GAUSS' KALENDER FORMEL FÜR WOCHENTAGE

BERNDT E. SCHWERDTFEGER

Meiner Mutter zum 85. Geburtstag

ZUSAMMENFASSUNG. Eine Formel von *C. F. Gauss* für die Bestimmung des Wochentages eines Datums wird beschrieben und ein konzeptioneller Beweis gegeben. Die *Julianische Tageszahl* wird abgeleitet. Meine Implementierung der Formel in C ist beigefügt.

VORWORT

Gauss' Formel berechnet den Wochentag, *Montag, . . . , Sonntag*, ausgehend vom Kalender Datum `Tag.Monat.Jahr`. Ich lernte sie etwa 1962 in dem Buch *Mathematical Recreations* [2] von *Maurice Kraitchik (1882–1957)* kennen. Die Formel wird dort ohne Beweis angeführt, zusammen mit einer Anwendung auf immerwährende Kalender.

Dieser Artikel erklärt die Formel und ihren mathematischen Hintergrund, einschließlich eines Beweises und der Konzepte der im Programm `gauss.c` implementierten Algorithmen.

Gauss hat seine Formel nie publiziert; sie erschien erst 1927 in seinen *Werken* [1, XI, 1.]. Für die Bequemlichkeit des Lesers habe ich diese Notiz im Anhang A.1 aufgenommen.

Berlin, 17. November 2009 B. E. Schwerdtfeger

1. GAUSS' FORMEL ERKLÄRT

1.1. **Gauss' Kalenderformel.** Gegeben seien d = Tag, m = Monat, y = Jahr. Wir spalten das Jahr in den Jahrhundertteil und das zweiziffrige Jahr innerhalb des Jahrhunderts. Für $m \geq 3$ für das Jahr $y = 100 \cdot c + g$, i.e. $c = [y/100]$, $g = y - 100 \cdot c$, so daß $0 \leq g \leq 99$. Für $m = 1, 2$ für das Vorjahr $y - 1 = 100 \cdot c + g$.

Gauss' Formel gemäß *Kraitchik* [2, chap. Five, The Calendar, p. 110–111] lautet:

$$w \equiv d + e + f + g + [g/4] \pmod 7$$

wo $w \equiv 1, \ldots$ zu *Montag*, . . . , etc. korrespondiert, und mit den folgenden Konstanten, die vom Monat (e) resp. Jahrhundert (f) abhängen:

Monat	m	1	2	3	4	5	6	7	8	9	10	11	12
	e	0	3	2	5	0	3	5	1	4	6	2	4

Gregorianischer Kalender	Jahrhundert	$c \bmod 4$	f
	$1600, 2000, \ldots$	0	0
	$1700, 2100, \ldots$	1	5
	$1800, 2200, \ldots$	2	3
	$1900, 2300, \ldots$	3	1

2010 *Mathematics Subject Classification.* Primary 68-04; Secondary 68N15.
Key words and phrases. Kalenderformel, immerwährender Kalender, Julianische Tageszahl.
 version 1.4, rev. 488, 4. März 2015.

Für den Julianischen Kalender gilt die gleiche Formel mit modifizierten Konstanten f gemäß der folgenden Tabelle:

Julianischer Kalender	Jahrhundert	$c \bmod 7$	f
	$0000, 0700, 1400, \ldots$	0	5
	$0100, 0800, 1500, \ldots$	1	4
	$0200, 0900, 1600, \ldots$	2	3
	$0300, 1000, 1700, \ldots$	3	2
	$0400, 1100, 1800, \ldots$	4	1
	$0500, 1200, 1900, \ldots$	5	0
	$0600, 1300, 2000, \ldots$	6	6

Einige immerwährende Kalender benutzen diese Formel (oder geben äquivalente Tabellen an). Eine — mit Auslassungen — kurze Begründung dieser Formel wird im nächsten Abschnitt 1.2 gegeben und ein vollständiger Beweis im Abschnitt 2.

1.2. **Kurze Begründung.** Die *Gregorianische Regel* für *Schaltjahre* lautet:

Ein normales Jahr hat 365 *Tage, mit einem extra Schalttag 29.02. alle vier Jahre, außer in Jahrhundertjahren, wenn das Jahrhundert nicht durch* 4 *teilbar ist.*

Mit unseren Variablen liest sich dies so

$$y = 100 \cdot c + g \quad \text{ist ein Schaltjahr} :\Longleftrightarrow$$
$$(g \neq 0 \text{ und } g \equiv 0 \bmod 4) \text{ oder } (g = 0 \text{ und } c \equiv 0 \bmod 4)$$

Diese Regel implementiert eine Periodizität von 400 Jahren, die genau $3 \cdot 24 + 25 = 97$ Schaltjahre und $3 \cdot 76 + 75 = 303$ normale Jahre haben.

