Analyses of the geolocation accuracy that can be obtained from shipborne sensors by use of time difference of arrival (TDOA), scanphase, and angle of arrival (AOA) measurements

Gudrun Høye

Forsvarets forskningsinstitutt/Norwegian Defence Research Establishment (FFI)

25 March 2010

FFI-rapport 2010/00737

1080

P: ISBN 978-82-464- 1746-2
E: ISBN 978-82-464- 1747-9

Keywords

Geolokalisering

Tidsdifferansemålinger

Skannfasemålinger

Peilinger

ESM

Maritim

Approved by

Berit Jahnsen	Project Manager
Vidar S. Andersen	Director

English summary

We have performed a theoretical study of the geolocation accuracy that can be obtained from shipborne sensors by use of the localization methods: 1) time difference of arrival (TDOA), 2) scanphase, and 3) angle of arrival (AOA).

Each method has been studied separately and in combination with the other methods. Several sensor geometries and types of movement have been investigated. Software for the simulations was written in Matlab v7.7.

The simulations showed that geolocation accuracies of about 5 m at 10 km distance and 100 m at 50 km distance from the sensors could be obtained after 15 min of observations with two sensors. The emitter was assumed to be stationary during the observations. If only one sensor was used, the corresponding accuracies were 100 m and 3000 m.

Momentaneous geolocation could be obtained if two or more sensors were used. With four sensors the geolocation accuracy was then found to be approximately 40 m at 10 km distance and 800 m at 50 km distance from the sensors.

The geolocation error dependency on different parameters was also investigated. The geolocation error was found to be proportional to the error in the measured parameter, proportional to the inverse of the distance between the sensors, proportional to the inverse of the trajectory length and proportional to the inverse of the square root of the number of measurements. Further, the simulations showed that the geolocation error increases rapidly with increasing distance from the sensors.

Additional factors that may affect the geolocation accuracy, but that have not been included in our calcuations, are errors in the reported sensor position, emitter movement during the observation period and limitations in de-interleaving capability and data transfer capacity.

Recommendations for geometry and trajectory choices for the sensors are included in the report.

Sammendrag

I denne rapporten har vi gjort en teoretisk studie av hvilke geolokaliseringsnøyaktigheter man kan oppnå fra maritime sensorer ved bruk av lokaliseringsmetodene: 1) tidsdifferansemålinger (TDOA), 2) skannfasemålinger og 3) peilinger (AOA).

Metodene er studert hver for seg og i ulike kombinasjoner. Flere forskjellige sensorgeometrier og bevegelsesmønstre har vært undersøkt. Programvaren for simuleringene ble skrevet i Matlab v7.7.

Simuleringene viste at det var mulig å oppnå geolokaliseringsnøyaktigheter på rundt 5 m på 10 km avstand og 100 m på 50 km avstand fra sensorene etter 15 min med observasjoner når det ble brukt to sensorer. Dette forutsatte at emitteren sto tilnærmet i ro mens observasjonene ble gjort. Dersom kun én sensor ble brukt var tilsvarende nøyaktigheter 100 m og 3000 m.

Momentan geolokalisering var mulig hvis to eller flere sensorer ble benyttet. Med fire sensorer var det da mulig å oppnå geolokaliseringsnøyaktigheter på ca 40 m på 10 km avstand og 800 m på 50 km avstand fra sensorene.

Det ble også undersøkt hvordan feilen i geolokaliseringen avhenger av ulike parametere. Feilen i geolokaliseringen ble funnet å øke proporsjonalt med feilen i den målte parameteren, omvendt proporsjonalt med avstanden mellom sensorene, omvendt proporsjonalt med avstanden sensorene tilbakelegger og omvendt proporsjonalt med kvadratroten av antallet målinger som blir gjort. Videre viste simuleringene at feilen øker raskt med økende avstand fra sensorene.

Andre faktorer som påvirker geolokaliseringsnøyaktigheten, men som ikke er tatt med i våre beregninger, er unøyaktighet i de angitte sensorposisjonene, emitterbevegelse i løpet av observasjonstiden samt begrensninger i pulssortering og dataoverføringskapasitet.

Anbefalinger for valg av geometri og bevegelsesmønstre for sensorene er inkludert i rapporten.

Contents

1 Introduction

New and more precise geolocation methods are considered for implementation on Norwegian navy vessels. The most promising methods for geolocating emitters from shipborne sensors are:

1) Time difference of arrival (TDOA) measurements
2) Scanphase measurements
3) Angle of arrival (AOA) measurements

Precise angle of arrival measurements can be performed with the new direction-finding sensors that are now being installed on the Norwegian frigates (Figure 1.1). Time difference of arrival measurements will in addition require the installation of a very precise clock on the vessels.

The TDOA and scanphase methods have already been studied for geolocation of emitters from airborne sensors [1], [8], [14]. In this report we will do a theoretical study of the geolocation accuracy that can be obtained from shipborne sensors. All three localization methods will be considered.

Chapter 2 presents the theory for the geolocation accuracy analyses, while Chapter 3 describes the simulations that were performed for the analyses. The results are presented and discussed in Chapter 4 with a summary of the main findings after each of the sections 4.1-4.4. Recommendations for trajectory choices are given in Chapter 4.6, while Chapter 5 summarizes the work of the report. The appendices give additional information on some subjects. The software used for the simulations is described in Appendix E.

Figure 1.1 Fridtjof Nansen class frigate.

2 Theory

This chapter presents the necessary theory for the geolocation accuracy analyses. Three different localization methods are studied:

1) Time difference of arrival (TDOA)
2) Scanphase
3) Angle of arrival (AOA)

The TDOA and scanphase methods require the use of 2 sensors, while the AOA method requires the use of one direction-finding (DF) sensor. In addition, the TDOA method requires the use of a very precise clock, such as for instance a rubidium clock, that can measure time with a precision of a few tens of nanoseconds. For the scanphase method a clock precision of about 1 ms is sufficient. Table 2.1 gives an overview of the methods that can be used for different combinations of sensor equipment.

Equipment:	**Method:**
1 DF-sensor	AOA
2 sensors	Scanphase
2 sensors + very precise clock	TDOA and/or Scanphase
2 DF-sensors(*)	AOA and/or Scanphase
2 DF-sensors(*) + very precise clock	TDOA and/or Scanphase and/or AOA

(*) 1 DF-sensor is sufficient in order to apply the AOA method.

Table 2.1 Overview of the methods that can be used for different combinations of sensor equipment.

The geolocation theory for the three methods is presented in Section 2.1, while Section 2.2 describes characteristics of the methods.

We do not consider the actual *geolocation* of the emitter in this study. For this a geolocation estimator would be needed. Instead, we assume that the emitter position is already known (estimated), and determine the *geolocation accuracy* that corresponds to this position. The geolocation accuracy can be calculated as the Cramer-Rao Lower Bound (CRLB) or the Circular Error Probable (CEP). These two methods for error estimation are described in Section 2.3.

In this report we study the geolocation accuracy for scenarios where both the emitter and the sensors are placed on the *sea surface*. This is a 2-dimensional problem, which can be approximated by the assumption that both the emitter and the sensors are placed in the same *horizontal plane* (flat Earth approximation). For studies of the geolocation *accuracy* this is sufficient. Cartesian coordinates (x, y) can then be used in the calculations.

2.1 Localization methods

The three localization methods are presented below. The theory and equations are valid for the general 3-dimensional case. The 2-dimensional special case is obtained by setting the sensor and emitter altitudes equal to zero in the equations.

2.1.1 Time Difference of Arrival (TDOA)

The time difference of arrival (TDOA) method for emitter localization is based on measuring the TDOA for the emitted radar pulses at two sensors, see Figure 2.1.

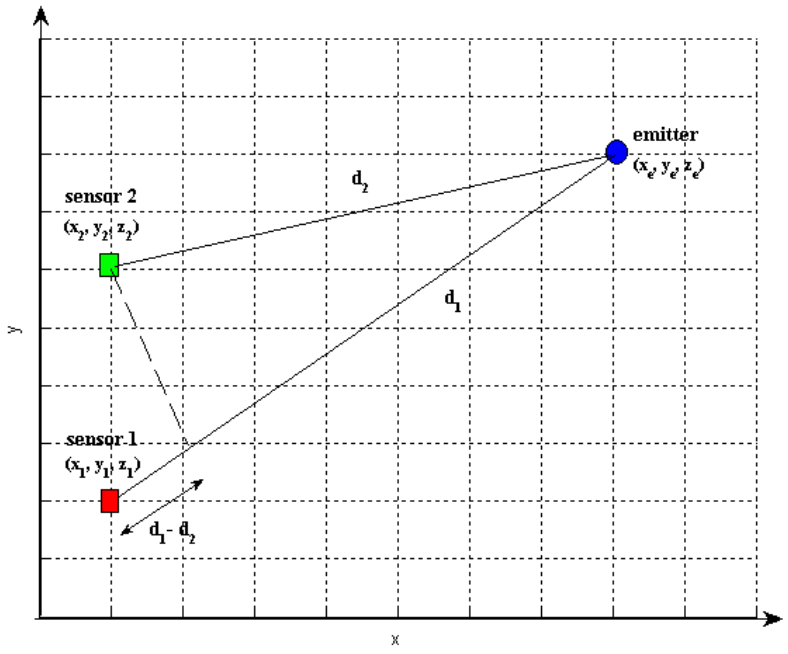


Figure 2.1 TDOA principle.

The time difference of arrival (TDOA) is given by

$$TDOA = \frac{1}{c}(d_1 - d_2) \tag{2.1}$$

where c is the speed of light and d_1 and d_2 are the distances between sensor 1 and the emitter and sensor 2 and the emitter respectively:

$$\begin{aligned} d_1 &= \sqrt{(x_1 - x_e)^2 + (y_1 - y_e)^2 + (z_1 - z_e)^2} \\ d_2 &= \sqrt{(x_2 - x_e)^2 + (y_2 - y_e)^2 + (z_2 - z_e)^2} \end{aligned} \tag{2.2}$$

Here (x_1, y_1, z_1) is the position of sensor 1, (x_2, y_2, z_2) is the position of sensor 2, and (x_e, y_e, z_e) is the emitter position.

Based on Equation (2.1) possible emitter positions can be determined when the time difference of arrival is known (measured). Equation (2.1) does not give a unique position for the emitter, since the equation contains three unknowns (x_e, y_e, z_e). In the 3-dimensional case the possible emitter positions are located on a circular hyperboloid with the sensors at the focal points (see Appendix A.3-A.4). In the 2-dimensional case where the sensors and the emitter are in the same plane, the possible emitter positions are located along a hyperbola with the sensors at the focal points (see Appendix A.1-A.2). This is illustrated in Figure 2.2.

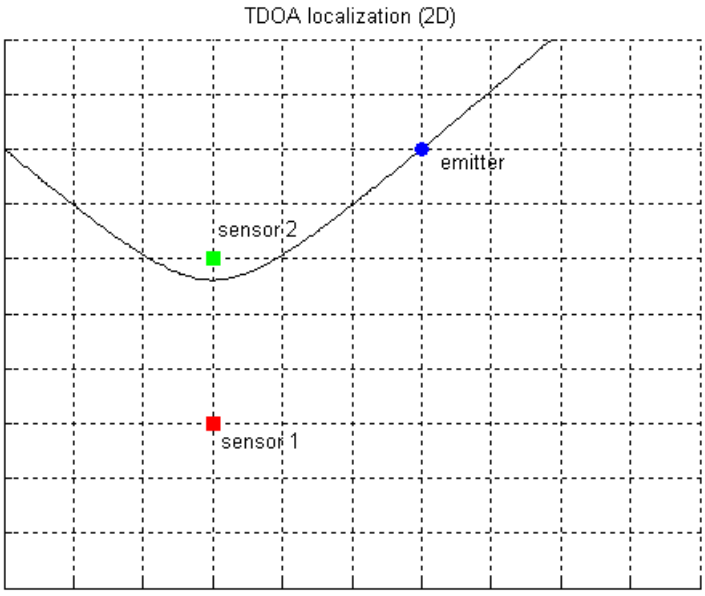


Figure 2.2 TDOA localization in 2 dimensions. Red and green squares show the positions of sensor 1 and 2, while the blue circle shows the actual emitter position. The black curve (hyperbola) shows the possible emitter positions based on the measured TDOA.

2.1.2 Scanphase

Scanphase localization is based on measuring the radar rotation time and the radar mainlobe time difference of arrival at two sensors. Together this information is used to determine the aperture angle υ, see Figure 2.3.

The aperture angle υ is given by

$$\upsilon = \arccos\left(\frac{d_{12}^2 - d_1^2 - d_2^2}{2d_1 d_2}\right) \tag{2.3}$$

where

$$d_1 = \sqrt{(x_1 - x_e)^2 + (y_1 - y_e)^2}$$
$$d_2 = \sqrt{(x_2 - x_e)^2 + (y_2 - y_e)^2} \qquad (2.4)$$
$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Here (x_1, y_1) is the position of sensor 1, (x_2, y_2) is the position of sensor 2, and (x_e, y_e) is the emitter position. Note that the sensor and emitter altitudes are not relevant for the calculations since their values do not affect the measured radar rotation time or the measured time difference of arrival for the mainlobe. For this reason, the scanphase method can not be used alone to locate emitters in three dimensions.

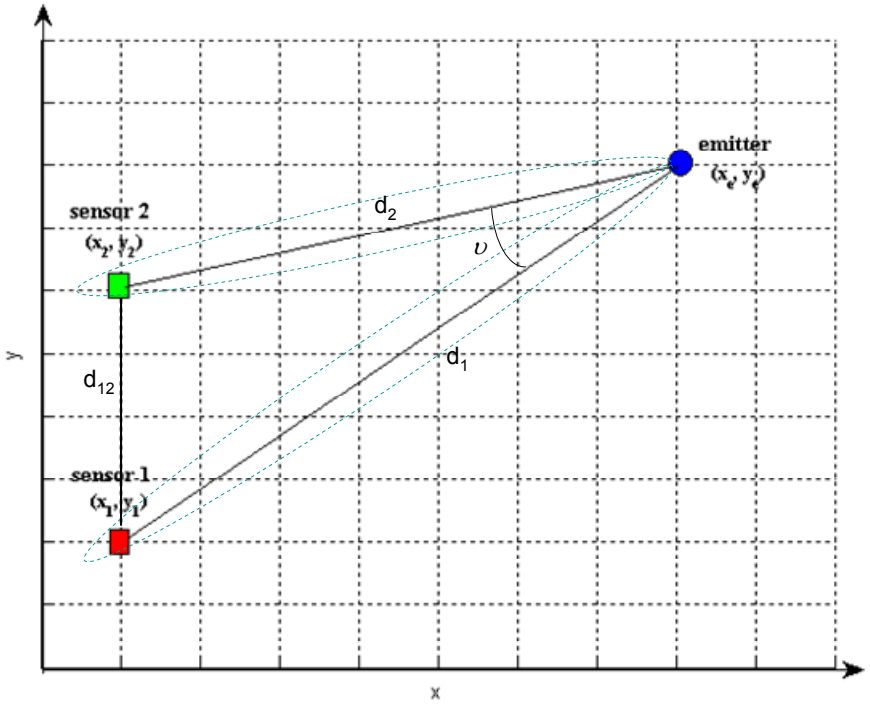


Figure 2.3 Scanphase principle.

Based on Equation (2.3) possible emitter positions can be determined in two dimensions when the aperture angle is known. The aperture angle is not measured directly but can be calculated from

$$\upsilon = \frac{2\pi \cdot \Delta T}{T_{rot}} \qquad (2.5)$$

where T_{rot} is the measured radar rotation time and ΔT is the measured radar mainlobe time difference of arrival at sensors 1 and 2.

Equation (2.3) does not give a unique position for the emitter, since the equation contains two unknowns (x_e, y_e). The possible emitter positions are located on two circles (see Appendix B). This is illustrated in Figure 2.4.

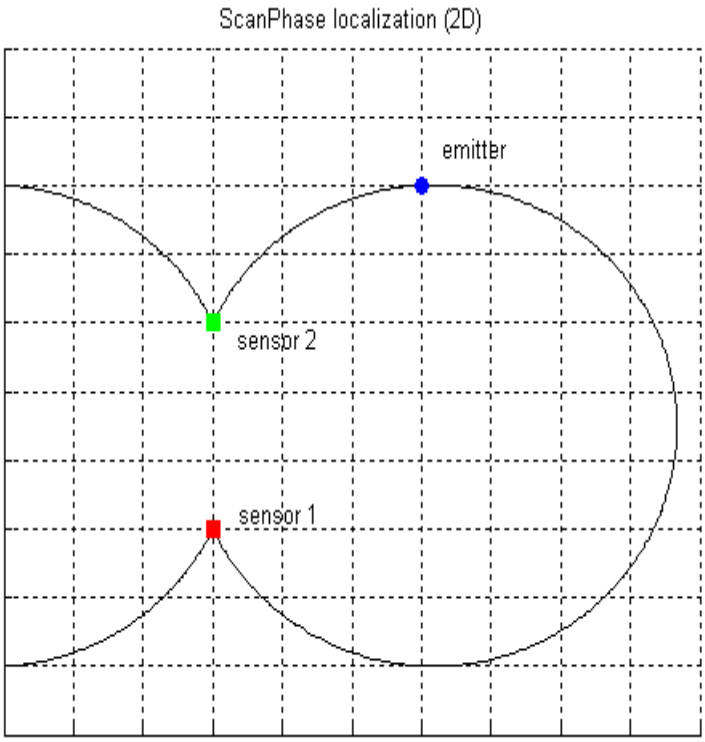


Figure 2.4 Scanphase localization in 2 dimensions. Red and green squares show the positions of sensor 1 and 2, while the blue circle shows the actual emitter position. The black circles show the possible emitter positions based on the measured aperture angle.

2.1.3 Angle of Arrival (AOA)

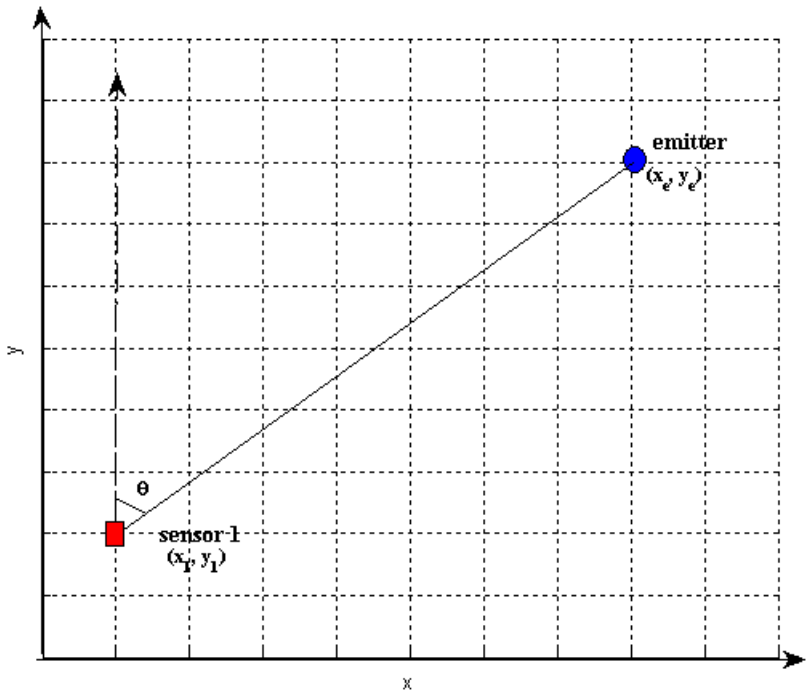


Figure 2.5 AOA principle.

Angle of arrival (AOA) localization is based on measuring the angle of arrival θ of the radar pulses that are received from an emitter at the sensor, see Figure 2.5. This method requires the use of one sensor only.

The angle of arrival θ is given by

$$\theta = \arctan\left(\frac{x_e - x_1}{y_e - y_1}\right) \tag{2.6}$$

where θ is measured relative to the y-axis in the horizontal plane as shown in Figure 2.5. Here (x_1, y_1) is the position of the sensor and (x_e, y_e) is the emitter position. Note that the sensor and emitter altitudes are not relevant for the calculations since their values do not affect the measured angle of arrival. For this reason, the AOA method can not be used alone to locate emitters in three dimensions.

Based on Equation (2.6) possible emitter positions can be determined when the angle of arrival is known (measured). Equation (2.6) does not give a unique position for the emitter, since the equation contains two unknowns (x_e, y_e). The possible emitter positions are located along the straight line

$$y = y_1 + \tan\theta \cdot (x - x_1) \tag{2.7}$$

This is illustrated in Figure 2.6.

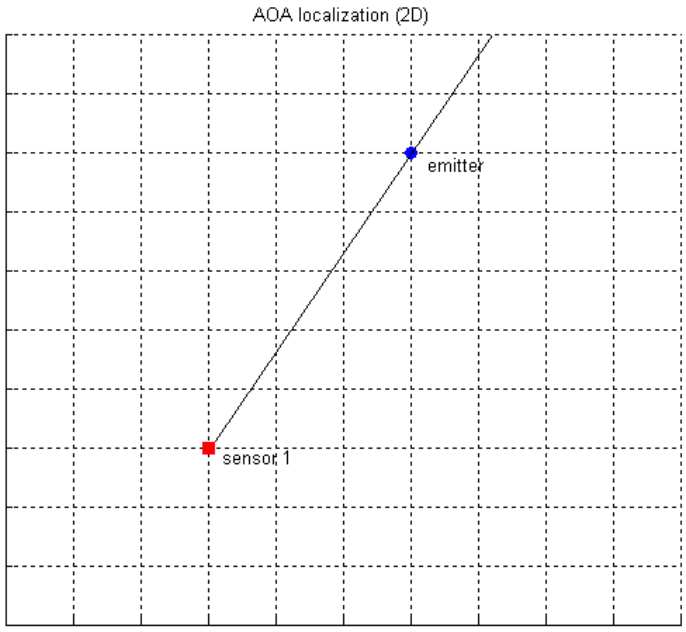


Figure 2.6 AOA localization in 2 dimensions. The red square shows the position of the sensor, while the blue circle shows the actual emitter position. The black straight line shows the possible emitter positions based on the measured angle of arrival.

2.2 Characteristics of the methods

We will in this section study some characteristics of the three methods. We will look at the symmetry of the methods, the presence of mirror images, the methods' ability to geolocate the emitter for different geometries and types of sensor movement, and geolocation error behaviour and error bounds. The following terms will be used in the discussion, see also Figure 2.7:

Sensor line: The straight line that goes through both sensors.

External sensor line: a) Part of the sensor line that is *not between* the sensors.
b) Part of the sensor line that is *not between* the first and last sensor position, if the sensors move after each other along the sensor line.

Middle line: The line that crosses the sensor line at a right angle midway between the sensors.

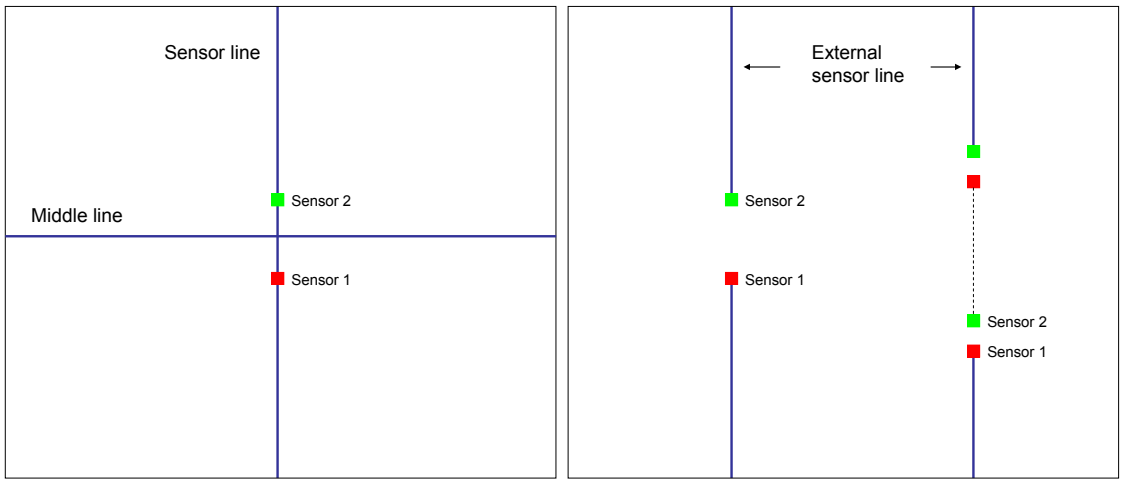


Figure 2.7 Sensor line, middle line, and external sensor line.

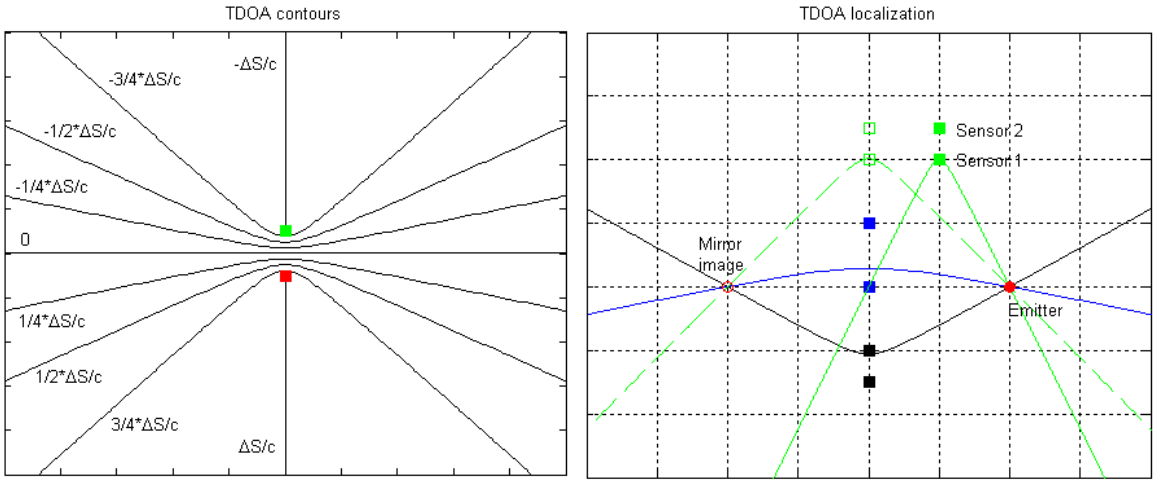
2.2.1 Symmetry and mirror images

The three methods show different behaviour with respect to symmetry and presence of mirror images.

2.2.1.1 Time difference of arrival (TDOA)

Figure 2.8a) shows TDOA contours in the *xy*-plane. The TDOA can have values in the range $[-\Delta S/c \;\; \Delta S/c]$, where ΔS is the distance between the sensors and c is the speed of light. The TDOA contours, which describe possible emitter positions for given TDOA, are symmetric around the sensor line. This symmetry creates mirror images under certain conditions, as we will see below.

Figure 2.8b) shows TDOA localization. Two measurements at different sensor positions give *two* possible emitter positions (at the cross-section between the two corresponding curves). At least three measurements at different sensor positions are therefore required to locate the emitter unambiguously. This can be obtained by moving the sensors or by using more than two sensors (three is sufficient if the sensors are not placed along the same line). Note, however, that due to the symmetry of the system a mirror image will be created independent of the number of measurements if the sensors move only along the sensor line. In order to resolve this ambiguity some measurements must be performed when the sensors are located away from the sensor line.



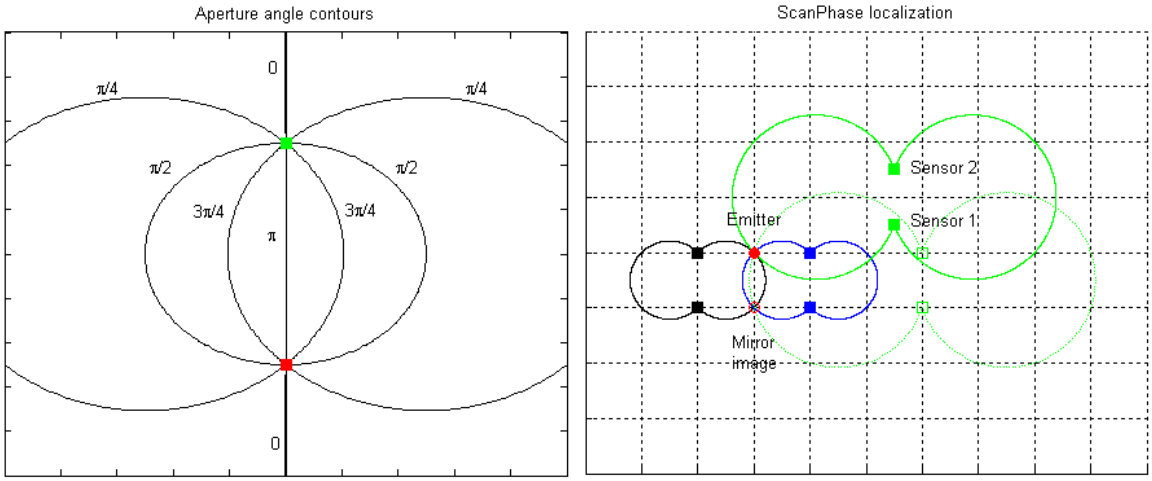
a) b)

Figure 2.8 The TDOA method. Left figure shows TDOA contours in the xy-plane. Here ΔS is the distance between the sensors and c is the speed of light. Right figure shows TDOA localization. Each measurement is represented by a different colour (black, blue, and green). The dashed green line is an alternative to the green solid line and demonstrates the creation of mirror images.

2.2.1.2 Scanphase

Figure 2.9a) shows aperture angle contours in the *xy*-plane. The aperture angle can have values in the range [0 π]. The aperture angle contours, which describe possible emitter positions for given aperture angle, are symmetric both around the sensor line and the middle line. This symmetry creates mirror images under certain conditions.

Figure 2.9b) shows scanphase localization. Two measurements at different sensor positions give *two or more* possible emitter positions. At least three measurements at different sensor positions are therefore required to locate the emitter unambiguously. This can be obtained by moving the sensors or by using more than two sensors (three is sufficient if the sensors are not placed along the same line). For the scanphase method the symmetry of the system creates a mirror image if the sensors move only along the sensor line *or* only along the middle line. In order to resolve this ambiguity, either the radar rotation direction must be known or some measurements must be performed when the sensors are located away from the sensor line *or* the middle line.



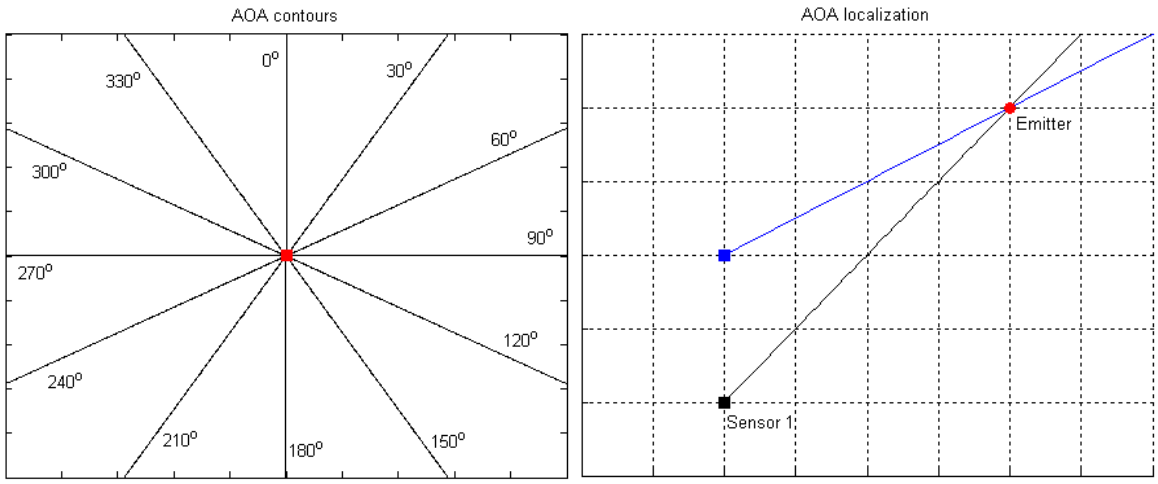
a) b)

Figure 2.9 The scanphase method. Left figure shows aperture angle contours in the xy-plane. Right figure shows scanphase localization. Each measurement is represented by a different colour (black, blue, and green). The dashed green line is an alternative to the green solid line and demonstrates the creation of mirror images.

2.2.1.3 Angle of arrival (AOA)

Figure 2.10a) shows AOA contours in the *xy*-plane. The AOA can have values in the range [0º 360º>.

Figure 2.10b) shows AOA localization. At least two measurements at different sensor positions are required to locate the emitter. This can be obtained by moving the sensor or by using more than one sensor. Note that the AOA method does not create mirror images.



a) b)

Figure 2.10 The AOA method. Left figure shows angle of arrival contours in the xy-plane. Right figure shows AOA localization. Each measurement is represented by a different colour (black and blue).

2.2.1.4 Summary

Both the TDOA method and the scanphase method have inherent symmetries and therefore create mirror images. These two methods require at least three measurements at different sensor positions to geolocate the emitter unambiguously. The AOA method, on the other hand, does not create mirror images and requires only two measurements at different sensor positions to geolocate the emitter. If two sensors are used, the AOA method allows for *stationary* sensors while both the TDOA method and the scanphase method require *moving* sensors.

2.2.2 Geometry and sensor movement

The methods' ability to geolocate emitters depends on geometry and sensor movement.

2.2.2.1 Time difference of arrival (TDOA)

We saw in the previous section that sensor movement along the sensor line gives ambiguous emitter position independent of the number of measurements for the TDOA method. Further, Figure 2.8a) in the previous section shows that an emitter located on the external sensor line can *not* be geolocated if the sensors move only along this line. This information is summarized in Figure 2.11a).

Figure 2.8a) also shows that an emitter located on the middle line can *not* be geolocated if the sensors move next to each other only along this line. This information is summarized in Figure 2.12a).

An emitter located on the line along which the sensors move (except when this line is equal to the sensor line or the middle line), can only be geolocated if the emitter is close to the sensors. This is pictured in Figure 2.13a). The reason for this is the following: At long distances from the sensors the emitter is always close to the asymptote of the hyperbola that describes the possible emitter positions. At the same time, when the emitter is located on the line that the sensors move along (while being far from the sensors) this line does in fact become the asymptote for *all* the sensor positions, see Figure 2.14a). As a result, the geolocation accuracy is very poor along this line, except very close to the sensors.

Geolocation is also poor along the central sensor line (see Figure 4.1 for definition), except close to the sensors. This is illustrated in Figure 2.12a) and Figure 2.13a). The reason is that the hyperbolas that describe the possible emitter positions intersect at small angles along this line, see Figure 2.15a).

Finally, consider the case when two sensors move on a circular path opposite to each other and at a constant distance from each other. This situation is pictured in Figure 2.15b). One might think that this type of sensor movement would be beneficial, since the lines where geolocation is not possible or very poor will be removed. However, as can be seen from Figure 2.15b), the hyperbolas that describe the possible emitter positions will intersect with small angles for all emitter positions (outside the circle described by the sensor movement) in this case, i.e., the

geolocation accuracy will instead be poor everywhere. Circular sensor movement is therefore not beneficial for the TDOA method.

2.2.2.2 Scanphase

We saw in the previous section that sensor movement along the middle line gives ambiguous emitter position independent of the number of measurements for the scanphase method. This information is summarized in Figure 2.12b).

Similarly, we saw that sensor movement along the sensor line gives ambiguous emitter position independent of the number of measurements. Further, Figure 2.9a) in the previous section shows that an emitter located on the external sensor line can *not* be geolocated if the sensors move only along this line. This information is summarized in Figure 2.11a). This is similar to what we saw for the TDOA method and is true also if the two methods are combined.

The geolocation accuracy is very poor for an emitter located on the line along which the sensors move (except when this line is equal to the sensor line, in which case geolocation is not possible at all). This is pictured in Figure 2.12b) and Figure 2.13b). The reason for this is that the circles that describe the possible emitter positions do not intersect in this case but rather touch each other as tangents, see Figure 2.14b).

Finally, consider the case when two sensors move on a circular path opposite to each other and at a constant distance from each other. This situation is pictured in Figure 2.16a). The circles that describe the possible emitter positions intersect at large angles for all emitter positions, resulting in good geolocation accuracy everywhere. Circular sensor movement could therefore be beneficial for the scanphase method. This type of movement does not generate any lines where geolocation is very poor or not possible. Note, however, that an mirror image is created opposite the emitter.

2.2.2.3 Angle of arrival (AOA)

Figure 2.10a) shows that an emitter located on the line along which the sensor moves (but *not between* the sensor's start and end position) can not be geolocated. If two sensors are used, an emitter can not be geolocated if it is located on the external sensor line and the sensors are stationary or move only along this line. This is illustrated in Figure 2.11b). This is also true if the AOA method is combined with either the TDOA method or the scanphase method or both.

Finally, consider the case when the sensor moves on a circular path. This is pictured in Figure 2.16b). The lines that describe the possible emitter positions intersect at angles comparable to the angles we would get if the sensor moved along a straight line. However, geolocation is now possible everywhere. Circular sensor movement could therefore be beneficial for the AOA method when only one sensor is used.

2.2.2.4 Summary

The AOA method, or the AOA method in combination with either the TDOA method or the scanphase method or both, gives unambiguous geolocation everywhere, except on the external sensor line where geolocation is not possible if the sensors are stationary or move only along this line.

The TDOA method and the scanphase method give ambiguous geolocation when the sensors move only along the sensor line. The scanphase method also gives ambiguous geolocation when the sensors move only along the middle line. For both methods it is not possible to geolocate emitters that are located on the line along which the sensors move.

If the TDOA and scanphase methods are combined, geolocation is unambiguous everywhere except when the sensors are stationary or move only along the sensor line. When this is the case, geolocation is ambiguous and it is not possible to geolocate an emitter on the external sensor line.

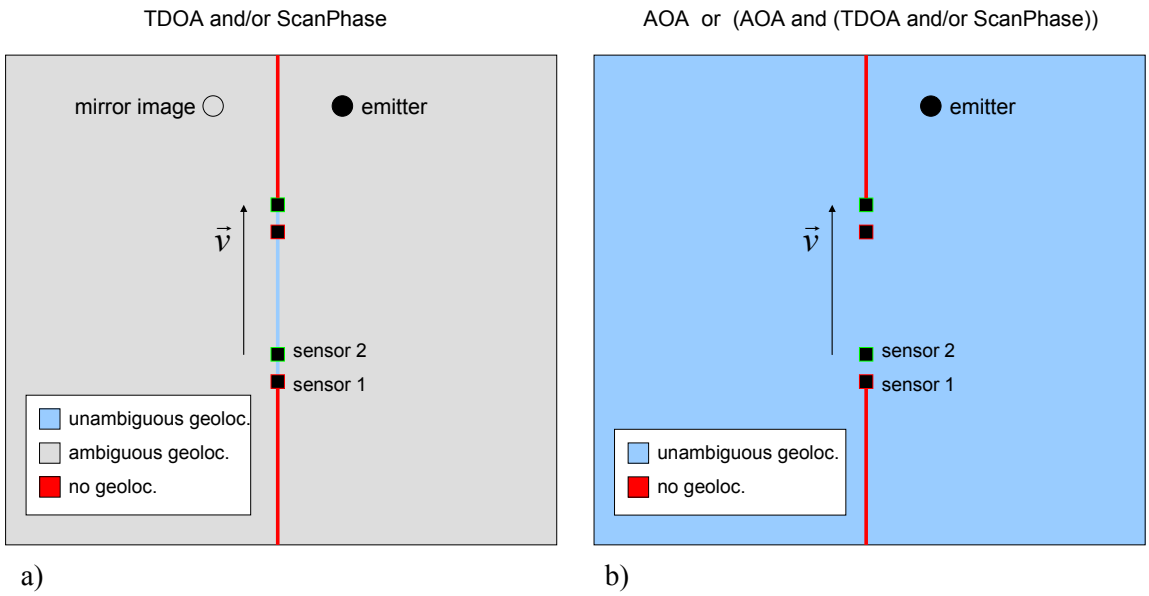


Figure 2.11 Sensors moving after each other along a straight line. This movement of the sensors affects the geolocation for all methods and combinations of methods.

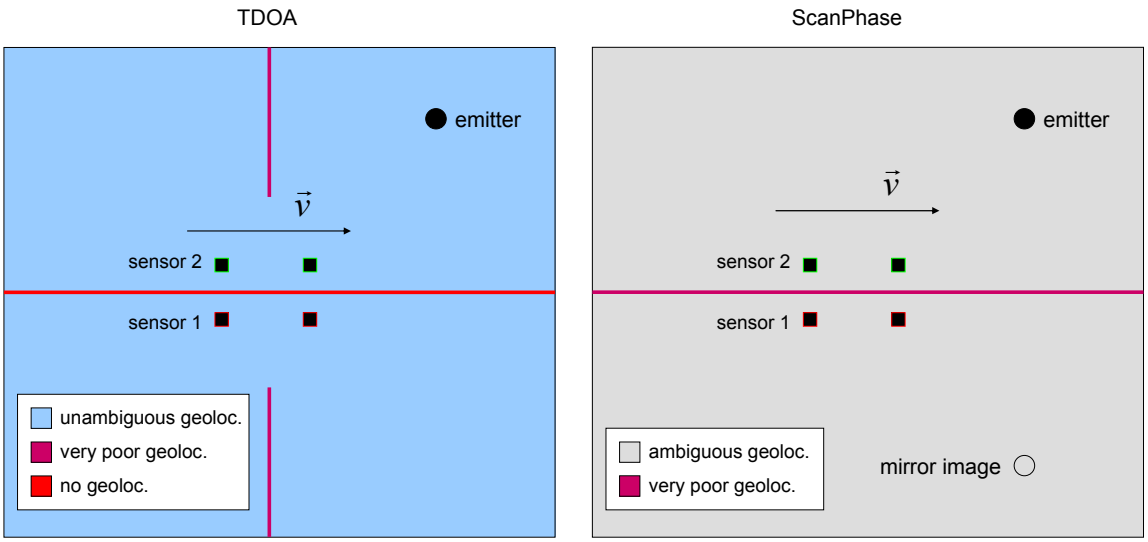


Figure 2.12 Sensors moving next to each other along a straight line. This movement of the sensors affects the geolocation for the TDOA method and the scanphase method.

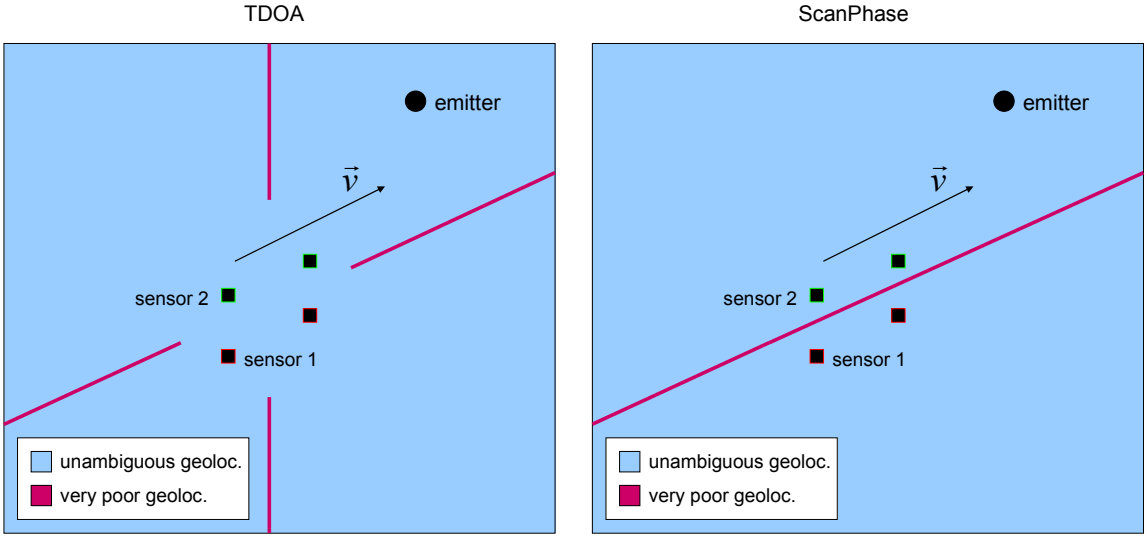
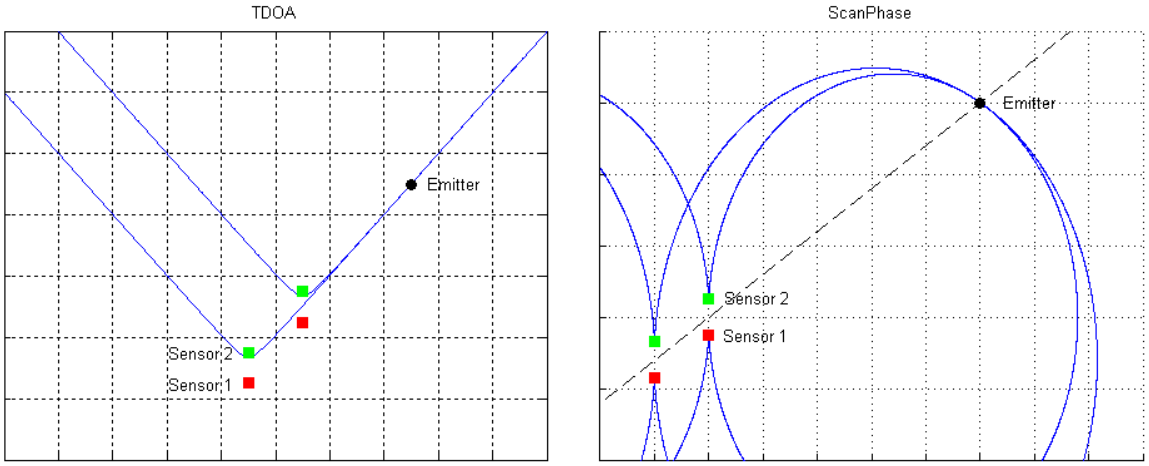
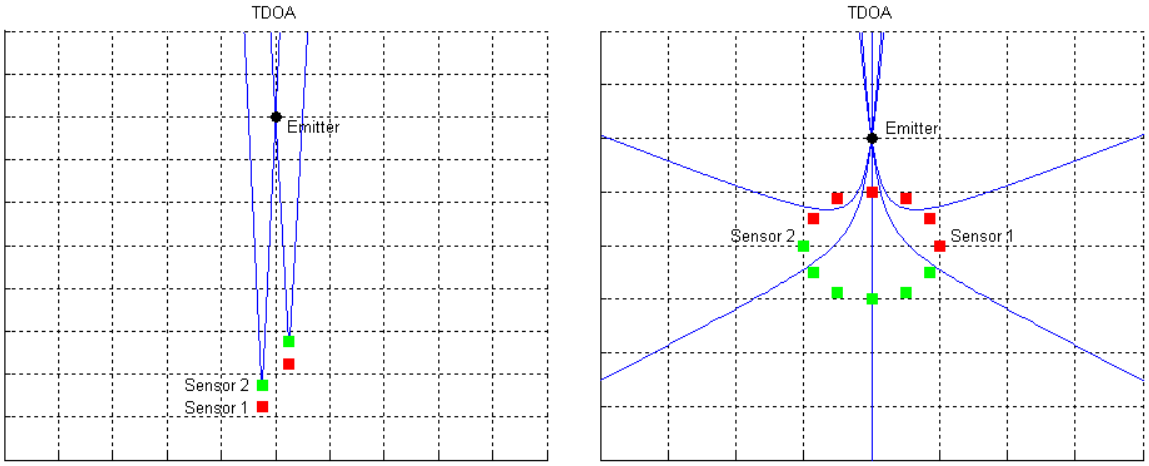


Figure 2.13 Sensors moving along a tilted straight line. This movement of the sensors affects the geolocation for the TDOA method and the scanphase method.



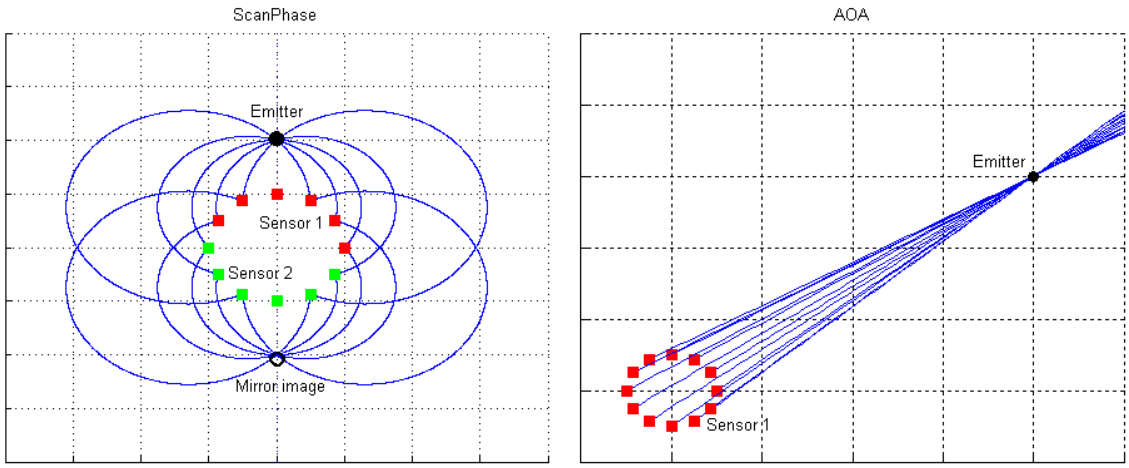
a) b)

Figure 2.14 Geolocation of emitter that is located on the line along which the sensors move. Left figure shows the geometry for the TDOA method. Right figure shows the geometry for the scanphase method.



a) b)

Figure 2.15 Left figure shows geolocation with the TDOA method of an emitter that is located on the central sensor line. Right figure shows geolocation with the TDOA method when the sensors move on a circular path opposite each other.



a) b)

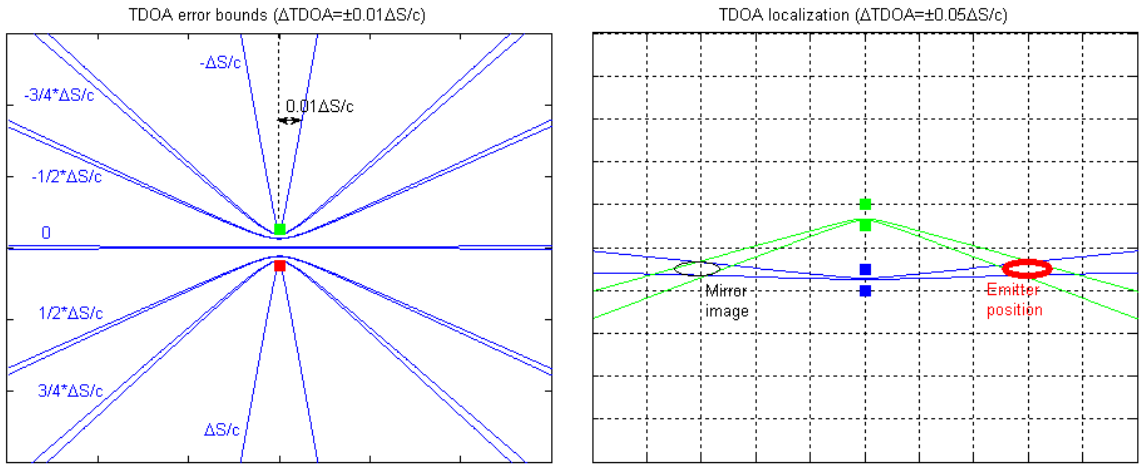
Figure 2.16 Left figure shows geolocation with the scanphase method when the sensors move on a circular path opposite each other. Right figure shows geolocation with the AOA method when the sensor moves on a circular path.

