Developed by Jonathan Scheffe
7/10/2020, University of Florida

Solar Time and Solar Time Python Calculator

Solar Time

Solar time is used in all sun-angle relationships. It is based on the apparent angular motion of the sun across the sky, with solar noon the time the sun crosses the meridian of the observer. Two corrections are needed to convert from standard time. 1) *Correction for difference in longitude between observer's meridian and the meridian at which local standard time is based*. 2) *Correction from the equation of time which accounts for perturbations in the earth's rate of rotation*. The equation used to calculate solar time is below.

$$\text{solar time} - \text{standard time} = 4(L_{st} - L_{loc}) + E$$

L_{st} is the standard meridian for the local time zone, L_{loc} is the longitude of the location in question and E is the equation of time in minutes. E can be determined graphically from the figure below (from Duffie and Beckmann, Solar Engineering of Thermal Processes, 4th Edition) or from the equations below from Duffie and Beckmann (Duffie and Beckmann, Solar Engineering of Thermal Processes, 4th Edition).

$$E = 229.2(0.000075 + 0.001868\cos B - 0.032077\sin B - 0.014615\cos 2B - 0.04089\sin 2B)$$

$$B = (n-1)\frac{360}{365} \quad 1 \le n \le 365$$

Here, n is the day in the year and B has units degrees. The units for the right hand side of the equation used to calculate solar time are minutes.

To determine L_{st}, multiply the difference in time between local standard clock time and Greenwich Mean Time (GMT) by 15°. This relationship comes from the fact that the sun takes 4 minutes to traverse 1° of longitude. Thus, if your local standard clock is 1 hour behind GMT then LST is 15°.

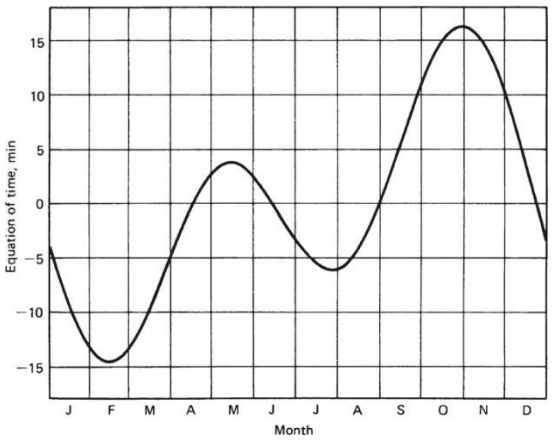

Figure 1. Equation of time verses month in the year. From Duffie and Beckmann, Solar Engineering of Thermal Processes, 4th Edition

You must account for Daylight Savings Time (DST): for example, DST in the United States takes place between the dates of March 8th and November 1st. During this time, the clocks are advanced by 1 hour and thus the difference between the clocks in the United States and GMT changes by 1 hour. For example, if DST is 14:00 then Local Standard Time (LST) which should be used to determine solar time and L_{st}, is 13:00.

Exercise

Note: these calculations can be done by hand and checked with the python solar time calculator that is appended to the end of this document. The calculator can be used for any location within the United States but has not been adapted for other locations yet.

1) In Gainesville, FL what is the solar time corresponding to 10:30 am DST on August 24th? Longitude of Gainesville = 82.3°. To help with difference between local clock time and Greenwich Mean Time, or Coordinated Universal Time, UTC you can use https://www.timeanddate.com/time/zones/edt. For Gainesville the difference between LST and GMT is 5 hours.

Solution:
Current time is 10:30 am DST, thus
LST = 9:30 am

Because the difference between LST and GMT is 5 hours, the standard meridian, L_{st} is

L_{st} = 5*15° = 75°

This is the meridian at which local standard time is based. The longitude, L_{st}, of Gainesville is provided and is

L_{st} = 82.3°

Here we will use a graphical estimation to determine E using the figure on the prior page.

E ≈ -2

Rearranging the equation used to calculate solar time, we see that

$$\text{solar time} = \text{standard time} + 4(L_{\text{st}} - L_{\text{loc}}) + E$$

$$\textbf{solar time} = \mathbf{9{:}30 + 4(75° - 82.3°) - 2 \approx 8{:}59}$$

The offset because of $4(L_{\text{st}} - L_{\text{loc}}) + E$ was equal to -31.2 minutes which was rounded to -31 minutes.

