How to compute planetary positions

By **Paul Schlyter, Stockholm, Sweden**
email: pausch@stjarnhimlen.se or WWW: http://stjarnhimlen.se/

Chapters

1. Foreword

Below is a description of how to compute the positions for the Sun and Moon and the major planets, as well as for comets and minor planets, from a set of orbital elements.

The algorithms have been simplified as much as possible while still keeping a fairly good accuracy. The accuracy of the computed positions is a fraction of an arc minute for the Sun and the inner planets, about one arc minute for the outer planets, and 1-2 arc minutes for the Moon. If we limit our accuracy demands to this level, one can simplify further by e.g. ignoring the difference between mean, true and apparent positions.

The positions computed below are for the 'equinox of the day', which is suitable for computing rise/set times, but not for plotting the position on a star map drawn for a fixed epoch. In the latter case, correction for precession must be applied, which is most simply performed as a rotation along the ecliptic.

These algorithms were developed by myself back in 1979, based on a preprint from T. van Flandern's and K. Pulkkinen's paper "Low precision formulae for planetary positions", published in the Astrophysical Journal Supplement Series, 1980. It's basically a simplification of these algorithms, while keeping a reasonable accuracy. They were first implemented on a HP-41C programmable pocket calculator, in 1979, and ran in less than 2 KBytes of RAM! Nowadays considerable more accurate algorithms are available of course, as well as more powerful computers. Nevertheless I've retained these algorithms as what I believe is the simplest way to compute solar/lunar positions with an accuracy of 1-2 arc minutes.

2. Introduction

The text below describes how to compute the positions in the sky of the Sun, Moon and the major planets out to Neptune. The algorithm for Pluto is taken from a fourier fit to Pluto's position as computed by numerical integration at JPL. Positions of other celestial bodies as well (i.e. comets and asteroids) can also be computed, if their orbital elements are available.

These formulae may seem complicated, but I believe this is the simplest method to compute planetary positions with the fairly good accuracy of about one arc minute (=1/60 degree). Any further simplifications will yield lower accuracy, but of course that may be ok, depending on the application.

3. A few words about accuracy

The accuracy requirements are modest: a final position with an error of no more than 1-2 arc minutes (one arc minute = 1/60 degree). This accuracy is in one respect quite optimal: it is the highest accuracy one can strive for, while still being able to do many simplifications. The simplifications made here are:

1: Nutation and aberration are both ignored.
2: Planetary aberration (i.e. light travel time) is ignored.
3: The difference between Terrestial Time/Ephemeris Time (TT/ET), and Universal Time (UT) is ignored.
4: Precession is computed in a simplified way, by a simple addition to the ecliptic longitude.
5: Higher-order terms in the planetary orbital elements are ignored. This will give an additional error of up to 2 arc min in 1000 years from now. For the Moon, the error will be larger: 7 arc min 1000 years from now. This error will grow as the square of the time from the present.
6: Most planetary perturbations are ignored. Only the major perturbation terms for the Moon, Jupiter, Saturn, and Uranus, are included. If still lower accuracy is acceptable, these perturbations can be ignored as well.
7: The largest Uranus-Neptune perturbation is accounted for in the orbital elements of these planets. Therefore, the orbital elements of Uranus and Neptune are less accurace, especially in the distant past and future. The elements for these planets should therefore only be used for at most a few centuries into the past and the future.

4. The time scale

The time scale in these formulae are counted in days. Hours, minutes, seconds are expressed as fractions of a day. Day 0.0 occurs at 2000 Jan 0.0 UT (or 1999 Dec 31, 0:00 UT). This "day number" d is computed as follows (y=year, m=month, D=date, UT=UT in hours+decimals):

```
d = 367*y - 7 * ( y + (m+9)/12 ) / 4 + 275*m/9 + D - 730530
```
Note that ALL divisions here should be INTEGER divisions. In Pascal, use "div" instead of "/", in MS-Basic, use "\" instead of "/". In Fortran, C and C++ "/" can be used if both y and m are integers. Finally, include the time of the day, by adding:

```
d = d + UT/24.0          (this is a floating-point division)
```

5. The orbital elements

The primary orbital elements are here denoted as:

```
N = longitude of the ascending node
i = inclination to the ecliptic (plane of the Earth's orbit)
w = argument of perihelion
a = semi-major axis, or mean distance from Sun
e = eccentricity (0=circle, 0-1=ellipse, 1=parabola)
M = mean anomaly (0 at perihelion; increases uniformly with time)
```

Related orbital elements are:

```
w1 = N + w   = longitude of perihelion
L  = M + w1  = mean longitude
q  = a*(1-e) = perihelion distance
Q  = a*(1+e) = aphelion distance
P  = a ^ 1.5 = orbital period (years if a is in AU, astronomical units)
T  = Epoch_of_M - (M(deg)/360_deg) / P  = time of perihelion
v  = true anomaly (angle between position and perihelion)
E  = eccentric anomaly
```

One *Astronomical Unit (AU)* is the Earth's mean distance to the Sun, or 149.6 million km. When closest to the Sun, a planet is in *perihelion*, and when most distant from the Sun it's in *aphelion*. For the Moon, an artificial satellite, or any other body orbiting the Earth, one says *perigee* and *apogee* instead, for the points in orbit least and most distant from Earth.

To describe the position in the orbit, we use three angles: Mean Anomaly, True Anomaly, and Eccentric Anomaly. They are all zero when the planet is in perihelion:

Mean Anomaly (M): This angle increases uniformly over time, by 360 degrees per orbital period. It's zero at perihelion. It's easily computed from the orbital period and the time since last perihelion.

True Anomaly (v): This is the actual angle between the planet and the perihelion, as seen from the central body (in this case the Sun). It increases non-uniformly with time, changing most rapidly at perihelion.

