

# LASSO

Localize, Attribute  
SatelliteS in Orbit  
P20151

## TDoA Algorithm



<https://twitter.com/tanyaofmars/status/8938928634859192>

# The TDoA simulator

## Agenda

1. Time Difference of Arrival (TDoA) Introduction.
2. The Satellite Problem
3. Finding the Best Solution on a Plane
4. Fitting a Line through the solution on each plane.
5. Unit Tests
6. Simulating the Inputs
7. Deriving the Uncertainty based on Simulated Inputs

# Purpose Of Slides

- Inform reader about how the Time Difference of Arrival algorithm works.
- This is not designed as presentation slides. The slides are filled with information and comments. It is meant to be read like a paper (but with lots of pictures).

# TDoA Basics

By Anthony Iannuzzi

TDoA data is receiver locations and time differences between each receiver.

$$\mathbf{R}_1 = [x_1, y_1, z_1]$$

$$\mathbf{R}_2 = [x_2, y_2, z_2]$$

$$\mathbf{R}_n = [x_n, y_n, z_n]$$

The receivers must all be measured in the same frame.

TD represents the Time Difference data. It is an upper triangular matrix.

Each entry is the time difference between the receivers represented by that row and column number.

$$TD = \begin{matrix} & \begin{matrix} 1 & 2 & \cdots & n \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ \vdots \\ n \end{matrix} & \begin{bmatrix} 0 & t_{12} & \cdots & t_{1n} \\ 0 & 0 & \ddots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

The diagonals are zero as a station compared to itself has no time difference. The lower triangle contains redundant information and is set to 0.

# TDoA data creates hyperbolas and hyperboloids.

In 2 Dimensions

$$\frac{4(x'')^2}{\delta^2} - \frac{4(y'')^2}{4D^2 - \delta^2} = 1$$

In 3 Dimensions

$$\frac{4(x'')^2}{\delta^2} - \frac{4(y'')^2}{4D^2 - \delta^2} - \frac{4(z'')^2}{4D^2 - \delta^2} = 1$$

where

$\delta$  is the distance difference

$D$  is half the distance between stations

$\delta$  the distances differences is calculated by

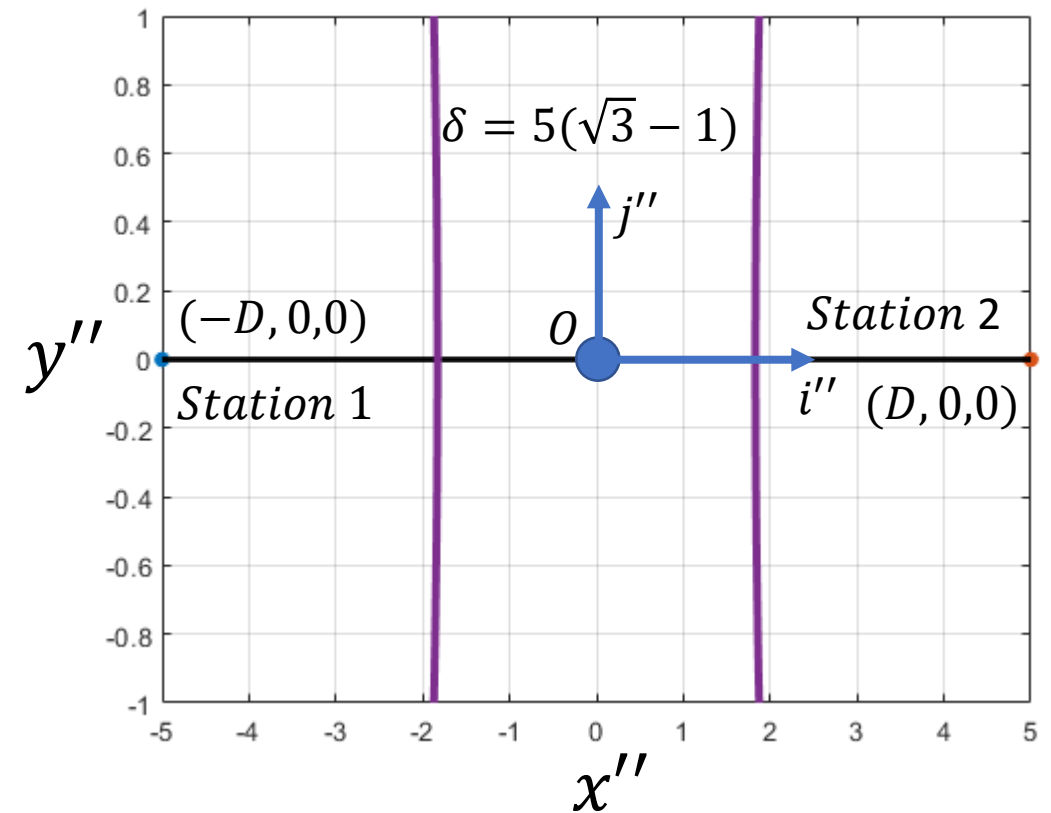
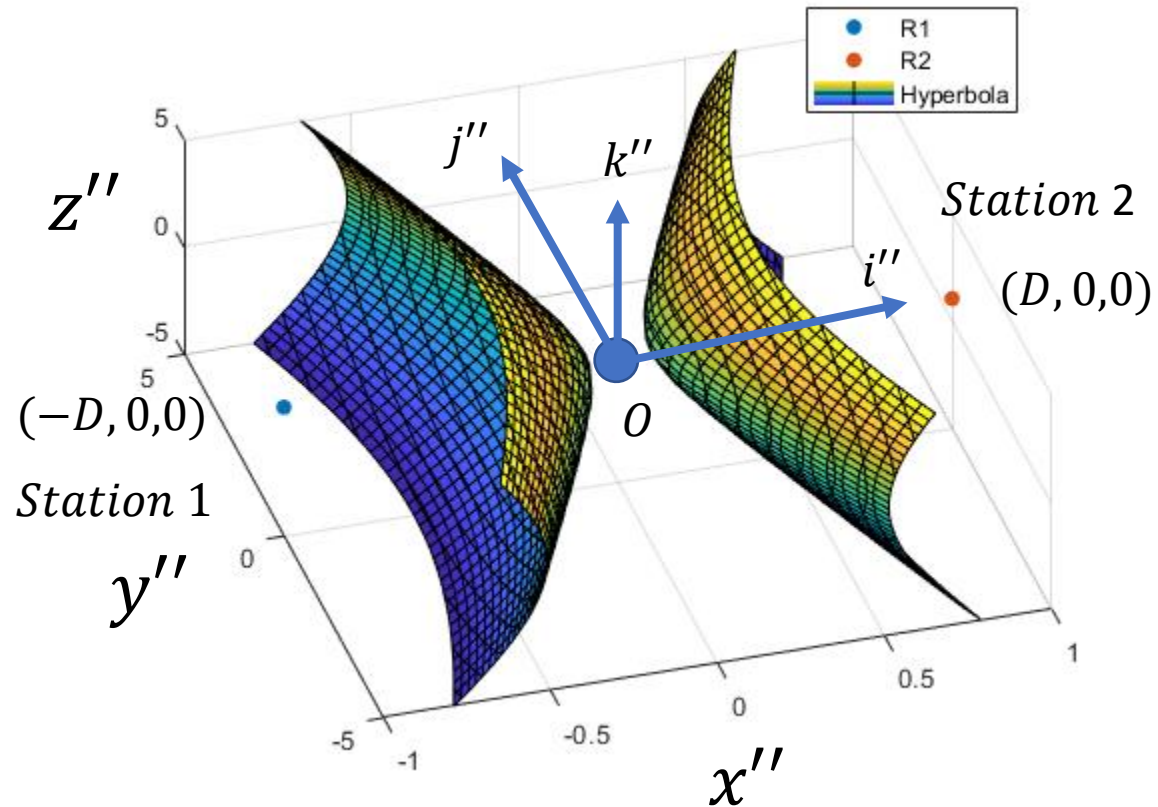
$$\delta = TD * 3e8$$

$3e8$  is the speed of light in  $m/s$ .

The half distance between stations is

$$D = \frac{||\mathbf{R}_2 - \mathbf{R}_1||}{2}$$

# Graphically what hyperboloid and hyperbola look like.

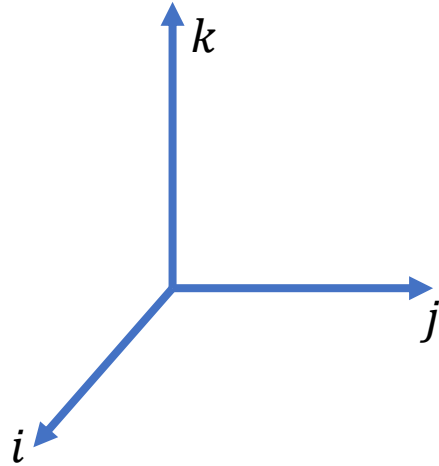


These Hyperbolas assume the stations are located at  $(\pm D, 0, 0)$ . This is called the **Body Frame**. We will now transform this hyperbola into the **Fixed Frame**; the frame the stations are measured in.



# Coordinate Transformation Introduction

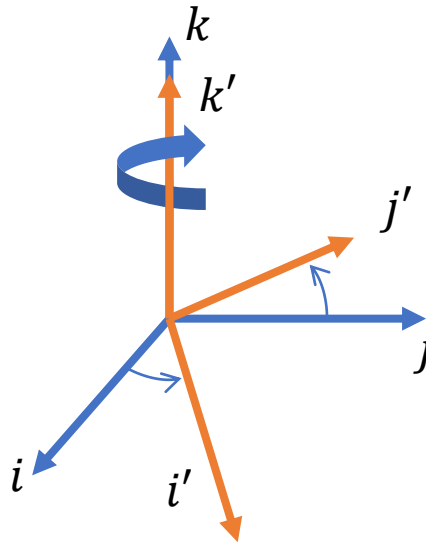
*Azimuth, Elevation Rotations*



**Fixed Frame**

$$RM = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A point measured in the body frame,  $\mathbf{r}_b$  can be converted to the fixed frame,  $\mathbf{r}_f$  by multiplying by the rotation matrix

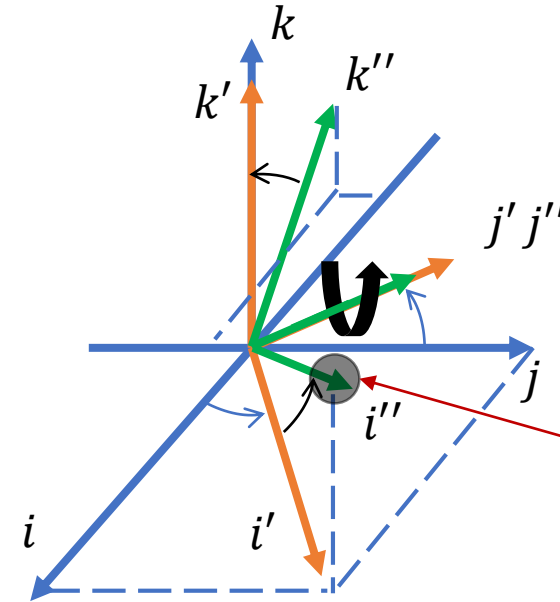


**Intermediate Frame**

$$RM_z = \begin{pmatrix} \cos a & -\sin a & 0 \\ \sin a & \cos a & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$RM_{b \rightarrow f} = RM_z * RM_y$$

$$\mathbf{r}_f = RM_{b \rightarrow f} * \mathbf{r}_b$$



The point we want to transform

**Body Frame**

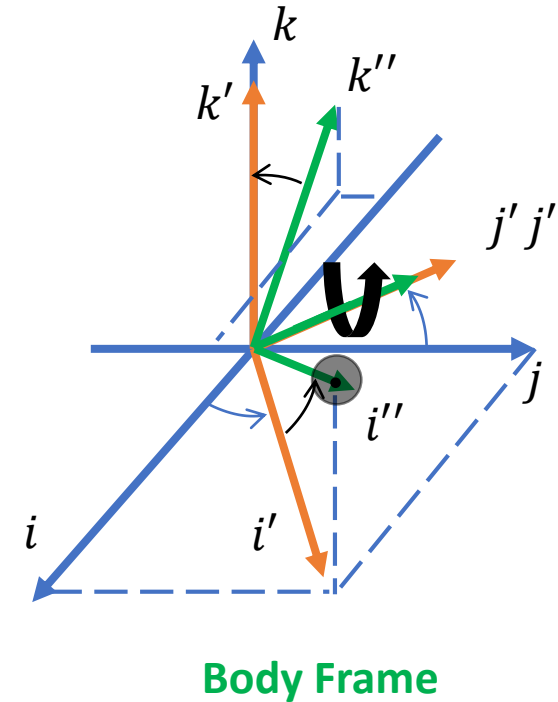
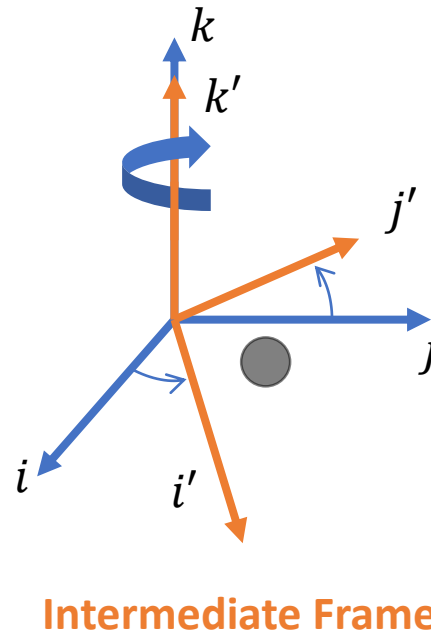
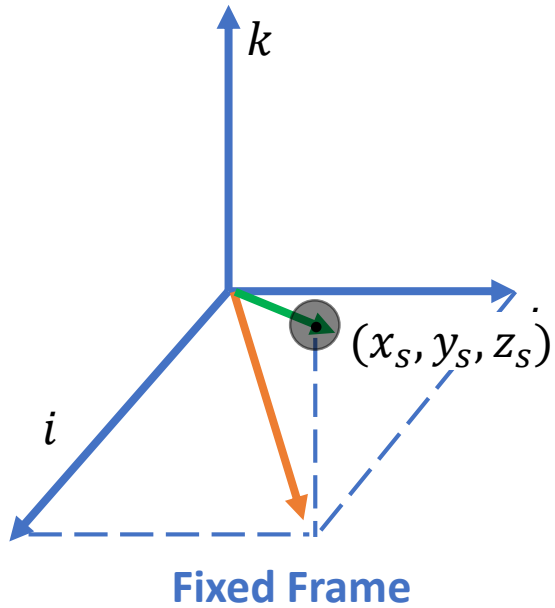
$$RM_y = \begin{pmatrix} \cos b & 0 & -\sin b \\ 0 & 1 & 0 \\ \sin b & 0 & \cos b \end{pmatrix}$$

$$RM_{b \rightarrow f} = \begin{pmatrix} \cos a \cos b & -\sin a & -\cos a \sin b \\ \sin a \cos b & \cos a & -\sin a \sin b \\ \sin b & 0 & \cos b \end{pmatrix}$$



# Calculating Azimuth and Elevation

*How to calculate Azimuth, Elevation Rotations*



Given:  $[x_s, y_s, z_s]$

Azimuth

$$a = \tan^{-1} \frac{y_s}{x_s}$$

Elevation

$$b = \tan^{-1} \frac{z_s}{\sqrt{x_s^2 + y_s^2}}$$

If we have the point measured in the fixed frame, then we can calculate the azimuth and elevation with respect to the fixed frame.

# Transforming Coordinate Axes that have different origins.

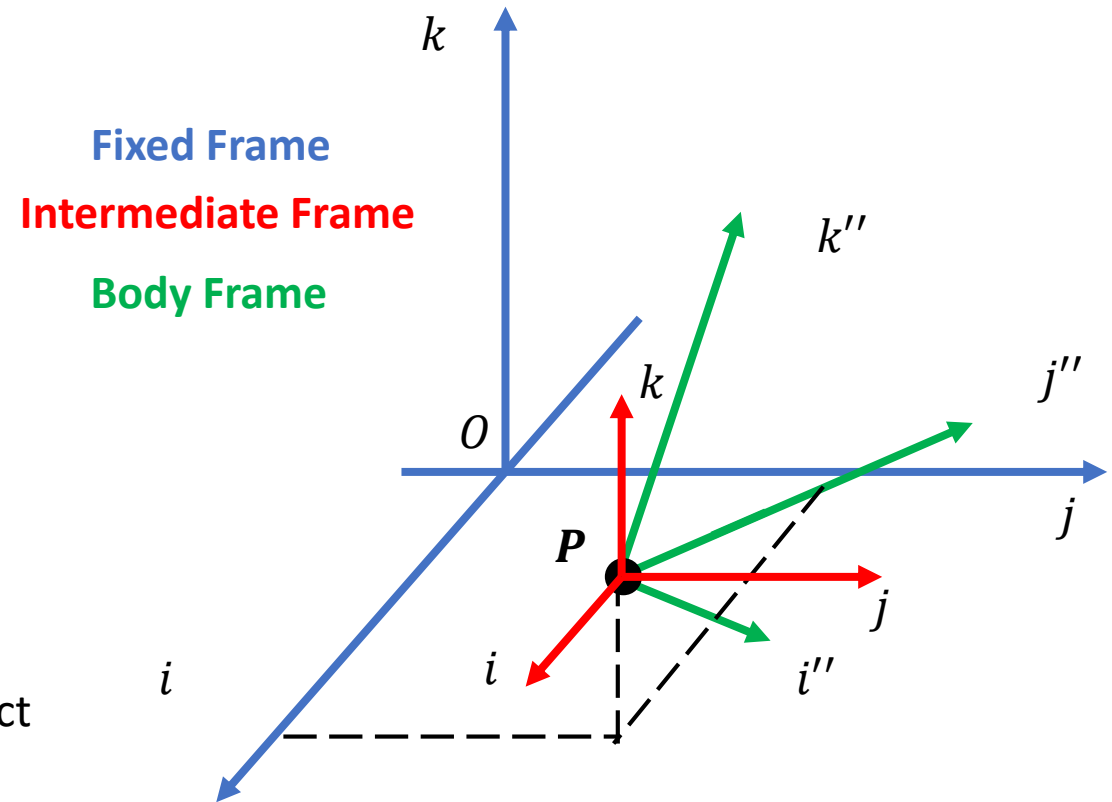
1. Rotate to the Fixed coordinate frame orientation.
2. Then shift coordinate frame.

Given:  $\mathbf{P} = [x_0, y_0, z_0]$

$$\mathbf{r}_f = \underbrace{RM_{b \rightarrow f} * \mathbf{r}_b^T}_{1.} - \underbrace{\mathbf{P}^T}_{2.}$$

Physically, each entry in the RM can be regarded as a dot product between each body and fixed frame axis:

$$RM_{b \rightarrow f} = \begin{array}{c} i'' \\ j'' \\ k'' \end{array} \begin{array}{ccc} i & j & k \\ \cos a \cos b & -\sin a & -\cos a \sin b \\ \sin a \cos b & \cos a & -\sin a \sin b \\ \sin b & 0 & \cos b \end{array}$$



# Transforming the Hyperbola or Hyperboloid to the Fixed Frame. Known Data.

Given Absolute Position of Stations:

$$\mathbf{R}_1 = [x_1, y_1, z_1]$$

$$\mathbf{R}_2 = [x_2, y_2, z_2]$$

$$D = \frac{||\mathbf{R}_2 - \mathbf{R}_1||}{2}$$

$$\mathbf{O} = \mathbf{R}_1 + \frac{\mathbf{R}_2 - \mathbf{R}_1}{2}$$


x axis always points from station 1 to 2.

$$[x_s, y_s, z_s] = \mathbf{R}_2 - \mathbf{O}$$

# Transforming the Hyperbola or Hyperboloid to the Fixed Frame. The new Equation

Let  $\mathbf{r} = [x - x_0, y - y_0, z - z_0]^T$ , then the hyperboloid becomes:

$$\frac{4(\mathbf{r} \cdot \mathbf{RM}(:, 1))^2}{\delta^2} - \frac{4(\mathbf{r} \cdot \mathbf{RM}(:, 2))^2}{4D^2 - \delta^2} - \frac{4(\mathbf{r} \cdot \mathbf{RM}(:, 3))^2}{4D^2 - \delta^2} = 1$$

 dot product


Substitute  $[x'', y'', z'']$  for

$\mathbf{RM} * \mathbf{r}^T$  split amongst the terms.

