Multidimensional and Multi-Parameter Fortran-Based Curve Fitting Tools

Daniel Tsegay and *Alem Mebrahtu

Department of Physics, College of Natural and Computational Sciences, Mekelle University, P.O. Box. 3044, Mekelle, Ethiopia (*alem.mebrahtu@mu.edu.et)

ABSTRACT

The Levenberg-Marquardt algorithm has become a popular method in nonlinear curve fitting works. In this paper, following the steps of Levenberg-Marquardt algorithm, we extend the framework of the algorithm to two and three dimensional real and complex functions. This work briefly describes the mathematics behind the algorithm, and also elaborates how to implement it using FORTRAN 95 programming language. The advantage of this algorithm, when it is extended to surfaces and complex functions, is that it makes researchers to have a better trust during fitting. It also improves the generalization and predictive performance of 2D and 3D real and complex functions.

Keywords: Levenberg-Marquardt algorithm, Nonlinear curve fitting and Least square fitting technique.

1. INTRODUCTION

Levenberg-Marquardt (LM) algorithm is an iterative technique (Levenberg, 1944; Kelley, 1999; Avriel, 2003; Marquardt, 1963; Bates & Watts, 1988; Box, et al., 1969; and Gill, et al., 1981) which helps in locating the discrepancy between a given model and the corresponding data. Such functions are usually expressible as sum of squares of nonlinear functions. The LM algorithm has become a standard technique for nonlinear least-square problems (Lourakis, 2005; Lampton, 1997**;** Arumugam, 2003; Coope, 1993; and Madsen, et al., 2004) and can be thought of as a combination of steepest descent and the Gauss-Newton methods. The paper is presented as follows: In section one, we present a brief introduction about the LM algorithm. In section two we discuss about the least square fitting technique. Section three elaborates Vanilla Gradient descent method. In the fourth section we present Newton's method. A more detailed discussion of LG algorithm is presented in section five. Section six discusses about the implementation of the LM algorithm. In the last section we present a brief summary of the paper.

1.1. Least-Square' Fitting Technique

Suppose we have a set of N experimental data points $\{x_i, y_i, \cdots, f_i, \sigma_i\}$, where $i = 1,\ldots,$ N for which we need to make a fitting. Here $X_i \equiv (x_i, y_i, \ldots)$ *are* the data coordinates, f_i is the data value and σ_i is the data error bar. Next we take a model which can estimate the values of f as a function of $X_i \equiv (x_i, y_i, \ldots)$ and a set of internal variable parameters $P \equiv (p_1, p_2, \ldots, p_M)$: $f(X, P)$.

Let us construct the chi-square function:

$$\chi^2(P) \equiv \sum_{i=1}^{N} \left(\frac{f_i - f(X_i, P)}{\sigma_i} \right)^2 = \sum_{i=1}^{N} r_i^2(P) \tag{1}$$

where $r_i(P_c) = \dfrac{f_i - f(X_i, P_c)}{\sigma_i}$ is called residue function. The goal of the least square method is to determine the parameters P of the regression function $f(X, P)$ so as to minimize the squared deviations between f_i and $f(X_i, P)$ for all data points: $i = 1 \cdots N$. If we assume that all measured values of f_i are *normally distributed* with standard deviations given by σ_i, then 'statistically-the-best' match would correspond to the *minimal* value of χ^2. Thus, the suitable model is essentially the one which gives the minimum value of the chi-square with respect to the parameters. That is why the method itself is called the 'least-square' technique. Of course, the error bars are determined not only by a statistical noise, but also by systematic inaccuracies, which are very difficult to estimate and are not normally distributed. However, to move on, we assume that they are some how accounted for by the values σ_i. Other approaches that are useful in determining the best-fit parameters for non-linear functions $f(X, P)$ by minimizing χ^2 iteratively include Newton's method and Gradient descent method.

1.2. Vanilla Gradient Descent Method

The Gradient descent method is simply an instinctive moving in the 'steepest descent' direction, which is apparently determined by the minus-gradient:

$$\beta_k \equiv -\frac{1}{2} \frac{\partial \chi^2(P_c)}{\partial p_k} = \sum_{i=1}^{N} r_i(P_c) \frac{\partial r_i(P_c)}{\partial p_k} = \sum_{i=1}^{N} \frac{f_i - f(X_{i,}, P_c)}{\sigma_1^2} \frac{\partial f}{\partial p_k}(X_i, P_c)$$

or

$$\begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{bmatrix} = \begin{bmatrix} \frac{\partial r_1(P_c)}{\partial p_1} & \frac{\partial r_2(P_c)}{\partial p_1} & \cdots & \frac{\partial r_M(P_c)}{\partial p_1} \\ \frac{\partial r_1(P_c)}{\partial p_2} & \frac{\partial r_2(P_c)}{\partial p_2} & \cdots & \frac{\partial r_M(P_c)}{\partial p_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_1(P_c)}{\partial p_M} & \frac{\partial r_2(P_c)}{\partial p_M} & \cdots & \frac{\partial r_M(P_c)}{\partial p_M} \end{bmatrix} \begin{bmatrix} r_1(P_c) \\ r_2(P_c) \\ \vdots \\ r_M(P_c) \end{bmatrix}. \qquad (2)$$

In compact form $\beta = -\frac{1}{2}\nabla\chi^2 = [J]^T r(P)$,

Where J is called Jacobian matrix of the residue $r_i(P_c)$ which is defined in Eqn. 1. The one-half coefficient is put to simplify the formulas. To improve the fit, we can shift the parameters $p_{kc} \rightarrow p_{kc} + \delta p_k$, where $\delta p_k = cons\tan t \times \beta_k$

$$\begin{bmatrix} \delta p_1 \\ \delta p_2 \\ \vdots \\ \delta p_M \end{bmatrix} = cons \tan t \times \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{bmatrix}. \qquad (3)$$

The steepest descent strategy is justified, when one is far from the minimum, but suffers from slow convergence in the plateau close to the minimum, especially in the multi-parameter space. Logically we would like large steps down the gradient at locations where the gradient (slope) is small (near the plateau) and small steps when the gradient is large not to rattle out of the minimum. Moreover, it has no information about the scale or the value of the constant and one can see that $\delta p_k = cons\tan t \times \beta_k$ has a problem with the unit dimensions.

