# A Study of Maximum Likelihood Estimation with Non-linear Iterative Method

Yizheng Liao*

August 30, 2010

## 1 Introduction

In Rife and Boorstyn's work [1], a maximum likelihood estimation (MLE) algorithm was presented to estimate the parameters of a single tone. However, they provide few details about the implementation of this specific algorithm. The purpose of this document is providing a detail study of Rife's algorithm. Section 2 will provide background information about Rife's algorithm. Sections 3 and 4 will present the estimation algorithms of complex signal model and real signal model. Section 5 will discuss the results.

## 2 Background

### 2.1 Complex Signal Observation Model

In Rife's paper[1], the received complex signal model is

$$z(t) := b_0 \exp(j(\omega_0 t + \theta_0)) + w(t) \tag{1}$$

where $b_0$, $\omega_0$ and $\theta_0$ denote the unknown amplitude, frequency and phase of the signal respectively, and $w(t)$ denotes zero-mean proper complex additive white Gaussian Noise (AWGN).The received signal is sampled at a constant sampling frequency rate $f_s := 1/T$ to produce the discrete-time observation

$$z[n] := z(t_0 + nT) = b_0 \exp(j(\omega_0(t_0 + nT) + \theta_0)) + \eta[n] \tag{2}$$

for $n := 0, ..., N-1$ where $t_0$ denotes the time of the first sample and $\eta[n]$ is a zero-mean proper complex Gaussian random variable with $\text{var}\{\text{Re}(\eta[k])\} = \text{var}\{\text{Im}(\eta[k])\} = \sigma^2$ and $\text{cov}\{\text{Re}(\eta[k]), \text{Im}(\eta[k])\} = 0$. Here we assumed that $\eta[n]$ are independent and identically distributed (i.i.d) for $n := 0, ..., N-1$.

The $N$-sample observation of (2) is provided as an input to a phase and frequency estimator. The phase and frequency estimates generated by the estimators are denoted as $\hat{\theta}$ and $\hat{\omega}$ respectively, and the resulting phase and frequency errors are denoted as $\tilde{\theta} := \theta_0 - \hat{\theta}$ and $\tilde{\omega} := \omega_0 - \hat{\omega}$, respectively.

*The author is an undergraduate student in the Eletrical and Computer Engineering, Worcester Polytehnic Institute, Worcester, MA 01609 USA. email: liaoyizheng@wpi.edu

## 2.2 Real Signal Observation Model

As discussed in [2], the received real signal model is the real part of (2). Therefore, the real signal observation is

$$x[n] := b_0 \cos(\omega_0(t_0 + nT) + \theta_0) + \eta[n] \tag{3}$$

for $n := 0, ..., N-1$ where $t_0$ denotes the time of the first sample and $\eta[n]$ is zero-mean proper real AWGN with $\text{var}\{\eta[k]\} = \sigma^2$. The $N$-sample observation of (3) is provided as the input to a phase and frequency estimator, which generated the phase estimate, $\hat{\theta}$, and the frequency estimate, $\hat{\omega}$. The resulting phase and frequency errors are denoted as $\tilde{\theta} := \theta_0 - \hat{\theta}$ and $\tilde{\omega} := \omega_0 - \hat{\omega}$, respectively.

## 2.3 Complex Signal Cramer-Rao Lower Bound

In this estimation system,we use Cramer-Rao Lower Bound (CRLB) to measure system inaccuracy. The Fisher Information Matrix for CRLB for complex signal is [1, 3]

$$\mathbf{J}(\beta) := \frac{1}{\sigma^2} \begin{bmatrix} b_0^2 T^2 (n_0^2 N + 2n_0^2 P + Q) & 0 & b_0^2 T (n_0 N + P) \\ 0 & N & 0 \\ b_0^2 T (n_0 N + P) & 0 & b_0^2 N \end{bmatrix} \tag{4}$$

where

$$\begin{aligned} P &:= \sum_{n=0}^{N-1} n = \frac{N(N-1)}{2} && (5) \\ Q &:= \sum_{n=0}^{N-1} n^2 = \frac{N(N-1)(2N-1)}{6} \\ \beta &:= [\omega, b, \theta]^T && (6) \end{aligned}$$

and $t_0 = n_0 T$ is the time at which the first sample is taken.

When all three parameters are unknown, after inverting all the variations of $\mathbf{J}$, the variances obtain the following set of bounds:

$$\begin{aligned} \text{var}\{\hat{b_0}\} &\geq \frac{\sigma^2}{N} && (7) \\ \text{var}\{\hat{\omega_0}\} &\geq \frac{12\sigma^2}{b^2 T^2 N (N^2 - 1)} && (8) \\ \text{var}\{\hat{\theta_0}\} &\geq \frac{12\sigma^2 (n_0^2 N + 2n_0 P + Q)}{b_0^2 N^2 (N^2 - 1)} && (9) \end{aligned}$$

## 2.4 Real Signal Cramer-Rao Lower Bound

When the received signal is real, the Fisher Information Matrix changes to the approximations in [4]

$$\mathbf{I}(\gamma) := \frac{1}{\sigma^2} \begin{bmatrix} \frac{2}{N} & 0 & 0 \\ 0 & 2b_0^2 \pi^2 Q & \pi b_0^2 P \\ 0 & \pi b_0^2 P & \frac{N b_0^2}{2} \end{bmatrix} \tag{10}$$

where

$$\gamma := [b, \omega, \theta]^T$$

Hence, after inverting all the variations of $\mathbf{I}$ the variances obtain the following set of bounds:

$$\begin{aligned} \text{var}(\hat{b_0}) &\geq \frac{2\sigma^2}{N} && (11) \\ \text{var}(\hat{\omega_0}) &\geq \frac{24\sigma^2}{(2\pi)^2 N(N^2-1)} && (12) \\ \text{var}(\hat{\theta_0}) &\geq \frac{4(2N-1)\sigma^2}{N(N+1)} && (13) \end{aligned}$$

