A Hardware Oriented Algorithm for 3D AOA Mobile Positioning

Marwan Hasan

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the Requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
June 2014
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işik Aybay
Chair, Department of Computer Engineering

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 of Science in Computer Engineering.

Asst. Prof. Dr. Emre Özen
Co-Supervisor

Assoc. Prof. Dr. Mustafa İlkan
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Mustafa İlkan _____

2. Assoc. Prof. Dr. Muhammad Salamah _____

3. Asst. Prof. Dr. Gürcü Öz _____

4. Asst. Prof. Dr. Emre Özen _____

5. Asst. Prof. Dr. Ahmet Ünveren _____

ABSTRACT

The determination of a mobile object's location in a cellular network becomes very important with the new US Federal Communication Commission (FCC) standards regarding the wireless Enhanced 911 (E911) emergency calling systems. Most commonly used methods for location can be addressed as Time of Arrival (TOA), Time difference of Arrival (TDOA) and Angle of Arrival (AOA). The importance of finding the location of a mobile object in 3D becomes much more important especially after Federal Communication Commission's announcement which asks about the vertical position estimation of a mobile object.

There exist general approaches to simple algorithms for 2D positioning techniques in cellular networks and adhoc networks. Such an approach is missing for 3D positioning. The aim of this project is via using AOA signal measurement technique to form a simple algorithm for 3D positioning that could be implemented both as hardware and software.

Four new 3D AOA algorithms; MEM-1, MEM-2, DMEM-1 and DMEM-2 are proposed in this study. At the end of simulation from the results it is clear that our proposed algorithms outperform the traditional algorithm in terms of computational cost and execution simplicity.

Keywords : Positioning algorithm, Angle of Arrival, 3D positioning.

ÖZ

ABD Federal İletişim Komisyonunun (FİK) kablosuz genişletilmiş acil çağrı (E911) standartlarında değişikliğe gitmesi ile mobil cihazların hücresel ağlar içerisindeki yerlerinin belirlenmesi çok önemli bir hal almıştır. Konum bulmak için yaygın olarak kullanılan yöntemler; varış zamanı (VZ), varış zamanı farkı (VZF) ve varış açısı (VA) olarak listelenebilir. Mobil cihazların konumunu üç boyutlu olarak bulma, FİK'nun mobil cihazların dikey konum tahmininin gerekli olduğunu açıklamasından sonra önem kazanmıştır.

Halihazırda hücresel ağlar içerisindeki mobil cihazların yerlerinin iki boyutlu olarak belirlenmesini sağlayan algoritmalar mevcuttur. Fakat üç boyutlu yer belirlemede kullanılabilecek algoritmalar yoktur. Bu çalışmanın amacı, VA yöntemini kullanarak üç boyutlu yer belirlemede kullanılabilecek bir algoritma geliştirmektir.

Bu çalışmada MEM-1, MEM-2, DMEM-1 ve DMEM-2 adlarında dört tane yeni algoritma geliştirilmiştir. Bu algoritmalar VA yöntemini kullanarak mobil cihazların yerlerinin üç boyutlu olarak bulunmasını sağlamaktadırlar. Simülasyon sonuçları geliştirilen algoritmaların geleneksel olandan daha verimli çalıştığını göstermektedir.

Anahtar kelimeler : konum bulma algoritması, varış açısı, üç boyutlu yer belirleme

To Family

ACKNOWLEDGMENT

I am deeply indebted to my Assoc. Prof. Dr. Mustafa İlkan for the continuous support for my master study and thesis. I could not have imagined having a better advisor from him for my master study.

I would like to thank for my co-supervisor Asst. Prof. Dr. Emre Özen for his help and valuable comments.

My special thanks for Prof. Dr. Işık Aybay chairman of the Department of Computer Engineering for his help.

Words cannot express my gratefulness to my dear family for their support to me for travel from Iraq to North Cyprus for my study. Thanks for all of them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AOA	Angle of Arrival
BS	Base Station
DMEM-1	Dynamic Mustafa Emre Marwan One
DMEM-2	Dynamic Mustafa Emre Marwan Two
FCC	Federal Communication Commission
FPGA	Field Programmable Gate Array
GPS	Global Positioning System
LOB	Line of Bearing
LOS	Line of Sight
MEM-1	Mustafa Emre Marwan -1
MEM-2	Mustafa Emre Marwan -2
MO	Mobile Object
MS	Mobile Station
PSTN	Public Switched Telephone Network
RISA	Reduced Instruction Set Computer
SDB	Salamah Doukhnitch Bayramer
SMLC	Service Mobile Location Center
TDOA	Time Difference of Arrival
TOA	Time of Arrival

Chapter 1

INTRODUCTION

In recent years Global Positioning system (GPS) has attracted a considerable attention as it is useful for the purposes of system administration and emergency situations that depend on knowing the exact position of mobile devices that are vital for many services. GPS services can be used in security, tour guide, emergency rescue and location based billing and other applications. [1]

GPS is useful for specifying the position of a mobile object. In the issue of location-sensitive billing, network operator can provide distinctive rates depending on the location of the mobile phone. This neglects the usage of Public Switched Telephone Network (PSTN) to offer competitive prices for variant calls and provides accurate information to network operators. From the standpoint of security, it also enables automatic determination of position information in emergency cases similar to those available for fixed phones calls, and this improve response time and efficiency used in resources.

The position of a mobile station (MS) can be determined through the use of GPS. This can be used in dispatching fleets of vehicles and detecting traffic congestion. And also can be used in Intelligent Transport Systems (ITS).

Estimation to the position of a MS in wireless cellular networks is an important issue for the network performance enhancement. Long-term MS positioning can be monitored to provide inputs for better cellular network planning and in the short-term for matching the requirements of Federal Communications Commission (FCC) to locate 911 emergency calls.

In the middle of 1996, the FCC pronounced its delegation to perform gradually increasing emergency service for cellular system users. That could be referred to Position Location site (PL) information which permits agencies to supply quick and efficient emergency service to help anyone in need. For that, in mid 90's, FCC handled three phase regulations under position determination E-911 service. In the first phase, the providers of this service can verify transmitting Mobile Identification Number (MIN) from those who are demanding 911 call ignoring all validation and charge cases, this phase was applied one year after FCC regulation settlement. In the second phase, service providers must supply a technique that includes cell site location for matching callers' phone numbers. In the third phase, the service providers could fulfill the goal of determining the location of mobile users who are seeking 911 calls with at least 67 percent of success throughout a circle of a 125 meters radius.

In order to locate cellular mobile station, two main classes are used; network based where one or more (BS) receive (MS) signals and determine the location inside the server. Because of the large number of messages between (BS) and (MS), the network is expensive even though the handsets don't have to be changed. The second class is the mobile based, where the mobile station reports its position based on measurements made form received signals. The position is reported via a wireless

network to the service provider. These methods support high position accuracy although legacy phones cannot be used. Different techniques can be used to determine the location of a MS in any cellular system. Positioning measurement can be done via time of arrival (TOA), time difference of arrival (TDOA), angle of arrival (AOA), cell identity (Cell-ID) and more [2], [3], [4]. However, AOA, TOA and TDOA methods can work with the existing cellular infrastructure without using any extra investments; otherwise, they are low cost methods that simplify integration through a high accuracy. AOA is a method which includes two base stations, where smart antennas can be used at the base/mobile station to determine angle of the received information.

This study focuses on the AOA technique which can be addressed as a low cost method with more applications. And also this thesis is devoted to the new location estimation algorithms in 3D that can be implemented as hardware.

There is a considerable advantage in selecting AOA which depends on wireless location system used for carriers of cellular and personal services because there is no need to change the existing MS which surpass 50 million. Aside from tracking the MS, it is possible to monitor the movement of reverse traffic channel.

Consequently, the background of positioning techniques and traditional method are reviewed in Chapter 2. Chapter 3 contains the proposed hardware-oriented 3D positioning algorithms for AOA technique. Simulation results and comparisons between the traditional and the proposed techniques have been presented in Chapter 4. Finally, conclusion is revealed in Chapter 5.

Chapter 2

POSITIONING TECHNIQUES

There are a number of technical alternatives for assigning positions of the user equipment. Most of the traditional techniques use site attributes of mobile signal for location, such as; 1) the identity of the cell (cell-ID), 2) the time of arrival (TOA), 3) the time difference of arrival (TDOA), and 4) the angle of arrival (AOA), which will be discussed in this chapter.

2.1 Cell-ID

Estimating the location of a mobile device drive by determining the location of the base station or antenna can be considered as the most basic and simple way of assigning locations throughout cell areas, this is demonstrated in Figure 2.1. Cell-ID is the simplest form of GPS technology, which focuses on the network. In spite of the bad locating accuracy of this method; it is simple, fast and cheap.

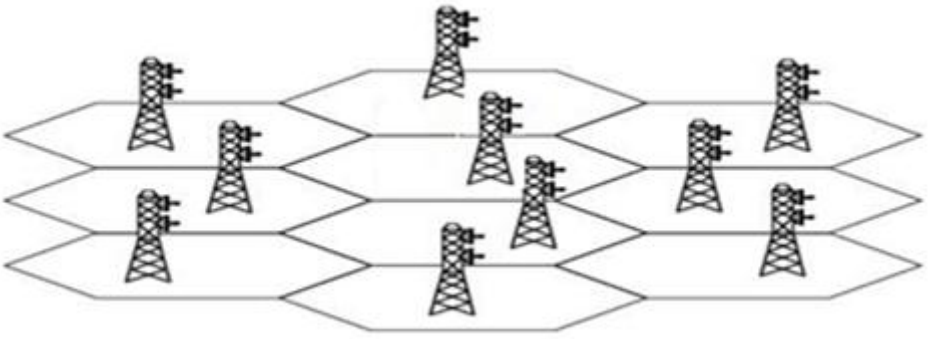

Figure 2.1: Cell-ID Positioning Method

For positioning method using Cell-ID, location measurement of MS's position could be conducted by registering MS to the assigned cell depending on the information available in the network and the MS. The geographic location of a Cell-ID can be determined with the use of the network operator knowledge. This is known as mast-based location which uses coverage database at the Serving Mobile Location Center (SMLC). Mobile network always knows the whereabouts of the mobile phone that is registered to region level, during a call it can determine the intended cell among the cells in the region communicating with this phone. Data base and MS roaming subscribers can be supported by estimating the mobile object's location done by cell center. According to the accuracy level, the cell size can be determined. The simplest shape of positioning technology is the network centric. The main disadvantage is that the size of the cells could be different between rural and urban areas, leading to changes in the position accuracy.

2.1.1 Time of Arrival (TOA) Positioning Algorithms

TOA technique finds the location of moving objects building at the intersection of measuring the distance from the base stations BS using circles group relying on the proportional relation of radio wave propagation time with its converse [11].

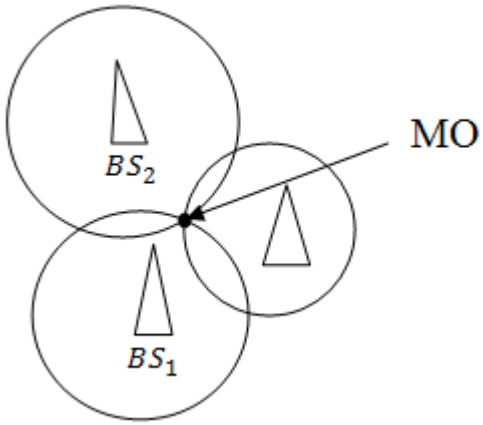

Figure 2.2: Time of Arrival

In order to determine the location exactly, there is a need for three base stations of measurements, that is demonstrated in Figure 2.3. The circles can be used in 2D space as areas in 3D space.

The range between BS_i and the MO can be expressed as

$$D_i = \| x_i - x_{MO} \| = \sqrt{(x_i - x_{MO})^2 + (y_i - y_{MO})^2} \quad (2.1)$$

Where x_i, y_i, z_i are known as coordinates of the i^{th} base station and D_is are the range estimations respectively. The Mobile Object (MO) location which will be estimated is noted as x_{mo}, y_{mo} and z_{mo} for x, y, z coordinates. Then via combining and substituting the formulation for (2.2), x_{mo}, y_{mo} and z_{mo} could be estimated.

The formulation could be as follow :

$$D_i = \sqrt{(x_i - x_{MO})^2 + (y_i - y_{MO})^2 + (z_i - z_{MO})^2} \quad (2.2)$$

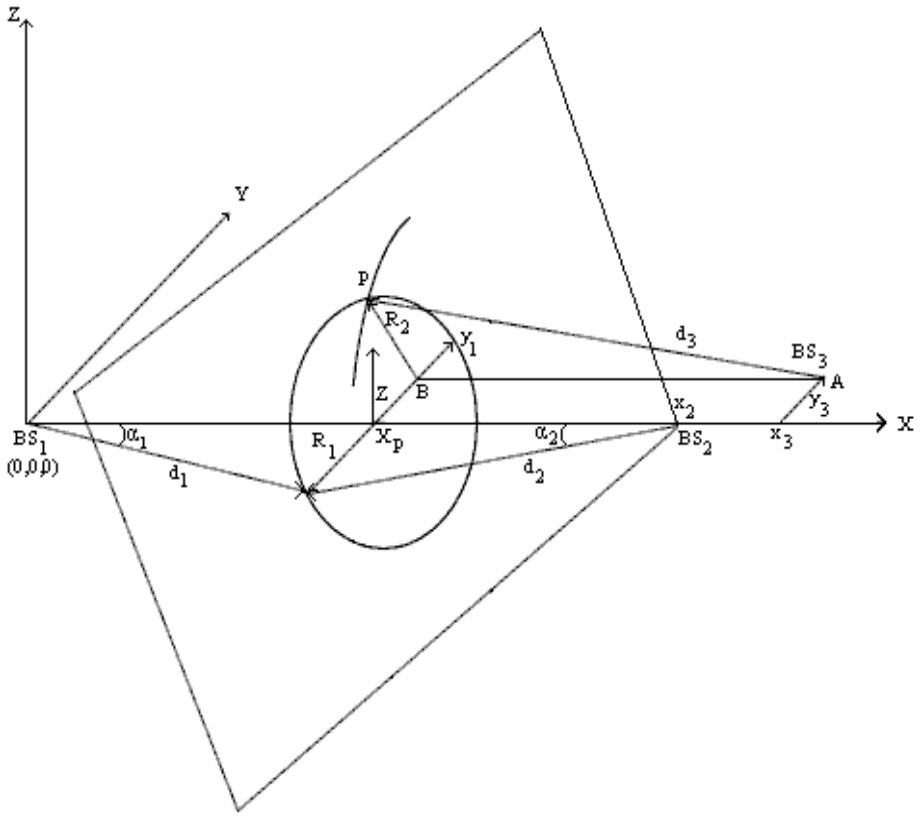

Figure 2.3: TOA 3D Position[11]

Assume that the coordinates of BS_1, BS_2 and BS_3 in a local right-handed orthogonal coordinate system are (X_1,Y_1, Z_1), (X_2,Y_2, Z_2) and (X_3,Y_3, Z_3), respectively. The axis is parallel to a station baseline, and another axis is orthogonal to the two station baseline, or station plane. Therefore according to the TOA from each of the three base stations, the mobile object (MO) position (X_p, Y_p, Z_p) is the intersection of the three spheres centered at BS_1 $(X_1, 0, 0)$, BS_2 $(X_2, Y_2, 0)$ and BS_3 $(X_3, Y_3, 0)$ with radiuses d_1, d_2 and d_3, respectively[11]. The conventional algorithm for 3-D positioning which can be decomposed into a sequence of 2-D rotations.

2.1.2 Time Difference of Arrival (TDOA) Positioning Algorithm

Instead of absolute time, this method relies on using time difference of arrived signals. For example, signal can be received by three antennas at variant times while it can be sent out at unknown time by a mobile device, as in Figure 2.4.

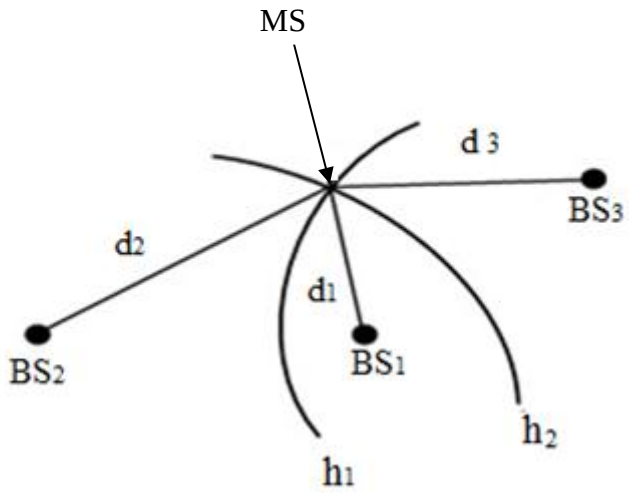

Figure 2.4: TDOA Positioning Method

The TDOA is known as a hyperbolic system because a hyperbolic curve can be addressed by two base stations related to the constant distance difference due to time difference. Also TDOA does not require any time reference point to determine Round Trip Delay (RTD), while TOA does. Estimating the time difference of arriving signal from the source at multiple base stations is the core work of TDOA.

The TDOA can be referred as Hyperbolic Position Location (HPL) method where the intersection of two hyperbolas specifies the position. When another hyperbola which defines another base station takes place in the process, then the intersection of hyperbolas results in location estimation of MS by utilizing both pairs of BSs for that. Consequently, the function of relative BS geometric locations can determine the accuracy of the system.

2.1.3 Angle of Arrival (AOA) Positioning Algorithm

The Angle of Arrival technique that determines the position of a mobile object by utilizing triangulation for position estimation by knowing the angles between the mobile station and two base station in the Line of Sight (LOS) as shown in Figure 2.5. AOA also does not demand precise timing signal on each side.

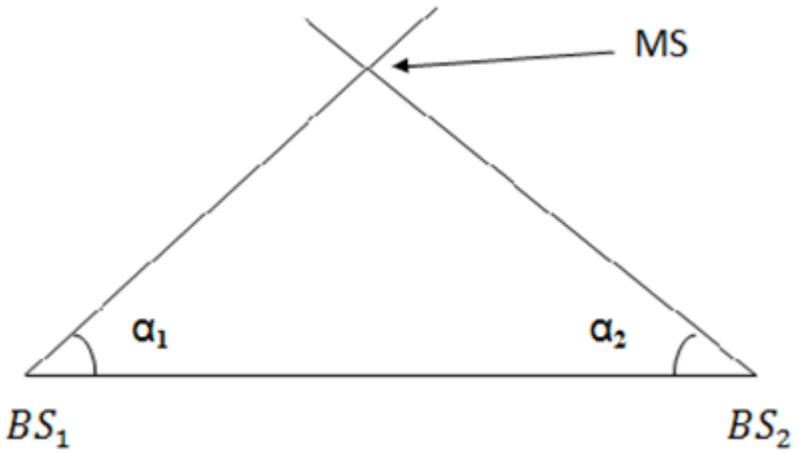

Figure 2.5: AOA Positioning Method

In mobile radio systems, because of the hardness of employing antenna arrays in a mobile station, the antenna arrays are typically located at base station. Those antennas or specifically directional antennas are usually used to measure AOA. Throughout the current technology, antenna array could be located in the BS properly. Otherwise, AOA is measured in the up link even the angle measurements are performed in the uplink or downlink. The measurement of AOA limits the source location along a line in the direction of the angle that connects the BS and the MS which is called Line of Bearing (LOB).