The exact answer using the python calculator and equation for the equation of time is:

$$\textbf{solar time} = \mathbf{8{:}58{:}17}$$

where
E = -2.52, n = 237, B = 232.8

Note that the latitude is requested in the python calculator but the calculation is independent of latitude. You can input Gainesville's, which is 29.65°, or use a different value.

2) In Gainesville, FL what is the local time corresponding to 12:00 solar time on August 24th? Longitude of Gainesville = 82.3°. To help with difference between local clock time and Greenwich Mean Time, or Coordinated Universal Time, UTC you can use https://www.timeanddate.com/time/zones/edt. For Gainesville the difference between LST and GMT is 5 hours.

Solution:
Solar time = 12:00

Because the difference between LST and GMT is 5 hours, the standard meridian, L_{st} is

L_{st} = 5*15° = 75°

This is the meridian at which local standard time is based. The longitude, L_{st}, of Gainesville is provided and is

L_{st} = 82.3°

Here we will use a graphical estimation to determine E using the figure on the prior page.

E ≈ -2

Rearranging the equation used to calculate solar time, we see that

$$\text{standard time} = \text{solar time} - 4(L_{\text{st}} - L_{\text{loc}}) - E$$

$$\text{standard time} = 12{:}00 - 4(75° - 82.3°) + 2 \approx 12{:}31$$

However, recall that on August 24th, it is DST, thus the local time should be 1 hour greater or