## 2.5 Complex Signal Maximum Likelihood Estimation

According to [1], when all three parameters are unknown, the frequency should be estimated at first. Upon receiving the discrete time observations according to (2) for $n := 0, ..., N-1$, the maximum likelihood frequency estimate can be computed as [1]

$$\hat{\omega} = \max_{\omega} |A(\omega)| \tag{14}$$

where

$$A(\omega) := \frac{1}{N} \sum_{n=0}^{N-1} z[n] \exp(-jn\omega T) \tag{15}$$

Once the maximum likelihood frequency estimate has been computed, the maximum likelihood phase and amplitude follow as

$$\hat{\theta} = \angle\{\exp(-j\hat{\omega}t_0) A(\hat{\omega})\} \tag{16}$$

$$\hat{b} = |A(\hat{\omega})| \tag{17}$$

## 2.6 Real Signal Maximum Likelihood Estimation

For the real signal MLE, still, when all the three parameters are unknown, the frequency should be estimated at first. Since the signal model is changed, the formula (15) need to be re-computed as well. Rife's algorithm simply took the real part of (15), which is

$$V(\omega) := \frac{1}{N} \sum_{n=0}^{N-1} x_n \exp(-jn\omega T)$$

where $x_n$ is (3). Then the frequency estimate was generated by maximizing $|V(\omega)|$. However, in experiment, we found this approximate may not work. The reason is that if $w_0$ is near 0 or $\frac{1}{2}\omega_s$, this approximate will be incorrect. Therefore, we use a more accurate model from [4].

According (3), the maximum likelihood estimation of amplitude $b_0$, frequency $\omega_0$, and phase $\theta_0$ is found by minimizing

$$\mathbf{D}(b, \omega, \theta) := \sum_{n=0}^{N-1} (x[n] - b\cos(\omega nT + \theta))^2$$

Suppose $\alpha_1 := b\cos\theta$ and $\alpha_2 := -b\sin\theta$, then

$$b = \sqrt{\alpha_1^2 + \alpha_2^2} \tag{18}$$

$$\theta = \arctan\left(\frac{-\alpha_2}{\alpha_1}\right) \tag{19}$$

Also, let

$$\mathbf{c} := [1, \cos(2\pi\omega T), ..., \cos(2\pi\omega T(N-1))]^T$$
$$\mathbf{s} = [1, \sin(2\pi\omega T), ..., \sin(2\pi\omega T(N-1))]^T$$

Then, we have

$$\mathbf{D}^{'}(\alpha_1, \alpha_2, \omega) = (\mathbf{x} - \alpha_1\mathbf{c} - \alpha_2\mathbf{s})^T(\mathbf{x} - \alpha_1\mathbf{c} - \alpha_2\mathbf{s}) = (\mathbf{x} - \mathbf{H}\alpha)^T(\mathbf{x} - \mathbf{H}\alpha)$$

where $\alpha = [\alpha_1, \alpha_2]^T$ and $\mathbf{H} = [\mathbf{c}, \mathbf{s}]$. In order to get the maximum likelihood frequency estimate, we need to maximize

$$\begin{aligned}
\mathbf{R}(\omega) &:= \begin{bmatrix} \mathbf{c}^T\mathbf{x} \\ \mathbf{s}^T\mathbf{x} \end{bmatrix}^T \begin{bmatrix} \mathbf{c}^T\mathbf{c} & \mathbf{c}^T\mathbf{s} \\ \mathbf{s}^T\mathbf{c} & \mathbf{s}^T\mathbf{s} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{c}^T\mathbf{x} \\ \mathbf{s}^T\mathbf{x} \end{bmatrix} \\
&= \begin{bmatrix} \sum_{n=0}^{N-1} x_n \cos(Tn\omega) \\ \sum_{n=0}^{N-1} x_n \sin(Tn\omega) \end{bmatrix}^T \\
&\quad \begin{bmatrix} \sum_{n=0}^{N-1} \cos^2(Tn\omega) & \sum_{n=0}^{N-1} \cos(Tn\omega)\sin(Tn\omega) \\ \sum_{n=0}^{N-1} \cos(Tn\omega)\sin(Tn\omega) & \sum_{n=0}^{N-1} \sin_2(Tn\omega) \end{bmatrix}^{-1} \begin{bmatrix} \sum_{n=0}^{N-1} x_n \cos(Tn\omega) \\ \sum_{n=0}^{N-1} x_n \sin(Tn\omega) \end{bmatrix}
\end{aligned} \tag{20}$$

Therefore, once the maximum likelihood frequency estimate has been estimated correctly, we can computer $\hat{\alpha}$ by using

$$\hat{\alpha} = (\mathbf{H}^T\mathbf{H}^{-1}\mathbf{H}^T\mathbf{x})$$

Then the maximum likelihood phase and amplitude can be generated by using (18) and (19).

## 3 Complex Signal Estimation

In this section, we use (2) as the signal model. In [1], the authors presented a two-part search routine to compute $\hat{\omega}$. The first part calculates $|A(\omega)|$ for a set of $\omega$ values between zeros and $\omega_s$, which is Nyquist frequency, and identifies the $\omega$ that maximizes $|A(\omega)|$ over this set of $\omega$ values. This part is called the *coarse search*. The second part locates the local maximum closet to the value of $\omega$ picked out by the coarse search. This part is called the *fine search*.

