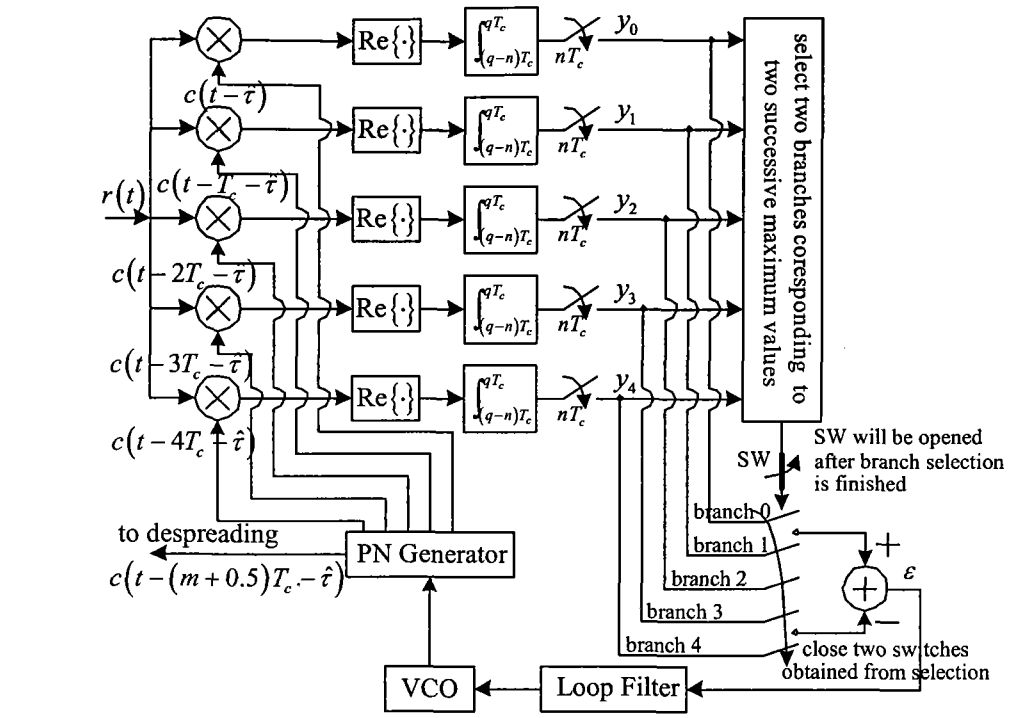
Chapter 3

Modified Extended Range DLL Using Selection Branch Algorithms

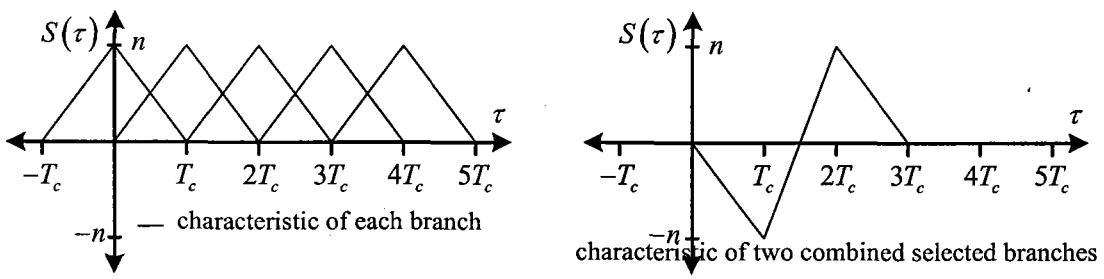
Traditional PN code synchronization methods usually assume that the received PN code phase can be anywhere in the entire PN code period. This is true when the range of the propagation delay is comparable to the PN period, such as the global positioning system (GPS) and satellite communications. On the other hand, in a mobile communication system with GPS controlling clock on both the base and mobile stations, such as, IS-95 [80] and cdma2000 [81], the delay of the received PN code phase is due to the propagated distance and so it is always delayed when compared with the local code phase of the receiver. Furthermore, the propagation delay is small, especially in the case of microcell or picocell structure. If the cell radius does not exceed one km, the phase delay is limited to four chip durations at the IS-95 chip rate (1.2288 Mcps) [80, 82]. In such a case, the PN code phase synchronization at the receiver may combine the acquisition process and tracking process into one. However, the conventional delay locked-loop (DLL) can not be used since its pull-in range does not cover the extended range (four chip durations) of the incoming code phase. Therefore, the DLL must be modified to cover the extended pull-in range.

Two methods have been proposed by Wilde [34, 35] to accomplish the extended pull-in range of the DLL. However, they exhibit trade-off among complexity, pull-in range, and tracking jitter noise. The first method [34] increases the code phase spacing ($2\delta T_c$, where T_c is the chip duration, $\delta \in (0, 1]$) between the advanced and retarded versions of locally generated PN code, but the maximum achievable pull-in range is limited to only four chip durations (when δ=1). Therefore, it cannot be used for pull-in range larger than four chips. Also, the average time to lock becomes slower. The second method [35] uses additional correlators, each of which possesses a different code phase, to broaden the pull-in range. Unfortunately, these added correlators increase the tracking jitter noise. To solve the jitter noise problem, Wilde has also proposed a modified DLL called correlation branch selection DLL (SelDLL) [36].

In the SelDLL scheme, shown on Figure 3.1(a), the received code phase is detected by first selecting a pair of correlators which have the two highest consecutive correlated values (output of integrate-and-dump, I&D). This pair of correlators is then used to form the conventional DLL. Consequently, the tracking jitter noise is relieved to the same level as that of the ordinary DLL. However, difficulty occurs when the incoming code phase delay is in the neighborhood of integer number of the chip duration. In such a situation, one correlated value is near the peak value while the other is close to zero. Coupled with the effect of noise, an incorrect pair may be selected, inducing a selection loss. Furthermore, even though the incoming code phase is not located in the neighborhood of an integer number, it still suffers from wrong selection of two correct branches. Owing to these situations, the selected correlators cannot track the received code phase, incurring penalty time (time spent in an attempt to synchronize) as a consequence, and the correlator selection process must be performed all over again.



(a) Structure

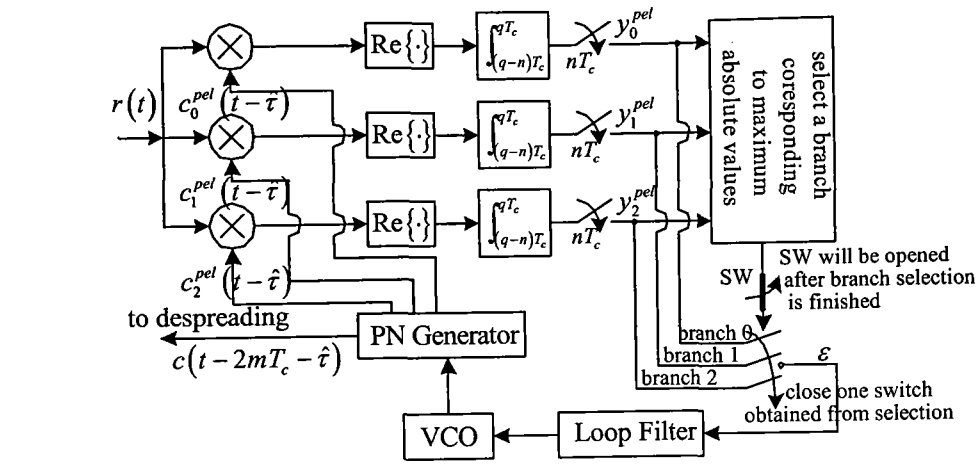


(b) Discriminator before selection (c) Discriminator after selection

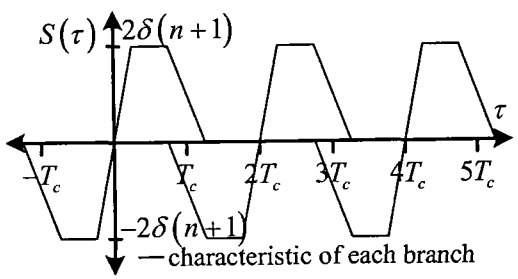
Figure 3.1. Structure and discriminator of SelDLL.

