Least-Squares Estimation of Nonlinear Parameters

Sasanka Uppuluri, 2019102036; Viswanadh Kandala, 2019112011

Abstract— **This paper is written as part of the term paper component of Signal Detection and Estimation Theory course. We attempt to implement the Least Square Estimation of Non Linear parameters using the Levenberg-Marquardt method, discuss a few applications and show the experimental results obtained from out implementation.**

I. INTRODUCTION

Many of the least-squares methods for nonlinear parameter estimation are based on expansion of Taylor series and corrections calculated by assuming a linear model and excluding second degree terms. These methods have problems of divergence of the successive iterates. Various modifications of the gradient descent method have also been introduced but they face the issue of slow convergence. As a result, a new method based on maximum neighbourhood method is developed which combined the Taylor series expansion and the gradient descent method known as the Levenberg-Marquardt method.

The term paper is organised as follows Section II contains the problem statement, Section III explains Gauss-Newton method and Gradient Descent method, Section IV explains the Levenberg-Marquardt method, Section V mentions the applications of leastsquares methods, Section VI describes the experiments we perform and we also show the results obtained for those experiments.We have also included a few proofs in the Appendix section.

II. PROBLEM STATEMENT

The model to be fitted to the data is defined as

$$E(y) = f(x_1, x_2, ..., x_m; b_1, b_2, ..., b_k) = f(\mathbf{x}, \mathbf{b}) \quad (1)$$

where $x_1, x_2, ..., x_m$ are independent variables, $b_1, b_2, ..., b_k$ are the k parameters to estimate, E(y) is the expected value of the dependent variable y.

To define the objective function, let the data points be denoted by

$$(Y_i, X_{1i}, X_{2i}, ..., X_{mi}), i = 1, 2, ..., n \quad (2)$$

The objective function to minimize to get the k parameters is given as

$$\Phi = \sum_{n=1}^{\infty} [Y_i - \hat{Y}_i]^2 = ||\mathbf{Y} - \hat{Y}||^2 (3)$$

where $\hat{Y}_i$ is the value of predicted y at the ith data point.

III. ALGORITHMS IN CURRENT USE

A. Gauss-Newton Algorithm

This method is similar to expanding f in a Taylor series as follows

$$\langle Y(\mathbf{X_i}, \mathbf{b} + \delta_t)\rangle = f(\mathbf{X_i}, \mathbf{b}) + \sum_{j=1}^{k} \left(\frac{\partial f_i}{\partial b_j}\right)(\delta_t)_j, \quad (4)$$

this can also be written in vectorized form as

$$\langle \mathbf{Y} \rangle = \mathbf{f_0} + \mathbf{P}\boldsymbol{\delta}_\mathbf{t}. \quad (5)$$

The vector δ_t is a small correction to b, which can be found by least-squares method of setting $\frac{\partial \langle \Phi \rangle}{\partial \delta_j} = 0$, for all j. This can be written as

$$A\boldsymbol{\delta}_\mathbf{t} = \mathbf{g}, \quad (6)$$

where

$$A^{[k\times k]} = P^T P, \quad (7)$$

$$P^{[n\times k]} = \left(\frac{\partial f_i}{\partial b_j}\right) \quad (8)$$

where $i = 1, 2, 3, ...n; j = 1, 2, ..., k$

$$g^{[k\times 1]} = \left(\sum_{i=1}^{n} (Y_i - f_i)\frac{\partial f_i}{\partial b_j}\right) = P^T(\mathbf{Y} - \mathbf{f}_0) = \quad (9)$$

where $j = 1, 2, ...k$

In practice it is found helpful to correct **b** by only a fraction of $\boldsymbol{\delta}_\mathbf{t}$; otherwise the extrapolation may be beyond the region where f can be adequately represented by and would cause divergence of the iterates. A few methods have been proposed to prevent this but even so, failure to converge is not uncommon.

