DFT-Based High Resolution Frequency Estimation Using Three Samples

Şadi Çetinkaya

Submitted to the
Institute of Graduate Studies and Research
in Partial fulfillment of the requirement for the Degree of

Master of Science
in
Electrical and Electronic Engineering

Eastern Mediterranean University
May 2013
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Electrical and Electronic Engineering.

Prof. Dr. Aykut Hocanın
Chair, Electrical and Electronic Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Electrical and Electronic Engineering.

Prof. Dr. Osman Kükrer
Supervisor

Examining Committee

1. Prof. Dr. Aykut Hocanın _____

2. Prof. Dr. Osman Kükrer _____

3. Prof. Dr. Hüseyin Özkaramanlı _____

ABSTRACT

Estimation of the parameters of a complex sinusoid in noise usually consists of two steps; a coarse frequency estimation found by applying the N-point DFT of an N length input. Then a search method is applied around the peak frequency.

Different search methods can be applied around the peak frequency. In this thesis, we try to compare the method proposed by Candan with the Jacobsen, Macleod and Quinn's estimators. The performance measure of these algorithms will be compared in terms of the Cramer-Rao lower bound.

Different experiments have been implemented with different number of observations. Simulations show that as the number of observations becomes larger, these methods converge to the Cramer-Rao lower bound expressed as RMS error. However, Candan's method has shown the best performance among all other algorithms.

Keywords: Frequency Estimation, DFT, Jacobsen Estimator, Quinn Estimator, Macleod Estimator, Cramer-Rao Lower Bound.

ÖZ

Bir gürültülü karmaşık üstelin parametre kestirimi genellikle iki adımdan oluşur: Uzunluğu N olan bir veriye N-noktalı bir DFT uygulanmak suretiyle yaklaşık sıklık kestirimi, daha sonra ise bu sıklık etrafında bir arama yönteminin uygulanması.

Yaklaşık kestirilmiş sıklık etrafındaki arama için farklı yöntemler uygulanabilir. Bu tezde, Candan tarafından önerilmiş yöntemin, Jacobsen, Macleod ve Quinn'in kestirim yöntemleri ile karşılaştırılmasına çalışılmıştır. Bu algoritmaların karşılaştırılması için Cramer-Rao alt sınırı başarım ölçütü olarak alınmıştır.

Farklı sayıda gözlemler ile birçok deney yapılmıştır. Benzetim çalışmaları, bu yöntemlerin, gözlem sayısı arttıkça MSE'nin karekökü cinsinden ifade edilen Cramer-Rao alt sınırına yakınsadığını göstermiştir. Candan'ın yöntemi, diğer algoritmalara göre en iyi başarımı göstermiştir.

Anahtar kelimeler: Sıklık kestirimi, DFT, Jacobsen kestirimi, Quinn Kestirimi, Macleod kestirimi, Cramer-Rao alt sınırı.

To My Family

ACKNOWLEDGMENTS

I would like to greatly thank my supervisor Prof. Dr. Osman Kükrer for his support, guidance, help, knowledge and huge confidence that he provided to me.

Also, my regards to my friends who supported and helped me.

Finally, words will not be enough to thank my family for their infinite support and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ ABBREVIATIONS

N	Number of samples in a complex exponential
P_n	Noise Power
P_s	Signal Power
R[k]	*N*- point DFT of $r(n)$
δ	The change in the frequency
ω	The real-valued frequency in radians per sample
$\tilde{\omega}(k)$	The DFT of $\omega(n)$
σ^2	Variance
AWGN	Additive White Gaussian Noise
CRB	Cramer-Rao Lower Bound
dB	Decibel
DFT	Discrete Fourier Transform
DTFT	Discrete-Time Fourier Transform
H.O.T.	Higher Order Terms
i.i.d.	Identically Independent Distributions
RMSE	Root-Mean-Square-Error
SNR	Signal-To-Noise Ratio

Chapter 1

INTRODUCTION

1.1 Introduction

Estimation of the parameters of a complex sinusoid corrupted by white noise is an important problem in signal processing, and has applications in many areas such as power spectrum estimation, array and radar signal processing. The requirements regarding computational load of the estimator are particularly crucial in radar signal processing where a very large number of hypothesis tests on received complex signals are carried out per second, [1].

There are many fields that require high accuracy of frequency estimation, especially in engineering and scientific applications. Many research subjects and theories concentrate on this estimation. The frequency estimation problem is presented in the literature in two categories. One of them is the frequency-domain method (non-parametric methods). This category includes the Fourier-Transform based methods such as the periodogram and correlogram methods which depend on the Fourier Transform. Time-domain methods (parametric methods), on the other hand, have high performance in accuracy of estimation when the model order is well-known [2], [3]. Usually a DFT of data having length N, with a frequency resolution of $2\pi/N$ in the estimate may be required in many applications to increase the frequency

estimate's resolution at the expense of increasing the computational load. A two-step search is described in [4] aimed at improving the accuracy of the frequency estimate. In the first step, approximate search is performed by computing the N-point DFT. Then, a refined search in the neighborhood of this coarse estimate is performed in order to improve the estimate. The resolution of this two-step search is limited to the size of the grid used in the fine search. In [7], [8], [10], [11], an alternative strategy for the second step is proposed. Instead of searching on a fine grid, the high resolution estimate is obtained from a function in terms of the DFT samples computed in the first step. The methods proposed in [7], [8], [10] employ three DFT coefficients, while the method of Provencher uses only two DFT samples, [11]. These methods require much less computation than the fine grid search. Moreover, they yield a high resolution estimate instead of the limited resolution of the grid used. Jacobsen has proposed a simple expression for fine frequency estimation in the DFT domain. This expression is presented without any proof and is based on empirical observations. In this thesis, we review the derivation of Jacobsen's formula and justify Candan's proposal for bias correction. It is found that the correction is particularly effective when SNR is high. Nevertheless, it requires minimal increase in computational cost, hence can be used at all SNR values.

In this thesis, performance comparisons between the most well-known parametric methods will be done in terms of bias, variance and root-mean-square-error (RMSE).

Chapter 2

FREQUENCY ESTIMATION

2.1 Problem Description

Estimation of the frequency of a single tone sinusoid in a noisy environment is becoming a necessity in many applications. Usually this is done by estimating the Fourier transform of the collected samples. However, the use of Fourier transform methods is an obstacle because of the high computational cost of these methods. Different methods that avoid using Fourier transform and, at the same time, provide a good estimation of the frequency of the signal of interest have been proposed. Some of the well-known techniques will be discussed in the following chapter.

A single complex sinusoid signal embedded in AWGN (additive white Gaussian noise), is written as

$$r[n] = Ae^{j\omega n} + w[n] \qquad (2.1)$$

In (2.1), A is the complex-valued amplitude and ω is the discrete-time frequency in radians/sample.

The magnitude spectrum of $r[n]$ for the noise-free case is shown in Fig 2.1. The frequency of the complex exponential given as $\omega = 2\pi(k+\delta)/N$ is also shown there.

Estimation of δ, where $|\delta| < 1/2$, using three samples including the DFT coefficient with maximum magnitude of the DFT spectrum becomes our goal in this thesis. In the first stage, the DFT of the data vector $r[n]$ of length N is computed as follows,

$$R[k] = \sum_{n=0}^{N-1} r[n] e^{-j2\pi/Nnk}$$

Here $R[k]$ is the DFT coefficient which is complex in general. The expectation of the maximum value in the DFT magnitudes (k in Fig. 2.1) is in the vicinity of the true frequency ω, if the input $SNR = \frac{|A|^2}{\sigma_w^2}$ is sufficiently large. Estimation of the complex signal's frequency for sufficiently high signal-to-noise ratios (SNR's) is our interest in this thesis. This is important in radar signal processing applications. The DFT bin can be defined where the peak resides. The DFT coefficients to the left and right of this peak are:

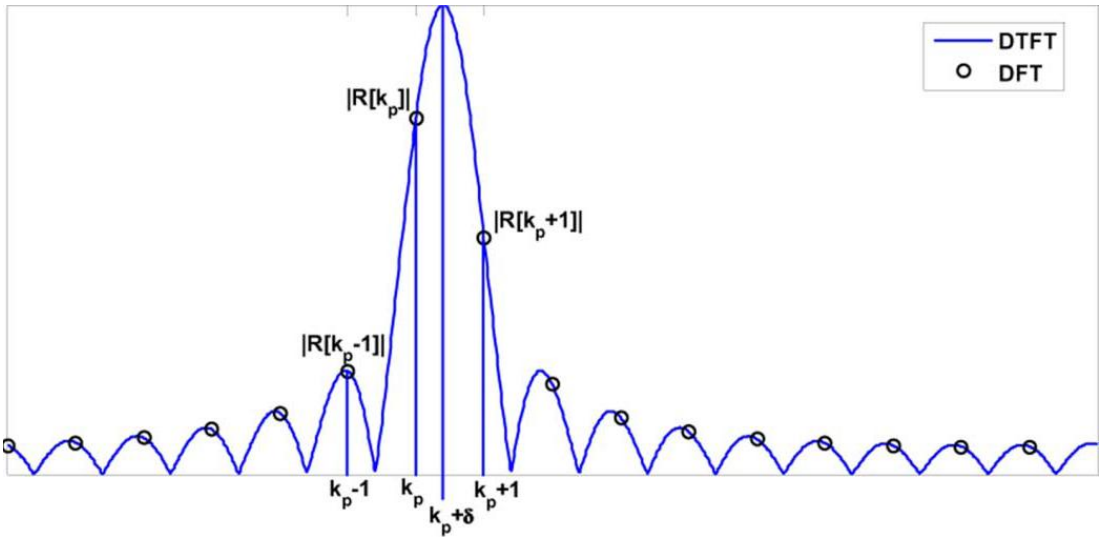

Fig 2.1: Magnitude spectrum of the complex exponential waveform with frequency $\omega = 2\pi(k+\delta)/N$ radians per sample, [6].

$$R[k-1] = A\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(\delta+1)n} + \tilde{\omega}[k-1] = Af(\delta+1) + \tilde{\omega}[k-1] \tag{2.2}$$

$$R[k] = A\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}\delta n} + \tilde{\omega}[k] = Af(\delta) + \tilde{\omega}[k] \tag{2.3}$$

$$R[k+1] = A\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(\delta-1)n} + \tilde{\omega}[k+1] = Af(\delta-1) + \tilde{\omega}[k+1] \tag{2.4}$$

Here $\tilde{w}[k]$ is the DFT of $w[n]$ which is also white and jointly Gaussian distributed. The function $f(.)$ in equations (2.2), (2.3) and (2.4) is given as:

$$f(\alpha) = \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}\alpha n} \tag{2.5}$$

where α is an arbitrary argument for $f(.)$.

Estimation of δ by utilizing three DFT coefficients in the vicinity of the peak is our goal. When the estimate $\hat{\delta}$ is determined, the resulting frequency estimate is obtained as $\hat{\omega} = 2\pi(k+\hat{\delta})/N$, where k is the bin index of the DFT peak evaluated in the approximate search step and $\hat{\delta}$ is the estimate produced by $R[k\text{-}1]$, $R[k]$ and $R[k\text{+}1]$.

Chapter 3

FREQUENCY ESTIMATORS

3.1 Introduction

In this chapter, the frequency estimators which are considered to be among the most important ones in the literature are presented and discussed. These methods are Quinn's, MacLeod's and Candan's estimators. A common feature of these methods is that they are all based on estimating frequency using three DFT samples, which are the one with maximum magnitude and those on the two sides of this maximum. The mathematical expressions employed in the estimators are given and the basic procedures in their algorithms are outlined.

3.2 Quinn Estimator

In this method, three DFT samples around the output peak are used for the estimation of the frequency of a complex exponential. This method does not use the method of periodogram to estimate the frequency, but instead it uses the complex DFT coefficients of data. This operation needs extra cost for computations in comparison with other methods. Quinn's method [10] can be summarized as:

$$\alpha_1 = real\left(\frac{R[k-1]}{R[k]}\right)$$
$$\alpha_2 = real\left(\frac{R[k+1]}{R[k]}\right)$$
$$\delta_1 = \frac{\alpha_1}{1-\alpha_1} \quad \text{and} \quad \delta_2 = \frac{-\alpha_2}{1-\alpha_2} \tag{3.1}$$
$$\text{if } \delta_1 > 0 \text{ and } \delta_2 > 0, \quad \hat{\delta} = \delta_2$$
$$\text{else} \quad \hat{\delta} = \delta_1$$

In Quinn's method, the noise is not assumed to be i.i.d Gaussian, but the data is assumed to be strictly stationary and ergodic, which are very weak assumptions.

3.3 MacLeod Estimator

This method provides more accurate computation of the frequency estimation in terms of bias and variance of estimation. It is applicable to single or multi-sinusoid complex signals.The basic mathematical model of this method [7] is given by:

$$d = \frac{real\left(R[k-1]R^*[k] - R[k+1]R^*[k]\right)}{real\left(2\left|R[k]\right|^2 + R[k-1]R^*[k] + R[k+1]R^*[k]\right)}$$
$$\hat{\delta} = \left(\frac{\sqrt{1+8d^2}-1}{4d}\right) \tag{3.2}$$

3.4 Jacobsen Estimator

This method achieves fast and accurate estimation by the Discrete Fourier Transform (DFT). The idea of the estimation in this method is based on three samples of DFT and the peak of the frequency that will be estimated [5]. It basically uses the complex DFT values rather than the magnitudes.

$$\hat{\delta} = real\left(\frac{R[k-1]-R[k+1]}{2R[k]-R[k-1]-R[k+1]}\right) \tag{3.3}$$

Equation (3.3) provides potential for reduction in computations by avoiding the nontrivial magnitude calculations in (3.2).