1.3. Newton's Method

Newton's method is an algorithm used for finding roots of equations in one or more dimensions. Let us expand $\nabla\chi^2(P)$ using a Taylor's series around the current points, $P_c \equiv (p_{1c}, p_{2c}, \ldots p_{Mc})$, we get

$$\nabla\chi^2(P) = \nabla\chi^2(P_c) + [\delta P]^T \cdot \nabla^2\chi^2(P_c) + \text{higher order terms} \qquad (4)$$

$$\nabla\chi^2(P) = \left[\frac{\partial\chi^2(P_c)}{\partial p_1},\ \frac{\partial\chi^2(P_c)}{\partial p_2},\ \cdots,\ \frac{\partial\chi^2(P_c)}{\partial p_M}\right] + \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1M} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{M1} & \alpha_{M2} & \cdots & \alpha_{MM} \end{bmatrix} \begin{bmatrix} \delta p_1 \\ \delta p_2 \\ \vdots \\ \delta p_M \end{bmatrix}$$ + higher

order terms,

where $\delta p_k = p_k - p_{kc}$, $\delta P^T = [\delta p_1,\ \delta p_2,\ \cdots,\ \delta p_M]$ and

$$\alpha = \begin{bmatrix} \frac{\partial^2\chi^2}{\partial p_1\partial p_1} & \frac{\partial^2\chi^2}{\partial p_1\partial p_2} & \cdots & \frac{\partial^2\chi^2}{\partial p_1\partial p_M} \\ \frac{\partial^2\chi^2}{\partial p_2\partial p_1} & \frac{\partial^2\chi^2}{\partial p_2\partial p_2} & \cdots & \frac{\partial^2\chi^2}{\partial p_2\partial p_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2\chi^2}{\partial p_M\partial p_1} & \frac{\partial^2\chi^2}{\partial p_M\partial p_2} & \cdots & \frac{\partial^2\chi^2}{\partial p_M\partial p_M} \end{bmatrix}.$$

Note that $\frac{\partial\chi^2(P_c)}{\partial p_k}$ is the gradient vector of χ^2 with respect to p_k evaluated at P_c and

$\alpha_{kl} \equiv \frac{1}{2}\frac{\partial^2\chi^2(P_c)}{\partial p_x\partial p_l}$ is the second order gradient vector of χ^2 (is called Hessian matrix) evaluated at P_c .

Near the current points P_c, we can approximate the value of $\chi^2(P)$ up to the second order, as

$\nabla\chi^2(P) = \nabla\chi^2(P_c) + [\delta P]^T \cdot \nabla^2\chi^2(P_c)$.

Assuming the chi-square function is quadratic around P_c and solving for the minimum values of the parameters P by setting $\nabla\chi^2(P) = 0$, we get the update rule (the next iteration point) for Newton's methods:

$$[\alpha][\delta P]^T = -\nabla\chi^2(P_c) \Leftrightarrow \sum_{l=1}^{M}\alpha_{kl}\delta p_l = \beta_k \tag{5}$$

$$[\delta P]^T = -[\alpha]^{-1}\nabla\chi^2(P_c) \Rightarrow P = P_c - [\alpha]^{-1}\nabla\chi^2(P_c). \tag{6}$$

The chi function (which is quadratic) to be minimized has almost parabolic shape. The Hessian matrix, which is proportional to the curvature of χ^2 , is given by

$$\alpha_{kl} \equiv \frac{1}{2}\frac{\partial^2 \chi^2(P_c)}{\partial p_k \partial p_l} = \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\left\{\frac{\partial f(X_i,P_c)}{\partial p_k}\frac{\partial f(X_i,P_c)}{\partial p_1} - \left[f_i - f(X_i,P_c)\right]\frac{\partial^2 f(X_i,P_c)}{\partial p_k \partial p_l}\right\}$$

(7)

(the one-half here is also added for the sake of simplicity). The components α_{kl} of the Hessian matrix in Eqn. (7) depends both on the first derivative, $\frac{\partial f(X_i,P_c)}{\partial p_k}$, and second derivative, $\frac{\partial^2 f(X_i,P_c)}{\partial p_k \partial p_l}$, of the basic function with respect to their parameters. The Second derivative can be ignored when it is zero, or small enough to be negligible when compared to the term involving the first derivative. In practice, this is quite often small enough to neglect. If one looks at Eqn. (7) carefully, the second derivative is multiplied by $\left[f_i - f(X_i,P_c)\right]$. For the successful model, this term should just be the random measurement error of each point. This error can have either sign, and should in general be uncorrelated with the model. Therefore, the second derivative terms tend to cancel out when summed over time i . Inclusion of second derivative term can in fact be destabilizing if the model fits badly or is contaminated by outlier points that are unlikely to be offset by compensating points of opposite sign. So, instead of Eqn. (7) we shall define the α-matrix simply as: $\alpha_{kl} \equiv \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\frac{\partial f(X_i,P_c)}{\partial p_k}\frac{\partial f(X_i,P_c)}{\partial p_l}$ which is equivalent to

$$\alpha = [J]^T J = \begin{bmatrix} \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\frac{\partial f(X_i,P_c)}{\partial p_1}\frac{\partial f(X_i,P_c)}{\partial p_1} & \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\frac{\partial f(X_i,P_c)}{\partial p_1}\frac{\partial f(X_i,P_c)}{\partial p_2} & \cdots & \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\frac{\partial f(X_i,P_c)}{\partial p_1}\frac{\partial f(X_i,P_c)}{\partial p_M} \\ \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\frac{\partial f(X_i,P_c)}{\partial p_2}\frac{\partial f(X_i,P_c)}{\partial p_1} & \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\frac{\partial f(X_i,P_c)}{\partial p_2}\frac{\partial f(X_i,P_c)}{\partial p_2} & \cdots & \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\frac{\partial f(X_i,P_c)}{\partial p_2}\frac{\partial f(X_i,P_c)}{\partial p_M} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\frac{\partial f(X_i,P_c)}{\partial p_M}\frac{\partial f(X_i,P_c)}{\partial p_1} & \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\frac{\partial f(X_i,P_c)}{\partial p_M}\frac{\partial f(X_i,P_c)}{\partial p_2} & \cdots & \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\frac{\partial f(X_i,P_c)}{\partial p_M}\frac{\partial f(X_i,P_c)}{\partial p_M} \end{bmatrix}.$$

(8)

After computing, numerically or analytically, the gradient and Hessian matrices for the current set of parameters, one can immediately move to the minimum by shifting the parameters $p_k \rightarrow p_k + \delta p_k$, where the displacement vector δp_k is determined from the linear system derived in Eqn. (5), i.e.,

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1M} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{M1} & \alpha_{M2} & \cdots & \alpha_{MM} \end{bmatrix} \begin{bmatrix} \delta p_1 \\ \delta p_2 \\ \vdots \\ \delta p_M \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{bmatrix} \Leftrightarrow \begin{bmatrix} \delta p_1 \\ \delta p_2 \\ \vdots \\ \delta p_M \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1M} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{M1} & \alpha_{M2} & \cdots & \alpha_{MM} \end{bmatrix}^{-1} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{bmatrix}.$$

(9)

One of the problems associated with Newton's method (Levenberg, 1944; Kelley, 1999; Madsen, et al., 2004; and Lawson & R.J. Hanson, 1974) is its divergence after successive iterations. At the instant when $\chi^2(P_c + \delta P)$ diverges we would like to retreat to its previous value $\chi^2(P_c)$ and then decrease the steps, δP and try again.