In order to mitigate the probability of selection loss, an improved scheme is introduced, shown in Figure 3.2(a). It is called parallel early-late DLL (PelDLL). In this scheme, the received signal is correlated with parallel early-late PN sequences, each of which is phase-shifted by two-chip durations from the previous one. Thereafter, only one branch possessing the highest correlated value is selected for further utilization in tracking the incoming PN code phase. However, for PelDLL, if the initial incoming code phase is in the neighborhood of the locking points, which are $0T_c$, $2T_c$, and $4T_c$ for branch 0, 1, and 2, respectively, an incorrect branch is likely to be selected by the selection algorithm. A solution of this problem is proposed, called parallel extended range DLL (PerDLL), as shown in Figure 3.3(a). In PerDLL, the discriminator characteristic and selection algorithm have been modified. However, the noise variance in each branch increases.

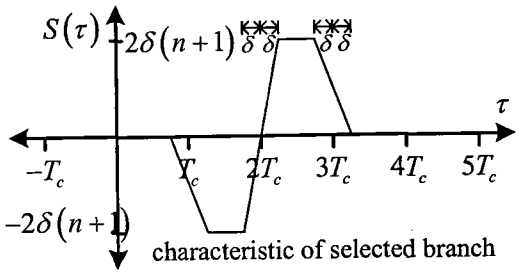
Obviously, the process of the above systems (SelDLL, PelDLL, PerDLL) comprise of two steps: branch selection and fine adjustment of the local code phase. The latter operation is the conventional DLL, the performance of which has been already shown in the literature [1, 29]. Therefore, only the performance of the former is considered here and it is evaluated as the probability of correct branch selection. Results show improvement in the modified schemes.



(a) Structure



(b) Discriminator before selection



(c) Discriminator after selection

Figure 3.2. Structure and discriminator of PelDLL.

It is assumed that the carrier phase is known, so we have coherent carrier demodulation. It is also assumed that no data is being transmitted during the synchronization process. Therefore, the received signal, $r(t)$, is represented in the equivalent baseband model as

$$r(t)=\sqrt{2P}c(t-\tau)+\eta(t) \tag{3.1}$$

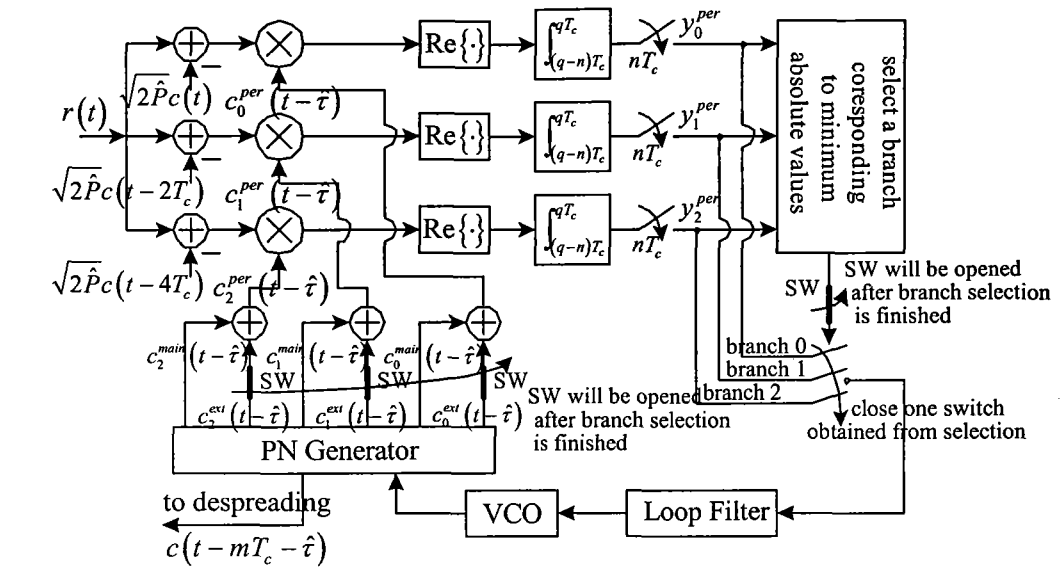
where P is the average signal power, $\eta(t)$ is complex envelope of zero mean white Gaussian noise with two-sided power spectral density $2N_0$, τ is the code phase delay, and $c(t)$ is the PN spreading waveform, which can be expressed as

$$c(t)=\sum_{i=-\infty}^{\infty} c_i P_{T_c}(t-iT_c) \tag{3.2}$$

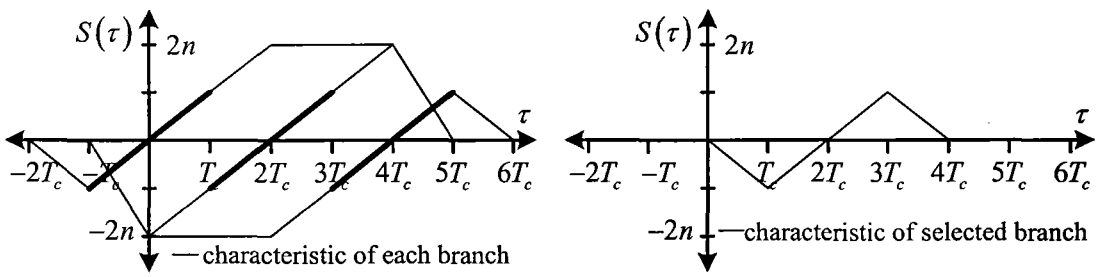
where c_i is a periodic binary m-sequence, with period N, i.e., $c_i \in \{1,-1\}$ and $c_i = c_{i+N}$ for all i, T_c is the chip duration, $P_{T_c}(t)$ is defined as a unit amplitude rectangular pulse shape in the interval $[0,\ T_c)$. The code phase delay τ is modeled as a uniform random variable. It can be written without loss of generality as

$$\tau=(\beta+\gamma)T_c, \tag{3.3}$$

where γ is a uniform random variable in the interval $[0,\ 1)$, while β is a discrete uniform random variable. For a microcell structure with $0 \le \tau < 4T_c$, the probability mass function (pmf) of β is



(a) Structure



(b) Discriminator before selection (c) Discriminator after selection

Figure 3.3. Structure and discriminator of PerDLL.

$$p(\beta)=\begin{cases}1/4, & \beta\in\{0,1,2,3\}\\ 0, & \text{otherwise}\end{cases}. \tag{3.4}$$

The contents of this chapter are organized as follows. In Section 3.1, 3.2, and 3.3, the schemes of SelDLL, PelDLL, PerDLL are described and the corresponding probabilities of correct selection are derived. Results and discussions of these schemes are presented in Section 3.4. Finally, the chapter ends with Section 3.5, which is a summary.

3.1 Correlation Branch Selection DLL (SelDLL)

In this section, we analyze the performance of SelDLL scheme, since the performance of SelDLL in terms of the probability of correct selection was not presented in [36]. For the SelDLL [36], the received signal $r(t)$ is correlated with a bank of five parallel local PN sequences, each of which is phase-shifted by one chip duration from the previous one, as shown on Figure 3.1(a). The two largest integrate-and-dump outputs are selected. Subsequently, such two corresponding branches are connected with the summation operator to produce the error signal ε, with the branch having the higher label being the branch with positive gain and the branch having the lower label being the branch with negative gain. For example, if branch labeled 1 and 2 are selected because they have the two highest outputs, the branch labeled 2 will be the positive-gain branch and branch labeled 1 will be the negative-gain branch. The local code phase $\hat{\tau}$, which was initially set at 0, is adjusted according to the error signal. The value m at the output of the PN generator is the minimum value between j and k, where j and k are PN phases of the

selected branches. The selected branches are still used to adjust the local code phase until synchronization is completed. However, when the system perceives that the tracking can not be achieved, the process described above will be repeated again.

