Calculating Celestial Coordinates
from Orbital Parameters
Using Javascript

Robin Newhouse

Physics 210, Department of Physics and Astronomy,
University of British Columbia, Vancouver, BC, Canada V6T 1Z1
(Dated: December 2, 2010)

Abstract

Using the simplicity and surety of celestial mechanics, one is able to calculate the position of a celestial body in the night sky with relative ease. However, multiple coordinate systems are employed, and to track a moving body, many calculations must be repeated many times. Though this may be tedious for a human mind, properly instructed computers will happily make these calculations thousands of times per second. This web tool is designed to take a very limited set of parameters and calculate many time-dependant variables. Using these variables, a body’s Right Ascension and Declination are returned to the user. These variables are also put into an easily graphable data file to visualize one period of orbit.

*robinnewhouse@gmail.com | http://www.physics.ubc.ca/~newhouse/p210/
I. METHOD

In calculating motion along transformed coordinate systems, I assumed there would be no major complications as I programmed. Though this was the case for certain steps, there were things about this function that proved to be less simple than expected. Also, being able to visualize the reality that many of the equations described took a lot more thought than I had expected. The following sections will describe each step of the program and why it is necessary.

A. Main Goals

My main goal was to output a celestial body’s right ascension and declination as seen from the earth, using publicly available orbital elements. Though this was possible to accomplish in a number of programming languages, I wanted this to be publicly available and simple to use, so I decided to make a website to host the program. I wrote the program in Javascript partly for the convenience of Java’s ability to operate on HTML pages, and partly for the learning experience, as I had never used Java before.

B. Parsing

The first step was to determine the orbital elements and what they mean. The names and brief descriptions of these elements are available on the website of the Minor Planet Center. [http://www.minorplanetcenter.org/iau/MPEph/NewObj Ephems.html](http://www.minorplanetcenter.org/iau/MPEph/NewObj Ephems.html) The Minor Planet Center is also the website that has the one-line format orbital elements for all currently documented comets and minor planets in the solar system. Getting the necessary numbers was only a matter of pasting the one-line orbital elements for a body, reading each part one
at a time, and assigning the number to a corresponding variable. the information available from the elements is as follows.

Example: 0081P 2010 02 22.6884 1.598039 0.537292 41.7877 136.0965 3.2374 20100723 7.0 6.0 81P/Wild MPC 67148
Comet Number: Characters 0-4
Orbit Type: Characters 4-5
Provisional Designation: Characters 6-12
Time of perihelion passage: Characters 14-29
Perihelion Distance: Characters 31-39
Eccentricity: Characters 41-49
Argument of Perihelion 51-59
Longitude of the Ascending Node: Characters 61-69
Inclination: Characters 71-79
Object Name: Characters 102-116

FIG. 1. The orbit of a body around the sun with orbital elements

\[ r = \text{distance between planet X and the Sun} \]
\[ P = \text{perihelion (point where X is closest to Sun)} \]
\[ \varpi = \text{first point of Aries} \]
\[ i = \text{angle between plane of Sun’s equator and planet X’s orbit} \]
\[ \omega = \text{longitude of perihelion} \]
\[ \Omega = \text{longitude of ascending node} \]
C. Finding Mean Parameters

From these orbital elements, I needed to calculate the time dependent quantities. What this step aims to do is find the true anomaly, \( \phi \), which is the angle between the body's current position and the body's perihelion on the plane of its own orbit. In this situation, anomaly is synonymous with angle. The term originates from the fact that the observed locations of a planet often showed small deviations from the predicted data\(^2\).

To do this, the mean anomaly must first be calculated. The mean anomaly is the true anomaly of the mean planet, a theoretical planet with the same semi-major axis and period as the real planet, but with a perfectly circular orbit. The mean anomaly by definition increases linearly with time.

From this mean anomaly, the eccentric anomaly can be calculated, then the true anomaly from that. However, the equation relating the mean and eccentric anomalies is given by \( M = E - e \sin E \), which is a “trancendental equation” and has no general solution. It must be repeatedly calculated and guessed, down to a certain error. Doing this by hand would be tedious, but in javascript takes only a fairly simple loop command.

Once the Eccentric anomaly is solved, a simple equation gives the true anomaly, \( \phi \), and the radius, \( r \).