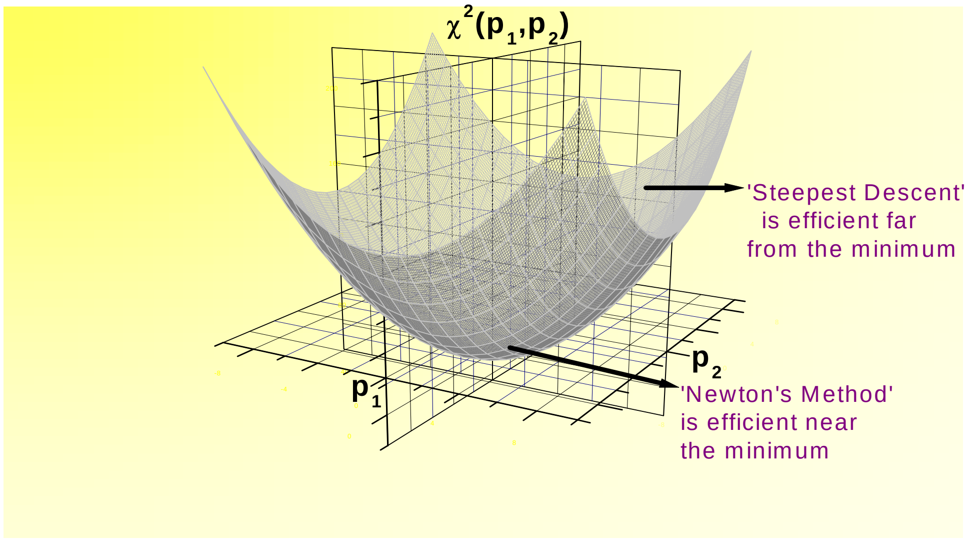
$\chi^2(p_1,p_2)$
'Steepest Descent' is efficient far from the minimum
p_2
p_1
'Newton's Method' is efficient near the minimum

Figure 1. Graph of the chi function: The chi-square (χ^2) function versus two arbitrary experimental parameters P_1 and P_2.

1.4. The Levenberg-Marquardt Algorithm

In order for the chi-square function to converge to a minimum rapidly, one needs a large step in the direction along with the low curvature (near the minimum) and a small step in the direction with the high curvature (i.e. a steep incline). The gradient descent and Gauss-Newton iterations provide additional advantages. The LM algorithm is based on the self-adjustable balance between the two minimizing strategies: the Vanilla Gradient Descent and the Inverse Hessian methods.

Coming back to the steepest descent technique χ^2 is dimensionless but β_k has the same dimension as $\frac{1}{p_k}$, as indicated in Eqn. (3). The constant of proportionality between β_k and δp_k must therefore have the dimension of p_k^2. For instance, if the parameter p_k is measured in kg, then β_k has obviously the units of kg^{-1} so the constant must have a dimension of kg^{-2}. Therefore the unit cannot be the same for all parameters since they are generally measured in different units (p_1 in Seconds, p_2 in Meter… p_M in Ampere). Marquadt surmised that the components of the Hessian matrix must hold at least some information about the order-of–magnitude scale and dimension. Among the components of α-matrix the reciprocal of the diagonal elements α_{kk}^{-1} have these dimensions. Hence he suggested that this must set the scale of the constant. To avoid the scale becoming too large, it is divided by a dimensionless positive damping term, λ (being positive ensures that δp_k is a descent direction). Eqn. (3) is then replaced by

$$\begin{bmatrix} \delta p_1 \\ \delta p_2 \\ \vdots \\ \delta p_M \end{bmatrix} = \frac{1}{\lambda}\begin{bmatrix} \alpha_{11}^{-1} & 0 & \cdots & 0 \\ 0 & \alpha_{22}^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_{MM}^{-1} \end{bmatrix}\begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{bmatrix}. \qquad (10)$$

In more compact form, $\delta p_k = \frac{1}{\lambda \alpha_{kk}} \beta_k$.

In order to combine Eqns. (9) and (10), Marquardt defined a diagonally-enhanced new α'-matrix: $\alpha'_{kl} = \alpha_{kl}(1 + \delta_{kl}\lambda)$, where the value of the Kronicker delta function is given by $\delta_{kl} = \begin{cases} 0 & for \quad k \neq l \\ 1 & for \quad k = l \end{cases}$ such that

$$\begin{bmatrix} \alpha'_{11} & \alpha'_{12} & \cdots & \alpha'_{1M} \\ \alpha'_{21} & \alpha'_{22} & \cdots & \alpha'_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha'_{M1} & \alpha'_{M2} & \cdots & \alpha'_{MM} \end{bmatrix} = \begin{bmatrix} \alpha_{11}(1+\lambda) & \alpha_{12} & \cdots & \alpha_{1M} \\ \alpha_{21} & \alpha_{22}(1+\lambda) & \cdots & \alpha_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{M1} & \alpha_{M2} & \cdots & \alpha_{MM}(1+\lambda) \end{bmatrix}$$

(11)

where λ is a dimensionless constant, and α_{kl} is replaced with α'_{kl} in Eqn. (5) which yields

$$\sum_{l=1}^{M} \alpha'_{kl}\delta p_1 = \beta_k \text{ or}$$

$$\begin{bmatrix} \delta p_1 \\ \delta p_2 \\ \vdots \\ \delta p_M \end{bmatrix} = \begin{bmatrix} \alpha_{11}(1+\lambda) & \alpha_{12} & \cdots & \alpha_{1M} \\ \alpha_{21} & \alpha_{22}(1+\lambda) & \cdots & \alpha_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{M1} & \alpha_{M2} & \cdots & \alpha_{MM}(1+\lambda) \end{bmatrix}^{-1} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{bmatrix}. \qquad (12)$$

For very small value of λ, the displacement vector δp_k, obtained from Eqn. (12) is close to the one, obtained by the pure Inverse Hessian technique, Eqn. (9), which is a good step in the final stages of the iteration, near the minima. If $\chi^2 = 0$ (or very small), then we can get (almost) quadratic final convergence. However, if λ is very large, then the matrix α'_{kl} is forced in to being diagonally dominant, so Eqn. (12) goes over to be identical to Eqn. (10), this is good if the current iterate is far from the solution. It means that, by increasing the parameter λ we approach the 'steepest descent' limit (i.e. a short step in the steepest descent direction). Thus, the damping term λ influences both the direction and the size of the step, and this leads us to make a method without a specific line search. To reduce the computational errors (especially near the minimum point), it is recommended to find the derivatives of the model function $\chi^2(X,P)$ *analytically*. Let's first prepare the LM algorithm, with flow chart. The minimization process is iterative. One starts with a reasonably small value of λ. At every successful iteration: $(\chi^2_{mew} < \chi^2_{cur})$, it is *reduced* by a factor of 10, moving towards the 'inverse Hessian' regime. Otherwise it retreats to the 'steepest descent' regime by being *increased* by a factor of 10. The stop criteria are necessary to avoid an endless iteration cycle. When one or more combination of the following stopping criteria are satisfied, then the fitting process stops:

i. When the total number of iterations entered by the user attains.

ii. When the minimum value of $\chi^2(P_c)$ to exit iteration attains.

iii. When the absolute shift of the chi square, $|\chi^2(P_c+\delta P) - \chi^2(P_c)|$ below some a certain threshold or decreases by negligible amount. The program can also be set to 'PAUSE' when $\chi^2(P_c+\delta P)$ a start to diverge then continues after press enter key.

