Hardware Oriented Self Localization Method Using NLOS Signal of Single Base Station

Abdul Bahadur Khan

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Technology
in
Information Technology

Eastern Mediterranean University
February 2024
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master in Information Technology.

Assoc. Prof. Dr. Ece Çelik
Director, School of Computing and Technology

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master in Information Technology.

Asst. Prof. Dr. Cem Yağlı
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Emre Özen _____

2. Asst. Prof. Dr. Öykü Akaydın _____

3. Asst. Prof. Dr. Cem Yağlı _____

ABSTRACT

Self-localization of Mobile object is the process of finding the location of mobile object using wireless network. It can be found in many ways like trilateration which found the location of mobile object using three base stations. Mobile object (MO) location can also be found using approach time of arrival (TOA). While researchers were founding the location of mobile object they has to compromise weather on the accuracy or resources because it's quite difficult to have the best results using minimum amount of resources.

This thesis will explain how I found the location of mobile object? It will explain all the techniques and approaches like static and dynamic CORDIC algorithm, Line of sight (LOS), Non Line of Sight (NLOS), Angle of arrival (AOA), Angle of departure (AOD) and Time difference of arrival (TDOA). Using all the above approaches we have found the location of mobile object using minimum amount of resources with maximum accuracy and the simulation results are also mentioned in this thesis.

The objective of this research is to achieve best results with minimum resources and this is done with static and dynamic CORDIC algorithm. CORDIC algorithm is used for real time mobile object localization which uses add -shift technique. CORDIC algorithm also compute trigonometric, logarithmic, division and square root calculations to locate MO which is faster and easier.

To accomplish the objective of this research, a system is designed which shows that mobile object location can be found in 3D space using the concept of wireless cellular

network. The CORDIC algorithm is changed to static and dynamic to achieve the maximum accuracy using one base station in 3D space.

Keywords: Mobile Object, Base Station, Cordic, Line of Sight, Non-line of Sight

ÖZ

Mobil nesnenin kendi kendine yerelleştirilmesi, kablosuz ağ kullanılarak mobil nesnenin konumunun bulunması işlemidir. Üç baz istasyonu kullanarak mobil nesnenin konumunu bulan trilaterasyon gibi birçok yolla bulunabilir. Mobil nesnenin konumu, varış zamanı yaklaşımı kullanılarak da bulunabilir. Mobil nesnelerin yerini bulurken, araştırmacıların doğruluk veya kaynaklar konusunda hava koşullarından ödün vermesi gerekir çünkü minimum miktarda kaynak kullanarak en iyi sonuçları elde etmek oldukça zordur.

Bu tez, hareketli nesnenin konumunu nasıl bulduğumu açıklayacaktır. Statik ve dinamik algoritması, Görüş Hattı Görüş Hattı Olmayan, Varış Açısı, Kalkış Açısı ve Varış Zaman Farkı gibi tüm teknik ve yaklaşımları açıklayacaktır. Yukarıdaki yaklaşımların tümünü kullanarak, minimum miktarda kaynak kullanarak maksimum doğrulukla hareketli nesnenin konumunu bulduk ve simülasyon sonuçlarından da bu tezde bahsedildi.

Anahtar Kelimeler: Mobil Nesnenin Konumu, Varış Saati, Kalkış Saati, Mobil Nesne, Baz Istasyonu, Görüş Hattı, Görüş Hattı Olmayan

DEDICATION

To My Family

ACKNOWLEDGMENT

I am highly obligated to my supervisor Assoc. Prof. Dr. Cem Yagli for his supervision without his own supervision, advising and unbelievable hard work and achievement of this software would have been in doubt. I am thankful to the Assoc. Prof. Dr. Cem Yagli for their hard work, experience and continuous support in this project. I also want to acknowledge my family members who have been a continuous sources of inspiration for me and bring me the value of honor & endless hard working.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	2 Dimension
3D	3 Dimension
AOA	Angle of Arrival
AOD	Angle of Departure
BC	Bahadur-Cem
BS	Base Station
CORDIC	Coordinate Rotation Digital Computer
DARM	Dynamic angle rotation method
FARM	Fixed angle rotation method
LOS	Line of Sight
MS	Mobile Object
NLOS	Non Line of Sight
TDOA	Time Difference of Arrival
TOA	Time of Arrival

Chapter 1

INTRODUCTION

1.1 Introduction

Positioning of Mobile Object in wireless cellular network has become very interesting and popular in now day's technological life. Location of mobile object is found in different ways like trilateration in 3D positioning using three different base stations which are placed on different locations that are already known using the signaling approach that is time of arrival (TOA). This technique is used in Navigation, guidance, tracking and finding has also added value in finding the mobile location in 2D and 3D space. Positioning of mobile object also plays important role in finding the location of different objects in our daily life like, trackers, lost and found of objects at home and electronic bracelets.

The main contribution of this thesis is to convert the 2D in to 3D, which means that to locate mobile object (MO) location of in 3 dimension (3D) space through one base station. This developed positioning technique uses static and dynamic coordinate rotation digital computer (CORDIC) algorithm to achieve maximum accuracy with minimum amount of resources.

Many techniques are applied to localize the MO where the researchers have to compromise on results or resources because it is quite difficult to achieve the best results with minimum amount of resources in shape of computational cost and energy.

The objective of this research is to achieve best results with minimum resources and this is done with static and dynamic CORDIC algorithm. CORDIC algorithm is used for real time mobile object localization which uses add -shift technique. CORDIC algorithm also compute trigonometric, logarithmic, division and square root calculations to localize the MO which is faster as well as easier.

To objective of this research, a system is designed which shows that mobile object location can be found in 3D space using the concept of wireless cellular network. The CORDIC algorithm is changed to static and dynamic to achieve the maximum accuracy using one base station in 3D space.

1.2 Navigation Through Stars

In the early stages of life people used to navigate through stars, sun and moon. Celestial Navigation, Polaris (North Star) and Star Charts and Constellations was used to determine the position on land and sea, while Timekeeping concept was used to estimate the time and was used for long distance journey. This technique was facing the complication when the weather is cloudy and rainy because they were unable to determine the position of stars, sun and moon.

1.3 Magnetic Compass

Magnetic compass was invented in 13th century. It was initially used when the weather was cloudy and rainy and the people were unable to find the North Star. The results were also not accurate as they were not sure that it pointing to North Pole or not.

1.4 Determination of Latitude Through Sextant

In the early stages of Navigation one of the most challenging task was to find the latitude. It was quite difficult for them to find the latitude due to the strenuous energizing humidity. In 1484 Martin Benaim a German cartographer discovered

Sextant device which was used above the horizon of stars and stars to measure the angle and to determine latitude.

1.5 Determination of Longitude Using Chronometer

In 1764 John Harrison discovered chronometer to find the longitude. Harrison's chronometer was used by James Cook to circumvent the Globe. The calculations of longitude in 1779 were proven all right within 8 miles. Charts were rapidly developed around the world. James Cook charts were used for 200 years. He also won British prize.

1.6 Radio Waves

In 1935 Robert Watson-Watt a British physicist produced First radar. An object's range, position in space, size, form, velocity, and direction of motion may all be ascertained using radar. Using radio waves. Radar can also control air traffic, detecting weather and tracking spacecraft.

1.7 Rise of Global Positioning System (GPS)

GPS was initiated in 1973. It is maintained and Operated by the U.S. Department of Defense. GPS was launched into space in 1978. In start 11 satellites were in used now its 32. Accuracy is within 16 feet's. Many people came to the conclusion that GPS is any satellite orbiting around the world bus it is the one that is operating by the US. Other countries have their own satellite systems like European Union is using a satellite system that is called Galileo where as China is operating BeiDou Navigation Satellite System. Russia has its own satellite system called GLONASS. There are many more Navigation systems that are operated by a specific country or region.

Chapter 2

MODEL

2.1 Introduction

Location estimation of MO in 2 dimension space and 3 dimension space got more attention now a days in technological life as cellular and wireless communication is advancing day by day. People are using mobile phones too much as it is providing us too much facilities like, phone calls, internet which can be accessible everywhere without any barrier. In the same way Mobile applications are very useful and people are in touch with them every day. Smartphones are very powerful devices with different number of sensors that shares real-time geolocation information, advertisement, entertainment and security services which is the most important aspect. Many research and studies are already done on the location estimation of MO and they are supported by the governments [6] [7]. The MO can be find in different ways as it was found by Global Positioning System (GPS) in the 1978 which was initiated in 1973 by US government. In first GPS was using 11 satellites in start but now a days they are using 32 satellites. Basically GPS geolocation orbits around the earth for the purpose of communication between satellites that send and receives the data from earth. The accuracy for GPS is 16 feet which is very low as compared to latest technologies and advancements and now Bluetooth and Wi-Fi technologies can be used for identifying the geographic position mostly indoor. Positioning through Wi-Fi is also known as WPS and WiPS, which is based on wireless access point and hotspot

of Wi-Fi. The common technique for location estimation of Wi-Fi is the measurement of receiving signal strength.

2.2 Techniques Used in Location Estimation

There are many techniques developed by the researchers to find the location of MO. Here I am discussing some of them which are very useful to find the location of MO.

1. Location positioning through Cellular Networks.
2. Use of Radio Frequency Identification in Location Estimation.

2.2.1 Location Positioning Through Cellular Networks

Positioning through cellular network is also one of the most used technique for the location estimation of mobile objects, in which radio waves are transmitted by cell tower which are used for the communication of MO. But as compared to GPS this method is not that much accurate.

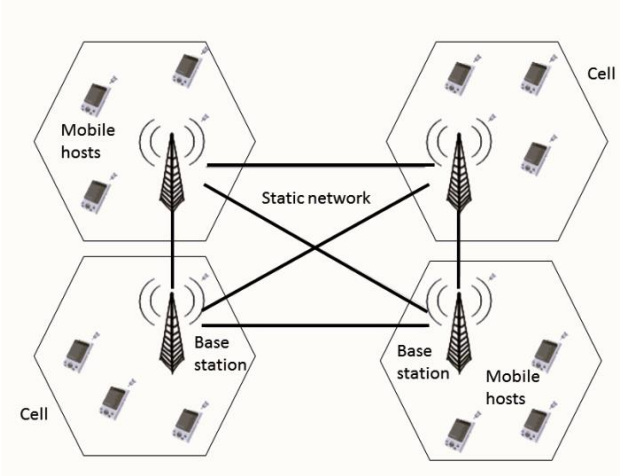

Figure 2.1: Cellular Communication

Cellular networks are mostly in operation in mobile phones for the purpose of text messages, internet connectivity and voice calls. It is often used by Internet of things (IOT) for control and remote monitoring.

2.2.2 Use of Radio Frequency Identification in Location Estimation

RFID is the combination of different effectively different methods. There is a static location in the RFID scanner. When the other networks ping off the scanner can be logged. When the Scanner of RFID is activated, when it records the access the location can be tagged. So, the location is identified by the scanner through device accessing.

2.3 Different Approaches in Localization

In finding the location of MO through BS there are different approaches that helps us to identify the location of MO. Like angle of departure (AOD), Basically AOD is the angle which is generated at BS when the signal is sent to the MO from BS. We now the AOD from base station which gives us the angle which aodxy, we can get the rotation angle $\sigma = \arcsin^{2-k}$ where 4<k>11 where k depends on the accuracy that is needed by increasing and decreasing the k it will affect the level of accuracy.

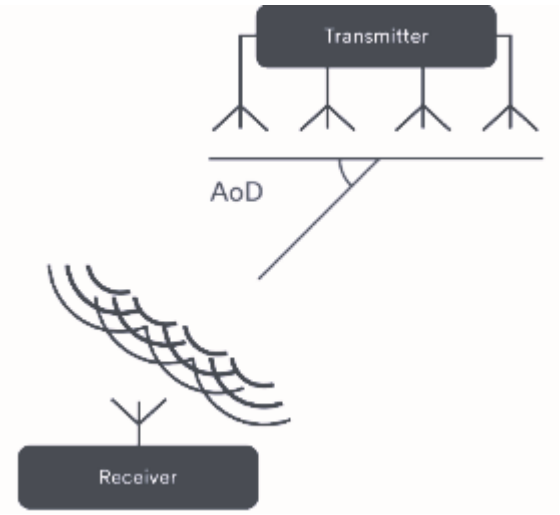

Figure 2.2: Angle of Departure

In the same way angle of arrival (AOA) at MO is also known which is $aoa_{xy.}$ AOA is calculated on the side of MO. As we can see in figure 2.3 the AOA is calculated once the signal reaches the MO.

