POSITIONING ALGORITHM FOR CELLULAR NETWORKS USING TIME DIFFERENCE OF ARRIVAL

SUBMITTED TO IITM

A Dual Degree Project Report

submitted by

ALEKHYA TUMMA

EE11B060

for the award of the degree

of

DUAL DEGREE IN ELECTRICAL ENGINEERING

(With specialization in Communications)

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2016

THESIS CERTIFICATE

This is to certify that the thesis titled **POSITIONING ALGORITHM FOR CELLULAR NETWORKS USING TIME DIFFERENCE OF ARRIVAL**, submitted by **Alekhya Tumma (EE11B060)**, to the Indian Institute of Technology Madras, Chennai for the award of the degree of **Dual Degree in Electrical Engineering with specialization in Communications**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. 1
Dr. Arun Pachai Kannu
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 20th May 2016

ABSTRACT

Locating the position of a mobile user with a high degree of accuracy is a research interest that holds the key to a breakthrough in many service challenges faced by operators in the wireless communication world. The benefits of success in this field ranges from value added services and efficient advertising to crime detection and fighting. Many technologies have been developed which made use of different algorithms to provide answers to these challenges but with varying degrees of accuracy, operational challenges, ease of deployment and cost of installation. Mobile position estimation technologies use techniques such as Time Difference of Arrival (TDOA), Angle of Arrival (AOA), Time of Arrival (TOA), Received Signal Strength (RSS) indication, GPS systems, and Cell ID. The interest of this research is to investigate the performance of positioning algorithms in wireless cellular networks based on time difference of arrival (TDOA) measurements provided by the base stations. The localization process of the mobile station results in a non-linear least squares estimation problem which cannot be solved analytically. Therefore, we use iterative algorithms to determine an estimate of the mobile station position. The well-known Gauss-Newton method fails to converge for certain geometric constellations, and thus, it is not suitable for a general solution in cellular networks. Another algorithm is the steepest descent method which has a slow convergence in the final iteration steps. Levenberg-Marquardt method acts more like a gradient-descent method when the parameters are far from their optimal value, and acts more like the Gauss-Newton method when the parameters are close to their optimal value. Hence, we apply the Levenberg-Marquardt algorithm as a new approach in the cellular network localization framework. We show that this method meets the best trade-off between accuracy and computational complexity.

ACKNOWLEDGMENTS

I owe my gratitude to all those people who have made this dissertation possible and because of whom my project experience has been one that I will cherish forever.
My deepest gratitude is to my mentor, Prof. Dr. Arun Pachai Kannu. I have been amazingly fortunate to have mentor who gave me the freedom to explore on my own and at the same time the guidance to recover when my steps faltered. This taught me how to question thoughts and express ideas. His patience and support helped me overcome many crisis situations and finish this dissertation. Above all, I would like to thank my parents, friends who were always there supporting and encouraging me, which help me in completion of this project. Last but not the least; I thank God Almighty for making everything possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I: Introduction

With the astonishing growth of wireless technologies, the requirement of providing universal location services by wireless technologies is growing. Positioning in wireless networks became very important in recent years and services and applications based on accurate knowledge of the location of the Mobile Station (MS) will play a fundamental role in future wireless systems. In addition to vehicle navigation, fraud detection, resource management, automated billing, it is stated by the Federal Communications Commission (FCC) that all wireless service providers have to deliver the location of all Enhanced 911 (E911) callers with specified accuracy. The process of obtaining a terminal's location by exploiting wireless network infrastructure and utilizing wireless communication technologies is called wireless positioning. Mobile station location usually implies the coordinates of the Mobile station that may be in two or three dimensions, and usually includes information such as the latitude and longitude of where it is located. Such information is made available by measuring some properties of the radio signals transmitted or received by the cellular phone. Mobile Station localization using Global Navigation Satellite Systems (GNSSs) such as the Global Positioning System (GPS) deliver very accurate position information only under good environmental conditions and consume high power.

Therefore, exploiting already available resources of the cellular networks gained interest. Generally the Time of Arrival (ToA), Time Difference of Arrival (TDoA), Received Signal Strength (RSS) and Angle of Arrival (AoA) obtained from the Base Stations (BS) are the parameters used in the process of localization or positioning. Using these metrics the whole problem of positioning turns into a minimization problem where the minimum of a cost function gives the accurate position of the mobile station.