B. Gradient Descent Method

In this method the correction in **b** is in the direction of the negative gradient of Φ. The step size is given as

$$\boldsymbol{\delta}_\mathbf{g} = -\left(\frac{\partial \Phi}{\partial b_1}, \frac{\partial \Phi}{\partial b_2}, \ldots, \frac{\partial \Phi}{\partial b_k}\right)^T \quad (10)$$

Various modified steepest-descent methods have been employed to compensate partially for the typically poor conditioning of the Φ surface which leads to very slow convergence of the gradient methods.Even here, controlling the step size carefully once the direction of the correction vector needs to be done. Even so, slow convergence is the rule rather than the exception.

IV. Levenberg-Marquardt algorithm

A. Theoritical basis of algorithm

The theoretical basis of the algorithm is contained in several theorems which follow. Let $\lambda \geq 0$ and let $\boldsymbol{\delta}_\mathbf{l}$ satisfy the equation

$$\left(A^{(r)} + \lambda^{(r)} I\right) \boldsymbol{\delta}_\mathbf{l}^{(r)} = \mathbf{g}^{(r)} \tag{11}$$

Theorem 1: $\boldsymbol{\delta}_\mathbf{l}$ minimizes Φ on the sphere whose radius $\|\boldsymbol{\delta}\|$ satisfies

$$\|\boldsymbol{\delta}|^2 = \|\boldsymbol{\delta}_\mathbf{l}\|^2 . \tag{12}$$

Theorem 2: $\|\boldsymbol{\delta}_\mathbf{l}(\lambda)\|$ decreases to zero monotonically as $\lambda \to \infty$.

Theorem 3: $\boldsymbol{\delta}_\mathbf{l}$ rotates from $\boldsymbol{\delta}_\mathbf{t}$ to $\boldsymbol{\delta}_\mathbf{g}$ monotonically as $\lambda \to \infty$.

B. Scale of measurement

The properties of the gradient methods are not scale invariant. It becomes necessary to scale the **b**-space in some convenient manner. We shall choose to scale the **b**-space in units of the standard deviations of the derivatives $\frac{\partial f_i}{\partial b_j}$ taken over the sample points i = 1, 2, .. , n.

This choice of scale transforms A into the matrix of simple correlation coefficients among the $\frac{\partial f_i}{\partial b_j}$ This also improves the numerical aspects of computing procedures. We define a scaled matrix A^* and a scaled vector g^*

$$A^* = a^*_{jj'} = \frac{a_{jj'}}{\sqrt{a_{jj}\sqrt{a_{j'j'}}}} \tag{13}$$

and

$$g^* = \left(g^*_j\right) = \left(\frac{g_j}{\sqrt{a_{jj}}}\right) \tag{14}$$

and solve for the Taylor series correction using

$$A^* \boldsymbol{\delta}^*_\mathbf{t} = g^* \tag{15}$$

Then

$$\delta_j = \frac{\delta^*_j}{\sqrt{a_{jj}}} \tag{16}$$

C. Construction of the algorithm

$\boldsymbol{\delta}_\mathbf{t}$ is almost $\pi/2$ away from $\boldsymbol{\delta}_\mathbf{g}$ in most problems because of the severe elongation of surface Φ, which makes the Gauss-Newton method slow. An idea to improve it is to interpolate between $\boldsymbol{\delta}_\mathbf{t}$ and $\boldsymbol{\delta}_\mathbf{g}$. In order to minimize Φ locally, equation for r^{th} iteration can be constructed as

$$\left(A^{*(r)} + \lambda^{(r)} I\right) \boldsymbol{\delta}_\mathbf{l}^{*(r)} = \mathbf{g}^{*(r)} \tag{17}$$

This equation is then solved for $\boldsymbol{\delta}_\mathbf{l}^{*(r)}$. We then use equation (16) to obtain $\boldsymbol{\delta}_\mathbf{l}^{(r)}$

