Chapter 5
NONLINEAR SYSTEM MODELING WITH MAXIMUM LIKELYHOOD ESTIMATION

Chapter 5 introduces the Maximum Likelihood Estimation, a well established statistical tool, of network parameters. The theory is extended for the training of ANN for nonlinear modeling. Comparison of the performance is also made with the EKF methods developed in chapter 4.

Division of Electronics, SOE, CUSAT
5.1 Introduction

Many approaches are available to estimate the weights for training the neural network model in nonlinear system identification. So far the Back Propagation algorithm and Extended Kalman Filter algorithm have been examined in details. Maximum likelihood is a well established procedure for statistical estimation [99-100] and is implemented for modeling nonlinear systems and the performance is evaluated. In this procedure first formulate a log likelihood function and then optimize it with respect to the parameter vector of the probabilistic model under consideration [116-119]. The same four nonlinear systems are used for modeling.

5.2 Maximum Likelihood Estimation

The term “maximum likelihood estimate” with the desired asymptotic properties usually refers to a log of the likelihood equation that globally maximizes the likelihood function $L(x)$ [90-91]. In other words the ML estimate $x_{ML}$ is that value of the parameter vector $x$ for which the conditional probability density function $P(z/x)$ is maximum[115].
Chapter 5 Nonlinear system modeling with maximum likelihood estimation

The maximum likelihood estimate $\mathbf{x}_{\text{ML}}$ of the target parameters $\mathbf{x}$ from $N$ independent measurements is the mode of the conditional probability density function (likelihood function):

$$p\left(\frac{z}{x}\right) = \frac{1}{(2\pi)^{n/2}} \exp\left(-\frac{|(z-x)|^2}{2\sigma^2}\right)$$  \hspace{1cm} (5.1)

In the log likelihood function, $\log(P(z|x)) = -\frac{1}{2} (r_k^2)$, $r_k$ is the residual, $r_k = (d_k - z_k) / \sigma_k$, $\sigma_k$ is the standard deviation, $d_k$-desired value and $z_k$-estimated value. Maximizing log likelihood function $\log(P(z|x))$ is equivalent to minimizing the negative log likelihood function $L(z, x)$.

Switching over to the parameter $w$ in place of $x$, and by using the negative log-likelihood function $L(z, w)$, the ML problem is reformulated as a nonlinear least square problem:

$$\text{Minimize}(L(x, z)) \text{where} L(x, z) = \sum_{k=1}^{N} \frac{1}{2} r_k^2$$  \hspace{1cm} (5.2)

The ML estimate must satisfy the following optimality condition:

$$\Delta_x L(z, w_{\text{ML}}) = J(w_{\text{ML}})^T R(w_{\text{ML}}) = 0$$  \hspace{1cm} (5.3)

$R(w)$ is given by

$$R(w) = [r_1(w) \ldots \ldots r_N(w)]^T$$  \hspace{1cm} (5.4)

where $r_k = (d_k - z_k) / \sigma_k$.

and $J(w)$ the $N x n$ Jacobian matrix, and $J(w)^T = \Delta_w r(w)^T$  \hspace{1cm} (5.5)
The operator $\Delta w$ is defined as,

$$\Delta w = \left[ \frac{\partial}{\partial w_1} \quad \frac{\partial}{\partial w_2} \quad \ldots \quad \frac{\partial}{\partial w_N} \right]^T$$  \hspace{1cm} (5.6)

One could employ many optimization methods to find the ML Estimate [12].

Two well known optimization techniques chosen are:
1. Gauss-Newton method [90]
2. Conjugate-Gradient method [90]

### 5.3 System modeling using Gauss-Newton Method

A feed forward neural network model similar to earlier cases is designed for the identification of the same nonlinear systems and trained using Gauss-Newton method. The Gauss-Newton method is applicable to a cost function that is expressed as the sum of error squares.

$$E(w) = \sum_{k=1}^{N} \frac{1}{2} r_k^2$$  \hspace{1cm} (5.7)

The error signal $r(k)$ is a function of adjustable parameter vector $w$. Given an operating point $w(n)$, one could linearise the dependence of $r(k)$ on $w$ by writing,

$$r'(k, w) = r(k, w(n)) + \frac{\partial r(k)}{\partial w} w = w(n) (w - w(n)), \quad k=1,2,\ldots,n$$  \hspace{1cm} (5.8)

Equivalently, by using matrix notation one may write

$$r'(k, w) = r(k, w(n)) + J(n)^T (w - w(n))$$  \hspace{1cm} (5.9)