Eccentric Anomaly (E): This is an auxiliary angle used in Kepler's Equation, when computing the True Anomaly from the Mean Anomaly and the orbital eccentricity.

Note that for a circular orbit (eccentricity=0), these three angles are all equal to each other.

Another quantity we will need is ecl, the *obliquity of the ecliptic*, i.e. the "tilt" of the Earth's axis of rotation (currently 23.4 degrees and slowly decreasing). First, compute the "d" of the moment of interest ([section 3](#)). Then, compute the obliquity of the ecliptic:

```
ecl = 23.4393 - 3.563E-7 * d
```

Now compute the orbital elements of the planet of interest. If you want the position of the Sun or the Moon, you only need to compute the orbital elements for the Sun or the Moon. If you want the position of any other planet, you must compute the orbital elements for that planet *and* for the Sun (of course the orbital elements for the Sun are really the orbital elements for the Earth; however it's customary to here pretend that the Sun orbits the Earth). This is necessary to be able to compute the geocentric position of the planet.

Please note that a, the semi-major axis, is given in Earth radii for the Moon, but in Astronomical Units for the Sun and all the planets.

When computing M (and, for the Moon, when computing N and w as well), one will quite often get a result that is larger than 360 degrees, or negative (all angles are here computed in degrees). If negative, add 360 degrees until positive. If larger than 360 degrees, subtract 360 degrees until the value is less than 360 degrees. Note that, in most programming languages, one must then multiply these angles with pi/180 to convert them to radians, before taking the sine or cosine of them.

Orbital elements of the Sun:

```
N = 0.0
i = 0.0
w = 282.9404 + 4.70935E-5 * d
a = 1.000000   (AU)
e = 0.016709 - 1.151E-9 * d
M = 356.0470 + 0.9856002585 * d
```

Orbital elements of the Moon:

```
N = 125.1228 - 0.0529538083 * d
i = 5.1454
w = 318.0634 + 0.1643573223 * d
a = 60.2666   (Earth radii)
e = 0.054900
M = 115.3654 + 13.0649929509 * d
```
Orbital elements of Mercury:

```
N =  48.3313 + 3.24587E-5 * d
i = 7.0047 + 5.00E-8 * d
w =  29.1241 + 1.01444E-5 * d
a = 0.387098   (AU)
e = 0.205635 + 5.59E-10 * d
M = 168.6562 + 4.0923344368 * d
```
Orbital elements of Venus:

```
N =  76.6799 + 2.46590E-5 * d
i = 3.3946 + 2.75E-8 * d
w =  54.8910 + 1.38374E-5 * d
a = 0.723330   (AU)
e = 0.006773 - 1.302E-9 * d
M =  48.0052 + 1.6021302244 * d
```
Orbital elements of Mars:

```
N =  49.5574 + 2.11081E-5 * d
i = 1.8497 - 1.78E-8 * d
w = 286.5016 + 2.92961E-5 * d
a = 1.523688   (AU)
e = 0.093405 + 2.516E-9 * d
M =  18.6021 + 0.5240207766 * d
```
Orbital elements of Jupiter:

```
N = 100.4542 + 2.76854E-5 * d
i = 1.3030 - 1.557E-7 * d
w = 273.8777 + 1.64505E-5 * d
a = 5.20256   (AU)
e = 0.048498 + 4.469E-9 * d
M =  19.8950 + 0.0830853001 * d
```
Orbital elements of Saturn:

```
N = 113.6634 + 2.38980E-5 * d
i = 2.4886 - 1.081E-7 * d
w = 339.3939 + 2.97661E-5 * d
a = 9.55475   (AU)
e = 0.055546 - 9.499E-9 * d
M = 316.9670 + 0.0334442282 * d
```
Orbital elements of Uranus:

```
N =  74.0005 + 1.3978E-5 * d
i = 0.7733 + 1.9E-8 * d
w =  96.6612 + 3.0565E-5 * d
a = 19.18171 - 1.55E-8 * d   (AU)
e = 0.047318 + 7.45E-9 * d
M = 142.5905 + 0.011725806 * d
```
Orbital elements of Neptune:

```
N = 131.7806 + 3.0173E-5 * d
i = 1.7700 - 2.55E-7 * d
w = 272.8461 - 6.027E-6 * d
a = 30.05826 + 3.313E-8 * d   (AU)
e = 0.008606 + 2.15E-9 * d
M = 260.2471 + 0.005995147 * d
```
Please note than the orbital elements of Uranus and Neptune as given here are somewhat less accurate. They include a long period perturbation between Uranus and Neptune. The period of the perturbation is about 4200 years. Therefore, these elements should not be expected to give results within the stated accuracy for more than a few centuries in the past and into the future.

6. The Sun

a) The position

The position of the Sun is computed just like the position of any other planet, but since the Sun always is moving in the ecliptic, and since the eccentricity of the orbit is quite small, a few simplifications can be made. Therefore, a separate presentation for the Sun is given.

Of course, we're here really computing the position of the Earth in its orbit around the Sun, but since we're viewing the sky from an Earth-centered perspective, we'll pretend that the Sun is in orbit around the Earth instead.

First, compute the eccentric anomaly E from the mean anomaly M and from the eccentricity e (E and M in degrees):

```
    E = M + e*(180/pi) * sin(M) * ( 1.0 + e * cos(M) )
```
or (if E and M are expressed in radians):

```
    E = M + e * sin(M) * ( 1.0 + e * cos(M) )
```
Note that the formulae for computing E are not exact; however they're accurate enough here.