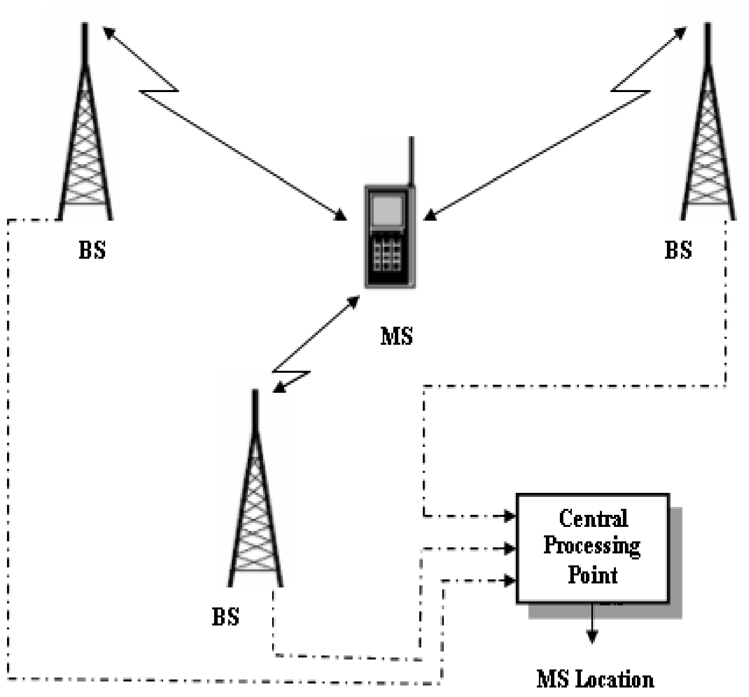

Figure 1: Layout of a cellular network for localization if mobile station

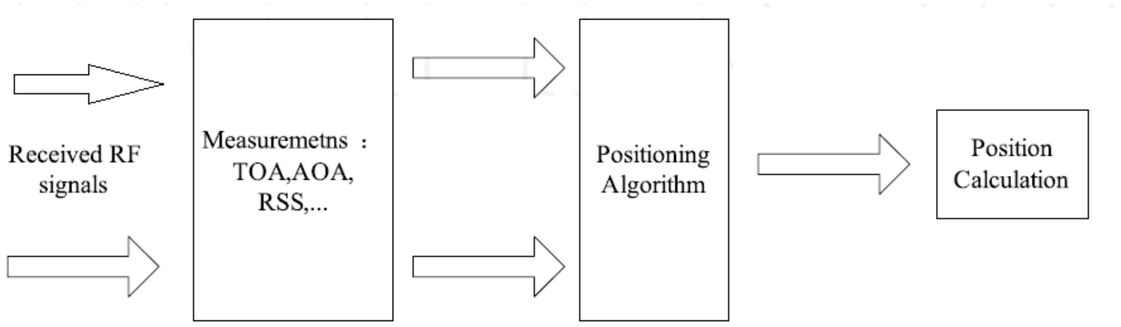

Figure 2: Basic elements of a wireless positioning system

To cope with the resulting non-linear estimation problem where no analytical solution is possible, the iterative methods such as Gauss-Newton (GN) and the Steepest Descent (SD) method are used as standard solutions. The well-known Gauss-Newton method fails to converge for certain geometric constellations, and thus, it is not suitable for a general solution in cellular networks. Steepest descent method has a slow convergence in the final iteration steps. Levenberg-Marquardt method acts more like a gradient-descent method when the parameters are far from their optimal value, and acts more like the Gauss-Newton method when the parameters are close to their optimal value. Hence, we apply the Levenberg-Marquardt algorithm as a new approach in the cellular network localization framework. This method meets the best trade-off between accuracy and computational complexity. Therefore, the iterative Levenberg-Marquardt (LM) algorithm is implemented in the context of positioning in cellular networks using Time Difference of Arrival as the metric. The variation in accuracy of estimating the mobile station position with respect to increase in the number of base stations, different noise powers and the robustness of the algorithm is analyzed.

CHAPTER II: Background and Literature Review

Positioning of a mobile station can be achieved considering already available information from the cellular network such as Cell ID, Angle of Arrival (AoA), Received Signal Strength (RSS), Time of Arrival (ToA) and Time Difference of Arrival (TDoA).
The positioning techniques using these parameters can be categorized into two groups:

- **Signal Strength and network parameter based techniques**

  This technique uses Cell ID and Received Signal Strength (RSS) as parameters.

a. Cell ID

Positioning algorithms using Cell ID as the metric to estimate the location of the mobile station converts the ID of the cell of the base station to which the mobile station is connected to a geographic position. This the most primitive way of localization. These algorithms can only estimate the position of the cell where the base station to which the mobile station is registered. Accuracy of the estimate depends on the cell size, cell type and range of the serving base station.

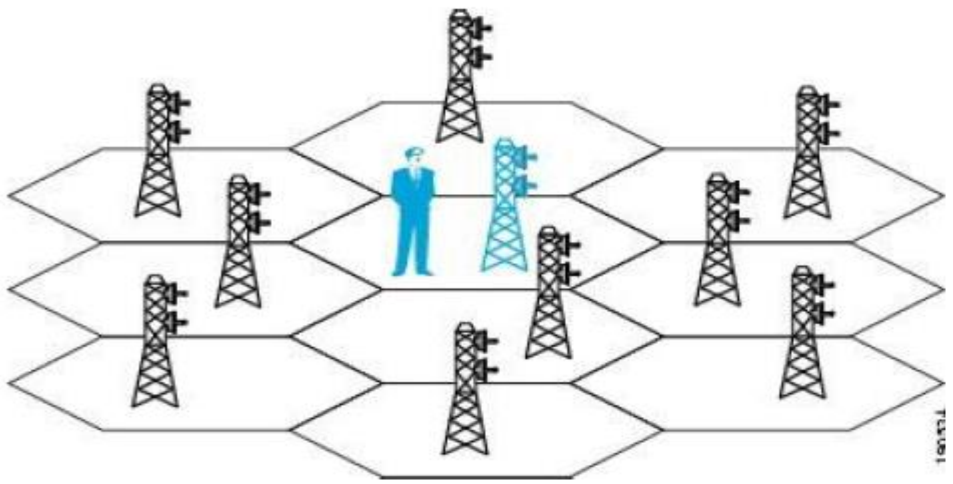

Figure 3: Location of the cell with the serving base station (in blue) is estimated

b. Received Signal Strength (RSS)

The distance between the base station and mobile station is calculated based on the signal power transmitted by the base station, its location and the type of channel through which it is transmitted.

- **Triangulation or Trilateration**

  This technique uses concepts of trigonometry and geometry to estimate the position. Hence, it uses Angle of Arrival, Time of Arrival and Time Difference of Arrival as the metrics.

a. Angle of Arrival

These methods are also referred to as Direction of Arrival methods. Angle of Arrival method uses multi array antennas to estimate the direction of arrival of the signal. Therefore, one AoA measurement give the coordinate of the mobile station along one axis in an n dimensional space. Hence, using multi array antennas gives a complete estimate of the mobile station position. Angle of Arrival method gives high accuracy only under the condition of lie of sight. Decrease in accuracy is observed in areas with high level of scattering.

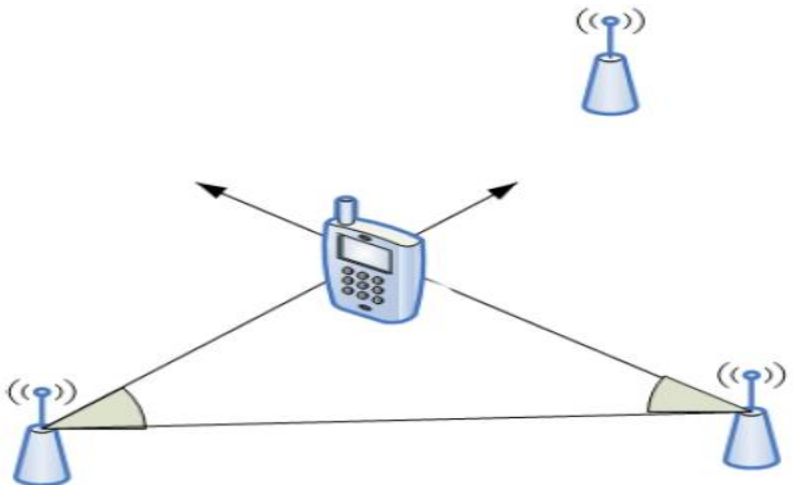

Figure 4: Angle of Arrival Technique

b. Time of Arrival

In Time of Arrival method the time taken by the signal from the mobile station to arrive at different base stations is observed. As time of arrival at a base station is the distance between the base station and mobile station scaled by the speed of the signal or light, the distance between base station and mobile station can be considered equal to the time of arrival. The position of the mobile station can be estimated by finding the point of intersection of circles considering these distances as the radius and base stations as the centers of the circles.

Time of Arrival technique is widely used but for this technique to give accurate results all the base stations and the mobile station should be synchronized in time which is difficult to achieve in practical cellular network. Using difference in Time of Arrivals eliminates the problem of time synchronization. Hence, Time Difference of Arrival gained popularity and is more efficient.