The updated parameter vector $w(n+1)$ is then defined by

$$w(n+1) = \arg \min_{w} \left\{ \frac{1}{2} r'(k, w)^2 \right\}$$  \hspace{1cm} (5.10)
The squared Euclidean norm of \( r(n,w) \) is,
\[
\frac{1}{2} r(n,w)^2 = \frac{1}{2} r(n)^2 + r(n)^T J(n) (w - w(n)) + \frac{1}{2} (w - w(n))^T J(n)^T J(n) (w - w(n)) \quad (5.11)
\]
Hence differentiating this expression with respect to \( w \) and setting the result equal to zero, it is possible to obtain,
\[
J(n)^T r(n) + J(n)^T J(n) (w - w(n)) = 0 \quad (5.12)
\]
Solving this equation for \( w \),
\[
w(n+1) = w(n) - (J(n)^T J(n) + \delta I)^{-1} J(n)^T r(n) \quad (5.13)
\]
which describes the pure form of the Gauss-Newton method.

However, for the Gauss-Newton iteration to be computable, the matrix product \( J(n)^T J(n) \) must be nonsingular. To guard against the possibility that \( J(n) \) being rank deficient, the usual practice is to add the diagonal matrix \( \delta I \) to the matrix \( J(n)^T J(n) \). The parameter \( \delta \) is a small positive constant chosen to ensure that, \( J(n)^T J(n) + \delta I \) is positive definite for all \( n \). The update equation accordingly becomes,
\[
w(n+1) = w(n) - (J(n)^T J(n) + \delta I)^{-1} J(n)^T r(n) \quad (5.14)
\]
where \( J(n) \) is the Jacobian matrix equal to \( \Delta w r(n) \)

### 5.4 Performance Analysis of MLE (Gauss-Newton)

The same set of four nonlinear systems, as in Chapter 4 is modeled with MLE also. The performance analysis is done by plotting the mean square error in
each case. Among the many systems modeled, the outputs of two nonlinear systems are presented in fig. 5.1 to fig. 5.8. The results are further used to get a conclusion on the model accuracy, its consistency and generalization capability.

5.4.1 Nonlinear system \( y = \sin(t^2 + t) \)

![Graph showing superposition of model and desired output with the MLE algorithm and the error vector (data set-1)](image)

Fig. 5.1 Superposition of model and desired output with the MLE algorithm and the error vector (data set-1)
Fig 5.1 shows the overlapped network and the model output for easy comparison of the performance. The second plot is the error vector over the training as well as validation samples. The overall MSE is plotted in Fig 5.2 below.

![MSE plot](image)

Fig 5.2 MSE for the result described in Fig 5.1
The overall MSE in this case is coming down to around 0.06356 only, compared to 0.0906 in EKF for the same nonlinear system. At the same time, the consistency of the model over a number of the systems is found to be generally good. The results obtained with the remaining data sets are described in the following figures.

5.4.2 Ambient Noise in the sea

![Figure 5.3: Superposition of model and desired output with the MLE algorithm and the error vector (data set-2)]
The average MSE for the first nonlinear system $y = \sin(t + t^2)$, is 0.0635 for MLE (G-N) and 0.0906 in EKF as demonstrated in figures 5.1 and 5.2. The MSE for ambient noise in the sea is 0.0083 for MLE and 0.0045 for EKF algorithm.
5.4.3 Acoustic source- ‘A’

Fig.5.5 Superposition of model and desired output with the MLE algorithm and the error vector (data set-3)
Fig 5.6 MSE for the result described in Fig 5.5

The overall MSE for the acoustic source-A is around 0.0118
5.4.4 Acoustic source- ‘B’

![Graph showing superposition of model and desired output with the MLE algorithm and the error vector (data set-4)](image_url)

In Fig 5.8, the MSE corresponding to this result is presented. It can be noticed that the MSE is coming down to around 0.005.
The overall MSE for the acoustic source-B is around 0.005.
5.5 System Identification using Conjugate – Gradient method

The conjugate gradient method belongs to a class of second order optimization methods known collectively as Conjugate direction methods [90].

Let $E_{av}(w)$ denote the cost function averaged over the training sample. Using Taylor series, it is possible to expand $E_{av}(w)$ about the current point $w(n)$ on the error surface, considering the second order terms.

