Converting Telescope Coordinate Systems.

Halverson 12/6/2018

How to convert from the Alt-Az (Altitude-Azimuth) coordinates our telescope uses to Equatorial coordinates needed to locate and track objects in the sky.

General Strategy

I am calling the scope's Alt-Az system the "A" system. and the Equatorial coordinates the "E" system.

General strategy to convert from A to E: (scope coordinates to sky coordinates)

1. Convert the pointing angles of the scope to x,y,z coordinates of a point a distance away, in the direction the scope is pointing. The distance R doesn't actually matter. It could be 10 meters away, or 1000 light years away.

   In short, $(\theta,\varphi,R)_A \rightarrow (x,y,z)_A$

2. Convert the $(x,y,z)_A$ coordinates to $(x,y,z)_E$ coordinates, in the rotated coordinate system where the z axis points to the north celestial pole. (The z axis points to Polaris, the "North Star".)

   In short, $(x,y,z)_A \rightarrow (x,y,z)_E$

3. Convert the equatorial x,y,z coordinates to equatorial angles.

   In short, $(x,y,z)_E \rightarrow (\theta,\varphi,R)_E$

Again, R is just a placeholder kept to make the math easier to understand. In the computer code we will probably set R to 1, or eliminate it completely.

We will also need to convert E coordinates to A. With a few minor differences, the same process will apply.

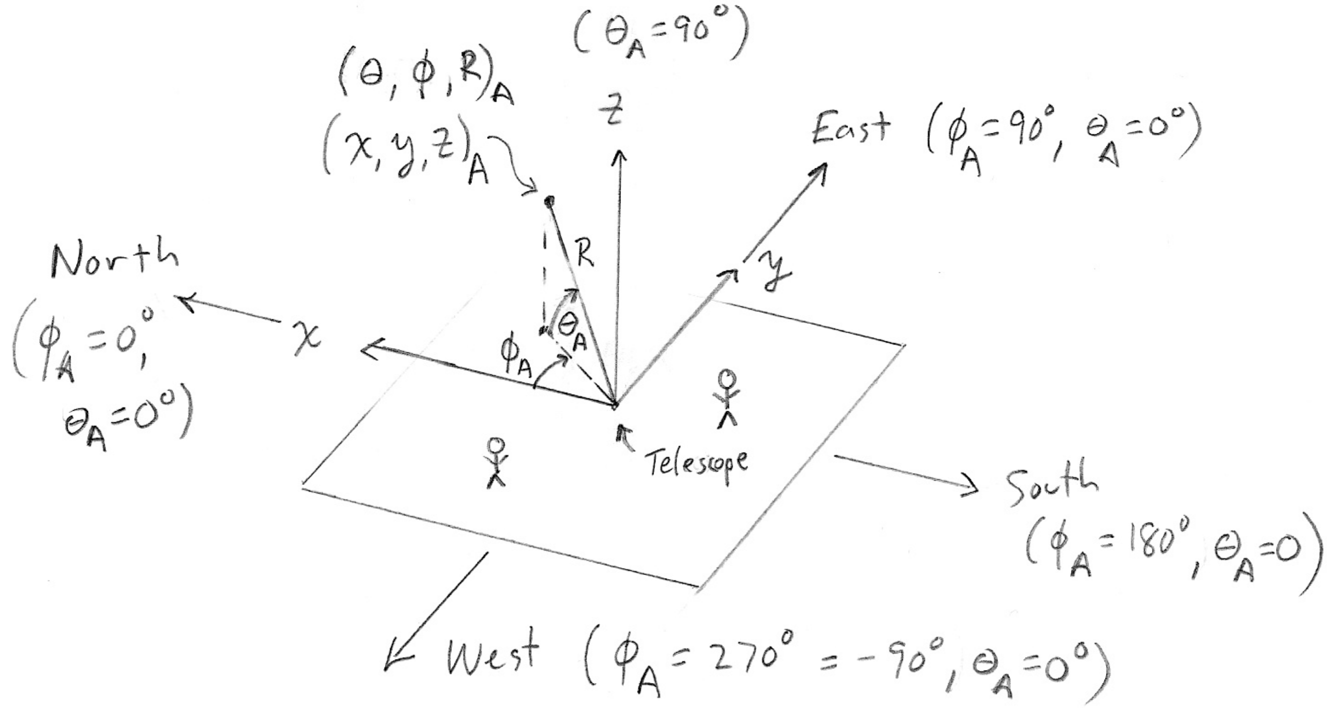
HELP!!!

I NEED A STUDENT TO WORK THE MATH FOR E TO A COORDINATES CONVERSION.