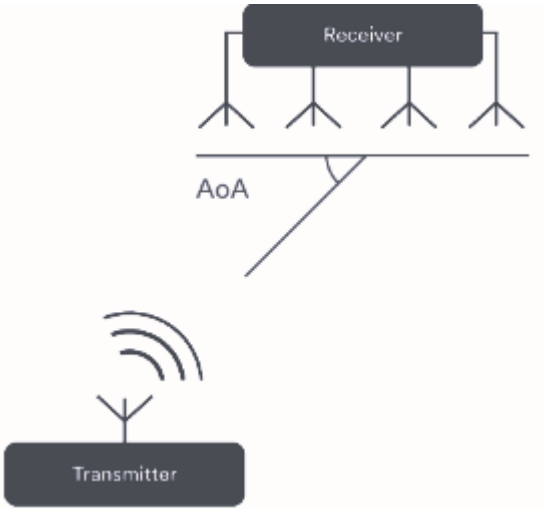

Figure 2.3: Angle of Arrival

The signal are received at MO in two different ways which plays important role in location estimation they are:

1. Line of Sight (LOS).
2. Non Line of Sight (NLOS).

2.3.1 Line of Sight (LOS)

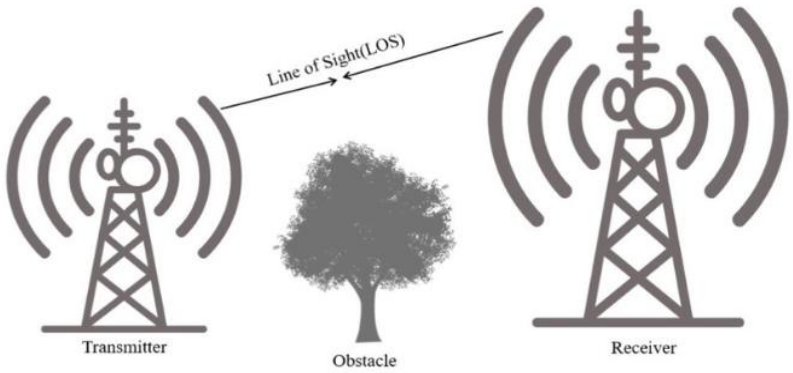

Figure 2.4: Line of sight

In this case the signals are received directly by the MO from BS without colliding with any obstacle in the way. We can say that the signal is traveled from BS to MO in the

shape of a straight line. As we can see in figure 2.4 the signal is approaching MO from BS in straight line. Using the technique of AOA and AOD explained in section 2.3 the location of MO is found.

2.3.2 Non Line of Sight (NLOS)

In NLOS the transmitted signal is reaching the MO after colliding with any obstacle in the way.

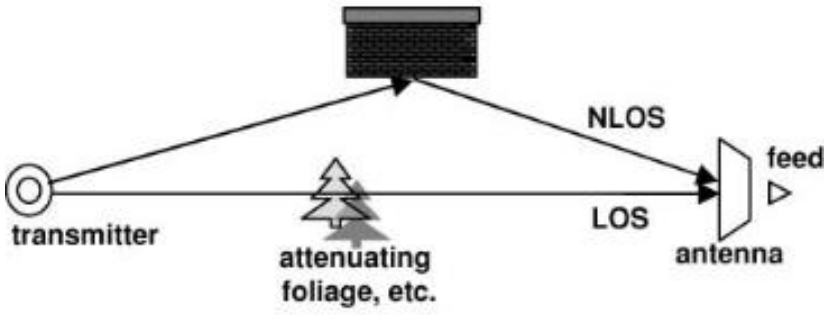

Figure 2.5: Non Line of Sight

As we can see in figure 2.5 the signal cannot reach the MO directly as there is an obstacle in the way. The transmitted signal is bounced with the object and then reached the MO, which is the NLOS case of this study in 2D.

2.4 The Coordinate Rotation Digital Computer (CORDIC)

In 1956 Jack E. Volder developed CORDIC algorithm [1] in the department of aero electronics of Convair in order to replace analog resolver in navigation computer of B-58 bomber with more precise and faster real time digital solution. Therefore, CORDIC is also known as digital resolver. His research and study proposed the CORDIC algorithm to solve the cosine and sine functions. His results also mention to execute the exponential functions, hyperbolic coordinate rotations and algorithms with revised CORDIC algorithm. In 1971 John Stephen Walther modify the CORDIC algorithm to

calculate square roots, divisions, multiplications, natural algorithm, hyperbolic functions and natural exponentials. Originally, CORDIC is designed to implement only binary numerals which are zeros and ones. CORDIC is using simple add and shift operations for different computing functions like, algorithmic and hyperbolic function, complex and real multiplications, square root calculations, trigonometric calculations, divisions and linear system solutions. CORDIC is also used in image processing and signal, robotics, 3D graphics and communication system.

2.4.1 Rotating and Lengthening Mode of CORDIC

In CORDIC algorithm the rotation is achieved by two different ways. CORDIC is a repetitive algorithm which is used to measure the rotations of 2D vectors in circular hyperbolic and linear coordinates system using shift and add operations. Its use is in the field of image processing, signal processing, filtering and algebra matrix etc. The first one is by rotating the input vector by different small angles θd which is equal to the sum of all rotated angles, θi i=0, 1… n. In this way the total rotated angle can be calculated. The second part is vectoring mode, where the vector(x, y) is rotated by an angle θ to get a new vector (x', y'). In this method the length R and the angle α toward x-axis are calculated. In this method the vector is continuously rotated with x-axis so that the y-axis is moved to zero. The sum of all rotated angles is equal to α, and the value of x-axis is equal to length R of vector (x, y).

Chapter 3

CONDUCTING A STUDY TO EXPLORE AN ALTERNATIVE SOLUTION

3.1 Introduction

In this solution we found the location of MO using one BS and different scatters using the concept of LOS and NLOS. For LOS there can be only one AOD but for NLOS there can be multiple AODs. As the location of BS and scatters are known but the location of MO is unknown so using the concept of AOD from BS and AOA at MO we have found the MO using both LOS and NLOS as BS and MO has antenna arrays to generate and receive the signals. In real life signals have multiple movements, they can approach to the targeted object in many ways like it can be directly, indirectly (scattered with one object or many object). The signals also gets weaker and weaker with increasing distance and by scattering with multiple object. So here we consider only directly received signals for LOS and single scattered signals for NLOS.

3.2 Geometric Model

Different components of this model are mentioned in the Figure 2.1. These components are one BS one MO and different scatters. As the location of BS is known but the location of MO and scatter is unknown. Using the concept of angle of departure AOD from BS and angle of arrival AOA on the other hand at MO the estimated position of scatters and MO has been found as mentioned in Figure 3.1

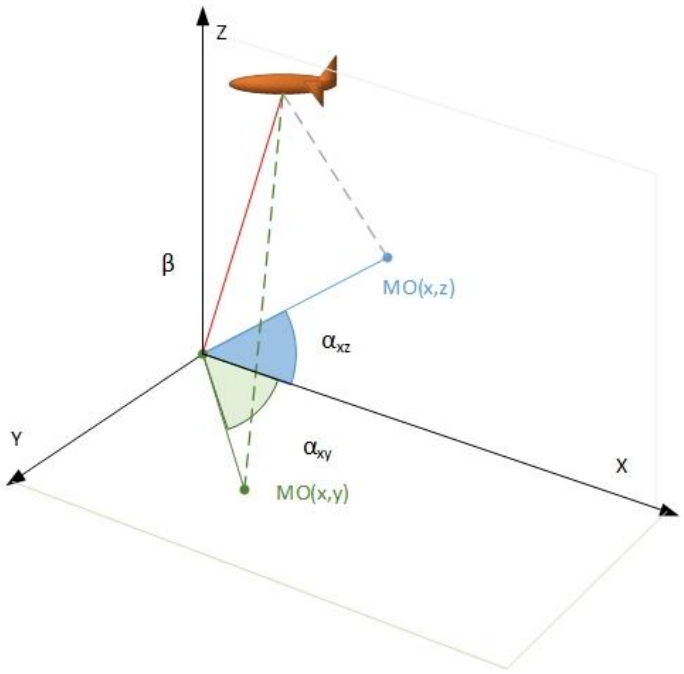

Figure 3.1: The Mobile Object in 3D Space

In this study, it is aimed to be found a new, accurate and computationally cost-effective self-localization technique that will serve to MOs in 3D Space. For simplifying the self-localization process, it will be calculated first in 2D Space (on the X-Y plane). Then the solution will be carried to the 3D space (with calculating its position on the X-Z plane). Figure 3. 1 depicts the 2D space projections on the X-Y and X-Z panels of a moving object traveling in 3D space.

On this model, from the time past between transmitting and receiving of the signal, its total travelling distance (Dl) can be calculated. Also, via the antenna arrays on the BS and MO, (AOD=α) which is the angle of departure and angle of arrival is (AOA=β) can be measured. The positions of the scatters and the mobile object is unknown. Through the known values Dl, AOD and AOA, the possible locations of scatters may be on the line OS and Mobile Object may be on the line SM accordingly. The OSM is an isosceles triangle hence OS=ON=dl.

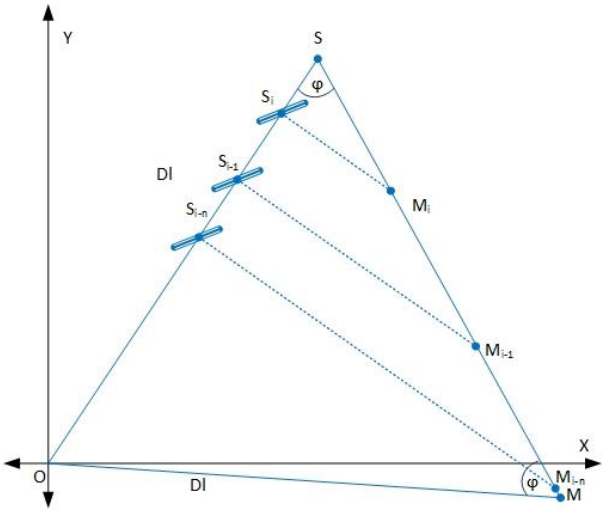

Figure 3.2: Possible Positions of Scatters and the Mobile Object in X-Y Plane

There are two interesting points are existing on the SM line, where on the point S both scatter and MO is sharing the same coordinate. This is the first point where the transmitted signal is reaching the MO directly (LOS case). The other LOS case is where the MO is on the point M. On the other hand, the transmitted signal can reach the MO on SM line through scattering the object which is Non line of sight (NLOS) way to reach the MO.

3.3. Mathematical Calculation

As we can see in figure 3.3. We have two triangles in our model where the intersection point of the two triangle will be expected location of the MO. We know the angle of departure for first transmitted signal on X-Y plan which is aod_{1xy} and it explains the model in 2D space to convert it into 3D space we have X-Z plane where the angle of departure is aod_{1xz} which converts the model to 3D space. On the side of MO the arriving angle on X-Y plan is aoa_{1xy} and on X-Z plan the arriving angle is $aoa_{1xz.}$ We also generated the second signal which is bouncing on SC2 and intersecting the first triangle on one point which is the expected location of the mobile object for NLOS

case. For second triangle the angle of departure on X-Y plan is aod_{2xy} and on X-Z plan the angle of departure is aod_{2xz}. Once the signal is bounced and reach the MO so the angle of arrival on X-Y plan is aoa_{2xy}, and the angle of arrival on X-Z plan is aoa_{2xz}. Using all the known parameters we found the location of MO.

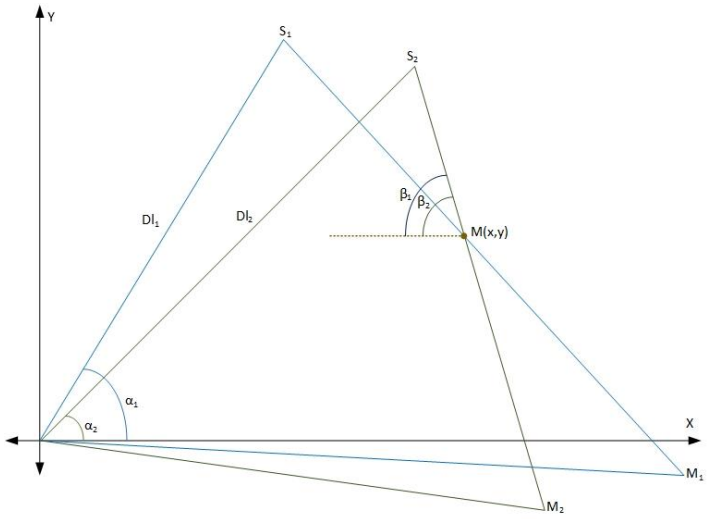

Figure 3.3: Self-positioning through two transmitted signals of same BS

The Mathematical solution is mainly based on intersecting SM lines and finding coordinates of the intersection point where it is the place of MO (See Figure 3.3). First, we must write the equations of SN lines of signals in terms of known values. For this purpose, the attributes of a signal must be considered as depicted in the Figure 3.4 for and Figure 3.5 for X-Y and X-Z plains accordingly.