The new trial vector is

$$\mathbf{b}^{(r+1)} = \mathbf{b}^{(r)} + \boldsymbol{\delta}_\mathbf{l}^{(r)} \tag{18}$$

Equation (18) will lead to a new sum of squares $\Phi^{(r+1)}$. It is essential that we select $\lambda^{(r)}$ such that

$$\Phi^{(r+1)} \leq \Phi^{(r)} \tag{19}$$

The idea is that in each iteration we need to minimize Φ in the maximum neighbourhood over which the linearized function will be an adequate representation of the nonlinear function. We need to choose small value of $\lambda^{(r)}$ when better convergence is required.

Larger values of $\lambda^{(r)}$ are only necessary when we need to satisfy equation (19). Larger value generally implies steepest-descent; e.g., rapid initial progress followed by progressively slower progress.

Keeping all these things in our mind, we construct our algorithm as follows

1) Let $v > 1$, and initialize $\lambda^{(0)} = 10^{-2}$.

2) Compute $\Phi\left(\lambda^{(r-1)}/v\right)$ and $\Phi\left(\lambda^{(r-1)}\right)$.
   i) If $\Phi\left(\lambda^{(r-1)}/v\right) \leq \Phi^{(r)}$, let $\lambda^{(r)} = \lambda^{(r-1)}/v$.
   ii) If $\Phi\left(\lambda^{(r-1)}/v\right) > \Phi^{(r)}$ and $\Phi\left(\lambda^{(r-1)}\right) \leq \Phi^{(r)}$, let $\lambda^{(r)} = \lambda^{(r-1)}$.
   iii) If $\Phi\left(\lambda^{(r-1)}/v\right) > \Phi^{(r)}$ and $\Phi\left(\lambda^{(r-1)}\right) > \Phi^{(r)}$, increase λ by successive multiplication by v until for some smallest w, $\Phi\left(\lambda^{(r-1)} v^w\right) \leq \Phi^{(r)}$

3) Set $\mathbf{b}^{(r)} \leftarrow \mathbf{b}^{(r-1)} + \delta_l$, where $\boldsymbol{\delta}_\mathbf{l}$ is obtained using equations (16) and (17) with $\lambda = \lambda^{(r)}$.

4) Set $r \leftarrow r + 1$.

5) Repeat steps 2 through 4.

V. Applications

In this section, we discuss a few applications of the algorithm.

A. Surface fitting

In computer vision, robotics, and geometric modelling it is common to derive an implicit representation of an object from raw shape data obtained with an imaging or touch sensing system. Such an implicit representation often takes the form of the zero set of a polynomial of even degree. In surface fitting, a family of quadratic polynomials with stably bounded zero sets is generally considered. The effectiveness of representation and computational efficiency of fitting are considered while choosing the family. The parameters to be determined are polynomial coefficients using a least-squares method. For example, minimizing the sum of squares of the distances from individual data points to the polynomial surface. Let's say, we need to fit a surface over n data points $p_1, p_2, ..., p_n$ in $R^2 or R^3$. We consider the family of polynomial curves/surfaces of form

$$f(X, a_0, a_1, ..., a_k) = 0 \tag{20}$$

where $X = (x, y) or (x, y, z)$ and $a_0, a_1, ..., a_k$ are the coefficients.

Let the distance from p_i to the surface mentioned in equation(20) be d_i, where i = 1, ..., n. Then, the fitting problem can then be modeled in a least squares fashion as follows

$$\min_{a_0,\ldots a_k} \sum_{i=0}^{k} d_i^2 \tag{21}$$

Here, d_i, for $1 \leq i \leq n$ cannot be determined until $a_0, \ldots, a_k$ are known, which according to equation (21) depend on the knowledge of d_i. To get out of this 'chicken-and-egg' situation, we approximate d_i as follows. Let q_i be the closest point to p_i on the surface to be determined. Obviously, $f(q_i; a_0, \ldots, a_k) = 0$. The best fitting f will place q_i close enough to p_i. So value of f can be approximated at q_i using Taylor's series at p_i, discarding all terms of the second order and above

$$f(q_i; a_0, .., a_k) \approx f(p_i, a_0, .., a_k) + \nabla f(p_i; a_0, .., a_k) \cdot (q_i - p_i) \tag{22}$$