Anders ausgedrückt, 400 aufeinanderfolgende Jahre im Gregorianischen Kalender bestehen immer aus genau $97 \cdot 366 + 303 \cdot 365 = 146.097$ Tagen, was zu einer Durchschnittslänge eines Gregorianischen Jahres von $\frac{146097}{400} = 365,2425$ Tagen führt (oder, wenn man das vorzieht, das aus 31.556.952 Sekunden besteht).

Eine kurze Begründung dieser Formel (mit ein paar Abkürzungen) läuft wie folgt:

Der erste Ausdruck d, der Fortschritt in Tagen, ist offensichtlich.

Da $365 \equiv 1 \pmod 7$ fällt der erste und letzte Tag jeden normalen Jahres auf den gleichen Wochentag. Das erklärt den Ausdruck g (Fortschritt von Jahr zu Jahr). Der andere Ausdruck $[g/4]$ berücksichtigt den extra Tag alle vier Jahre.

Wieviele Wochentage haben wir von einem Jahrhundert zum nächsten zu verschieben ? $100 \equiv 2 \pmod 7$, plus die 24 Schaltjahre im Jahrhundert ergeben $24 \equiv 3 \pmod 7$, also addiert jedes Jahrhundert $2 + 3 = 5$ Tage.

Für den Julianischen Kalender haben wir 25 Schaltjahre in einem Jahrhundert, deshalb verschiebt es sich hier um $2 + 4 = 6$ Tage.

Damit sind die f Werte erklärt: $5 + 5 \equiv 3 \pmod 7$, $5 + 5 + 5 \equiv 1 \pmod 7$. Ihre Abhängigkeit von $c \pmod 4$ und die Lücke von $c \equiv 3$ nach $c \equiv 0 \pmod 4$ liegt an der Schaltjahrhundert Regel (im Gregorianischen Kalender).

Damit bleiben die 'magischen' Werte für e zu erklären. Sehen wir uns dazu die folgende Tabelle an: mit der Anzahl $d(m)$ von Tagen in einem Monat, der Summe $e(m)$ der Tage in den vorhergegangenen Monaten (die Anzahl der Tage seit Beginn des Jahres, in normalen Jahren) und ihre Werte modulo 7:

m	1	2	3	4	5	6	7	8	9	10	11	12
$d(m)$	31	28	31	30	31	30	31	31	30	31	30	31
$e(m)$	0	31	59	90	120	151	181	212	243	273	304	334
(mod 7)	0	3	3	6	1	4	6	2	5	0	3	5

Der 'magische' Wert ist $e = e(m)$ für die Monate $m = 1, 2$ und $e = e(m) - 1$ für die anderen Monate ... (mod 7).

Es bleibt der Leserin als Übungsaufgabe, diese Argumentation in einen untadeligen mathematischen Beweis zu verwandeln und ein konzeptionelles Vorgehen zu finden.

Sie mag aber genauso gut das Programm im Anhang B studieren: die Funktion `offset` in `gauss.c` gibt eine andere Parametrisierung eines Datums mittels *Jahr* und *Verschiebung (offset)*; die Funktion `date` ist dazu invers.

Oder man liest den im Abschnitt 2 entwickelten konzeptionellen Ansatz.

1.3. **Historische Perspektive.** Der international gebräuchliche Kalender ist vom Römischen Kalender abgeleitet, wie er im Jahre 46 v. Chr. von Julius Caesar reformiert wurde, als er alle vier Jahre die Schalttage einfügte (*Julianischer Kalender*). Die Durchschnittslänge eines Julianischen Jahres ist daher 365.25 Tage.

Das durch den Durchgang der Sonne durch den Frühlingspunkt definierte *Sonnen–*Jahr ist meßbar kürzer, nämlich: 365 Tage, 5 Stunden, 48 Minuten und 45, 51 Sekunden, oder 31.556.925, 51 s, oder 365, 242193402 Tage. Das sind ungefähr 11 Minuten und 15 Sekunden weniger als das Julianische Durchschnittsjahr.

Im Laufe der Jahrhunderte summierte sich diese Diskrepanz und führte zum Zurückbleiben des Julianischen Jahres hinter der Zeit: es hatte zuviele Schalttage eingefügt. Im 16. Jahrhundert betrug die Abweichung etwa 12 Tage. *Ugo Boncompagni*, seit 1572 Papst *Gregor XIII*, setzte 1582 durch eine Kalenderreform den *Gregorianischen Kalender* mit den oben beschriebenen Schaltjahr Regeln ein. Auf Donnerstag den 4. Oktober 1582 im alten Kalender folgte Freitag der 15. Oktober 1582.

Auch die Gregorianischen Regeln wären nicht für alle Zeit ausreichend: es ist auch zu lang (26, 49 Sekunden) und in $24 \cdot 60 \cdot 60/26, 49 \approx 3261$ Jahren würde sich ein Tag aufsummieren. Da *Atomuhren* genauer gehen als Himmelskörper, wurde 1968 die *koordinierte Weltzeit (UTC)* eingeführt. Seither sorgen *Schaltsekunden* für die Synchronisierung von Weltzeit mit Sonnenzeit.