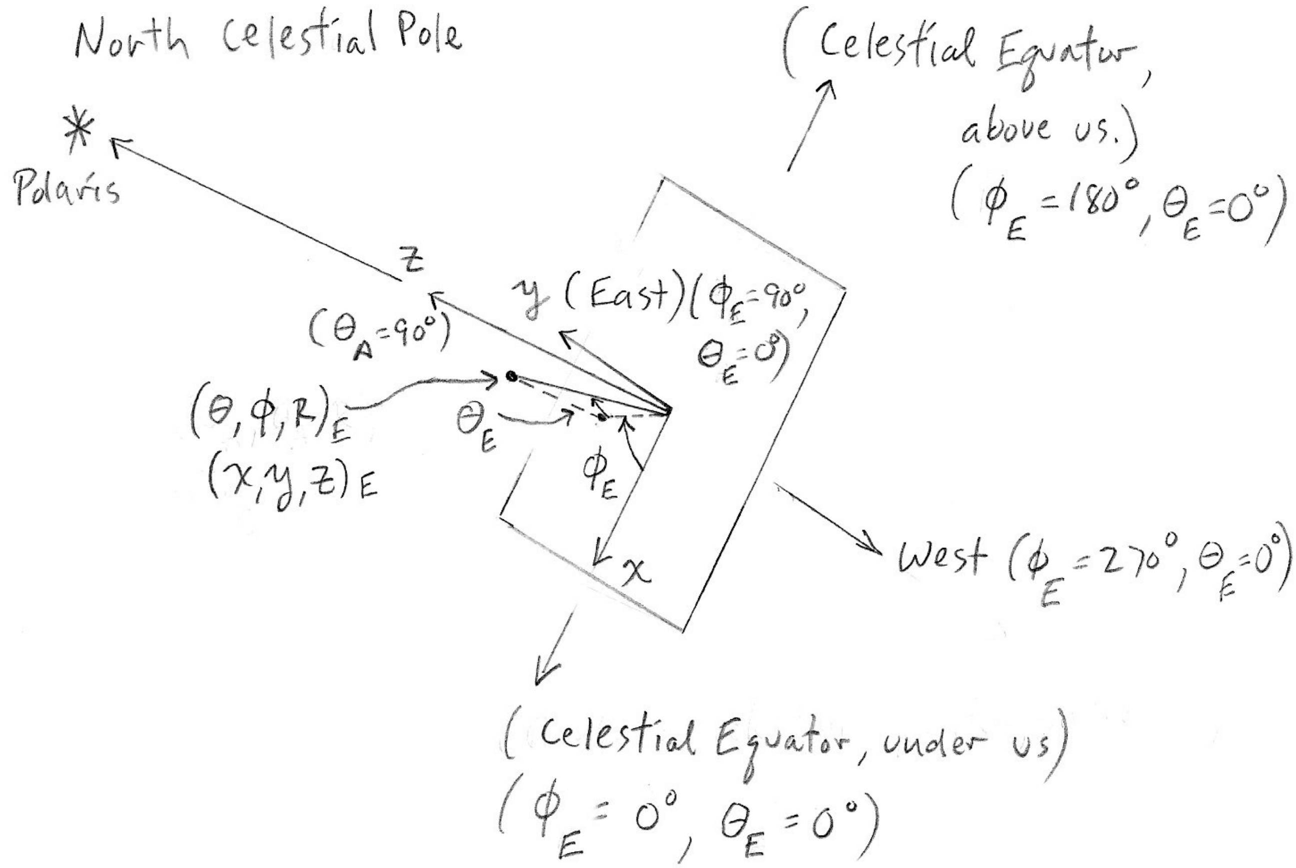
Update 1/22/2019: I have done the math for E to A conversion and written the Python code for the conversions math to be built in to the telescope control software. I have added the math and the Python code to this document. I also corrected a couple mistakes. Scroll down and take a look.

Coordinate systems

"A" = ALT-AZ COORDINATE SYSTEM:



"E" = EQUATORIAL COORDINATE SYSTEM



Converting from A to E - details

Step 1: ***(θ,φ,R)*** $_A$ **→** ***(x,y,z)*** $_A$

$$x_A = R\cos(\theta_A)\cos(\varphi_A)$$
$$y_A = R\cos(\theta_A)\sin(\varphi_A)$$
$$z_A = R\sin(\theta_A)$$

Step 2: ***(x,y,z)*** $_A$ **→** ***(x,y,z)*** $_E$

This part is complicated and I will break it into smaller sub-steps.

