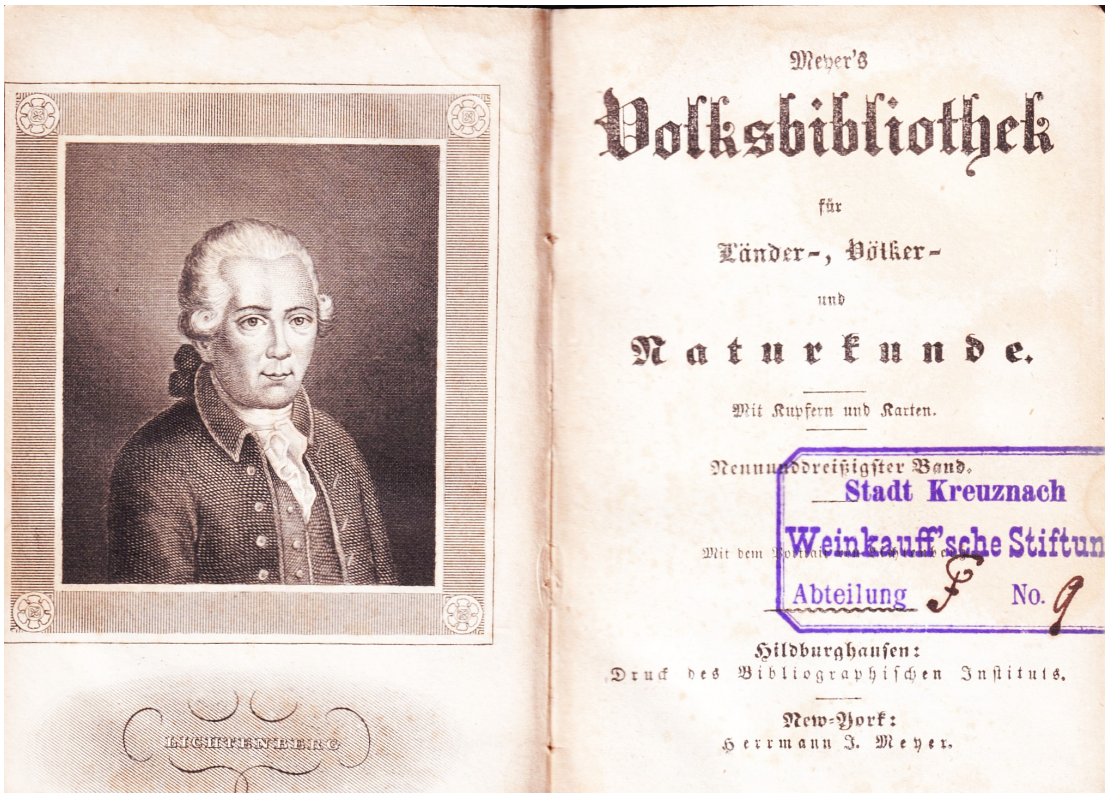
Der *Raum* ist ebenfalls ein Element unseres Bewusstseins und die Bedingung dafür, dass wir die Dinge *nebeneinander* ordnen können (Ballung). Das Kausalitätsprinzip ist die 3. Form der empirischen Vorstellung nach Schopenhauer, die hier nicht weiter untersucht werden soll. Seine kurze Zusammenfassung über die Beziehung zwischen Zeit und dreidimensionalem Raum zur Kausalität findet man im Anhang.

Weiterhin gilt nach der von Einstein begründeten Relativitätstheorie, dass die Zeit nicht vom Raum zu trennen ist. Ohne Raum keine Zeit und ohne Zeit kein Raum. Es gibt die Abhängigkeit der Zeit vom Raum, so Einstein.

Kaum Bilder zur Illustration (Erleuchtung, Erklärung)

In Meyers Volksbibliothek für Länder--, Völker-, und Naturkunde, die ab 1854 in 102 Bänden erschien, findet man viele interessante Artikel und Beschreibungen, jedoch sehr wenige gedruckten Bilder. Pro Band höchstens ein Kupfer oder eine Landkarte. Bewundernswert, dass man damals den Anspruch hatte, das Wissen der Zeit auf 22.000 Seiten zu publizieren. Nicht durch Stichworte, sondern einzelne Artikel. Der spätere Meyers Herausgeber, Dr. Friedrich Hofmann, wendet sich wie folgt an die Leser: "*An Mannigfaltigkeit des Stoffs fehlte es unseren Bänden nicht. Wir sind in denselben über die Alpen gestiegen und zu Königsgräbern gegangen, haben vor Fürstenschlössern gestanden in Thälern und auf Bergen, durchwanderten die neue Welt und das Morgenland, weilten in Bädern und Goldgräbereien, blickten auf Trauermale verlorener Nationen, schwammen auf den Strömen der Heimath und auf den Meeren der Ferne, betrachteten die Bauten der Vergangenheit und die Werke der Restauration, freuten uns der Werkstätten und Paläste der Industrie und der Markthallen der Landwirthschaft, wir sahen Zwingburgen des Herrscherthums und Ehrendome der Kunst und des Ruhms, finstere Glaubensfestungen und wonnevolle Stätten des Friedens*".