# Hyperboloid in Fixed Frame

Let  $\mathbf{r} = [x - x_0, y - y_0, z - z_0]^T$ , then the hyperboloid becomes:

$$\frac{4(\mathbf{r} \cdot \mathbf{RM}(:, 1))^2}{\delta^2} - \frac{4(\mathbf{r} \cdot \mathbf{RM}(:, 2))^2}{4D^2 - \delta^2} - \frac{4(\mathbf{r} \cdot \mathbf{RM}(:, 3))^2}{4D^2 - \delta^2} = 1$$

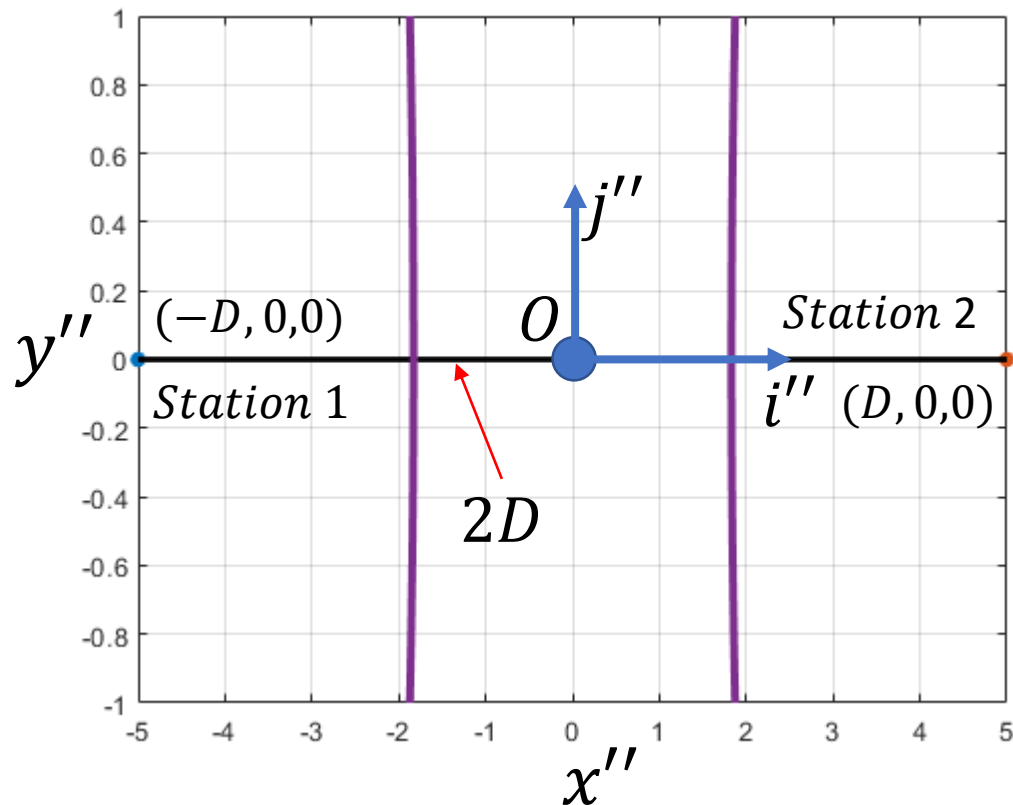
 *dot product*

Substitute  $[x'', y'', z'']$  for

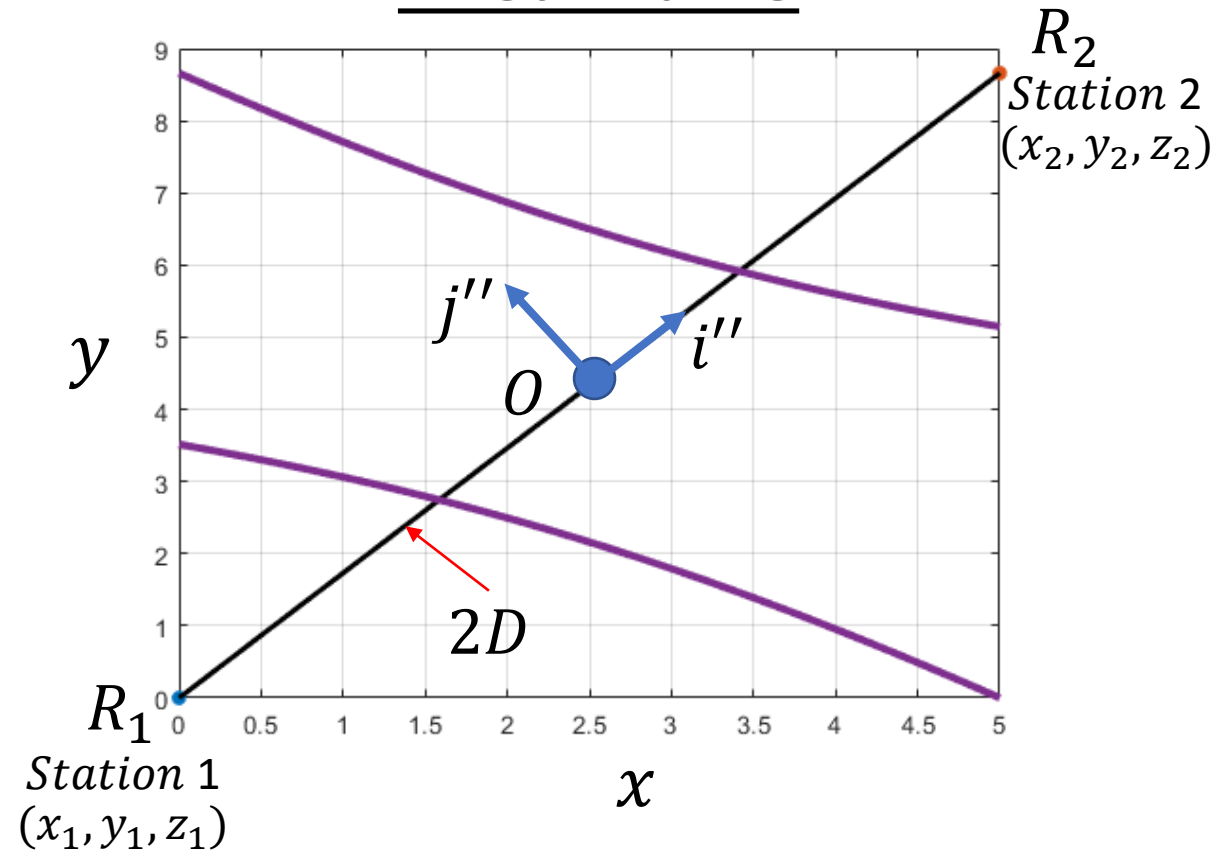
$\mathbf{RM} * \mathbf{r}^T$  split amongst the terms.

# Transforming the Hyperbola or Hyperboloid to the Fixed Frame. Visualization

**Body Frame**



**Fixed Frame**



# Creating the Hyperboloids and Hyperbolas algorithm.

Inputs is *TDoA Data*

- Receiver Locations in the **Fixed Frame**
- Distance Difference

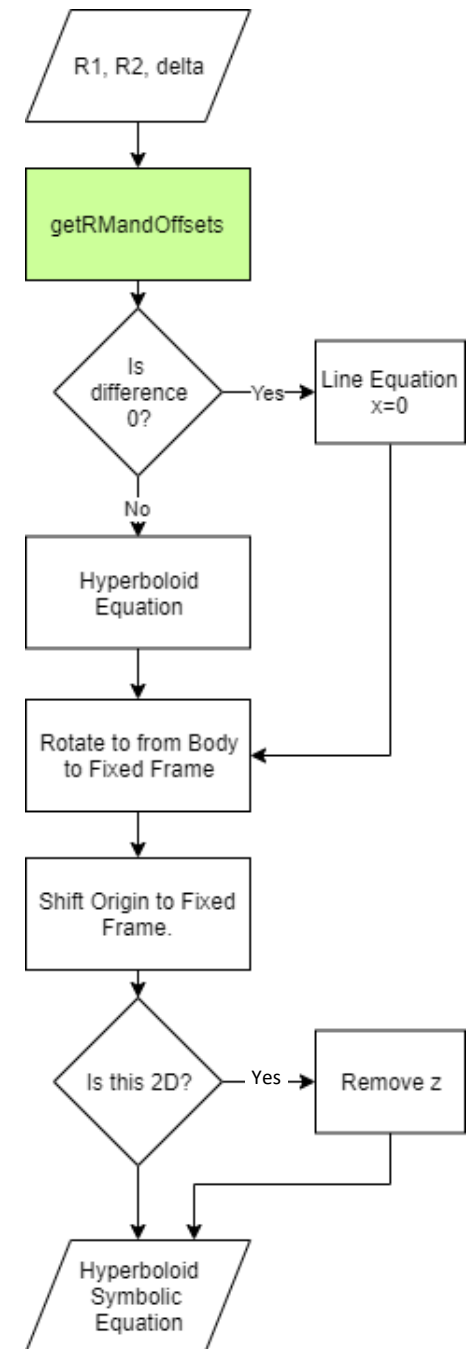
Algorithm:

1. Calculate  $\mathbf{RM}_{b \rightarrow f}$  and offset  $P$
2. Create Hyperboloid in **Body Frame**.
3. Transform to **Fixed Frame**.

Outputs:

- Symbolic Equation of Hyperboloid

$$\frac{4(\mathbf{r} \cdot \mathbf{RM}(:, 1))^2}{\delta^2} - \frac{4(\mathbf{r} \cdot \mathbf{RM}(:, 2))^2}{4D^2 - \delta^2} - \frac{4(\mathbf{r} \cdot \mathbf{RM}(:, 3))^2}{4D^2 - \delta^2} = 1$$





# Changing to a 1 sided Hyperbola

2 sided Hyperbola

$$\frac{4(x'')^2}{\delta^2} - \frac{4(y'')^2}{4D^2 - \delta^2} - \frac{4(z'')^2}{4D^2 - \delta^2} = 1$$

1 sided Hyperbola (solve for  $x''$ )

$$x'' = \delta \sqrt{\frac{1}{4} + \frac{(y'')^2}{4D^2 - \delta^2} + \frac{(z'')^2}{4D^2 - \delta^2}}$$

where

$\delta$  is the distance difference

$D$  is half the distance between stations

Choose the Hyperbola side based on the sign of the distance difference.

$$|\delta| = 5(\sqrt{3} - 1) = -3.66$$

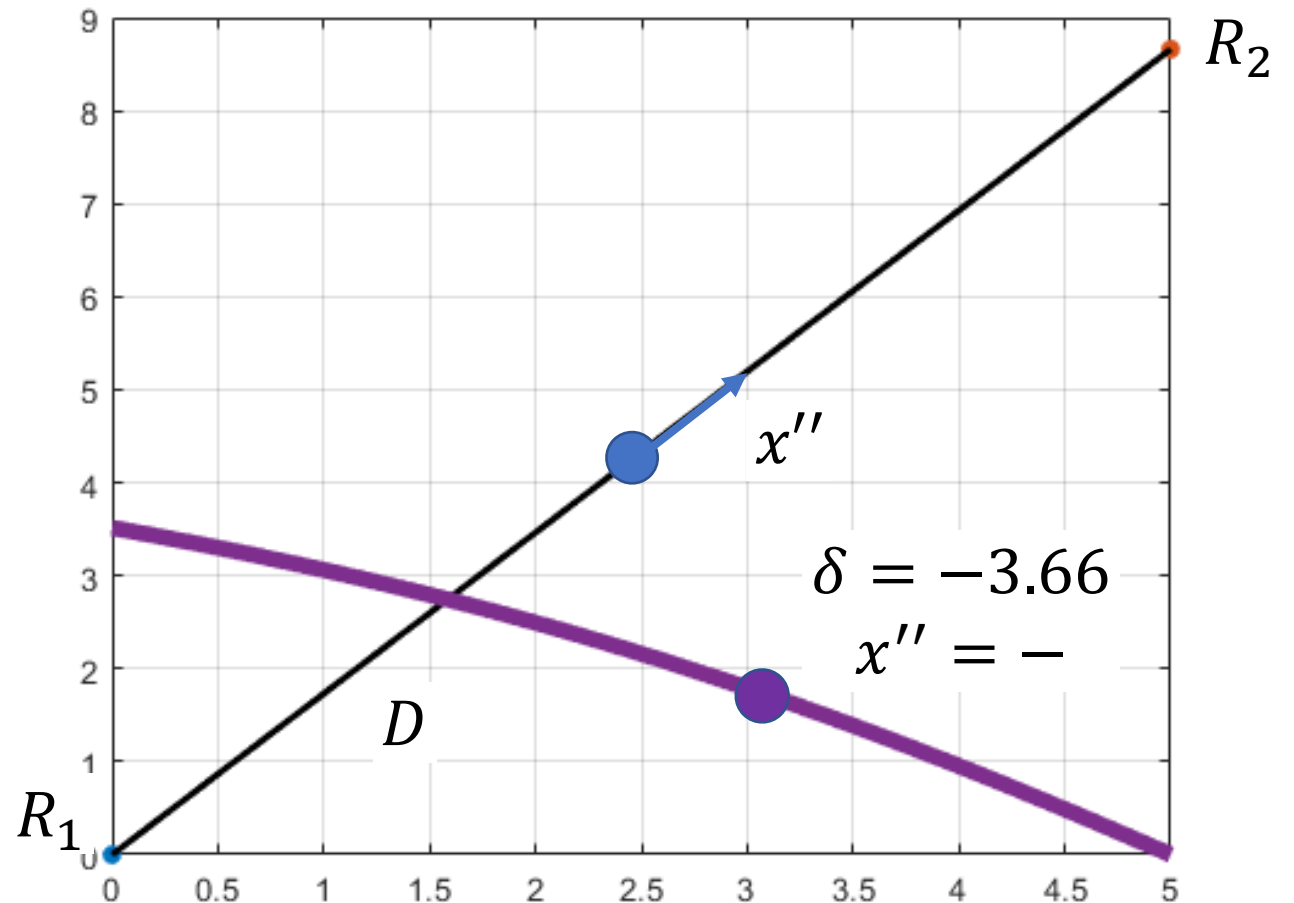
The sign of  $x''$  is wholly dictated by sign of delta.

$$x'' = \delta \sqrt{\frac{1}{4} + \frac{(y'')^2}{4D^2 - \delta^2} + \frac{(z'')^2}{4D^2 - \delta^2}}$$

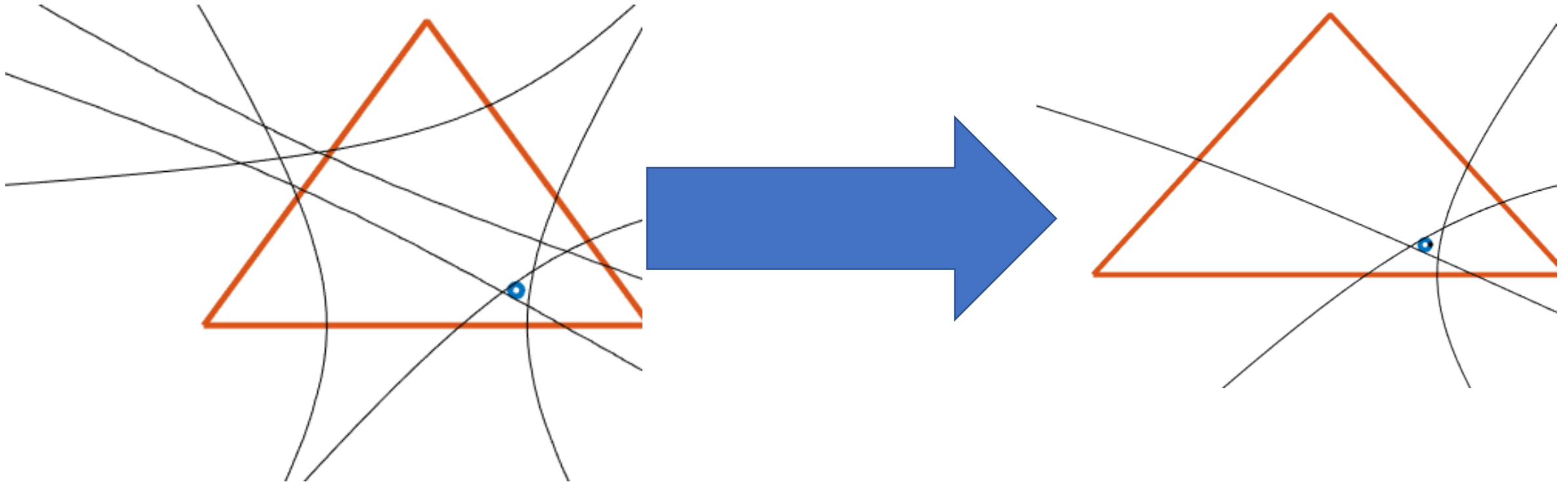
where

$\delta$  is the distance difference

$D$  is half the distance between stations



# Solution is much less ambiguous with 1 sided Hyperbolas



But how many hyperbolas are created from  $n$  number of stations?  
How do we find the “best solution”?

TDoA data creates  $\frac{n(n-1)(n-2)}{6}$  hyperboloids.

Recall the TDoA Data

$$\mathbf{R}_1 = [x_1, y_1, z_1]$$

$$\mathbf{R}_2 = [x_2, y_2, z_2]$$

$$\mathbf{R}_n = [x_n, y_n, z_n]$$

$$TD = \begin{matrix} & \begin{matrix} 1 & 2 & \cdots & n \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ \vdots \\ n \end{matrix} & \begin{bmatrix} 0 & t_{12} & \cdots & t_{1n} \\ 0 & 0 & \ddots & t_{2n} \\ \vdots & \vdots & \ddots & \ddots \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

For  $n = 3$ , we have 3 Hyperboloids.

For  $n = 4$ , we have 6 Hyperboloids.

Number of unique Hyperboloids.

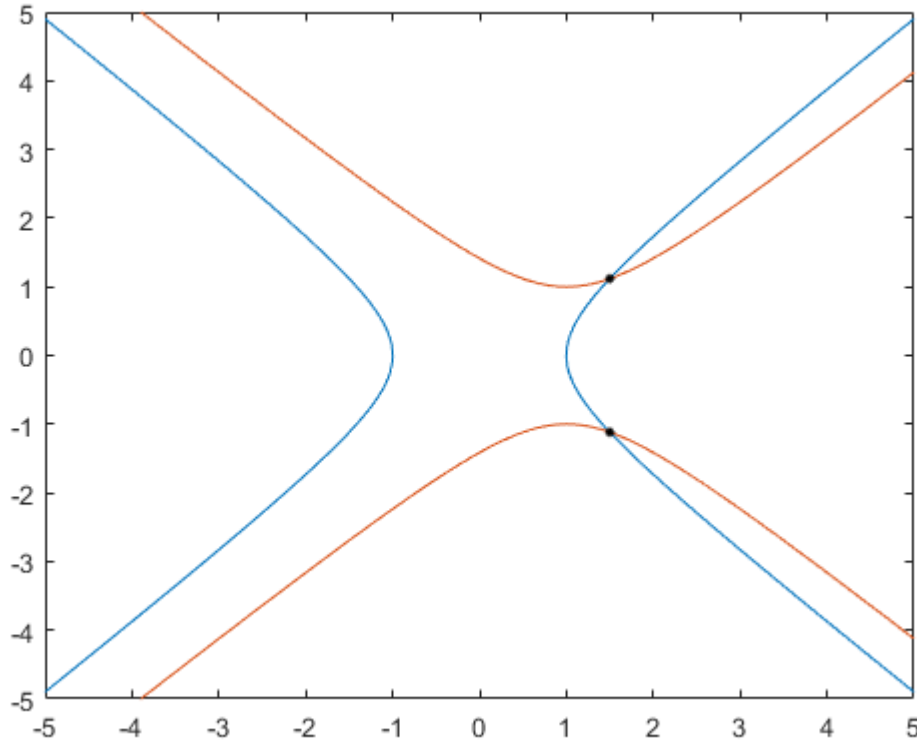
Given  $n = 3$ , we have

$$H_{12}, H_{13}, H_{23}$$

Only 2 of these hyperboloids are unique.

For  $n = 4$ , we have 3 unique Hyperboloids.

# Intersecting Hyperbolas in 2D.



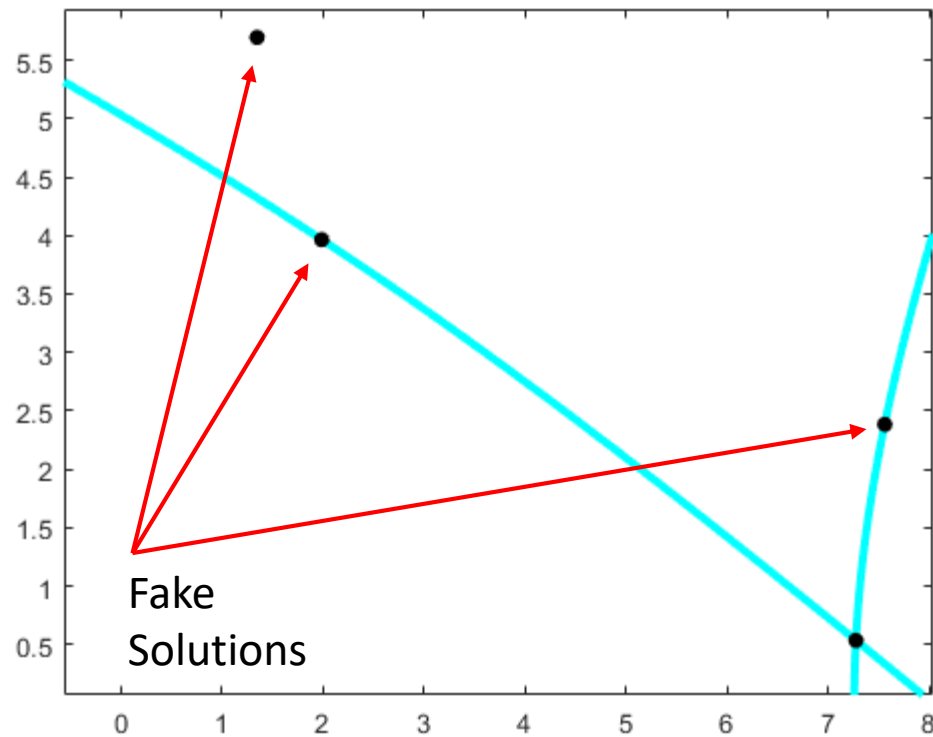
$$H_A = \frac{4(\mathbf{r}_A \cdot \mathbf{RM}_A(:, 1))^2}{\delta_A^2} - \frac{4(\mathbf{r}_A \cdot \mathbf{RM}_A(:, 2))^2}{4D_A^2 - \delta_A^2} = 1$$

$$H_B = \frac{4(\mathbf{r}_B \cdot \mathbf{RM}_B(:, 1))^2}{\delta_B^2} - \frac{4(\mathbf{r}_B \cdot \mathbf{RM}_B(:, 2))^2}{4D_B^2 - \delta_B^2} = 1$$

Intersection is  $H_A - H_B = 0$

- This works for both one-sided and two-sided Hyperbolas.
- For one-sided Hyperbolas, must remove fake solutions by plugging in all solutions back into  $H_A$  and  $H_B$  to verify they are indeed on the Hyperbola.