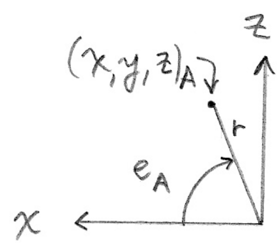
Sub-step 2.1: $y_A \rightarrow y_E$

This is easy! The y coordinates are the same in the two systems.

$$y_E = y_A$$

Sub-step 2.2: $(x,z)_A \rightarrow (e_A, r)$

We now look at the coordinate systems a bit differently. We pretend we are West of the telescope and look at it sitting in the East:



This is a good way to look at it because from this vantage point the transition from "A" (Alt-Az) to "E" (Equatorial) coordinates looks like a simple rotation.

In the diagram, we have defined a new angle called e_A related to the altitude θ_A but different depending on the current azimuth angle φ_A of the telescope. To calculate e_A first notice that

$$z_A = r\sin(e_A) \text{ and } x_A = r\cos(e_A).$$

Now put these two thing together like this:

$$\frac{z_A}{x_A} = \frac{r\sin(e_A)}{r\cos(e_A)} = \tan(e_A)$$

We now have an easy way to find e_A:

$$e_A = \arctan\left(\frac{z_A}{x_A}\right)$$

We also define a new radial distance r related to the original R, but foreshortened because of our point-of-view standing West of the telescope, looking East. We find it using the pythagorean theorem:

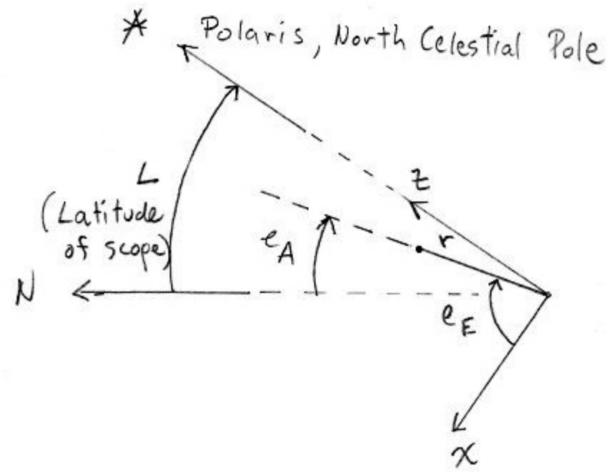
$$r = \sqrt{x_A^2 + z_A^2}$$

(Another way to find r is $r = R\cos(\varphi_A)$ however if φ_A. is greater than 90 degrees, the cosine will be negative, so we actually would need to use the absolute value $r = R\left|\cos(\varphi_A)\right|$)

Sub-step 2.3: $e_A \rightarrow e_E$

The E system is rotated relative to the A system by an amount that depends on our latitude L. Here, in Los Angeles, L=34.05 degrees which means we are that many degrees North of the equator, and that Polaris, the North Star, is that many degrees above the horizon, to the North.

The E system's x and z axes are rotated L degrees counter-clockwise from the A system's x and z axis. In the diagram below, you see how it is similar to the previous digram, but rotated ccw. (The y axis is pointed away from us, so we don't se it.)



(I corrected a mistake in this diagram and in the following step. The angles didn't add up right.)

For the conversion $e_A \rightarrow e_E$ first note that the angle from the x axis to the z axis is 90 degrees so

$$90^0 = e_E - e_A + L$$

Solve for e_E:

$$e_E = 90^0 + e_A - L$$

No conversion is needed for r, since the distance from the origin is unaffected by a rotation.

Sub-step 2.4: $(r, e_E) \rightarrow (x,z)_E$

We now want to get x and z in the E system. (We already know y; it didn't change.) It isn't too hard:

$$x_E = r\cos(e_E)$$
$$z_E = r\sin(e_E)$$

(I corrected a mistake here. I had the cos and sin reversed.)

We are now done with step 2. We know $(x,y,z)_E$.

Step 3: ***(x,y,z)***$_E$ ***→ (θ,φ,R)***$_E$

Since R doesn't change in a rotation, we already know this part of the answer. It is the same R we started with. (And it doesn't really matter what R is.)

We also know this:

$$x_E = R\cos(\theta_E)\cos(\varphi_E)$$
$$y_E = R\cos(\theta_E)\sin(\varphi_E)$$
$$z_E = R\sin(\theta_E)$$

If we can work these equations backward, then we're done.