3.4 Candan Estimator

This method is a recently introduced novel way of frequency estimation. The novelty is that it combines Jacobsen's formula and bias correction. In this way, we can get high performance for bias correction for large SNR with little extra cost of computation. Thus, at any SNR level it can be used [6].

In order to be able to express Jacobsen's estimator in terms of δ, $f(\delta)$ in (2.5) is expanded in a Taylor series around δ=0. For small δ, the terms involving powers of δ higher then two can be neglected to enable solution. For the derivation of the estimation equation consider the expansion of $g(z)=\sum_{n=0}^{N-1} e^{zn}$ as a Taylor series around z=0:

$$\begin{aligned} g(z)=\sum_{n=0}^{N-1} e^{zn} &= \sum_{n=0}^{N-1}\sum_{k=0}^{\infty}\frac{(zn)^k}{k!} \\ &= \sum_{k=0}^{\infty}\frac{z^k}{k!}\sum_{n=0}^{N-1} n^k \\ &= \sum_{k=0}^{\infty}\frac{z^k}{k!}S_k(N-1) \end{aligned} \qquad (3.4)$$

The integer power summations (k^{th}) between 0 and N-1 can be denoted by $S_k(N\text{-}1) = 0^k + 1^k + \cdots\cdots + (N\text{-}1)^k$. The sequence $S_k(N)$ for k =0, 1, 2 can be written as follows:

$$\begin{aligned} S_0(N) &= N+1 \\ S_1(N) &= \frac{N(N+1)}{2} \\ S_2(N) &= \frac{N(N+1)(2N+1)}{6} \end{aligned} \qquad (3.5)$$

Higher orders for $S_k(N)$ can been written in terms of Bernoulli numbers [12]. Separating the summation in (3.4) for even and odd values of k, the $f(\alpha)$ sequence becomes

$$f(\alpha)=\sum_{k=0}^{\infty}(-1)^{k}\frac{S_{2k}(N-1)(2\pi)^{2k}}{(2k)!N^{2k}}\alpha^{2k}+j\sum_{k=0}^{\infty}(-1)^{k}\frac{S_{2k+1}(N-1)(2\pi)^{2k+1}}{(2k+1)!N^{2k+1}}\alpha^{2k+1} \tag{3.5}$$

Defining c_k as:

$$c_k=\frac{S_k(N-1)(2\pi)^{k}}{(k)!N^{k}} \tag{3.6}$$

the Taylor series expansion of (3.5) can be written as follows:

$$\begin{aligned} f(\alpha)&=\sum_{k=0}^{\infty}c_k(j\alpha)^k \\ &=\left(c_0-c_2\alpha^2+c_4\alpha^4-\cdots\right) \\ &=j\left(c_1\alpha-c_3\alpha^3+c_5\alpha^5-\cdots\right) \end{aligned} \tag{3.7}$$

An estimator for δ can be constructed by evaluating $f(\alpha)$ at $\alpha=\delta+$p where p=–1,0 ,+1 and forming the following second-order differences:

$$\begin{aligned} &f(\delta+1)-f(\delta-1)= \\ &\qquad 2\left\{\left(-2c_2+4c_4-6c_6+\cdots\right)\delta+j\left(c_1-c_3+c_5-\cdots\right)\right\}+H.O.T. \\ &f(\delta+1)-2f(\delta)+f(\delta-1)= \\ &\qquad 2\left\{\left(-c_2+c_4-c_6+\cdots\right)+j\left(-3c_3+5c_5-7c_5+\cdots\right)\right\}+H.O.T. \end{aligned} \tag{3.8}$$

The abbreviation *H.O.T.* means the higher order terms of δ, and in the right-hand side of equation (3.8) we evaluate the infinite sum by odd and even indexed c_k terms as follows

$$\sum_{k=0}^{\infty} c_k j^k = f(1) = \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}n} = 0$$

$$\sum_{k=1}^{\infty} k c_k j^k = f'(1) = j\frac{2\pi}{N}\sum_{n=0}^{N-1} n e^{j\frac{2\pi}{N}n} \tag{3.9}$$

$$= \pi \cot\left(\frac{\pi}{N}\right) - j\pi.$$

Rewriting (3.9);

$$\begin{aligned} &\left(c_0 - c_2 + c_4 - c_6 + c_8 - \cdots\right) \\ &+ j\left(c_1 - c_3 + c_5 - \cdots\right) = 0 \\ &\left(-2c_2 + 4c_4 - 6c_6 + \cdots\right) \\ &+ j\left(c_1 - 3c_3 + 5c_5 - \cdots\right) = \pi \cot\left(\frac{\pi}{N}\right) - j\pi \end{aligned} \tag{3.10}$$

Using (3.6) for k =0, it can be shown that c_0=N and c_1= π(N-1). By substituting c_0 and c_1 into the equations given in (3.10), equation (3.8) can be simplified as:

$$f(\delta+1) - f(\delta-1) = 2\left\{\pi \cot\left(\frac{\pi}{N}\right)\delta\right\} + H.O.T.$$

$$f(\delta+1) - 2f(\delta) + f(\delta-1) = 2\left\{-N - j\pi N\delta\right\} + H.O.T. \tag{3.11}$$

In this procedure, by neglecting the higher order terms, the ratio of the differences is obtained as

$$\frac{f(\delta+1) - f(\delta-1)}{f(\delta+1) - 2f(\delta) + f(\delta-1)} = \frac{\pi \cot(\pi/N)\delta}{-N - j\pi N\delta}$$

$$= \frac{-\pi N \cot(\pi/N)}{N^2 + \pi^2 N^2 \delta^2}\delta + j\frac{\pi^2 N \cot(\pi/N)}{N^2 + \pi^2 N^2 \delta^2}\delta^2 \tag{3.12}$$

Because δ^2 is small compared to δ, (3.12) can be simplified as follows:

$$real\left\{\frac{f(\delta+1)-f(\delta-1)}{f(\delta+1)-2f(\delta)+f(\delta-1)}\right\}=\frac{-\pi\cot(\pi/N)}{N}\delta \tag{3.13}$$

When the signal-to-noise ratio (SNR) is large; that is when $|A|^2>>\sigma_w^2$, DFT samples around the peak value, that is $R[k\text{-}1]$, $R[k]$ and $R[k+1]$, in (2.2), (2.3) and (2.4), can be approximated by $Af(\delta+1)$, $Af(\delta)$ and $Af(\delta\text{-}1)$, respectively. Then an estimate for δ can be produced via the substitutions $f(\delta)\rightarrow R[k_p]$, $f(\delta-1)\rightarrow R[k_p+1]$, and $f(\delta+1)\rightarrow R[k_p\text{-}1]$ into (3.13):

$$\hat{\delta}=\frac{\tan(\pi/N)}{(\pi/N)}real\left(\frac{R[k-1]-R[k+1]}{2R[k]-R[k-1]-R[k+1]}\right) \tag{3.14}$$

Candan's estimator is based on equation (3.14) and it can be shown that as $N\rightarrow\infty$ this estimator converges to the Jacobsen estimator.

Chapter 4

EXPERIMENTAL RESULTS

4.1. Introduction

In this chapter, the performance of the frequency estimation algorithm proposed by Candan, discussed in detail in chapter 3, is investigated and compared with some well-known estimators presented in the literature.

MATLAB Software Package is used for implementing the various frequency estimators for simulations. The performances of the estimators of Candan, Quinn, Macleod, Jacobsen and the Cramer-Rao bound for frequency estimation variance (given by (4.1)) are compared and discussed in detail. A summary of the estimators is shown in Table 4.1. Given the data length N and signal-to-noise-ratio SNR, the Cramer-Rao lower bound for a complex sinusoid is given by [13]

$$CRB_w = \frac{6}{N(N^2-1)\times SNR}$$

where (4.1)

$$SNR = \frac{|A|^2}{\sigma_w^2}$$

Table4.1: DFT-based Fine Resolution Frequency Estimators.

Quinn	$\alpha_1 = real\left(\frac{R[k-1]}{R[k]}\right)$ $\alpha_2 = real\left(\frac{R[k+1]}{R[k]}\right)$ $\delta_1 = \frac{\alpha_1}{1-\alpha_1}$ and $\delta_2 = \frac{-\alpha_2}{1-\alpha_2}$ if $\delta_1 > 0$ and $\delta_2 > 0$, $\hat{\delta} = \delta_2$ else $\hat{\delta} = \delta_1$
MacLeod	$d = \frac{real\left(R[k-1]R^*[k] - R[k+1]R^*[k]\right)}{real\left(2\lvert R[k]\rvert^2 + R[k-1]R^*[k] + R[k+1]R^*[k]\right)}$ $\hat{\delta} = \left(\frac{\sqrt{1+8d^2}-1}{4d}\right)$
Jacobsen	$\hat{\delta} = real\left(\frac{R[k-1]-R[k+1]}{2R[k]-R[k-1]-R[k+1]}\right)$
Candan	$\hat{\delta} = \frac{\tan(\pi/N)}{(\pi/N)} real\left(\frac{R[k-1]-R[k+1]}{2R[k]-R[k-1]-R[k+1]}\right)$

The main Matlab functions of the methods listed in Table 4.1 are given in Appendix A.1.

In the first part of the simulations, the frequency of the signal described by the complex exponential waveform

$$x(n) = Ae^{j\omega n} + v(n) \qquad (4.2)$$

is being estimated, where $v(n)$ is a complex white Gaussian noise with zero mean and variance $\sigma^2 = 0.5$. Here we estimate δ and we denote the estimated value by $\hat{\delta}$ to distinguish it from the actual value. So the estimated frequency is given by:

$$\hat{\omega} = \frac{2\pi(k+\hat{\delta})}{N} \tag{4.3}$$

The signal data length is selected to be 64 ($N = 64$) and the simulations were done for 1000 independent trials ($M = 1000$) with signal-to-noise ratio (SNR= $10\log_{10}\left(\frac{P_s}{P_n}\right) = 15.6$ dB).

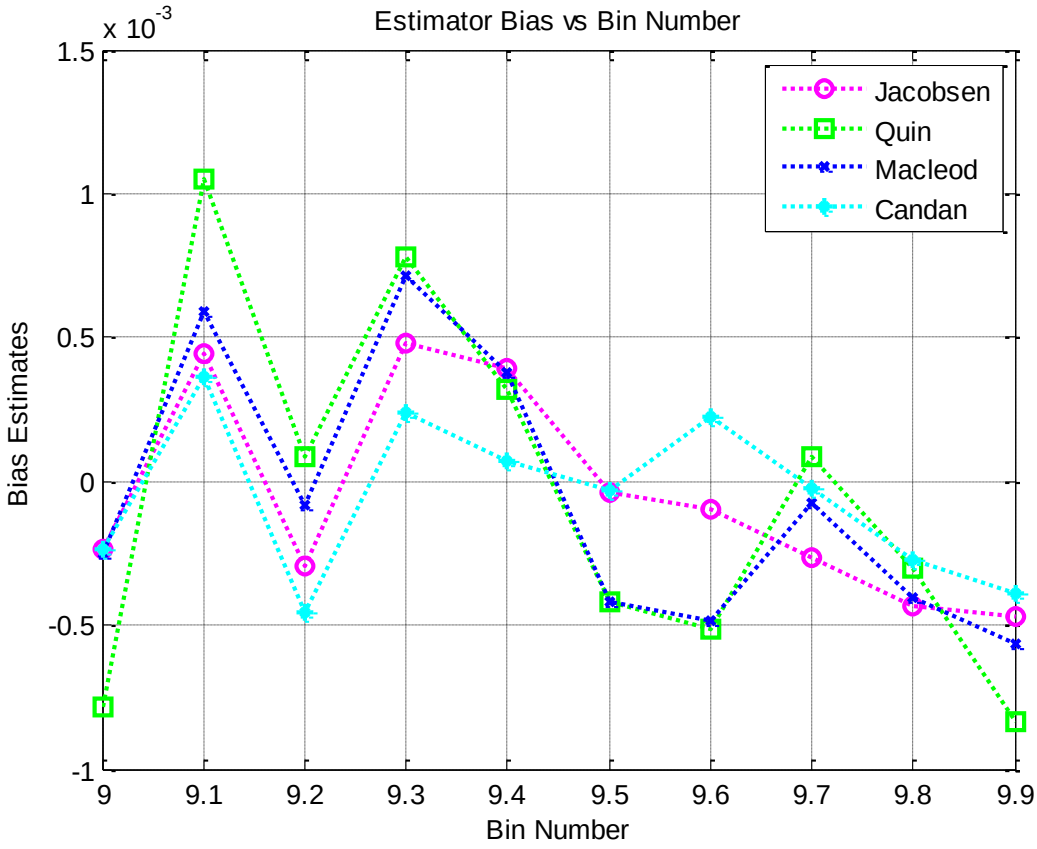

Fig. 4.1: Estimators Bias.

Fig. 4.1 shows that the method proposed by Candan [6] (a modified version of Jacobsen's method) provides almost the best estimation of δ with lower bias (in average vs. bin number) than the other methods (Quinn, Jacobsen and Macleod). The bias of the estimators is calculated using $\hat{\omega}_{bias} = \left(\frac{2\pi}{MN}\right)\sum_{i=1}^{M}\left(\delta - \hat{\delta}\right)$.