2.1.4 Traditional AOA Algorithm

The angle of arrival depend on 2-D position technique of location is based on lines coming from the base stations which determines the position of the mobile station (MS) at the position of intersection using bearings. Consider(X, Y) as the mobile objects position that will be obtained as shown in Figure 2.7. The coordinates of the MS-vector can be calculated using X_1 which is the absolute distance between MS and BS_1, the 1st vector angle becomes α_1. X_2 becomes the displacement out dated in distance separating the base station account $BS_2$and the X coordinates of MS, and α_2 is defined as the angle of the vector.

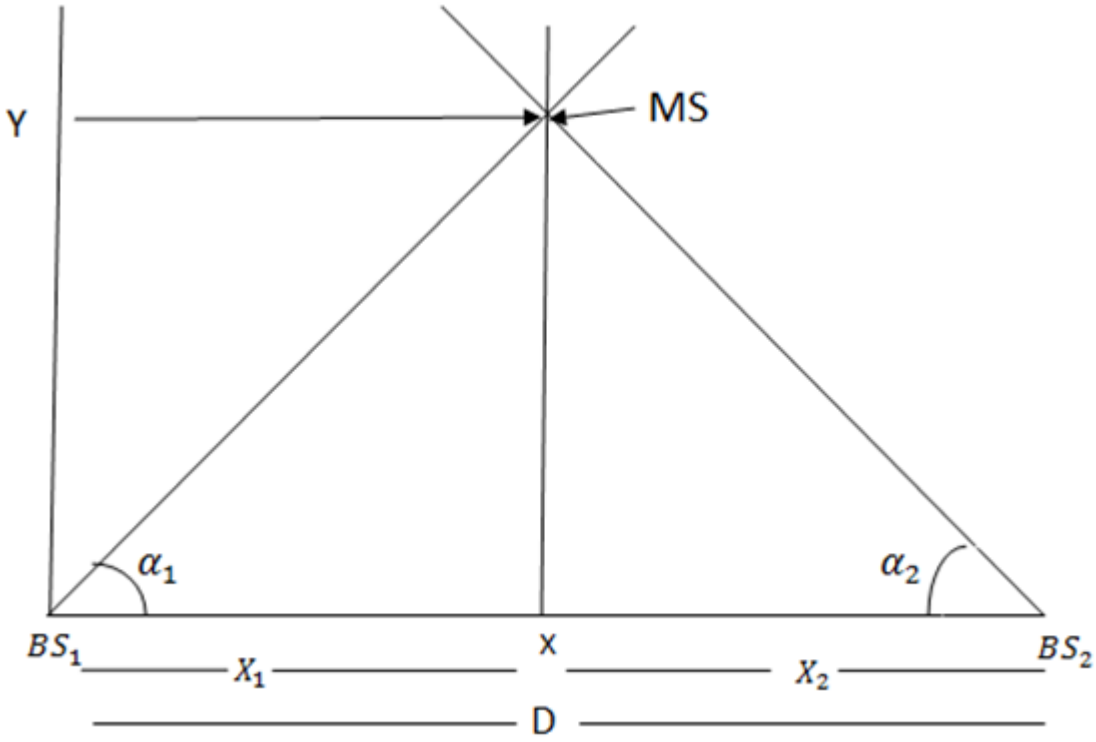

Figure 2.6: Traditional AOA Algorithm

Using simple equations it can be written as :

$Y_1 = X_1 . \tan \alpha_1$ (1)

$Y_2 = X_2 . \tan \alpha_2$ (2)

$D = x_1 + x_2$ (3)

So base station one and base station two, the distance between them is D.

then, equation (3) is written as follow as:

$x_1 = D - x_2$ (4)

By deducting (2) from (1), equation (5) is obtained :

$X_1 . \tan \alpha_1 - X_2 . \tan \alpha_2 = 0$ (5)

Substituting equation four into equation five gives :

$X_2 = (D . \tan \alpha_1) / (\tan \alpha_1 - \tan \alpha_2)$ (6)

After determining X_2, X_1 (that is allocation of MS at x-axis) which is calculated from equation four, and then y is calculated from equation one or two.

2.1.5 SDB Algorithm

The algorithm is based on AOA which will use the same origin of information as the conventional one [12]. Solely subtract, add and shift operations have been used to find the mobile location without implementing any trigonometric equation. Therefore it can be carried out via hardware using, for instance, FPGA.

Mainly the algorithm is formed from two parts, in the first part, parallel and circular rotations of the vectors to determine the parameters cos (α_1), sin (α_1), cos (α_2), sin (α_2). Whereas parallel vector rotation and parallel incrementation are used in the second part to find the MS position. For finding the sin and cos of α_1 and sin and cos of α_2, parallel circular rotations of vectors can be used as stated in Figure 2.7 below :

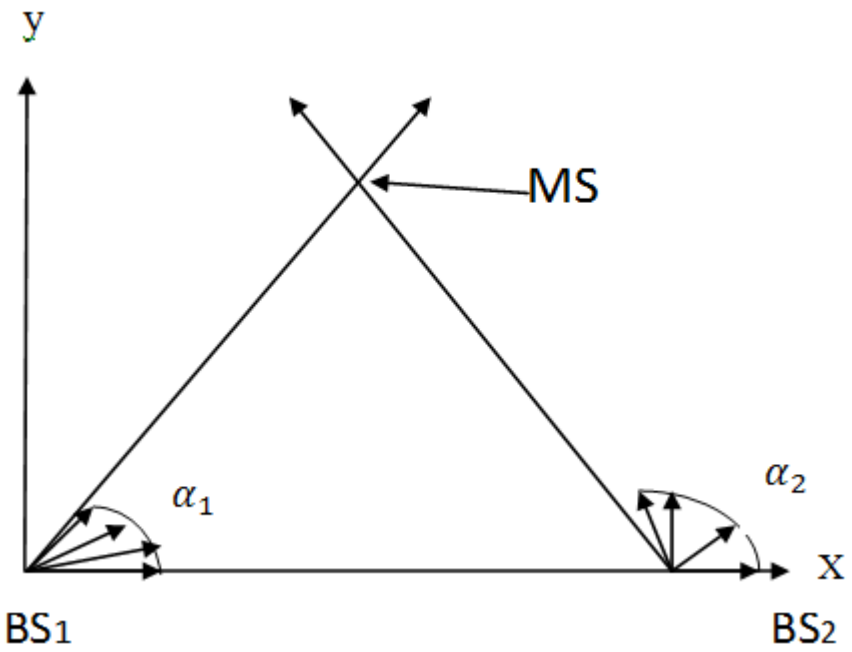

Figure 2.7: Circular Vector Rotation

The circulation rotation matrix M can be derived from equation (5) and is written as

$$\mathbf{M} = \begin{pmatrix} \cos\alpha & \pm\sin\alpha \\ \pm\sin\alpha & \cos\alpha \end{pmatrix} \tag{7}$$

The step rotation angle in radians which is α. Taking the sin function as :

$$\text{Sin } \alpha = 2^{-k} = v \tag{8}$$

Since

$$cos^2\alpha + \sin^2\alpha = 1 \tag{9}$$

Then it is possible to denote the *cos* function as:

$$\cos\alpha = \sqrt{1 - \sin^2\alpha} = \sqrt{1 - 2^{-2k}} \tag{10}$$

Then, it can be written as :

$$\sqrt{1 - 2 - 2k} \leq 1 - 2^{-2k-1} \tag{11}$$

Inequality (11) correct where is squaring for each sides gives :

$$1 - 2^{-2k} \leq 1 - 2.\ 2^{-2k-1} + 2^{-4k-2}$$

$$= 1 - 2^{-2k} + 2^{-4k-2}$$

For k more than or equal to four, one acquire $2^{-4k-2} \leq 2^{-18} < 10^{-5}$. However, can be an accuracy of ε = 10^{-5}, cos σ could converge as :

Cos σ = 1– 2^{-2k-1} = 1– $2^{-(2k+1)}$ = 1- u (12)

Therefore, vector coordinates are repeatedly rotated for using matrix (7) So, the rotation equations for the first vector are written as :

$x_{1,i+1}$ = $x_{1,i}$ - $x_{1,i}u$ - $y_{1,i}v$ (13)

$y_{1,i+1}$ = $y_{1,i}$ - $y_{1,i}u$ + $x_{1,i}v$ (14)

Where σ_i is the step rotation angle with initial values which is $x_{1,0}$ = 1, $y_{1,0}$ = 0. So the cumulative angle is written as **:**

σ_{i+1} = σ_i+ 2^{-k} (15)

The stop condition which is the 1st rotation will be :

$\Delta\sigma_1$ = σ_1 - $\sigma_{i+1} \leq \varepsilon$ (16)

And the last iterated $x_{1,i+1}$ = cos (α_1), and $y_{1,i+1}$ = sin (α_1).

Likely, the stop condition which is the second rotation where the initial values are $x_{2,0}$ =1, $y_{2,0}$ = 0, will be as follows :

$\Delta\ \alpha_2$ = (α_2 – π/2) - $\alpha_{i+1} \leq \varepsilon$ (17)

And the last iteration $x_{2,i+1} = sin\ (\alpha_2)$, $y_{2,i+1}$ = cos (α_2).

The second part of the algorithm can be implemented after finding the sin and cos of α_1 and sin and cos of α_2. The vital idea is the incremental length of the vector as shown in Figure2.8. This vector holds polar coordinates (R, α) and increase factor of ± 2^{-k} which can be increased in length R. Also its orthogonal coordinates are modified and will be written as :

x_{i+1} = x_i + s 2^{-k} cos α (18)

y_{i+1} = y_i + s 2^{-k} sin α (19)

Where s denotes the incremental factor. For each given $2^{-k}\cos(\alpha)$ and $2^{-k}\sin(\alpha)$ can be calculated.

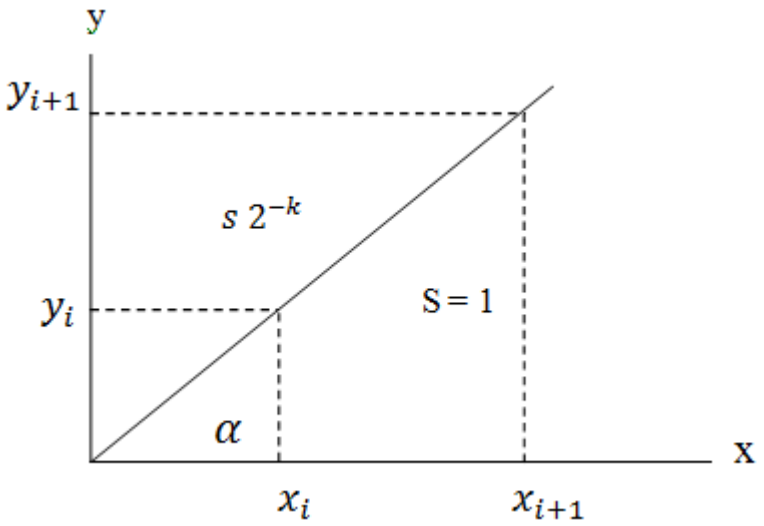

Figure 2.8: Vector Length Incrementation

Figure 2.9, below shows the flowchart that demonstrates the procedure of finding the mobile position by the proposed algorithm.

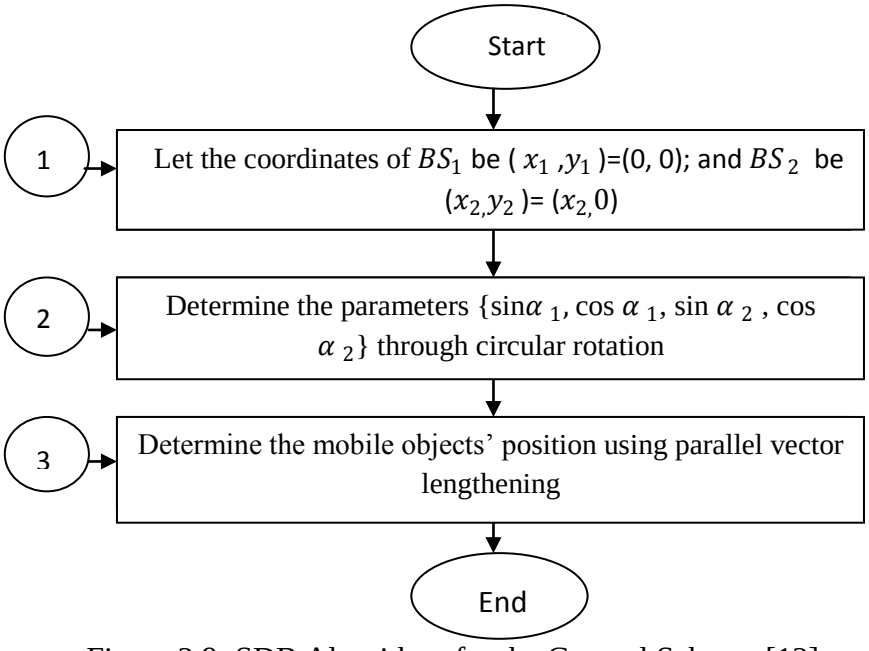

Figure 2.9: SDB Algorithm for the General Scheme [12]

As illustrated in Figure2.10, the coordinates of both BS_s which could be set down as the point of origin to implement parallel vector increments. The vectors are increased for both d_1 and d_2 until their heads intersects.

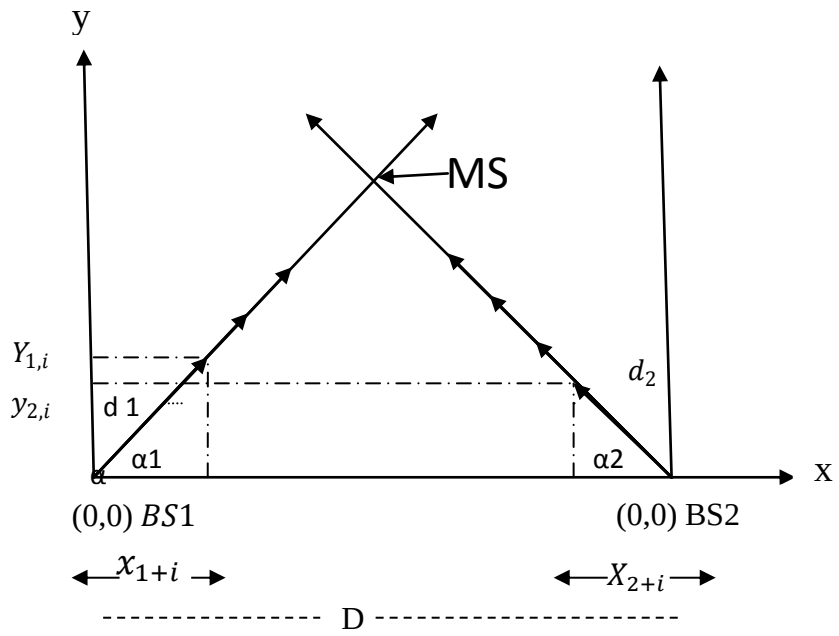

Figure 2.10: Parallel Incrementation of Vectors

Before the beginning of operation, it can be check to find the one with smaller angle (i.e min $\{\pi - \alpha_2\}$), because it needs more increments. This is because of the equation (18), (19), and the vectors where y coordinates of the vectors heads after lengthening depends on α_1 and α_2.Thus, if the value $(\pi - \alpha_2)$ is proposed as the greatest, the conditions of vector lengthening will be as follows :

While (| D – ($x_{1,i}$ + $x_{2,i}$) |) > ε
Increment d_2
While ($y_{2,i}$ - $y_{1,i}$) > ε)
Increment d_1
End while
End while

The new coordinates of the first vector after the first step from the lengthening are :

$x_{1,i+1} = x_{1,i} + s_i$. ΔD. cos α_1 (20)

$y_{1,i+1} = y_{1,i} + s_i$. ΔD. sin α_1 (21)

After one step of lengthening, the coordinates of the vector, similarly for the second vector will become as follows :

$x_{2,i+1} = x_{2,i} + s_i$. ΔD . cos α_2 (22)

$y_{2,i+1} = y_{2,i} + s_i$. ΔD . sin α_2 (23)

Where ΔD is 2^{-k}, and s_i is a sign parameter (i.e. s_i = -1). In order to speed up the process, the commencing values which correspond to the parallel increments could be written as :

$x_{1,0}$ = 0.5 cos (α_1),

$y_{1,0}$ = 0.5 sin (α_1),

$y_{2,0}$ = 0.5 cos (π - α_2),

$y_{2,0}$ = 0.5 sin (π - α_2),

Also s_i = sign (D - ($x_{1,0}$ + $x_{2,0}$)).

The location of the MS can be viewed from point of intersection (x, y) = ($x_{1,i}$, $y_{1,i}$) which can be acquired from the final iteration.

If values at the beginning which denotes the vector coordinates are above theposition of the mobile station, s_i = -1, decrease (as illustrated in Figure 2.11) will be applied instead of increase.

