SUN ALTITUDES FOR SEXTANT PRACTICE
A Mathcad 8 Prof. Document Prepared October 2000

Roger L. Mansfield
E-mail: astroger@att.net
Webpage: http://astroger.com

(Updated to PTC's Mathcad Prime 10.0 on 2024 July 27)

Given a list of times of interest, together with your geographical location, i.e., your geodetic latitude, longitude, and height above mean sea level, this worksheet will calculate the sun's apparent altitude at each time in the list. Thus, if you have taken a sequence of "sun shots" with a sextant at precisely known Greenwich mean times and you input the times to this worksheet, the worksheet will tell you what your sextant-measured sun altitudes should be.

For each time of interest, the basic steps in the calculation are as follows:

1. Calculate the sun's apparent geocentric equatorial (also called "Earth-centered, inertial" or ECI) cartesian coordinates, referred to the true equator and equinox of date.

2. Calculate your own ECI cartesian coordinates.

3. Calculate the sun's topocentric, local horizon-referenced cartesian coordinates.

4. Convert the sun's topocentric, horizon-referenced cartesian coordinates to apparent azimuth, altitude, and topocentric distance.

5. Correct the sun's apparent altitude for atmospheric refraction.

The procedure which follows requires that a sequence of local times of interest be supplied via a text file named "TIMES.PRN". One sun altitude will be computed for each time point in the sequence, in sequential order, but the times themselves need not be in increasing temporal order. See the end of the worksheet for a discussion of accuracy.

We will need to define some basic conversion factors to go from degrees to radians, from degrees to arc-seconds, and from revolutions to arc-seconds. We will need Earth's mean equatorial radius in meters, and the Gaussian constant.

$DegPerRad := \frac{180}{\pi}$ $\qquad$ $SecPerDeg := 3600.0$

$SecPerRev := 360.0 \cdot SecPerDeg$ $\qquad$ $a_e := 6378135.0$

$k := 0.01720209895$ $\qquad$ We will need the Gaussian constant associated with motion of a planet around the sun.

$\mathbf{ORIGIN} \equiv 1$ $\qquad$ We set the Mathcad ORIGIN to 1 so that vector and matrix subscripts start with unity rather than with zero.

We now define the observer's geographical location. Using the fact that a Mathcad worksheet is "live", you can, of course, change this location to any latitude, longitude, and height of interest.

OBSERVER'S GEOGRAPHICAL LOCATION

$$\phi := \frac{33 + \frac{57}{60} + \frac{24.0}{3600}}{DegPerRad}$$

We set the geodetic latitude to 33 degrees, 57 arc-minutes, and 24 arc-seconds (see Comments immediately below).

$$\lambda := \frac{118 + \frac{27}{60} + \frac{06.0}{3600}}{DegPerRad}$$

We set the longitude to 118 degrees, 27 arc-minutes, and 6 arc-seconds, west, but then subtract this quantity from 360 degrees (2π radians) to convert the longitude from west to east. East longitude will work better in the calculations of Step 2, below.

$$\lambda := 2 \cdot \pi - \lambda$$

$$H := 8.0 \cdot 0.3048$$

We set the height above sea level to 8.0 feet, and multiply by the conversion factor 0.3048 meters per foot to convert the height to meters. (Later on we will divide by a_e to convert the height to Earth radii.)

Comments In celestial navigation, the figure of Earth is assumed to be spherical. However, we assume an oblate spheroidal Earth in our calculation of sun altitudes because the resulting altitudes are then more accurate, i.e., simulate realworld measurements with greater fidelity. Geodetic latitude can be defined as the angle that a line, normal to the oblate spheroid and passing through the observer, makes with Earth's equator. The angle of this definition is subtended at the geocenter only when the observer is at a pole or at the equator, whereas on a spherical Earth, all latitude-defining lines normal to Earth's surface pass through the geocenter.

Speaking of realworld measurements, the test location and times chosen for this worksheet are indeed realworld: they are based upon actual sun shots taken by Richard R. Shiffman and documented in his Mathcad worksheet, "Sextant Noon-Day Sun Sightings" [1].

TIMES OF SEXTANT SIGHTINGS OF THE SUN ("SUN SHOTS")

We input the sequence of times of interest via text file "TIMES.PRN". The format of the times is HH MM SS, and for convenience, the times are specified as local. Thus we need to specify the date on which the sextant measurements were taken, and the time zone offset in hours, so as to be able to convert the times to Greenwich mean times.

$$Times := \text{READPRN}(\text{"TIMES.PRN"})$$

$$Year := 1993$$

$$Month := 4$$

We assume for this example that the date of the sextant sightings was 1993 April 18, and that they were taken from a location keeping Pacific Daylight Time, 7 hours slow on Greenwich mean time (GMT). (See again [1].)

$Day := 18$

$Offset := \frac{7.0}{24}$

We convert the time zone offset from GMT from hours to days, which will work better in our calculations below.

$n := \text{length}\left(Times^{\langle 1 \rangle}\right)$

We use Mathcad's length function to count the number of times/measurements.

$n = 30$

We define, then invoke a procedural function to convert each of the local times to a Greenwich mean time on the date of interest. Note that each of the times in array **GMT** is a fraction of a day that lies between zero and unity.

$$GMTCalc(Times, k) := \left\| \begin{array}{l} \text{for } i \in 1..k \\ \quad \left\| \; t_i \leftarrow \left(\dfrac{Times_{i,1} + \dfrac{Times_{i,2} + \dfrac{Times_{i,3}}{60}}{60}}{24} \right) \right. \\ t + Offset \end{array} \right.$$

$GMT := GMTCalc(Times, n)$

Our astronomical model of the sun's apparent motion in the sky (actually, ECI space) will require that time be input in Julian days of Terrestrial Time, abbreviated "TT" (and previously known as Ephemeris Time, or "ET"). We thus define and invoke below a procedural function that converts days since 1900 January 0.0 to a Julian days.

DayCount specifies the count of days from the beginning of the year, up through the last day of the previous month of any non-leap year. **JED19** calculates the number of Julian days corresponding to Year, Month, and Day. Note that **JED19** is intended to be used with any Gregorian calendar date since 1900 January 0.0, having JED = 2415019.5.