Then compute the Sun's distance r and its true anomaly v from:

```
    xv = r * cos(v) = cos(E) - e
    yv = r * sin(v) = sqrt(1.0 - e*e) * sin(E)

    v = atan2( yv, xv )
    r = sqrt( xv*xv + yv*yv )
```
(note that the r computed here is later used as rs)

atan2() is a function that converts an x,y coordinate pair to the correct angle in all four quadrants. It is available as a library function in Fortran, C and C++. In other languages, one has to write one's own atan2() function. It's not that difficult:

```
    atan2( y, x ) = atan(y/x)                if x positive
    atan2( y, x ) = atan(y/x) +- 180 degrees if x negative
    atan2( y, x ) = sign(y) * 90 degrees     if x zero
```
See these links for some code in Basic or Pascal. Fortran and C/C++ already has atan2() as a standard library function.

Now, compute the Sun's true longitude:

```
    lonsun = v + w
```
Convert lonsun,r to ecliptic rectangular geocentric coordinates xs,ys:

```
    xs = r * cos(lonsun)
    ys = r * sin(lonsun)
```
(since the Sun always is in the ecliptic plane, zs is of course zero). xs,ys is the Sun's position in a coordinate system in the plane of the ecliptic. To convert this to equatorial, rectangular, geocentric coordinates, compute:

```
    xe = xs
    ye = ys * cos(ecl)
    ze = ys * sin(ecl)
```
Finally, compute the Sun's Right Ascension (RA) and Declination (Dec):

```
    RA  = atan2( ye, xe )
    Dec = atan2( ze, sqrt(xe*xe+ye*ye) )
```

b) Sidereal Time

Quite often we need a quantity called Sidereal Time. The Local Sideral Time (LST) is simply the RA of your local meridian. The Greenwich Mean Sideral Time (GMST) is the LST at Greenwich. And, finally, the Greenwich Mean Sidereal Time at 0h UT (GMST0) is, as the name says, the GMST at Greenwich Midnight. However, we will here extend the concept of GMST0 a bit, by letting "our" GMST0 be the same as the conventional GMST0 at UT midnight but also let GMST0 be defined at any other time such that GMST0 will increase by 3m51s every 24 hours. Then this formula will be valid at any time:

```
GMST = GMST0 + UT
```
We also need the Sun's mean longitude, Ls, which can be computed from the Sun's v and w as follows:

```
Ls = v + w
```
The GMST0 is easily computed from Ls (divide by 15 if you want GMST0 in hours rather than degrees), GMST is then computed by adding the UT, and finally the LST is computed by adding your local longitude (east longitude is positive, west negative).

Note that "time" is given in hours while "angle" is given in degrees. The two are related to one another due to the Earth's rotation: one hour is here the same as 15 degrees. Before adding or subtracting a "time" and an "angle", be sure to convert them to the same unit, e.g. degrees by multiplying the hours by 15 before adding/subtracting:

```
GMST0 = Ls + 180_degrees
GMST = GMST0 + UT
LST  = GMST + local_longitude
```
The formulae above are written as if times are expressed in degrees. If we instead assume times are given in hours and angles in degrees, and if we explicitly write out the conversion factor of 15, we get:

```
GMST0 = (Ls + 180_degrees)/15 = Ls/15 + 12_hours
GMST = GMST0 + UT
LST  = GMST + local_longitude/15
```

7. The position

a) Of the Moon and of the planets

First, compute the eccentric anomaly, E, from M, the mean anomaly, and e, the eccentricity. As a first approximation, do (E and M in degrees):

```
E = M + e*(180/pi) * sin(M) * ( 1.0 + e * cos(M) )
```
or, if E and M are in radians:

```
E = M + e * sin(M) * ( 1.0 + e * cos(M) )
```
If e, the eccentricity, is less than about 0.05-0.06, this approximation is sufficiently accurate. If the eccentricity is larger, set E0=E and then use this iteration formula (E and M in degrees):

```
E1 = E0 - ( E0 - e*(180/pi) * sin(E0) - M ) / ( 1 - e * cos(E0) )
```
or (E and M in radians):

```
E1 = E0 - ( E0 - e * sin(E0) - M ) / ( 1 - e * cos(E0) )
```
For each new iteration, replace E0 with E1. Iterate until E0 and E1 are sufficiently close together (about 0.001 degrees). For comet orbits with eccentricites close to one, a difference of less than 1E-4 or 1E-5 degrees should be required.

If this iteration formula won't converge, the eccentricity is probably too close to one. Then you should instead use the formulae for near-parabolic or parabolic orbits.

Now compute the planet's distance and true anomaly:

```
xv = r * cos(v) = a * ( cos(E) - e )
yv = r * sin(v) = a * ( sqrt(1.0 - e*e) * sin(E) )

v = atan2( yv, xv )
r = sqrt( xv*xv + yv*yv )
```

b) In space

Compute the planet's position in 3-dimensional space:

```
xh = r * ( cos(N) * cos(v+w) - sin(N) * sin(v+w) * cos(i) )
yh = r * ( sin(N) * cos(v+w) + cos(N) * sin(v+w) * cos(i) )
zh = r * ( sin(v+w) * sin(i) )
```
For the Moon, this is the geocentric (Earth-centered) position in the ecliptic coordinate system. For the planets, this is the heliocentric (Sun-centered) position, also in the ecliptic coordinate system. If one wishes, one can compute the ecliptic longitude and latitude (this must be done if one wishes to correct for perturbations, or if one wants to precess the position to a standard epoch):

```
lonecl = atan2( yh, xh )
latecl = atan2( zh, sqrt(xh*xh+yh*yh) )
```
As a check one can compute sqrt(xh*xh+yh*yh+zh*zh), which of course should equal r (except for small round-off errors).

8. Precession

If one wishes to compute the planet's position for some standard epoch, such as 1950.0 or 2000.0 (e.g. to be able to plot the position on a star atlas), one must add the correction below to lonecl. If a planet's and not the Moon's position is computed, one must also add the same correction to lonsun, the Sun's longitude. The desired Epoch is expressed as the year, possibly with a fraction.

```
lon_corr = 3.82394E-5 * ( 365.2422 * ( Epoch - 2000.0 ) - d )
```
If one wishes the position for today's epoch (useful when computing rising/setting times and the like), no corrections need to be done.