2.2.3 Error bounds

The different localization methods are assosiated with errors in the measured parameters (TDOA, aperture angle or angle of arrival) which results in errors in the geolocation of the emitter. Figure 2.17-Figure 2.19 show error bounds for the different methods.

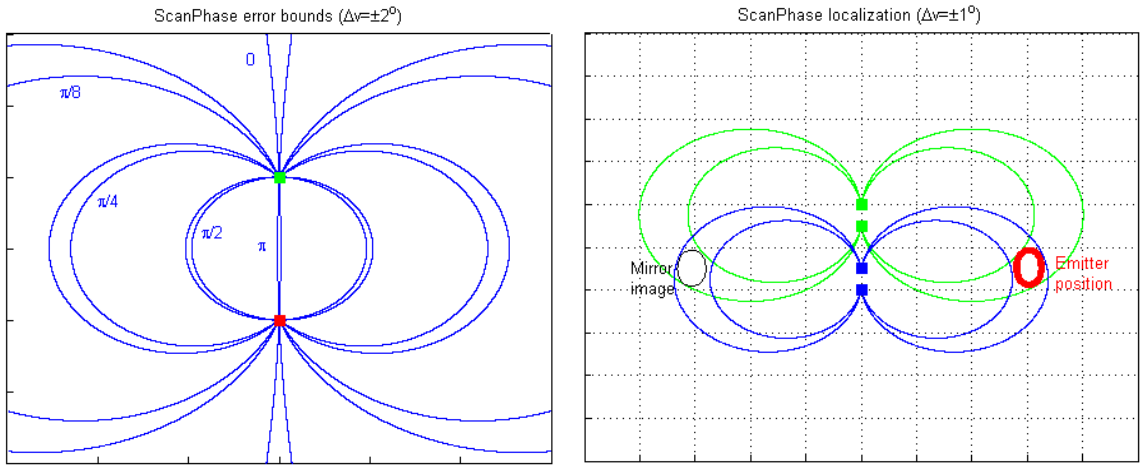
For all three methods the geolocation error increases with increasing distance from the sensors (for given change in TDOA, aperture angle or angle of arrival), see Figure 2.17a), Figure 2.18a), and Figure 2.19a). For the TDOA method the position error is largest close to the sensor line and decreases towards the middle line.

Figure 2.17b), Figure 2.18b), and Figure 2.19b) show examples of the resulting geolocation error for each of the three methods. The geolocation error depends on both the error (variance) in the measured parameter and the geometry of the system (sensor and emitter positions). When all three methods are combined, the emitter position can be uniquely determined with only one set of measurements, see Figure 2.20. However, the geolocation error decreases if the number of measurements is increased.



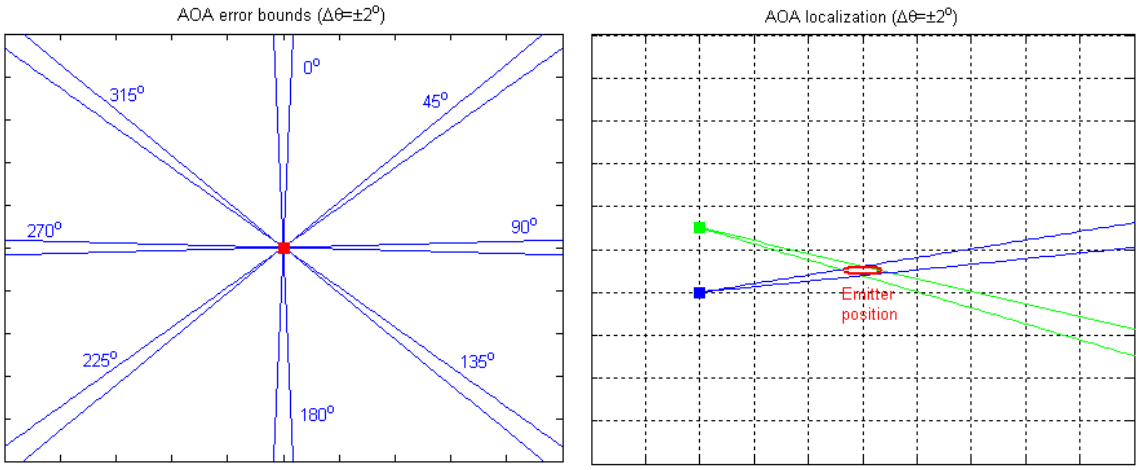
a) b)

Figure 2.17 Error bounds for the TDOA method. Here ΔS is the distance between the sensors and c is the speed of light.



a) b)

Figure 2.18 Error bounds for the scanphase method.



a) b)

Figure 2.19 Error bounds for the AOA method.

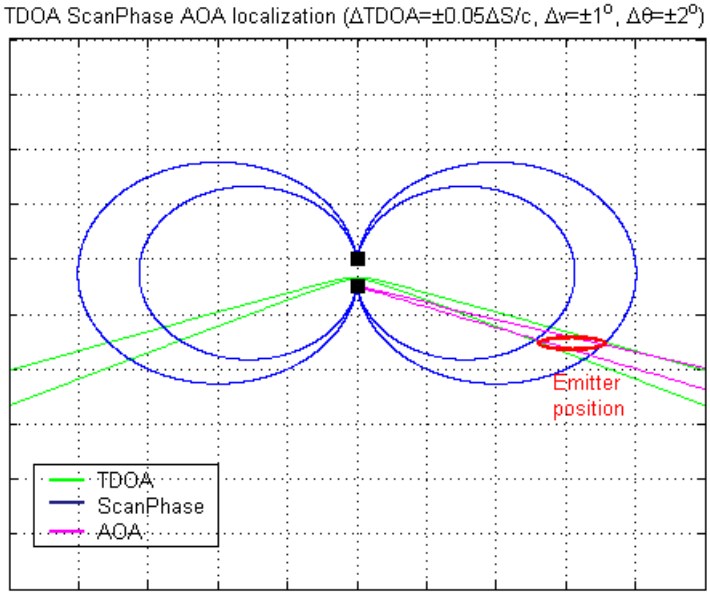


Figure 2.20 Geolocation with error bounds when all three methods are combined. Here ΔS is the distance between the sensors and c is the speed of light. The TDOA error is given as a fraction of the distance between the sensors, ΔTDOA=±0.05ΔS/c. For instance, if ΔS=30 km this corresponds to ΔTDOA=6 μs.

2.3 Error estimates

Geolocation error estimates can be found from the Cramer-Rao Lower Bound (CRLB) or the Circular Error Probable (CEP). The CRLB (Section 2.3.1) is the theoretical lower bound for the geolocation error. Good geolocation estimators should be able to achieve geolocation accuracies close to this value. The CRLB can be visualized as an error *ellipse* (2D) and gives information about both the size and the direction of the error. The CEP (Section 2.3.2) is calculated from the CRLB and can be visualized as an error *circle* (2D). It preserves the statistical properties, but does not give directional information. Figure 2.21 shows a contour plot based on values for the CEP-radius together with the corresponding CRLB error ellipses at selected emitter positions.

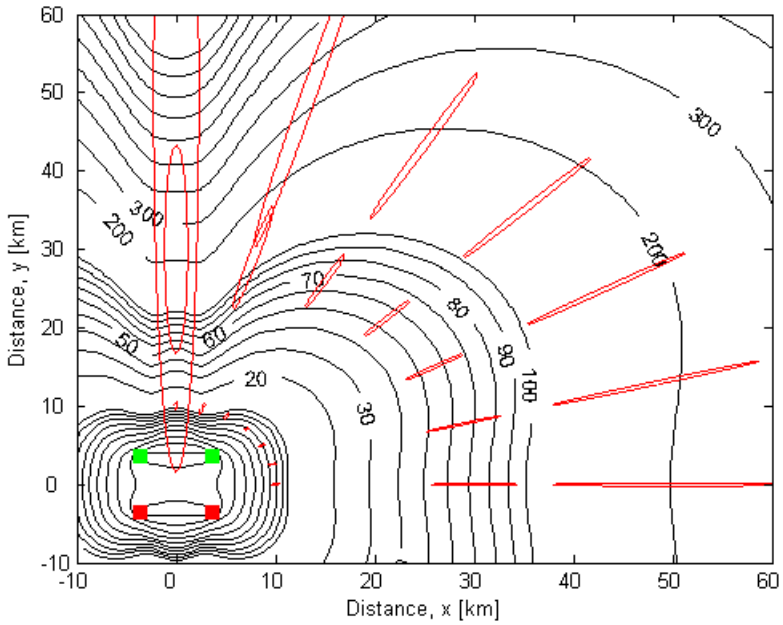


Figure 2.21 CEP-contours (black) together with CRLB error ellipses (red) at selected emitter positions. All three localization methods were combined for the calculations. The sensors (red and green squares) were placed 7.2 km apart and moved next to each other along a straight line for 7.2 km. A total of 361 sensor positions were included in the calculations (only start and end positions shown). The TDOA, aperture angle, and angle of arrival errors were set to respectively 50 ns, 1°, and 2°. The error ellipses in the figure are enlarged 40 times.

2.3.1 Cramer-Rao Lower Bound (CRLB)

Cramer-Rao Lower Bound (CRLB) is the theoretical lower bound for the geolocation error. Its value depends upon:

1) The geometry of the system; emitter and sensor position(s)
2) The variances σ^2 for the measured parameters
3) The number of measurements N

In addition, the relation between the measured parameter $\tilde{S}$ (TDOA, aperture angle, angle of arrival) and the emitter and sensor position(s) $\vec{p}_e$ and $\vec{p}_i$ must be known:

$$\tilde{S} = S(\vec{p}_e, \vec{p}_i) \tag{2.8}$$

so that the partial derivatives $\partial\tilde{S} / \partial\vec{p}_e$ can be calculated either analytically or numerically. These relations are given by Equation (2.1) for the TDOA, Equation (2.3) for the aperture angle, and Equation (2.6) for the angle of arrival.

The CRLB can be determined in 1, 2, or 3 dimensions. In the 1-dimensional case the CRLB defines an error *line*, in the 2-dimensional case an error *ellipse*, and in the 3-dimensional case an error *ellipsoid*. The 1-dimensional case is presented in Appendix D and illustrates the principles behind the method.

In this report we study the CRLB in 2 dimensions. The size and orientation of the CRLB error ellipse can be determined from the CRLB covariance matrix $\mathbf{K}$, by solving the corresponding eigenvalue problem[1]

$$\mathbf{K} \cdot \vec{X} = \omega^2 \vec{X} \tag{2.9}$$

where ω^2 is the eigenvalue and $\vec{X}$ is the eigenvector.

The CRLB covariance matrix $\mathbf{K}$ is given by (valid also for the 1-dimensional and 3-dimensional cases)

$$\mathbf{K} = \left[\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^{\mathbf{T}} \right]^{-1} \tag{2.10}$$

and is for the 2-dimensional case a 2×2 matrix.

$\mathbf{A}$ is the $N \times N$ variance matrix given by

$$\mathbf{A} = \begin{pmatrix} \sigma_1^2 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & & \sigma_n^2 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \sigma_N^2 \end{pmatrix} \tag{2.11}$$

where σ_1^2 is the variance for the n'th measurement (TDOA, aperture angle or angle of arrival) and N is the total number of measurements.

[1] For a rigorous mathematical derivation of the CRLB covariance matrix, see Appendix A in [5].

$\mathbf{B}$ is the $2 \times N$ (2-dimensional case) partial derivatives matrix given by

$$\mathbf{B} = \begin{pmatrix} \delta x_1 & \dots & \delta x_n & \dots & \delta x_N \\ \delta y_1 & \dots & \delta y_n & \dots & \delta y_N \end{pmatrix} \tag{2.12}$$

where $\delta x_n = \partial S_n / \partial x_e$ and $\delta y_n = \partial S_n / \partial y_e$ are the partial derivatives for the n'th measurement (TDOA, aperture angle or angle of arrival). Analytical expressions for the partial derivatives are given in Appendix C.

The orientation and size of the semi axes of the error ellipse are given by the eigenvectors (orientation) and eigenvalues (size), see Figure 2.22, and can be expressed by the matrix $\mathbf{D}$

$$\mathbf{D} = \mathbf{X} \cdot \sqrt{\mathbf{C}} \tag{2.13}$$

Here $\mathbf{X}$ is the eigenvector matrix where the *i*'th column corresponds to the *i*'th eigenvector $\vec{X}_i$ and $\mathbf{C}$ is the eigenvalue matrix

$$\mathbf{C} = \begin{pmatrix} \omega_1^2 & 0 \\ 0 & \omega_2^2 \end{pmatrix} \tag{2.14}$$

where ω_i^2 is the *i*'th eigenvalue.

The vector to a point on the CRLB error ellipse can then be found from

$$\begin{bmatrix} x(\beta) \\ y(\beta) \end{bmatrix} = \mathbf{D} \cdot \begin{bmatrix} \cos\beta \\ \sin\beta \end{bmatrix} \tag{2.15}$$

where β is the angle to a specific point on the unit circle, see Figure 2.22. The probability that an emitter is located within a 1σ CRLB error ellipse is 39.4%, see Table 2.2.

	CRLB probability		
	1σ	**2σ**	**3σ**
1D	68.3%	95.5%	99.7%
2D	39.4%	86.5%	98.9%
3D	20.0%	73.9%	97.1%

Table 2.2 The probability that an emitter is located within a 1σ, 2σ, or 3σ error line (1D), error ellipse (2D) or error ellipsoid (3D).

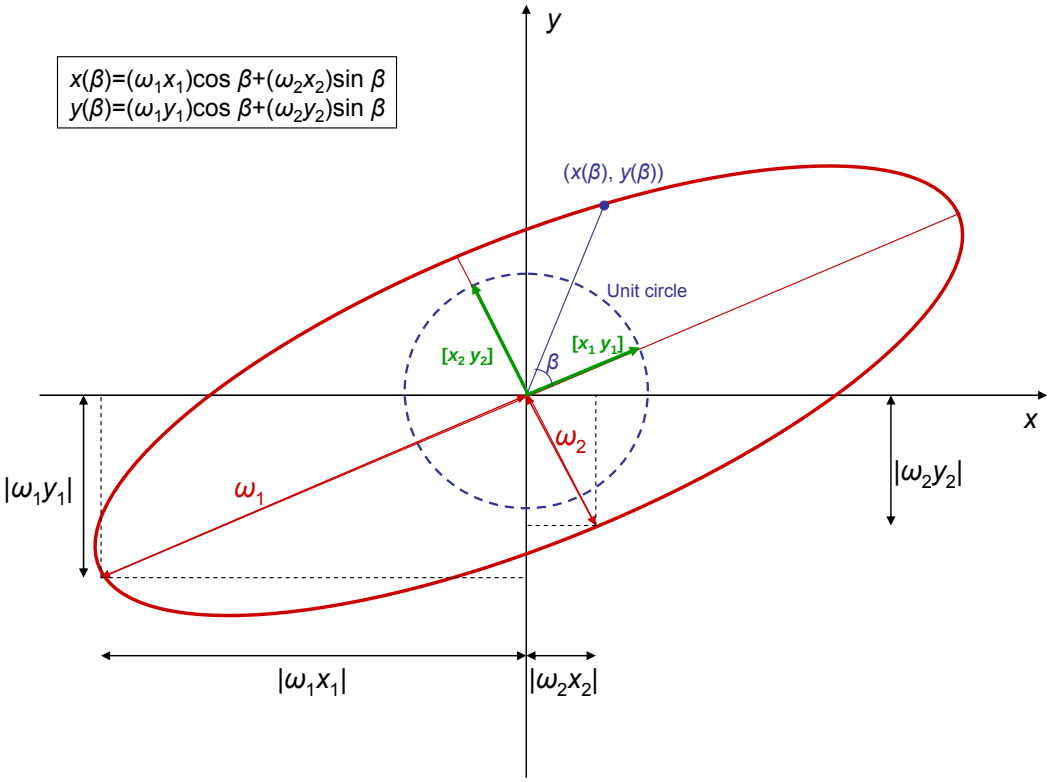


Figure 2.22 CRLB error ellipse. The orientation of the error ellipse is given by the eigenvectors $[x_1\ y_1]$ and $[x_2\ y_2]$, and the shape is determined by the eigenvalues ω_1 and ω_2 (major and minor semi-axes).

2.3.2 Circular Error Probable (CEP)

Circular Error Probable (CEP) is defined as the probability that a random realization of a two dimensional vector lies inside a circle with a given radius. CEP circles do not give direction information, but preserve the statistical properties, see Figure 2.23. The CEP can be found from the CRLB covariance matrix and is given by [2]:

$$P(R,\omega_1,\omega_2)=\frac{1}{2\pi\omega_1\omega_2}\iint_{x^2+y^2\le R^2} dxdy\cdot\exp\left(-\frac{\left(\frac{x}{\omega_1}\right)^2+\left(\frac{y}{\omega_2}\right)^2}{2}\right) \tag{2.16}$$

where ω_1^2 and ω_2^2 are the eigenvalues of the CRLB covariance matrix, and P is the probability that a random point will fall into a circle of radius R.

Substituting $x/\omega_1 = r\cos\theta$ and $y/\omega_2 = r\sin\theta$ into Equation (2.16) gives [3]

$$P(R,\omega_1,\omega_2)=\frac{1}{2\pi}\iint_{r^2\le R^2/(\omega_1^2\cos^2\theta+\omega_2^2\sin^2\theta)} r\cdot\exp\left(-\frac{r^2}{2}\right)drd\theta \tag{2.17}$$

Integrating over r further gives

$$P(R,\omega_1,\omega_2) = \frac{1}{2\pi}\int_0^{2\pi} d\theta \left[-\exp\left(-\frac{r^2}{2} \right) \right]_{r=0}^{r=R/\sqrt{\omega_1^2\cos^2\theta+\omega_2^2\sin^2\theta}} \\ = 1 - \frac{2}{\pi}\int_0^{\pi/2} f(\theta)d\theta \tag{2.18}$$

where

$$f(\theta) = \exp\left(-\frac{R^2}{2\left(\omega_1^2\cos^2\theta+\omega_2^2\sin^2\theta\right)} \right) \tag{2.19}$$

Since $f(\theta)$ is a periodic analytic function, the integral in Equation (2.18) can be evaluated to high accuracy using the standard trapezoidal rule [4]. This gives for the CEP:

$$P(R,\omega_1,\omega_2) = 1 - \frac{1}{N}\left[\tfrac{1}{2}f(\theta_0) + \tfrac{1}{2}f(\theta_N) + \sum_{k=1}^{N-1} f(\theta_k) \right] \tag{2.20}$$

where

$$\theta_k = \frac{k}{N}\cdot\frac{\pi}{2}, \qquad k = 0,1,2,\ldots,N \tag{2.21}$$

and N is the number of subdivisions of the interval $[0,\pi/2]$.

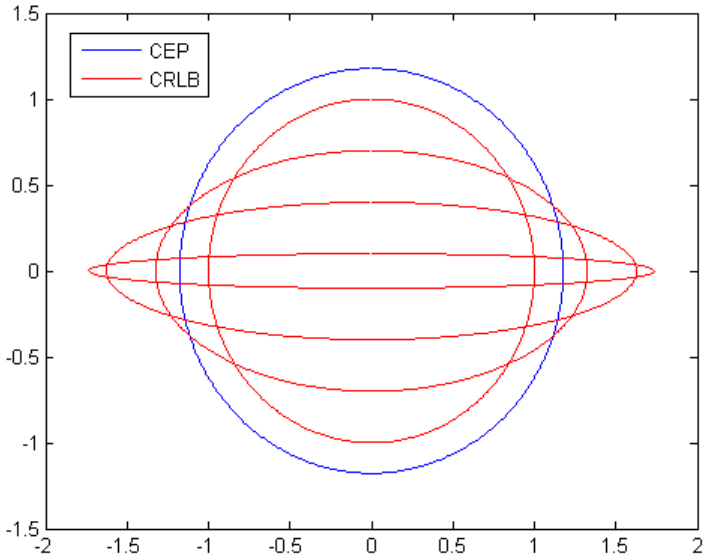


Figure 2.23 Example of different CRLB error ellipses (1σ) that all correspond to the same CEP circle (50% probability).

3 Simulations

In this chapter we describe the simulations that were performed as basis for the geolocation accuracy analyses. The results are presented in Chapter 4. Software for the simulations is written in Matlab and is given in Appendix E.

Before performing the simulations we should know the approximate range of the sensors. Radar waves do not travel along perfectly straight lines, but curve somewhat along the Earth surface. As a result, it is possible to see radars at some distance behind the visual horizon. For a sensor at 50 m height above the sea surface, the radar horizon is at approximately 30 km distance. If the emitter is at the same height as the sensor, this increases to 60 km distance, see Figure 3.1. If conditions are favourable, the radar signal may be subject to 'ducting', in which case the signal is trapped close to the surface and may travel long distances following the Earth's curvature. When this happens, the sensor range increases compared to what is shown here.

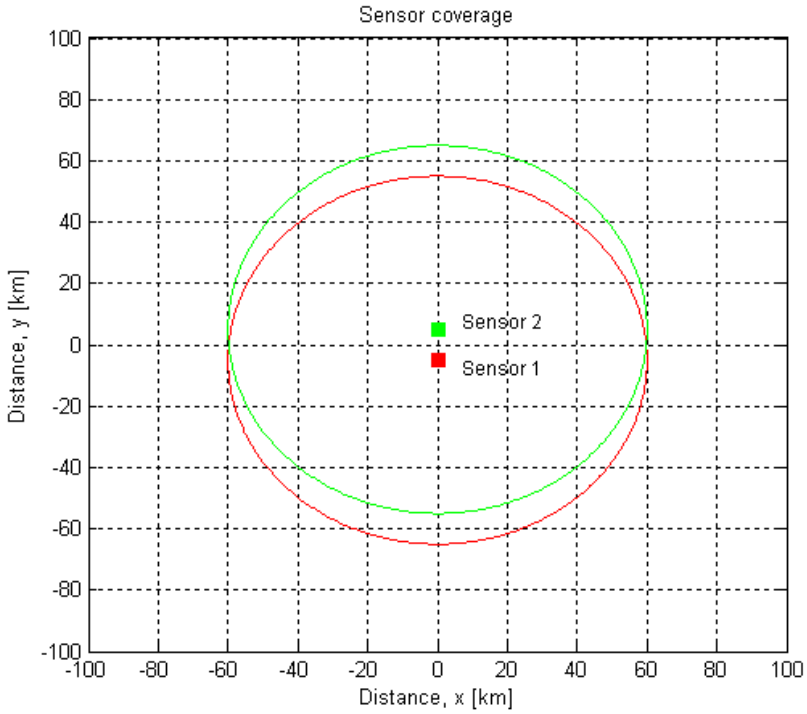


Figure 3.1 Sensor coverage. The red and green circles mark the radar horizon for each of the sensors.

We study three different localization methods; Time difference of arrival (TDOA), scanphase, and angle of arrival (AOA). The scanphase method depends on the emitters having rotating radars. We have assumed a radar rotation period of 2.5 s (24 rpm) for the calculations. One measurement can be made per mainlobe pass, i.e., per radar rotation period, and we have therefore defined a measurement period to be 2.5 s long. The TDOA and AOA methods can use pulses from both the mainlobe and the sidelobes. We therefore define a TDOA (or angle of

arrival) measurement to be the average value for the TDOA (or angle of arrival) measured over the measurement period. The measurements are associated with corresponding variances. We have assumed that the TDOA variance is $(50\ \text{ns})^2$, the aperture angle variance is $(1^\circ)^2$, and the angle of arrival variance is $(2^\circ)^2$, see Table 3.1. An error in the TDOA of 50 ns corresponds to an error in the measured pulse time of arrival (TOA) of about 35 ns (on each of the two sensors), which can be obtained with a very precise clock such as for instance a rubidium clock.

The method (CRLB) for determining the geolocation *accuracy* does not distinguish between a scenario where all measurements are made simultaneously by many sensors or a scenario where the measurements are made one after the other by one (or a pair of) sensor(s) over a finite time period. Hence, neither observation time nor number of sensors (only number of sensor positions) are directly relevant for the calculations. However, in order to keep the simulation results realistic for a maritime scenario, we will make some assumptions both about the number of sensors used, the observation time and other relevant parameters.

We have set the maximum acceptable observation time to be 15 min, and the maximum vessel (sensor) speed to be 8 m/s (16 knots). The maximum distance a sensor can move during the observation time is then 7.2 km. The maximum distance per measurement period is 20 m and the maximum number of measurement periods is 360. Values for all key parameters are listed in Table 3.1.

Radar rotation period	2.5 s (24 rpm)
Measurement period	2.5 s
Maximum observation time	15 min
Maximum number of measuremement periods	360
Maximum vessel (sensor) speed	8 m/s (16 knots)
Maximum sensor movement during the measurement period	20 m
Maximum sensor movement during the observation time	7.2 km
Variance, TDOA	$(50\ \text{ns})^2$
Variance, aperture angle	$(1^\circ)^2$
Variance, angle of arrival	$(2^\circ)^2$

Table 3.1 Key parameters for the simulations.

We have investigated the following methods or combination of methods:

1. AOA (single sensor)
2. Scanphase
3. TDOA
4. (2x)AOA
5. (2x)AOA & Scanphase
6. TDOA & Scanphase

7. TDOA & (2x)AOA
8. TDOA & (2x)AOA & Scanphase

The TDOA and scanphase methods require the use of two sensors, while AOA can be performed with only one sensor. For the AOA method we have therefore looked at what can be achieved both with two sensors and with only one sensor. When the AOA method is combined with one of the other methods, two sensors are always used. Note that for AOA with two sensors a measurement period contains two measurements, while for the other two methods (or AOA with only one sensor) a measurement period contains only one measurement.

The sensors can be placed at different distances from each other and may be stationary or move with respect to the emitter. For AOA (single sensor), scanphase or TDOA the sensors must always move some distance in order to obtain a geolocation for the emitter, see Section 2.2. For AOA with two sensors, or a combination of two or more methods, the sensors may be stationary during the observation period. A single measurement period is then sufficient to make a geolocation for the emitter, i.e., momentaneous geolocation can be performed. We assume that the emitter does not move during the observation period.

Each calculation is done for 1 emitter position (stationary emitter), m sensor (pair) positions, and N measuremements with corresponding variances. The sensor positions can be the same for all m (stationary sensors) or vary for different m (moving sensors). The number of measurement periods equals the number of sensor positions m, while the number of measurements is equal to or larger than the number of sensor positions, i.e., $N \geq m$, depending on which method or combination of methods are used. For instance, the combination TDOA &(2x)AOA gives three measurements per sensor position, i.e., $N = 3m$.

Input and output from the simulations are:

Input:

- Emitter position, (x_e, y_e)
- Sensor 1 position(s), (x_1, y_1)
- Sensor 2 position(s), (x_2, y_2)
- Variance σ^2 for each measured parameter (TDOA, aperture angle or angle of arrival)

Output:

- CEP-radius at the emitter position
- Major semi-axis of the CRLB error ellipse at the emitter position
- Minor semi-axis of the CRLB error ellipse at the emitter position

4 Results and discussion

We will in this chapter present and discuss the results from the simulations of the geolocation accuracy.

Section 4.1 presents the general results for the geolocation accuracy for the different methods or combinations of methods and for different geometries and sensor movements. Section 4.2 discusses CRLB error ellipses relative CEP-radius and Section 4.3 presents results for the instantaneous geolocation accuracy when more than two sensors are used. Section 4.4 discusses the geolocation accuracy dependency on different parameters and Section 4.5 discusses additional factors that may affect the geolocation accuracy. Trajectory recommendations are given in Section 4.6. Finally, Section 4.7 describes further work.

The results in this chapter are mainly presented as CEP-contour plots, where the CEP-contours give the CEP-radius (50% probability) in meters and are shown on intervals [1, 2, ... 10], [10, 20, ... 100], [100, 200, ... 1000], and so on. Sensor positions are marked by red and green squares and only the start and end positions are shown.

The following symbols are used in the figures:

Sensor distance:	ΔS
Trajectory length:	ΔL
Number of sensor (pair) positions:	m
Number of measurements:	$N=k \cdot m$
Error in measured TDOA:	$\Delta TDOA$
Error in measured aperture angle:	$\Delta \nu$
Error in measured angle of arrival:	$\Delta \theta$

The value of k depends on the method or combination of methods used:

AOA (single sensor):	$k=1$
TDOA:	$k=1$
Scanphase:	$k=1$
AOA:	$k=2$
TDOA & Scanphase:	$k=2$
AOA & Scanphase:	$k=3$
TDOA & AOA:	$k=3$
TDOA & AOA & Scanphase:	$k=4$

In addition, the terms ”central sensor line”, ”central middle line”, and ”tilted (45º) line are used in the discussion. These terms are defined in Figure 4.1.

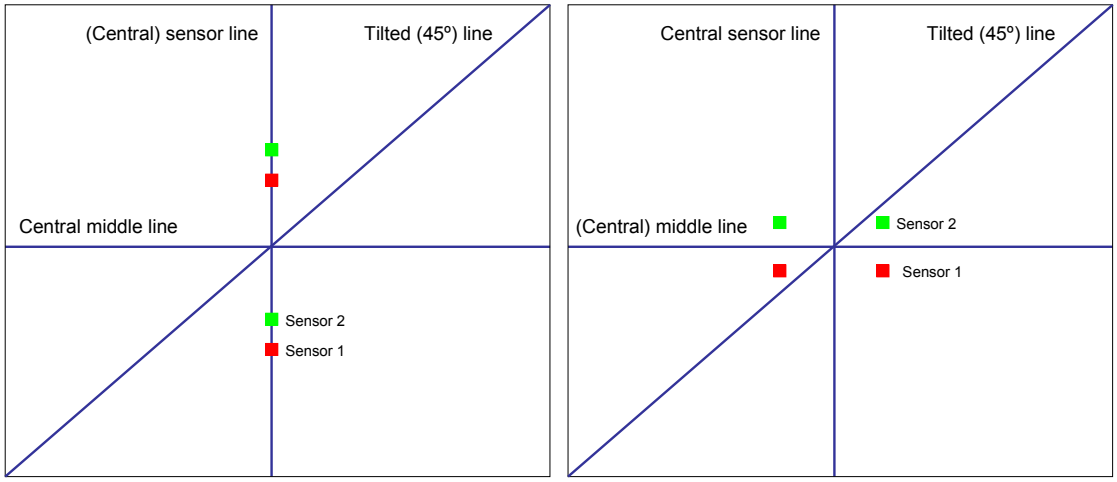


Figure 4.1 Central sensor line, central middle line, and tilted (45°) line.

4.1 Characteristics of the methods and geolocation accuracy

We first performed broad general simulations where we studied the following cases for the different methods and combinations of methods:

- Sensors moving *after* each other
- Sensors moving *next* to each other
- Sensors moving on a *circular* path
- *Stationary* sensors
- *Single* measurement

The CEP-radius (50% probability) was calculated at different emitter positions (2-dimensional grid with 1 km between the grid points) within a 200×200 km large area around the sensors. Based on this CEP-radius contour plots were generated, showing the geolocation accuracy that can be obtained at different points around the sensors. These results are shown in Figure 4.2-Figure 4.16. A summary of the findings are given in Table 4.8.

4.1.1 TDOA

The TDOA method requires the use of 2 moving sensors. We will discuss the following cases:

1) Sensors moving *next to* each other
2) Sensors moving *after* each other
3) Sensors moving along a *tilted* line
4) Sensors moving along a *zig-zag* path

The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 361. The error in the measured TDOA is set to 50 ns.

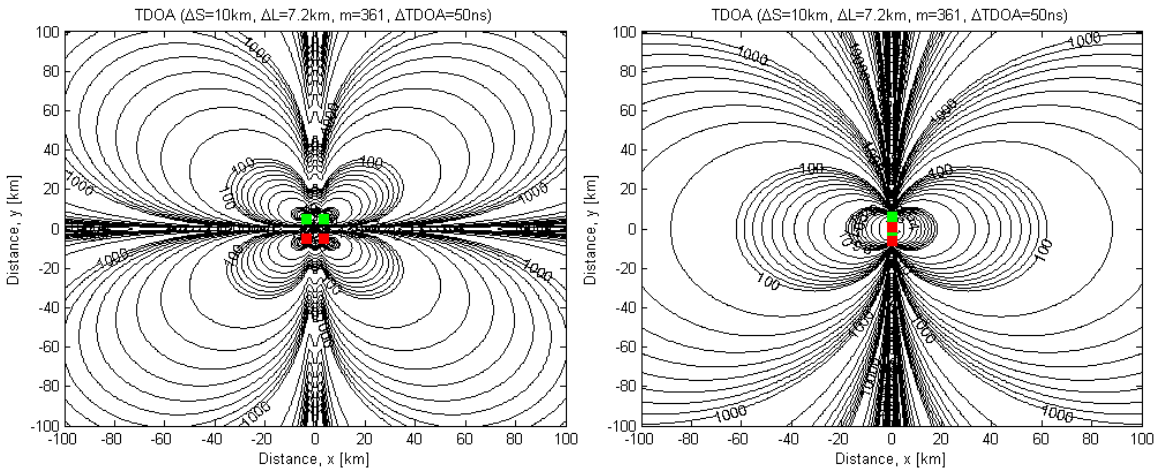


Figure 4.2 CEP-contours for the TDOA method. Left figure shows sensors moving next to each other. Right figure shows sensors moving after each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 361. The error in the measured TDOA is set to 50 ns.

Figure 4.2 (left) shows CEP-contours when sensors move *next to* each other. The CEP-contours are symmetric around the central sensor line and the central middle line. The geolocation accuracy is very poor along the central sensor line, while the error goes to infinity along the central middle line, i.e., geolocation is not possible here. This is consistent with the predictions in Section 2.2.2.1, see Figure 2.12a). The geolocation accuracy is best along the tilted (45º) line. Here the geolocation error is about 10 m at 10 km distance, increasing to 100 m at 50 km distance, and 500 m at 100 km distance from the sensors. No mirror image is created when the sensors move next to each other.

Figure 4.2 (right) shows CEP-contours when sensors move *after* each other. The CEP-contours are symmetric around the central sensor line and the central middle line. The geolocation error goes to infinity along the central sensor line, i.e., geolocation is not possible here. This is consistent with the predictions in Section 2.2.2.1, see Figure 2.11a). The geolocation accuracy is best along the central middle line. Here the geolocation error is about 5 m at 10 km distance, increasing to 70 m at 50 km distance, and 300 m at 100 km distance from the sensors. The geolocation accuracy is better when the sensors move after each other compared to when they move next to each other. This is consistent with earlier findings [8]. Note that when sensors move after each other a mirror image is created (Section 2.2.1.1).

Figure 4.3 (left) shows CEP-contours when sensors move along the *tilted* (45º) line. The geolocation accuracy is very poor along the central sensor line and along the line that the sensors move along. This is consistent with the predictions in Section 2.2.2.1, see Figure 2.13a). The CEP-contours appear as a somewhat distorted version of the case when the sensors move next to each other. Geolocation accuracies are close to what is achieved when sensors move after each other, with geolocation errors about 5 m at 10 km distance, increasing to 80 m at 50 km distance, and 300 m at 100 km distance from the sensors. No mirror image is created.

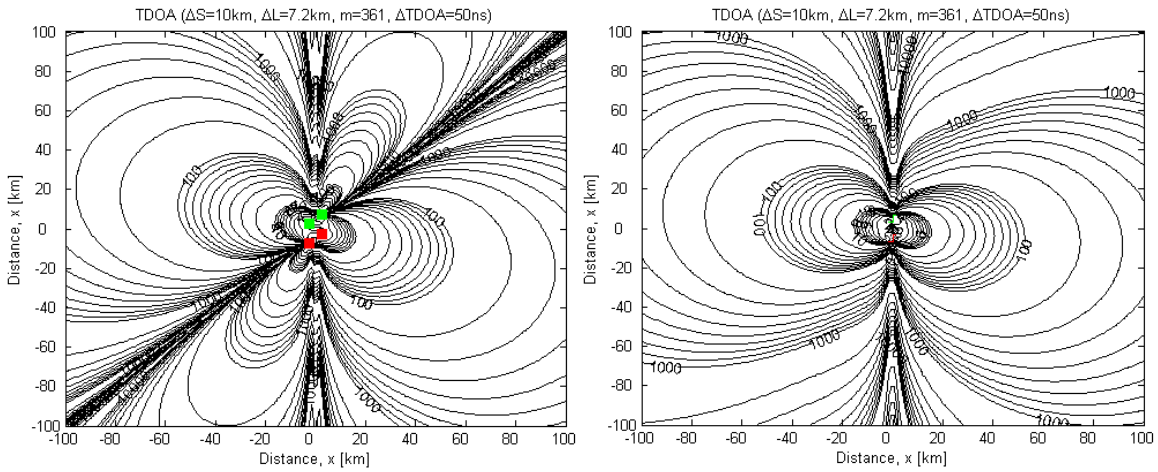


Figure 4.3 CEP-contours for the TDOA method. Left figure shows sensors moving along a tilted line. Right figure shows sensors moving along a zig-zag path. The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 361. The error in the measured TDOA is set to 50 ns.

Figure 4.3 (right) shows CEP-contours when sensors move along a *zig-zag* path[2]. The CEP-contours are similar to when sensors move after each other, only slightly rotated. Geolocation accuracies are close to what is achieved when sensors move after each other, with geolocation errors about 5 m at 10 km distance, increasing to 90 m at 50 km distance, and 400 m at 100 km distance from the sensors. Note, however, that geolocation along the central sensor line is now possible (though with very poor accuracy). This is because the sensor movement now is a combination of sensors moving next to each other and sensors moving after each other, bringing together the best from both. No mirror image is created.

Circular sensor movement was also investigated, but gave in general poor geolocation accuracy everywhere consistent with predictions in Section 2.2.2.1, see also Figure 2.15b).

4.1.1.1 Summary

The TDOA method requires the use of 2 moving sensors. Geolocation is not possible along the direction in which the sensors move. The geolocation accuracy is best when the sensors move after each other, but a mirror image is then created. In order to resolve the mirror image and improve the geolocation accuracy along the central sensor line a zig-zag path is recommended. The geolocation accuracies that can be obtained are then about 5 m at 10 km distance, 90 m at 50 km distance, and 400 m at 100 km distance from the sensors. Table 4.1 summarizes the results for the TDOA method.

[2] Sensors first move *next to* each other along a 1.8 km long straight line, then *after* each other along a 2x1.8 km long straight line, and then again *next to* each other along a 1.8 km straight line. The total trajectory is 7.2 km long, and the sensor distance is always 10 km.

Geometry/ trajectory	Mirror image	Line with no/poor geoloc.	Geolocation accuracy		
			10 km	50 km	100 km
Next to	No	2	10 m	100 m	500 m
After	Yes	1	5 m	70 m	300 m
Tilted	No	2	50 m	80 m	300 m
Zig-zag	No	1	5 m	90 m	400 m

Table 4.1 Summary of results for the TDOA method. The number of measurements is 361.

4.1.2 Scanphase

The scanphase method requires the use of 2 moving sensors. We will discuss the following cases:

1) Sensors moving *next to* each other
2) Sensors moving *after* each other
3) Sensors moving along a *tilted* line
4) Sensors moving along a *circular* path

The trajectory length is 7.2 km, the sensor distance is 10 km (except when sensors move along a circular path, in which case the sensor distance i 4.584 km in order to keep the trajectory length the same), and the number of measurements is 361. The error in the measured aperture angle is set to 1º.

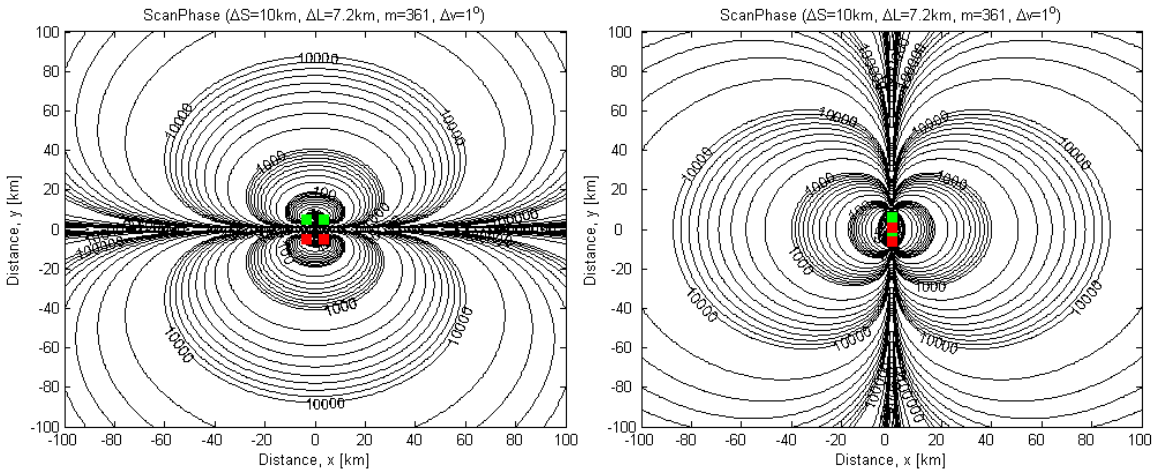


Figure 4.4 CEP-contours for the scanphase method. Left figure shows sensors moving next to each other. Right figure shows sensors moving after each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 361. The error in the measured aperture angle is set to 1°.

Figure 4.4 (left) shows CEP-contours when sensors move *next to* each other. The CEP-contours are symmetric around the central sensor line and the central middle line. Note that the geolocation accuracy is very poor along the central middle line. This is consistent with the predictions in Section 2.2.2.2, see Figure 2.12b). The geolocation accuracy is best along the central sensor line.

Here the geolocation error is about 20 m at 10 km distance, increasing to 2000 m at 50 km distance, and 20 km at 100 km distance from the sensors. Note that when sensors move next to each other a mirror image is created (Section 2.2.1.2).

Figure 4.4 (right) shows CEP-contours when sensors move *after* each other. The CEP-contours are symmetric around the central sensor line and the central middle line. Note that the geolocation error goes to infinity along the central sensor line, i.e., geolocation is not possible here. This is consistent with the predictions in Section 2.2.2.2, see Figure 2.11a). The geolocation accuracy is best along the central middle line. Here the geolocation error is about 20 m at 10 km distance, increasing to 2000 m at 50 km distance, and 20 km at 100 km distance from the sensors. This is the same as when the sensors move next to each other. The CEP-contours appear similar in the two cases, only rotated 90º with respect to each other. Note that when sensors move after each other a mirror image is created (Section 2.2.1.2).

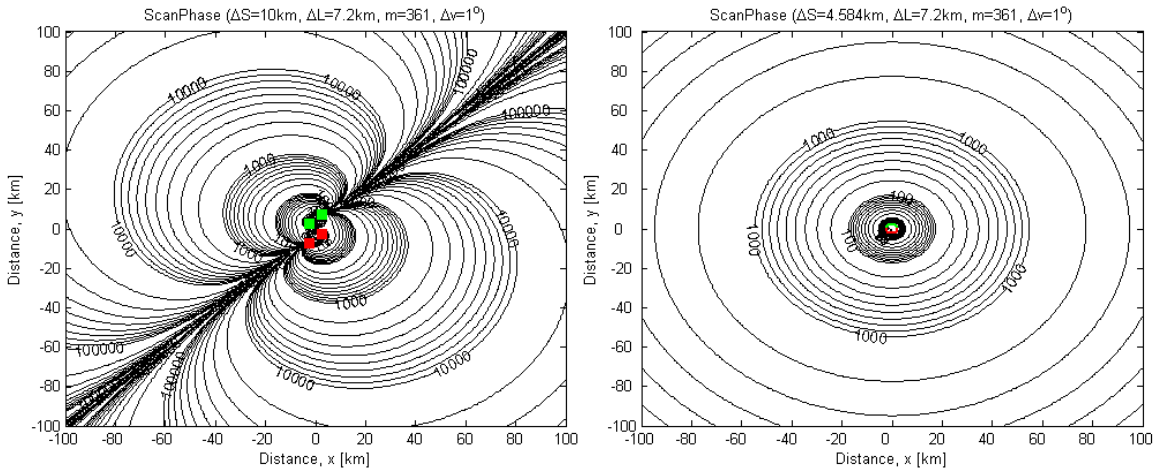


Figure 4.5 CEP-contours for the scanphase method. Left figure shows sensors moving along a tilted line. The sensor distance is 10 km. Right figure shows sensors moving along a circular path. The sensor distance is 4.584 km. The trajectory length is in both cases 7.2 km, and the number of measurements is 361. The error in the measured aperture angle is set to 1°.

Figure 4.5 (left) shows CEP-contours when sensors move along the *tilted* (45º) line. The geolocation accuracy is very poor along the line that the sensors move along. This is consistent with the predictions in 2.2.2.2, see Figure 2.13b). The CEP-contours appear as a slightly distorted version of the CEP-contours for the two previous cases (Figure 4.4), with similar geolocation accuracies. The geolocation error is about 20 m at 10 km distance, increasing to 2000 m at 50 km distance, and 20 km at 100 km distance from the sensors. No mirror image is created.

Figure 4.5 (right) shows CEP-contours when sensors move along a *circular* path. The CEP-contours are circular, and there are no lines where the geolocation accuracy is very poor or where the error goes to infinity, i.e., geolocation is possible everywhere. This is consistent with the predictions in 2.2.2.2, see Figure 2.16a). The geolocation error is about 40 m at 10 km distance, increasing to 900 m at 50 km distance, and 4000 m at 100 km distance from the sensors. Note that

the geolocation accuracy is considerably better (except close to the sensors) than for the previous three cases. The geolocation error also increases much slower with increasing distance from the sensors. Note, however, that when sensors move along a circular path a mirror image is created (Section 2.2.1.2).

4.1.2.1 Summary

The scanphase method requires the use of 2 moving sensors. Geolocation is not possible along the direction in which the sensors move. The geolocation accuracy is best when the sensors move along a circular path. Geolocation is then possible everywhere and the geolocation accuracies that can be obtained are about 40 m at 10 km distance, 900 m at 50 km distance, and 4000 m at 100 km distance from the sensors. However, a mirror image is created. Table 4.2 summarizes the results for the scanphase method.

Geometry/ trajectory	**Mirror image**	**Line with no/poor geoloc.**	**Geolocation accuracy**		
			10 km	**50 km**	**100 km**
Next to	Yes	1	20 m	2000 m	20 km
After	Yes	1	20 m	2000 m	20 km
Tilted	No	1	20 m	2000 m	20 km
Circular	Yes	-	40 m	900 m	4000 m

Table 4.2 Summary of results for the scanphase method. The number of measurements is 361.

4.1.3 AOA

The AOA method requires the use of either 1 moving sensor or 2 sensors that may be stationary. We will discuss the following cases:

1) Two sensors moving *next to* each other
2) Two sensors moving *after* each other
3) Two *stationary* sensors
4) Two sensors - *single* measurement
5) One sensor moving along a *straight* line
6) One sensor moving along a *circular* path

The sensor distance is 10 km (two sensors) and the trajectory length is 7.2 km (moving sensors). When two sensors are used, the number of measurements is 2x361 (except for the single measurement case where the number of measurements is 2x1). When one sensor is used, the number of measurements is 361. The error in the measured angle of arrival is set to 2º.

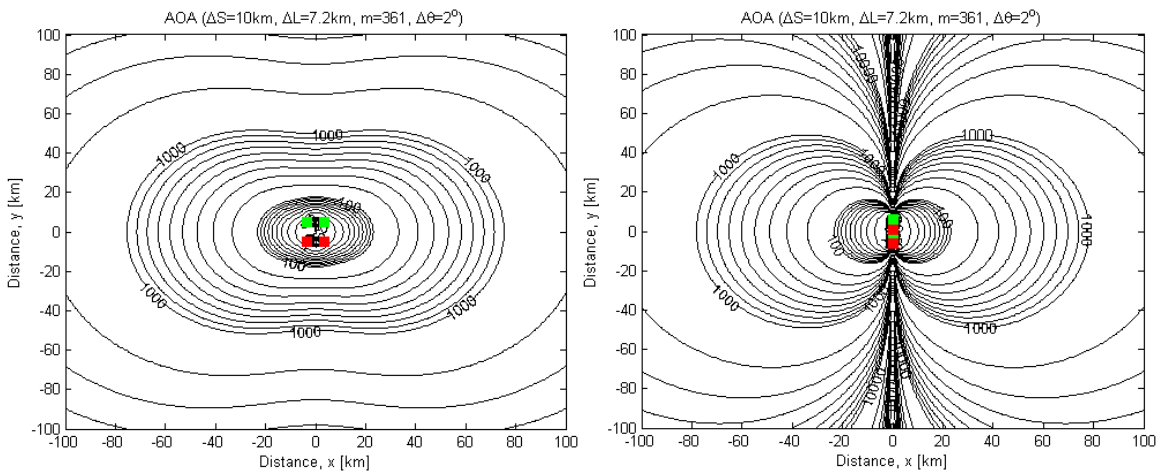


Figure 4.6 CEP-contours for the AOA method. Left figure shows sensors moving next to each other. Right figure shows sensors moving after each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 2x361. The error in the measured angle of arrival is set to 2°.

Figure 4.6 (left) shows CEP-contours when sensors move *next to* each other. The CEP-contours are symmetric around the central sensor line and the central middle line. There are no lines where the geolocation accuracy is very poor or the error goes to infinity, i.e., geolocation is possible everywhere. This is consistent with the predictions in Section 2.2.2.3. For the given choice of sensor distance and trajectory length, the geolocation accuracy is best along the central middle line. Here the geolocation error is about 30 m at 10 km distance, increasing to 500 m at 50 km distance, and 2000 m at 100 km distance from the sensors. In general, longer sensor distances give better accuracy along the central middle line, while longer trajectories give better accuracy along the central sensor line. The direction along which the geolocation accuracy is best will therefore vary with the choice of these parameters.