For the coarse search, we used the discrete Fourier transform (DFT) to find $\omega$ which maximizes $|A(\omega)|$. In fact, the $M$-point DFT is a sampled version of $A(\omega)$ at frequencies

$\omega = \frac{2\pi k}{MT}$ for $k := 0, ..., M-1$. Usually, we use the fast Fourier transform (FFT) to compute $A(\frac{2\pi k}{MT})$ for $k := 0, ..., M-1$. Then we select the index $k$ at which $A(\frac{2\pi k}{MT})$ attains its maximum magnitude, i.e.,

$$\hat{\omega} = \max_{\omega \in \Omega} |A(\omega)| \tag{21}$$

where $\Omega := \left\{0, \frac{2\pi}{MT}, ..., \frac{2\pi(M-1)}{MT}\right\}$. However, the accuracy of this method is strongly affected by the number of points of FFT. Therefore, in [3], the FFT magnitude is interpolated to improve the estimation accuracy. For example, suppose the maximum in (21) occurs at FFT index $\hat{k}$. A quadratic fit $y = ax^2 + bx + c$ in the neighbourhood of the maximum can be computed given the frequencies $x \in \left\{\frac{2\pi(\hat{k}-1)}{MT}, \frac{2\pi\hat{k}}{MT}, \frac{2\pi(\hat{k}+1)}{MT}\right\}$ and FFT magnitudes $y = |A(x)|$. Then maximum likelihood frequency estimate can be computed using standard techniques as $\hat{\omega} = \frac{-b}{2a}$. This is method is described as the Quad MLE in this document.

Although the coarse search can locate the maximum likelihood frequency by using FFT, and the Quad MLE can improve the precision further, both results may not be accurate enough to attain the CRLB at high SNR. Figure 1 and Figure 2 show that for Mean Squared Error(MSE) the coarse search only MLE cannot achieve the CRLB in the entire SNR range. For the Quad MLE, only when the value of M is $2^{16}$, the MSE can overlap the CRLB up to 60dB SNR. Therefore, the fine search is required to improve estimation accuracy. The MLE method contains both the coarse search and the fine search is described as the full method in this document. In [1], the secant method was mentioned to locate the value of $\omega$ closet to $\hat{\omega}$ that maximizes $|A(\omega)|$. However, the author did not provide many details about it.

## 3.1 Secant Method

The secant method is a iterative method used to find roots of a non-linear system. The iteration formula is [5]:

$$\begin{aligned} x_n &:= \frac{x_{n-2} f(x_{n-1}) - x_{n-1} f(x_{n-2})}{f(x_{n-1}) - f(x_{n-2})} \\ &= x_{n-1} - f(x_{n-1}) \frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})} \end{aligned} \tag{22}$$

The two initial values, $x_0$ and $x_1$, are chosen to lie close to root.

Finding the maximum of $|A(\omega)|$ is equivalent to find the root of the first order derivative of $|A(\omega)|$. Therefore, (2) can be re-written as $z[n] = x[n] + jy[n] = \sum_{n=0}^{N-1} x_n + jy_n$. Then (15) can be written in rectangular form as:

$$A(\omega) := B(\omega) + jC(\omega) \tag{23}$$

where

$$B(\omega) := \frac{1}{N} \sum_{n=0}^{N-1} x_n \cos(n\omega T) + y_n \sin(n\omega T) \tag{24}$$

$$C(\omega) := \frac{-1}{N}\sum_{n=0}^{N-1} x_n \sin(n\omega T) - y_n \cos(n\omega T) \tag{25}$$

Also, the maximization of $|A(\omega)|$ is equivalent to the maximization of

$$|A(\omega)|^2 := A(\omega)A^*(\omega) = B^2(\omega) + C^2(\omega) = U(\omega)$$

Hence, we have

$$\frac{dU}{d\omega} := 2B\frac{dB}{d\omega} + 2C\frac{dC}{d\omega} \tag{26}$$

where

$$\frac{dB}{d\omega} := \frac{1}{N}\sum_{n=0}^{N-1} Tn(y_n \cos(Tn\omega) - x_n \sin(Tn\omega))$$

$$\frac{dC}{d\omega} := \frac{-1}{N}\sum_{n=0}^{N-1} Tn(x_n \cos(Tn\omega) + y_n \sin(Tn\omega))$$

Therefore, (22) becomes

$$\omega_n = \omega_{n-1} - U^{'}(\omega_{n-1})\frac{\omega_{n-1} - \omega_{n-2}}{U^{'}(\omega_{n-1}) - U^{'}(\omega_{n-2})}$$

When $|\omega_n - \omega_{n-1}| \leq \epsilon$, $U^{'}(\omega_n) = U^{'}(\omega_{n-1})$, or $U^{'}(\omega_n) = 0$, the iteration will stop. In our experiment, we choose the first condition to terminate the process. The algorithm of the secant method is described in Algorithm 1.

**Algorithm 1** The Secant Method

```
Import $\omega_0, \omega_1$
$\Delta = 0, n = 1$
while $|\Delta| \geq \epsilon$ do
  $n = n + 1$
  $\Delta = U^{'}(\omega_{n-1}) * \frac{\omega_{n-1} - \omega_{n-2}}{U^{'}(\omega_{n-1}) - U^{'}(\omega_{n-2})}$
  $\omega_n = \omega_{n-1} - \Delta$
end while
return $\omega_n$
```

## 3.2 Bisection Method

Since the secant method cannot ensure convergence when the interval is not very small [5], an alternatively iterative method, the bisection method, is introduced here. If $f$ is a continuous function on the interval $[a, b]$ and $f(a)f(b) < 0$, then the bisection method converges to the root of $f$ by halving the range. The best estimation is the midpoint of the smallest range found. Therefore, after $n$ steps, the absolute error is

$$\frac{|b - a|}{2^n}$$

The algorithm of the bisection method is described in Algorithm 2.