# Removing the Fake Solutions for One-sided Hyperbolas.



$$\mathbf{r}_A \cdot \mathbf{RM}_A(:, 1) = \delta_A \sqrt{\frac{1}{4} + \frac{(\mathbf{r}_A \cdot \mathbf{RM}_A(:, 2))^2}{4D_A^2 - \delta_A^2}}$$

$$\mathbf{r}_B \cdot \mathbf{RM}_B(:, 1) = \delta_B \sqrt{\frac{1}{4} + \frac{(\mathbf{r}_B \cdot \mathbf{RM}_B(:, 2))^2}{4D_B^2 - \delta_B^2}}$$

Intersection is  $H_A - H_B = 0$

- For one-sided Hyperbolas, must remove fake solutions by plugging in all solutions back into  $H_A$  and  $H_B$  to verify they are indeed on the Hyperbola.
- Fake solutions come from squaring both sides when going to solve the one-sided equation.

# The one-sided method breaks down when the sign of the Time Difference is questionable.

- The one-sided method requires signed time differences.
- Consider the following time differences:

$$(A) \ TD = 1.2\mu s \pm 0.1\mu s$$

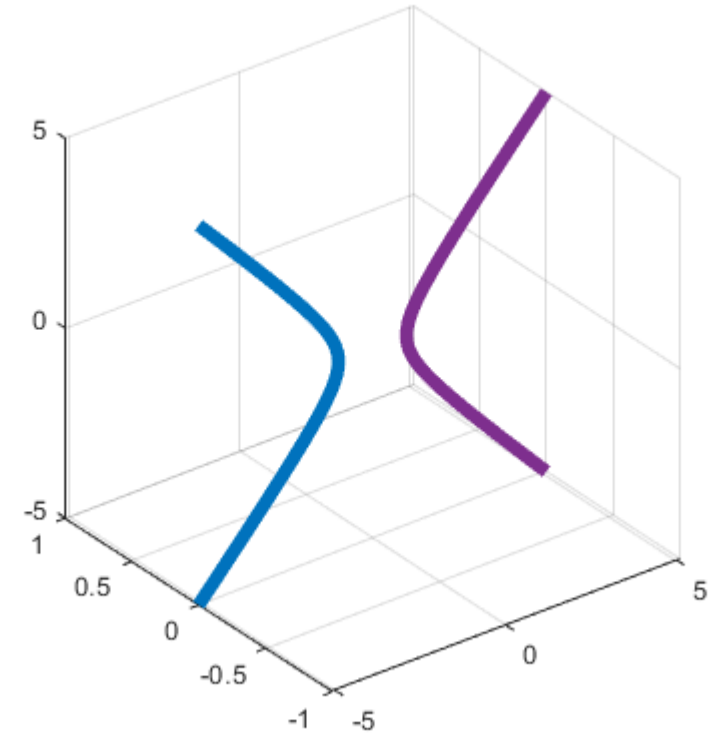
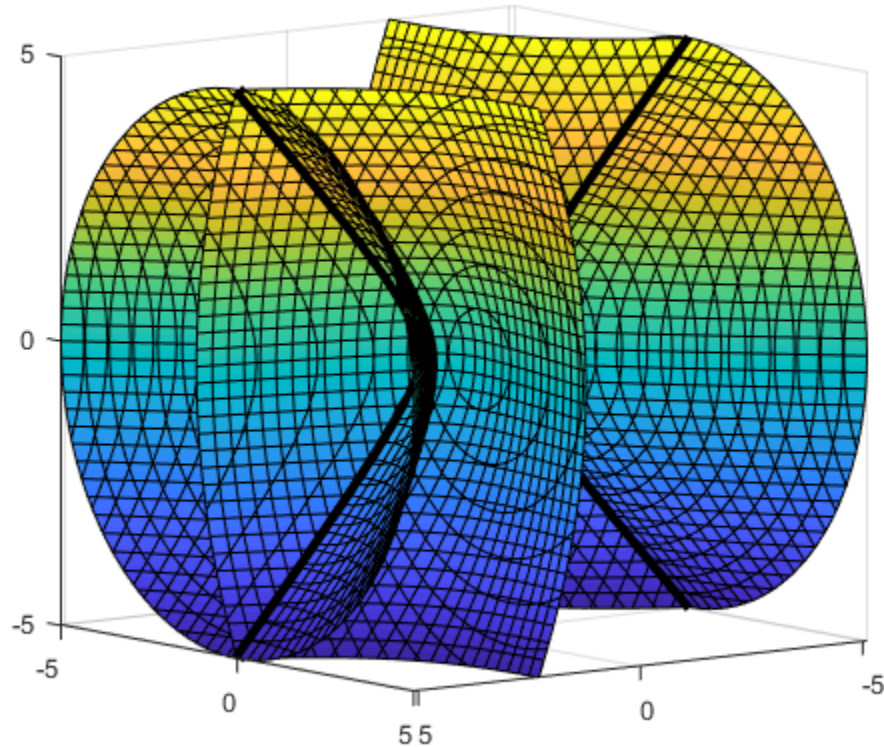
$$(B) \ TD = 0.05\mu s \pm 0.1\mu s$$

- The sign of TD is unambiguous in case (A).
- The sign of TD is ambiguous in case (B) with respect to its uncertainty value.

*Time differences approach zero as the emitter goes directly in between two receivers. TD Error becomes more significant at these points and the one-sided method may begin breaking down.*



# Intersecting 2 Hyperboloids results in a Hyperbola.



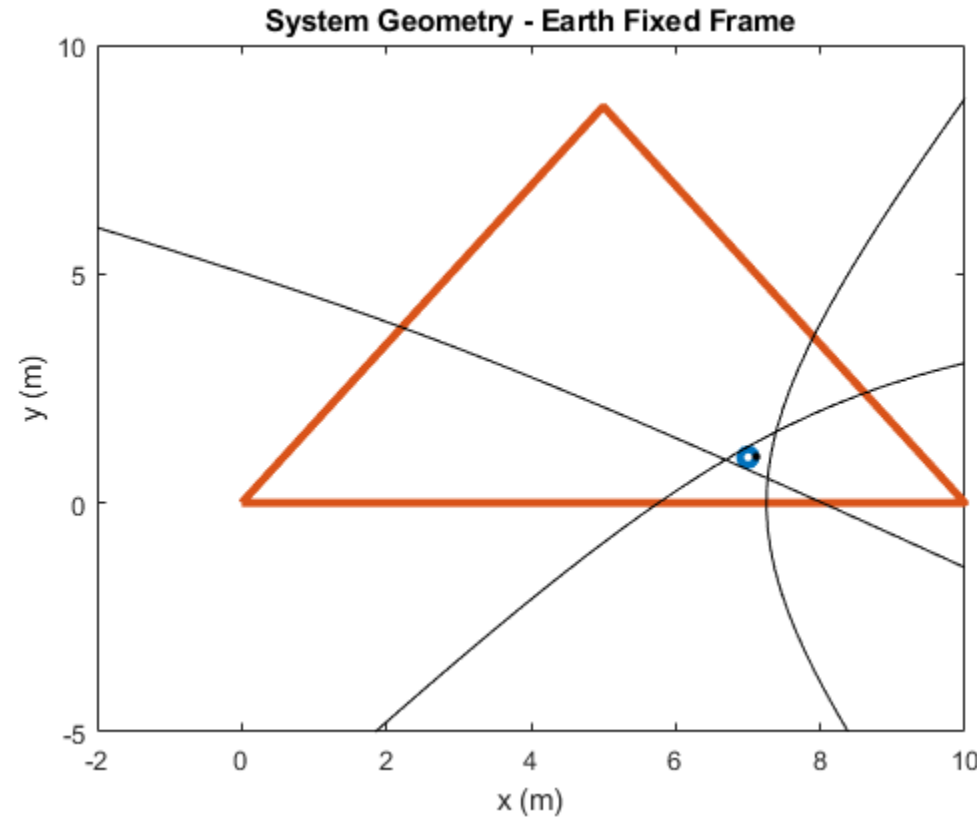
If the Hyperboloids are one-sided, the result is a one-sided Hyperbola.  
Still working on whether we can mathematically derive this hyperbola.

# Finding Best Solution

By Anthony Iannuzzi

# For one-sided Hyperbolas, finding the best solution is the average of 3 points.

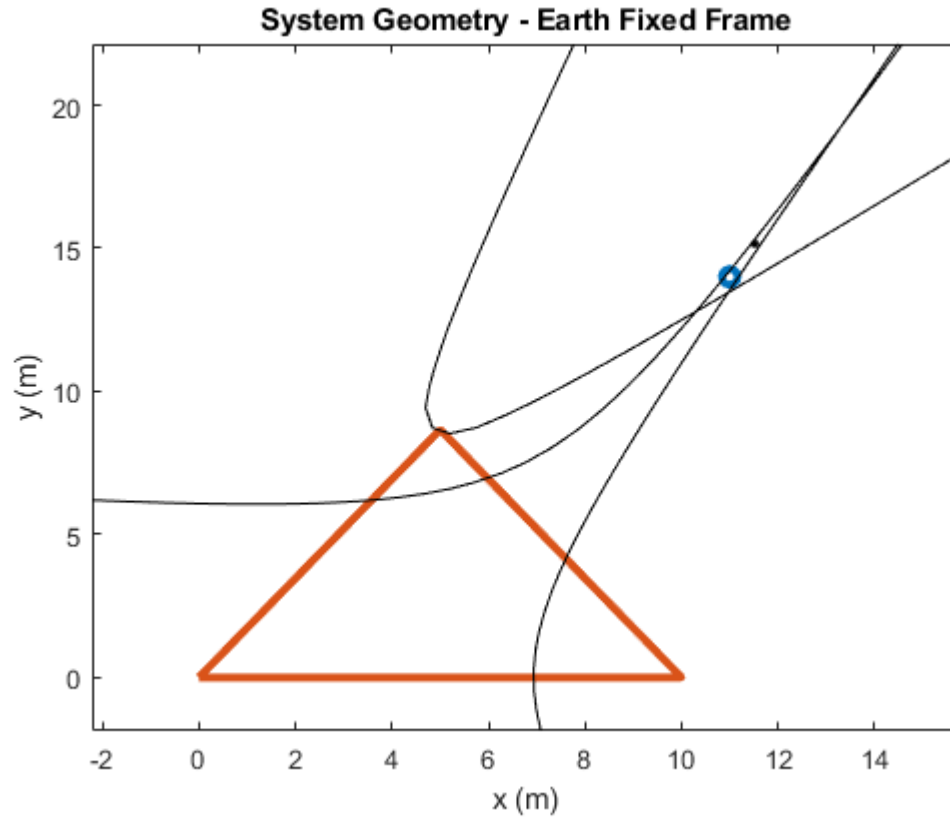
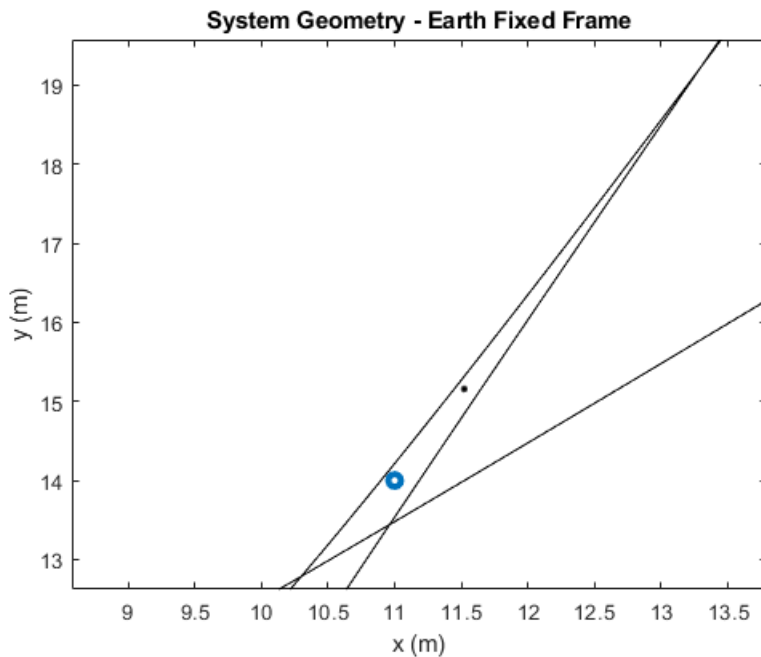
Three one-sided hyperbolas intersect at 3 points only.



- The blue circle is the correct answer.
- The black dot in the center of the enclosed triangular shape is the "best solution".

# TDoA in 2D is very sensitive to errors outside its “triangle of receivers”.

TDoA error generally looks like this:



- The blue circle is the correct answer.
- The black dot in the center of the enclosed triangular shape is the “best solution”.

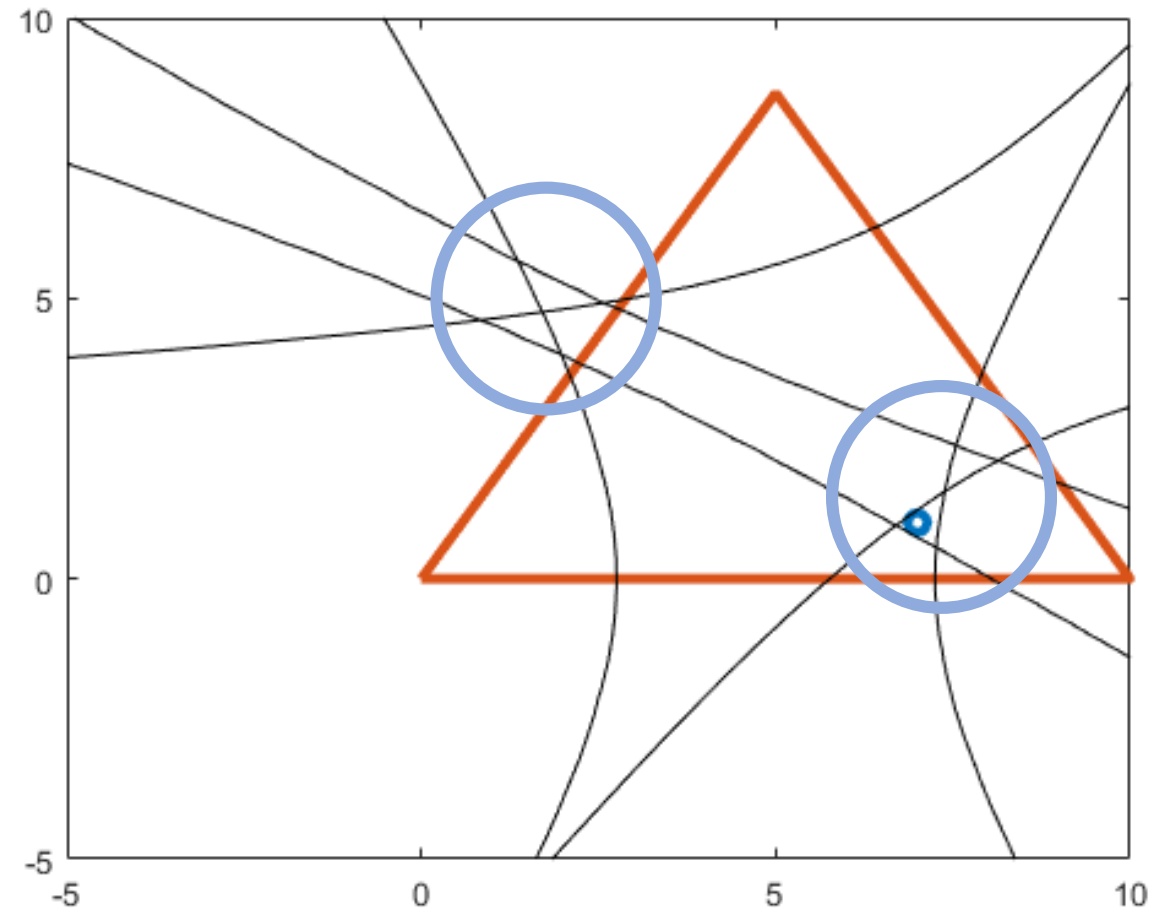
Position error is large, but the direction is still accurate.

# Finding the best solution for two-sided hyperbolas is significantly harder.

- Two-sided hyperbolas intersect in 12 different locations.
- Which cluster is the best solution?
- The one with the smallest area?

Approach:

Find all clustered locations by intersecting the hyperbolas 2 at a time.



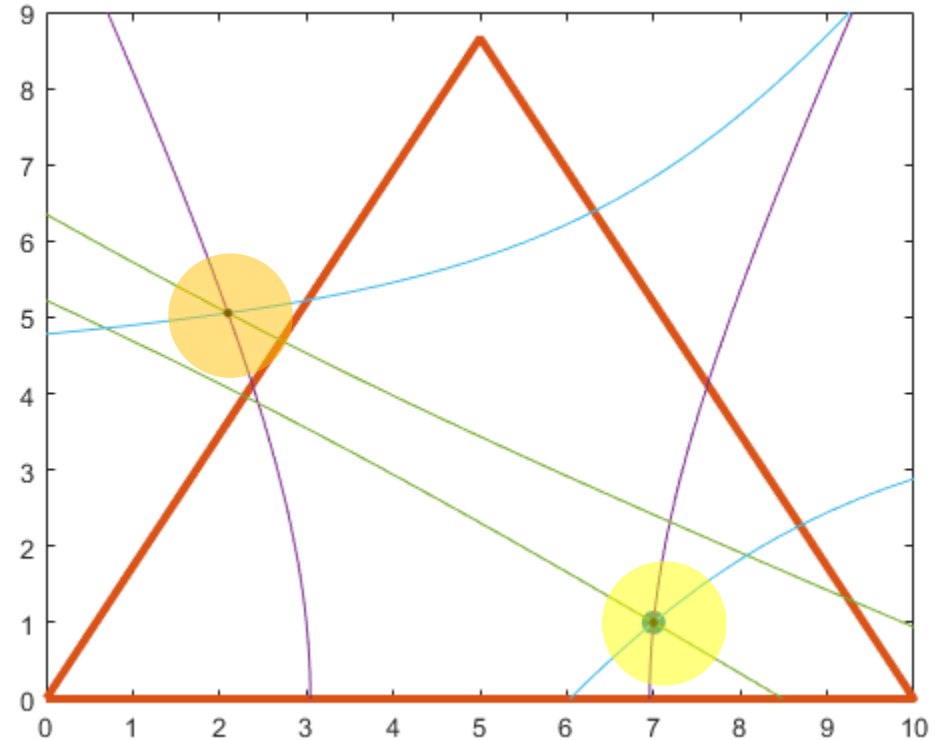
Example with no error: 2 clusters are close, but they still form a triangle.

1	7.0001	0.9998
	7.1886	2.3197
	2.4457	3.8727
	2.0964	5.0562

2	1.9290	-5.5781
	7.0000	1.0001
	2.0963	5.0564
	9.3175	9.0656

3	6.9888	0.9970
	8.0753	1.8799
	0.6839	4.8581
	2.0965	5.0565

- Numerical precision prevents the 3 points from lining up *exactly*.
- Intersecting 3 hyperbolas results in no solution.

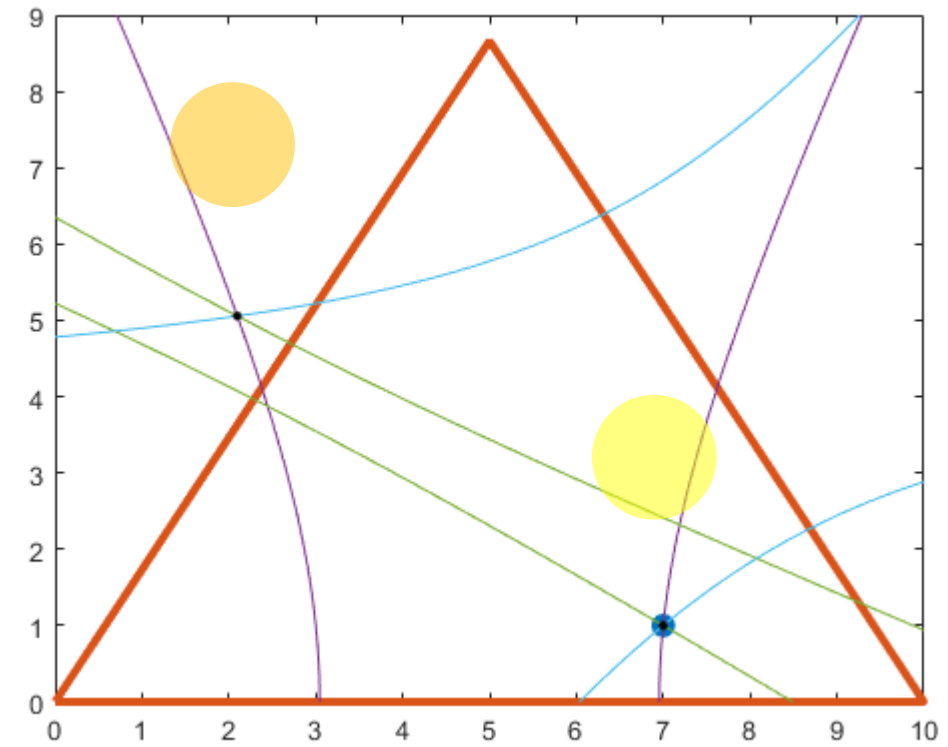


$$H_A - H_B = 0$$

Where  $H_A, H_B$  are equations of Hyperboloids.

