Sunrise . . . Sunset

By Dr. T.S. Kelso

March/April 1997

Up until now, our investigations in this column have centered on making predictions about the coordinates—for example, the azimuth and elevation—of a satellite at a specific time of interest. While the process of making these calculations accurately and efficiently is not a trivial matter, it only addresses one facet of orbital analysis.

One of the biggest questions typically faced by an orbital analyst is the question of when (or even if) some particular event will happen. There are many such instances which might be of interest: the rise or set of a satellite, the time of an equator crossing, the time of culmination (the highest elevation during a satellite pass), the entry and exit times for an eclipse, or the time of closest approach to another satellite.

As such, two of the most important tools available to an orbital analyst are tools for efficiently determining the roots and extrema (the minima and maxima) of equations. Finding roots corresponds to finding crossing phenomena, such as rise or set times, while finding extrema corresponds to finding optimal circumstances, such as the closest approach of a satellite. This column will be devoted to selecting the best method for each tool and to developing an understanding for how to apply them—skills which will be absolutely essential in some of our upcoming investigations.

The Problem

For the sake of this column, let's assume our task is to find the rise times of the satellite ***NOAA 14*** on 19 January 1997 for an observer located in Montgomery, Alabama. This type of task is a typical requirement of any ground station wishing to download weather imagery during an overhead pass of any one of a number of polar-orbiting weather satellites. Of course, the computer used at most weather ground stations performs many tasks—often including capturing the imagery—and tracks more than a single weather satellite, so we'd like to develop an algorithm that is as efficient as possible.

A Simple Approach

Perhaps the simplest way to approach such a problem is to generate an ephemeris (or table) of look angles for ***NOAA 14*** for the period of interest. We could then read through the ephemeris to determine when the satellite's elevation went from negative to positive to find the rise time. A graph of this ephemeris is shown in figure 1.