9. Perturbations

a) Of the Moon

If the position of the Moon is computed, and one wishes a better accuracy than about 2 degrees, the most important perturbations has to be taken into account. If one wishes 2 arc minute accuracy, all the following terms should be accounted for. If less accuracy is needed, some of the smaller terms can be omitted.

First compute:

```
Ms, Mm              Mean Anomaly of the Sun and the Moon
Nm                  Longitude of the Moon's node
ws, wm              Argument of perihelion for the Sun and the Moon
Ls = Ms + ws        Mean Longitude of the Sun   (Ns=0)
Lm = Mm + wm + Nm   Mean longitude of the Moon
D = Lm - Ls         Mean elongation of the Moon
F = Lm - Nm         Argument of latitude for the Moon
```

Add these terms to the Moon's longitude (degrees):

```
-1.274 * sin(Mm - 2*D)          (the Evection)
+0.658 * sin(2*D)               (the Variation)
-0.186 * sin(Ms)                (the Yearly Equation)
-0.059 * sin(2*Mm - 2*D)
-0.057 * sin(Mm - 2*D + Ms)
+0.053 * sin(Mm + 2*D)
+0.046 * sin(2*D - Ms)
+0.041 * sin(Mm - Ms)
-0.035 * sin(D)                 (the Parallactic Equation)
-0.031 * sin(Mm + Ms)
-0.015 * sin(2*F - 2*D)
+0.011 * sin(Mm - 4*D)
```

Add these terms to the Moon's latitude (degrees):

```
-0.173 * sin(F - 2*D)
-0.055 * sin(Mm - F - 2*D)
-0.046 * sin(Mm + F - 2*D)
+0.033 * sin(F + 2*D)
+0.017 * sin(2*Mm + F)
```

Add these terms to the Moon's distance (Earth radii):

```
-0.58 * cos(Mm - 2*D)
-0.46 * cos(2*D)
```

All perturbation terms that are smaller than 0.01 degrees in longitude or latitude and smaller than 0.1 Earth radii in distance have been omitted here. A few of the largest perturbation terms even have their own names! The Evection (the largest perturbation) was discovered already by Ptolemy a few thousand years ago (the Evection was one of Ptolemy's epicycles). The Variation and the Yearly Equation were both discovered by Tycho Brahe in the 16'th century.

The computations can be simplified by omitting the smaller perturbation terms. The error introduced by this seldom exceeds the sum of the amplitudes of the 4-5 largest omitted terms. If one only computes the three largest perturbation terms in longitude and the largest term in latitude, the error in longitude will rarley exceed 0.25 degrees, and in latitude 0.15 degrees.

b) Of Jupiter, Saturn and Uranus

The only planets having perturbations larger than 0.01 degrees are Jupiter, Saturn and Uranus. First compute:

```
Mj    Mean anomaly of Jupiter
Ms    Mean anomaly of Saturn
Mu    Mean anomaly of Uranus (needed for Uranus only)
```

Perturbations for Jupiter. Add these terms to the longitude:

```
-0.332 * sin(2*Mj - 5*Ms - 67.6 degrees)
-0.056 * sin(2*Mj - 2*Ms + 21 degrees)
+0.042 * sin(3*Mj - 5*Ms + 21 degrees)
-0.036 * sin(Mj - 2*Ms)
+0.022 * cos(Mj - Ms)
+0.023 * sin(2*Mj - 3*Ms + 52 degrees)
-0.016 * sin(Mj - 5*Ms - 69 degrees)
```

Perturbations for Saturn. Add these terms to the longitude:

```
+0.812 * sin(2*Mj - 5*Ms - 67.6 degrees)
-0.229 * cos(2*Mj - 4*Ms - 2 degrees)
+0.119 * sin(Mj - 2*Ms - 3 degrees)
+0.046 * sin(2*Mj - 6*Ms - 69 degrees)
+0.014 * sin(Mj - 3*Ms + 32 degrees)
```

For Saturn: *also* add these terms to the latitude:

```
-0.020 * cos(2*Mj - 4*Ms - 2 degrees)
+0.018 * sin(2*Mj - 6*Ms - 49 degrees)
```

Perturbations for Uranus: Add these terms to the longitude:

```
+0.040 * sin(Ms - 2*Mu + 6 degrees)
+0.035 * sin(Ms - 3*Mu + 33 degrees)
-0.015 * sin(Mj - Mu + 20 degrees)
```

The "great Jupiter-Saturn term" is the largest perturbation for both Jupiter and Saturn. Its period is 918 years, and its amplitude is 0.332 degrees for Jupiter and 0.812 degrees for Saturn. These is also a "great Saturn-Uranus term", period 560 years, amplitude 0.035 degrees for Uranus, less than 0.01 degrees for Saturn (and therefore omitted). The other perturbations have periods between 14 and 100 years. One should also mention the "great Uranus-Neptune term", which has a period of 4220 years and an amplitude of about one degree. It is not included here, instead it is included in the orbital elements of Uranus and Neptune.

For Mercury, Venus and Mars we can ignore all perturbations. For Neptune the only significant perturbation is already included in the orbital elements, as mentioned above, and therefore no further perturbation terms need to be accounted for.

10. Coordinates

a) Geocentric (Earth-centered) coordinates

Now we have computed the heliocentric (Sun-centered) coordinate of the planet, and we have included the most important perturbations. We want to compute the geocentric (Earth-centerd) position. We should convert the perturbed lonecl, latecl, r to (perturbed) xh, yh, zh:

```
xh = r * cos(lonecl) * cos(latecl)
yh = r * sin(lonecl) * cos(latecl)
zh = r             * sin(latecl)
```

If we are computing the Moon's position, this is already the geocentric position, and thus we simply set xg=xh, yg=yh, zg=zh. Otherwise we must also compute the Sun's position: convert lonsun, rs (where rs is the r computed [here]) to xs, ys:

```
xs = rs * cos(lonsun)
ys = rs * sin(lonsun)
```

(Of course, any correction for precession should be added to lonecl *and* lonsun *before* converting to xh,yh,zh and xs,ys).