$DayCount := \begin{bmatrix} 0 & 31 & 59 & 90 & 120 & 151 & 181 & 212 & 243 & 273 & 304 & 334 \end{bmatrix}^T$

```
JED19(Year, Month, Day) :=
    JED ← 2415019.5
    for Y ∈ 1900 .. Year
        if Y < Year
            if mod(Y, 4) ≠ 0
                JED ← JED + 365
            else
                if mod(Y, 100) = 0
                    if mod(Y, 400) = 0
                        JED ← JED + 366
                    else
                        JED ← JED + 365
                else
                    JED ← JED + 366
        else
            JED ← JED + DayCount_Month + Day
            if mod(Y, 4) ≠ 0
                JED ← JED + 0
            else
                if Month > 2
                    if mod(Y, 100) = 0
                        if mod(Y, 400) = 0
                            JED ← JED + 1
                        else
                            JED ← JED + 0
                    else
                        JED ← JED + 1
                else
                    JED ← JED + 0
    JED
```

```
JDTCalc(Y, M, D, k) :=
    for i ∈ 1 .. k
        JDT_i ← JED19(Y, M, D)
    JDT
```

$$JDT := JDTCalc(Year, Month, Day, n) + GMT$$

What we have now are n Julian dates, but the times are still GMTs, or in more modern terminology, Universal times (UTs). We need to convert these UTs to TTs. To do this we need to know the time difference TT-UT, which is also known as the "Reduction to Terrestrial Time", which we will add to each JDT in the **JDT** array to make it a TT rather than a UT.

Now the quantity TT - UT is composed of two parts:

(a) the difference between Terrestrial Time and International Atomic Time (TAI), which is a fixed difference of 32.184 seconds,

(b) the difference between TAI and UT, which is known precisely for times in the past, but must be estimated for future dates, i.e.,

TT - UT = (TT - TAI) + (TAI - UT) = 32.184 + (TAI - UT) seconds.

TAI - UT for the past eleven years is

Date, Jan 1.0 UT	TAI - UT, in seconds
1990	25.0
1991	26.0
1992	26.0
1993	27.0
1994	28.0
1995	29.0
1996	30.0
1997	30.0
1998	31.0
1999	32.0
2000	32.0

(See [2], p. K9 for the full table.)

Since there are 86400 seconds in a day, we thus have on 1993 January 1.0 UT that TT - UT, which we will denote as TTUT, is

$$TTUT := \frac{32.184 + 27.0}{86400.0}$$

For the example at hand we convert all of the times of the sextant measurements to Julian days of Terrestrial Time by adding TTUT to each element of **JDT**.

$$JDT := JDT + TTUT$$

Given the precise Terrestrial times of the sextant measurements, we will be able to compute accurate sun altitudes using the solar ephemeris model that we now develop.

1. Calculate the sun's apparent ECI cartesian coordinates, referred to the true equator and equinox of date.

This step has the following parts:

a. Calculate the sun's coordinates referred to the mean equator and equinox of the J2000.0 epoch.

b. Convert the sun's coordinates from true to astrometric by correcting for aberration or "light-time", i.e., the amount of time it takes for the sun's light to travel from the sun to Earth.

c. Apply a precession matrix to refer the coordinates to the mean equator and equinox of date.

d. Apply a nutation matrix to refer the coordinates to the true equator and equinox of date.

To accomplish Step 1a, we will use a procedural function, **GSUN**, which will calculate the sun's ECI position and velocity, given mean orbital elements at an arbitrary epoch. **GSUN** will invoke **E2PV**, a function which calculates position and velocity given the elements of an elliptical orbit and the time elapsed since periapsis. **E2PV** is similar to the function **U2PV**, which is defined in the worksheet, "Ephemeris of a Comet via Uniform Path Mechanics" [3]. But note that while **U2PV** works for elliptical, parabolic, and hyperbolic paths, **E2PV** only works for elliptical orbits.

Note also that functions **PQEQ**, **E2PV**, and **GSUN** have ORIGIN = 1 subscripts, but the corresponding functions in [3] have ORIGIN = 0 subscripts.

$$PQEQ(i, \Omega, \omega, p, q) := \left\| \begin{array}{l} P_1 \leftarrow \cos(\Omega) \cdot \cos(\omega) - \sin(\Omega) \cdot \cos(i) \cdot \sin(\omega) \\ P_2 \leftarrow \sin(\Omega) \cdot \cos(\omega) + \cos(\Omega) \cdot \cos(i) \cdot \sin(\omega) \\ P_3 \leftarrow \sin(i) \cdot \sin(\omega) \\ Q_1 \leftarrow -(\cos(\Omega) \cdot \sin(\omega) + \sin(\Omega) \cdot \cos(i) \cdot \cos(\omega)) \\ Q_2 \leftarrow -(\sin(\Omega) \cdot \sin(\omega) - \cos(\Omega) \cdot \cos(i) \cdot \cos(\omega)) \\ Q_3 \leftarrow \sin(i) \cdot \cos(\omega) \\ p \cdot P + q \cdot Q \end{array} \right.$$

Function **PQEQ** performs the Euler angle rotations needed to transform position and velocity in the orbit plane reference frame (also called the perifocal, or PQW reference frame) to position and velocity in the ECI reference frame. The orbital inclination, i, the right ascension of ascending node, Ω, and the argument of periapsis, ω, are the three Euler angles. We have broken out function **PQEQ** because function **E2PV** performs the Euler angle transformation twice: first it transforms the position, then the velocity.

$$
\begin{aligned}
E2PV(K, q, e, i, \Omega, \omega, \Delta t) := \;
& a \leftarrow \frac{q}{(1-e)} \\
& n \leftarrow K \cdot a^{\frac{-3}{2}} \\
& p \leftarrow q \cdot (1+e) \\
& M \leftarrow n \cdot \Delta t \\
& E \leftarrow M \\
& \Delta E \leftarrow E \\
& \text{while } |\Delta E| \geq 0.00000001 \\
& \quad f \leftarrow E - e \cdot \sin(E) - M \\
& \quad Df \leftarrow 1 - e \cdot \cos(E) \\
& \quad E_{new} \leftarrow E - \frac{f}{Df} \\
& \quad \Delta E \leftarrow E_{new} - E \\
& \quad E \leftarrow E_{new} \\
& rcosv \leftarrow a \cdot (\cos(E) - e) \\
& rsinv \leftarrow a \cdot \sqrt{1-e^2} \cdot \sin(E) \\
& r \leftarrow PQEQ(i, \Omega, \omega, rcosv, rsinv) \\
& rmag \leftarrow \sqrt{r \cdot r} \\
& rmagdot \leftarrow \frac{-K}{\sqrt{p}} \cdot \frac{rsinv}{rmag} \\
& rvdot \leftarrow \frac{K}{\sqrt{p}} \cdot \left(e + \frac{rcosv}{rmag}\right) \\
& v \leftarrow PQEQ(i, \Omega, \omega, rmagdot, rvdot) \\
& \text{augment}(r, v)
\end{aligned}
$$

The rationale for the name **E2PV** is that "**E2PV** transforms **E**lliptical orbital elements and time since periapsis to (**2**) **P**osition and **V**elocity". **E2PV** employs the classical notation of two-body orbit propagation, while **U2PV** in [3] employs the notation of Uniform Path Mechanics (UPM), as described in [3].