First let's do φ_E:

$$\frac{y_E}{x_E} = \frac{R\cos(\theta_E)\sin(\varphi_E)}{R\cos(\theta_E)\cos(\varphi_E)} = \frac{\sin(\varphi_E)}{\cos(\varphi_E)} = \tan(\varphi_E)$$

therefore

$$\boxed{\varphi_E = \arctan\left(\frac{y_E}{x_E}\right)} \quad \text{<---THE ANSWER, part 1}$$

To get θ_E we have:

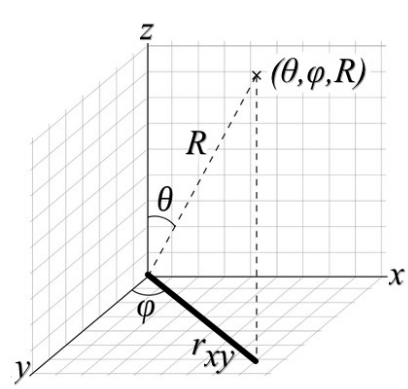
$$z_E = R\sin(\theta_E)$$
$$\sin(\theta_E) = \frac{z_E}{R}$$
$$\theta_E = \arcsin\left(\frac{z_E}{R}\right)$$

(If we happen to make R=1, then θ_E=arcsin(z_E), but I haven't decided this yet.)

In a computer the formula

$$\theta_E = \arcsin\left(\frac{z_E}{R}\right)$$

gives low accuracy answers near 90 degrees (why is that?) but there is a better way that is always accurate. To understand it, think of the distance from the z axis that the (x,y,z) point is. Call this distance "r_{xy}" because it is the "projection of R on to the xy plane." Here is a digram with r_{xy} in bold (from Wikipedia, with modifications[1]):



$$r_{xy}^2 = x_E^2 + y_E^2$$
$$r_{xy} = \sqrt{x_E^2 + y_E^2}$$

Now we have the better way to get θ_E :

$$\tan(\theta_E) = \frac{z_E}{r_{xy}} = \frac{z_E}{\sqrt{x_E^2 + y_E^2}}$$

so

$$\boxed{\theta_E = \arctan\left(\frac{z_E}{\sqrt{x_E^2 + y_E^2}}\right)} \quad \text{<---THE ANSWER, part 2}$$

(This accuracy problem is very common in computer simulations, game design and target tracking so most computer languages, including python, have a special arctan function, called “ATAN2” for just this purpose.)

Conclusion:

We now have the math needed to convert the telescope’s Alt-Az pointing information (which comes from the stepping motor control) to the sky’s natural Equatorial coordinates.

Footnotes:

1) I have changed the xyz system from “right handed’ to “left-handed” to agree with this explanation. It came from https://en.wikipedia.org/wiki/Spherical_coordinate_system

Reverse Conversion: Equatorial to Alt-Az: (Steps flagged in red became Python code)

1) $(\theta_E, \phi_E) \rightarrow (x, y, z)_E$

$$\left.\begin{aligned} x_E &= R\cos(\theta_E)\cos(\phi_E) \\ y_E &= R\cos(\theta_E)\sin(\phi_E) \\ z &= R\sin(\theta_A) \end{aligned}\right\}$$