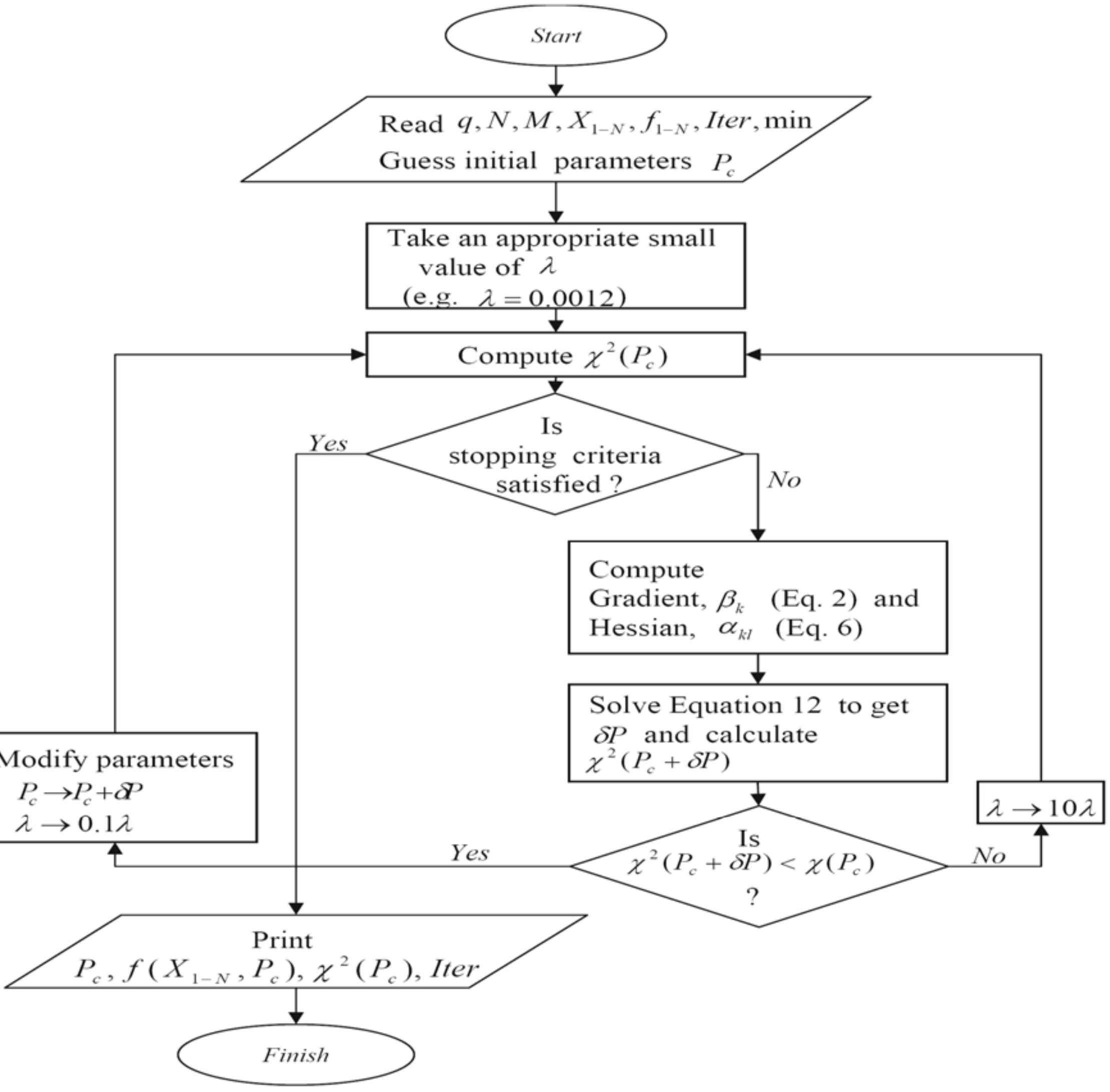
Start
Read $q, N, M, X_{1-N}, f_{1-N}, Iter$, min
Guess initial parameters P_c
Take an appropriate small value of λ (e.g. $\lambda = 0.0012$)
Compute $\chi^2(P_c)$
Is stopping criteria satisfied ?
Yes
No
Compute Gradient, β_k (Eq. 2) and Hessian, α_{kl} (Eq. 6)
Solve Equation 12 to get δP and calculate $\chi^2(P_c + \delta P)$
Is $\chi^2(P_c + \delta P) < \chi(P_c)$?
Yes
No
$\lambda \rightarrow 10\lambda$
Modify parameters $P_c \rightarrow P_c + \delta P$ $\lambda \rightarrow 0.1\lambda$
Print $P_c, f(X_{1-N}, P_c), \chi^2(P_c), Iter$
Finish

Figure 2. The LM Algorithm with a flow chart.

The update rule is used as follows. If the error goes down following an update, it implies that our quadratic assumption on χ^2 is working and reduce λ (usually by a factor of 10) to reduce the influence of gradient descent. On the other hand, if the error goes up, we would like to follow the gradient more and so λ is increased by the same factor. If the initial guess is good but χ^2 does not fall down to the required minimum value, we have to change the initial value of λ slightly.

2. IMPLEMENTATION OF THE LM ALGORITHM

In this paper Gauss's elimination and Gauss's Jordan matrix inversion methods are used to determine the shift parameters. Among the several tests made on real and complex non linear functions, only three examples are illustrated to see how much this method is effective and faster than the other methods.

2.1. Test on real three dimensional wave function

The first test is applied to two dimensional data coordinate (x_i, y_i) and data value f_i where $i = 1 - 210$, e.g., at $i = 7$ $(x_7 = -7, y_7 = -1, f_7 = 6.452)$.

Table 1. Experimental data for irregularly shaped surface.