Define function **GSUN** to calculate the geocentric ecliptic position and velocity of the sun as a function of the Julian date, with epoch at 2000 January 1.5 TT (JD = 2451545.0). Note that k and DegPerRad, both as defined above, are "global" arguments of this function, i.e., they are defined in the worksheet outside of the function, and prior to its definition. So also are SecPerDeg and SecPerRev. The solar model constants in **GSUN** were taken from [4].

$$GSUN(JD) := \left\| \begin{array}{l} JD_o \leftarrow 2451545.0 \\ T_c \leftarrow \dfrac{JD - JD_o}{36525.0} \\ a \leftarrow 1.00000011 - 0.00000005 \cdot T_c \\ e \leftarrow 0.01671022 - 0.00003804 \cdot T_c \\ q \leftarrow a \cdot (1 - e) \\ \mu \leftarrow 1.00000304 \\ K \leftarrow k \cdot \sqrt{\mu} \\ n \leftarrow K \cdot a^{\frac{-3}{2}} \\ \omega \leftarrow \dfrac{102.94719 + \dfrac{1198.28 \cdot T_c}{SecPerDeg}}{DegPerRad} \\ i \leftarrow \dfrac{0.00005 - \dfrac{46.94 \cdot T_c}{SecPerDeg}}{DegPerRad} \\ \Omega \leftarrow 0.0 \\ L \leftarrow \dfrac{100.46435 + \dfrac{1293740.63 + 99 \cdot SecPerRev}{SecPerDeg} \cdot T_c}{DegPerRad} \\ T \leftarrow JD - \dfrac{\mathrm{mod}(L - \omega, 2 \cdot \pi)}{n} \\ \Delta t \leftarrow JD - T \\ PV \leftarrow E2PV(K, q, e, i, \Omega, \omega, \Delta t) \\ r_{EM} \leftarrow PV^{\langle 1 \rangle} \\ v_{EM} \leftarrow PV^{\langle 2 \rangle} \\ L_M \leftarrow \dfrac{\mathrm{mod}(218.0 + 481268.0 \cdot T_c, 360.0)}{DegPerRad} \\ -\mathrm{augment}\left(\begin{bmatrix} r_{EM_1} - 0.0000312 \cdot \cos(L_M) \\ r_{EM_2} - 0.0000312 \cdot \sin(L_M) \\ r_{EM_3} \end{bmatrix}, v_{EM} \right) \end{array} \right.$$

GSUN is based upon **HGEO** in [3]. It takes advantage of the fact that the geocentric ecliptic cartesian coordinates of the sun are the negatives of the heliocentric ecliptic cartesian coordinates of the geocenter. Note that the minus sign is applied in the very last line of **GSUN**.

Define function **ECEQ** to convert from geocentric ecliptic coordinates to geocentric equatorial coordinates at the J2000 epoch.

$$ECEQ(r) := \left\| \begin{array}{l} \varepsilon \leftarrow \dfrac{23.4392911}{DegPerRad} \\ M \leftarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varepsilon) & -\sin(\varepsilon) \\ 0 & \sin(\varepsilon) & \cos(\varepsilon) \end{bmatrix} \\ M \cdot r \end{array} \right.$$

We have now defined what we need to compute the ECI cartesian coordinates of the sun, referred to the mean equator and equinox of J2000.0, at the specified times.

To accomplish Step 1b, we need a function that performs the light-time correction. (For a reference, see [5], p. 320.)

$$LTIM(PV) := \left\| \begin{array}{l} r \leftarrow PV^{\langle 1 \rangle} \\ v \leftarrow PV^{\langle 2 \rangle} \\ \Delta \leftarrow \sqrt{r \cdot r} \\ r - 0.00578 \cdot \Delta \cdot v \end{array} \right.$$

To accomplish Step 1c, we need a precession matrix function, **PRECSS**, defined as follows. (For a reference, see [5], p. 318.)

$$PRECSS(r, JD) := \left\| \begin{array}{l} T \leftarrow \dfrac{(JD - 2451545.0)}{36525.0} \\ P_{1,1} \leftarrow 1.0 - 0.00029724 \cdot T^2 - 0.00000013 \cdot T^3 \\ P_{1,2} \leftarrow -0.02236172 \cdot T - 0.00000677 \cdot T^2 + 0.00000222 \cdot T^3 \\ P_{1,3} \leftarrow -0.00971717 \cdot T + 0.00000207 \cdot T^2 + 0.00000096 \cdot T^3 \\ P_{2,1} \leftarrow -P_{1,2} \\ P_{2,2} \leftarrow 1.0 - 0.00025002 \cdot T^2 - 0.00000015 \cdot T^3 \\ P_{2,3} \leftarrow -0.00010865 \cdot T^2 \\ P_{3,1} \leftarrow -P_{1,3} \\ P_{3,2} \leftarrow P_{2,3} \\ P_{3,3} \leftarrow 1.0 - 0.00004721 \cdot T^2 \\ P \cdot r \end{array} \right.$$

To accomplish Step 1d, we need a nutation matrix function, **NUTATE**, as defined follows.
(For a reference, see [5], p. 320.)

$$NUTATE(r, JD) := \left\| \begin{array}{l} d \leftarrow JD - 2451545.0 \\ \varepsilon \leftarrow \dfrac{23.4392911}{DegPerRad} \\ \Delta\psi \leftarrow -0.0048 \cdot \sin\left(\dfrac{125.0 - 0.05295 \cdot d}{DegPerRad}\right) - 0.0004 \cdot \sin\left(\dfrac{200.9 + 1.97129 \cdot d}{DegPerRad}\right) \\ \Delta\psi \leftarrow \dfrac{\Delta\psi}{DegPerRad} \\ \Delta\varepsilon \leftarrow 0.0026 \cdot \cos\left(\dfrac{125.0 - 0.05295 \cdot d}{DegPerRad}\right) + 0.0002 \cdot \cos\left(\dfrac{200.9 + 1.97129 \cdot d}{DegPerRad}\right) \\ \Delta\varepsilon \leftarrow \dfrac{\Delta\varepsilon}{DegPerRad} \\ N \leftarrow \begin{bmatrix} 1.0 & -\Delta\psi \cdot \cos(\varepsilon) & -\Delta\psi \cdot \sin(\varepsilon) \\ \Delta\psi \cdot \cos(\varepsilon) & 1.0 & -\Delta\varepsilon \\ \Delta\psi \cdot \sin(\varepsilon) & \Delta\varepsilon & 1.0 \end{bmatrix} \\ N \cdot r \end{array} \right.$$

We now accomplish Steps 1b, 1c, and 1d by defining and invoking procedural function **APPSUN**.