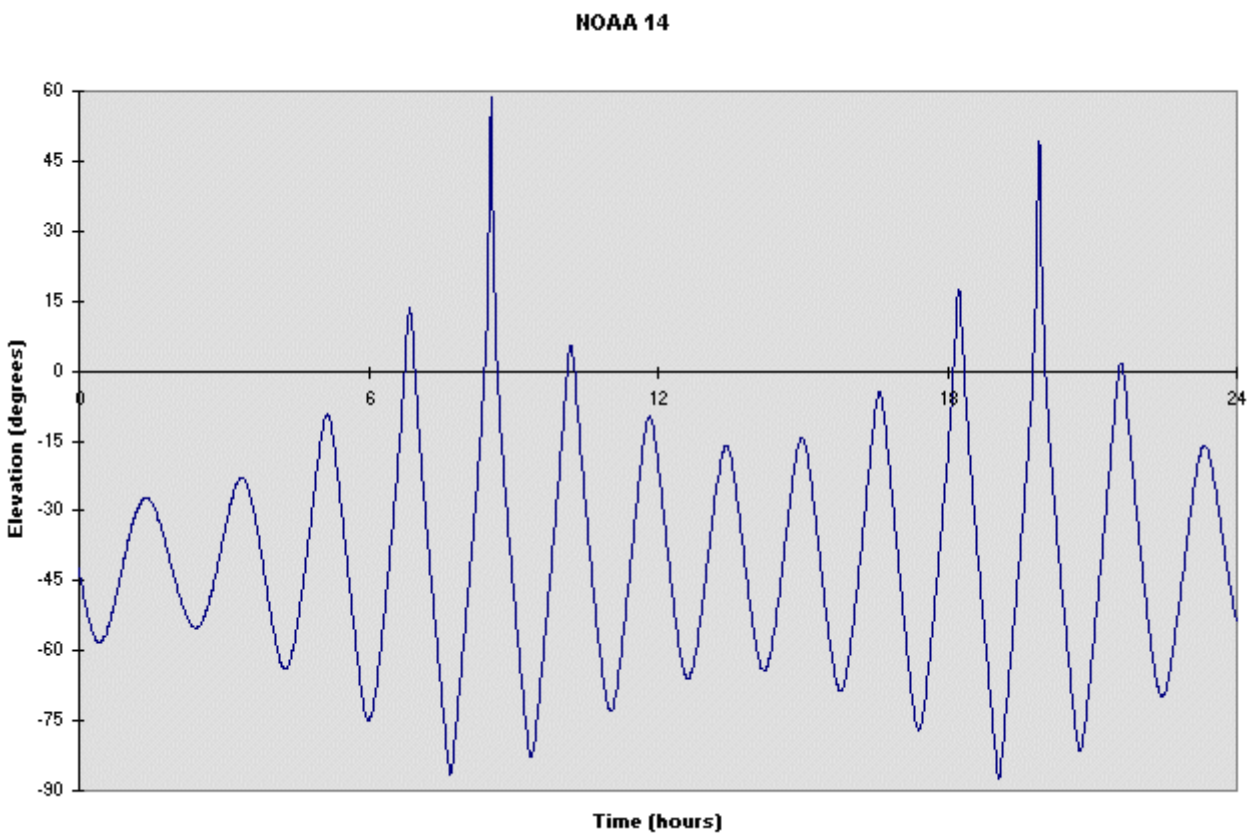

Figure 1. NOAA 14 Elevation Over Time

The first problem that presents itself is just what size time step we should use for our ephemeris. Make it too large and we may completely miss a pass. Make it too small and we'll spend the majority of our time calculating ephemeris values which we won't even use. For example, if we use an interval of one minute, we'll calculate 1,440 satellite positions just to find six rise times. And, to double our accuracy requires doubling our computations.

An Improved Solution

A more effective way to find our rise times would be to start with large steps and then reduce our step size as we get close to a solution. The ***bisection method*** uses just such an approach. However, before we can apply the bisection method, we must know that the interval we want to search contains a root.

Mathematically, a root exists in the interval (***a***,***b***) if ***f***(***a***) and ***f***(***b***) (where the function ***f*** represents the satellite's elevation) have opposite signs. This conclusion, of course, depends upon the function ***f*** being continuous over the interval (***a***,***b***). Since ***NOAA 14*** (or any other existing satellite) does not have the ability to warp space, any function of its position will be continuous.

Before we tackle the issue of how to select the proper interval, let's do a quick comparison of the efficiency gain which results from using this approach. For an interval of one day, it could take as many as 1,440 evaluations to find our rise time to an accuracy of one minute. If we use the bisection method, however, we would start with an interval of 1,440 minutes and then halve it, repeating the process with the smaller interval

that now contains our rise time. To achieve an accuracy of one minute would require only ***eleven*** calculations—an improvement of two orders of magnitude! And, doubling our accuracy now only requires one additional calculation rather than another 1,440! This property is known as ***linear convergence***.

The bisection method has several advantages: speed and the guarantee of convergence (i.e., if the interval contains a root, it will find it). To apply it, however, we must determine the proper interval for our search.

Bracketing the Target

While we now have a much more efficient means to home in on our prey, we have actually just traded one problem for another. Instead of searching the entire time period of interest, we must first find a time period where one end of the interval corresponds to an elevation above the horizon and the other to one below the horizon. An examination of figure 1 shows this to be nearly as difficult as our original problem. And, if we're not careful, we can select an interval which has more than one rise time—the bisection method is only guaranteed to find one of them.

We can be much more effective in the hunt for our prey, however, if we first understand something about its characteristics. In developing an algorithm such as the one we're working on, it helps to first visualize our quarry, as we've done in figure 1. As with any satellite orbit (with the possible exception of a perfectly geostationary orbit), the first thing we notice is that our function is periodic. This result is completely expected since both the satellite's orbit and the earth's rotation are periodic. In fact, closer examination of figure 1 shows there are actually two periods—one which corresponds to the satellite's orbit and the other to when the observer rotates under the orbital plane (which happens roughly twice a day).