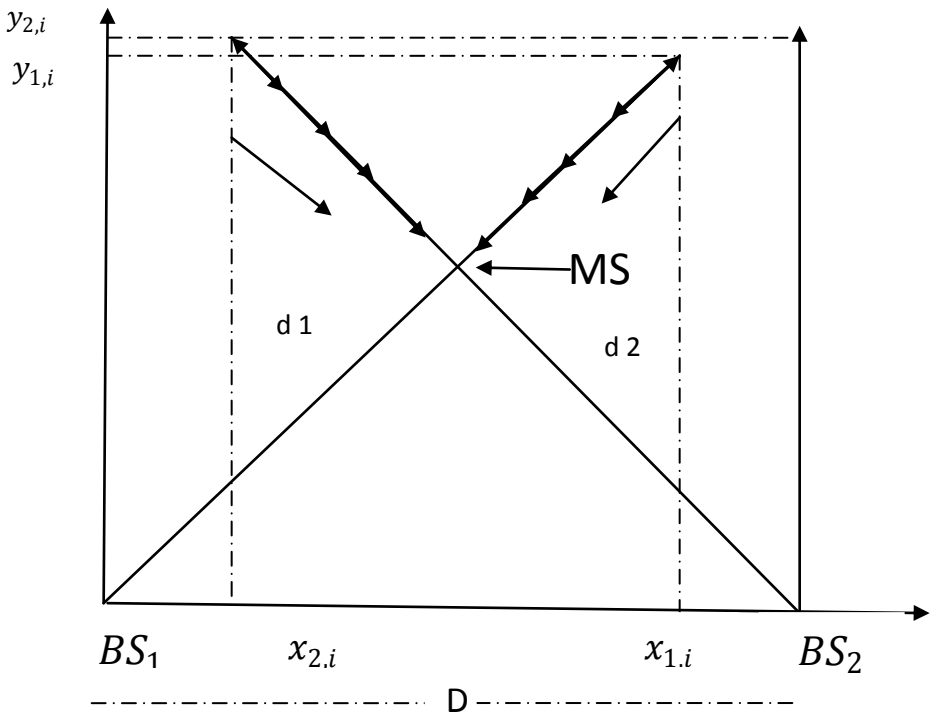

Figure 2.11: Parallel Decrementation of Vectors

Hence, if the value (π - α_2) is supposed to be the greater, then the terms increase are written :

While (| D – ($x_{1,i}$ + $x_{2,i}$) |) > ε
decrement d_2
While ($y_{2,i}$ - $y_{1,i}$) > ε)
decrement d_1
End while
End while

The location of the MS can be viewed from the point intersection (x, y) = ($x_{1,i}$, $y_{1,i}$) which can be acquired from the final iteration.

Chapter 3

PROPOSED HARDWARE-ORIENTED ALGORITHM FOR AOA

This method is based on determining the angle of the signals that reaches the antenna. In this chapter a new hardware-oriented algorithm is presented. This approach can be preceded or carried out using some simple operations such as shift, add and subtract. This will be helpful to let the algorithms being more attractive to Reduced Instruction Set Computer (RISC) processors, as they are to Field-Programmable Gate Array (FPGE) chips as well. The reasons of preferring to use RISC processors more than other types available in the market are such that; they are faster and cheaper to design, test and manufacture.

Figure 3.1 demonstrates the location of a mobile object in 3D space where α_1, α_2, α_3 and α_4 are the angles that are going to be used to determine the location of MS via AOA based algorithms.

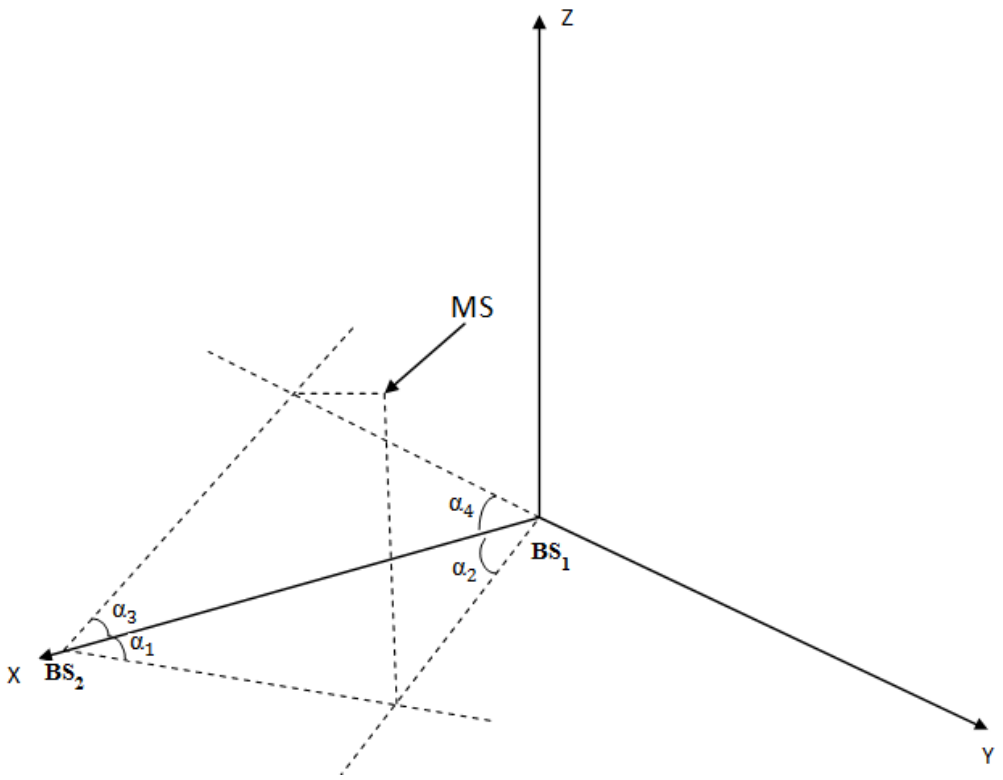

Figure 3.1: Mobile Object's Position in 3D Space

3.1 MEM-1 Algorithm

Fixed points of access to the network are usually used by most known wireless networks which are called base stations (BS). If the position of two fixed transmitters are known then the receiver can figure out its own position by determining the angle at which these transmitters are located with respect to each other. With respect to AOA, (x_p, y_p, z_p) represents the coordinate of the position that comes out from the intersection of the two spheres centered at $BS_1(0,\ 0,\ 0)$, $BS_2\,(x_1,\ 0,\ 0)$ with radius*es* d_1 and d_2 respectively. A mobile station will be located between BS_1 and BS_2 with x_p, y_p with z_p, if the vectors of these spheres intersect a mobile station. Therefore for BS_1 there is α_3 and for BS_2 there is α_4.

For finding the sin and cos of α_3. The parallel circular rotations of vectors that can be used are stated in Figure 3.2 below :

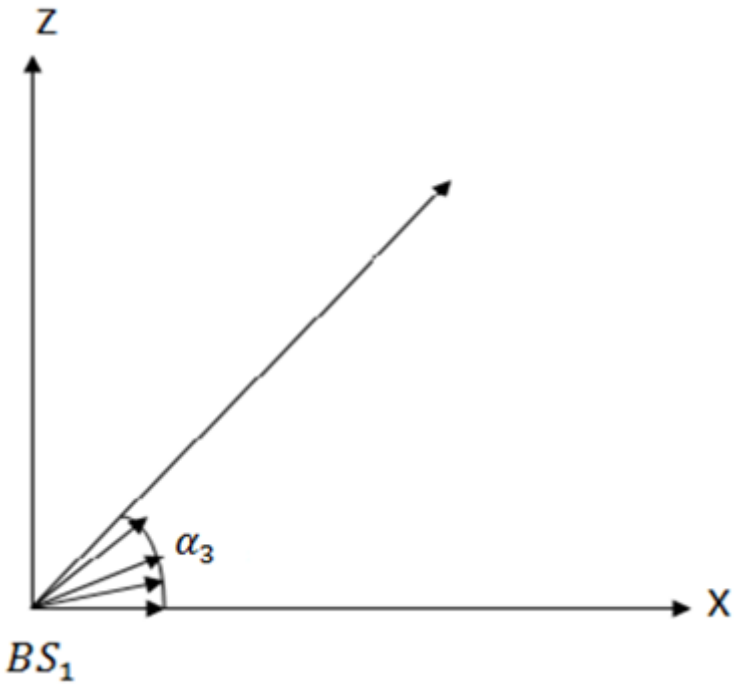

Figure 3.2: Circular Vector Rotation

The circulation rotation matrix M can be derived from equation (5) and is written as

$$\mathbf{M} = \begin{pmatrix} \cos\alpha & \pm\sin\alpha \\ \pm\sin\alpha & \cos\alpha \end{pmatrix} \tag{24}$$

The step rotation angle in radians which is α. Taking the sin function as :

$$\text{Sin } \alpha = 2^{-k} = v \tag{25}$$

Since

$$cos^2\alpha + \sin^2\alpha = 1 \tag{26}$$

Then it is possible to denote the *cos* function as:

$$\cos\alpha = \sqrt{1 - \sin^2\alpha} = \sqrt{1 - 2^{-2k}} \tag{27}$$

Then, it can be written as :

$$\sqrt{1 - 2 - 2k} \leq 1 - .2^{-2k-1} \tag{28}$$

Inequality (28) correct where is squaring for each sides gives :

$2^{-2k} \leq$ 1-2. $2^{-2k-1} + 2^{-4k-2}$

= 1- $2^{-2k} + 2^{-4k-2}$

For k more than or equal to four, one acquire $2^{-4k-2} \leq 2^{-18} < 10^{-5}$. However, for an accuracy of ε= 10^{-5}, cos σ could convergeas :

Cos σ = 1– 2^{-2k-1} = 1– $2^{-(2k+1)}$= 1- u (29)

Therefore, vector coordinates are repeatedly rotated using matrix (24). So, the rotation equations for the first vector can be written as :

$x_{1,i+1} = x_{1,i} - x_{1,i}u - z_{1,i}v$ (30)

$z_{1,i+1} = z_{1,i} - z_{1,i}u + x_{1,i}v$ (31)

Where σ_i is the step rotation angle with initial values which is $x_{1,0}$= 1, $z_{1,0}$ = 0. So the cumulative angle is written as **:**

$\sigma_{i+1} = \sigma_i + 2^{-k}$ (32)

The stop condition which is the 1st rotation will be :

$\Delta \sigma_1 = \sigma_1 - \sigma_{i+1} \leq \varepsilon$ (33)

And the last iterated $x_{1,i+1}$ = cos (α_3), and $z_{1,i+1}$ = sin (α_3).

Since X can be calculated, it is possible to lengthen the vector on plane X-Z and when vector reaches to the actual size of X then Z will be found as shown in Figure 3.3 below :

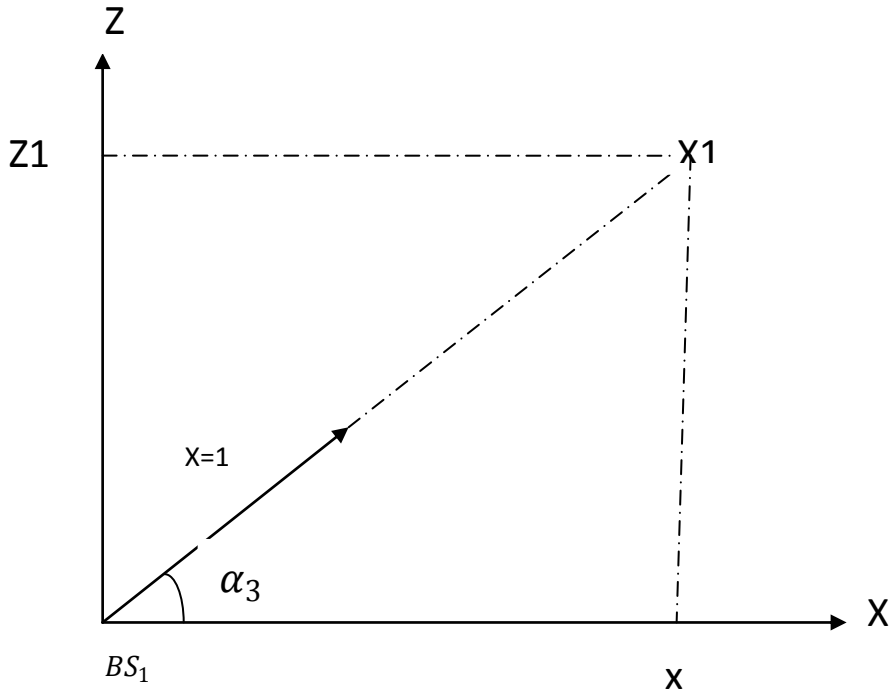

Figure 3.3: Vector Lengthening for Finding Z1

The second part of the algorithm can be implemented after finding the sin and cos of α_3. As stated in Figure 3.4, the vital idea is the incremental of length for the vector. This vector holds polar coordinates (R, α) and increases by the factor of $\pm 2^{-k}$ which can be increased in length R. Also its orthogonal coordinates are modified and written as :

$$x_{i+1} = x_i + s2^{-k} \cos \alpha \quad (34)$$

$$z_{i+1} = z_i + s\, 2^{-k} \sin \alpha \quad (35)$$

where *s* denotes the incremental factor. For each given 2^{-k}, cos (α) and sin (α) can be calculated.

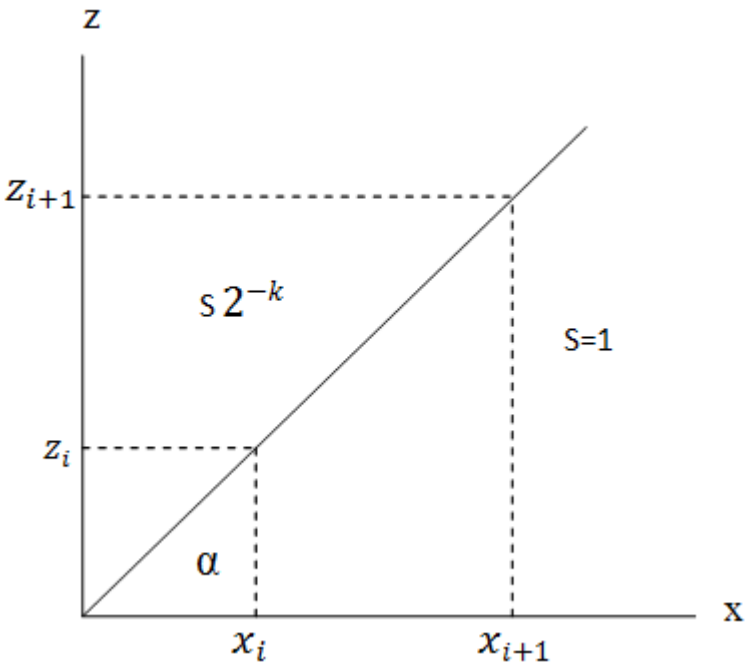

Figure 3.4: Vector Lengthening on X-Z axis

The flowchart of the proposed algorithm that can find the location of the MS for the third dimension (Z) is show in Figure 3.5.

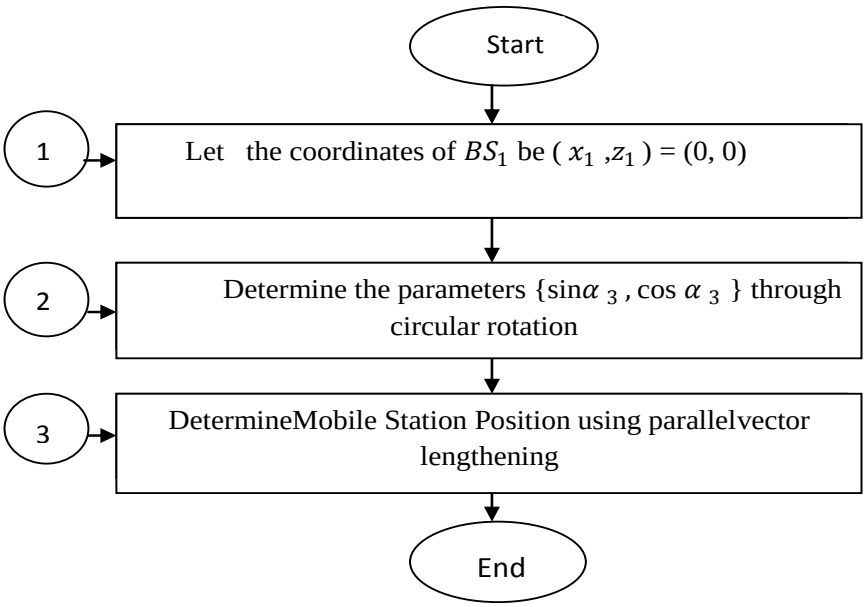

Figure 3.5: General Scheme of MEM-1 Algorithm

The location of the MS can be viewed from point of intersection (x, z) = $(x_{1,i},\ z_{1,i})$ which can be acquired from the final iteration.

3.2 MEM-2Algorithm

The proposed algorithm is based on AOA which will use the same origin of information as the conventional one. Solely subtract, add and shift operations have been used to find the mobile location without implementing any trigonometric equation. The algorithm consists of two parts, in the first part, parallel and circular rotations of the vectors will take place and the parameters cos (α_1), sin (α_1), cos (α_2), sin (α_2) for x with y and cos (α_3), sin (α_3), cos (α_4), sin (α_4) for x with z will be determined. Whereas vector lengthening is used in the second part to find the position of MS. For finding the sin and cos of α_1 and sin and cos of α_2. The parallel circular rotations of vectors can be applied as stated in Figure 3.6 below :

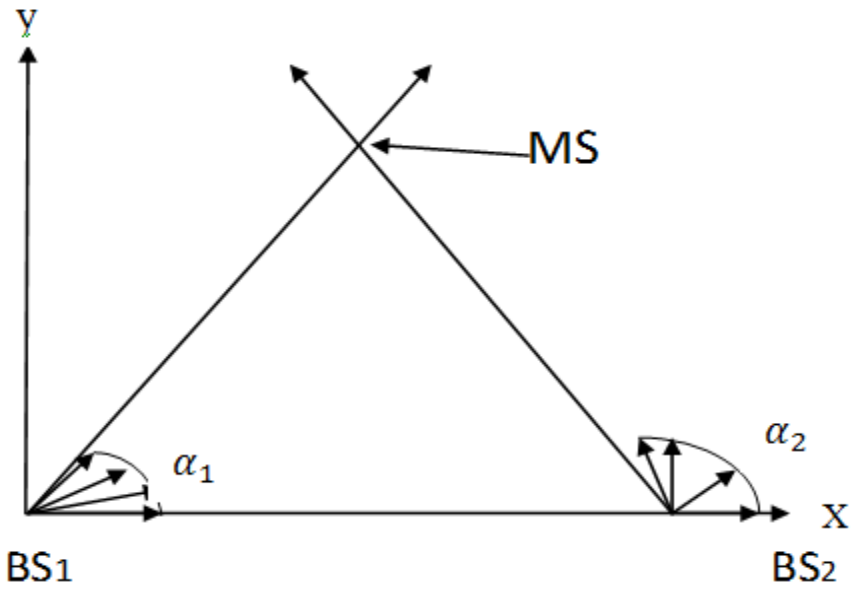

Figure 3.6: Circular Vector Rotation

The circulation rotation matrix M can be derived from equation (5) and is written as

$$M = \begin{pmatrix} \cos\alpha & \pm\sin\alpha \\ \pm\sin\alpha & \cos\alpha \end{pmatrix} \quad (36)$$

The step rotation angle in radians where sin function is taking the α as :

$$\text{Sin } \alpha = 2^{-k} = v \quad (37)$$

$$cos^2\alpha + \sin^2\alpha = 1 \quad (38)$$

Then it is possible to denote the *cos* function as:

$$\cos \alpha = \sqrt{1 - \sin^2\alpha} = \sqrt{1 - 2^{-2k}} \quad (39)$$

Then, it can be written as :

$$\sqrt{1 - 2 - 2k} \leq 1 - .2^{-2k-1} \quad (40)$$

Inequality (40) holds where squaring each sides gives :

$2^{-2k} \leq$ 1- 2. 2^{-2k-1} + 2^{-4k-2}

= 1- 2^{-2k} + 2^{-4k-2}

For k more than or equal to four, one acquire $2^{-4k-2} \leq 2^{-18} < 10^{-5}$. However, for an accuracy of $\varepsilon = 10^{-5}$, cos σ could converge as :

$$\text{Cos } \sigma = 1 - 2^{-2k-1} = 1 - 2^{-(2k+1)} = 1 - u \quad (41)$$

Therefore, vector coordinates are repeatedly rotated using matrix (36). So, the rotation equations for the first vector are written as :

$$x_{1,i+1} = x_{1,i} - x_{1,i}u - y_{1,i}v \quad (42)$$

$$y_{1,i+1} = y_{1,i} - y_{1,i}u + x_{1,i}v \quad (43)$$

Where σ_i is the step rotation angle with initial values which is $x_{1,0} = 1$, $y_{1,0} = 0$. So the cumulative angle is written as :

$$\sigma_{i+1} = \sigma_i + 2^{-k} \quad (44)$$

The stop condition which is the 1st rotation will be :

$$\Delta \sigma_1 = \sigma_1 - \sigma_{i+1} \leq \varepsilon \quad (45)$$

And the last iterated $x_{1,i+1}$ = cos (α_1), and $y_{1,i+1}$= sin (α_1).