x_i	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7
y_i														
-7	-1.029	6.743	13.3	15.99	14.91	12.72	19.56	-33.59	-14.18	-3.027	2.91	2.546	-2.389	-8.428
-6	-7.211	0.544	6.464	7.324	2.644	-6.841	-20.33	4.615	5.203	10.67	13.85	11.29	4.334	-3.384
-5	-9.661	-5.837	-3.782	-5.371	-10.97	-22.67	-49.52	40.49	21.73	18.62	17.54	13.99	8.64	3.572
-4	-7.136	-8.94	-11.96	-15.41	-18.64	-26.28	-52.52	54.98	26.54	16.56	12.09	9.181	8.176	8.796
-3	-1.112	-6.983	-13.68	-17.6	-16.22	-15.65	-27.89	40.53	16.97	5.508	0.567	-0.605	3.058	9.637
-2	4.853	-0.665	-7.916	-11.2	-4.794	3.837	10.87	5.224	-2.158	-8.837	-10.48	-10.26	-4.33	5.997
-1	6.452	7.76	2.596	-0.96	10.17	22.71	41.83	-30.86	-21.57	-19.46	-13.81	-14.99	-11.16	0.942
0	1.905	14.72	12.53	6.545	21.22	31.66	47.46	-47.46	-31.66	-21.22	-6.545	-12.53	-14.72	-1.905
1	-0.942	11.16	14.99	13.81	19.46	21.57	30.86	-41.83	-22.71	-10.17	0.96	-2.596	-7.76	-6.452
2	-5.997	4.33	10.26	10.48	8.837	2.158	-5.224	-10.87	-3.837	4.794	11.2	7.916	0.665	-4.853
3	-9.637	-3.058	0.605	-0.567	-5.508	-16.97	-40.53	27.89	15.65	16.22	17.6	13.68	6.983	1.112
4	-8.796	-8.176	-9.181	-12.09	-16.56	-26.54	-54.98	52.52	26.28	18.64	15.41	11.96	8.94	7.136
5	-3.572	-8.64	-13.99	-17.54	-18.62	-21.73	-40.49	49.52	22.67	10.97	5.371	3.782	5.837	9.661
6	3.384	-4.334	-11.29	-13.85	-10.67	-5.203	-4.615	20.33	6.841	-2.644	-7.324	-6.464	-0.544	7.211
7	8.428	2.389	-2.546	-2.91	3.027	14.18	33.59	-19.56	-12.72	-14.91	-15.99	-13.3	-6.743	1.029

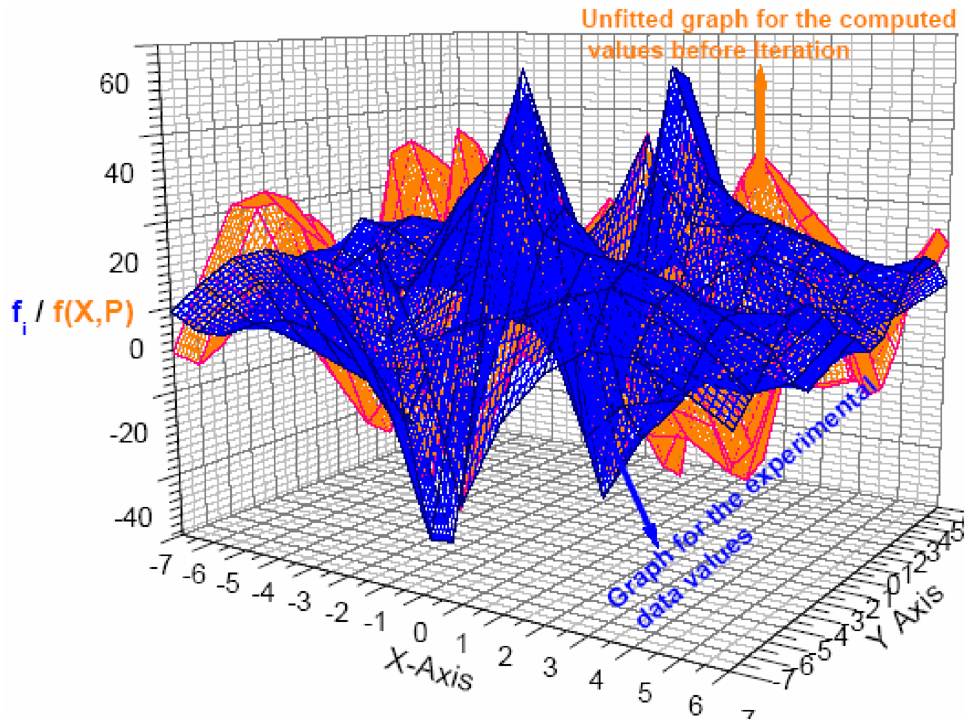

Figure 3 (a). Graphs of experimental values f_i (blue) and numerical or computed values $f(X, P_c)$ (yellow) before iteration.

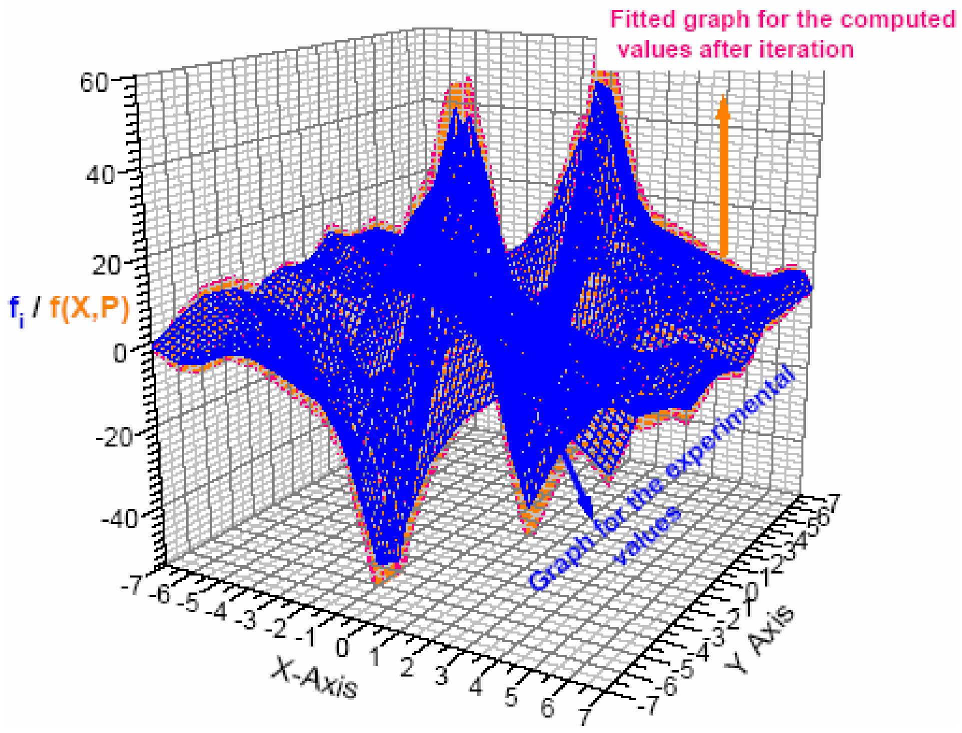

Figure 3 (b). Graphs of experimental values f_i (blue) and numerical or computed values $f(X, P_c)$ (yellow) after iteration

From the above results (Table.1), one can easily see that the data (surface) follows the wave function having the form

$$f(X,P) = f(x,y,p_1,p_2,p_3) = p_1\frac{\sin(p_3x)}{\left(y^2+1\right)} - p_2\frac{\cos(y)}{x} - p_3\sin(y)\cos(x).$$

We have then made a fitting, using the LM approach, in order to find the values of the parameters $\left(p_1, p_{2,}, p_3\right)$ that best fit $f(X_i,P)$ with f_i (see Fig. 3 (a) and (b)).

In this case the dimension is $q=2$ and the numbers of parameters are $M=3$. After initializing $\left(p_1, p_{2,}, p_3\right)$ the values found from the iteration are $\chi^2 = 0.0$, $p_1 = 7.0,$ $p_2 = 11.0$ and $p_3 = 54.0$.