First sort the data with respect to x then y.  
Differentiate the data. Single out the data below 0.1 difference.

	x sorted	y sorted	difference	difference
1	0.6839	4.8581	1.2451	-10.4362
2	1.9290	-5.5781	0.1673	10.6345
3	2.0963	5.0564	0.0001	-0.0002
4	2.0964	5.0562	0.0001	0.0003
5	2.0965	5.0565	0.3492	-1.1838
6	2.4457	3.8727	4.5431	-2.8757
7	6.9888	0.9970	0.0112	0.0031
8	7.0000	1.0001	0.0001	-0.0003
9	7.0001	0.9998	0.1885	1.3199
10	7.1886	2.3197	0.8866	-0.4399
11	8.0753	1.8799	1.2423	7.1858
12	9.3175	9.0656		





Decide how many solutions there are by looking for which singled out points are clustered.

Index	Diff - 1	x sorted	y sorted	difference	difference
1		0.6839	4.8581	1.2451	-10.4362
2		1.9290	-5.5781	0.1673	10.6345
3	0	2.0963	5.0564	0.0001	-0.0002
4	2	2.0964	5.0562	0.0001	0.0003
5		2.0965	5.0565	0.3492	-1.1838
6		2.4457	3.8727	4.5431	-2.8757
7	0	6.9888	0.9970	0.0112	0.0031
8		7.0000	1.0001	0.0001	-0.0003
9	0	7.0001	0.9998	0.1885	1.3199
10		7.1886	2.3197	0.8866	-0.4399
11		8.0753	1.8799	1.2423	7.1858
12		9.3175	9.0656		

- Collect the indices of all singled out differences.

$$\text{Indices} = [3, 4, 7, 8]$$

- Take the differences of these differences.

$$\text{diff}(\text{Indices}) = [1, 3, 1]$$

$$\text{diff}(\text{Indices}) - 1 = [0, 2, 0]$$

- The number of solutions is equal to the number of non-zero terms.

The critical points to collect are the beginning, end, and all non-zero terms in Diff-1.

Index	Diff - 1	x sorted	y sorted	difference	difference
1		0.6839	4.8581	1.2451	-10.4362
2		1.9290	-5.5781	0.1673	10.6345
3	0	2.0963	5.0564	0.0001	-0.0002
4	2	2.0964	5.0562	0.0001	0.0003
5		2.0965	5.0565	0.3492	-1.1838
6		2.4457	3.8727	4.5431	-2.8757
7	0	6.9888	0.9970	0.0112	0.0031
8		7.0000	1.0001	0.0001	-0.0003
9	0	7.0001	0.9998	0.1885	1.3199
10		7.1886	2.3197	0.8866	-0.4399
11		8.0753	1.8799	1.2423	7.1858
12		9.3175	9.0656		

$$\text{diff}(\text{Indices}) - 1 = [0, 2, 0]$$

- The number of solutions is equal to the number of non-zero terms.
- The critical points identify switching between solutions.
- The first critical point is always 0.
- The last critical point is the number of singled out points. (4 in this ex.)

$$\text{criticalPoints} = [0, 2, 4]$$

# Use critical points to identify the indices of the solution

Index	Diff - 1	x sorted	y sorted	difference	difference
1		0.6839	4.8581	1.2451	-10.4362
2		1.9290	-5.5781	0.1673	10.6345
3	0	2.0963	5.0564	0.0001	-0.0002
4	2	2.0964	5.0562	0.0001	0.0003
5		2.0965	5.0565	0.3492	-1.1838
6		2.4457	3.8727	4.5431	-2.8757
7	0	6.9888	0.9970	0.0112	0.0031
8		7.0000	1.0001	0.0001	-0.0003
9		7.0001	0.9998	0.1885	1.3199
10		7.1886	2.3197	0.8866	-0.4399
11		8.0753	1.8799	1.2423	7.1858
12		9.3175	9.0656		

*Indices* = [3,4,7,8]

*criticalPoints* = [0,2,4]

We now iterate through critical points, collecting all data between the previous and current critical point:

*Indices*(*split*(*i*) + 1 : *split*(*i* + 1))

For the first solution we have

*Indices*(1 : 2) = [3,4]

For the second solution we have

*Indices*(3 : 4) = [7,8]

The last point in the cluster is the point directly above the identified ranges.

Index	Diff - 1	x sorted	y sorted	difference	difference
1		0.6839	4.8581	1.2451	-10.4362
2		1.9290	-5.5781	0.1673	10.6345
3	0	2.0963	5.0564	0.0001	-0.0002
4	2	2.0964	5.0562	0.0001	0.0003
5		2.0965	5.0565	0.3492	-1.1838
6		2.4457	3.8727	4.5431	-2.8757
7	0	6.9888	0.9970	0.0112	0.0031
8		7.0000	1.0001	0.0001	-0.0003
9		7.0001	0.9998	0.1885	1.3199
10		7.1886	2.3197	0.8866	-0.4399
11		8.0753	1.8799	1.2423	7.1858
12		9.3175	9.0656		

For the first solution we have

$$\text{Indices}(1:2) = [3,4]$$

For the second solution we have

$$\text{Indices}(3:4) = [7,8]$$

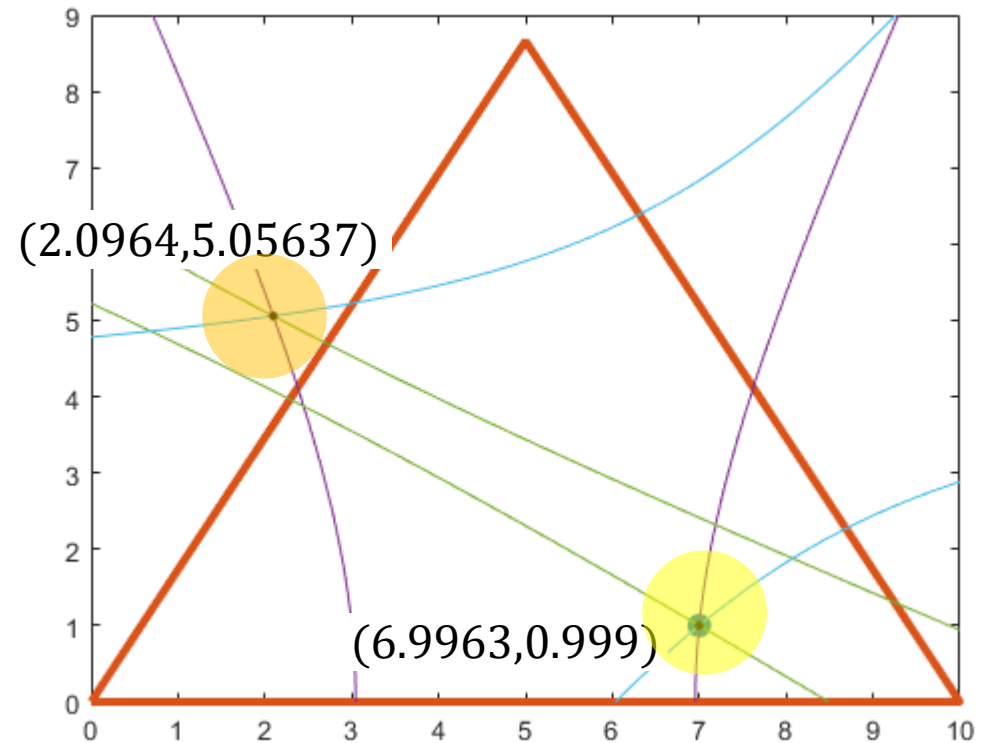
The diff function lags one element behind. We add the next consecutive element to each solution:

$$\text{soln1} = [3,4,5]$$

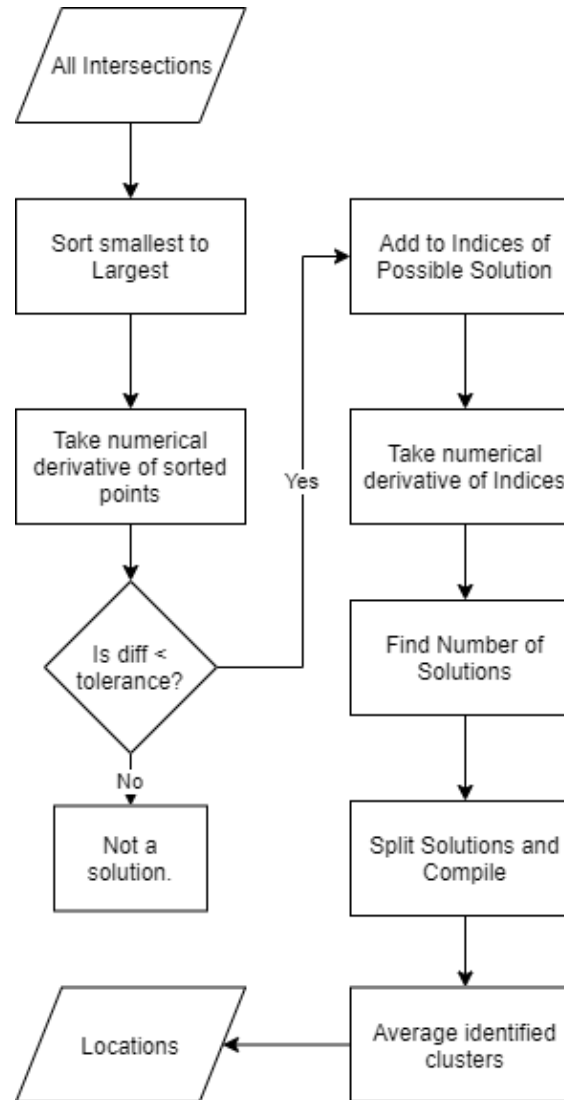
$$\text{soln2} = [7,8,9]$$

# Finally, average the solution together.

Index	Diff - 1	x sorted	y sorted	difference	difference
1		0.6839	4.8581	1.2451	-10.4362
2		1.9290	-5.5781	0.1673	10.6345
3	0	2.0963	5.0564	0.0001	-0.0002
4	2	2.0964	5.0562	0.0001	0.0003
5		2.0965	5.0565	0.3492	-1.1838
6		2.4457	3.8727	4.5431	-2.8757
7	0	6.9888	0.9970	0.0112	0.0031
8		7.0000	1.0001	0.0001	-0.0003
9		7.0001	0.9998	0.1885	1.3199
10		7.1886	2.3197	0.8866	-0.4399
11		8.0753	1.8799	1.2423	7.1858
12		9.3175	9.0656		

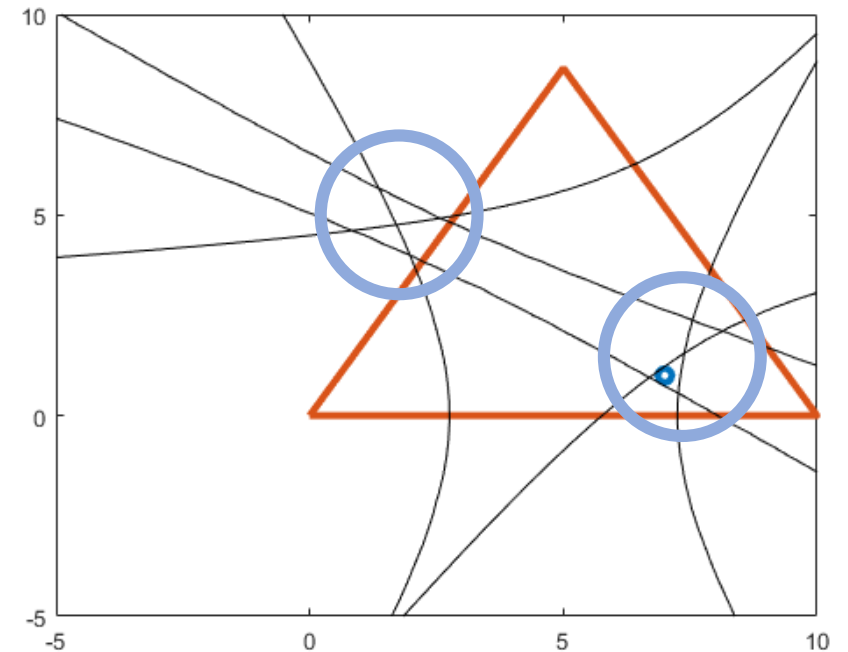


# Choose the “Best” Solution Flowchart



# Limitations to this algorithm

- This algorithm works well for 2 clusters of points.
- It does not work well when there are 4 clusters of points with roughly the same size:
- Averaging the points makes sense for a nequilateral triangle. It might not for an isosceles triangle.

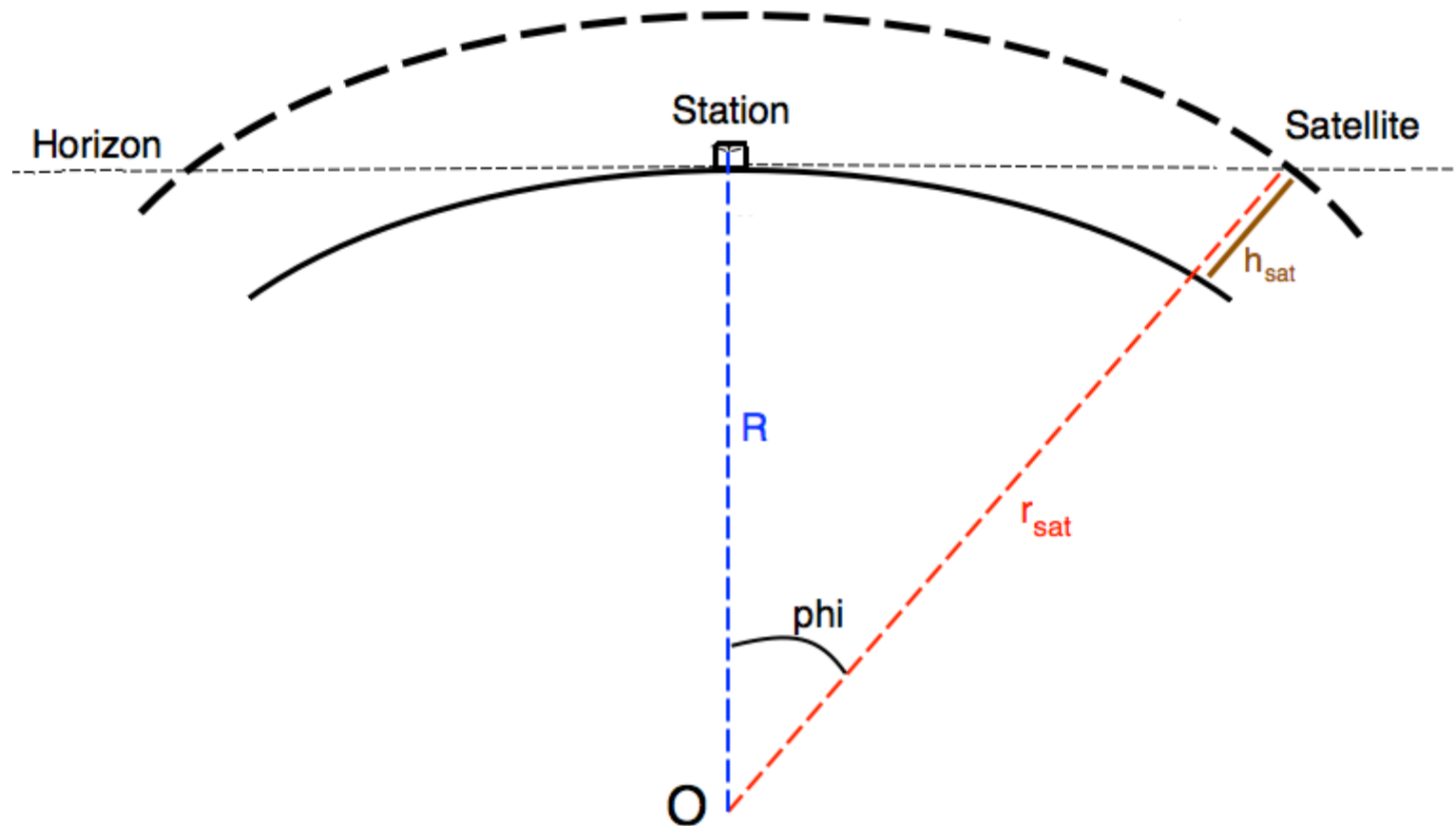




# The Satellite Problem

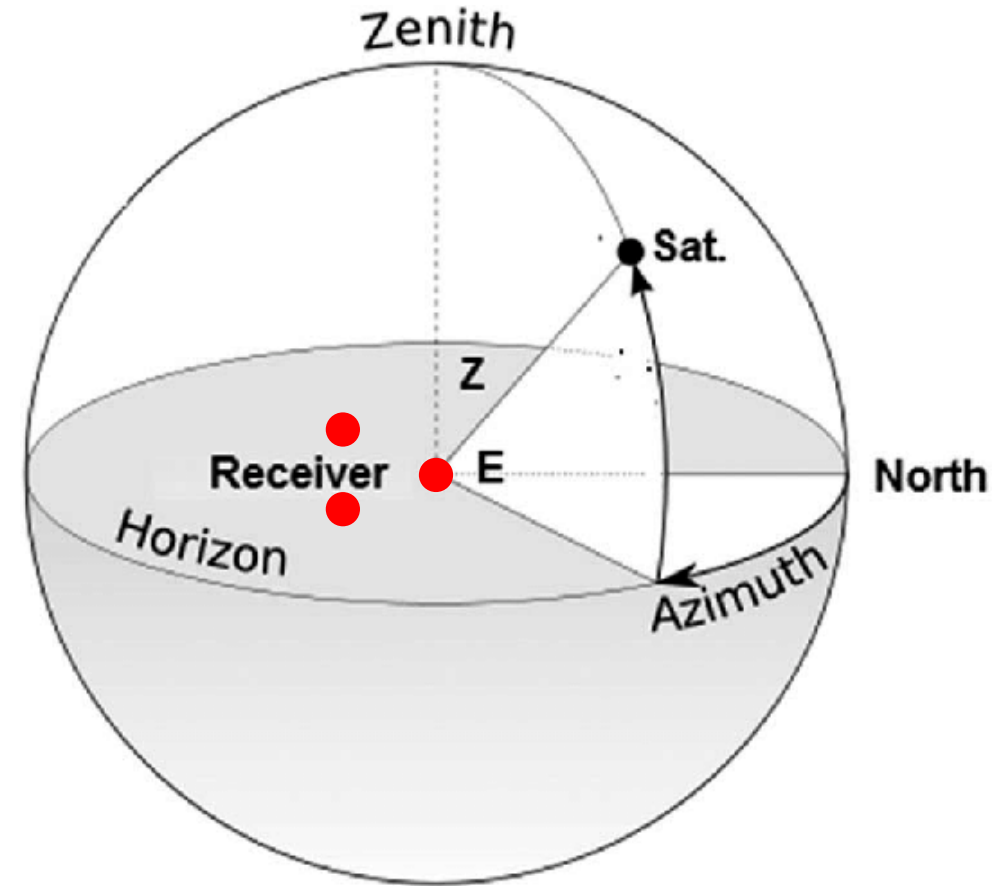
By Anthony Iannuzzi

The satellite is much closer to the station near elevation  $90^\circ$  than at lower elevations.



# The satellite is far away from the plane of the receivers except when elevation is $\sim 0^\circ$

- The satellite is far away from the receivers.
- For the case of elevation  $\sim 0^\circ$ , the satellite is far outside the “triangle”.
- For any other elevation, the satellite is on a different plane.