Likely, the stop condition which is the 2nd rotation where the initial values are $x_{2,0}$ =1, $y_{2,0}$ = 0, will be as follows :

$$\Delta \alpha_2 = (\alpha_2 - \pi/2) - \alpha_{i+1} \leq \varepsilon \quad (46)$$

And the last iteration $x_{2,i+1} = sin(\alpha_2)$, $y_{2,i+1}$ = cos (α_2).

The second part of the algorithm can be implemented after finding the sin and cos of α_1 and sin and cos of α_2. The vital idea is the vector lengthening as shown in Figure 3.7. This vector holds polar coordinates (R, α) with increase factor of $\pm 2^{-k}$ which can be used to increase the length of R. Also its orthogonal coordinates are modified and will be written as :

$$x_{i+1} = x_i + s2^{-k} \cos \alpha \quad (47)$$

$$y_{i+1} = y_i + s\, 2^{-k} \sin \alpha \quad (48)$$

Where (s) denotes the incremental factor. For each given k $2^{-k}\cos(\alpha)$ and 2^{-k} sin (α) can be calculated.

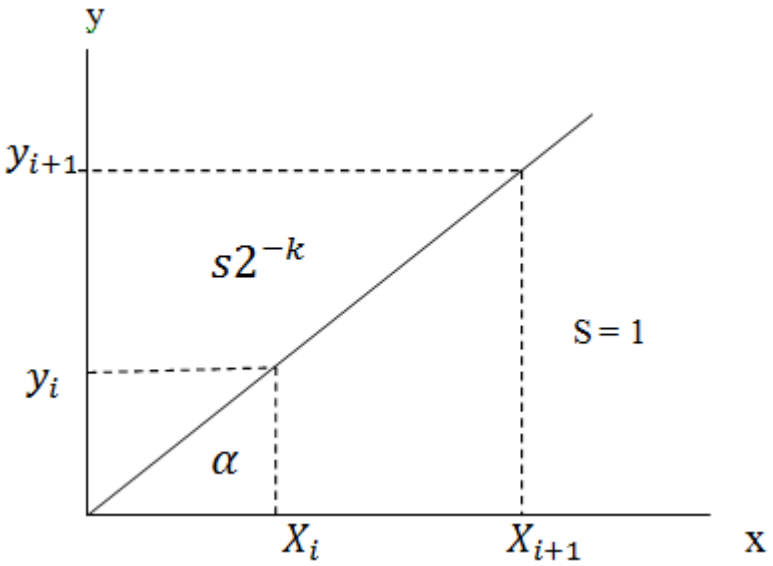

Figure 3.7: Vector Lengthening on X-Y axis

Figure 3.8, below shows the flowchart that demonstrates the procedure of finding the MS position by the proposed algorithm.

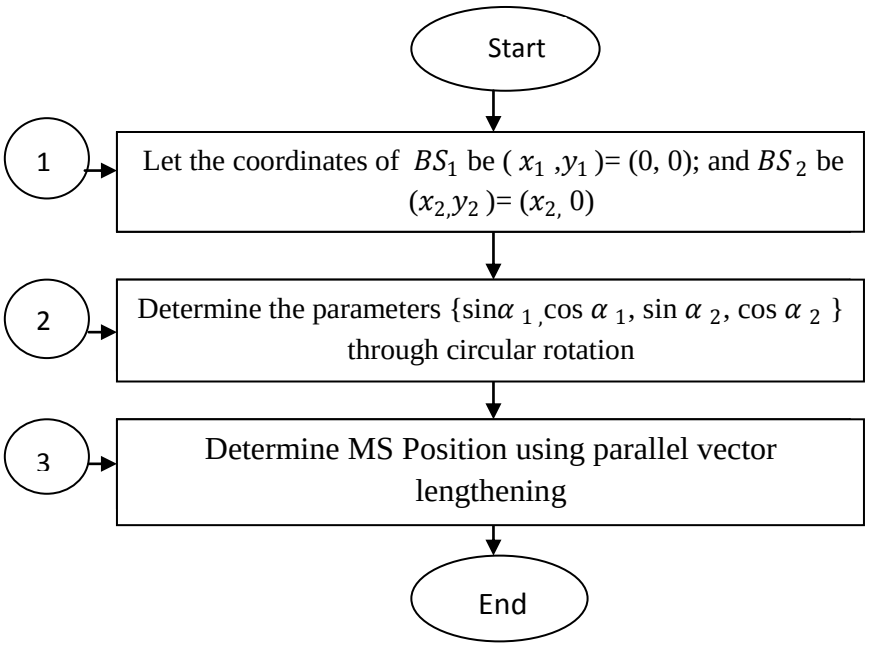

Figure 3.8: General Scheme of MEM-2 Algorithm on X-Y axis

As illustrated in Figure 3.9, the coordinates of both BS_Swhich could be set down as the point of origin to implement parallel vector incrementation. The vectors are lengthened until their heads intersects.

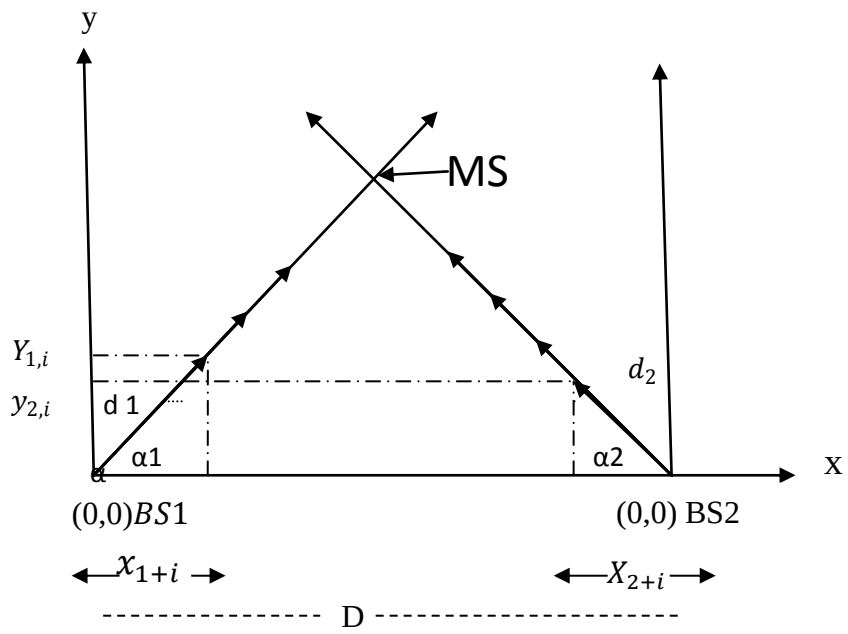

Figure 3.9: The Parallel Incrementation of Vectors

Before the beginning of operation, the angles can be compared to findthe smaller angle (i.e min $\{\pi - \alpha_2\}$), because it needs more incrementation. This is because of the equation (47) and (48), and the algorithm increasesthe vectors sizes where y coordinates of the vectors heads meets after incrementation based on $\alpha_1$and α_2. Thus, if the value (π-α_2) is proposed as the greatest, the conditions of vector lengthening will become as follows :

While (| D – ($x_{1,i}$ + $x_{2,i}$) |) > ε
Increment d_2
While ($y_{2,i}$ - $y_{1,i}$) > ε)
Increment d_1
End while
End while

The new coordinates of the vector orientation for the first vector one step away from the rotation are :

$x_{1,i+1} = x_{1,i} + s_i$. ΔD. cos α_1 (49)

$y_{1,i+1} = y_{1,i} + s_i$. ΔD. sin α_1 (50)

After one step of rotation, the coordinates of the vector head, similarly for second vector will become as follows:

$x_{2,i+1} = x_{2,i} + s_i$. ΔD. cos α_2 (51)

$y_{2,i+1} = y_{2,i} + s_i$. ΔD. sin α_2 (52)

Where ΔD is 2^{-k}, and s_i is a sign parameter (i.e. si = -1). For fast convergence, the commencing values which correspond to the parallel increments will be written as below :

$x_{1,0}$ = 0.5 cos (α_1),

$y_{1,0}$ = 0.5 sin (α_1),

$x_{2,0}$ = 0.5 cos (π - α_2),

$y_{2,0}$ = 0.5 sin (π - α_2),

Also s_i = sign (D - ($x_{1,0}$ + $x_{2,0}$)).

The location of the MS can be viewed from point of intersection(x, y) = ($x_{1,i}$, $y_{1,i}$) which is acquired from the final iteration. So, if the initial values of vector coordinates which can be lie above the coordinates of position of the mobile station. s_i = -1, and decrease (as illustrated in Figure 3.10) will be applied instead of increase.

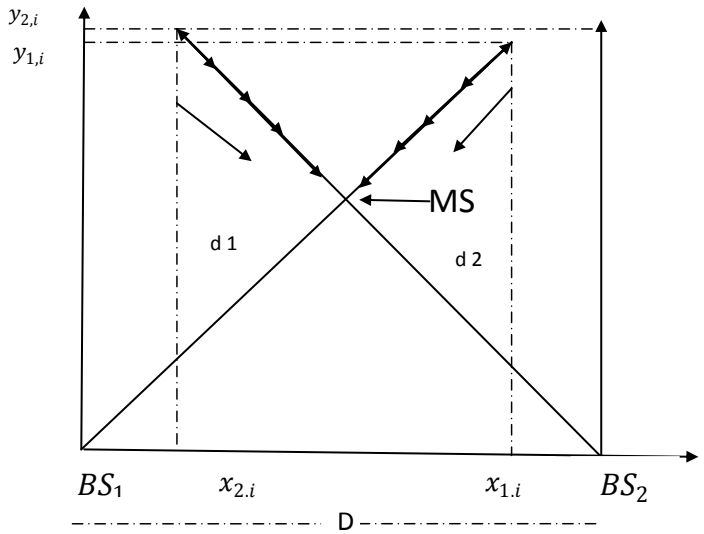

Figure 3.10: The Parallel Decrementation of Vectors

While (| D – ($x_{1,i}$ + $x_{2,i}$) |) > ε
decrement d_2
While ($y_{2,i}$ - $y_{1,i}$) > ε)
decrement d_1
End while
End while

For finding the sin and cos of α_3 and sin and cos of α_4. The parallel circular rotations of vectors can be used as :

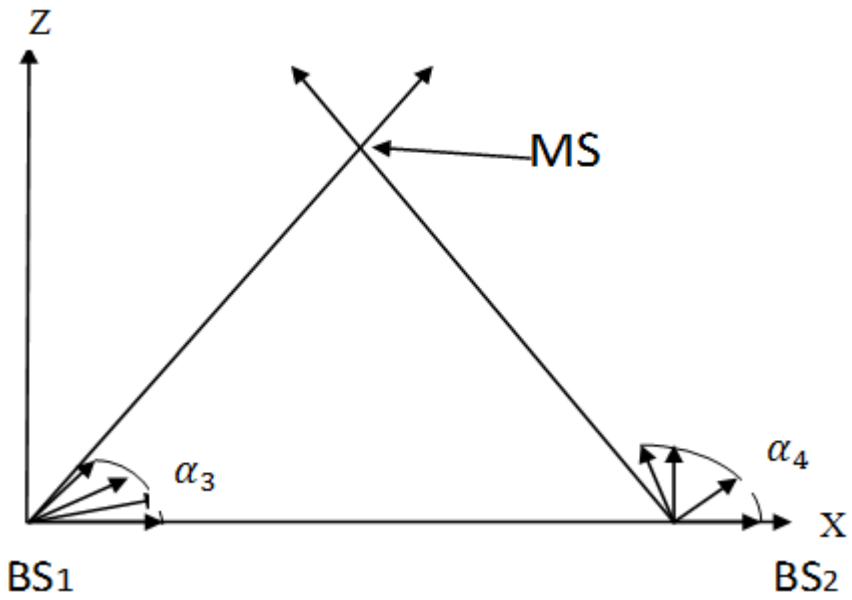

Figure 3.11: Circular Vector Rotation

The circular rotation matrix M can be derived from equation (5) and is written as

$$\mathbf{M} = \begin{pmatrix} \cos\alpha & \pm\sin\alpha \\ \pm\sin\alpha & \cos\alpha \end{pmatrix} \tag{53}$$

The step rotation angle in radians where the sin function is taking α as:

$$\text{Sin } \alpha = 2^{-k} = v \tag{54}$$

Since

$$cos^2\alpha + \sin^2\alpha = 1 \tag{55}$$

Then it is possible to denote the *cos* function as :

$$\cos\alpha = \sqrt{1 - \sin^2\alpha} = \sqrt{1 - 2^{-2k}} \tag{56}$$

where it can be written as :

$$\sqrt{1 - 2 - 2k} \leq 1 - 2^{-2k-1} \tag{57}$$

Inequality (57) holds where squaring each sides gives :

$2^{-2k} \leq 1\text{-}2.2^{-2k-1} + 2^{-4k-2}$

$= 1\text{-}\ 2^{-2k} + 2^{-4k-2}$

For *k* more than or equal to four, one acquire $2^{-4k-2} \leq 2^{-18} < 10^{-5}$. However, for an accuracy of $\varepsilon = 10^{-5}$, cosσcould converge as:

$$\text{Cos } \sigma = 1 - 2^{-2k-1} = 1 - 2^{-(2k+1)} = 1 - u \quad (58)$$

Therefore, vector coordinates are repeatedly rotated for using matrix (53). So, the rotation equations for the first vector are written as :

$$xx_{1,i+1} = xx_{1,i} - xx_{1,i}\, u - z_{1,i}\, v \quad (59)$$

$$z_{1,i+1} = z_{1,i} - z_{1,i}\, u + xx_{1,i} \quad (60)$$

Where σ_i is the step rotation angle with initial values $xx_{1,0} = 1$, $z_{1,0} = 0$. The cumulative angle is written as **:**

$$\sigma_{i+1} = \sigma_i + 2^{-k} \quad (61)$$

The stop condition which is the 1st rotation will be :

$$\Delta\, \sigma_1 = \sigma_1 - \sigma_{i+1} \leq \varepsilon \quad (62)$$

And the last iterated $xx_{1,i+1} = \cos(\alpha_3)$, and $z_{1,i+1} = \sin(\alpha_3)$.

Likely, the stop condition which is the 2nd rotation where the initial values are $xx_{2,0} = 1$, $z_{2,0} = 0$, will be as follows :

$$\Delta\, \alpha_4 = (\alpha_4 - \pi/2) - \alpha_{i+1} \leq \varepsilon \quad (63)$$

And the last iteration $xx_{2,i+1} = sin(\alpha_4)$, $z_{2,i+1} = \cos(\alpha_4)$.

The second part of the algorithm can be implemented after finding the sin and cos of α_3 and sin and cos of α_4. The vital idea is the length incrementation of the vectors as shown in Figure 3.12. This vector holds polar coordinates (R, α) and increase factor of $\pm 2^{-k}$ which can be used to increase the length of R. And also its orthogonal coordinates are modified and can be written as :

$$xx_{i+1} = xx_i + s\, 2^{-k} \cos \alpha \quad (64)$$

$$z_{i+1} = z_i + s\, 2^{-k} \sin \alpha \quad (65)$$

Where *s* denotes for the increment factor. Foreachgiven k, 2^{-k} cos (α) and 2^{-k} sin(α) can be calculated.

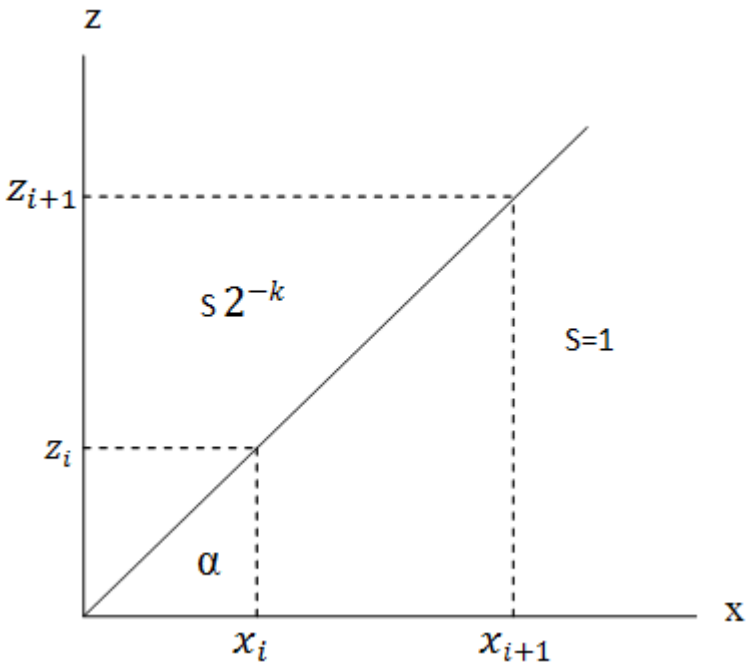

Figure 3.12: Vector Lengthening on X-Z axis

The flowchart of the proposed algorithm that can find the location of MS is demonstrated in Figure 3.13.