In dieser Arbeit werden alle Daten behandelt, als wäre der Gregorianische Kalender in alle Ewigkeit gültig (so genannter *proleptischer* Gregorianischer Kalender). Es wird auch die astronomische Nummerierung der Jahre durch ganze, positive wie negative Zahlen verwendet: $\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$

Das Programm `gauss.c` ignoriert die historischen Gegebenheiten und die von 46 v.Chr. bis 1582 (und in einigen Ländern darüber hinaus) in Kraft befindlichen Julianischen Regeln. Stattdessen behandelt es jedes Datum nach den Gregorianischen Regeln, allerdings wird auch das Julianische Datum ausgegeben.

2. Konzeptioneller Ansatz

2.1. **Heuristische Bemerkung.** Im Prinzip sehen die Tage von heute in die Vergangenheit oder Zukunft aus wie die Menge $\mathbb{Z}$ der ganzen rationalen Zahlen. Aber es gibt keinen natürlichen *Tag Null* und man hat eine willkürliche Wahl zu treffen. Wir werden einen historischen Vorschlag (1849 von John F. Herschel, nach Ideen von Joseph Justus Scaliger im 16. Jhd.) für diese Nummerierung in Abschnitt 2.4 sehen, der heute noch in der Astronomie üblich ist.

Die Menge T der Daten ist eine andere sonderbare Nummerierung von $\mathbb{Z}$ durch ein Tripel $t = (y, m, d)$ von Jahr, Monat, Tag (in einigen Ländern `d.m.y`, in anderen

`d/m/y` oder gar `m/d/y` notiert; das ISO 8601 Format schreibt `y-m-d` vor und dies wird auch in meinem Programm benutzt.

Die gesuchte Abbildung von T zur Menge der Wochentage {Montag,. . . , Sonntag} sollte so einfach sein wie die kanonische Abbildung $\mathbb{Z} \to \mathbb{F}_7$, der ganzen Zahlen modulo 7. Wir werden einen *Gauss Abbildung* genannten Kandidaten

$$\omega : T \longrightarrow \mathbb{F}_7$$

in einer ziemlich natürlichen Weise konstruieren und eine explizite Formel mit den Koordinaten y, m, d angeben — sie erweist sich als die *Formel von Gauss* (8) in Korollar 2.3.

2.2. **Schaltjahr Indikator.** Für ein Jahr $y \in \mathbb{Z}$ setzen wir das *Jahrhundert* $c = [y/100]$ und den Rest $g = y - 100 \cdot c$, so daß $0 \leq g \leq 99$. Der (Gregorianische) Schaltjahr Indikator auf der Menge der Jahre ist eine Boolsche Abbildung

$$\lambda : \mathbb{Z} \longrightarrow \{0, 1\}$$

gegeben durch

$$\lambda(y) = \begin{cases} 1 & \text{falls} \quad (g \neq 0 \wedge g \equiv 0 \bmod 4) \vee (g = 0 \wedge c \equiv 0 \bmod 4), \\ 0 & \text{sonst} \end{cases}$$

Die Teilmenge der Schaltjahre wird bezeichnet mit

$$\Lambda = \{y \in \mathbb{Z} \mid \lambda(y) = 1\}$$

Es ist

$$\Lambda = 400\mathbb{Z} \cup \bigcup_{k \not\equiv 0 \bmod 25} 4k + 100\mathbb{Z}$$

Wir definieren für einen Monat m und ein Jahr y

$$\lambda(y, m) = \begin{cases} \lambda(y) & \text{falls } m > 2 \\ 0 & \text{für } m = 1, 2 \end{cases}$$

Da wir beim Zählen der Schaltjahre durch die Jahrhunderte den Überblick behalten müssen, notieren wir einige Eigenschaften von λ, die mehr oder weniger offensichtlich aus der Definition folgen.

Lemma 2.1. *Sei* $0 \leq g < 100$. *Dann ist*

$$\sum_{l=1}^{c} \lambda(100 \cdot l) = [c/4] \qquad c \geq 0 \tag{1}$$

$$\sum_{k=1}^{g} \lambda(100 \cdot c + k) = [g/4] \tag{2}$$

$$\sum_{x=1}^{100c+g} \lambda(x) = [c/4] + 24 \cdot c + [g/4] \qquad c \geq 0 \tag{3}$$

$$-\sum_{x=1+100c+g}^{0} \lambda(x) = [c/4] + 24 \cdot c + [g/4] \qquad c < 0 \tag{4}$$

Beweis. Die Relationen (1), (2) sind offensichtlich wegen der Schaltjahr Regel. Zu (3):

$$\begin{aligned}\sum_{y=0}^{100c+g}\lambda(y) &= \sum_{l=0}^{c-1}\sum_{k=0}^{99}\lambda(100\cdot l+k)+\sum_{k=0}^{g}\lambda(100\cdot c+k)\\ &= \sum_{l=0}^{c}\lambda(100\cdot l)+\sum_{l=0}^{c-1}\sum_{k=1}^{99}\lambda(100\cdot l+k)+\sum_{k=1}^{g}\lambda(100\cdot c+k)\\ &= [c/4]+1+\sum_{l=0}^{c-1}24+[g/4]\\ &= [c/4]+1+24\cdot c+[g/4] \qquad \text{und } \lambda(0)=1.\end{aligned}$$

