ON CALENDAR FORMULAS

BERNDT E. SCHWERDTFEGER

For Ralph

Abstract. This paper deals with calendar formulas for calculating the day of the week and the *Julian* day number as used in Astronomy. It describes my implementation of these formulas in a C program.

Preface

Ten years ago I published an article on *Gauss'* calendar formula [3], which determines the day of the week of the 1^{st} of January of a year A as

$$w \equiv 5 \cdot (A-1) \bmod 4 + 4 \cdot (A-1) \bmod 100 + 6 \cdot (A-1) \bmod 400 \pmod 7$$

where $w = 0$ is *Monday*, $w = 1$ is *Tuesday*, ..., $w = 6$ is *Sunday* [2, XI, 1.].

Here I treat the *Julian* day number j introduced by *John Herschel* in 1849. The *Gauss* formula can be bypassed by $j(y, m, d)$ as $w \equiv j \pmod 7$.

Berlin, 4 June 2010
 version 1.3

1. Leap years through the times

In this article I use the ISO 8601 format for dates: `y-m-d`, in formulas written as (y, m, d). The use of integers for years signifies that I denote 1 BC = 0, etc.

The calendar currently used worldwide is the *Gregorian* calendar, which is derived from the Roman calendar as reformed by *Julius Caesar* in -45 and corrected by *Augustus (Octavian)* in -7. The *Julian* rule prescribes: every four years is a *leap* year. In the early years after the reform by mistake every third year was made a leap year and *Augustus* corrected it (leaving out several leap years). Since year 8 the rule was followed until 1582.

Then a reform by Pope *Gregory XIII* refined this rule by dropping the leap day in years divisible by 100, but not by 400. This adjustment was necessary to keep the *vernal equinox* near the 21^{st} of March. To bring the calendar back into synch the *Gregorian* reform also dropped ten days in October: the last day of the *Julian* calendar was Thursday, 4 October 1582 and was followed by the first day of the *Gregorian* calendar: Friday, 15 October 1582.

Here I treat the dates as if the *Julian* calendar had been in use since eternity until `1582-10-04`, (so called *proleptic* calendar). Keep in mind, though, this is historically not correct before *Caesar*'s time; also, the current numbering of the

2010 *Mathematics Subject Classification.* Primary 68-04; Secondary 68N15.
Key words and phrases. Gauss' calendar formula, Julian day number, intervals in congruence classes, Gauss' floor function.

years was invented by *Dionysius Exiguus* in 525 and adopted by the venerable *Bede* in 731 and thereafter only slowly promulgated.

Dates since `1582-10-15` are assumed to follow the *Gregorian* rules forever. The year 1582 was shorter than ordinary years by 10 days, it had only 355 days. Finally the dates `1582-10-5`, `1582-10-6`, ... `1582-10-14` are false, they never existed.

2. Gauss ansatz

In [2, XI, 1.] *Gauss* conceives a numbering of the days by starting at some particular date and linearly extending the numeration to the future. For the 1^{st} of January he writes down a simple formula, reflecting the *Gregorian* leap year rules in a supple way ("geschmeidige Art"). His note is reproduced below as appendix A.2.

We will generalize his approach to arbitrary dates (including the *Julian* era) and extending to the past. We will take a different starting point, which was proposed in 1849 by *John Herschel*: the *Julian* day number, starting at `-4712-1-1`. This is used in *Astronomy*, in particular, where it denotes *noon* of a day.

We consider the group operation of $\mathbf{Z}$ on the set $T \subset \mathbf{Z} \times \{1, \ldots, 12\} \times \{1, \ldots, 31\}$ of dates as in [3, §2.3 shifting dates]:

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

where $n + t$ is defined as t shifted by n days, earlier ($n < 0$) or later ($n > 0$) in time. For a year $a \in \mathbf{Z}$ we let $T_a \subset T$ be the finite set of dates of that year: $t = (y, m, d) \in T_a \Leftrightarrow y = a$. If a is a *leap* year, the cardinal is $|T_a| = 366$, otherwise $|T_a| = 365$, except for $|T_{1582}| = 355$. We can also write $|T_a| = 365 + \ell(a)$ for all $a \neq 1582$, where $\ell(a) = 1$ for a *leap* year and $\ell(a) = 0$ otherwise.