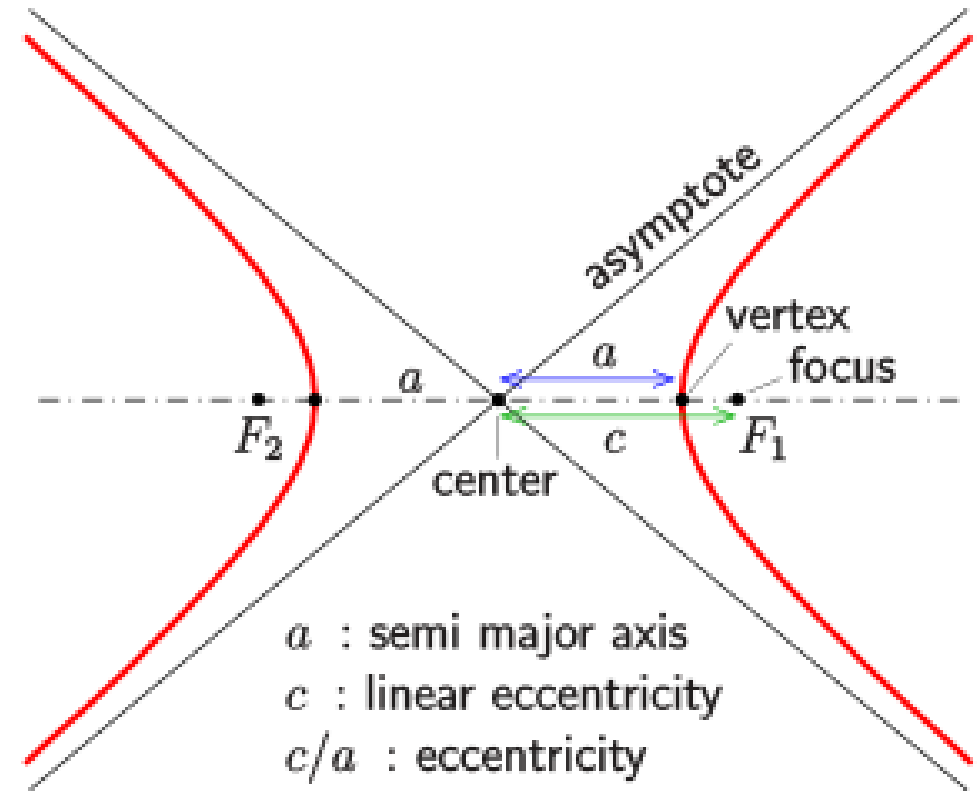
[https://www.researchgate.net/figure/Satellite-Azimuth-and-elevation-angle\\_fig1\\_334197029](https://www.researchgate.net/figure/Satellite-Azimuth-and-elevation-angle_fig1_334197029)

# Far away, hyperbolas look like cones.

- Hyperbolas approach a line as  $x$  or  $y$  approach infinity.
- A 2D one-sided hyperbola is:

$$x'' = \delta \sqrt{\frac{1}{4} + \frac{(y'')^2}{4D^2 - \delta^2}}$$

If  $y''$  is large, then the  $\frac{1}{4}$  can be neglected.



<https://en.wikipedia.org/wiki/Hyperbola>

Far away, hyperbolas look like cones.

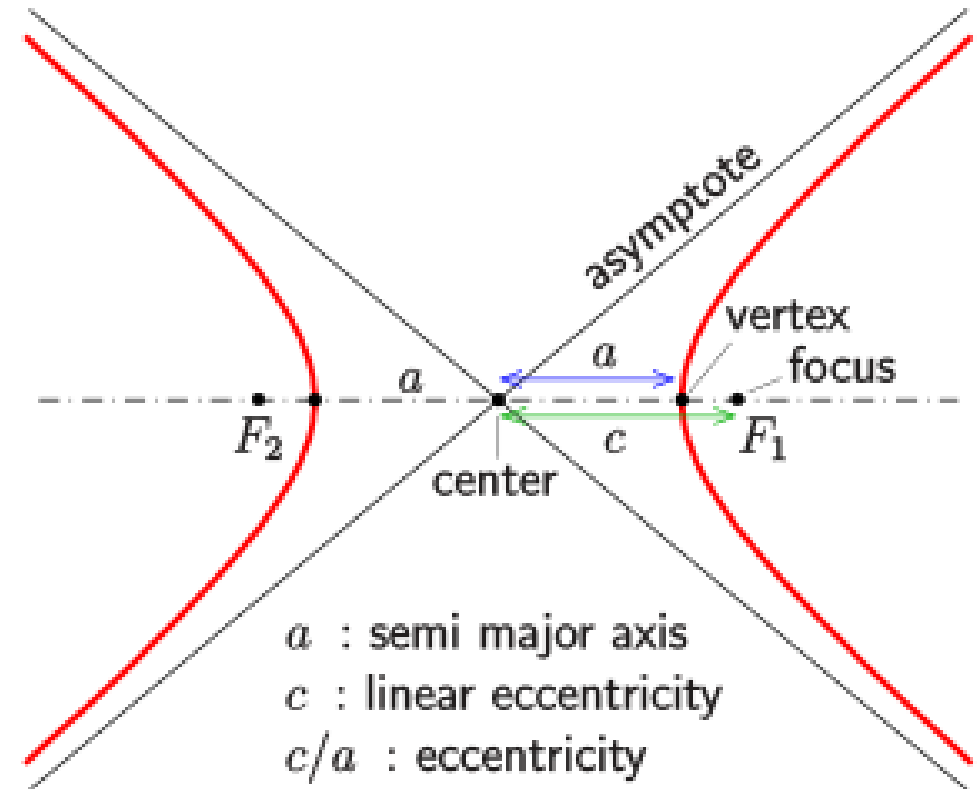
$$x'' = \delta \sqrt{\frac{1}{4} + \frac{(y'')^2}{4D^2 - \delta^2}}$$

If  $y''$  is large, then the  $\frac{1}{4}$  can be neglected.

$$x'' = \delta \sqrt{\frac{(y'')^2}{4D^2 - \delta^2}}$$

$$x'' = \delta |y''| \sqrt{\frac{1}{4D^2 - \delta^2}}$$

- This is the equation of the asymptotes.



<https://en.wikipedia.org/wiki/Hyperbola>

The line equation is

$$x = \frac{a}{b} |y|$$

Where  $b^2 = 4D^2 - \delta^2$

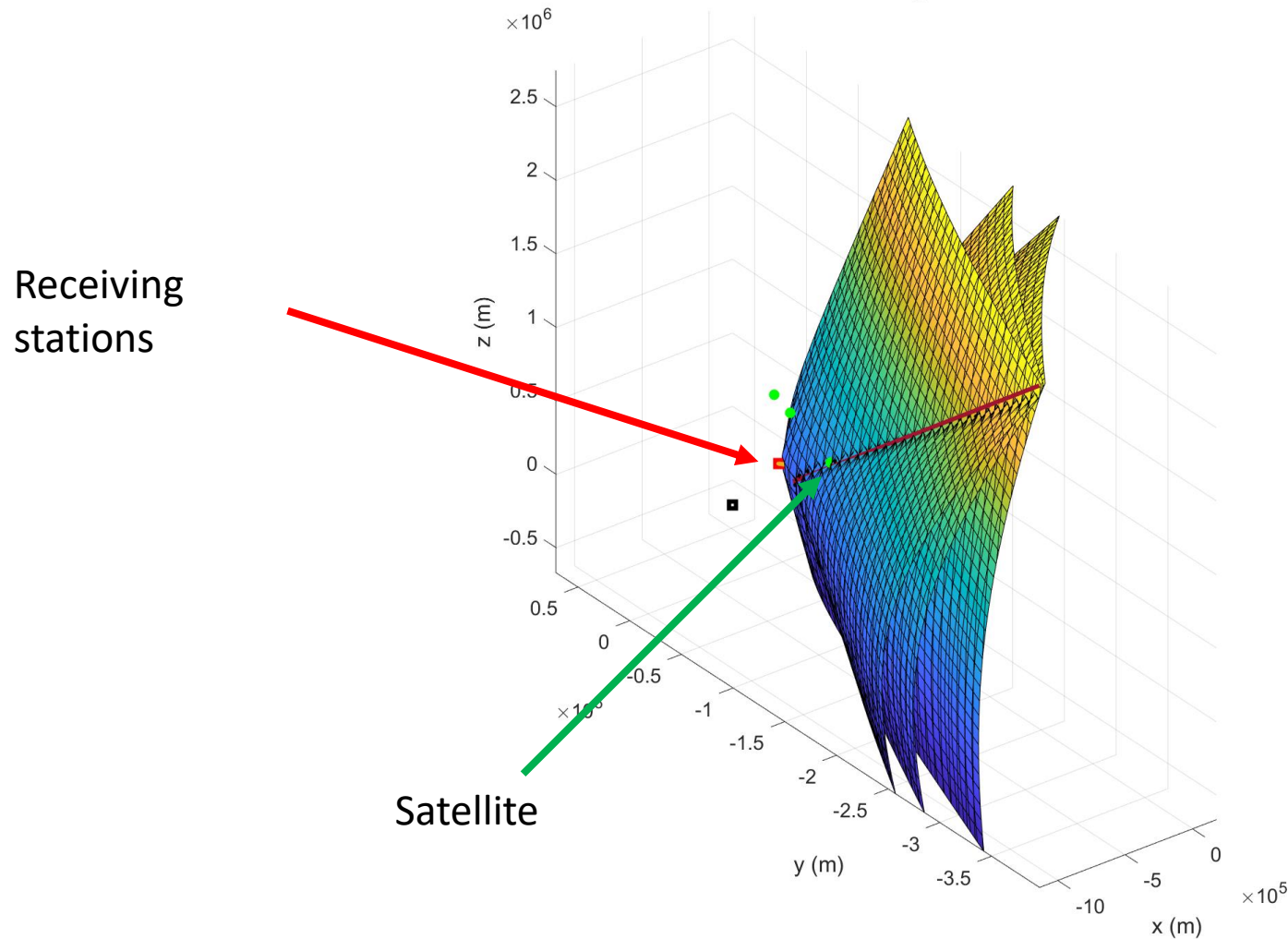
# The strategy for getting the satellite direction.

- 2 Hyperboloids intersect on a hyperbola.
- Far away from the foci, a hyperbola looks like a line.
- The direction of the satellite can be derived from a line.

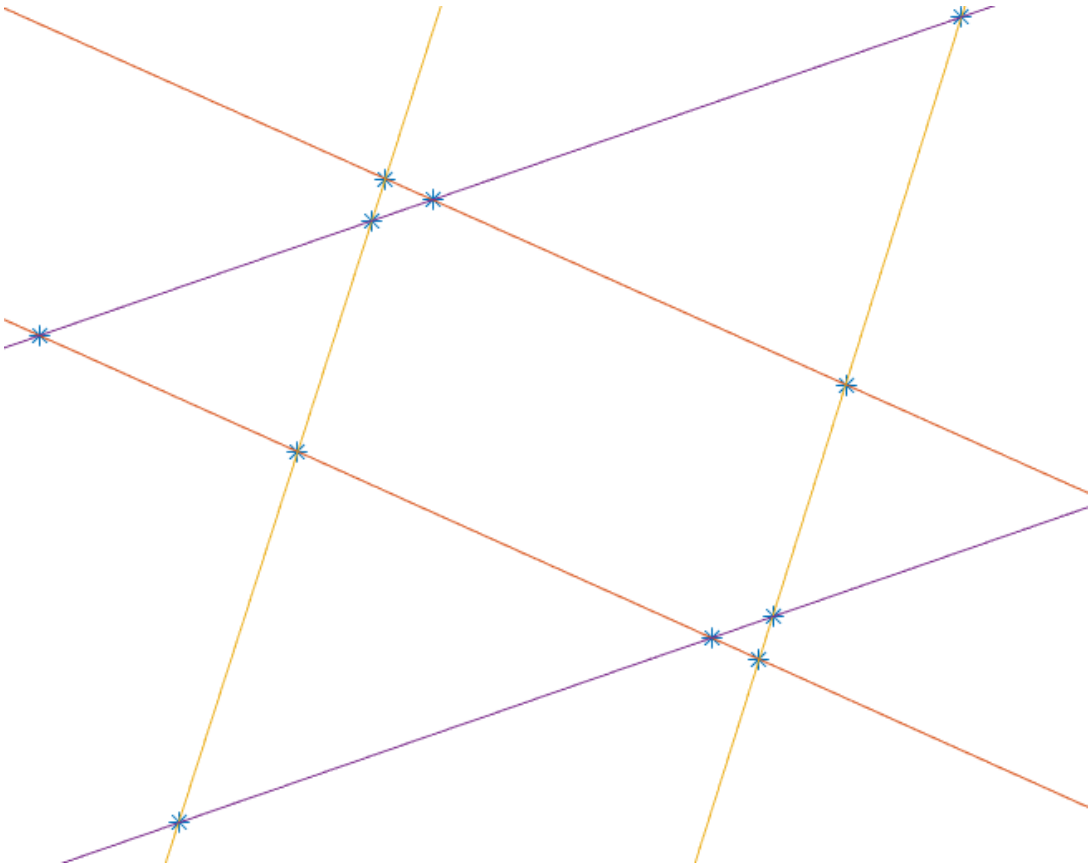
1. Draw the hyperboloids.
2. Solve multiple 2D problems on different planes.
3. Fit a line to the 2D solutions.

More on specifics in a later section.

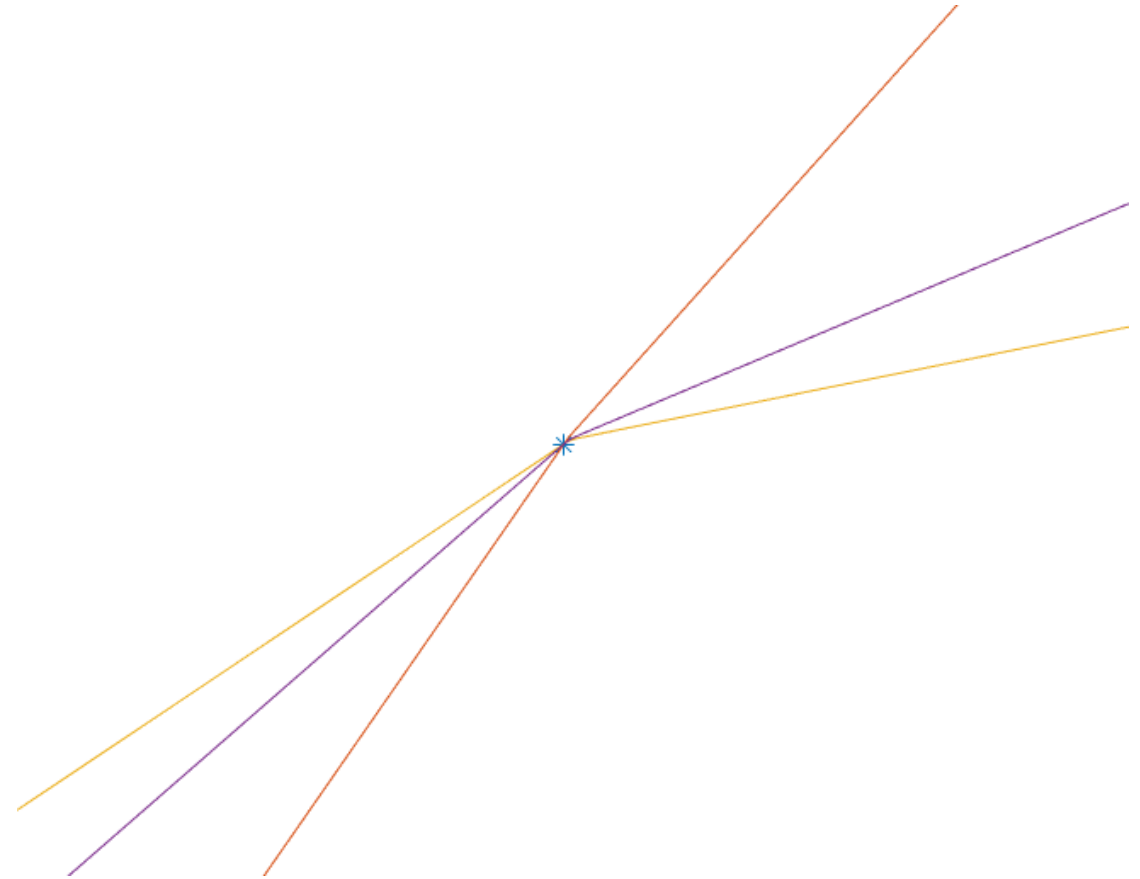
# What do the hyperboloids look like for the satellite problem?



# What does the localizing the satellite look like on a plane far away from the receivers?

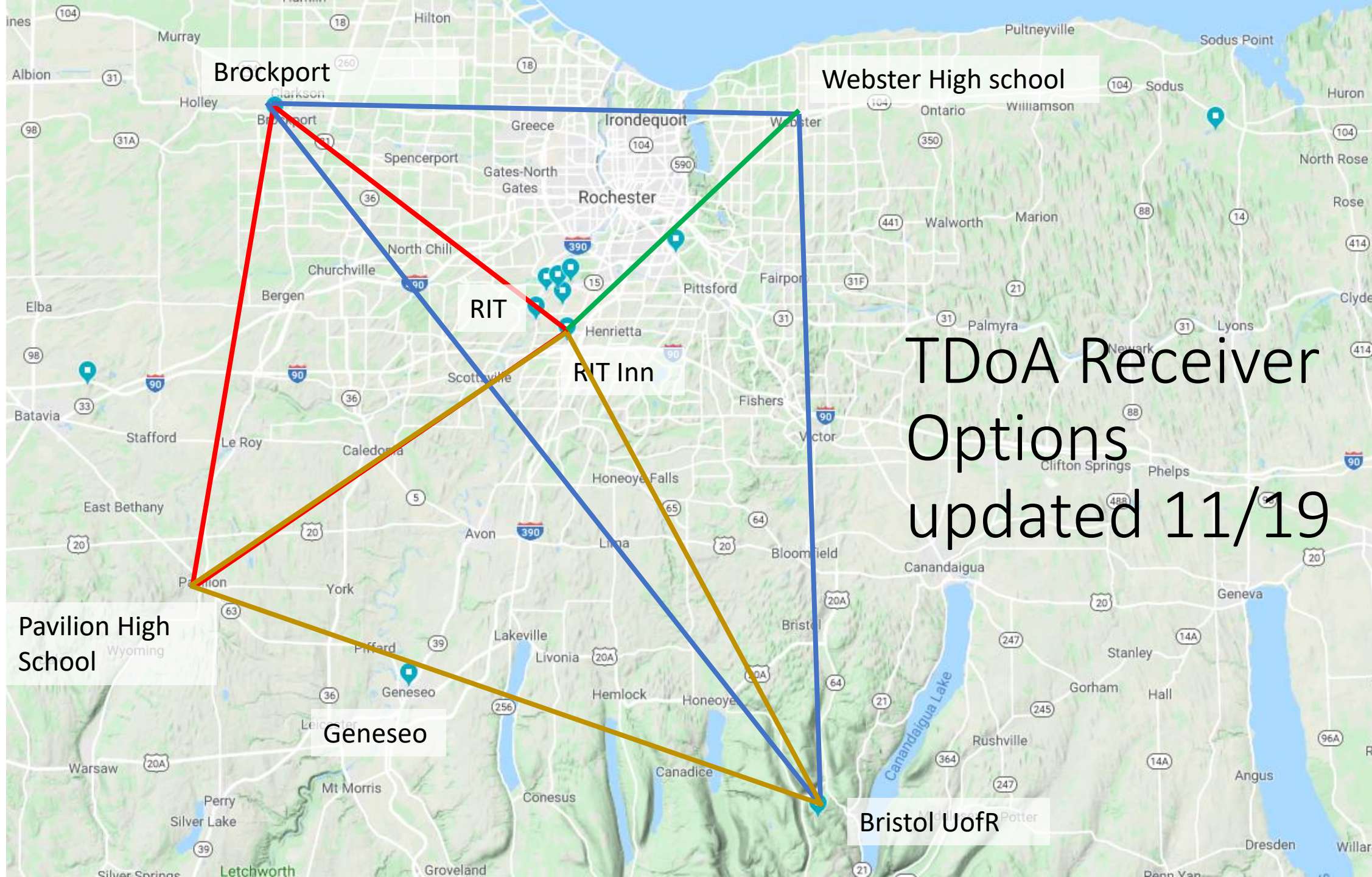


2-sided Hyperbolas



1-sided Hyperbolas





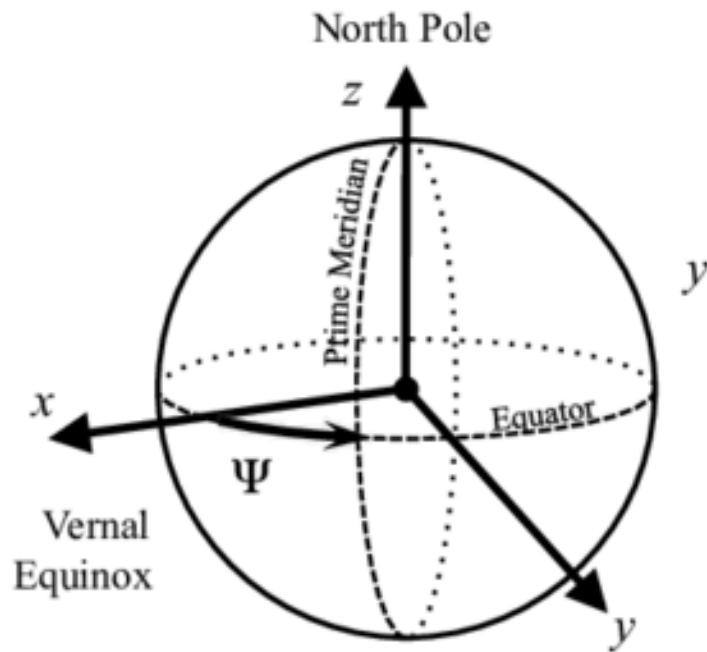
# Coordinate Transformations Overview.