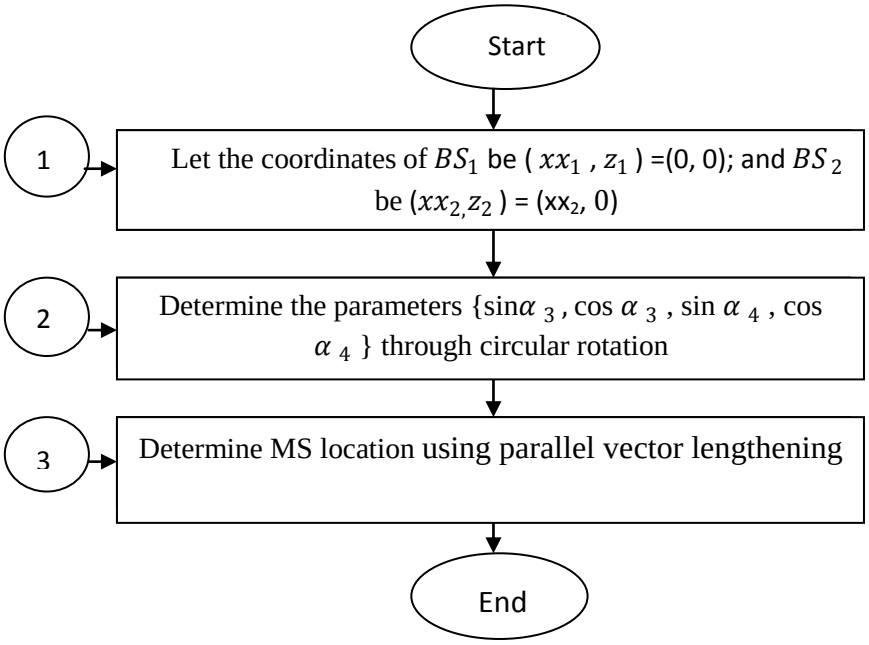

Figure 3.13: General Scheme of MEM-2 Algorithm on X-Z axis

In Figure 3.14, the coordinates of BSs which could be considered as the origin to implement parallel vector incremenation demonstrated. Both vectors are increased until their heads intersect each other.

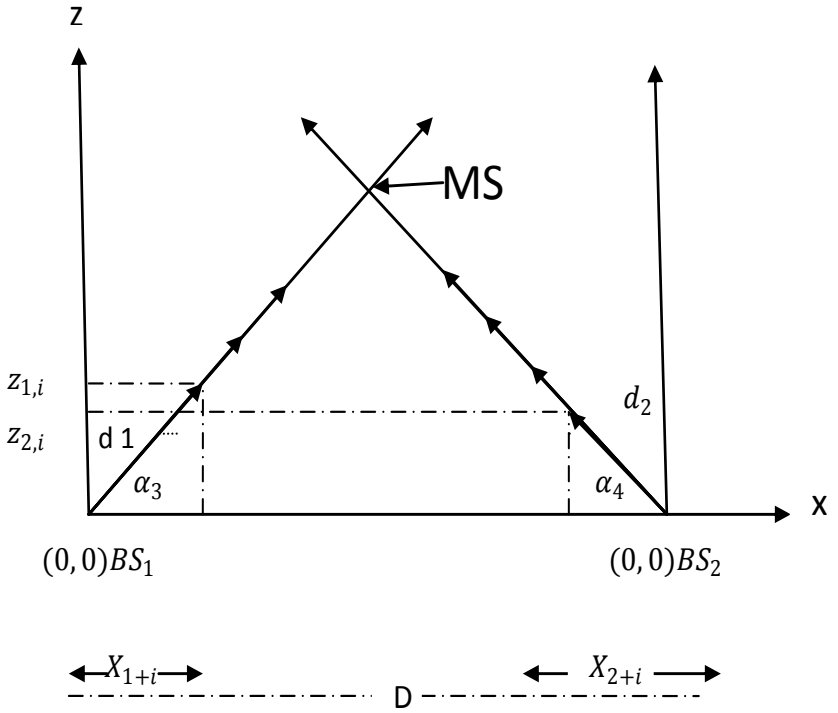

Figure 3.14: Parallel Incrementation of Vectors on X-Z axis

Before the beginning of operation, the angles can be compared to find the smaller angle (i.e min $\{\pi-\alpha_2\}$), because it needs more incrementation. This is because of the equation (64) and (65), and the algorithm increasesthe vectors sizes where y coordinates of the vectors heads meets after incrementation based on α_1 and α_2. Thus, if the value $(\pi-\alpha_2)$ is proposed to be the greatest, the conditions of vector lengthening will be as follows:

While (| D – ($xx_{1,i}$ + $xx_{2,i}$) |) > ε
Increment d_2
While ($z_{2,i}$ - $z_{1,i}$) > ε)
Increment d_1
End while
End while

The new coordinates of the vector orientation for the first vector one step away from the rotation are :

$xx_{1,i+1} = xx_{1,i} + s_i . \Delta D. \cos \alpha_3$ (66)

$z_{1,i+1} = z_{1,i} + s_i . \Delta D. \sin \alpha_3$ (67)

After one step of rotation, the coordinates of the vectors' head, similarly for the second vector will become as follows :

$xx_{2,i+1} = xx_{2,i} + s_i . \Delta D. \cos \alpha_4$ (68)

$z_{2,i+1} = z_{2,i} + s_i . \Delta D. \sin \alpha_4$ (69)

Where ΔD is 2^{-k}, and s_i is a sign parameter (i.e. si = -1). In order to speed-up the process, the values before starting the process can be written as follow as :

$xx_{1,0} = 0.5 \cos(\alpha_3)$,

$z_{1,0} = 0.5 \sin(\alpha_3)$,

$xx_{2,0} = 0.5 \cos(\pi - \alpha_4)$,

$z_{2,0} = 0.5 \sin(\pi - \alpha_4)$,

Also s_i = sign (D - ($xx_{1,0}$ + $xx_{2,0}$)).

The location of the MS can be found from the intersection point (x, z) = ($xx_{1,i}$, $z_{1,i}$) which is acquired from the last iteration.

If the values of vectors coordinates before starting lies above the MSs' coordinates, then s_i= -1, and decrease (as illustrated in Figure 3.15) will be applied instead of increase.

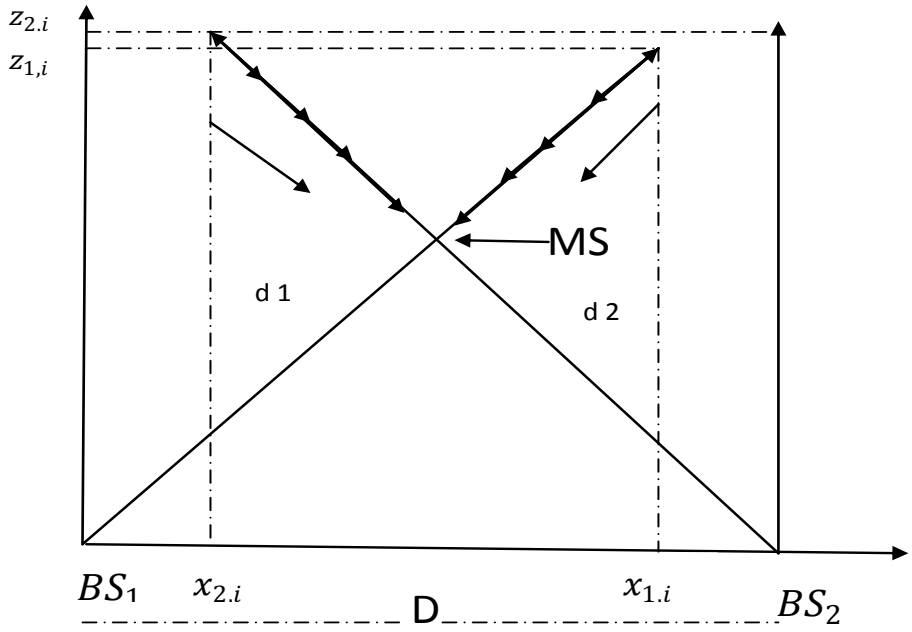

Figure 3.15: Parallel Decrementation of Vectors on X-Z axis

While (| D – ($xx_{1,i}$ + $xx_{2,i}$) |) > ε
decrement d_2
While ($z_{2,i}$ - $z_{1,i}$) > ε)
decrement d_1
End while
End while

The location of the MS is the intersection point that will be in(x, z) = ($xx_{1,i}$, $z_{1,i}$) which is obtained from the final iteration.

3.3 DMEM-1 Algorithm

Aiming to decrease the number of iterations for vector rotations and vector lengthening, MEM-1 algorithm has been modified. DMEM-1 algorithm uses varying step sizes (2^{-k}) where in MEM-1 the step size is constant. For the 1st step of the algorithm where the equation is iterated repeatedly, the value of k is changing dynamically according to the accumulated angle that can be used to define the angle accumulated in equation (15). Based on the distance between their current heads positions, if that distance is large rotation should be done via using larger step size (i.e.k = 5). When the distance becomes smaller, vector rotation lengthening is considered with smaller step sizes (i.e. k =11). In order to find the step size, the formulas below can be used :

$\Delta \alpha_1 = \alpha_1 - \sigma_{i+1}$

While $\sigma_1 \rangle \sigma_{i+1}$
While $\Delta\sigma_1 \rangle 2^{-k}$
Rotate
End
Increment K by 1
End

The stop condition for the first rotation will be :

$\Delta \alpha_1 = \alpha_1 - \sigma_{i+1} \leq \varepsilon$ (70)

And the last iteration $x_{1,i+1} = cos(\alpha_1)$, $y_{1,i+1} = \sin(\alpha_1)$.

Likely, the stop condition for the 2nd rotation and the starting values are :

$x_{2,0} = 1, y_{2,0} = 0$

$\Delta \alpha_2 = (\alpha_2 - \pi/2) - \sigma_{i+1} \leq \varepsilon$ (71)

And the last iteration $x_{2,i+1} = sin(\alpha_2)$, $y_{2,i+1} = \cos(\alpha_2)$.

In the second step, which is the vector lengthening k was valued constant (MEM-1 algorithm) but for DMEM-1 k can be change dynamically. For DMEM-1 the step size has been based on the displacement of X coordinates between the vectors' head and the position of MS's x coordinate. Convergence can be assured, via defining the k with :

$$\Delta D = \begin{cases} 2^{-k} \; if \; \Delta x_i \geq 2^{-k} \; and \cos(\alpha_{min}) \leq 0.5 \\ 2^{-m-t} \; if \; 2^{-m} \leq \Delta x_i \leq 2^{-m+1} \; and \sin(\alpha_{min}) \geq 2^{-t}, K < M \leq n, 1 < t \leq n-m, \end{cases} \quad (72)$$

Where Δx_i = | D – ($x_{1,i}$ + $x_{2,i}$) | .Practically k is 4 and n equals to 11.

In order to converge faster, the values at the beginning for parallel vector lengthening can be written as :

$x_{1,0}$ = 0.5 cos (α_1),

$y_{1,0}$ = 0.5 sin (α_1),

$x_{2,0}$ = 0.5 cos (π - α_2),

$y_{2,0}$ = 0.5 sin (π - α_2),

Also s_i = sign (D - ($x_{1,0}$ + $x_{2,0}$)).

The location of the MS can be viewed from point of intersection (x, y) = ($x_{1,i}$, $y_{1,i}$) which is acquired from the final iteration.

σ_i is the step size with initial values which is $x_{1,0}$= 1, $z_{1,0}$ = 0. Then the cumulative angle can be written as:

$$\sigma_{i+1} = \sigma_i + 2^{-k} \quad (73)$$

The stop condition which is the 1st rotation will be :

$$\Delta \alpha_1 = \alpha_1 - \sigma_{i+1} \leq \varepsilon \quad (74)$$

And from the last iteration $x_{1,i+1}$= cos (α_3), and $z_{1,i+1}$ = sin (α_3).

$$\Delta D = \begin{cases} 2^{-k} \ if \ \Delta x_i \geq 2^{-k} \ and \cos(\alpha_{min}) \leq 0.5 \\ 2^{-m-t} \ if \ 2^{-m} \leq \Delta x_i \leq 2^{-m+1} \ and \sin(\alpha_{min}) \geq 2^{-t}, K < M \leq n, 1 < t \leq n-m, \end{cases} \quad (75)$$

Where $\Delta x_i = (x_1 - xx_{1,i})$. Practically k is equal to 4 and n equal to 11.

When the vectors coordinates are above the MS coordinates, then s_i = -1, as a result a decrease will be applied instead of an increase.

3.4 DMEM-2 Algorithm

In DMEM-2 algorithm the step size for vector lengthening and circular rotations may vary from one iteration to another. Where σ_i is the step rotation angle with initial values xx_1 =1, $z_{1,0}$= 0. The cumulative angle can be written as :

$\Delta \alpha_1 = \alpha_1 - \sigma_{i+1}$

While $\sigma_1 > \sigma_{i+1}$
While $\Delta\sigma_1 > 2^{-k}$
Rotate
End
Increment K by 1
End

The stop condition for the first rotation will be :

$$\Delta \alpha_1 = \alpha_1 - \sigma_{i+1} \leq \varepsilon \quad (76)$$

And from the last iteration $xx_{1,i+1}$ = cos (α_3), and $z_{1,i+1}$ = sin (α_3).

Likely, the stop condition for the second rotation where the initial values are

$xx_{2,0}$ =1, $z_{2,0}$ = 0, will be as follows :

$$\Delta \alpha_4 = (\alpha_4 - \pi/2) - \sigma_{i+1} \leq \varepsilon \quad (77)$$

And from the last iteration $xx_{2,i+1} = sin(\alpha_4)$, $z_{2,i+1}$ = cos (α_4).

In the third step which is the vector lengthening k was valued constant (MEM-2 algorithm) but for DMEM-2 k can be change dynamically. For DMEM-2 the step

size has been based on the displacement of X coordinates between the vectors' head and the position of MS's X coordinate. Convergence can be assured, via defining the k with :

$$\Delta D = \begin{cases} 2^{-k} \ if \ \Delta x_i \geq 2^{-k} \ and \cos(\alpha_{min}) \leq 0.5 \\ 2^{-m-t} \ if \ 2^{-m} \leq \Delta x_i \leq 2^{-m+1} \ and \sin(\alpha_{min}) \geq 2^{-t}, K < M \leq n, 1 < t \leq n-m, \end{cases} \quad (78)$$

Where Δx_i = D – (($xx_{1,0}$+ $xx_{1,0}$)). Practically k is equal 4 and n equal to 1.

In order to converge faster, the values at the beginning for parallel vector lengthening can be written as :

$xx_{1,0}$ = 0.5 cos (α_3),

$z_{1,0}$ = 0.5 sin (α_3),

$xx_{2,0}$ = 0.5 cos (π - α_4),

$z_{2,0}$ = 0.5 sin (π - α_4),

Also s_i = sign (D - ($xx_{1,0}$ + $xx_{2,0}$)).

The location of the MS can be achieved from point of intersection (x, z) =($xx_{1,i}$, $z_{1,i}$) which is acquired from the final iteration.

$$\Delta A = \begin{cases} 2^{-k} \ if \ \Delta x_i \geq 2^{-k} \ and \cos(\alpha_{min}) \leq 0.5 \\ 2^{-m-t} \ if \ 2^{-m} \leq \Delta x_i \leq 2^{-m+1} \ and \sin(\alpha_{min}) \geq 2^{-t}, K < M \leq n, 1 < t \leq n-m, \end{cases} \quad (79)$$

where ΔA_i =(α_3 - ro_i). Practically k is equal4 and n equal to 11. When α_i= 1, 2, 3, 4.

When the vectors coordinates are above the MS's coordinates, then s_i = -1, as a result a decrease would utilized instead of an increase. In order to determine the position of MS in 3D AOA four algorithms presented. The Matlab simulation results for the algorithms can be found in the next chapter. The level of accuracy, exact location of MS and the relation between these two will also be discussed**.**

Chapter 4

SIMULATION RESULTS

To carry out the analysis, to locate the MS in areal coverage of the two base stations (BS) and to code the programs for traditional, MEM-1, MEM-2, DMEM-1 and DMEM-2 algorithms, Matlab 8.2 package has been used. In order to achieve the 95% of the degree of confidence, the experiments were repeated for the mobile station positions that have arbitrarily been chosen.

To compare the algorithm initiated and developed in this study with the traditional algorithm, the calculation of the average computational cost is needed to determine the position of MS in each case. This is done by multiplying the number of operations by their corresponding weights. 20 bits accuracy has been considered for weights of the operations, see Table 4.1[13].

Table 4.1: Weight of Operations[13]

Operation	Addition	Subtraction	Shift	Multiplication	Division	Sin	Cos	Tan
Weight	1	1	1	40	40	404	404	1448

Figure 4.1 illustrates the average error in determining sin (α) and cos (α) versus the step rotation angle $\sigma = 2^{-k}$. This average is obtained by calculating the sin (α) and cos (α) for $0 \leq \alpha \leq \pi/2$.

As it is seen from the Figure 4.1, the breaking point for accurate solution of sin and cos function is 2^{-8}**.** Therefore σ = 2^{-8} used for our MEM-1 and MEM-2 algorithms.