Now convert from heliocentric to geocentric position:

```
xg = xh + xs
yg = yh + ys
```

```
        zg = zh
```
We now have the planet's geocentric (Earth centered) position in rectangular, ecliptic coordinates.

b) Equatorial coordinates

Let's convert our rectangular, ecliptic coordinates to rectangular, equatorial coordinates: simply rotate the y-z-plane by ecl, the angle of the obliquity of the ecliptic:

```
xe = xg
ye = yg * cos(ecl) - zg * sin(ecl)
ze = yg * sin(ecl) + zg * cos(ecl)
```
Finally, compute the planet's Right Ascension (RA) and Declination (Dec):

```
RA  = atan2( ye, xe )
Dec = atan2( ze, sqrt(xe*xe+ye*ye) )
```
Compute the geocentric distance:

```
rg = sqrt(xg*xg+yg*yg+zg*zg) = sqrt(xe*xe+ye*ye+ze*ze)
```
Thie completes our computation of the equatorial coordinates.

c) Azimuthal coordinates

To find the azimuthal coordinates (azimuth and altitude) we proceed by computing the HA (Hour Angle) of the object. But first we must compute the LST (Local Sidereal Time), which we do as described in 5b above. When we know LST, we can easily compute HA from:

```
HA = LST - RA
```
HA is usually given in the interval -12 to +12 hours, or -180 to +180 degrees. If HA is zero, the object can be seen directly to the south. If HA is negative, the object is to the east of south, and if HA is positive, the object is to the west of south. IF your computed HA should fall outside this interval, add or subtract 24 hours (or 360 degrees) until HA falls within this interval.

Now it's time to convert our objects HA and Decl to local azimuth and altitude. To do that, we also must know lat, our local latitude. Then we proceed as follows:

```
x = cos(HA) * cos(Decl)
y = sin(HA) * cos(Decl)
z = sin(Decl)

xhor = x * sin(lat) - z * cos(lat)
yhor = y
zhor = x * cos(lat) + z * sin(lat)

az  = atan2( yhor, xhor ) + 180_degrees
alt = asin( zhor ) = atan2( zhor, sqrt(xhor*xhor+yhor*yhor) )
```
This completes our calculation of the local azimuth and altitude. Note that azimuth is 0 at North, 90 deg at East, 180 deg at South and 270 deg at West. Altitude is of course 0 at the (mathematical) horizon, 90 deg at zenith, and negative below the horizon.

11. The Moon's topocentric position

The Moon's position, as computed earlier, is geocentric, i.e. as seen by an imaginary observer at the center of the Earth. Real observers dwell on the surface of the Earth, though, and they will see a different position - the topocentric position. This position can differ by more than one degree from the geocentric position. To compute the topocentric positions, we must add a correction to the geocentric position.

Let's start by computing the Moon's parallax, i.e. the apparent size of the (equatorial) radius of the Earth, as seen from the Moon:

```
mpar = asin( 1/r )
```
where r is the Moon's distance in Earth radii. It's simplest to apply the correction in horizontal coordinates (azimuth and altitude): within our accuracy aim of 1-2 arc minutes, no correction need to be applied to the azimuth. One need only apply a correction to the altitude above the horizon:

```
alt_topoc = alt_geoc - mpar * cos(alt_geoc)
```
Sometimes one need to correct for topocentric position directly in equatorial coordinates though, e.g. if one wants to draw on a star map how the Moon passes in front of the Pleiades, as seen from some specific location. Then we need to know the Moon's geocentric Right Ascension and Declination (RA, Decl), the Local Sidereal Time (LST), and our latitude (lat).

Our astronomical latitude (lat) must first be converted to a geocentric latitude (gclat), and distance from the center of the Earth (rho) in Earth equatorial radii. If we only want an approximate topocentric position, it's simplest to pretend that the Earth is a perfect sphere, and simply set:

```
gclat = lat,  rho = 1.0
```
However, if we do wish to account for the flattening of the Earth, we instead compute:

```
gclat = lat - 0.1924_deg * sin(2*lat)
rho   = 0.99833 + 0.00167 * cos(2*lat)
```
Next we compute the Moon's geocentric Hour Angle (HA) from the Moon's geocentric RA. First we must compute LST as described in [5b](#) above, then we compute HA as:

```
HA = LST - RA
```
We also need an auxiliary angle, g:

```
g = atan( tan(gclat) / cos(HA) )
```
Now we're ready to convert the geocentric Right Ascension and Declination (RA, Decl) to their topocentric values (topRA, topDecl):

```
topRA  = RA  - mpar * rho * cos(gclat) * sin(HA) / cos(Decl)
topDecl = Decl - mpar * rho * sin(gclat) * sin(g - Decl) / sin(g)
```
(Note that if decl is exactly 90 deg, cos(Decl) becomes zero and we get a division by zero when computing topRA, but that formula breaks down only very close to the celestial poles anyway and we never see the Moon there. Also if gclat is precisely zero, g becomes zero too, and we get a division by zero when computing topDecl. In that case, replace the formula for topDecl with

```
topDecl = Decl - mpar * rho * sin(-Decl) * cos(HA)
```
which is valid for gclat equal to zero; it can also be used for gclat extremely close to zero).

This correction to topocentric position can also be applied to the Sun and the planets. But since they're much farther away, the correction becomes much smaller. It's largest for Venus at inferior conjunction, when Venus' parallax is somewhat larger than 32 arc seconds. Within our aim of obtaining a final accuracy of 1-2 arc minutes, it might barely be justified to correct to topocentric position when Venus is close to inferior conjunction, and perhaps also when Mars is at a favourable opposition. But in all other cases this correction can safely be ignored within our accuracy aim. We only need to worry about the Moon in this case.