We have obvious maps (*offset*) $x : T_a \longrightarrow \{1, \ldots, 366\}$ counting the days of a year from the first to the last. When $e(m)$ is the number of days in previous months

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

then for $t \in T_a$, $a \neq 1582$, $x(t) = \begin{cases} d + e(m) & \text{for } m \leq 2 \\ d + e(m) + \ell(a) & \text{for } m > 2 \end{cases}$

Let $z : T \longrightarrow \mathbf{Z}$ be any *linear* map, such that $z(n + t) = n + z(t)$. There is *essentially* one, only depending on the *origin* t_0 such that $z(t_0) = 0$. The choice $t_0 = (-4712, 1, 1)$ will be denoted by j, the *Julian* day number.

For two dates $t, t' \in T$, $t = (a, m, d), t' = (a', m', d')$ and $t < t'$ we have

$$z(t') - z(t) = x(t') - x(t) + \sum_{y=a}^{a'-1} |T_y|$$

and by the above (assuming $y \neq 1582$) we have

$$\sum_{y=a}^{a'-1} |T_y| = (a' - a) \cdot 365 + \sum_{y=a}^{a'-1} \ell(y)$$

The last sum, the number of leap years in the interval $[a, a' - 1]$, is written in the *supple way* by *Gauss* for $a \equiv 1 \bmod 400$

$$\sum_{y=a}^{a'-1} \ell(y) = \left[\frac{a' - a}{4}\right] - \left[\frac{a' - a}{100}\right] + \left[\frac{a' - a}{400}\right]$$

which simply reflects the *Gregorian* leap year rule. Hence

$$z(t') - z(t) = x(t') - x(t) + (a' - a) \cdot 365 + \left[\frac{a' - a}{4}\right] - \left[\frac{a' - a}{100}\right] + \left[\frac{a' - a}{400}\right]$$

The interesting point now is, that this formula is not only true for $t < t'$ but for $t > t'$ as well, as will be derived in the next section.

3. Counting intervals in congruence classes

We prove some simple properties of the *Gauss floor* function $[x]$, which he introduced in 1808 (see [1], p. 459).

Lemma 3.1. *For $m, n \in \mathbf{Z}$, $m > 0$ we have*

$$\left[\frac{-1 - n}{m}\right] = -1 - \left[\frac{n}{m}\right]$$

Proof. We write $n = q \cdot m + r$ with $0 \leq r < m$ such that $q = [n/m]$. Then $0 \leq m - r - 1 < m$ and as $-1 - n = -(q + 1) \cdot m + (m - r - 1)$ we have $[(-1 - n)/m] = -q - 1$. □

Lemma 3.2. *For $a, n, m \in \mathbf{Z}$, $n > 0$, $m > 0$ with $a \equiv 1 \pmod{m}$ we have*

$$|[a, a + n - 1] \cap m\mathbf{Z}| = \left[\frac{n}{m}\right] \tag{1}$$

$$|[a - n, a - 1] \cap m\mathbf{Z}| = -\left[\frac{-n}{m}\right] \tag{2}$$

Proof. For (1): let $n = q \cdot m + r$ with $0 \leq r < m$ and $a = 1 + k \cdot m$. The set

$$[a, a + n - 1] \cap m\mathbf{Z} = \{a - 1 + m, \ldots, a - 1 + q \cdot m = a + n - 1 - r\}$$

has $q = [n/m]$ elements.

For (2): let $n - 1 = q \cdot m + r$ with $0 \leq r < m$ and $a = 1 + k \cdot m$. Then $a - n = k \cdot m - q \cdot m - r = (k - q) \cdot m - r$ and the set

$$[a - n, a - 1] \cap m\mathbf{Z} = \{a - n + r, a - n + r + m, \ldots, a - n + r + q \cdot m = a - 1\}$$

contains $1 + q$ elements, which by lemma 3.1 is $= 1 + [(n - 1)/m] = -[-n/m]$. □

We will apply this to the *leap* years in the *Julian* resp. *Gregorian* era in the interval $a \leq y < a'$ for $a < a'$ and in the interval $a' \leq y < a$ for $a' < a$.