Zum Beispiel gibt es im neununddreißigsten Band dieses alten Lexikons vor 222 Textseiten nur *ein* Bild des Physikers und Schriftstellers Georg Christoph Lichtenberg (1742 – 1799), der als jüngstes von 18 Kindern seiner Eltern besonders durch seine Aphorismen berühmt wurde, die erst nach seinem Tod erschienen sind.



Im Verhältnis zum Schriftdruck war die Wiedergabe von Bildern aufwändig und teuer, es gab hierfür eigene Berufe wie Lithograph oder Reprofotograf. Seit der Erfindung des Buchdrucks wurden zur Erklärung von Texten zunächst Holzschnitte verwendet, die selten auch handkoloriert waren. Die Herstellung von Bildern durch Holzschnitt wurde ab 1600 von Kupferstich, Stahlstich und Lithographie abgelöst. Letztere farbig besonders schön, als Ergebnis der aufwändigen Drucktechnik meist kleine Kunstwerke.
Auch Computer Monitore und Drucker haben sich lange gegen Bilder gesträubt, mein hochwertiger IBM 8514-Bildschirm mit Grafikbeschleuniger von 1989 konnte wohl schon von APL erzeugte hochauflösende Bilder darstellen, jedoch gab es lange keine Möglichkeit, diese schönen Bilder und Grafiken auch zu drucken. Man verwies mich damals beim Kundendienst lakonisch auf die Verfahren der Photochemie.

Die Postkarte als Vorläufer von Selfies

Schon 1865 schlägt der Postreformer und später von Bismarck zum Generalpostdirektor ernannte, Heinrich von Stephan, die Einführung eines „offenen Postblattes“ als einfache und kostengünstige Alternative zum Brief vor. Es gab lange auch Widerstand gegen die Einführung der Postkarte, weil man um die guten Sitten und das Briefgeheimnis besorgt war. Vom Publikum wurde dieses neue Medium der Kommunikation jedoch weltweit begeistert angenommen. Die Farbigkeit der Ansichtskarte erhöhte ihre Beliebtheit, die Ansichtskarte wird zum Bildungs- und Sammlungsobjekt. Allein im Jahr 1900 befördert die deutsche Reichspost 440 Millionen Ansichtskarten. Schon damals macht der Begriff der *Bilderflut* die Runde. Postkarten wurden verschickt und gesammelt, um die Daheimgebliebenen zu erfreuen, in Staunen zu versetzen und die oft mutigen Reisenden zu bewundern. So schrieb damals ein Reisender in einer englischen Zeitschrift: „Kürzlich erstieg ich gemeinsam mit einer größeren Gesellschaft den Rigi. Unmittelbar nachdem wir den Gipfel erklommen hatten, rannte jeder zum nahegelegenen Hotel und raufte sich um Postkarten. Fünf Minuten später schrieb ein jeder, als ginge es ums liebe Leben. Ich gewann den Eindruck, dass diese ganze Gesellschaft nicht um der Erfahrung selber willen den Berg erstiegen hatte, sondern um eine Postkarte loszuwerden.“

Was heute WhatsApp, Signal und Selfies fast spielerisch leisten, konnte früher die Postkarte unter dem Aspekt von Raum und Zeit meist sogar noch besser. Für das Thema von Bedeutung ist die Tatsache, dass die von der Post beförderten Ansichtskarten meist die Raum-Zeit Anforderung genial erfüllen: das Bildmotiv und der Aufnahmeort aufgedruckt. Auf der Rückseite der Postkarte schrieb man früher ordentlich, wie zuvor auf Briefen, das Datum und außerdem gab es Briefmarke und Poststempel, oft sogar zwei, so dass man die Beförderungszeit in Tagen und teilweise sogar in Stunden feststellen konnte. Außerdem werden auf der Ansichtskarte der Verlag und der Fotograf bzw. der Lithograph genannt. Diese Fülle von Informationen sollte uns heute anspornen, die schon vor 120 Jahren selbstverständlichen Kriterien für die neuzeitlichen gesendeten und gespeicherten digitalen Bilder hinsichtlich Raum und Zeit ebenfalls zu erfüllen.

Diese 1899 an einem Tag als Ansichtskarte gelaufene echte Lithographie schrieb ganz liebevoll eine Freundin aus Oberneukirch an ihre Freundin in Eibau/Oberlausitz.
Beim heutigen Versand der Fotos mit WhatsApp ist zu beachten, dass dieser, wie auch andere Messenger, die Fotos komprimiert und sämtliche Metadaten aus den Fotos entfernt, so dass diese kein Aufnahmedatum mehr haben, sondern nur ein Dateidatum, welches gesetzt wird, wenn die Datei auf das Smartphone geschrieben wird. Außerdem geht dabei auch der Aufnahmestandort verloren.