For simplifying the solution problem will be thought in 2D (on X-Y plane) and then the 3th dimension (X-Z) will be added to the solution. On the Figure 3.4, the attributes α_{xyi}, β_{xyi} and dl_i are known. Considering the triangle $\widehat{OS_iA_i}$ on the same figure the following attributes can be derived:

$$xs_i = dl_{xyi}Cos(\alpha_{xyi}),\ ys_i = dl_{xyi}Sin(\alpha_{xyi}) \tag{3.1}$$

$$xa_i = \frac{ys_i}{\tan(\beta_{xyi})} + xs_i \tag{3.2}$$

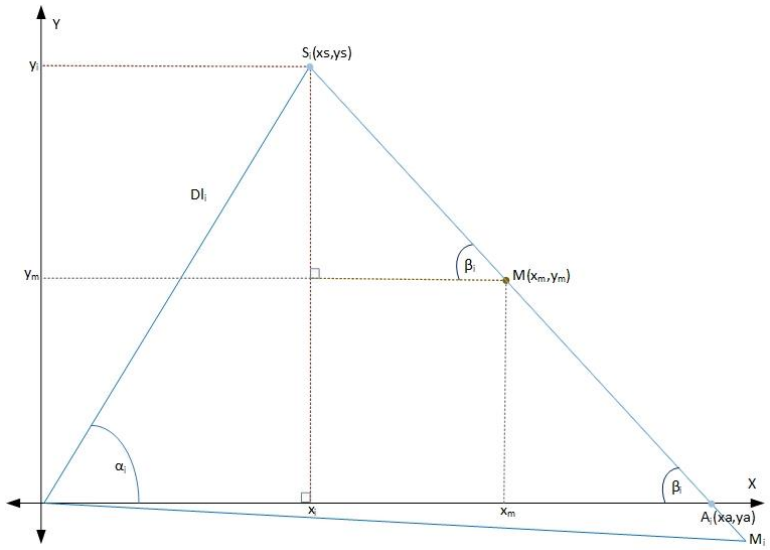

Figure 3.4: The attributes of a SM line

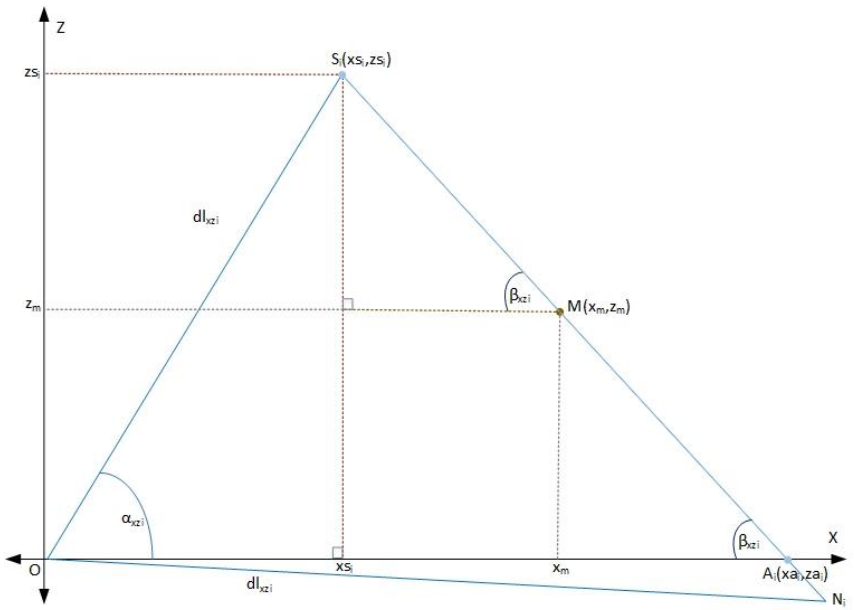

Figure 3.5: The attributes of a SM line on X-Z plane

So, the equation of the line $\overline{S_\iota A_\iota}$ can be derived as:

$$(x-xs_i)(ys_i-ya_i)=(y-ys_i)(xs_i-xa_i) \quad (3.3)$$

$$(ys_i-ya_i)x + (xa_i-xs_i)y=(ys_i-ya_i)xs_i+(xs_i-ya_i)ys_i \quad (3.4)$$

Similarly, if the $\overline{S_\iota A_\iota}$ line which is on the Figure 3.5 is considered the following equation (3.5) will be derived:

$(zs_i\text{-}za_i)x + (xa_i\text{-}xs_i)z=(zs_i\text{-}za_i)xs_i+(xs_i\text{-}za_i)zs_i$ (3.5)

The summation of the equations (3.4) and (3.5) will give us the following equation:

$(ys_i\text{-}ya_{i+}zs_i\text{-}za_i)x +(xa_i\text{-}xs_i)y+(xa_i\text{-}xs_i)z =(ys_i\text{-}ya_{i+}zs_i\text{-}za_i)xs_i++(xs_i\text{-}ya_i)ys_i +(xs_i\text{-}ya_i)ys_i$

Let $b_{1i}= (ys_i\text{-}ya_{i+}zs_i\text{-}za_i)$, $b_{2i}= (xa_i\text{-}xs_i)$, $b3i=(xa_i\text{-}xs_i)$, and

$c_i=(ys_i\text{-}ya_{i+}zs_i\text{-}za_i)xs_i+(xs_i\text{-}ya_i)ys_i +(xs_i\text{-}ya_i)ys_i$, so the equation can be rewritten as:

$b_{1i}\, x + b_{2i}\, y + b_{3i}\, z = c_i$ (3.6)

So, there are three unknowns (x, y, and z that are going to be the planes of the MO), hence in the mathematical solution (MS) it requires three equations (= $\overline{SA}$ lines = transmitted signals), hence i=1..3. Using the (3.6), the linear equation of this model will be like this:

$$A\ r = c \tag{3.8}$$

Where $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ and $a_{ij} = b_{ji}$, $r = [x \quad y \quad z]^T$,

$$c = [c_1 \quad c_2 \quad c_3]^T \tag{3.9}$$

So the solution will be

$$r = A^{-1}\ c \tag{3.10}$$

3.4 Computational Cost of the Mathematical Solution

Table 3.1: Computational costs (weights) of mathematical operations

Operation	**Addition**	**Subtraction**	**Shift**	**Multiplication**
Comp. Cost	1	1	1	40
Operation	**Division**	**Sin**	**Cos**	**Tan**
Comp. Cost	40	404	404	1448

Operation	**Cot**	**Square root**		
Comp. Cost	1448	100		

In Section 3.3, formulation of mathematical solutions were described and the localization process of this model is explained in detail as the final equation is shown at (3.10). Table 3.1 shows the computational cost of the mathematical operation of this model. As output of the system depends on the cost of the system.

Table 3.2: calculating the computational cost of the equation coefficients.

Parameter of vector A	**CCost**
$x_{si}=dl_{xyi}Cos(\alpha_{xyi})$, for i=1..3	444x3=1332
$x_{si}=dl_{xzi}Cos(\alpha_{xzi})$, for i=1..3	444x3=1332
$y_{si}=dl_{xyi}Sin(\alpha_{xyi})$, for i=1..3	444x3=1332
$z_{si}=dl_{xzi}Sin(\alpha_{xzi})$, for i=1..3	444x3=1332
$xa_i=ys_i/tan(\beta_{xyi}) + xs_i$, for i=1..3	1489x3=4467
$xa_i=zs_i/tan(\beta_{xzi}) + xs_i$, for i=1..3	1489x3=4467
$ya_i=0$, for i=1..3	0
$za_i=0$, for i=1..3	0
$b1_i=ys_i-ya_{i+}zs_i-za_i$, for i=1..3	3x3=9
$b2_i=xa_i-xs_i$, for i=1..3	1x3=3
$b3_i= xa_i-xs_i$, for i=1..3	1x3=3
$c_i=(ys_i-ya_{i+}zs_i-za_i)xs_i+(xs_i-ya_i)ys_i +(xs_i-ya_i)ys_i$, for i=1..3	127x3=381
Sub Total	14658

Table 3.3: Computational cost of vector b.

Parameter of vector b	**CCost**
b1= ys_i-ya_i+zs_i-za_i	22
b2= xa_i-xs_i	62
b3= xa_i-xs_i	62
The total computational cost of vector B	106

Thereafter, Vector A is simplified as in (3.35)

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix} \quad (3.35)$$

Let say E = $(A^T A) = \begin{bmatrix} d_1 & d_2 & d_3 \\ d_4 & d_5 & d_6 \\ d_7 & d_8 & d_9 \end{bmatrix}$

Table 3.4: The computational cost calculation of E

Parameters of vector E	**CCost**
$d_1 = c_1 c_1 + c_4 c_4 + c_7 c_7$	45
$d_2 = c_1 c_2 + c_4 c_5 + c_7 c_8$	45
$d_3 = c_1 c_3 + c_4 c_6 + c_7 c_9$	45
$d_4 = c_2 c_1 + c_5 c_4 + c_8 c_7$	45
$d_5 = c_2 c_2 + c_5 c_5 + c_8 c_8$	45
$d_6 = c_2 c_3 + c_5 c_6 + c_8 c_9$	45
$d_7 = c_3 c_1 + c_6 c_4 + c_9 c_7$	45
$d_8 = c_3 c_2 + c_6 c_5 + c_9 c_8$	45
$d_9 = c_3 c_3 + c_6 c_6 + c_9 c_9$	45
The total computational cost of E	405

To solve the equation first the inverse of the matrix A should be calculated in this way:

$A^{-1} = \frac{1}{\det(A)} adj(A)$. The determinants of each coefficient will be calculated in the Table 3.4.

Table 3.5: Calculating the computational cost of determinants of a_{ij} coefficients.

Processes	**CCost**
det(A) = a_{11} det(a_{11}) - a_{12} det(a_{12}) + a_{13} det(a_{13})	122
det(a_{11}) = a_{22} a_{33} – a_{23} a_{32}	81
det(a_{12}) = a_{21} a_{33} -a_{23} a_{31}	81
det(a_{13}) = a_{21} a_{32} -a_{22} a_{31}	81
Sub Total	365

On the next step, let adj(A) = E^T, so E has to be as in (3.11).

$$E = \begin{bmatrix} +\det(E_1) & -\det(E_4) & +\det(E_7) \\ -\det(E_2) & +\det(E_5) & -\det(E_8) \\ +\det(E_3) & -\det(E_6) & +\det(E_9) \end{bmatrix} \quad (3.11)$$

Where each determinant can be rewritten as:

$$+\det(E_1) = +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}, -\det(E_2) = -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}, +\det(E_3) = +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

$$-\det(E_4) = +\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix}, +\det(E_5) = +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix}, -\det(E_6) = +\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix}$$

$$+\det(E_7) = +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix}, -\det(E_8) = -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix}, +\det(E_9) = +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$$

Table 3.6: Calculating the computational cost of E.

Processes	**CCost**
$+\det(E_1)= +a_{22}\, a_{33} - a_{32}\, a_{23}$	81
$-\det(E_2)= -a_{21}\, a_{33} + a_{31}\, a_{23}$	81
$+\det(E_3)= +a_{21}\, a_{32} - a_{31}\, a_{22}$	81
$-\det(E_4)= -a_{12}\, a_{33} + a_{32}\, a_{13}$	81
$+\det(E_5)= +a_{11}\, a_{33} - a_{31}\, a_{13}$	81
$-\det(E_6)= -a_{11}\, a_{32} + a_{31}\, a_{12}$	81
$+\det(E_7)= +a_{12}\, a_{23} - a_{22}\, a_{13}$	81
$-\det(E_8)= -a_{11}\, a_{23} + a_{13}\, a_{21}$	81
$+\det(E_9)= +a_{11}\, a_{22} - a_{21}\, a_{12}$	81
Sub Total	729

Let det(A)=g and then write the $A^{-1} = \frac{adj(A)}{\det(A)} = \frac{adj(A)}{g}$ as the following:

$$A^{-1} = \begin{bmatrix} \frac{+\det(E_1)}{g} & \frac{-\det(E_4)}{g} & \frac{+\det(E_7)}{g} \\ \frac{-\det(E_2)}{g} & \frac{+\det(E_5)}{g} & \frac{-\det(E_8)}{g} \\ \frac{-\det(E_3)}{g} & \frac{-\det(E_6)}{g} & \frac{+\det(E_9)}{g} \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

The computational cost A^{-1} is 9 x 41= 369.