As the left hand side of the above equation is zero(from equation(20)), we have

$$\nabla f(p_i; a_0, \ldots, a_k) \cdot (q_i - p_i) \approx -f(p_i; a_0, \ldots a_k) \tag{23}$$

As q_i and p_i are very close to each other, the vector $q_i - p_i$ and the gradient $\nabla f(p_i; a_0, \ldots a_k)$ are nearly parallel. The above equation yields an approximation

$$d_i = \|q_i - p_i\| \approx \frac{|f(p_i; a_0, \ldots a_k)|}{\|\nabla f(p_i; a_0, \ldots, a_k)\|}. \tag{24}$$

Substituting d_i in equation (21) , the fitting problem can be reformulated as

$$\min_{a_0,\ldots,a_k} \frac{f^2(p_i; a_0, \ldots, a_k)}{\|\nabla f(p_i; a_0, \ldots, a_k)\|^2} \tag{25}$$

B. Surface patch reconstruction

A robotic hand can reconstruct an unknown surface patch by touch. The goal is to track along three concurrent curves on the surface while collecting tactile data points (x_k, y_k, z_k), $1 \leq k$, in the meantime. Each curve represents the intersection of patch with a separate plane (referred to as the sampling plane) within which the tracking motion is presently constrained. Let p be the intersection point of the three curves. By fitting a parabola to the data points along each curve, its curvature at p is estimated. The surface's normal curvature at p in the tangent direction of this curve is the product of the (estimated) curvature with the cosine of the angle between the sampling plane and the normal plane containing the tangent direction. Thus three normal curvatures are obtained. From these curvature and the relative orientations of the corresponding tangent vectors at p, we solve for the principal curvatures k_1 and k_2 and the Darboux frame at the point p. Under this frame, the surface patch locally takes the form

$$z(x, y) = \frac{1}{2}\left(k_1 x^2 + k_2 y^2\right) + \sum_{3 \leq i+j \leq d} a_{ij} x^i y^j, \tag{26}$$

Here the terms of degree greater than two are added to describe a larger area of the surface. The coefficients of the polynomial gathered into a vector a, are determined in a least-squares sense as

$$\min_a f(a)$$

where

$$f(a) = \frac{1}{n} \sum_{k=1}^{n} \left(z(x_k, y_k) - z_k\right)^2. \tag{27}$$

VI. Comparisons between the methods

In this section, we take a function and try to do curve fitting. We compare the results obtained between Gauss-Newton, Gradient-Descent and Levenberg-Marqaurdt methods.

A. Experiments

We take our function to be estimated as a kind of inverted gaussian function whose pdf is

$$f(x) = A * (1 - \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}) \tag{28}$$

Our goal is to estimate the three parameters A, μ and σ so that we obtain the curve itself.

Baseline: The original curve has parameters A=10, $\mu = 0$ and $\sigma = 20$ We take the initial guess to be $[\hat{A}, \hat{\mu}, \hat{\sigma}] = [1, 1, 1]$. The learning rate η is taken to be 0.02 and we take 150 observations. We then estimate the parameters.

Experiment-1: The initial guess is changed to $[\hat{A}, \hat{\mu}, \hat{\sigma}] = [5, 10, 15]$.

Experiment-2: The number of observations is reduced to 50.

Experiment-3: A gaussian noise of $\mu = 0.5$ and $\sigma = 2$ is added to the observations.