Neuerdings kann man über WhatsApp auch Fotos mit maximaler Auflösung einschließlich der EXIF-Daten versenden. Man geht in etwa so vor wie beim Email-Versand und verschickt ein oder mehrere Fotos mit dem Büroklammersymbol als Anhang wie ein Dokument. Der Empfänger der Nachricht muss solcherart empfangenen Bilddokument gegebenenfalls den ursprünglichen Dateityp, meist *jpg*, wieder einstellen.
Warum macht man heute so viele Fotos, welche Anlässe gibt es für die Bilderflut?
Hauptsächlich Porträts, Gruppen, Essen, festliche Anlässe, Familienfeiern und Reisen.

Format und Attribute von Bilddateien

Als Formate für Bilddateien sind das JPEG-Format, TIFF, GIF, PDF und RAW derzeit allgemein verbreitet. Darüber hinaus gibt es weniger bekannte Formate wie PSD, BMP, EPS, um nur einige zu nennen.
Die eigentlichen Bildinformationen, die für die Abbildung eines Bildes notwendig sind, lassen sich durch weitere Attribute, wie Aufnahmezeit und –ort sowie Apparat, Belichtungszeit etc ergänzen. Diese Informationen sollen standardmäßig mit der Datei zusammen als Metainformationen gespeichert werden. Es besteht der Anspruch, dass diese Metadaten vor unbeabsichtigten Manipulationen der Bilddatei geschützt sein sollen. Zum Beispiel kann der bei der ursprünglichen Entstehung der Bilddatei gespeicherte Aufnahmezeitpunkt während der Bearbeitung im Dateisystem verändert werden. Das Dateisystem von Computern verwendet üblicherweise verschiedene Zeitangaben. Beim Kopieren der Dateien auf andere Datenträger wird meist nicht das tatsächliche Erstelldatum/Aufnahmedatum übernommen, sondern z.B. setzt Microsoft Windows das Datum des Zeitpunkts des Kopierens als neues Dateidatum „Erstellt“. Auch beim Umbenennen der Datei kann sich das Datum ändern. Das Änderungsdatum kann so zum Erstelldatum werden und das so wesentliche Aufnahmedatum verloren gehen.

Rettung durch EXIF

Die Metadaten werden heute normalerweise im sogenannten Exif-Format in die Bilddateien geschrieben, ohne diese selbst zu beeinflussen, in den sogenannten *Header*. Das ist der Bereich am Anfang einer Bilddatei, der sich noch vor den eigentlichen Bildinformationen befindet. Die Metadaten sind vergleichbar den Informationen, die man früher neben oder unter den Fotos im Album oder auf der Rückseite seiner Fotos in der losen Sammlung festgehalten hat. Um auf diese Weise den Reiseverlauf in Raum und Zeit für die Nachwelt festzuhalten. Bei Diapositiven war das Gedächtnis stärker gefordert. Selbst die beschrifteten Dias ließen sich während der Vorführung im Dunkeln nur nach Reiseablauf in etwa nach Ort und Zeit bestimmen.
Inzwischen speichern zum Glück praktisch jedes Smartphone und viele Digitalkameras diese zusätzlichen Informationen bei jeder Aufnahme in der Bilddatei mit ab.
Das so nützliche Exif gibt es seit 1995 und es wurde seitdem ständig verbessert und allgemein verfügbar gemacht. EXIF steht für Exchangeable Image File Format. Es handelt sich um eine standardisierte Art der Speicherung von nützlichen Metadaten in digitalen Bilddateien. Diese umfassen zahlreiche technische Informationen über die Entstehung des Bildes, darunter Uhrzeit und Datum der Aufnahme, verwendetes Kameramodell, Objektiv

und Aufnahmeeinstellungen, Geo-Daten u.a. die Höhe über Meeresspiegel und viele andere Informationen.
Zum Anzeigen und Bearbeiten von Exif-Einträge der Bilddateien gibt es zunächst Phil Harveys *ExifTool*, eine viel genutzte *Perl*-Bibliothek, quasi der Standard für Programmierer. Da Perl für sehr viele Plattformen zur Verfügung steht, gilt somit das Gleiche auch für das ExifTool. Für Windows, Linux und Mac OS X stehen ausführbare *Kommandozeilen*-Versionen zur Verfügung, die ohne ein installiertes Perl laufen. Auch mit APL und dem ExifTool lassen sich die Exif Daten lesen und verändern, also zum Beispiel ursprüngliche Aufnahmezeit und -ort – falls vorhanden – ermitteln und gegebenenfalls korrigieren.
Außerdem können im Exif-Header fast beliebige weitere Informationen als sogenannte „Tags“ (engl. „tag“ = Kennzeichen, Markierung) hinterlegt werden. Die Tags bestehen jeweils aus einem Paar von Namen und Wert. Die Werte können verschiedenen Typs und verschiedener Länge sein. Zur eindeutigen Identifizierung hat jedes Tag eine Nummer (Tag-ID). Die Installation des ExifTools ist sehr einfach, man benötigt nur die Exe-Datei.

EXIF-Beispiele

Mit APL2 und Exiftool sollen die Exif Informationen aus einer Bilddatei gelesen werden. Alle Bilder werden gesucht, die im Verzeichnis *apl* mit „friedr“ beginnen.

```
ρR←('c:\users\barg\exiftool E:\apl\friedr*')PIPE ''
77 76
```

Die erste Zahl zeigt die Anzahl der Zeilen in EXIF an, die gelesen wurde.
Es handelt sich bei diesem Beispiel um ein Bild von Caspar David Friedrich, genannt „Hügel und Bruchacker bei Dresden“.

