```
//
//  NASASun.swift
//  Predict
//
//  Created by Rob Hawley on 9/30/20.
//  Copyright © 2020 Rob Hawley. All rights reserved.
//

// This will calculate everything one would want to know about the sun at a
 particular
// location, date, and time.  Note that Date is broken into a date object
 where h,m,s are zero
// and a time object which is the UTC that you are interested in.
// This is for ease of the caller.  If you care only about the day set time to
  the timezone
// long /15 and add 12 for roughly local noon

// from
 https://www.esrl.noaa
 .gov%2Fgmd%2Fgrad%2Fsolcalc%2FNOAA_Solar_Calculations_day.xls
// The first letter refers to the column in the spreadsheet

// B3 is latitude     // note that ObsConditions uses radians for angles
// B4 is longitude
// B5 is the time zone, but this calculation is UTC so that is 0.0


import Foundation


class SolarCalcs : NSObject {
    var JD : Double  // Julian Day

    fileprivate var statsLoc : ObsConditions

    var atDate : Date = Date.distantPast
    var  g_julCentury : Double
    var i_geomMeanLongSun : Double
    var j_geomMeanAnomSun : Double
    var k_EccentEarthOrbit : Double
    var l_SunEqofCtr : Double
    var m_SunTrueLong : Double
    var p_SunAppLong : Double
    var q_MeanObliqEcliptic : Double
    var r_ObliqCorr : Double
    var t_SunDeclin : Double
    var u_vary : Double
    var v_EqofTime : Double
    var ab_TrueSolarTime : Double
    var ac_HourAngle : Double
    let ad_SolarZenithAngle : Double
```

```
    var ah_SolarAzimuthAngle : Double

    // Additional items calculated after sunsetRise is called

    // in UTC
    var w_HA_Sunrise : Double = -1.0
    var x_SolarNoon : Double = -1.0
    var y_SunriseTime : Double = -1.0 // in frac of days
    var sunriseMinusOne = false
    var sunriseDate : Date = Date.distantPast
    var z_SunsetTime : Double = -1.0 // in frac of days
    var SunsetPlusOne = false
    var sunsetDate : Date = Date.distantPast

// atTime relative to T0
///////////////////////////////////////////////////
     init(loc: ObsConditions, atDay: Date, atTime: TDTHours) {
        //let pi = 3.14159265359

        statsLoc = loc
        atDate = atDay
        JD = atDay.julianDate()
        //let at = BE.T0 + atTime - BE.ΔT/3600 <- make sure caller does this
        let at_days = atTime/24.0
        //NSLog("atdays=" + prt5(at_days))
        JD = JD + at_days
        //NSLog("JD=" + prt5(JD))

        g_julCentury     = (JD-2451545.0)/36525.0
        // i_geomMeanLongSun=MOD(280.46646+G151*(36000.76983 +
         G151*0.0003032),360)
        i_geomMeanLongSun = MOD(number: 280.46646+g_julCentury*(36000.76983 +
         g_julCentury*0.0003032), divisor: 360)
        // j_geomMeanAnomSun=357.52911+G151*(35999.05029 - 0.0001537*G151)
        j_geomMeanAnomSun = 357.52911+g_julCentury*(35999.05029 -