$E_{av}(w(n)+\Delta w(n)) = E_{av}(w(n)) + g(n)^T \Delta w(n) + \frac{1}{2} \Delta w(n)^T H(n) \Delta w(n) + \text{third and higher order terms} \ (5.15)$

$g(n) = \frac{\partial E_{av}(w)}{\partial w} \bigg|_{w=w(n)} \quad (5.16)$

and $H(n)$ is the local Hessian matrix defined by

$H(n) = \frac{\partial^2 E_{av}(w)}{\partial w^2} \bigg|_{w=w(n)} \quad (5.17)$

For the Minimization of equation (4.42), third and higher order terms are neglected, differentiate w.r.t $\Delta w(n)$ and equate to zero, to obtain

$\Delta w(n) = H(n)^{-1} g(n) \quad (5.18)$

However the computation of $H(n)$ at every point $w(n)$ is difficult. On the other hand, the unconstrained minimization of the quadratic error
function $E_{av}$ can be done using a set of A-conjugate vectors $\{S_0, S_1, S_2, \ldots, S_{W-1}\}$ is defined by

$$S_i^T A S_j = 0 \text{ for } i \neq j$$  \hspace{1cm} (5.19)

The update equation is given by

$$w(n+1) = w(n) + \zeta(n) S(n) \quad n=0,1,2,\ldots,W-1$$  \hspace{1cm} (5.20)

where $w(0)$ is an arbitrary starting vector and $\zeta(n)$ is a scalar defined by

$$E_{av} (w(n) + \zeta(n) S(n)) = \min E_{av} (w(n) + \zeta S(n))$$  \hspace{1cm} (5.21)

where $\zeta$ is obtained from a one-dimensional minimization problem.

$$\zeta(n) = -S(n)^T A E(n) / S(n)^T A S(n): \quad n=0,1,\ldots,W-1$$  \hspace{1cm} (5.22)

where $E(n)$ is the error vector $w(n) - w^*$

But for the conjugate direction method to work, it requires the availability of a set of A conjugate vectors $(S(0), S(1), \ldots, S(W-1))$, and the final position $w^*$, which is not available. So the following procedure is adopted.

It is a special form of conjugate direction method. Here the successive direction vectors are generated as A-conjugate versions of the successive gradient vectors of the quadratic function $E_{av}$ as the method progresses. Except for $n=0$, the set of direction vectors $\{S(n)\}$ is not specified beforehand but rather it is determined in a sequential manner at successive steps of the method.

Define residual as the steepest descent direction:

$$r(n) = g(n) - H(n) \Delta w(n)$$  \hspace{1cm} (5.24)
Then to proceed $S(n)$ is taken as a linear combination of $r(n)$ and $S(n-1)$ as shown by,

$$S(n) = r(n) + \beta(n) S(n-1), \ n=1,2,\ldots,w-1.$$  \hspace{1cm} (5.25)

where $\beta(n)$ is a scaling factor.

Similar to gradient direction method,

$$\beta(n) = - S(n-1)^T A r(n) / S(n-1)^T A S(n-1)$$  \hspace{1cm} (5.26)

Using equations for $S(n)$ and $\beta(n)$ it is possible to define vectors $S(0)$, $S(1),\ldots,S(n-1)$. But equations need knowledge of matrix $A$. So it is required to evaluate $\beta(n)$ without explicit knowledge of $A$.

The formula defining $\beta(n)$,

$$\beta(n) = r(n)^T (r(n) - r(n-1))/r(n-1)^T r(n-1)$$  \hspace{1cm} (5.27)

This is known as Polak-Rebiere formula. A summary of the algorithm is now presented next [90-92]

**Initialization**

Unless prior knowledge on the weight vector $w$ is available, choose the initial value $w(0)$ as random.

**Computation**

1. For $w(0)$, compute the gradient vector $g(0)$.

2. Set $S(0) = r(0) = -g(0)$

3. At time step $n$, use a line search to find $\zeta(n)$ that minimizes $E_{av}(\zeta)$ sufficiently, representing the cost function $E_{av}$ expressed as a function of $\zeta$
for fixed values of $w$ and $S$.

4. Test to determine if the Euclidean norm of the residual $r(n)$ has fallen below a specified value, that is a small fraction of the initial value $r(0)$.