Die mit APL2 gelesenen Exif-Daten, hier nur eine Auswahl der Zeilen:

```
ExifTool Version Number         : 12.81
File Name                       : Friedrich.jpg
File Size                       : 305 kB
File Modification Date/Time     : 2024:04:06 11:38:30+02:00
File Type                       : JPEG
Make                            : samsung
Create Date                     : 2024:02:27 15:30:21
Shutter Speed Value             : 1/40
Camera Model Name               : SM-G973F     (Modell S10)
Aperture Value                  : 1.5
Exposure Time                   : 1/40
GPS Altitude                    : 57 m Above Sea Level
GPS Latitude                    : 53 deg 33' 21.21" N
GPS Longitude                   : 10 deg 0' 6.05" E
```

Wenn man diese GPS-Daten z.B. bei GeoSetter eingibt, wird die Hamburger Kunsthalle angezeigt.
Um die Funktionalität von EXIF in vollem Umfang zu gewährleisten, sind einige Vorbedingungen zu erfüllen und gewisse Einstellungen zu beachten.

Ursachen für fehlende oder falsche GEO- oder Zeitdaten in EXIF

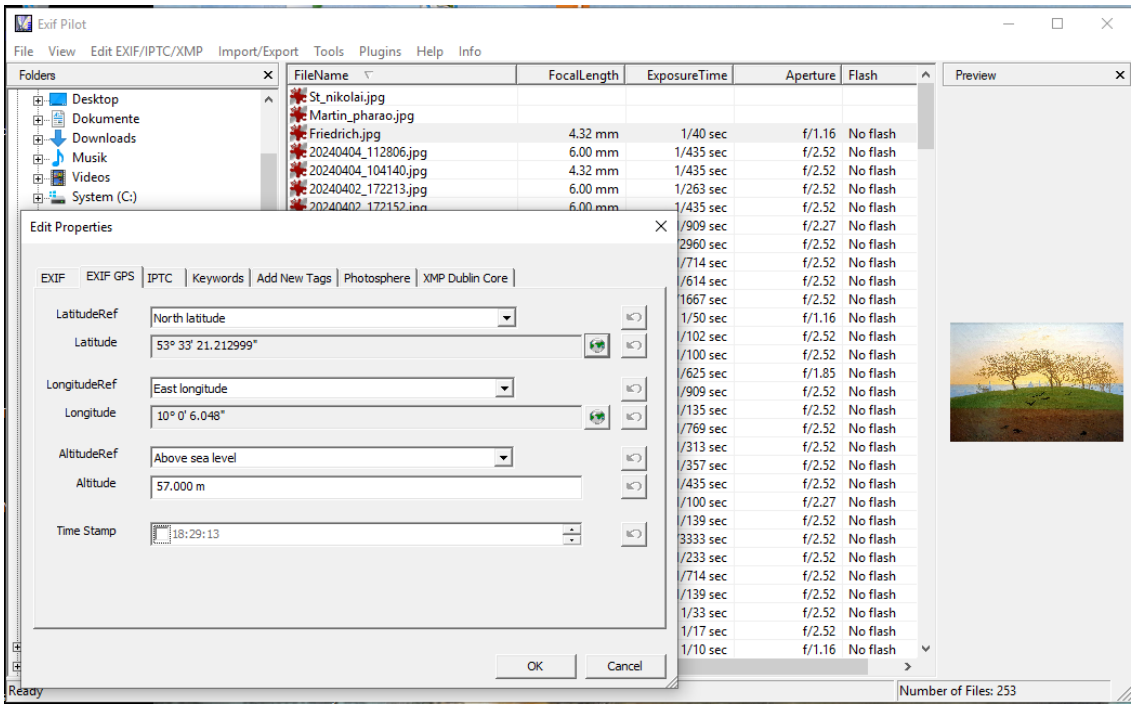
- Die Kamera /das Mobiltelefon kann kein GPS, das Modul fehlt gänzlich.
- Bei der Kamera/dem Mobiltelefon war GPS nicht eingestellt.
  [GPS muss beim Mobiltelefon unter Einstellungen (Standortdienste) sowie beim Kameraprogramm (Geotagging) eingestellt sein.]
- Es gab keinen GPS-Empfang (geschlossene Räume) oder der Empfang war ungenau.
- Die Dateien wurden umkopiert oder z.B. die Größe oder andere Eigenschaften verändert. Weitere Ursachen können eine Namensänderung, einfacher Dateiversand mit Whatsapp, Signal, Facebook etc. sein. Oder die Bilder wurden in Emails mit einer Bildkompression verschickt. Bei normalem Bildversand mit Email mit Anhang bleibt EXIF komplett erhalten.
- Gescannte Fotos, hier wird als Apparat der Scanner und der Standort des Scanners als Aufnahmeort gespeichert. Das gescannte Bild wurde in der Regel zu anderer Zeit aufgenommen.
  (Im Windows Explorer lässt sich nur das Datum, nicht aber die Zeit eingeben oder verändern.)
- Die Synchronisation von Aufnahmezeit und –datum von verschiedenen Teilnehmern und Kameras einer Gruppe stimmt nicht.
  (Probleme durch Ortszeit-Heimatzeit bzw. Sommerzeit- Winterzeit Umstellung.)