**Algorithm 2** The Bisection Method

```
Import $\omega_0, \omega_1$
if $U'(\omega_0) \leq 0$ then
    $lo = \omega_0$
    $hi = \omega_1$
else
    $lo = \omega_1$
    $hi = \omega_0$
end if
$\omega_2 = lo + \frac{(hi-lo)}{2}$
$n = 2$
while $(\omega_{n-1} \neq lo) AND (\omega_{n-1} \neq hi)$ do
    $n = n + 1$
    if $U'(\omega_{n-1}) \leq 0$ then
        $lo = \omega_{n-1}$
    else
        $hi = \omega_{n-1}$
    end if
    $\Delta = \frac{hi-lo}{2}$
    $\omega_n = lo + \Delta$
    if $|\Delta| \leq \epsilon$ then
        BREAK
    end if
end while
return $\omega_n$
```

## 3.3 Hybrid Iteration Method

Since the initial point of the bisection method have different signs, hence, we can ensure that the bisection method can converge finally. However, the convergence of the bisection method is linear. Therefore, the convergence speed is slow. Thus, for the fine search of maximum likelihood estimation, we use a hybrid iterative method. The primary method is the secant method. When the iterative steps and $|U(\omega_0) - U(\omega_n)|$ are large, the algorithm will shift to the bisection method. In our experiment, when the iteration step is over 1000 or $|U(\omega_0) - U(\omega_n)| \geq 0.1$, the iteration method will shift to the bisection method.

In some literatures, like [6] and [7], Newton's Method was used as the iterative method. However, as discussed in [2], when $U''(\omega)$ was very small, Newton's method did not converge but the secant did. In addition, the function $U''(\omega)$ is not easy to calculate.

# 4 Real Signal Estimation

The maximum likelihood estimation algorithm of real signal is similar to that of complex signal. For the coarse search routine, we still use FFT to find the index $k$ at which $V(\frac{2\pi k}{MT})$ attains its maximum magnitude. After locate the index $k$, we shift to the fine search routine. In order to apply the hybrid iteration method, firstly, we compute the first order derivative of (20) respect to $\omega$, which is

$$\frac{d\mathbf{R}}{d\omega} = \left(\frac{d\mathbf{F}}{d\omega}\right)^T \mathbf{G}^{-1}\mathbf{F} + \mathbf{F}^T\left(\frac{d\mathbf{G}^{-1}}{d\omega}\right)\mathbf{F} + \mathbf{F}^T\mathbf{G}^{-1}\left(\frac{d\mathbf{F}}{d\omega}\right) \tag{27}$$

where

$$\begin{aligned}
\frac{d\mathbf{F}}{d\omega} &= \begin{bmatrix} \sum_{n=0}^{N-1} -x_n Tn\sin(Tn\omega) \\ \sum_{n=0}^{N-1} x_n Tn\cos(Tn\omega) \end{bmatrix} \\
\frac{d\mathbf{G}}{d\omega} &= \begin{bmatrix} \sum_{n=0}^{N-1} -Tn\sin(2Tn\omega) & \sum_{n=0}^{N-1} Tn\cos(2Tn\omega) \\ \sum_{n=0}^{N-1} Tn\cos(2Tn\omega) & \sum_{n=0}^{N-1} Tn\sin(2Tn\omega) \end{bmatrix} \\
\frac{d\mathbf{G}^{-1}}{d\omega} &= -\mathbf{G}^{-1}\frac{d\mathbf{G}}{d\omega}\mathbf{G}^{-1}
\end{aligned}$$

Now we can use the Algorithm 1 and Algorithm 2 to locate $\hat{\omega}$. After finding $\hat{\omega}$, we can compute $\hat{\alpha}$ and then estimate $\hat{b}$ and $\hat{\theta}$ by using (18) and (19). Then we can estimate $\hat{b}$ and $\hat{\theta}$ by using (18) and (19).

## 5 Numerical Results and Discussion

This section presents numerical results comparing phase and frequency estimators of the coarse search only MLE, the Quad MLE and the full MLE for different values of $M$. All of the results in this section assume an observation with $N = 2000$ samples at $f_s = 16$kHz. 5000 realizations of the complex signal using (2) or the real signal using (3) and AWGN were generated with fixed $b_0 = 1$ and random independent uniformly distributed phase and frequency centred at 1020Hz according to

$$\theta_0 \sim U(-\pi, \pi)$$

$$\omega_0 \sim U(2\pi \cdot 1010, 2\pi \cdot 1030)$$

The error boundary $\epsilon$ is $1 \cdot 10^{-6}$.The independent AWGN in each realization was generated with independent real and imaginary components for complex signal and with independent real component for real signal. The AWGN has zero mean and variance $\sigma^2$.

Figure 1 and 2 show the mean squared frequency and phase estimation error, respectively, of the coarse search only MLE, the Quad MLE and the full MLE frequency and phase estimators as a function of SNR $:= 10\log_{10}(b_0^2/\sigma^2)$ for three values of $M$ for the complex signal. Figure 3 and 4 show same thing as Figure 1 and Figure 2 respectively but for the real signal. The Quad MLE is implemented with quadratic interpolation as discussed in previous section.

For complex signal, the coarse search only MLE method provides the worst estimation performance among all of the MLE methods. The Quad MLE with $M = 2^{12}$ have the same performance of the coarse search only MLE with $M = 2^{16}$. For Quad MLE, the

