Doppler Shift-Orbit Determination From Multiple Ground Stations

Winston Moy
AE 548: Astrodynamics
wsmoy@umich.edu

Abstract—This paper describes methods of using Doppler shift in orbit determination, both past and present, and explains certain limitations of each method. It will then discuss a method for determining a satellite’s position and velocity using simultaneous Doppler measurements by multiple ground stations.

TABLE OF CONTENTS
1. INTRODUCTION ................................................. 1
2. PROBLEM STATEMENT AND SETUP .................. 2
3. ATTEMPTS AT A CLOSED-FORM SOLUTION ..... 2
4. A LESS-ELEGANT NUMERICAL SOLUTION ...... 3
5. RESULTS ............................................................ 4
6. FUTURE WORK ................................................. 5
7. CONCLUSION ..................................................... 6
REFERENCES ......................................................... 6

1. INTRODUCTION
Since the inception of mankind’s national space programs in the 1950’s, there has always been a need to locate and track spacecraft in orbit around Earth. Methods of orbit determination such as those discussed in class can be quite costly. They often require a high-power radar array, high-precision optics, or other manner of specialized equipment. An alternative method of satellite tracking that can be performed with a non-directional antenna utilizes Doppler shift, which is a byproduct of General Relativity. Doppler measurements require only a single calibrated signal and a sensitive receiver. They give scalar information on a moving object’s rate of closure.

One of the earliest instances of Doppler orbit determination was in 1959, spurred by the launch of Sputnik 1. Listening to Sputnik fly overhead, U.S. scientists William Guier and George Weiffenbach at APL’s research center realized that Sputnik’s 20.005 MHz radio beacon could be used to estimate its velocity. By plotting the Doppler shift of a flyover it was also possible to ascertain Sputnik’s point of closest approach, a technique carried over from APL’s work done with guided missiles. Their work, something of a hobby at the time, caught the eye of APL’s lab director and was allocated computing time on a Univac 1200F in order to establish the method’s ultimate accuracy. For near-earth satellites, measured data from Sputnik was found to produce a “useful” set of orbit parameters. Future work reversed this process of orbit determination to provide a ground positioning system intended for use on ballistic missile submarines.

Guier and Weiffenbach’s work succeeded because the 6 orbit parameters are not degenerate and could be uniquely worked out from the measurements taken. In 1960, Imre Izsak published a different method of Doppler orbit determination using three ground stations and two timed readings instead of tracking a satellite from a single station as it flew from horizon to horizon. It required certain assumptions however, most importantly that the satellite’s range from one of the ground stations was known. From there, integration over the time-step would reveal the satellite position relative to the remaining ground stations.

Continuous improvements in Doppler determination of orbits were the precursor for the modern (and French) Doppler Orbitography and Radiopositioning Integrated by Satellite (DORIS) system. The primary function of DORIS is to allow a satellite to determine its own orbit with high-precision, and has a secondary function of supplementing ground-position systems like GPS. Unlike previous Doppler-based systems, however, DORIS does not rely on ground stations to sense and pinpoint a satellite’s location. Instead, fifty to sixty radio beacons are scattered around the globe and their transmissions are picked up on the satellite, allowing it to continually refine its knowledge of its orbital elements. It is credited with allowing the measurement of earth’s surface features down to an accuracy of 2 centimeters.

Figure 1 - Locations of DORIS beacons as mapped on Google Earth.
2. PROBLEM STATEMENT AND SETUP

The objective of the orbit determination techniques discussed below is to determine the position and velocity of a satellite (with a radio beacon) relative to multiple ground stations using readings taken at a single instant in time with zero prior knowledge of the target’s initial states. The problem was initially attempted in two dimensions, with the intention of generalizing the method for three dimensions.

The problem begins with \( n \)-number of Doppler readings from ground stations whose positions are known absolutely, and \( n \) corresponds to the number of unknown states. The origin is arbitrarily taken to be one of the ground stations, and the coordinate system is Cartesian.

\[
\begin{align*}
\vec{r}_1 &= \vec{r}_2 + \vec{m}_2 \\
& \vdots \\
\vec{r}_1 &= \vec{r}_n + \vec{m}_n 
\end{align*}
\]

where \( \vec{r}_1 \) is the position of the satellite relative to the origin, \( \vec{m}_i \) are the positions of the ground stations, and \( \vec{r}_1 \) is the position vector of the satellite relative to the origin.

\[
d_1 = v \cos \theta
\]

Figure 2 - 2D problem setup with 3 ground stations.