2) $(x, y, z)_E \rightarrow (x, y, z)_A$

2.1) $y_A = y_E \leftarrow$

2.2) Looking from the West, to the East

$(x, z)_E \rightarrow (e_E, r)$

$$e_E = \arctan\left(\frac{z_E}{x_E}\right) \leftarrow$$

$$r = \sqrt{x_E^2 + z_E^2} \leftarrow$$

2.3) $90^\circ = e_E - e_A + L$, solve for e_A

$$e_A = e_E + L - 90^\circ \leftarrow$$

2.4) $(r, e_A) \rightarrow (x, z)_A$

$$x_A = r\cos(e_A)$$

$$z_A = r\sin(e_A) \qquad (y_A = y_E)$$

3) $(x, y, z)_A \rightarrow (\theta, \phi, R)_A$

$$\phi_A = \arctan\left(\frac{y_A}{x_A}\right)$$

$$\theta_A = \arctan\left(\frac{z_A}{\sqrt{x_A^2 + y_A^2}}\right)$$

Python Code:

```
from collections import namedtuple   #This is a add-on feature to allow multi-part
                                     #variables such as  (x,y,z) and (theta,phi)

from math import *                   #Import all of the math library, ie sin, cos, tan...

L=radians(34.0636051)    #This is the latitude of Stern MASS from Google Maps that I
added to Stellarium
#L=radians(34.0095291)    #This is the latitude of Alhambra used in Stellarium

def A_to_E(thetaA,phiA):  #thetaA is the Altitude angle, phiA is the Azimuth angle, in
degrees.
  # thetaA is zero at the horizon, increasing to 90 degrees at the zenith (straight
overhead)
  # phiA is zero due north increases to 90 degrees due East, 180 for South, 270 for West.
  print
  print "thetaA=",thetaA," phiA=",phiA
  R=1000.0
  #Convert the angles to radians
  thetaA=radians(thetaA)
  phiA=radians(phiA)
  #---------------------------STEP 1     theta, phi, R --> x,y,z
  xA=R*cos(thetaA)*cos(phiA)
  yA=R*cos(thetaA)*sin(phiA)
  zA=R*sin(thetaA)
  print "xA=",xA," yA=",yA," zA=",zA
  #---------------------------STEP 2    (x,y,z)A --> (x,y,z)E
  #---------------------------substep 2.1  yA --> yE
  yE=yA
  #---------------------------substep 2.2  (x,z)A --> (eA,r)
  eA=atan2(zA,xA)
  r=sqrt(xA*xA+zA*zA)
  print "r=",r
  #---------------------------substep 2.3  eA --> eE
  eE=radians(90.0)+eA-L
  print "eA=",degrees(eA)," eE=",degrees(eE)
  #---------------------------substep 2.4  eE --> (x,z)E
  xE=r*cos(eE)
  zE=r*sin(eE)
  print "xE=",xE," yE=",yE," zE=",zE
  #---------------------------STEP 3     (x,y,z)E --> (theta, phi,R)E
  phiE=atan2(yE,xE)
  thetaE=atan2(zE,sqrt(xE*xE+yE*yE))
  #----------------------------We are done
  #convert the angles from radians back to degrees
  thetaE=degrees(thetaE)
  phiE=degrees(phiE)
  if phiE < 0.0:
    phiE += 360.0    #We don't want negative phi values.
  reverse_phiE=360.0-phiE
  #if reverse_phiE > 360.0:
  #  reverse_phiE -= 360.0
  print "thetaE=",thetaE," phiE=",phiE,"   360-phiE=",reverse_phiE
  angles = namedtuple('angles','theta phi')
  return angles(theta=thetaE,phi=phiE)

def E_to_A(thetaE,phiE):
```

```
  # thetaE is the Declination angle.  It is zero at the celestial equator, increasing to
90 at the celestial North Pole.
  # phiE is the angle around the equatorial plane
  # phiE is zero at the point below the horizon under the north celestial pole.  It
increase
  # to 90 degree at due East, to 180 overhead/south and 270, due west.
  print
  print "thetaE=",thetaE," phiA=",phiE
  R=1000.0
  #Convert the angles to radians
  thetaE=radians(thetaE)
  phiE=radians(phiE)
  #--------------------------STEP 1     theta, phi, R --> x,y,z
  xE=R*cos(thetaE)*cos(phiE)
  yE=R*cos(thetaE)*sin(phiE)
  zE=R*sin(thetaE)
  print "xE=",xE," yE=",yE," zA=",zE
  #--------------------------STEP 2    (x,y,z)E --> (x,y,z)A
  #--------------------------substep 2.1  yE --> yA
  yA=yE
  #--------------------------substep 2.2  (x,z)E --> (eE,r)
  eE=atan2(zE,xE)
  r=sqrt(xE*xE+zE*zE)
  print "r=",r
  #--------------------------substep 2.3  eE --> eA
  eA=eE+L-radians(90.0)
  print "eE=",degrees(eE)," eA=",degrees(eA)
  #--------------------------substep 2.4  r,eA --> (x,z)A
  xA=r*cos(eA)
  zA=r*sin(eA)
  print "xA=",xA," yA=",yA," zA=",zA
  #--------------------------STEP 3     (x,y,z)A --> (theta, phi,R)A
  phiA=atan2(yA,xA)
  thetaA=atan2(zA,sqrt(xA*xA+yA*yA))
  #---------------------------We are done
  #convert the angles from radians back to degrees
  thetaA=degrees(thetaA)
  phiA=degrees(phiA)
  if phiA < 0.:
    phiA += 360.0     #We don't want negative phi values.
  #reverse_phiA=360.0-phiA
  #if reverse_phiA > 360.0:
  #  reverse_phiA -= 360.0
  print "thetaA=",thetaA," phiA=",phiA
  #,"   360-phiE=",reverse_phiE
  angles = namedtuple('angles','theta phi')
  return angles(theta=thetaA,phi=phiA)

# Test the conversion function
#newthetaE,newphiE=A_to_E(10.,20.)
#print "newthetaE=",newthetaE," newphiE=",newphiE

# Test the reverse conversion
#newthetaA,newphiA=E_to_A(newthetaE,newphiE)
#print "newthetaA=",newthetaA," newphiA=",newphiA
```