B. Results

The code has been implemented in python and its resultant plots are shown. We plot
a) The original distribution and the estimated distribution in one plot
b) We also plot the Objective function($||\Phi||^2$) vs Number of iterations for convergence
We plot the graphs for all the 4 experiments and for all the 3 methods for easy comparison and analysis.
Both original and estimated curves are plotted in the same figure. They fit perfectly sometimes, so look closely to differentiate between them

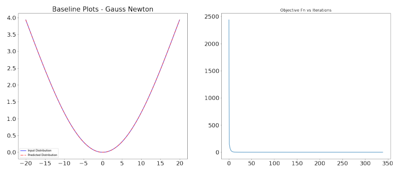

Fig. 1. Baseline Plot for Gauss-Newton

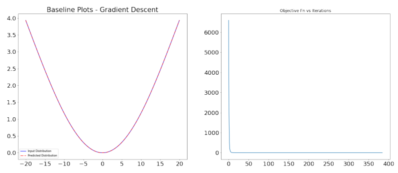

Fig. 2. Baseline Plot for Gradient Descent

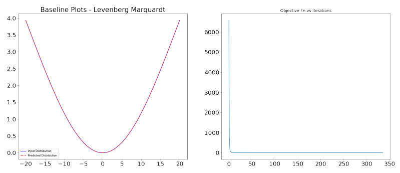

Fig. 3. Baseline Plot for Levenberg-Marquardt

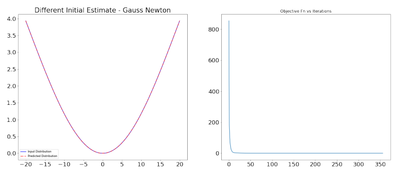

Fig. 4. Experiment-1 Plot for Gauss-Newton

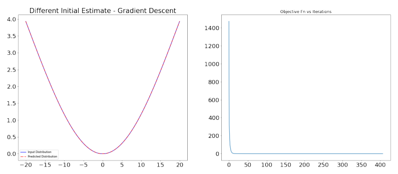

Fig. 5. Experiment-1 Plot for Gradient Descent

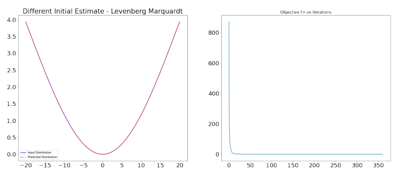

Fig. 6. Experiment-1 Plot for Levenberg-Marquardt

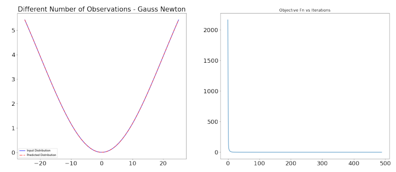

Fig. 7. Experiment-2 Plot for Gauss-Newton

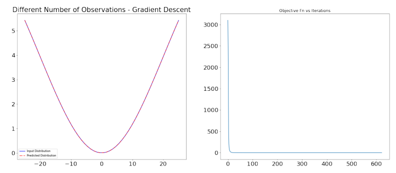

Fig. 8. Experiment-2 Plot for Gradient Descent

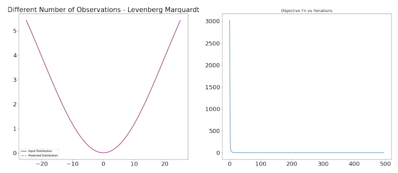

Fig. 9. Experiment-2 Plot for Levenberg-Marquardt

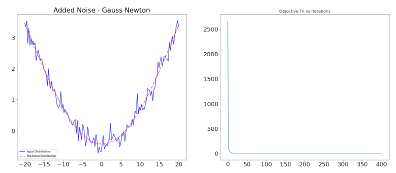

Fig. 10. Experiment-3 Plot for Gauss-Newton

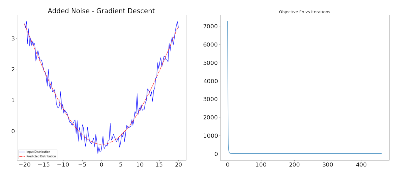

Fig. 11. Experiment-3 Plot for Gradient Descent

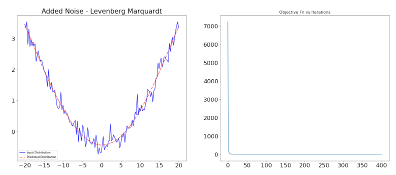

Fig. 12. Experiment-3 Plot for Levenberg-Marquardt

C. Analysis

The predictions made by Gauss-Newton and Levenberg-Marquardt methods are better as compared to Gradient Descent method as the gradient approximation involves linearly approximated second order terms.

Unlike Gradient Descent, Gauss-Newton and Levenberg-Marquardt methods have a normalisation proportional to the gradient magnitude. So very large gradients are normalised such that they do not cause oscillations and very small gradients are boosted for updates. The number of iterations required to converge is in the order: Gauss-Newton $\leq$ Levenberg-Marquardt $\leq$ Gradient Descent.