Then for calculating the $r = A^{-1} c$.

Table 3.7: The computational cost calculation of E^{-1}

Processes	**CCost**
$det(E) = d_1\ det(E_1) - d_2\ det(E_2) + d_3\ det(E_3)$	106
$det(E_1) = d_1\ det(E_{1,1})$	62
$det(E_{1,1}) = d_9\ d_5 – d_8\ d_6$	22
$det(E2) = d_2\ det(E_{,2})$	62
$det(E_{1,2}) = d_9\ d_4 – d_7\ d_6$	22
$det(E3) = d_3\ det(E_{1,3})$	62
$det(E_{1,3}) = d_8\ d_4 – d_7\ d_5$	22
The total computational cost of E^{-1}	358

Table 3.8: Calculating the Computation Cost of solving the r

Processes	CCost
$r_1 = x = h_1c_1 + h_2c_2 + h_3c_3$	122
$r_2 = y = h_4c_1 + h_5c_2 + h_6c_3$	122
$r_3 = z = h_7c_1 + h_8c_2 + h_9c_3$	122
Sub Total	366

Total Computation Cost of the mathematical solution is calculated as 16487 CPU cycles.

Chapter 4

PROPOSED MODEL OF SOLUTION

4.1 Introduction

According to the Introduction and literature review, the objective of this research is to introduce a system for finding the location of mobile object which is hardware and software oriented, this solution is fast, accurate, and cost effective. We decided to use coordinate rotation digital computer (CORDIC) algorithm and two fast variation of CORDIC algorithm, they are dynamic angle rotation method (DARM) and fixed angle rotation method (FARM). We also develop a new model which is called BC (Bahadur-Cem) algorithm which will be discussed in more detail in section 4.2.

4.2 BC Algorithm

The BC algorithm contains total of two steps. The first step is used to find the elements on X-Y axis in 2 dimension space for MO, while the other step is used to transform the plane from 2 dimension to 3 dimension by adding the X-Z plane to the model. CORDIC algorithm is placed with trigonometric functions in order to decrease the computational cost of calculation and speed it up because trigonometric algorithms are heavy weighted.

There are two different methods used in BC to discard the algorithms trigonometric, these are FARM and DARM. BC algorithm is executed with both FARM and DARM individually. These are BC-FARM and BC-DARM. These are discussed in more detail in section 4.2.1 and 4.2.2.

4.2.1 BC-FARM Algorithm

FARM is using repetitive stepping angle with fixed rotation while moving toward Mobile object. In BC-FARM algorithm we are using FARM instead of trigonometric functions.

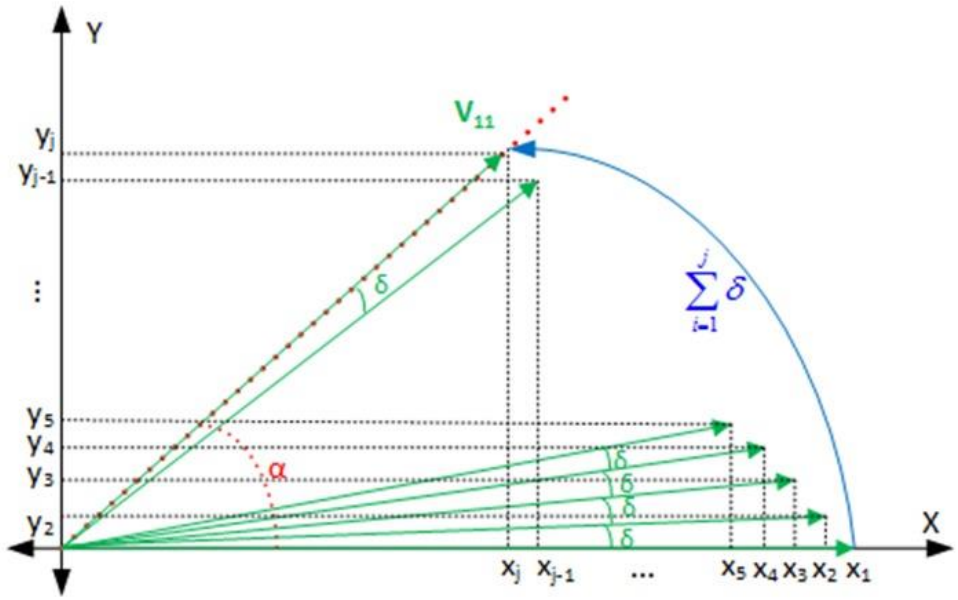

Figure 4.1: Fix Angle Rotation Method FARM

As illustrated in the Figure 4.1 FARM angle in radians is determined by step size σ = arcsine (2^{-k}), here k is used to determine the accuracy, circular rotation matrix is M. Here, v = 2^{-k}, u = $2^{-(2k+1)}$ and the operator which will decide the direction of rotation is s = ±1. Here y1, y2, y3 until yj and x1, x2, x3 until xj represents total number of rotations between MO and BS, combining all of them together gives the total distance between BS and MO. Using binary shift and basic arithmetic operations instead of trigonometric functions to decrease the computational cost of the process. As a result, FARM is very useful when we have limited hardware resources.

Algorithm 4.1: Emulating the sin, cos functions in BC-FARM

Input: α, σ, ϵ, y, z, d_angle
Output y_j, z_j

Begin

```
if (α > π/2) start
     sign ← −1
     α ← π/2 − α
 terminate

 while((α-d_angle) > ϵ) start
   old_y ← y
   old_z ← z
   y ← old_y − old_y ∗ u + sign ∗ old_z ∗ v
   z ← old_z − old_z ∗ u − sign ∗ old_y ∗ v
  d_angle ← d_angle + 2−k
   terminate while
```

End

As mentioned in algorithm 3.1, in the initial step of BC-FARM, the vector V11 (mentioned in Figure 3.1) is placed and then it is continuously rotated with the angle α on X-Y plan. Here α is aod1 (angle of departure) and d1 is DL1. DL1 is starting from origin (O) which is placed on x-axis and then it is rotated step by step on the direction of counterclockwise with fix angle σ to reach the total rotation of angle to aod1 which is the angle that we want to achieve. The loop will stop when (α-c_angle) > ϵ).

4.2.2 BC-DARM Algorithm

Dynamic Angle Rotation Method (DARM) uses irregular rotations with unsteady rotation angle to reach the targeted object more closely. DARM is changing the rotation angle by itself in irregular iterations. Here BC-DARM is used with the same concept of CORDIC algorithm with changing rotation angle $\sigma i = \arcsin(2^{-k})$ where i=1...7 and k=5...11. The repetitive iterations start with σ = 1 and k = 5, so the equation will be $\sigma 1 = \arcsin(2^{-5})$. After that it will continue in I = 7 and k = 11. When all these

iterations are completed dynamically then we can find the location of MO by summing all the iterations together.

If we increase the non-fixed iteration it will increase the cost of computation directly but we will move more closely to targeted object. On the other hand, if we decrease the number of iterations it will automatically decrease the cost but we have to compromise on accuracy. So here we found the best result with minimum cost as mentioned in figure 5.3.

Chapter 5

SIMULATION

The general model explained in Figure 3.1, 3.2 and 3.3 and the BC algorithm explained in section 3.2 which has further two parts BC-FARM and BC-DARM which are mentioned in detail in section 4.2.1 and section 4.2.2.

5.1 Simulation Result

BC-DARM recorded computational cost and accuracy level with different value of k is mentioned in Figure 5.1. Figure 5.1 shows the accuracy of BC-FARM which is improving exponentially when the value of k is increased. In the same way, in BC-DARM the accuracy is also getting better and better when the value of k in increased, but, here in BC-DARM the value of k is increased automatically, while the stepping angle in rotations is changing dynamically by itself in every iteration to give fixed accuracy value which is root men square root.

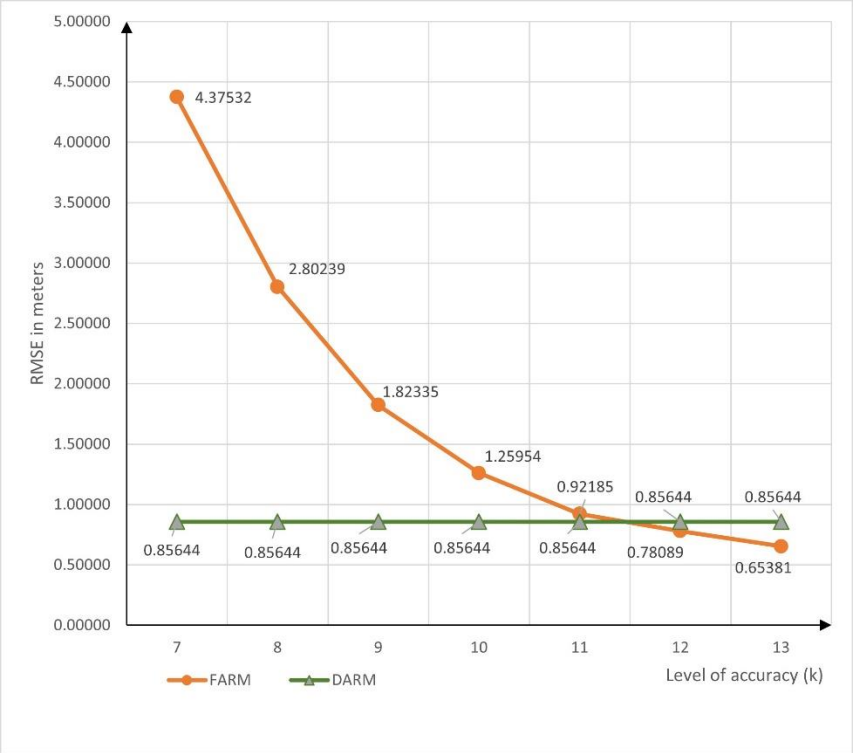

Figure 5.1: Accuracy of BC-FARM and BC-DA RM

Figure 5.2 shows the computational cost of BC-FARM and BC-DARM, as we can see in the figure the computational cost is increasing for both BC-FARM and BC-DARM as the value of k is increased. Which means that somehow we have to compromise on accuracy or computational cost, because it is quite difficult to have the best result with minimum amount of resources.

As compared to the previous research and studies this model is showing much better results with calculated mathematical calculations.

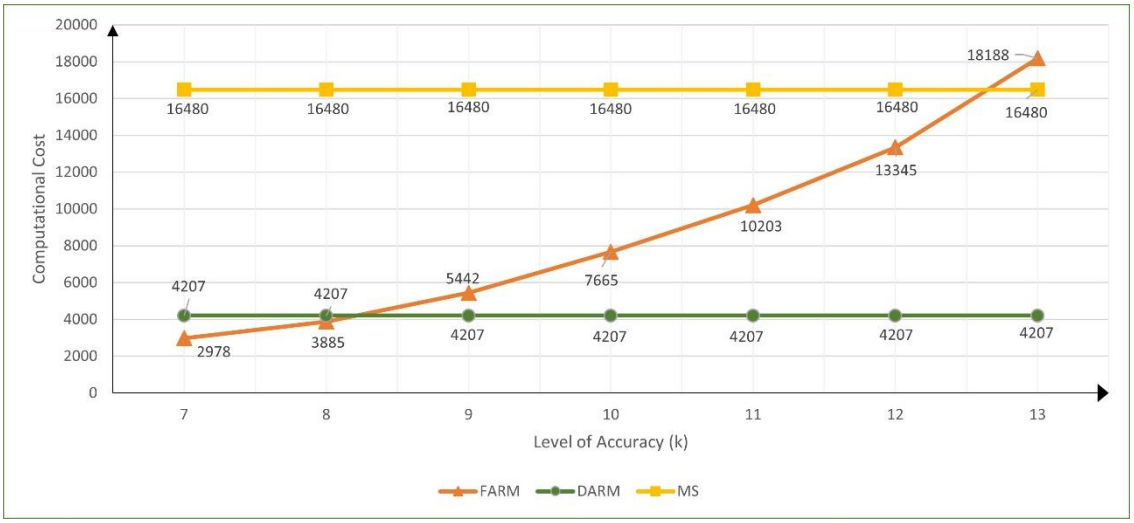

Figure 5.2 Level of Accuracy of BC-FARM, BC-DARM and MS

5.2 Comparison with other Studies

As compared to previous research and studies the CPU cycles are calculated as 24860 with location estimation with three base stations. For two base stations the total CPU cycles were calculated as 20745. In my research the total Computation Cost of the mathematical solution is calculated as 16487 CPU cycles. As compared to other studies I can say that this system is working much better than other systems,

Chapter 6

CONCLUSION

In this thesis, the detail of newly developed MO localization technique is explained. This technique is known as Bahadur-Cem (BC) algorithm. This technique is further divided into two parts theses are fixed angle rotation method (FARM) and dynamic angle rotation method (DARM) which using the concept of CORDIC algorithm with simple add and shift concept with both single bounce signal SBS non line of sight NLOS signal and directly approaching signal which is known as line of sight LOS.