Zu (4): aus Symmetrie $\lambda(-x)=\lambda(x)$, folglich

$$\sum_{x=1+100c+g}^{0}\lambda(x)=\sum_{x=0}^{-1-100c-g}\lambda(x)$$

und wir sind in der Situation von (3) mit $-1-100c-g=100(-1-c)+(100-1-g)$, $-1-c\geq 0$, $0\leq 100-1-g<100$,

$$\begin{aligned}\sum_{x=1+100c+g}^{0}\lambda(x) &= 1+\sum_{x=1}^{-1-100c-g}\lambda(x)=\\ &= 1+\left[\frac{-1-c}{4}\right]+24(-c-1)+\left[\frac{100-1-g}{4}\right]=\\ &= 1+\left[\frac{-1-c}{4}\right]-24\cdot c-24+25+\left[\frac{-1-g}{4}\right]=\\ &= -[c/4]-24\cdot c-[g/4]\end{aligned}$$

wo wir die Relation $1+\left[\frac{-1-n}{4}\right]=-\left[\frac{n}{4}\right]$, gültig für jedes $n\in\mathbb{Z}$, benutzt haben. □

2.3. **Datum verschieben.** Die Menge T gültiger 'Datumstage' ist Teilmenge von

$$\mathbb{Z}\times\{1,\dots,12\}\times\{1,\dots,31\}$$

definiert durch

$$\begin{aligned}T := &\ \mathbb{Z}\times\{1,3,5,7,8,10,12\}\times\{1,\dots,31\}\\ &\cup\mathbb{Z}\times\{4,6,9,11\}\times\{1,\dots,30\}\\ &\cup\mathbb{Z}\times\{2\}\times\{1,\dots,28\}\\ &\cup\Lambda\times\{2\}\times\{29\}\end{aligned}$$

Die letzte Menge in der Vereinigung enthält die Schalttage (29. Februar).

Wir haben eine endliche Faserungsabbildung

$$\pi : T\longrightarrow\mathbb{Z}$$

gegeben durch die erste Projektion $\pi(y,m,d)=y$. Normale Fasern $\pi^{-1}(y)$, $y\notin\Lambda$, haben 365 Elemente, während 'Schalt' Fasern ($y\in\Lambda$) 366 Tage haben, i.e. die Anzahl der Tage einer Faser ist

$$|\pi^{-1}(y)| = 365+\lambda(y) \tag{5}$$

Wir verwenden die natürliche Ordnung der Tage: $t<t'$ wenn t' später als t ist. Die Anzahl der Tage vom Beginn eines Jahres bis zum Tag $t=(y,m,d)$ ist

$$|\{t'\in\pi^{-1}(y)\mid t'\leq t\}| = d+e(m)+\lambda(y,m) \tag{6}$$

Wir haben eine natürliche *Verschiebe* Operation der additiven Gruppe $\mathbb{Z}$ auf T:

$$\begin{aligned} \sigma : \mathbb{Z} \times T &\longrightarrow T \\ (n, t) &\longmapsto n + t \end{aligned}$$

wobei $n + t$ definiert ist als t verschoben um n Tage, früher ($n < 0$) oder später ($n > 0$). Wir schreiben auch $\sigma_t(n) = \sigma(n, t) = n + t$, wie auch $t' - t = n$ für $t' = n + t$.

Beispiele

$$\begin{aligned} 1 + (2000, 2, 28) &= (2000, 2, 29) \\ 1 + (2001, 2, 28) &= (2001, 3, 1) \\ -365 + (2001, 1, 1) &= (2000, 1, 2) \\ -152930 + (2001, 6, 30) &= (1582, 10, 15) \\ 1872 + (1947, 2, 4) &= (1952, 3, 21) \\ (y, m, d) &= d + e(m) + \lambda(y, m) + (y - 1, 12, 31) \end{aligned}$$

Diese Operation ist einfach transitiv. Daher erzeugt jedes $t \in T$ einen Isomorphismus

$$\begin{aligned} \sigma_t : \mathbb{Z} &\xrightarrow{\sim} T \\ n &\longmapsto n + t \end{aligned}$$

der *linear* ist: $\sigma_t(n + m) = n + \sigma_t(m)$. Die inverse σ_t^{-1} kombiniert mit $\mathbb{Z} \to \mathbb{F}_7$ gibt uns *Gauss* Abbildungen

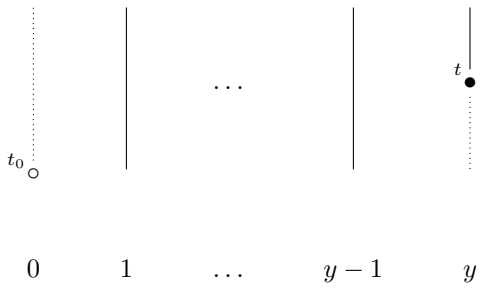
$$\omega_t : T \longrightarrow \mathbb{F}_7$$

mit der Eigenschaft $\omega_t(n+u) = n+\omega_t(u) = \omega_{-n+t}(u)$. Deshalb gibt es nur 7 davon, die voneinander durch eine Konstante $\in \mathbb{F}_7$ abweichen.