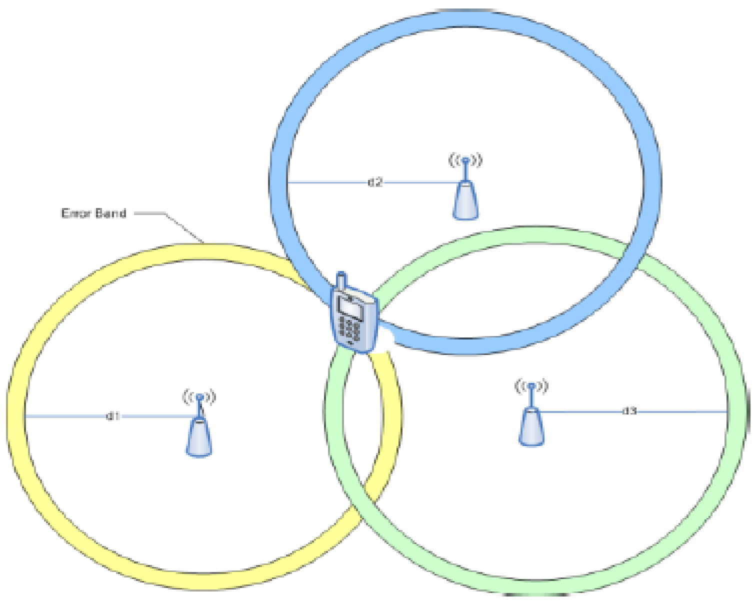

Figure 5: Triangulation technique for Time of Arrival

c. **Time Difference of Arrival**

In Time Difference of Arrival method one particular base station is considered as the reference base station and all the differences in time of arrivals of the signals received at the base stations are computed with respect to the reference base station. The information about the locations of all the base stations is available. This eliminates the problem of time synchronization experienced in Time of Arrival technique. Therefore, each time difference of arrival represents a hyperbolic curve on which the mobile station resides. Intersection of these hyperbolic curves gives the position of the mobile station. As we are considering difference in the time of arrivals a minimum number of three base stations are necessary to find the mobile station location.

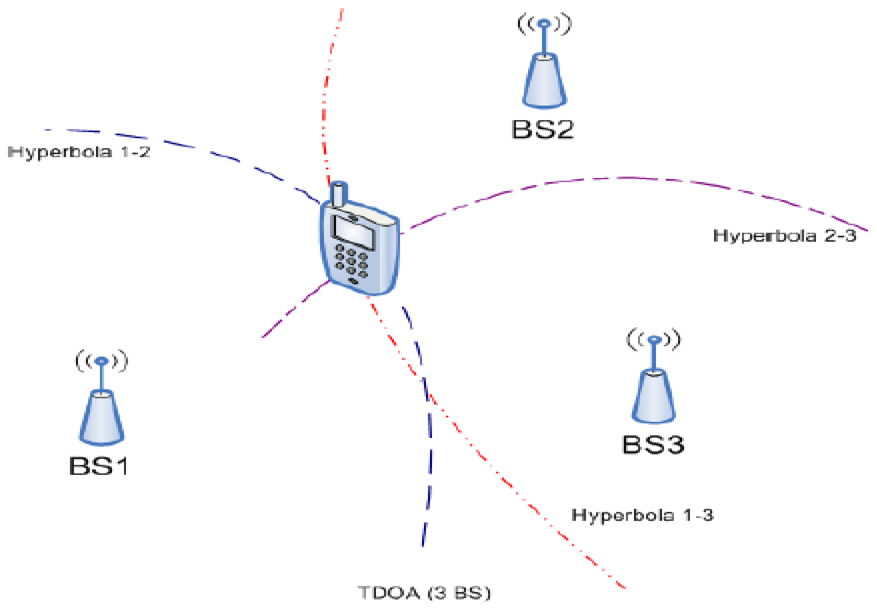

Figure 6: Time Difference of Arrival Technique

Advantages of using Time Difference of Arrivals

1. Eliminates the problem of time synchronization between base stations and mobile stations which is present when only Time of Arrivals are considered.
2. If a major reflector affects the signal components going to all the receivers, the timing error may get cancelled or reduced in the time difference operation.
3. All changes take place only at the infrastructure level.

Using TDoA measurements to find the location of mobile station:

Consider a cellular network with cell radius R with time synchronized base stations. The mobile station is located at X.

$$X = [\, x \, , y \,]^T$$

N_{bs} be the number of base stations nearest to the mobile station. Let the positions of the base stations be denoted by X_v where $v \in \{1,2, \ldots .. N_{BS}\}$.

$$X_v = [\ x_v\ ,\ y_v\ ]^T$$

The distance between each base station and mobile station is given by r_v

$$r_v = \left||X_v - X|\right| = \sqrt{(x_v - x)^2 + (y_v - y)^2} \quad \text{----------------------} \ (1)$$

Considering BS1 as the reference base station and the distances and TDoAs as equivalent, the time difference of arrival for vth base station be $d_{v,1}$

$$d_{v,1}(X) = \left(r_v(X) - r_1(X)\right)$$
$$= \sqrt{(x_v - x)^2 + (y_v - y)^2} - \sqrt{(x_1 - x)^2 + (y_1 - y)^2} \quad \text{----------------------} \ (2)$$

As BS1 is the reference base station we get $N_{BS} - 1$ independent equations for the time difference of arrivals.

$$d(X) = \ [d_{2,1}(X), d_{3,1}(X), \ldots \ldots \ldots d_{N_{BS},1}(X)]^T \quad \text{----------------------} \ (3)$$

The corresponding observed TDoAs information obtained from the cellular network is represented by the matrix‘d’.

$$d = \ [d_{2,1}, d_{3,1}, \ldots \ldots \ldots d_{N_{BS},1}]^T \quad \text{----------------------} \ (4)$$

$$d = d(X) + n \quad \text{----------------------} \ (5)$$

Where $n = \ [n_{2,1}, n_{3,1} \ldots \ldots \ldots n_{N_{BS},1}]^T$ is the zero mean Additive White Gaussian Noise (AWGN) with covariance matrix

$$\textstyle\sum_n = \ E\{nn^T\} = Variance \quad \text{----------------------} \ (6)$$

(Since the mean is zero)

Positioning Algorithms

The algorithms considered for positioning of wireless mobile station using time difference of arrivals result in a nonlinear square equations which require iterative methods to find the location of the mobile station. With the system model of TDoA described and applying weighted nonlinear least square approach the cost function to be minimized for the estimation of the location of mobile station is

$$\varepsilon(X) = \left(d - d(X)\right)^T \sum\nolimits_n^{-1} (d - d(X)) \qquad \text{----------------------} \ (7)$$

$$= d^T \sum\nolimits_n^{-1} d - 2d^T \sum\nolimits_n^{-1} d(X) + d(X)^T \sum\nolimits_n^{-1} d(X)$$

With respect to X yielding

$$X = \operatorname{argmin}_x \varepsilon\,(X) \qquad \text{----------------------} \ (8)$$

In the general case, there exists no closed-form solution to the non-linear two-dimensional optimization problem given by the above equation, and hence, iterative approaches that are necessary in solving are

1. Gauss Newton algorithm
2. Steepest Descent algorithm
3. Levenberg-Marquardt algorithm

All methods for non-linear optimization are iterative: From a starting point X_0 the method produces a series of vectors X_1, X_2, …..which converges to X^*, a local minimizer for the given function. Most methods have measures which enforce the descending condition

$$F(X_{K+1}) < F(X_K)$$

This prevents convergence to a maximizer and also makes it less probable that we converge towards a saddle point. If the given function has several minimizers the result will depend on the starting point X_0. Which of the minimum value is found is not known and hence, the minimum value found is not necessarily the minimum value closest to X_0. In many cases the method produces vectors which converge towards the minimum in two clearly different stages. When X_0 is far from the solution the method should produce iterates which move steadily towards X^*. In this "global stage" of the iteration we are satisfied if the errors do not increase except in the very first steps, ie

$$\left||e_{K+1}(X)\right|| < ||e_K(X)|| \ \text{ For } K > K_{max} \qquad \text{----------------------} \ (9)$$

$$\text{Where } e_K(X) = X_K - X^*$$

In the final stage of the iteration where X_K *is closer to* X^* faster convergence of the algorithm is desired.