Ergänzung oder Korrektur der EXIF-Daten

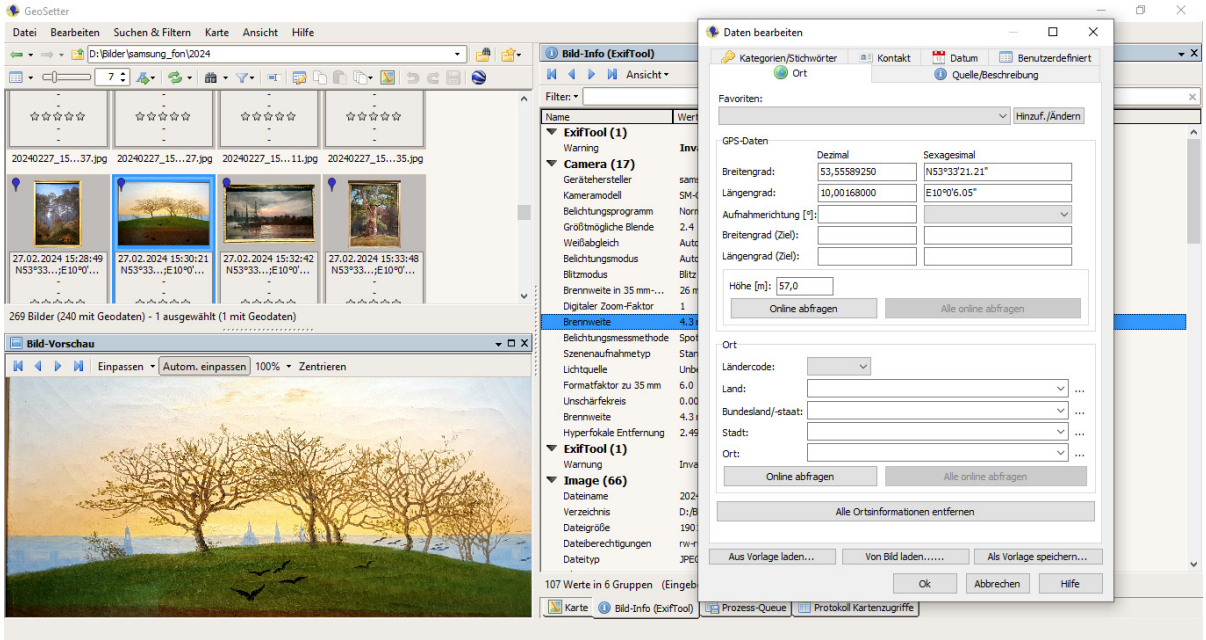
Variante 1: Eigenes Vorwissen oder Erinnerungen sind vorhanden
Folgende Möglichkeiten bieten sich auf dem Computer oder Smartphone an:

Das ExifTool von Phil Harveys. Dieses Kommandozeilenwerkzeug lässt sich mit vielen Programmiersprachen verwenden und einbinden, um EXIF-Daten zu lesen und zu ergänzen.
Mit dem Open-Source-Tool ExifToolGUI von Bogdan Hrastnik bekommt man, wer möchte, eine praktische Oberfläche für das ExifTool.
Hier werden beispielhaft die unter Windows freien Computerprogramme Exif-Pilot und GeoSetter vorgestellt.