- Receivers are usually in geodetic coordinates  
*latitude, longitude, altitude*
- Satellite coordinates are usually in orbital plane or an Inertial frame, like J2000.

$$x_f, y_f, z_f$$

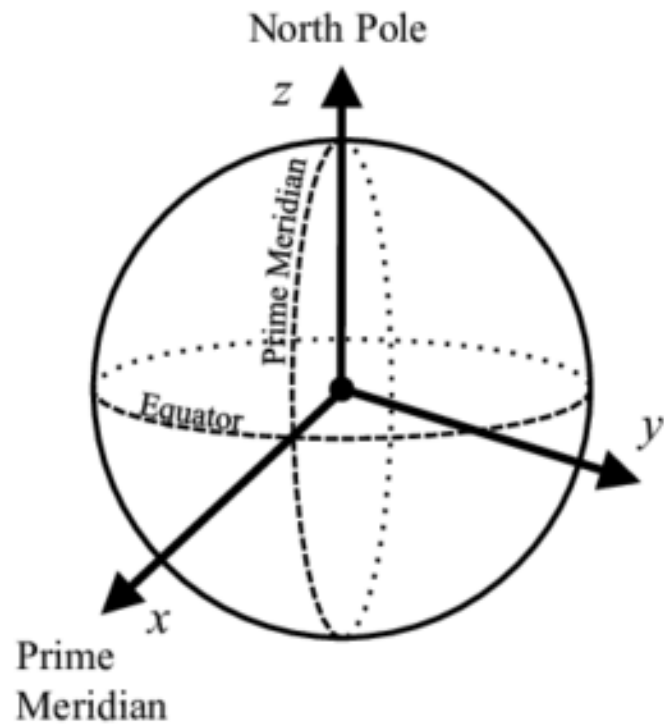
- To convert the receivers to x,y,z coordinates, the shape of the Earth must be taken into account.
- To convert the satellite to Earth fixed coordinates, the rotation rate and orientation of the Earth must be taken into account.

# The 3 Earth frames that are important.



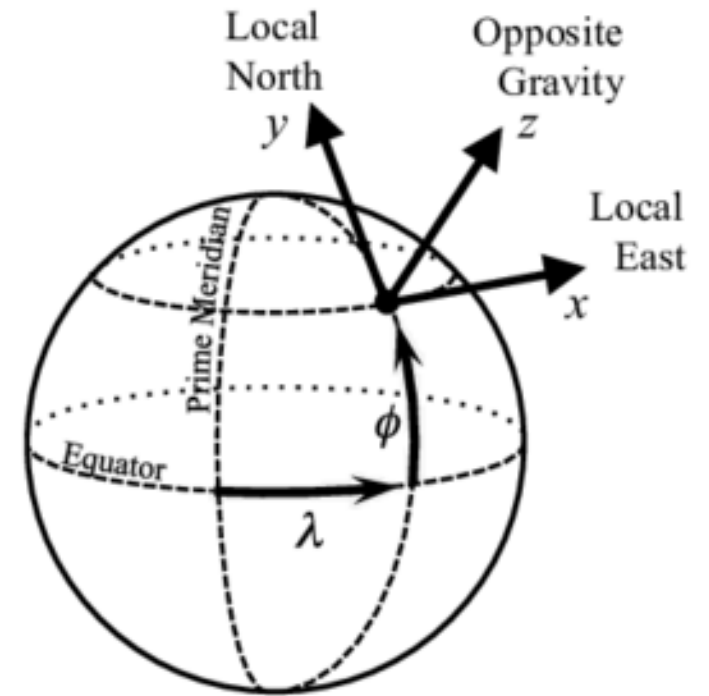
EARTH CENTRED INERTIAL:  $I$

Where the satellite is measured.



EARTH CENTRED FIXED:  $F$

Ground station coordinates



TOPOCENTRIC:  $T$

Satellite coordinates with respect to a station.

# Transforming from Geodetic to Earth Fixed

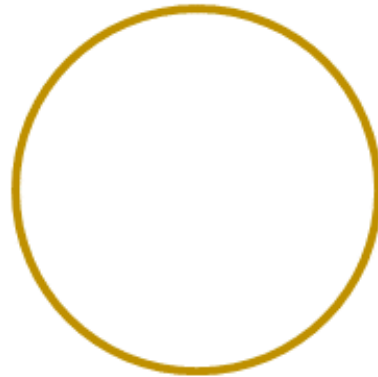
## Spherical Model

Analytical (Lat,Long)  $\rightarrow$  (x,y,z)

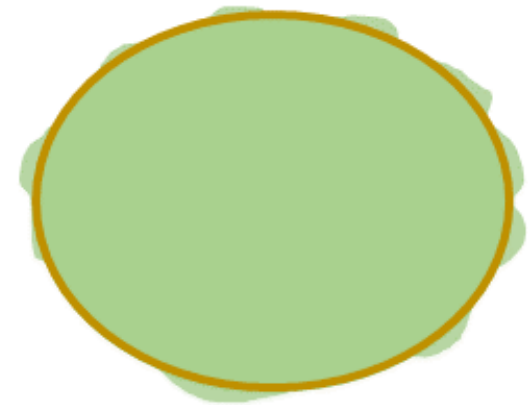
Planet earth



Spheres



WGS84



## WGS84 Model

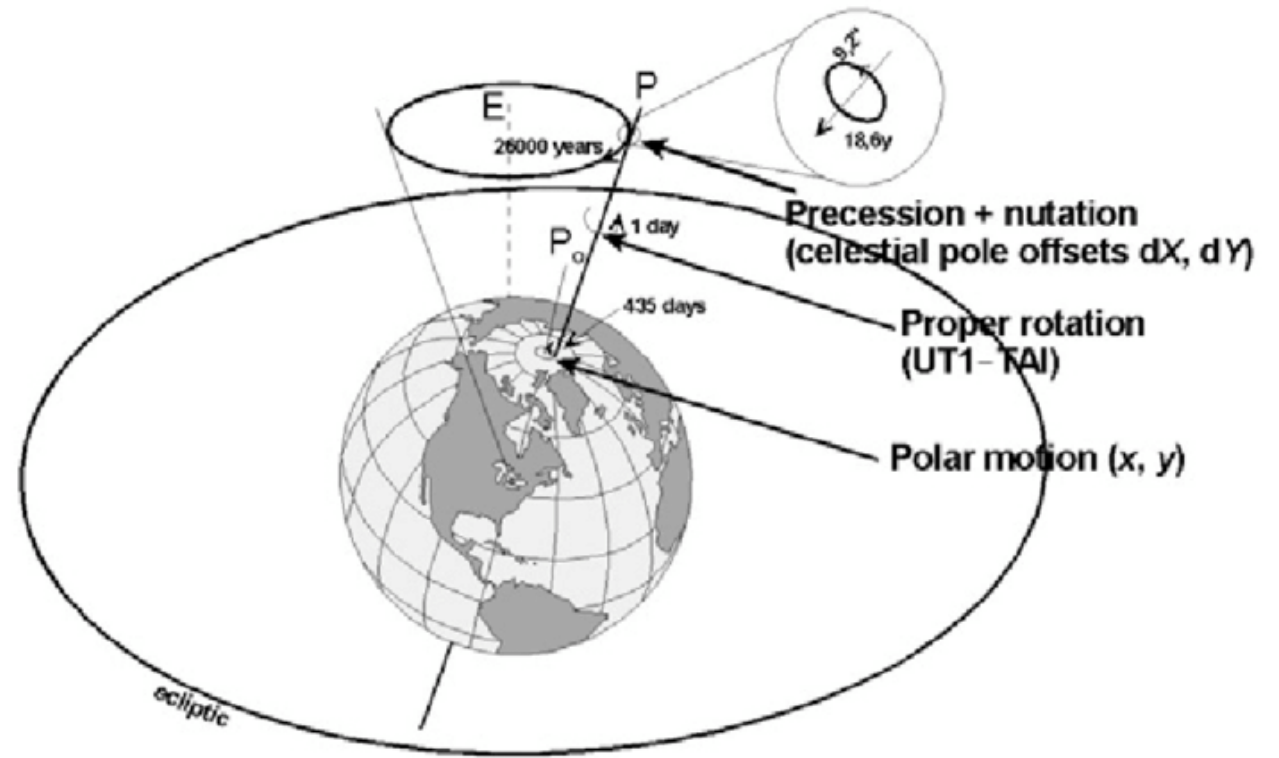
More accurate model.

Matlab and Orekit Built-in functions available. Analytical solution is more



# Transforming from Earth Inertial to Earth Fixed.

- The Earth rotates once in about 24 hours
- It undergoes one precession period in 26,000 years.
- It undergoes one nutation period every 18.6 years
- The poles wander one full period in 435 days.



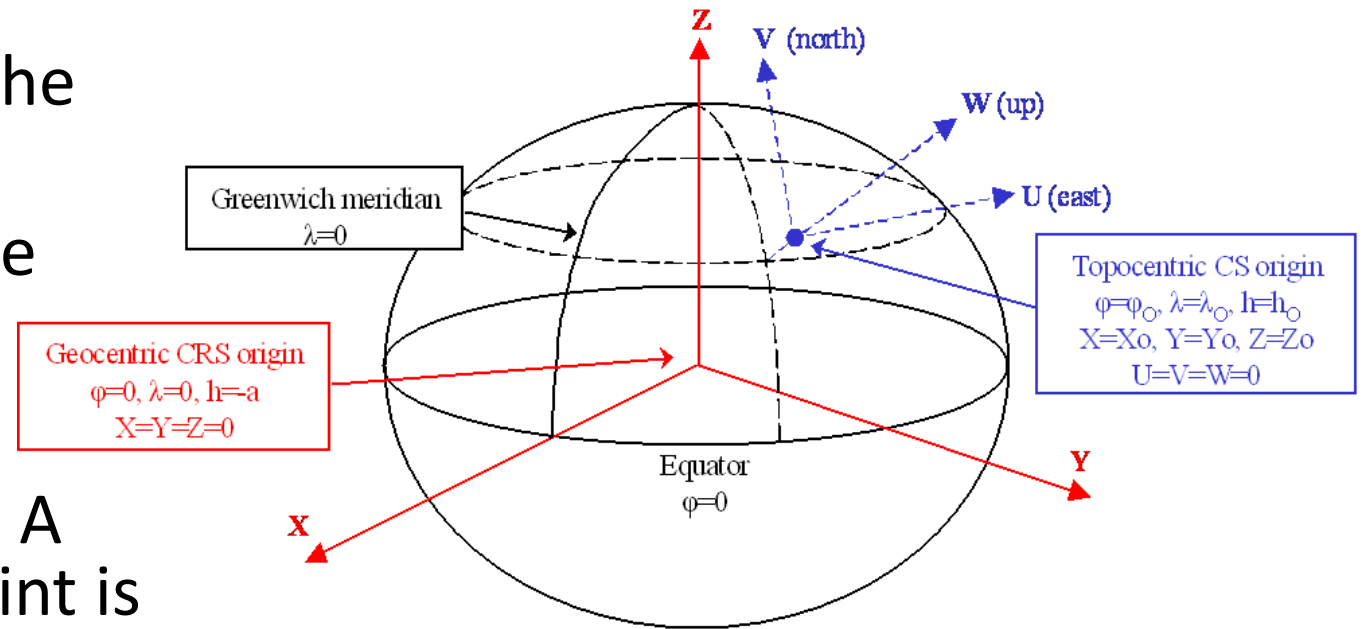
[https://www.researchgate.net/figure/Earth-orientation-parameters\\_fig1\\_289388318](https://www.researchgate.net/figure/Earth-orientation-parameters_fig1_289388318)

Orekit handles these transformations.

# Transforming from Earth Fixed to Topocentric

Topocentric frame orientation:

- The z-axis points normal to the surface
- The y-axis points towards true north
- The x-axis points east.
- Frame describes the horizon. A positive zenith means the point is in-view.
- The Azimuth/Elevation conversions can be applied to these coordinates to get a direction.



<http://www.hydrometronics.com/ecef.html>

You have reached the point where the slides are updated and cohesive.  
This document is a work-in-progress.

11/17/2019.

# Finding the Direction

By Anthony Iannuzzi



# Points on Planes to a Line Fit.

Parametric Equation of a Line:  $r = r_0 + tr_d$

Where

$$r = [x, y, z]$$

$r_0 = [x_0, y_0, z_0]$  starting point of line

$r_d = [x_d, y_d, z_d]$  direction vector

$n$  is number of points

$D = 2$  for a linear line fit

$$y = mx + b$$

**Fit a line to the data points from each plane.**

For each dimension, do a linear regression fit

$$y = Mw$$

$$x = (M^T M)^{-1} * M^T * y$$

For 4 points:

$$\begin{matrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ nx1 \end{matrix} = \begin{matrix} \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ 1 & t_3 \end{bmatrix} \\ nxD \end{matrix} * \begin{matrix} \begin{bmatrix} b \\ m \end{bmatrix} \\ Dx1 \end{matrix}$$

# Line to Azimuth and Elevation.

Parametric Equation of a Line:  $r = r_0 + tr_d$

Where

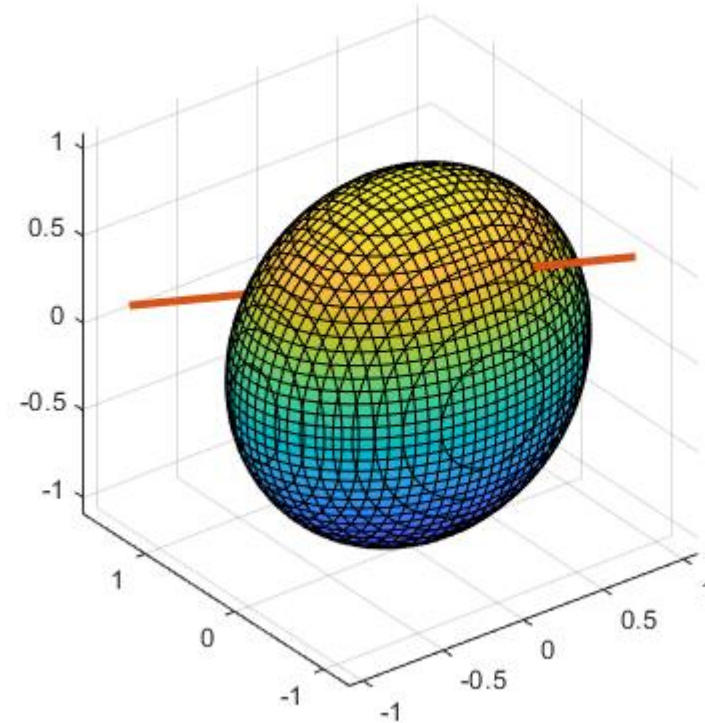
$$r = [x, y, z]$$

$r_0 = [x_0, y_0, z_0]$  starting point of line

$r_d = [x_d, y_d, z_d]$  direction vector

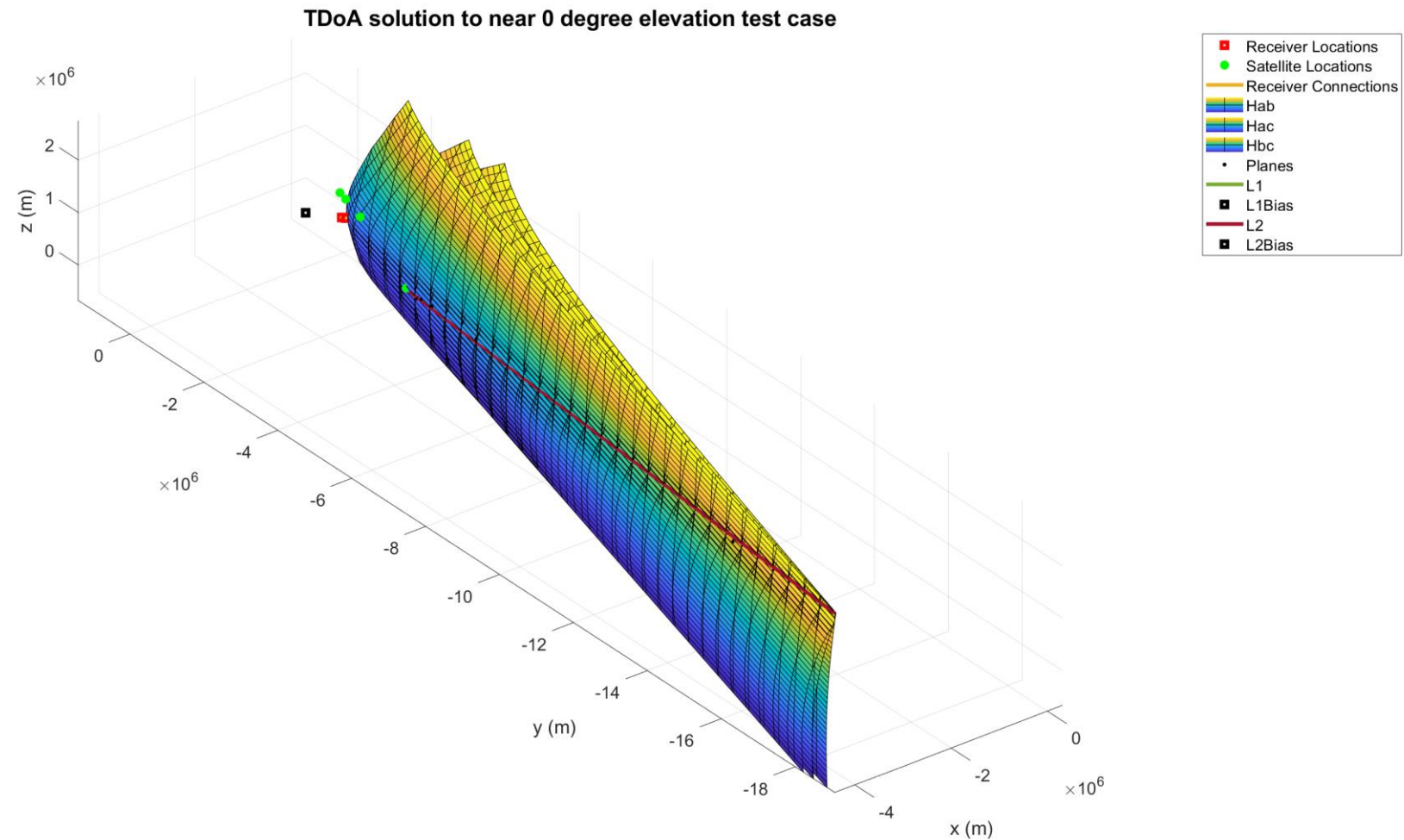
$$\text{Earth: } R_{\oplus}^2 = x^2 + y^2 + z^2$$

1. Intersect Line and Earth Model
  - a) Earth is sphere.
  - b) Constitutes our “reference frame location”
2. Convert Line direction vector to Azimuth and Elevation

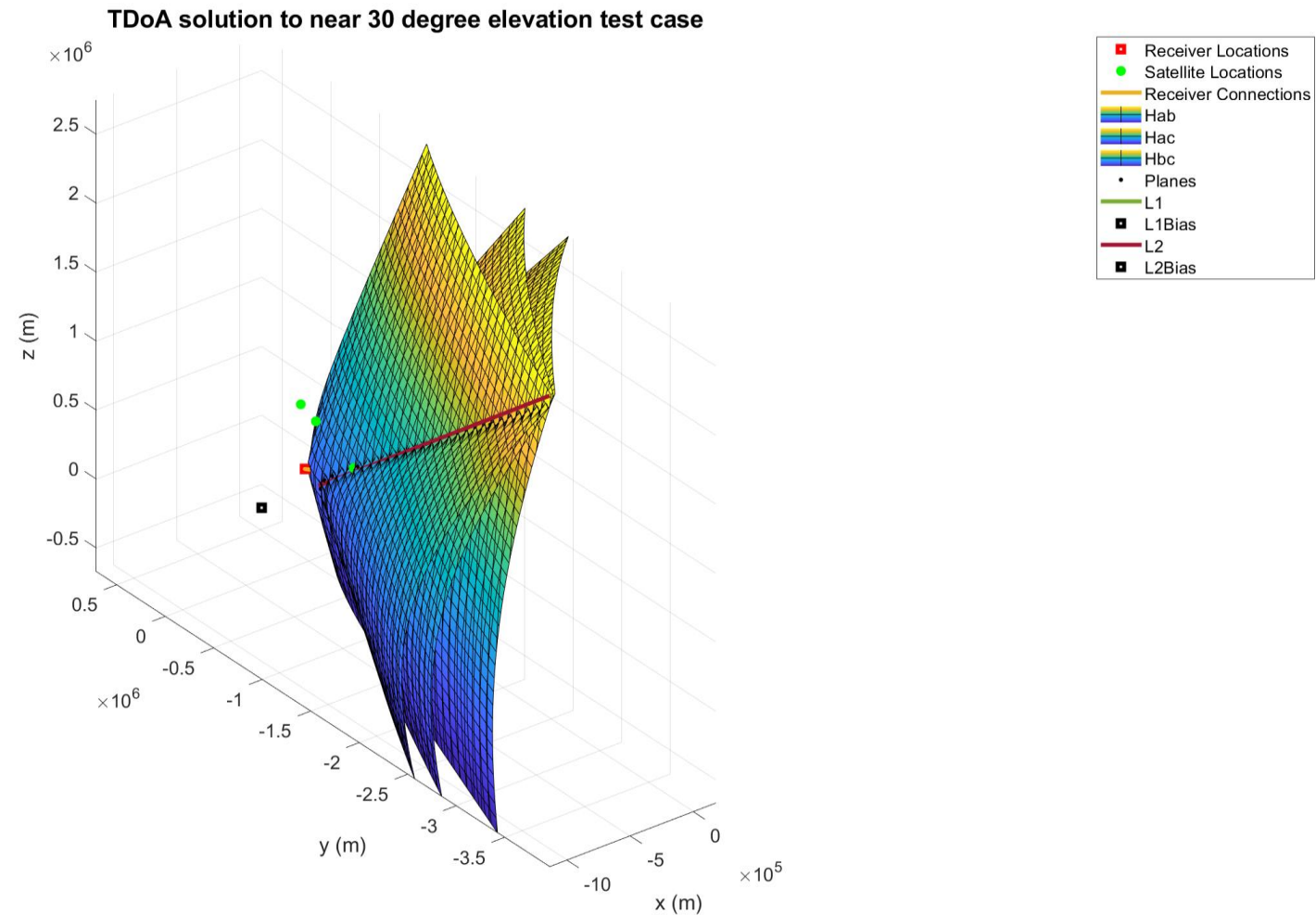


$$az = \tan^{-1} \frac{y_d}{x_d} \quad el = \tan^{-1} \frac{z_d}{\sqrt{x_d^2 + y_d^2}}$$