The results of both BC-FARM and BC-DARM were outstanding and gave much better accuracy and computational cost as compared to previously developed approaches. BC algorithm is used with CORDIC algorithm to improve the accuracy and reduce the Computational cost.

In this thesis, we consider BC algorithm for single bounced signal SBS for non-line of sight NLOS, but in future BC algorithms can be used for multi-bounced scattered non line of sight NLOS signal and also for line of sight LOS arriving signals for better performance.

REFERENCES

Bertrand, A., Doclo, S., Gannot, S., Ono, N., & van Waterschoot, T. (2015). Special issue on wireless acoustic sensor networks and ad hoc microphone arrays. *Signal Processing*, *107*(C), 1-3.

Changela, A., Zaveri, M., & Verma, D. (2023). A Comparative Study on CORDIC Algorithms and Applications. *Journal of Circuits, Systems and Computers*, *32*(05), 2330002.

Chen, W., Wu, T., Tang, W., Jin, K., & Huang, G. (2020, May). Implementation Method of CORDIC Algorithm to Improve DDFS Performance. In *2020 IEEE 3rd International Conference on Electronics Technology (ICET)* (pp. 58-62). IEEE.

Choi, H., Geeves, M., Alsalam, B., & Gonzalez, F. (2016, March). Open source computer-vision based guidance system for UAVs on-board decision making. In *2016 IEEE aerospace conference* (pp. 1-5). IEEE.

Ferrer, M., de Diego, M., Piñero, G., & Gonzalez, A. (2015). Active noise control over adaptive distributed networks. *Signal Processing*, *107*, 82-95.

Gergen, S., Nagathil, A., & Martin, R. (2015). Classification of reverberant audio signals using clustered ad hoc distributed microphones. *Signal Processing*, *107*, 21-32.

Mahdavi, H., & Timarchi, S. (2020). Improving architectures of binary signed-digit CORDIC with generic/specific initial angles. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *67*(7), 2297-2304.

Markovich-Golan, S., Bertrand, A., Moonen, M., & Gannot, S. (2015). Optimal distributed minimum-variance beamforming approaches for speech enhancement in wireless acoustic sensor networks. *Signal Processing*, *107*, 4-20.

Rullan-Lara, J. L., Salazar, S., & Lozano, R. (2011, April). Uav real-time location using a wireless sensor network. In *2011 8th Workshop on Positioning, Navigation and Communication* (pp. 18-23). IEEE.

Salgar, M. S. P., Sachidanand, B. N., Metigoudar, V. M., & Zirmite, M. P. P. (2014). Range based Localization Scheme for 3D Wireless Sensor Network using Joint Distance and Angle Information: A Brief Review. *International Journal of Engineering Research*, *3*(6).

Sharma, N. K., Rathore, S., & Khan, M. R. (2020, January). A comparative analysis on coordinate rotation digital computer (CORDIC) algorithm and its use on computer vision technology. In *2020 First International Conference on Power, Control and Computing Technologies (ICPC2T)* (pp. 106-110). IEEE.

Shi, Z., He, Q., & Liu, Y. (2020). Accelerating parallel Jacobi method for matrix eigenvalue computation in DOA estimation algorithm. *IEEE Transactions on Vehicular Technology*, *69*(6), 6275-6285.

Tatlas, N. A., Potirakis, S. M., Mitilineos, S. A., & Rangoussi, M. (2015). On the effect of compression on the complexity characteristics of wireless acoustic sensor network signals. *Signal Processing*, *107*, 153-163.

Tang, W., & Xu, F. (2020). A noniterative Radix-8 CORDIC algorithm with low latency and high efficiency. *Electronics*, *9*(9), 1521.

Wang, G., Chen, S., Ye, J., & Zhang, F. (2020). A chirp signal generator without multiplier based on improved CORDIC algorithm. *Journal of the Chinese Institute of Engineers*, *43*(6), 532-540.

Wang, Y., Luo, Y., Wang, Z., Shen, Q., & Pan, H. (2020). GH CORDIC-Based Architecture for Computing $ N $ th Root of Single-Precision Floating-Point Number. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *28*(4), 864-875.

Zeng, Y., & Hendriks, R. C. (2015). Distributed estimation of the inverse of the correlation matrix for privacy preserving beamforming. *Signal Processing*, *107*, 109-122.

APPENDICES

Appendix A: MATLAB Source Code of SBS NLOS Sample Generator

```
clc;
clear;
%-------- Parameters for Sampling ----------------------
sample_size=1000;
mlex=500; % Possible max length of the experment field in x-axis
mley=500; % Possible max length of the experment field in y-axis
mlez=500; % Possible max length of the experment field in z-axis
r_min_MOx=0.70; % Percentage of the total field length MO positioning starts
r_max_MOx=1.00; % Percentage of the total field length MO positioning ends
r_min_MOy=0.50; % Percentage of the total field length MO positioning starts
r_max_MOy=0.20; % Percentage of the total field length MO positioning ends
r_min_MOz=0.50; % Percentage of the total field length MO positioning starts
r_max_MOz=0.10; % Percentage of the total field length MO positioning ends

r_min_SC=0.20; % Percentage of the total field length SC positioning start
r_max_SC=0.60; % Percentage of the total field length SC positioning ends
%-------------------------------------------------------------------------
```

```
bsx=0; bsy=0; bsz=0;

formatOut = 'yyyymmdd-HHMMSS';

str=datestr(now,formatOut);

fileoutname=strcat('SAMPLESET-SBS-',str,'.txt');

%--------------------------------------------------------------------------



fw = fopen(fileoutname,'w');

fprintf(strcat('(1) Out file ',fileoutname,' is created!!!\n'));

fprintf('\n%5s \t %6s \t %6s \t %6s \t','S.NO','BS(x)', 'BS(y)','BS(z)');

fprintf('%6s \t %6s \t %6s \t','S1(x)', 'S1(y)','S1(z)');

fprintf('%6s \t %6s \t %6s \t','S2(x)', 'S2(y)','S2(z)');

fprintf('%8s \t %8s \t %8s \t','Mo(x)   ', 'Mo(y)   ','Mo(z)   ');

fprintf('%8s \t %8s \t %8s \t %8s \t','dl11xy ', 'dl11xz ','dl12xy ', 'dl12xz ');

fprintf('%8s \t %8s \t %8s \t %8s \t','aod1xy ', 'aod1xz ','aoa1xy ', 'aoa1xz ');

fprintf('%8s \t %8s \t %8s \t %8s \t','dl21xy ', 'dl21xz ','dl22xy ', 'dl22xz ');

fprintf('%8s \t %8s \t %8s \t %8s \n','aod2xy ', 'aod2xz ','aoa2xy ', 'aoa2xz ');

i=0;

while (i<sample_size)

i = i +1;
```

```
%---- Generating the position of the Mobile Object --------------------

low_x= ceil(mlex*r_min_MOx);

high_x= ceil(mlex*r_max_MOx);

mx=low_x+ceil(rand()*(high_x-low_x));


low_y= ceil(mley*r_min_MOy);

high_y= ceil(mley*r_max_MOy);

my=low_y+ceil(rand()*(high_y-low_y));


low_z= ceil(mlez*r_min_MOz);

high_z= ceil(mlez*r_max_MOz);

mz=low_z+ceil(rand()*(high_z-low_z));


%------------ Generating the positions of the scatters ----------------

%---- Placing the Scatter-1 ------------------------------------------

low_x= ceil(mlex*0.10);     %-- x range starts from 10%

high_x= mx-ceil(mlex*0.10); %-- x range ends (mx - 10%)

s1x=low_x+ceil(rand()*(high_x-low_x));
```

```
low_y= my + ceil(mley*0.10);%-- y range starts at (my + 10%)

high_y= ceil(mley*1.00);    %-- y range ends at 100%

s1y=low_y+ceil(rand()*(high_y-low_y));


low_z= ceil(mlez*r_min_SC); %-- z range starts at (mz + 10%)

high_z= ceil(mlez*1.00);%-- z range ends at 100%

s1z=low_z+ceil(rand()*(high_z-low_z));


%---- Placing the Scatter-2 ----------------------------------------

% If the SCATTER-1 is after the half of the region, place the SCATTER-2

% at most(10%) before the SCATTER-1 else place the SCATTER-2 at least

% (10%) after the SCATTER-1

%--------------------------------------------------------------------

if (s1x>(high_x/2))

high_x= s1x-ceil(mlex*0.10);

s2x=low_x+ceil(rand()*(high_x-low_x));

else

low_x= s1x+ceil(mlex*0.10);

s2x=low_x+ceil(rand()*(high_x-low_x));

end
```

```
if (s1y>(high_y/2))

high_y= s1y-ceil(mley*0.10);

s2y=low_y+ceil(rand()*(high_y-low_y));

else

low_y= s1y+ceil(mley*0.10);

s2y=low_y+ceil(rand()*(high_y-low_y));

end


if (s1z>high_z/2)

high_z= s1z-ceil(mlez*0.10);

s2z=low_x+ceil(rand()*(high_z-low_z));

else

low_z= s1z+ceil(mlez*0.10);

s2z=low_z+ceil(rand()*(high_z-low_z));

end

%-----------------------------------------------------------------------


%---- Calculating the DLs and aod, aoa values for each BS-SC-MO beams -

dl11_xy=sqrt((s1x)^2+(s1y)^2);

dl11_xz=sqrt((s1x)^2+(s1z)^2);
```

```
dl12_xy=sqrt((s1x-mx)^2+(s1y-my)^2);

dl12_xz=sqrt((s1x-mx)^2+(s1z-mz)^2);


dl21_xy=sqrt((s2x)^2+(s2y)^2);

dl21_xz=sqrt((s2x)^2+(s2z)^2);


dl22_xy=sqrt((s2x-mx)^2+(s2y-my)^2);

dl22_xz=sqrt((s2x-mx)^2+(s2z-mz)^2);


aod1_xy =asind(s1y/dl11_xy);

aod1_xz =asind(s1z/dl11_xz);

aod2_xy =asind(s2y/dl21_xy);

aod2_xz =asind(s2z/dl21_xz);


aoa1_xy =asind(abs(s1y-my)/dl12_xy);

aoa1_xz =asind(abs(s1z-mz)/dl12_xz);

aoa2_xy =asind(abs(s2y-my)/dl22_xy);

aoa2_xz =asind(abs(s2z-mz)/dl22_xz);


%---------- Printing the data row ----------------------------------
```

```
if (length(int2str(i))==1)

str = strcat('000',int2str(i));

end

if (length(int2str(i))==2)

str = strcat('00',int2str(i));

end

if (length(int2str(i))==3)

str = strcat('0',int2str(i));

end

if (length(int2str(i))==4)

str = int2str(i);

end

if

((aod1_xy==aod2_xy)||(aod1_xz==aod2_xz)||(aoa1_xy==aoa2_xy)||(aoa1_xz==aoa2

_xz))

continue;

end;



fprintf('%s. \t',str);

fprintf(fw,'%8.4f\t%8.4f\t%8.4f\t',bsx, bsy, bsz);
```

```
fprintf('%8.4f\t%8.4f\t%8.4f\t',bsx, bsy, bsz);

fprintf(fw,'%8.4f\t%8.4f\t%8.4f\t',s1x, s1y, s1z);
fprintf('%8.4f\t%8.4f\t%8.4f\t',s1x, s1y, s1z);

fprintf(fw,'%8.4f\t%8.4f\t%8.4f\t',s2x, s2y, s2z);
fprintf('%8.4f\t%8.4f\t%8.4f\t',s2x, s2y, s2z);

fprintf(fw,'%8.4f\t%8.4f\t%8.4f\t',mx, my, mz);
fprintf('%8.4f\t%8.4f\t%8.4f\t',mx, my, mz);

fprintf(fw,'%8.4f\t%8.4f\t%8.4f\t%8.4f\t',dl11_xy, dl11_xz, dl12_xy, dl12_xz);
fprintf('%8.4f\t%8.4f\t%8.4f\t%8.4f\t',dl11_xy, dl11_xz, dl12_xy, dl12_xz);

fprintf(fw,'%8.4f\t%8.4f\t%8.4f\t%8.4f\t',aod1_xy, aod1_xz, aoa1_xy, aoa1_xz);
fprintf('%8.4f\t%8.4f\t%8.4f\t%8.4f\t',aod1_xy, aod1_xz, aoa1_xy, aoa1_xz);

fprintf(fw,'%8.4f\t%8.4f\t%8.4f\t%8.4f\t',dl21_xy, dl21_xz, dl22_xy, dl22_xz);
fprintf('%8.4f\t%8.4f\t%8.4f\t%8.4f\t',dl21_xy, dl21_xz, dl22_xy, dl22_xz);
```

```
fprintf(fw,'%8.4f\t%8.4f\t%8.4f\t%8.4f',aod2_xy, aod2_xz, aoa2_xy, aoa2_xz);

fprintf('%8.4f\t%8.4f\t%8.4f\t%8.4f',aod2_xy, aod2_xz, aoa2_xy, aoa2_xz);



if (i<sample_size)

fprintf('\t\n');

fprintf(fw,'\t\n');

end;



end % while (i<=sample_size)

%--- Ending Processes ----------------------------------

fclose(fw);

fprintf(strcat('\n\nOut file\t',fileoutname,' is closed!!!'));

load gong

sound(y,Fs)

h = msgbox('Operation Completed----');
```