The function now have the from $f(x,y) = 7\frac{\sin(11x)}{\left(y^2+1\right)} - 54\frac{\cos(y)}{x} - 11\sin(y)\cos(x)$.

As one can see from the above results, the LM model is highly useful when it is implemented to complicated-shaped surfaces. What is also important here is here that selecting an appropriate type of function (such as sine, power, decay, etc functions) and lambda. The shift parameters are not that much changed by normalized random errors only minimum of chi-function increases. Hence, based on the above two figures (Figs. 3 (a) and (b)), one can conclude that new equations/relations and modifications to the already existing formulas can be obtained from experimental data having disturbed/complicated surfaces.

2.2. Test made on complex two dimensional function

In ellipsometery the complex ratio $\rho = \frac{r_p}{r_s} = \tan\Psi e^{j\Delta}$ is measured, commonly expressed in terms of the two real parameters Ψ and Δ i.e. $\rho = \tan\Psi e^{j\Delta}$. The inversion of this formula to get suitable value of real and imaginary part of the refractive index is some what difficult to do analytically, and even numerically inversion of complex functions using LM algorithm is not yet well developed.

Let us consider an oblique reflection and transmission of optical plane wave at the planner interface between two semi-infinite homogeneous optically isotropic media air and glass with complex index of refraction $n = n_r + jk$. The ratio of the complex reflection coefficient, ρ, is related to the angle of incident by

$$\rho = \tan\Psi e^{j\Delta} = \frac{\left[\sin^2\theta - \cos\theta_0\sqrt{\left(n_r + jk\right)^2 - \sin^2\theta}\right]^2}{\sin^2\theta - \left(n_r + jk\right)^2\cos\theta}.$$

The algorithm has been tested on an actual data taken in a *PSA*-ellipsometry on acrylic glass sample for a wave length of light $450nm$. After successive iterations the following results has been recorded.

Table 2. Experimental data and computed values of ρ and n.

i^{th} *measurement*	Data coordinate $(\theta_i/\deg)$	Experimental data values $f_i = \rho_i$	Values found from the successive iterations: Computed values $f(X_i,P_c) = \rho(\theta_i,n)$	Actual error $\rho_i - \rho(\theta_i,n)$
1	52+j0	-0.11726-j0.00134	-1.1721756E-01-j1.3177083E-03	-4.2445958E-05-j2.2291672E-05
2	54+j0	-0.06301-j0.00135	-6.2998131E-02-j1.3587392E-03	-1.1868775E-05+j8.7391818E-06
3	55+j0	-0.03577-j0.00135	-3.5782781E-02-j1.3766416E-03	1.2781471E-05+j2.6641530E-05
4	56+j0	-0.00847-j0.00143	-8.5111084E-03-j1.3926749E-03	4.1108578E-05-j3.7325081E-05
N=4 $q = 1,\ m = 1,$ Initialization $p_{1c} = n = 3 + j0.3$			$\mathrm{CHI} = 1.1195837\mathrm{E}-09 - j7.0279005\mathrm{E}-10$ $\mathrm{ABS(CHI)} = 1.32188\mathrm{E}-9$ $n = 1.50009620 + j2.90234271\mathrm{E}-3$	

The real and imaginary part of the refractive index of the glass found from the iteration is $n_r = 1.5000962$ and $k = 0.00290234271$ respectively. The fitted values of the reflection coefficient have up to 5 decimal precision (one can also get high precision by selecting appropriate lambda till the errors arise only form the experiment imperfection and machine error. The interesting thing doing with complex function is, we only solve the derivative of $\rho(\theta, n)$ with respect to n i.e. $\frac{d\rho(\theta,n)}{dn}$ to find n_r and k (not $\frac{\rho(\theta,n)}{dn_r}$ and $\frac{\rho(\theta,n)}{dk}$). During interpolation and extrapolation, unlike the Aitkens and Lagrange interpolations, graphs interpolated using LM model follow the right path (with little regression).

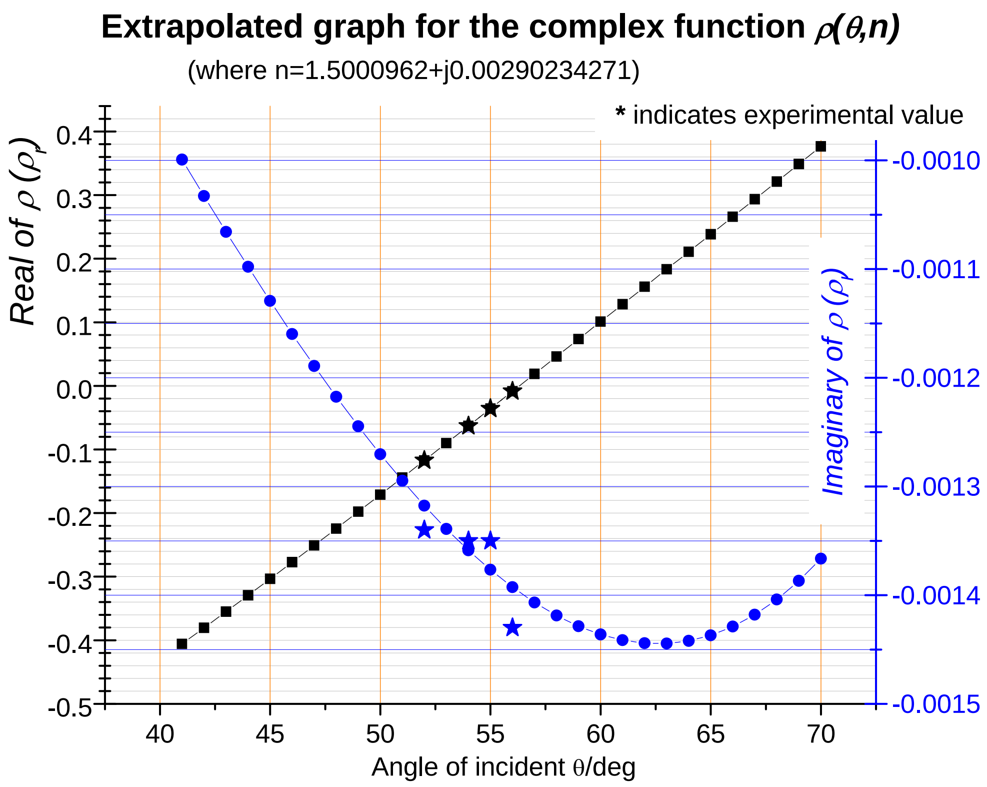

Figure 4. Extrapolated graph for the complex function $\rho(\theta, n)$ with * and ▪ representing experimental and numerical values respectively.