$$APPSUN(JDT) := \left\| \begin{array}{l} n \leftarrow \text{length}(JDT) \\ \text{for } i \in 1..n \\ \quad \left\| \begin{array}{l} JD \leftarrow JDT_i \\ PV \leftarrow GSUN(JD) \\ r \leftarrow LTIM(PV) \\ r \leftarrow ECEQ(r) \\ r \leftarrow NUTATE(PRECSS(r, JD), JD) \\ \alpha \leftarrow \text{angle}\left(r_1, r_2\right) \cdot \dfrac{DegPerRad}{15} \\ \delta \leftarrow \text{asin}\left(\dfrac{r_3}{\sqrt{\left(r_1\right)^2 + \left(r_2\right)^2 + \left(r_3\right)^2}}\right) \cdot DegPerRad \\ \text{if } i = 1 \\ \quad \| \; Table \leftarrow [JD \;\; \alpha \;\; \delta] \\ \text{else} \\ \quad \| \; Table \leftarrow \text{stack}(Table, [JD \;\; \alpha \;\; \delta]) \end{array} \right. \\ Table \end{array} \right.$$

$M := APPSUN(JDT)$

To see what the apparent ECI equatorial coordinates of the sun turn out to be, we define a formatting function, **FMT**, and apply it to the solar ephemeris generated via **APPSUN**.

$$
\begin{aligned}
FMT(M,N) := \;& \text{for } j \in 1..N \\
& \quad h_r \leftarrow M_{j,2} + \frac{0.5}{36000} \\
& \quad h \leftarrow \text{floor}(h_r) \\
& \quad m \leftarrow 60 \cdot (h_r - h) \\
& \quad s \leftarrow \frac{\text{floor}(600 \cdot (m - \text{floor}(m)))}{10} \\
& \quad m \leftarrow \text{floor}(m) \\
& \quad H_{j,1} \leftarrow h \\
& \quad P_{j,1} \leftarrow m \\
& \quad S_{j,1} \leftarrow s \\
& A \leftarrow \text{augment}(H, P) \\
& A \leftarrow \text{augment}(A, S) \\
& \text{for } j \in 1..N \\
& \quad d_r \leftarrow \left|M_{j,3}\right| + \frac{0.5}{3600} \\
& \quad d \leftarrow \text{floor}(d_r) \\
& \quad m \leftarrow 60 \cdot (d_r - d) \\
& \quad s \leftarrow \text{floor}(60 \cdot (m - \text{floor}(m))) \\
& \quad m \leftarrow \text{floor}(m) \\
& \quad H_{j,1} \leftarrow d \\
& \quad \text{if } M_{j,3} < 0 \\
& \quad \quad H_{j,1} \leftarrow -d \\
& \quad P_{j,1} \leftarrow m \\
& \quad S_{j,1} \leftarrow s \\
& A \leftarrow \text{augment}(A, H) \\
& A \leftarrow \text{augment}(A, P) \\
& A \leftarrow \text{augment}(A, S) \\
& A \leftarrow \text{augment}\left(M^{\langle 1 \rangle}, A\right)
\end{aligned}
$$

In the formatted array, note that the right ascensions are rounded to the nearest tenth of a time second and the declinations are rounded to the nearest whole arc-second.

$$FMT(M,n)=\begin{bmatrix} 2449096.319701 & 1 & 46 & 58 & 11 & 2 & 20 \\ 2449096.320384 & 1 & 46 & 58.1 & 11 & 2 & 21 \\ 2449096.320963 & 1 & 46 & 58.3 & 11 & 2 & 22 \\ 2449096.321623 & 1 & 46 & 58.4 & 11 & 2 & 23 \\ 2449096.322282 & 1 & 46 & 58.5 & 11 & 2 & 24 \\ 2449096.323266 & 1 & 46 & 58.8 & 11 & 2 & 25 \\ 2449096.324065 & 1 & 46 & 58.9 & 11 & 2 & 26 \\ 2449096.324771 & 1 & 46 & 59.1 & 11 & 2 & 27 \\ 2449096.32528 & 1 & 46 & 59.2 & 11 & 2 & 27 \\ 2449096.326264 & 1 & 46 & 59.4 & 11 & 2 & 29 \\ 2449096.326958 & 1 & 46 & 59.6 & 11 & 2 & 29 \\ 2449096.327629 & 1 & 46 & 59.7 & 11 & 2 & 30 \\ 2449096.328231 & 1 & 46 & 59.9 & 11 & 2 & 31 \\ 2449096.329041 & 1 & 47 & 0.1 & 11 & 2 & 32 \\ 2449096.329562 & 1 & 47 & 0.2 & 11 & 2 & 33 \\ 2449096.330257 & 1 & 47 & 0.3 & 11 & 2 & 34 \\ 2449096.330916 & 1 & 47 & 0.5 & 11 & 2 & 34 \\ 2449096.331472 & 1 & 47 & 0.6 & 11 & 2 & 35 \\ 2449096.332236 & 1 & 47 & 0.8 & 11 & 2 & 36 \\ 2449096.332965 & 1 & 47 & 0.9 & 11 & 2 & 37 \\ 2449096.333613 & 1 & 47 & 1.1 & 11 & 2 & 38 \\ 2449096.334308 & 1 & 47 & 1.2 & 11 & 2 & 39 \\ 2449096.336541 & 1 & 47 & 1.7 & 11 & 2 & 41 \\ 2449096.337329 & 1 & 47 & 1.9 & 11 & 2 & 42 \\ 2449096.337965 & 1 & 47 & 2.1 & 11 & 2 & 43 \\ 2449096.338602 & 1 & 47 & 2.2 & 11 & 2 & 44 \\ 2449096.339261 & 1 & 47 & 2.3 & 11 & 2 & 45 \\ 2449096.339771 & 1 & 47 & 2.5 & 11 & 2 & 45 \\ 2449096.340291 & 1 & 47 & 2.6 & 11 & 2 & 46 \\ 2449096.340824 & 1 & 47 & 2.7 & 11 & 2 & 47 \end{bmatrix}$$

This completes Step 1. But since the accuracy of our computed sun altitudes depends so critically upon our solar ephemeris model, we should take the time at this point to see how good our solar ephemeris model really is. We define a new time array, **Apr1993**, consisting of the Julian date at the beginning of each day of April 1993, Terrestrial Time, then calculate the sun's apparent right ascension and declination at each of these times. We compare the results with those obtained using the U.S. Naval Observatory's MICA program [6].

```
Apr1993 := | JD ← 2449078.5
           | for i ∈ 0..29
           |   | JDT[i+1] ← JD + i
           | JDT
```

$$M := APPSUN(Apr1993)$$

Compare **APPSUN** values with R.A. and Dec. as generated by the U.S. Naval Observatory's MICA 1990-2005 program:

$FMT(M,30) =$

2449078.5	0	41	28.7	4	27	44
2449079.5	0	45	7.4	4	50	52
2449080.5	0	48	46.2	5	13	54
2449081.5	0	52	25.1	5	36	51
2449082.5	0	56	4.1	5	59	41
2449083.5	0	59	43.3	6	22	26
2449084.5	1	3	22.8	6	45	4
2449085.5	1	7	2.4	7	7	35
2449086.5	1	10	42.3	7	29	58
2449087.5	1	14	22.4	7	52	14
2449088.5	1	18	2.8	8	14	23
2449089.5	1	21	43.5	8	36	23
2449090.5	1	25	24.5	8	58	14
2449091.5	1	29	5.9	9	19	57
2449092.5	1	32	47.6	9	41	30
2449093.5	1	36	29.7	10	2	54
2449094.5	1	40	12.1	10	24	8
2449095.5	1	43	55	10	45	12
2449096.5	1	47	38.3	11	6	5
2449097.5	1	51	21.9	11	26	48
2449098.5	1	55	6	11	47	19
2449099.5	1	58	50.6	12	7	38
2449100.5	2	2	35.6	12	27	46
2449101.5	2	6	21	12	47	41
2449102.5	2	10	6.9	13	7	24
2449103.5	2	13	53.3	13	26	53
2449104.5	2	17	40.2	13	46	9
2449105.5	2	21	27.5	14	5	11
2449106.5	2	25	15.3	14	24	0
2449107.5	2	29	3.7	14	42	34

```
                              Sun

              Apparent Geocentric Positions,
             True Equator and Equinox of Date

    Date,               Right              Declina-
     TDT              Ascension              tion

                     h  m   s             °  '   "
1993 Apr 01.0      0 41 28.421     +  4 27 41.70
1993 Apr 02.0      0 45 07.085     +  4 50 49.15
1993 Apr 03.0      0 48 45.850     +  5 13 51.31
1993 Apr 04.0      0 52 24.738     +  5 36 47.83
1993 Apr 05.0      0 56 03.769     +  5 59 38.40
1993 Apr 06.0      0 59 42.970     +  6 22 22.72
1993 Apr 07.0      1 03 22.365     +  6 45 00.47
1993 Apr 08.0      1 07 01.980     +  7 07 31.35
1993 Apr 09.0      1 10 41.837     +  7 29 55.07
1993 Apr 10.0      1 14 21.958     +  7 52 11.28
1993 Apr 11.0      1 18 02.362     +  8 14 19.66
1993 Apr 12.0      1 21 43.067     +  8 36 19.86
1993 Apr 13.0      1 25 24.089     +  8 58 11.54
1993 Apr 14.0      1 29 05.446     +  9 19 54.34
1993 Apr 15.0      1 32 47.151     +  9 41 27.93
1993 Apr 16.0      1 36 29.220     + 10 02 51.93
1993 Apr 17.0      1 40 11.667     + 10 24 06.02
1993 Apr 18.0      1 43 54.506     + 10 45 09.84
1993 Apr 19.0      1 47 37.751     + 11 06 03.05
1993 Apr 20.0      1 51 21.413     + 11 26 45.30
1993 Apr 21.0      1 55 05.504     + 11 47 16.25
1993 Apr 22.0      1 58 50.036     + 12 07 35.56
1993 Apr 23.0      2 02 35.017     + 12 27 42.89
1993 Apr 24.0      2 06 20.457     + 12 47 37.90
1993 Apr 25.0      2 10 06.363     + 13 07 20.25
1993 Apr 26.0      2 13 52.742     + 13 26 49.61
1993 Apr 27.0      2 17 39.599     + 13 46 05.63
1993 Apr 28.0      2 21 26.942     + 14 05 07.99
1993 Apr 29.0      2 25 14.774     + 14 23 56.35
1993 Apr 30.0      2 29 03.102     + 14 42 30.38
```

We see that the right ascensions agree to within about one second of time, while the declinations can be off by up to about four seconds of arc. Sampling of other 30-day dates typically yields agreement to within about a second of R.A. and about six arc-seconds (i.e., about a tenth of an arc-minute) of Dec.

2. Calculate the observer's ECI cartesian coordinates.

This step has the following two parts:

a. Convert geodetic latitude, longitude, and height to Earth-fixed, Greenwich (EFG) coordinates.

b. Apply a simplified model of Earth rotation, based upon the time elapsed since the reference epoch 2000 January 0.0 UT to the instant of observation, to obtain the observer's ECI cartesian coordinates

This simplified model is embodied in the following equation, called "Newcomb's formula", which gives the mean sidereal time at Greenwich as a function of time elapsed in days since 2000 January 0.0 UT.

$$\theta_G(days) := \text{mod}\left(\frac{98.98215}{DegPerRad} + \frac{360.98564735}{DegPerRad} \cdot days, 2 \cdot \pi\right)$$

$JD_o := 2451543.5$

To implement Newcomb's formula we need the Julian date corresponding to 2000 January 0.0 UT.

$GMT := JDT - TTUT$

We also need the UTs corresponding to **JDT**'s TTs.

$$f := \frac{1}{298.26}$$

To calculate the observer's ECI coordinates we need the flattening factor associated with Earth's oblateness.

$$e_e := \sqrt{2 \cdot f - f^2}$$

The quantity e_e is the eccentricity of the reference ellipse associated with Earth's meridional cross-section.

$$e_e^{\,2} = 0.006694$$

$ERPAU := 23454.79842$

We need the number of Earth radii in one astronomical unit (A.U.), so that we can convert the observer's ECI cartesian coordinates to A.U. before we subtract them from the sun's coordinates in A.U.

Define and invoke function **SENPOS** to create a 3xn matrix of ECI observer positions, thereby effecting Step 2.

$$SENPOS(t, \phi, \lambda, H) := \left\| \begin{array}{l} \text{for } i \in 1..n \\ \quad \left\| \begin{array}{l} \theta \leftarrow \theta_G\left(t_i\right) + \lambda \\ G_1 \leftarrow \dfrac{1}{\sqrt{1 - e_e^{\,2} \cdot \sin(\phi)^2}} + \dfrac{H}{a_e} \\ G_2 \leftarrow \dfrac{\left(1 - e_e^{\,2}\right)}{\sqrt{1 - e_e^{\,2} \cdot \sin(\phi)^2}} + \dfrac{H}{a_e} \\ R^{\langle i \rangle} \leftarrow \begin{bmatrix} G_1 \cdot \cos(\phi) \cdot \cos(\theta) \\ G_1 \cdot \cos(\phi) \cdot \sin(\theta) \\ G_2 \cdot \sin(\phi) \end{bmatrix} \end{array} \right. \\ \dfrac{R}{ERPAU} \end{array} \right.$$

$$R := SENPOS\left(GMT - JD_o, \phi, \lambda, H\right)$$

Display R-transpose so that document does not switch from page mode to draft mode.