The correlated output described above of the m^{th} branch, m=0, 1, 2, 3, 4, at $t = qT_c$ is expressed as

$$y_m = \int_{(q-n)T_c}^{qT_c} \mathrm{Re}\left\{\left[\sqrt{2P}c(t-\tau)+\eta(t)\right]c(t-mT_c)\right\}dt = \sqrt{2P}T_c s(q,\beta,\gamma,m,n)+\eta_{m,n} \quad (3.5)$$

where nT_c is the integration length of the integrate-and-dump circuit and q is a discrete uniform random variable with pmf

$$p(q) = \begin{cases} 1/N, & q \in \{0,1,\ldots,N-1\} \\ 0, & \text{otherwise} \end{cases}, \quad (3.6)$$

$s(q,\beta,\gamma,m,n)$ and $\eta_{m,n}$ are the signal and noise terms, respectively, given by

$$s(q,\beta,\gamma,m,n) = \gamma\sum_{i=0}^{n-1} c_{q-n-m+i}c_{q-n-\beta-1+i} + (1-\gamma)\sum_{j=0}^{n-1} c_{q-n-m+j}c_{q-n-\beta+j}, \quad (3.7)$$

$$\eta_{m,n} = \int_{(q-n)T_c}^{qT_c} \mathrm{Re}\{z(t)\}c(t-mT_c)dt. \quad (3.8)$$

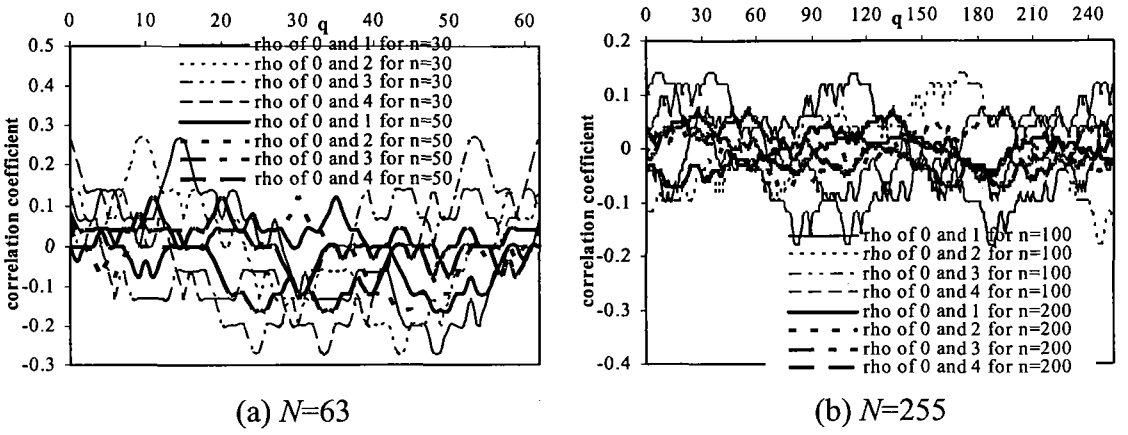
Obviously, the signal term $s(q,\beta,\gamma,m,n)$ is the weighted sum of two partial autocorrelations and it depends on q, β, γ, m, and n. $\eta_{m,n}$ is a zero mean Gaussian random variable with variance nN_0T_c. The derivations of (3.7) and noise variance are presented in Appendices A and B, respectively. However, for convenience, in the sequel, $s(q,\beta,\gamma,m,n)$ and $\eta_{m,n}$ are abbreviated as s_m and η_m, respectively.

In the noise-free environment, before the selection is performed, the discriminator characteristic, $S(\tau)$, of each branch behaves as a shifted autocorrelation of the PN signal as shown in Figure 3.1(b). It should be noted that the number of correlated PN code is selected to cover the range of the phase uncertainty region. In order to synchronize to the incoming code phase, two branches are selected and used to form a DLL. For example, $S(\tau)$ behaves as shown in Figure 3.1(c) when branches 1 and 2 are selected and combined to work as a DLL. However, due to noise, the selected pair may be a wrong pair. This induces selection loss, which is the complement of correct selection. The probability of correct selection can be calculated as follows.

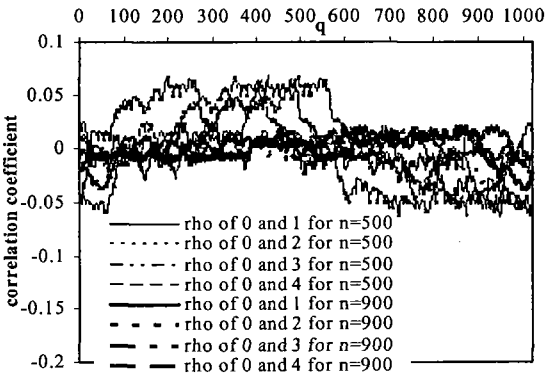
Let H_β be the hypothesis that the incoming code phase τ is in the range $[\beta, \beta+1)T_c$, $\beta = 0, 1, 2, 3$, i.e.,

$$\begin{aligned} H_0:&\ \tau \in [0,1)T_c, & \Pr(H_0) &= \Pr(\beta=0) = 1/4 \\ H_1:&\ \tau \in [1,2)T_c, & \Pr(H_1) &= \Pr(\beta=1) = 1/4 \\ H_2:&\ \tau \in [2,3)T_c, & \Pr(H_2) &= \Pr(\beta=2) = 1/4 \\ H_3:&\ \tau \in [3,4)T_c, & \Pr(H_3) &= \Pr(\beta=3) = 1/4 \end{aligned} \quad (3.9)$$

Owing to η_m, y_m is also Gaussian random variable with mean $\sqrt{2P}T_c s_m$ and variance $\sigma_m^2 = nN_0T_c$, i.e., its probability density function (pdf) is given by



(a) N=63 (b) N=255



(c) N=1023

Figure 3.4. Correlation coefficient of SelDLL versus q at specific n and N.

$$f_m\left(y_m \middle| H_\beta, q, n, N\right) = \frac{1}{\sqrt{2\pi}\sigma_m} e^{\frac{-\left(y_m - \sqrt{2P}T_c s_m\right)^2}{2\sigma_m^2}}. \tag{3.10}$$

The correlation between branches must be examined. As shown in Appendix C, the correlation coefficient $\rho_{m,k}$ between y_m and y_k, $m \neq k$, is approximated as

$$\tilde{\rho}_{m,k} = \frac{-L + 3\left[\sqrt{\left(1 - \frac{n'}{N}\right)n'}\right] - \frac{n'}{N}}{n}, \quad \text{for } m \neq k. \tag{3.11}$$

As n and N becomes large, the value of $\tilde{\rho}_{m,k}$ can be approximated as zero. This approximation is also confirmed by numerical results shown in Figures 3.4 and 3.5. Figure 3.4 plots the correlation coefficient $\rho_{m,k}(q,n,N)$ for various m and k, defined in (B.1), versus q for specific n and N. Moreover, Figure 3.5 shows the average over q of the absolute value of the correlation coefficient, $\left|\rho_{m,k}(n,N)\right|_{ave}$ with n being a parameter for specific N, where $\left|\rho_{m,k}(n,N)\right|_{ave}$ is defined as

$$\left|\rho_{m,k}(n,N)\right|_{ave} = \frac{\sum_{q=0}^{N-1}\left|\rho_{mk}(q,n,N)\right|}{N} \tag{3.12}$$

and $|\cdot|$ is the absolute operator. As seen, the correlation coefficient is small for all possible pairs of m and k when n and N are large. Since y_m and y_k, $m \neq k$, are jointly Gaussian,