5. Update the weight vector $w(n+1) = w(n) + \zeta(n) S(n)$.

6. For $w(n+1)$, use back propagation to compute the updated gradient vector $g(n+1)$.

7. Set $r(n+1) = -g(n+1)$

8. Use Polak-Ribiere method to calculate:
   \[
   \beta(n+1) = \max\{ r(n+1)( r(n+1) - r(n) / r(n)^T r(n) ), 0 \}
   \]

9. Update the direction vector $S(n+1) = r(n+1) + \beta(n+1) S(n)$

10. Set $n = n+1$ and go back to step 3.

**Stopping Criterion**

Terminate the algorithm when the following condition is satisfied.

\[
|r(n)| \leq \varepsilon |r(0)|
\]

where $\varepsilon$ is a prescribed small number.
5.6 Performance analysis of MLE (Conjugate-Gradient)

The same nonlinear systems are modeled with MLE (Conjugate - gradient algorithm). The results are further used to get a conclusion on the model accuracy, its consistency and generalization capability.

5.6.1 Nonlinear system with output \( y = \sin(t^2 + t) \)

![Graph showing superposition of model and desired output with MLE-CG algorithm and the error vector](image)

Fig.5.9 Superposition of model and desired output with MLE-CG algorithm and the error vector (data set-1)
Fig 5.10 MSE for the result described in Fig 5.9

The error in MLE algorithm, MSE over the number of epochs is around 0.0825
5.6.2 Ambient noise in the sea

Fig. 5.11 Superposition of model and desired output with MLE-CG algorithm and the error vector (data set-2)
It can be observed that the validation error (MSE) is around 0.0122
5.6.3 Acoustic source- ‘A’

![Graph showing model and desired output with error vector data set-3]

Fig. 5.13 Superposition of model and desired output with MLE-CG algorithm and the error vector (data set-3)
It can be observed that the validation error (MSE) is around 0.0122
5.6.4 Acoustic source- ‘B’

Fig.5.15 Superposition of model and desired output with MLE-CG algorithm and the error vector (data set-4)
Fig 5.16 shows that the validation error (MSE) is around 0.0122.

Thus all the systems have been modeled with Gauss-Newton and Conjugate gradient methods. A detailed performance measure has been provided with
the plots of mean square error (MSE) in each case. The error trajectory and final values of MSE can be used for the performance assessment. A detailed comparison of the EKF and its variances with the current method of MLE is presented in the next section.

5.7 Comparison between the various MLE methods for modeling

Table 5.1 Performance comparison of EKF, EKF with EM and MLE in the MSE sense

<table>
<thead>
<tr>
<th>System</th>
<th>Mean Square Error (MSE)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Simple EKF</td>
<td>EKF with EM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gauss-Newton</td>
<td>Conjugate Gradient</td>
</tr>
<tr>
<td>$y = \sin(t + t^2)$</td>
<td>0.0906</td>
<td>4.132x10^{-4}</td>
<td>0.0635</td>
</tr>
<tr>
<td>Ambient noise</td>
<td>0.0045</td>
<td>8.06x10^{-4}</td>
<td>0.0083</td>
</tr>
<tr>
<td>Acoustic source ‘A’</td>
<td>0.0054</td>
<td>0.0065</td>
<td>0.0118</td>
</tr>
<tr>
<td>Acoustic source ‘B’</td>
<td>0.0038</td>
<td>0.000174</td>
<td>0.0092</td>
</tr>
</tbody>
</table>

Table 5.1 shows the performance comparison of EKF and MLE. The EKF algorithm converges faster and has marginally good performance compared to BP algorithm and MLE. It is also consistent for all the nonlinear systems modeled. The performance of EKF can be again improved by EM algorithm as shown in the table. The MLE algorithm also gives good results and
computationally efficient but in problems where faster convergence is required, as in adaptive filters and real world problems; Kalman Estimation has to be used.

5.8 Conclusion

In an attempt to compare the performance of the EKF based algorithms for estimating the network parameters, Maximum Likelihood Estimation (MLE) for modeling nonlinear systems has been implemented and the performance results are compared. The same Feed forward neural network is used as the model structure. Four nonlinear systems are modeled using two different techniques of implementing the MLE viz. the Gauss-Newton method and Conjugate gradient methods. The results show good performance of the estimation technique in respect of MSE. A comparison is also made among all the different methods tried out viz. the EKF algorithm, the EKF algorithm with EM, the Gauss-Newton method and the Conjugate gradient method. It is seen that the performance of MLE is good but inferior to that of EKF in terms of mean square error. It could be due to the variances in the choice of the parameters like β in the Conjugate gradient descent algorithm. On the other hand, the EKF algorithm and its variants seem to be performing very well with minimum dependence on the choice of parameters.