D. Acquiring Coordinates

Now, the three time independent terms \( i, \Omega \) and \( \omega \) and the two time dependent terms \( \phi \) and \( r \), can be used to calculate the heliocentric ecliptic coordinates of the body. These are the rectangular \( x, y, \) and \( z \) coordinates of the body with the \( x \) axis pointing through the point of Aries (the earth’s ascending node) and the \( xy \) plane parallel to the orbital plane of the earth. These coordinates follow an elliptical path around the origin with an increase in time and represent the path of the body around the sun.
E. Translating to Geocentric Ecliptical Coordinates

The path around the sun has now been calculated, but I’m interested in where we can see the body from the Earth. To do this, the exact same process and calculations must be made for the heliocentric ecliptic coordinates of the Earth. I just used an exact copy of the function used for the body’s coordinates, but used predefined values for the Earth.

Now to obtain *geocentric* coordinates we need simply to translate the origin of the coordinate system by the form

\[
\text{geocentric coordinates} = \text{heliocentric body coordinates} - \text{heliocentric earth coordinates}
\]

F. Rotating to Equatorial Coordinates

Now we have values for where the body is in relation to the earth as they both revolve around the sun. But these coordinates are in relation to the orbital plane of the earth, which is inclined by 23.439° to the equator of the earth to which we usually relate our astronomical measurements. To correct for this, the coordinate system is rotated by 23.439°.

G. Converting to Right Ascension and Declination Coordinates

Now the coordinates only go through three simple equations to give right ascension, declination and radial distance from earth. Right ascension is usually given in hours, minutes, seconds format while declination is given in degrees.

II. MATHEMATICS

Though there are many steps to solving the position of a planet in the night sky each individual step is solved, for the most part, by a relatively simple equation. Following are the mathematics the program uses to calculate the position of a celestial body. I’ve also included the actual script that correlates to the mathematics described.
A. Values into $a$

Solving for the orbits semi-major axis, $a$, requires knowing at least two orbital elements, in this case $D$, the distance of perihelion, and $e$, the eccentricity of the orbit.

\[ a = \frac{D}{1 - e} \]  

(1)

The script:

```javascript
var D=distp ;
var e=ecc ;
var a=D/(1-e) ;
document.forms["calc"].a.value = a;
```

B. Values into $P$

One of Kepler’s laws of planetary motion state a very convenient fact about planets: the square of the period is equal to the cube of the semi-major axis (when measured in years and Astronomical Units). Using what was calculated for $a$ in eq. 1, this relation can find the period, $P$.

\[ P^2 = a^3 \]  

(2)

\[ P = \sqrt{a^3} \]  

(3)

The script:

```javascript
var PP=a*a*a ;
var P=Math.sqrt(PP);
document.forms["calc"].P.value = P;
```

C. Solving for $M$

An unexpected issue was encountered while solving for $M$. Though the mean anomaly is simply the planets velocity multiplied by the time elapsed since it last passed perihelion,

\[ M = 2\pi(t - T)/P \]  

(4)
the time since last perihelion is not given. Subtracting the time of last recorded perihelion from the time of observation and taking the remainder of the difference divided by the bodys period from \[3\] solves this issue. That is, unless the time of observation is less than the time of last recorded perihelion. This is easily resolvable however by subtracting multiple periods from the time of perihelion until it is less than the time of observation. This is a reasonable correction because the planet will be in the same point of its orbit no matter how many periods are added or subtracted.

The script:

```javascript
var t=time;
var Q=timep;
var L=Q; //last perihelion
while (L >= t)
{
    L = L - P;
}

var M=2*pi*(Math.abs(t-L)%P)/P;
document.forms["calc"].M.value = M*deg;
document.forms["calc"].L.value = t-(Math.abs(t-L)%P);
```

### D. Solving for \( E \)

This part is probably the most complicated calculation of the whole script. The eccentric anomaly is described by

\[
M = E - e \sin E
\] (5)

which is a transcendental equation, meaning it has no general solution. It can be approximated to a high degree of accuracy though by performing multiple iterations, essentially guessing and checking. A script I found on [http://www.jgiesen.de/kepler/kepler.html](http://www.jgiesen.de/kepler/kepler.html) does exactly the iteration I was looking for, so I borrowed it to put in my script and made a few modifications to attain a more accurate approximation. The script defines a function

\[
F(E) = E - e \sin E(t) - M(t)
\] (6)

and adjusts the function \( E(t) \) by

\[
E(t) = E(t) - F(E)/(1.0 - e \cos(E(t)))
\] (7)
until \( F(E) \) is satisfactorily small