Another performance measure is the variance of the estimator, calculated using

$$\sigma_{\omega}^{2} = \frac{4\pi^{2}}{N^{2}}\left(\frac{1}{M}\right)\sum_{i=1}^{M}\left(\delta - \hat{\delta}\right)^{2}$$

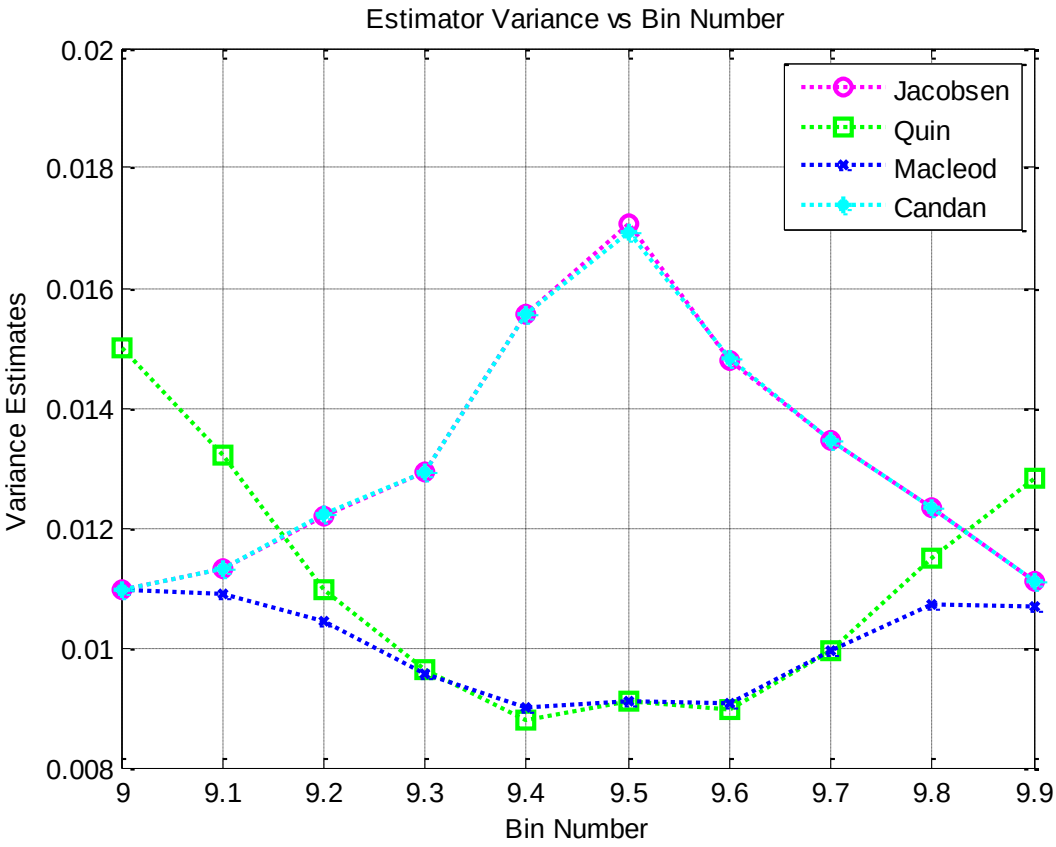

Fig. 4.2: Estimator Variance.

Fig. 4.2 shows that even though Candan's and Jacobsen's methods have higher variances than the others for bin numbers in the range 9.2.-9.8 their variances decrease when the bin number starts increasing.

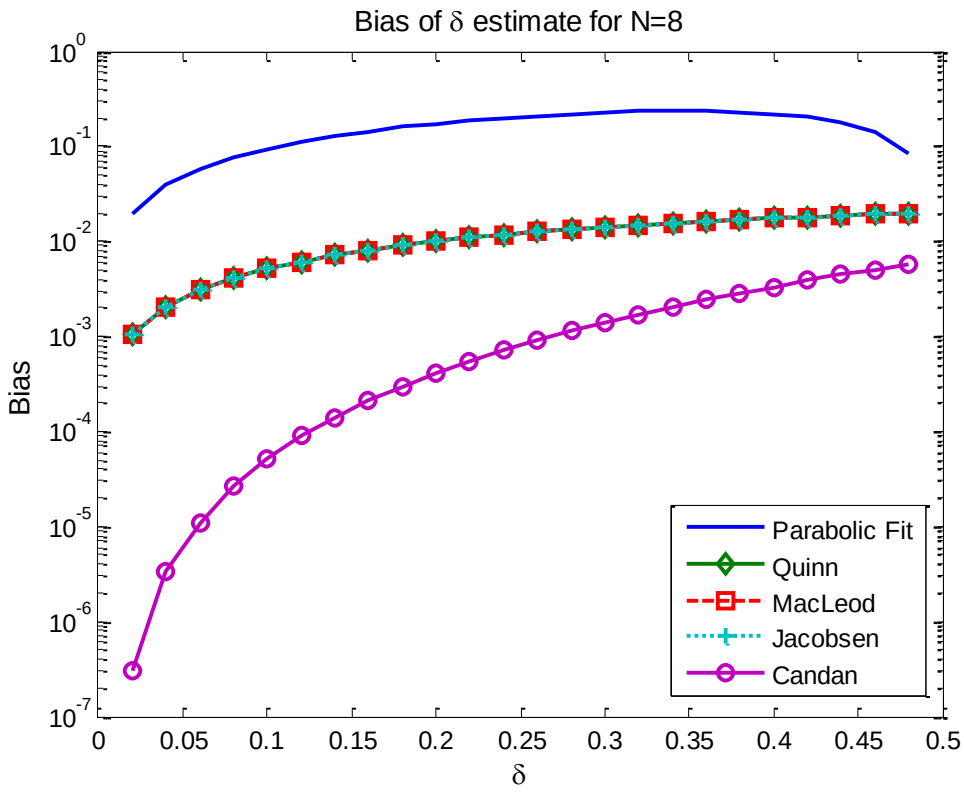

Fig. 4.3: Frequency bias of estimators with white Gauusian noise.

In Fig. 4.3, a comparison is made among the algorithms, given in Table 4.1, in terms of the bias estimate when N=8 and the given signal is without noise. Frequency estimation is a process which is nonlinear; hence it should be expected that all frequency estimators are inherently biased. The bias decreases as $N \to \infty$ or as the SNR increases. Fig. 4.3 shows that Candan's estimator has the lowest bias among the others.

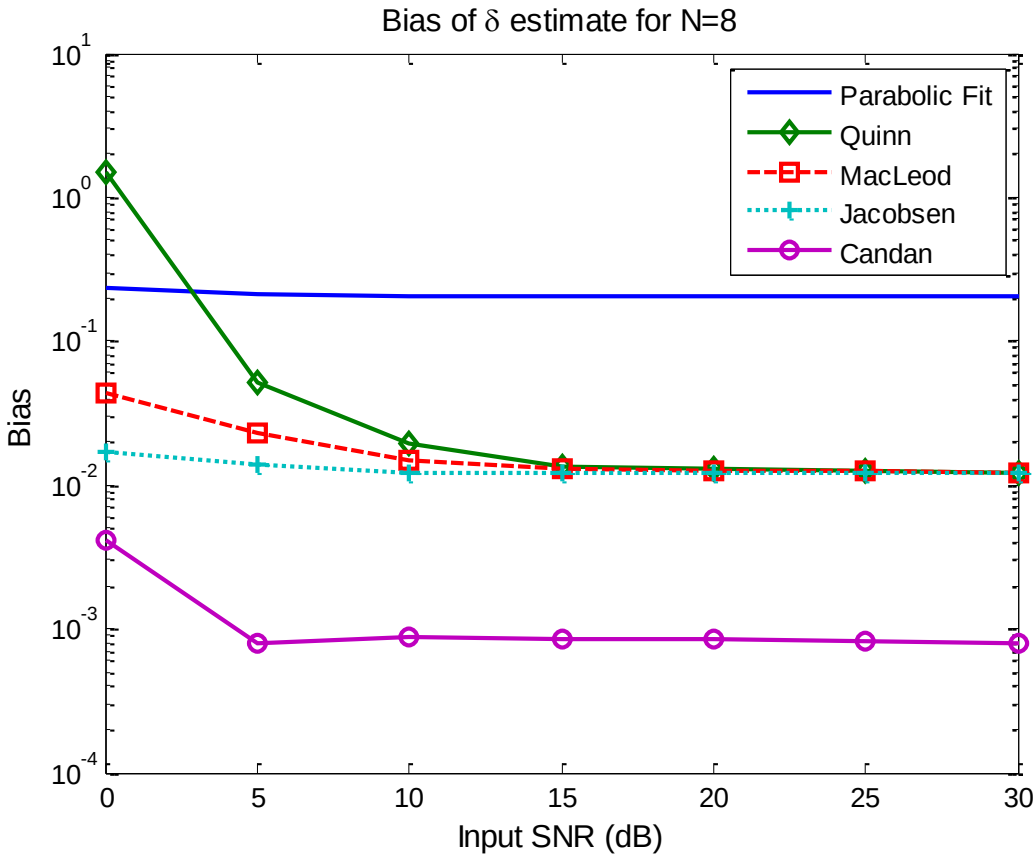

Fig. 4.4: Bias variation as SNR changes for $\delta = 0.25, N = 8$.

Fig.4.5 shows the effect of noise on the bias. For this computation, the parameter is fixed to the specific value, which is $\delta = 0.25$, and SNR is varied between 0 to 80 dB (in Fig. 4.4 we show 0-30 dB for zooming purpose). As can be observed from this figure, Candan's estimator provides the lowest bias value among the others.

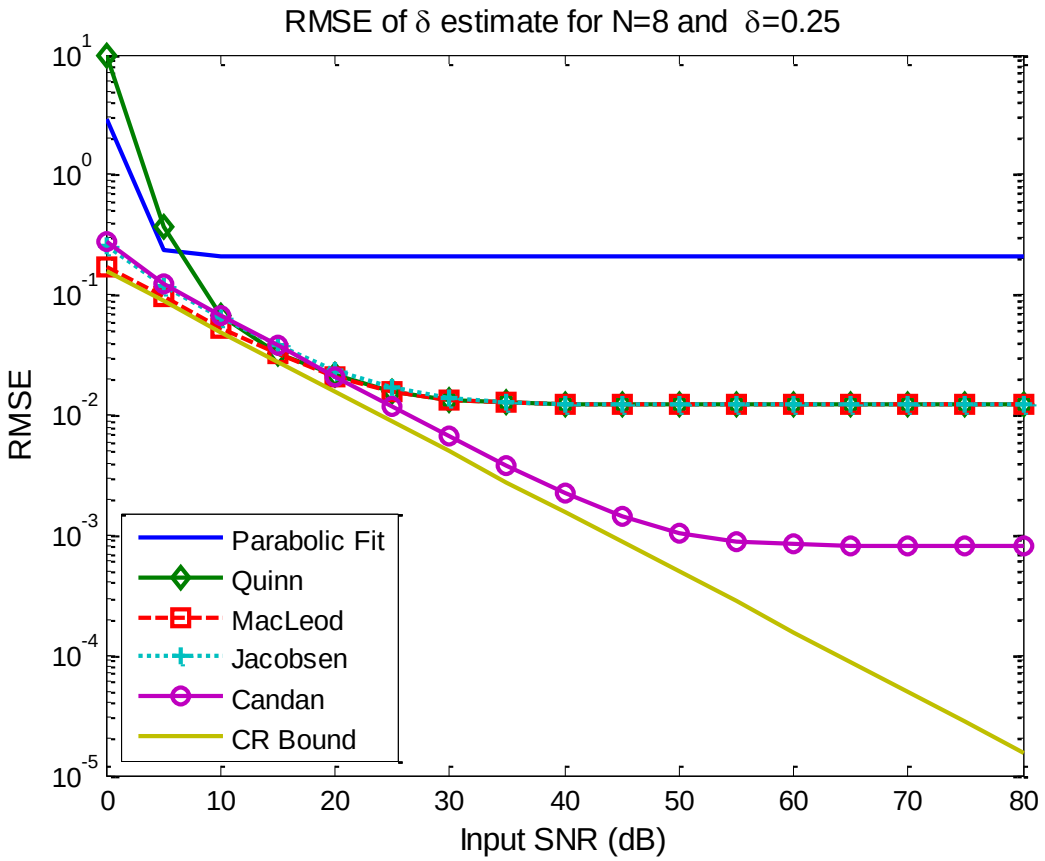

Fig. 4.5: RMSE variation as SNR changes for $\delta = 0.25$ and $N = 8$.

Fig. 4.5 shows the root-mean-square-error (RMSE) of the estimators and the Cramer–Rao lower bound which is given by (4.1). Here, it should be noted that the Cramer-Rao bound is not normally applicable for estimation algorithms which are biased. However, it is still a useful indicator of performance if the bias of the estimator is much smaller then the variance of the error. Indeed, it is clearly noticeable in this figure that, the estimator bias is dominant in the RMSEs produced by all the estimators, including Candan's estimator, when the SNR is sufficiently high.