In the *Julian* case this set is $[a, a' - 1] \cap 4\mathbf{Z}$ for $a < a'$ and $[a', a - 1] \cap 4\mathbf{Z}$ for $a' < a$.

In the *Gregorian* case this set is $[a, a' - 1] \cap 4\mathbf{Z} \setminus [a, a' - 1] \cap 100\mathbf{Z} \cup [a, a' - 1] \cap 400\mathbf{Z}$ for $a < a'$ and $[a', a - 1] \cap 4\mathbf{Z} \setminus [a', a - 1] \cap 100\mathbf{Z} \cup [a', a - 1] \cap 400\mathbf{Z}$ for $a' < a$. Remark that the union is *disjoint*.

Corollary 3.3. *Let $t, t' \in T$, $t = (a, m, d), t' = (a', m', d')$.*

In the Julian *era: $t, t' \leq$ 1582-10-04, for $a \equiv 1 \pmod{4}$*

$$z(t') - z(t) = x(t') - x(t) + (a' - a) \cdot 365 + \left[\frac{a' - a}{4}\right] \tag{3}$$

In the Gregorian *era: $t, t' \geq$ 1582-10-15, for $a \equiv 1 \pmod{400}$*

$$z(t') - z(t) = x(t') - x(t) + (a' - a) \cdot 365 + \left[\frac{a' - a}{4}\right] - \left[\frac{a' - a}{100}\right] + \left[\frac{a' - a}{400}\right] \tag{4}$$

Proof. The corollary is clear for $a < a'$. For $a' < a$ we remark that $z(t) - z(t') = x(t) - x(t') + (a - a') \cdot 365 + \sum_{y=a'}^{a-1} \ell(y)$ and applying Lemma 3.2 (2) with $n = a - a'$ we get

$$z(t) - z(t') = x(t) - x(t') + (a - a') \cdot 365 - \left[\frac{a' - a}{4}\right] + \left[\frac{a' - a}{100}\right] - \left[\frac{a' - a}{400}\right]$$

with the last two terms only in the *Gregorian* case. Multiplying by -1 gives the asserted formula. □

4. Programming notes

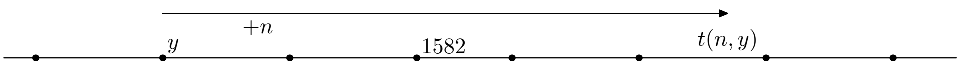
In this section I explain some points of the program `cal.c` in appendix A.1.

There are four functions: `main`, `date`, `leap` and `quot`, the latter three reused from `gauss` [3]. `leap` implements the *leap* function $\ell(y)$ and `quot` the *floor* function $[n/m]$ for $m > 0$. Function `date` needs some explanation.

4.1. Function `date`.

It is called with an *offset* n and by setting y to a *starting* year. Think of it as a *linear* function $t : \mathbf{Z} \times \mathbf{Z} \to T$, $t(n, y) = (a, m, d)$ with $t(k + n, y) = k + t(n, y)$. It is understood that $n = 1$ corresponds to the first day of the year y, $t(1, y) = (y, 1, 1)$, hence offset $n = 0$ corresponds to New Year's Eve, $t(0, y) = (y - 1, 12, 31)$.

Another special point is that I have *not* implemented the exceptional year 1582, which only has 355 days. That is, I have to avoid situations like the following, crossing the critical year. In the figure I start in the *Julian* era and add an offset getting me to the *Gregorian* era.



Similarly, situations have to be avoided starting in the *Gregorian* era and taking us back to the *Julian* era by a *negative* offset.