2.3. Test on complex two dimensional power function

The third test was made on complex three dimensional power functions (their derivatives are logarithmic functions). Consider the following experimental data:

Table 3. Experimental data on 2D power functions.

i	x_i	y_i	f_i	
1	6+j2	- 1- j 6	151.1271	j 41.47818
2	5+j8	29+j 0	-318.893	j 710.7169
3	-3+-j0.5	-7+j 1	34.97808	j 96.72046
4	-4+j 2	0+j 5	61.8854	- j 24.1816
5	-5+j 5	-9.9- j 3	260.2891	j 413.5324
6	-6- j 1	-4+j 1	14.13067	j 120.9102
N =6				

The data is fitted with the function $f(x, y, p_1, p_2, p_3, p_4) = x^{p_1} + y^{p_2} + p_3xy + p_4$. For this case the value of $q = 2$ and $M = 4$. During Initialization of the parameters with $p_1 = 2 - 0.5j$, $p_2 = 2 + 0.5j$, $p_3 = 2 + 0j0.5$ and $p_4 = 2 + j0.5$ (equivalent to $P_c = 2,2,2,-0.5,0.5,0.5,0.5$), the appropriate value of λ used near 0.001 is 0.0012.

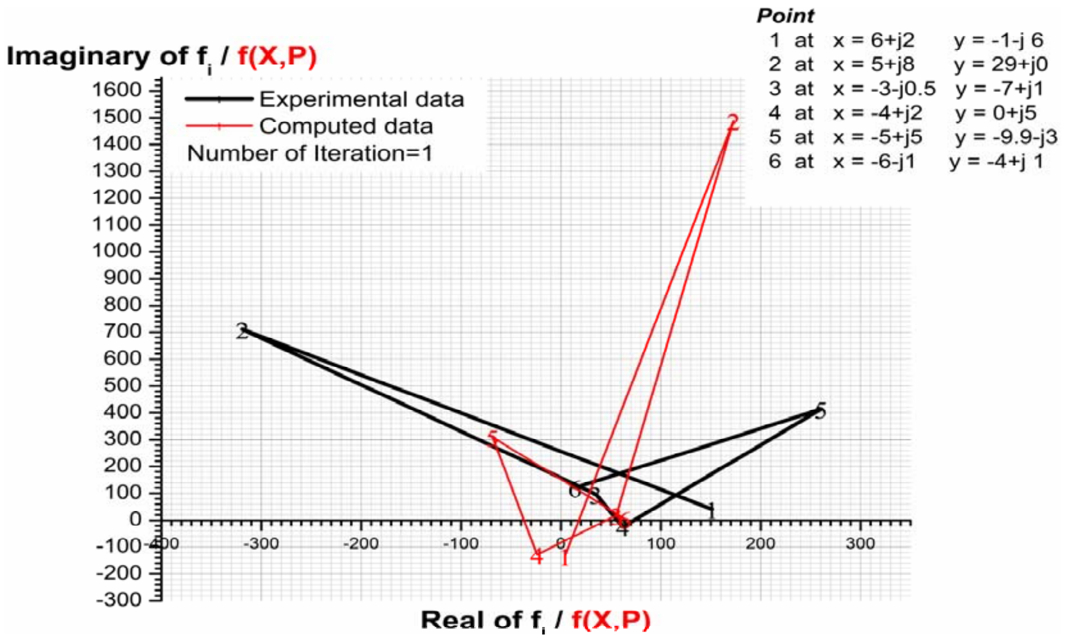

Figure 5 (a). Graphs of the experimental and numerical data at different number of iterations.

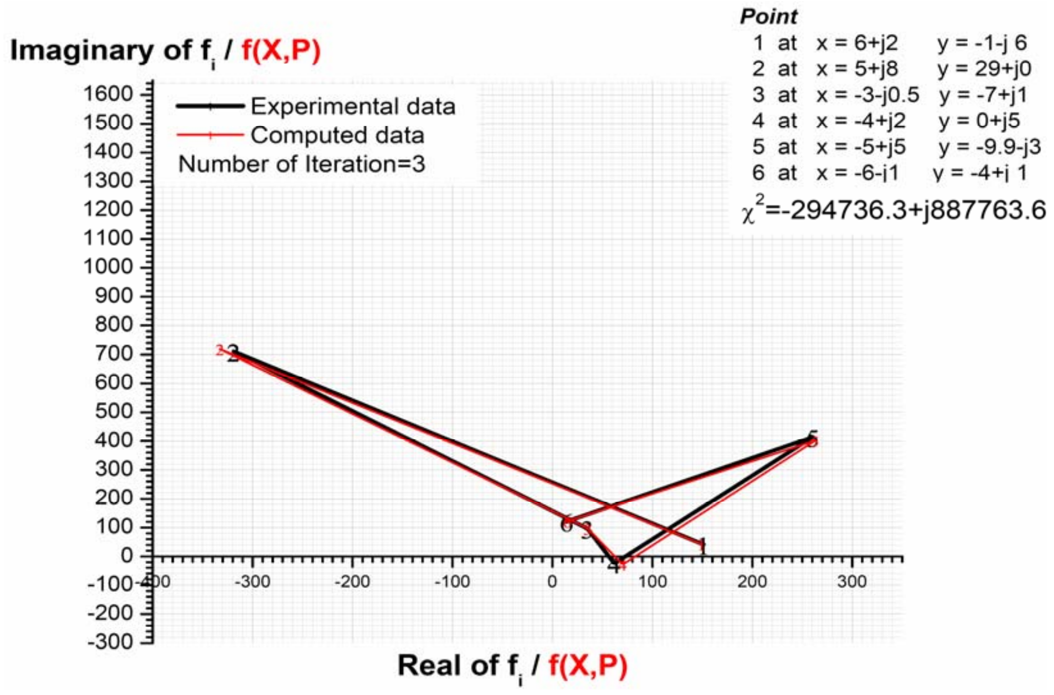

Figure 5 (b). Graphs of the experimental and numerical data at different number of iterations.

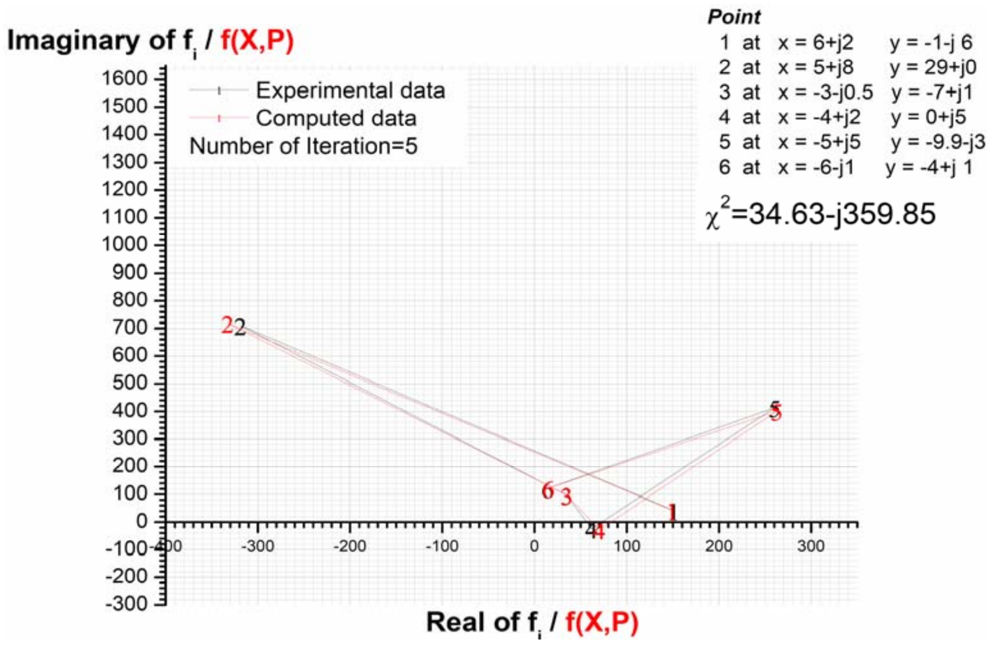

Figure 5 (c). Graphs of the experimental and numerical data at different number of iterations.