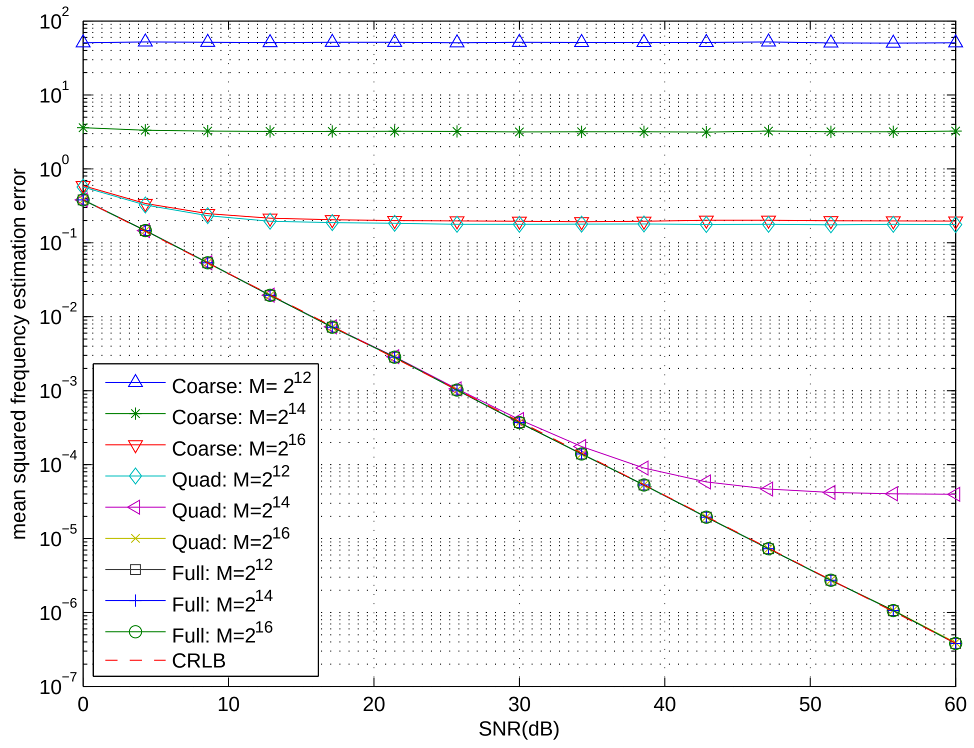

Figure 1: Mean squared frequency estimation error for complex signal

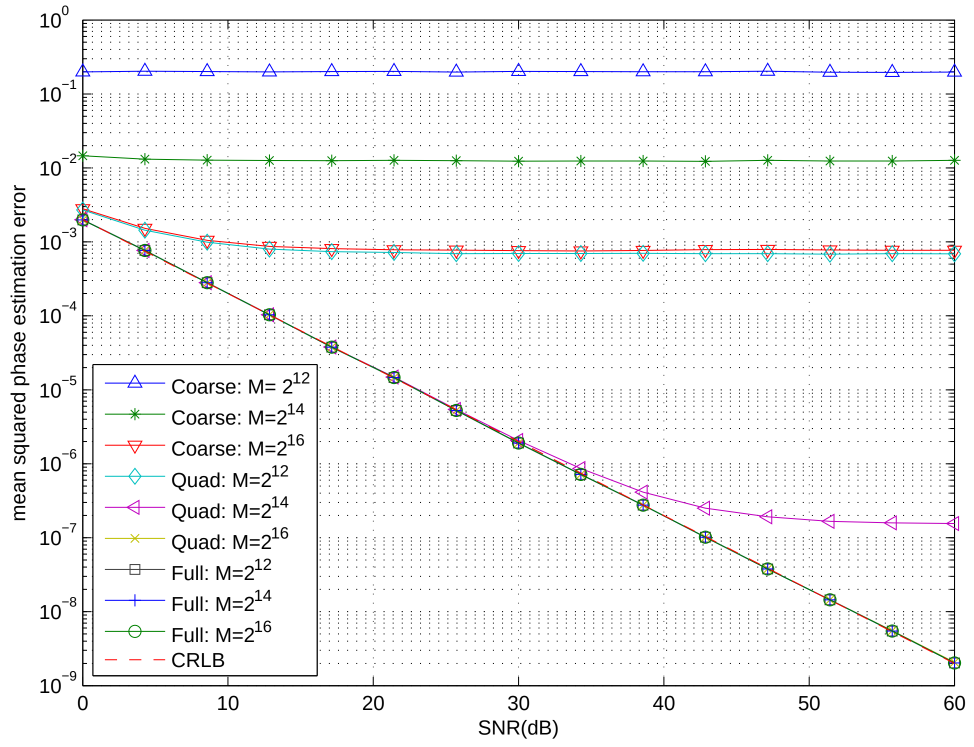

Figure 2: Mean squared phase estimation error for complex signal

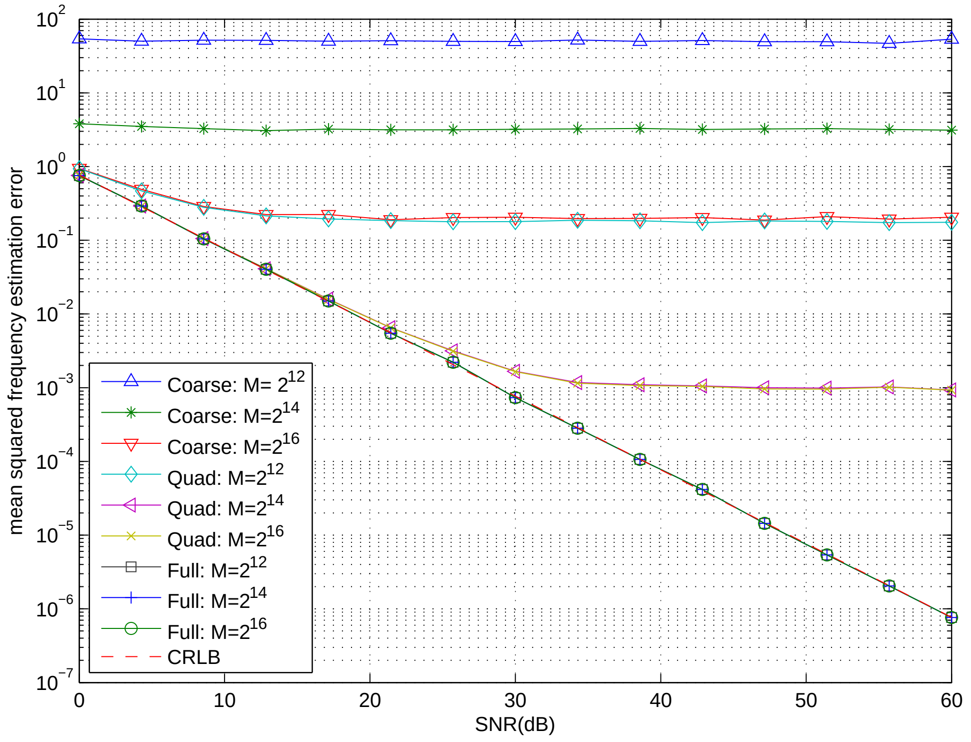

Figure 3: Mean squared frequency estimation error for real signal

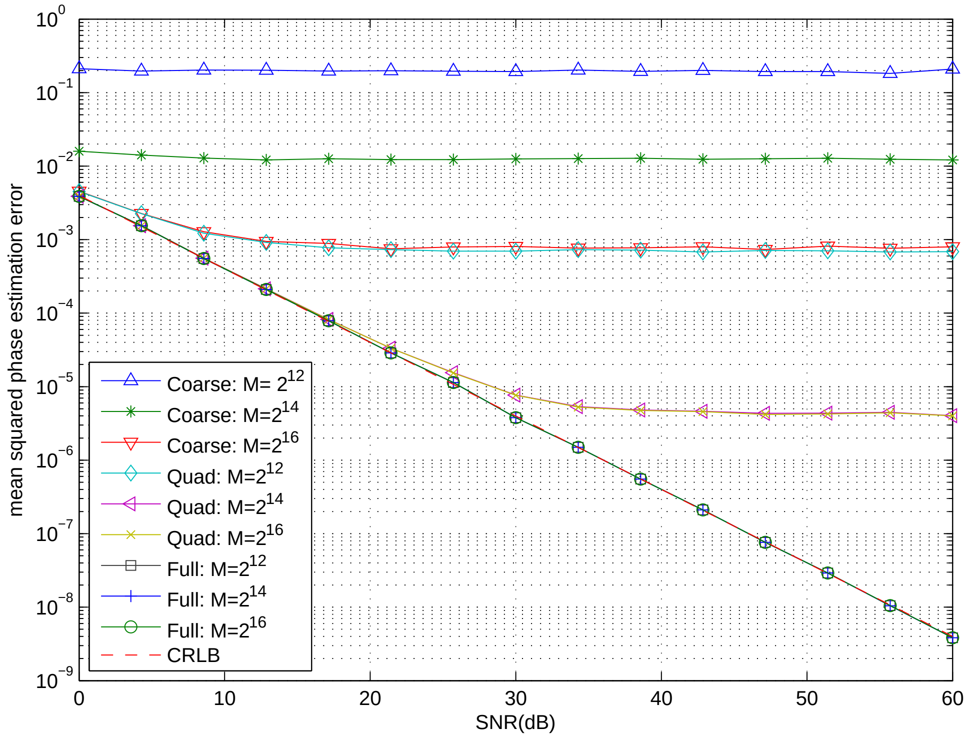

Figure 4: Mean squared phase estimation error for real signal

CRLB can be achieved only when the value of $M$ is $2^{16}$. For the full MLE, the CRLB can always be achieve.

For real signal, the worst estimation method is still the coarse search only MEL method. The CRLB cannot be achieved for the entire