Jetzt wähle ich $t_0 = (0, 12, 31)$ als den willkürlichen Beginn unserer Zählung.

Sei $t = (y, m, d) \in T$ irgendein Datum. Um $\omega(t) = \omega_{t_0}(t)$ zu berechnen, müssen wir ein $n \in \mathbb{Z}$ bestimmen mit $n = t - t_0$. Dann ist $\omega(t) = n \mod 7 \in \mathbb{F}_7$.

Dazu werden wir durch die Fasern von t bis t_0 gehen, von der Faser $\pi^{-1}(y)$ zur Faser $\pi^{-1}(1)$, und dabei die Anzahl der Tage im Vorübergehen zählen müssen.



(in diesem Bild ist $y > 0$ angenommen).

Satz 2.2.

$$t - t_0 = d + e(m) + 365 \cdot (y - 1) + [c/4] + 24 \cdot c + [g/4] \tag{7}$$

Beweis. Sei zunächst $y > 0$ angenommen, dann ist wegen (6) und (5)

$$\begin{aligned} t - t_0 &= d + e(m) + \lambda(y,m) + \sum_{x=1}^{y-1} |\pi^{-1}(x)| = \\ &= d + e(m) + \lambda(y,m) + \sum_{x=1}^{y-1} (365 + \lambda(x)) = \\ &= d + e(m) + 365 \cdot (y-1) + \begin{cases} \sum_{x=1}^{y} \lambda(x) & m \geq 3 \\ \sum_{x=1}^{y-1} \lambda(x) & m = 1,2 \end{cases} \\ &= d + e(m) + 365 \cdot (y-1) + \sum_{x=1}^{100c+g} \lambda(x) = \\ &= d + e(m) + 365 \cdot (y-1) + [c/4] + 24 \cdot c + [g/4] \end{aligned}$$

wo wir $y = 100 \cdot c + g$ resp. $y - 1 = 100 \cdot c + g$ ersetzt haben, und die Relation (3) aus Lemma 2.1 verwendet haben.

Jetzt nehmen wir $y \leq 0$ an, wir sind in der Situation folgenden Bildes:

$$y-1 \qquad y \qquad \ldots \qquad -1 \qquad 0$$

und, indem wir ebenso die Fasern zählen, erhalten wir

$$\begin{aligned} t - t_0 &= d + e(m) + \lambda(y,m) - \sum_{x=y}^{0} |\pi^{-1}(x)| = \\ &= d + e(m) + \lambda(y,m) - \sum_{x=y}^{0} (365 + \lambda(x)) = \\ &= d + e(m) + 365 \cdot (y-1) - \sum_{x=1+100c+g}^{0} \lambda(x) = \\ &= d + e(m) + 365 \cdot (y-1) + [c/4] + 24 \cdot c + [g/4] \end{aligned}$$

wo wir $y = 100 \cdot c + g$ resp. $y - 1 = 100 \cdot c + g$ ersetzt haben und die Relation (4) vom Lemma 2.1 verwendeten. □

Korollar 2.3. *(Gauss' Formel)*

$$\omega(y,m,d) = d + e + f + g + [g/4] \tag{8}$$

wo $f = 5 \cdot r$ *wenn* $c = q \cdot 4 + r$, $0 \leq r \leq 3$ *und* $e = \begin{cases} e(m) & \text{für } m = 1,2 \\ e(m) - 1 & \text{für } m > 2 \end{cases}$

Beweis. Wir haben $\omega(t) = t - t_0 \mod 7$, folglich nach dem Satz

$$\omega(t) = d + e(m) + y - 1 + [c/4] + 3 \cdot c + [g/4]$$

Für $m = 1, 2$, sowie für $m > 2$ ist

$$e(m) + y - 1 = e(m) - 1 + y = e + 100 \cdot c + g$$

deshalb

$$\begin{aligned}\omega(t) &= d + e + 5 \cdot c + [c/4] + g + [g/4] = \\ &= d + e + 5 \cdot (q \cdot 4 + r) + q + g + [g/4] = \\ &= d + e + f + g + [g/4]\end{aligned}$$

