Finding the date of Easter with Gauss's Algorithm
An implementation in Python

Konstantinos Thanos

Mathematician, MSc

03/03/2020

In 1800, mathematician Carl Friedrich Gauss presented his algorithm for finding the date of Easter. This calculation considered to be the most important computation of the age and it has the name *Computus*. Computus determines Easter by the first Sunday on or after spring Equinox on 21 of March.

The Metonic Cycle

Meton of Athens was a greek mathematician, astronomer and engineer who lived in the 5th century BC. Meton observed that a period of 19 solar (tropical) years is almost equal to 235 synodic months. A synodic month is defined as the time needed between two consecutive moon phases (new or full moon) and is equal to 29.530588 days. A tropical year is the time that Sun takes to return to the same position in the cycle of seasons and it is equal to 365.24219 days.

$$19 \text{ tropical years} = 6939.602 \text{ days}$$
$$235 \text{ synodic months} = 6939.688 \text{ days}$$

The Golden Number

The Golden Number is the number that points to the position of each year in the Metonic Cycle. It has been decided that this golden number should start from a year that has full moon on 1st of January. Chronologists found that a year like this was the first AD year and gave the golden number 2 to this year. The golden number of a year can be found by dividing the year by 19, and adding 1 to the remainder of the division.

$$gn(year) = year\ mod\ 19 + 1$$

From now on we will be using the symbolism $[x]_n = x\ mod\ n$ to denote the remainder of the devision x divided by n and also we will represent a year by "Y". So we can rewrite the golden number definition as

$$gn(Y) = [Y]_{19} + 1$$

```
def golden_num(year):
  return year%19 + 1
```

Golden Number calculation

```
# Examples
years = [1955, 1989, 2004, 2020]
for year in years:
    print(”ga(%d) : %d”%(year, golden_num(year)))
```

Examples

```
ga(1955) : 18
ga(1989) : 14
ga(2004) : 10
ga(2020) : 7
```

Epact (epacti)

Epact or Julian comes from the greek word epacti (επακτή). We use the name *epacti* of a year to describe the age of the Moon in days on 31 of December of the previous year. The epacti can have a value from 0 to 29. To find the epacti (from now on ”E”) of a year we use the formula

$$E(Y) = [11 \cdot [Y]_{19} + 8]_{30}$$

From the equation that we use to find the golden number of a year we can see the relation between golden number and epacti from the new formula

$$E(Y) = [11 \cdot (gn(Y) - 1) + 8]_{30}$$

or equivalently

$$E(Y) = [11 \cdot gn(Y) - 3]_{30}$$

```
def epacti(year):
    return (11*gn(year) - 3)%30
```

Epacti calculation

```
# Examples
years = [1955, 1989, 2004, 2020]
for year in years:
    print(”Epacti(%d) : %d”%(year, epacti(year)))
```

Examples

```
Epacti(1955) : 15
Epacti(1989) : 1
Epacti(2004) : 17
Epacti(2020) : 14
```

The values that the epacti of a year can take are in the next table with relation to the golden number of the year.

```python
print ('  golden number  ' + '||' '\t' + 'epacti')
print('==================================')
for gd in range(1, 20):
    print('\t', gd, '\t', '||', '\t', (11*gd-3)%30)
```

Relation between golden number and epacti

```
golden number  ||  epacti
============================
        1          ||      8
        2          ||      19
        3          ||      0
        4          ||      11
        5          ||      22
        6          ||      3
        7          ||      14
        8          ||      25
        9          ||      6
       10          ||      17
       11          ||      28
       12          ||      9
       13          ||      20
       14          ||      1
       15          ||      12
       16          ||      23
       17          ||      4
       18          ||      15
       19          ||      26
```

Proposition P1

If $R = [Y]_{19}$ and $E(Y) = [11 \cdot [Y]_{19} + 8]_{30}$ then

$$[19 \cdot R + 16]_{30} = 24 - [11 \cdot R + 8]_{30} = 24 - E(Y)$$

Proof

$$\begin{aligned} [19 \cdot R + 16]_{30} &= \big[[19 \cdot R]_{30} + [16]_{30}\big]_{30} \\ &= \big[[-11 \cdot R]_{30} + 16\big]_{30} \\ &= [-11 \cdot R + 24 - 8]_{30} \\ &= [24 - (11 \cdot R + 8)]_{30} \\ &= \big[[24]_{30} - [11 \cdot R + 8]_{30}\big]_{30} \\ &= 24 - [11 \cdot R + 8]_{30} \end{aligned}$$

Pascha Full Moon date

Pascha Full Moon date is the ecclesiastical full moon date on or after 21 of March. As we saw before the age of moon on 31 of December of the previous year will be

$$E(Y) = [11 \cdot [Y]_{19} + 8]_{30}$$

From 1 of January to 30 of March there are about $31 + 28,25 + 30 = 89,25$ days (every 4 years one additional day) and the age of moon is $E(Y)$ days. On the 31/03/year Moon will be $E(Y) + 1$ days of age. So the first day of Moon is on $31 - E(Y)$ of March and the full moon will be 13 days later on $44 - E(Y)$ of March with the Julian Calendar. So in the Gregorian calendar which we are using today the date of full Moon will be on $44 - E(Y) + 13$ in days of March.