SNR range. For the Quad MLE, the MSE overlaps the CRLB before 20dB. In addition, the performance cannot be improved further when the value of $M$ is $2^{14}$. For the complete MLE, the CRLB can always be achieved.

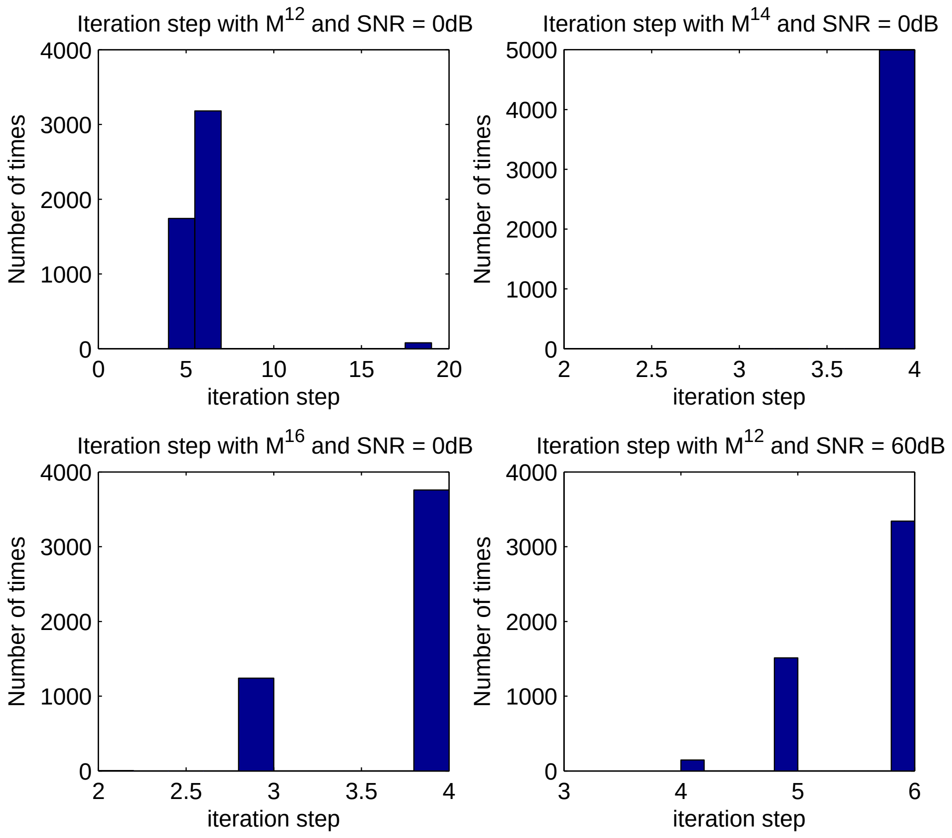

Figure 5: Iteration Step

In Figure 5, the iteration is completed within 19 steps when the SNR is 0dB and the value of $M$ is $2^{12}$. With the increase of the value of $M$, the iteration can be done within four steps. When $M$ is $2^{12}$ and the SNR is increased to 60dB, the iteration takes six steps at most. Compared with the Quad MLE, the complete MLE method has much better performance with lower complexity.