Appendix B: MATLAB Source Code of SBS_NLOS_Fixed Algorithm

```
% 3D Self Navigation with Single Base Station signals thru NLOS

% arriving apporach with FIXED ANGLE ROTATION METHOD

%-----------------------------------------------------------------

clc;

clear;

format long;

%-------- Definitions ------------------------------------------

k = 6;

v = 2^-k;

u = 2^-(2*k+1);

err =10^-k;

loop_trsh=10000; % Trashold value for loop count

step_val=1; % Stepping value in meters

turn_allowed=45;   % Number of maximum sample rows will be processed

%samplesetfile='SAMPLESET-SBS-20230524-134633.txt';

samplesetfile='SAMPLESET-SBS-20230623-173430.txt';

cnt = 0;           % Counter for reading sample rows

simresult_filename = strcat('simres-SBS-fixed-k', int2str(k),'-', ...
```

```
samplesetfile(15:33));

fprintf('\n 00 - Simulation out file name: %s',simresult_filename);

%---- For Statistics ----------------------------------------------------

t_err_x= 0.00000;

t_err_y= 0.00000;

t_err_z= 0.00000;

t_rmse = 0.00000;

t_cost = 0.00000;

total_cost= 0.00000;

tt_cost= 0.00000;

fprintf('\n 01 - Parameters are defined and initialized. k=%f', k);


fr=fopen(samplesetfile,'r');

fprintf('\n 02 - Sample Set file: %s is is opened for reading', samplesetfile);


fw=fopen(simresult_filename,'w');

fprintf(fw,'%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t
%8s\t%8s\t%8s\t%8s\t%8s\n', ...

'    xm','    ym','    zm','    x1','    y1','    x2','    y2','    x3','    z3','    x1','
y1','    x2','    y2','(xm-x1)','(ym-y1)','(zm-z3)','  RMSE  ','(-COST-)');
```

```
fprintf('\n 03 - Out file \t %s is created!!!', simresult_filename);

fprintf('\n 04 - Reading and processing the sample sets');

%----- Reading --------------------------------------------------------

while ~feof(fr)

total_cost=0.00000;

t_cost=0.00000;

cnt=cnt+1;

if(cnt>turn_allowed)

break;

end; % end of if(cnt>turn_allowed)

fprintf('\n-------------------------------------------------------');

fprintf('-------------------------------------');

fprintf('\n\t Sample Set %3d is in process ...',cnt);


bsx=fscanf(fr,'%f',1);

bsy=fscanf(fr,'%f',1);

bsz=fscanf(fr,'%f',1);


s1x = fscanf(fr,'%f',1);

s1y = fscanf(fr,'%f',1);
```

```
s1z = fscanf(fr,'%f',1);


s2x = fscanf(fr,'%f',1);

s2y = fscanf(fr,'%f',1);

s2z = fscanf(fr,'%f',1);


mx = fscanf(fr,'%f',1);

my = fscanf(fr,'%f',1);

mz = fscanf(fr,'%f',1);


dl11_xy = fscanf(fr,'%f',1);

dl11_xz = fscanf(fr,'%f',1);

dl12_xy = fscanf(fr,'%f',1);

dl12_xz = fscanf(fr,'%f',1);

dl1_xy  = dl11_xy + dl12_xy;

dl1_xz  = dl11_xz + dl12_xz;


aod1_xy = fscanf(fr,'%f',1);

aod1_xz = fscanf(fr,'%f',1);

aoa1_xy = fscanf(fr,'%f',1);
```

```
aoa1_xz = fscanf(fr,'%f',1);


dl21_xy = fscanf(fr,'%f',1);

dl21_xz = fscanf(fr,'%f',1);

dl22_xy = fscanf(fr,'%f',1);

dl22_xz = fscanf(fr,'%f',1);

dl2_xy  = dl21_xy + dl22_xy;

dl2_xz  = dl21_xz + dl22_xz;


aod2_xy = fscanf(fr,'%f',1);

aod2_xz = fscanf(fr,'%f',1);

aoa2_xy = fscanf(fr,'%f',1);

aoa2_xz = fscanf(fr,'%f',1);


fprintf('\n\n\t Knowns: BS(%f,%f), aod1_xy:%f, DL1_xy:%f, aoa1_xy: %f', ...

bsx, bsy, aod1_xy, dl1_xy, aoa1_xy);

fprintf('\n\t         BS(%f,%f), aod2_xy:%f, DL2_xy:%f, aoa2_xy: %f', ...

bsx,bsy,aod2_xy, dl2_xy, aoa2_xy);

fprintf('\n\n\t (A)Placing the stepping vectors v11, v12, v21, v22 ');

fprintf('onto the graph (Stepping value = %f meters)',step_val);
```

```
%-----------------------------------------------------------------------

% PLACING THE STEPPER VECTORS ON THE GRAPH

% Stepping vectors are going to be used in gredually broken the DL

% (It is going to be used to simulate the sin and cos functions)

%-----------------------------------------------------------------------

% (1) Placing the v11: The stepping_value is laid on x-axis,

% and then rotated until it reaches to the angle rad_aod1_xy

% xs1, ys1 are the coordinates of the Arrow head of the vector v11

t_cost=0.00000;

rad_aod1_xy=(aod1_xy*pi)/180;

ro=0.00000; % current angle after one step rotation

xs1=step_val;

ys1=0.00000;

if (rad_aod1_xy>pi/2)

sign=-1;

delta=pi+sign*rad_aod1_xy;

else

sign=1;

delta=sign*rad_aod1_xy;

end
```

```
% Rotating the vector until it reaches to delta (targetted angle)

while((delta-ro)>err)

t_cost=t_cost+1;

oldx1=xs1;

xs1=xs1-xs1*u-ys1*v;

ys1=ys1-ys1*u+oldx1*v;

ro=ro+2^-k;

end %while

fprintf('\n\t\t Stepping vector v11(x,y) is placed with the angle ');

fprintf('(%f) as %f -->aod1xy(%f), err: %f', aod1_xy, ro, ...

delta,(delta-ro));

fprintf('\n\t\t Calculated arrow head position V11');

fprintf('(%f,%f) with cost:%5d',xs1,ys1, t_cost);

total_cost=total_cost + t_cost;

t_cost=0;



%-----------------------------------------------------------------------

% (2) The stepping_value is laid on x-axis, and then rotated

% until it reaches to the angle rad_aoa1_xy

% xs2, ys2 are the coordinates of the arrow head of the vector V12
```

```
xs2=step_val;

ys2=0.00000;

ro=0.00000;

rad_aoa1_xy=(aoa1_xy*pi)/180;

if (rad_aoa1_xy>pi/2)

sign=-1;

delta=pi+sign*rad_aoa1_xy;

else

sign=1;

delta=sign*rad_aoa1_xy;

end

while((delta-ro)>err)

t_cost=t_cost+1;

oldx2=xs2;

xs2=xs2-xs2*u-ys2*v;

ys2=ys2-ys2*u+oldx2*v;

ro=ro+2^-k;

end %while((delta-ro)>=err)

fprintf('\n\t\t Stepping vector v12(x,y) is placed with the angle ');

fprintf('(%f) as %f -->aoa1_xy(%f), err: %f', aoa1_xy, ...
```

```
ro, delta,(delta-ro));

fprintf('\n\t\t Calculated arrow head position of V12');

fprintf('(%f,%f) with cost:%5d',xs2,ys2, t_cost);

total_cost=total_cost + t_cost;

t_cost=0;



% (3) Placing the v21: The stepping_value is laid on x-axis,

% and then rotated until it reaches to the angle rad_aod2_xy

% xs4, ys4 are the coordinates of the Arrow head of the vector v21

rad_aod2_xy=(aod2_xy*pi)/180;

ro=0.00000; % current angle after one step rotation

xs3=step_val;

ys3=0.00000;

if (rad_aod2_xy>pi/2)

sign=-1;

delta=pi+sign*rad_aod2_xy;

else

sign=1;

delta=sign*rad_aod2_xy;

end
```

```
% Rotating the vector until it reaches to delta (targetted angle)

while((delta-ro)>err)

t_cost=t_cost+1;

oldx3=xs3;

xs3=xs3-xs3*u-ys3*v;

ys3=ys3-ys3*u+oldx3*v;

ro=ro+2^-k;

end %while

fprintf('\n\t\t Stepping vector v21(x,y) is placed with the angle ');

fprintf('(%f) as %f -->aod2xy(%f), err: %f', aod2_xy, ...

ro, delta,(delta-ro));

fprintf('\n\t\t Calculated arrow head position V21');

fprintf('(%f,%f) with cost:%5d',xs3,ys3, t_cost);

total_cost=total_cost + t_cost;

t_cost=0;



%-----------------------------------------------------------------------

% (4) The stepping_value is laid on x-axis, and then rotated

% until it reaches to the angle rad_aoa2_xy

% xs4, ys4 are the coordinates of the arrow head of the vector V22
```

```
xs4=step_val;

ys4=0.00000;

ro=0.00000;

rad_aoa2_xy=(aoa2_xy*pi)/180;

if (rad_aoa2_xy>pi/2)

sign=-1;

delta=pi+sign*rad_aoa2_xy;

else

sign=1;

delta=sign*rad_aoa2_xy;

end

while((delta-ro)>err)

t_cost=t_cost+1;

oldx4=xs4;

xs4=xs4-xs4*u-ys4*v;

ys4=ys4-ys4*u+oldx4*v;

ro=ro+2^-k;

end %while((delta-ro)>=err)

fprintf('\n\t\t Stepping vector v22(x,y) is placed with the angle ');

fprintf('(%f) as %f -->aoa2_xy(%f), err: %f', aoa1_xy, ...
```

```
ro, delta,(delta-ro));

fprintf('\n\t\t Calculated arrow head position of V22');

fprintf('(%f,%f) with cost:%5d',xs4,ys4, t_cost);

total_cost=total_cost + t_cost;

t_cost=0;



%-------------------------------------------------------------------------

% (5) The stepping_value is laid on x-axis, and then rotated

% until it reaches to the angle rad_aod1_xz

% xs5, ys5 are the coordinates of the arrow head of the vector V31

xs5=step_val;

zs5=0.00000;

ro=0.00000;

rad_aod1_xz=(aod1_xz*pi)/180;

if (rad_aod1_xz>pi/2)

sign=-1;

delta=pi+sign*rad_aod1_xz;

else

sign=1;

delta=sign*rad_aod1_xz;
```

```
end

while((delta-ro)>err)

t_cost=t_cost+1;

oldx5=xs5;

xs5=xs5-xs5*u-zs5*v;

zs5=zs5-zs5*u+oldx5*v;

ro=ro+2^-k;

end %while((delta-ro)>=err)

fprintf('\n\t\t Stepping vector v31(x,z) is placed with the angle ');

fprintf('(%f) as %f -->aod1_xz(%f), err: %f', aod1_xz, ...

ro, delta,(delta-ro));

fprintf('\n\t\t Calculated arrow head position of V31');

fprintf('(%f,%f) with cost:%5d',xs5,zs5, t_cost);

total_cost=total_cost + t_cost;

t_cost=0;



%-------------------------------------------------------------------------

% (6) The stepping_value is laid on x-axis, and then rotated

% until it reaches to the angle rad_aoa1_xz

% xs6, ys6 are the coordinates of the arrow head of the vector V32
```

```
xs6=step_val;

zs6=0.00000;

ro=0.00000;

rad_aoa1_xz=(aoa1_xz*pi)/180;

if (rad_aoa1_xz>pi/2)

sign=-1;

delta=pi+sign*rad_aoa1_xz;

else

sign=1;

delta=sign*rad_aoa1_xz;

end

while((delta-ro)>err)

t_cost=t_cost+1;

oldx6=xs6;

xs6=xs6-xs6*u-zs6*v;

zs6=zs6-zs6*u+oldx6*v;

ro=ro+2^-k;

end %while((delta-ro)>=err)

fprintf('\n\t\t Stepping vector v32(x,z) is placed with the angle ');

fprintf('(%f) as %f -->aoa1_xz(%f), err: %f', aoa1_xz, ...
```

```
ro, delta,(delta-ro));

fprintf('\n\t\t Calculated arrow head position of V32');

fprintf('(%f,%f) with cost:%5d',xs6,zs6, t_cost);

total_cost=total_cost + t_cost;

t_cost=0;




%---------------------------------------------------------------------

% Positionning the vectors with length DL1 and DL2 with angles

% aod1 and aod2 respectively

%---------------------------------------------------------------------

fprintf('\n\t (B)Placing the vector DL1 and DL2 to the graph ... ');

% Positioning the vector (DL1)-----------------------------------------

rad_aod1_xy=(aod1_xy*pi)/180;

ro=0.00000; % current angle after one step rotation