Since we know that a root is bracketed by a minimum on one side and a maximum on the other, let's turn our attention to first developing a tool to find extrema. As it turns out, the approach is similar but subtly different. In searching for a root, splitting the interval in half turns out to be the optimal approach since only two points are needed to determine whether an interval contains a root. To find a minimum, however, requires three points.

To bracket a minimum over the interval (***a***,***b***), there must be a point ***c*** such that ***f***(***c***) is less than ***f***(***a***) and ***f***(***b***) and ***c*** is our best guess at the minimum (see figure 2). To continue our search, we choose a point ***x***, either in the interval (***a***,***c***) or (***c***,***b***). Let's say we choose the first interval. If ***f***(***x***) < ***f***(***c***), then the minimum is in the interval (***a***,***c***) and ***x*** is our new best minimum value (see the left side of figure 2); otherwise, the minimum is in the interval (***x***,***b***) and ***c*** is still our best minimum value (see the right side of figure 2). It turns out that the most efficient means of searching for the minimum (without considering the shape of our function) is to use the ***golden section search***.

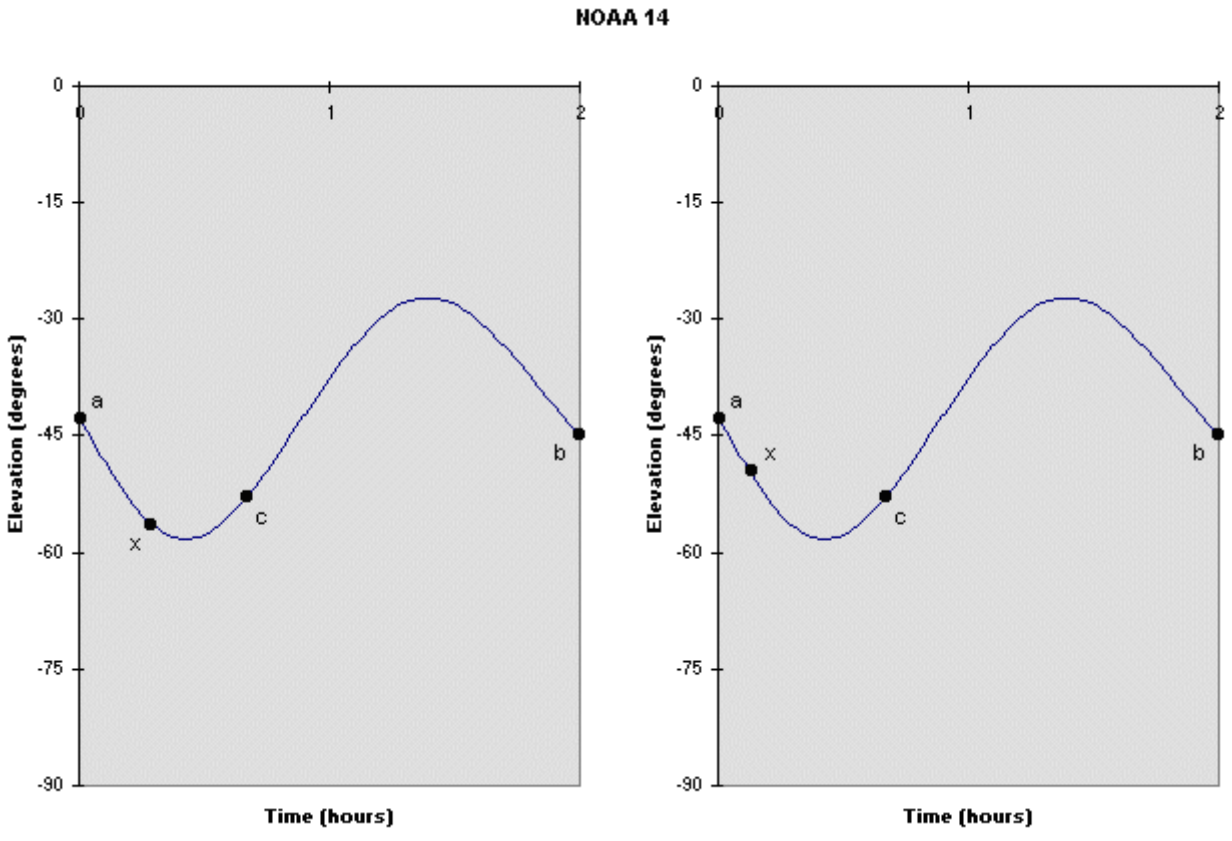

Figure 2. Bracketing a Minimum

It can be shown that the optimal choice for a new point within our bracketing interval is at approximately 38 percent (or 62 percent) of the interval. The name ***golden section*** or ***golden mean*** comes from the ancient Greeks who discovered this ratio. Using the golden section search will guarantee convergence and, like the bisection method, also converges linearly.

Tying It All Together

At first inspection, it would seem we've simply traded one problem for another. However, another look at figure 1 reveals that bracketing an extremum (minimum or maximum) should be much easier than bracketing a root. Each orbit of our target satellite produces a minimum and a maximum but does not necessarily produce a root. We make use of this information in the following approach.

To find the first pass of ***NOAA 14*** on 19 January 1997, let's examine the interval one orbital period ***before*** the beginning of the day (actually, we'll use slightly more than the orbital period since the interval between minima varies). The characteristics of our function assure us that there is at least one minimum in this interval. Finding the initial midpoint guess can be done using the golden section method—if the value at our new point is less than at the two endpoints, we search for a minimum; if it's greater, we search for a maximum; otherwise, we pick the other golden section value.