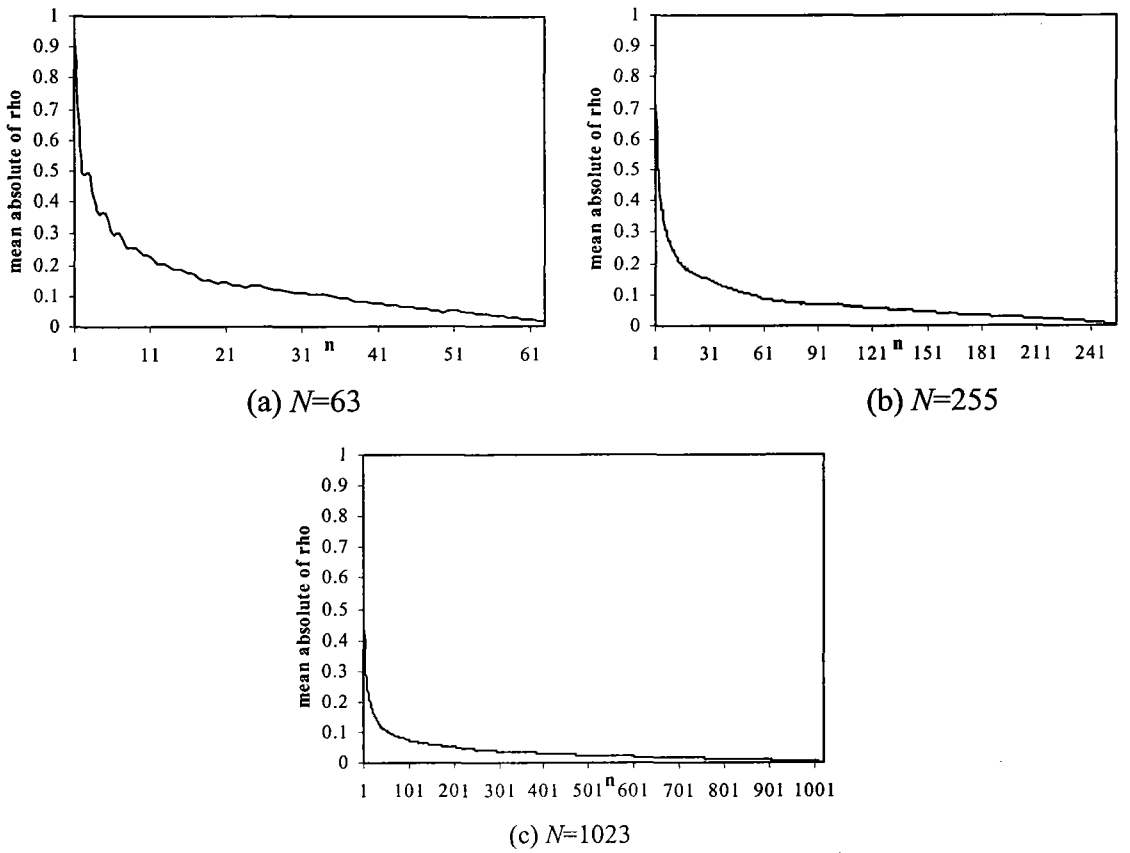


Figure 3.5. Mean absolute correlation coefficient of SelDLL versus n at specific N.

they can consequently be assumed to be independent. Therefore, the joint pdf can be completely expressed as the product of the marginal pdf's, i.e.,

$$f(y_0,y_1,y_2,y_3,y_4|\psi)=f_0(y_0|\psi)\cdot f_1(y_1|\psi)\cdot f_2(y_2|\psi)\cdot f_3(y_3|\psi)\cdot f_4(y_4|\psi), \quad (3.13)$$

where ψ is a given event. Because of (3.3) and (3.10), $f_m(y_m|H_\beta,q,n,N)$ may be written as $f_m(y_m|\beta,\gamma,q,n,N)$.

Given H_β, the correct branch selection is achieved if the β^{th} and $(\beta+1)^{\text{th}}$ branches are selected, which occurs if the corresponding correlated outputs y_β and $y_{\beta+1}$ are larger than the others. As described in Appendix D, the probability of correct selection given H_β is

$$\Pr(Y_\beta,Y_{\beta+1}>Y_j,Y_k,Y_l|\beta,\gamma,q,n,N)=$$
$$\int_{-\infty}^{\infty}\int_{-\infty}^{y_\beta}\int_{-\infty}^{y_{\beta+1}}\int_{-\infty}^{y_{\beta+1}}\int_{-\infty}^{y_{\beta+1}} f(y_\beta,y_{\beta+1},y_j,y_k,y_l|\beta,\gamma,q,n,N)dy_l dy_k dy_j dy_{\beta+1} dy_\beta$$
$$+\int_{-\infty}^{\infty}\int_{y_\beta}^{\infty}\int_{-\infty}^{y_\beta}\int_{-\infty}^{y_\beta}\int_{-\infty}^{y_\beta} f(y_\beta,y_{\beta+1},y_j,y_k,y_l|\beta,\gamma,q,n,N)dy_l dy_k dy_j dy_{\beta+1} dy_\beta . \quad (3.14)$$

Here, β, $\beta+1$, j, k, and l take different values. The complete expression for each given H_β is also presented in Appendix D, given by (D.4)-(D.7) for H_0, H_1, H_2, and H_3 respectively. From (3.4), (3.6), (3.9), (3.10), (D.4)-(D.7), the probability of correct selection, averaged over β, γ, and q, can be expressed as

$$P_{ave}\left(\text{correct selection}\,|n,N\right)=$$

$$\sum_{\beta=0}^{3}\int_{0}^{1}\sum_{q=0}^{N-1}\frac{1}{N}\Pr\left(H_\beta\right)\Pr\left(Y_\beta\ \&\ Y_{\beta+1}>Y_j,Y_k,Y_l\,|\beta,\gamma,q,n,N\right)d\gamma\,. \qquad (3.15)$$

3.2 Parallel Early-Late DLL (PelDLL)

In this modified scheme, three branches are used to detect the incoming code phase and only one is selected to perform further synchronization as shown in Figure 3.2. Owing to one branch selection, there is a probability of selection loss, but a better performance is obtained compared to SelDLL.

In Figure 3.2(a), the correlation outputs are compared and only the branch possessing the maximum absolute value is selected to further produce the error signal ε by closing the switch corresponding to such branch. Therefore, the value of $\hat{\tau}$, initially set to zero, is obtained from this error signal. The selected branch will be used to adjust the local code phase until synchronization is reached. However, if the system realizes that the tracking can not be achieved, the process described above will be repeated again.

Notice that to obtain the discriminator characteristic shown in Figure 3.2(b), the local code used to correlate in each branch is the summation of two PN sequences with different shifted phases. Unfortunately, it results in an increase of noise variance in the branch.