□

2.4. **Julianische Tageszahl.** Die *Julianische Tageszahl* ist definiert als Anzahl vergangener Tage seit `-4713-11-24` (das ist *Julianisch* `-4712-01-01`), folglich gegeben durch $j(t) = t - t_1 = (t - t_0) - (t_1 - t_0)$, wo $t_1 = (-4713, 11, 24)$. Aus der Gleichung (7) ergibt sich

$$t_1 - t_0 = 24 + e(11) + 365 \cdot (-4713 - 1) + [-48/4] + 24 \cdot (-48) + [87/4] = -1721425$$

also

$$j(t) = 1721060 + d + e(m) + 365 \cdot y + [c/4] + 24 \cdot c + [g/4] \tag{9}$$

Literatur

[1] Carl Friedrich Gauss, *Werke*, Niedersächsische Staats- und Universitätsbibliothek, Göttingen, 1863. http://gdz.sub.uni-goettingen.de/no_cache/dms/load/toc/?IDDOC=38910.
[2] Maurice Kraitchik, *Mathematical Recreations*, George Allen & Unwill Ltd., London, 1955.

Anhang A. Die Formel in Gauss' Nachlass

Gauss hat seine Formel zur Findung des Wochentages nie publiziert. Sie erschien 1927 in [1, XI, 1.], S. 206, mit Bemerkungen von *Alfred Loewy* zur Herleitung.

A.1. **Den Wochentag des 1. Januar eines Jahres zu finden.**

Handschriftliche Eintragung in: Sammlung astronomischer Tafeln, unter Aufsicht der Kgl. Preußischen Akademie der Wissenschaften, I. Band, Berlin 1776. — 1798 von Gauss erworben.

Bezeichnet man den kleinsten positiven Rest einer Grösse A nach dem Modulus m durch $R : A \pmod m$, so lassen sich alle Vorschriften des Gregorianischen Kalenders auf folgende geschmeidige Art darstellen:

1.

Wenn man die Tage vom 1^{ten} Januar 1701 an zählt, d. i. diesen mit 1, den 2^{ten} mit 2, den 31^{ten} Dec. 1700 mit 0, den 30^{ten} mit -1 etc. bezeichnet, so ist der 1^{te} Januar in irgend einem Jahre A

$$\begin{aligned}= 1 + (A - 1701)365 &+ \frac{1}{4}\big((A - 1701) - R : (A - 1701) \bmod 4\big) \\ &- \frac{1}{100}\big((A - 1701) - R : (A - 1701) \bmod 100\big) \\ &+ \frac{1}{400}\big((A - 1601) - R : (A - 1601) \bmod 400\big)\end{aligned}$$

2.

Die Wochentage Sonntag, Montag, etc. mit $0, 1$ etc. bezeichnet, ist der 1^{te} Januar irgend eines Jahres, qua Wochentag,

$$\left.\begin{aligned}\equiv 6 + A &+ (2A + 5R : (A - 1) \bmod 4) \\ &+ (3A + 4R : (A - 1) \bmod 100) \\ &+ (A + 2 + 6R : (A - 1) \bmod 400)\end{aligned}\right\} \bmod 7.$$

Also von 1601 bis 2000

$$\equiv 6 + 6A + 5R : (A-1) \bmod 4 + 4R : (A-1) \bmod 100.$$

Allgemein

$$\equiv 1 + 5R : (A-1) \bmod 4 + 4R : (A-1) \bmod 100 + 6R : (A-1) \bmod 400.$$

A.2. **Vergleich.** Der Leser mag die Herleitung von *Loewy* in [1] nachlesen. Ich behaupte, dass die Formel in A.1 identisch ist zu (8). Sei $y = A - 1 = 100c + g$ und $c = 4q + r$, dann ist bei *Gauss* $R : y \bmod m = y - [y/m]m$, also

$$\begin{aligned} R : y \bmod 4 &= R : g \bmod 4 = g - 4[g/4] \\ R : y \bmod 100 &= g \\ R : y \bmod 400 &= 100r + g \end{aligned}$$

In $\mathbb{F}_7$ gerechnet ist der Wochentag des 1. Januars des Jahres A nach A.1

$$\begin{aligned} &= 1 + 5R : y \bmod 4 + 4R : y \bmod 100 + 6R : y \bmod 400 = \\ &= 1 + 5g - 20[g/4] + 4g + 12r + 6g = \\ &= 1 + g + [g/4] + 5r = \omega(A, 1, 1) \qquad \text{nach Formel (8).} \quad \square \end{aligned}$$

Anhang B. Das Programm `gauss.c`

Das Programm `gauss` zeigt an

- den Wochentag,
- das Gregorianische Datum,
- das Julianische Datum (`JC`),
- die Tageszahl im Jahr (`D#`),
- die Woche im Jahr (`W#`),
- die Julianische Tageszahl (`J#`),
- die Unix Tageszahl (`X#`) (Tage seit 1.1.1970)

Die Eingabe Syntax ist `datum [verschiebung]`, wobei die *verschiebung* irgendeine ganze Zahl sein kann und *datum* im ISO 8601 Format `Jahr-Monat-Tag` sein muß. Im Programm wird statt von *verschiebung* von *offset* geredet. Eine Woche geht von Montag bis Sonntag, ihre Nummerierung folgt ISO.