The script:

```javascript
//solve for E (eccentric anomaly) using multiple iterations
// arguments:
// ec=eccentricity, m=mean anomaly,
// dp=number of decimal places
var K=\pi/180.0;
var maxIter=200, i=0;
var delta=Math.pow(10,-30);
var E, F;
m=M;
ec=e;
// m=2.0*\pi*(m-Math.floor(m));
if (ec<0.8) E=m; else E=180.0;//>
F = E - ec*Math.sin(m) - m;
while ((Math.abs(F)>delta) && (i<maxIter)) {
  E = E - F/(1.0-ec*Math.cos(E));
  F = E - ec*Math.sin(E) - m;
  i = i + 1;
}
//done
```

E. Solving for \( \phi \)

\( \phi \) is the true anomaly and can be solved from \( E \) and \( e \)

\[
\phi = 2\arctan\left(\frac{1-e}{1+e}\tan\frac{E}{2}\right)
\] (8)

The script:

```javascript
//using E and e solve for phi
var phi=mod2pi(2*Math.atan(Math.sqrt((1 + e)/(1 - e))*Math.tan(0.5*E)));
document.forms["calc"].phi.value = phi*deg;
```
F. Solving for $r$

Another simple calculation is solving for the radial distance from the Sun to the body.

$$r = a \frac{1 - e^2}{1 + e \cos \phi} \quad (9)$$

The script:

```javascript
//using phi solve for r radial distance
var r=a*(1-e*e)/(1+e*Math.cos(phi));
document.forms["calc"].r.value = r;
```

G. Calculating heliocentric ecliptic coordinates

Now that the proper time-dependant and time-independent terms are known, the heliocentric ecliptic coordinates, $x$, $y$ and $z$, of the body can be calculated.

$$x = r \cos(\Omega) \cos(\omega + \phi) - r \sin(\Omega) \sin(\omega + \phi) \cos(i) \quad (10)$$

$$y = r \sin(\Omega) \cos(\omega + \phi) + r \cos(\Omega) \sin(\omega + \phi) \cos(i) \quad (11)$$

$$z = r \sin(\omega + \phi) \sin(i) \quad (12)$$

The script:

```javascript
//calculate heliocentric ecliptic coordinates
var xec = r*(Math.cos(omega)*Math.cos(w+phi)
             -Math.sin(omega)*Math.sin(w+phi)*Math.cos(incl));
document.forms["calc"].xec.value = xec; //uses inc (rad) not incl (degrees)
var yec = r*(Math.sin(omega)*Math.cos(w+phi)
             +Math.cos(omega)*Math.sin(w+phi)*Math.cos(incl));
document.forms["calc"].yec.value = yec;
var zec = r*Math.sin(w+phi)*Math.sin(incl);
document.forms["calc"].zec.value = zec;
```
H. Calculating the Same for Earth

To use geocentric coordinates the exact same calculations must be done for the position of the Earth in relation to the Sun. This is just a matter of copying the entire script up to this point but using predefined values for the Earth. In specific they are as follows:

- time of perihelion = 2010.00821355 years
- distance of perihelion = 0.98328978 AU
- eccentricity = 0.01671022
- argument of perihelion = 102.93768193 degrees
- longitude of ascending node = 0 degrees
- inclination = 0 degrees

I. Translating Heliocentric to Geocentric coordinates

Probably the easiest part of this mess is transforming the coordinates. The calculations are as follows:

\[
\begin{align*}
x_{\text{geo}} &= x_{\text{body}} - x_{\text{earth}} \quad (13) \\
y_{\text{geo}} &= y_{\text{body}} - y_{\text{earth}} \quad (14) \\
z_{\text{geo}} &= z_{\text{body}} - z_{\text{earth}} \quad (15)
\end{align*}
\]