In these examples, the value of $M$ does not affect the performance of the complete MLE method. In addition, the computation complexity of the fine research routine is low. Therefore, this method points a potential method for real-time implementation in the future.

## 6 Conclusion

This document presented a performance comparison among three different maximum likelihood estimation methods. We developed Rife' method and implement it for different $M$ values. The numerical results show that the full MLE method can always achieve Cramer-Rao lower bound over a wide range of signal to noise ratios. The discussion of iteration steps purposes the low computational complexity of the full MLE method.

## References

[1] D. Rife and R. Boorstyn, "Single-tone parameter estimation from discrete-time observations," *IEEE Transactions on Information Theory*, vol. 20, no. 5, pp. 591–598, 1974.

[2] D. Rife, *Digital tone parameter estimation in the presence of Gaussian noise.* PhD thesis, Polytechnic Institute of Brooklyn, 1973.

[3] D. R. Brown III, Y. Liao, and N. Fox, "Low-complexity real-time single-tone phase and frequency estimation," in *2010 Military Communication Conference*, (San Jose, CA), OCT 31 - NOV 3 2010.

[4] S. Kay, *Fundamentals of statistical signal processing: estimation theory.* 1993.

[5] F. Hildebrand, *Introduction to numerical analysis.* Dover Pubns, 1987.

[6] T. Abatzoglou, "A fast maximum likelihood algorithm for frequency estimation of a sinusoid based on Newton's method," *IEEE transactions on acoustics, speech, and signal processing*, vol. 33, no. 1, pp. 77–89, 1985.

[7] D. Dorfman and E. Alf, "Maximum likelihood estimation of parameters of signal detection theory?a direct solution," *Psychometrika*, vol. 33, no. 1, pp. 117–124, 1968.

# A Complex Signal Estimation Function

## A.1 Main Function

```
function [ahat,omegahat,thetahat,counter] = ML_fun(N,s,nfft,fs,el,sindex,eindex)

% Input Paramter:
% N: length of signal
% s: input signal
% nfft: number of points of FFT
% fs: sampling frequency
% el: the error bound
% sindex: start index of signal
% eindex: end index of signal

% Output Parameter
% ahat: amplitude estimate
% omegahat: frequency estimate (in rad/s)
% thetahat: phase estimate
% counter: iteration steps

A = fft(s,nfft)/N;
[junk,index] = max(abs(A(1:nfft/2)));
n = sindex:eindex;

x0 = (index-2);
x1 = index;
y0 = dA_fun(N,s,x0,nfft,n,fs);
y1 = dA_fun(N,s,x1,nfft,n,fs);

if abs(A_fun(N,s,x0,nfft,n,fs)) > abs(A_fun(N,s,x1,nfft,n,fs))
        temp = x0;
        x0 = x1;
        x1 = temp;
        temp = y0;
        y0 = y1;
        y1 = temp;
end

counter = 0;
delta = 1;
peak = abs(A_fun(N,s,index-1,nfft,n,fs));
while abs(delta) > el
%Secant method
        delta = y1*(x1-x0)/(y1-y0);
        x0 = x1;
        x1 = x1-delta;
        y0 = y1;
        y1 = dA_fun(N,s,x1,nfft,n,fs);
        counter = counter+1;
        dd = peak - abs(A_fun(N,s,x1,nfft,n,fs));

        if (counter > 1000)||(abs(dd)>0.1)
        %shift to Bisection method
                aa = index-2;
                bb = index;
                while dA_fun(N,s,aa,nfft,n,fs)*dA_fun(N,s,bb,nfft,n,fs) >= 0
                        if dA_fun(N,s,aa,nfft,n,fs)<0
                                aa = aa-1;
                        else if dA_fun(N,s,bb,nfft,n,fs) > 0
                                        bb = bb+1;
                                 end
                        end
                end
                if dA_fun(N,s,aa,nfft,n,fs) <= 0
                        lo = aa;
                        hi = bb;
                else
                        lo = bb;
                        hi = aa;
                end

                mid = lo+(hi-lo)/2;
                counter = 0;
                while (mid ~= lo) && (mid ~= hi)
                        if dA_fun_real(N,s,mid,nfft,n,fs) <=0
                                lo = mid;
                        else
                                hi = mid;
                        end
                        delta = (hi-lo)/2;
                        mid = lo + delta;
                        if abs(delta) < el
                                break;
                        end
                        counter = counter+1;
                end
                x1 = mid;
                break;
        else continue;
        end
end

omegahat = x1/nfft*fs*2*pi;
thetahat_secant(kk,i) = angle(exp(-j*(omegahat_secant(i)*t0))*A_fun(N,s,x1,nfft,n,fs)
ahat = abs(A_fun(N,s,x1,nfft,n,fs));
```

## A.2 Function $U(\omega)$

```
function value = A_fun(N,xx,k,nfft,nn,fs)
% Input Paramter:
% N: length of signal
% xx: input signal
% k: index of FFT
% nfft: number of points of FFT
% nn: discrete index
% fs: sampling frequency

% Output Parameter:
% value: A(k)

w = 2*pi*k*fs/nfft;

T = 1/fs;
j = sqrt(-1);

value = (1/N)*sum(xx.*exp(-j*nn*w*T));


end
```