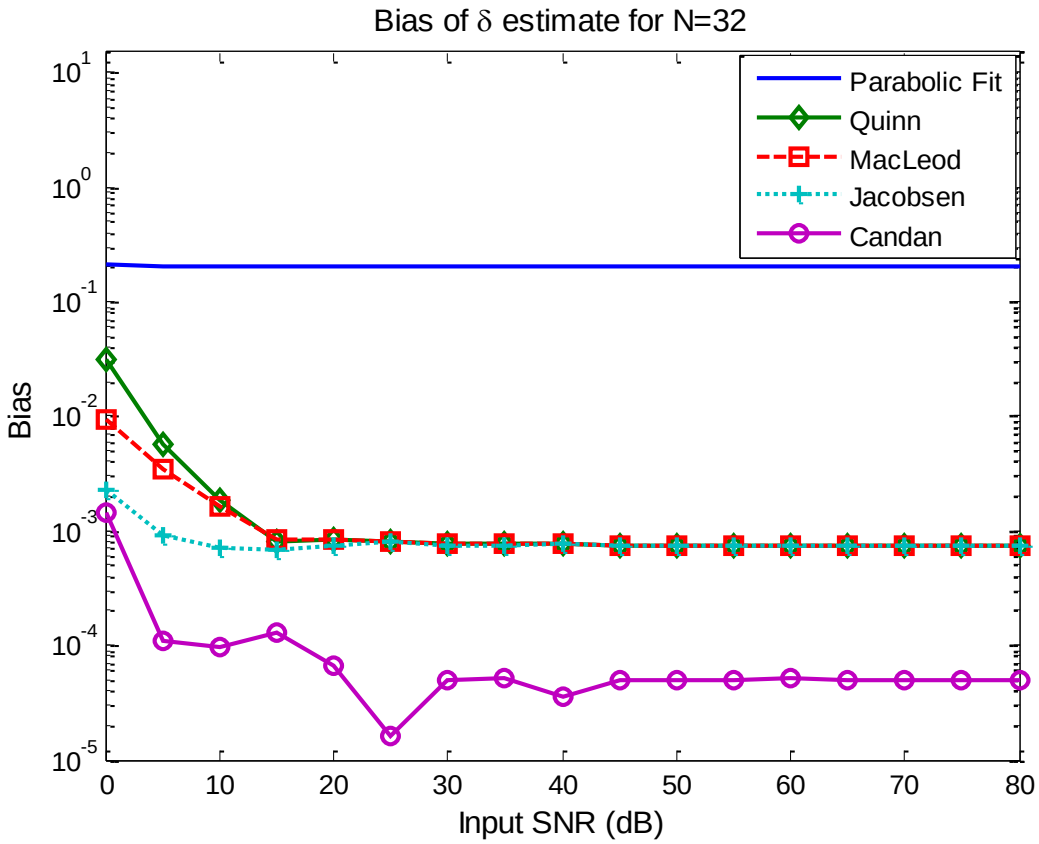

Fig. 4.6: Bias variation as SNR changes for $\delta = 0.25$ and $N = 32$.

In Figures 4.6 and 4.7, N is taken as N=32. Fig. 4.6 shows how bias is affected by noise present in the signal. In this figure, the frequency parameter delta is fixed to a predetermined value $\delta = 0.25$, and SNR is again varied between 0 to 80 dB. In this figure it can be observed that the variances of the estimators tend to the bias values produced in the absence of noise, as SNR increases, given in Fig. 4.3. However, Candan's estimator has the lowest bias among all estimators.

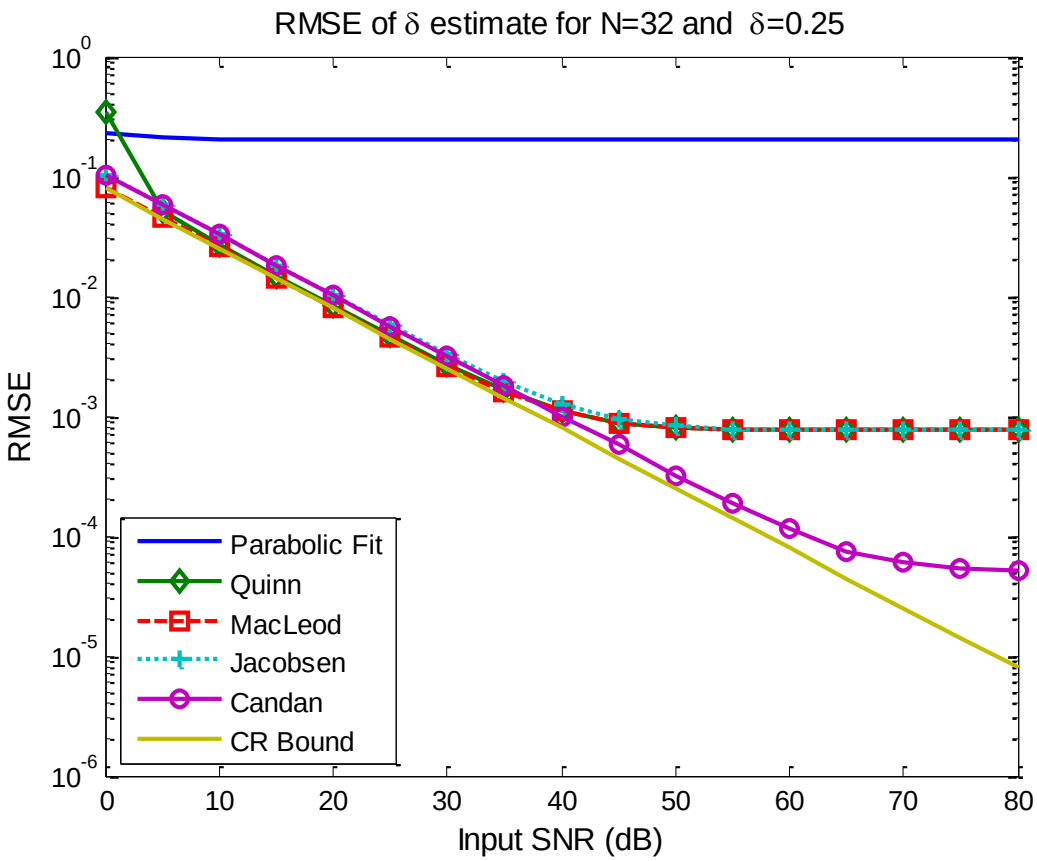

Fig. 4.7: RMSE variation as SNR changes for $\delta = 0.25$ and $N = 32$.

Fig. 4.7 depicts the comparison of RMSEs of the methods for *N*=32. It should be pointed out that, as the number of observations gets larger, the estimators' performance (especially Candan's one) gets closer to the Cramer-Rao lower bound.

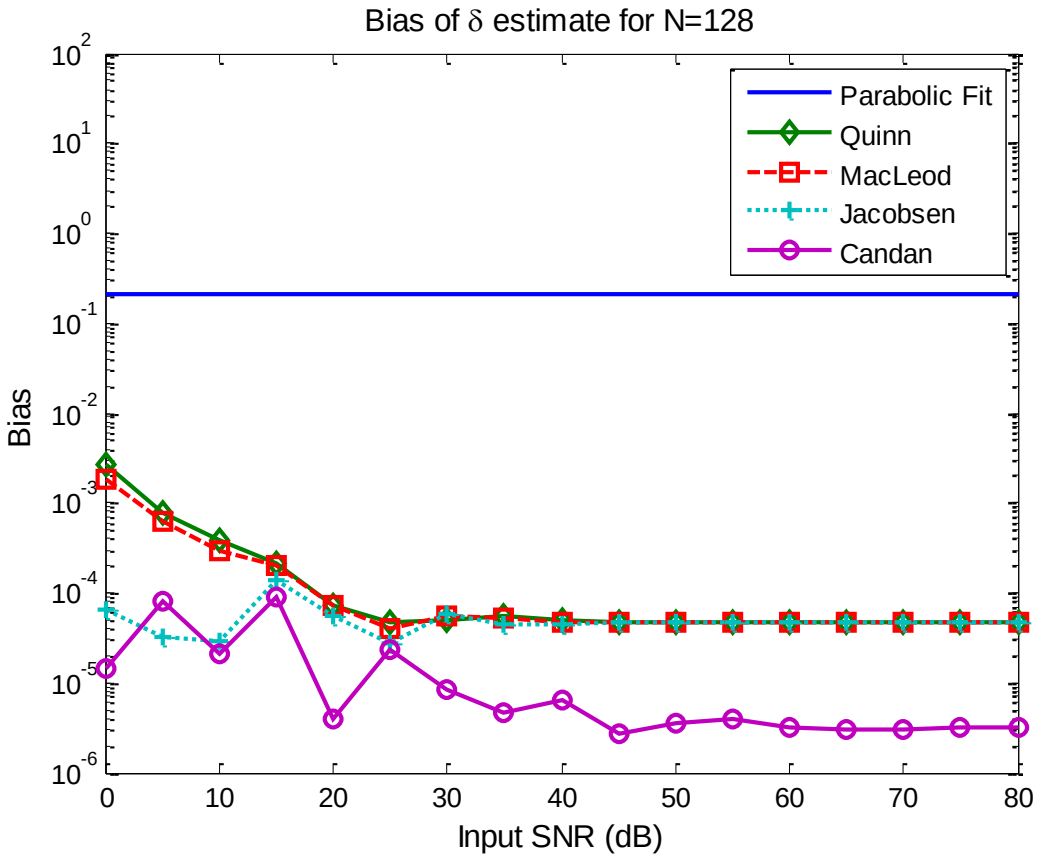

Fig. 4.8: Bias variation as SNR changes for $\delta = 0.25$ and $N = 128$.

In Figures 4.8 and 4.9, N is further increased to N=128. Fig. 4.8, again, shows the bias when the exponential is burried in noise. In this figure, the frequency parameter δ is fixed to the predetermined value, which is $\delta = 0.25$, and SNR is again varied between 0 to 80 dB. As the figure shows, the estimators can become close to the bias values in the absence of noise, as depicted in Fig. 4.3. Also, as before, Candan's estimator still has the lowest bias among all estimators.

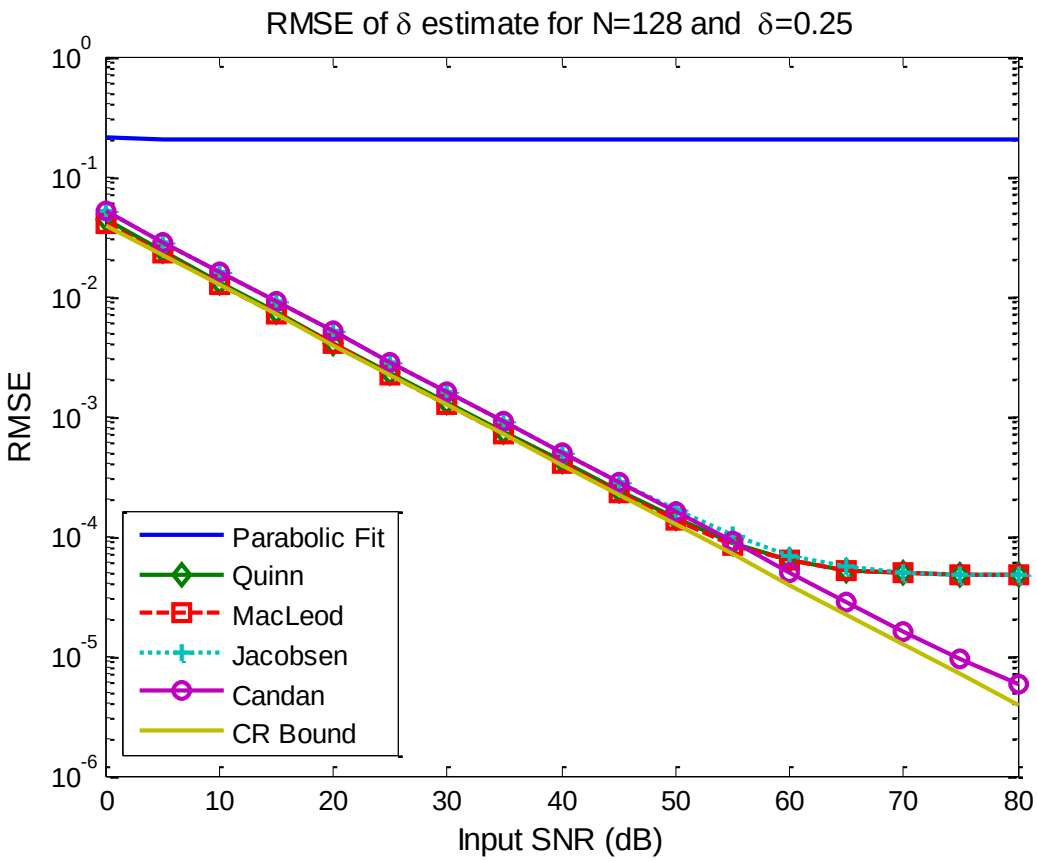

Fig. 4.9: RMSE variation as SNR changes for $\delta = 0.25$ and $N = 128$.

Fig. 4.9 displays the RMSE when the data length is increased to N=128. It should be stressed that, as the number of observations gets larger, the estimators' performance (especially Candan's one) gets closer to the Cramer-Rao lower bound. This, in turn, shows the convergence to the Cramer-Rao lower bound.