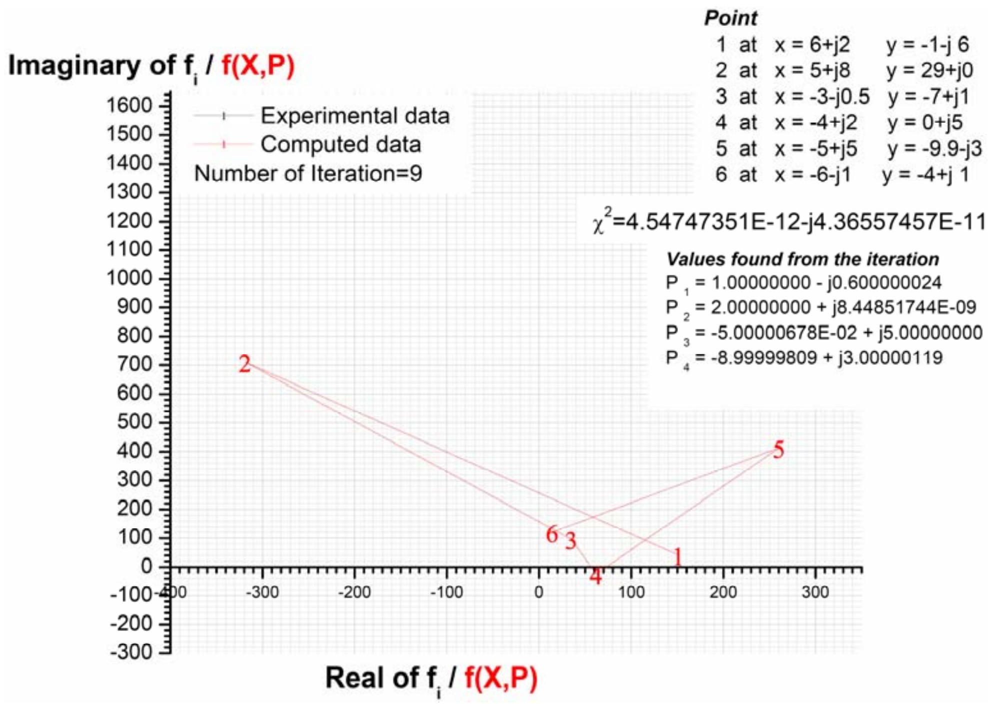

Figure 5 (d). Graphs of the experimental and numerical data at different number of iterations.

The function becomes $f(x, y, p_1, p_2, p_3, p_4) = x^{(1-j0.6)} + y^2 + (-0.05 + j5)xy - 9 + j3$. From the Figs. 5 (a)-(d), we can see that the LM is not affected by the order of the data (ascending or descending).

Based on the above results we can conclude that the LM algorithm is popular method and has the following advantages

(i) The parameters converge rapidly around the minimum in multi dimensional surfaces with complicated landscapes.

(ii) Even though the initial guess is poor, LM fits partly/most of the parameters to make fresh start.

(iii) The convergence speed needed to reach the minimum, is not significantly influenced by the number of parameters.

(iv) The shift parameters are not that much changed by normalized random errors. Only the minimum of the chi-function increases.

(v) Normalized random errors do not bring much change on the convergence speed, etc. Like any other non-linear optimization techniques, the LM algorithm method in finding global minimum is not guaranteed (this however can be secured by initializing parameters with a better guess).

3. SUMMARY

We extended the framework of the LM algorithm to real and complex multi-dimensional functions. The results show that LM is very efficient when Gradient Descent and Newton's methods separately failed to converge. In this paper we developed two programs (one for real and the other for complex or imaginary values) that work for any number of parameters, any number of dimensions and coordinate systems: Cartesian, Curvilinear etc. We believe that the algorithm also provides a concert support when someone wants to make a check at the instant of a fitting or when solving complex functions. Last but not least the LM method develops user's trust on the algorithm during fitting complicated surfaces and/or graphs.

4. ACKNOWLEDGEMENTS

We would like to acknowledge the moral support of all our staff members at the Department of Physics and the material support of the same department, College of Natural and Computational Sciences, Mekelle University. We are also grateful for the referees (internal and external) and the editors of the Momona Ethiopian Journal of Science for their critical and constructive comments and to the opportunity the Journal has given us to publish our paper in the first volume.

5. REFERENCES

Arumugam, M. 2003. EMPRR: A High-dimensional-Based Piecewise Regression Algorithm. pp. 4-16.

Avriel, M. 2003. Nonlinear Programming: Analysis and Methods. Dover Publishing. ISBN 0-486-43227-0.

Bates, D. M.& Watts, D. G. 1988. Nonlinear Regression and Its Applications. Wiley, New York.

Box, M. J., Davies D. & Swann, W.H. 1969. Non-Linear optimisation Techniques. Oliver & Boyd.

Coope, I. D. 1993. Circle fitting by linear and nonlinear least squares. Journal of Optimization Theory and Applications, 76 (2), Plenum Press, New York,.

Gill, P. R., Murray, W. & Wright, M. H. 1981. The Levenberg-Marquardt Method §4.7.3 in Practical Optimization. Academic Press, London, pp. 136-137.

Kelley, C. T. 1999. Iterative Methods for Optimization, SIAM Frontiers in Applied Mathematics, 18, ISBN 0-89871-433-8.

Lampton, M. 1997. Damping-Undamping, Strategies for the Levenberg-Marquardt Nonlinear Least-Squares Method. Computers in Physics Journal, 11(1): 110 – 115.

Lawson, C.L. & Hanson, R.J. 1974. Solving Least Squares Problems. Prentice-Hall.

Levenberg, K. 1944. A Method for the Solution of Certain Non-Linear Problems in Least Squares. The Quarterly of Applied Mathematics, 2: 164–168.

Lourakis, I. A. 2005. A Brief Description of the Levenberg-Marquardt Algorithm Implemented by levmar, Institute of Computer Science Foundation for Research and Technology - Hellas (FORTH), Vassilika Vouton.

Madsen, K., Nielsen, H.B. & Tingleff, O. 2004. Methods for Non-linear Least Squares Problems, Informatics and Mathematical Modeling. 2nd edition, Technical University of Denmark.

Marquardt, D. 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. SIAM Journal on Applied Mathematics, 11: 431–441.