Steepest Descent algorithm

The steepest descent method is a general minimization method which updates parameter values in the direction opposite to the gradient of the objective function. It is recognized as a highly convergent algorithm for finding the minimum of simple objective functions. For problems with thousands of parameters, gradient descent methods are sometimes the only viable method.

The minimum of function is where the gradient of the function goes to zero. The gradient of the cost function derived from the TDoA is

$$\begin{aligned}\nabla\varepsilon(X) &= \nabla\left(\left(d - d(X)\right)^T \textstyle\sum_n^{-1} \left(d - d(X)\right)\right) \\ &= -\left(d - d(X)\right)^T \textstyle\sum_n^{-1} \left(\nabla d(X)\right) \\ &= -\left(d - d(X)\right)^T \textstyle\sum_n^{-1} J \qquad \text{---------------------- (10)}\end{aligned}$$

Where J = ∇ d(X)

$$J = \begin{bmatrix} \frac{x-x_2}{r_2} - \frac{x-x_1}{r_1} & & \frac{y-y_2}{r_2} - \frac{y-y_1}{r_1} \\ \frac{x-x_3}{r_3} - \frac{x-x_1}{r_1} & & \frac{y-y_3}{r_3} - \frac{y-y_1}{r_1} \\ \frac{x-x_4}{r_4} - \frac{x-x_1}{r_1} & & \frac{y-y_4}{r_4} - \frac{y-y_1}{r_1} \\ & . & \\ & . & \\ & . & \\ & . & \\ \frac{x-x_n}{r_n} - \frac{x-x_1}{r_1} & & \frac{y-y_n}{r_n} - \frac{y-y_1}{r_1} \end{bmatrix} \qquad \text{---------------------- (11)}$$

The Jacobian matrix J represents the sensitivity of $\varepsilon(X)$ to the parameters in X. Based on this descent direction and considering a step size of **'μ'** the perturbation **'h'** that moves the parameters of X in the steepest descent direction is

$$h = \mu(\nabla\varepsilon(X)) \qquad \text{Where } \mu>0$$

$$= -\mu\left(d - d(X)\right)^T \textstyle\sum_n^{-1} J$$

Therefore the value of X in the next iteration is given by

$$X_{K+1} = X_K - \mu\left(d - d(X)\right)^T \textstyle\sum_n^{-1} J \qquad \text{---------------------- (12)}$$

The easiest way to find a step size is to choose a constant μ for all iteration steps. The optimum step size for an iteration K is given by optimum line search algorithms which use nonlinear one dimensional optimization.

$$\mu_K = argmin_\mu \varepsilon(X_{K+1}) \qquad \text{---------------------- (13)}$$

Drawbacks of Steepest Descent method

- High computational effort in evaluating the step size μ using nonlinear one dimensional optimization.
- Slow convergence in the final iteration steps of the algorithm.
- Due to the slow convergence in the final iteration steps there is a possibility to run out of local minima.
- The curvature of the error surface may not be the same in all directions. For example, if there is a long and narrow valley in the error surface, the component of the gradient in the direction that points along the base of the valley is very small while the component along the valley walls is large. This results in motion more in the direction of the walls even though we have to move a long distance along the base and small distance along the walls. This is called the **"Error valley problem"**

Gauss Newton algorithm

Like the steepest descent algorithm, Gauss Newton algorithm also uses the weighted nonlinear least square model of the cost function obtained using TDoAs. The whole basis of Gauss Newton algorithm is using the Taylor series expansion to linearize the TDoA vector obtained in equation (3).

Applying the Taylor series expansion to the vector d(X) we get

$$d(X+h) = d(X) + \nabla d(X)(h) \qquad \text{---------------------- (14)}$$

$$= d(X) + \mathrm{J}(h) \qquad \text{Where } \mathrm{J}=\nabla d(X)$$

Substituting the above expression for d(X) in equation (7)

$$\varepsilon(X+h) = d^T \textstyle\sum_n^{-1} d + d(X)^T \sum_n^{-1} d(X) - 2d^T \sum_n^{-1} d(X) \quad \text{---------------------- (15)}$$
$$-2\big(d-d(X)\big)^T \textstyle\sum_n^{-1} Jh + h^T J^T \sum_n^{-1} Jh$$

This shows that the cost function is quadratic in terms of the perturbation h. The perturbation h that minimizes the cost function is obtained by equating the gradient of the cost function to zero.

$$\partial \varepsilon / \partial \mathrm{h} = 0$$

$$L'(h) = \frac{\partial \varepsilon(X+h)}{\partial h} = -2\big(d-d(X)\big)^T \textstyle\sum_n^{-1} J + 2h^T J^T \sum_n^{-1} J = 0$$

The gradient and Hessian of L (h) are

$$L'(h) = J^T \textstyle\sum_n^{-1} (d-d(X)) \quad \text{And} \quad L''(h) = J^T \sum_n^{-1} J$$

We can see that the Hessian matrix is independent of h. It is symmetric and if J has full rank, that is the columns are linearly independent then the Hessian is also positive definite. This implies that the cost function $\varepsilon(X)$ has a unique minimum value which can be found by solving

$$[J^T \textstyle\sum_n^{-1} J]h = J^T \sum_n^{-1} (d-d(X))$$
$$h = \left[J^T \textstyle\sum_n^{-1} J\right]^{-1} (J^T \sum_n^{-1} (d-d(X))) \quad \text{---------------------- (16)}$$

Therefore, the value of X in the next iteration is given by

$$X_{K+1} = X_K + \alpha\left[J^T \textstyle\sum_n^{-1} J\right]^{-1} (J^T \sum_n^{-1} (d-d(X))) \quad \text{---------------------- (17)}$$
$$= X_K + \alpha A_K^{-1} g_K$$

In classical Gauss Newton method $\alpha = 1$ in all steps. The method with line search can be shown to have guaranteed convergence provided that

a. $\varepsilon(X) < \varepsilon(X_0)$ is bounded.
b. The jacobian matrix J(X) has full rank in all steps.

Disadvantages of Gauss Newton algorithm

- The Gauss Newton algorithm provides higher convergence rates and higher accuracy only for initial values close to the actual mobile station position.

- For poor initial values and bad geometric conditions the algorithm results in a rank-deficient, and thus, non-invertible matrix A_K for certain constellations of MS and BSs. In this case the algorithm diverges

From the algorithms described above it is observed that the Steepest Descent and Gauss Newton algorithms are complementary to each other. In the cases where Steepest Descent method has slow convergence, Gauss Newton method can be used. To cope up with the disadvantages of Gauss Newton algorithm and Steepest Descent algorithm that is robustness and slow convergence Levenberg-Marquardt algorithm which is a blend of both Gauss Newton and Steepest Descent is used. The following section describes the implementation of Levenberg-Marquardt algorithm.