If you want to compute topocentric coordinates for the planets too, you do it the same way as for the Moon, with one exception: the Moon's parallax is replaced by the parallax of the planet (ppar), as computed from this formula:

```
ppar = (8.794/3600)_deg / r
```
where r is the distance of the planet from the Earth, in astronomical units.

12. The elongation and physical ephemerides of the planets

When we finally have completed our computation of the heliocentric and geocentric coordinates of the planets, it could also be interesting to know what the planet will look like. How large will it appear? What's its phase and magnitude (brightness)? These computations are much simpler than the computations of the positions.

Let's start by computing the apparent diameter of the planet:

```
d = d0 / R
```
R is the planet's geocentric distance in astronomical units, and d is the planet's apparent diameter at a distance of 1 astronomical unit. d0 is of course different for each planet. The values below are given in seconds of arc. Some planets have different equatorial and polar diameters:

```
Mercury      6.74"
Venus       16.92"
Earth       17.59" equ     17.53" pol
Mars         9.36" equ      9.28" pol
Jupiter    196.94" equ    185.08" pol
Saturn     165.6"  equ    150.8"  pol
Uranus      65.8"  equ     62.1"  pol
Neptune     62.2"  equ     60.9"  pol
```
The Sun's apparent diameter at 1 astronomical unit is 1919.26". The Moon's apparent diameter is:

```
d = 1873.7" * 60 / r
```
where r is the Moon's distance in Earth radii.

Two other quantities we'd like to know are the phase angle and the elongation.

The phase angle tells us the phase: if it's zero the planet appears "full", if it's 90 degrees it appears "half", and if it's 180 degrees it appears "new". Only the Moon and the inferior planets (i.e. Mercury and Venus) can have phase angles exceeding about 50 degrees.

The elongation is the apparent angular distance of the planet from the Sun. If the elongation is smaller than about 20 degrees, the planet is hard to observe, and if it's smaller than about 10 degrees it's usually not possible to observe the planet.

To compute phase angle and elongation we need to know the planet's heliocentric distance, r, its geocentric distance, R, and the distance to the Sun, s. Now we can compute the phase angle, FV, and the elongation, elong:

```
elong = acos( ( s*s + R*R - r*r ) / (2*s*R) )

FV    = acos( ( r*r + R*R - s*s ) / (2*r*R) )
```
When we know the phase angle, we can easily compute the phase:

```
phase  =  ( 1 + cos(FV) ) / 2  =  hav(180_deg - FV)
```
hav is the "haversine" function. The "haversine" (or "half versine") is an old and now obsolete trigonometric function. It's defined as:

```
hav(x)  =  ( 1 - cos(x) ) / 2  =   sin^2 (x/2)
```
As usual we must use a different procedure for the Moon. Since the Moon is so close to the Earth, the procedure above would introduce too big errors. Instead we use the Moon's ecliptic longitude and latitude, mlon and mlat, and the Sun's ecliptic longitude, mlon, to compute first the elongation, then the phase angle, of the Moon:

```
elong = acos( cos(slon - mlon) * cos(mlat) )

FV = 180_deg - elong
```
Finally we'll compute the magnitude (or brightness) of the planets. Here we need to use a formula that's different for each planet. FV is the phase angle (in degrees), r is the heliocentric and R the geocentric distance (both in AU):

```
Mercury:   -0.36 + 5*log10(r*R) + 0.027 * FV + 2.2E-13 * FV**6
```

```
Venus:     -4.34 + 5*log10(r*R) + 0.013 * FV + 4.2E-7  * FV**3
Mars:      -1.51 + 5*log10(r*R) + 0.016 * FV
Jupiter:   -9.25 + 5*log10(r*R) + 0.014 * FV
Saturn:    -9.0  + 5*log10(r*R) + 0.044 * FV + ring_magn
Uranus:    -7.15 + 5*log10(r*R) + 0.001 * FV
Neptune:   -6.90 + 5*log10(r*R) + 0.001 * FV

Moon:      +0.23 + 5*log10(r*R) + 0.026 * FV + 4.0E-9 * FV**4
```

** is the power operator, thus FV**6 is the phase angle (in degrees) raised to the sixth power. If FV is 150 degrees then FV**6 becomes ca 1.14E+13, which is a quite large number.

For the Moon, we also need the heliocentric distance, r, and geocentric distance, R, in AU (astronomical units). Here r can be set equal to the Sun's geocentric distance in AU. The Moon's geocentric distance, R, previously computed i Earth radii, must be converted to AU's - we do this by multiplying by sin(17.59"/2) = 1/23450. Or we could modify the magnitude formula for the Moon so it uses r in AU's and R in Earth radii:

```
Moon:      -21.62 + 5*log10(r*R) + 0.026 * FV + 4.0E-9 * FV**4
```

Saturn needs special treatment due to its rings: when Saturn's rings are "open" then Saturn will appear much brighter than when we view Saturn's rings edgewise. We'll compute ring_mang like this:

```
ring_magn = -2.6 * sin(abs(B)) + 1.2 * (sin(B))**2
```

Here B is the tilt of Saturn's rings which we also need to compute. Then we start with Saturn's geocentric ecliptic longitude and latitude (los, las) which we've already computed. We also need the tilt of the rings to the ecliptic, ir, and the "ascending node" of the plane of the rings, Nr:

```
ir = 28.06_deg
Nr = 169.51_deg + 3.82E-5_deg * d
```

Here d is our "day number" which we've used so many times before. Now let's compute the tilt of the rings:

```
B = asin( sin(las) * cos(ir) - cos(las) * sin(ir) * sin(los-Nr) )
```

This concludes our computation of the magnitudes of the planets.