In the m^{th} branch, the received signal is correlated by $c_m^{pel}(t)$, which is an early-late PN signal, shifted by $2mT_c$, i.e.,

$$c_m^{pel}(t)=c\left(t-2mT_c-\delta T_c\right)-c\left(t-2mT_c+\delta T_c\right) \qquad (3.16)$$

where $m=0,\ 1,\ 2$, and δT_c is a half of early-late spacing, with $\delta\in(0,\ 0.5]$. The output of the m^{th} integrate-and-dump circuit is

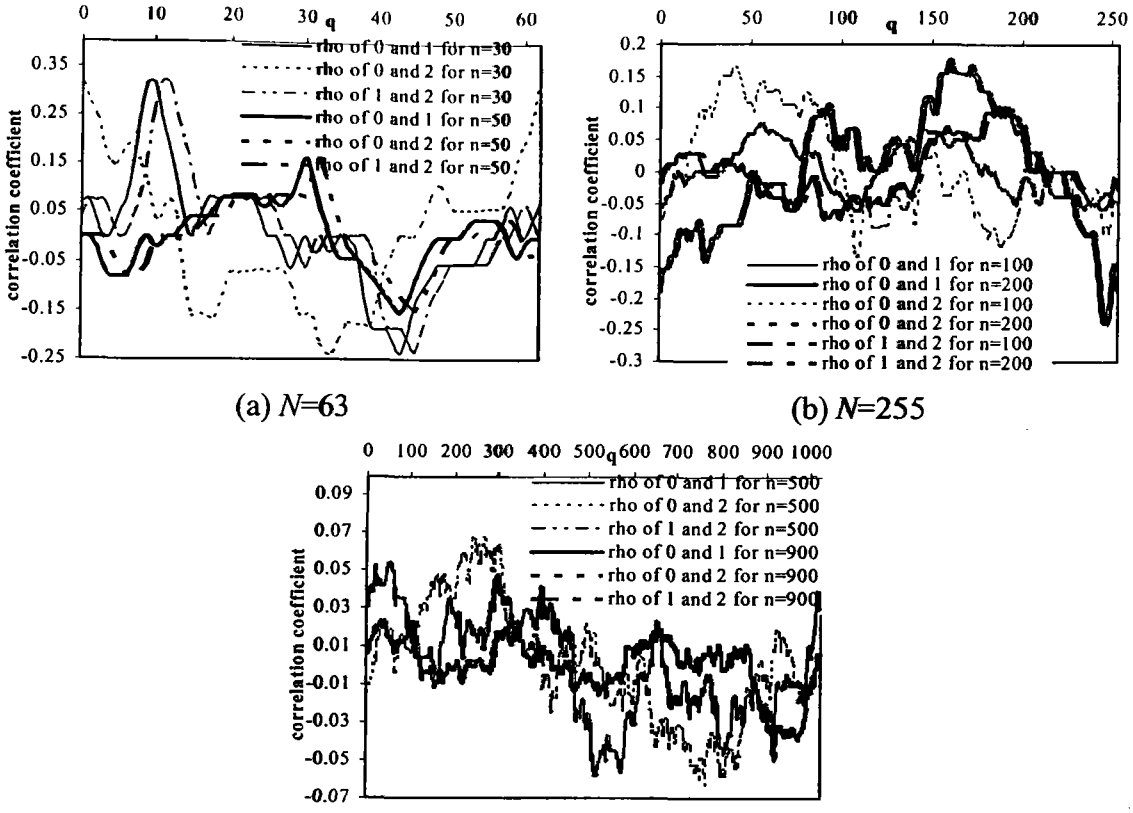
$$y_m^{pel}=\int_{(q-n)T_c}^{qT_c}\text{Re}\left\{\left[\sqrt{2P}c\left(t-\tau\right)+z\left(t\right)\right]c_m^{pel}\left(t\right)\right\}dt=\sqrt{2P}T_c s_m^{pel}+\eta_m^{pel} \qquad (3.17)$$

where s_m^{pel} and η_m^{pel} are the signal and noise terms, given by

$$\begin{aligned}s_m^{pel}=\sum_{i=0}^{n-1}\Bigg\{&\left(\frac{\delta+\gamma}{2}-\frac{|\delta-\gamma|}{2}\right)c_{q-n+i-2m-1}c_{q-n+i-\beta-1}+\left(\frac{\delta-\gamma}{2}+\frac{|\delta-\gamma|}{2}\right)c_{q-n+i-2m-1}c_{q-n+i-\beta}\\&+\left(\frac{|(1-\delta)-\gamma|}{2}+\frac{|\gamma-\delta|}{2}-\frac{1}{2}\right)c_{q-n+i-2m}c_{q-n+i-\beta-1}+\left(\frac{1}{2}-\frac{|\delta-\gamma|}{2}-\frac{|(1-\delta)-\gamma|}{2}\right)c_{q-n+i-2m}c_{q-n+i-\beta}\\&-\left(\frac{\gamma-(1-\delta)}{2}+\frac{|\gamma-(1-\delta)|}{2}\right)c_{q-n+i-2m+1}c_{q-n+i-\beta-1}\\&-\left(\frac{1+\delta-\gamma}{2}-\frac{|(1-\delta)-\gamma|}{2}\right)c_{q-n+i-2m+1}c_{q-n+i-\beta}\Bigg\},\end{aligned} \qquad (3.18)$$

$$\eta_m^{pel}=\int_{(q-n)T_c}^{qT_c}\text{Re}\left\{z\left(t\right)\right\}c_m^{pel}\left(t\right)dt\,. \qquad (3.19)$$

The value of s_m^{pel} depends on β, γ, q, n, N, and δ, while η_m^{pel} is a zero mean Gaussian random variable with variance



(a) N=63 (b) N=255

(c) N=1023

Figure 3.6. Correlation coefficient of PelDLL versus q at specific n and N.

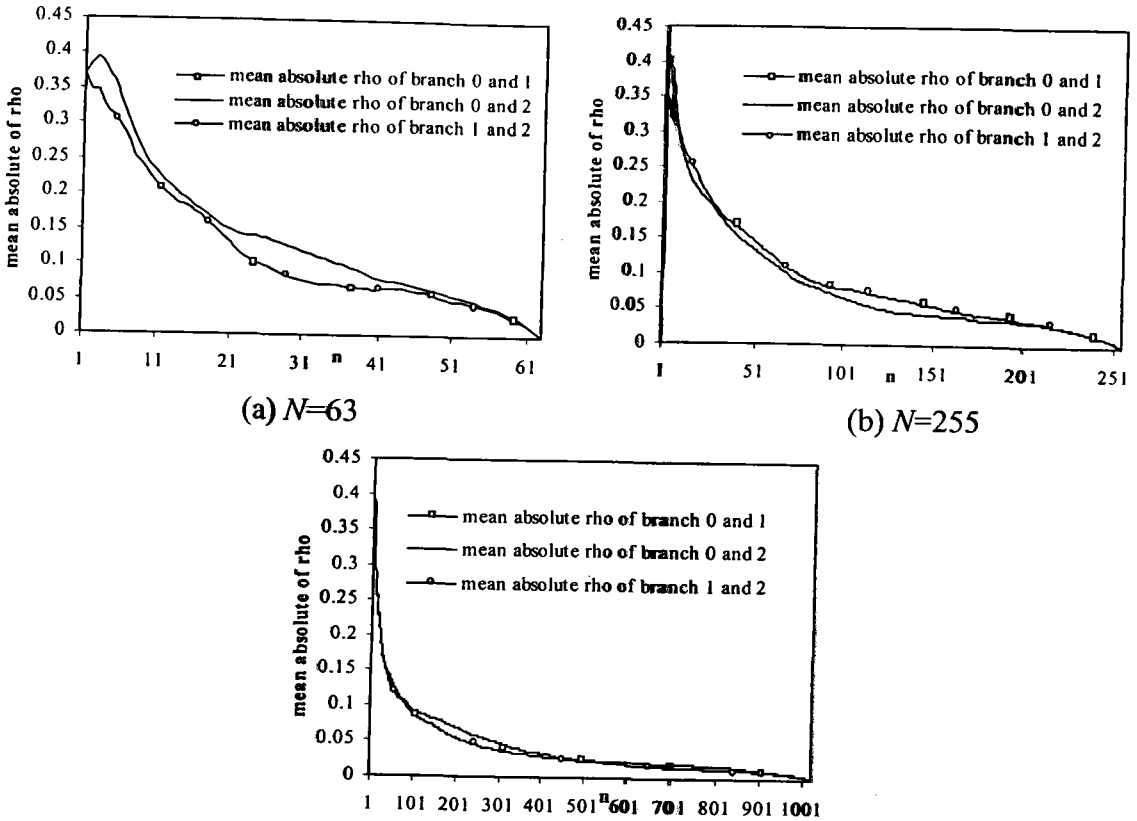
$$\sigma^2_{pel,m} = \delta N_0 T_c \left[4n - 2\sum_{i=0}^{n-1} \left(c_{q-n+i-2m-1} c_{q-n+i-2m} + c_{q-n+i-2m} c_{q-n+i-2m+1} \right) \right]. \tag{3.20}$$

The derivations of (3.18) and (3.20) are given in Appendix E and F, respectively.