4.2. Some *Julian* day numbers.

We need to understand some of the constants in the `main` function, gathered in this table

$$\begin{array}{rrr} j(0,12,31) = 1,721,423 & j(1581,12,31) = 2,298,883 & j(1582,10,4) = 2,299,160 \\ j(1582,10,15) = 2,299,161 & j(1582,12,31) = 2,299,238 & j(2000,12,31) = 2,451,910 \end{array}$$

Using (3) with $t' = (-4712, 1, 1)$ and $t = (1, 1, 1)$ we get

$$j(-4712, 1, 1) - j(1, 1, 1) = -4713 \cdot 365 + \left[\frac{-4713}{4}\right] = -1720245 - 1179 = -1721424$$

As by definition $j(-4712, 1, 1) = 0$ we obtain $j(1, 1, 1) = 1721424$.

The next value is $j(1582, 1, 1) = j(1, 1, 1) + 1581 \cdot 365 + [1581/4] = 1721424 + 577065 + 395 = 2298884$. $j(1582, 10, 4) = j(1581, 12, 31) + 4 + e(10) = 2298883 + 4 + 273 = 2299160$. $j(1582, 12, 31) = j(1581, 12, 31) + 355 = 2299238$. Using (4) we have $j(2001, 1, 1) = j(1583, 1, 1) + 418 \cdot 365 + 1 + [418/4] - [418/100] + [418/400] = 2299239 + 152570 + 1 + 104 - 4 + 1 = 2451911$.

4.3. **The `main` function.** The first half of the `main` function does housekeeping and parameter checking.

The following lines

```
if (y < 1582 || y == 1582 && m*100+d < 1005) // Julian era
   j = 1721423 + d + e[m] + (y-1)*365 + quot(y-1,4);
else {                                       // Gregorian era
   x = y - 2001;
   j = 2451910 + d + e[m] + x*365 + quot(x,4) - quot(x,100) + quot(x,400);
}
```

set the *Julian* day number $j = j(t)$ depending on the era, making use of the formulas (3) resp. (4), with the values just calculated.

The next two lines adjust for leap year and add the *offset* from the command line.

```
   if (m>2) j+=leap(y);                       // correction for leap year
   j += n;                                    // add offset
```

Eventually, the correct dates are calculated by calling function `date` either for a *Julian* date starting with 1582, or for a *Gregorian* date starting with year 1583 – to avoid the crossing of the critical year described above.

```
if (j < 2299161) {                     // Julian era
   y=1582;
   date(j-2298883,&y,&m,&d);           // calculate the date
}
else {                                 // Gregorian era
   y=1583;
   date(j-2299238,&y,&m,&d);           // calculate the date
}
```

The remainder of the `main` function is obvious.

Appendix A.

A.1. **The program `cal.c`.** The input syntax is `date [offset]`, where *date* is in ISO 8601 format `year-mm-dd` and *offset* can be any integer. Weeks run from Monday to Sunday, its numbering follows ISO.

The program checks for correct dates under the assumption that

- all dates previous to `1582-10-04` are dates in the proleptic *Julian* calendar
- all dates past `1582-10-15` are dates in the proleptic *Gregorian* calendar
- the dates in the range `1582-10-05` to `1582-10-14` are rejected as incorrect

The program `cal` displays

- the day of the week (Mon, Tue, Wed, Thu, Fri, Sat, Sun)
- the date (*Julian* for $\leq$ `1582-10-04`, *Gregorian* for $\geq$ `1582-10-15`)
- the Julian day number (`J#`)
- the day of the year (`D#`)
- the week of the year (`W#`)

The program operates correctly in the range of dates `-5,877,908-3-15` to `+5,874,898-6-3` corresponding to values of j in the range $-2^{31} - 1 + 2,298,883 \leq j \leq 2^{31} - 1 = 2,147,483,647$. This range originates from the overflow in the type `long int` of j.