         0.0001537*g_julCentury)
        // k_EccentEarthOrbit =
         0.016708634-G162*(0.000042037+0.0000001267*G162)
        k_EccentEarthOrbit =
         0.016708634-g_julCentury*(0.000042037+0.0000001267*g_julCentury)
        //
         l_SunEqofCtr=SIN(RADIANS(J151))*(1.914602-G151*(0.004817+0
         .000014*G151))+SIN(RADIANS(2*J151))*(0.019993-0
         .000101*G151)+SIN(RADIANS(3*J151))*0.000289
        l_SunEqofCtr =
         sin(rad(j_geomMeanAnomSun))*(1.914602-g_julCentury*(0.004817+0
         .000014*g_julCentury))+sin(rad(2*j_geomMeanAnomSun))*(0.019993-0
         .000101*g_julCentury)+sin(rad(3.0*j_geomMeanAnomSun))*0.000289
        //m_SunTrueLong = =I153+L153
        m_SunTrueLong = i_geomMeanLongSun + l_SunEqofCtr
```

```
// p_SunAppLong
 =M151-0.00569-0.00478*SIN(RADIANS(125.04-1934.136*G151))
p_SunAppLong =
 m_SunTrueLong-0.00569-0.00478*sin(rad(125.04-1934.136*g_julCentury))
// q_MeanObliqEcliptic =
 23+(26+((21.448-G162*(46.815+G162*(0.00059-G162*0.001813))))/60)/60
q_MeanObliqEcliptic =
 23.0+(26.0+((21.448-g_julCentury*(46.815+g_julCentury*(0
 .00059-g_julCentury*0.001813))))/60.0)/60.0
// r_ObliqCorr = Q162+0.00256*COS(RADIANS(125.04-1934.136*G162))
r_ObliqCorr =
 q_MeanObliqEcliptic+0.00256*cos(rad(125.04-1934.136*g_julCentury))
// t_SunDeclin = =DEGREES(ASIN(SIN(RADIANS(R162))*SIN(RADIANS(P162))))
t_SunDeclin = deg(asin(sin(rad(r_ObliqCorr))*sin(rad(p_SunAppLong))))

// u_vary = =TAN(RADIANS(R162/2))*TAN(RADIANS(R162/2))
u_vary = tan(rad(r_ObliqCorr/2.0))*tan(rad(r_ObliqCorr/2.0))

// v_EqofTime =
 4*DEGREES(U162*SIN(2*RADIANS(I162))-2*K162*SIN(RADIANS
 (J162))+4*K162*U162*SIN(RADIANS(J162))*COS(2*RADIANS(I162))-0
 .5*U162*U162*SIN(4*RADIANS(I162))-1.25*K162*K162*SIN(2*RADIANS(J162)))
let v1 = u_vary * sin(2.0 * rad(i_geomMeanLongSun))
let v2 =  2.0 * k_EccentEarthOrbit*sin(rad(j_geomMeanAnomSun))
let v3 = 4.0 * k_EccentEarthOrbit * u_vary *
 sin(rad(j_geomMeanAnomSun))*cos(2.0*rad(i_geomMeanLongSun))
let v4 = 0.5 * u_vary*u_vary*sin(4.0*rad(i_geomMeanLongSun))
let v5 = 1.25 *
 k_EccentEarthOrbit*k_EccentEarthOrbit*sin(2.0*rad(j_geomMeanAnomSun))
v_EqofTime = 4.0 * deg(v1 - v2 + v3 - v4 - v5)




// ab_TrueSolarTime = MOD(E162*1440+V162+4*$B$4-60*$B$5,1440)
// $B$5 is time zone. We are using UTC
// TST is in minutes
ab_TrueSolarTime = MOD(number:
 at_days*1440.0+v_EqofTime+4.0*(loc.longitudeAsDeg())-60.0*0.0,
 divisor: 1440)

// ac_HourAngle = =IF(AB162/4<0,AB162/4+180,AB162/4-180)


if ab_TrueSolarTime/4.0 < 0.0 {
    ac_HourAngle = ab_TrueSolarTime/4.0 + 180.0
} else {
    ac_HourAngle = ab_TrueSolarTime/4.0 - 180.0
}
```

```
    // ad_SolarZenithAngle
     =DEGREES(ACOS(SIN(RADIANS($B$3))*SIN(RADIANS(T162))+COS(RADIANS
     ($B$3))*COS(RADIANS(T162))*COS(RADIANS(AC162))))
    ad_SolarZenithAngle =