13. The position

a) Of Pluto

No analytical theory has ever been constructed for the planet Pluto. Our most accurate representation of the motion of this planet is from numerical integrations. Yet, a "curve fit" may be performed to these numerical integrations, and the result will be the formulae below, valid from about 1800 to about 2100. Compute d, our day number, as usual ([section 3](#)). Then compute these angles:

```
S  =    50.03  +  0.033459652 * d
P  =   238.95  +  0.003968789 * d
```

Next compute the heliocentric ecliptic longitude and latitude (degrees), and distance (a.u.):

```
lonecl = 238.9508  +  0.00400703 * d
          - 19.799 * sin(P)      + 19.848 * cos(P)
          +  0.897 * sin(2*P)    -  4.956 * cos(2*P)
          +  0.610 * sin(3*P)    +  1.211 * cos(3*P)
          -  0.341 * sin(4*P)    -  0.190 * cos(4*P)
          +  0.128 * sin(5*P)    -  0.034 * cos(5*P)
          -  0.038 * sin(6*P)    +  0.031 * cos(6*P)
          +  0.020 * sin(S-P)    -  0.010 * cos(S-P)

  latecl =  -3.9082
          -  5.453 * sin(P)      - 14.975 * cos(P)
          +  3.527 * sin(2*P)    +  1.673 * cos(2*P)
          -  1.051 * sin(3*P)    +  0.328 * cos(3*P)
          +  0.179 * sin(4*P)    -  0.292 * cos(4*P)
          +  0.019 * sin(5*P)    +  0.100 * cos(5*P)
          -  0.031 * sin(6*P)    -  0.026 * cos(6*P)
                                 +  0.011 * cos(S-P)

  r     =   40.72
          +  6.68 * sin(P)       +  6.90 * cos(P)
          -  1.18 * sin(2*P)     -  0.03 * cos(2*P)
          +  0.15 * sin(3*P)     -  0.14 * cos(3*P)
```

Now we know the heliocentric distance and ecliptic longitude/latitude for Pluto. To convert to geocentric coordinates, do as for the other planets.

b) Of asteroids

For asteroids, the orbital elements are often given as: N,i,w,a,e,M, where N,i,w are valid for a specific epoch (nowadays usually 2000.0). In our simplified computational scheme, the only significant changes with the epoch occurs in N. To convert N_Epoch to the N (today's epoch) we want to use, simply add a correction for precession:

```
  N = N_Epoch + 0.013967 * ( 2000.0 - Epoch ) + 3.82394E-5 * d
```

where Epoch is expressed as a year with fractions, e.g. 1950.0 or 2000.0

Most often M, the mean anomaly, is given for another day than the day we want to compute the asteroid's position for. If the daily motion, n, is given, simply add n * (time difference in days) to M. If n is not given, but the period P (in days) is given, then n = 360.0/P. If P is not given, it can be computed from:

```
  P = 365.2568984 * a**1.5 (days) = 1.00004024 * a**1.5   (years)
```

** is the power-of operator. a**1.5 is the same as sqrt(a*a*a).

When all orbital elements has been computed, proceed as with the other planets ([section 6](#)).

c) Of comets

For comets having elliptical orbits, M is usually not given. Instead T, the time of perihelion, is given. At perihelion M is zero. To compute M for any other moment, first compute the "day number" d of T ([section 3](#)), let's call this dT. Then compute the "day number" d of the moment for which you want to compute a position, let's call this d. Then M, the mean anomaly, is computed like:

```
M = 360.0 * (d-dT)/P     (degrees)
```
where P is given in days, and d-dT of course is the time since last perihelion, also in days.

Also, a, the semi-major axis, is usually not given. Instead q, the perihelion distance, is given. a can easily be computed from q and e:

```
a = q / (1.0 - e)
```
Then proceed as with an asteroid ([section 16](#)).

14. Orbits

a) Parabolic

If the comet has a parabolic orbit, a different method has to be used. Then the orbital period of the comet is infinite, and M (the mean anomaly) is always zero. The eccentricity, e, is always exactly 1. Since the semi-major axis, a, is infinite, we must instead directly use the perihelion distance, q. To compute a parabolic orbit, we proceed like this:

Compute the "day number", d, for T, the moment of perihelion, call this dT. Compute d for the moment we want to compute a position, call it d ([section 3](#)). The constant k is the Gaussian gravitational constant: *k = 0.01720209895 exactly!*

Then compute:

```
H = (d-dT) * (k/sqrt(2)) / q**1.5
```
where q**1.5 is the same as sqrt(q*q*q). Also compute:

```
h = 1.5 * H
g = sqrt( 1.0 + h*h )
s = cbrt( g + h ) - cbrt( g - h )
```
cbrt() is the cube root function: cbrt(x) = x**(1.0/3.0). The formulae has been devised so that both g+h and g-h always are positive. Therefore one can here safely compute cbrt(x) as exp(log(x)/3.0) . In general, cbrt(-x) = -cbrt(x) and of course cbrt(0) = 0.

Instead of trying to compute some eccentric anomaly, we compute the true anomaly and the heliocentric distance directly:

```
v = 2.0 * atan(s)
r = q * ( 1.0 + s*s )
```
When we know the true anomaly and the heliocentric distance, we can proceed by computing the position in space ([section 7](#)).

b) Near-parabolic

The most common case for a newly discovered comet is that the orbit isn't an exact parabola, but very nearly so. It's eccentricity is slightly below, or slightly above, one. The algorithm presented here can be used for eccentricities between about 0.98 and 1.02. If the eccentricity is smaller than 0.98 the elliptic algorithm (Kepler's equation/etc) should be used instead. No known comet has an eccentricity exceeding 1.02.