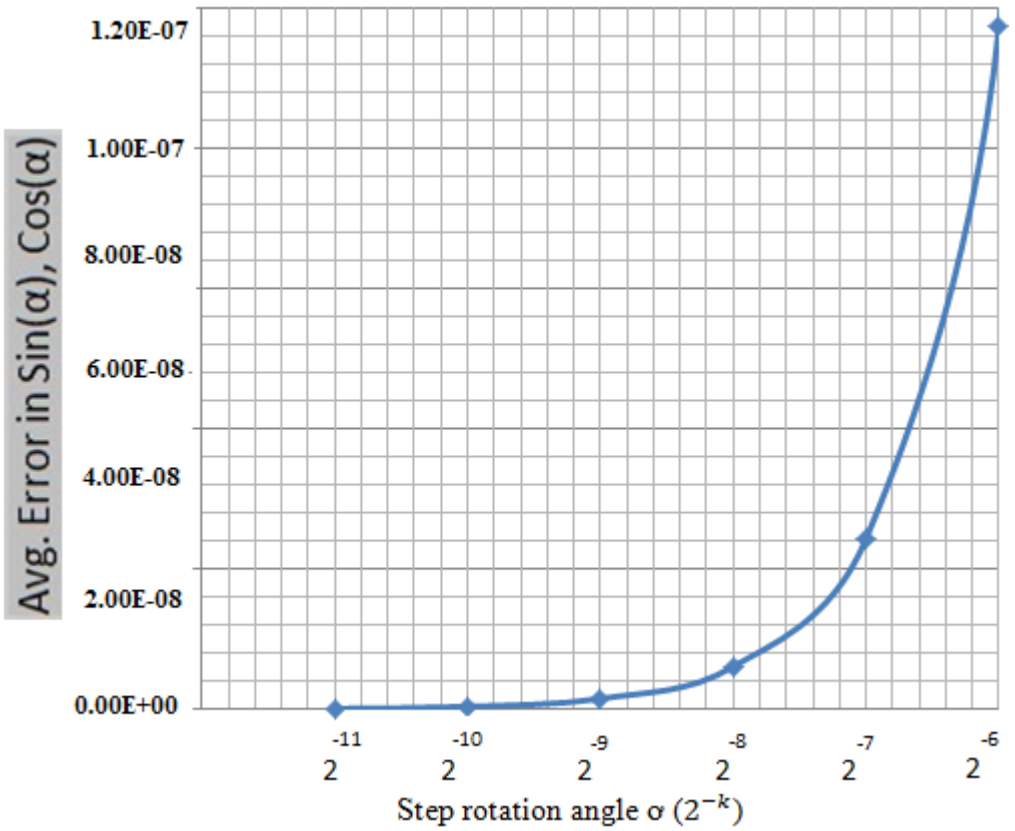

Figure 4.1: Average Error in sin (α), cos (α) Versus the Rotation Angle

Figure 4.2 illustrates the maximum error in meters while trying to determine the position of MS against step size ΔD = 2^{-k}. All proposed algorithms will satisfy the E-911 standards at a certain step size.

Accuracy of MEM-2 is better when compared to MEM-1 because in algorithm MEM-1 to find the Z coordinate we are using the X coordinate that is found in the first step of algorithm which is an approximation. In MEM-2 to find the Z coordinate the first step of algorithm will be repeated for X-Z axis and as a result the accuracy will be preserved for the Z coordinate.

Due to the dynamic change in step size DMEM-1 and DMEM-2 has a fixed accuracy. DMEM-2's accuracy is better than the DMEM-1's one, because DMEM-2 calculates the location on Z-axis without using the values that are calculated to find X and Y coordinates' values. DMEM-1 in order to find the location on Z-axis uses the value that is found for X-axis which is an estimation. This is the source of having worse accuracy then DMEM-2.

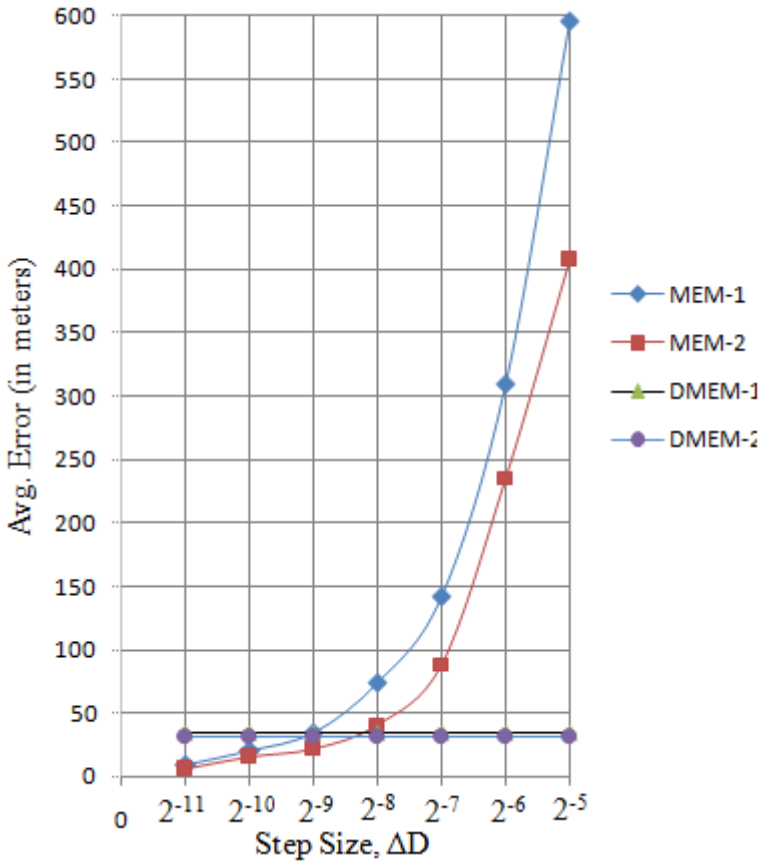

Figure 4.2: Error Estimation of the Mobile Location Versus ΔD

For the simulation results, generally the confidence interval is a measure for the accuracy of the results when the precision is satisfied, so it will be invisible in the manuscript. This can be determined from the equation shown hereinafter. If $\bar{z}$ *is* the model average and $\bar{z} = \frac{1}{n}\sum_{i=1}^{n} zi$ demonstrates sample set of size n (i.e number of runs),

S is standard deviation and S = $\sqrt{s^2}$ where $S^2 = \sum_{i=1}^{n} \frac{(z_i - \bar{z})^2}{n-1}$, $\bar{z}$ is the average of population with n-1 as the degree of freedom. For the results of simulation model, n = 10, α/2 = (1- 0.95) /2 (the aim is to calculate for 95% confidence interval).

Table 4.2: Confidence interval calculation for MEM-2 algorithm

Number of Runs	Error in Location Estimation
1	40.36619
2	46.52688
3	49.02575
4	49.05569
5	49.148
6	50.0805
7	54.69675
8	55.99025
9	60.44481
10	61.56756
mean	63.53338

The upper and lower bound limits could be calculated as :

$\bar{z} - t_{\alpha/2,n-1} \frac{S}{\sqrt{n}}$ and $\bar{z} + t_{\alpha/2,n-1} \frac{S}{\sqrt{n}}$

Ƒ = n-1= 9

95% =100 (1- α) %, then α could be calculated as α = 0.05

When $t_{\alpha/2,f} = t_{0.025,9}$ (using the table named as percentage point of the t distribution in [14]). For 95% confidence, equation (1) should be satisfied, $t_{\frac{\alpha}{2},n-1}$ = 2.262 and

achieved. As it can be seen in Table 4.2, the obtained results for error through the simulation are between lower 50.22082and upper53.15958 bounds of the interval with 95% confidence.

Table 4.3: Average Computational Costs versus Step Size

Step size	Computational Cost of traditional Algorithm	Average computational of cost MEM-1	Average computational cost for MEM-2	Average computational cost for DMEM-1	Average computational cost for DMEM-2
2^{-5}	7128	4831.73	7550.43	2805.07	4405.44
2^{-6}		9489.55	14452.84		
2^{-7}		18630.64	28648.12		
2^{-8}		35503.47	58998.35		
2^{-9}		75758.01	114122.7		
2^{-10}		150122.5	235386.2		
2^{-11}		295695.6	470531.9		

As it can be seen from table 4.3 above, MEM-1 only for step size 2^{-5} outperforms the traditional one which is not sufficient for E-911 standards. For the location estimation, DMEM-1 and DMEM-2 outperform the traditional algorithm and also satisfies the E-911 standards for the location estimation of a MS.

Chapter 5

CONCLUSION

The advantages of low-cost and broader applicability of AOA based wireless location systems have been highlighted in this work. It seems that mobile-based position determination will become more and more attractive than those which depend on network.

Since the energy source of a Mobile Station is a battery pack, energy consumption should be reduced to a minimum. To attain this, minimizing and simplifying the instructions carried out in MS represents an essential factor in the determination of location.

The main idea of this work was to propose a new hardware oriented algorithm that extends the angle of arrival 2D positioning into 3D space. To achieve this goal four new algorithms proposed. These are MEM-1, MEM-2, DMEM-1 and DMEM-2.

Every process in these algorithms use simple shift and add operations as a result RISC processors could be used to implement the proposed algorithms as hardware.

The results demonstrate that an accuracy of a good level is obtainable which satisfy the E911 standards for all four proposed algorithms. The traditional algorithm out performed by the proposed algorithms DMEM-1 and DMEM-2 and demands less time in terms of computational cost.

The distinct advantages of proposed algorithms over the traditional one are : low computational overhead and implementation simplicity. The algorithms named MEM-1 and MEM-2 show worse results (in terms of computational cost) when compared to the tradition one. The proposed algorithms DMEM-1 and DMEM-2 have 61% and 40% reduction for the computational cost respectively. DMEM-2 algorithm has a better accuracy when compared to DMEM-1 algorithm.

As a work for future, a new algorithm for 3D TDOA positioning could be proposed and a survey for 3D TOA, AOA and TDOA could be presented.

REFERENCES

[1] H.Laitinen et al : cellularlocationTechnology, CELLO project Technical Report, CELLO-WP2-VTT-DO3-007-Int, November , 2001.

[2] Y.Zhao, ‘’Standardization of mobile phonepositioningfor 3G systems,”IEEE communicationsMaganzine, No.4, vol. 40,(July 2002) 108-116.

[3] M. Salamah, E. Doukhnitch, and D. Devrim, ‘‘ A Fast Hardware-Oriented Algorithm for Cellular Mobiles Positioning, ”Lecture Notes on Computer Science LNCS 3280, PP. 267-277, October, 2004.

[4] Jean-Michel Muller :Elementary Function Algorithms and Implementation, Birkhauser, 1997.

[5] Yuan Zhang, ShutangLiu, ZhongtianJia, Localization Using Joint Distance and Angle Information for 3D Wireless Sensor Networks, IEEE Communication Letters, 2012.

[6] E. Doukhnitch, M. Salamah, E. Ozen, An efficientapproachfortrilateration in 3D positioning,Computer Communications, 2008.

[7] M. Salamah, E. Doukhnitch, General Approachesto Simple Algorithmsfor 2D positioning techniques in Cellular Networks, Computer Communications, 2008.

[8] M. Salamah, E. Douknitch, An efficientalgorithmfor mobile objectlocalization, Journal of CommunicationSystems, 2008.

[9] I. Jami, M. Ali, and R. F. Ormondroyd : Comparison of Methods of Locating and Tracking Cellular Mobiles, Novel Methods of Location and Tracking of Cellular Mobiles and Their System, Ref. No. 1999/046, IEE Colloquium, London UK, 1/1-1/6.

[10] E. Doukhnitch, M. Salamah, and A. Sandouka “A Novel Hardware-Oriented Algorithms for TDOA Positioning Technique in Cellular Networks, “ Accepted for publication in the International Symposium on Mathematical Methods in Engineering MME-06, Ankara, Turkey.

[11] E. Ozen, "Design of a Low Computational Cost Algorithm for 3-D TOA Mobile Positioning”, Ph.D. Thesis, 2009.

[12] C. Bayramer, " A hardware-oriented Algorithm for Angle of Arrival Positioning Technique n Cellular Network", M.Sc. Thesis, 2006.

[13] Jean- Michael Muller : Elementary Function Algorithms and Implementation. Birkhauser (1997).

[14] Montgomery D., Design and Analysis of Experiments, John wile & Sons, Inc., 1993.

APPENDICES

Appendix A : Source code of the MEM-1 Algorithm

```
clc;
clear;
format long;
fr=fopen('rnds.txt','r');
fw=fopen('results.txt','w');
cost=0;
costm=0;
while ~feof(fr)
    alphainp1=fscanf(fr,'%d',1);
    alphainp2=fscanf(fr,'%d',1);
    w1=fscanf(fr,'%d',1); % wont be used
    alphainp3=fscanf(fr,'%d',1);
    alphainp4=fscanf(fr,'%d',1);
    w2=fscanf(fr,'%d',1); %wont be used
%initial inputs
alpha1=(alphainp1*pi)/180;
alpha2=(alphainp2*pi)/180;
alpha3=(alphainp3*pi)/180;
% pause on; %to stop
D=16; % distance between two base stations. x=8 y=6 hyp=10
%constants
k=8;
v=2^-k;
u=2^-(2*k+1);
err=2^-4;

% first side rotation for angle x1,y1
x1=1;
y1=0;
ro=0;
if (alpha1>pi/2)
    sign=-1;
    delta=pi-alpha1;
else
    sign=1;
    delta=alpha1;
end

while((delta-ro)>=err)
 oldx1=x1;
 x1=x1-x1*u-y1*v;
 cost=cost+4;
 y1=y1-y1*u+oldx1*v;
 cost=cost+4;
ro=ro+2^-k;
 cost=cost+2;
end%while
cosalpha1=sign*x1;
sinalpha1=y1;
% end of first side rotation fo angle

% 3d z-side rotation for angle x1,y1
xx1=1;
z1=0;
ro=0;
```

```
if (alpha3>pi/2)
    sign=-1;
    delta=pi-alpha3;
else
    sign=1;
    delta=alpha3;
end

while((delta-ro)>=err)
 oldxx1=xx1;
 xx1=xx1-xx1*u-z1*v;
 cost=cost+4;
 z1=z1-z1*u+oldxx1*v;
 cost=cost+4;
ro=ro+2^-k;
 cost=cost+2;
end%while
cosalpha3=sign*xx1;
sinalpha3=z1;
% end of 3d-z side rotation fo angle

% second side rotation for angle x2,y2
x2=1;
y2=0;
ro=0;
if (alpha2>pi/2)
    sign=-1;
    delta=pi-alpha2;
else
    sign=1;
    delta=alpha2;
end
while((delta-ro)>=err)
 oldx2=x2;
 x2=x2-x2*u-y2*v;
 cost=cost+4;
 y2=y2-y2*u+oldx2*v;
 cost=cost+4;
ro=ro+2^-k;
 cost=cost+2;
end%while
cosalpha2=sign*x2; % in documentation sin
sinalpha2=y2; %documentation cos
% end of second side rotation for angle

% to speed up convergence
%x1=0.5*cosalpha1;
%y1=0.5*sinalpha1;
%x2=0.5*cosalpha2;
%y2=0.5*sinalpha2;
%sign=(D-(abs(x1)+abs(x2)))/abs(D-(abs(x1)+abs(x2)));
% end of speed up

increment=2^-k;
if ((pi-alpha2)>=alpha1)
while (abs(D-(abs(x1)+abs(x2)))>err)
      x2=x2+sign*increment*cosalpha2;
      cost=cost+3;
      y2=y2+sign*increment*sinalpha2;
      cost=cost+3;
while (abs(y2)-abs(y1)>err)
```

```
        x1=x1+sign*increment*cosalpha1;
        cost=cost+3;
        y1=y1+sign*increment*sinalpha1;
        cost=cost+3;
end% inner while
end% while
else%if
while (abs(D-(abs(x1)+abs(x2)))>err)
      x1=x1+sign*increment*cosalpha1;
      cost=cost+3;
      y1=y1+sign*increment*sinalpha1;
      cost=cost+3;
while (abs(y2-y1)>err)
        x2=x2+sign*increment*cosalpha2;
        cost=cost+3;
        y2=y2+sign*increment*sinalpha2;
        cost=cost+3;
end% inner while
end% while
end%if

%finding rotation for 3d z-dimension
      xx1=cosalpha3;
      z1=sinalpha3;
      increment=2^-k;
while ((abs(x1)-abs(xx1))>err)
        xx1=xx1+sign*increment*cosalpha3;
        cost=cost+3;
        z1=z1+sign*increment*sinalpha3;
        cost=cost+3;
end% inner while
%end finding rotation for 3d z-dimension

% manual calculation
if (alphainp1==alphainp2)
 mx2=D/2; %replaced with trigonometric formula;
costm=costm+40;
else
 mx2=(D*tan(alphainp1))/(tan(alphainp1)-tan(alphainp2));
costm=costm+4425;
end
mx1=D-mx2;
costm=costm+1;
my1=mx1*tan(alphainp1);
costm=costm+1488;
my2=mx2*tan(alphainp2);
costm=costm+1488;
cost=cost+4;
err1=mx1-x1;
err2=my1-y1;
err3=mx2-x2;
err4=my2-y2;
err5=mx1-xx1; % from x-z axis
err6=x1-xx1; % from x-z axis
costm=costm+6;

% end of manual calculation

fprintf(fw,'%4d\t%4d\t%4d\t%4d\t%4d\t%4d\t%2.8f\t%2.8f\t%2.8f\t%2.8f
\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.
8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\n',alphainp1,alphainp2,w1,alphainp1,a
```

```
lphainp1,w2,x1,y1,x2,y2,xx1,z1,mx1,my1,mx2,my2,err1,err2,err3,err4,e
rr5,err6,cost,costm);

% %calculated first base station stuff
% hyp1=sqrt(x1^2+y1^2);
% calculatedalpha1=(asind(y1/hyp1)*pi)/180;
% disp('********From First Base Station********')
% disp(strcat('x1=',num2str(x1,8)));
% disp(strcat('y1=',num2str(y1,8)));
% disp(strcat('hyp1=',num2str(hyp1,8)));
% disp(strcat('alpha1=',num2str(alpha1,8)));
% disp(strcat('angle=',num2str(alpha1*180/pi,8)));
% disp(strcat('calculatedalpha1=',num2str(calculatedalpha1,8)));
%
disp(strcat('calculatedangle=',num2str(calculatedalpha1*180/pi,8)));
%
% %calculated second base station stuff
% hyp2=sqrt(x2^2+y2^2);
% calculatedalpha2=pi-((asind(y2/hyp2)*pi)/180);
% disp('********From Second Base Station********')
% disp(strcat('x2=',num2str(x2,8)));
% disp(strcat('y2=',num2str(y2,8)));
% disp(strcat('hyp2=',num2str(hyp2,8)));
% disp(strcat('alpha2=',num2str(alpha2,8)));
% disp(strcat('angle2=',num2str(alpha2*180/pi,8))); %alpha2=((180-
asind(0.6))*pi)/180;
% disp(strcat('calculatedalpha2=',num2str(calculatedalpha2,8)));
%
disp(strcat('calculatedangle2=',num2str(calculatedalpha2*180/pi,8)))
;

end% while ~feof(fid)
k=5:11;
i=1;
for k=5:11
v(i)=2^(-k);
u(i)=2^-(2*k+1);
errr(i)=abs(u(i)-1+sqrt(1-(v(i))^2));
fprintf('%4d\t',errr(i));
i=i+1;
end
p=plot(v,errr,'b:+');
set(p,'Color','blue','LineWidth',2);
hold on;
fclose(fr);
fclose(fw);
```