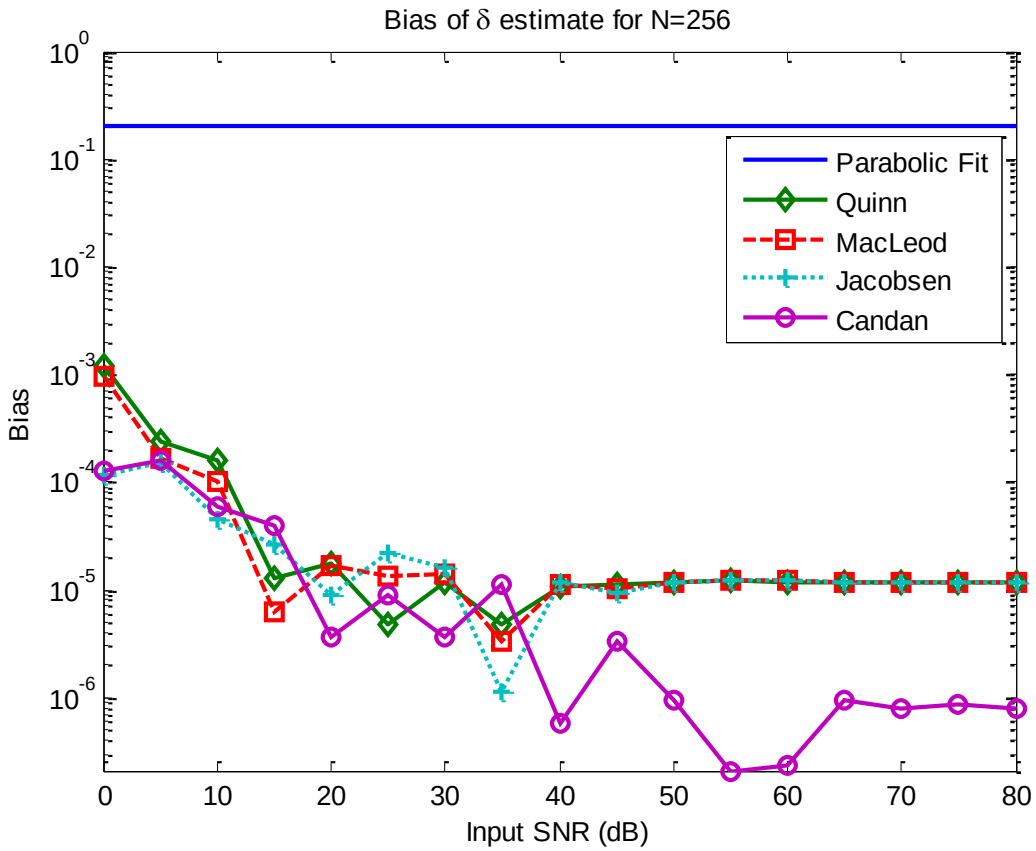

Fig. 4.10: Bias variation as SNR changes for $\delta = 0.25$ and $N = 256$.

In Figures 4.10 and 4.11, the number of observations is given as N=256. Fig. 4.10 shows how bias is affected by noise present in the signal. In this figure, the frequency parameter δ is fixed to the predetermined, which is $\delta = 0.25$, and SNR is again varied between 0 to 80 dB. As can be seen in this figure, the estimator RMSEs tend to the bias values for the noise free case (see Fig. 4.3.), as SNR increases. However, Candan's estimator again has the lowest bias among all estimators.

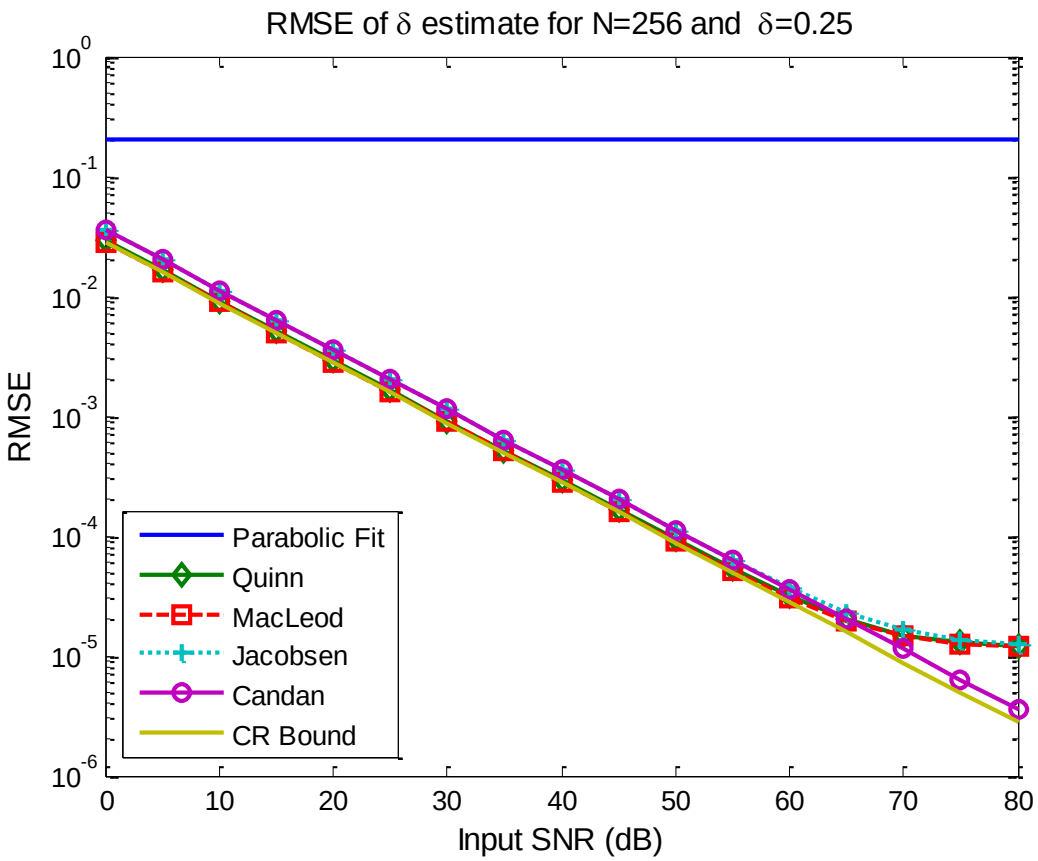

Fig. 4.11: RMSE variation as SNR changes for $\delta = 0.25$ and $N = 256$.

Fig. 4.11 shows the RMSE for N=256. In this figure, Candan's estimator approaches to the Cramer-Rao lower bound (only 1 dB difference).

In Figs 4.1-4.11, additive white Gaussian noise case is investigated. In order to show the effect of the noise type on the performance of the mentioned techniques, additive colored Gaussian noise (ACGN) case will be investigated. The colored Gaussian noise is created by filtering the white Gaussian noise using a 2-coefficient low-pass filter with impulse response $h = [0.707\ 0.707]$ and magnitude response shown in Fig. 4.12.

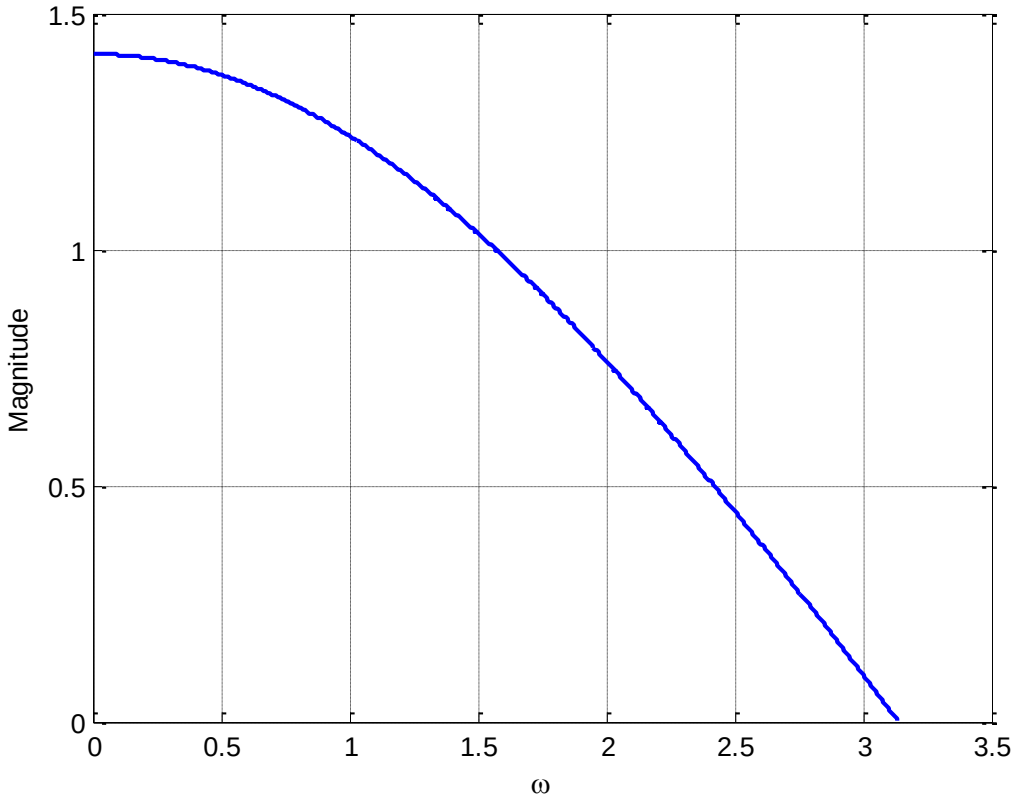

Fig. 4.12: The Magnitude response of the filter used in creating colored noise ($h = [0.707\ 0.707]$).

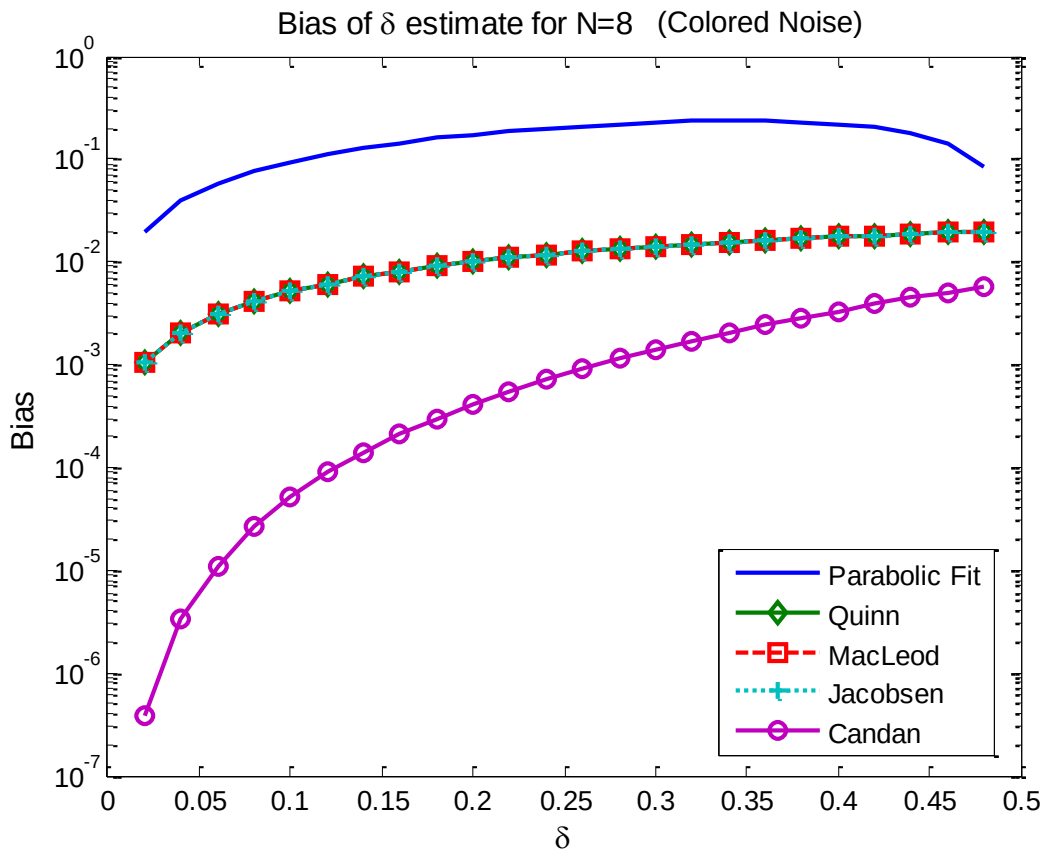

Fig. 4.13: Frequency bias of estimators with colored Gaussian noise.

In Fig. 4.13, we compare the performance of the algorithms under ACGN. The other parameters are the same as those in Fig. 4.3. From Fig. 4.13 we see that the performance of Candan's estimator is less than that of the AWGN case by 0.5 dB while it is less by 1 dB for the other estimators. This shows the advantage of Candan's over the others even in ACGN case.

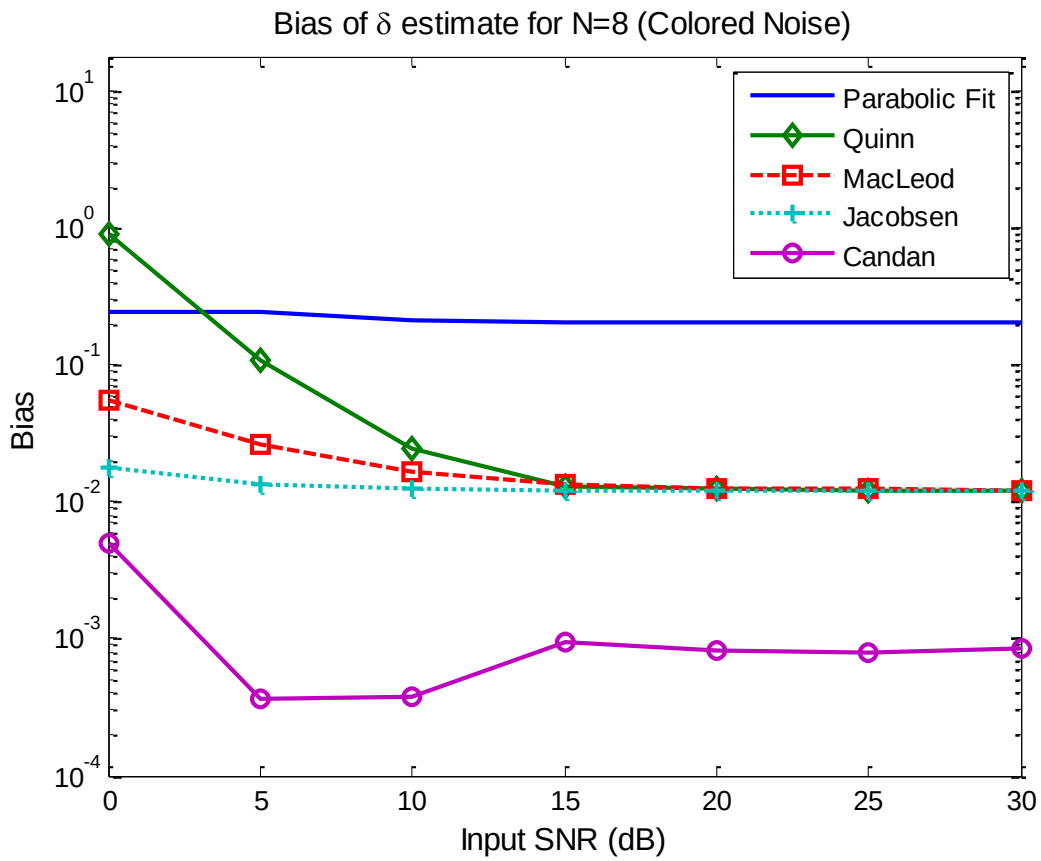

Fig. 4.14: Bias variation as SNR changes for $\delta = 0.25$ and $N = 8$ in colored Gaussian noise.