- If $E(Y) \leq 23$ ($44 - E(Y) \geq 21$) then we have Pascha Full Moon on

$$\begin{aligned} 44 - E(Y) + 13 &= 57 - E(Y) \text{ of March} \\ &= 57 - [11 \cdot [Y]_{19} + 8]_{30} - 31 \text{ of April} \\ &= 26 - [11 \cdot [Y]_{19} + 8]_{30} \\ \text{(by P1)} &= 26 + [19 \cdot [Y]_{19} + 16]_{30} - 24 \\ &= 2 + [19 \cdot [Y]_{19} + 16]_{30} \end{aligned}$$

- If $E(Y) > 23$ ($44 - E(Y) < 21$) then this full Moon is not Pascha Full Moon so in this case we should wait the next Pascha Full Moon after 30 days on

$$\begin{aligned} 44 - E(Y) + 30 &= 74 - E(Y) \text{ of March} \\ &= 74 - [11 \cdot [Y]_{19} + 8]_{30} - 31 \text{ of April} \\ &= 56 - [11 \cdot [Y]_{19} + 8]_{30} \\ &= 56 + [19 \cdot [Y]_{19} + 16]_{30} - 24 \\ &= 32 + [19 \cdot [Y]_{19} + 16]_{30} \\ ([32]_{30} = 2) &= 2 + [19 \cdot [Y]_{19} + 16]_{30} \end{aligned}$$

So in both cases ($E(Y) \leq 23$ and $E(Y) > 23$) we found that the date of Pascha Full Moon is

$$\underline{\text{Pascha Full Moon date}} : 2 + [19 \cdot [Y]_{19} + 16]_{30} \text{ of April}$$

Orthodox Eeaster Algorithm

In 1800 mathematician Carl Friedrich Gauss presented his algorithm for finding the date of Easter of a year. Next we can see the steps (1 step per calculation) of his algorithm with some explanation on each step.

Step 1 : $a = year\ mod\ 19 = [Y]_{19}$

Determines year’s position inside the Metonic cycle.

Step 2 : $b = year\ mod\ 4 = [Y]_4$

Counts the days which correspond to leap years. Every four years one additional day.

Step 3 : $c = year\ mod\ 7 = [Y]_7$

Deals with the fact that a non-leap year is 1 day longer than 52 weeks.

Step 4 : $d = (19 \cdot a + 15)\ mod\ 30 = [19 \cdot a + 15]_{30}$

This result defines the number of days that need to be added on 21st of March in order to find the date of Pascha Full Moon.

Step 5 : $e = (2 \cdot b + 4 \cdot c + 6 \cdot d + 6)\ mod\ 7 = [2 \cdot b + 4 \cdot c + 6 \cdot d + 6]_7$

This result defines the number of days until the first Sunday after Pascha Full Moon date. This Sunday is Easter Sunday.

Step 6 : $easter_{date} = d + e + 4$

Calculates the date of Orthodox Easter.

```
def orth_easter_date(year):
    a = year%19
    b = year%4
    c = year%7
    d = (19*a + 15)%30
    e = (2*b + 4*c + 6*d + 6)%7
    easter = d + e + 4
    month = ”April”
    if easter > 30: # If date > 30 of April we should move to May
        easter -= 30
        month = ”May”
    result = str(easter) + ” ” + month
    return result
```

Gauss Algorithm for Orthodox Easter date calculation

```
# Examples
# Find the dates of Orthodox Easter for years in range 2000-2100
print(”Orthodox Easter dates for years 2000-2100”)
for year in range(2000,2101):
    print(str(year) + ” : ” + orth_easter_date(year))
```