In a noise-free situation, $S(\tau)$ of this scheme before selection is performed can be displayed as three parallel early-late autocorrelation functions centered at $0T_c$, $2T_c$, and $4T_c$ as shown on Figure 3.2(b). Among the three correlator branches, the branch possessing the maximum absolute value is selected for tracking the incoming code phase. Now $S(\tau)$ is only one of the three early-late autocorrelation functions, which is shown in Figure 3.2(c) for the case that branch 1 is selected as an example.

The probability of correct selection can be calculated as follows. The correlated value, y_m^{pel}, is a Gaussian random variable with mean $\sqrt{2PT_c}s_m^{pel}$ and variance specified in (3.20). As shown in Appendix F, the correlation coefficient $\rho_{m,k}^{pel}$ between y_m^{pel} and y_k^{pel} of the PelDLL scheme can be expressed as

$$\rho_{m,k}^{pel} = \frac{\sum_{i=q-n}^{q-1} \left[\left(c_{i-2m-1} - c_{i-2m} \right)\left(c_{i-2k-1} - c_{i-2k} \right) + \left(c_{i-2m} - c_{i-2m+1} \right)\left(c_{i-2k} - c_{i-2k+1} \right) \right]}{\sqrt{\left[4n - 2\sum_{j=q-n}^{q-1} \left(c_{j-2m-1} c_{j-2m} + c_{j-2m} c_{j-2m+1} \right) \right] \left[4n - 2\sum_{v=q-n}^{q-1} \left(c_{v-2k-1} c_{v-2k} + c_{v-2k} c_{v-2k+1} \right) \right]}}. \tag{3.21}$$



(a) N=63 (b) N=255

(c) N=1023

Figure 3.7. Mean absolute correlation coefficient of PelDLL versus n at specific N.

We may approximate the jointly Gaussian random variables y_0^{pel}, y_1^{pel}, and y_2^{pel} as independent random variables when n is sufficiently large, which is confirmed by numerical results shown in Figure 3.6 and 3.7. Figure 3.6 plots the correlation coefficient of j and k, $\rho_{j,k}^{pel}(q,n,N)$, defined in (3.21) for specific n and N with q being a parameters. In addition, Figure 3.7 shows the average (over q) of the absolute value of the correlation coefficient, $\left|\rho_{j,k}^{pel}(n,N)\right|_{ave}$, versus n for specific N, where $\left|\rho_{j,k}^{pel}(n,N)\right|_{ave}$ is defined as

$$\left|\rho_{j,k}^{pel}(n,N)\right|_{ave} = \frac{\sum_{q=0}^{N-1}\left|\rho_{j,k}^{pel}(q,n,N)\right|}{N}. \tag{3.22}$$

As seen, the correlation coefficient is small when n is large enough. Therefore, the joint pdf can be approximated by the product of individual marginal pdf's, i.e.,

$$f\left(y_0^{pel}, y_1^{pel}, y_2^{pel}|\psi\right) = f_0\left(y_0^{pel}|\psi\right)\cdot f_1\left(y_1^{pel}|\psi\right)\cdot f_2\left(y_2^{pel}|\psi\right). \tag{3.23}$$

Where ψ is some set of parameters. The marginal pdf of y_m^{pel} is

$$f_m\left(y_m^{pel}|\beta,\gamma,q,\delta,n,N\right) = \frac{1}{\sqrt{2\pi}\sigma_{pel,m}} e^{\frac{-\left(y_m^{pel}-\sqrt{2P}T_c s_m^{pel}\right)^2}{2\sigma_{pel,m}^2}}. \tag{3.24}$$

Given hypothesis H_0, H_1, H_2, and H_3, defined as (3.9), the correct branch is 0, 1, 1, and 2, respectively. Therefore, the probability of correct branch selection under H_0 is

$$\Pr\left(\left|Y_0^{pel}\right| > \left|Y_1^{pel}\right|, \left|Y_2^{pel}\right| \middle| 0,\gamma,q,\delta,n,N\right) =$$

$$\int_{-\infty}^{\infty}\int_{-y_0^{pel}}^{y_0^{pel}}\int_{-y_0^{pel}}^{y_0^{pel}} f\left(y_0^{pel},y_1^{pel},y_2^{pel}\middle|0,\gamma,q,\delta,n,N\right)dy_2^{pel}dy_1^{pel}dy_0^{pel} \quad (3.25)$$

The average probability of correct branch selection is obtained by averaging (3.25) over q and γ, i.e.,

$$\Pr\left(\left|Y_0^{pel}\right|>\left|Y_1^{pel}\right|,\left|Y_2^{pel}\right|\middle|0,\delta,n,N\right)=\int_0^1\sum_{q=0}^{N-1}\frac{1}{N}\Pr\left(\left|Y_0^{pel}\right|>\left|Y_1^{pel}\right|,\left|Y_2^{pel}\right|\middle|0,\gamma,q,\delta,n,N\right)d\gamma. \quad (3.26)$$

Similar results under H_1, H_2, and H_3 can be obtained, as presented in Appendix G. After averaging over H_β, β=0, 1, 2, 3, we obtain the probability of correct selection with δ, n, N being the parameters:

$$\begin{aligned}P_{ave}^{pel}(\text{correct selection}|\delta,n,N)&=\Pr(H_0)\Pr\left(\left|Y_0^{pel}\right|>\left|Y_1^{pel}\right|,\left|Y_2^{pel}\right|\middle|0,\delta,n,N\right)\\&+\Pr(H_1)\Pr\left(\left|Y_1^{pel}\right|>\left|Y_0^{pel}\right|,\left|Y_2^{pel}\right|\middle|1,\delta,n,N\right)+\Pr(H_2)\Pr\left(\left|Y_1^{pel}\right|>\left|Y_0^{pel}\right|,\left|Y_2^{pel}\right|\middle|2,\delta,n,N\right)\\&+\Pr(H_3)\Pr\left(\left|Y_2^{pel}\right|>\left|Y_0^{pel}\right|,\left|Y_1^{pel}\right|\middle|3,\delta,n,N\right). \quad (3.27)\end{aligned}$$

3.3 Parallel Extended Range DLL (PerDLL)

As observed in the previous section that the PelDLL scheme's branch selection algorithm may not operate properly in the event that the initial code phase delay is in the neighborhood of $0T_c$, $2T_c$, and $4T_c$ which are the loop's locking points. This is because around these values the correlated output of the appropriate branch is close to zero. Due to the maximum absolute value selection algorithm, selection loss is induced. Therefore, a new scheme called PerDLL is proposed to reduce such effect. The idea is described below.

The structure of PerDLL and its corresponding discriminator characteristic are presented in Figure 3.3. First, the incoming signal in the m^{th} branch, m=0, 1, 2, is subtracted by $\sqrt{2\hat{P}}c(t-2mT_c)$, where $\hat{P}$ is the estimated power of the received signal, in order to provide a zero correlation output at $\tau=2mT_c$. Then, the result in the m^{th} branch is correlated with $c_m^{per}(t-\hat{\tau})=c_m^{main}(t-\hat{\tau})+c_m^{ext}(t-\hat{\tau})$. The correlation results are input to the selection algorithm. The branch possessing the minimum absolute correlation value is selected for tracking. Let m be the label of the selected branch. After branch selection, the local sequence is chosen as $c_m^{main}(t-\hat{\tau})$. The tracking process after the selection uses only the selected branch to adjust the local code phase. However, if the system realizes that the tracking can not be achieved, the process described above will be repeated again.