## A.3 Function $U^{'}(\omega)$

```
function da = dA_fun( N,xx,k,nfft,nn,fs )
% Input Paramter:
% N: length of signal
% xx: input signal
% k: index of FFT
% nfft: number of points of FFT
% nn: discrete index
% fs: sampling frequency

% Output Parameter:
% da: A'(k)

w = 2*pi*k*fs/nfft;

T = 1/fs;

n = nn;
sine = sin(T*n*w);
cose = cos(T*n*w);

x = real(xx);
y = imag(xx);

B = sum(x.*cose+y.*sine);
B = B/N;
C = sum(x.*sine - y.*cose);
C = -C/N;

dB = sum(n.*(y.*cose - x.*sine));
dB = dB*T/N;
dC = sum(n.*(x.*cose + y.*sine));
dC = -T/N*dC;

da = 2*B*dB+2*C*dC;


end
```

# B Real Signal Estimation Function

## B.1 Main Function

```
function [ahat,omegahat,thetahat,counter] = ML_fun(N,s,nfft,fs,el,sindex,eindex)

% Input Paramter:
% N: length of signal
% s: input signal
% nfft: number of points of FFT
% fs: sampling frequency
% el: the error bound
% sindex: start index of signal
% eindex: end index of signal

% Output Parameter
% ahat: amplitude estimate
% omegahat: frequency estimate (in rad/s)
% thetahat: phase estimate
% counter: iteration steps

A = fft(s,nfft)/N;
[junk,index] = max(abs(A(1:nfft/2)));
n = sindex:eindex;

x0 = (index-2);
x1 = index;
y0 = dA_fun_real(N,s,x0,nfft,n,fs);
y1 = dA_fun_real(N,s,x1,nfft,n,fs);

if abs(A_fun_real(N,s,x0,nfft,n,fs)) > abs(A_fun_real(N,s,x1,nfft,n,fs))
        temp = x0;
        x0 = x1;
        x1 = temp;
        temp = y0;
        y0 = y1;
        y1 = temp;
end

counter = 0;
delta = 1;
peak = abs(A_fun_real(N,s,index-1,nfft,n,fs));
while abs(delta) > el
        delta = y1*(x1-x0)/(y1-y0);
        x0 = x1;
        x1 = x1-delta;
        y0 = y1;
        y1 = dA_fun_real(N,s,x1,nfft,n,fs);
        counter = counter+1;
        dd = peak - abs(A_fun_real(N,s,x1,nfft,n,fs));

        if (counter > 1000)||(abs(dd)>0.1)
                aa = index-2;
                bb = index;
                while dA_fun_real(N,s,aa,nfft,n,fs)*dA_fun_real(N,s,bb,nfft,n,fs) >=
                        if dA_fun_real(N,s,aa,nfft,n,fs)<0
```

```
                                aa = aa-1;
                        else if dA_fun_real(N,s,bb,nfft,n,fs) > 0
                                        bb = bb+1;
                                 end
                        end
                end
                if dA_fun_real(N,s,aa,nfft,n,fs) <= 0
                        lo = aa;
                        hi = bb;
                else
                        lo = bb;
                        hi = aa;
                end

                mid = lo+(hi-lo)/2;
                counter = 0;
                while (mid ~= lo) && (mid ~= hi)
                        if dA_fun_real(N,s,mid,nfft,n,fs) <=0
                                lo = mid;
                        else
                                hi = mid;
                        end
                        delta = (hi-lo)/2;
                        mid = lo + delta;
                        if abs(delta) < el
                                break;
                        end
                        counter = counter+1;
                end
                x1 = mid;
                break;
        else continue;
        end
end

omegahat = x1/nfft*fs*2*pi;
cose = cos(omegahat*n(:)/fs);
sine = sin(omegahat*n(:)/fs);
cc = cose'*s(:);
ss = sine'*s(:);
Ha = cose'*cose;
Hb = cose'*sine;
Hc = sine'*cose;
Hd = sine'*sine;
alpha = [Hd -Hb; -Hc Ha]*[cc;ss]./(Ha*Hd-Hb*Hc);
thetahat = atan2(-alpha(2),alpha(1));
ahat = sqrt(alpha(1)^2+alpha(2)^2);
```

## B.2 Function $R(\omega)$

```
function value = A_fun_real(N,xx,k,nfft,nn,fs)

% Input Paramter:
% N: length of signal
% xx: input signal
% k: index of FFT
% nfft: number of points of FFT
% nn: discrete index
% fs: sampling frequency

% Output Parameter:
% value: A(k)

ww = 2*pi*k*fs/nfft;

n = nn';
x = xx';
T = 1/fs;
w = ww*T;

% these are the cos/sin vectors just below (7.63) in Steven Kay
c = cos(w*n);
s = sin(w*n);

% compute (7.65) from Steven Kay
a = c'*x;
b = s'*x;
A = c'*c;
B = c'*s;
C = s'*c;
D = s'*s;

in = [D -B; -C A]./(A*D-B*C);

value = [a;b]'*in*[a;b];
```

## B.3 Function $R^{'}(\omega)$

```
function value = dA_fun_real( N,xx,k,nfft,nn,fs )
% Input Paramter:
% N: length of signal
% xx: input signal
% k: index of FFT
% nfft: number of points of FFT
% nn: discrete index
% fs: sampling frequency

% Output Parameter:
% value: A'(k)

ww = 2*pi*k*fs/nfft;

T = 1/fs;
w2 = 2*ww*T;
w = ww*T;
n = nn';
x = xx';

c = cos(w*n);
c2 = cos(w2*n);
s = sin(w*n);
s2 = sin(w2*n);

dA = [sum((-1)*s.*x.*n*T); sum(c.*x.*n*T)];
dB = [sum((-1)*s2.*n*T) sum(c2.*n*T); sum(c2.*n*T) sum(s2.*n*T)];
dC = dA;
dA = dC';

p = c'*x;
q = s'*x;

A = [p;q]';
C = [p;q];

a = c'*c;
b = c'*s;
%c = s'*c;
c = b;
d = s'*s;

%B = [a b; c d];
B_i = [d -b; -c a]./(a*d-b*c);

value = dA*B_i*C+A*((-1)*B_i*dB*B_i)*C + A*B_i*dC;

end
```