Wenn man mit Eingabebeispielen experimentiert, sieht man, dass `gauss` ein falsches Datum toleriert (und es korrigiert): Eingabe von `1999-2-29` wird mit `1999-03-01` beantwortet. Leere Eingabe beendet das Programm.

B.1. **Frühere Implementierungen.** Die Unterroutinen gehen auf eine Implementierung in REXX in den späten 1980ern im IBM VM/CMS Betriebssystem zurück. Sie wurden benutzt, um wöchentliche Übersichten von Fehlerstatistiken in einem Projekt zu generieren. Zu Beginn eines Schaltjahres (1996) habe ich sie alle in ein kleines REXX Programm (`gauss.rex`) gepackt und schrieb ein *Rationale* des Algorithmus und einen Beweis der Kalenderformel. Diese grobe Erläuterung von *Gauss'* Formel war auch der erste Beitrag auf meiner Homepage (Juni 1999).

Im März 2004 portierte ich das Programm in Perl und im März 2009 in C. Seit 2004 wird auch das Julianische Datum die *Julianische Tageszahl* ausgegeben.

Die Funktion `date` implementiert die Operation der ganzen Zahlen auf den Datumstagen und `offset` gibt zu einem Datum die Tageszahl im Jahr zurück. Die Funktion `leap` implementiert die Gregorianische Schaltjahr Regel und `weekday` gibt das Ergebnis von *Gauss'* Formel zurück. Ich weiche im Programm vom obigen Text ab und benutze als Nummerierung der Wochentage Montag = 0, ..., Sonntag = 6. Auf diese Weise ist die *Julianische Tageszahl* $\equiv$ zum Wochentag. Die Subroutine `julian` berechnet die Julianische Tageszahl.

Für positive y gilt in REXX `c = y % 100` (Division ganzer Zahlen), und der Rest ist `g = y // 100`. Dies ist falsch für negative y, wo REXX stattdessen `y//100 = -[-y/100]` ergibt. C leidet unter derselben falschen *Gaussklammer* wie die anderen Programmiersprachen: der Euklidische Algorithmus gibt die Formel $a = q \cdot b + r$ mit $q = [a/b]$ und $0 \leq r < b$ für positive b. Wenn a negativ ist, gibt die C Funktion a/b ein falsches Ergebnis, außer falls b Teiler von a ist. Die Routinen `quot` und `mod` korrigieren die Behandlung der *Gaussklammer* $[x]$ für negative Argumente (negative Jahre sind erlaubt).

Diese Implementierungen sind auf verschiedenen Plattformen ausgeführt worden:

- REXX unter VM, OS/2, DOS, Windows, Linux
- Perl unter Linux, AiX, Windows
- C unter Windows (XP, Vista), Linux