% Step-1: Lie down DL1 to x-axis (xr1=DL1, yr1=0) angle=0

xr1=dl1_xy;

yr1=0.00000;

if (rad_aod1_xy>pi/2)
```

```
sign=-1;

delta=pi+sign*rad_aod1_xy;

else

sign=1;

delta=sign*rad_aod1_xy;

end

% Step-2: Rotate the vector until angle (ro) meets the aod1_xy (delta)

% xr1, yr1 are the coordinates of the Arrow head of the vector DL1

while((delta-ro)>err)

t_cost=t_cost+1;

oldxr1=xr1;

xr1=xr1-xr1*u-yr1*v;

yr1=yr1-yr1*u+oldxr1*v;

ro=ro+2^-k;

end %while

fprintf('\n\t\t Vector DL1(x,y)=%f is placed with the angle ', dl1_xy);

fprintf('(%f) as %f -->aod1xy(%f), err: %f', aod1_xy, ...

ro, delta,(delta-ro));

fprintf('\n\t\t Calculated arrow head position of DL1');

fprintf('(%f,%f) with cost:%5d',xr1,yr1, t_cost);
```

```
total_cost=total_cost + t_cost;

t_cost=0;



% Positioning the vector (DL2)---------------------------------------

rad_aod2_xy=(aod2_xy*pi)/180;

ro=0.00000; % current angle after one step rotation



% Step-1: Lie down DL2 to x-axis (xr3=DL, yr1=0) angle=0

xr3=dl2_xy;

yr3=0.00000;

if (rad_aod2_xy>pi/2)

sign=-1;

delta=pi+sign*rad_aod2_xy;

else

sign=1;

delta=sign*rad_aod2_xy;

end

% Step-2: Rotate the vector until angle (ro) meets the aod2_xy (delta)

% xr3, yr3 are the coordinates of the Arrow head of the vector DL1

while((delta-ro)>err)
```

```
t_cost=t_cost+1;

oldxr3=xr3;

xr3=xr3-xr3*u-yr3*v;

yr3=yr3-yr3*u+oldxr3*v;

ro=ro+2^-k;

end %while

fprintf('\n\t\t Vector DL2(x,y)=%f is placed with the angle ', dl2_xy);

fprintf('(%f) as %f -->aod2xy(%f), err: %f', aod2_xy, ...

ro, delta,(delta-ro));

fprintf('\n\t\t Calculated arrow head position of DL2');

fprintf('(%f,%f) with cost:%5d',xr3,yr3, t_cost);

total_cost=total_cost + t_cost;

t_cost=0;



% Positioning the vector (DL1_xz)----------------------------------------

rad_aod1_xz=(aod1_xz*pi)/180;

ro=0.00000; % current angle after one step rotation



% Step-1: Lie down DL1_xz to x-axis (xr5=DL1, zr5=0) angle=0

xr5=dl1_xz;
```

```
zr5=0.00000;
if (rad_aod1_xz>pi/2)
sign=-1;
delta=pi+sign*rad_aod1_xz;
else
sign=1;
delta=sign*rad_aod1_xz;
end
% Step-2: Rotate the vector until angle (ro) meets the aod1_xz (delta)
% xr5, zr5 are the coordinates of the Arrow head of the vector DL1_xz
while((delta-ro)>err)
t_cost=t_cost+1;
oldxr5=xr5;
xr5=xr5-xr5*u-zr5*v;
zr5=zr5-zr5*u+oldxr5*v;
ro=ro+2^-k;
end %while
fprintf('\n\t\t Vector DL1(x,z)=%f is placed with the angle ', dl1_xz);
fprintf('(%f) as %f -->aod1xz(%f), err: %f', aod1_xz, ...
ro, delta,(delta-ro));
```

```
fprintf('\n\t\t Calculated arrow head position of DL1_xz');

fprintf('(%f,%f) with cost:%5d',xr5,zr5, t_cost);

total_cost=total_cost + t_cost;

t_cost=0;



%-----------------------------------------------------------------

% Apply the vector breaking algorithm to both DL-1 and DL-2

% until xr3==xr4  (or |xr3-xr4|< error)

% (x of the head of R3 and R4 are going to met each other on MO)

%-----------------------------------------------------------------

fprintf('\n\t (C)Applying the Vector Breaking Algorithms to ');

fprintf('DL-1 and DL-2 to find the 2D position of the MO(x,y)');



lr2x=0.000; % length of R2 on x-axis;

lr2y=0.000; % length of R2 on y-axis;

lr4x=0.000; % length of R4 on x-axis;

lr4y=0.000; % length of R4 on y-axis;



xr2=xr1; %initially headX of both R1 and R2 are pointing the same x

yr2=yr1; %initially headY of both R1 and R2 are pointing the same y
```

```
xr4=xr3; %initially headX of both R3 and R4 are pointing the same x

yr4=yr3; %initially headY of both R3 and R4 are pointing the same y


r2len_x=0.00000000000001; % Accumulated length or R2 in X-axis

r2len_y=0.00000000000001; % Accumulated length or R2 in Y-axis

r4len_x=0.00000000000001; % Accumulated length or R4 in X-axis

r4len_y=0.00000000000001; % Accumulated length or R4 in Y-axis

lp=0;

%fprintf('\n\t\t xr2>?xr4 (%f>%f)', xr2, xr4);

%-- Concurently breaking DL1 and DL2 until they intersect on the MO

while((xr2-xr4)> err)   % while R2 NOT meets R4 continue

if(lp>loop_trsh)

break;

end;

lp=lp+1;

xr1 = xr1 - xs1;

yr1 = yr1 - ys1;

R1=sqrt(xr1^2+yr1^2);              % current R1


xr2 = xr1 + r2len_x; % x value of the head of R2
```

```
yr2 = yr1 - r2len_y; % y value of the head of R2

R2=sqrt((xr2-xr1)^2+(yr2-yr1)^2);% current R2



%fprintf('\n \t xr1:%f, yr1:%f', xr1, yr1);

t_cost=t_cost+246;



while((dl1_xy-(R1+R2))>err)



% Extend the length of R2

% with adding steping vector V12 to r2len_x and r2len_y

r2len_x = r2len_x + xs2;

r2len_y = r2len_y + ys2;

% positioning the head of the R2

xr2 = xr1 + r2len_x; % x value of the head of R2

yr2 = yr1 - r2len_y; % y value of the head of R2

R2=sqrt((xr2-xr1)^2+(yr2-yr1)^2);

%fprintf('\n \t\t xr2:%f, yr2:%f,', xr2, yr2);

if((xr1<0)||(yr1<0)||(xr2<0)||(yr2<0))

break;

end;
```

```
if((xr3<0)||(yr3<0)||(xr4<0)||(yr4<0))

break;

end;

t_cost=t_cost+126;

end;  %-- end of while((Dl1-(R1 +sqrt(xtemp2^2+ytemp2^2)))>err)



%-------- Now, break the DL-2 until y4 meets y2 ----------

while((yr4-yr2)>err)

% Break the R3 one step, extend the R4 as DL2 = R3 + R4

% Substracting the steping vector V21 from R3

xr3 = xr3 - xs3;

yr3 = yr3 - ys3;

R3=sqrt(xr3^2+yr3^2);              % current R3



xr4 = xr3 + r4len_x; % x value of the head of R4

yr4 = yr3 - r4len_y; % y value of the head of R4

R4=sqrt((xr4-xr3)^2+(yr4-yr3)^2);% current R2



%fprintf('\n \t \t \t xr3:%f, yr3:%f', xr3, yr3);

t_cost=t_cost+246;
```

```
while((dl2_xy-(R3+R4))>err)

% Extend the length of R4

% with adding steping vector V22 to r4len_x and r4len_y

r4len_x = r4len_x + xs4;

r4len_y = r4len_y + ys4;

% positioning the head of the R4

xr4 = xr3 + r4len_x; % x value of the head of R4

yr4 = yr3 - r4len_y; % y value of the head of R4

R4=sqrt((xr4-xr3)^2+(yr4-yr3)^2);

t_cost=t_cost+126;

%fprintf('\n \t \t \t \t xr4:%f, yr4:%f,', xr4, yr4);



end;  %-- end of while((Dl2-(R3+R4))>err)

end; % end of while((yr1-yr3)<err)

end; %-- end of while((xr2-xr4)> err)



if ((xr5+2)>xr2)

fprintf('\n\t\t[(!)Sample set is skiped since xr5(%f)>xr2(%f)]', ...

xr5, xr2);

cnt = cnt - 1;
```

```
continue;

end;

%-------------------------------------------------------------

%---- Carrying the 2D solution to 3D, by finding the z-value:

% We will break the DL1_xz until xr5 meets xr2

%-------------------------------------------------------------

r6len_x = 0.000000000000;

r6len_z = 0.000000000000;

xr6 = xr5; % x-value of arrow heads (R5, R6) are equalized

zr6 = zr5; % y-value of arrow heads (R5, R6) are equalized

if(xr6>xr2)

while((xr6-xr2)>err)

xr5=xr5-xs5;

zr5=zr5-zs5;

R5=sqrt(xr5^2+zr5^2);              % current R1_xz

xr6 = xr5 + r6len_x; % x value of the head of R6

zr6 = zr5 - r6len_z; % y value of the head of R6

R6=sqrt((xr6-xr5)^2+(zr6-zr5)^2);% current R6

t_cost=t_cost+246;

while((dl1_xz-(R5+R6))>err)
```

```
% Extend the length of R6

% with adding stepping vector V32 to r6len_x and r6len_z

r6len_x = r6len_x + xs6;

r6len_z = r6len_z + zs6;

% positioning the head of the R6

xr6 = xr5 + r6len_x; % x value of the head of R6

zr6 = zr5 - r6len_z; % y value of the head of R6

R6=sqrt((xr6-xr5)^2+(zr6-zr5)^2);

t_cost=t_cost+126;

end;  %-- end of while((dl1_xz-(R5+R5))>err)

end;  %-- end of while((xr6-xr2)>err)

else

while((xr2-xr6)>err)

xr5=xr5-xs5;

zr5=zr5-zs5;

R5=sqrt(xr5^2+zr5^2);              % current R1_xz

xr6 = xr5 + r6len_x; % x value of the head of R6

zr6 = zr5 - r6len_z; % y value of the head of R6

R6=sqrt((xr6-xr5)^2+(zr6-zr5)^2);% current R6

t_cost=t_cost+246;
```

```
while((dl1_xz-(R5+R6))>err)

% Extend the length of R6

% with adding stepping vector V32 to r6len_x and r6len_z

r6len_x = r6len_x + xs6;

r6len_z = r6len_z + zs6;

% positioning the head of the R6

xr6 = xr5 + r6len_x; % x value of the head of R6

zr6 = zr5 - r6len_z; % y value of the head of R6

R6=sqrt((xr6-xr5)^2+(zr6-zr5)^2);

t_cost=t_cost+126;

end;  %-- end of while((dl1_xz-(R5+R5))>err)

end; %while(xr2-xr5>err)

end; % end of else

total_cost=total_cost + t_cost;
```

```
if(abs(mx - xr2)< abs(mx - xr4))

err_x = abs(mx - xr2);

else

err_x = abs(mx - xr4);

end;

t_err_x = t_err_x + err_x;



if(abs(my - yr2)< abs(my - yr4))

err_y = abs(my - yr2);

else

err_y = abs(my - yr4);

end;

t_err_y = t_err_y + err_y;



err_z = abs(mz - zr6);

t_err_z = t_err_z + err_z;

rmse = sqrt((err_x^2+err_y^2+err_z^2)/3); % Root Mean Square Error

t_rmse = t_rmse + rmse;

fprintf('\n\t\t => MO(%f,%f,%f) estimated with the cost:%f ', ...

mx, my,mz, t_cost);
```

```
fprintf('\n\t\t => R2(%f,%f,%f) <-> R4(%f,%f,%f)', ...

xr2,yr2,zr6, xr4, yr4, zr6);

fprintf('Avg MO(x,y,z)=(%f,%f,%f)',((xr2+xr4)/2), ((yr2+yr4)/2), zr6);

fprintf('\n\t\t => err-1(%f,%f,%f); ', mx-xr2, my-yr2, mz-zr6);

fprintf('err-2(%f,%f,%f); ',mx-xr4, my-yr4, mz-zr6);

fprintf('avg err(%f, %f, %f) ',(mx-(xr2+xr4)/2), ...

(my-(yr2+yr4)/2), mz-zr6);

fprintf('\n\t\t Total Cost: %10d\t RMSE:%f meters\n',total_cost,rmse);

tt_cost = tt_cost + total_cost;

t_cost=0;

x1=xr2; y1=yr2; x2 = xr4; y2=yr4;

fprintf(fw,'%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t
%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8d\n', ...

mx,my,mz,xr2,yr2,xr4,yr4,xr6,zr6,x1,y1,x2,y2,err_x,err_y,err_z, rmse,total_cost);

end; % end of while ~feof(fr)

cnt = cnt - 1;

fprintf('\n\n Mean Error(x,y,z):%f, %f, %f \t m-RMSE: %f \t Total Cost:
%f\n',t_err_x/cnt, t_err_y/cnt, t_err_z/cnt, t_rmse/cnt, tt_cost/cnt);

fprintf(fw,'\n Average Errors: (mx-x1): %f, (my-y1): %f, (mz-z3): %f, m-RMSE: %f,
(Avg-Cost: %f \n',t_err_x/cnt, t_err_y/cnt, t_err_z/cnt, t_rmse/cnt, tt_cost/cnt);
```

```
fclose(fr);

fclose(fw);



load gong

sound(y,Fs)

h = msgbox('Operation Completed----');
```