Figure 4.6 (right) shows CEP-contours when sensors move *after* each other. The CEP-contours are symmetric around the central sensor line and the central middle line. Note that the geolocation error goes to infinity along the central sensor line, i.e., geolocation is not possible here. This is consistent with the predictions in Section 2.2.2.3, see Figure 2.11b). The geolocation accuracy is best along the central middle line. Here the geolocation error is about 30 m at 10 km distance, increasing to 500 m at 50 km distance, and 2000 m at 100 km distance from the sensors. These values are the same as when the sensors move next to each other.

Figure 4.7 (left) shows CEP-contours when sensors are *stationary*. The CEP-contours are similar to the case when sensors move after each other (Figure 4.6), with similar geolocation accuracies. The geolocation error is about 30 m at 10 km distance, increasing to 500 m at 50 km distance, and 2000 m at 100 km distance from the sensors.

Figure 4.7 (right) shows CEP-contours when a single measurement (per sensor) is made (momentaneous geolocation). The CEP-contour geometry is the same as in the previous case, but the geolocation accuracy is much poorer due to the reduced number of measurements. The

geolocation error is here about 500 m at 10 km distance, increasing to 9000 m at 50 km distance, and 40 km at 100 km distance from the sensors.

Circular movement with two sensors was also investigated, but gave poorer geolocation accuracy than when the sensors move next to or after each other.

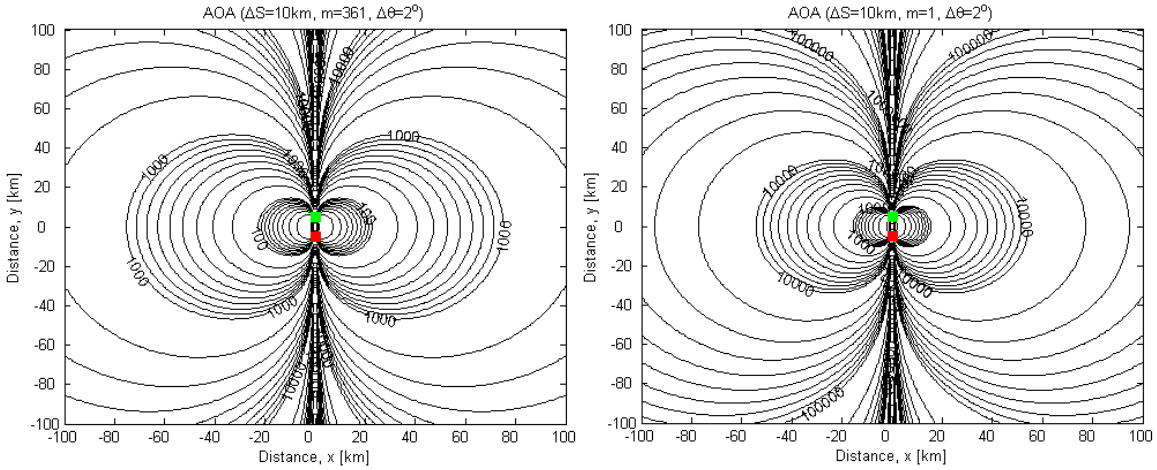


Figure 4.7 CEP-contours for the AOA method. Sensors are stationary. Left figure shows results for stationary sensors when the number of measurements is 2x361. Right figure shows results with only 2x1 measurements. The sensor distance is 10 km and the error in the measured angle of arrival is set to 2°.

Figure 4.8 (left) shows CEP-contours when one sensor moves along a *straight* line. The CEP-contour geometry is the same as when two sensors move after each other. However, the geolocation accuracy is worse since the number of measurements is reduced by a factor 2, and the single sensor covers a shorter path than the two sensors combined do. The geolocation accuracy is best along the central middle line. Here the geolocation error is about 100 m at 10 km distance, increasing to 2000 m at 50 km distance, and 6000 m at 100 km distance from the sensors.

Figure 4.8 (right) shows CEP-contours when one sensor moves along a *circular* path. The CEP-contours are circular, and there are no lines where the geolocation accuracy is very poor or the error goes to infinity, i.e., geolocation is possible everywhere. This is consistent with the predictions in 2.2.2.3, see Figure 2.16b). However, the geolocation accuracies are somewhat poorer than when the sensor moves along a straight line, with geolocation errors about 100 m at 10 km distance, increasing to 3000 m at 50 km distance, and 10 km at 100 km distance from the sensors.

Note that no mirror image is created in any of the cases above.

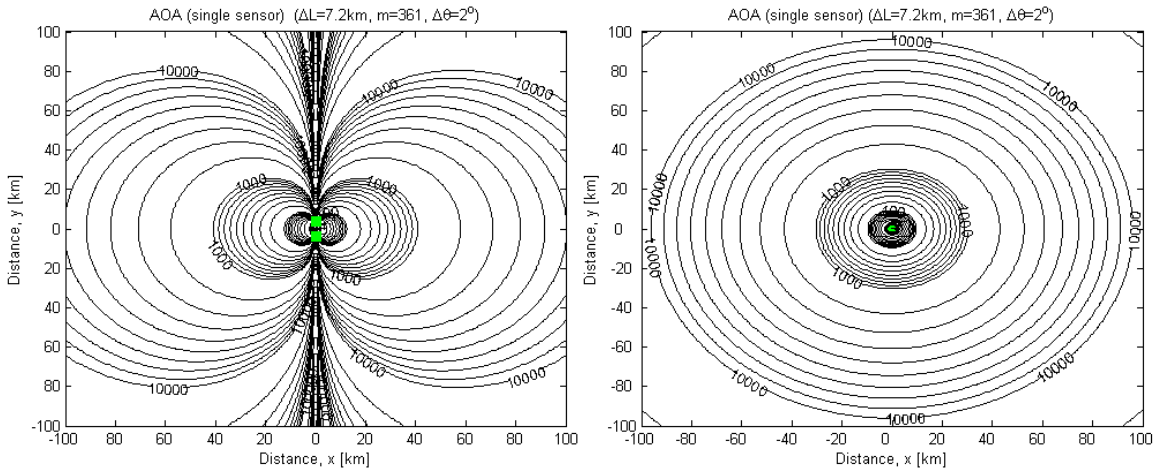


Figure 4.8 CEP-contours for the AOA method. The figures show what can be obtained with the use of only 1 sensor. Left figure shows sensor movement along a straight line. Right figure shows sensor movement along a circular path. The trajectory length is 7.2 km, and the number of measurements is 361. The error in the measured angle of arrival is set to 2°.

4.1.3.1 Summary

The AOA method requires the use of 1 moving sensor or 2 sensors that may be stationary. When one sensor is used, the best overall performance is obtained when the sensor moves on a circular path. Geolocation is then possible everywhere, and the geolocation accuracies that can be obtained are about 100 m at 10 km distance, 3000 m at 50 km distance, and 10 km at 100 km distance from the sensor. When two sensors are used, movement next to each other gives the best results. Geolocation is then possible everywhere, and the geolocation accuracies that can be obtained are about 30 m at 10 km distance, 500 m at 50 km distance, and 2000 m at 100 km distance from the sensors. Momentaneous geolocation can be obtained with corresponding accuracies of about 500 m, 900 m, and 40 km. The AOA method does not create mirror images. Table 4.3 summarizes the results for the AOA method.

# sensors	Geometry/ trajectory	Mirror image	Line with no/poor geoloc.	Geolocation accuracy		
				10 km	50 km	100 km
1	Straight	No	1	100 m	2000 m	6000 m
	Circular	No	-	100 m	3000 m	10 km
2	Next to	No	-	30 m	500 m	2000 m
	After	No	1	30 m	500 m	2000 m
	Stationary	No	1	30 m	500 m	2000 m
	Single	No	1	500 m	9000 m	40 km

Table 4.3 Summary of results for the AOA method. The number of measurements is 361 (one sensor), 2x361 (two sensors), or 2x1 (single measurement).

4.1.4 TDOA & Scanphase

When the TDOA and scanphase methods are combined the sensors may be stationary. We will discuss the following cases:

1) Sensors moving *next to* each other
2) Sensors moving *after* each other
3) *Stationary* sensors
4) *Single* measurement

The sensor distance is 10 km, the trajectory length is 7.2 km (moving sensors), and the number of measurements is 2x361 (or 2x1 for single measurement). The errors in the measured TDOA and aperture angle are set to 50 ns and 1º respectively.

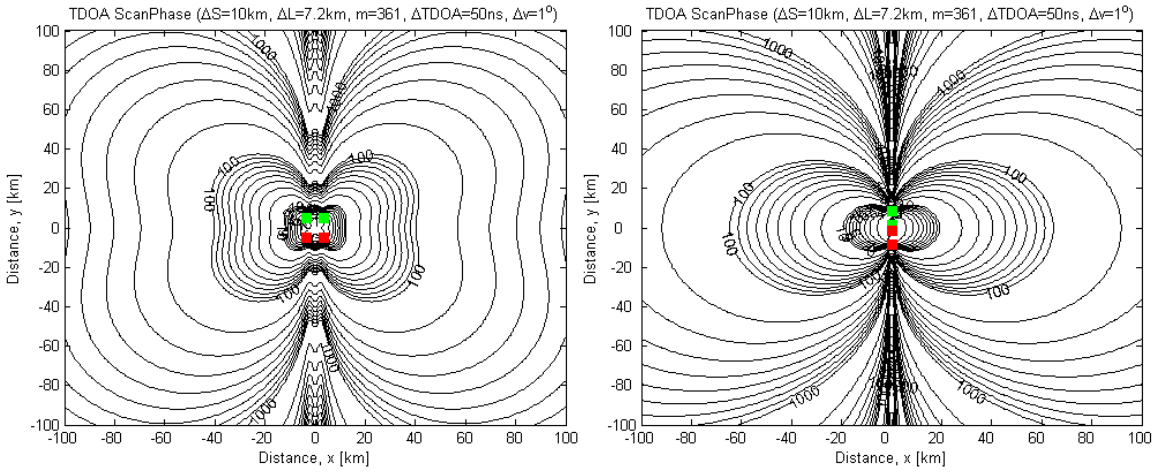


Figure 4.9 CEP-contours when the TDOA and scanphase methods are combined. Left figure shows sensors moving next to each other. Right figure shows sensors moving after each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 2x361. The errors in the measured TDOA and aperture angle are set to 50 ns and 1° respectively.

Figure 4.9 (left) shows CEP-contours when sensors move *next to* each other. The CEP-contours are similar to the CEP-contours for the TDOA method alone (Figure 4.2) with similar geolocation accuracies. The reason for this is that the TDOA method is the most accurate of the two methods and therefore dominates the results. The geolocation accuracy is best along the tilted (45º) line. Here geolocation errors are about 10 m at 10 km distance, increasing to 100 m at 50 km distance, and 500 m at 100 km distance from the sensors. Note, however, that geolocation is now also possible with good accuracy along the central middle line, even though this was not possible for either of the two methods alone. The reason for this is that the curves describing the possible emitter positions for the two methods (see Figure 2.8a) and Figure 2.9a)) intersect at right angles along this line, resulting in good geolocation accuracy when the methods are combined. No mirror image is created when the sensors move next to each other.

Figure 4.9 (right) shows CEP-contours when sensors move *after* each other. The CEP-contours are similar to the CEP-contours for the TDOA method alone (Figure 4.2) with slightly improved geolocation accuracies. The geolocation error is best along the central middle line. Here the geolocation error is about 3 m at 10 km distance, increasing to 60 m at 50 km distance, and 250 m at 100 km distance from the sensors. However, geolocation is still not possible along the central sensor line. The reason for this is that the curves describing the possible emitter positions for the two methods (see Figure 2.8a) and Figure 2.9a)) are parallel along this line. Combining the two methods does therefore not improve the results here. Note that when sensors move after each other a mirror image is created (Section 2.2.1.1 and Section 2.2.1.2).

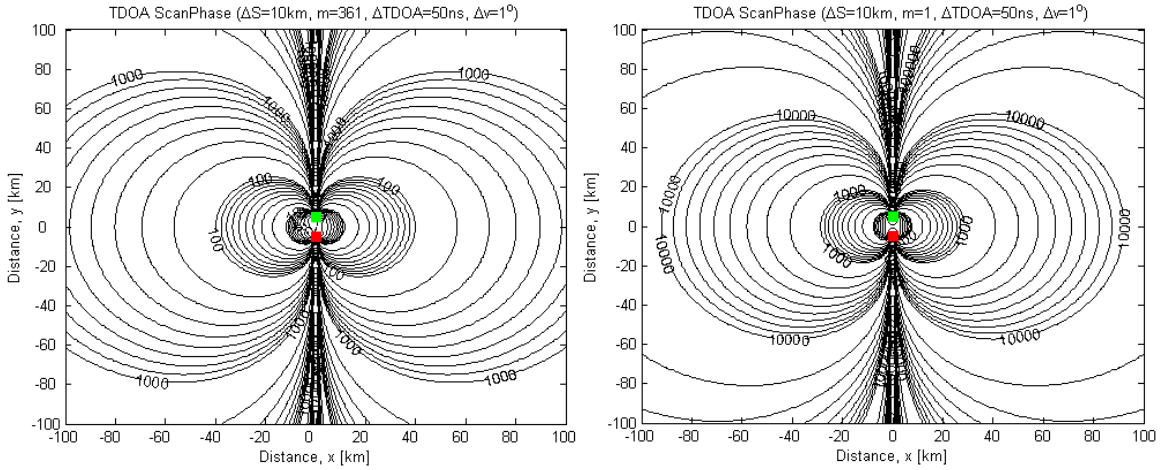


Figure 4.10 CEP-contours when the TDOA and scanphase methods are combined. Left figure shows results for stationary sensors when the number of measurements is 2x361. Right figure shows results with only 2x1 measurements. The sensor distance is 10 km and the errors in the measured TDOA and aperture angle are set to 50 ns and 1° respectively.

Figure 4.10 (left) shows CEP-contours when sensors are *stationary*. The CEP-contours are symmetric around the central sensor line and the central middle line. Note that the geolocation error goes to infinity along the central sensor line, i.e., geolocation is not possible here. This is similar to the case when sensors move after each other. The geolocation accuracy is best along the central middle line. Here the geolocation error is about 10 m at 10 km distance, increasing to 200 m at 50 km distance, and 600 m at 100 km distance from the sensors. This is somewhat worse than when sensors move after or next to each other. Note that when sensors are stationary a mirror image is created.

Figure 4.10 (right) shows CEP-contours when a *single* measurement (per method) is made (momentaneous geolocation). The CEP-contour geometry is the same as in the previous case, but the geolocation accuracy is much poorer due to the reduced number of measurements. The geolocation error is here about 200 m at 10 km distance, increasing to 3000 m at 50 km distance, and 10 km at 100 km distance from the sensors.

Circular sensor movement was also investigated, but gave poorer geolocation accuracy than when sensors move next to or after each other.

4.1.4.1 Summary

The combination of the TDOA and scanphase methods requires the use of 2 sensors that may be stationary. The geolocation accuracy is best when the sensors move after each other, but a mirror image is then created and geolocation is not possible along the central sensor line. The best overall results are obtained when the sensors move next to each other. Geolocation is then possible everywhere, though with poor accuracy along the central sensor line. No mirror image is created. The geolocation accuracy is best along the central middle line, with geolocation errors of about 10 m at 10 km distance, 100 m at 50 km distance, and 500 m at 100 km distance from the sensors. Momentaneous geolocation can be obtained with corresponding errors of about 200 m, 3000 m, and 10 km. Table 4.4 summarizes the results when the TDOA and scanphase methods are combined.

Geometry/ trajectory	**Mirror image**	**Line with no/poor geoloc.**	**Geolocation accuracy**		
			10 km	**50 km**	**100 km**
Next to	No	1	10 m	100 m	500 m
After	Yes	1	3 m	60 m	250 m
Stationary	Yes	1	10 m	200 m	600 m
Single	Yes	1	200 m	3000 m	10 km

Table 4.4 Summary of results when the TDOA and scanphase methods are combined. The number of measurements is 2x361 (or 2x1 for single measurement).

4.1.5 AOA & Scanphase

When the AOA and scanphase methods are combined the sensors may be stationary. We will discuss the following cases:

1) Sensors moving *next to* each other
2) Sensors moving *after* each other
3) *Stationary* sensors
4) *Single* measurement

The sensor distance is 10 km, the trajectory length is 7.2 km (moving sensors), and the number of measurements is 3x361 (or 3x1 for single measurement). The errors in the measured AOA and aperture angle are set to 2º and 1º respectively.

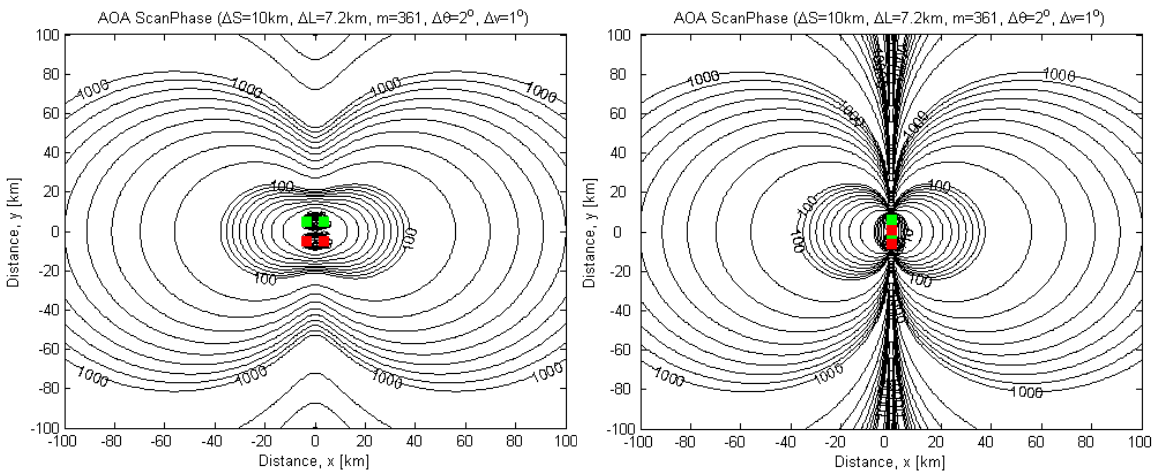


Figure 4.11 CEP-contours when the AOA and scanphase methods are combined. Left figure shows sensors moving next to each other. Right figure shows sensors moving after each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 3x361. The errors in the measured angle of arrival and aperture angle are set to 2° and 1° respectively.

Figure 4.11 (left) shows CEP-contours when sensors move *next to* each other. The CEP-contours are symmetric around the central sensor line and the central middle line. There are no lines where the geolocation accuracy is very poor or the error goes to infinity, i.e., geolocation is possible everywhere. This is similar to what we saw for the AOA method alone. The geolocation accuracy is best along the central middle line and is considerably better than for each method alone. The geolocation error is about 20 m at 10 km distance, increasing to 200 m at 50 km distance, and 600 m at 100 km distance from the sensors.

Figure 4.11 (right) shows CEP-contours when sensors move *after* each other. The CEP-contour symmetry is the same as for each of the methods alone (Figure 4.4 and Figure 4.6), but the geolocation accuracy is considerably better when the methods are combined. Note, however, that even when the methods are combined the geolocation error goes to infinity along the central sensor line, i.e., geolocation is not possible here. The reason for this is that the curves describing the possible emitter positions for the two methods (see Figure 2.9a) and Figure 2.10a)) are parallel along this line. Combining the two methods does therefore not improve the results here. The geolocation accuracy is best along the central middle line. Here the geolocation error is about 20 m at 10 km distance, increasing to 200 m at 50 km distance, and 600 m at 100 km distance from the sensors. This is the same as when the sensors move next to each other.

Figure 4.12 (left) shows CEP-contours when sensors are *stationary*. The CEP-contours are similar to the case when sensors move after each other (Figure 4.11), with similar geolocation accuracies. The geolocation error is about 20 m at 10 km distance, increasing to 200 m at 50 km distance, and 600 m at 100 km distance from the sensors. The geolocation accuracy is considerably better than for the AOA method alone (Figure 4.7).

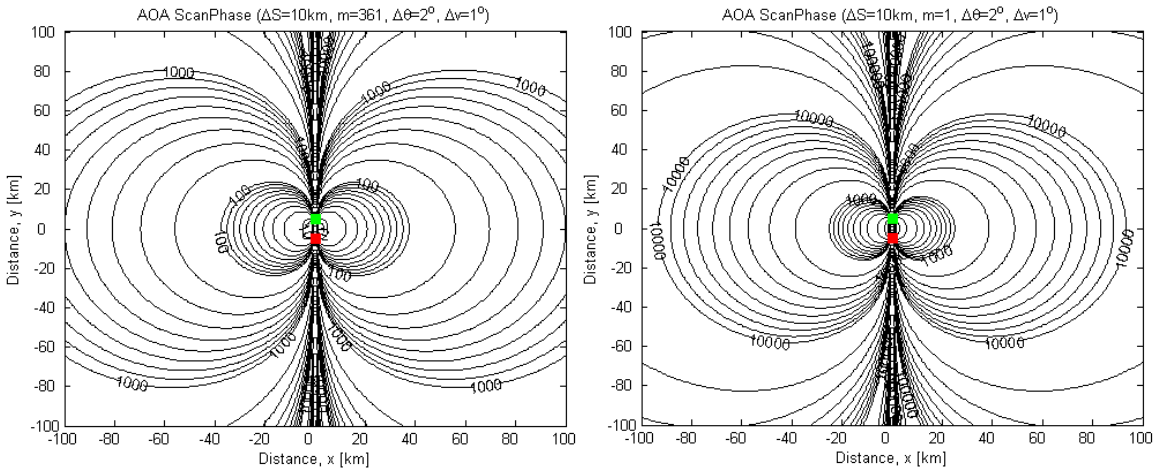


Figure 4.12 CEP-contours when the AOA and scanphase methods are combined. Left figure shows results for stationary sensors when the number of measurements is 3x361. Right figure shows results with only 3x1 measurements. The sensor distance is 10 km and the errors in the measured angle of arrival and aperture angle are set to 2° and 1° respectively.

Figure 4.12 (right) shows CEP-contours when a *single* measurement (i.e., 3x1) is made (momentaneous geolocation). The CEP-contour geometry is the same as in the previous case, but the geolocation accuracy is much poorer due to the reduced number of measurements. The geolocation accuracy is, however, considerably better than for the AOA method alone (Figure 4.7). The geolocation error is about 300 m at 10 km distance, increasing to 3000 m at 50 km distance, and 10 km at 100 km distance from the sensors.

Circular sensor movement was also investigated, but gave somewhat poorer geolocation accuracy than when sensors move next to or after each other.

Note that when the AOA and scanphase methods are combined no mirror image is created.

4.1.5.1 Summary

The combination of the AOA and scanphase methods requires the use of 2 sensors that may be stationary. The best results are obtained when the sensors move next to each other. Geolocation is then possible everywhere. The geolocation accuracy is best along the central middle line, with geolocation errors of about 20 m at 10 km distance, 200 m at 50 km distance, and 600 m at 100 km distance from the sensors. The combination of the AOA and scanphase methods does not create mirror images. Momentaneous geolocation can be obtained with corresponding errors of about 300 m, 3000 m, and 10 km. Table 4.5 summarizes the results when the AOA and scanphase methods are combined.

Geometry/ trajectory	Mirror image	Line with no/poor geoloc.	Geolocation accuracy		
			10 km	50 km	100 km
Next to	No	0	20 m	200 m	600 m
After	No	1	20 m	200 m	600 m
Stationary	No	1	20 m	200 m	600 m
Single	No	1	300 m	3000 m	10 km

Table 4.5 Summary of results when the AOA and scanphase methods are combined. The number of measurements is 3x361 (or 3x1 for single measurement).

4.1.6 TDOA & AOA

When the TDOA and AOA methods are combined the sensors may be stationary. We will discuss the following cases:

1) Sensors moving *next to* each other
2) Sensors moving *after* each other
3) *Stationary* sensors
4) *Single* measurement

The sensor distance is 10 km, the trajectory length is 7.2 km (moving sensors), and the number of measurements is 3x361 (or 3x1 for single measurement). The errors in the measured TDOA and angle of arrival are set to 50 ns and 2º respectively.

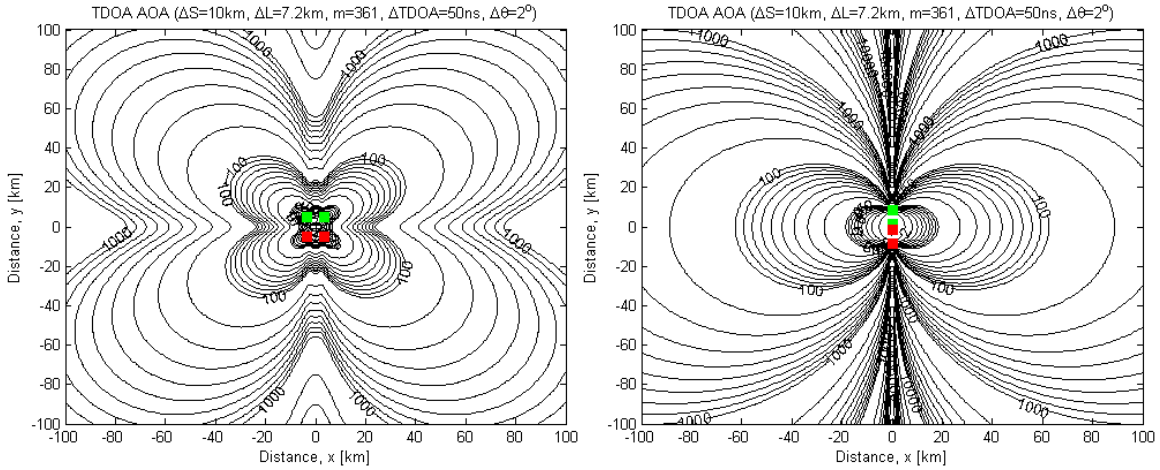


Figure 4.13 CEP-contours when the TDOA and AOA methods are combined. Left figure shows sensors moving next to each other. Right figure shows sensors moving after each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 3x361. The errors in the measured TDOA and angle of arrival are set to 50 ns and 2° respectively.

Figure 4.13 (left) shows CEP-contours when sensors move *next to* each other. The CEP-contours are similar to the CEP-contours for the TDOA method alone (Figure 4.2) with similar geolocation accuracies. Note, however, that geolocation is now also possible with reasonably good accuracy along the central middle line due to the inclusion of the AOA method. The geolocation accuracy is best along the tilted (45º) line. Here geolocation errors are about 10 m at 10 km distance, increasing to 100 m at 50 km distance, and 500 m at 100 km distance from the sensors. The geolocation accuracy is somewhat better than for the AOA method alone (Figure 4.6).

Figure 4.13 (right) shows CEP-contours when sensors move *after* each other. The CEP-contours are similar to the CEP-contours for the TDOA method alone (Figure 4.2) with similar geolocation accuracies. The geolocation accuracy is best along the central middle line. Here geolocation errors are about 5 m at 10 km distance, increasing to 70 m at 50 km distance, and 300 m at 100 km distance from the sensors. The geolocation accuracy is considerably better than for the AOA method alone (Figure 4.6).

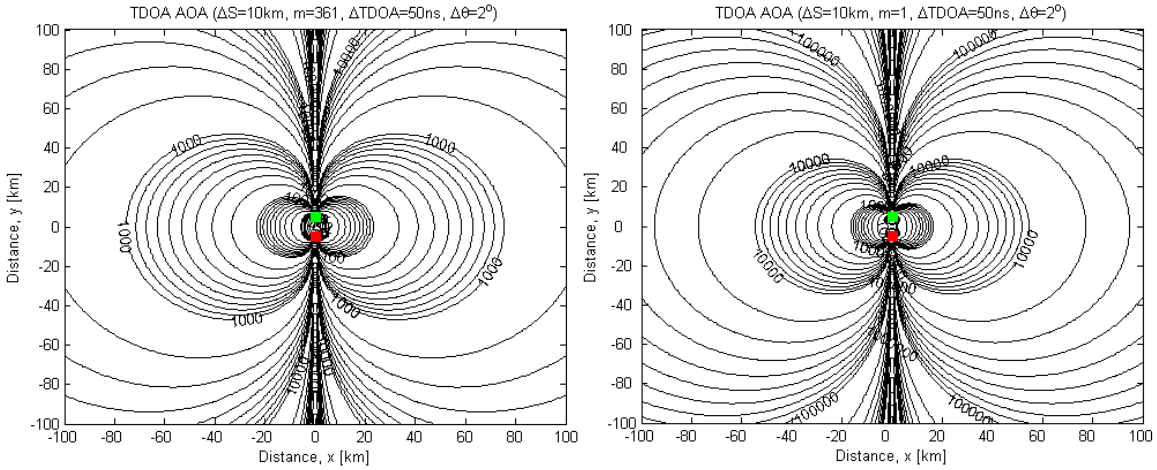


Figure 4.14 CEP-contours when the TDOA and AOA methods are combined. Left figure shows results for stationary sensors when the number of measurements is 3x361. Right figure shows results with only 3x1 measurements. The sensor distance is 10 km and the errors in the measured TDOA and angle of arrival are set to 50 ns and 2° respectively.

Figure 4.14 (left) shows CEP-contours when sensors are *stationary*. The CEP-contours are similar to the CEP-contours for the AOA method alone (Figure 4.7) with similar geolocation accuracies. The geolocation accuracy is best along the central middle line. Here geolocation errors are about 30 m at 10 km distance, increasing to 500 m at 50 km distance, and 2000 m at 100 km distance from the sensors.

Figure 4.14 (right) shows CEP-contours when a *single* measurement (i.e., 3x1) is made (momentaneous geolocation). The CEP-contour geometry is the same as in the previous case, but the geolocation accuracy is much poorer due to the reduced number of measurements. The geolocation accuracy is best along the central middle line. Here geolocation errors are about 500 m at 10 km distance, increasing to 9000 m at 50 km distance, and 40 km at 100 km distance from the sensors.

Circular sensor movement was also investigated, but gave considerably poorer geolocation accuracy than when sensors move next to or after each other.

Note that it appears to be very little or no improvement in the geolocation accuracy by combining the two methods. The reason for this is that the curves describing the possible emitter positions for the two methods (see Figure 2.8a) and Figure 2.10a)) are almost parallell everywhere, i.e., they intersect at very small angles. As a result, very little gain can be obtained in the geolocation accuracy by combining the two methods.

Note that when the TDOA and AOA methods are combined no mirror image is created.

4.1.6.1 Summary

The combination of the TDOA and AOA methods requires the use of 2 sensors that may be stationary. The geolocation accuracy is best when the sensors move after each other, but geolocation is then not possible along the central sensor line. The best overall results are obtained when the sensors move next to each other. Geolocation is then possible everywhere. The geolocation accuracy is best along the tilted (45º) line, with geolocation errors of about 10 m at 10 km distance, 100 m at 50 km distance, and 500 m at 100 km distance from the sensors. Momentaneous geolocation can be obtained with corresponding errors of aabout 500 m, 9000 m, and 40 km. The combination of the TDOA and AOA methods does not create mirror images. Table 4.6 summarizes the results when the TDOA and AOA methods are combined.

Geometry/ trajectory	**Mirror image**	**Line with no/poor geoloc.**	**Geolocation accuracy**		
			10 km	**50 km**	**100 km**
Next to	No	0	10 m	100 m	500 m
After	No	1	5 m	70 m	300 m
Stationary	No	1	30 m	500 m	2000 m
Single	No	1	500 m	9000 m	40 km

Table 4.6 Summary of results when the TDOA and AOA methods are combined. The number of measurements is 3x361 (or 3x1 for single measurement).

4.1.7 TDOA & Scanphase & AOA

When all three methods are combined the sensors may be stationary. We will discuss the following cases:

1) Sensors moving *next to* each other
2) Sensors moving *after* each other
3) *Stationary* sensors
4) *Single* measurement

The sensor distance is 10 km, the trajectory length is 7.2 km (moving sensors), and the number of measurements is 4x361 (except for the single measurement case where the number of measurements is 4x1). The errors in the measured TDOA, aperture angle, and angle of arrival are set to 50 ns, 1º, and 2º respectively.

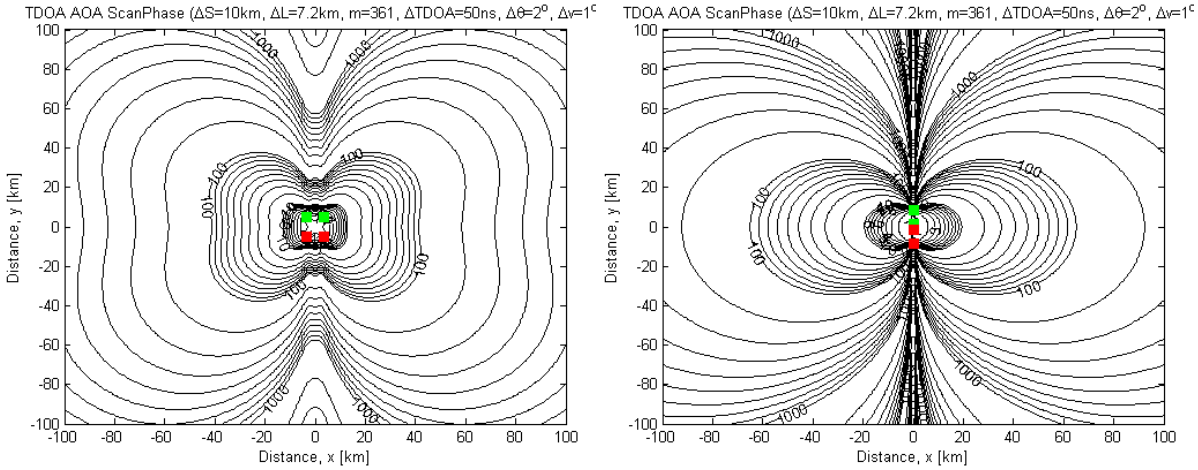


Figure 4.15 CEP-contours when all three methods are combined. Left figure shows sensors moving next to each other. Right figure shows sensors moving after each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 4x361. The errors in the measured TDOA, aperture angle, and angle of arrival are set to 50 ns, 1°, and 2° respectively.

Figure 4.15 (left) shows CEP-contours when sensors move *next to* each other. The CEP-contours are similar to the CEP-contours for the TDOA and scanphase methods combined (Figure 4.9), with slightly improved geolocation accuracies due to the inclusion of the AOA method. The geolocation accuracy is best along the tilted (45º) line. Here geolocation errors are about 5 m at 10 km distance, increasing to 100 m at 50 km distance, and 400 m at 100 km distance from the sensors.

Figure 4.15 (right) shows CEP-contours when sensors move *after* each other. The CEP-contours are similar to the CEP-contours for the TDOA and scanphase method combined (Figure 4.9), with similar geolocation accuracies. The CEP-contours are also similar to the CEP-contours for the TDOA and AOA methods combined (Figure 4.13) and for the TDOA method alone (Figure 4.2), with slightly improved geolocation accuracies. The geolocation accuracy is best along the central middle line. Here geolocation errors are about 3 m at 10 km distance, increasing to 60 m at 50 km distance, and 250 m at 100 km distance from the sensors.

Figure 4.16 (left) shows CEP-contours when sensors are *stationary*. The CEP-contours are similar to the CEP-contours for the TDOA and scanphase methods combined (Figure 4.10) and the AOA and scanphase methods combined (Figure 4.12), with similar geolocation accuracies. The geolocation accuracy is better than for the TDOA and AOA method combined (Figure 4.14) or the AOA method alone (Figure 4.7). The geolocation accuracy is best along the central middle line. Here geolocation errors are about 10 m at 10 km distance, increasing to 200 m at 50 km

distance, and 600 m at 100 km distance from the sensors.

Figure 4.16 (right) shows CEP-contours when a *single* measurement is made (momentaneous geolocation). The CEP-contour geometry is the same as in the previous case, but the geolocation accuracy is much poorer due to the reduced number of measurements. The geolocation accuracy is best along the central middle line. Here geolocation errors are about 200 m at 10 km distance, increasing to 3000 m at 50 km distance, and 10 km at 100 km distance from the sensors.

Circular sensor movement was also investigated, but gave considerably poorer geolocation accuracy than when sensors move next to or after each other.

Note that when all three methods are combined no mirror image is created.

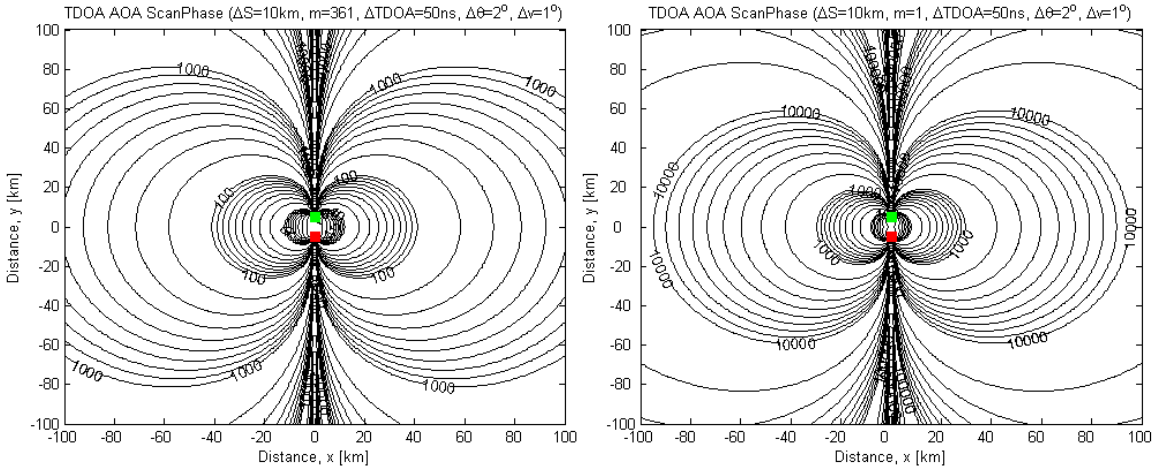


Figure 4.16 CEP-contours when all three methods are combined. Sensors are stationary. Left figure shows results when the number of measurements is 4x361. Right figure shows results with only 4x1 measurements. The sensor distance is 10 km and the errors in the measured TDOA, aperture angle, and angle of arrival are set to 50 ns, 1°, and 2° respectively.

4.1.7.1 Summary

The combination of all three methods requires the use of 2 sensors that may be stationary. The geolocation accuracy is best when the sensors move after each other, but geolocation is then not possible along the central sensor line. The best overall results are obtained when the sensors move next to each other. Geolocation is then possible everywhere. The geolocation accuracy is best along the tilted (45º) line, with geolocation errors of about 5 m at 10 km distance, 100 m at 50 km distance, and 400 m at 100 km distance from the sensors. Momentaneous geolocation can be obtained with corresponding errors of about 200 m, 3000 m, and 10 km. The combination of all three methods does not create mirror images. Table 4.7 summarizes the results when all three methods are combined.

Geometry/ trajectory	Mirror image	Line with no/poor geoloc.	Geolocation accuracy		
			10 km	50 km	100 km
Next to	No	0	5 m	100 m	400 m
After	No	1	3 m	60 m	250 m
Stationary	No	1	10 m	200 m	600 m
Single	No	1	200 m	3000 m	10 km

Table 4.7 Summary of results when all three methods are combined. The number of measurements is 4x361 (or 4x1 for single measurement).

4.1.8 Summary

We have in this section studied the localization methods TDOA, scanphase, and AOA with respect to characteristics of the methods and the geolocation accuracy that can be obtained. We have also looked at different combinations of the methods. Table 4.8 presents the main findings.

The CEP-contours for the different methods or combinations of methods are found to be symmetric around the central sensor line and the central middle line, when the sensors are stationary or move next to or after each other along a straight line. Circular sensor movement gives circular symmetry for the CEP-contours.

When sensors are stationary or moving after each other, geolocation is not possible along the central sensor line. Mirror images are also often created. When sensors move next to each other the problem with mirror images disappears (except when the scanphase method is used alone), and geolocation is possible everywhere in most cases[3].

Moving sensors do in general give better geolocation accuracies than stationary sensors. For the TDOA method alone the best overall results are obtained when the sensors move along a *zig-zag* path. For the scanphase method alone the best results are obtained when the sensors move along a *circular* path. This is also the case for the AOA method alone when only one sensor is used. In all other cases, the best overall results are obtained when the sensors move *next to* each other.

Combining two or more methods does in general give better results than applying one method alone. It usually removes the problem with mirror images and often makes geolocation possible along lines where geolocation is not possible with one method alone. However, not all combinations of methods are equally beneficial. The TDOA and AOA methods do for instance have emitter position curves that intersect at very small angles (except close to the sensors). Very little gain in geolocation accuracy can therefore be obtained by combining these two methods. The scanphase method, on the other hand, has emitter position curves that often intersect at close to right angles with the emitter position curves from the other two methods. Combining the

[3] The only exception is when the scanphase or TDOA methods are used alone. In this case geolocation is never possible along the direction in which the sensors move.

scanphase method with either the TDOA method or the AOA method or both is therefore often beneficial. An example of this can be seen from Table 4.8 for the single measurement case (last three columns), where the geolocation accuracy for the AOA method alone or the AOA and TDOA methods combined is approximately three times poorer than for the scanphase method in combination with either the TDOA method or the AOA method or both. In general, the results obtained by combining two or more methods will be dominated by the most accurate method.

The best overall performance is obtained when all three methods are combined and the sensors move next to each other. Geolocation is then possible everywhere and no mirror image is created. The geolocation accuracy is in this case best along the tilted (45º) line, with geolocation errors about 5 m at 10 km distance, increasing to 100 m at 50 km distance, and 400 m at 100 km distance from the sensors. To obtain these results 361 measurements was used, which corresponds to 15 min of observation time when the measurement period is 2.5 s. For momentaneous geolocation (single measurement) the error is about 200 m at 10 km distance, increasing to 3000 m at 50 km distance, and 10 km at 100 km distance from the sensors. The geolocation accuracy is in this case best along the central middle line.

Method	Requirements	# Sensors	Best geometry/ trajectory	Mirror image	Line with poor geolocation(*)	Geolocation accuracy					
						m=361			m=1		
						R=10km	R=50km	R=100km	R=10km	R=50km	R=100km
TDOA	· Very precise clock	2	zig-zag	No	1	5 m	90 m	400 m	-	-	-
Scanphase	-	2	circular	Yes	-	40 m	900 m	4000 m	-	-	-
AOA	· DF-sensor	1	circular	No	-	100 m	3000 m	10 km	-	-	-
		2	next to	No	-	30 m	500 m	2000 m	500 m	9000 m	40 km
TDOA & Scanphase	· Very precise clock	2	next to	No(**)	1	10 m	100 m	500 m	200 m	3000 m	10 km
AOA & Scanphase	· DF-sensor	2	next to	No	-	20 m	200 m	600 m	300 m	3000 m	10 km
TDOA & AOA	· DF-sensor · Very precise clock	2	next to	No	-	10 m	100	500 m	500 m	9000 m	40 km
TDOA & AOA & Scanphase	· DF-sensor · Very precise clock	2	next to	No	-	5 m	100 m	400 m	200 m	3000 m	10 km

(*) Momentaneous geolocation (single measurement, m=1) is never (for any method or combination of methods) possible along the sensor line.

(**) For momentaneous geolocation (single measurement, m=1) a mirror image is present.

Table 4.8 Results for the three localization methods. The geolocation accuracy is given for different distances R from the sensors along the direction with best geolocation accuracy.

4.2 CEP-radius and CRLB error ellipses

We will in this section study the CRLB error ellipses for the different methods in order to gain more insight into the behaviour of the error. It is interesting to know if the error is approximately the same in all directions (for given emitter position) or if the error is significantly smaller in one direction than another. The CEP-radius gives information only about the size of the geolocation error, while the CRLB error ellipse gives information about both the size and the direction of the error.

We have calculated the CEP-radius with corresponding CRLB error ellipses for each method alone and for all three methods combined. The CRLB error ellipses were calculated for emitter positions at distances 10 km, 30 km, and 50 km and angular positions 0º, 15º, 30º, 45º, 60º, 75º, and 90º. The results are presented in Figure 4.17-Figure 4.22 and Table 4.9-Table 4.14.

4.2.1 TDOA

Figure 4.17 shows CEP-contours and CRLB error ellipses with corresponding minor and major semi-axis contours for the TDOA method when sensors move next to each other. The sensor distance is 7.2 km, the trajectory length is 7.2 km, and the number of measurements is 361. The error in the measured TDOA is set to 50 ns.

Figure 4.17a) shows the CEP-contours. Consistent with the findings in Section 4.1.1, we see that the CEP-radius is large (poor geolocation accuracy) along the central sensor line and that the CEP-radius goes to infinity along the central middle line, i.e., geolocation is not possible here. The geolocation accuracy is best along the tilted (45º) line with values better than approximately 200 m for distances up to 60 km from the sensors, see Table 4.9.

	Central middle line			**Tilted (45º) line**			**Central sensor line**		
	R=10km	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**
CEP	∞	∞	∞	8 m	60 m	200 m	20 m	400 m	3000 m
Major	∞	∞	∞	10 m	90 m	350 m	20 m	500 m	5000 m
Minor	1 m	3 m	7 m	2 m	5 m	9 m	5 m	50 m	200 m

Table 4.9 Values for the CEP-radius and the minor and major semi-axes of the CRLB error ellipse at selected emitter positions for the TDOA method. The values correspond to Figure 4.17.

Figure 4.17b) shows corresponding CRLB error ellipses (enlarged ten times) at selected emitter positions. The error ellipses are very large at the central sensor line (far from the sensors) and become infinitely large at the central middle line. This is consistent with the results seen for the CEP-radius. At large distances the error ellipses are also very elongated and they become more elongated when approaching the central sensor line or the central middle line. The difference between the minor and major semi-axis can be a factor ten or more, see Table 4.9. Note that the error ellipses are elongated in the radial direction everywhere, i.e., the TDOA method is very

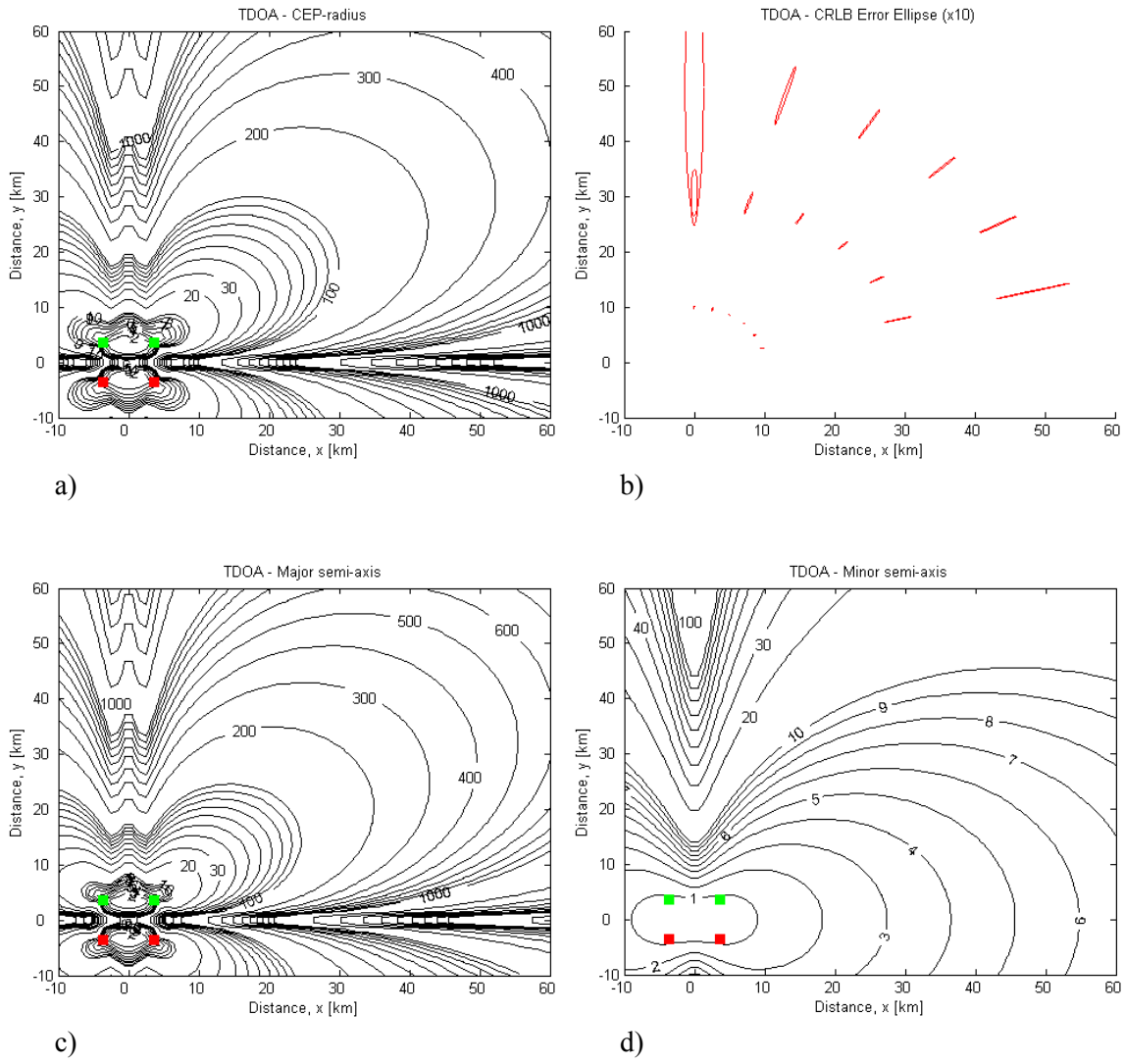


Figure 4.17 The TDOA method. a) CEP-contours. b) CRLB error ellipses at selected emitter positions (enlarged ten times). c) Major semi-axis contours. d) Minor semi-axis contours. The sensors are moving next to each other. The sensor distance is 7.2 km, the trajectory length is 7.2 km, and the number of measurements is 361. The error in the measured TDOA is set to 50 ns.

suitable for localization in the *angular* direction, but less accurate in the radial direction (especially at large distances from the sensors).

Figure 4.17c) shows the major semi-axis contours. They have the same geometry as the CEP contours, but somewhat larger values, see Table 4.9.

Figure 4.17d) shows the minor semi-axis contours. They have a different geometry from the CEP contours, and the values are very much smaller, see Table 4.9. The smallest values (highest accuracy) are found along the central middle line where the minor semi-axis is smaller than approximately 7 m for distances up to 60 km from the sensors. Thus, even though geolocation is not possible along this line (infinite CEP-radius), the angular position of an emitter located here can be determined very accurately.

4.2.2 Scanphase

Figure 4.18 shows CEP-contours and CRLB error ellipses with corresponding minor and major semi-axis contours for the scanphase method when sensors move next to each other. The sensor distance is 7.2 km, the trajectory length is 7.2 km, and the number of measurements is 361. The error in the measured aperture angle is set to 1º.

Figure 4.18a) shows the CEP-contours. Consistent with the findings in Section 4.1.2, we see that the CEP-radius is large (poor geolocation accuracy) along the central middle line. The geolocation accuracy is best along the central sensor line with values better than approximately 5000 m for distances up to 60 km from the sensors, see Table 4.10.