CHAPTER III: Levenberg-Marquardt algorithm in solving nonlinear least square problem

The Levenberg-Marquardt algorithm is an iterative technique that locates the minimum of a multivariate function that is expressed as the sum of squares of nonlinear real valued functions. It has become a standard technique to solve nonlinear least square problems. The well-known Gauss-Newton method fails to converge for certain geometric constellations, and thus, it is not suitable for a general solution in cellular networks. Steepest descent method has a slow convergence in the final iteration steps. Levenberg-Marquardt method is a blend of both Steepest Descent and Gauss Newton which acts more like a gradient-descent method when the parameters are far from their optimal value, and acts more like the Gauss-Newton method when the parameters are close to their optimal value. Hence, we apply the Levenberg-Marquardt algorithm as a new approach in the cellular network localization framework. This method meets the best trade-off between accuracy and computational complexity. Therefore, the iterative Levenberg-Marquardt (LM) algorithm is implemented in the context of positioning in cellular networks using Time Difference of Arrival as the metric. Levenberg-Marquardt algorithm is also called **"Damped Gauss Newton algorithm"** as the basis of it is to modify the correction term in the Gauss Newton algorithm.

The correction term in the Gauss Newton algorithm is given by

$$h = \left[J^T \textstyle\sum_n^{-1} J\right]^{-1} (J^T \textstyle\sum_n^{-1} (d - d(X)))$$

Where $\left[J^T \sum_n^{-1} J\right]$ is the Hessian matrix i.e. the double derivative of the cost function $\varepsilon(X)$.

Levenberg-Marquardt algorithm updates this correction term by adding a damping parameter **"λ"** such that the algorithm is a blend of both Steepest Descent and Gauss Newton algorithms. Therefore, the update rule proposed by Levenberg-Marquardt is

$$h = \left[J^T \textstyle\sum_n^{-1} J + \lambda I\right]^{-1} (J^T \textstyle\sum_n^{-1} (d - d(X))) \quad \text{---------------------} \; (18)$$

Where I is the Identity matrix.

From the analysis of Gauss Newton method, the convergence of Taylor series expansion and the convergence speed directly depends on the choice of the Mobile station's initial coordinates. This iterative method must start with an initial guess which is in the condition of close to the true solution to avoid local minima. Selection of such a starting point is not simple in practice. To solve this problem, steepest decent method with the properties of fast convergence at the initial iteration and small computation complexity is applied at the first several iterations to get the correct mobile station coordinates. Therefore, Levenberg-Marquardt algorithm assumes a high value for 'λ' initially for it to follow the Steepest Descent method and then on iteration changes the value of 'λ' such that the algorithm behaves like Gauss Newton when the coordinates are close to the actual value.

The damping parameter 'λ' has several effects:

- For all λ>0 the coefficient matrix is positive definite, and this ensures that h is a descent direction.
- For larger values of λ we get

  $$h = \frac{1}{\lambda}((J^T \Sigma_n^{-1} (d - d(X))))$$

  i.e. a short step in the Steepest Descent direction. This is good if the current iterate is far from the solution.
- If λ is very small then the perturbation h is equal to that in the Gauss Newton algorithm which is a good step in the final stages of iteration when X is close to the actual solution X^*. If $\varepsilon(X^*) = 0$ or very small then the algorithm almost as quadratic convergence.

Hence, the damping parameter influences both the direction and size of the step, and this leads us to a method without specific line search as in the case of Gauss Newton. The choice of initial λ value should be related to the size of the Hessian matrix

$$A = J^T \Sigma_n^{-1} J, \qquad \text{e.g. by letting}$$

$$\lambda_0 = \tau . max_i\{A_{ii}^0\} \qquad \text{----------------------} (19)$$

Where τ is chosen by the user.

During iteration the size of λ can be updated as described below. The updating is controlled by the gain ratio or the gain in error ratio

$$\rho = (\varepsilon(X) - \varepsilon(X+h))/(L(0) - L(h)) \qquad \text{----------------------} (20)$$

Where the denominator is the gain predicted by

$$\varepsilon(X+h) = L(h) = \varepsilon(X) \ - 2\big(d - d(X)\big)^T \Sigma_n^{-1} Jh + h^T J^T \Sigma_n^{-1} Jh$$

Therefore,

$$L(0) - L(h) = \big(d - d(X)\big)^T \Sigma_n^{-1} Jh + \left(\tfrac{1}{2}\right) h^T J^T \Sigma_n^{-1} Jh$$

$$= 2h^T(\lambda h + J^T \Sigma_n^{-1} (d - d(X))) \qquad \text{----------------------} (21)$$

A large value of ρ indicates that L (h) is a good approximation to $\varepsilon(X+h)$ and λ is decreased by a factor so that the next Levenberg-Marquardt step is closer to the Gauss Newton step. If ρ is small then L(h) is a poor approximation and we should increase λ with the two fold aim of getting closer to the Steepest Descent direction and reducing the step length. In this way, LM can defensively navigate a region of the parameter space in which the model is highly nonlinear. This algorithm is adaptive as it can control its own damping factor.

The algorithm terminates when at least one of the following conditions is met:

- When the magnitude of the gradient of the cost function is very small or drops below a threshold.
- The relative change in the magnitude of h drops below a threshold or
- When maximum number of iterations is reached.

The above algorithm has the disadvantage that if the value of λ is large, the calculated Hessian matrix is not used at all. We can derive some advantage out of the second derivative even in such cases by scaling each component of the gradient according to the curvature. This should result in larger movement along the directions where the gradient is smaller so that the classic "error valley" problem does not occur any more. This crucial insight was provided by Marquardt. He replaced the identity matrix with the diagonal of the Hessian resulting in the Levenberg-Marquardt update rule. Therefore, the final update rule in Levenberg-Marquardt algorithm is given by

$$h = \left[J^T \Sigma_n^{-1} J + \lambda J^T \Sigma_n^{-1} J\right]^{-1} (J^T \Sigma_n^{-1} (d - d(X))) \qquad \text{----------------------} \ (22)$$

It is to be noted that while the LM method is in no way optimal but is just a heuristic, it works extremely well in practice. The only flaw is its need for matrix inversion as part of the update. Even though the inverse is usually implemented using clever pseudo-inverse methods such as singular value decomposition, the cost of the update becomes prohibitive after the model size increases to a few thousand parameters. For moderately sized models (of a few hundred parameters) however, this method is much faster than say, steepest descent method. The Levenberg-Marquardt algorithm provides fast convergence and is very robust against inaccurate initial values unlike Gauss Newton method. The Levenberg-Marquardt algorithm is summarized below.