As for the purely parabolic orbit, we start by computing the time since perihelion in days, d-dT, and the perihelion distance, q. We also need to know the eccentricity, e. The constant k is the Gaussian gravitational constant: *k = 0.01720209895 exactly!*

Then we can proceed as:

```
a = 0.75 * (d-dT) * k * sqrt( (1 + e) / (q*q*q) )
b = sqrt( 1 + a*a )
W = cbrt(b + a) - cbrt(b - a)
f = (1 - e) / (1 + e)

a1 = (2/3) + (2/5) * W*W
a2 = (7/5) + (33/35) * W*W + (37/175) * W**4
a3 = W*W * ( (432/175) + (956/1125) * W*W + (84/1575) * W**4 )

C = W*W / (1 + W*W)
g = f * C*C
w = W * ( 1 + f * C * ( a1 + a2*g + a3*g*g ) )

v = 2 * atan(w)
r = q * ( 1 + w*w ) / ( 1 + w*w * f )
```

This algorithm yields the true anomaly, v, and the heliocentric distance, r, for a nearly-parabolic orbit. Note that this algorithm will fail very far from the perihelion; however the accuracy is sufficient for all comets closer than Pluto.

15. Rise and set times.

(this subject has received a [document of its own](#))

16. Validity of orbital elements.

Due to perturbations from mainly the giant planets, like Jupiter and Saturn, the orbital elements of celestial bodies are constantly changing. The orbital elements for the Sun, Moon and the major planets, as given here, are valid for a long time period. However, orbital elemets given for a comet or an asteroid are valid only for a limited time. How long they are valid is hard to say generally. It depends on many factors, such as the accuracy you need, and the magnitude of the perturbations the comet or asteroid is subjected to from, say, Jupiter. A comet might travel in roughly the same orbit several orbital periods, experiencing only slight perturbations, but suddenly it might pass very close to Jupiter and get its orbit changed drastically. To compute this in a reliable way is quite complicated and completely out of scope for this description. As a rule of thumb, though, one can assume that an asteriod, if one uses the orbital elements for a specific epoch, one or a few revolutions away from that moment will have an error in its computed position of at least one or a few arc minutes, and possibly more. The errors will accumulate with time.

22. Links to other sites.

Astronomical Calculations by Keith Burnett: http://www.xylem.f2s.com/kepler/
Free BASIC programs can be found at ftp://seds.lpl.arizona.edu/pub/software/pc/general/ in: *ast.exe* (needs GWBASIC!) and *duff2ed.exe* (Pete Duffett-Smiths programs)
Books from **Willmann-Bell** about Math and Celestial Mechanics: http://www.willbell.com/math/index.htm
John Walker's freeware program **Home Planet + other stuff**: http://www.fourmilab.ch/
Elwood Downey's **Xephem** and **Ephem** programs, with C source code: http://www.clearskyinstitute.com/xephem/.
Ephem can also be found at ftp://seds.lpl.arizona.edu/pub/software/pc/general/ as *ephem421.zip*
Steven Moshier: Astronomy and numerical software source codes: http://www.moshier.net/
Dan Bruton's astronomical software links: http://www.physics.sfasu.edu/astro/software.html
Mel Bartel's software (much ATM stuff): http://www.efn.org/~mbartels/tm/software.html
Almanac data from **USNO**: http://aa.usno.navy.mil/data/
Asteroid orbital elements from **Lowell Observatory**: http://asteroid.lowell.edu/
SAC downloads: http://www.saguaroastro.org/content/downloads.htm
Earth Satellite software from **AMSAT**: http://www.amsat.org/amsat/ftpsoft.html
IMCCE (formerly Bureau des Longitudes): http://www.imcce.fr/
VSOP87: ftp://ftp.imcce.fr/pub/ephem/planets/vsop87/

DE403/404/410/414 at JPL: ftp://ssd.jpl.nasa.gov/pub/eph/export/
SSEphem at NRAO: ftp://ftp.cv.nrao.edu/NRAO-staff/rfisher/SSEphem/
Some catalogues at **CDS, Strasbourg, France** - high accuracy orbital theories:
Overview: http://cdsweb.u-strasbg.fr/cgi-bin/qcat?VI/
Precession & mean orbital elements: http://cdsweb.u-strasbg.fr/cgi-bin/qcat?VI/66/
ELP2000-82 (orbital theory of Moon): http://cdsweb.u-strasbg.fr/cgi-bin/qcat?VI/79/
VSOP87 (orbital theories of planets): http://cdsweb.u-strasbg.fr/cgi-bin/qcat?VI/81/
Astronomical Data Center http://adc.gsfc.nasa.gov/adc.html has lots of catalogs. Some of them are:
Asteroid orbital elements 1998: ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1245/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1245/
JPL ephemeris DE118/LE62: ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1093A/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1093A/
JPL ephemeris DE200/LE200: ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1094A/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1094A/
USNO ZZCAT: ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1157/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1157/
XZ catalog of zodiacal stars: ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1201/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1201/
Tycho Reference Catalog: ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1250/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1250/
Tycho 2 catalog: ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1259/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1259/
USNO A2.0 catalog (very large): ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1252/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1252/
HST Guide Star catalog 1.2: ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1254/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1254/
HST Guide Star catalog 1.3: ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1255/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1255/
Tycho 2 catalog: ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1259/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1259/
AC 2000.2 catalog: ftp://adc.gsfc.nasa.gov/pub/adc/archives/catalogs/1/1275/ ftp://adc.astro.umd.edu/pub/adc/archives/catalogs/1/1275/
Other similar services are available at: CDS (France) - ADAC (Japan) - CAD (Russia)

The original ZC (Zodiacal Catalogue):
http://stjarnhimlen.se/zc/
http://web.archive.org/web/20030604102426/sorry.vse.cz/~ludek/zakryty/pub.phtml#zc
http://web.archive.org/web/20030728014108/http://sorry.vse.cz/~ludek/zakryty/pub/