Figure 4.14 shows that none of the estimators bias is affected by the ACGN at high SNR, i.e., noise type does not affect the bias estimate. However, Candan estimator's performance becomes constant at SNR=15 dB while it became constant at SNR=5 dB in the case of AWGN.

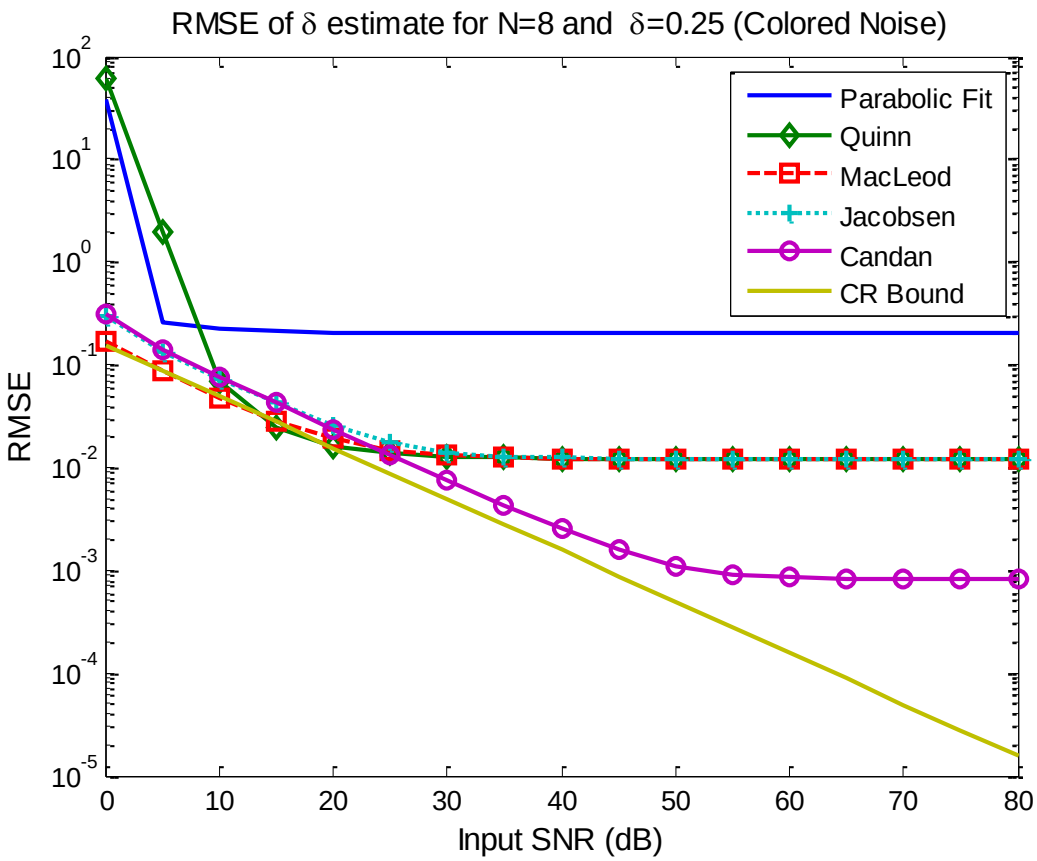

Fig. 4.15: RMSE variation as SNR changes for $\delta = 0.25$ and $N = 8$ in colored Gaussian noise.

Figure 4.15 shows that the estimators' RMSE performances are again not affected by the noise type when the signal length is short (i.e. N=8)

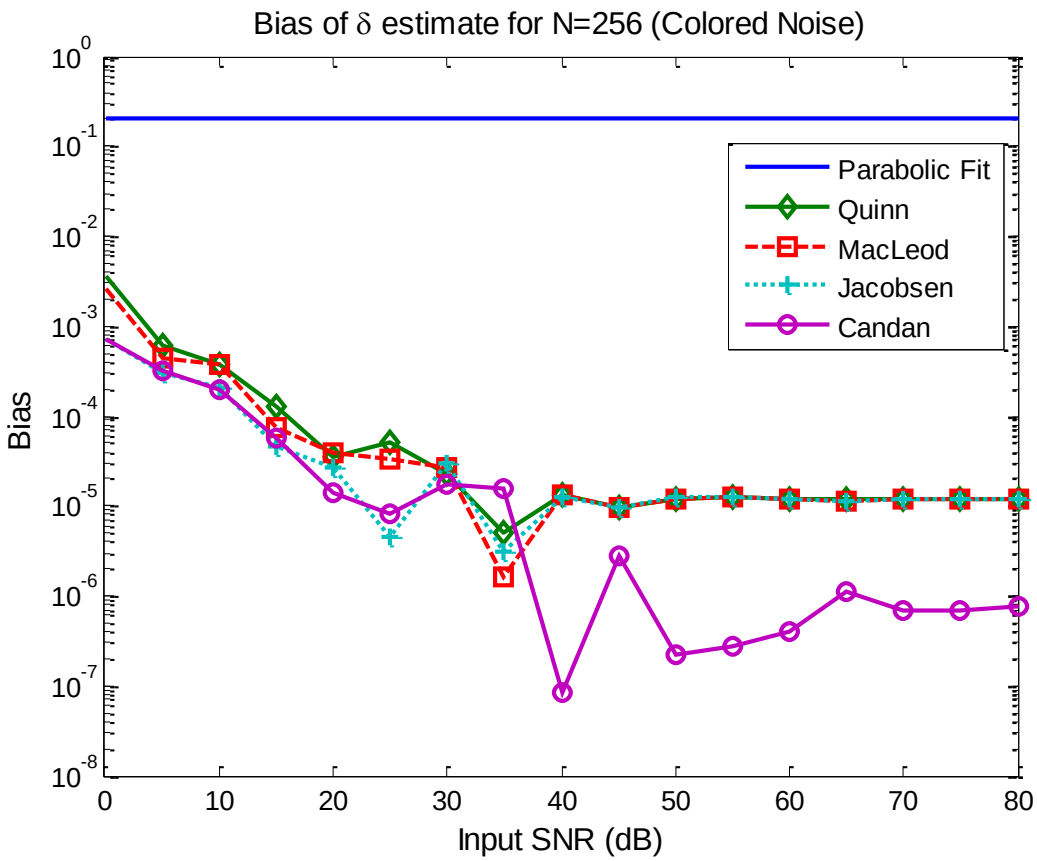

Fig. 4.16: Bias variation as SNR changes for $\delta = 0.25$ and $N = 256$ in colored Gaussian noise.

Figure 4.16 shows that the estimators' biases are the same as those of the AWGN case when the SNR is very high (i.e. SNR>40 dB). However, their biases are high by 4-5 dB at very low SNR's in the case of ACGN than that of the AWGN.

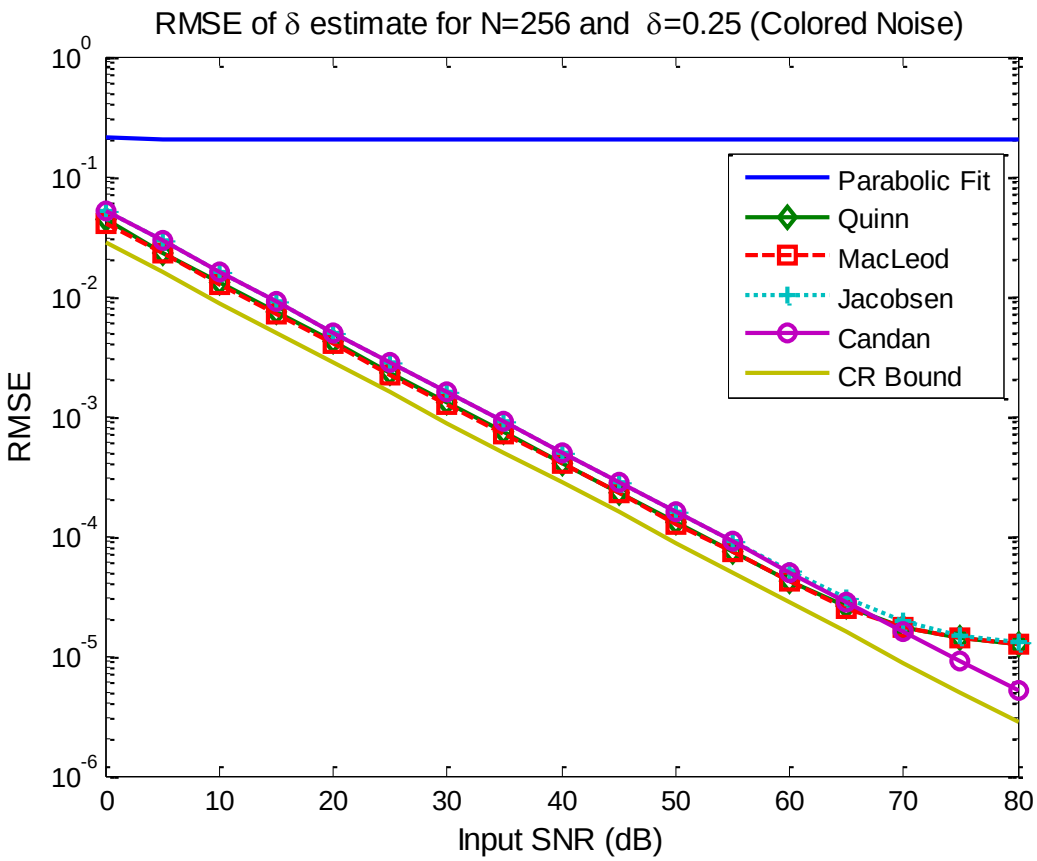

Fig. 4.17: RMSE variation as SNR changes for $\delta = 0.25$ and $N = 256$ in colored Gaussian noise.

From Fig. 4.17, we see that Candan's estimator is worse by 0.5 dB in ACGN case than that of the AWGN. However, in AWGN case Quinn and Macleod estimators were exactly reaching CR-lower bound when SNR<60 dB but in the ACGN case they perform worse by 1 dB.

Chapter 5

CONCLUSIONS AND FUTURE WORK

Conclusions

In this thesis, the performance of the frequency estimator proposed by Candan is compared with the estimators of Jacobsen, Macleod and Quinn.

The Candan estimator requires a small number of calculations for each estimate, and produces results with variances very close to the CR lower bound when the SNR is sufficiently high. These features render this estimator particularly suited to radar signal processing applications, where it is required to compute Doppler frequencies of unknown targets accurately and efficiently. The work presented in this thesis has confirmed the viability of this estimator.

Different experiments have been implemented with different numbers of observations. Simulations show that Candan's estimator performs much better than the other algorithms in terms of bias, variance and root-mean-square error (RMSE). Also, it has been shown that as the number of observations becomes large, these methods converge to the Cramer-Rao lower bound in terms of RMSE. However, Candan's method has shown the best performance among all the other algorithms.

Future Work

A possible future work could be the derivation of similar high resolution estimators by using alternative windows such as those of Hamming and Hanning.

Also, trying to improve the performance of the Candan estimator in order to reach the Cramer-Rao lower bound for short length data without increasing the number of computations would be a distinct potential future work.

REFERENCES

[1] M. A. Richards, *Fundamentals of Radar Signal Processing*, New York: McGraw-Hill, 2005.

[2] S. L. Marple, *Digital spectral analysis*, Prentice-Hall, Englewood Cliffs, 1987.

[3] D. C. Rife and R. R. Boorstyn, "Multiple tone parameter estimation from discrete-time observations," *Bell Syst. Tech. J.*, pp. 1389–1410, Nov. 1976.

[4] H. L. V. Trees, *Detection, Estimation and Modulation Theory, Part 1*, New York: Wiley, 1971.

[5] E. Jacobsen and P. Kootsookos, "Fast, accurate frequency estimators," *IEEE Signal Processings Magazine*, vol. 24, pp. 123–125, May 2007.

[6] C. Candan, "A method for fine resolution frequency estimation from three DFT samples," *IEEE Signal Processing Letters*, vol. 18, no. 6, pp. 351-354, June 2011.

[7] M. D. Macleod, "Fast nearly ML estimation of the parameters of real or complex single tones or resolved multiple tones," *IEEE Transactions on Signal Processing*, vol. 46, no. 1, pp. 141–148, Jan. 1998.