Daten Ergänzung mit Exif-Pilot



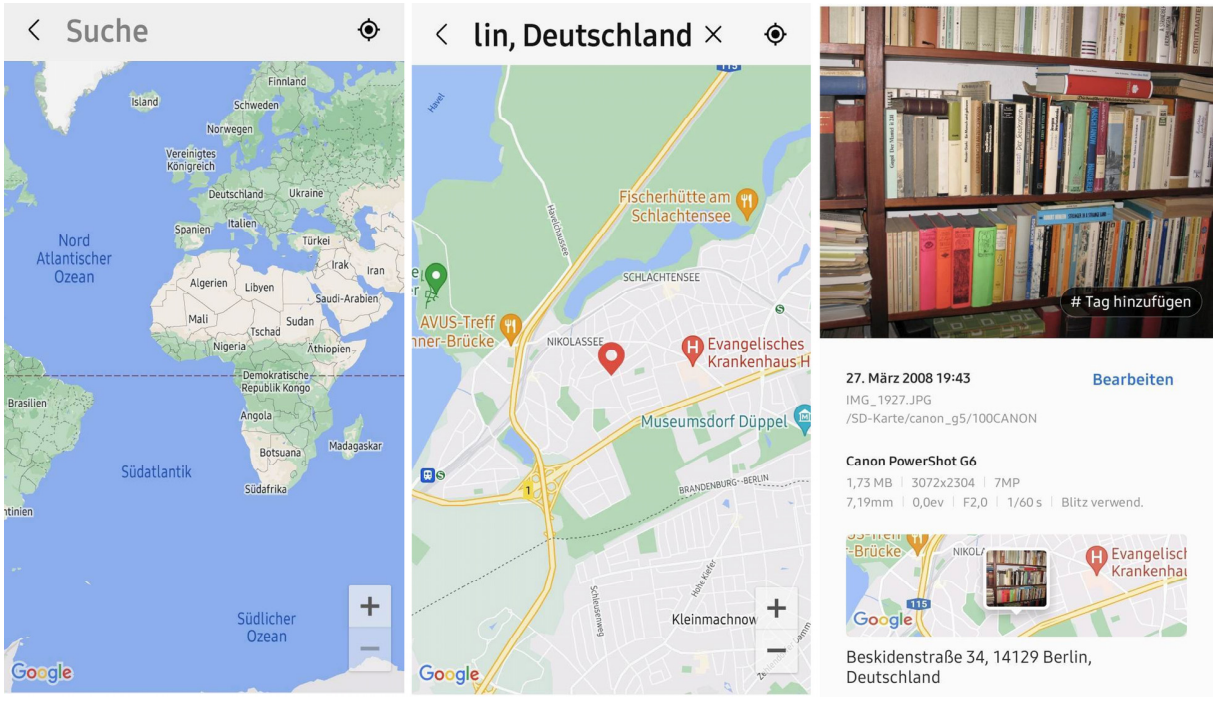
EXIF-Daten ergänzen mit dem komfortablen GeoSetter



Das komfortabelste Programm, welches ich gefunden habe, läuft nur auf einem Android Smartphone, speziell ab dem Samsung S10, nämlich die Gallery Version 14.5.04.4. Mit dieser App lassen sich auch die EXIF-Daten beliebiger Blöcke von Dateien gleichzeitig sehr benutzerfreundlich einstellen und bearbeiten.
Nach dem Starten von Gallery öffnet man zuerst einen bestimmten Ordner und wählt mit dem Dreipunktmenü eine oder mehrere Dateien durch Markieren aus. Danach kann man weiter über ein Dreipunktmenü *Standort bearbeiten* auswählen.

Die nächsten Schritte der Bearbeitung werden hier graphisch veranschaulicht:

1. Eingabe Adresse, Land, Stadt, Gebirge
2. Standort anzeigen gegebenenfalls Korrektur
3. Anzeige Bild sowie EXIF Aufnahmedatum und -ort



Den Aufnahmeort muss man anfangs nicht mit höchster Genauigkeit angeben, es reichen Ortsbezeichnungen wie Hessen, Mainz, Harz oder Allgäu. Gallery ist so intelligent, das es die übergeordneten Kategorien kennt. Wenn man z.B. Zehlendorf als Ort eingibt, weiß das Programm, dass Zehlendorf zu Berlin gehört. Bei Ortsteilen ist diese Verknüpfung nicht gesichert.
Wenn man mit Gallery die Fotos nach einer bestimmten Ortsbezeichnung sucht, erfolgt die Ausgabe der Bilder nach dem Datum gruppiert.
Analog kann man Zeit-Suchvorgänge starten nach Monat und Jahr. Die Ergebnisse sind nicht immer zuverlässig, bieten Anlass für nötige Korrekturen bei den Bildern.
Eine weitere interessante Variante für die Bildersuche bietet Gallery durch die Personenabfrage, wobei das Programm mit automatischer Bilderkennung arbeitet. Die erkannten Bilder kann man mit den Personennamen ergänzen.
Variante 2: keine Erinnerungen oder fremde Fotos, deshalb Anwendung von Bilderkennungsprogrammen

Um vergessene Ansichten oder gänzlich unbekannte Bilder zu erkennen, empfiehlt sich das Programm *lens* von Google. Man kann es auf verschiedene Weise auf allen Plattformen aus dem Browser, aus dem Google Programm *Fotos* oder mit der eigens installierten lens-App starten und oft zu verblüffenden Ergebnissen kommen.
In der Abbildung zeigt der Pfeil auf den Startknopf von *lens* in der Google Suchzeile.

Mit *lens* lassen sich vielerlei Objekte, wie Pflanzen, Tiere, Gebäude, Fahrzeuge, Uniformen, Kunstwerke etc. erkennen. Eine gewisse Bekanntheit der Objekte ist natürlich Voraussetzung für den Versuch, schon anderweitig *Bekanntes* wiederzufinden. Ein Gruppenfoto vor dem Brandenburger Tor wird sich stets erfolgreich erkennen lassen.
Üblicherweise werden die Objekte einzeln erkannt. Google Lens arbeitet auf Grundlage der künstlichen Intelligenz. Diese Anwendung ist in der Lage, aus gespeicherten Fotos oder Live-Aufnahmen zusätzliche Informationen abzurufen, mit den eigenen Bildern zu vergleichen und die gesuchten Objekte zu identifizieren. Ähnliche Objekte oder Ansichten werden von lens vorgeschlagen und bei relativ bekannten Ansichten ist fast immer das gesuchte Objekt dabei. Sobald man sein Bild in diesen Vorschlägen wiedergefunden hat, kann man die eigenen Fotos mit den Raum/Zeit-Daten bzw. *tags* ergänzen.