$$R^{\mathrm{T}} = \begin{bmatrix} 0.0000325095 & 0.0000140135 & 0.0000236799 \\ 0.0000324489 & 0.0000141532 & 0.0000236799 \\ 0.0000323971 & 0.0000142714 & 0.0000236799 \\ 0.0000323375 & 0.000014406 & 0.0000236799 \\ 0.0000322773 & 0.0000145403 & 0.0000236799 \\ 0.0000321866 & 0.0000147401 & 0.0000236799 \\ 0.000032112 & 0.0000149018 & 0.0000236799 \\ 0.0000320454 & 0.0000150445 & 0.0000236799 \\ 0.000031997 & 0.0000151472 & 0.0000236799 \\ 0.0000319025 & 0.0000153453 & 0.0000236799 \\ 0.000031835 & 0.0000154847 & 0.0000236799 \\ 0.0000317693 & 0.0000156192 & 0.0000236799 \\ 0.0000317098 & 0.0000157396 & 0.0000236799 \\ 0.000031629 & 0.0000159012 & 0.0000236799 \\ 0.0000315767 & 0.0000160049 & 0.0000236799 \\ 0.0000315064 & 0.0000161429 & 0.0000236799 \\ 0.000031439 & 0.0000162738 & 0.0000236799 \\ 0.0000313818 & 0.0000163837 & 0.0000236799 \\ 0.0000313026 & 0.0000165345 & 0.0000236799 \\ 0.0000312263 & 0.0000166782 & 0.0000236799 \\ 0.000031158 & 0.0000168055 & 0.0000236799 \\ 0.0000310841 & 0.0000169417 & 0.0000236799 \\ 0.0000308426 & 0.0000173775 & 0.0000236799 \\ 0.0000307561 & 0.0000175302 & 0.0000236799 \\ 0.0000306855 & 0.0000176534 & 0.0000236799 \\ 0.0000306145 & 0.0000177763 & 0.0000236799 \\ 0.0000305403 & 0.0000179034 & 0.0000236799 \\ 0.0000304827 & 0.0000180013 & 0.0000236799 \\ 0.0000304235 & 0.0000181013 & 0.0000236799 \\ 0.0000303626 & 0.0000182032 & 0.0000236799 \end{bmatrix}$$

3. Calculate the sun's topocentric, local horizon-referenced cartesian coordinates.

This step has the following parts:

a. Subtract the observer's ECI coordinates from the sun's ECI coordinates.

b. Apply an orthogonal rotation matrix, based upon the geodetic latitude and the right ascension of the observer, to convert the sun's topocentric ECI coordinates to topocentric, horizon-referenced coordinates.

First we define an orthogonal rotation matrix, **SEZ**, which transforms topocentric horizon-referenced cartesian coordinates to topocentric ECI coordinates. We will need the transpose.

$$SEZ(\phi, \theta) := \begin{bmatrix} \sin(\phi) \cdot \cos(\theta) & -\sin(\theta) & \cos(\theta) \cdot \cos(\phi) \\ \sin(\phi) \cdot \sin(\theta) & \cos(\theta) & \sin(\theta) \cdot \cos(\phi) \\ -\cos(\phi) & 0 & \sin(\phi) \end{bmatrix}$$

Note that ϕ is the geodetic latitude and θ is the right ascension of the observer.

4. Convert the sun's topocentric, horizon-referenced ECI coordinates to altitude, azimuth, and topocentric distance.

We combine Steps 3 and 4 into a procedural function, **ALTSUN**, which provides the sun's altitude and azimuth in degrees, and topocentric distance in A.U. Note that **ALTSUN** is similar in its loop structure to **APPSUN**.

```
ALTSUN(JDT) := | n ← length(JDT)
               | for i ∈ 1 .. n
               |   | JD ← JDT_i
               |   | PV ← GSUN(JD)
               |   | r ← LTIM(PV)
               |   | r ← ECEQ(r)
               |   | r ← NUTATE(PRECSS(r, JD), JD)
               |   | r_top ← r − R^<i>
               |   | θ ← θ_G(GMT_i − JD_o) + λ
               |   | ρ ← (SEZ(φ, θ))^T · r_top
               |   | ρmag ← sqrt(ρ · ρ)
               |   | Alt ← asin(ρ_3 / ρmag) · DegPerRad − asin(4.6525 · 10^-3 / ρmag) · DegPerRad
               |   | Azi ← mod(3 · π − angle(ρ_1, ρ_2), 2 · π) · DegPerRad
               |   | if i = 1
               |   |   | Table ← [i  Times_i,1  Times_i,2  Times_i,3  Alt  Azi  ρmag]
               |   | else
               |   |   | Table ← stack(Table, [i  Times_i,1  Times_i,2  Times_i,3  Alt  Azi  ρmag])
               | Table
```

$$SunData := ALTSUN(JDT)$$

Invoking **ALTSUN** yields the following table of the sun's altitudes and azimuths in degrees, and topocentric distances in A.U. Note the use of the array **Times** to refer the predicted measurements back to the times of the sun shots.

Meas. #	Time hh	mm	ss	Altitude, degrees	Azimuth, degrees	Distance, A.U.
1	12	39	23	66.6032095	171.42130852	1.00431664
2	12	40	22	66.63279354	172.03136916	1.00431682
3	12	41	12	66.65617986	172.54944189	1.00431698
4	12	42	9	66.68094829	173.14116313	1.00431715
5	12	43	6	66.70369474	173.73399026	1.00431733
6	12	44	31	66.73384606	174.61988619	1.0043176
7	12	45	40	66.7549931	175.34047265	1.00431782
8	12	46	41	66.77119683	175.97844852	1.00431801
9	12	47	25	66.78143025	176.43910475	1.00431815
10	12	48	50	66.79774066	177.32996123	1.00431842
11	12	49	50	66.80650533	177.95941477	1.00431861
12	12	50	48	66.81281274	178.56824843	1.0043188
13	12	51	40	66.81665666	179.1143166	1.00431897
14	12	52	50	66.81912579	179.84959845	1.00431919
15	12	53	35	66.81907327	180.32231861	1.00431934
16	12	54	35	66.81700696	180.952563	1.00431953
17	12	55	32	66.81293129	181.55114844	1.00431972
18	12	56	20	66.80790336	182.05503617	1.00431988
19	12	57	26	66.79860956	182.74749509	1.00432009
20	12	58	29	66.78716994	183.40793504	1.0043203
21	12	59	25	66.77489852	183.9944408	1.00432049
22	13	0	25	66.75955812	184.62215546	1.00432069
23	13	3	38	66.69488943	186.63522193	1.00432133
24	13	4	46	66.66656124	187.34180107	1.00432156
25	13	5	41	66.6415473	187.91209901	1.00432175
26	13	6	36	66.6146607	188.4812365	1.00432193
27	13	7	33	66.58482736	189.06976158	1.00432213
28	13	8	17	66.56043233	189.52309642	1.00432228
29	13	9	2	66.53425633	189.98581942	1.00432243
30	13	9	48	66.50622104	190.45782603	1.00432259

$SunData =$ (matrix above)

We should note at this point that when the observer sights on the sun with a sextant, he or she typically "brings the lower limb of the sun down to the horizon" in the process of making the altitude measurement.