[8] B. G. Quinn, “Estimating frequency by interpolation using Fourier coefficients,” *IEEE Transaction on Signal Processing*, vol. 42, no. 5, pp. 1264–1268, May 1994.

[9] B. Kasztenny and E. Rosolowski, “Two new measuring algorithms for generator and transformer relaying*,” IEEE Trans. Power Del*., vol. 13, no. 4, pp. 1053–1959, Oct. 1998.

[10] B. G. Quinn, “Estimation of frequency, amplitude, and phase from the DFT of a time series,” *IEEE Transactions on Signal Processing*, vol. 45, no. 3, pp. 814–817, Mar. 1997.

[11] S. Provencher, “Estimation of complex single-tone parameters in the DFT domain,” *IEEE Transactions on Signal Processing*, vol. 58, no. 7, pp. 3879–3883, 2010.

[12] D. Knuth, “Johann Faulhaber and sums of powers,” *Math. Comput*., vol. 61, no. 203, pp. 277–294, 1993.

[13] B. Bischl, U. Liggesand C Weihs, “Frequency estimation by DFT interpolation: a comparison of methods,” *Technical Report*, Dortmund University, SFB 475 2009.

APPENDIX

A.1. MATLAB Codes (Performance Measure vs. Bin Number)

```
clear all; clc;
tstlen=64;              % Effective length of test vector.
NZ=1;                   % Switch to enable (1) or disable (0) noise.
NOISE=0.25;             % Set noise level.
N=1000;                 % Number of trials in test.
Amp=4;                  % Exponential amplitude

quinerr=zeros(size(1:N));   % Allocate vector for results.
quaderr=zeros(size(1:N));   % Allocate vector for results.
macldерr=zeros(size(1:N));  % Allocate vector for results.
candanerr=zeros(size(1:N)); % Allocate vector for results.

quinest=zeros(size(1:N));   % Allocate vector for results.
quadest=zeros(size(1:N));   % Allocate vector for results.
macldest=zeros(size(1:N));  % Allocate vector for results.
candanest=zeros(size(1:N)); % Allocate vector for results.

SNR=zeros(size(1:N));       % Allocate vector for results.

K=10;                   % Number of bins to test.
quinr=zeros(size(1:K));     % Allocate vector for results.
quadr=zeros(size(1:K));     % Allocate vector for results.
macldr=zeros(size(1:K));    % Allocate vector for results.
candanr=zeros(size(1:K));   % Allocate vector for results.

quinvar=zeros(size(1:K));   % Allocate vector for results.
quadvar=zeros(size(1:K));   % Allocate vector for results.
macldvar=zeros(size(1:K));  % Allocate vector for results.
candanvar=zeros(size(1:K)); % Allocate vector for results.

quinbias=zeros(size(1:K));  % Allocate vector for results.
quadbias=zeros(size(1:K));  % Allocate vector for results.
macldbias=zeros(size(1:K)); % Allocate vector for results.
candanbias=zeros(size(1:K));% Allocate vector for results.

targ=zeros(size(1:K));      % Allocate vector for results.

M=0;                    % Current test number.

binstrt = 9.0;          % Starting bin number.
binstep = 0.1;          % Bin delta step size.
binend = 9.9;           % Ending bin number.

for bin = binstrt: binstep: binend,

M=M+1;                  % Current test number.
```

```
target=bin;                % Calculate desired target result
targ(M)=bin;

fprintf('Peak is at %f.\n',bin);

for I = 1:N,               % Run trials.

phz=2*pi*rand(1);
 % Generate signal.
cw=Amp*exp(j*((2*pi*(1:tstlen)*bin./(tstlen))+phz));
 % Calculate signal power.

sigp=sum(abs(cw(1:tstlen).^2));

cwn=cw;

if(NZ==1)
  % Generate noise.

  % Set signal level and add noise (noise variance=1 on I and on Q)
nzi=sqrt(NOISE)*randn(1,tstlen);
nzq=sqrt(NOISE)*randn(1,tstlen);
nzv=nzi+j*nzq;

nzp=sum(abs(nzv(1:tstlen).^2));

if(nzp>0)
SNR(I)=10*log10(sigp/nzp);                    % Calculate SNR.
else
  % Use this for no noise.

SNR(I)=100.0;                              end

cwn=cw+nzv;

end
% End noise.

dftshrt(1:tstlen)=fft(cwn(1:tstlen));          % DFT.
magshrt(1:tstlen)=abs(dftshrt);                % DFT magnitude.
 % Find raw peak magnitude and location.
 [rawmag,rawind]=max(magshrt);

% Isolated 3 samples around peak.
pk3vect(1:3)=dftshrt(rawind-1:rawind+1);
quinest(I)=rawind-1+quin(pk3vect); % Do Quinn's first estimation.
quinerr(I)=target-quinest(I);      % Calculate and save error.
 % Do modified quadratic estimation.
quadest(I)=rawind-1+quadterp(pk3vect);
quaderr(I)=target-quadest(I);      % Calculate and save error.



macldest(I)=rawind-1+macleod(pk3vect);% Do Macleod's estimation.
maclderr(I)=target-macldest(I);    % Calculate and save error.
candanest(I)=rawind-1+candan(tstlen,pk3vect);% Do Candan's
candanerr(I)=target-candanest(I);  % Calculate and save error.
end                                % End inner for loop.
```

```
quinvar(M)=sqrt(mean(quinerr.^2));     % Calculate result rms error.
quadvar(M)=sqrt(mean(quaderr.^2));     % Calculate result rms error.
macldvar(M)=sqrt(mean(maclderr.^2));  % Calculate result rms error.
candanvar(M)=sqrt(mean(candanerr.^2)); % Calculate result rms error.

SNRmn=mean(SNR);                    % Calculate result mean.

quinbias(M)=mean(quinerr);           % Calculate bias.
quadbias(M)=mean(quaderr);           % Calculate bias.
macldbias(M)=mean(maclderr);         % Calculate bias.
candanbias(M)=mean(candanerr);       % Calculate bias.

quinr(M)=mean(quinest);              % Calculate average result.
quadr(M)=mean(quadest);              % Calculate average result.
macldr(M)=mean(macldest);            % Calculate average result.
macldr(M)=mean(macldest);            % Calculate average result.
candanr(M)=mean(candanest);          % Calculate average result.

end;                                 % End bin loop.

xs(1:M)=binstrt+(((1:M)-1)*binstep);% Generate scale for plot

figure(1);
plot(xs(1:M),targ(1:M),'r-',xs(1:M),quadr(1:M),'mo',xs(1:M),quinr(1:M),...
    'gs', xs(1:M),macldr(1:M),'bx',xs(1:M),...
candanr(1:M),'c*');
title('Average Peak Location Estimates (bin) vs Bin Number'); grid on
legend('Real Value','Jacobsen','Quin','Macleod','Candan');
xlabel('Bin Number')
ylabel('Average Peak Location Estimates')
figure(2);

plot(xs(1:M),quadvar(1:M),'mo:',xs(1:M),quinvar(1:M),'gs:',...
    ,xs(1:M),macldvar(1:M),'bx:',xs(1:M),...
candanvar(1:M),'c*:');
title('Estimator Variance vs Bin Number'); grid on
legend('Jacobsen','Quin','Macleod','Candan');
xlabel('Bin Number')
ylabel('Variance Estimates')

figure(3);
plot(xs(1:M),quadbias(1:M),'mo:',xs(1:M),quinbias(1:M),'gs:',...
     ,xs(1:M),macldbias(1:M),'bx:',xs(1:M),...
candanbias(1:M),'c*:');
title('Estimator Bias vs Bin Number'); grid on
legend('Jacobsen','Quin','Macleod','Candan');
xlabel('Bin Number')
ylabel('Bias Estimates')


function x=candan(tstlen,y)
% Candan
% Performs a quadratic-fit peak location interpolation on a three-  % element input vector y.
% Returns -0.5< x < 0.5, which is the fraction of the sample       % spacing about the center
% element where the peak is estimated to be.
NN=tstlen;

x=(tan(pi/NN)/(pi/NN)).*real((y(1)-y(3))/((2*y(2))-y(1)-y(3)));
```

```
function [x] = macleod(Y)

% Y is a three-element complex vector with the
% DFT output magnitude maximizer as the center element.
%
% Returns -0.5< x < 0.5, which is the fraction of the sample
% spacing (i.e., bin width) about the center element where the
% peak is estimated to be.


ref = Y(2);                 % Isolate phase reference.
R = real(Y.*conj(ref));% Generate phase corrected coefficients
gamma = (R(1)-R(3))/((2*R(2))+R(1)+R(3));       % Calculate offset.
delta = (sqrt(1 + 8*gamma*gamma)-1)/(4*gamma);  % Final estimate.
x = delta;



function x=quadterp(y)
% Jacobsen
% Performs a quadratic-fit peak location interpolation on a three-  %element input vector y.
% Returns -0.5< x < 0.5, which is the fraction of the sample% spacing about the center
% element where the peak is estimated to be.

x=real((y(1)-y(3))/((2*y(2))-y(1)-y(3)));


function [x] = quin(pk3vect)

% pk3vect is a three-element complex vector with the
% DFT output magnitude maximizer as the center element.
% Returns -0.5< x < 0.5, which is the fraction of the sample
% spacing (i.e., bin width) about the center element where the
% peak is estimated to be.

alpha1=real(pk3vect(1)/pk3vect(2));
alpha2=real(pk3vect(3)/pk3vect(2));

delta1= alpha1/(1-alpha1);
delta2=-alpha2/(1-alpha2);

if ((delta1>0) & (delta2>0))
   x=delta2;
else
      x=delta1;
end
```