Figure 4.18b) shows corresponding CRLB error ellipses at selected emitter positions. The error ellipses are very large far from the sensors and become increasingly large when approaching the central middle line. This is consistent with the results seen for the CEP-radius. At large distances the error ellipses are also very elongated and they become more elongated when approaching the central middle line. The difference between the minor and major semi-axis can be a factor ten or more, see Table 4.10. Note that the error ellipses are elongated in the radial direction close to the central sensor line (i.e., best accuracy in the *angular* direction), while being elongated in the angular direction close to the central middle line (i.e., best accuracy in the *radial* direction). This makes the scanphase method very suitable as a complementary method to the TDOA method which is most accurate in the *angular* direction only.

Figure 4.18c) shows the major semi-axis contours. They have the same geometry as the CEP contours, but somewhat larger values, see Table 4.10.

Figure 4.18d) shows the minor semi-axis contours, which are circular with values very much smaller than the CEP contour values, see Table 4.10. The minor semi-axis is smaller than approximately 500 m for distances up to 60 km from the sensors in all directions. Thus, even though geolocation is very poor along the central middle line (large CEP-radius), the radial position of an emitter located here can be determined accurately.

	Central middle line			**Tilted (45º) line**			**Central sensor line**		
	R=10km	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**
CEP	∞	∞	∞	40 m	800 m	6000 m	20 m	600 m	5000 m
Major	∞	∞	∞	50 m	1000 m	9000 m	30 m	800 m	7000 m
Minor	10 m	100 m	500 m	10 m	100 m	500 m	10 m	100 m	500 m

Table 4.10 Values for the CEP-radius and the minor and major semi-axes of the CRLB error ellipse at selected emitter positions for the scanphase method. The values correspond to Figure 4.18.

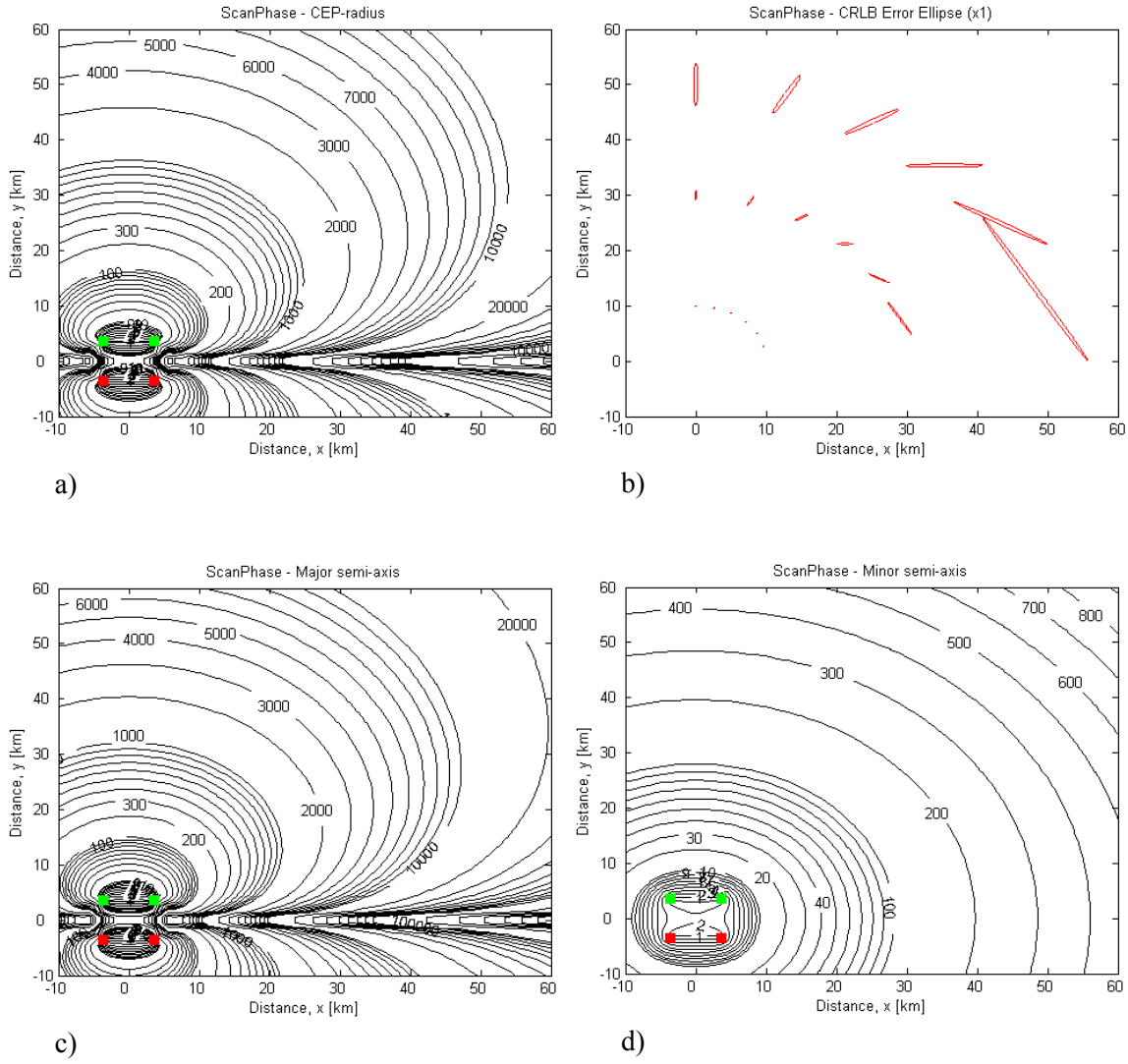


Figure 4.18 The scanphase method. a) CEP-contours. b) CRLB error ellipses at selected emitter positions. c) Major semi-axis contours. d) Minor semi-axis contours. The sensors are moving next to each other. The sensor distance is 7.2 km, the trajectory length is 7.2 km, and the number of measurements is 361. The error in the measured aperture angle is set to 1°.

4.2.3 AOA

Figure 4.19 shows CEP-contours and CRLB error ellipses with corresponding minor and major semi-axis contours for the AOA method when sensors move next to each other. The sensor distance is 7.2 km, the trajectory length is 7.2 km, and the number of measurements is 2x361. The error in the measured angle of arrival is set to 2º.

Figure 4.19a) shows the CEP-contours. Consistent with the findings in Section 4.1.3, we see that geolocation is possible everywhere. The geolocation accuracy is best along the central middle line with values better than approximately 900 m for distances up to 60 km from the sensors, see Table 4.11.

Figure 4.19b) shows corresponding CRLB error ellipses (enlarged ten times) at selected emitter positions. The error ellipses are much larger far from the sensors than at closer distances. At large distances the error ellipses are also very elongated and they become more elongated when approaching the central sensor line. The difference between the minor and major semi-axis can be a factor ten or more, see Table 4.11. Note that the error ellipses are elongated in the radial direction everywhere, i.e., the AOA method is very suitable for localization in the *angular* direction, but less accurate in the radial direction (especially at large distances from the sensors).

Figure 4.19c) shows the major semi-axis contours. They have the same geometry as the CEP contours, but somewhat larger values, see Table 4.11.

Figure 4.19d) shows the minor semi-axis contours, which are circular (except very close to the sensors) with values very much smaller than the CEP contour values, see Table 4.11. The minor semi-axis is smaller than approximately 80 m for distances up to 60 km from the sensors in all directions.

	Central middle line			**Tilted (45º) line**			**Central sensor line**		
	R=10km	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**
CEP	30 m	200 m	900 m	40 m	300 m	1000 m	30 m	400 m	1500 m
Major	40 m	300 m	1500 m	50 m	400 m	1500 m	40 m	500 m	2000 m
Minor	15 m	40 m	80 m	10 m	40 m	80 m	10 m	40 m	80 m

Table 4.11 Values for the CEP-radius and the minor and major semi-axes of the CRLB error ellipse at selected emitter positions for the AOA method when two sensors are used. The values correspond to Figure 4.19.

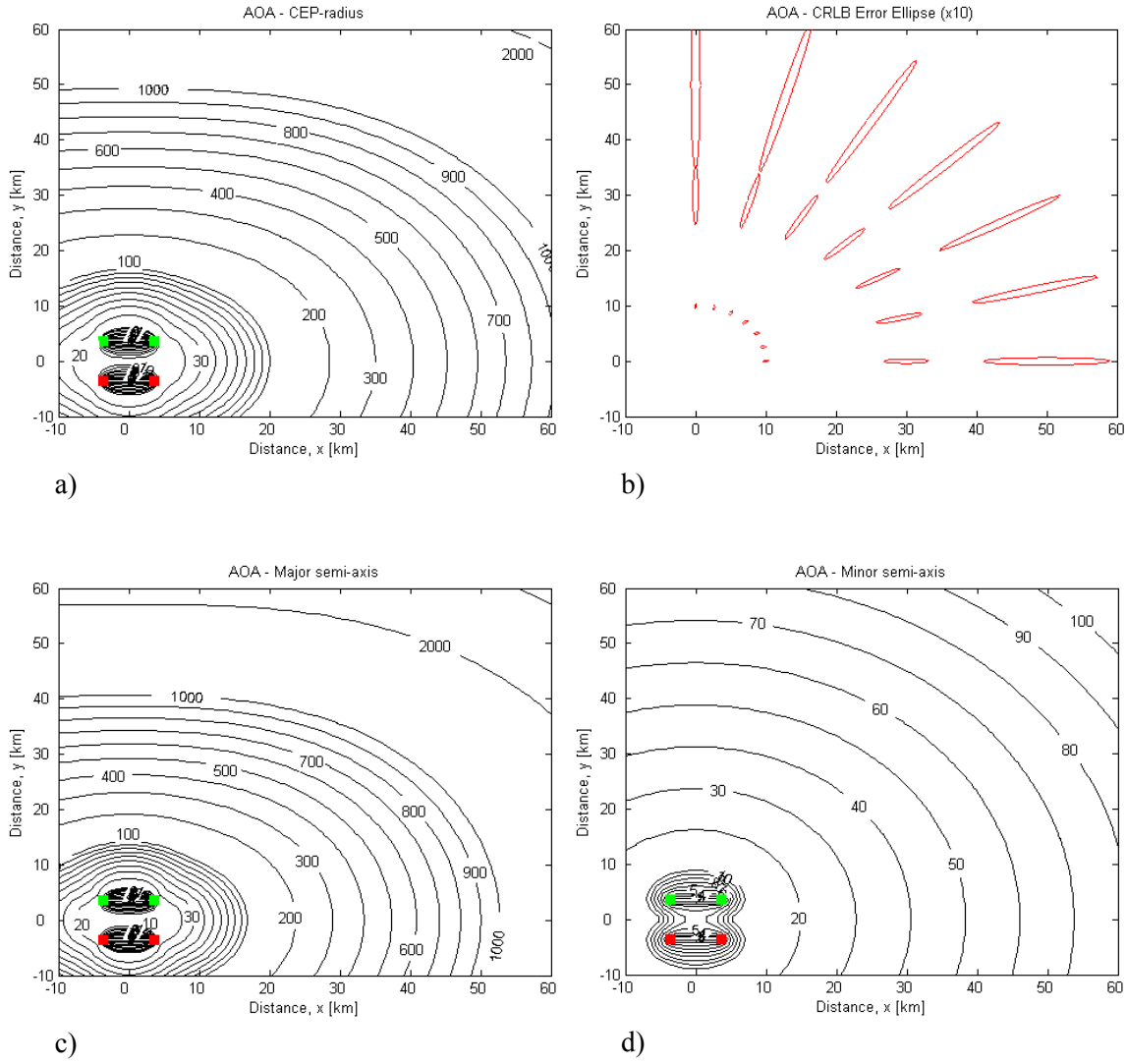


Figure 4.19 The AOA method. a) CEP-contours. b) CRLB error ellipses at selected emitter positions (enlarged ten times). c) Major semi-axis contours. d) Minor semi-axis contours. The sensors are moving next to each other. The sensor distance is 7.2 km, the trajectory length is 7.2 km, and the number of measurements is 2x361. The error in the measured angle of arrival is set to 2°.

Figure 4.20 shows CEP-contours and CRLB error ellipses with corresponding minor and major semi-axis contours for the AOA method when one sensor is used. The sensor moves on a 7.2 km long circular path and the number of measurements is 361. The error in the measured angle of arrival is set to 2º.

Figure 4.20a) shows the CEP-contours. Consistent with the findings in Section 4.1.3, we see that the CEP-contours are circular and that geolocation is possible everywhere with values better than approximately 4000 m for distances up to 60 km from the sensors, see Table 4.12.

Figure 4.20b) shows corresponding CRLB error ellipses (enlarged three times) at selected emitter positions. The error ellipses are much larger far from the sensors than at closer distances. At large distances the error ellipses are also very elongated and they become more elongated when approaching the central sensor line. The difference between the minor and major semi-axis can be a factor ten or more, see Table 4.12. Note that the error ellipses are elongated in the radial direction also when only one sensor is used.

Figure 4.20c) shows the major semi-axis contours. They have the same geometry as the CEP contours, but somewhat larger values, see Table 4.12.

Figure 4.20d) shows the minor semi-axis contours, which are circular with values very much smaller than the CEP contour values, see Table 4.12. The minor semi-axis is smaller than approximately 80 m for distances up to 60 km from the sensors in all directions.

	Central middle line			**Tilted (45º) line**			**Central sensor line**		
	R=10km	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**
CEP	100 m	1000 m	4000 m	100 m	1000 m	4000 m	100 m	1000 m	4000 m
Major	200 m	1500 m	6000 m	200 m	1500 m	6000 m	200 m	1500 m	6000 m
Minor	15 m	40 m	80 m	15 m	40 m	80 m	15 m	40 m	80 m

Table 4.12 Values for the CEP-radius and the minor and major semi-axes of the CRLB error ellipse at selected emitter positions for the AOA method when one sensor is used. The values correspond to Figure 4.20.

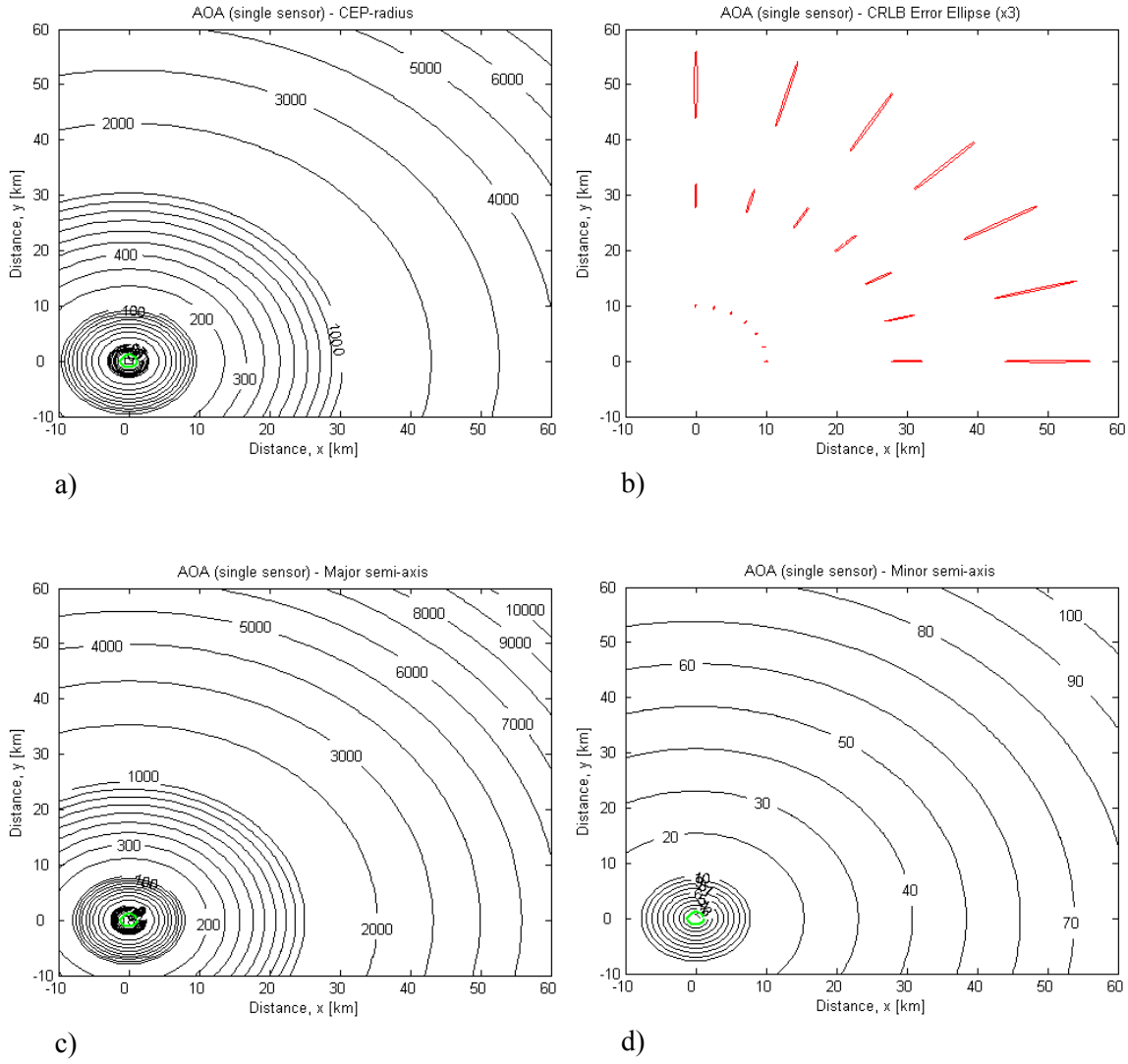


Figure 4.20 The AOA method. a) CEP-contours. b) CRLB error ellipses at selected emitter positions (enlarged three times). c) Major semi-axis contours. d) Minor semi-axis contours. Only 1 sensor is used for the measurements. The sensor moves on a 7.2 km long circular path. The number of measurements is 361 and the error in the measured angle of arrival is set to 2°.

4.2.4 TDOA & AOA & Scanphase

Figure 4.21 shows CEP-contours and CRLB error ellipses with corresponding minor and major semi-axis contours when sensors move next to each other and all three methods are combined. The sensor distance is 7.2 km, the trajectory length is 7.2 km, and the number of measurements is 4x361. The errors in the measured TDOA, aperture angle, and angle of arrival are set to 50 ns, 1º , and 2º respectively.

Figure 4.21a) shows the CEP-contours. Consistent with the findings in Section 4.1.7, we see that geolocation is possible everywhere. The geolocation accuracy is best along the tilted (45º) line with values better than approximately 200 m for distances up to 60 km from the sensors, see Table 4.13.

Figure 4.21b) shows corresponding CRLB error ellipses (enlarged fourty times) at selected emitter positions. The error ellipses are much larger far from the sensors than at closer distances and become larger when approaching the central sensor line. The error ellipses are also very elongated at large distances. This is consistent with earlier findings for each of the methods alone. The difference between the minor and major semi-axis can be a factor ten or more, see Table 4.13. Note that the error ellipses are elongated in the radial direction everywhere, i.e., when all three methods are combined the accuracy is best in the *angular* direction.

Figure 4.21c) shows the major semi-axis contours. They have the same geometry as the CEP contours, but somewhat larger values, see Table 4.13.

Figure 4.21d) shows the minor semi-axis contours. They have a different geometry from the CEP contours, and the values are very much smaller, see Table 4.13. The smallest values (best accuracy) are found along the central middle line where the minor semi-axis is smaller than approximately 7 m for distances up to 60 km from the sensors.

	Central middle line			**Tilted (45º) line**			**Central sensor line**		
	R=10km	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**
CEP	8 m	70 m	300 m	8 m	60 m	200 m	10 m	200 m	1000 m
Major	10 m	100 m	400 m	10 m	80 m	300 m	20 m	300 m	2000 m
Minor	1 m	3 m	7 m	2 m	5 m	9 m	4 m	30 m	70 m

Table 4.13 Values for the CEP-radius and the minor and major semi-axes of the CRLB error ellipse at selected emitter positions when all three methods are combined. The values correspond to Figure 4.21.

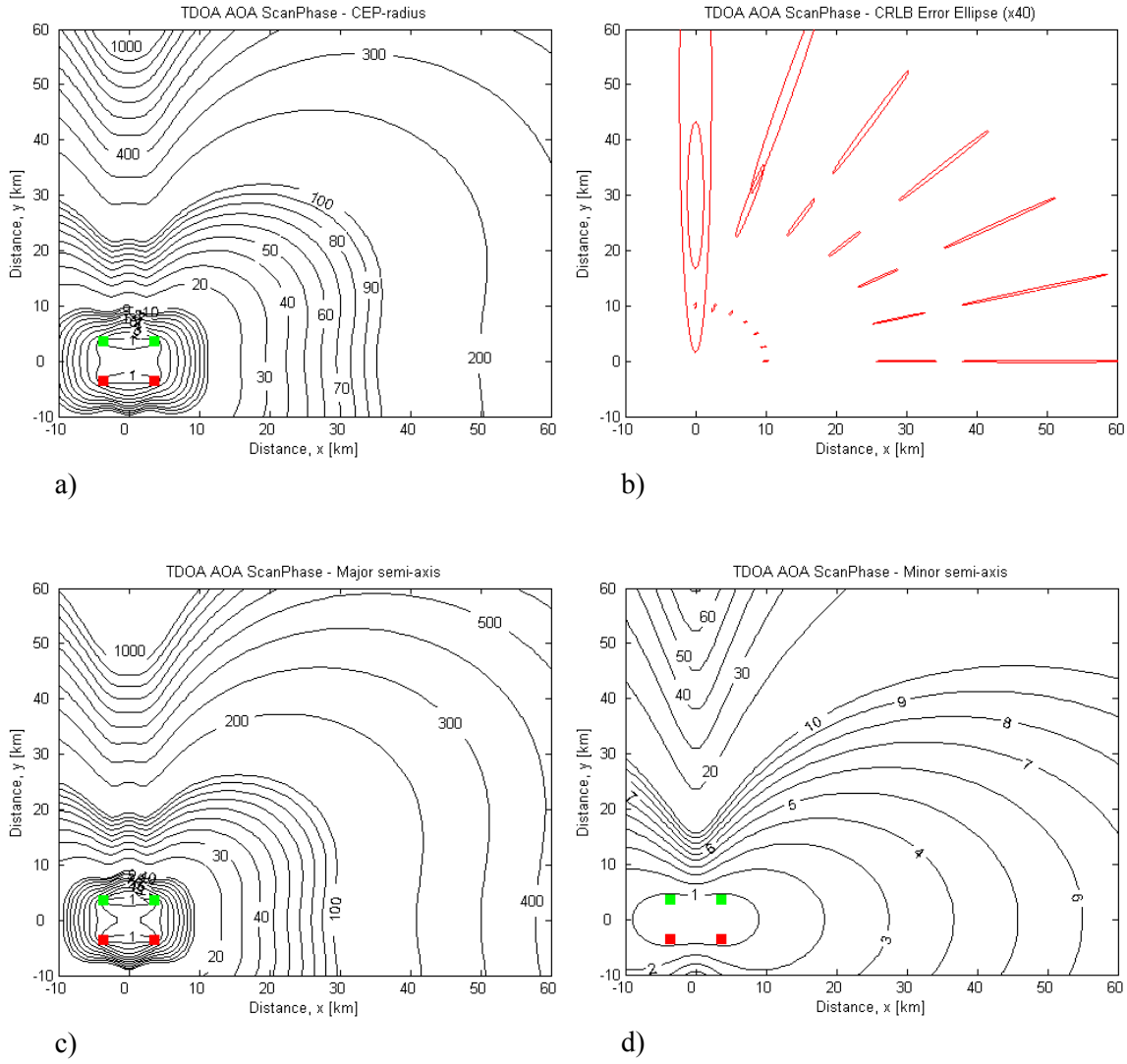


Figure 4.21 All three methods combined. a) CEP-contours. b) CRLB error ellipses at selected emitter positions (enlarged forty times). c) Major semi-axis contours. d) Minor semi-axis contours. The sensors are moving next to each other. The sensor distance is 7.2 km, the trajectory length is 7.2 km, and the number of measurements is 4x361. The errors in the measured TDOA, aperture angle, and angle of arrival are set to 50 ns, 1°, and 2° respectively.

Figure 4.22 shows CEP-contours and CRLB error ellipses with corresponding minor and major semi-axis contours when all three methods are combined and the sensors are stationary and only a single measurement (4x1) is made (momentaneous geolocation). The sensor distance is 7.2 km and the errors in the measured TDOA, aperture angle, and angle of arrival are set to 50 ns, 1º , and 2º respectively.

Figure 4.22a) shows the CEP-contours. Consistent with the findings in Section 4.1.7, we see that the CEP-radius goes to infinity along the central sensor line, i.e., geolocation is not possible here. The geolocation accuracy is best along the central middle line with values better than approximately 6000 m for distances up to 60 km from the sensors, see Table 4.14.

Figure 4.22b) shows corresponding CRLB error ellipses at selected emitter positions. The error ellipses are much larger far from the sensors than at closer distances and become infinintely large at the central sensor line. The error ellipses are also very elongated at large distances. This is consistent with earlier findings for each of the methods alone. The difference between the minor and major semi-axis can be a factor ten or more, see Table 4.14. Note that the error ellipses are elongated in the radial direction everywhere also when the sensors are stationary.

Figure 4.22c) shows the major semi-axis contours. They have the same geometry as the CEP contours, but somewhat larger values, see Table 4.14.

Figure 4.22d) shows the minor semi-axis contours. Their geometry is somewhat similar to the geometry of the CEP contours, but the values are very much smaller, see Table 4.14. The smallest values (best accuracy) are found along the central middle line where the minor semi-axis is smaller than approximately 150 m for distances up to 60 km from the sensors. Note that the semi-minor axis have finite values along the central sensor line, as opposed to the semi-major axis and the CEP-radius that are infinitely large here. Thus, even though geolocation is not possible along the central sensor line the angular position of an emitter located here can be determined fairly accurately.

	Central middle line			**Tilted (45º) line**			**Central sensor line**		
	R=10km	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**	**R=10km**	**R=30km**	**R=60km**
CEP	200 m	1500 m	6000 m	200 m	2000 m	8000 m	∞	∞	∞
Major	300 m	2000 m	8000 m	300 m	3000 m	10 km	∞	∞	∞
Minor	20 m	60 m	150 m	30 m	90 m	200 m	100 m	700 m	2000 m

Table 4.14 Values for the CEP-radius and the minor and major semi-axes of the CRLB error ellipse at selected emitter positions when all three methods are combined. The sensors are stationary and only 4x1 measurements are made. The values correspond to Figure 4.22.

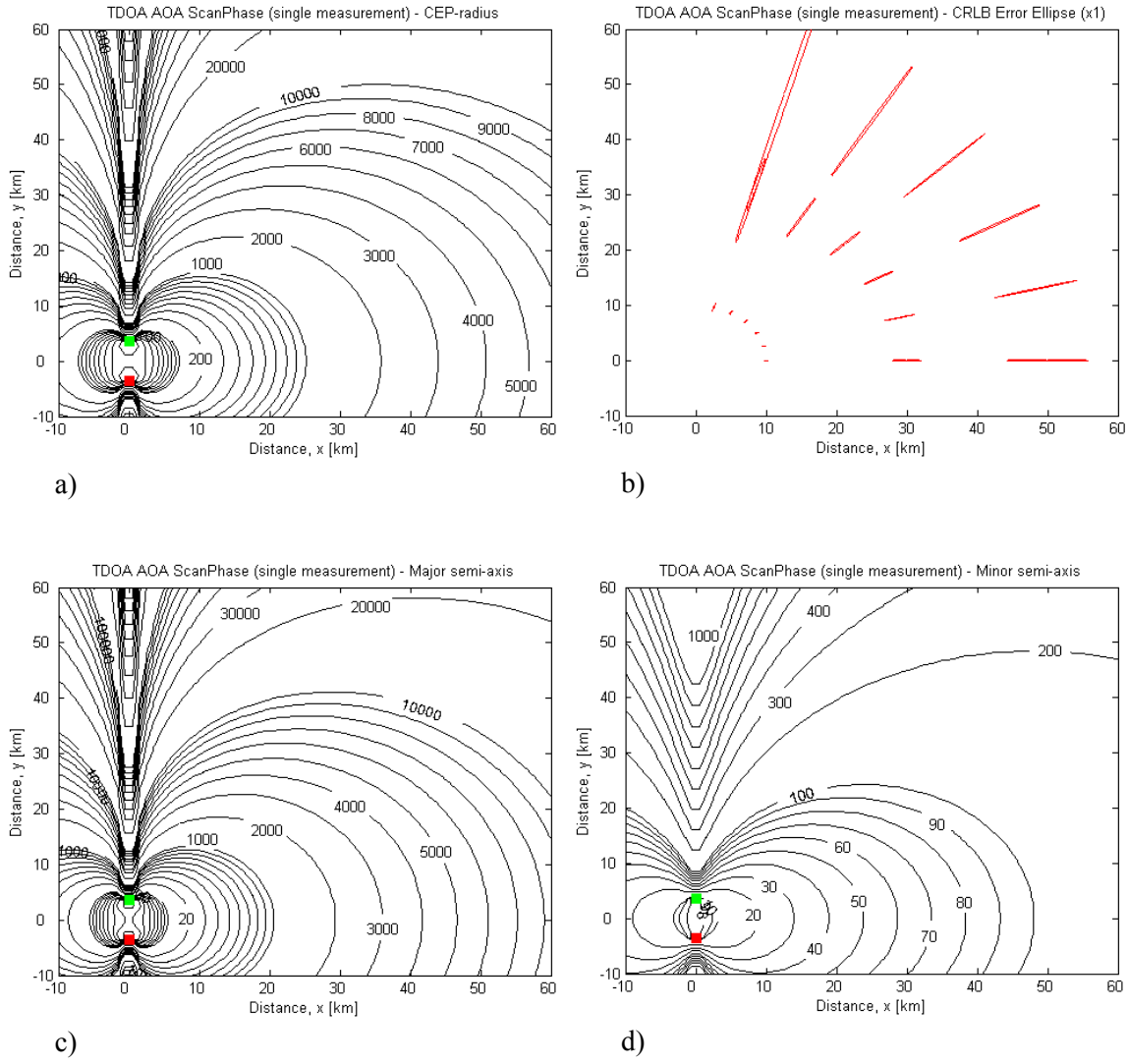


Figure 4.22 All three methods combined. a) CEP-contours. b) CRLB error ellipses at selected emitter positions. c) Major semi-axis contours. d) Minor semi-axis contours. The sensors are stationary. The sensor distance is 7.2 km and the number of measurements is 4x1. The errors in the measured TDOA, aperture angle, and angle of arrival are set to 50 ns, 1°, and 2° respectively.

4.2.5 Summary

We have studied the CRLB error ellipses for the three methods in order to learn more about the direction of the error. We have seen that the CRLB error ellipses become larger and more elongated at longer distances from the sensors and that the differences between the minor and major semi-axes in many cases are very large. The major semi-axis may be a factor ten or more larger than the corresponding minor semi-axis. The CEP-radius is largely dominated by the major semi-axis. The accuracy in the minor semi-axis direction may therefore be considerably better than the CEP-radius indicates. The minor semi-axis is directed in the angular direction for the TDOA and AOA methods. For the scanphase method the minor semi-axis may also be in the radial direction. In some cases it will be possible to determine either the angular or the radial position of an emitter even though geolocation is not possible.

4.3 Momentaneous geolocation with more than two sensors

Momentaneous geolocation is of particular interest since it gives the emitter position within very short time. So far we have discussed what can be obtained if two sensors are used for the measurements (the single measurement cases discussed in Section 4.1). We will now look at what can be achieved if three or four sensors are used to obtain momentaneous geolocation. Note that geolocation errors due to emitter movement is not an issue for momentaneous geolocation since a shipborne emitter will not move significantly during the very short observation period.

When more than two sensors are used for the measurements, it is possible to combine the sensors in several different ways (pairs) in order to obtain the measurements (TDOA or aperture angle). However, care must be taken when performing the calculations in order to obtain the correct geolocation accuracies. This will be explained for the TDOA method below. The same will be the case for the scanphase method.

Imagine a scene with N sensors, where each sensor makes one pulse time of arrival (TOA) measurement. Two sensors are required to make one TDOA measurement based on the measured TOA pair. With N sensors in the scene we get $N(N-1)/2$ pulse pairs that can be used to generate a TDOA measurement. However, the errors in the TDOA measurements will not be independent of each other since the same TOA measurement is used in more than one pulse pair. In fact, each TOA measurement will be used $(N-1)$ times if all possible pulse pairs are used in the calculations.

When calculating the geolocation accuracy for a scene with N sensors, we have calculated as if we have $N(N-1)/2$ pulse pairs with $N(N-1)$ independent TOA measurements. However, since we do in fact only have N independent TOA measurements (one for each sensor) we must correct the resulting values for the geolocation errors with a factor $\sqrt{(N-1)}$. If three sensors are used for the calculations we must multiply the resulting geolocation accuracy with a factor $\sqrt{2}$. If four sensors are used we must multiply with a factor $\sqrt{3}$.

4.3.1 Three sensors

We have looked at the momentaneous geolocation accuracy that can be obtained when three sensors are used. The sensors were placed in an equilateral triangle with 10 km distance between the sensors. The errors in the measured time difference of arrival, aperture angle, and angle of arrival were set to 50 ns, 1º, and 2º respectively.

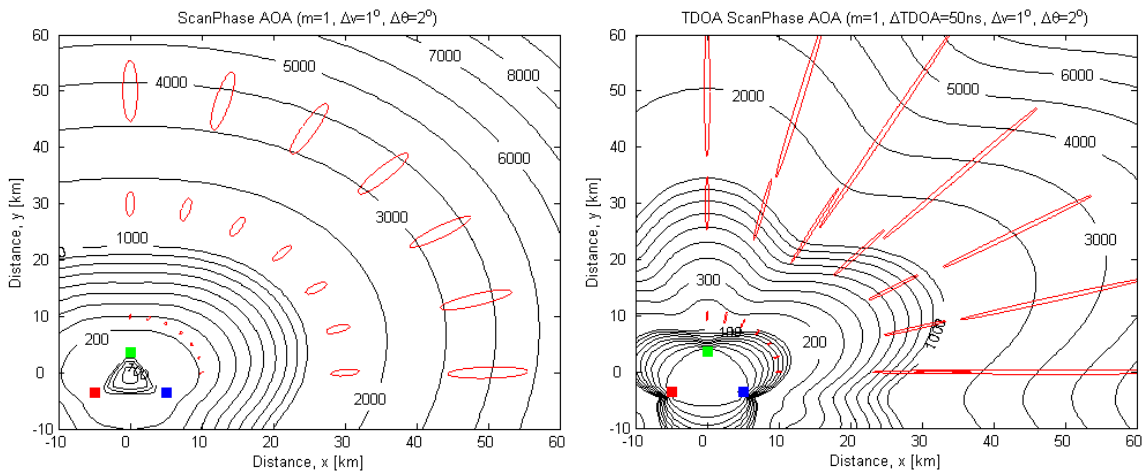


Figure 4.23 CEP-contours and CRLB error ellipses (red) for single measurement with three sensors. Left figure shows results when the scanphase and AOA methods are combined. Right figure shows results when all three methods are combined (error ellipses enlarged four times). The sensors are placed in an equilateral triangle with 10 km distance between the sensors. The errors in the measured time difference of arrival, aperture angle, and angle of arrival are set to 50 ns, 1°, and 2° respectively.

We calculated the geolocation accuracy for all methods and combination of methods. For the scanphase and AOA methods the CEP-contours were found to be approximately circular (slight triangular shape for the AOA method), and there were no lines where geolocation was very poor or not possible. Figure 4.23 (left) shows the results when the scanphase and AOA methods are combined. Geolocation accuracies of about 200 m at 10 km distance, 1500 m at 30 km distance, and 3500 m at 50 km distance from the sensors can then be obtained. This is very similar to the results obtained with two sensors (Figure 4.12 and Table 4.5). However, with three sensors geolocation is possible everywhere, while with two sensors geolocation is not possible along the line through the sensors.

For the TDOA method alone the CEP-contours were found to have the shape of a flower with six petals. Further, there were three lines (one through each sensor pair) where geolocation was very poor. Figure 4.23 (right) shows the results when the TDOA method is combined with the scanphase and AOA methods. Geolocation accuracies of about 70 m at 10 km distance, 700 m at 30 km distance, and 2000 m at 50 km distance from the sensors can then be obtained. This is somewhat better than what can be obtained when only two sensors are used (Figure 4.16 and Table 4.7). Also, with three sensors geolocation is possible everywhere, while with two sensors geolocation is not possible along the line through the sensors. The geolocation accuracy improves when all three methods are combined compared to when only the scanphase and AOA methods are combined.

4.3.2 Four sensors

We have also looked at the momentaneous geolocation accuracy that can be obtained when four sensors are used. The sensors were placed in a square with 10 km long sides. The errors in the measured time difference of arrival, aperture angle, and angle of arrival were set to 50 ns, 1º, and 2º respectively.

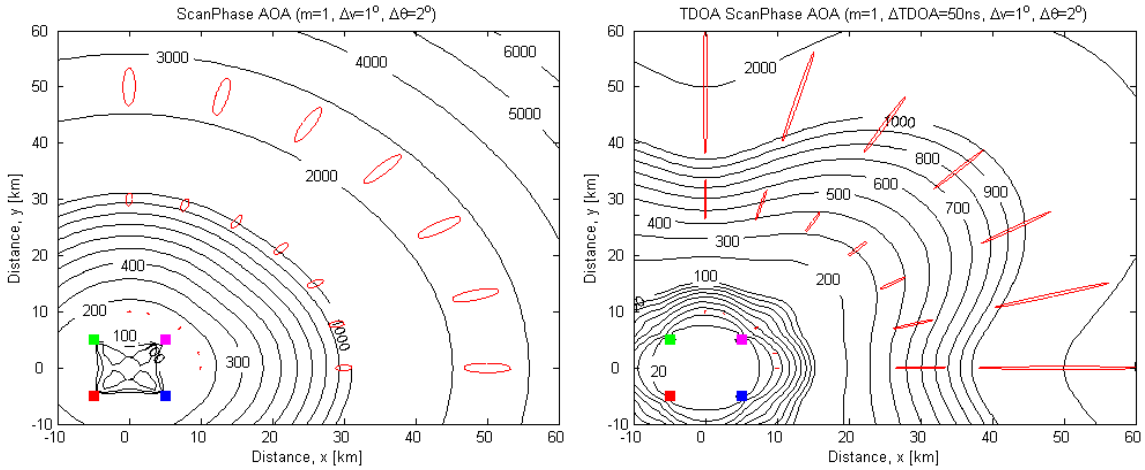


Figure 4.24 CEP-contours and CRLB error ellipses (red) for momentaneous geolocation with four sensors. Left figure shows results when the scanphase and AOA methods are combined. Right figure shows results when all three methods are combined (error ellipses enlarged four times). The sensors are placed in a square with 10 km long sides. The errors in the measured time difference of arrival, aperture angle, and angle of arrival are set to 50 ns, 1º, and 2º respectively.

We calculated the geolocation accuracy for all methods and combination of methods. For the scanphase and AOA methods alone the CEP-contours were found to be approximately circular, and there were no lines where geolocation was very poor or not possible. Figure 4.24 (left) shows the results when the scanphase and AOA methods are combined. Geolocation accuracies of about 200 m at 10 km distance, 1000 m at 30 km distance, and 2500 m at 50 km distance from the sensors can then be obtained. This is somewhat better than the results obtained with three sensors (Figure 4.23 and Table 4.15).

For the TDOA method alone the CEP-contours were found to have the shape of a flower with four petals, and there were no lines where geolocation was very poor or not possible. Figure 4.24 (right) shows the results when the TDOA method is combined with the scanphase and AOA methods. Geolocation accuracies of about 40 m at 10 km distance, 300 m at 30 km distance, and 800 m at 50 km distance from the sensors can then be obtained. This is approximately two times better than what can be obtained with three sensors (Figure 4.23 and Table 4.15). The geolocation accuracy improves when all three methods are combined compared to when only the scanphase and AOA methods are combined.

4.3.3 Summary

The use of more than two sensors for momentaneous geolocation removes lines along which geolocation is not possible and improves the geolocation accuracy. The best results are obtained when all three methods are combined. Table 4.15 shows the geolocation accuracies that can be obtained at different distances from the sensors when the scanphase and AOA methods are combined and when all three methods are combined.

Method	**# sensors**	**Geolocation accuracy**		
		10 km	**30 km**	**50 km**
Scanphase & AOA	3	200 m	1500 m	3500 m
	4	200 m	1000 m	2500 m
TDOA & Scanphase & AOA	3	70 m	700 m	2000 m
	4	40 m	300 m	800 m

Table 4.15 Geolocation accuracy for momentaneous geolocation with 3 or 4 sensors.

4.4 Parameter dependency

In this section we study the behaviour of the geolocation accuracy with respect to the following parameters:

- Error in measured parameter (ΔTDOA, Δv or $\Delta\theta$)
- Number of measurements (N)
- Sensor distance (ΔS)
- Trajectory length (ΔL)
- Radial distance (R)

Knowledge about this dependency is very useful, since it makes it possible to estimate quickly the geolocation accuracy for one set of values for the parameters when the geolocation accuracy is known (already calculated) for another set of values.

The following emitter positions were used for the calculations;

(x=30 km, y=10 km), (x=50 km, y=50 km), (x=20 km, y=60 km)

Figure 4.25-Figure 4.36 and Table 4.16 presents the results. A summary is given in Table 4.17.

4.4.1 Error in measured parameter

Figure 4.25-Figure 4.27 show the CEP-radius dependency on the error in the measured parameter (ΔTDOA, Δv or $\Delta\theta$) for the three methods. The sensors move next to each other and the sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 361 (or 2x361 for the AOA method). Upper left and right figures show CEP-contours for $\Delta TDOA$=50 ns

and 100 ns, Δv=0.5º and 1º, and $\Delta\theta$=1º and 2º respectively. The lower figures show the CEP-radius as a function of the error in the measured parameter at selected emitter positions.

From the figures it is clear that the CEP-radius is proportional to the error in the measured parameter for all three methods, i.e.,

$$\text{CEP-radius} \sim \Delta TDOA,\ \Delta\upsilon \text{ or } \Delta\theta \tag{4.1}$$

The minor and major semi-axes of the CRLB error ellipses were found to have the same dependency on the error in the measured parameter as the CEP-radius.

The dependency on the error in the measured parameter will be the same also for other geometries.

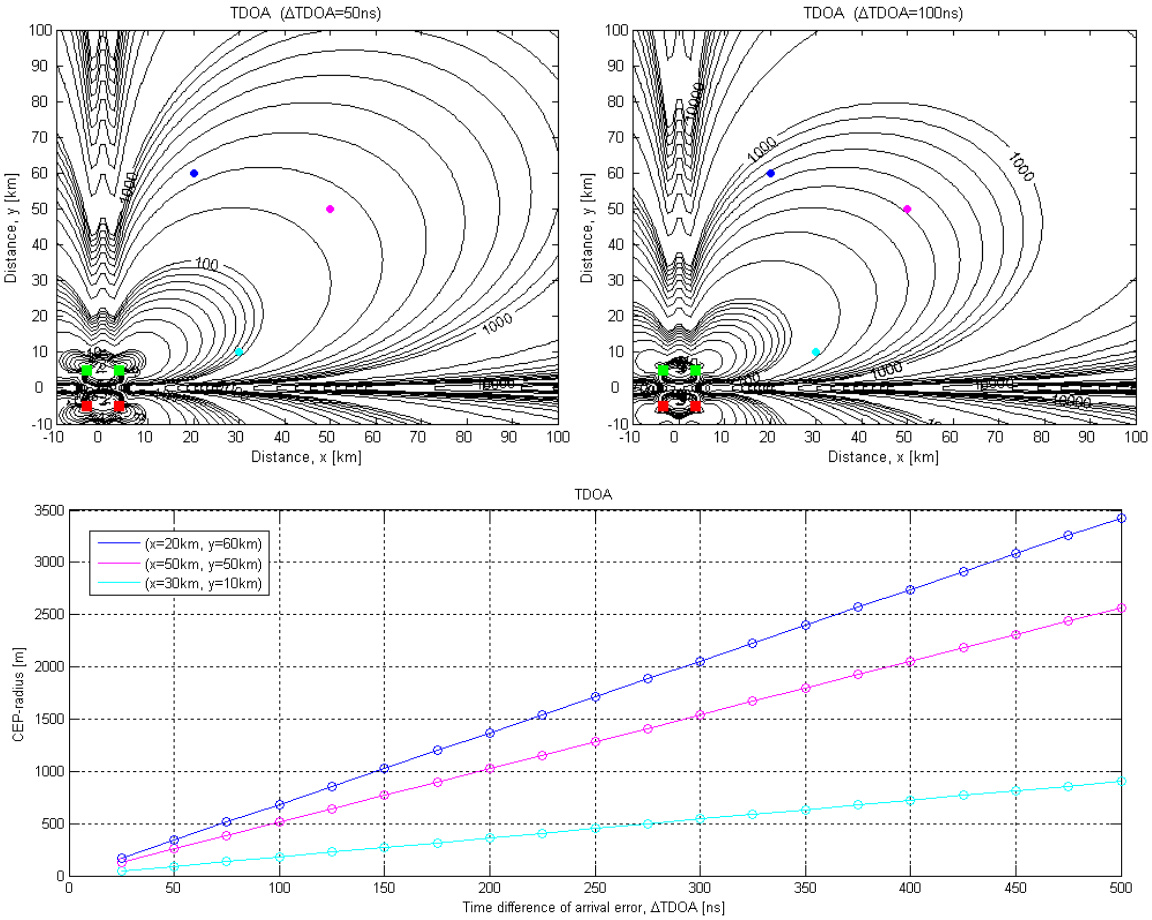


Figure 4.25 CEP-radius dependency on the error (ΔTDOA) in the measured TDOA. Upper left and right figures show CEP-contours for ΔTDOA=50 ns and 100 ns respectively. The sensors move next to each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 361. Lower figure shows CEP-radius as a function of ΔTDOA at selected emitter positions (marked by corresponding colours in the upper figures).

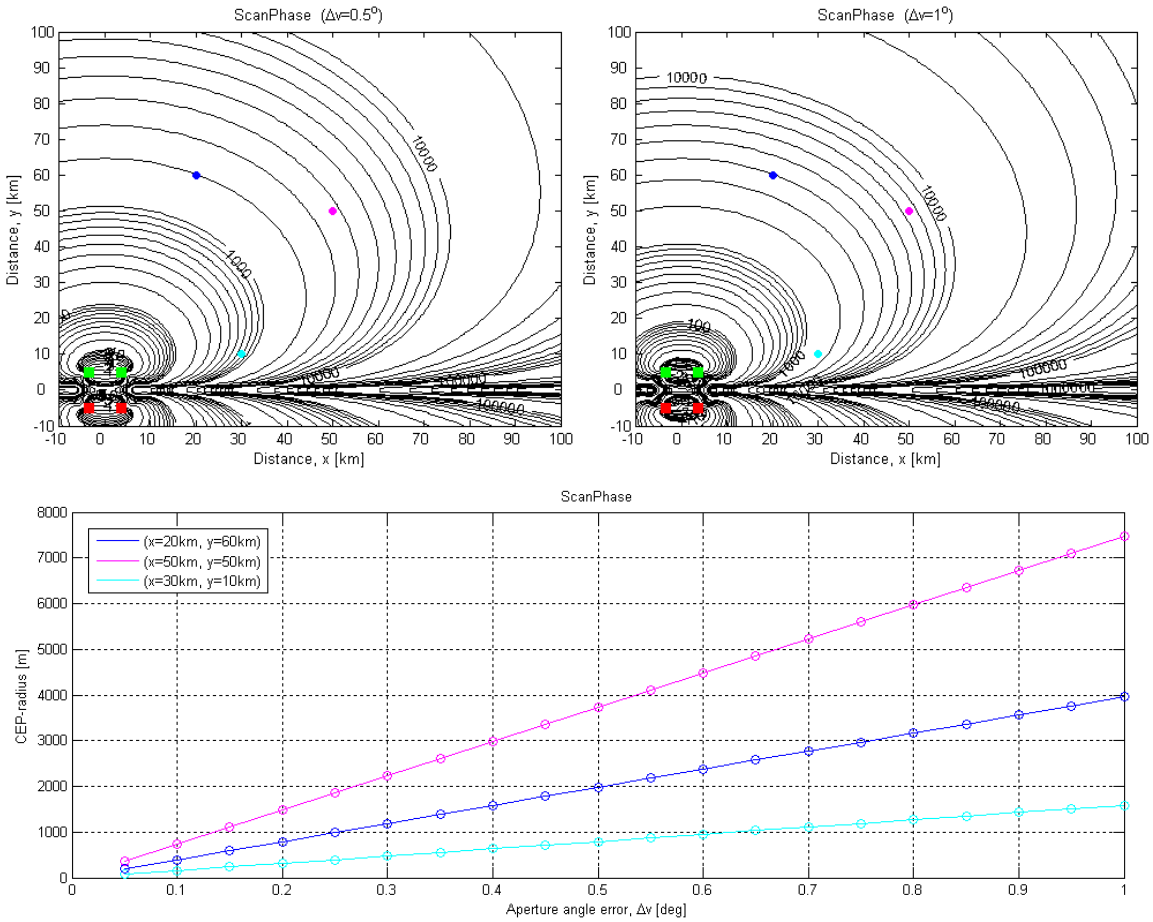


Figure 4.26 CEP-radius dependency on the error (Δv) in the measured aperture angle. Upper left and right figures show CEP-contours for Δv=0.5° and 1° respectively. The sensors move next to each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 361. Lower figure shows CEP-radius as a function of Δv at selected emitter positions (marked by corresponding colours in the upper figures).