The received signal, $r(t)$, is correlated with a bank of three parallel extended range local PN sequences, $c_m^{per}(t)$, m=0, 1, 2 , defined as

$$\begin{aligned}c_0^{per}(t)&=\left[c_0^{main}(t)\right]+\left[c_0^{ext}(t)\right]\\&=\left[c(t-T_c)-c(t+T_c)\right]+\left[2c(t-2T_c)+2c(t-3T_c)+2c(t-4T_c)\right]\\&=\sum_{i=-\infty}^{\infty}c_{0,i}^{per}P_{T_c}(t-iT_c)=\sum_{i=-\infty}^{\infty}(c_{i-1}-c_{i+1}+2c_{i-2}+2c_{i-3}+2c_{i-4})P_{T_c}(t-iT_c), \quad (3.28)\\c_1^{per}(t)&=\left[c_1^{main}(t)\right]+\left[c_1^{ext}(t)\right]\end{aligned}$$

$$= \left[c(t-3T_c) - c(t-T_c)\right] + \left[2c(t-4T_c) - 2c(t)\right]$$

$$= \sum_{i=-\infty}^{\infty} c_{1,i}^{per} P_{T_c}(t-iT_c) = \sum_{i=-\infty}^{\infty} (c_{i-3} - c_{i-1} + 2c_{i-4} - 2c_i) P_{T_c}(t-iT_c), \quad (3.29)$$

$$c_2^{per}(t) = \left[c_2^{main}(t)\right] + \left[c_2^{ext}(t)\right]$$

$$= \left[c(t-5T_c) - c(t-3T_c)\right] + \left[-2c(t-2T_c) - 2c(t-T_c) - 2c(t)\right]$$

$$= \sum_{i=-\infty}^{\infty} c_{2,i}^{per} P_{T_c}(t-iT_c) = \sum_{i=-\infty}^{\infty} (c_{i-5} - c_{i-3} - 2c_{i-2} - 2c_{i-1} - 2c_i) P_{T_c}(t-iT_c). \quad (3.30)$$

Let the initial value of $\hat{\tau}$ be zero. The output of the m^{th} integrate-and-dump circuit is

$$y_m^{per} = \int_{(q-n)T_c}^{qT_c} \mathrm{Re}\left\{\left[\sqrt{2P}c(t-\tau) - \sqrt{2\hat{P}}c(t-2mT_c) + z(t)\right] c_m^{per}(t)\right\} dt$$

$$= \sqrt{2P}T_c s_m^{per} - \sqrt{2\hat{P}}T_c B_m^{per} + \eta_m^{per} \quad (3.31)$$

where s_m^{per}, B_m^{per}, and η_m^{per} are signal, biased signal, and noise term, given by

$$s_0^{per} = \sum_{i=0}^{n-1} \left\{\left[\gamma c_{q-n+i-\beta-1} + (1-\gamma) c_{q-n+i-\beta}\right] \times \left(c_{q-n+i-1} - c_{q-n+i+1} + 2c_{q-n+i-2} + 2c_{q-n+i-3} + 2c_{q-n+i-4}\right)\right\}, \quad (3.32)$$

$$s_1^{per} = \sum_{i=0}^{n-1} \left\{\left[\gamma c_{q-n+i-\beta-1} + (1-\gamma) c_{q-n+i-\beta}\right] \times \left(c_{q-n+i-3} - c_{q-n+i-1} + 2c_{q-n+i-4} - 2c_{q-n+i}\right)\right\}, \quad (3.33)$$

$$s_2^{per} = \gamma \sum_{i=0}^{n-1} \left\{\left[\gamma c_{q-n+i-\beta-1} + (1-\gamma) c_{q-n+i-\beta}\right] \times \left(c_{q-n+i-5} - c_{q-n+i-3} - 2c_{q-n+i-2} - 2c_{q-n+i-1} - 2c_{q-n+i}\right)\right\}, \quad (3.34)$$

$$B_0^{per} = \sum_{i=0}^{n-1} c_{q-n+i} \left(c_{q-n+i-1} - c_{q-n+i+1} + 2c_{q-n+i-2} + 2c_{q-n+i-3} + 2c_{q-n+i-4}\right), \quad (3.35)$$

$$B_1^{per} = \sum_{i=0}^{n-1} c_{q-n+i-2} \left(c_{q-n+i-3} - c_{q-n+i-1} + 2c_{q-n+i-4} - 2c_{q-n+i}\right), \quad (3.36)$$

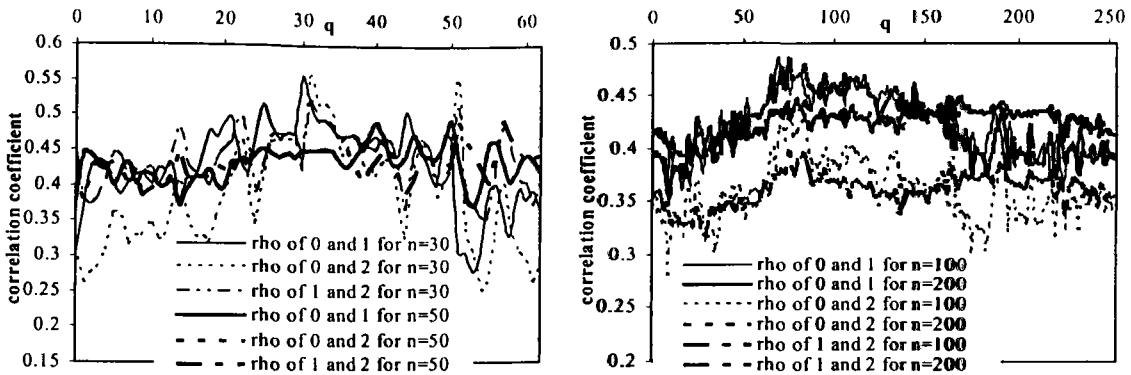
$$B_2^{per} = \sum_{i=0}^{n-1} c_{q-n+i-4} \left(c_{q-n+i-5} - c_{q-n+i-3} - 2c_{q-n+i-2} - 2c_{q-n+i-1} - 2c_{q-n+i}\right), \quad (3.37)$$

$$\eta_m^{per} = \int_{(q-n)T_c}^{qT_c} \mathrm{Re}\left\{z(t)\right\} c_m^{per}(t)\, dt. \quad (3.38)$$

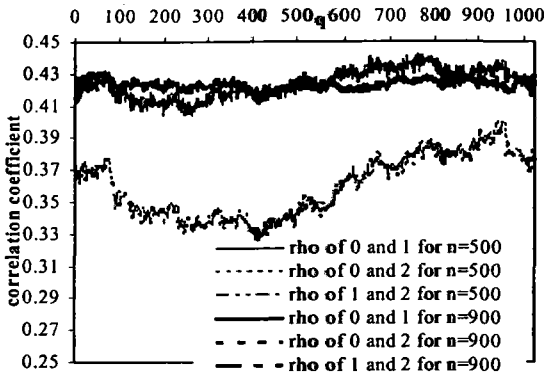
Eq. (3.32)-(3.34) are obtained by substituting $c_{q-n-m+i}$ and $c_{q-n-m+j}$ in (A.4) with $c_{m,q-n+i}^{per}$ and $c_{m,q-n+j}^{per}$, defined in (3.28)-(3.30), for m=0, 1, 2, respectively. The value of s_m^{per} depends on m, γ, n, N, while η_m^{per} is a zero mean Gaussian random variable with variance

$$\sigma_{per,0}^2 = N_0 T_c \sum_{i=0}^{n-1} \left[c_{q-n+i-1} - c_{q-n+i+1} + 2c_{q-n+i-2} + 2c_{q-n+i-3} + 2c_{q-n+i-4}\right]^2, \quad (3.39)$$

$$\sigma_{per,1}^2 = N_0 T_c \sum_{i=0}^{n-1} \left[c_{q-n+i-3} - c_{q-n+i-1} + 2c_{q-n+i-4} - 2c_{q-n+i}\right]^2, \quad (3.40)$$



(a) N=63 (b) N=255



(c) N=1023

Figure 3.8. Correlation coefficient of PerDLL versus q at specific n and N.