LEVENBERG-MARQUARDT ALGORITHM

ALGORITHM

```
1.  K = 0
2.  $V_K = 2$
3.  $A_K = J^T(X_K) \sum_n^{-1} J(X_K)$
4.  $g_K = J^T(X_K) \sum_n^{-1} (d - d(X_K))$
5.  $\lambda_K = \tau . max_i\{A_{ii}^K\}$
6.  Repeat
7.  $h_K = (A_K + \lambda_K A_K)^{-1} g_K$
8.  $X_{K+1} = X_K + h_K$
9.  $\rho_K = (\varepsilon(X_K) - \varepsilon(X_{K+1}))/(h_K^T(\lambda_K h_K + g_K))$
10. $if\ \rho_K > 0\ then$
            $A_{K+1} = J^T(X_{K+1}) \sum_n^{-1} J(X_{K+1})$
            $g_{K+1} = J^T(X_{K+1}) \sum_n^{-1} (d - d(X_{K+1}))$
            $\lambda_{K+1} = \lambda_K \max\{\left(\frac{1}{3}\right), 1 - (2\rho_K - 1)^3\}$
            $V_{K+1} = 2$
    Else
            $\lambda_{K+1} = \lambda_K V_K$
            $V_{K+1} = 2V_K$
11. End if
12. K=K+1
13. Until convergence
```

Table 1: Pseudo code of Levenberg-Marquardt algorithm

CHAPTER IV: RESULTS

The above algorithm is tested in a 1000km x 1000km grid divided into square cells of 100km x 100 km each with base stations located at the center of the cells. A minimum of 3 base stations are required to find the exact location of the mobile station as we need at least two Time Difference of Arrivals to estimate the position of the mobile station. It is assumed that the noise power is constant for all the involved links from base stations to the mobile stations with

$\sum_{n} = \sigma_n^2 I_{N_{BS-1}}$ Where σ_n is the standard deviation of the noise

The locations of the base stations are provided to the algorithm and the corresponding time difference of arrivals are calculated considering first base station as the reference base station. The maximum number of iterations required is 400.

The root mean square error of the estimate is computed using

$$RMSE = \sqrt{E\{||X - X'||\}} \qquad \text{---------------------- (23)}$$

Where X' is the estimate providing after 400 iteration steps.

Error in estimate vs Standard Deviation of noise:

Increasing standard deviation of noise results in increase of error in the estimate. The error considered is the Root mean square error where the error is averaged over 1000 different noise values having the same noise power. Therefore, the root mean square error vs variation curve is a monotonically increasing curve. The following figure represents the relation between root mean square error and standard deviation of noise.

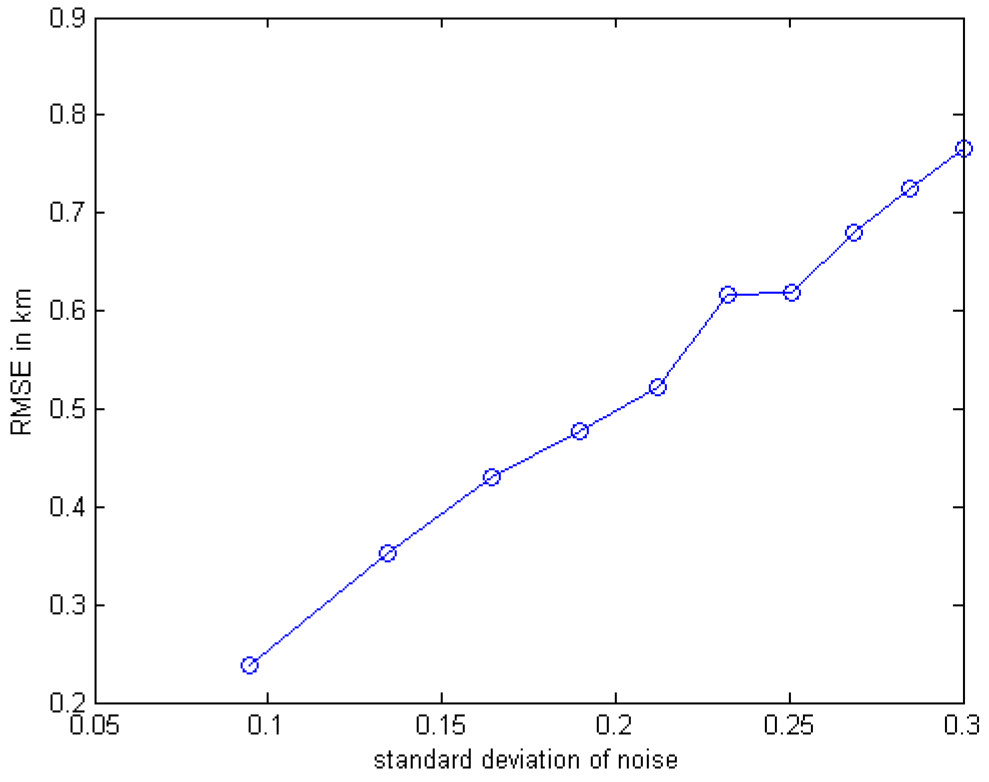

Figure 7: Root mean square error vs standard deviation of noise

Error in estimate vs Number of base stations serving the mobile station:

Increasing the number of base stations serving the mobile station increases the accuracy with which the position is estimated. This is because we have more information about the mobile position compared to earlier case where there are less number of base stations. Hence, root mean square error vs number of base stations serving the mobile station is a monotonically decreasing curve. The following result is obtained by varying the number of base stations from 3 to 13. The error obtained for particular number of base stations is averaged over 1000 noise realizations with a standard deviation of 0.3

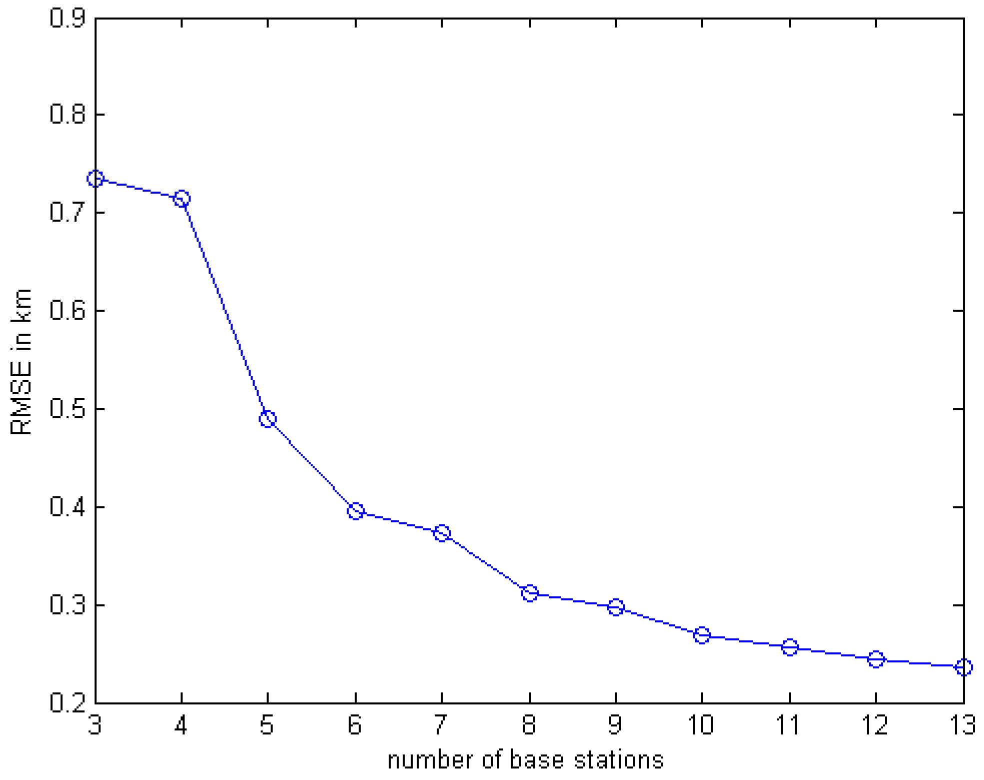

Figure 8: Root mean square error vs number of base stations

Error in estimate vs Number of iterations taken by the algorithm:

With increase in number of iterations the accuracy of the estimate of the position of mobile station increases. After a particular number of iterations the estimate value remains the same despite the increase in number of iterations. This is the final position estimated by the algorithm. As the accuracy of the estimate increases with increase in number of iterations the root mean square error vs number of iterations graph is a monotonically decreasing curve. The following result depicts this relation.