Examples

Orthodox Easter dates for years 2000-2100

2000 : 30 April
2001 : 15 April
2002 : 5 May
2003 : 27 April
2004 : 11 April
2005 : 1 May
2006 : 23 April
2007 : 8 April
2008 : 27 April
2009 : 19 April
2010 : 4 April
2011 : 24 April
2012 : 15 April
2013 : 5 May
2014 : 20 April
2015 : 12 April
2016 : 1 May
2017 : 16 April
2018 : 8 April
2019 : 28 April
2020 : 19 April
2021 : 2 May
2022 : 24 April
2023 : 16 April
2024 : 5 May
2025 : 20 April
2026 : 12 April
2027 : 2 May
2028 : 16 April
2029 : 8 April
2030 : 28 April
2031 : 13 April
2032 : 2 May
2033 : 24 April
2034 : 9 April
2035 : 29 April
2036 : 20 April
2037 : 5 April
2038 : 25 April
2039 : 17 April
2040 : 6 May
2041 : 21 April
2042 : 13 April
2043 : 3 May
2044 : 24 April
2045 : 9 April
2046 : 29 April
2047 : 21 April
2048 : 5 April
2049 : 25 April
2050 : 17 April
2051 : 7 May
2052 : 21 April
2053 : 13 April
2054 : 3 May
2055 : 18 April
2056 : 9 April
2057 : 29 April
2058 : 14 April
2059 : 4 May
2060 : 25 April
2061 : 10 April
2062 : 30 April
2063 : 22 April
2064 : 13 April
2065 : 26 April
2066 : 18 April
2067 : 10 April
2068 : 29 April
2069 : 14 April
2070 : 4 May
2071 : 19 April
2072 : 10 April
2073 : 30 April
2074 : 22 April
2075 : 7 April
2076 : 26 April
2077 : 18 April
2078 : 8 May
2079 : 23 April
2080 : 14 April
2081 : 4 May
2082 : 19 April
2083 : 11 April
2084 : 30 April
2085 : 15 April
2086 : 7 April
2087 : 27 April
2088 : 18 April
2089 : 1 May
2090 : 23 April
2091 : 8 April
2092 : 27 April
2093 : 19 April
2094 : 11 April
2095 : 24 April
2096 : 15 April
2097 : 5 May
2098 : 27 April
2099 : 12 April
2100 : 1 May

Catholic Eeaster Algorithm

```python
def cath_easter_date(year):
    a = year%19
    b = year%4
    c = year%7
    d = (19*a + 24)%30
    e = (2*b + 4*c + 6*d + 5)%7
    if (d+e)<=9:
        easter = 22 + d + e
        month = ”March”
    else:
        easter = d + e - 9
        month = ”April”
    result = str(easter) + ” ” + month
    return result
```

Gauss Algorithm for Catholic Easter date calculation

```python
# Examples
# Find the dates of Catholic Easter for years in range 2000-2100
print(”Catholic Easter dates for years 2000-2100”)
for year in range(2000,2101):
    print(str(year) + ” : ” + cath_easter_date(year))
```

Examples

Catholic Easter dates for years 2000-2100

2000 : 23 April
2001 : 15 April
2002 : 31 March
2003 : 20 April
2004 : 11 April
2005 : 27 March
2006 : 16 April
2007 : 8 April
2008 : 23 March
2009 : 12 April
2010 : 4 April
2011 : 24 April
2012 : 8 April
2013 : 31 March
2014 : 20 April
2015 : 5 April
2016 : 27 March
2017 : 16 April
2018 : 1 April
2019 : 21 April
2020 : 12 April
2021 : 4 April
2022 : 17 April
2023 : 9 April
2024 : 31 March
2025 : 20 April
2026 : 5 April
2027 : 28 March
2028 : 16 April
2029 : 1 April
2030 : 21 April
2031 : 13 April
2032 : 28 March
2033 : 17 April
2034 : 9 April
2035 : 25 March
2036 : 13 April
2037 : 5 April
2038 : 25 April
2039 : 10 April
2040 : 1 April
2041 : 21 April
2042 : 6 April
2043 : 29 March
2044 : 17 April
2045 : 9 April
2046 : 25 March
2047 : 14 April
2048 : 5 April
2049 : 25 April
2050 : 10 April
2051 : 2 April
2052 : 21 April
2053 : 6 April
2054 : 29 March
2055 : 18 April
2056 : 2 April
2057 : 22 April
2058 : 14 April
2059 : 30 March
2060 : 18 April
2061 : 10 April
2062 : 26 March
2063 : 15 April
2064 : 6 April
2065 : 29 March
2066 : 11 April
2067 : 3 April
2068 : 22 April
2069 : 14 April
2070 : 30 March
2071 : 19 April
2072 : 10 April
2073 : 26 March
2074 : 15 April
2075 : 7 April
2076 : 26 April
2077 : 11 April
2078 : 3 April
2079 : 23 April
2080 : 7 April
2081 : 30 March
2082 : 19 April
2083 : 4 April
2084 : 26 March
2085 : 15 April
2086 : 31 March
2087 : 20 April
2088 : 11 April
2089 : 3 April
2090 : 16 April
2091 : 8 April
2092 : 30 March
2093 : 12 April
2094 : 4 April
2095 : 24 April
2096 : 15 April
2097 : 31 March
2098 : 20 April
2099 : 12 April
2100 : 27 March