In the two dimensional simplification, there are eight unknowns, \( x \) and \( y \) corresponding to the satellite’s velocity and each of the 3 radii from the ground stations. Six of these unknowns are linked, the radii from each station. Their relation is described as follows:

\[
\vec{r}_1 = \vec{r}_2 + \vec{m}_2
\]

\[
\vdots
\]

\[
\vec{r}_1 = \vec{r}_n + \vec{m}_n
\]

\[ (1) \]

\[ (2) \]

\[ (3) \]

\[ (4) \]

\[ (5) \]

\[ (6) \]

\[ (7) \]

\[ (8) \]

\[ (9) \]

\[ (10) \]

\[ (11) \]

\( d \) is taken to be this radial velocity. It may be calculated as the velocity times the cosine of the angle between the velocity vector and the ground-station-to-satellite radius. It may also be calculated as follows:

\[
d_1 = \frac{\vec{v} \cdot \vec{r}_1}{||\vec{r}_1||}
\]

3. ATTEMPTS AT A CLOSED-FORM SOLUTION

Using the dot product definition and equation (2), the equations of radial velocity become:

\[
d_1 ||\vec{r}_1|| = r_{1x} v_x + r_{1y} v_y 
\]

\[
d_2 ||\vec{r}_1 - \vec{m}_2|| = (r_{1x} - m_{2x}) v_x + (r_{1y} - m_{2y}) v_y 
\]

\[
\vdots
\]

\[
d_n ||\vec{r}_1 - \vec{m}_n|| = (r_{1x} - m_{nx}) v_x + (r_{1y} - m_{ny}) v_y
\]

Because all equations could be found in terms of \( \vec{r}_1 \), all further references to the radius variable will be shortened to \( \vec{r} \). In order to simplify the determination of a solution to four equations for four unknowns, \( m_{ny} \) was assumed to be zero. This would be analogous to all ground stations residing on a single plane (not necessarily the earth being flat).

Another condition applied to the setup of the problem was to set \( v_y \) equal to zero. This was done to reduce the complexity of calculating the solution by minimizing instances of large substitutions. Not neglecting \( v_y \) would still result in the same methodology. Applied to a real world scenario, this assumption means that the target satellite’s orbit would have to be of low eccentricity and its position nearly overhead.

Beginning from equation (4), \( v_x \) was found to be:

\[
v_x = \frac{d_1 ||\vec{r}_1|| r_2}{r_x}
\]

This equation for \( v_x \) was put into equation (5), and several terms simplified for convenience of writing.

\[
\gamma = \left(\frac{d_1}{d_2 r_x}\right)^2
\]

\[
\theta = (r_x - m_{2x})^2
\]

\[
r_2^2 = \frac{\theta(y r_2^2 - 1)}{1 - y \theta}
\]

These equations (7 through 10) are substituted into equation (6) specialized to station 3, resulting in the following expression:

\[
d_3 ||\vec{r}_1 - \vec{m}_3|| = (r_{1x} - m_{3x}) v_x + (r_{1y} - m_{3y}) v_y
\]

\[ (11) \]
\[
\begin{align*}
\frac{d_3}{r_x} \sqrt{(r_x - m_{3x})^2 + \frac{(r_x - m_{2x})^2 \left( \frac{d_3}{d_2 r_x} \right)^2 (r_x - m_{2x})^2}{1 - \left( \frac{d_3}{d_2 r_x} \right)^2 (r_x - m_{2x})^2}} = \\
\frac{d_1}{r_x} \sqrt{r_x^2 + \frac{(r_x - m_{2x})^2 \left( \frac{d_3}{d_2 r_x} \right)^2 (r_x - m_{2x})^2}{1 - \left( \frac{d_3}{d_2 r_x} \right)^2 (r_x - m_{2x})^2}} (r_x - m_{3x})
\end{align*}
\] (12)

The equation is entirely in terms of \(r_x\) and known constants, but even the mighty Mathematica cannot solve it. It may be possible to use numerical method to solve for \(r_x\) from the form of equation (12) but if \(v_y\) and \(m_{ny}\) are not taken to be zero, and/or this method is applied in three-dimensional space, then the complexity increases exponentially. Equation (12) is problematic to solve at best, after only 2 substitutions. The full 3D form of the equation would require 5 substitutions, and would probably be finished sometime in February if attempted by hand.

4. A LESS-ELEGANT NUMERICAL SOLUTION

Because of the inherent difficulty in achieving a closed-form solution for determining a satellite’s position and velocity from Doppler measurements, it is necessary to turn to a numerical optimization method. The alternate method used in this paper essentially “meshes” 3D space into a grid of search locations. At each location, a unit velocity vector will be tested in discretized directions in the 3D. The calculated Doppler shifts resulting from each test location and direction will be compared to a given value, and the errors recorded. A ball, of radius slightly less than the original mesh resolution, around the test location with the smallest error residual will be re-meshed with finer resolution. The process will be repeated recursively until the minimum residual converges.