Gauss-Newton has a normalisation of magnitude of gradient, whereas Levenberg-Marquardt has a normalisation of λ + magnitude of gradient. For some cases (depending on whether λ is greater than or less than magnitude of gradient) it can wrongly increase or decrease gradients, which leads to more number of iterations to converge

When we make a different initialisation, we need more iterations to converge and this also involves some sudden drops in the cost function versus iteration maps. That is because we are approximating an exponential function and our initialization is far away. So initially, prediction values will come closer to ground truth rapidly.

Noisy samples have almost no effect on rate of convergence but it does have on the cost value and subsequently the predictions obtained, which are little off from the ground truth values. The following table shows the number of iterations for different algorithm and experiments.

Algorithm	BL	E1	E2	E3
Gauss-Newton	339	356	489	401
Gradient Descent	384	407	622	461
Levenberg-Marqaurdt	335	360	495	401

VII. Conclusion

The Levenberg-Marqaurdt algorithm combines the property of Gradient Descent methods ability to converge from an initial guess and the property of Gauss-Newton method to converge to the solution faster after the neighbourhood values are reached but is able to eliminate the shortcomings of both algorithms which is slow convergence and divergence from the solution.

Appendix A
Theorem-1 Proof

In order to find δ that minimizes

$$\Phi = \|\mathbf{Y} - \mathbf{f_0} - P\boldsymbol{\delta}\|^2$$

under the constraint

$$\|\delta\|^2 = \|\delta_l\|^2$$

Lagrange method is used where the final equation is given as

$$u(\boldsymbol{\delta}, \lambda) = \|\mathbf{Y} - \mathbf{f_0} - P\boldsymbol{\delta}\|^2 + \lambda\left(\|\boldsymbol{\delta}\|^2 - \|\boldsymbol{\delta}_l\|^2\right),$$

and λ is a Lagrange multiplier. After taking the derivatives, the equation that solves δ is given by

$$\left(P^T P + \lambda I\right)\boldsymbol{\delta} = P^T\left(\mathbf{Y} - \mathbf{f_0}\right) \tag{29}$$

and after premultiplying (29) by $\left(P^T P\right)^{-1}$

$$\boldsymbol{\delta} = \left(P^T P\right)^{-1} P^T\left(\mathbf{Y} - \mathbf{f_0}\right) - \left(P^T P\right)^{-1}\lambda\boldsymbol{\delta}$$

and substituting this in the Lagrange derivative will give the answer. Thus, it can be seen that equations (11) and (29) are identical. Thus, this stationary point is actually a minimum.