Appendix C: MATLAB Source Code of SBS_NLOS_Dynamic Algorithm

```
%One BS and AOD-TDOA-Dynamic Angle Rotation Method - 3D Positioning

clc;

clear;

format long;

%-------- Definitions ---------------------------------------------------

k_start = 7;

k_len=11;

step_val=1; % Stepping value in meters

turn_allowed=100;   % Number of maximum sample rows will be processed

s_date= datestr(now, 'yyyymmdd');

samplesetfile='SAMPLESET-20220927-SS1000-d.txt';

cnt = 0;            % Counter for reading sample rows

simresult_filename = strcat('g:simres-dynamic-',samplesetfile(11:25),'.txt');



%---- For Statistics -----------------------------------------------------

t_err_x= 0.0000;

t_err_y= 0.0000;

t_err_z= 0.0000;
```

```
t_rmse = 0.0000;

t_cost = 0.0000;

total_cost= 0.00000;

tt_cost= 0.00000;

fprintf('\n 01 - Parameters are defined and initialized SUCCESSFULLY !!!\t');


fr=fopen(samplesetfile,'r');

fprintf(strcat('\n 02 - File of sample sets (',samplesetfile,') is opened for reading'));


fw=fopen(simresult_filename,'w');

fprintf(fw,'%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t%8s\t
%8s\t%8s\t\n','  X','  Y','  Z','  x_M','  y_M','  z_M','  dl_xy','  dl_xy','
aod_xy','  aod_xz','Err X  ','Err Y  ','Err Z  ','  RMSE  ','(-COST-)');

fprintf(strcat('\n 03 - Out file \t',simresult_filename,' is created!!!'));


fprintf('\n 04 - Reading and processing the sample sets');

%----- Reading ----------------------------------------------------------

while ~feof(fr)

total_cost=0.00000;

t_cost=0.00000;
```

```
cnt=cnt+1;

if(cnt>turn_allowed)

break;

end; % end of if(cnt>turn_allowed)

fprintf('\n\t Sample row %3d is in process ...',cnt);

fprintf('\n\n\t\t\t (1) Reading current sample row ...',cnt);



bx=fscanf(fr,'%f',1);

by=fscanf(fr,'%f',1);

bz=fscanf(fr,'%f',1);



mx=fscanf(fr,'%f',1);

my=fscanf(fr,'%f',1);

mz=fscanf(fr,'%f',1);



dl_xy=fscanf(fr,'%f',1);

dl_xz=fscanf(fr,'%f',1);



aod_xy=fscanf(fr,'%f',1);

aod_xz=fscanf(fr,'%f',1);
```

```
fprintf('\n\t\t\t\t B(%f,%f,%f))-->M(%f,%f,%f))',bx,by,bz,mx,my,mz);

fprintf('\n\t\t\t\t dl_xy[%f], aod_xy[%f], dl_xz[%f], aod_xz[%f]', dl_xy, aod_xy,
dl_xz, aod_xz);


fprintf('\n\n\t\t\t (2) Lengthening and rotating the arrow head (x,y)...');


% PLACING THE STEPPER VECTORS ON THE GRAPH

% (1) The stepping_value (dl) is laid on x-axis, and then rotated

% until it reaches to the angle rad_aod_xy

% x1, y1 are the coordinates of the vector-1 head

rad_aod_xy=(aod_xy*pi)/180;

ro=0.00000; % current angle after one step rotation

%xs=step_val;

xs=dl_xy*1.00000; % Lay down the dl_xy on to x-axis and than rotate it

ys=0.00000;

if (rad_aod_xy>pi/2)

sign=-1;

delta=pi-rad_aod_xy;

else

sign=1;
```

```
delta=rad_aod_xy;

end

% delta: targeted angle in radian



% Rotating the vector until it reaches to delta

k=k_start;

err =10^-k;

while((delta-ro)>err)

t_cost=t_cost+1;

if(k>11) break;

end;  % if(k>11)

v = 2^-k;

u = 2^-(2*k+1);

err =10^-k;

txs = xs;

tys = ys;

tro = ro;



oldx=xs;

xs=xs-xs*u-ys*v;
```

```
ys=ys-ys*u+oldx*v;

ro=ro+2^-k;



if((delta-ro)<err)

%----- Rollback the last iteration and increase the k -------

xs = txs;

ys = tys;

ro = tro;

k=k+1;

t_cost = t_cost + 1;

end;  % if((delta-ro)<err)

end % while((delta-ro)>err)



fprintf('\n\t\t\t\t Stepping vector rotated with angle %f-->%f, err: %f',ro,delta,(delta-
ro));

fprintf('\n\t\t\t\t Calculated Arrowhead(x,y) vs MO(x,y) => (%f,%f) vs (%f,%f)
',xs,ys, mx, my);

fprintf('\n\t\t\t\t Error  is (%f,%f)',abs(mx-xs),abs(my-ys));

fprintf('\t\t\t\t Calculated Cost: %5d',t_cost);

total_cost=total_cost + t_cost;
```

```
t_cost=0;

%------ End of Lengthening and rotating arrowhead (x,y) processes ------


fprintf('\n\n\t\t\t (3) Lengthening and rotating the arrow head (x,z)...');


% PLACING THE STEPPER VECTORS ON THE GRAPH

% (1) The stepping_value (dl) is laid on x-axis, and then rotated

% until it reaches to the angle rad_aod_xz

% x1, y1 are the coordinates of the vector-1 head

rad_aod_xz=(aod_xz*pi)/180;

ro=0.00000; % current angle after one step rotation

%xs=step_val;

xs2=dl_xz*1.00000; % Lay down the dl_xy on to x-axis and than rotate it

zs=0.00000;

if (rad_aod_xz>pi/2)

sign=-1;

delta=pi-rad_aod_xz;

else

sign=1;

delta=rad_aod_xz;
```

```
end

% delta: targeted angle in radian



% Rotating the vector until it reaches to delta

k=k_start;

err =10^-k;

while((delta-ro)>err)

t_cost=t_cost+1;

if(k>11) break;

end;  % if(k>11)

v = 2^-k;

u = 2^-(2*k+1);

err =10^-k;

txs = xs2;

tzs = zs;

tro = ro;



oldx=xs2;

xs2=xs2-xs2*u-zs*v;

zs=zs-zs*u+oldx*v;
```

```
ro=ro+2^-k;


if((delta-ro)<err)
%----- Rollback the last iteration and increase the k -------
xs2 = txs;
zs = tzs;
ro = tro;
k=k+1;
t_cost = t_cost + 1;
end;  % if((delta-ro)<err)
end %while


fprintf('\n\t\t\t\t Stepping vector rotated with angle %f-->%f, err: %f',ro,delta,(delta-
ro));
fprintf('\n\t\t\t\t Calculated Arrowhead(x,z) vs MO(x,z) => (%f,%f) vs (%f,%f)
',xs2,zs, mx, mz);
fprintf('\n\t\t\t\t Error  is (%f,%f)',abs(mx-xs2),abs(mz-zs));
fprintf('\t\t\t\t Calculated Cost: %5d',t_cost);
total_cost=total_cost + t_cost;
t_cost=0;
```

```
%------ End of Lengthening and rotating arrowhead (x,y) processes ------


if(abs(mx-xs)<abs(mx-xs2))

t_err_x= t_err_x + abs(mx-xs);

err_x = abs(mx-xs);

else

t_err_x= t_err_x + abs(mx-xs2);

err_x = abs(mx-xs2);

end;

t_err_y= t_err_y + abs(my-ys);

err_y = abs(my-ys);

t_err_z= t_err_z + abs(mz-zs);

err_z = abs(mz-zs);



rmse = sqrt((err_x^2+err_y^2+err_z^2)/3); % Root Mean Square Error

fprintf('\n\n\t\t\t\t\t\t Total Cost: %5d\t RMSE:% f\n',total_cost,rmse);

t_rmse = t_rmse + rmse;
```

```
fprintf(fw,'%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\t
%8.5f\t%8.5f\t%8.5f\t%8.5f\t%8.5f\n',xs, ys, zs, mx,my,mz,dl_xy, dl_xz, aod_xy,
aod_xz,err_x,err_y,err_z, rmse,total_cost);

tt_cost=tt_cost + total_cost;

end; % end of while ~feof(fr)


fprintf('\n\n Mean Error(x,y,z):%f, %f, %f \t m-RMSE: %f \t Total Cost:
%f\n',t_err_x/cnt, t_err_y/cnt, t_err_z/cnt, t_rmse/cnt, tt_cost/cnt);

fprintf(fw,'\n Average Errors: (mx-x1): %f, (my-y1): %f, (mz-z3): %f, m-RMSE: %f,
(Avg-Cost: %f \n',t_err_x/cnt, t_err_y/cnt, t_err_z/cnt, t_rmse/cnt, tt_cost/cnt);


fclose(fr);

fclose(fw);


load gong

sound(y,Fs)

h = msgbox('Operation Completed----');
```