Appendix B : Source code of the MEM-2 Algorithm

```
clc;
clear;
Format long;
for k=12:12  %5:11
fr=fopen(strcat(pwd,'\','rnds.txt'),'r');
fw=fopen(strcat(pwd,'\','results2ndver',num2str(k),'.txt'),'w');
fprintf(fw,'%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t
%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n','a1','
a2','a3','a4','x1','y1','x2','y2','xz1','z1','xz2','z2','mx1','my1',
'mx2','my2','mx3','mz1','mx4','mz2','Ex1','Ex2','Ey1','Ey2','Exz1','
Exz2','Ez1','Ez2','Cost');
counter=0;
Eavgx1=0;
Eavgx2=0;
Eavgy1=0;
Eavgy2=0;
Eavgxz1 =0;
Eavgxz2 =0;
Eavgz1 =0;
Eavgz2 =0;
avgcost=0;
while ~feof(fr)
    cost=0;
    counter=counter+1;
    alphainp1=fscanf(fr,'%d',1);
    alphainp2=fscanf(fr,'%d',1);
    winp1=fscanf(fr,'%d',1); % wont be used
    alphainp3=fscanf(fr,'%d',1);
    alphainp4=fscanf(fr,'%f',1);
    winp2=fscanf(fr,'%f\n',1); %wont be used

%initial inputs
alpha1=(alphainp1*pi)/180;
alpha2=(alphainp2*pi)/180;
w1=(winp1*pi)/180;

alpha3=(alphainp3*pi)/180;
alpha4=(alphainp4*pi)/180;
w2=(winp2*pi)/180;
% pause on; %to stop
D=16; % distance between two base stations. x=8 y=6 hyp=10
%constants
%k=5;
v=2^-k;
u=2^-(2*k+1);
err=10^-5;

% first side rotation for angle x1,y1
x1=1;
y1=0;
ro=0;
if (alpha1>pi/2)
    sign=-1;
    delta=pi-alpha1;
else
```

```
    sign=1;
    delta=alpha1;
end

figure(1);
hold on;
title('1st rotation');
axis([-1 1 -1 1]);
grid on;
xlabel('X');
ylabel('Y');
while((delta-ro)>=err)
 oldx1=x1;
 x1=x1-x1*u-y1*v;
 y1=y1-y1*u+oldx1*v;
ro=ro+2^-k;
 plot(x1,y1,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
 cost=cost+9;
end%while
cosalpha1=sign*x1;
sinalpha1=y1;
 plot(cosalpha1,sinalpha1,'Color', [0.0, 0.0, 1.0], 'Marker', '+');
% end of first side rotation for angle

% second side rotation for angle x2,y2
figure(2);
hold on;
title('2nd rotation');
axis([-1 1 -1 1]);
grid on;
xlabel('X');
ylabel('Y');
x2=1;
y2=0;
ro=0;
if (alpha2>pi/2)
    sign=-1;
    delta=pi-alpha2;
else
    sign=1;
    delta=alpha2;
end
while((delta-ro)>=err)
 oldx2=x2;
 x2=x2-x2*u-y2*v;
 y2=y2-y2*u+oldx2*v;
ro=ro+2^-k;
  plot(x2,y2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
  cost=cost+9;
end%while
cosalpha2=sign*x2; % in documentation sin
sinalpha2=y2; %documentation cos
% end of second side rotation for angle
plot(cosalpha2,sinalpha2,'Color', [0.0, 0.0, 1.0], 'Marker', '+');
% to speed up convergence
%x1=0.5*cosalpha1;
%y1=0.5*sinalpha1;
%x2=0.5*cosalpha2;
%y2=0.5*sinalpha2;
%sign=(D-(abs(x1)+abs(x2)))/abs(D-(abs(x1)+abs(x2)));
% end of speed
```

```
figure(3);
hold on;
title('MS Location from Alpha1 & Alpha 2 on X Y axis');
axis([-16 16 -16 16]);
grid on;
xlabel('X');
ylabel('Y');

sign=1;
increment=2^-k;
if ((alpha2)>=alpha1)
while ((D-(x1+x2))>err)
      x2=x2+sign*increment*cosalpha2;
      y2=y2+sign*increment*sinalpha2;
      plot(x2,y2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
      cost=cost+4;
while ((y2-y1)>err)
        x1=x1+sign*increment*cosalpha1;
        y1=y1+sign*increment*sinalpha1;
        plot(x1,y1,'Color', [0.0, 1.0, 0.0], 'Marker', '+');
        cost=cost+4;
end% inner while
end% while
else%if
while ((D-(x1+x2))>err)
      x1=x1+sign*increment*cosalpha1;
      y1=y1+sign*increment*sinalpha1;
      plot(x1,y1,'Color', [0.0, 1.0, 0.0], 'Marker', '+');
      cost=cost+4;
while ((y1-y2)>err)
        x2=x2+sign*increment*cosalpha2;
        y2=y2+sign*increment*sinalpha2;
        plot(x2,y2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
        cost=cost+4;
end% inner while
end% while
end%i
% 3d z-side rotation for angle x1,y1
% first side rotation for angle xz1,z1
xz1=1;
z1=0;
ro=0;
if (alpha3>pi/2)
    sign=-1;
    delta=pi-alpha3;
else
    sign=1;
    delta=alpha3;
end

figure(4);
hold on;
title('1st rotation (X-Z)');
axis([-1 1 -1 1]);
grid on;
xlabel('X');
ylabel('Z');
while((delta-ro)>=err)
 oldxz1=xz1;
 xz1=xz1-xz1*u-z1*v;
 z1=z1-z1*u+oldxz1*v;
```

```
ro=ro+2^-k;
 plot(xz1,z1,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
 cost=cost+9;
end%while
cosalpha3=sign*xz1;
sinalpha3=z1;
 plot(cosalpha3,sinalpha3,'Color', [0.0, 0.0, 1.0], 'Marker', '+');
% end of first side rotation fo angle

% second side rotation for angle xz2,z2
figure(5);
hold on;
title('2nd rotation');
axis([-1 1 -1 1]);
grid on;
xlabel('X');
ylabel('Z');
xz2=1;
z2=0;
ro=0;
if (alpha4>pi/2)
    sign=-1;
    delta=pi-alpha4;
else
    sign=1;
    delta=alpha4;
end
while((delta-ro)>=err)
 oldxz2=xz2;
 xz2=xz2-xz2*u-z2*v;
 z2=z2-z2*u+oldxz2*v;
ro=ro+2^-k;
  plot(xz2,z2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
  cost=cost+9;
end%while
cosalpha4=sign*xz2; % in documentation sin
sinalpha4=z2; %documentation cos
% end of second side rotation for angle
plot(cosalpha4,sinalpha4,'Color', [0.0, 0.0, 1.0], 'Marker', '+');


% to speed up convergence
%xz1=0.5*cosalpha3;
%z1=0.5*sinalpha3;
%xz2=0.5*cosalpha4;
%z2=0.5*sinalpha4;
%sign=(D-(abs(xz1)+abs(xz2)))/abs(D-(abs(xz1)+abs(xz2)));
% end of speed up


figure(6);
hold on;
title('MS Location from Alpha3 & Alpha 4 on X Z axis');
axis([-16 16 -16 16]);
grid on;
xlabel('X');
ylabel('Z');

sign=1;
increment=2^-k;
```

```
if ((alpha4)>=alpha3)
while ((D-(xz1+xz2))>err)
      xz2=xz2+sign*increment*cosalpha4;
      z2=z2+sign*increment*sinalpha4;
      plot(xz2,z2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
      cost=cost+4;
while ((z2-z1)>err)
        xz1=xz1+sign*increment*cosalpha3;
        z1=z1+sign*increment*sinalpha3;
        plot(xz1,z1,'Color', [0.0, 1.0, 0.0], 'Marker', '+');
        cost=cost+4;
end% inner while
end% while
else%if
while ((D-(xz1+xz2))>err)
      xz1=xz1+sign*increment*cosalpha3;
      z1=z1+sign*increment*sinalpha3;
      plot(xz1,z1,'Color', [0.0, 1.0, 0.0], 'Marker', '+');
      cost=cost+4;
while ((z1-z2)>err)
        xz2=xz2+sign*increment*cosalpha4;
        z2=z2+sign*increment*sinalpha4;
        plot(xz2,z2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
        cost=cost+4;
end% inner while
end% while
end%if
%end finding rotation for 3d z-dimension
% manual calculation
mcost=0;
v1=D*sin(alpha2)/sin(w1);
mcost=mcost+40+404+40+404;
mx1=cos(alpha1)*v1;
mcost=mcost+404+40;
my1=mx1*tan(alpha1);
mcost=mcost+1448+40;

v2=D*sin(alpha1)/sin(w1);
mcost=mcost+40+404+40+404;
mx2=cos(alpha2)*v2;
mcost=mcost+404+40;
my2=mx2*tan(alpha2);
mcost=mcost+1448+40;
v3=D*sin(alpha4)/sin(w2);
mcost=mcost+40+404+40+404;
mx3=cos(alpha3)*v3;
mcost=mcost+404+40;
mz1=mx3*tan(alpha3);
mcost=mcost+1448+40;
v4=D*sin(alpha3)/sin(w2);
mcost=mcost+40+404+40+404;
mx4=cos(alpha4)*v4;
mcost=mcost+404+40;
mz2=mx4*tan(alpha4);
mcost=mcost+1448+40;
% end of manual calculation
%errors
Ex1=abs(mx1-x1); % error of X from BS1
Ex2=abs(mx2-x2); % error of X from BS2
Ey1=abs(my1-y1); % error of y from BS1
Ey2=abs(my2-y2); % error of Y from BS2
```

```
Exz1=abs(mx3-xz1);    % error of X from BS1
Exz2=abs(mx4-xz2);    % error of X from BS2
Ez1=abs(mz1-z1);    % error of Z from BS1
Ez2=abs(mz2-z2);    % error of Z from BS2
%end of erros
%average errors
Eavgx1=Eavgx1+Ex1;
Eavgx2=Eavgx2+Ex2;
Eavgy1=Eavgy1+Ey1;
Eavgy2=Eavgy2+Ey2;
Eavgxz1 =Eavgxz1 +Exz1;
Eavgxz2 =Eavgxz2 +Exz2;
Eavgz1 =Eavgz1 +Ez1;
Eavgz2 =Eavgz2 +Ez2;
avgcost=avgcost+cost;
%end of averages
fprintf(fw,'%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t
%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f
\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.
8f\t%2.8f\n',alpha1,alpha2,alpha3,alpha4,x1,y1,x2,y2,xz1,z1,xz2,z2,m
x1,my1,mx2,my2,mx3,mz1,mx4,mz2,Ex1,Ex2,Ey1,Ey1,Exz1,Exz2,Ez1,Ez2,cos
t);

close all;
end% while ~feof(fid)
Eavgx1=Eavgx1/counter;
Eavgx2=Eavgx2/counter;
Eavgy1=Eavgy1/counter;
Eavgy2=Eavgy2/counter;
Eavgxz1=Eavgxz1/counter;
Eavgxz2=Eavgxz2/counter;
Eavgz1=Eavgz1/counter;
Eavgz2=Eavgz2/counter;
avgcost=avgcost/counter;

fprintf(fw,'%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n','Eavgx1','Eavg
x2','Eavgy1','Eavgy2','Eavgxz1','Eavgxz2','Eavgz1','Eavgz2','AvgCost
','Mcost');
fprintf(fw,'%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t
%2.8f\t%2.8f\t%2.8f\n',Eavgx1,Eavgx2,Eavgy1,Eavgy2,Eavgxz1,Eavgxz2,E
avgz1,Eavgz2,avgcost,mcost);
fclose(fr);
fclose(fw);
end
h = msgbox('Operation Completed');
```

Appendix C : Source code of Mobile Object location Generator Algorithm

```
clc;
clear;
format long;
fid=fopen(strcat(pwd,'\','rnds.txt'),'w');
samplesize=100;
i=1;
while(i<=samplesize)
% creation of angles for x-y plate
a1=10+ceil(rand()*60);
a2=10+ceil(rand()*60);
w1=180-(a1+a2);

% arrangement of angles for x-z plate
D=16;
alpha1=(a1*pi)/180;
alpha2=(a2*pi)/180;
wr1=(w1*pi)/180;

v1=D*sin(a2)/sin(wr1);
x1=cos(alpha1)*v1;
x2=D-x1;


a3=10+ceil(rand()*60);
ta4=x2/x1*tand(a3);
a4=abs(atand(ta4));
w2=180-(a3+a4);
j=0;
while ((a4<10)||(w2<10))
 a3=10+ceil(rand()*60);
 ta4=x2/x1*tand(a3);
 a4=abs(atand(ta4));
 w2=180-(a3+a4);
 j=j+1;
if (j>1000)
break;
end
end

if (j<1000)
fprintf(fid,'%d\t%d\t%d\t%d\t%f\t%f\n',a1,a2,w1,a3,a4,w2);
i=i+1;
end
end%while
fclose(fid);
h = msgbox('Operation Completed')
```

Appendix D : Source code of the DMEM-1 Algorithm

```
clc;
clear;
format long;
fr=fopen(strcat(pwd,'\','rnds.txt'),'r');
fw=fopen(strcat(pwd,'\','results1stDynaver.txt'),'w');
fprintf(fw,'%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t
%s\t%s\t%s\t%s\t%s\t%s\t%s\n','          a1','          a2','
a3','          a4','          x1','          y1','          x2','
y2','          xx1','          z1','          mx1','          my1','
mx2','          my2','          mz','          Ex1','          Ex2','
Ey1','          Ey2','          Ez','          Cost');
counter=0;
Eavgx1=0;
Eavgx2=0;
Eavgy1=0;
Eavgy2=0;
Eavgz =0;
avgcost=0;
while ~feof(fr)
    cost=0;
    counter=counter+1;
    alphainp1=fscanf(fr,'%d',1);
    alphainp2=fscanf(fr,'%d',1);
    winp1=fscanf(fr,'%d',1); % wont be used
    alphainp3=fscanf(fr,'%d',1);
    alphainp4=fscanf(fr,'%f',1);
    winp2=fscanf(fr,'%f\n',1); %wont be used

%initial inputs
alpha1=(alphainp1*pi)/180;
alpha2=(alphainp2*pi)/180;
w1=(winp1*pi)/180;

alpha3=(alphainp3*pi)/180;
alpha4=(alphainp4*pi)/180;
w2=(winp2*pi)/180;
% pause on; %to stop
D=16; % distance between two base stations. x=8 y=6 hyp=10
%constants
%k=5;

err=10^-5;

% first side rotation for angle x1,y1
x1=1;
y1=0;
ro=0;
if (alpha1>pi/2)
    sign=-1;
    delta=pi-alpha1;
else
    sign=1;
    delta=alpha1;
end
```

```
figure(1);
hold on;
title('1st rotation');
axis([-1 1 -1 1]);
grid on;
xlabel('X');
ylabel('Y');
k=4;
while((delta-ro)>=err)
ro=ro+2^-k;
if ((abs(delta-ro)<2^-k) && (k<11))
ro=ro-2^-k;
     k=k+1;
ro=ro+2^-k;
     cost=cost+1+2+2;%1 for k +2 for ro +2 for u
end
 v=2^-k;
 u=2^-(2*k+1);
 oldx1=x1;
 x1=x1-x1*u-y1*v;
 y1=y1-y1*u+oldx1*v;
 plot(x1,y1,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
 cost=cost+9;
end%while
cosalpha1=sign*x1;
sinalpha1=y1;
 plot(cosalpha1,sinalpha1,'Color', [0.0, 0.0, 1.0], 'Marker', '+');

% end of first side rotation fo angle

% second side rotation for angle x2,y2
figure(2);
hold on;
title('2nd rotation');
axis([-1 1 -1 1]);
grid on;
xlabel('X');
ylabel('Y');
x2=1;
y2=0;
ro=0;

if (alpha2>pi/2)
    sign=-1;
    delta=pi-alpha2;
else
    sign=1;
    delta=alpha2;
end

k=4;
while((delta-ro)>=err)
ro=ro+2^-k;
if ((abs(delta-ro)<2^-k) && (k<11))
ro=ro-2^-k;
     k=k+1;
ro=ro+2^-k;
     cost=cost+1+2+2;%1 for k +2 for ro +2 for u
end
 v=2^-k;
 u=2^-(2*k+1);
```

```
 oldx2=x2;
 x2=x2-x2*u-y2*v;
 y2=y2-y2*u+oldx2*v;

  plot(x2,y2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
  cost=cost+9;
end%while
cosalpha2=sign*x2; % in documentation sin
sinalpha2=y2; %documentation cos
% end of second side rotation for angle
plot(cosalpha2,sinalpha2,'Color', [0.0, 0.0, 1.0], 'Marker', '+');

% to speed up convergence
%x1=0.5*cosalpha1;
%y1=0.5*sinalpha1;
%x2=0.5*cosalpha2;
%y2=0.5*sinalpha2;
%sign=(D-(abs(x1)+abs(x2)))/abs(D-(abs(x1)+abs(x2)));
% end of speed up

figure(3);
hold on;
title('MS Location from Alpha1 & Alpha 2 on X Y axis');
axis([-16 16 -16 16]);
grid on;
xlabel('X');
ylabel('Y');

sign=1;
k=4;
increment=2^-k;
if ((alpha2)>=alpha1)
while ((D-(x1+x2))>err)
      ox2=x2;
      oy2=y2;
      ox1=x1;
      oy1=y1;
      x2=x2+sign*increment*cosalpha2;
      y2=y2+sign*increment*sinalpha2;
      plot(x2,y2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
      cost=cost+4;
while ((y2-y1)>err)
        x1=x1+sign*increment*cosalpha1;
        y1=y1+sign*increment*sinalpha1;
        plot(x1,y1,'Color', [0.0, 1.0, 0.0], 'Marker', '+');
        cost=cost+4;
end% inner while
%dynaaproach
dif=D-(x1+x2);
if ((dif<0) && (k<11))
            k=k+1;
            increment=2^-k;
            x2=ox2;
            y2=oy2;
            x1=ox1;
            y1=oy1;
            cost=cost+2;
end
% end of dyna approach
end% while
else%if
```

```
while ((D-(x1+x2))>err)
      ox1=x1;
      oy1=y1;
      ox2=x2;
      oy2=y2;
      x1=x1+sign*increment*cosalpha1;
      y1=y1+sign*increment*sinalpha1;
      plot(x1,y1,'Color', [0.0, 1.0, 0.0], 'Marker', '+');
      cost=cost+4;
while ((y1-y2)>err)
        x2=x2+sign*increment*cosalpha2;
        y2=y2+sign*increment*sinalpha2;
        plot(x2,y2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
        cost=cost+4;
end% inner while
%dynaaproach
dif=abs(D-(x1+x2));
if ((dif<0) && (k<11))
            k=k+1;
            increment=2^-k;
            x2=ox2;
            y2=oy2;
            x1=ox1;
            y1=oy1;
            cost=cost+2;
end
end% while
end%if


% 3d z-side rotation for angle x1,y1
figure(4);
hold on;
title('3rd rotation');
axis([-1 1 -1 1]);
grid on;
xlabel('X');
ylabel('Z');
xx1=1;
z1=0;
ro=0;
if (alpha3>pi/2)
    sign=-1;
    delta=pi-alpha3;
else
    sign=1;
    delta=alpha3;
end

k=4;
while((delta-ro)>=err)
ro=ro+2^-k;
if ((abs(delta-ro)<2^-k) && (k<11))
ro=ro-2^-k;
     k=k+1;
ro=ro+2^-k;
     cost=cost+1+2+2;%1 for k +2 for ro +2 for u
end
 v=2^-k;
 u=2^-(2*k+1);
 oldxx1=xx1;
```

```
 xx1=xx1-xx1*u-z1*v;
 z1=z1-z1*u+oldxx1*v;
 plot(xx1,z1,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
 cost=cost+9;
end%while
cosalpha3=sign*xx1;
sinalpha3=z1;
plot(cosalpha3,sinalpha3,'Color', [0.0, 0.0, 1.0], 'Marker', '+');

% end of 3d-z side rotation for angle

figure(5);
hold on;
title('MS Location from Alpha3 Up to Z using Ms location foun d from
Alpha1 & Alpha 2 for X axis');
axis([-16 16 -16 16]);
grid on;
xlabel('X');
ylabel('Z');
plot(0,x1,'Color', [0.0, 1.0, 0.0], 'Marker', '+');
plot(0,x2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');

%finding rotation for 3d z-dimension
      xx1=cosalpha3;
      z1=sinalpha3;
      sign=1;
      k=4;
while ((x1-xx1)>err)
if ((abs(x1-xx1)<((2^-k)*cosalpha3)) && (k<11))
          k=k+1;
          cost=cost+1+1; % last +1 is for the next increment
end
         increment=2^-k;
        xx1=xx1+sign*increment*cosalpha3;
        z1=z1+sign*increment*sinalpha3;
        plot(xx1,z1,'Color', [0.0, 0.0, 1.0], 'Marker', '+');
        cost=cost+4;
end% inner while
%end finding rotation for 3d z-dimension
% manual calculation
mcost=0;
v1=D*sin(alpha2)/sin(w1);
mcost=mcost+40+404+40+404;
mx1=cos(alpha1)*v1;
mcost=mcost+404+40;
my1=mx1*tan(alpha1);
mcost=mcost+40+1448;

v2=D*sin(alpha1)/sin(w1);
mcost=mcost+40+404+40+404;
mx2=cos(alpha2)*v2;
mcost=mcost+404+40;
my2=mx2*tan(alpha2);
mcost=mcost+40+1448;


mz=mx1*tand(alphainp3);
mcost=mcost+40+1448;
% end of manual calculation
```

```
%errors
Ex1=abs(mx1-x1); % error of X from BS1
Ex2=abs(mx2-x2); % error of X from BS2
Ey1=abs(my1-y1); % error of y from BS1
Ey2=abs(my2-y2); % error of Y from BS2
Ez=abs(mz-z1);   % error of Z from BS1
%end of erros

%average errors
Eavgx1=Eavgx1+Ex1;
Eavgx2=Eavgx2+Ex2;
Eavgy1=Eavgy1+Ey1;
Eavgy2=Eavgy2+Ey2;
Eavgz =Eavgz +Ez;
avgcost=avgcost+cost;
%end of averages

fprintf(fw,'%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t
%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f
\t%2.8f\t%2.8f\t%2.8f\n',alpha1,alpha2,alpha3,alpha4,x1,y1,x2,y2,xx1
,z1,mx1,my1,mx2,my2,mz,Ex1,Ex2,Ey1,Ey2,Ez,cost);

close all;

end% while ~feof(fid)

Eavgx1=Eavgx1/counter;
Eavgx2=Eavgx2/counter;
Eavgy1=Eavgy1/counter;
Eavgy2=Eavgy2/counter;
Eavgz =Eavgz/counter;
avgcost=avgcost/counter;
fprintf(fw,'%s\t%s\t%s\t%s\t%s\t%s\t%s\n','Eavgx1','Eavgx2','Eavgy1'
,'Eavgy2','Eavgz','AvgCost','MCost');
fprintf(fw,'%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\n
',Eavgx1,Eavgx2,Eavgy1,Eavgy2,Eavgz,avgcost,mcost);

fclose(fr);
fclose(fw);
h = msgbox('Operation Completed');
```