Since the altitude of the sun is, by convention, the altitude of the sun's center above the sea horizon, the distance between the center of the sun and its limb, called the "solar semidiameter", must be added to the sextant-measured altitude before the altitude measurement is reduced.

But, since we are simulating sun altitude measurements, we must subtract the semidiameter from each computed altitude. This has been done in **ALTSUN**, above: the second term in the calculation of "Alt" is the solar semidiameter in degrees.

5. Correct the sun's altitudes for atmospheric refraction.

$$ZT := \begin{bmatrix} 0.0 \\ 10.00277778 \\ 20.0058 \\ 30.00944444 \\ 40.01361111 \\ 45.01638889 \\ 50.01944444 \\ 55.02333333 \\ 60.02805556 \\ 65.03472222 \\ 70.04416667 \\ 75.05972222 \\ 80.08861111 \\ 81.09805556 \\ 82.10944444 \\ 83.12333333 \\ 84.14138889 \\ 85.16472222 \\ 86.19611111 \\ 87.24027778 \\ 88.30638889 \\ 89.41138889 \\ 90.0 \\ 90.58972222 \end{bmatrix} \qquad RF := \begin{bmatrix} 0.0 \\ 10.0 \\ 21.0 \\ 34.0 \\ 49.0 \\ 59.0 \\ 70.0 \\ 84.0 \\ 101.0 \\ 125.0 \\ 159.0 \\ 215.0 \\ 319.0 \\ 353.0 \\ 394.0 \\ 444.0 \\ 509.0 \\ 593.0 \\ 706.0 \\ 865.0 \\ 1103.0 \\ 1481.0 \\ 1760.0 \\ 2123.0 \end{bmatrix}$$

ZT is a table of true zenith distances and **RF** is a table of the corresponding amounts of refraction that rays of light experience as they pass through the atmosphere at these zenith distances. (For a reference see [7].)

What we will do is to define a function, **RFN**, which computes the atmospheric refraction at 760 mm Hg and 10 degrees Celsius as a function of true altitude.

RFN interpolates between two bracketing values of true zenith distance to find the atm. refraction.

```
RFN(Alt) :=  n ← length(Alt)
             for i ∈ 1 .. n
                 Z ← 90.0 − Alt_i
                 if Z ≤ ZT_1
                     Rfn_i ← 0.0
                 else
                     if Z > ZT_24
                         Rfn_i ← 0.0
                     else
                         m ← 2
                         for j ∈ 2 .. 23
                             if Z > ZT_j
                                 m ← m + 1
                         Rfn_i ← ( RF_(m−1) + ((RF_m − RF_(m−1)) / (ZT_m − ZT_(m−1))) · (Z − ZT_(m−1)) ) / SecPerDeg
             Rfn
```

$$Rfn_i \leftarrow \frac{\left(RF_{m-1} + \frac{\left(RF_m - RF_{m-1}\right)}{\left(ZT_m - ZT_{m-1}\right)} \cdot \left(Z - ZT_{m-1}\right)\right)}{SecPerDeg}$$

$Altitude := SunData^{\langle 5 \rangle}$

$RefrAlt := Altitude + RFN(Altitude)$

$$N(m) := \left\| \begin{array}{l} n \leftarrow \text{length}(Altitude) \\ \text{for } i \in 1..n \\ \quad \| N_i \leftarrow i \\ N \end{array} \right.$$

$M := N(n)$

(See Appendix 1 for validation of function **RFN**.)

Note that the sun's altitude, as tabulated below, is by our calculations in **ALTSUN** the altitude of the sun's lower limb, as would be measured using a sextant. The second tabulation accounts for atmospheric refraction so as to simulate as accurately as possible the actual sextant-measured altitude.

SUN'S ALTITUDE (LOWER LIMB)

$\text{augment}(M, Altitude) =$

1	66.60321
2	66.63279
3	66.65618
4	66.68095
5	66.70369
6	66.73385
7	66.75499
8	66.7712
9	66.78143
10	66.79774
11	66.80651
12	66.81281
13	66.81666
14	66.81913
15	66.81907
16	66.81701
17	66.81293
18	66.8079
19	66.79861
20	66.78717
21	66.7749
22	66.75956
23	66.69489
24	66.66656
25	66.64155
26	66.61466
27	66.58483
28	66.56043
29	66.53426
30	66.50622

SEXTANT-MEASURED ALTITUDE

$\text{augment}(M, RefrAlt) =$

1	66.61027
2	66.63984
3	66.66322
4	66.68798
5	66.71072
6	66.74086
7	66.762
8	66.77819
9	66.78842
10	66.80473
11	66.81349
12	66.81979
13	66.82364
14	66.82611
15	66.82605
16	66.82399
17	66.81991
18	66.81489
19	66.8056
20	66.79416
21	66.78189
22	66.76656
23	66.70191
24	66.6736
25	66.64859
26	66.62171
27	66.59189
28	66.56751
29	66.54134
30	66.51331

DISCUSSION OF ACCURACY

The solar ephemeris model herein neglects the perturbations of the Earth-moon system by the other major planets. It uses a simple, mean-lunar-longitude-based model for the orbit of the geocenter as it revolves around the Earth-moon barycenter. The errors quantified at the end of Step 1 arise from the assumptions made in these models. The models of precession and nutation have an error tolerance of about an arc-second according to [5]. The Earth rotation model does not account for polar wander (the true pole can wander up to about 15m away from the mean pole).

Still, it is believed that the sun altitudes above are accurate to within about an arc-minute, and that the models and assumptions herein are good enough for sun-sight celestial navigation without printed tables.

The reader who wishes to improve upon the models in this worksheet might wish to start with Montenbruck & Pfleger [8] or with Heafner [9]. Montenbruck & Pfleger provide analytical expressions that account for planetary perturbations. Heafner shows how to access the highly accurate JPL ephemerides on CD-ROM.

REFERENCES

[1] Richard R. Shiffman, "Sextant Noon-Day Sun Sightings," Math in Action, MathSoft, Inc. (http://mathsoft.com/appsindex.html).

[2] Alan D. Fiala, et al., Astronomical Almanac for the Year 2000, U.S. Naval Observatory, Washington, D.C. U.S.A., and Rutherford Appleton Laboratory, Chilton, Didcot U.K., November 1998.