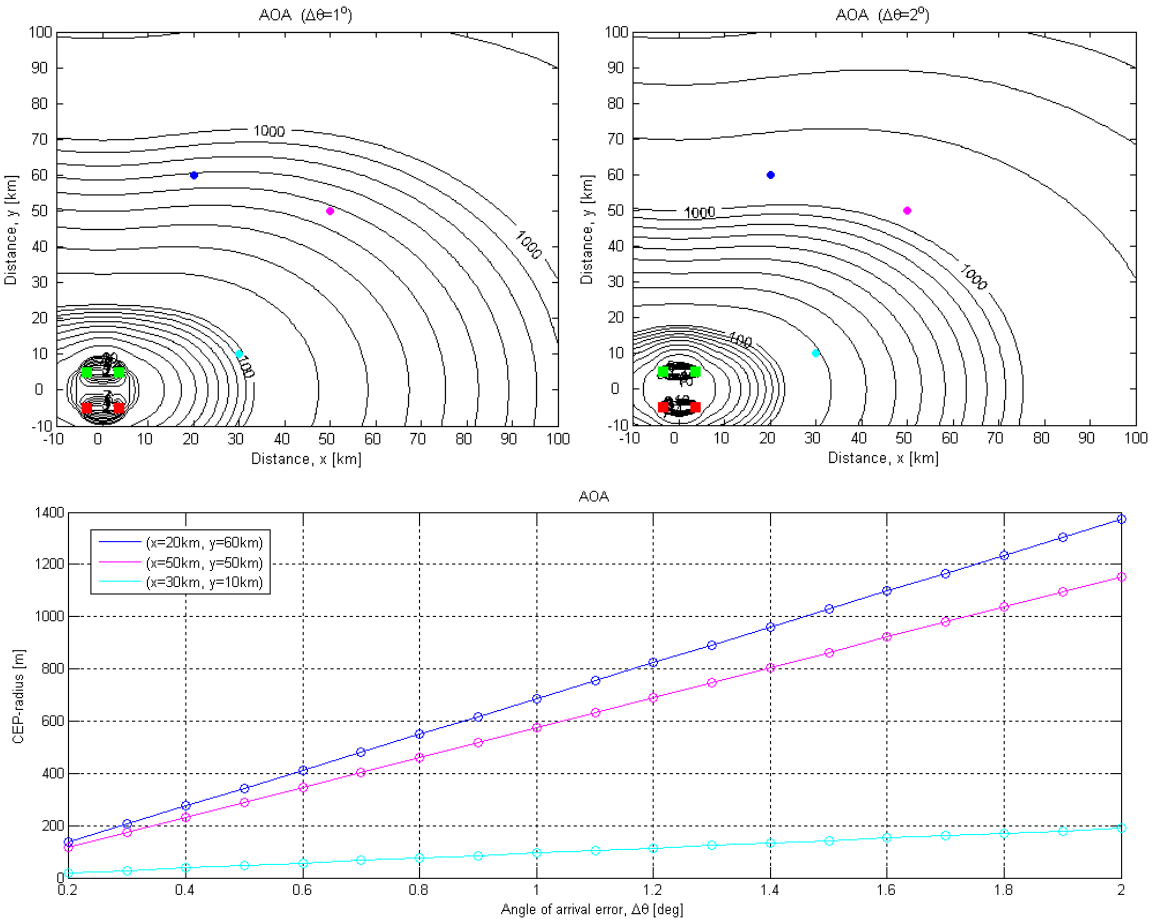


Figure 4.27 CEP-radius dependency on the error in the measured parameter ($\Delta\theta$) in the measured angle of arrival. Upper left and right figures show CEP-contours for $\Delta\theta$ =1° and 2° respectively. The sensors move next to each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the number of measurements is 361. Lower figure shows CEP-radius as a function of $\Delta\theta$ at selected emitter positions (marked by corresponding colours in the upper figures).

4.4.2 Number of measurements

Figure 4.28-Figure 4.30 show the CEP-radius dependency on the number of measurements (N) for the three methods. The sensors move next to each other and the sensor distance is 10 km, the trajectory length is 7.2 km, and the errors in the measured TDOA, aperture angle, and angle of arrival are set to 50 ns, 1º, and 2º respectively. Upper left and right figures show CEP-contours for N=36 and 361 (or N=2x36 and 2x361 for the AOA method), the middle figures show the CEP-radius as a function of the number of measurements at selected emitter positions, and the lower figures show the CEP-radius as a function of $1/\sqrt{N}$.

From the figures it is clear that the CEP-radius is proportional to the inverse of the square root of the number of measurements for all three methods, i.e.,

$$\text{CEP-radius} \sim \frac{1}{\sqrt{N}} \tag{4.2}$$

The minor and major semi-axes of the CRLB error ellipses were found to have the same dependency on the number of measurements as the CEP-radius.

The dependency on the number of measurements will be the same also for other geometries.

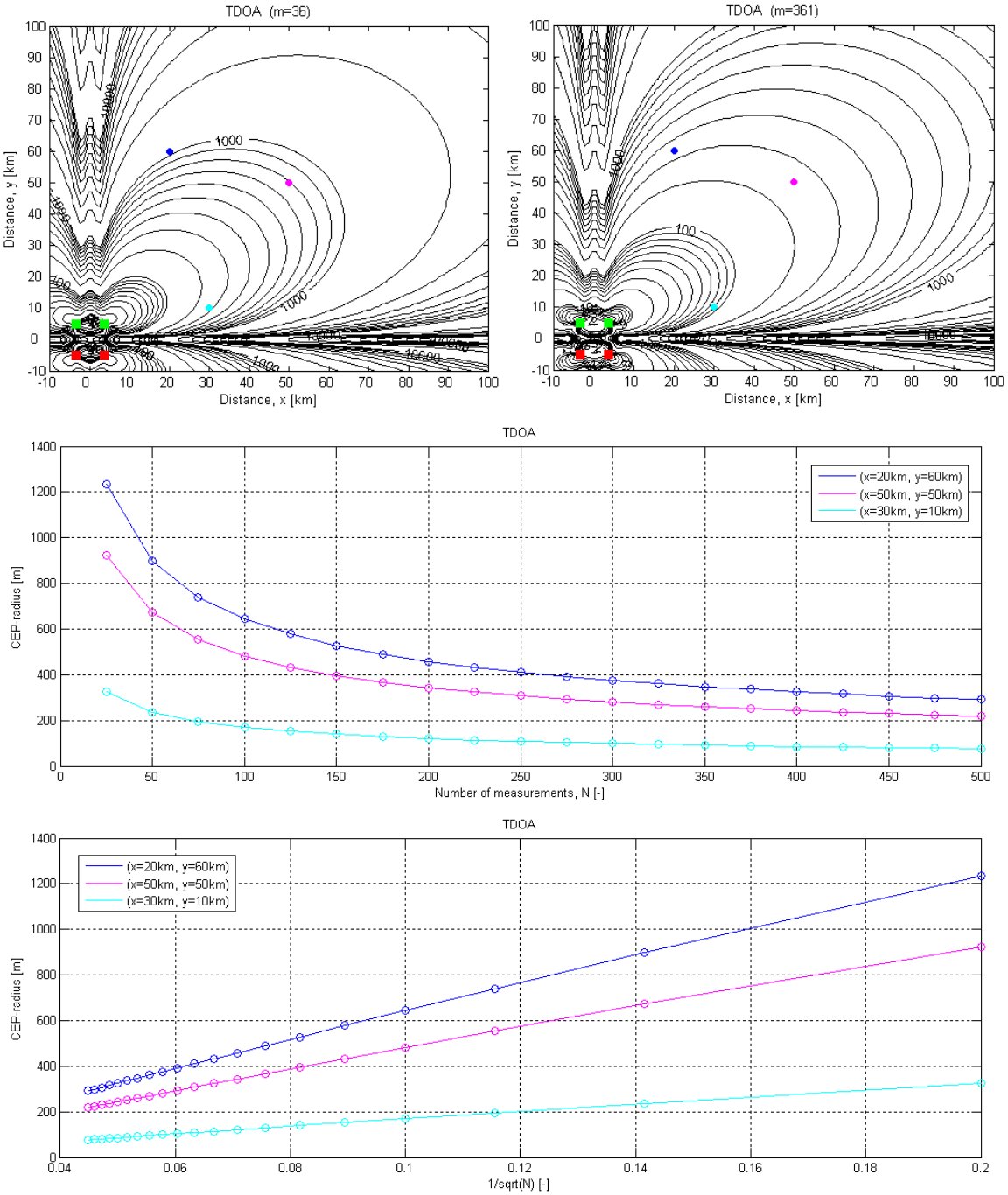


Figure 4.28 CEP-radius dependency on the number of measurements (N) for the TDOA method. Upper left and right figures show CEP-contours for N=36 and 361 respectively. The sensors move next to each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the error in the measured TDOA is set to 50 ns. The figure in the middle shows CEP-radius as a function of N at selected emitter positions (marked by corresponding colours in the upper figures), while the lower figure shows the CEP-radius as a function of 1/√N.

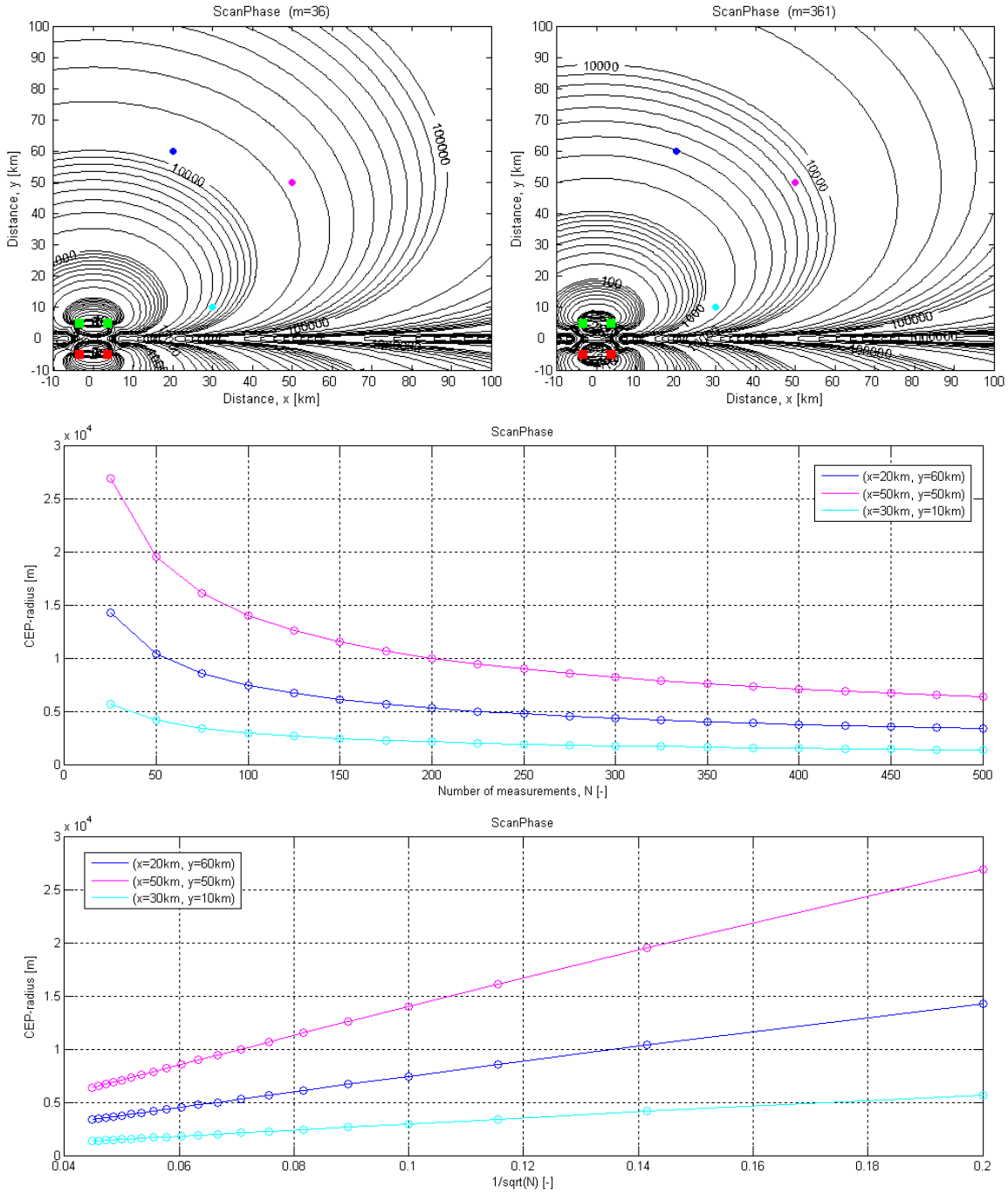


Figure 4.29 CEP-radius dependency on the number of measurements (N) for the scanphase method. Upper left and right figures show CEP-contours for N=36 and 361 respectively. The sensors move next to each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the error in the measured aperture angle is set to 1º. The figure in the middle shows CEP-radius as a function of N at selected emitter positions (marked by corresponding colours in the upper figures), while the lower figure shows the CEP-radius as a function of 1/√N.

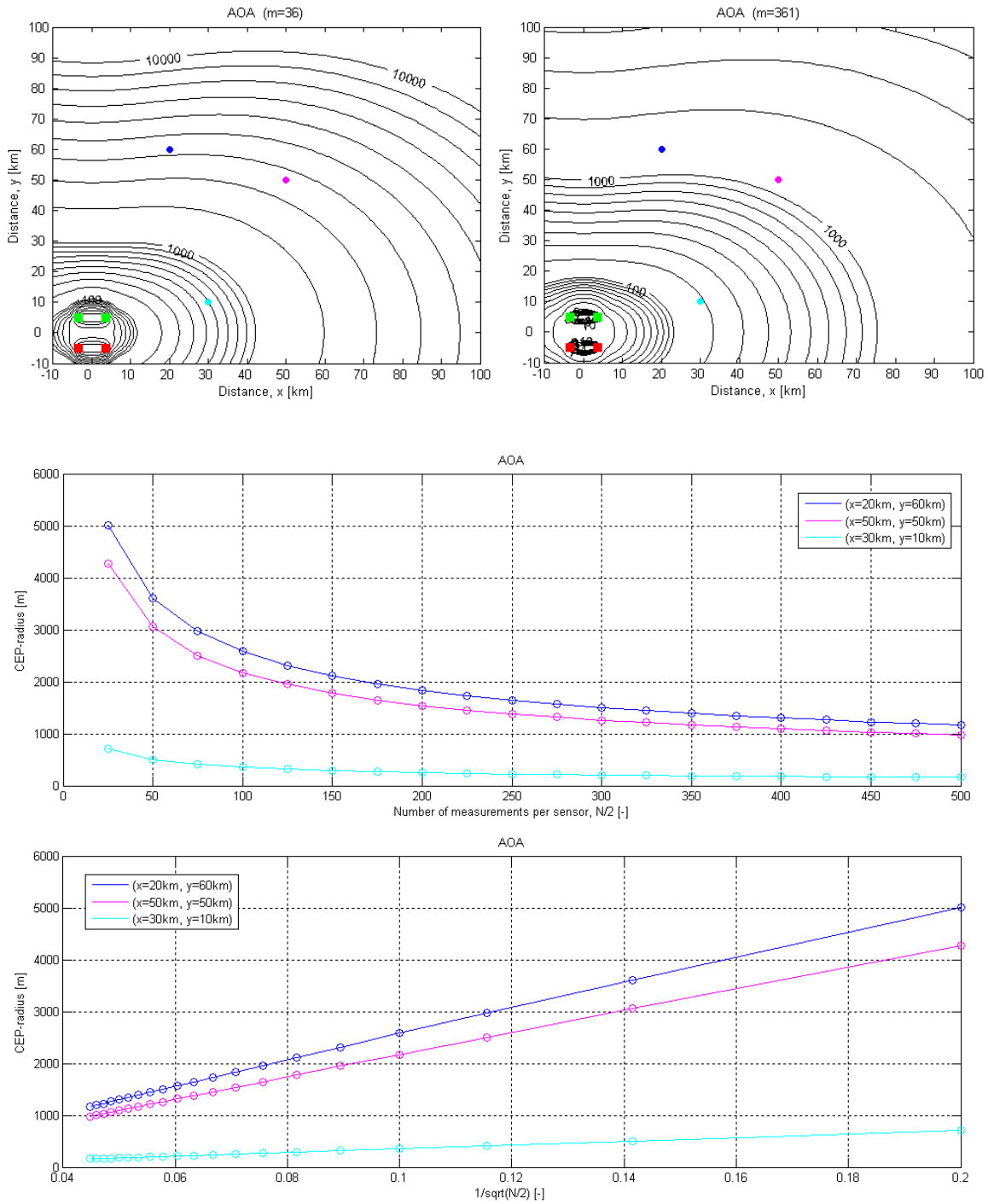


Figure 4.30 CEP-radius dependency on the number of measurements (N) for the AOA method. Upper left and right figures show CEP-contours for N=2x36 and 2x361 respectively. The sensors move next to each other. The sensor distance is 10 km, the trajectory length is 7.2 km, and the error in the measured angle of arrival is set to 2°. The figure in the middle shows CEP-radius as a function of N/2 at selected emitter positions (marked by corresponding colours in the upper figures), while the lower figure shows the CEP-radius as a function of 1/√(N/2).

4.4.3 Sensor distance

Figure 4.31-Figure 4.33 show the CEP-radius dependency on the sensor distance (ΔS) for the three methods. For the TDOA and scanphase methods the sensors move next to each other and the trajectory length is 7.2 km and the number of measurements is 361. For the AOA method the sensors are stationary and the number of measurements is 2x361. The errors in the measured TDOA, aperture angle, and angle of arrival are set to 50 ns, 1º, and 2º respectively. Upper left and right figures show CEP-contours for ΔS=1 km and 10 km, the middle figures show the CEP-radius as a function of sensor distance at selected emitter positions, and the lower figures show the CEP-radius as a function of $1/\Delta S$.

From the figures it is clear that the CEP-radius is proportional to the inverse of the sensor distance for all three methods, i.e.,

$$\text{CEP-radius} \sim \frac{1}{\Delta S} \tag{4.3}$$

For the TDOA and scanphase methods the minor and major semi-axes of the CRLB error ellipses were found to have the same dependency on the sensor distance as the CEP-radius. For the AOA method the major semi-axis has the same dependency as the CEP-radius, while the minor semi-axis is constant as a function of the sensor distance.

For the TDOA and scanphase methods the dependency on the sensor distance was found to be the same also for sensors moving after each other.

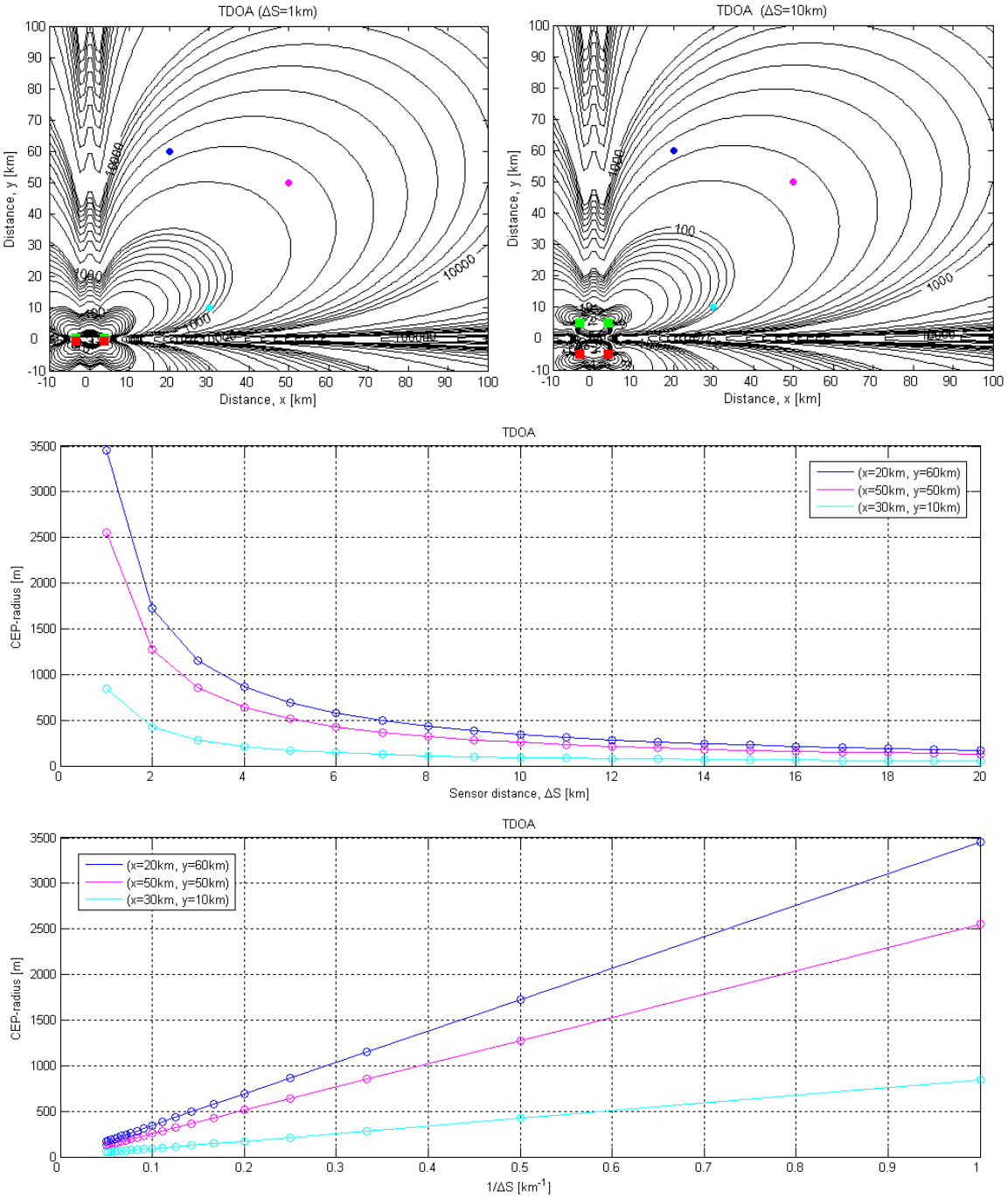


Figure 4.31 CEP-radius dependency on the sensor distance (ΔS) for the TDOA method. Upper left and right figures show CEP-contours for ΔS=1 km and 10 km respectively. The sensors move next to each other. The trajectory length is 7.2 km, the number of measurements is 361, and the error in the measured TDOA is set to 50 ns. The figure in the middle shows CEP-radius as a function of ΔS at selected emitter positions (marked by corresponding colours in the upper figures), while the lower figure shows the CEP-radius as a function of 1/ΔS.

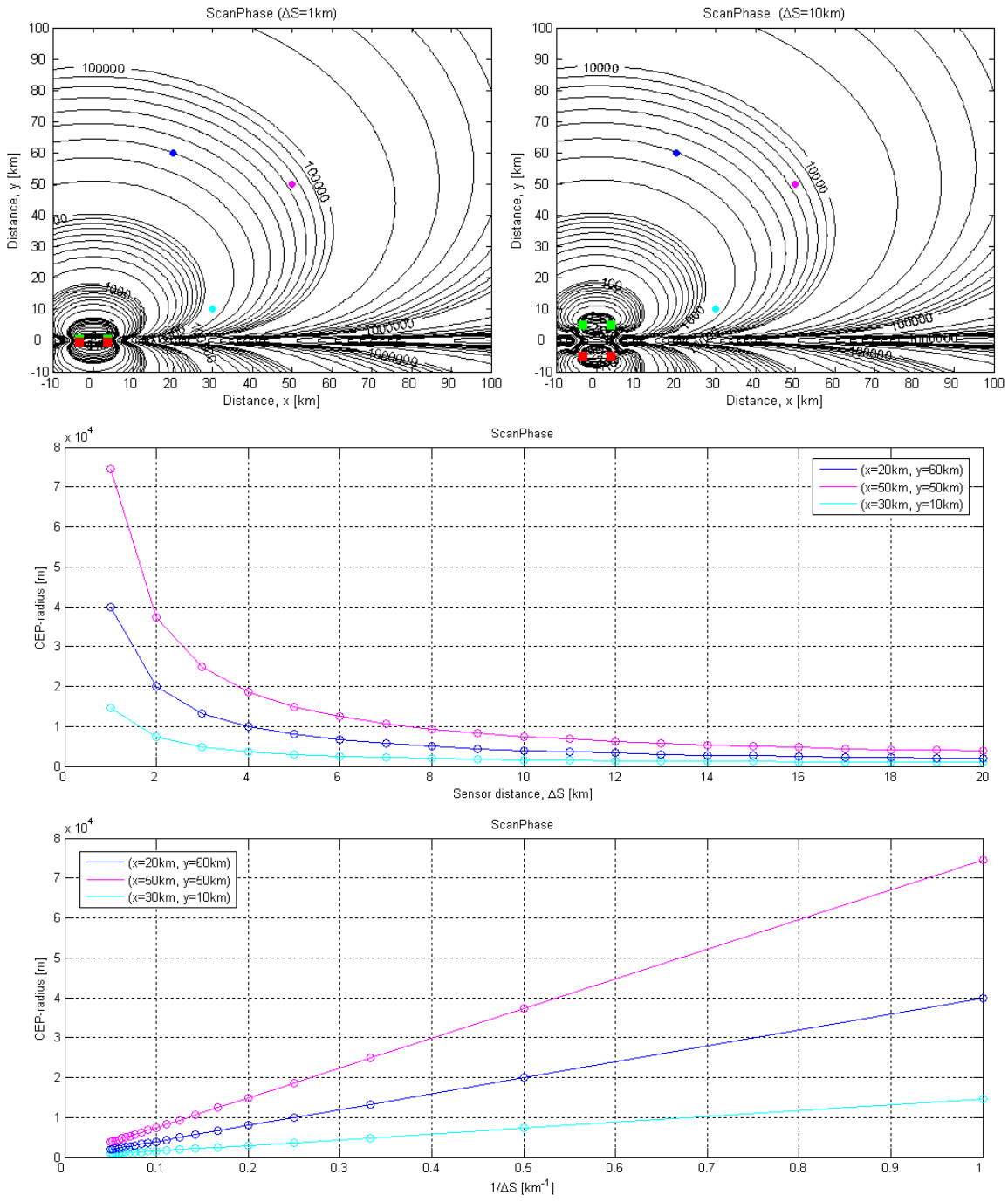


Figure 4.32 CEP-radius dependency on the sensor distance (ΔS) for the scanphase method. Upper left and right figures show CEP-contours for ΔS=1 km and 10 km respectively. The sensors move next to each other. The trajectory length is 7.2 km, the number of measurements is 361, and the error in the measured aperture angle is set to 1°. The figure in the middle shows CEP-radius as a function of ΔS at selected emitter positions (marked by corresponding colours in the upper figures), while the lower figure shows the CEP-radius as a function of 1/ΔS.

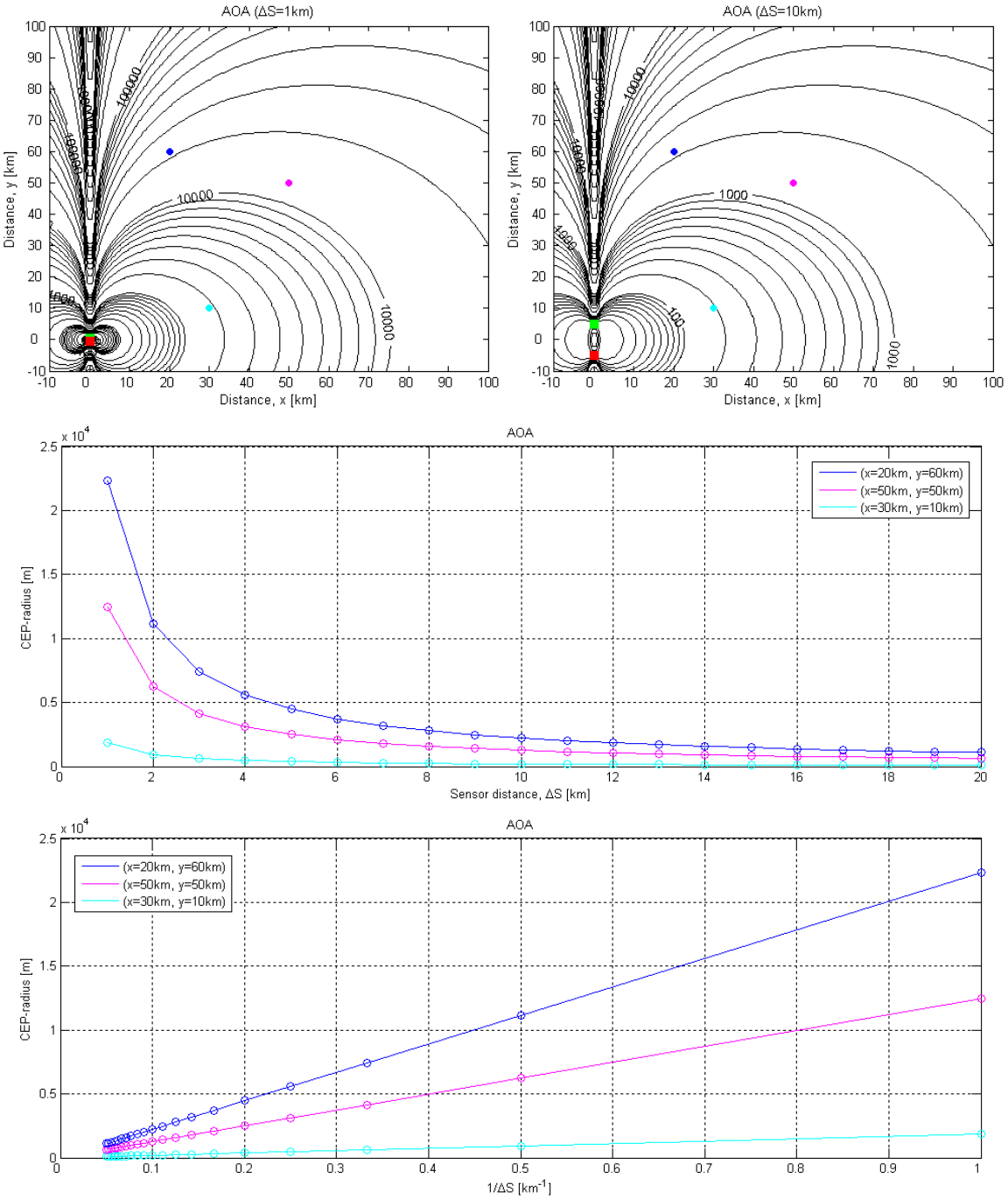


Figure 4.33 CEP-radius dependency on the sensor distance (ΔS) for the AOA method. Upper left and right figures show CEP-contours for ΔS=1 km and 10 km respectively. The sensors are stationary. The number of measurements is 2x361 and the error in the measured angle of arrival is set to 2º. The figure in the middle shows CEP-radius as a function of ΔS at selected emitter positions (marked by corresponding colours in the upper figures), while the lower figure shows the CEP-radius as a function of 1/ΔS.

4.4.4 Trajectory length

Figure 4.34-Figure 4.36 show the CEP-radius dependency on the trajectory length (ΔL) for the three methods. For the TDOA and scanphase methods the sensors move next to each other and the sensor distance is 10 km. For the AOA method one sensor is used that moves along a straight line. The number of measurements is 361 and the errors in the measured TDOA, aperture angle, and angle of arrival are set to 50 ns, 1º, and 2º respectively. Upper left and right figures show CEP-contours for ΔL=3.6 km and 7.2 km, the middle figures show the CEP-radius as a function of trajectory length at selected emitter positions, and the lower figures show the CEP-radius as a function of $1/\Delta L$.

From the figures it is clear that the CEP-radius is proportional to the inverse of the trajectory length for all three methods, i.e.,

$$\text{CEP-radius} \sim \frac{1}{\Delta L} \tag{4.4}$$

The major semi-axis of the CRLB error ellipses was found to have the same dependency on the trajectory length as the CEP-radius, while the minor semi-axis was found to be constant as a function of the trajectory length for all three methods.

For the TDOA and scanphase methods the dependency on the trajectory length was found to be the same also for sensors moving after each other.

The dependency on the trajectory length for circular sensor movement was also investigated for all three methods. The ratio between the trajectory length (ΔL) and the sensor distance (ΔS) was then kept constant, i.e., $\Delta L = \pi \cdot \Delta S$, so that the sensor or pair of sensors always completed one full circle. This means that we investigated the situation were both the trajectory length and the sensor distance increased or decreased with the same amount simultaneously.

For the AOA and scanphase methods the CEP-radius was in this case found to be proportional to the inverse of the trajectory length. For the TDOA method the CEP-radius was found to be proportional to the inverse of the cube of the trajectory length.

The major semi-axis was found to have the same dependency on the trajectory length as the CEP-radius for all three methods.

The minor semi-axis was found to be proportional to the inverse of the trajectory length for the TDOA and scanphase methods, and to be constant as a function of the trajectory length for the AOA method.

Note that the above described dependency on trajectory length for cirular sensor movement is in fact a dependency on both the trajectory length *and* the sensor distance.

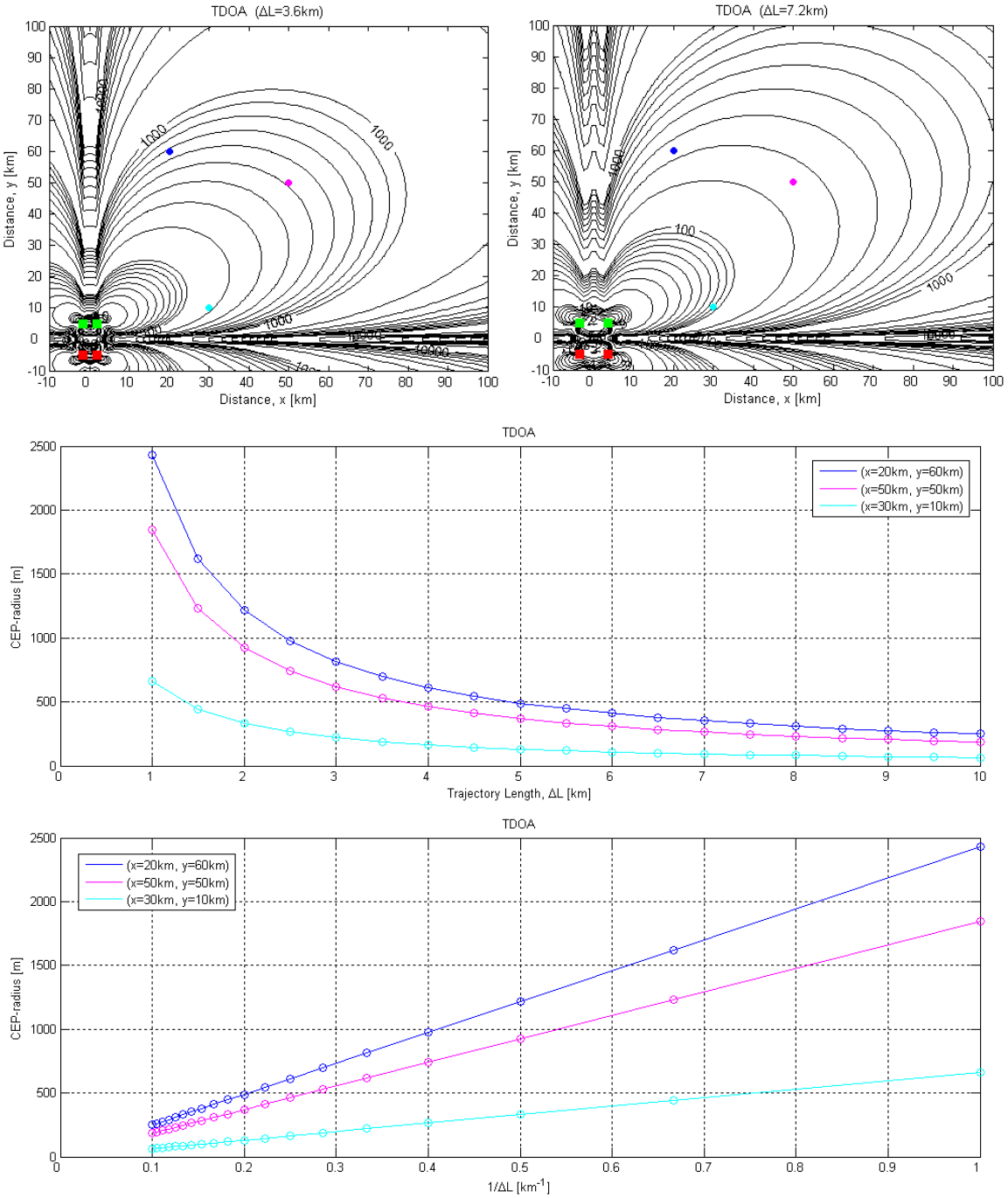


Figure 4.34 CEP-radius dependency on the trajectory length (ΔL) for the TDOA method. Upper left and right figures show CEP-contours for ΔL=3.6 km and 7.2 km respectively. The sensors move next to each other. The sensor distance is 10 km, the number of measurements is 361, and the error in the measured TDOA is set to 50 ns. The figure in the middle shows CEP-radius as a function of ΔL at selected emitter positions (marked by corresponding colours in the upper figures), while the lower figure shows the CEP-radius as a function of 1/ΔL.

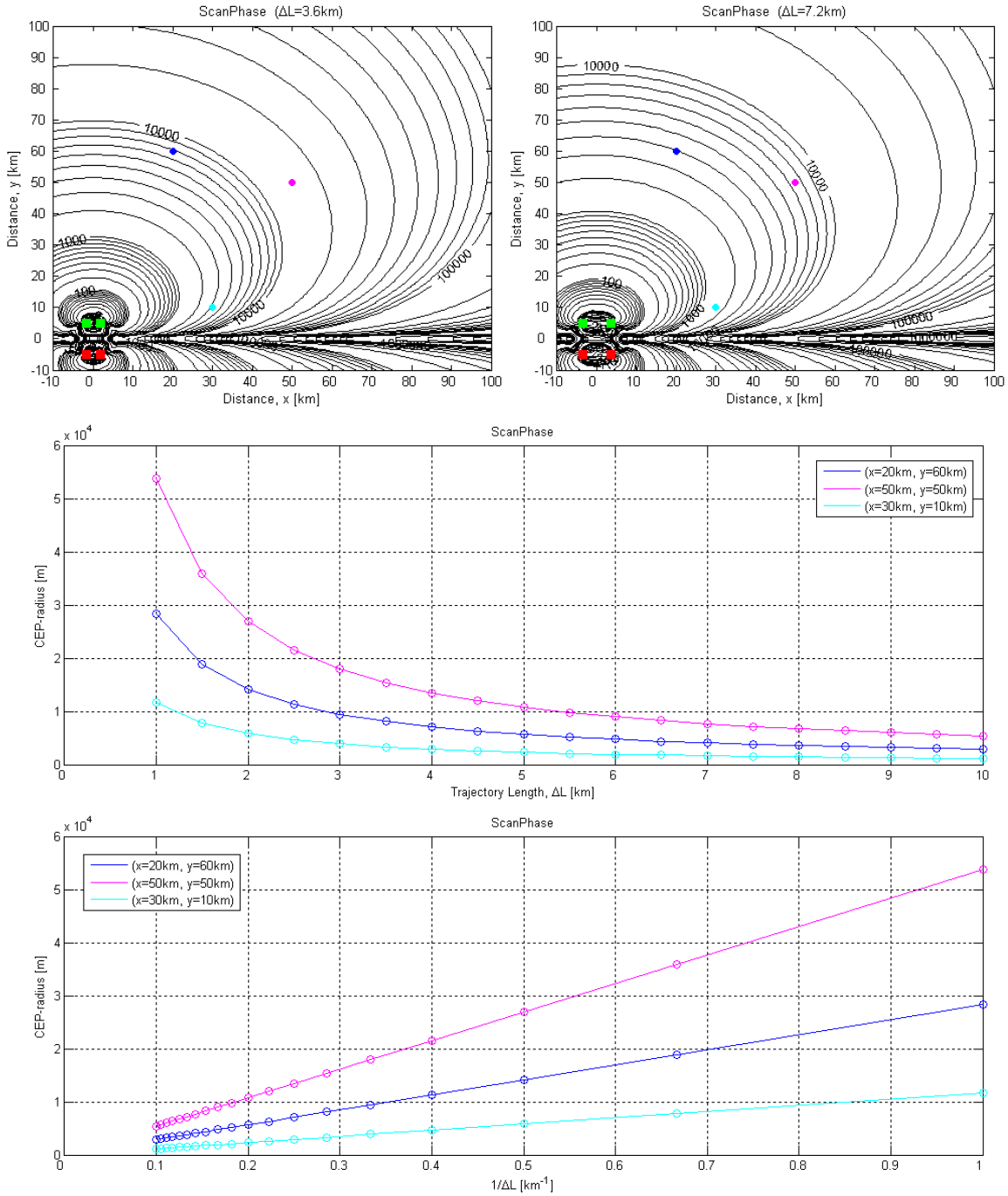


Figure 4.35 CEP-radius dependency on the trajectory length (ΔL) for the scanphase method. Upper left and right figures show CEP-contours for ΔL=3.6 km and 7.2 km respectively. The sensors move next to each other. The sensor distance is 10 km, the number of measurements is 361, and the error in the measured aperture angle is set to 1º. The figure in the middle shows CEP-radius as a function of ΔL at selected emitter positions (marked by corresponding colours in the upper figures), while the lower figure shows the CEP-radius as a function of 1/ΔL.

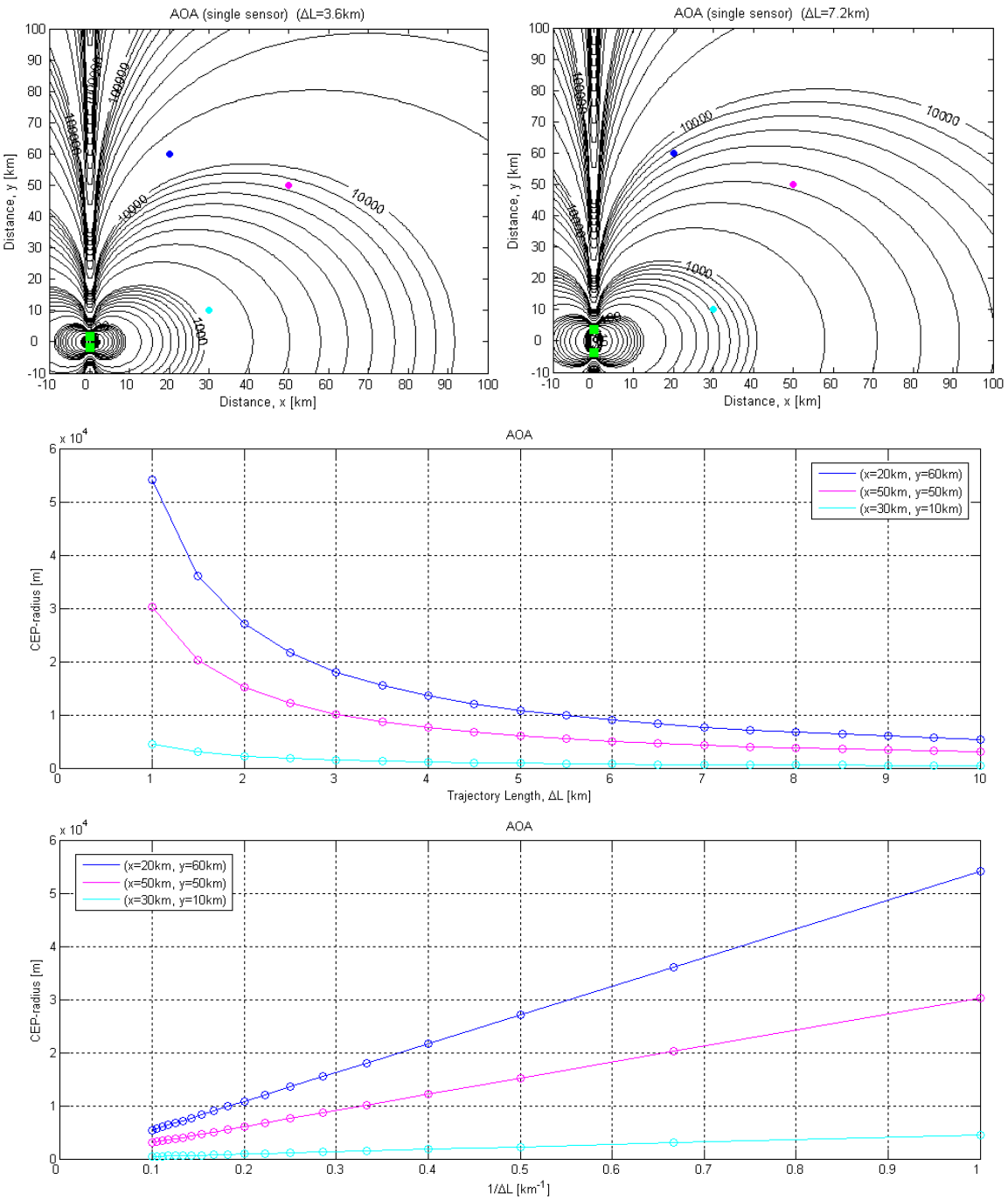


Figure 4.36 CEP-radius dependency on the trajectory length (ΔL) for the AOA method. Upper left and right figures show CEP-contours for ΔL=3.6 km and 7.2 km respectively. The number of measurements is 361, and the error in the measured angle of arrival is set to 2°. The figure in the middle shows CEP-radius as a function of ΔL at selected emitter positions (marked by corresponding colours in the upper figures), while the lower figure shows the CEP-radius as a function of 1/ΔL.

4.4.5 Radial distance

We have studied the CEP-radius dependency on the radial distance (R) from the sensors for the three methods for the following types of sensor movement:

1) Sensors moving *after* each other
2) Sensors moving *next to* each other
3) Sensors moving on a *circular* path

For sensors moving on a circular path, the CEP-radius dependency is the same in all directions. When the sensors move along a straight line (either after or next to each other), the CEP-radius dependency will be different in different directions. We have then looked at the CEP-radius dependency along the central sensor line (calculations performed 500 m to the side of the line), the tilted (45º) line, and the central middle line (calculations performed 500 m to the side of the line). Table 4.16 shows the results. The table also includes the minor and major semi-axes' dependency on the radial distance for the corresponding CRLB error ellipses.

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="2">Central sensor line</th><th colspan="2">Tilted (45º) line</th><th colspan="2">Central middle line</th><th rowspan="2">Circular</th></tr>
<tr><th>After</th><th>Next to</th><th>After</th><th>Next to</th><th>After</th><th>Next to</th></tr>
<tr><td rowspan="3">TDOA</td><td>CEP-radius</td><td rowspan="2">$\sim R^4$</td><td rowspan="2">$\sim R^3$</td><td colspan="2" rowspan="2">$\sim R^2$</td><td colspan="2" rowspan="2">$\sim R^3$</td></tr>
<tr><td>Major semi-axis</td></tr>
<tr><td>Minor semi-axis</td><td colspan="2">$\sim R^2$</td><td colspan="4">$\sim R$</td></tr>
<tr><td rowspan="3">Scanphase</td><td>CEP-radius</td><td rowspan="2">$\sim R^4$</td><td colspan="3" rowspan="2">$\sim R^3$</td><td rowspan="2">$\sim R^4$</td><td rowspan="2">$\sim R^2$</td></tr>
<tr><td>Major semi-axis</td></tr>
<tr><td>Minor semi-axis</td><td colspan="6">$\sim R^2$</td></tr>
<tr><td rowspan="3">AOA</td><td>CEP-radius</td><td rowspan="2">$\sim R^{3\,(*)}$</td><td colspan="5" rowspan="2">$\sim R^2$</td></tr>
<tr><td>Major semi-axis</td></tr>
<tr><td>Minor semi-axis</td><td colspan="6">$\sim R$</td></tr>
</table>

(*) Valid also for 2 stationary sensors and in the velocity direction for 1 sensor moving along a straight line.

Table 4.16 CEP-radius dependency on the radial distance from the sensors. Results for the minor and major semi-axes of the corresponding CRLB error ellipses are also included.

Table 4.16 shows that for all three methods the CEP-radius and the major semi-axis increase rapidly with increasing radial distance from the sensors. For the AOA method the CEP-radius and major semi-axis increase as $\sim R^2$ or $\sim R^3$ depending on the sensor type of movement and direction we look along. For the TDOA method the increase is $\sim R^2$, $\sim R^3$ or $\sim R^4$. For the scanphase method the CEP-radius and major semi-axis increase as $\sim R^3$ or $\sim R^4$ when the sensors move along a straight line, while only as $\sim R^2$ when the sensors move on a circular path. This supports the findings in Section 4.1.2, where we saw that circular sensor movement is particularly beneficial for the scanphase method.

The minor semi-axis increases more slowly as function of radial distance than the CEP-radius and the major semi-axis. For the AOA method the increase is linear (~R), while for the scanphase method the increase is proportional to the square of the radial distance (~R^2). For the TDOA method the increase can be both linear (~R) and proportional to the square of the radial distance (~R^2), depending on the sensor type of movement and direction we look along.

4.4.6 Summary

We have studied the CEP-radius dependency on different parameters for the three methods. The CEP-radius was found to be proportional to the error in the measured parameter (ΔTDOA, Δv, or $\Delta\theta$), proportional to the inverse of the sensor distance (ΔS), proportional to the inverse of the trajectory length (ΔL), and proportional to the inverse of the square root of the number of measurements (N). These results are in agreement with earlier findings for the TDOA method [8]. The CEP-radius dependency on the radial distance (R) from the sensors is ~R^2, ~R^3 or ~R^4 for the TDOA and scanphase methods and ~R^2 or ~R^3 for the AOA method depending on the geometry. Table 4.17 summarizes the results.

The major semi-axis of the corresponding CRLB error ellipses behave similarly to the CEP-radius for all parameters and methods. For the minor semi-axis there are some exceptions: The minor semi-axis is constant with respect to the trajectory length for all methods and with respect to the sensor distance for the AOA method. The minor semi-axis dependency on the radial distance from the sensors is ~R or ~R^2 for the TDOA method, ~R for the AOA method, and ~R^2 for the scanphase method.

Parameter	**CEP-radius dependency on different parameters**		
	TDOA	**Scanphase**	**AOA**
Error in measured parameter	~ ΔTDOA	~ Δv	~ $\Delta\theta$
Number of measurements (N)	~ $\frac{1}{\sqrt{N}}$		
Sensor distance (ΔS)	~ $\frac{1}{\Delta S}$ (†*)		
Trajectory length (ΔL)	~ $\frac{1}{\Delta L}$ (†**)		
Radial distance (R)	~ R^2, R^3, R^4		~ R^2, R^3

(†) TDOA and Scanphase methods: Valid for sensors moving next to or after each other.

(*) AOA-method: Valid for 2 stationary sensors.

(**) AOA-method: Valid for 1 sensor.

Table 4.17 CEP-radius dependency on different parameters.

4.5 Additional factors that may affect the geolocation accuracy

There are several additional factors that may affect the geolocation accuracy. We will consider some of these below.

4.5.1 Error in sensor position

Errors in the sensor position (Δp) can be converted to errors in the time difference of arrival (ΔTDOA), aperture angle (Δv), and angle of arrival ($\Delta\theta$).

4.5.1.1 Effect on the angle of arrival

Only sensor position errors in the angular direction (Δp_θ) affect the angle of arrival, see Figure 4.37. The error in the angle of arrival ($\Delta\theta_p$) resulting from an error in the sensor position can then be written

$$\Delta\theta_p = \frac{\Delta p_\theta}{R} \tag{4.5}$$

where R is the distance between the sensor and the emitter. A sensor position error of 10 m in the angular direction will then give an error in the angle of arrival of about 0.6º when the distance to the sensor is 1 km. The error decreases with increasing distance between the sensor and the emitter.