B.2. **Programm Liste** `gauss.c`. //

```
/* ------------------------------------------------------------------*

   Module:      gauss.c

   Description:

        This program explores Gauss' formula for weekday calculation
        in the proleptic Gregorian Calendar.

        For a detailed explanation see my calendar article:
        http://berndt-schwerdtfeger.de/cal/cal.pdf

        Input:  year-mm-dd [offset] (ISO 8601 Format)

        Output: Weekday, date, Julian Calendar date, day of the year,
                week of the year, Julian day number, Unix day number

   Subroutines: date, offset, weekday, julian, leap, quot, mod

   Sample: gauss 2010-01-0 +120
        Sun 2010-04-30, JC 2010-04-17, D# 120 W# 18 J# 2455317 X# 14729

   ------------------------------------------------------------------

   Copyright (C) 2010 Berndt E. Schwerdtfeger

   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.

   ------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
```

```
// set constants

char *wd[] = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};
int em[13] = {0,0,31,59,90,120,151,181,212,243,273,304,334};
const int JULIAN = 0;
const int GREGOR = 1;

// prototype statements for functions

void date(int,long,long*,int*,int*); // type, offset, year, month, day
long offset(long,int,int);           // = offset into year y
int  weekday(long,int,int);          // = {0|1|2|3|4|5|6}, 0 = Monday
long julian(long,int,int);           // = number of days since JC -4712/01/01
int  leap(int,long);                 // = 1 if year is a leap year
long quot(long,long);                // = [a/b] (corrected: works if a < 0)
long mod(long,long);                 // = a - quot(a,b)*b

// ------------------------------------------------------------------------
// main program
// ------------------------------------------------------------------------

int main(int argc, char *argv[]){

   if (argc == 1 || *argv[1] == '?'){
      printf("\ngauss $Revision: 1.13 $ $Date: 2010-12-03 09:10:07 $ \n");
      printf("-- Copyright (C) 2009-2010 Berndt E. Schwerdtfeger -- \n\n");
      printf("  Input:      year-mm-dd [offset] \n\n");
      printf("  Output:     Weekday, date, Julian Calendar date, day of the year,\n");
      printf("              week of the year, Julian day number, Unix day number\n\n");
      printf("  Sample: gauss 2010-01-00 +120\n");
      printf("  Sun 2010-04-30, JC 2010-04-17, D# 120 W# 18 J# 2455317 X# 14729\n\n");

     return EXIT_SUCCESS;               // end the program
   }                                    // argc > 1 here

   int  w,m,d,jm,jd;
   long y,jy,j,x,n=0;
   char line[80];

   sscanf(argv[1],"%ld-%d-%d",&y,&m,&d);

   if (argc == 3)
      sscanf(argv[2],"%ld",&n);         // get the offset
   else if (argc > 3) {
      printf("Too many parameters !\n");
      return EXIT_FAILURE;              // end the program
   }

   while(1)
   {
      n += offset(y,m,d);               // correct the offset
      date(GREGOR, n, &y, &m, &d);      // correct y, m, d
      j = julian(y,m,d);                // set Julian day number
      x = j - 2440588;                  // set Unix day number
      jy = -4712;                       // set initial julian year
      date(JULIAN, j+1, &jy, &jm, &jd); // correct jy, jm, jd
```

```
      n = offset(y,m,d);                // set offset in this year
      w = weekday(y,m,d);               // set weekday (Gauss)

      printf("%s %ld-%02d-%02d, JC %ld-%02d-%02d, ",wd[w],y,m,d,jy,jm,jd);
      printf("D# %03d W# %02d J# %ld X# %ld\n",n,(n-w+9)/7,j,x);
      fgets(line,sizeof line,stdin);    // read next
      if (line[0]=='\n')
         break;
      n = 0;
      sscanf(line,"%ld-%d-%d %ld",&y,&m,&d,&n);
   }                                    // end of while loop
   return EXIT_SUCCESS;                 // exit the program
};

// ------------------------------------------------------------------------
// subroutines: date, offset, weekday, julian, leap, quot, mod
// ------------------------------------------------------------------------
void date(int c, long n, long* y, int* m, int* d){

   int i;
   while (n > 365 + leap(c,*y)){        // if offset larger than # of
      n -= 365 + leap(c,*y);            // ... days in a year
      *y += 1;                          // ... find the correct year
   }

   while (n <= 0){                      // if offset is negative
      *y -= 1;                          // ... find the correct year
      n += 365 + leap(c,*y);            // ... and offset
   }

   i = leap(c,*y);                      // adjust for leap day
   *m = 12;
   while (n <= em[*m] + i){             // searching for the month
      *m -= 1;
      if (*m<3) i=0;
   }
   *d = n - em[*m] - i;                 // getting the day
}

// ------------------------------------------------------------------------
long offset(long y, int m, int d){

   int x = d + em[m];                   // offset into this year
   if (m > 2)
      x += leap(GREGOR,y);              // adjust for leap day
   return x;
}

// ------------------------------------------------------------------------
int weekday(long y, int m, int d){

   if (m < 3)
      y -= 1;

   long c = quot(y,100);
   int g = mod(y,100);
   int f = 5 * mod(c,4);
```

```
   int e = em[m];
   if (m > 2)
      e -= 1;
   return (-1 + d + e + f + g + g/4)%7; // Gauss' formula
}

// ------------------------------------------------------------------------
long julian(long y, int m, int d){

   long x = 0;
   if (m < 3) {
      y -= 1;
      x = 365;
   }
   long c = quot(y,100);
   int g = mod(y,100);
   x += em[m];
   return 1721060 + d + x + 365*y + quot(c,4) + 24*c + g/4;
}

// ------------------------------------------------------------------------
int leap(int c, long y){

   int i = 0;
   if (y%4 == 0)
      i = 1;
   if (c == GREGOR)
      if (y%100 == 0)                  // if century ..
         i -= (y%400 != 0);            // .. adjust
   return i;
}

// ------------------------------------------------------------------------
long quot(long a, long b){

   long x;
   x = a/b;
   if (a < 0 )                         // for negative numerator ..
      x -= (a%b != 0);                 // .. if remainder, subtract 1
   return x;
}

// ------------------------------------------------------------------------
long mod(long a,long b){
   return a - quot(a,b)*b;             // remainder always positive
}

//
```

Index

F

Frühlingspunkt . . . 3

G

Gauss Abbildung . . . 4, 6
Gregorianischer Kalender . . . 1, 3
Gregorianisches Datum . . . 9

I

immerwährende Kalender . . . 2

J

Jahr
  Schalt- . . . 2, 4
  Sonnen- . . . 3
Julianische Tageszahl . . . 1, 8, 9
Julianischer Kalender . . . 2, 3
Julianisches Datum . . . 9

K

Kalender
  Gregorianischer . . . 1, 3
  immerwährender . . . 2
  Julianischer . . . 2, 3
  proleptisch . . . 3

P

proleptisch . . . 3

S

Schaltjahre . . . 2, 4
Schaltsekunde . . . 3

U

Unix Tageszahl . . . 9

V

Verschiebe Operation . . . 6

W

Weltzeit . . . 3