Once we know where a minimum is, we can find the next maximum by searching the interval from our minimum forward one orbital period. The same is true when finding a maximum—searching forward one orbital period brackets the next minimum. We can continue this process forward to find each successive minimum and maximum, finding them all very quickly. Each time we find a maximum, we would determine whether it is positive and for a minimum, whether it is negative. We would then only search for a root when one was properly bracketed. This method takes maximum advantage of the characteristics of our function to find what we're looking for.

Making It More Efficient

One limitation of the bisection and golden section methods is that they both ignore the shape of the function under investigation. While they both guarantee convergence and converge linearly, there are methods which can produce superlinear convergence by taking the derivative of the function into account. One such method which you may be aware of is the ***Newton-Raphson method***. Unfortunately, this method only works when you can analytically determine the derivative of the function.

There are, however, adaptations of searches for roots and extrema—referred to as ***Brent's methods***—which can provide superlinear convergence without the need for a derivative for well-behaved functions (such as the one we're working with). Our basic search strategy is the same except that the bisection and golden search methods are replaced by the more efficient Brent's methods. A detailed discussion of these methods—together with source code—can be found in chapters 9 and 10 of ***Numerical Recipes in C: The Art of Scientific Computing*** (Second Edition) by William H. Press, et al. I highly recommend this reference.

Whether we use these tools to generate pass schedules, as we have here, or in looking for satellite transits across the sun or the moon, the ability to quickly search for when a particular event occurs—or whether it even happens at all—will be a most useful skill. Once you've mastered their use, you'll wonder how you ever did without them!

If you have any questions or comments regarding this column, please feel free to contact me at TS.Kelso@celestrak.com. Until next time, keep looking up!

Dr. T.S. Kelso [TS.Kelso@celestrak.com]
Follow CelesTrak on Twitter @TSKelso
Last updated: 2019 Dec 28 02:09:45 UTC
Accessed 69,662 times
Current system time: 2020 Jun 26 13:00:22 UTC
CelesTrak's Simple Privacy Policy