     deg(acos(sin(loc.latitude)*sin(rad(t_SunDeclin))+cos(loc
     .latitude)*cos(rad(t_SunDeclin))*cos(rad(ac_HourAngle))))

    // ah_SolarAzimuthAngle (deg cw from N) =
     =IF(AC162>0,MOD(DEGREES(ACOS(((SIN(RADIANS($B$3))*COS(RADIANS
     (AD162)))-SIN(RADIANS(T162)))/(COS(RADIANS($B$3))*SIN(RADIANS
     (AD162)))))+180,360),MOD(540-DEGREES(ACOS(((SIN(RADIANS($B$3))*COS
     (RADIANS(AD162)))-SIN(RADIANS(T162)))/(COS(RADIANS($B$3))*SIN(RADIANS
     (AD162))))),360))

    if ac_HourAngle > 0 {
        ah_SolarAzimuthAngle = MOD(number:
         deg(acos(((sin(loc
         .latitude)*cos(rad(ad_SolarZenithAngle)))-sin(rad(t_SunDeclin)))/
         (cos(loc.latitude)*sin(rad(ad_SolarZenithAngle)))))+180.0,
         divisor: 360)
    } else {
        ah_SolarAzimuthAngle = MOD(number: 540.0 -
         deg(acos(((sin(loc
         .latitude)*cos(rad(ad_SolarZenithAngle)))-sin(rad(t_SunDeclin)))/
         (cos(loc.latitude)*sin(rad(ad_SolarZenithAngle))))), divisor: 360)
    }



    //NSLog("Solar Azimuth=" + prt5(ah_SolarAzimuthAngle))


}

func sunsetRise(){

    //w_HA_Sunrise=DEGREES(ACOS(COS(RADIANS(90
     .833))/(COS(RADIANS($B$3))*COS(RADIANS(T162)))-TAN(RADIANS($B$3))*TAN
     (RADIANS(T162))))
    w_HA_Sunrise=deg(acos(cos(rad(90.833))/(cos(statsLoc
     .latitude)*cos(rad(t_SunDeclin)))-tan(statsLoc
     .latitude)*tan(rad(t_SunDeclin))))

    // sun does not rise
    if w_HA_Sunrise.isNaN  {
        sunriseDate = Date.distantPast
        sunsetDate = Date.distantPast
        return
    }

    //x_SolarNoon ==(720-4*$B$4-V162+$B$5*60)/1440
```

```
        x_SolarNoon =
         (720.0-4.0*(statsLoc.longitudeAsDeg())-v_EqofTime+0.0*60)/1440.0

        //NSLog("solar noon=" + fmtHours(x_SolarNoon*24.0))

        //y_SunriseTime = X162-W162*4/1440
        y_SunriseTime = x_SolarNoon-w_HA_Sunrise*4.0/1440.0
        let srInterval = y_SunriseTime*24.0*60.0*60.0
        sunriseDate = Date(timeInterval: srInterval, since: atDate)
        //NSLog("sunrise=" + fmtHours(y_SunriseTime*24.0))
        //NSLog("sunrise Date=" + sunriseDate.debugDescription)
        if y_SunriseTime < 0 {
            //NSLog("Sunrise previous day")
            sunriseMinusOne = true
            y_SunriseTime = y_SunriseTime + 1.0
        }



        //z_SunsetTime=X162+W162*4/1440
        z_SunsetTime=x_SolarNoon+w_HA_Sunrise*4.0/1440.0
        sunsetDate = Date(timeInterval: z_SunsetTime*24.0*60.0*60.0, since:
         atDate)
        //NSLog("sunset=" + fmtHours(z_SunsetTime*24.0))
        //NSLog("sunset Date=" + sunsetDate.debugDescription)

        if (z_SunsetTime > 1 ){
            //NSLog("Sunset Next Day")
            z_SunsetTime = z_SunsetTime - 1.0
            SunsetPlusOne = true
        }
        //NSLog("sunset=" + fmtHours(z_SunsetTime*24.0))

    }


}
```