Appendix B
Theorem-2 Proof

As matrix A is symmetric and positive definite, it can be diagonalized as $S^T AS = D, S$ is orthogonal and D has positive diagonal elements. Putting this in (11)

$$\delta_0 = S(D + \lambda I)^{-1} S^T \mathbf{g}$$

Let $\mathbf{v} = S^T\mathbf{g}$ then,

$$\begin{aligned}\|\boldsymbol{\delta}_l(\lambda)\|^2 &= \mathbf{g}^T S(D + \lambda I)^{-1} S^T S(D + \lambda I)^{-1} S^T \mathbf{g}\\ &= \mathbf{v}^T\left[(D + \lambda I)^2\right]^{-1}\mathbf{v}\\ &= \sum_{j=1}^{k}\frac{v_j^2}{\left(D_j + \lambda\right)^2}\end{aligned}$$

which clearly is a decreasing function in λ.

Appendix C
Theorem-3 Proof

From the above theory δ_g = g. Let γ be the angle between δ_l and δ_g, then

$$\begin{aligned}\cos\gamma &= \frac{\delta^T\mathbf{g}}{(\|\delta\|)(\|\mathbf{g}\|)}\\ &= \frac{\mathbf{v}^T(D+\lambda I)^{-1}\mathbf{v}}{\left(\mathbf{v}^T[(D+\lambda I)^2]^{-1}\mathbf{v}\right)^{1/2}(\mathbf{g}^T\mathbf{g})^{1/2}}\\ &= \frac{\sum_{j=1}^{k}\frac{v_j^2}{(D_j+\lambda)}}{\left[\sum_{j=1}^{k}\frac{v_j^2}{(D_j+\lambda)^2}\right]^{1/2}(\mathbf{g}^T\mathbf{g})^{1/2}}\end{aligned}$$

After differentiating and simplifying

$$\frac{d\cos\gamma}{d\lambda} = \frac{\left[\sum_{j=1}^{k} v_j^2\Pi_{1j}\right]\left[\sum_{j=1}^{k} v_j^2\Pi_{3j}\right] - \left[\sum_{j=1}^{k} v_j^2\Pi_{2j}\right]^2}{\left[\sum_{j=1}^{k}\frac{v_j^2}{(D_j+\lambda)^2}\right]^{3/2}\left[\Pi_{j=1}^{k}\left(D_j + \lambda\right)^2\right]^2\left(\mathbf{g}^T\mathbf{g}\right)^{1/2}}$$

where $\Pi_{1j} = \Pi_{j'=1,j'\neq j}^{k}\left(D_{j'} + \lambda\right), \Pi_{2j} = \Pi_{j'=1,j'\neq j}^{k}\left(D_{j'} + \lambda\right)^2, \Pi_{3j} = \Pi_{j'=1,j'\neq j}^{k}\left(D_{j'} + \lambda\right)^3$. The sign of the derivative depends on the numerator. The numerator can be changed to

$$\left[\sum_{j=1}^{k} v_j^2\Pi_{1j}\right]\left[\sum_{j=1}^{k} v_j^2\Pi_{3j}\right] - \left[\sum_{j=1}^{k}\left(v_j\Pi_{1j}^{1/2}\right)\left(v_j\Pi_{3j}^{1/2}\right)\right]^2.$$

By Cauchy Schwarz Inequality, the above equation is positive. Thus, the derivative is always positive. As a result, γ is monotonically decreasing function of λ. For very large value of $\lambda(\infty)$, the equation (11) gives $\frac{g}{\lambda}$, the angle between δ_0 and g approaches zero. When $\lambda = 0$, the vectors δ_0 and g meet at some angle between 0 and $\pi/2$. This analysis also shows γ is a continuous monotone decreasing function of λ

ACKNOWLEDGEMENT

We would like to thank our course instructor Dr. Praful Mankar for his constant guidance and are grateful for this opportunity provided through this term paper which helped us to understand the concepts taught in class with better clarity.

REFERENCES

[1] D. W. Marquardt. An algorithm for least-squares estimation of non-linear parameters. Journal of the Society for Industrial and Applied Mathematics, 11(2):431–441, 1963.