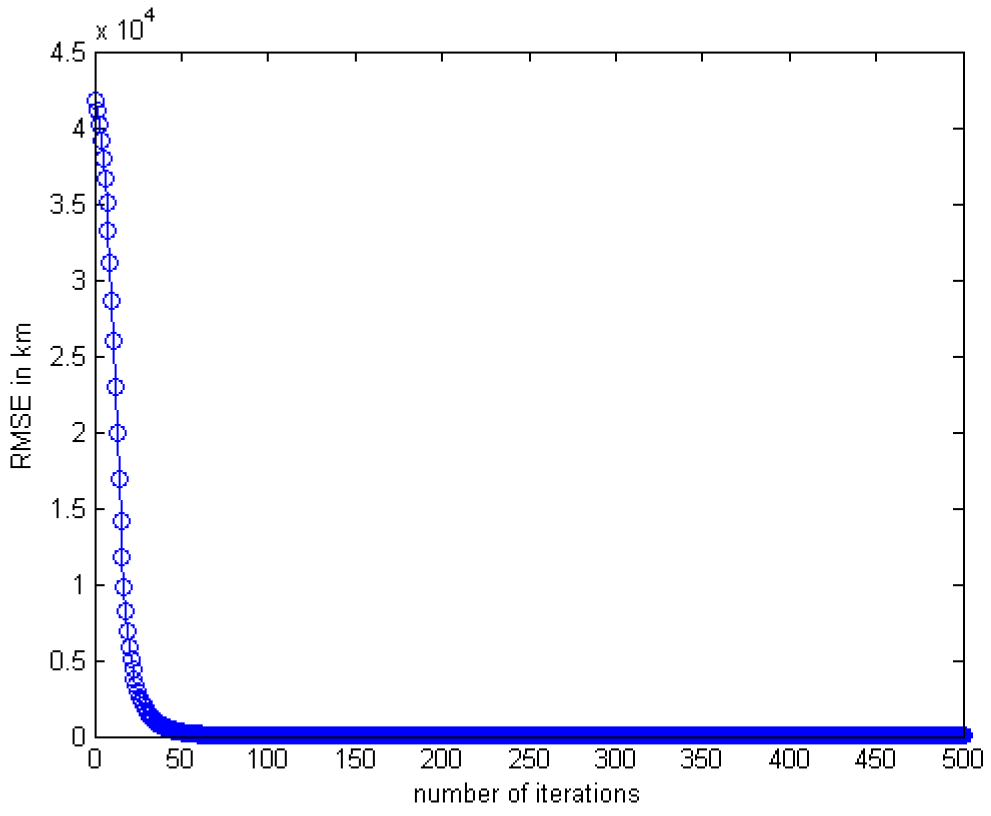

Figure 9: Root mean square error vs number of iterations

CHAPTER V: CONCLUSION

In this report, the performance of mobile station positioning in cellular networks using Time Difference of Arrival measurements is investigated. The standard Gauss Newton algorithm diverges for inaccurate initial values and the Steepest Descent method has a poor convergence behavior in the final iteration steps. To avoid these drawbacks, the Levenberg-Marquardt algorithm is proposed as a new approach in the positioning framework. Simulation results show that this method is suitable to estimate the mobile station location with high accuracy and low complexity. The results also show how the accuracy in the estimate changes with different parameters such as standard deviation of noise, number of base stations serving the mobile station and number of iterations involved.

REFERENCES

1. *Christian Mensing and Simon Plass, POSITIONING ALGORITHMS FOR CELLULAR NETWORKS USING TDOA, German Aerospace Center (DLR), Institute of Communications and Navigation.*

2. *Eneh Joy Nnenna, Orah Harris Onyekachi, Department of Electronics and Computer Engineering, Nnamdi Azikiwe University Awka. "Mobile Positioning Techniques in GSM Cellular Networks: A Comparative Performance Analysis".*

3. *http://www.ltu.se/cms_fs/1.51590!/nonlinear_least_squares.pdf*

4. *http://www.ananth.in/Notes_files/lmtut.pdf*

5. *D. Marquardt, "An Algorithm for Least Squares Estimation on Nonlinear Parameters," SIAM Journal on Applied Mathematics, vol. 11, pp. 431–441, 1963.*

6. *K. Levenberg, "A Method for the Solution of Certain Problems in Least Squares," Quarterly of Applied Mathematics, vol. 2, pp. 164–168, 1944.*