[3] Roger L. Mansfield, "Ephemeris of a Comet via Uniform Path Mechanics," Math in Action, MathSoft, Inc. (http://mathsoft.com/appsindex.html).

[4] P. Kenneth Seidelmann, et al., Explanatory Supplement to the Astronomical Almanac, University Science Books, Mill Valley, California (1992).

[5] H.M. Nautical Almanac Office, Royal Greenwich Observatory and Nautical Almanac Office, U.S. Naval Observatory, Planetary and Lunar Coordinates for the Years 1984-2000, London and Washington, January 1983.

[6] Nautical Almanac Office, U.S. Naval Observatory, Multi-Year Interactive Computer Almanac (MICA) 1990-2005, Willmann-Bell, Richmond, VA (http://www.willbell.com).

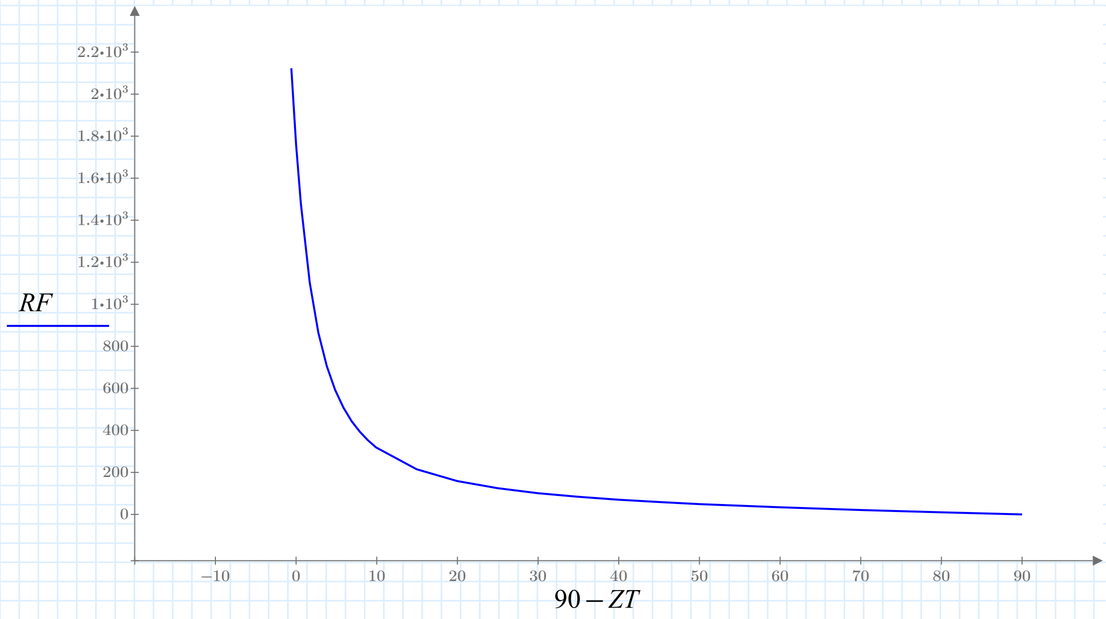
[7] C. W. Allen, Astrophysical Quantities, Athlone Press, University of London, Third Edition (1973), pp. 124-125.

[8] Oliver Montenbruck and Thomas Pfleger, Astronomy on the Personal Computer, Fourth Edition (2000), Springer-Verlag, New York.

[9] Paul J. Heafner, Fundamental Ephemeris Computations, Willmann-Bell, Richmond, VA (1999).

Appendix 1 - Validation of **RFN**

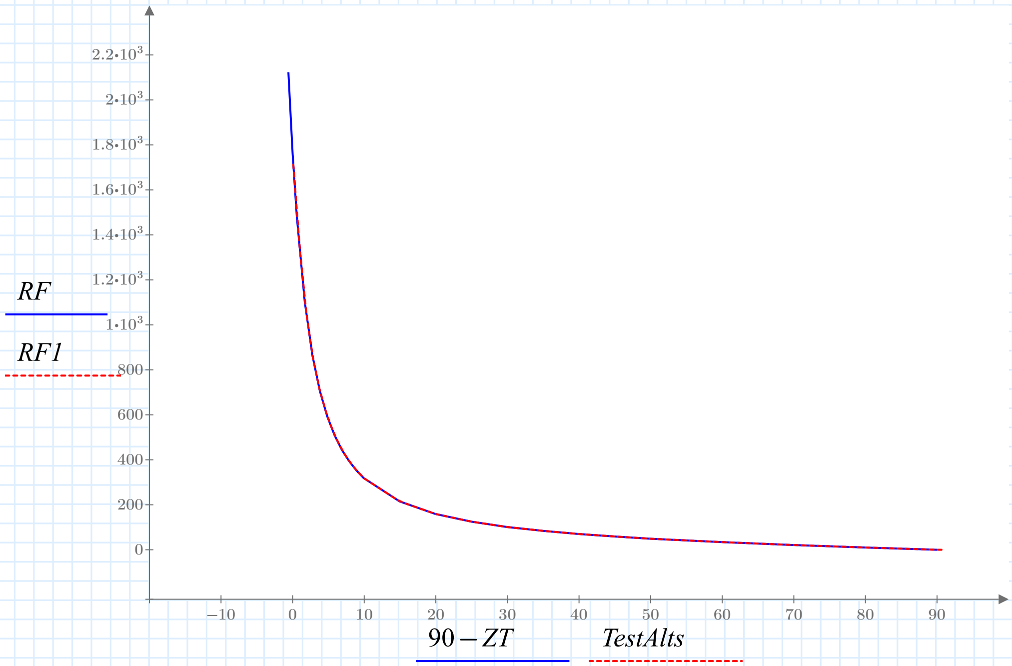
We need to verify that **RFN** interpolates correctly between all of the discrete points in the refraction table. What we can to is to plot the tabular values of refraction in blue, then define a vector that samples intermediate values and plot them in red. We will then be able to see, by inspection, that **RFN** is working correctly. First we plot the tabular refraction vs. altitude to see what the curve looks like.



Now we define a vector of sample points, **TestAlts**, with the sample points chosen to be within each tabular interval, and invoke **RFN** with argument **TestAlts**. We multiply each result by SecPerDeg since **RFN** converted the result from arc-seconds to degrees.

$$TestAlts := \left\| \begin{array}{l} Alt_1 \leftarrow 0.1 \\ \Delta Alt \leftarrow \dfrac{ZT_{24}}{100} \\ \text{for } i \in 2 \,..\, 101 \\ \quad \| \; Alt_i \leftarrow Alt_{i-1} + \Delta Alt \\ Alt \end{array} \right.$$

$$RF1 := RFN(TestAlts) \cdot SecPerDeg$$



We see that the interpolated points, in red, lie very close to the tabular points, in blue, which was to be expected if the logic of interpolation in procedural function **RFN** is correct.