# Direction from TDoA near 0 degree elevation.

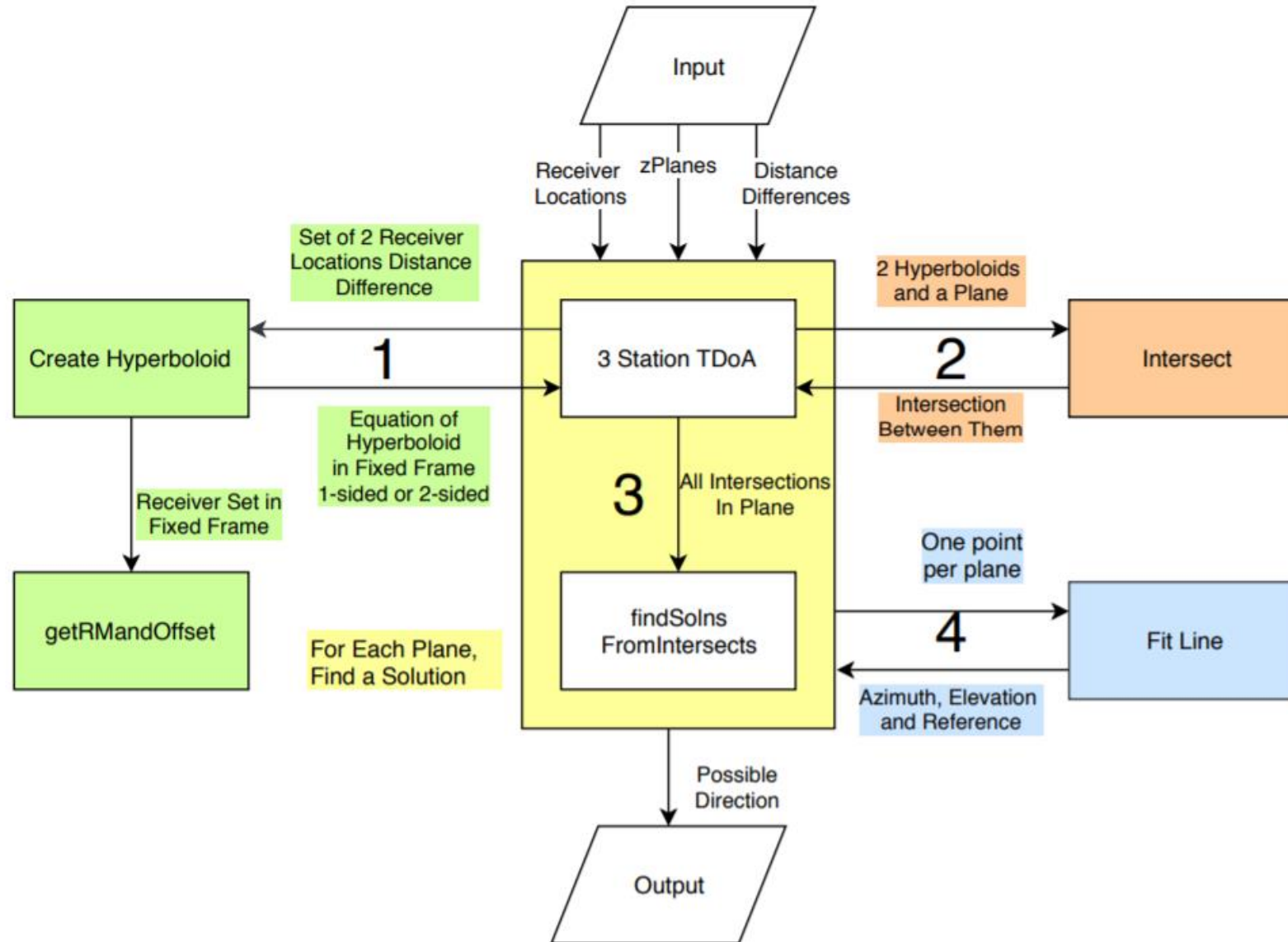


# Direction from TDoA at 30 degree elevation.

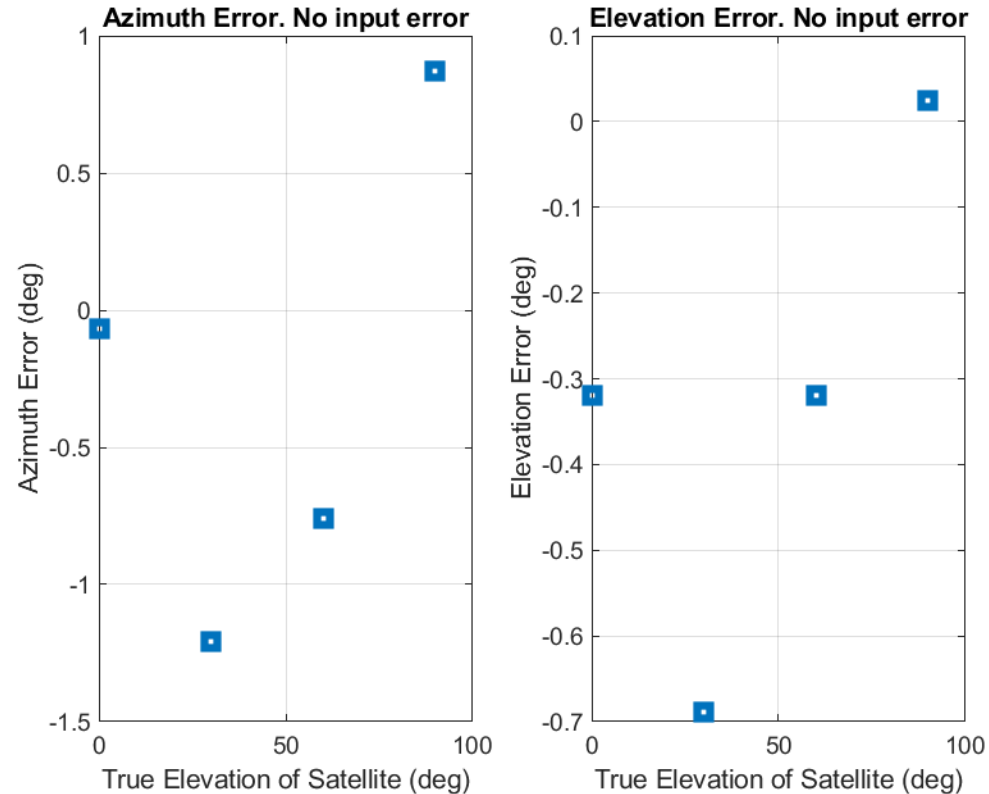


# Unit Tests

# 3 Station TDoA Simulator High level diagram.



# Algorithm Inherent Error based on Crude Ground Track is $\pm 1.25^\circ$



# Generating Inputs to the TDOA simulator

Original by Andrew DeVries

Modified by Anthony Iannuzzi



## Inputs:

- 1: Array of Latitudes, Longitudes and Elevations for G ground stations
2. Array of Latitude, Longitude, Elevation and Time Sync Errors for G ground stations
3. Array of Latitudes, Longitudes and Elevations for S satellites
4. Array of Latitude, Longitude and Elevation Errors for S satellites

getStruct.m

Fields: lat, long, elev, lat\_er, long\_er, elev\_er, coord[1x3], coord\_er[1x3], clk

geo2rect.m

[X Y Z], [Xerror Yerror Zerror]

GND (1xG ground stations)  
SAT (1xS satellites)

timeDiff.m

Loop through each  
Ground Station Pair for  
every satellite

gnd2sat.m

Distances between each  
ground station and the  
satellite (also input clk)

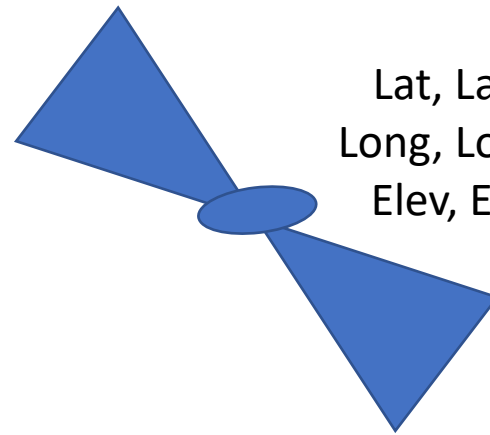
Time Differences for each  
pair for each satellite

dist2time.m

## Outputs:

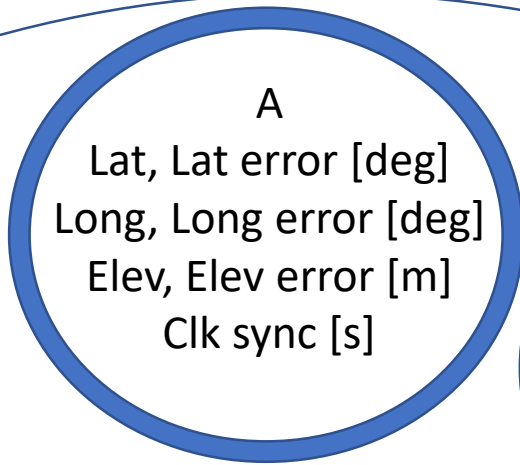
- 1: Time Differences
- 2: Time Difference Errors

# Inputs:



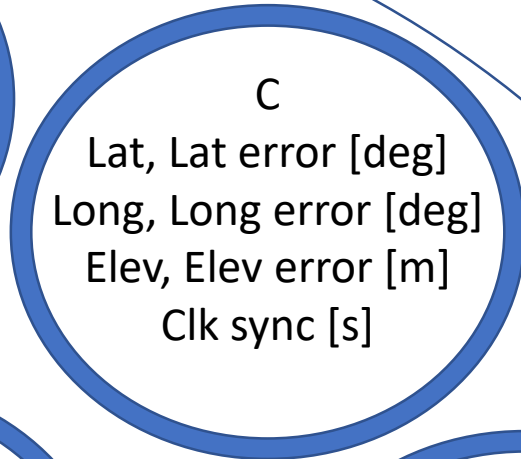
S1

Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]



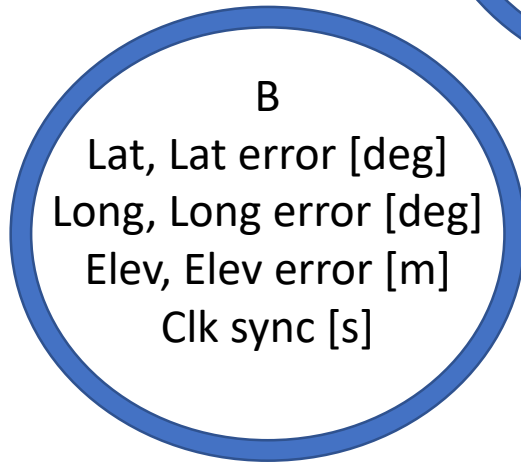
A

Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]  
Clk sync [s]



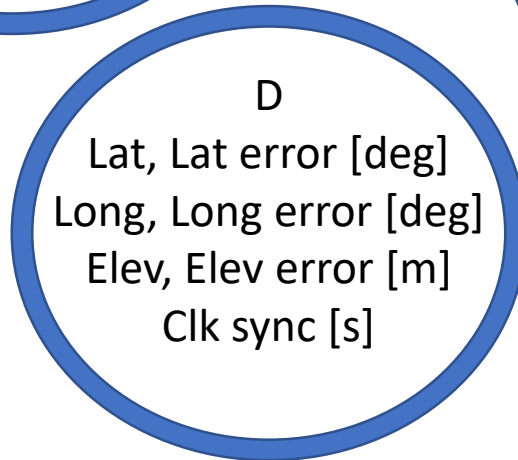
C

Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]  
Clk sync [s]



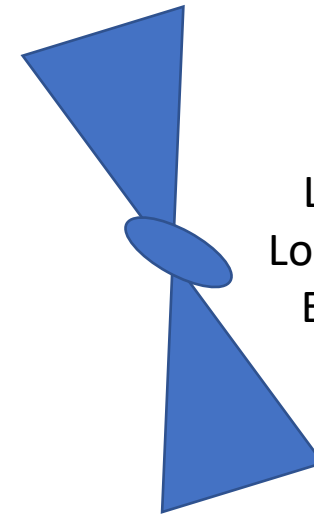
B

Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]  
Clk sync [s]



D

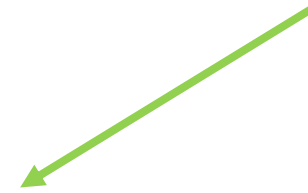
Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]  
Clk sync [s]



S2

Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]

All information captured in  
GND structure!



A

Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]  
Clk sync

C

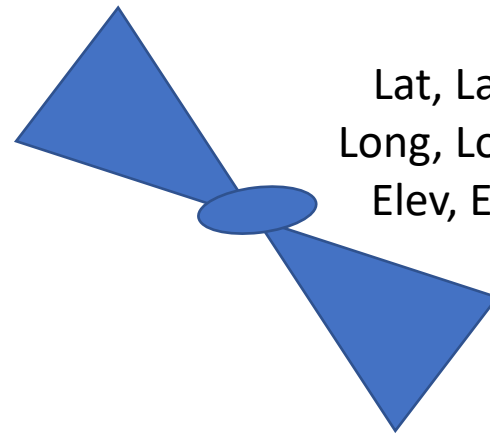
Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]  
Clk sync

B

Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]  
Clk sync

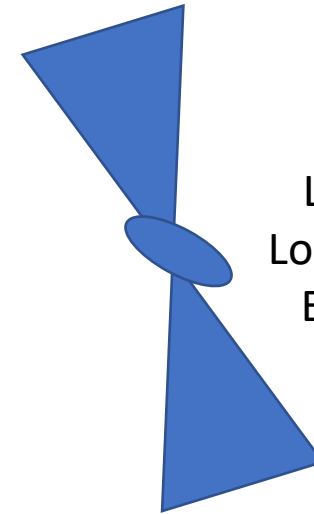
D

Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]  
Clk sync



S1  
Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]

All information captured in  
SAT structure!



S2  
Lat, Lat error [deg]  
Long, Long error [deg]  
Elev, Elev error [m]

# Outputs: Time Difference Matrix

N satellites  
G Ground Stations

	S1	S2	...	SN
Time Difference [s]	A-B	A-B		A-B
Time Difference Error [s]	A-B error	A-B error		A-B error
	A-C	A-C		A-C
	A-C error	A-C error		A-C error
	A-D	A-D		.
	A-D error	A-D error		.
	B-C	B-C		.
	B-C error	B-C error		.
	B-D	B-D		.
	B-D error	B-D error		.
	C-D	C-D		(G-1)-G
	C-D error	C-D error		(G-1)-G error

# Notes:

- timeDiff can run with any size GND and SAT structures
- The coordinate conversion assumes a spherical earth
- Satellite position is absolute Lat and Long, not azimuth and elevation
- The elevations values are in meters above sea level

# Changes

- Geo2rect can take any type of spheroid (really geo2ecef)
- measureInTopocentricFrame takes geodetic data turns into topocentric frame.
- getStruct records position in ECEF and Topocentric frame

# Estimating the Uncertainty

By Anthony Iannuzzi



The uncertainty of some output is equal to input error times the partial derivative.

For a One-at-a-Time analysis:

Given

$$g = f(x, y, z, \dots)$$

Then,

$$\Delta g = \sqrt{\left(\frac{\delta f}{\delta x} \Delta x\right)^2 + \left(\frac{\delta f}{\delta y} \Delta y\right)^2 + \left(\frac{\delta f}{\delta z} \Delta z\right)^2 + \dots}$$

The partial derivatives are called the **sensitivities**. These identify which variables the output is most affected in. Larger errors in these mean larger error in the final answer.

For complicated functions, the partial derivatives can be estimated.

For a One-at-a-Time analysis:

Given

$$g = f(x, y, z, \dots)$$

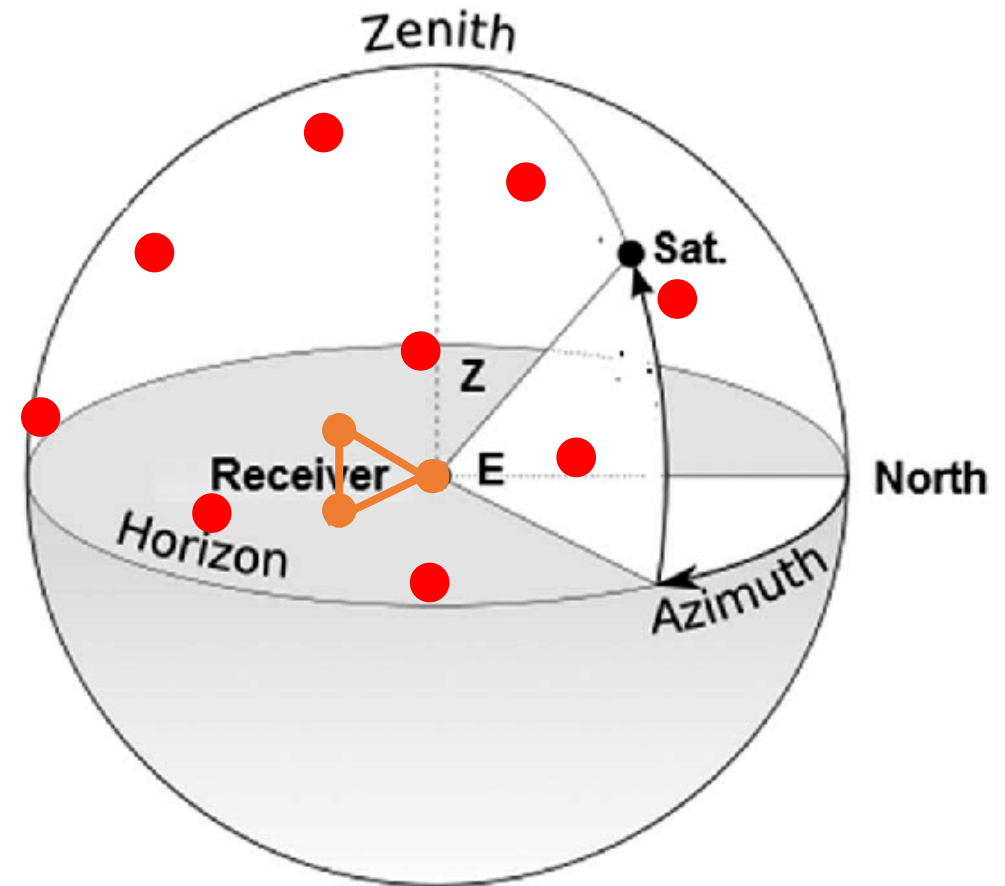
Then,

$$\frac{\delta f}{\delta x} = \frac{f(x + h, y, z, \dots) - f(x, y, z, \dots)}{h}$$

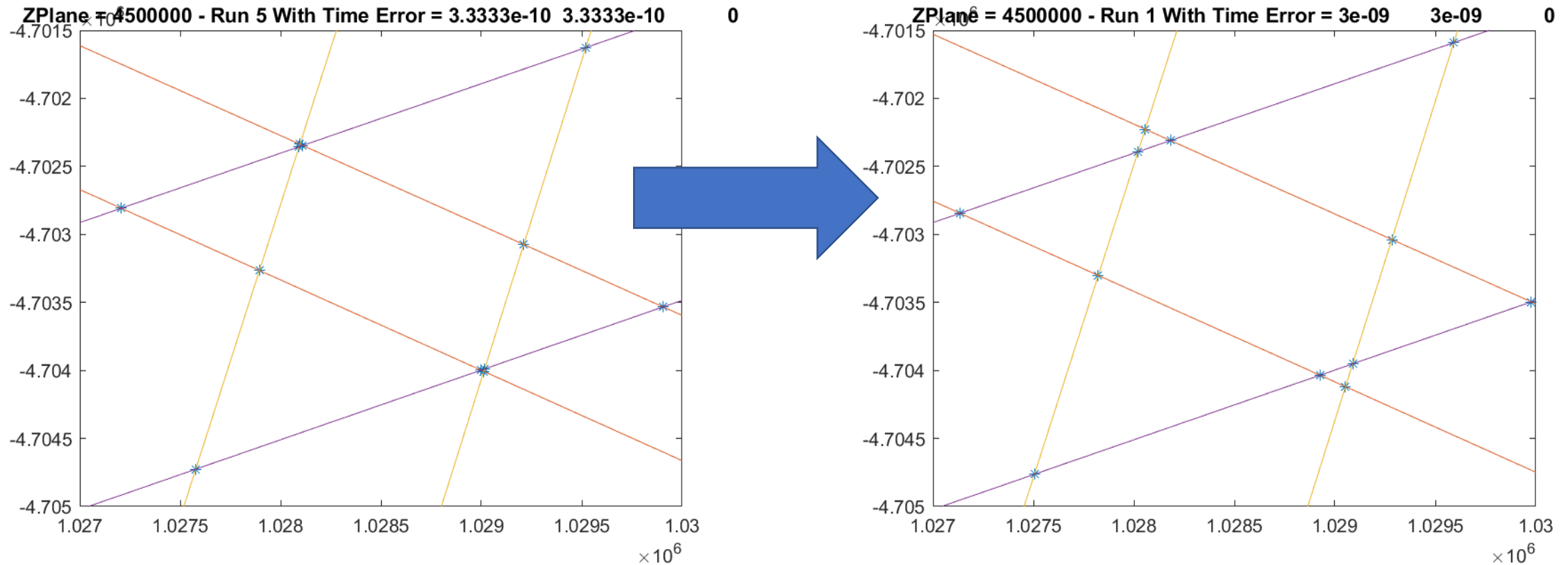
- By perturbing around the nominal inputs, we can find each partial derivative term.
- **Limitation:** the estimated partial derivatives only apply AT those nominal conditions.