Note : Some instructions on how to install and use colorama library on **Ubuntu 18.04** system. We are going to use this library to highlight with yellow color the years that have the same Orthodox and Catholic date.

```
# Update package list
$ sudo apt update
# Install pip for Python 3
$ sudo apt install python3-pip
# Verify installation by checking pip version
$ pip3 --version
# Install colorama using pip3
$ sudo pip3 install colorama
```

Installation instructions for colorama

In the following Python code we will print both Orthodox and Catholic Easter dates for years 2000-2100 and will mark the dates which are the same for both calendars.

```
from colorama import Fore, Back, Style
print(”Year \t Orthodox \t Catholic”)
for year in range(2000, 2101):
    blank = ’ ’
    print(((Back.YELLOW + str(year)) if orth_easter_date(year)==
            cath_easter_date(year) else year),(Style.RESET_ALL),
            ”\t”, orth_easter_date(year),(14 - len(orth_easter_date(year)))
            *blank, cath_easter_date(year))
```

Dates for Orthodox and Catholic Easter

Year	Orthodox	Catholic
2000	30 April	23 April
2001	15 April	15 April
2002	5 May	31 March
2003	27 April	20 April
2004	11 April	11 April
2005	1 May	27 March
2006	23 April	16 April
2007	8 April	8 April
2008	27 April	23 March
2009	19 April	12 April
2010	4 April	4 April
2011	24 April	24 April
2012	15 April	8 April
2013	5 May	31 March
2014	20 April	20 April
2015	12 April	5 April
2016	1 May	27 March
2017	16 April	16 April
2018	8 April	1 April
2019	28 April	21 April
2020	19 April	12 April
2021	2 May	4 April
2022	24 April	17 April
2023	16 April	9 April
2024	5 May	31 March
2025	20 April	20 April
2026	12 April	5 April
2027	2 May	28 March
2028	16 April	16 April
2029	8 April	1 April
2030	28 April	21 April
2031	13 April	13 April
2032	2 May	28 March
2033	24 April	17 April
2034	9 April	9 April
2035	29 April	25 March
2036	20 April	13 April
2037	5 April	5 April
2038	25 April	25 April
2039	17 April	10 April
2040	6 May	1 April
2041	21 April	21 April
2042	13 April	6 April
2043	3 May	29 March
2044	24 April	17 April
2045	9 April	9 April
2046	29 April	25 March
2047	21 April	14 April
2048	5 April	5 April
2049	25 April	25 April
2050	17 April	10 April
2051	7 May	2 April
2052	21 April	21 April
2053	13 April	6 April
2054	3 May	29 March
2055	18 April	18 April
2056	9 April	2 April
2057	29 April	22 April
2058	14 April	14 April
2059	4 May	30 March
2060	25 April	18 April
2061	10 April	10 April
2062	30 April	26 March
2063	22 April	15 April
2064	13 April	6 April
2065	26 April	29 March
2066	18 April	11 April
2067	10 April	3 April
2068	29 April	22 April
2069	14 April	14 April
2070	4 May	30 March
2071	19 April	19 April
2072	10 April	10 April
2073	30 April	26 March
2074	22 April	15 April
2075	7 April	7 April
2076	26 April	26 April
2077	18 April	11 April
2078	8 May	3 April
2079	23 April	23 April
2080	14 April	7 April
2081	4 May	30 March
2082	19 April	19 April
2083	11 April	4 April
2084	30 April	26 March
2085	15 April	15 April
2086	7 April	31 March
2087	27 April	20 April
2088	18 April	11 April
2089	1 May	3 April
2090	23 April	16 April
2091	8 April	8 April
2092	27 April	30 March
2093	19 April	12 April
2094	11 April	4 April
2095	24 April	24 April
2096	15 April	15 April
2097	5 May	31 March
2098	27 April	20 April
2099	12 April	12 April
2100	1 May	27 March

References

1. https://en.wikipedia.org/w/index.php?title=Computus&oldid=907024414

2. https://en.wikipedia.org/wiki/Metonic_cycle

3. https://en.wikipedia.org/wiki/Golden_number_(time)

4. https://www.maa.org/press/periodicals/convergence/servois-1813-perpetual-calendar-with-an-english-translation-gauss-calculation-for-the-date-of-easter

5. http://karamatskos.blogspot.com/2012/04/blog-post_07.html

6. http://users.sch.gr/geokasap/site/images/pdf/pasxa.pdf

7. Dimitris I. Bounakis. The date of Easter.