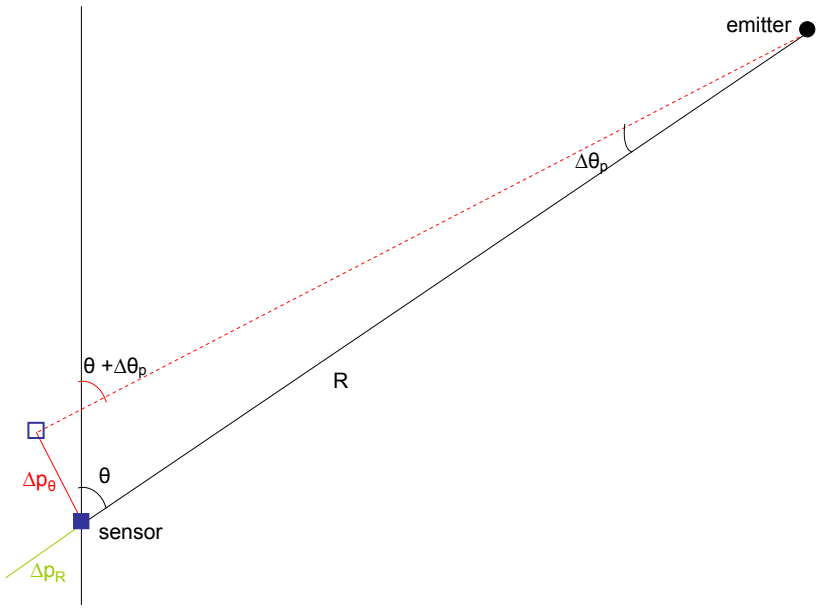


Figure 4.37 Error in the angle of arrival $\Delta\theta_p$ resulting from an error in the sensor position Δp_θ.

4.5.1.2 Effect on the aperture angle

Only sensor position errors in the angular direction (Δp_v) affects the aperture angle, see Figure 4.38. The error in the aperture angle (Δv_p^{tot}) resulting from errors in the sensor positions can then be written

$$\Delta v_p^{tot} = \sqrt{\left(\Delta v_{1p}\right)^2 + \left(\Delta v_{2p}\right)^2} = \sqrt{\left(\frac{\Delta p_{1v}}{R_1}\right)^2 + \left(\frac{\Delta p_{2v}}{R_2}\right)^2} \tag{4.6}$$

where R_1 is the distance between the emitter and sensor 1 and R_2 is the distance between the emitter and sensor 2. A sensor position error of 10 m in the angular direction will then give an error in the aperture angle of about 0.8º when the distance from the emitter to each sensor is 1 km. The error decreases with increasing distance between the sensors and the emitter.

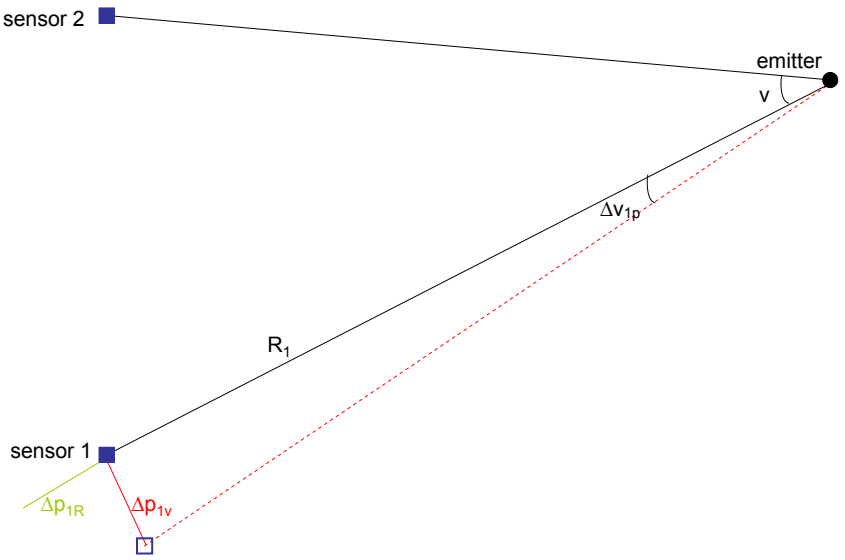


Figure 4.38 Error in the aperture angle Δv_{1p} resulting from an error in the sensor position Δp_{1v}.

4.5.1.3 Effect on the time difference of arrival

Only sensor position errors in the radial direction (Δp_R) affect the time difference of arrival, see Figure 4.39. The error in the time difference of arrival ($\Delta TDOA_p^{tot}$) resulting from errors in the sensor positions can then be written

$$\Delta TDOA_p^{tot} = \sqrt{\left(\Delta TDOA_{1p}\right)^2 + \left(\Delta TDOA_{2p}\right)^2} = \frac{1}{c}\sqrt{\left(\Delta p_{1R}\right)^2 + \left(\Delta p_{2R}\right)^2} \tag{4.7}$$

where c is the speed of light. A sensor position error of 10 m in the radial direction for each sensor will then give an error in the time difference of arrival of about 50 ns.

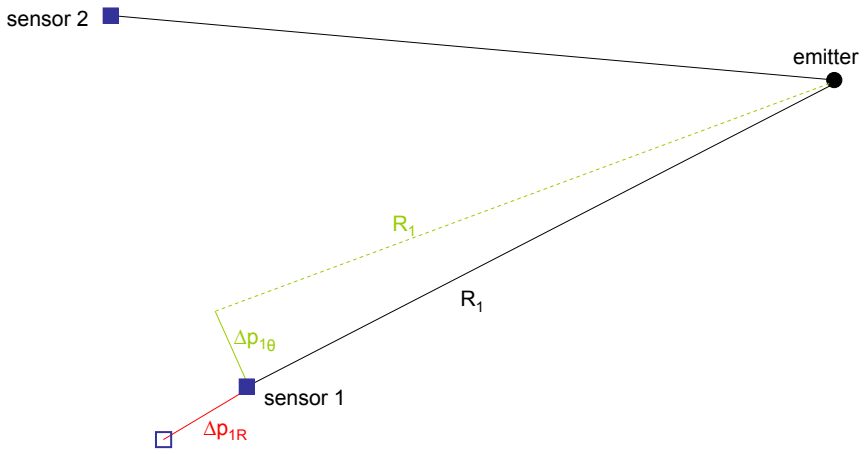


Figure 4.39 Error in the time difference of arrival $\Delta TDOA_{1p}$ resulting from an error in the sensor position Δp_{1R}.

4.5.2 Emitter movements

Emitter movements during the observation period will affect the geolocation accuracy. If the emitter movements are small compared to the CEP-radius, the additional geolocation error will be negligible. If the emitter movements are comparable to or larger than the CEP-radius, the geolocation accuracy will decrease compared to what could have been achieved. A shorter observation period should then be considered. Several separate measurements of the emitter's position could then instead be performed and used to estimate the speed and direction of the emitter.

4.5.3 Received energy

The amount of energy that the sensors receive from the emitter affects how precisely the time difference of arrival, the aperture angle, and the angle of arrival can be measured. How much energy that is received from a particular emitter is determined by several factors such as:

- The emitter power.
- The emitter antenna gain in the direction of the sensor.
- The distance between the sensor and the emitter.
- Losses due to atmospheric attenuation, etc.

Since the received energy decreases with the square of the distance, the geolocation error will in reality increase faster with increasing distance from the sensors than what we found in Section 4.4.5.

4.5.4 De-interleaving capability

The sensors may receive pulses from several emitters during the observation period. The ability to geolocate the emitters depends on the sensors' capability to separate pulses from different emitters, as well as the ability to connect a particular emitter seen by one sensor to the correct

emitter seen by the other sensor. For the TDOA method it is necessary to know which pulse on one sensor corresponds to which pulse on the other sensor, since this method is based on determining the pulse time difference of arrival at the two sensors. For the scanphase and AOA methods it is sufficient to identify that we look at the same emitter on both sensors.

In order to find pulses that potentially belong together on one sensor, the measured angle of arrival and frequency can be used.

In order to find which pulses on one sensor that belong together with which pulses on the other sensor, a TDOA histogram based on the measured times of arrival (TOA) can be used. The TOA for each pulse on one sensor is compared to the TOA for each pulse on the other sensor and the time difference of arrival (TDOA) is calculated for each pulse pair. By plotting a histogram that shows the number of pulse pairs as a function of TDOA, the true TDOAs can easily be identified, see Figure 4.40. When the true TDOAs are known, we also know which pulse on one sensor that belongs together with which pulse on the other sensor. In addition, we know which pulses belong to the same pulse train.

Errors in de-interleaving, such as combining wrong pulses from the two sensors or including pulses from other emitters, will affect the geolocation accuracy. One source of error in the de-interleaving process is multi-path, which may lead to wrong values for the measured time of arrival or angle of arrival for a pulse.

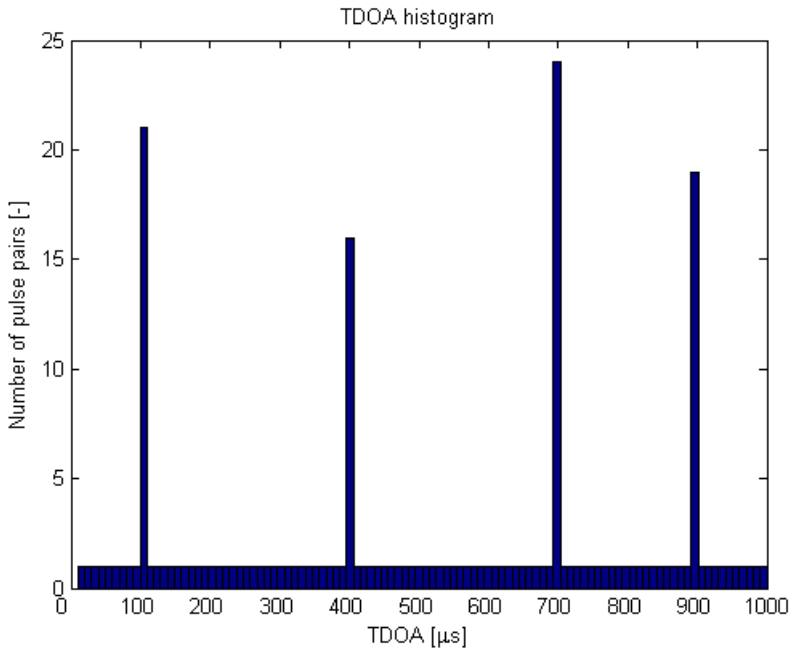


Figure 4.40 TDOA histogram. True TDOAs are in this example found at 100 µs, 400 µs, 700 µs, and 900 µs.

4.5.5 Data transfer capacity

The pulse density in a maritime scenario is often very high, since each emitter may generate up to 100 000 pulses per second. With several emitters present, the number of pulses may easily reach 1 million per second or more. In order to utilize all the information available for geolocation, each pulse received by one sensor should be transferred to the other. However, today's communication systems have limited capacity, and only a small part of the pulses can therefore be tranferred and used for geolocation purposes. This affects the geolocation accuracy directly since the geolocation error increases with a factor √2 when the number of pulses used for the calculations is reduced by a factor 2.

The following pulse parameters should be transferred from one sensor to the other:

- TOA [TDOA + Scanphase + de-interleaving]
- AOA [AOA + de-interleaving]
- Amplitude [Scanphase]
- Frequency [de-interleaving]

The sensors used for the measurements will normally produce pulse description words (PDWs) that contain this information. Each PDW may typically occupy 128 bits. We will consider three different data transfer rates which represent typical low, medium, and high rate data links of today, see Table 4.18. With the low rate data link (IDM technology, 10 kb/s) the number of pulses (or PDWs) that can be transferred per second is 78. With the medium rate data link (satellite communications, 512 kb/s) this number increases to 4000, while a modern data link (16 Mb/s) may transfer as many as 125 000 pulses per second. Note, however, that these numbers are theoretical values. The actual values will depend upon the choice of protocol and strategy for the data transfer and may be considerably lower than indicated here.

Data link	**Data transfer rate**	**# PDWs transferred per second**
NATO IDM	10 kb/s	78
Satellite communications	512 kb/s	4000
Modern data link	16 Mb/s	125 000

Table 4.18 Data transfer rates.

From Table 4.18 it is clear that even with a modern data link only a fraction of the pulses in a typical maritime scenario can be transferred in real time[4]. A possible way to reduce the amount of data that needs to be transferred is to use emitter description words (EDWs) instead of PDWs. This may reduce the amount of data by a factor ~100 or more, while giving approximately the same accuracy as if each pulse was transferred since average values are used for the transferred parameters [9].

[4] Note that both the IDM technology and satellite communications have a typical communications latency of 3-4 seconds. A future modern data linke may have a latency of 1-2 seconds.

The EDWs should contain the following information:

- Emitter ID (number or other identification tag)
- TOA (first pulse) [TDOA]
- PRI (average value) [TDOA]
- AOA (average value) [AOA]
- TOA for radar mainlobe [Scanphase]
- Number of pulses [TDOA + error estimate]

When EDWs are used, the pulses must be de-interleaved on each sensor separately before the data transfer takes place.

Since the amount of data in most cases will be considerably larger than the data transfer capacity, strategies for selecting the data to transfer must be considered. One possibility is to transfer all data that is collected in a short time interval, while ignoring data received outside this interval. Another possibility would be to use a limited amount of data from each emitter. It is also possible to consider only narrow frequency bands, or to agree in advance on which emitters should be ignored (for instance all emitters that are already known), focusing on emitters that are of particular interest.

4.6 Trajectory recommendations

When choosing the sensor trajectory for geolocation, the following general guidelines should be followed:

1) The sensors must be within range of each other.
2) The sensors must be able to observe the area of interest simultaneously.
3) The sensors should be as close as possible to the area of interest.
4) The distance between the sensors should be as long as possible.
5) The trajectory should be as long as possible.
6) The type of sensor movement should be:
    i) Circular (scanhase method alone or AOA method alone with one sensor)
    ii) Zig-zag (TDOA method alone)
    iii) Next to (all other combinations of methods)

The sensors must be within range of each other in order to be able to communicate and exchange data. Both sensors must also be able to observe the area of interest simultaneously for the geolocation measurements to be made.

Since the geolocation accuracy decreases with increasing distance to the target, the sensors should be as close as possible to the area of interest.

The distance between the sensors should be as long as possible since the geolocation accuracy increases with longer sensor distance. The upper limit on the sensor distance will be determined by the requirements that the sensors must be within range of each other and that both sensors must be able to observe the area of interest simultaneously.

The sensor trajectory should also be as long as possible, since a longer trajectory increases the geolcoation accuracy. The upper limit on the trajectory length will be determined by the maximum possible speed of the sensor and the maximum acceptable observation time before a geolocation is made. The requirement that both sensors must be able to observe the area of interest simultaneously may also impose some limitations. Geographical limitations such as proximity to land or sea borders that cannot be crossed come in addition.

The sensors should move along a circular path when the scanphase method is used alone or when the AOA method is used alone with one sensor. For the TDOA method alone a zig-zag path is the best option, while for all other methods and combinations of methods the sensors should move next to each other. Geolocation is then possible everywhere and the problem with mirror images is avoided. Figure 4.41 shows an example of trajectory choice.

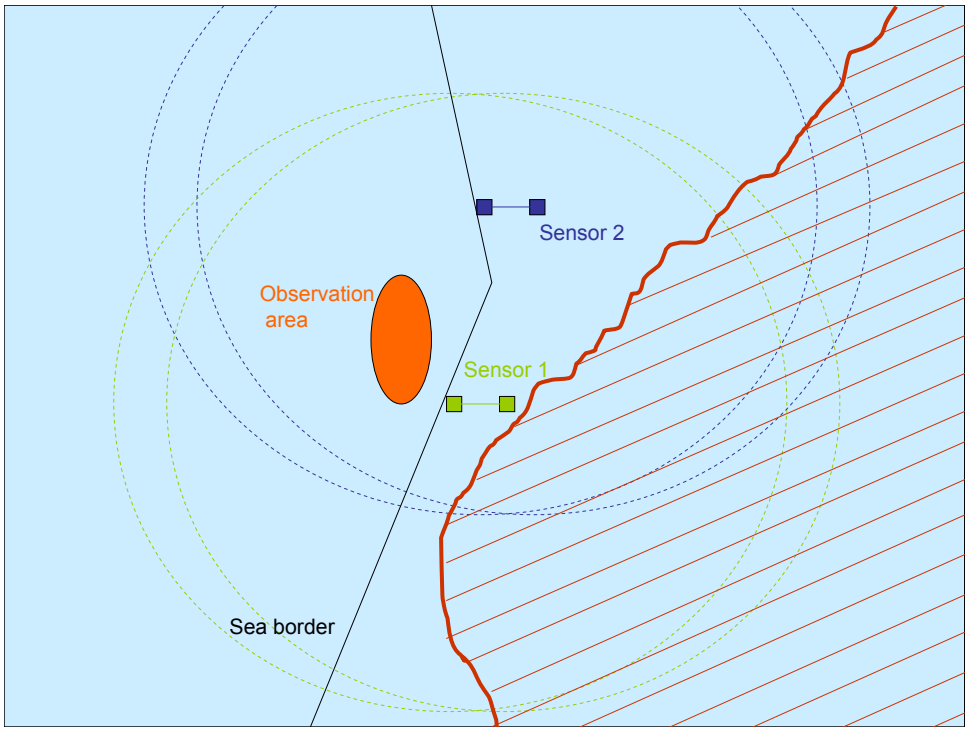


Figure 4.41 Example of trajectory choice when there are additional constraints such as proximity to land and a sea border that should not be crossed.

4.7 Further work

The geolocation accuracies that we have found in this report are based upon assumptions about the accuracy that can be obtained for the time difference of arrival, aperture angle, and angle of arrival measurements. The assumed values may or may not be close to real values. A good estimate of the geolocation accuracy depends on knowledge about the real values (typical size) of the errors in the measured parameters. This is something that should be investigated further, preferably through analyses of real data.

An important factor that may limit the geolocation accuracy that can be obtained, is the data transfer capacity. If only a small part of the data can be exchanged between the sensors, the geolocation accuracy will be noticeably worse than it could have been. Both the available data transfer rate and the transfer strategy will be of importance. Further investigations regarding realistic data transfer rates and optimum transfer strategies should therefore be performed.

The three localization methods have different requirements both to the sensors and to the additional equipment that will be necessary to do the measurements with sufficient accuracy. It should be discussed in more detail what additional equipment, including the necessary datalinks, that must be installed on the vessels in order to utilize the localization methods described in this report.

This report has focused on the geolocation *accuracy*. The next step should be to look at the actual geolocation estimation, i.e., how to find the position of the emitter. The estimation algorithm described in [10] and [11] could be used for this. This algorithm has already been applied by the GEOIDE projects at FFI for geolocation of emitters [15]. Another possibility would be to look at the Target Motion Analysis (TMA)-algorithm that is being used by the submarine group at FFI for geolocation and tracking of targets [12]. Some initial studies on the TMA-algorithm for use on maritime sensors have already been performed by two summer students at FFI [13].

5 Summary

New and more precise geolocation methods are considered for implementation on Norwegian navy vessels. The most promising methods for geolocating emitters from shipborne sensors are:

1) Time difference of arrival (TDOA) measurements
2) Scanphase measurements
3) Angle of arrival (AOA) measurements

Precise angle of arrival measurements can be performed with the new direction-finding sensors that are now being installed on the Norwegian frigates. Time difference of arrival measurements will in addition require the installation of a very precise clock, such as for instance a rubidium clock, that can measure time with a precision of a few tens of nanoseconds. Both the TDOA and scanphase methods require the use of two sensors.

In this report, we have studied the geolocation accuracy that can be obtained from shipborne sensors with the three methods. Each method has been studied separately, but we have also looked at different combinations of the methods. Several sensor geometries and types of movement were investigated.

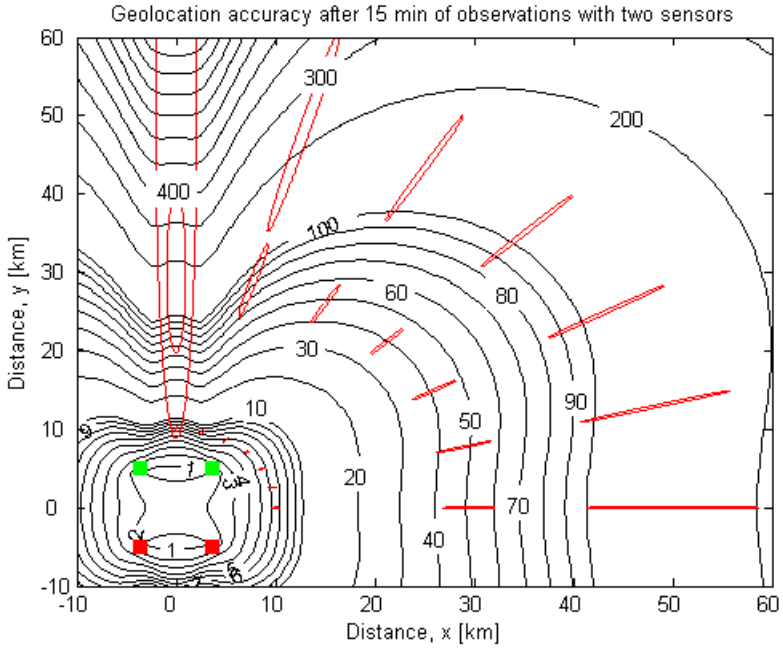


Figure 5.1 Geolocation accuracy (m) after 15 min of observations when all three methods are combined and the sensors (marked by red and green squares) move next to each other. The error ellipses (red) are enlarged 40 times.

The simulations showed that the overall best results are obtained when all three methods are combined and the two sensors move next to each other. Figure 5.1 shows an example of this. After 15 min of observations, geolocation accuracies of about 5 m at 10 km distance, 100 m at 50 km distance, and 400 m at 100 km distance from the sensors can then be obtained.

It is also possible to do geolocation with only one sensor. The AOA method is then used alone. In this case, circular sensor movement gives the overall best results with geolocation accuracies of about 100 m at 10 km distance, 3000 m at 50 km distance, and 10 km at 100 km distance after 15 min of observations.

Momentaneous geolocation can be performed if two or more sensors are used. With four sensors geolocation accuracies of about 40 m at 10 km distance and 800 m at 50 km distance from the sensors can be obtained. Figure 5.2 shows an example of this.

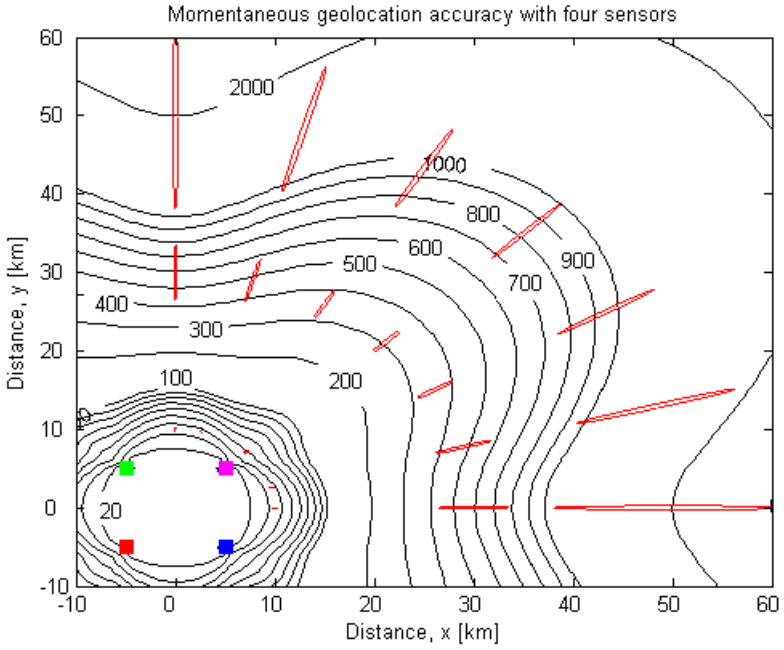


Figure 5.2 Momentaneous geolocation accuracy (m) when all three methods are combined and four sensors (marked by squares) are used. The error ellipses (red) are enlarged four times.

When choosing the trajectory for the sensors, the following general guidelines should be followed:

1) The sensors must be within range of each other.
2) The sensors must be able to observe the area of interest simultaneously.
3) The sensors should be as close as possible to the area of interest.
4) The distance between the sensors should be as long as possible.
5) The trajectory should be as long as possible.
6) The sensors should move next to each other.

Figure 5.3 shows an example of trajectory choice for a given scenario with proximity to land and a sea border that should not be crossed.

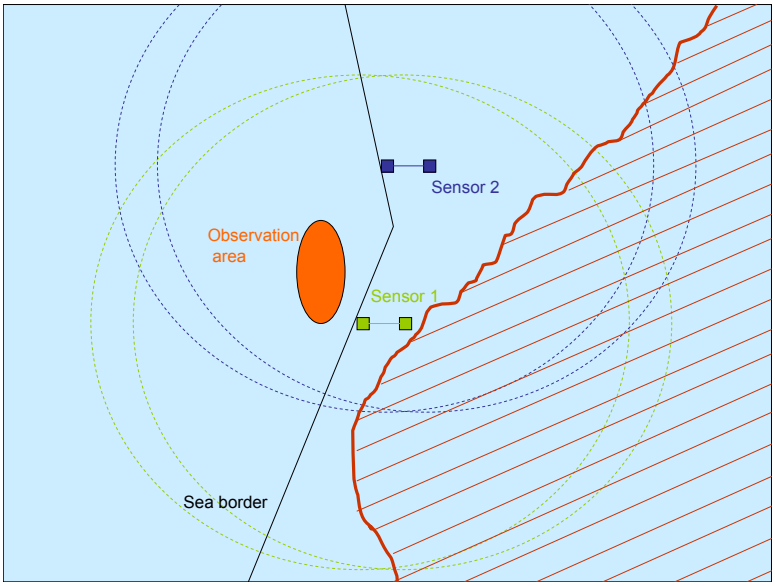


Figure 5.3 Example of trajectory choice.

The geolocation errors have been calculated as the Cramer-Rao Lower Bound (CRLB) and/or the Circular Error Probable (CEP), which gives the theoretical lower bound for the geolocation error. Good geolocation estimators should be able to achieve geolocation accuracies close to this. However, emitter movements and errors in the sensor positions will add to the geolocation error. Several other factors will also affect the geolocation accuracy, such as de-interleaving capability and data transfer capacity.

Further work should include analyses of real data in order to obtain more accurate information about the measurement errors involved and investigations of what could be realistic data transfer rates and optimum transfer strategies. A geolocation estimation algorithm should be implemented and used to perform geolocation of emitters with both simulated and real data. Finally, it should be investigated what additional equipment must be installed on the vessels in order to utilize the localization methods described in this report, including the necessary datalinks.

Appendix A Hyperbolic curves and the TDOA method

A.1 Hyperbolas

A hyperbola is one of three types of conic sections (the other two are ellipses and parabolas) [6]. The hyperbola consists of two disconnected curves called its arms or branches. It has a center (x_0, y_0) and two focal points (F_1 and F_2), see Figure A.1. The line joining the two focal points is called the semi-major (or transverse) axis. The line perpendicular to the semi-major axis and that passes through the center of the hyperbola, is called the semi-minor (or conjugate) axis. The hyperbola is symmetric about the semi-major and semi-minor axes resepectively. The two points at which the hyperbola crosses the semi-major axis are known as the vertices.

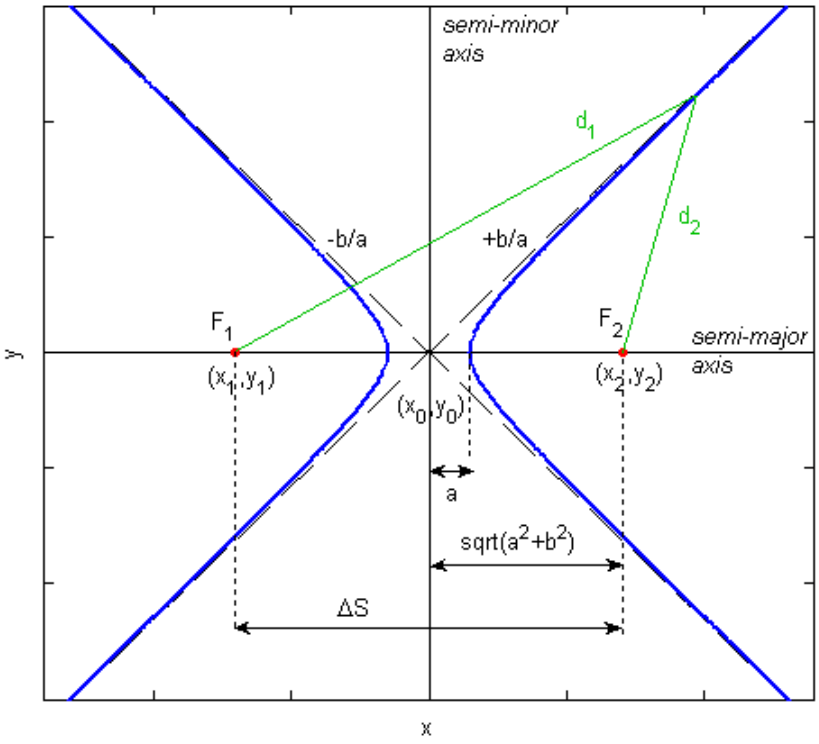


Figure A.1 Hyperbola.

The equation for a hyperbola that has its semi-major axis *parallell* to the x-axis (as shown in Figure A.1) is given by

$$\frac{(x-x_0)^2}{a^2} - \frac{(y-y_0)^2}{b^2} = 1 \qquad \text{(A.1)}$$

Here a is the distance from the center to each of the two vertices (see Figure A.1), and the parameter b is connected to the distance ΔS between the two focal points through

$$\tfrac{1}{2}\Delta S = \sqrt{a^2 + b^2} \tag{A.2}$$

The hyperbola has two asymptotes with slopes $\pm b/a$. The equation for the asymptotes is

$$y_k = y_0 \pm \tfrac{b}{a}(x - x_0) \tag{A.3}$$

An important property of the hyperbola is that the difference Δd between the distances d_1 and d_2 from each of the focal points to a point on the hyperbola is constant, i.e.,

$$\Delta d = \mid d_2 - d_1 \mid = 2a = \text{constant} \tag{A.4}$$

for all points on the hyperbolic curve.

A.2 Hyperbolas and the TDOA method (2D)

The time difference of arrival (TDOA) is given by Equation (2.1)

$$TDOA = \frac{1}{c}\left(d_1 - d_2\right) \tag{A.5}$$

where c is the speed of light and d_1 and d_2 are the distances between sensor 1 and the emitter and sensor 2 and the emitter respectively. Equation (A.5) can be rewritten on the form

$$\mid d_1 - d_2 \mid = \Delta d = c \cdot \mid TDOA \mid = \text{constant} \tag{A.6}$$

which fulfills the requirement for a hyperbola (Equation (A.4)) with the sensors located at the focal points and the possible emitter positions located along the hyperbolic curve.

Choosing a coordinate system $(x'y')$ with center at the center of the hyperbola and with the x'-axis coinciding with the semi-major axis (line through the sensors) and the y'-axis coinciding with the semi-minor axis, gives the following equation for the hyperbolic curve that represents the possible emitter positions $(x'y')$:

$$\frac{(x')^2}{a^2} - \frac{(y')^2}{b^2} = 1 \tag{A.7}$$

with the constant a given by (Equation (A.4) and Equation (A.6))

$$a = \frac{1}{2} c \mid TDOA \mid \tag{A.8}$$

and the constant b given by (from Equation (A.2))

$$b = \frac{1}{4}\Delta S^2 - a^2 = \frac{1}{2}\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 - (c \cdot TDOA)^2} \tag{A.9}$$

Here (x_1, y_1) and (x_2, y_2) are the sensor positions in the xy-coordinate system (see Figure A.2) and ΔS is the distance between the two sensors

$$\Delta S^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 \tag{A.10}$$

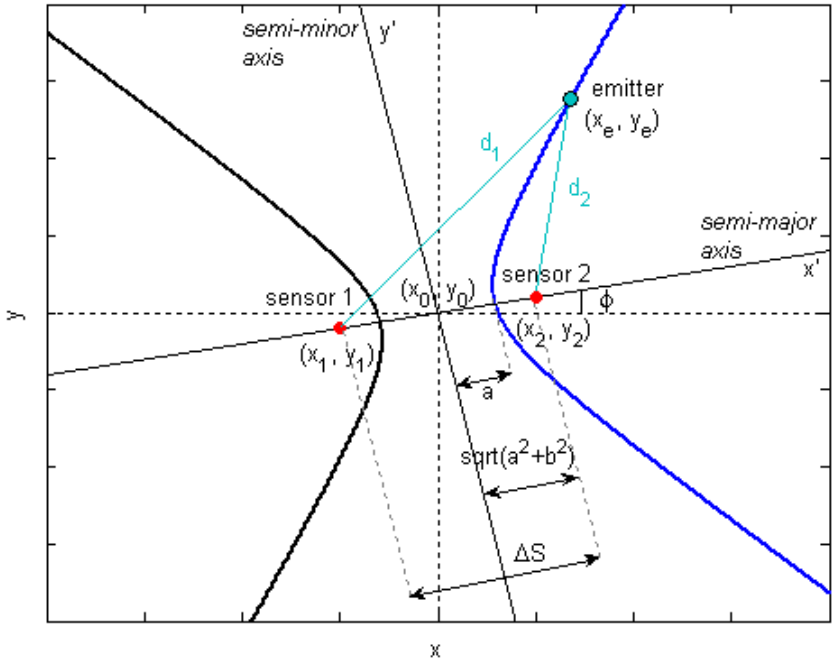


Figure A.2 Hyperbola and the TDOA-method. Red circles show the sensor positions, light blue circle shows the emitter position, and dark blue curve shows the possible emitter positions.

In order to determine the possible emitter positions in xy-coordinates, we must know the position (x_0, y_0) and rotation φ of the $x'y'$-coordinate system with respect to the xy-coordinate system (see Figure A.2). The possible emitter positions (x, y) can then be found from

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + R_\varphi^{2D} \cdot \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{A.11}$$

where R_φ^{2D} is the 2-dimensional rotation matrix

$$R_\varphi^{2D} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \tag{A.12}$$

with

$$\varphi = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \tag{A.13}$$

and (x_0, y_0) is the center of the hyperbola in xy-coordinates given by

$$\begin{aligned} x_0 &= \frac{x_1 + x_2}{2} \\ y_0 &= \frac{y_1 + y_2}{2} \end{aligned} \tag{A.14}$$

Which of the two branches of the hyperbola that corresponds to the actual possible emitter positions is determined by the sign of the measured time difference of arrival (TDOA).

A.3 Circular hyperboloids

Rotating the hyperbola described by Equation (A.1) around its semi-major axis gives a circular hyperboloid, see Figure A.3. The circular hyperboloid has the important property that the difference Δd between the distances d_1 and d_2 from each of the focal points to a point on the hyperboloid is

$$\Delta d = \mid d_2 - d_1 \mid = 2a = \text{constant} \tag{A.15}$$

for all points on the hyperboloid.

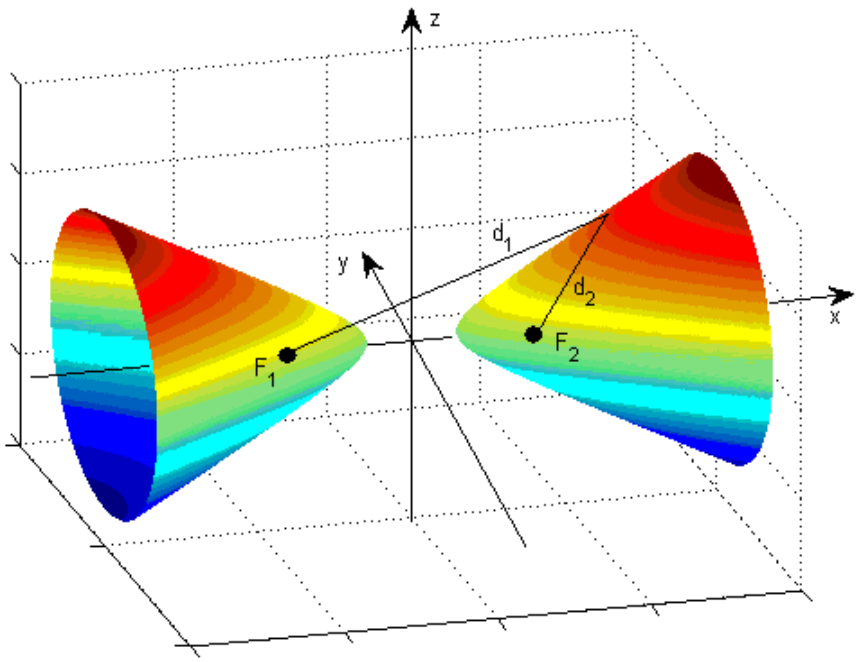


Figure A.3 Circular hyperboloid.

The equation for the circular hyperboloid can be found by noticing that the cross-section in the yz - plane is a circle with radius $(y_x - y_0)$, see Figure A.4. The equation for the circle is

$$(z - z_0)^2 + (y - y_0)^2 = (y_x - y_0)^2 \tag{A.16}$$

where (y_0, z_0) are the coordinates for the center of the circle and y_x can be found from Equation (A.1) by substituting $y = y_x$. This gives the following equation for the circular hyperboloid

$$\frac{(x - x_0)^2}{a^2} - \frac{(y - y_0)^2}{b^2} - \frac{(z - z_0)^2}{b^2} = 1 \tag{A.17}$$

where (x_0, y_0, z_0) is the center of the hyperboloid.

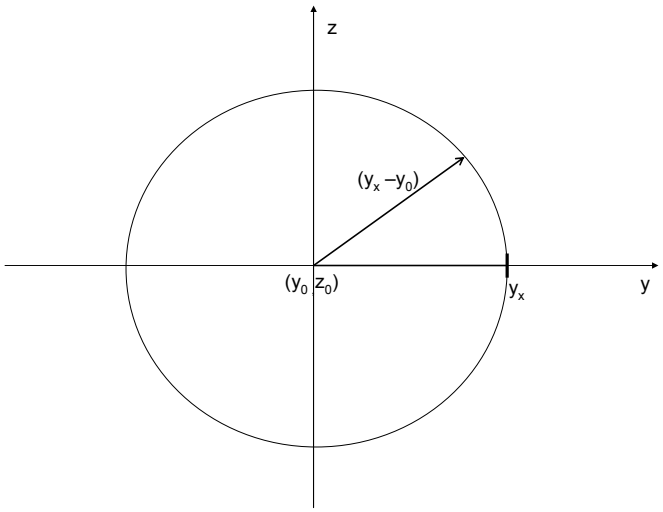


Figure A.4 Cross-section in the yz-plane of a circular hyperboloid.

A.3.1 Cross-section between the circular hyperboloid and a plane parallell to the xy-plane

The equation for the cross-section between the circular hyperboloid described by Equation (A.17) and a plane parallell to the xy-plane at $z = z_h$, can be found by substituting $z = z_h$ into Equation (A.17). This gives

$$\frac{(x - x_0)^2}{a_h^2} - \frac{(y - y_0)^2}{b_h^2} = 1 \tag{A.18}$$

where

$$a_h = a\sqrt{1+\frac{(z_h - z_0)^2}{b^2}}$$
$$b_h = b\sqrt{1+\frac{(z_h - z_0)^2}{b^2}} \qquad \text{(A.19)}$$

Equation (A.18) describes a hyperbola, i.e., the cross-section between the circular hyperboloid and a plane parallell to the xy-plane is itself a hyperbolic curve. This is illustrated in Figure A.5. The resulting hyperbola has constants a_h and b_h that are increased by a factor $\sqrt{1+(z_h - z_0)^2/b^2}$ compared to the constants a and b for the original hyperbola (at $z = z_0$) described by Equation (A.1). When $z_h = z_0$, Equation (A.18) reduces to Equation (A.1). Note that the focal points f_1^h and f_2^h for the resulting hyperbola do *not* coincide with the projected (onto the same plane) focal points F_1 and F_2 for the hyperboloid. Figure A.6 illustrates the resulting hyperbola for different $\Delta z = z_h - z_0$.

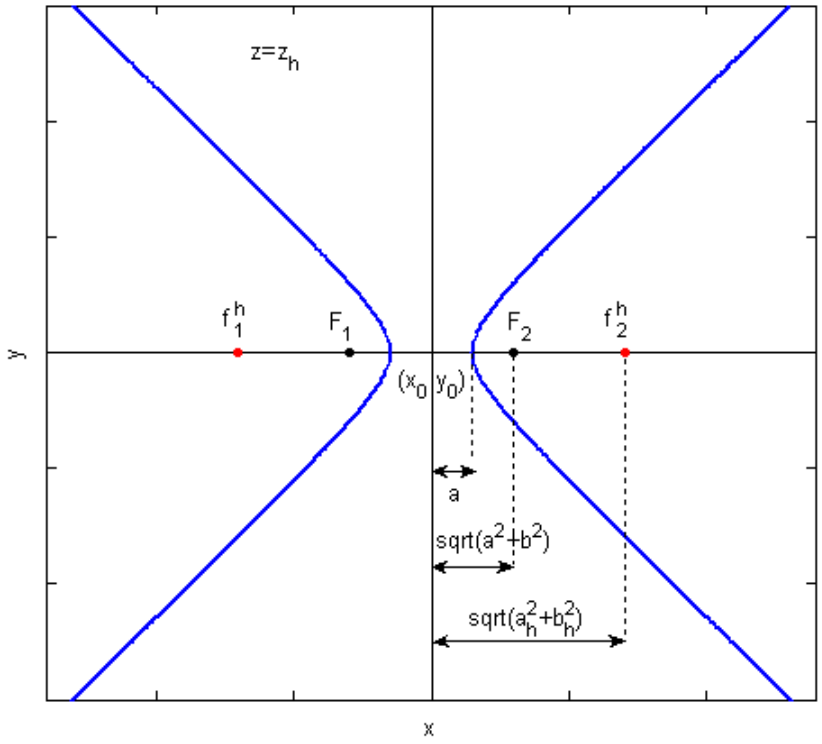


Figure A.5 Cross-section between a circular hyperboloid and a plane parallell to the xy-plane. The cross-section (blue curve) is a hyperbola with foci f_1^h and f_2^h that differ from the projected foci F_1 and F_2 of the circular hyperboloid.

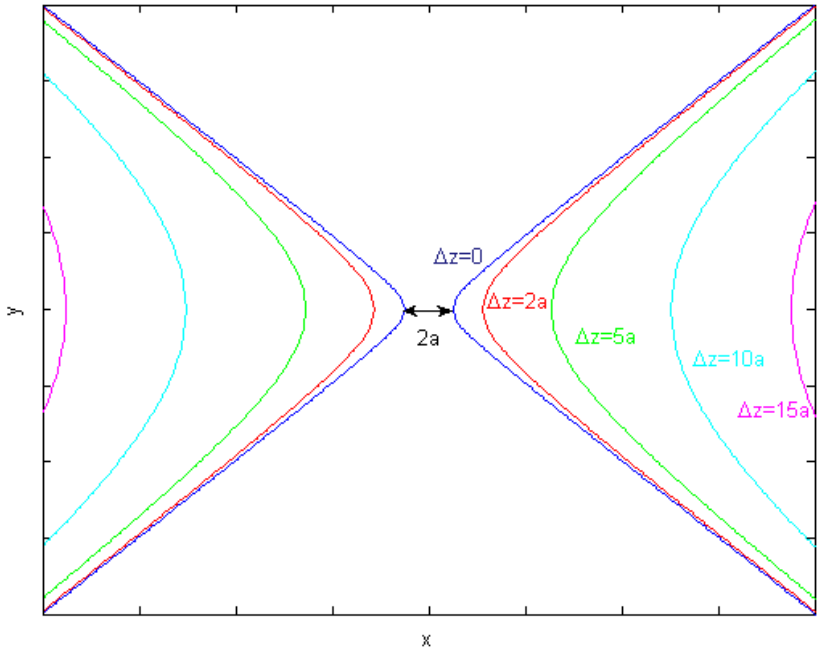


Figure A.6 Cross-sections (hyperbolas) between a circular hyperboloid and planes parallell to (but at different heights above) the xy-plane. Here 2a is the distance between the vertices. The parameter b in Equation (A.19) was set equal to the parameter a when the curves were calculated. Units are equal along both axes.

A.4 Circular hyperboloids and the TDOA method (3D)

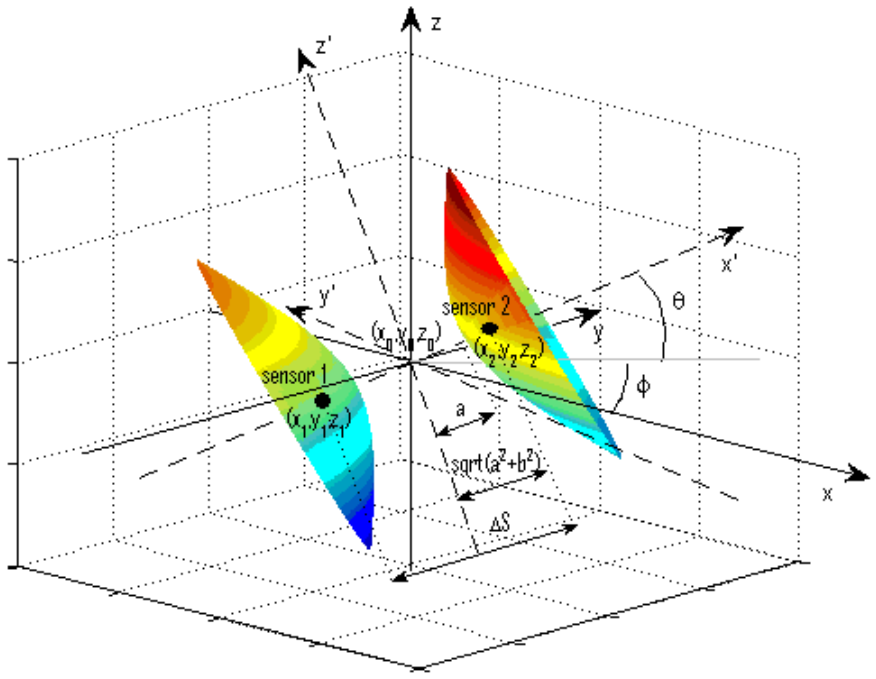


Figure A.7 Circular hyperboloids and the TDOA method. Black circles show the sensor positions. Possible emitter positions are located on one of the two hyperboloids.

When the TDOA method is used in three dimensions, the possible emitter positions describe a circular hyperboloid as described above in Section A.3. The equation for the circular hyperboloid is

$$\frac{(x')^2}{a^2} - \frac{(y')^2}{b^2} - \frac{(z')^2}{b^2} = 1 \tag{A.20}$$

in the $x'y'z'$-coordinate system, see Figure A.7. The center of the hyperboloid is at the center of the $x'y'z'$-coordinate system. The constant a is as before given by

$$a = \frac{1}{2} c \cdot |TDOA| \tag{A.21}$$

and the constant b is given by

$$b = \sqrt{\tfrac{1}{4}\Delta S^2 - a^2} = \tfrac{1}{2}\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 - (c \cdot TDOA)^2} \tag{A.22}$$

Here (x_1, y_1, z_1) and (x_2, y_2, z_2) are the sensor positions in the xyz-coordinate system, and ΔS is the distance between the two sensors in three dimensions

$$\Delta S^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 \tag{A.23}$$

In order to determine the possible emitter positions in xyz-coordinates, we must know the position (x_0, y_0, z_0) and rotation (φ, θ) of the $x'y'z'$-coordinate system with respect to the xyz-coordinate system (see Figure A.7). The possible emitter positions (x, y, z) can then be found from

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + R^{3D}_{y',\theta} \cdot R^{3D}_{z',\varphi} \cdot \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{A.24}$$

where $R^{3D}_{z,\varphi}$ is the 3-dimensional rotation matrix for rotation an angle φ around the z-axis

$$R^{3D}_{z,\varphi} = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.25}$$

with

$$\varphi = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \tag{A.26}$$

and $R_{y,\theta}^{3D}$ is the 3-dimensional rotation matrix for rotation an angle θ around the new y-axis

$$R_{y,\theta}^{3D} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \tag{A.27}$$

with

$$\theta = \arctan\left(\frac{z_2 - z_1}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \right) \tag{A.28}$$

The center (x_0, y_0, z_0) of the hyperboloid in xyz-coordinates is found from

$$\begin{aligned} x_0 &= \frac{x_1 + x_2}{2} \\ y_0 &= \frac{y_1 + y_2}{2} \\ z_0 &= \frac{z_1 + z_2}{2} \end{aligned} \tag{A.29}$$

Which of the two branches of the hyperboloid that corresponds to the actual possible emitter positions is determined by the sign of the measured time difference of arrival (TDOA).

A.4.1 Cross-section between the circular hyperboloid and the emitter plane

If the possible emitter positions are confined to a plane and both sensors are at the same altitude above this plane, the situation is equivalent to the situation described in Section A.3.1. The cross-section between the circular hyperboloid (describing the possible emitter positions in three dimensions) and the emitter plane will describe a hyperbolic curve and can be found from Equations (A.18)-(A.19) (which describe the cross-section in the coordinate system of the hyperboloid) and Equations (A.11)-(A.13) (which rotate the coordinate system in the *xy*-plane if the semi-major axis of the hyperboloid is not parallell to the x-axis).

Appendix B Circles and the Scanphase method

The circle is a special case of the ellipse, which is one of the three types of conic sections (the other two are hyperbolas and parabolas) [6]. Figure B.1 shows a circle with inscribed angles (marked in green). Inscribed angles are always excactly half the size of the central angle (marked in blue) that subtend the same arc (marked in red). All inscribed angles that subtend the same arc are therefore equal. Figure B.1 shows three inscribed angles that are all equal to θ (the corresponding central angle is 2θ). Corresponding inscribed angles (marked in yellow) that are inscribed onto (as opposed to subtend) the arc are equal to $\pi - \theta$. Inscribed angles that *subtend* the arc are always $\theta < \pi/2$, while inscribed angles that are *inscribed onto* the arc are always $\pi - \theta > \pi/2$.

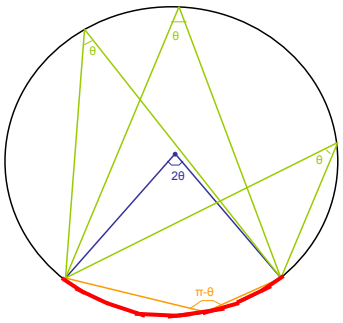


Figure B.1 Inscribed angles. The central angle is shown in blue. Examples of inscribed angles that subtend the arc (red part of the circle) are shown in green, while an example of an inscribed angle that is inscribed onto the arc is shown in yellow.

The scanphase method (see Section 2.1.2) uses the measured aperture angle υ to locate the emitter. The possible emitter positions must therefore be located on two circles with the sensors at the cross-section between the circles, see Figure B.2. The aperture angle plays the role of the inscribed angle and the emitter is found on one of the large arcs when the aperture angle is $\upsilon < \pi/2$ and on one of the small arcs when the aperture angle is $\upsilon > \pi/2$.