Initially, MATLAB was to be used to run the Doppler comparison program, but early testing of its efficiency ruled it out as being egregiously slow. The test pattern in figure 5 took approximately 7 minutes to generate. The test involved the creation of just fewer than 50,000 vectors over 15,000 points. These numbers are at least one order of magnitude lower than desired. The calculation of dot products is actually quite simple and can be done in C++ without the need to invoke any of MATLAB’s enormous, yet enormously capable math libraries.

MATLAB was however used to calculate test Doppler data to be used, since multiple sets of data could be created without the need to recompile a program. A set of Doppler data was created assuming a hexagonal array of ground stations, with the surface spanning the array forming the x-y plane of the coordinate system. This is the “flat Earth” approximation that was used in section 3. The data was then hardcoded into the C++ program. All units in this paper are based on kilometers.

Figure 4 - Early conceptualization of simulated Doppler array.

Figure 5 - Example of discretization of 3D space and test unit vectors in a 200x200x100 unit range. Bottom image is the same space rotated.
In the most rudimentary sense, the Doppler determination program is a comparator. Given a set of data points, it tests systematic guesses against the original and seeks to minimize the error residual.

The guesses in this case are taken from a 3D volume over the simulated Doppler detection array that is defined by X-Y-Z bounds and discretized by an arbitrary number of steps (variable: ‘ns’). At each point in 3D space, a pair of nested loops tests a series of velocity magnitudes and direction.

The test position and velocity are fed into a function that determines the apparent radial velocity (Doppler shift) that would be observed at each ground station. These numbers are compared to the original, given data, and the absolute value of the difference at each stations is summed. This residual value is what the program attempts to minimize.

Subsequent loops through this entire process shrink the region of interest (but preserving the number of discretized points) until the error residual is below a threshold accepted for convergence.

**Pseudo-Pseudocode**

1. Generate bounds and step size (i.e. dx, or dy)
2. For loop (var: i) steps through discretized x-axis.
   - For loop (var: j) steps through discretized x-axis.
     - For loop (var: k) steps through discretized x-axis.
       - For loop (var: z) determines magnitude of radius from each ground stations to satellite location guess.
       - For loop (var: m) steps through increasing velocity magnitudes.
         - For loop (var: l) tests different velocity directions.
           - For loop (var: z) sums residuals from each ground stations.
         - If residual is lower than any previous, save guess.
   - Optionally, if residuals have peaked… or rather ceased minimizing after a certain number of loops within i-Loop, skip.
3. If residual of best guess exceeds convergence criteria, center bounds of region of interest about the best guess, +/- the step size for each dimension.
4. Return to step 1.

**5. RESULTS**

The numerical approximation program was primarily tested on two datasets. The first dataset had the satellite resting on the x-axis with no y or z velocity. This was the simplest setup and was intended to test the function of the program, rather than its ability to achieve convergence under varied operating conditions.

```c
//Doppler-Based Radial Velocity Data Set 1
//Actual Sat Position is [60;0;600]
//Actual Sat Velocity is [30;0;0];
double d[] = {2.985111570629968,-1.968536144774360,-1.968536144774360,-6.722834029176015,-6.722834029176015,-11.141720290623110};
```

```c
//Doppler-Based Radial Velocity Data Set 2
//Actual Sat Position is [120;28;700]
//Actual Sat Velocity is [12;13;0];
//double d[] = {2.53817593758038,0.988668281386897,2.674579407662616,-2.67711172493201,0.983117498687291,-2.483013640465591};
```

The first Doppler dataset showed that the program functioned as intended and was able to uniquely locate the original satellite. With the convergence criteria specified, it took approximately 2 contractions of the search grid to determine.

**Sample of Program Output**

```
ErrMin = 60 -0.48   600     V:30    Ang= 1.09332e-15        resid = 0.0021706
ErrMin = 60 -0.4    600     V:30    Ang= 1.09332e-15        resid = 0.00180828
ErrMin = 60 -0.32   600     V:30    Ang= 1.09332e-15        resid = 0.00144618
ErrMin = 60 -0.24   600     V:30    Ang= 1.09332e-15        resid = 0.0010843
ErrMin = 60 -0.16   600     V:30    Ang= 1.09332e-15        resid = 0.000722642
ErrMin = 60 -0.08   600     V:30    Ang= 1.09332e-15        resid = 0.000361209
ErrMin = 60 0     600     V:30    Ang= 1.09332e-15        resid = 4.21885e-15
```

Figure 7 - Subset of program output during search showing decreasing error and convergence. Full text output occupies 55 pages and will be included as attachment.
The second dataset posed a greater challenge. Because of the asymmetry in the setup (the satellite not residing on one of the axes), minimization of the residual was more difficult. The finite resolution also created problems, insufficient grid resolution caused the program to miss the true location of minimum residual and terminate at a point slightly removed from the satellite’s true location. This occurrence was likely due to some form of sampling aliasing. Tweaking the search and discretization parameters would sometimes change whether the program over or underestimated certain states. The natural solution was to drastically increase the discretization of the search volume but then the search time increased to impossible lengths. Alternate solutions considered, but not fully explored, are discussed in the following section.