The script:

```javascript
var gxec=xec*1-exec*1;
var gyec=yec*1-eyec*1;
var gzec=zec*1-ezec*1;
```

J. Translating Geo. Ecliptic Coordinates to Geo. Equatorial coordinates

The following equations transform the new geocentric ecliptic coordinates to geocentric equatorial coordinates by tilting the coordinates system by the tilt of the earth, which is

\[
tilt = 23.439281^\circ
\]
The script:

```javascript
var tilt = 23.439281 / deg;
var gxeq = gxec * 1;
document.forms['calc'].xeq.value = gxeq;
var gyeq = gyec * Math.cos(tilt) - gzec * Math.sin(tilt);
document.forms['calc'].yeq.value = gyeq;
var gzeq = gyec * Math.sin(tilt) + gzec * Math.cos(tilt);
document.forms['calc'].zeq.value = gzeq;
```

K. Transforming to RA, DEC, & Radius

The final step is to transform these equatorial coordinates into the familiar right ascension (RA) and declination (Dec) coordinates used in astronomy. Since all the “ingredients” are already known, I decided to add in the radial distance from the earth to the body.

\[
RA = \arctan\left(\frac{y_{eq}}{x_{eq}}\right) \quad (19)
\]
\[
Dec = \arctan\left(\frac{z_{eq}}{x_{eq}^2 + y_{eq}^2}\right) \quad (20)
\]
\[
Radial = \sqrt{x_{eq}^2 + y_{eq}^2 + z_{eq}^2} \quad (21)
\]

This gives RA and Dec in radians (later converted to degrees) but traditional measurements for RA are in ArcHours and ArcMinutes, so I added an extra piece of script to adjust the output values.

The script:

```javascript
RArad = mod2pi(Math.atan2(gyeq, gxeq)); // convert into RA DEC coordinates;
DECrad = Math.atan(gzeq / (Math.sqrt(gxeq * gxeq + gyeq * gyeq)));
Raddist = Math.sqrt(gxeq * gxeq + gyeq * gyeq + gzeq * gzeq);
```
document.forms["calc"].RAdeg.value = RArad*deg

var raH = RArad*deg/15; // degrees to hours
var HH = Math.floor(raH); // setHours
var MM = 60*(raH - HH); // set Minutes
document.forms["calc"].RA.value = HH+":"+MM;
document.forms["calc"].DEC.value = DECrad*deg;
document.forms["calc"].Raddist.value = Raddist;

III. WEBSITE FEATURES

A. Dropdown

With the position-calculating function working nicely, I focused a bit on aesthetics and usability. I learned how to make a dropdown menu for many known comets. Selecting a comet from the list populates the orbital elements form and parses the element information into the corresponding forms.

B. Realtime

Another nice feature is to have the website recalculate the information in real-time. Though this looks like one of the most impressive features of the website, it really only took an extra few lines of code. Javascript has a handy function called `getTime()` that outputs the number of elapsed milliseconds since 1970. The substance of the real-time script is in:

document.forms["calc"].currentTime.value = currentTime.getTime()/31558464000+1970;

which is called upon each time a coordinate is calculated.

IV. ADDITIONS

Seeing that the original goal was finished I realized that to get there I had created a versatile orbit simulator. Though not entirely necessary, I thought it would be nice to see what the orbits being calculated actually looked like. At one point in the program I had
x, y, and z coordinates for the orbiting body and the earth that followed an elliptical orbit as real time increased. I simply had to add a function that calculated a large number of coordinates for one period of orbit and output them in a data file readable by 3D plotting software. The “output to DAT button does this nicely. Here are some orbits plotted in Gnuplot. The Gnuplot command used was:

\texttt{splot 'comet.txt' u 1:2:3, 'comet.txt' u 4:5:6, 'comet.txt' u 6:6:6}

\textbf{FIG. 2.} The orbits of many comets and Earth
A. Future Additions

This is a working project and with a few minor additions and corrections could have many additional features. For instance, it’s only a matter of adding a few more lines to the dropdown menu for the option of tracking planets, asteroid belt bodies, or even hypothetical bodies with unusual orbits. Other features could include a button that brings the user to a separate site that shows where in the sky the RA and Dec describe. A KML object could even be created for google sky so the user could track whatever real or theoretical object
they input.

V. REFERENCES