Appendix A

The following code gives the steps in the implementation of the algorithm. This code checks how the RMSE changes with different standard deviations of noise.

```
clear all
clc

fileID = fopen('position.txt','r')  %file containing the positions of the BS

A = fscanf(fileID,'%f %f',[2 inf]); %matrix used to save the positions of BS

fclose(fileID);
A= A';
p=size(A);                         % gives the number of base stations
j=1;
fileID = fopen('tdoa.txt','r');    % file containing the actual TDoA values
B = fscanf(fileID,'%f');           % matrix used to save TDoA values
fclose(fileID);

I= zeros(2,2);
for i=1:2
    for j=1:2
        if(i==j)
            I(i,j)=1;
        end
    end
end

var = 0;
a=10;
s = zeros(1,a);
v = zeros(1,a);
N = zeros(p(1,1)-1,1);

for l=1:a
var = var+0.009;                  % incrementing the variance of noise by
0.009
cov= zeros(p(1,1)-1,p(1,1)-1);    % covariance matrix of noise
for i=1:p(1,1)-1
    for j=1:p(1,1)-1
         if(i==j)
             cov(i,j)= var;
         end
     end
end
C=inv(cov);
s1 = zeros(1,500);
for m=1:1000
for j=1:(p(1,1)-1)
    g(j,1) = normrnd(0,sqrt(var));    % AWGN generated randomly
    N(j,1)=B(j,1)+g(j,1) ;
end
```

```
x=100; y=100;                        % initial position of the mobile station
M(1,1)=100;
M(2,1)=100;
count =1;
K = 2;
t=5;
for i=1:(p(1,1)-1)
        d(i,1)= -((sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2)-sqrt((x-A(1,1))^2+(y-
A(1,2))^2))-N(i,1)) ;
        J(i,1)= -(((x-A(i+1,1))/sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2))-((x-
A(1,1))/sqrt((x-A(1,1))^2+(y-A(1,2))^2)));  % jacobian matrix
        J(i,2)= -(((y-A(i+1,2))/sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2))-((y-
A(1,2))/sqrt((x-A(1,1))^2+(y-A(1,2))^2)));
end
e = d'*C*d;                       %cost function to be minimized
gradf = -J'*C*d;                  % gradient of cost function
dgradf = J'*C*J;                  % Hessian matrix
max = dgradf(1,1) ;
for i=1:2
    for j=1:2
        if((i==j)&&(dgradf(i,j)>max))
            max=dgradf(i,j);
        end
    end
end
max;
lambda =t*(max);               % initial value of update parameter
for count = 1:500
 h = (inv(dgradf+lambda*(dgradf*I))*gradf);
 M = M+h;
 x=M(1,1);
 y=M(2,1);
for i=1:(p(1,1)-1)
        d(i,1)= -((sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2)-sqrt((x-A(1,1))^2+(y-
A(1,2))^2))-N(i,1));
        J(i,1)= -(((x-A(i+1,1))/sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2))-((x-
A(1,1))/sqrt((x-A(1,1))^2+(y-A(1,2))^2)));
        J(i,2)= -(((y-A(i+1,2))/sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2))-((y-
A(1,2))/sqrt((x-A(1,1))^2+(y-A(1,2))^2)));
end
enew =d'*C*d;
r = (e-enew)/(2*h'*(lambda*h + gradf));      % error
e=enew;
if(r >= 0)
    gradf = -J'*C*d;
    dgradf = J'*C*J;
    q = 1-(2*r-1)^3;
    if(q > 1/3)
      lambda = lambda*q;
    else
        lambda = lambda*(1/3);
    end
    K=2;
else
   lambda = lambda*K;
   K = 2*K;
```

```
end
if(lambda<1e-7)
    lambda = 1e-7;
else if(lambda>1e7)
        lambda = 1e7;
    end
end
count;
lambda;
s1(1,count) = ((x-50)^2)+((y-300)^2);
end
s1(1,m) = ((x-50)^2)+((y-300)^2);
%df(:,m+100*(l-1))= d;
end
g=1;
for g=1:m
    s(1,l)=s(1,l)+s1(1,g);
end
s(1,l)=sqrt(s(1,l)/m);      % calculating the RMSE
v(1,l)= sqrt(var);
end
s;
plot(v,s,'-o');       %plot for RMSE vs standard deviation of noise
```

The following code gives how the RMSE changes with increase in number of base stations.

The implementation of the LM algorithm part is same above.

```
clear all
clc
fileID = fopen('position.txt','r');    %Text file containing the
positions of the BS
A = fscanf(fileID,'%f %f',[2 inf]);    % matrix used to save the
positions of BS
fclose(fileID);
A= A';
p=size(A);                             % gives the number of base
stations
j=1;
fileID = fopen('tdoa.txt','r');    % file containing the actual TDoA
values
B = fscanf(fileID,'%f');           % matrix used to save TDoA values
a=size(B);                         % gives the number of TDoA
measurements
fclose(fileID);
I= zeros(2,2);
```

```
for i=1:2
    for j=1:2
        if(i==j)
            I(i,j)=1;
        end
    end
end
var = 0;
var = var+0.09;

n=10;              %number of base stations
w=1000;
g1= zeros(n+3,w);
N = zeros(a(1,1),1);
for i=1:n+3
    for j=1:w
  g1(i,j) = normrnd(0,sqrt(var)); % AWGN generated randomly
    end
end
s = zeros(1,n+1);
bas = zeros(1,n+1);
for l=0:n
if(l>0)
A(p(1,1)+l,1)= (1000)*rand(1,1);   %positions of extra BS generated
randomly
A(p(1,1)+l,2)= (1000)*rand(1,1);
B(a(1,1)+l,1)= (sqrt((50-A(p(1,1)+l,1))^2+(300-A(p(1,1)+l,2))^2)-
sqrt((50-A(1,1))^2+(300-A(1,2))^2));
end
cov= zeros(p(1,1)-1+l,p(1,1)-1+l);  % covariance matrix of noise
for i=1:p(1,1)-1+l
    for j=1:p(1,1)-1+l
         if(i==j)
             cov(i,j)= var;
         end
     end
end
C=inv(cov);
s1 = zeros(1,w);
for m=1:w
for j=1:(p(1,1)-1+l)
    N(j,1)= B(j,1)+ g1(j,m);
end
x=100; y=100;    %initial value
M(1,1)=100;
M(2,1)=100;
count =1;
K = 2;
t=5;
for i=1:(p(1,1)-1+l)
        d(i,1)= -((sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2)-sqrt((x-
A(1,1))^2+(y-A(1,2))^2))-N(i,1));
        J(i,1)= -(((x-A(i+1,1))/sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2))-
((x-A(1,1))/sqrt((x-A(1,1))^2+(y-A(1,2))^2)));
        J(i,2)= -(((y-A(i+1,2))/sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2))-
((y-A(1,2))/sqrt((x-A(1,1))^2+(y-A(1,2))^2)));
```

```
end
e = d'*C*d;               %cost function to be minimized
gradf = -J'*C*d;          % gradient of cost function
dgradf = J'*C*J;          % Hessian matrix
max = dgradf(1,1) ;
for i=1:2
    for j=1:2
        if((i==j)&&(dgradf(i,j)>max))
            max=dgradf(i,j);
        end
    end
end
max;
lambda =t*(max);           % initial value of update parameter
for count = 1:500
 h = (inv(dgradf+lambda*(I))*gradf);
 M = M+h;
 x=M(1,1);
 y=M(2,1);
for i=1:(p(1,1)-1+l)
        d(i,1)= -((sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2)-sqrt((x-
A(1,1))^2+(y-A(1,2))^2))-N(i,1));
        J(i,1)= -(((x-A(i+1,1))/sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2))-
((x-A(1,1))/sqrt((x-A(1,1))^2+(y-A(1,2))^2)));
        J(i,2)= -(((y-A(i+1,2))/sqrt((x-A(i+1,1))^2+(y-A(i+1,2))^2))-
((y-A(1,2))/sqrt((x-A(1,1))^2+(y-A(1,2))^2)));
end
enew =d'*C*d;
r = (e-enew)/(2*h'*(lambda*h + gradf));    % error
e=enew;
if(r >= 0)
    gradf = -J'*C*d;
    dgradf = J'*C*J;
    q = 1-(2*r-1)^3;
    if(q > 1/3)
      lambda = lambda*q;
    else
        lambda = lambda*(1/3);
    end
    K=2;
else
   lambda = lambda*K;
   K = 2*K;
end
if(lambda<1e-10)
    lambda = 1e-10;
else if(lambda>1e10)
        lambda = 1e10;
    end
end
count;
lambda;
end
s1(1,m) = (((x-50)^2)+((y-300)^2));
for u=1:a(1,1)+l
df(u,m+(p(1,1)+l-3)*100)= d(u,1);
```

```
end
end
for g=1:m
    s(1,p(1,1)+l+1-3)=s(1,p(1,1)+l+1-3)+s1(1,g);
end
s(1,p(1,1)+l-3+1)=sqrt(s(1,p(1,1)+l+1-3)/m);   % calculating the RMSE
bas(1,p(1,1)+l-3+1)= p(1,1)+l;
end
 s;
 bas;
 lambda;
 s1;
 plot(bas,s,'-o');   %plot for RMSE vs no.of BS
```