Further testing of the program’s search parameters revealed that the convergence criterion was highly sensitive to direction. Sampling velocity vectors even every three degrees could easily overlook the true location of minimum residual. Originally, when testing velocity vectors between \(+/-\) pi with 50 increments, the program would determine an incorrect region of interest to investigate in the subsequent loop. Even doubling the granularity of the search failed to solve the problem. It was not until the program searched degree by degree that it converged to the true satellite position.

<table>
<thead>
<tr>
<th># Ang. Divisions</th>
<th>Actual Pos/Vel</th>
<th>Found Pos/Vel</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>[120 28 700]</td>
<td>[108.9 38.1 820.7]</td>
</tr>
<tr>
<td></td>
<td>[12 13 0]</td>
<td>20.67 @ 47.384°</td>
</tr>
<tr>
<td></td>
<td>aka 17.69 @ 47.29°</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>[121.98 26.2 668.7]</td>
<td>16.93 @ 47.26°</td>
</tr>
<tr>
<td>180</td>
<td>[120 28 700.45]</td>
<td>17.70 @ 47.29°</td>
</tr>
</tbody>
</table>

Fortunately, increasing the angular resolution of the search only affects a single loop, the innermost loop, and increasing the search time linearly rather than exponentially.

### 6. Future Work

The clear limiting factor in the brute force computation of orbital elements from Doppler data is the computer upon which the calculations are performed. On a linear CPU scheduling scheme, this process is relatively slow. The 4.04 GHz computer on which this Doppler determination simulation was run can perform roughly 100 gigaFLOPS. Using parallel execution on a consumer-level GPU (which any CSE grad student worth his or her salt should be able to do) a theoretical performance on the order of teraFLOPS could be achieved. This would absolutely exclude the use of programs like MATLAB, but the simple dot product calculations could easily be programmed in CUDA or another GPU-programming language. In fact, porting it from C++ code would require only a few additional lines for memory management and threading, with little change to the overall structure of the code.

The ability to dynamically change the resolution of the search grid based on the magnitude of the error residual would drastically improve efficiency, allowing the program to effectively skip areas unlikely to contain a solution. This is similar to how many FEA programs “auto-mesh” models, selectively investing computation time in regions of greater uncertainty.

Alternately, if something like a 3D pattern recognition technique could be developed that isolate features like the error residual plateauing in a particular region, then the subsequent contraction of the search matrix could be sized more appropriately.
One last interesting permutation of this sort of guess-and-check search would be to use a random Monte Carlo search, which often is employed in hunting for things of unknown distribution (ex. gas fields). Coupled with a dynamic mesh sizing algorithm, the hunt for a satellite’s true location from Doppler data could be drastically shortened.

Furthermore, if certain assumptions about the satellite’s position or velocity could be made, then even fewer iterations could be used. These assumptions include:

- Satellite’s orbit direction, prograde vs. retrograde
- Satellite’s approximate altitude
  - In addition to flattening the search grid, a better estimate of initial velocity could be applied.
- Any localization information to reduce the size of the search grid (ex. satellite is “inside” the ground station array/directly overhead)

One particular mathematical operation was found to drastically increase computation time, sqrt(). This arose from an earlier attempt to perform a rudimentary abs() function on Double data types. Instead, a Boolean statement was used to negate the residuals as necessary. Sqrt() is also used in calculating the magnitude of the radius between ground stations and satellite. If the necessary equations of Doppler shift are redone to use radius squared instead of a normal radius (squaring everything else), computation time may be further reduced.

Also for future investigation is this method’s robustness when error is introduced into the system. It has already been observed that the Doppler determination algorithm is highly sensitive to estimates of angular deviation. Small errors in the magnitude of recorded Doppler measurements may cause a satellite to be incorrectly located, or even cause convergence criteria to fail (i.e. an impossible set of Doppler measurements). The magnitude of this effect would need to be tested using appropriately noisy (or real) data.

The seventh loop to optimize for Z-velocity must also be initialized in the “full version” of the Doppler determination program. Currently, one complete run from initialization to convergence takes anywhere from 3 minutes to 15 minutes to complete. With the current discretization fineness, ascertaining another free state would take 100 times longer, and was therefore omitted for sanity.

7. CONCLUSION

Although not necessarily a simple or popular method for orbit determination, Doppler measurements are nonetheless powerful and contain a large amount of data that can be extracted. The numerical method described in this paper can determine a satellite’s position and velocity from Doppler data taken at a single point in time, which can then be used to ascertain its orbital elements. The program itself however has much room for optimization and, it serves primarily as a proof of concept.