Appendix E : Source code of the DMEM-2 Algorithm

```
clc;
clear;
format long;
fr=fopen(strcat(pwd,'\','rnds.txt'),'r');
fw=fopen(strcat(pwd,'\','results2ndDynaver.txt'),'w');
fprintf(fw,'%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t
%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n','a1','
a2','a3','a4','x1','y1','x2','y2','xz1','z1','xz2','z2','mx1','my1',
'mx2','my2','mx3','mz1','mx4','mz2','Ex1','Ex2','Ey1','Ey2','Exz1','
Exz2','Ez1','Ez2','Cost');
counter=0;
Eavgx1=0;
Eavgx2=0;
Eavgy1=0;
Eavgy2=0;
Eavgxz1 =0;
Eavgxz2 =0;
Eavgz1 =0;
Eavgz2 =0;
avgcost=0;
while ~feof(fr)
    cost=0;
    counter=counter+1;
    alphainp1=fscanf(fr,'%d',1);
    alphainp2=fscanf(fr,'%d',1);
    winp1=fscanf(fr,'%d',1); % wont be used
    alphainp3=fscanf(fr,'%d',1);
    alphainp4=fscanf(fr,'%f',1);
    winp2=fscanf(fr,'%f\n',1); %wont be used

%initial inputs
alpha1=(alphainp1*pi)/180;
alpha2=(alphainp2*pi)/180;
w1=(winp1*pi)/180;

alpha3=(alphainp3*pi)/180;
alpha4=(alphainp4*pi)/180;
w2=(winp2*pi)/180;
% pause on; %to stop
D=16; % distance between two base stations. x=8 y=6 hyp=10
%constants
err=10^-5;

% first side rotation for angle x1,y1
x1=1;
y1=0;
ro=0;
if (alpha1>pi/2)
    sign=-1;
    delta=pi-alpha1;
else
    sign=1;
    delta=alpha1;
end
```

```
figure(1);
hold on;
title('1st rotation');
axis([-1 1 -1 1]);
grid on;
xlabel('X');
ylabel('Y');

k=4;
while((delta-ro)>=err)
ro=ro+2^-k;
if ((abs(delta-ro)<2^-k) && (k<11))
ro=ro-2^-k;
     k=k+1;
ro=ro+2^-k;
     cost=cost+1+2+2;%1 for k +2 for ro +2 for u
end
 v=2^-k;
 u=2^-(2*k+1);
 oldx1=x1;
 x1=x1-x1*u-y1*v;
 y1=y1-y1*u+oldx1*v;
 plot(x1,y1,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
 cost=cost+9;
end%while
cosalpha1=sign*x1;
sinalpha1=y1;
 plot(cosalpha1,sinalpha1,'Color', [0.0, 0.0, 1.0], 'Marker', '+');
% end of first side rotation fo angle


% second side rotation for angle x2,y2
figure(2);
hold on;
title('2nd rotation');
axis([-1 1 -1 1]);
grid on;
xlabel('X');
ylabel('Y');
x2=1;
y2=0;
ro=0;
if (alpha2>pi/2)
    sign=-1;
    delta=pi-alpha2;
else
    sign=1;
    delta=alpha2;
end
k=4;
while((delta-ro)>=err)
ro=ro+2^-k;
if ((abs(delta-ro)<2^-k) && (k<11))
ro=ro-2^-k;
     k=k+1;
ro=ro+2^-k;
     cost=cost+1+2+2;%1 for k +2 for ro +2 for u
end
 v=2^-k;
 u=2^-(2*k+1);
 oldx2=x2;
```

```
 x2=x2-x2*u-y2*v;
 y2=y2-y2*u+oldx2*v;

  plot(x2,y2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
  cost=cost+9;
end%while
cosalpha2=sign*x2; % in documentation sin
sinalpha2=y2; %documentation cos
% end of second side rotation for angle
plot(cosalpha2,sinalpha2,'Color', [0.0, 0.0, 1.0], 'Marker', '+');



% to speed up convergence
%x1=0.5*cosalpha1;
%y1=0.5*sinalpha1;
%x2=0.5*cosalpha2;
%y2=0.5*sinalpha2;
%sign=(D-(abs(x1)+abs(x2)))/abs(D-(abs(x1)+abs(x2)));
% end of speed up


figure(3);
hold on;
title('MS Location from Alpha1 & Alpha 2 on X Y axis');
axis([-16 16 -16 16]);
grid on;
xlabel('X');
ylabel('Y');

sign=1;
k=4;
increment=2^-k;
if ((alpha2)>=alpha1)
while ((D-(x1+x2))>err)
      ox2=x2;
      oy2=y2;
      ox1=x1;
      oy1=y1;
      x2=x2+sign*increment*cosalpha2;
      y2=y2+sign*increment*sinalpha2;
      plot(x2,y2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
      cost=cost+4;
while ((y2-y1)>err)
        x1=x1+sign*increment*cosalpha1;
        y1=y1+sign*increment*sinalpha1;
        plot(x1,y1,'Color', [0.0, 1.0, 0.0], 'Marker', '+');
        cost=cost+4;
end% inner while
%dynaaproach
dif=D-(x1+x2);
if ((dif<0) && (k<11))
            k=k+1;
            increment=2^-k;
            x2=ox2;
            y2=oy2;
            x1=ox1;
            y1=oy1;
            cost=cost+2;
end
```

```
% end of dyna approach
end% while
else%if
while ((D-(x1+x2))>err)
      ox1=x1;
      oy1=y1;
      ox2=x2;
      oy2=y2;
      x1=x1+sign*increment*cosalpha1;
      y1=y1+sign*increment*sinalpha1;
      plot(x1,y1,'Color', [0.0, 1.0, 0.0], 'Marker', '+');
      cost=cost+4;
while ((y1-y2)>err)
        x2=x2+sign*increment*cosalpha2;
        y2=y2+sign*increment*sinalpha2;
        plot(x2,y2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
        cost=cost+4;
end% inner while
%dynaaproach
dif=abs(D-(x1+x2));
if ((dif<0) && (k<11))
            k=k+1;
            increment=2^-k;
            x2=ox2;
            y2=oy2;
            x1=ox1;
            y1=oy1;
            cost=cost+2;
end
end% while
end%if



% 3d z-side rotation for angle x1,y1
% first side rotation for angle xz1,z1
xz1=1;
z1=0;
ro=0;
if (alpha3>pi/2)
    sign=-1;
    delta=pi-alpha3;
else
    sign=1;
    delta=alpha3;
end

figure(4);
hold on;
title('1st rotation (X-Z)');
axis([-1 1 -1 1]);
grid on;
xlabel('X');
ylabel('Z');
k=4;
while((delta-ro)>=err)
ro=ro+2^-k;
if ((abs(delta-ro)<2^-k) && (k<11))
ro=ro-2^-k;
     k=k+1;
ro=ro+2^-k;
```

```
     cost=cost+1+2+2;%1 for k +2 for ro +2 for u
end
  v=2^-k;
  u=2^-(2*k+1);

 oldxz1=xz1;
 xz1=xz1-xz1*u-z1*v;
 z1=z1-z1*u+oldxz1*v;

 plot(xz1,z1,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
 cost=cost+9;
end%while
cosalpha3=sign*xz1;
sinalpha3=z1;
 plot(cosalpha3,sinalpha3,'Color', [0.0, 0.0, 1.0], 'Marker', '+');
% end of first side rotation for angle

% second side rotation for angle xz2,z2
figure(5);
hold on;
title('2nd rotation');
axis([-1 1 -1 1]);
grid on;
xlabel('X');
ylabel('Z');
xz2=1;
z2=0;
ro=0;
if (alpha4>pi/2)
    sign=-1;
    delta=pi-alpha4;
else
    sign=1;
    delta=alpha4;
end
k=4;
while((delta-ro)>=err)
ro=ro+2^-k;
if ((abs(delta-ro)<2^-k) && (k<11))
ro=ro-2^-k;
     k=k+1;
ro=ro+2^-k;
     cost=cost+1+2+2;%1 for k +2 for ro +2 for u
end
  v=2^-k;
  u=2^-(2*k+1);
 oldxz2=xz2;
 xz2=xz2-xz2*u-z2*v;
 z2=z2-z2*u+oldxz2*v;
  plot(xz2,z2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
  cost=cost+9;
end%while
cosalpha4=sign*xz2; % in documentation sin
sinalpha4=z2; %documentation cos
% end of second side rotation for angle
plot(cosalpha4,sinalpha4,'Color', [0.0, 0.0, 1.0], 'Marker', '+');


% to speed up convergence
%xz1=0.5*cosalpha3;
```

```
%z1=0.5*sinalpha3;
%xz2=0.5*cosalpha4;
%z2=0.5*sinalpha4;
%sign=(D-(abs(xz1)+abs(xz2)))/abs(D-(abs(xz1)+abs(xz2)));
% end of speed up


figure(6);
hold on;
title('MS Location from Alpha3 & Alpha 4 on X Z axis');
axis([-16 16 -16 16]);
grid on;
xlabel('X');
ylabel('Z');

sign=1;
k=4;
increment=2^-k;
if ((alpha4)>=alpha3)
while ((D-(xz1+xz2))>err)
      oxz2=xz2;
      oz2=z2;
      oxz1=xz1;
      oz1=z1;
      xz2=xz2+sign*increment*cosalpha4;
      z2=z2+sign*increment*sinalpha4;
      plot(xz2,z2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
      cost=cost+4;
while ((z2-z1)>err)
        xz1=xz1+sign*increment*cosalpha3;
        z1=z1+sign*increment*sinalpha3;
        plot(xz1,z1,'Color', [0.0, 1.0, 0.0], 'Marker', '+');
        cost=cost+4;
end% inner while
%dynaaproach
dif=D-(xz1+xz2);
if ((dif<0) && (k<11))
            k=k+1;
            increment=2^-k;
            xz2=oxz2;
            z2=oz2;
            xz1=oxz1;
            z1=oz1;
            cost=cost+2;
end
% end of dyna approach
end% while
else%if
while ((D-(xz1+xz2))>err)
      oxz1=xz1;
      oz1=z1;
      oxz2=xz2;
      oz2=z2;
      xz1=xz1+sign*increment*cosalpha3;
      z1=z1+sign*increment*sinalpha3;
      plot(xz1,z1,'Color', [0.0, 1.0, 0.0], 'Marker', '+');
      cost=cost+4;
while ((z1-z2)>err)
        xz2=xz2+sign*increment*cosalpha4;
        z2=z2+sign*increment*sinalpha4;
        plot(xz2,z2,'Color', [1.0, 0.0, 0.0], 'Marker', '+');
```

```
        cost=cost+4;
end% inner while
%dynaaproach
dif=D-(xz1+xz2);
if ((dif<0) && (k<11))
            k=k+1;
            increment=2^-k;
            xz2=oxz2;
            z2=oz2;
            xz1=oxz1;
            z1=oz1;
            cost=cost+2;
end
% end of dyna approach
end% while
end%if
%end finding rotation for 3d z-dimension

% manual calculation
mcost=0;
v1=D*sin(alpha2)/sin(w1);
mcost=mcost+40+404+40+404;
mx1=cos(alpha1)*v1;
mcost=mcost+404+40;
my1=mx1*tan(alpha1);
mcost=mcost+1448+40;

v2=D*sin(alpha1)/sin(w1);
mcost=mcost+40+404+40+404;
mx2=cos(alpha2)*v2;
mcost=mcost+404+40;
my2=mx2*tan(alpha2);
mcost=mcost+1448+40;

v3=D*sin(alpha4)/sin(w2);
mcost=mcost+40+404+40+404;
mx3=cos(alpha3)*v3;
mcost=mcost+404+40;
mz1=mx3*tan(alpha3);
mcost=mcost+1448+40;

v4=D*sin(alpha3)/sin(w2);
mcost=mcost+40+404+40+404;
mx4=cos(alpha4)*v4;
mcost=mcost+404+40;
mz2=mx4*tan(alpha4);
mcost=mcost+1448+40;
% end of manual calculation
%errors
Ex1=abs(mx1-x1); % error of X from BS1
Ex2=abs(mx2-x2); % error of X from BS2
Ey1=abs(my1-y1); % error of y from BS1
Ey2=abs(my2-y2); % error of Y from BS2

Exz1=abs(mx3-xz1);    % error of X from BS1
Exz2=abs(mx4-xz2);    % error of X from BS2
Ez1=abs(mz1-z1);    % error of Z from BS1
Ez2=abs(mz2-z2);    % error of Z from BS2
%end of erros
```

```
%average errors
Eavgx1=Eavgx1+Ex1;
Eavgx2=Eavgx2+Ex2;
Eavgy1=Eavgy1+Ey1;
Eavgy2=Eavgy2+Ey2;
Eavgxz1 =Eavgxz1 +Exz1;
Eavgxz2 =Eavgxz2 +Exz2;
Eavgz1 =Eavgz1 +Ez1;
Eavgz2 =Eavgz2 +Ez2;
avgcost=avgcost+cost;
%end of averages

fprintf(fw,'%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t
%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f
\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.
8f\t%2.8f\n',alpha1,alpha2,alpha3,alpha4,x1,y1,x2,y2,xz1,z1,xz2,z2,m
x1,my1,mx2,my2,mx3,mz1,mx4,mz2,Ex1,Ex2,Ey1,Ey1,Exz1,Exz2,Ez1,Ez2,cos
t);

close all;
end% while ~feof(fid)
Eavgx1=Eavgx1/counter;
Eavgx2=Eavgx2/counter;
Eavgy1=Eavgy1/counter;
Eavgy2=Eavgy2/counter;
Eavgxz1=Eavgxz1/counter;
Eavgxz2=Eavgxz2/counter;
Eavgz1=Eavgz1/counter;
Eavgz2=Eavgz2/counter;
avgcost=avgcost/counter;

fprintf(fw,'%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n','Eavgx1','Eavg
x2','Eavgy1','Eavgy2','Eavgxz1','Eavgxz2','Eavgz1','Eavgz2','AvgCost
','Mcost');
fprintf(fw,'%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t%2.8f\t
%2.8f\t%2.8f\t%2.8f\n',Eavgx1,Eavgx2,Eavgy1,Eavgy2,Eavgxz1,Eavgxz2,E
avgz1,Eavgz2,avgcost,mcost);
fclose(fr);
fclose(fw);

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