# For the satellite problem, the nominal input that changes is the time differences.

- The driver of the time differences is the satellite location in the sky.  
 $Direction = f(TD, Receivers)$
- Receivers have error associated with them, but their nominal values never change.
- The Time Differences change depending on the satellite actual azimuth, elevation, and range.
  - We will estimate sensitivities for a “net” of the horizon.

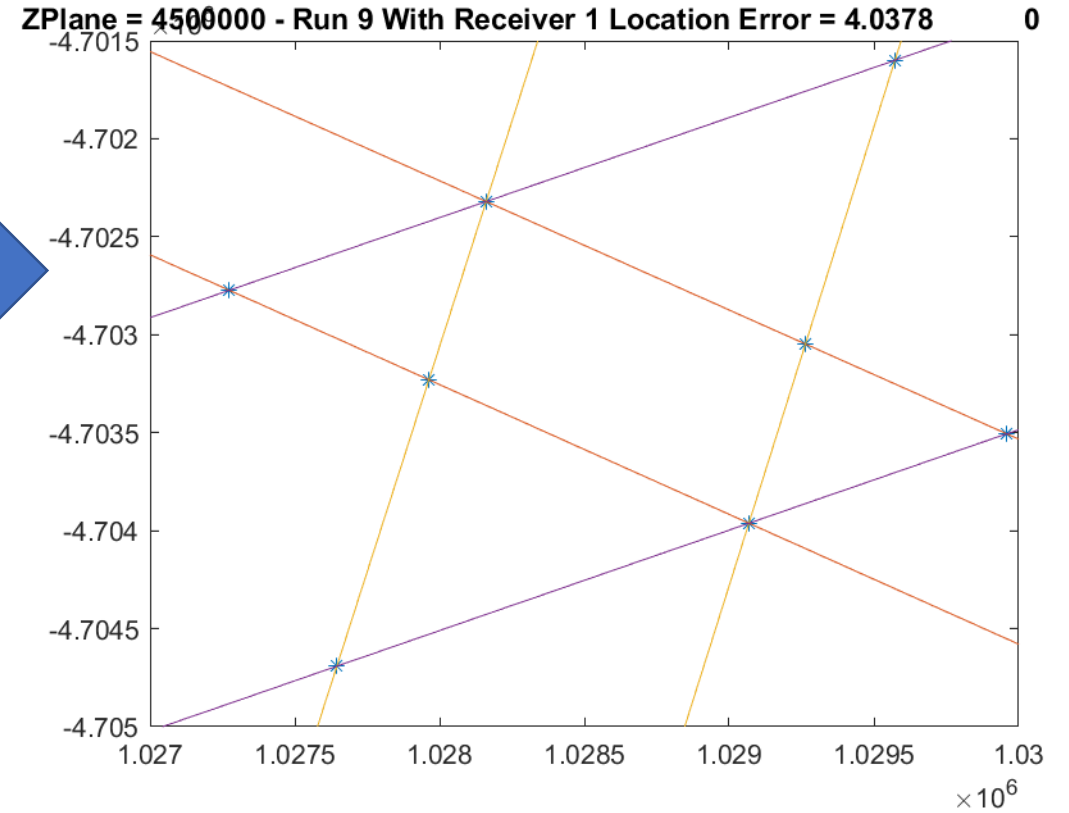
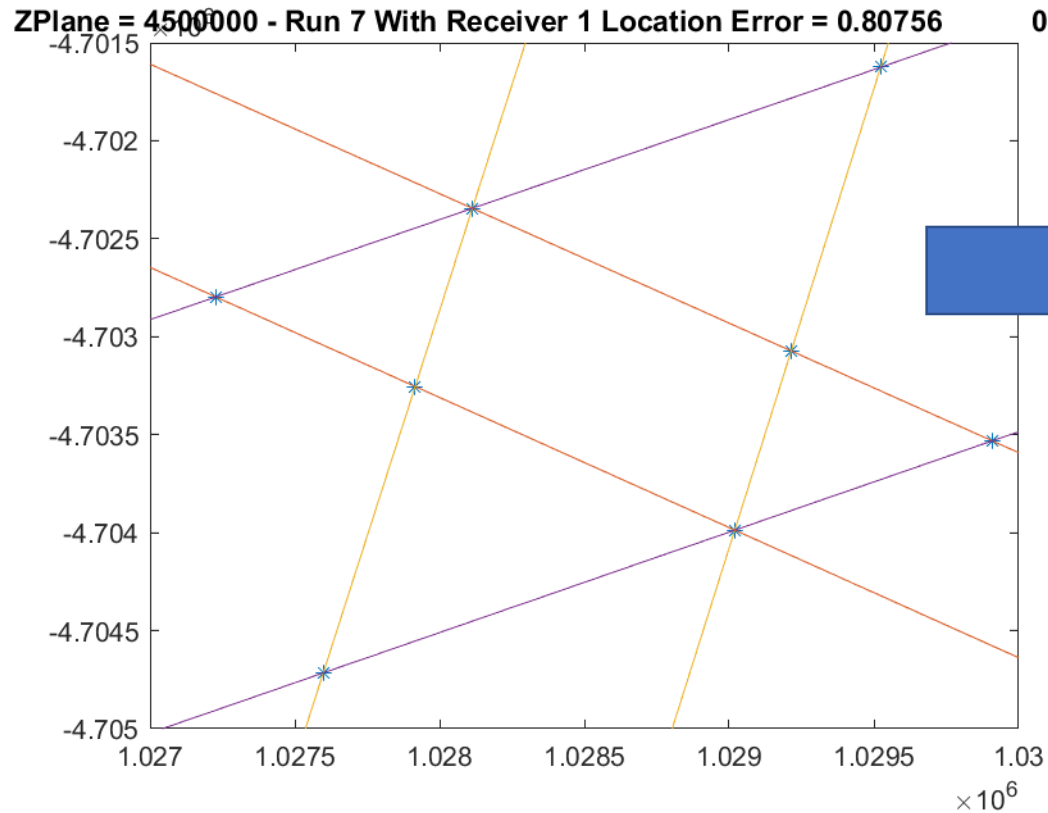


Errors in time tend to increase cluster spacing, increasing the area of potential locations. Precision Loss



But first... How do the hyperbolas change when there are errors in the system?

Errors in location shift 2 of the hyperbolas, but the cluster size is constant. Accuracy Loss.



But first... How do the hyperbolas change when there are errors in the system?

You have reached the point where the slides are updated and cohesive.  
This document is a work-in-progress.

11/17/2019.

# More than 3 base stations with small errors.

1. Poorly Constructed problem.
2. Slopes of each line are too similar – near parallel.
3. Slight deviations result in huge errors.

	No error	0.001% Location Err.	2% Time Error
Position Err. (m)	625	256,000	16,000

### 3 Base Stations with small errors.

- Satellite is over Sentinel still. Stations are located on Ellingson, The Hill, and RIT Inn.
- Time error is 300 picoseconds

Resulting Errors from Time Sync. One at a Time.

Receiver	Error (rad)
A	$1.5 \times 10^{-4}$
B	$3.4 \times 10^{-4}$
C	$3.75 \times 10^{-4}$



### 3 Base Stations with small errors.

- Location error is Lat: 0.00005 deg, Long: 0.0001 deg, Altitude: 0.9m

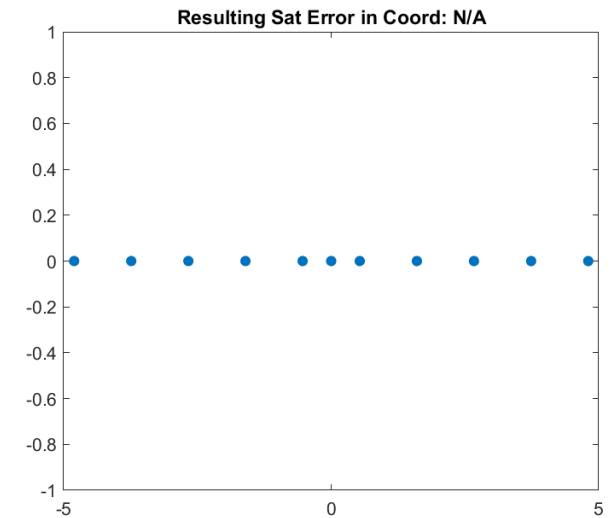
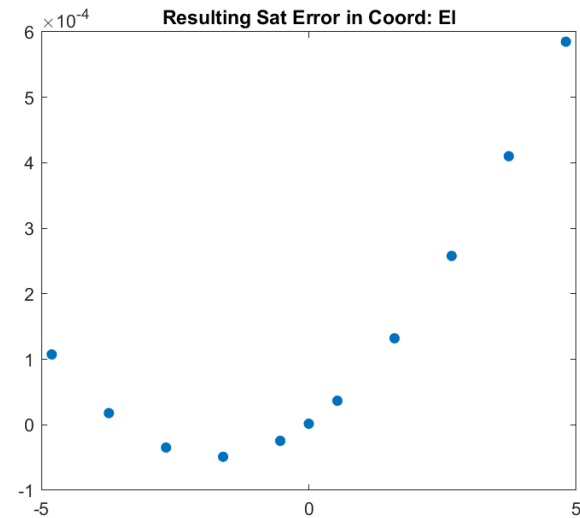
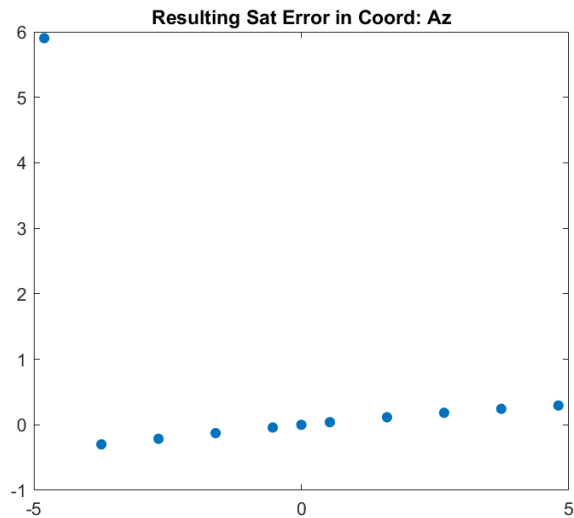
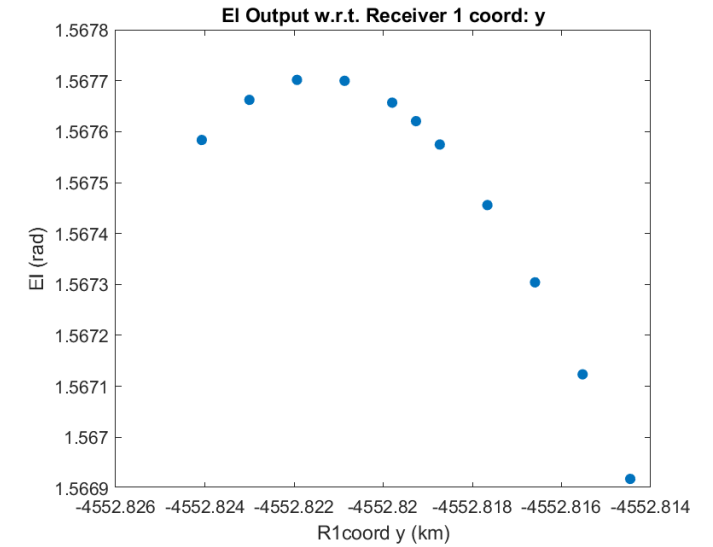
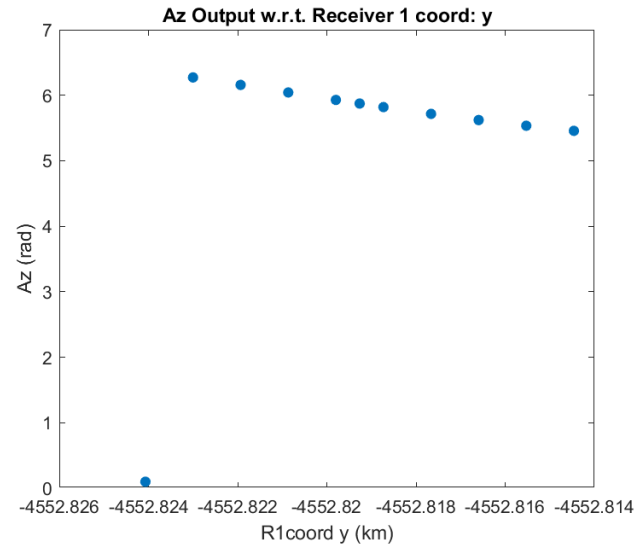
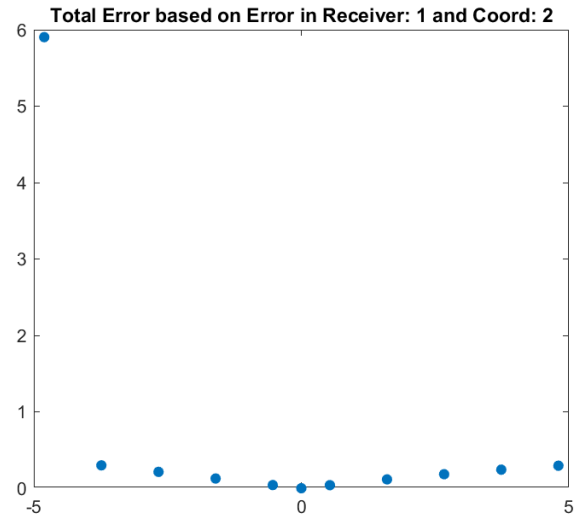
Derived Location Error

Receiver	X (m)	Y (m)	Z (m)
A	7.268079	4.798699	4.67656
B	7.2629	4.802649	4.675313
C	7.267696	4.801882	4.677395

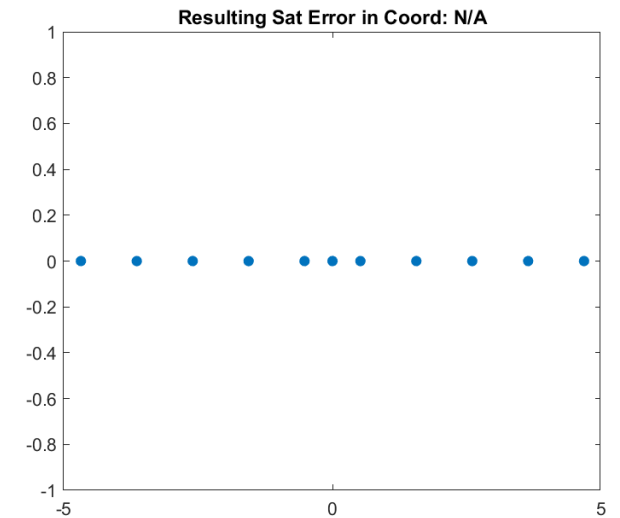
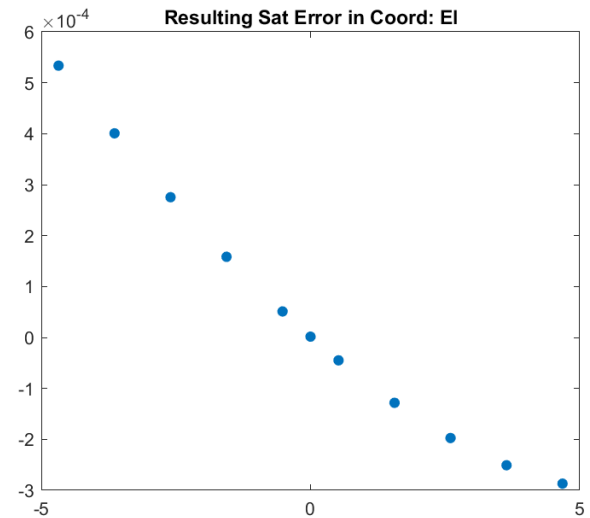
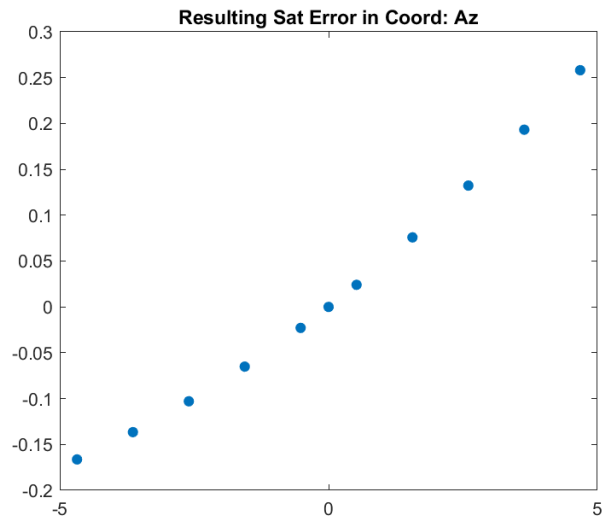
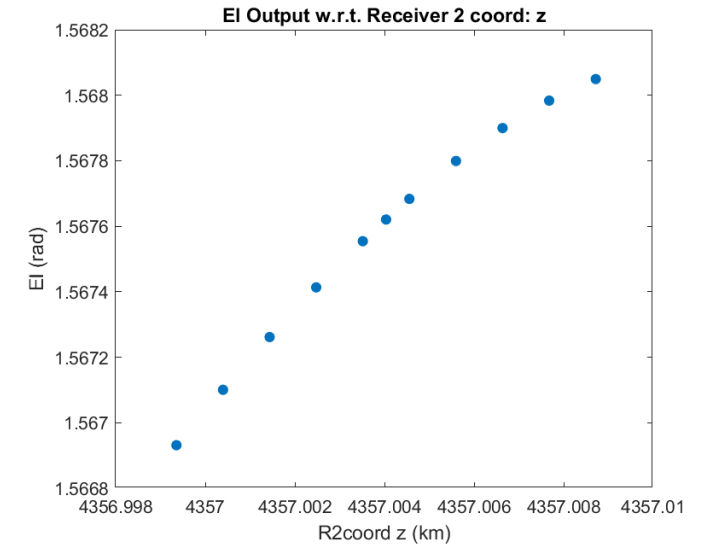
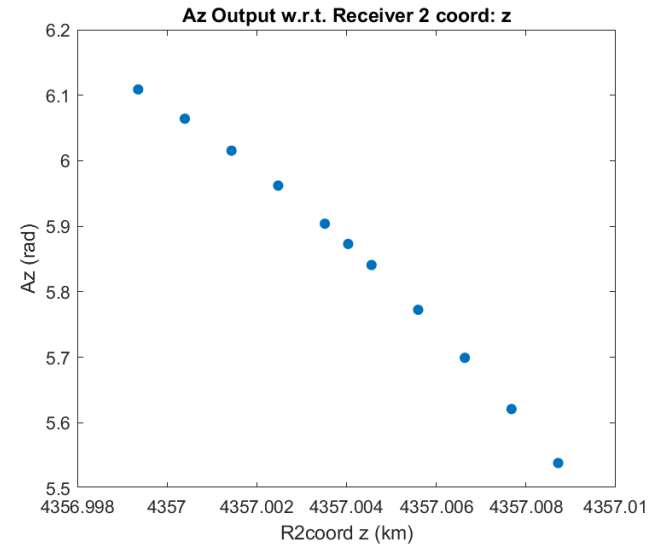
Resulting Errors from Location Error. One at a Time.

Receiver	From X (rad)	From Y (rad)	From Z (rad)
A	$7 \times 10^{-4}$	$22 \times 10^{-4}$	$20 \times 10^{-4}$
B	$4 \times 10^{-4}$	$12 \times 10^{-4}$	$12 \times 10^{-4}$
C	$4.5 \times 10^{-4}$	$14 \times 10^{-4}$	$14 \times 10^{-4}$

# One at a time Plot



# One at a time Plot



# Derived Sensitivities

Using “total error”

Parameter	Sensitivity (rad/m) & (rad/s)
$x_a$	9.60E-05
$y_a$	4.58E-04
$z_a$	4.27E-04
$x_b$	5.48E-05
$y_b$	2.49E-04
$z_b$	2.56E-04
$x_c$	6.16E-05
$y_c$	2.91E-04
$z_c$	2.99E-04
$t_a$	4.92E+05
$t_b$	1.13E+06
$t_c$	1.24E+06

# Solving with more than 3 stations

By Anthony Iannuzzi

# More than 3 base stations.

1. Create  $m$  sets of 3 Hyperboloids.
2. Solve each of these individually
3. Intersect the resulting line fits via least squares.
4. Output: Single Point.

$$y = Mw$$

$$x = (M^T M)^{-1} * M^T * y$$

For 2 lines:

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ x_{02} \\ y_{02} \\ z_{02} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_d & 0 \\ 0 & 1 & 0 & y_d & 0 \\ 0 & 0 & 1 & z_d & 0 \\ 1 & 0 & 0 & 0 & x_{d2} \\ 0 & 1 & 0 & 0 & y_{d2} \\ 0 & 0 & 1 & 0 & z_{d2} \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ t \\ s \end{bmatrix}$$

$3n \times 1$        $3n \times (3 + n)$        $(3 + n) \times 1$

$$r = r_0 + tr_d$$

$$r_2 = r_{02} + sr_{d2}$$