Ordnung der Objekte durch ihre Attribute

Infolge der Ausstattung der Objekte (Bilder) mit Attributen durch EXIF wird die ursprüngliche Speicherung der Objekte in Ordnern obsolet. Die Suchabfragen in Gallary nach Raum oder Zeit einzeln bzw. auch Raum *oder* Zeit (SQL Klausel group by Ort, Zeit) liefern den gewünschten Output über alle Ordner hinweg. Eine wirklich faszinierende Möglichkeit, die sich aber erst eröffnet, wenn man die EXIF-Daten hinsichtlich Raum und Zeit vervollständigt.
Eine kleine Mühe, die in der Zukunft reichlich belohnt wird.

C.C. Friedrich,
Schiffe auf der Reede

Abschließend ein Aphorismus des großen Lichtenberg, der zum Thema passt, wie ich hoffe: „Leute nennen wir rasend, wenn sich die Ordnung ihrer Begriffe nicht mehr aus der Folge der Begebenheiten in unserer ordentlichen Welt bestimmen lässt; deswegen ist gewiss eine sorgfältige Betrachtung der Natur, oder auch die Mathematik, das sicherste Mittel wider Raserei; die Natur ist sozusagen das Laufseil, woran unsere Gedanken geführt werden, dass sie nicht ausschweifen."

Anlage, Raum, Zeit, Kausalität

Arthur Schopenhauer, *Über das Sehen und die Farben*, Verlag Hartknoch Leipzig 1816, die Rechtschreibung zeitgemäß

„Das Kind, in den ersten Wochen seines Lebens, empfindet mit allen Sinnen: aber es schaut nicht an, es apprehendiert nicht: daher starrt es dumm in die Welt hinein. Bald jedoch fängt es an, den Verstand gebrauchen zu lernen, das ihm vor aller Erfahrung bewußte Gesetz der Kausalität anzuwenden und es mit den ebenso a priori gegebenen Formen aller Erkenntnis, Zeit und Raum, *zu verbinden: so gelangt es von der Empfindung zur Anschauung, zur Apprehension: und nunmehr blickt es mit klugen, intelligenten Augen in die Welt. Da aber jedes Objekt auf alle fünf Sinne verschieden wirkt, diese Wirkungen dennoch auf ein und dieselbe Ursache zurückleiten, welche sich eben dadurch als Objekt darstellt; so vergleicht das die Anschauung erlernende Kind die verschiedenartigen Eindrücke, welche es vom nämlichen Objekt erhält, betastet, was es sieht, besieht was es betastet, geht dem Klang nach zu dessen Ursache, nimmt Geruch und Geschmack zu Hilfe, bringt schließlich auch für das Auge die Entfernung und Beleuchtung in Anschlag, lernt die Wirkung des Lichts und des Schattens kennen und schließlich, mit vieler Mühe, auch die Perspektive, deren Kenntnis zustande kommt durch die Vereinigung der Gesetze des Raums mit dem der Kausalität, die beide a priori im Bewußtsein liegen und der Anwendung bedürfen, wobei nun sogar die Veränderungen, welche, beim Sehen in verschiedene Entfernungen, teils die innere Konformation der Augen, teils die Lage beider Augen gegeneinander erleidet, in Anschlag gebracht werden müssen: und alle diese Kombinationen macht für den Verstand schon das Kind, für die Vernunft, d. h. in abstracto, erst der Optiker. Dergestalt also verarbeitet das Kind die mannigfaltigen Data der Sinnlichkeit, nach den ihm a priori bewußten Gesetzen des Verstandes, zur Anschauung, mit welcher allererst die Welt als Objekt für dasselbe da ist. Viel später lernt es die Vernunft gebrauchen: dann fängt es an, die Rede zu verstehen, zu sprechen und eigentlich zu denken."*

Physik

Quantencomputer: Gewissheit aus dem Zufall ziehen

Berlin. Quantencomputer werden mit zunehmender Größe und Komplexität undurchschaubar. Mit Methoden der mathematischen Physik ist es nun einem Team gelungen, aus zufälligen, Datensequenzen konkrete Zahlen abzuleiten, die als Maßstab für die Leistungsfähigkeit eines Quantencomputersystems dienen können. An der Arbeit, die nun in Nature communications veröffentlicht ist, waren Experten des Helmholtz-Zentrum Berlin, der Freien Universität Berlin, des Qusoft Forschungszentrum Amsterdam, der Universität Kopenhagen sowie des Technology Innovation Institute Abu Dhabi beteiligt.

Mit Quantencomputern lassen sich insbesondere Quantensysteme deutlich effizienter berechnen und zum Beispiel Probleme in der Materialforschung lösen. Je größer und komplexer jedoch Quantencomputer werden, desto weniger lassen sich die Prozesse durchschauen, die zum Ergebnis führen. Um solche Quantenoperationen zu charakterisieren und die Fähigkeiten von Quantencomputern mit der klassischen Rechenleistung bei denselben Aufgaben fair zu vergleichen, werden daher passende Werkzeuge gebraucht. Ein solches Werkzeug mit überraschenden Talenten hat nun ein Team um Prof. Jens Eisert und Ingo Roth entwickelt.