The equation for each of the two circles is

$$\frac{(x-x_c)^2}{R^2} + \frac{(y-y_c)^2}{R^2} = 1 \tag{B.1}$$

where R is the radius given by (see Figure B.2)

$$R = \frac{d_{12}}{2\sin\upsilon} \tag{B.2}$$

and d_{12} is the distance between the two sensors

$$d_{12} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{B.3}$$

The center coordinates (x_c, y_c) for each of the two circles can be found from

$$\begin{aligned} x_c &= x_0 \pm a\sin\varphi \\ y_c &= y_0 \pm a\cos\varphi \end{aligned} \tag{B.4}$$

where the minus sign is used to find the center (x_{c1}, y_{c1}) for the first circle and the plus sign is used to find the center (x_{c2}, y_{c2}) for the second circle, see Figure B.2. Here a is given by

$$a = \frac{d_{12}}{2\tan\upsilon} \tag{B.5}$$

and

$$\varphi = \arctan\left(\frac{-(y_2 - y_0)}{x_2 - x_0}\right) = \arctan\left(\frac{-(y_1 - y_0)}{x_1 - x_0}\right) \tag{B.6}$$

with

$$\begin{aligned} x_0 &= \frac{x_1 + x_2}{2} \\ y_0 &= \frac{y_1 + y_2}{2} \end{aligned} \tag{B.7}$$

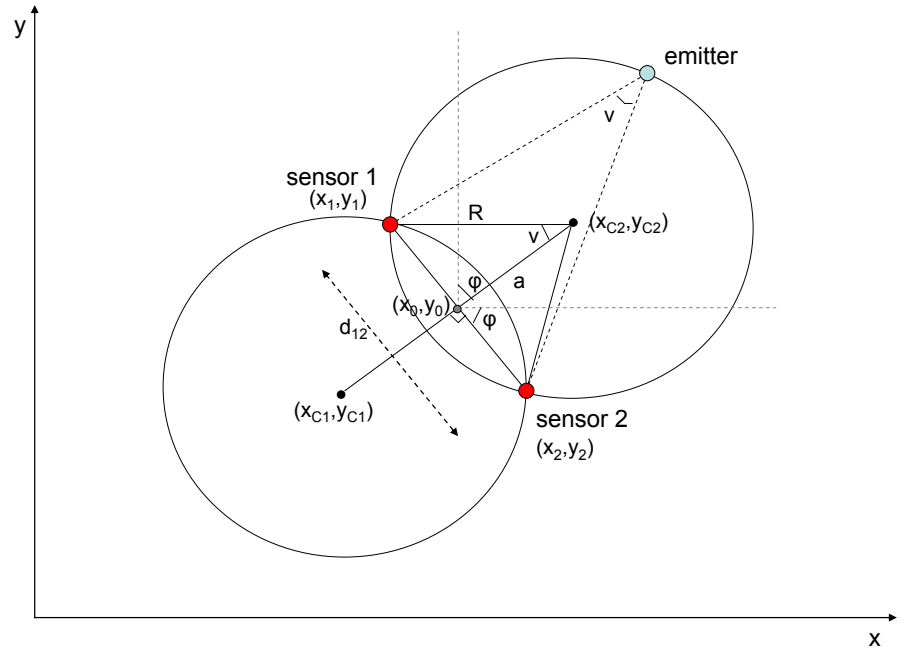


Figure B.2 Circles and the scanphase method. The red filled circles show the sensor positions, the blue filled circle shows the emitter position, and the big black circles show the possible emitter positions.

Appendix C Partial derivatives

C.1 Time difference of arrival

The time difference of arrival TDOA is given by Equation (2.1). Its partial derivatives with respect to the emitter coordinates x_e and y_e are given by

$$\begin{aligned}\frac{\partial(TDOA)}{\partial x_e} &= \frac{1}{c}\cdot\left[\frac{(x_2-x_e)}{d_2}-\frac{(x_1-x_e)}{d_1}\right]\\ \frac{\partial(TDOA)}{\partial y_e} &= \frac{1}{c}\cdot\left[\frac{(y_2-y_e)}{d_2}-\frac{(y_1-y_e)}{d_1}\right]\end{aligned} \tag{C.1}$$

where c is the speed of light, (x_1, y_1) and (x_2, y_2) are the sensor coordinates and d_1 and d_2 are given by Equation (2.2).

C.2 Aperture angle

The aperture angle υ is given by Equation (2.3). Its partial derivatives with respect to the emitter coordinates x_e and y_e are given by[5]

$$\begin{aligned}\frac{\partial\upsilon}{\partial x_e} &= -\frac{1}{\sqrt{1-u^2}}\cdot\frac{\partial u}{\partial x_e}\\ \frac{\partial\upsilon}{\partial y_e} &= -\frac{1}{\sqrt{1-u^2}}\cdot\frac{\partial u}{\partial y_e}\end{aligned} \tag{C.2}$$

where

$$\begin{aligned}u &= \frac{d_{12}^2-d_1^2-d_2^2}{-2d_1d_2}\\ \frac{\partial u}{\partial x_e} &= -\frac{u}{a}\cdot\frac{\partial a}{\partial x_e}-\frac{2}{a}\cdot\left[d_1\frac{\partial d_1}{\partial x_e}+d_2\frac{\partial d_2}{\partial x_e}\right]\\ \frac{\partial u}{\partial y_e} &= -\frac{u}{a}\cdot\frac{\partial a}{\partial y_e}-\frac{2}{a}\cdot\left[d_1\frac{\partial d_1}{\partial y_e}+d_2\frac{\partial d_2}{\partial y_e}\right]\end{aligned} \tag{C.3}$$

[5] We have used that $d/du(\arccos u) = -1/\sqrt{1-u^2}$ [7].

Here

$$
\begin{aligned}
a &= -2d_1 d_2 \\
\frac{\partial a}{\partial x_e} &= -2 \cdot \left[d_2 \frac{\partial d_1}{\partial x_e} + d_1 \frac{\partial d_2}{\partial x_e} \right] \\
\frac{\partial a}{\partial y_e} &= -2 \cdot \left[d_2 \frac{\partial d_1}{\partial y_e} + d_1 \frac{\partial d_2}{\partial y_e} \right]
\end{aligned}
\tag{C.4}
$$

with d_1, d_2, and d_{12} given by Equation (2.4) and

$$
\begin{aligned}
\frac{\partial d_1}{\partial x_e} &= -\frac{x_1 - x_e}{d_1} \\
\frac{\partial d_2}{\partial x_e} &= -\frac{x_2 - x_e}{d_2} \\
\frac{\partial d_1}{\partial y_e} &= -\frac{y_1 - y_e}{d_1} \\
\frac{\partial d_2}{\partial y_e} &= -\frac{y_2 - y_e}{d_2}
\end{aligned}
\tag{C.5}
$$

where (x_1, y_1) and (x_2, y_2) are the sensor coordinates.

C.3 Angle of arrival

The angle of arrival θ is given by Equation (2.6). Its partial derivatives with respect to the emitter coordinates x_e and y_e are given by[6]

$$
\begin{aligned}
\frac{\partial \theta}{\partial x_e} &= \frac{y_e - y_1}{(x_e - x_1)^2 + (y_e - y_1)^2} \\
\frac{\partial \theta}{\partial y_e} &= -\frac{x_e - x_1}{(x_e - x_1)^2 + (y_e - y_1)^2}
\end{aligned}
\tag{C.6}
$$

where (x_1, y_1) are the sensor coordinates.

[6] We have used that $d/du(\arctan u) = 1/(1+u^2)$ [7].

Appendix D The Cramer-Rao lower bound (CRLB) in 1D

We will in this appendix try to explain in an intuitive way how to determine the CRLB in 1 dimension. The goal is to understand the general principles behind the CRLB by looking at how the CRLB is obtained in the 1-dimensional case.

The process of determining the CRLB can be split into two steps:

1. Determine the geolocation error Δx_n that corresponds to the variance σ_n^2 for the n'th measurement.
2. Determine the total geolocation error ΔX that results from combining all N measurements.

The geolocation error Δx_n that corresponds to the variance σ_n^2 for the n'th measurement is in the 1-dimensional case given by

$$\Delta x_n = \frac{\sigma_n}{\partial S_n / \partial x_e} \tag{D.1}$$

where $\partial S_n / \partial x_e$ is the partial derivative of the n'th measured parameter S_n at the emitter position x_e.

The resulting total geolocation error ΔX from N measurements $S_1,...,S_N$ with corresponding variances $\sigma_1^2,...,\sigma_N^2$ is then given by[7]

$$\Delta X = \sqrt{\frac{1}{\sum_{n=1}^{N}\left(\frac{1}{\Delta x_n}\right)^2}} = \sqrt{\frac{1}{\sum_{n=1}^{N}\left(\frac{\partial S_n/\partial x_e}{\sigma_n}\right)^2}} \tag{D.2}$$

Equation (D.2) can be written on the general form

$$(\Delta X)^2 = \left[\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^{\mathbf{T}}\right]^{-1} = \mathbf{K} = \omega^2 \tag{D.3}$$

where $\mathbf{A}$ is the $N \times N$ variance matrix

[7] In the case when the geolocation error is the same for all N measurements, i.e., when $\Delta x_n = \Delta x$ for all n=1,..., N, Equation (D.2) gives the familiar result $\Delta X = \Delta x/\sqrt{N}$ (the resulting total error from N measurements is reduced by a factor $\sqrt{N}$ compared to the error from a single measurement).

$$\mathbf{A} = \begin{pmatrix} \sigma_1^2 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & & \sigma_n^2 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \sigma_N^2 \end{pmatrix} \tag{D.4}$$

and $\mathbf{B}$ is the $1 \times N$ partial derivatives matrix

$$\mathbf{B} = \left[\frac{\partial S_1}{\partial x_e} \cdots \frac{\partial S_n}{\partial x_e} \cdots \frac{\partial S_N}{\partial x_e} \right] \tag{D.5}$$

The expression $\left[\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^{\mathbf{T}} \right]^{-1}$ in Equation (D.3) is called the CRLB covariance matrix[8], usually denoted $\mathbf{K}$, i.e., and is valid also for higher dimensions. In the 1-dimensional case $\mathbf{K}$ becomes a number (1×1 matrix) which we may denote ω^2. This gives for the CRLB error line

$$\Delta X = \omega \tag{D.6}$$

where the value of ω can be calculated when the variance matrix $\mathbf{A}$ and the partial derivatives matrix $\mathbf{B}$ are known. In fact, ω^2 is the eigenvalue of the eigenvalue problem

$$\mathbf{K} \cdot \vec{X} = \omega^2 \vec{X} \tag{D.7}$$

that must be solved in order to find the size and orientation of the CRLB error line/ellipse/ ellipsoid. In the 1-dimensional case the CRLB is an error line with length ω, in the 2-dimensional case an error ellipse with semi-axes ω_1 and ω_2, and in the 3-dimensional case an error ellipsoid with semi-axes ω_1, ω_2, and ω_3. The orientation of the error line/ellipse/ellipsoid is in each case determined by the corresponding eigenvectors. Figure 2.22 in Section 2.3.1 shows an example of an error ellipse (2D).

[8] For a rigorous mathematical derviation of the CRLB covariance matrix, see Appendix A in [5].

Appendix E Software

The software for the simulations was written in Matlab v7.7. The code was based on software for geolocation accuracy simulations of the TDOA and scanphase methods written by E O Løvli [1]. The software has been modified to increase speed and clearity of the code, and routines for geolocation accuracy simulations of the AOA method (and the AOA method in combination with the TDOA method and/or the scanphase method) have been added.

In this report we have studied the case when the sensors and the emitter are in the same plane. However, the software is valid also when this is not the case, i.e., when the sensors are at some altitude above the emitter.

The software is grouped into three folders:

1) CEP
2) CRLB
3) EmitterPosition

The CEP-folder contains the routines that are used for calculating the CEP-radius and plotting the CEP-contours, see pages 116-130. The CRLB-folder contains the routines that are used for calculating the CRLB covariance matrix and plotting CRLB error ellipses, see pages 131-153. Since the CEP-radius is calculated from the CRLB error ellipse, these routines are also used by some of the routines in the CEP-folder. The EmitterPosition-folder contains routines for calculating and plotting possible emitter positions for given time difference of arrival, aperture angle or angle of arrival measurements, see pages 154-177.

```
function [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_TDOA(sensor1, sensor2, ...
                                                         stddev, limits, nPoints);
% Plot CEP radius contours
%
% [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_TDOA(sensor1, sensor2, stddev, ...
%                                                              limits, nPoints)
%
% Output parameters: X            - Array of x-coordinates [m]
%                    Y            - Array of y-coordinates [m]
%                    Cep2D_radius - Array of CEP radii [m]
%                                   for 2D (xy) localization
%                    omega1       - Minor semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%                    omega2       - Major semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%
% Input parameters: sensor1  - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2  - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev   - Standard deviation [ns] for the TDOA
%                              measurements
%                   limits   - Vector giving minimum and maximum values for
%                              the x- and y-coordinates [m]: [x_min x_max y_min y_max]
%                   nPoints  - Number of points in the grid along one
%                              dimension (x or y, the same value is used
%                              for both dimensions)
%
% Output files: TDOA_Cep2D
%
% Input files: none
%
% External functions: ComputeCep2D
%                     ComputeCrlb2D_TDOA
%                     ComputeCrlb2DMinMajAxes
%
% Last modified: 18.09.09
%
% ------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ------------------------------------

path('..\CRLB\', path);

x_min=limits(1);
x_max=limits(2);
y_min=limits(3);
y_max=limits(4);

% Calculate the variance for the TDOA measurements [s^2]
variance=(stddev.*1e-9).^2;

% Calculate CEP radius in grid
[X,Y]=meshgrid(linspace(x_min,x_max,nPoints),linspace(y_min,y_max,nPoints));
Cep2D_radius=zeros(size(X));
[i_max,j_max]=size(X);
```

```
function [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_ScanPhase(sensor1, sensor2,...
                                                        stddev, limits, nPoints);
% Plot CEP radius contours
%
% [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_ScanPhase(sensor1, sensor2, stddev, ...
%                                                             limits, nPoints)
%
% Output parameters: X            - Array of x-coordinates [m]
%                    Y            - Array of y-coordinates [m]
%                    Cep2D_radius - Array of CEP radii [m]
%                                   for 2D (xy) localization
%                    omega1       - Minor semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%                    omega2       - Major semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%
% Input parameters: sensor1  - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2  - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev   - Standard deviation [deg] for the Aperture
%                              Angle measurements
%                   limits   - Vector giving minimum and maximum values for
%                              the x- and y-coordinates [m]: [x_min x_max y_min y_max]
%                   nPoints  - Number of points in the grid along one
%                              dimension (x or y, the same value is used
%                              for both dimensions)
%
% Output files: ScanPhase_Cep2D
%
% Input files: none
%
% External functions: ComputeCep2D
%                     ComputeCrlb2D_ScanPhase
%                     ComputeCrlb2DMinMajAxes
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

path('..\CRLB\', path);

x_min=limits(1);
x_max=limits(2);
y_min=limits(3);
y_max=limits(4);

% Calculate the variance for the Aperture Angle measurements [rad^2]
variance=(stddev.*pi./180).^2;

% Calculate CEP radius in grid
[X,Y]=meshgrid(linspace(x_min,x_max,nPoints),linspace(y_min,y_max,nPoints));
Cep2D_radius=zeros(size(X));
[i_max,j_max]=size(X);
```

```
for i=1:1:i_max
    for j=1:1:j_max
        emitter=[X(i,j) Y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_ScanPhase(sensor1,sensor2,emitter,variance);
        [omega1(i,j), omega2(i,j)] = ComputeCrlb2DMinMajAxes(Crlb2D_CovMatrix);
        [Cep2D_radius(i,j)] = ComputeCep2D(omega1(i,j), omega2(i,j));
    end
end

% Plot CEP radius contours
figure;
m0=1:1:9;
m1=10:10:90;
m2=100:100:900;
m3=1e3:1e3:9e3;
m4=1e4:1e4:9e4;
m5=1e5:1e5:9e5;
m6=1e6:1e6:9e6;
m7=1e7:1e7:9e7;
m8=1e8:1e8:9e8;
m9=1e9:1e9:1e10;
v1=[m0 m1 m2 m3 m4 m5 m6 m7 m8 m9];
[C,h]=contour(X,Y,Cep2D_radius,v1,'k');
v2=[10 100 1000 1e4 1e5 1e6];
clabel(C,h,v2,'Color','k');
xlabel('Distance, x [km]');
ylabel('Distance, y [km]');
title('ScanPhase');

% Values on figure axes when x_min=y_min=-100km and x_max=y_max=100km
set(gca, 'XTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'YTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'XTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);
set(gca, 'YTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);

% Save results to file
save outputfiles\Cep2D_ScanPhase X Y Cep2D_radius omega1 omega2
```

```
for i=1:1:i_max
    for j=1:1:j_max
        emitter=[X(i,j) Y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOA(sensor1, sensor2, emitter, variance);
        [omega1(i,j), omega2(i,j)] = ComputeCrlb2DMinMajAxes(Crlb2D_CovMatrix);
        [Cep2D_radius(i,j)] = ComputeCep2D(omega1(i,j), omega2(i,j));
    end
end

% Plot CEP radius contours
figure;
m0=1:1:9;
m1=10:10:90;
m2=100:100:900;
m3=1e3:1e3:9e3;
m4=1e4:1e4:9e4;
m5=1e5:1e5:9e5;
m6=1e6:1e6:9e6;
m7=1e7:1e7:9e7;
m8=1e8:1e8:9e8;
m9=1e9:1e9:1e10;
v1=[m0 m1 m2 m3 m4 m5 m6 m7 m8 m9];
[C,h]=contour(X,Y,Cep2D_radius,v1,'k');
v2=[10 100 1000 1e4 1e5 1e6];
clabel(C,h,v2,'Color','k');
xlabel('Distance, x [km]');
ylabel('Distance, y [km]');
title('TDOA');

% Values on figure axes when x_min=y_min=-100km and x_max=y_max=100km
set(gca, 'XTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'YTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'XTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);
set(gca, 'YTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);

% Save results to file
save outputfiles\Cep2D_TDOA X Y Cep2D_radius omega1 omega2
```

```
function [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_AOA(sensor,stddev,limits,nPoints);
% Plot CEP radius contours
%
% [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_AOA(sensor,stddev,limits,nPoints)
%
% Output parameters: X            - Array of x-coordinates [m]
%                    Y            - Array of y-coordinates [m]
%                    Cep2D_radius - Array of CEP radii [m]
%                                   for 2D (xy) localization
%                    omega1       - Minor semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%                    omega2       - Major semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%
% Input parameters: sensor   - nx3 matrix specifying n sensor xyz-position(s) [m]
%                   stddev   - Standard deviation [deg] for the AOA
%                              measurements
%                   limits   - Vector giving minimum and maximum values for
%                              the x- and y-coordinates [m]: [x_min x_max y_min y_max]
%                   nPoints  - Number of points in the grid along one
%                              dimension (x or y, the same value is used
%                              for both dimensions)
%
% Output files: AOA_Cep2D
%
% Input files: none
%
% External functions: ComputeCep2D
%                     ComputeCrlb2D_AOA
%                     ComputeCrlb2DMinMajAxes
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

path('..\CRLB\', path);

x_min=limits(1);
x_max=limits(2);
y_min=limits(3);
y_max=limits(4);

% Calculate the variance for the AOA measurements [rad^2]
variance=(stddev.*pi./180).^2;

% Calculate CEP radius in grid
[X,Y]=meshgrid(linspace(x_min,x_max,nPoints),linspace(y_min,y_max,nPoints));
Cep2D_radius=zeros(size(X));
[i_max,j_max]=size(X);
for i=1:1:i_max
    for j=1:1:j_max
        emitter=[X(i,j) Y(i,j) 0];
```

```
        [Crlb2D_CovMatrix] = ComputeCrlb2D_AOA(sensor, emitter, variance);
        [omega1(i,j), omega2(i,j)] = ComputeCrlb2DMinMajAxes(Crlb2D_CovMatrix);
        [Cep2D_radius(i,j)] = ComputeCep2D(omega1(i,j), omega2(i,j));
    end
end

% Plot CEP radius contours
figure;
m0=1:1:9;
m1=10:10:90;
m2=100:100:900;
m3=1e3:1e3:9e3;
m4=1e4:1e4:9e4;
m5=1e5:1e5:9e5;
m6=1e6:1e6:9e6;
m7=1e7:1e7:9e7;
m8=1e8:1e8:9e8;
m9=1e9:1e9:1e10;
v1=[m0 m1 m2 m3 m4 m5 m6 m7 m8 m9];
[C,h]=contour(X,Y,Cep2D_radius,v1,'k');
v2=[10 100 1000 1e4 1e5 1e6];
clabel(C,h,v2,'Color','k');
xlabel('Distance, x [km]');
ylabel('Distance, y [km]');
title('AOA');

% Values on figure axes when x_min=y_min=-100km and x_max=y_max=100km
set(gca, 'XTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'YTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'XTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);
set(gca, 'YTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);

% Save results to file
save outputfiles\Cep2D_AOA X Y Cep2D_radius omega1 omega2
```

```
function [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_ScanPhaseAOA(sensor1, sensor2, ...
                                                 stddev_AA, stddev_AOA, limits, nPoints);
% Plot CEP radius contours
%
% [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_ScanPhaseAOA(sensor1, sensor2, ...
%                                             stddev_AA, stddev_AOA, limits, nPoints)
%
% Output parameters: X            - Array of x-coordinates [m]
%                    Y            - Array of y-coordinates [m]
%                    Cep2D_radius - Array of CEP radii [m]
%                                   for 2D (xy) localization
%                    omega1       - Minor semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%                    omega2       - Major semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%
% Input parameters: sensor1     - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2     - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev_AA   - Standard deviation [deg] for the Aperture
%                                 Angle measurements
%                   stddev_AOA  - Standard deviation [deg] for the AOA
%                                 measurements
%                   limits      - Vector giving minimum and maximum values for the
%                                 x- and y-coordinates [m]: [x_min x_max y_min y_max]
%                   nPoints     - Number of points in the grid along one
%                                 dimension (x or y, the same value is used
%                                 for both dimensions)
%
% Output files: ScanPhaseAOA_Cep2D
%
% Input files: none
%
% External functions: ComputeCep2D
%                     ComputeCrlb2D_ScanPhaseAOA
%                     ComputeCrlb2DMinMajAxes
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

path('..\CRLB\', path);

x_min=limits(1);
x_max=limits(2);
y_min=limits(3);
y_max=limits(4);

% Calculate the variance for the Aperture Angle measurements [rad^2]
var_AA=(stddev_AA.*pi./180).^2;

% Calculate the variance for the AOA measurements [rad^2]
var_AOA=(stddev_AOA.*pi./180).^2;
```

```
% Calculate CEP radius in grid
[X,Y]=meshgrid(linspace(x_min,x_max,nPoints),linspace(y_min,y_max,nPoints));
Cep2D_radius=zeros(size(X));
[i_max,j_max]=size(X);
for i=1:1:i_max
    for j=1:1:j_max
        emitter=[X(i,j) Y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_ScanPhaseAOA(sensor1, sensor2, ...
                                                    emitter, var_AA, var_AOA);
        [omega1(i,j), omega2(i,j)] = ComputeCrlb2DMinMajAxes(Crlb2D_CovMatrix);
        [Cep2D_radius(i,j)] = ComputeCep2D(omega1(i,j), omega2(i,j));
    end
end

% Plot CEP radius contours
figure;
m0=1:1:9;
m1=10:10:90;
m2=100:100:900;
m3=1e3:1e3:9e3;
m4=1e4:1e4:9e4;
m5=1e5:1e5:9e5;
m6=1e6:1e6:9e6;
m7=1e7:1e7:9e7;
m8=1e8:1e8:9e8;
m9=1e9:1e9:1e10;
v1=[m0 m1 m2 m3 m4 m5 m6 m7 m8 m9];
[C,h]=contour(X,Y,Cep2D_radius,v1,'k');
v2=[10 100 1000 1e4 1e5 1e6];
clabel(C,h,v2,'Color','k');
xlabel('Distance, x [km]');
ylabel('Distance, y [km]');
title('ScanPhaseAOA');

% Values on figure axes when x_min=y_min=-100km and x_max=y_max=100km
set(gca, 'XTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'YTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'XTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);
set(gca, 'YTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);

% Save results to file
save outputfiles\Cep2D_ScanPhaseAOA X Y Cep2D_radius omega1 omega2
```

```
function [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_TDOAScanPhase(sensor1, sensor2, ...
                                                  stddev_TDOA, stddev_AA, limits, nPoints);
% Plot CEP radius contours
%
% [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_TDOAScanPhase(sensor1, sensor2, ...
%                                             stddev_TDOA, stddev_AA, limits, nPoints)
%
% Output parameters: X            - Array of x-coordinates [m]
%                    Y            - Array of y-coordinates [m]
%                    Cep2D_radius - Array of CEP radii [m]
%                                   for 2D (xy) localization
%                    omega1       - Minor semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%                    omega2       - Major semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%
% Input parameters: sensor1     - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2     - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev_TDOA - Standard deviation [ns] for the TDOA measurements
%                   stddev_AA   - Standard deviation [deg] for the Aperture Angle
%                                 measurements
%                   limits      - Vector giving minimum and maximum values for the
%                                 x- and y-coordinates [m]: [x_min x_max y_min y_max]
%                   nPoints     - Number of points in the grid along one
%                                 dimension (x or y, the same value is used
%                                 for both dimensions)
%
% Output files: TDOAScanPhase_Cep2D
%
% Input files: none
%
% External functions: ComputeCep2D
%                     ComputeCrlb2D_TDOAScanphase
%                     ComputeCrlb2DMinMajAxes
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

path('..\CRLB\', path);

x_min=limits(1);
x_max=limits(2);
y_min=limits(3);
y_max=limits(4);

% Calculate the variance for the TDOA measurements [s^2]
var_TDOA=(stddev_TDOA.*1e-9).^2;

% Calculate the variance for the Aperture Angle measurements [rad^2]
var_AA=(stddev_AA.*pi./180).^2;
```

```
% Calculate CEP radius in grid
[X,Y]=meshgrid(linspace(x_min,x_max,nPoints),linspace(y_min,y_max,nPoints));
Cep2D_radius=zeros(size(X));
[i_max,j_max]=size(X);
for i=1:1:i_max
    for j=1:1:j_max
        emitter=[X(i,j) Y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAScanPhase(sensor1, sensor2, ...
                                                         emitter, var_TDOA, var_AA);
        [omega1(i,j), omega2(i,j)] = ComputeCrlb2DMinMajAxes(Crlb2D_CovMatrix);
        [Cep2D_radius(i,j)] = ComputeCep2D(omega1(i,j), omega2(i,j));
    end
end

% Plot CEP radius contours
figure;
m0=1:1:9;
m1=10:10:90;
m2=100:100:900;
m3=1e3:1e3:9e3;
m4=1e4:1e4:9e4;
m5=1e5:1e5:9e5;
m6=1e6:1e6:9e6;
m7=1e7:1e7:9e7;
m8=1e8:1e8:9e8;
m9=1e9:1e9:1e10;
v1=[m0 m1 m2 m3 m4 m5 m6 m7 m8 m9];
[C,h]=contour(X,Y,Cep2D_radius,v1,'k');
v2=[10 100 1000 1e4 1e5 1e6];
clabel(C,h,v2,'Color','k');
xlabel('Distance, x [km]');
ylabel('Distance, y [km]');
title('TDOAScanPhase');

% Values on figure axes when x_min=y_min=-100km and x_max=y_max=100km
set(gca, 'XTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'YTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'XTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);
set(gca, 'YTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);

% Save results to file
save outputfiles\Cep2D_TDOAScanPhase X Y Cep2D_radius omega1 omega2
```

```
function [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_TDOAAOA(sensor1, sensor2, ...
                                          stddev_TDOA, stddev_AOA, limits, nPoints);
% Plot CEP radius contours
%
% [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_TDOAAOA(sensor1, sensor2, stddev_TDOA, ...
%                                                    stddev_AOA, limits, nPoints)
%
% Output parameters: X            - Array of x-coordinates [m]
%                    Y            - Array of y-coordinates [m]
%                    Cep2D_radius - Array of CEP radii [m]
%                                   for 2D (xy) localization
%                    omega1       - Minor semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%                    omega2       - Major semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%
% Input parameters: sensor1     - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2     - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev_TDOA - Standard deviation [ns] for the TDOA measurements
%                   stddev_AOA  - Standard deviation [deg] for the AOA measurements
%                   limits      - Vector giving minimum and maximum values for the
%                                 x- and y-coordinates [m]: [x_min x_max y_min y_max]
%                   nPoints     - Number of points in the grid along one
%                                 dimension (x or y, the same value is used
%                                 for both dimensions)
%
% Output files: TDOAAOA_Cep2D
%
% Input files: none
%
% External functions: ComputeCep2D
%                     ComputeCrlb2D_TDOAAOA
%                     ComputeCrlb2DMinMajAxes
%
% Last modified: 18.09.09
%
% ------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ------------------------------------

path('..\CRLB\', path);

x_min=limits(1);
x_max=limits(2);
y_min=limits(3);
y_max=limits(4);

% Calculate the variance for the TDOA measurements [s^2]
var_TDOA=(stddev_TDOA.*1e-9).^2;

% Calculate the variance for the AOA measurements [rad^2]
var_AOA=(stddev_AOA.*pi./180).^2;

% Calculate CEP radius in grid
```

```
[X,Y]=meshgrid(linspace(x_min,x_max,nPoints),linspace(y_min,y_max,nPoints));
Cep2D_radius=zeros(size(X));
[i_max,j_max]=size(X);
for i=1:1:i_max
    for j=1:1:j_max
        emitter=[X(i,j) Y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAAOA(sensor1, sensor2, emitter, ...
                                                   var_TDOA, var_AOA);
        [omega1(i,j), omega2(i,j)] = ComputeCrlb2DMinMajAxes(Crlb2D_CovMatrix);
        [Cep2D_radius(i,j)] = ComputeCep2D(omega1(i,j), omega2(i,j));
    end
end

% Plot CEP radius contours
figure;
m0=1:1:9;
m1=10:10:90;
m2=100:100:900;
m3=1e3:1e3:9e3;
m4=1e4:1e4:9e4;
m5=1e5:1e5:9e5;
m6=1e6:1e6:9e6;
m7=1e7:1e7:9e7;
m8=1e8:1e8:9e8;
m9=1e9:1e9:1e10;
v1=[m0 m1 m2 m3 m4 m5 m6 m7 m8 m9];
[C,h]=contour(X,Y,Cep2D_radius,v1,'k');
v2=[10 100 1000 1e4 1e5 1e6];
clabel(C,h,v2,'Color','k');
xlabel('Distance, x [km]');
ylabel('Distance, y [km]');
title('TDOAAOA');

% Values on figure axes when x_min=y_min=-100km and x_max=y_max=100km
set(gca, 'XTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'YTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'XTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);
set(gca, 'YTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);

% Save results to file
save outputfiles\Cep2D_TDOAAOA X Y Cep2D_radius omega1 omega2
```

```
function [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_TDOAScanPhaseAOA(sensor1, ...
                        sensor2, stddev_TDOA, stddev_AA, stddev_AOA, limits, nPoints);
% Plot CEP radius contours
%
% [X,Y,Cep2D_radius,omega1,omega2]=PlotCep2D_TDOAScanPhaseAOA(sensor1, sensor2, ...
%                                  stddev_TDOA, stddev_AA, stddev_AOA, limits, nPoints)
%
% Output parameters: X            - Array of x-coordinates [m]
%                    Y            - Array of y-coordinates [m]
%                    Cep2D_radius - Array of CEP radii [m]
%                                   for 2D (xy) localization
%                    omega1       - Minor semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%                    omega2       - Major semi-axis [m] for the CRLB error
%                                   ellipse (2D)
%
% Input parameters: sensor1     - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2     - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev_TDOA - Standard deviation [ns] for the TDOA measurements
%                   stddev_AA   - Standard deviation [deg] for the Aperture
%                                 Angle measurements
%                   stddev_AOA  - Standard deviation [deg] for the AOA measurements
%                   limits      - Vector giving minimum and maximum values for the
%                                 x- and y-coordinates [m]: [x_min x_max y_min y_max]
%                   nPoints     - Number of points in the grid along one
%                                 dimension (x or y, the same value is used
%                                 for both dimensions)
%
% Output files: TDOAScanPhaseAOA_Cep2D
%
% Input files: none
%
% External functions: ComputeCep2D
%                     ComputeCrlb2D_TDOAScanPhaseAOA
%                     ComputeCrlb2DMinMajAxes
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

path('..\CRLB\', path);

x_min=limits(1);
x_max=limits(2);
y_min=limits(3);
y_max=limits(4);

% Calculate the variance for the TDOA measurements [s^2]
var_TDOA=(stddev_TDOA.*1e-9).^2;

% Calculate the variance for the Aperture Angle measurements [rad^2]
var_AA=(stddev_AA.*pi./180).^2;
```

```
% Calculate the variance for the AOA measurements [rad^2]
var_AOA=(stddev_AOA.*pi./180).^2;

% Calculate CEP radius in grid
[X,Y]=meshgrid(linspace(x_min,x_max,nPoints),linspace(y_min,y_max,nPoints));
Cep2D_radius=zeros(size(X));
[i_max,j_max]=size(X);
for i=1:1:i_max
    for j=1:1:j_max
        emitter=[X(i,j) Y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAScanPhaseAOA(sensor1, sensor2, ...
                                                emitter, var_TDOA, var_AA, var_AOA);
        [omega1(i,j), omega2(i,j)] = ComputeCrlb2DMinMajAxes(Crlb2D_CovMatrix);
        [Cep2D_radius(i,j)] = ComputeCep2D(omega1(i,j), omega2(i,j));
    end
end

% Plot CEP radius contours
figure;
m0=1:1:9;
m1=10:10:90;
m2=100:100:900;
m3=1e3:1e3:9e3;
m4=1e4:1e4:9e4;
m5=1e5:1e5:9e5;
m6=1e6:1e6:9e6;
m7=1e7:1e7:9e7;
m8=1e8:1e8:9e8;
m9=1e9:1e9:1e10;
v1=[m0 m1 m2 m3 m4 m5 m6 m7 m8 m9];
[C,h]=contour(X,Y,Cep2D_radius,v1,'k');
v2=[10 100 1000 1e4 1e5 1e6];
clabel(C,h,v2,'Color','k');
xlabel('Distance, x [km]');
ylabel('Distance, y [km]');
title('TDOAScanPhaseAOA');

% Values on figure axes when x_min=y_min=-100km and x_max=y_max=100km
set(gca, 'XTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'YTick', [-100e3 -80e3 -60e3 -40e3 -20e3 0 20e3 40e3 60e3 80e3 100e3]);
set(gca, 'XTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);
set(gca, 'YTickLabel', [-100 -80 -60 -40 -20 0 20 40 60 80 100]);

% Save results to file
save outputfiles\Cep2D_TDOAScanPhaseAOA X Y Cep2D_radius omega1 omega2
```

```
function [Cep2D_radius] = ComputeCep2D(omega1, omega2);
% Calculate the CEP radius
%
% [Cep2D_radius] = ComputeCep2D(omega1, omega2)
%
% Output parameters: Cep2D_radius - CEP radius [m] for 2D (xy) localization
%
% Input parameters: omega1 - minor semi-axis for the CRLB error ellipse (2D)
%                   omega2 - major semi-axis for the CRLB error ellipse (2D)
%
% Output files: none
%
% Input files: none
%
% External functions: none
%
% Last modified: 18.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

% Set the initial value for the CEP radius equal to the major semi-axis for
% the CRLB error ellipse (2D)
Cep2D_radius=omega2;
delta=Cep2D_radius;


P=1;              %Set initial CEP probability equal to 1
P_lowlim =0.4999; %Gives error <0.1% compared to when using 0.499999999999
P_highlim=0.5001; %Gives error <0.1% compared to when using 0.500000000001
N=10;             %Gives error <0.1% compared to when using N=1000
i_max=1000;       %Maximum number of iterations
i=0;
% Compute the CEP radius
while P>P_highlim || P<P_lowlim;
    k=0:1:N;
    theta=(k./N).*pi./2;
    f=exp(-Cep2D_radius.^2./(2.*(omega1^2.*cos(theta).^2+omega2.^2*sin(theta).^2)));
    P=1-(f(1)./2+f(N+1)./2+sum(f(2:N)))./N;
    delta=delta./2;
    if P>P_highlim
        Cep2D_radius=Cep2D_radius-delta;
    elseif P<P_lowlim;
        Cep2D_radius=Cep2D_radius+delta;
    end
    i=i+1;
    if i>i_max
        disp('Error! Maximum number of iterations exceeded')
        Cep2D_radius=NaN;
        break
    end
end
```

```
function PlotCrlb2D_TDOA(sensor1, sensor2, stddev, radius, angle);
% Plot CRLB error ellipses
%
% PlotCrlb2D_TDOA(sensor1, sensor2, stddev, radius, angle)
%
% Output parameters: none
%
% Input parameters: sensor1 - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2 - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev  - Standard deviation [ns] for the TDOA measurements
%                   radius  - vector specifying at which radii [m] to plot
%                             the error ellipse
%                   angle   - vector specifying at which angles [deg] to plot
%                             the error ellipse
%
% Output files: none
%
% Input files: none
%
% External functions: ComputeCrlb2D_TDOA
%                     PlotCrlb2D_ErrorEllipse
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Calculate the variance for the TDOA measurements [s^2]
variance=(stddev.*1e-9).^2;

% Convert from degrees to radians
angle=angle.*pi./180;

% Calculate the CRLB covariance matrix at each given point
[xx,i_max]=size(radius);
[xx,j_max]=size(angle);
for i=1:1:i_max
    for j=1:1:j_max
        x(i,j)=radius(i).*cos(angle(j));
        y(i,j)=radius(i).*sin(angle(j));
        emitter=[x(i,j) y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOA(sensor1, sensor2, emitter, variance);
        PlotCrlb2DErrorEllipse([x(i,j) y(i,j)], Crlb2D_CovMatrix);
        hold on;
    end
end
```

```
function PlotCrlb2D_ScanPhase(sensor1, sensor2, stddev, radius, angle);
% Plot CRLB error ellipses
%
% PlotCrlb2D_ScanPhase(sensor1, sensor2, stddev, radius, angle)
%
% Output parameters: none
%
% Input parameters: sensor1 - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2 - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev  - Standard deviation [deg] for the Aperture Angle
%                             measurements
%                   radius  - vector specifying at which radii [m] to plot
%                             the error ellipse
%                   angle   - vector specifying at which angles [deg] to plot
%                             the error ellipse
%
% Output files: none
%
% Input files: none
%
% External functions: ComputeCrlb2D_ScanPhase
%                     PlotCrlb2D_ErrorEllipse
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Calculate the variance for the Aperture Angle measurements [rad^2]
variance=(stddev.*pi./180).^2;

% Convert from degrees to radians
angle=angle.*pi./180;

% Calculate the CRLB covariance matrix at each given point
[xx,i_max]=size(radius);
[xx,j_max]=size(angle);
for i=1:1:i_max
    for j=1:1:j_max
        x(i,j)=radius(i).*cos(angle(j));
        y(i,j)=radius(i).*sin(angle(j));
        emitter=[x(i,j) y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_ScanPhase(sensor1, sensor2, ...
                                                     emitter, variance);
        PlotCrlb2DErrorEllipse([x(i,j) y(i,j)], Crlb2D_CovMatrix);
        hold on;
    end
end
```

```
function PlotCrlb2D_AOA(sensor, stddev, radius, angle);
% Plot CRLB error ellipses
%
% PlotCrlb2D_AOA(sensor, stddev, radius, angle)
%
% Output parameters: none
%
% Input parameters: sensor - nx3 matrix specifying n sensor xyz-position(s) [m]
%                   stddev - Standard deviation [deg] for the AOA measurements
%                   radius - vector specifying at which radii [m] to plot
%                            the error ellipse
%                   angle  - vector specifying at which angles [deg] to plot
%                            the error ellipse
%
% Output files: none
%
% Input files: none
%
% External functions: ComputeCrlb2D_AOA
%                     PlotCrlb2D_ErrorEllipse
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Calculate the variance for the AOA measurements [rad^2]
variance=(stddev.*pi./180).^2;

% Convert from degrees to radians
angle=angle.*pi./180;

% Calculate the CRLB covariance matrix at each given point
[xx,i_max]=size(radius);
[xx,j_max]=size(angle);
for i=1:1:i_max
    for j=1:1:j_max
        x(i,j)=radius(i).*cos(angle(j));
        y(i,j)=radius(i).*sin(angle(j));
        emitter=[x(i,j) y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_AOA(sensor, emitter, variance);
        PlotCrlb2DErrorEllipse([x(i,j) y(i,j)], Crlb2D_CovMatrix);
        hold on;
    end
end
```

```
function PlotCrlb2D_ScanPhaseAOA(sensor1, sensor2, stddev_AA, stddev_AOA, ...
                                                                radius, angle);
% Plot CRLB error ellipses
%
% PlotCrlb2D_ScanPhaseAOA(sensor1, sensor2, stddev_AA, stddev_AOA, radius, angle)
%
% Output parameters: none
%
% Input parameters: sensor1     - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2     - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev_AA   - Standard deviation [deg] for the aperture angle
%                                 measurements
%                   stddev_AOA  - Standard deviation [deg] for the AOA measurements
%                   radius      - vector specifying at which radii [m] to plot
%                                 the error ellipse
%                   angle       - vector specifying at which angles [deg] to plot
%                                 the error ellipse
%
% Output files: none
%
% Input files: none
%
% External functions: ComputeCrlb2D_ScanPhaseAOA
%                     PlotCrlb2D_ErrorEllipse
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Calculate the variance for the Aperture Angle measurements [rad^2]
var_AA=(stddev_AA.*pi./180).^2;

% Calculate the variance for the AOA measurements [rad^2]
var_AOA=(stddev_AOA.*pi./180).^2;

% Convert from degrees to radians
angle=angle.*pi./180;

% Calculate the CRLB covariance matrix at each given point
[xx,i_max]=size(radius);
[xx,j_max]=size(angle);
for i=1:1:i_max
    for j=1:1:j_max
        x(i,j)=radius(i).*cos(angle(j));
        y(i,j)=radius(i).*sin(angle(j));
        emitter=[x(i,j) y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_ScanPhaseAOA(sensor1, sensor2, ...
                                                    emitter, var_AA, var_AOA);
        PlotCrlb2DErrorEllipse([x(i,j) y(i,j)], Crlb2D_CovMatrix);
        hold on;
    end
end
```

```
function PlotCrlb2D_TDOAScanPhase(sensor1, sensor2, stddev_TDOA, stddev_AA, ...
                                                                    radius, angle);
% Plot CRLB error ellipses
%
% PlotCrlb2D_TDOAScanPhase(sensor1, sensor2, stddev_TDOA, stddev_AA, radius, angle)
%
% Output parameters: none
%
% Input parameters: sensor1      - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2      - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev_TDOA  - Standard deviation [ns] for the TDOA measurements
%                   stddev_AA    - Standard deviation [deg] for the aperture angle
%                                  measurements
%                   radius       - vector specifying at which radii [m] to plot
%                                  the error ellipse
%                   angle        - vector specifying at which angles [deg] to plot
%                                  the error ellipse
%
% Output files: none
%
% Input files: none
%
% External functions: ComputeCrlb2D_TDOAScanPhase
%                     PlotCrlb2D_ErrorEllipse
%
% Last modified: 18.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

% Calculate the variance for the TDOA measurements [s^2]
var_TDOA=(stddev_TDOA.*1e-9).^2;

% Calculate the variance for the Aperture Angle measurements [rad^2]
var_AA=(stddev_AA.*pi./180).^2;

% Convert from degrees to radians
angle=angle.*pi./180;

% Calculate the CRLB covariance matrix at each given point
[xx,i_max]=size(radius);
[xx,j_max]=size(angle);
for i=1:1:i_max
    for j=1:1:j_max
        x(i,j)=radius(i).*cos(angle(j));
        y(i,j)=radius(i).*sin(angle(j));
        emitter=[x(i,j) y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAScanPhase(sensor1, sensor2, ...
                                                    emitter, var_TDOA, var_AA);
        PlotCrlb2DErrorEllipse([x(i,j) y(i,j)], Crlb2D_CovMatrix);
        hold on;
    end
end
```

```
function PlotCrlb2D_TDOAAOA(sensor1, sensor2, stddev_TDOA, stddev_AOA, radius, angle);
% Plot CRLB error ellipses
%
% PlotCrlb2D_TDOAAOA(sensor1, sensor2, stddev_TDOA, stddev_AOA, radius, angle)
%
% Output parameters: none
%
% Input parameters: sensor1      - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2      - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev_TDOA  - Standard deviation [ns] for the TDOA measurements
%                   stddev_AOA   - Standard deviation [deg] for the AOA measurements
%                   radius       - vector specifying at which radii [m] to plot
%                                  the error ellipse
%                   angle        - vector specifying at which angles [deg] to plot
%                                  the error ellipse
%
% Output files: none
%
% Input files: none
%
% External functions: ComputeCrlb2D_TDOAAOA
%                     PlotCrlb2D_ErrorEllipse
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Calculate the variance for the TDOA measurements [s^2]
var_TDOA=(stddev_TDOA.*1e-9).^2;

% Calculate the variance for the AOA measurements [rad^2]
var_AOA=(stddev_AOA.*pi./180).^2;

% Convert from degrees to radians
angle=angle.*pi./180;

% Calculate the CRLB covariance matrix at each given point
[xx,i_max]=size(radius);
[xx,j_max]=size(angle);
for i=1:1:i_max
    for j=1:1:j_max
        x(i,j)=radius(i).*cos(angle(j));
        y(i,j)=radius(i).*sin(angle(j));
        emitter=[x(i,j) y(i,j) 0];
        [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAAOA(sensor1, sensor2, emitter, ...
                                                   var_TDOA, var_AOA);
        PlotCrlb2DErrorEllipse([x(i,j) y(i,j)], Crlb2D_CovMatrix);
        hold on;
    end
end
```

```
function PlotCrlb2D_TDOAScanPhaseAOA(sensor1, sensor2, stddev_TDOA, stddev_AA, ...
                                                        stddev_AOA, radius, angle);
% Plot CRLB error ellipses
%
% PlotCrlb2D_TDOAScanPhaseAOA(sensor1, sensor2, stddev_TDOA, stddev_AA, ...
%                                                        stddev_AOA, radius, angle)
%
% Output parameters: none
%
% Input parameters: sensor1      - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2      - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   stddev_TDOA  - Standard deviation [ns] for the TDOA measurements
%                   stddev_AA    - Standard deviation [deg] for the aperture angle
%                                  measurements
%                   stddev_AOA   - Standard deviation [deg] for the AOA measurements
%                   radius       - vector specifying at which radii [m] to plot
%                                  the error ellipse
%                   angle        - vector specifying at which angles [deg] to plot
%                                  the error ellipse
%
% Output files: none
%
% Input files: none
%
% External functions: ComputeCrlb2D_TDOAScanPhaseAOA
%                     PlotCrlb2D_ErrorEllipse
%
% Last modified: 18.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

% Calculate the variance for the TDOA measurements [s^2]
var_TDOA=(stddev_TDOA.*1e-9).^2;

% Calculate the variance for the Aperture Angle measurements [rad^2]
var_AA=(stddev_AA.*pi./180).^2;

% Calculate the variance for the AOA measurements [rad^2]
var_AOA=(stddev_AOA.*pi./180).^2;

% Convert from degrees to radians
angle=angle.*pi./180;

% Calculate the CRLB covariance matrix at each given point
[xx,i_max]=size(radius);
[xx,j_max]=size(angle);
for i=1:1:i_max
    for j=1:1:j_max
        x(i,j)=radius(i).*cos(angle(j));
        y(i,j)=radius(i).*sin(angle(j));
        emitter=[x(i,j) y(i,j) 0];
```

```
        [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAScanPhaseAOA(sensor1, sensor2, ...
                                                emitter, var_TDOA, var_AA, var_AOA);
        PlotCrlb2DErrorEllipse([x(i,j) y(i,j)], Crlb2D_CovMatrix);
        hold on;
    end
end
```

```
function PlotCrlb2DErrorEllipse(center, Crlb2D_CovMatrix);
% Plot CRLB error ellipse (2D)
%
% PlotCrlb2DErrorEllipse(Center, Crlb2D_CovMatrix)
%
% Output parameters: none
%
% Input parameters: center           - Center position (x0,y0) for
%                                      the error ellipse [m]
%                   Crlb2D_CovMatrix - CRLB covariance matrix for
%                                      2D (xy) localization
%
% Output files: none
%
% Input files: none
%
% External functions: none
%
% Last modified: 18.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

% Check if the CRLB covariance matrix contains infinite or NaN elements
if sum(sum(isinf(Crlb2D_CovMatrix)))>0 || sum(sum(isnan(Crlb2D_CovMatrix)))>0;
    return;
end

% Compute eigenvectors and eigenvalues of the CRLB covariance matrix
[EigenvectorMatrix, EigenvalueMatrix]=eig(Crlb2D_CovMatrix);

% Compute minor and major semi-axes of the CRLB error ellipse
omega1=sqrt(EigenvalueMatrix(1,1));
omega2=sqrt(EigenvalueMatrix(2,2));

% Compute x- and y-components of the two eigenvectors
x1=EigenvectorMatrix(1,1);
x2=EigenvectorMatrix(1,2);
y1=EigenvectorMatrix(2,1);
y2=EigenvectorMatrix(2,2);

% Center coordinates [m] for the CRLB error ellipse
x0=center(1);
y0=center(2);

% Compute corresponding x- and y-values for the CRLB error ellipse
beta=0:0.01:2.*pi;
x=x0+omega1.*x1.*cos(beta)+(omega2.*x2).*sin(beta);
y=y0+omega1.*y1.*cos(beta)+(omega2.*y2).*sin(beta);

% Plot the CRLB error ellipse
plot(x,y,'r-');
```

```
function [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOA(sensor1, sensor2, emitter, variance);
% Calculate the CRLB covariance matrix for 2D (xy) localization
%
% [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOA(sensor1, sensor2, emitter, variance)
%
% Output parameters: Crlb2D_CovMatrix - CRLB covariance matrix for
%                                       2D (xy) localization
%
% Input parameters: sensor1  - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2  - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   emitter  - 1x3 matrix specifying the emitter xyz-position [m]
%                   variance - Variance for the TDOA measurements [s^2]
%
% Output files: none
%
% Input files: none
%
% External functions: Compute_dTDOA
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

dim=2;   % 2 dimensions

% Find number of measurements (n)
[n,xx]=size(sensor1);

% Generate nxn variance matrix (A)
A=eye(n).*variance;

% Calculate the partial derivatives of the time difference of arrival
% with respect to x and y
[dTDOA_dx dTDOA_dy]=Compute_dTDOA(sensor1, sensor2, emitter);

% Generate 2xn partial derivatives matrix (B)
B=zeros(dim,n);
B(1,:)=dTDOA_dx;
B(2,:)=dTDOA_dy;

% Calculate CRLB covariance matrix
Crlb2D_CovMatrix=inv(B*inv(A)*B');
```

```
function [Crlb2D_CovMatrix] = ComputeCrlb2D_ScanPhase(sensor1, sensor2, ...
                                                      emitter, variance);
% Calculate the CRLB covariance matrix for 2D (xy) localization
%
% [Crlb2D_CovMatrix] = ComputeCrlb2D_ScanPhase(sensor1, sensor2, emitter, variance)
%
% Output parameters: Crlb2D_CovMatrix - CRLB covariance matrix for
%                                       2D (xy) localization
%
% Input parameters: sensor1  - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2  - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   emitter  - 1x3 matrix specifying the emitter xyz-position [m]
%                   variance - Variance for the Aperture Angle measurements [rad^2]
%
% Output files: none
%
% Input files: none
%
% External functions: Compute_dApertureAngle
%
% Last modified: 18.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

dim=2;   % 2 dimensions

% Find number of measurements (n)
[n,xx]=size(sensor1);

% Generate nxn variance matrix (A)
A=eye(n).*variance;

% Calculate the partial derivatives of the aperture angle with respect to x and y
[dApertureAngle_dx dApertureAngle_dy]=Compute_dApertureAngle(sensor1, sensor2,emitter);

% Generate 2xn partial derivatives matrix (B)
B=zeros(dim,n);
B(1,:)=dApertureAngle_dx;
B(2,:)=dApertureAngle_dy;

% Calculate CRLB covariance matrix
Crlb2D_CovMatrix=inv(B*inv(A)*B');
```

```
function [Crlb2D_CovMatrix] = ComputeCrlb2D_AOA(sensor, emitter, variance);
% Calculate the CRLB covariance matrix for 2D (xy) localization
%
% [Crlb2D_CovMatrix] = ComputeCrlb2D_AOA(sensor, emitter, variance)
%
% Output parameters: Crlb2D_CovMatrix - CRLB covariance matrix for
%                                       2D (xy) localization
%
% Input parameters: sensor   - nx3 matrix specifying n sensor xyz-position(s) [m]
%                   emitter  - 1x3 matrix specifying the emitter xyz-position [m]
%                   variance - Variance for the AOA measurements [rad^2]
%
% Output files: none
%
% Input files: none
%
% External functions: Compute_dAOA
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

dim=2;   % 2 dimensions

% Find number of measurements (n)
[n,xx]=size(sensor);

% Generate nxn variance matrix (A)
A=eye(n).*variance;

% Calculate the partial derivatives of AOA with respect to x and y
[dAOA_dx dAOA_dy]=Compute_dAOA(sensor, emitter);

% Generate 2xn partial derivatives matrix (B)
B=zeros(dim,n);
B(1,:)=dAOA_dx;
B(2,:)=dAOA_dy;

% Calculate CRLB covariance matrix
Crlb2D_CovMatrix=inv(B*inv(A)*B');
```

```
function [Crlb2D_CovMatrix] = ComputeCrlb2D_ScanPhaseAOA(sensor1, sensor2, ...
                                                         emitter, var_AA, var_AOA);
% Calculate the CRLB covariance matrix for 2D (xy) localization
%
% [Crlb2D_CovMatrix] = ComputeCrlb2D_ScanPhaseAOA(sensor1, sensor2, emitter, ...
%                                                                  var_AA, var_AOA)
%
% Output parameters: Crlb2D_CovMatrix - CRLB covariance matrix for
%                                       2D (xy) localization
%
% Input parameters: sensor1  - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2  - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   emitter  - 1x3 matrix specifying the emitter xyz-position [m]
%                   var_AA   - Variance for the Aperture Angle measurements [rad^2]
%                   var_AOA  - Variance for the Aperture Angle measurements [rad^2]
%
% Output files: none
%
% Input files: none
%
% External functions: Compute_dApertureAngle
%                     Compute_dAOA
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

dim=2;   % 2 dimensions

% Find number of measurements (n)
[n,xx]=size(sensor1);

% Generate 3nx3n variance matrix (A)
z=zeros(n,n);
a1=eye(n).*var_AA;
a2=eye(n).*var_AOA;
A=[a1 z z ; z a2 z ; z z a2];

% Calculate the partial derivatives of the aperture angle and
% angle of arrival with respect to x and y
[dApertureAngle_dx dApertureAngle_dy]=Compute_dApertureAngle(sensor1, sensor2,emitter);
[dAOA1_dx dAOA1_dy]=Compute_dAOA(sensor1, emitter);
[dAOA2_dx dAOA2_dy]=Compute_dAOA(sensor2, emitter);

% Generate 2x3n partial derivatives matrix (B)
B=zeros(dim,3.*n);
B(1,1:n)=dApertureAngle_dx;
B(2,1:n)=dApertureAngle_dy;
B(1,n+1:2.*n)=dAOA1_dx;
B(2,n+1:2.*n)=dAOA1_dy;
B(1,2.*n+1:3.*n)=dAOA2_dx;
B(2,2.*n+1:3.*n)=dAOA2_dy;
```

```
% Calculate CRLB covariance matrix
Crlb2D_CovMatrix=inv(B*inv(A)*B');
```

```
function [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAScanPhase(sensor1, sensor2, ...
                                                    emitter, var_TDOA, var_AA);
% Calculate the CRLB covariance matrix for 2D (xy) localization
%
% [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAScanPhase(sensor1, sensor2, emitter, ...
%                                                          var_TDOA, var_AA)
%
% Output parameters: Crlb2D_CovMatrix - CRLB covariance matrix for
%                                       2D (xy) localization
%
% Input parameters: sensor1  - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2  - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   emitter  - 1x3 matrix specifying the emitter xyz-position [m]
%                   var_TDOA - Variance for the TDOA measurements [s^2]
%                   var_AA   - Variance for the Aperture Angle measurements [rad^2]
%
% Output files: none
%
% Input files: none
%
% External functions: Compute_dTDOA
%                     Compute_dApertureAngle
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

dim=2;   % 2 dimensions

% Find number of measurements (n)
[n,xx]=size(sensor1);

% Generate 2nx2n variance matrix (A)
z=zeros(n,n);
a1=eye(n).*var_TDOA;
a2=eye(n).*var_AA;
A=[a1 z; z a2];

% Calculate the partial derivatives of the time difference of arrival and
% aperture angle with respect to x and y
[dTDOA_dx dTDOA_dy]=Compute_dTDOA(sensor1, sensor2, emitter);
[dApertureAngle_dx dApertureAngle_dy]=Compute_dApertureAngle(sensor1, sensor2,emitter);

% Generate 2x2n partial derivatives matrix (B)
B=zeros(dim,2.*n);
B(1,1:n)=dTDOA_dx;
B(2,1:n)=dTDOA_dy;
B(1,n+1:2.*n)=dApertureAngle_dx;
B(2,n+1:2.*n)=dApertureAngle_dy;

% Calculate CRLB covariance matrix
Crlb2D_CovMatrix=inv(B*inv(A)*B');
```

```
function [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAAOA(sensor1, sensor2, emitter, ...
                                                    var_TDOA, var_AOA);
% Calculate the CRLB covariance matrix for 2D (xy) localization
%
% [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAAOA(sensor1, sensor2, emitter, ...
%                                                      var_TDOA, var_AOA)
%
% Output parameters: Crlb2D_CovMatrix - CRLB covariance matrix for
%                                       2D (xy) localization
%
% Input parameters: sensor1  - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2  - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   emitter  - 1x3 matrix specifying the emitter xyz-position [m]
%                   var_TDOA - Variance for the TDOA measurements [s^2]
%                   var_AOA  - Variance for the AOA measurements [rad^2]
%
% Output files: none
%
% Input files: none
%
% External functions: Compute_dTDOA
%                     Compute_dAOA
%
% Last modified: 18.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

dim=2;   % 2 dimensions

% Find number of measurements (n)
[n,xx]=size(sensor1);

% Generate 3nx3n variance matrix (A)
z=zeros(n,n);
a1=eye(n).*var_TDOA;
a2=eye(n).*var_AOA;
A=[a1 z z ; z a2 z ; z z a2];

% Calculate the partial derivatives of the time difference of arrival and
% angle of arrival with respect to x and y
[dTDOA_dx dTDOA_dy]=Compute_dTDOA(sensor1, sensor2, emitter);
[dAOA1_dx dAOA1_dy]=Compute_dAOA(sensor1, emitter);
[dAOA2_dx dAOA2_dy]=Compute_dAOA(sensor2, emitter);

% Generate 2x3n partial derivatives matrix (B)
B=zeros(dim,3.*n);
B(1,1:n)=dTDOA_dx;
B(2,1:n)=dTDOA_dy;
B(1,n+1:2.*n)=dAOA1_dx;
B(2,n+1:2.*n)=dAOA1_dy;
B(1,2.*n+1:3.*n)=dAOA2_dx;
B(2,2.*n+1:3.*n)=dAOA2_dy;
```

```
% Calculate CRLB covariance matrix
Crlb2D_CovMatrix=inv(B*inv(A)*B');
```

```
function [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAScanPhaseAOA(sensor1, sensor2, ...
                                                  emitter, var_TDOA, var_AA, var_AOA);
% Calculate the CRLB covariance matrix for 2D (xy) localization
%
% [Crlb2D_CovMatrix] = ComputeCrlb2D_TDOAScanPhaseAOA(sensor1, sensor2, emitter, ...
%                                                        var_TDOA, var_AA, var_AOA)
%
% Output parameters: Crlb2D_CovMatrix - CRLB covariance matrix for
%                                       2D (xy) localization
%
% Input parameters: sensor1  - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2  - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   emitter  - 1x3 matrix specifying the emitter xyz-position [m]
%                   var_TDOA - Variance for the TDOA measurements [s^2]
%                   var_AA   - Variance for the Aperture Angle measurements [rad^2]
%                   var_AOA  - Variance for the AOA measurements [rad^2]
%
% Output files: none
%
% Input files: none
%
% External functions: Compute_dTDOA
%                     Compute_dApertureAngle
%                     Compute_dAOA
%
% Last modified: 18.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

dim=2;   % 2 dimensions

% Find number of measurements (n)
[n,xx]=size(sensor1);

% Generate 4nx4n variance matrix (A)
z=zeros(n,n);
a1=eye(n).*var_TDOA;
a2=eye(n).*var_AA;
a3=eye(n).*var_AOA;
A=[a1 z z z; z a2 z z; z z a3 z; z z z a3];

% Calculate the partial derivatives of the time difference of arrival,
% aperture angle, and angle of arrival with respect to x and y
[dTDOA_dx dTDOA_dy]=Compute_dTDOA(sensor1, sensor2, emitter);
[dApertureAngle_dx dApertureAngle_dy]=Compute_dApertureAngle(sensor1, sensor2,↙
emitter);
[dAOA1_dx dAOA1_dy]=Compute_dAOA(sensor1, emitter);
[dAOA2_dx dAOA2_dy]=Compute_dAOA(sensor2, emitter);

% Generate 2x4n partial derivatives matrix (B)
B=zeros(dim,4.*n);
B(1,1:n)=dTDOA_dx;
```

```
B(2,1:n)=dTDOA_dy;
B(1,n+1:2.*n)=dApertureAngle_dx;
B(2,n+1:2.*n)=dApertureAngle_dy;
B(1,2.*n+1:3.*n)=dAOA1_dx;
B(2,2.*n+1:3.*n)=dAOA1_dy;
B(1,3.*n+1:4.*n)=dAOA2_dx;
B(2,3.*n+1:4.*n)=dAOA2_dy;

% Calculate CRLB covariance matrix
Crlb2D_CovMatrix=inv(B*inv(A)*B');
```

```
function [dTDOA_dx dTDOA_dy] = Compute_dTDOA(sensor1, sensor2, emitter);
% Calculate the partial derivatives for the time difference of arrival(s) for given ↙
emitter and sensor position(s)
%
% [dTDOA_dx dTDOA_dy] = Compute_dTDOA(sensor1, sensor2, emitter)
%
% Output parameters: dTDOA_dx - nx1 matrix specifying the partial derivatives
%                               of the TDOA(s) with respect to x [s/m]
%                    dTDOA_dy - nx1 matrix specifying the partial derivatives
%                               of the TDOA(s) with respect to y [s/m]
%
% Input parameters: sensor1 - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2 - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   emitter - 1x3 matrix specifying the emitter xyz-position [m]
%
% Output files: none
%
% Input files: none
%
% External functions: none
%
% Last modified: 18.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

c=2.99792458e8;  % Speed of light [m/s]

% Sensor1 positions [m]
x_s1=sensor1(:,1);
y_s1=sensor1(:,2);
z_s1=sensor1(:,3);

% Sensor2 positions [m]
x_s2=sensor2(:,1);
y_s2=sensor2(:,2);
z_s2=sensor2(:,3);

% Emitter position [m]
x_e=emitter(1);
y_e=emitter(2);
z_e=emitter(3);

% Calculate distances d_1 and d_2 between sensor1 and the emitter and sensor2 and
% the emitter [m]
d_1=sqrt((x_s1-x_e).^2+(y_s1-y_e).^2+(z_s1-z_e).^2);
d_2=sqrt((x_s2-x_e).^2+(y_s2-y_e).^2+(z_s2-z_e).^2);

% Calculate the partial derivatives for the TDOA [s/m]
dTDOA_dx=1./c.*((x_s2-x_e)./d_2-(x_s1-x_e)./d_1);
dTDOA_dy=1./c.*((y_s2-y_e)./d_2-(y_s1-y_e)./d_1);
```

```
function [dAA_dx dAA_dy] = Compute_dApertureAngle(sensor1, sensor2, emitter);
% Calculate the partial derivatives for the aperture angle(s) for given emitter and ↙
sensor position(s)
%
% [dAA_dx dAA_dy] = Compute_dApertureAngle(sensor1, sensor2, emitter)
%
% Output parameters: dAA_dx - nx1 matrix specifying the partial derivatives
%                             of the aperture angle(s) with respect
%                             to x [rad/m]
%                    dAA_dy - nx1 matrix specifying the partial derivatives
%                             of the aperture angle(s) with respect
%                             to y [rad/m]
%
% Input parameters: sensor1 - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2 - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   emitter - 1x3 matrix specifying the emitter xyz-position [m]
%
% Output files: none
%
% Input files: none
%
% External functions: none
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Sensor1 positions [m]
x_s1=sensor1(:,1);
y_s1=sensor1(:,2);

% Sensor2 positions [m]
x_s2=sensor2(:,1);
y_s2=sensor2(:,2);

% Emitter positions [m]
x_e=emitter(1);
y_e=emitter(2);

d_1 =sqrt((x_s1-x_e).^2+(y_s1-y_e).^2);
d_2 =sqrt((x_s2-x_e).^2+(y_s2-y_e).^2);
d_12=sqrt((x_s1-x_s2).^2+(y_s1-y_s2).^2);

dd1_dx=-(x_s1-x_e)./d_1;
dd2_dx=-(x_s2-x_e)./d_2;

dd1_dy=-(y_s1-y_e)./d_1;
dd2_dy=-(y_s2-y_e)./d_2;

a=-2.*d_1.*d_2;
da_dx=-2.*(d_2.*dd1_dx+d_1.*dd2_dx);
da_dy=-2.*(d_2.*dd1_dy+d_1.*dd2_dy);
```

```
u=(d_12.^2-d_1.^2-d_2.^2)./(-2.*d_1.*d_2);
du_dx=-u./a.*da_dx-2./a.*(d_1.*dd1_dx+d_2.*dd2_dx);
du_dy=-u./a.*da_dy-2./a.*(d_1.*dd1_dy+d_2.*dd2_dy);

% Calculate the partial derivatives for the aperture angle [rad/m]
dAA_dx=-1./sqrt(1-u.^2).*du_dx;
dAA_dy=-1./sqrt(1-u.^2).*du_dy;
```

```
function [dAOA_dx dAOA_dy] = Compute_dAOA(sensor, emitter);
% Calculate the partial derivatives for the angle(s) of arrival for given emitter and↙
sensor position(s)
%
% [dAOA_dx dAOA_dy] = Compute_dAOA(sensor, emitter)
%
% Output parameters: dAOA_dx - nx1 matrix specifying the partial derivatives
%                              of the angle(s) of arrival with respect
%                              to x [rad/m]
%                    dAOA_dy - nx1 matrix specifying the partial derivatives
%                              of the angle(s) of arrival with respect
%                              to y [rad/m]
%
% Input parameters: sensor  - nx3 matrix specifying n sensor xyz-position(s) [m]
%                   emitter - 1x3 matrix specifying the emitter xyz-position [m]
%
% Output files: none
%
% Input files: none
%
% External functions: none
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Sensor positions [m]
x_s1=sensor(:,1);
y_s1=sensor(:,2);

% Emitter position [m]
x_e=emitter(1);
y_e=emitter(2);

x=x_e-x_s1;
y=y_e-y_s1;

% Calculate the partial derivatives for the angle of arrival [rad/m]
dAOA_dx=y./(x.^2+y.^2);
dAOA_dy=-x./(x.^2+y.^2);
```

```
function TDOA_Plot2D_EmitterPos_DiffSensorPos(x_max, sensor1, sensor2, emitter);
% Plot emitter positions in 2D for different sensor positions (emitter plane at z=0)
%
% TDOA_Plot2D_EmitterPos_DiffSensorPos(x_max, sensor1, sensor2, emitter)
%
% Output parameters: none

% Input parameters: x_max   - Maximum x-value used for the calculations [m]
%                   sensor1 - nx3 vector describing n sensor 1 xyz-positions [m]
%                   sensor2 - nx3 vector describing n sensor 2 xyz-positions [m]
%                   emitter - 1x3 vector describing emitter xyz-position [m]
%
% Output files: none
%
% Input files: none
%
% External functions: Compute_TDOA
%                     TDOA_Compute2D_EmitterPositions
%
% Last modified: 17.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

% Compute TDOA values
[TDOA]=Compute_TDOA(sensor1, sensor2, emitter);

figure;
grid on;
hold on;

[n_max, xx]=size(TDOA);
for n=1:1:n_max
    % Compute possible emitter positions
    [xy_U_n xy_L_n]=TDOA_Compute2D_EmitterPositions(x_max, sensor1(n,:), ...
                                                    sensor2(n,:), TDOA(n));

    % Plot possible emitter positions
    plot(xy_U_n(:,1), xy_U_n(:,2), 'b-', xy_L_n(:,1), xy_L_n(:,2), 'b-');

    % Plot sensor1 and sensor2 position
    plot(sensor1(n,1), sensor1(n,2), 'sr', 'MarkerFaceColor', 'r');
    plot(sensor2(n,1), sensor2(n,2), 'sg', 'MarkerFaceColor', 'g');
end

% Plot emitter position
plot(emitter(1), emitter(2),'ok', 'MarkerFaceColor', 'k');
```

```
function [xy_U xy_L]=TDOA_Plot2D_EmitterPositions(x_max, sensor1, sensor2, TDOA);
% Plot emitter positions in 2D (emitter plane at z=0)
%
% [xy_U xy_L]=TDOA_Plot2D_EmitterPositions(x_max, sensor1, sensor2, TDOA)
%
% Output parameters: xy_U - nx2 vector with [x y]-values for the upper (U)
%                           branch of the hyperbolic surface that describes
%                           the possible emitter positions [m]
%                    xy_L - nx2 vector with [x y]-values for the lower (L)
%                           branch of the hyperbolic surface that describes
%                           the possible emitter positions [m]

% Input parameters: x_max   - Maximum x-value used for the calculations [m]
%                   sensor1 - 1x3 vector describing sensor 1 xyz-position [m]
%                   sensor2 - 1x3 vector describing sensor 2 xyz-position [m]
%                   TDOA    - Time difference of arrival [s]
%
% Output files: none
%
% Input files: none
%
% External functions: TDOA_Compute2D_EmitterPositions
%
% Last modified: 17.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Compute possible emitter positions
[xy_U xy_L]=TDOA_Compute2D_EmitterPositions(x_max, sensor1, sensor2, TDOA);

% Plot possible emitter positions
plot(xy_U(:,1), xy_U(:,2), 'b-', xy_L(:,1), xy_L(:,2), 'b-');
```

```
function [xy_U xy_L]=TDOA_Compute2D_EmitterPositions(x_max, sensor1, sensor2, TDOA);
% Compute emitter positions in 2D (emitter plane at z=0)
%
% [xy_U xy_L]=TDOA_Compute2D_EmitterPositions(x_max, sensor1, sensor2, TDOA)
%
% Output parameters: xy_U - nx2 vector with [x y]-values for the upper (U)
%                           branch of the hyperbolic surface that describes
%                           the possible emitter positions [m]
%                    xy_L - nx2 vector with [x y]-values for the lower (L)
%                           branch of the hyperbolic surface that describes
%                           the possible emitter positions [m]

% Input parameters: x_max   - Maximum x-value used for the calculations [m]
%                   sensor1 - 1x3 vector describing sensor 1 xyz-position [m]
%                   sensor2 - 1x3 vector describing sensor 2 xyz-position [m]
%                   TDOA    - Time difference of arrival [s]
%
% Output files: none
%
% Input files: none
%
% External functions: ComputeHyperbolicCrossSectionRotated
%
% Last modified: 17.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

c=2.99792458e8; % Speed of light [m/s]

% Sensor1 position [m]
x1=sensor1(1);
y1=sensor1(2);
z1=sensor1(3);

% Sensor2 position [m]
x2=sensor2(1);
y2=sensor2(2);
z2=sensor2(3);

if z2~=z1
    disp(' ');
    disp('ERROR! Both sensors must have the same height.');
    disp(' ');
    return;
end

% Calculate the distance (delta_S) between the sensors
delta_x=x2-x1;
delta_y=y2-y1;
delta_S=sqrt(delta_x.^2+delta_y.^2);

% Calculate constants a and b
```

```
a=c.*abs(TDOA)./2;
b=sqrt(delta_S.^2./4-a.^2);

% Find the center coordinates for the hyperbolic surface
x0=(x1+x2)./2;
y0=(y1+y2)./2;
center=[x0 y0];

% Calculate the rotation angle about the z-axis
phi=atan2(delta_y,delta_x);

% Calculate the rotated hyperbolic cross section with the emitter plane (z=0)
z_h=-z1;
[xy_rot_LU xy_rot_LL xy_rot_RU xy_rot_RL]=ComputeHyperbolicCrossSectionRotated...
                                                    (x_max, a, b, center, phi, z_h);

if TDOA<0
    xy_U=xy_rot_LU;
    xy_L=xy_rot_LL;
else
    xy_U=xy_rot_RU;
    xy_L=xy_rot_RL;
end
```

```
function [xy_rot_LU xy_rot_LL xy_rot_RU xy_rot_RL]=ComputeHyperbolicCrossSectionRotated
(x_max, a, b, center, phi, z_h);
% Compute rotated hyperbolic cross section
%
% [xy_rot_LU xy_rot_LL xy_rot_RU xy_rot_RL]=ComputeHyperbolicCrossSectionRotated(x_max,
a, b, center, phi, z_h)
%
% Output parameters: xy_rot_LU - nx2 vector with [x y]-values for the
%                                left (L) upper (U) branch of the
%                                hyperbolic cross section
%                    xy_rot_LL - nx2 vector with [x y]-values for the
%                                left (L) lower (L) branch of the
%                                hyperbolic cross section
%                    xy_rot_RU - nx2 vector with [x y]-values for the
%                                right (R) upper (U) branch of the
%                                hyperbolic cross section
%                    xy_rot_RL - nx2 vector with [x y]-values for the
%                                right (R) lower (L) branch of the
%                                hyperbolic cross section
%
% Input parameters: x_max  - Maximum x-value used for the calculations [m]
%                   a      - Major semi-axis of the original hyperbolic surface [m]
%                   b      - Minor semi-axis of the original hyperbolic surface [m]
%                   center - Center coordinates [x0,y0] for the hyperbola [m]
%                   phi    - Rotation angle [rad]
%                   z_h    - Coordinate of the cross section plane (z=z_h)
%
% Output files: none
%
% Input files: none
%
% External functions: ComputeHyperbolaRotated
%
% Last modified: 18.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Check if the hyperbolic cross section exists. If it does, calculate the
% parameters a_h and b_h.
if b==0
    if z_h==0
        a_h=a;
        b_h=b;
    else
        disp(' ');
        disp('ERROR!! There is no hyperbolic cross section with the specified plane
z=z_h.');
        disp(' ');
        return;
    end
else
    a_h=a.*sqrt(1+z_h.^2./b.^2);
```

```
    b_h=b.*sqrt(1+z_h.^2./b.^2);
end

% Calculate the rotated hyperbolic cross section
[xy_rot_LU xy_rot_LL xy_rot_RU xy_rot_RL]=ComputeHyperbolaRotated(x_max, a_h, b_h, ...
                                                                  center, phi);
```

```
function [xy_rot_LU xy_rot_LL xy_rot_RU xy_rot_RL]=ComputeHyperbolaRotated(x_max, ...
                                                    a, b, center, phi);
% Compute rotated hyperbola
%
% [y_rot_LU y_rot_LL y_rot_RU y_rot_RL]=ComputeHyperbolaRotated(x_max, a, b, ...
%                                                      center, phi)
%
% Output parameters: xy_rot_LU - nx2 vector with rotated [x y]-values for the
%                                left (L) upper (U) branch of the hyperbola
%                    xy_rot_LL - nx2 vector with rotated [x y]-values for the
%                                left (L) lower (L) branch of the hyperbola
%                    xy_rot_RU - nx2 vector with rotated [x y]-values for the
%                                right (R) upper (U) branch of the hyperbola
%                    xy_rot_RL - nx2 vector with rotated [x y]-values for the
%                                right (R) lower (L) branch of the hyperbola
%
% Input parameters: x_max  - Maximum x-value used for the calculations [m]
%                   a      - Major semi-axis of the hyperbola [m]
%                   b      - Minor semi-axis of the hyperbola [m]
%                   center - Center coordinates [x0,y0] for the hyperbola [m]
%                   phi    - Rotation angle [rad]
%
% Output files: none
%
% Input files: none
%
% External functions: ComputeHyperbola
%
% Last modified: 17.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

% Center coordinates
x0=center(1);
y0=center(2);

% Compute original (not rotated) hyperbola
[xy_LU xy_LL xy_RU xy_RL]=ComputeHyperbola(x_max, a, b);

% Rotate the hyperbola
[xy_rot_LU]=xy2xRyR(xy_LU, x0, y0, phi);
[xy_rot_LL]=xy2xRyR(xy_LL, x0, y0, phi);
[xy_rot_RU]=xy2xRyR(xy_RU, x0, y0, phi);
[xy_rot_RL]=xy2xRyR(xy_RL, x0, y0, phi);


function [xy_R]=xy2xRyR(xy, x0, y0, phi)
% Rotate from [x y] to [x_R y_R]
x=xy(:,1);
y=xy(:,2);
R=[cos(phi) -sin(phi); sin(phi) cos(phi)];
x_R=x0 + R(1,1).*x + R(1,2).*y;
```

```
y_R=y0 + R(2,1).*x + R(2,2).*y;
xy_R=[x_R y_R];
```

```
function [xy_LU xy_LL xy_RU xy_RL]=ComputeHyperbola(x_max, a, b)
% Compute hyperbola
%
% [xy_LU xy_LL xy_RU xy_RL]=ComputeHyperbola(x_max, a, b)
%
% Output parameters: xy_LU - nx2 vector with [x y]-values for the left (L)
%                            upper (U) branch of the hyperbola
%                    xy_LL - nx2 vector with [x y]-values for the left (L)
%                            lower (L) branch of the hyperbola
%                    xy_RU - nx2 vector with [x y]-values for the right (R)
%                            upper (U) branch of the hyperbola
%                    xy_RL - nx2 vector with [x y]-values for the right (R)
%                            lower (L) branch of the hyperbola
%
% Input parameters: x_max - Maximum x-value used for the calculations [m]
%                   a     - Major semi-axis of the hyperbola [m]
%                   b     - Minor semi-axis of the hyperbola [m]
%
% Output files: none
%
% Input files: none
%
% External functions: none
%
% Last modified: 17.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

n=1000;
x_min=a;
dx=(x_max-x_min)./n;
x=x_min:dx:x_max;

% Check if the hyperbola can be calculated
if (a==0)&&(b==0)
    disp(' ');
    disp('ERROR!! The system will have an infinite number of solutions when a=b=0.');
    disp(' ');
    xy_LU=NaN;
    xy_LL=NaN;
    xy_RU=NaN;
    xy_RL=NaN;
    return;
end

% Calculate the right upper branch of the hyperbola
if a==0
    y_min=0;
    y_max=x_max;
    dy=(y_max-y_min)./n;
    y=y_min:dy:y_max;
    x=zeros(1,n+1);
```

```
else
    y=b.*sqrt(x.^2./a.^2-1);
end

% Set values for both branches of the hyperbola
xy_LU=[-x'  y'];
xy_LL=[-x' -y'];
xy_RU=[ x'  y'];
xy_RL=[ x' -y'];
```

```
function [TDOA] = Compute_TDOA(sensor1, sensor2, emitter);
% Calculate the time difference(s) of arrival for given emitter and sensor positions
%
% [TDOA] = Compute_TDOA(sensor1, sensor2, emitter)
%
% Output parameters: TDOA - nx1 matrix specifying the time difference(s) of arrival [s]
%
% Input parameters: sensor1 - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2 - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   emitter - 1x3 matrix specifying the emitter xyz-position [m]
%
% Output files: none
%
% Input files: none
%
% External functions: none
%
% Last modified: 17.09.09
%
% ------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ------------------------------------

c=2.99792458e8; % Speed of light [m/s]

% Emitter position [m]
x_e=emitter(1);
y_e=emitter(2);
z_e=emitter(3);

% Sensor1 positions [m]
x_s1=sensor1(:,1);
y_s1=sensor1(:,2);
z_s1=sensor1(:,3);

% Sensor2 positions [m]
x_s2=sensor2(:,1);
y_s2=sensor2(:,2);
z_s2=sensor2(:,3);

% Calculate the TDOA [s]
d1=sqrt((x_s1-x_e).^2+(y_s1-y_e).^2+(z_s1-z_e).^2);
d2=sqrt((x_s2-x_e).^2+(y_s2-y_e).^2+(z_s2-z_e).^2);
TDOA=(d1-d2)./c;
```

```
function ScanPhase_Plot2D_EmitterPos_DiffSensorPos(sensor1, sensor2, emitter);
% Plot emitter positions in 2D for different sensor positions (emitter plane at z=0)
%
% ScanPhase_Plot2D_EmitterPos_DiffSensorPos(sensor1, sensor2, emitter)
%
% Output parameters: none
%
% Input parameters: sensor1 - nx3 vector describing n sensor 1 xyz-positions [m]
%                   sensor2 - nx3 vector describing n sensor 2 xyz-positions [m]
%                   emitter - 1x3 vector describing emitter xyz-position [m]
%
% Output files: none
%
% Input files: none
%
% External functions: Compute_ApertureAngle
%                     ScanPhase_Compute2D_EmitterPositions
%
% Last modified: 17.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Compute aperture angles
[ApertureAngle] = Compute_ApertureAngle(sensor1, sensor2, emitter);

figure;
grid on;
hold on;

[n_max, xx]=size(ApertureAngle);
for n=1:1:n_max
    % Compute possible emitter positions
    [x1_n y1_n x2_n y2_n]=ScanPhase_Compute2D_EmitterPositions(sensor1(n,:), ...
                                                sensor2(n,:), ApertureAngle(n));

    % Plot possible emitter positions
    plot(x1_n, y1_n, 'b.', 'MarkerSize', 3);
    plot(x2_n, y2_n, 'b.', 'MarkerSize', 3);

    % Plot sensor1 and sensor2 position
    plot(sensor1(n,1), sensor1(n,2), 'sr', 'MarkerFaceColor', 'r');
    plot(sensor2(n,1), sensor2(n,2), 'sg', 'MarkerFaceColor', 'g');
end

% Plot emitter position
plot(emitter(1), emitter(2),'ok', 'MarkerFaceColor', 'k');
```

```
function [x1 y1 x2 y2]=ScanPhase_Plot2D_EmitterPositions(sensor1, sensor2, ...
                                                         ApertureAngle);
% Plot emitter positions in 2D
%
% [x1 y1 x2 y2]=ScanPhase_Plot2D_EmitterPositions(sensor1, sensor2, ApertureAngle)
%
% Output parameters: x1 - vector with x-values for circle arc 1 [m]
%                    y1 - vector with y-values for circle arc 1 [m]
%                    x2 - vector with x-values for circle arc 2 [m]
%                    y2 - vector with y-values for circle arc 2 [m]
%
% Input parameters: sensor1       - 1x3 vector describing sensor 1 xyz-position [m]
%                   sensor2       - 1x3 vector describing sensor 2 xyz-position [m]
%                   ApertureAngle - Aperture angle [rad]
%
% Output files: none
%
% Input files: none
%
% External functions: ScanPhase_Compute2D_EmitterPositions
%
% Last modified: 17.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

% Calculate possible emitter positions
[x1 y1 x2 y2]=ScanPhase_Compute2D_EmitterPositions(sensor1, sensor2, ApertureAngle);

% Plot possible emitter positions
figure;
hold on;
plot(x1, y1, 'b.', 'MarkerSize', 3);
plot(x2, y2, 'b.', 'MarkerSize', 3);
```

```
function [x1 y1 x2 y2]=ScanPhase_Compute2D_EmitterPositions(sensor1, sensor2, ...
                                                            ApertureAngle);
% Compute emitter positions in 2D
%
% [x1 y1 x2 y2]=ScanPhase_Compute2D_EmitterPositions(sensor1, sensor2, ApertureAngle)
%
% Output parameters: x1 - vector with x-values for circle arc 1 [m]
%                    y1 - vector with y-values for circle arc 1 [m]
%                    x2 - vector with x-values for circle arc 2 [m]
%                    y2 - vector with y-values for circle arc 2 [m]
%
% Input parameters: sensor1       - 1x3 vector describing sensor 1 xyz-position [m]
%                   sensor2       - 1x3 vector describing sensor 2 xyz-position [m]
%                   ApertureAngle - Aperture angle [rad]
%
% Output files: none
%
% Input files: none
%
% External functions: ComputeCircle
%
% Last modified: 17.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

% Check if Aperture Angle is within allowed limits [0 pi]
if (ApertureAngle>pi) || (ApertureAngle<0)
    disp(' ');
    disp('ERROR!! Aperture Angle must be between 0 and pi.');
    disp(' ');
    x1=NaN;
    y1=NaN;
    x2=NaN;
    y2=NaN;
    return;
end

% Coordinates for sensor 1 (x_s1,y_s1) and sensor 2 (x_s2,y_s2)
x_s1=sensor1(1);
y_s1=sensor1(2);
x_s2=sensor2(1);
y_s2=sensor2(2);

% Check for the special cases when the Aperture Angle is equal to 0 or 180 deg.
[x1 y1 x2 y2 state]=CheckSpecialCases(x_s1, y_s1, x_s2, y_s2, ApertureAngle);
if state==1
    return;
end

% Calculate the distance (d_12) between the sensors
d_12=sqrt((x_s2-x_s1).^2+(y_s2-y_s1).^2);
```

```
% Calculate the parameter a
a=d_12./(2.*tan(ApertureAngle));

% Find the center coordinates (x0,y0) for the sensors
x0=(x_s1+x_s2)./2;
y0=(y_s1+y_s2)./2;

% Calculate the angle phi
phi=atan2(-(y_s2-y0),(x_s2-x0));

% Find the center coordinates (x_c1, y_c1) for circle 1
x_c1=x0-a.*sin(phi);
y_c1=y0-a.*cos(phi);

% Find the center coordinates (x_c2, y_c2) for circle 2
x_c2=x0+a.*sin(phi);
y_c2=y0+a.*cos(phi);

% Calculate the radius R of the circle
R=d_12./(2.*sin(ApertureAngle));

% Compute circle arc 1
[x1 y1]=ComputeCircleArc(x_c1, y_c1, R, x_s1, y_s1, x_s2, y_s2, ApertureAngle);

% Compute circle arc 2
[x2 y2]=ComputeCircleArc(x_c2, y_c2, R, x_s1, y_s1, x_s2, y_s2, ApertureAngle);



%%%%%%%%%%%%%%%%%%% INTERNAL FUNCTIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x_arc y_arc]=ComputeCircleArc(x_c, y_c, R, x_s1, y_s1, x_s2, y_s2, AA);
% Find arc of the circle with center (x_c,y_c) and radius R
% that corresponds to given sensor positions (x_s1,y_s1) and (x_s2,y_s2)
% and aperture angle (AA). The sensor positions lie on the circle.

% Calculate the circle with center (x_c,y_c) and radius R
[x y]=ComputeCircle(x_c, y_c, R);

% Calculate the angle theta [rad] around the circle
theta=atan2(y-y_c,x-x_c);
index=find(theta<0);
theta(index)=theta(index)+2.*pi;

% Calculate angle theta_s1 [rad] for sensor position (x_s1,x_s1)
theta_s1=atan2(y_s1-y_c,x_s1-x_c);
if theta_s1<0
    theta_s1=theta_s1+2.*pi;
end

% Calculate angle theta_s2 [rad] for sensor position (x_s2,x_s2)
theta_s2=atan2(y_s2-y_c,x_s2-x_c);
if theta_s2<0
    theta_s2=theta_s2+2.*pi;
end
```

```
% Find index in theta array that corresponds to theta_s1 and theta_s2
[xx,index_s1]=min(abs(theta-theta_s1));
[xx,index_s2]=min(abs(theta-theta_s2));

% Find max index in theta array + index_low and index_high
[xx, index_max]=size(theta);
index_low = min(index_s1,index_s2);
index_high=max(index_s1,index_s2);

D1=index_high-index_low;
D2=index_max-D1;

% Select x-values and y-values from array according to the appropriate
% indices (depends upon the aperture angle (AA))
if AA==pi./2
    x_arc=x;
    y_arc=y;
elseif ((AA<pi./2)&&(D1>=D2)) || ((AA>pi./2)&&(D1<D2))
    x_arc=x(index_low:index_high);
    y_arc=y(index_low:index_high);
else
    x_arc(1:index_low)     =x(1:index_low);
    x_arc(index_low+1:D2+1)=x(index_high:index_max);
    y_arc(1:index_low)     =y(1:index_low);
    y_arc(index_low+1:D2+1)=y(index_high:index_max);
end


function [x1 y1 x2 y2 state]=CheckSpecialCases(x_s1, y_s1, x_s2, y_s2, AA);
% Check and set values for the special cases when the Aperture Angle (AA)
% is equal to 0 or pi.
state=0;
n=100;
k=(y_s2-y_s1)./(x_s2-x_s1);
if AA==0
    % Set values when the Aperture Angle (AA) is equal to 0.
    if x_s1==x_s2
        Dy=10.*abs(y_s2-y_s1);
        y_min=min(y_s1,y_s2)-Dy;
        y_max=max(y_s1,y_s2)+Dy;
        delta_y=Dy./n;
        y1=min(y_s1,y_s2):-delta_y:y_min;
        x1=y1;
        x1=min(x_s1,x_s2);
        y2=max(y_s1,y_s2):delta_y:y_max;
        x2=y2;
        x2=max(x_s1,x_s2);
    else
        if abs(k)<=1
            Dx=10.*abs(x_s2-x_s1);
            x_min=min(x_s1,x_s2)-Dx;
            x_max=max(x_s1,x_s2)+Dx;
            delta_x=Dx./n;
            x1=min(x_s1,x_s2):-delta_x:x_min;
            y1=y_s1+k.*(x1-x_s1);
            x2=max(x_s1,x_s2):delta_x:x_max;
```

```
            y2=y_s2+k.*(x2-x_s2);
        else
            Dy=10.*abs(y_s2-y_s1);
            y_min=min(y_s1,y_s2)-Dy;
            y_max=max(y_s1,y_s2)+Dy;
            delta_y=Dy./n;
            y1=min(y_s1,y_s2):-delta_y:y_min;
            x1=x_s1+k.^(-1).*(y1-y_s1);
            y2=max(y_s1,y_s2):delta_y:y_max;
            x2=x_s2+k.^(-1).*(y2-y_s2);
        end
    end
    state=1;
elseif AA==pi
    % Set values when the Aperture Angle (AA) is equal to pi.
    if abs(k)<=1
        delta_x=(x_s2-x_s1)./n;
        x1=x_s1:delta_x:x_s2;
        y1=y_s1+k.*(x1-x_s1);
        x2=x1;
        y2=y1;
    else
        delta_y=(y_s2-y_s1)./n;
        y1=y_s1:delta_y:y_s2;
        x1=x_s1+k.^(-1).*(y1-y_s1);
        x2=x1;
        y2=y1;
    end
    state=1;
else
    x1=NaN;
    y1=NaN;
    x2=NaN;
    y2=NaN;
end
```

```
function [x y]=ComputeCircle(x_c, y_c, R);
% Compute circle
%
% [x y]=ComputeCircle(x_c, y_c, R)
%
% Output parameters: x - vector with x-values for the circle [m]
%                    y - vector with y-values for the circle [m]
%
%
% Input parameters: x_c - x-coordinate for the center of the circle [m]
%                   y_c - y-coordinate for the center of the circle [m]
%                   R   - Radius of the circle [m]
%
% Output files: none
%
% Input files: none
%
% External functions: none
%
% Last modified: 17.09.09
%
% --------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% --------------------------------------

theta_min=0;
theta_max=2.*pi;
delta_theta=0.1.*pi./180;    % Steps of 0.1 deg
theta=theta_min:delta_theta:theta_max;

% Compute x- and y-values for the circle
x=x_c+R.*cos(theta);
y=y_c+R.*sin(theta);
```

```
function [ApertureAngle] = Compute_ApertureAngle(sensor1, sensor2, emitter);
% Calculate the aperture angle(s) for given emitter and sensor positions
%
% [ApertureAngle] = Compute_ApertureAngle(sensor1, sensor2, emitter)
%
% Output parameters: ApertureAngle - nx1 matrix specifying the aperture angle(s)
%                                    (on the interval [0 pi]) [rad]
%
% Input parameters: sensor1 - nx3 matrix specifying n sensor1 xyz-position(s) [m]
%                   sensor2 - nx3 matrix specifying n sensor2 xyz-position(s) [m]
%                   emitter - 1x3 matrix specifying the emitter xyz-position [m]
%
% Output files: none
%
% Input files: none
%
% External functions: none
%
% Last modified: 17.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

c=2.99792458e8; % Speed of light [m/s]

% Emitter position [m]
x_e=emitter(1);
y_e=emitter(2);
z_e=emitter(3);

% Sensor1 positions [m]
x_s1=sensor1(:,1);
y_s1=sensor1(:,2);
z_s1=sensor1(:,3);

% Sensor2 positions [m]
x_s2=sensor2(:,1);
y_s2=sensor2(:,2);
z_s2=sensor2(:,3);

% Calculate the Aperture Angle [rad]
d12=sqrt((x_s1-x_s2).^2+(y_s1-y_s2).^2);
d1 =sqrt((x_s1-x_e).^2 +(y_s1-y_e).^2);
d2 =sqrt((x_s2-x_e).^2+(y_s2-y_e).^2);
ApertureAngle=acos((d12.^2-d1.^2-d2.^2)./(-2.*d1.*d2));
```

```
function AOA_Plot2D_EmitterPos_DiffSensorPos(sensor, emitter);
% Plot emitter positions in 2D for different sensor positions (emitter plane at z=0)
%
% AOA_Plot2D_EmitterPos_DiffSensorPos(sensor, emitter)
%
% Output parameters: none
%
% Input parameters: sensor  - nx3 vector describing n sensor xyz-positions [m]
%                   emitter - 1x3 vector describing emitter xyz-position [m]
%
% Output files: none
%
% Input files: none
%
% External functions: Compute_AOA
%                     AOA_Compute2D_EmitterPositions
%
% Last modified: 17.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Compute angles of arrival [rad]
[AOA] = Compute_AOA(sensor, emitter);
index=find(AOA<0);
AOA(index)=AOA(index)+2.*pi;

figure;
grid on;
hold on;

[n_max, xx]=size(AOA);
for n=1:1:n_max
    % Compute possible emitter positions
    [x_n y_n]=AOA_Compute2D_EmitterPositions(sensor(n,:), AOA(n));

    % Plot possible emitter positions
    plot(x_n, y_n, 'b-');

    % Plot sensor position
    plot(sensor(n,1), sensor(n,2), 'sr', 'MarkerFaceColor', 'r');
end

% Plot emitter position
plot(emitter(1), emitter(2),'ok', 'MarkerFaceColor', 'k');
```

```
function [x y]=AOA_Plot2D_EmitterPositions(sensor, AOA);
% Plot emitter positions in 2D
%
% [x y]=AOA_Compute2D_EmitterPositions(sensor, AOA)
%
% Output parameters: x - vector with x-values [m]
%                    y - vector with y-values [m]
%
% Input parameters: sensor - 1x3 vector describing sensor xyz-position [m]
%                   AOA    - Angle of arrival [rad]
%
% Output files: none
%
% Input files: none
%
% External functions: AOA_Compute2D_EmitterPositions
%
% Last modified: 17.09.09
%
% ---------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% ---------------------------------------

% Calculate possible emitter positions
[x y]=AOA_Compute2D_EmitterPositions(sensor, AOA);

% Plot possible emitter positions
figure;
plot(x, y, 'b-');
```

```
function [x y]=AOA_Compute2D_EmitterPositions(sensor, AOA);
% Compute emitter positions in 2D
%
% [x y]=AOA_Compute2D_EmitterPositions(sensor, AOA)
%
% Output parameters: x - vector with x-values [m]
%                    y - vector with y-values [m]
%
% Input parameters: sensor - 1x3 vector describing sensor xyz-position [m]
%                   AOA    - Angle of arrival [rad]
%
% Output files: none
%
% Input files: none
%
% External functions: none
%
% Last modified: 17.09.09
%
% ------------------------------------
% Author: Gudrun Kristine Høye
%         Forsvarets Forskningsinstitutt
%         Norway
% ------------------------------------

n=100;
limit=1e5;
x_s=sensor(1);
y_s=sensor(2);

% Check if AOA is within allowed limit [0 2pi]
if (AOA<0) || (AOA>2.*pi)
    disp(' ');
    disp('ERROR!! Angle of arrival must be between 0 and 2pi.');
    disp(' ');
    x=NaN;
    y=NaN;
    return;
end

% Calculate possible emitter positions
if ((AOA>=pi./4)&&(AOA<=3.*pi./4)) || ((AOA>=5.*pi./4)&&(AOA<=7.*pi./4))
    if (AOA>=pi./4)&&(AOA<=3.*pi./4)
        Dx=limit;
    else
        Dx=-limit;
    end
    k_y=cos(AOA)./sin(AOA);
    x_lim=x_s+Dx;
    delta_x=(x_lim-x_s)./n;
    x=x_s:delta_x:x_lim;
    y=y_s+k_y.*(x-x_s);
else
    if (AOA>7.*pi./4)||(AOA<pi./4)
        Dy=limit;
    else
```

```
        Dy=-limit;
    end
    k_x=sin(AOA)./cos(AOA);
    y_lim=y_s+Dy;
    delta_y=(y_lim-y_s)./n;
    y=y_s:delta_y:y_lim;
    x=x_s+k_x.*(y-y_s);
end
```

```
function [AOA] = Compute_AOA(sensor, emitter);
% Calculate the angle(s) of arrival for given emitter and sensor position(s)
%
% [AOA] = Compute_AOA(sensor, emitter)
%
% Output parameters: AOA - nx1 matrix specifying the angle(s) of arrival
%                          (on the interval <-pi pi]) [rad]
%
% Input parameters: sensor  - nx3 matrix specifying n sensor xyz-position(s) [m]
%                   emitter - 1x3 matrix specifying the emitter xyz-position [m]
%
% Output files: none
%
% Input files: none
%
% External functions: none
%
% Last modified: 17.09.09
%
% ------------------------------------
% Author: Gudrun Kristine Høye, PhD
%         Forsvarets Forskningsinstitutt
%         Norway
% ------------------------------------

% Sensor positions [m]
x_s=sensor(:,1);
y_s=sensor(:,2);

% Emitter position [m]
x_e=emitter(1);
y_e=emitter(2);

x=x_e-x_s;
y=y_e-y_s;

% Calculate angle of arrival [rad]
AOA=atan2(x,y);
```

References

[1] E O Løvli (2005): Geolocalization with two airborne ESM-sensors, Master Thesis, NTNU, 2005.

[2] J T Gillis (1991): Computation of the circular error probability integral, IEEE Transactions on Aerospace and Electronic Systems, 27 (Nov. 1991), 906-910.

[3] J T Gillis (1993): Comments on "Computation of the circular error probability integral", IEEE Transactions on Aerospace and Electronic Systems, Vol. 29, No. 2 (April 1993), 553-555.

[4] P J Davis and P Rabinowitz (1975): Methods of Numerical Integration, New York: Academic Press, 1975, 40 and 241-243.

[5] T Smestad (2000): Metoder og dataprogram for analyser av posisjonering med ESM-sensorer, FFI/NOTAT-2000/02397.

[6] Wikipedia.

[7] S Barnett and T M Cronin (1986): Matemathical formulae, 4th edition, Longman Group Limited.

[8] E O Løvli and T Smestad (2006): (U) Geolokalisering med tidsdifferense mellom to fly – analyser og dataprogram, FFI/RAPPORT-2006/01307, BEGRENSET.

[9] T Smestad et al (2005): (U) Reduksjon av kommunikasjon i TDOA-system, FFI/NOTAT-2005/04034, BEGRENSET.

[10] P H Andersen (2005): En generell algoritme for stedfesting nær jordoverflaten ved hjelp av peilinger, FFI/NOTAT-2005/03979.

[11] P H Andersen (2005): En generell algoritme for stedfesting nær jordoverflaten ved hjelp av tidsdifferansemålinger, FFI/NOTAT-2005/03980.

[12] A Melkevik (2001): (U) ULA – Automatic target motion analyses with multisensor integration. Presentation of A-MSI filter version 2, FFI/RAPPORT-2001/01142, BEGRENSET.

[13] M Kirkeby og K Mygland (2009): TMA-algoritmen for ESM, FFI, 7. aug 2009.

[14] T Smestad, J Hammerstad, S Hauge, K Lie, E B Nilssen, G Skille, R Sundgot (2009): "Statusrapport for 1037 Geolokalisering og identifikasjon fra fly (GEOIDE), FFI-rapport 2009/00777, BEGRENSET.

[15] L Grønvold og S Oftebro (2006): "LabView program for lokalisering av båter med tidsinformasjon fra to sensorer", Hovedprosjekt Elektro, Høgskolen i Oslo, 2006.