DST ≈ 13:31, or 1:31 pm

Appendix

```
# Jonathan Scheffe - Solar Time Calculator 20180903
from datetime import datetime,date,time,timedelta
# import time
import math
#from geopy.geocoders import Nominatim
#import matplotlib.pyplot as plt
import numpy as np

understand = str(input("To be used for locations within the continental United States. Understand? (Enter Y/N)"))
if understand == "Y":
   today = date.today() #todays date in year,month,day
   use_today = str(input("Do you want to use today's date (Y/N)?"))
   if use_today == "N":
      month_enter = int(input("What month are you interested in (e.g. 2,3, etc)? "))
      day_enter = int(input("What day are you interested in(e.g. 2,3, etc)? "))
      date_wanted = date(today.year,month_enter,day_enter)# date of interest for calculation in current year,month,day
   else:
      date_wanted = date(today.year,today.month,today.day)# date of interest for calculation in current year,month,day

   date_reference = date(today.year,1,1) # date on Jan 1st of current year
   day_number = (date_wanted-date_reference).days+1 # number of days between date of interest and Jan 1st
   print ("Day in year is", day_number)
   latitude = float(input("What latitude are you interested in(degrees. N. of Equator is positive and S. is negative)? "))# latitude of interest in degrees
   longitude = float(input("What longitude are you interested in(degrees. W. of Greenwich merridian is positive)? "))# longitude of interest in degrees
   timezone = str(input("What timezone are you interested in(Eastern = E, Central  = C, Mountain = M, Pacific = P)? "))# timezone of interest
   if timezone == "E":
      time_difference = 5
   elif timezone == "C":
      time_difference = 6
   elif timezone == "M":
      time_difference = 7
   elif timezone == "P":
      time_difference = 8

   # Determination of B
   def B(n):
      return (n-1)*360/365
```

```
   print("B = ",B(day_number))

   # Equation of time calculation
   def E(n):
      return 229.2*(0.000075+0.001868*np.cos(np.radians(B(n)))-0.032077*np.sin(np.radians(B(n)))-
0.014615*np.cos(np.radians(2*B(n)))-0.04089*np.sin(np.radians(2*B(n))))
   print("E = ",E(day_number),"minutes")

   # Declination angle calculation
   def delta(n):
      return 23.45*np.sin(np.radians(360*(284+n)/365))
   print("delta = ",delta(day_number),"degrees")

   # x = np.arange(0, 365, 5)# these next four lines plot declination angle verses time of year
   # y = delta(x)
   # plt.plot(x, y)
   # plt.show()

   # Determine standard and utc times currently
   #standard_time = datetime.time(datetime.now()) #standard time based on current location
   utc_time = datetime.time(datetime.utcnow()) #utc time
   #print (utc_time)
   #print (standard_time)

   if date(today.year,11,4)>date_wanted>date(today.year,3,11): #this checks if the date of interest is
observing DST
      print ("It is Daylight Savings Time (DST)")
      #print (standard_time.hour-1,":", standard_time.minute,":", standard_time.second) # if it is DST
then subtract 1 hour from local time

   else:
      #print (standard_time.hour,":", standard_time.minute,":", standard_time.second)
      print ("It is NOT Daylight Savings Time (DST)")

   use_today_time = str(input("Do you want to use today's current time (Y/N)?"))

   if use_today_time == "Y":
      print ("UTC (Greenwich Mean) Time is ",utc_time)
      if date(today.year,11,4)>date_wanted>date(today.year,3,11): #this checks if the date of interest is
obsevring DST
         standard_time = datetime.utcnow() - timedelta(hours = time_difference-1)
         hours_from_utc = utc_time.hour-standard_time.hour+1 # determine hour difference from
standard time to UTC accounting for DST
         time_longitude = 15*hours_from_utc # determine longitude at which local time is based - in
degrees
         solar_time_difference = 4*(time_longitude-longitude)+ E(day_number)# time difference in
minutes from local meridian
         print ("Standard Time is ",datetime.time(standard_time))
```

```
        print ("Solar Time is ",datetime.time(standard_time - timedelta(hours=1,minutes=-
solar_time_difference)))
        solar_time = datetime.time(standard_time - timedelta(hours=1,minutes=-solar_time_difference))
      else:
        standard_time = datetime.utcnow() - timedelta(hours = time_difference)
        hours_from_utc = utc_time.hour-standard_time.hour # determine hour difference from standard
time to UTC accounting for DST
        time_longitude = 15*hours_from_utc # determine longitude at which local time is based - in
degrees
        solar_time_difference = 4*(time_longitude-longitude)+ E(day_number)# time difference in
minutes from local meridian
        print ("Standard Time is ",datetime.time(standard_time))
        print ("Solar Time is ",datetime.time(standard_time - timedelta(minutes=-solar_time_difference)))
        solar_time =datetime.time(standard_time - timedelta(minutes=-solar_time_difference))

   else:
      a=datetime.strptime(input('Specify time (on clock) in HHMMSS format: '), "%H%M%S")
      #print ("Standard Time is ",a.hour,":", a.minute,":", a.second)
      print ("Clock Time is ",datetime.time(a))
      if date(today.year,11,4)>date_wanted>date(today.year,3,11): #this checks if the date of interest is
observing DST
        #standard_time = datetime.utcnow() - timedelta(hours = time_difference-1)
        standard_time = a - timedelta(hours = 1)
        print ("Standard Time is ",standard_time.time())

        utc_time = datetime.time(a + timedelta(hours = time_difference-1))
        hours_from_utc = utc_time.hour-standard_time.hour # determine hour difference from standard
time to UTC accounting for DST
        time_longitude = 15*hours_from_utc # determine longitude at which local time is based - in
degrees
        solar_time_difference = 4*(time_longitude-longitude)+ E(day_number)# time difference in
minutes from local meridian
        print ("UTC (Greenwich Mean) Time is ",utc_time)
        #print ("solar time difference ",solar_time_difference)
        solar_time = standard_time - timedelta(minutes=-solar_time_difference)
        print ("Solar Time is ",solar_time.time())

      else:
        standard_time = a
        print ("Standard Time is ",standard_time.time())

        utc_time = datetime.time(a + timedelta(hours = time_difference))
        hours_from_utc = utc_time.hour-standard_time.hour # determine hour difference from standard
time to UTC accounting for DST
        time_longitude = 15*hours_from_utc # determine longitude at which local time is based - in
degrees
        solar_time_difference = 4*(time_longitude-longitude)+ E(day_number)# time difference in
minutes from local meridian
```

```
        print ("UTC (Greenwich Mean) Time is ",utc_time)
        #print ("solar time difference ",solar_time_difference)
        solar_time = standard_time - timedelta(minutes=-solar_time_difference)
        print ("Solar Time is ",solar_time.time())
else:
   print ("Sorry, please run again.")

#Determine hour angle
hour_angle = ((solar_time.hour-12)*60 + solar_time.minute)*15/60
print ("Hour angle = ", hour_angle,"degrees")
```