Program listing `cal.c`. *//*

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

   Module:      cal.c

   Description: Calendar formulas

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

        Output: day of the week, date, Julian day number,
                day of the year, week of the year

   Example: cal 1777-04-30 +84005
            Mon 2007-04-30 J# 2454221 D# 120 W# 18

   Documentation: http://berndt-schwerdtfeger.de/cal/calj.pdf

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

   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  e[13] = {0,0,31,59,90,120,151,181,212,243,273,304,334};

// function prototypes

void date(long,long*,int*,int*);     // offset, year, month, day
int  leap(long);                     // = 1 if year is a leap year
long quot(long,long);                // = [a/b] (corrected: works if a < 0)

// ---------------------------------------------------------------------
// main function
// ---------------------------------------------------------------------

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

   if (argc == 1 || *argv[1] == '?'){
```

```
      printf("\ncal $Revision: 105 $, $Date: 2011-04-05 20:46:18 +0200 (Di, 05 Apr 2011) $ \n");
      printf("-- Copyright (C) 2010 Berndt E. Schwerdtfeger -- \n\n");
      printf("Input:  year-mm-dd [offset] \n\n");
      printf("Output: {Mon|..|Sun} year-mm-dd J# jjjjjjj D# nnn W# ww\n\n");
      printf("Example: cal 1777-04-30 +84005 \n");
      printf("Mon 2007-04-30 J# 2454221 D# 120 W# 18\n\n");
      return EXIT_SUCCESS;              // end the program
   }                                    // argc > 1 here
   int  w,m,d;
   long x,y,j,n=0;

   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
   }
   if (y==1582 && m==10 && d>4 && d<15) {
      printf("This date is invalid, it did never exist !\n");
      return EXIT_FAILURE;              // end the program
   }
   if (y < 1582 || y == 1582 && m*100+d < 1005) // Julian era
      j = 1721423 + d + e[m] + (y-1)*365 + quot(y-1,4);
   else {                               // Gregorian era
      x = y - 2001;
      j = 2451910 + d + e[m] + x*365 + quot(x,4) - quot(x,100) + quot(x,400);
   }
   if (m>2) j+=leap(y);                 // correction for leap year
   j += n;                              // add offset
   if (j < 2299161) {                   // Julian era
      y=1582;
      date(j-2298883,&y,&m,&d);         // calculate the date
   }
   else {                               // Gregorian era
      y=1583;
      date(j-2299238,&y,&m,&d);         // calculate the date
   }
   n = d + e[m];                        // offset into this year
   if (m > 2)
      n += leap(y);                     // adjust for leap day
   if (y==1582 && n>278) n-=10;         // ten days dropped in 1582
   w = j%7;                             // set weekday
   if (w<0) w+=7;
   printf("%s %ld-%02d-%02d J# %ld ", wd[w],y,m,d,j);
   printf("D# %03d W# %02d\n", n,(n-w+9)/7);
   return EXIT_SUCCESS;                 // exit the program
};

// ------------------------------------------------------------------------
// function definitions
// ------------------------------------------------------------------------

void date(long n, long* y, int* m, int* d){

   int i;
```

```
   while (n > 365 + leap(*y)){          // if offset larger than # of
      n -= 365 + leap(*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(*y);              // ... and offset
   }
   i = leap(*y);                        // adjust for leap day
   *m = 12;
   while (n <= e[*m] + i){              // searching for the month
      *m -= 1;
      if (*m<3) i=0;
   }
   *d = n - e[*m] - i;                  // setting the day
}

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

   int i = 0;
   if (y%4 == 0)
      i = 1;
   if (y > 1582 && y%100 == 0 && y%400 != 0)
      i = 0;
   return i;
}

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

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

//
```

A.2. **Gauss' calendar formula in his Nachlass.** *Gauss* did not publish his formula for finding the day of the week. It appeared posthumously in his *Werke* in 1927 [2, XI, p. 206]. The *bracket* notation for the floor function $[x]$ was not yet used, *Gauss* introduced it in 1808.

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

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

References

[1] Carl Friedrich Gauss, *Untersuchungen über höhere Arithmetik*, 2nd (reprinted), AMS Chelsea Publishing, Providence, 1981.

[2] ———, *Werke* (1863), available at http://gdz.sub.uni-goettingen.de/dms/load/toc/?PID=PPN235957348.

[3] Berndt E. Schwerdtfeger, *Gauss' calendar formula for the day of the week* (1999), available at http://berndt-schwerdtfeger.de/wp-content/uploads/pdf/cal.pdf.