Roth, der aktuell am Technology Innovation Institute in Abu Dhabi eine Gruppe aufbaut, erläutert: „Aus den Ergebnissen zufällig gewählter Experimente können wir mit mathematischen Methoden nun viele verschiedene Zahlen extrahieren, die zeigen, wie nah die Operationen im statistischen Mittel an den gewünschten Operationen sind. Damit kann man aus den gleichen Daten viel mehr lernen als zuvor. Und zwar – das ist das Entscheidende – wächst die benötigte Datenmenge nicht linear sondern nur logarithmisch." Dies konnte das Team sogar mathematisch beweisen. Konkret bedeutet das: Um hundertmal so viel zu lernen, werden nur doppelt so viel Daten gebraucht. Eine enorme Verbesserung.

„Es geht hier um das Benchmarking von Quantencomputern", sagt Eisert, der eine gemeinsame Forschungsgruppe zu theoretischer Physik am Helmholtz-Zentrum Berlin und der Freien Universität Berlin leitet. „Wir haben gezeigt, wie man mit randomisierten Daten solche Systeme kalibrieren kann. Das ist eine sehr wichtige Arbeit für die Entwicklung von Quantencomputern."

Kontakt:

Ingo Roth, Quantum Research Center, Technology Innovation Institute (TII), Abu Dhabi, UAE, E-Mail: ingo.roth@tii.ae.
Prof. Jens Eisert, Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany
Helmholtz-Zentrum Berlin für Materialien und Energie, 14109 Berlin,
E-Mail: jenseisert@gmail.com

Originalpublikation:

Helsen, J., Ioannou, M., Kitzinger, J. et al. Shadow estimation of gate-set properties from random sequences. Nat Commun 14, 5039 (2023). https://doi.org/10.1038/41467-023-39382-9
https://www.nature.com/articles/s41467-023-39382-9

APL-Journal
42. Jg. 2023, ISSN 1438-4531

Herausgeber: Prof. Dr. Dieter Kilsch, APL-Germany e.V., Mannheim, Homepage: https://apl-germany.de/, E-Mail: d.kilsch@th-bingen.de

Redaktion: Dipl.-Volksw. Martin Barghoorn (verantw.), Lückhoffstr. 8, 14129 Berlin, E-Mail: Martin@Barghoorn.com

Verlag: RHOMBOS-VERLAG, Bernhard Reiser, Berlin, Postfach 67 02 17, D-10207 Berlin, Tel. (030) 261 9461, eMail: verlag@rhombos.de, Internet: https://rhombos.de/

Erscheinungsweise: halbjährlich

Erscheinungsort: Berlin

Druck: dbusiness.de GmbH, Berlin

Fotonachweis: Martin Barghoorn (Umschlagseite 1 und 4, Seite 29, 31

Allgemeine Informationen

(Stand 2022)

APL-Germany e.V. ist ein gemeinnütziger Verein mit Sitz in Düsseldorf. Sein Zweck ist es, die Programmiersprache APL, sowie die Verbreitung des Verständnisses der Mensch-Maschine Kommunikation zu fördern. Für Interessenten, die zum Gedankenaustausch den Kontakt zu anderen APL-Benutzern suchen, sowie für solche, die sich aktiv an der Weiterverbreitung der Sprache APL beteiligen wollen, bietet APL-Germany den adäquaten organisatorischen Rahmen.

Auf Antrag, über den der Vorstand entscheidet, kann jede natürliche oder juristische Person Mitglied werden. Organe des Vereins sind die mindestens einmal jährlich stattfindende Mitgliederversammlung sowie der jeweils auf zwei Jahre gewählte Vorstand.

1. Vorstandsvorsitzender

Prof. Dr. Dieter Kilsch,
Dumontstraße 12, 55313 Mainz,
Tel. 06131 6982200, E-Mail:
d.kilsch@th-bingen.de.

2. Vorstandsvorsitzender:

Martin Barghoorn
Lückhoffstr. 8, 14129 Berlin,
E-Mail: Martin@Barghoorn.com

Schatzmeister

Jürgen Beckmann
Im Freudenheimer Grün 10
68259 Mannheim
Tel. 0621 7 98 08 40,
eMail: JBecki@onlinehome.de

Beitragssätze

Persönliche Mitglieder:

Natürliche Personen	32,- EUR*
Studenten / Schüler	11,- EUR*

Institutionelle Mitglieder:

Jurist./natürl. Pers.	500,- EUR*

* Jahresbeitrag

Bankverbindung
BVB Volksbank eG Bad Vilbel
BLZ 518 613 25
Konto-Nr. 523 2694

Hinweis:
Wir bitten alle Mitglieder, uns Adressänderungen und neue Bankverbindungen immer sofort mitzuteilen. Geben Sie bei Überweisungen den Namen und/oder die Mitgliedsnummer an.

https://apl-germany.de/

www.apl-germany.de

APL Germany e.V.