A.2. MATLAB Codes (Performance Measure vs. SNR)

```
%FIGURE BIAS (MC_deneme_f_bias_plot)
clear all,
N=8;
SNR_dB=100;
noise_type=2; % select 1 for colored noise, otherwise white noise
MC_run=1000;
wbin_vec=setdiff(2.02:0.02:2.48,2);
hh=[0.707 0.707];
indexwbin=0;
error=zeros(length(wbin_vec),5);
SNR=10^(SNR_dB/10);
for wbin=wbin_vec,
   indexwbin=indexwbin+1;
      delta=wbin-round(wbin);

      %Generate noise
      if noise_type==1
         s=repmat(sqrt(SNR)*exp(j*wbin*2*pi/N*(0:N-1)'+j*rand(1)*2*pi),[1 MC_run]) +...
           sqrt(1/2)*(filter(hh,1,randn(N,MC_run))+j*filter(hh,1,randn(N,MC_run)));
      else
         s=repmat(sqrt(SNR)*exp(j*wbin*2*pi/N*(0:N-1)'+j*rand(1)*2*pi),[1 MC_run]) +...
           sqrt(1/2)*(randn(N,MC_run)+j*randn(N,MC_run));
      end


[wcandan,wquinn,wjacobsen,wmcl,w_dftmag]=find_Dop_peak_around_abin_f(s,round(wbin)+1);
      error=[w_dftmag' wquinn' wmcl' wjacobsen'  wcandan']-delta-round(wbin);
   bias(indexwbin,:) = mean(error);
   MSE(indexwbin,:)  = var(error)+mean(error).^2;
   varer(indexwbin,:) = var(error);
   dum = CR_calc(SNR_dB,N,wbin);
   CR(indexwbin) = dum(3,3);
end;
figure,
semilogy(wbin_vec-round(wbin_vec),abs(bias(:,1)),'linewidth',2);  hold all;
semilogy(wbin_vec-round(wbin_vec),abs(bias(:,2)),'d-','linewidth',2);
semilogy(wbin_vec-round(wbin_vec),abs(bias(:,3)),'s--','linewidth',2);
semilogy(wbin_vec-round(wbin_vec),abs(bias(:,4)),'+:','linewidth',2);
semilogy(wbin_vec-round(wbin_vec),abs(bias(:,5)),'o-','linewidth',2);
legend('Parabolic Fit','Quinn','MacLeod','Jacobsen','Candan');
h=xlabel('\delta'); set(h,'fontsize',11) % increase the title size
h=ylabel('Bias'); set(h,'fontsize',11) % increase the title size
h=title(['Bias of \delta estimate for N=' num2str(N)]);
set(h,'fontsize',11);

% RMSE=sqrt(MSE);

% figure(2),semilogy(wbin_vec-round(wbin_vec),RMSE,SNR_dB_vec,sqrt(CR));
legend('new','quinn','jacobsen','MacLeod','DFT Mag','CR')
% xlabel('\delta'); ylabel('RMSE'); title(['N=' num2str(N)]);
%plot(error),
hold off;
disp('press space for next figure');
pause;
%%%%%%%%%
%FIGURE RMSE

clear all,
noise_type=2; % select 1 for colored noise, otherwise white noise
```

```
hh=[0.707 0.707];
N=8;
SNR_dB_vec=[0:5:80];
MC_run=20e3;
%wbin=2.1;
%delta=wbin-round(wbin);

indexSNR=0;
error=zeros(MC_run,5);
for SNR_dB=SNR_dB_vec,
   SNR=10^(SNR_dB/10);
   indexSNR=indexSNR+1;
      wbin=2.25;
      delta=wbin-round(wbin);


       %Generate noise
      if noise_type==1
         s=repmat(sqrt(SNR)*exp(j*wbin*2*pi/N*(0:N-1)'+j*rand(1)*2*pi),[1 MC_run]) +...
           sqrt(1/2)*(filter(hh,1,randn(N,MC_run))+j*filter(hh,1,randn(N,MC_run)));
      else
         s=repmat(sqrt(SNR)*exp(j*wbin*2*pi/N*(0:N-1)'+j*rand(1)*2*pi),[1 MC_run]) +...
           sqrt(1/2)*(randn(N,MC_run)+j*randn(N,MC_run));
      end



[wcandan,wquinn,wjacobsen,wmcl,w_dftmag]=find_Dop_peak_around_abin_f(s,round(wbin)+1);
      error=[w_dftmag' wquinn' wmcl' wjacobsen'  wcandan']-delta-round(wbin);
   bias(indexSNR,:) = mean(error);
   MSE(indexSNR,:)  = var(error)+mean(error).^2;
   varer(indexSNR,:) = var(error);
   dum = CR_calc(SNR_dB,N,wbin);
   CR(indexSNR) = dum(3,3);
end;
%figure(1),semilogy(SNR_dB_vec,abs(bias)); legend('new','quinn','jacobsen','MacLeod','DFT Mag');
%xlabel('Sample SNR'); ylabel('Bias'); title(['N=' num2str(N)]);
figure,
semilogy(SNR_dB_vec,abs(bias(:,1)),'linewidth',2);  hold all;
semilogy(SNR_dB_vec,abs(bias(:,2)),'d-','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,3)),'s--','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,4)),'+:','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,5)),'o-','linewidth',2);
legend('Parabolic Fit','Quinn','MacLeod','Jacobsen','Candan');
h=xlabel('Input SNR (dB)'); set(h,'fontsize',11) % increase the title size
h=ylabel('Bias'); set(h,'fontsize',11) % increase the title size
h=title(['Bias of \delta estimate for N=' num2str(N)]);
set(h,'fontsize',11);
hold off;

RMSE=sqrt(MSE);

% figure(2),semilogy(SNR_dB_vec,RMSE,SNR_dB_vec,sqrt(CR));
legend('new','quinn','jacobsen','MacLeod','DFT Mag','CR')
% xlabel('Sample SNR'); ylabel('RMSE'); title(['N=' num2str(N)]);
%plot(error),
hold off;


figure,
semilogy(SNR_dB_vec,abs(RMSE(:,1)),'linewidth',2);  hold all;
```

```
semilogy(SNR_dB_vec,abs(RMSE(:,2)),'d-','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,3)),'s--','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,4)),'+:','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,5)),'o-','linewidth',2);
semilogy(SNR_dB_vec,sqrt(CR),'linewidth',2)
legend('Parabolic Fit','Quinn','MacLeod','Jacobsen','Candan','CR Bound');
h=xlabel('Input SNR (dB)'); set(h,'fontsize',11) % increase the title size
h=ylabel('RMSE'); set(h,'fontsize',11) % increase the title size
h=title(['RMSE of \delta estimate for N=' num2str(N) ' and  \delta=' num2str(delta) ]);
set(h,'fontsize',11);


disp('press space for next figure');
pause;

clear all,
noise_type=2; % select 1 for colored noise, otherwise white noise
hh=[0.707 0.707];
N=32;
SNR_dB_vec=[0:5:80];
MC_run=20e3;
%wbin=2.1;
%delta=wbin-round(wbin);

indexSNR=0;
error=zeros(MC_run,5);
for SNR_dB=SNR_dB_vec,
   SNR=10^(SNR_dB/10);
   indexSNR=indexSNR+1;
      wbin=2.25;
      delta=wbin-round(wbin);

       %Generate noise
      if noise_type==1
         s=repmat(sqrt(SNR)*exp(j*wbin*2*pi/N*(0:N-1)'+j*rand(1)*2*pi),[1 MC_run]) +...
           sqrt(1/2)*(filter(hh,1,randn(N,MC_run))+j*filter(hh,1,randn(N,MC_run)));
      else
         s=repmat(sqrt(SNR)*exp(j*wbin*2*pi/N*(0:N-1)'+j*rand(1)*2*pi),[1 MC_run]) +...
           sqrt(1/2)*(randn(N,MC_run)+j*randn(N,MC_run));
      end



[wcandan,wquinn,wjacobsen,wmcl,w_dftmag]=find_Dop_peak_around_abin_f(s,round(wbin)+1);
      error=[w_dftmag' wquinn' wmcl' wjacobsen'  wcandan']-delta-round(wbin);
   bias(indexSNR,:) = mean(error);
   MSE(indexSNR,:)  = var(error)+mean(error).^2;
   varer(indexSNR,:) = var(error);
   dum = CR_calc(SNR_dB,N,wbin);
   CR(indexSNR) = dum(3,3);
end;
%figure(1),semilogy(SNR_dB_vec,abs(bias)); legend('new','quinn','jacobsen','MacLeod','DFT Mag');
%xlabel('Sample SNR'); ylabel('Bias'); title(['N=' num2str(N)]);
figure,
semilogy(SNR_dB_vec,abs(bias(:,1)),'linewidth',2);  hold all;
semilogy(SNR_dB_vec,abs(bias(:,2)),'d-','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,3)),'s--','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,4)),'+:','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,5)),'o-','linewidth',2);
legend('Parabolic Fit','Quinn','MacLeod','Jacobsen','Candan');
h=xlabel('Input SNR (dB)'); set(h,'fontsize',11) % increase the title size
```

```
h=ylabel('Bias'); set(h,'fontsize',11) % increase the title size
h=title(['Bias of \delta estimate for N=' num2str(N)]);
set(h,'fontsize',11);
hold off;

RMSE=sqrt(MSE);

% figure(2),semilogy(SNR_dB_vec,RMSE,SNR_dB_vec,sqrt(CR));
legend('new','quinn','jacobsen','MacLeod','DFT Mag','CR')
% xlabel('Sample SNR'); ylabel('RMSE'); title(['N=' num2str(N)]);
%plot(error),
hold off;


figure,
semilogy(SNR_dB_vec,abs(RMSE(:,1)),'linewidth',2);  hold all;
semilogy(SNR_dB_vec,abs(RMSE(:,2)),'d-','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,3)),'s--','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,4)),'+:','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,5)),'o-','linewidth',2);
semilogy(SNR_dB_vec,sqrt(CR),'linewidth',2)
legend('Parabolic Fit','Quinn','MacLeod','Jacobsen','Candan','CR Bound');
h=xlabel('Input SNR (dB)'); set(h,'fontsize',11) % increase the title size
h=ylabel('RMSE'); set(h,'fontsize',11) % increase the title size
h=title(['RMSE of \delta estimate for N=' num2str(N) ' and  \delta=' num2str(delta) ]);
set(h,'fontsize',11);


disp('press space for next figure');
pause;


clear all,
noise_type=2; % select 1 for colored noise, otherwise white noise
hh=[0.707 0.707];
N=128;
SNR_dB_vec=[0:5:80];
MC_run=20e3;
%wbin=2.1;
%delta=wbin-round(wbin);

indexSNR=0;error=zeros(MC_run,5);
for SNR_dB=SNR_dB_vec,
   SNR=10^(SNR_dB/10);
   indexSNR=indexSNR+1;
      wbin=2.25;
      delta=wbin-round(wbin);

       %Generate noise
      if noise_type==1
         s=repmat(sqrt(SNR)*exp(j*wbin*2*pi/N*(0:N-1)'+j*rand(1)*2*pi),[1 MC_run]) +...
           sqrt(1/2)*(filter(hh,1,randn(N,MC_run))+j*filter(hh,1,randn(N,MC_run)));
      else
         s=repmat(sqrt(SNR)*exp(j*wbin*2*pi/N*(0:N-1)'+j*rand(1)*2*pi),[1 MC_run]) +...
           sqrt(1/2)*(randn(N,MC_run)+j*randn(N,MC_run));
      end



[wcandan,wquinn,wjacobsen,wmcl,w_dftmag]=find_Dop_peak_around_abin_f(s,round(wbin)+1);
      error=[w_dftmag' wquinn' wmcl' wjacobsen'  wcandan']-delta-round(wbin);
```

```
   bias(indexSNR,:) = mean(error);
   MSE(indexSNR,:)  = var(error)+mean(error).^2;
   varer(indexSNR,:) = var(error);
   dum = CR_calc(SNR_dB,N,wbin);
   CR(indexSNR) = dum(3,3);
end;
%figure(1),semilogy(SNR_dB_vec,abs(bias)); legend('new','quinn','jacobsen','MacLeod','DFT Mag');
%xlabel('Sample SNR'); ylabel('Bias'); title(['N=' num2str(N)]);
figure,
semilogy(SNR_dB_vec,abs(bias(:,1)),'linewidth',2);  hold all;
semilogy(SNR_dB_vec,abs(bias(:,2)),'d-','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,3)),'s--','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,4)),'+:','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,5)),'o-','linewidth',2);
legend('Parabolic Fit','Quinn','MacLeod','Jacobsen','Candan');
xlabel('Input SNR (dB)'); ylabel('Bias'); title(['Bias of \delta estimate for N=' num2str(N)]);
hold off;

RMSE=sqrt(MSE);

% figure(2),semilogy(SNR_dB_vec,RMSE,SNR_dB_vec,sqrt(CR));
legend('new','quinn','jacobsen','MacLeod','DFT Mag','CR')
% xlabel('Sample SNR'); ylabel('RMSE'); title(['N=' num2str(N)]);
%plot(error),
hold off;

figure,
semilogy(SNR_dB_vec,abs(RMSE(:,1)),'linewidth',2);  hold all;
semilogy(SNR_dB_vec,abs(RMSE(:,2)),'d-','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,3)),'s--','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,4)),'+:','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,5)),'o-','linewidth',2);
semilogy(SNR_dB_vec,sqrt(CR),'linewidth',2)
legend('Parabolic Fit','Quinn','MacLeod','Jacobsen','Candan','CR Bound');
h=xlabel('Input SNR (dB)'); set(h,'fontsize',11) % increase the title size
h=ylabel('RMSE'); set(h,'fontsize',11) % increase the title size
h=title(['RMSE of \delta estimate for N=' num2str(N) ' and  \delta=' num2str(delta) ]);
set(h,'fontsize',11);
hold off;

clear all,
noise_type=2; % select 1 for colored noise, otherwise white noise
hh=[0.707 0.707];
N=256;
SNR_dB_vec=[0:5:80];
MC_run=20e3;
%wbin=2.1;
%delta=wbin-round(wbin);

indexSNR=0;
error=zeros(MC_run,5);
for SNR_dB=SNR_dB_vec,
   SNR=10^(SNR_dB/10);
   indexSNR=indexSNR+1;
      wbin=2.25;
      delta=wbin-round(wbin);

       %Generate noise
      if noise_type==1
         s=repmat(sqrt(SNR)*exp(j*wbin*2*pi/N*(0:N-1)'+j*rand(1)*2*pi),[1 MC_run]) +...
           sqrt(1/2)*(filter(hh,1,randn(N,MC_run))+j*filter(hh,1,randn(N,MC_run)));
```

```
        else
           s=repmat(sqrt(SNR)*exp(j*wbin*2*pi/N*(0:N-1)'+j*rand(1)*2*pi),[1 MC_run]) +...
             sqrt(1/2)*(randn(N,MC_run)+j*randn(N,MC_run));
        end



[wcandan,wquinn,wjacobsen,wmcl,w_dftmag]=find_Dop_peak_around_abin_f(s,round(wbin)+1);
        error=[w_dftmag' wquinn' wmcl' wjacobsen'  wcandan']-delta-round(wbin);
    bias(indexSNR,:) = mean(error);
    MSE(indexSNR,:)  = var(error)+mean(error).^2;
    varer(indexSNR,:) = var(error);
    dum = CR_calc(SNR_dB,N,wbin);
    CR(indexSNR) = dum(3,3);
end;
%figure(1),semilogy(SNR_dB_vec,abs(bias)); legend('new','quinn','jacobsen','MacLeod','DFT Mag');
%xlabel('Sample SNR'); ylabel('Bias'); title(['N=' num2str(N)]);
figure,
semilogy(SNR_dB_vec,abs(bias(:,1)),'linewidth',2);  hold all;
semilogy(SNR_dB_vec,abs(bias(:,2)),'d-','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,3)),'s--','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,4)),'+:','linewidth',2);
semilogy(SNR_dB_vec,abs(bias(:,5)),'o-','linewidth',2);
legend('Parabolic Fit','Quinn','MacLeod','Jacobsen','Candan');
h=xlabel('Input SNR (dB)'); set(h,'fontsize',11) % increase the title size
h=ylabel('Bias'); set(h,'fontsize',11) % increase the title size
h=title(['Bias of \delta estimate for N=' num2str(N)]);
set(h,'fontsize',11);
hold off;

RMSE=sqrt(MSE);

% figure(2),semilogy(SNR_dB_vec,RMSE,SNR_dB_vec,sqrt(CR));
legend('new','quinn','jacobsen','MacLeod','DFT Mag','CR')
% xlabel('Sample SNR'); ylabel('RMSE'); title(['N=' num2str(N)]);
%plot(error),
hold off;


figure,
semilogy(SNR_dB_vec,abs(RMSE(:,1)),'linewidth',2);  hold all;
semilogy(SNR_dB_vec,abs(RMSE(:,2)),'d-','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,3)),'s--','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,4)),'+:','linewidth',2);
semilogy(SNR_dB_vec,abs(RMSE(:,5)),'o-','linewidth',2);
semilogy(SNR_dB_vec,sqrt(CR),'linewidth',2)
legend('Parabolic Fit','Quinn','MacLeod','Jacobsen','Candan','CR Bound');
h=xlabel('Input SNR (dB)'); set(h,'fontsize',11) % increase the title size
h=ylabel('RMSE'); set(h,'fontsize',11) % increase the title size
h=title(['RMSE of \delta estimate for N=' num2str(N) ' and  \delta=' num2str(delta) ]);
set(h,'fontsize',11);


disp('press space for next figure');
pause;
```