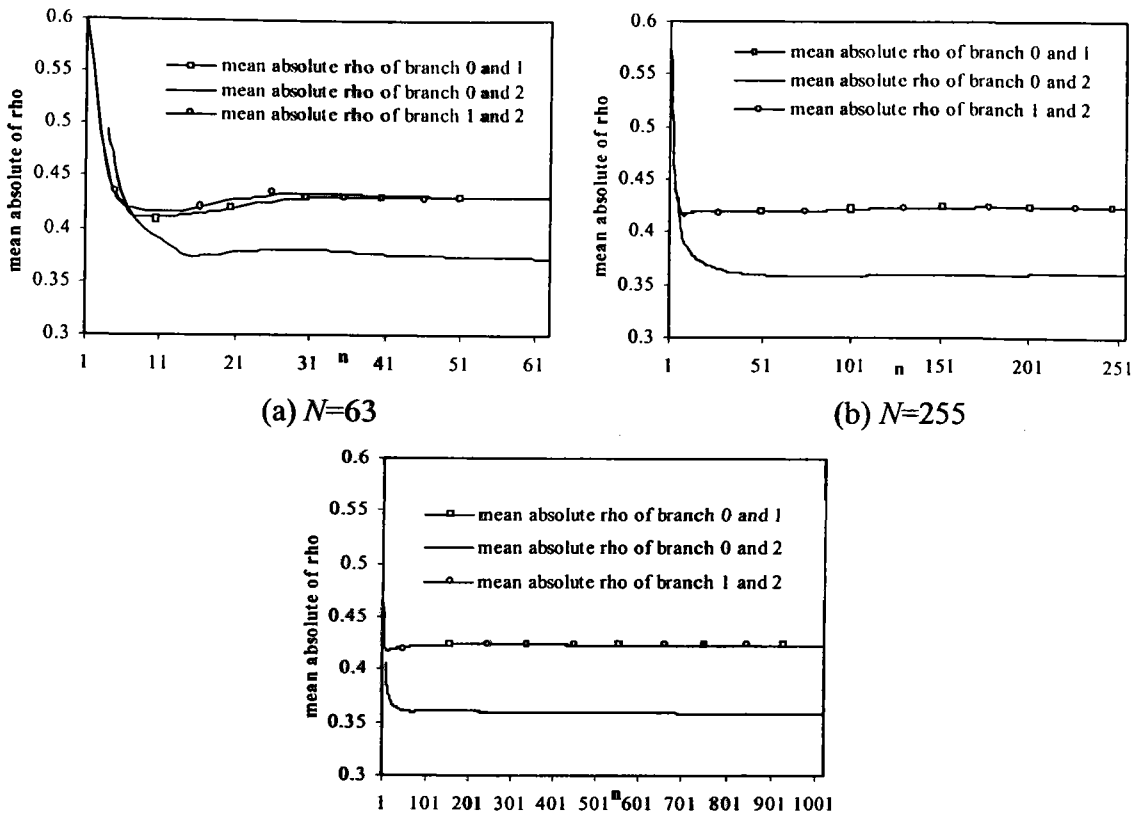
$$\sigma^2_{per,2} = N_0 T_c \sum_{i=0}^{n-1} \left[c_{q-n+i-5} - c_{q-n+i-3} - 2c_{q-n+i-2} - 2c_{q-n+i-1} - 2c_{q-n+i} \right]^2, \tag{3.41}$$

as shown in Appendix H.

In a noise-free situation, the discriminator characteristic $S(\tau)$ of this scheme before branch selection can be displayed as three parallel extended range autocorrelation functions as shown in Figure 3.3(b) with locking points at $0T_c$, $2T_c$, and $4T_c$. Among the three correlator branches, there is one branch possessing the minimum absolute value at the integrate-and-dump output (this is represented as the bold line in Figure 3.3(b)). Such branch is selected for tracking the incoming code phase, resulting in $S(\tau)$ such as the one shown in Figure 3.3(c) for the case that branch 1 is selected.

The probability of selection loss of this scheme can be calculated as follows. Owing to η_m^{per}, y_m^{per} is also a Gaussian random variable. Since each branch has its correlation characteristic overlapping with the others, the random variables y_0^{per}, y_1^{per}, and y_2^{per} are dependent random variables. This is confirmed by the results shown in Figures 3.8 and 3.9. Figure 3.8 plots the correlation coefficient $\rho_{m,k}^{per}(q,n,N)$ between branches m and k,

$$\rho_{m,k}^{per}(q,n,N) = \frac{\sum_{i=q-n}^{q-1} c_{m,i}^{per} c_{k,i}^{per}}{\sqrt{\left[\sum_{j=q-n}^{q-1} \left(c_{m,j}^{per}\right)^2\right]\left[\sum_{v=q-n}^{q-1} \left(c_{k,v}^{per}\right)^2\right]}}, \tag{3.42}$$



(a) N=63 (b) N=255

(c) N=1023

Figure 3.9. Mean absolute correlation coefficient of PerDLL versus n at specific N.

which is defined in (H.1). Figure 3.9 shows the mean of the absolute value of the correlation coefficient, $\left|\rho_{m,k}^{per}(n,N)\right|_{ave}$, defined as

$$\left|\rho_{m,k}^{per}(n,N)\right|_{ave}=\frac{\sum_{q=0}^{N-1}\left|\rho_{m,k}^{per}(q,n,N)\right|}{N}. \tag{3.43}$$

We see from Figures 3.8 and 3.9 that the correlation cannot be neglected. Therefore, the joint pdf of three random variables is [83]

$$f(y_0^{per},y_1^{per},y_2^{per}|\beta,\gamma,q,n,N)=\frac{\exp\left\{-\frac{1}{2}\left(\mathbf{y}^{per}-\mathbf{s}^{per}\right)^{\mathrm{T}}\mathbf{K}^{-1}\left(\mathbf{y}^{per}-\mathbf{s}^{per}\right)\right\}}{(2\pi)^{3/2}|\mathbf{K}|^{1/2}}, \tag{3.44}$$

where $(\bullet)^{\mathrm{T}}$, $(\bullet)^{-1}$, and $|\bullet|$ are a transpose, inverse, and determinant operations, respectively. $\mathbf{y}^{per}$ and $\mathbf{s}^{per}$ are column vectors while $\mathbf{K}$ is the covariance matrix, given by

$$\mathbf{y}^{per}=\left[y_0^{per},y_1^{per},y_2^{per}\right]^{\mathrm{T}}, \tag{3.45}$$

$$\mathbf{s}^{per}=\left[\sqrt{2P}T_c s_0^{per}-\sqrt{2\hat{P}}T_c B_0^{per},\sqrt{2P}T_c s_1^{per}-\sqrt{2\hat{P}}T_c B_1^{per},\sqrt{2P}T_c s_2^{per}-\sqrt{2\hat{P}}T_c B_2^{per}\right]^{\mathrm{T}}, \tag{3.46}$$

$$\mathbf{K}=\begin{bmatrix}\sigma_{per,0}^2 & Cov\left(Y_0^{per},Y_1^{per}\right) & Cov\left(Y_0^{per},Y_2^{per}\right)\\ Cov\left(Y_0^{per},Y_1^{per}\right) & \sigma_{per,1}^2 & Cov\left(Y_1^{per},Y_2^{per}\right)\\ Cov\left(Y_0^{per},Y_2^{per}\right) & Cov\left(Y_1^{per},Y_2^{per}\right) & \sigma_{per,2}^2\end{bmatrix}, \tag{3.47}$$

where $Cov\left(Y_j^{per}, Y_k^{per}\right)$ is the covariance of Y_j^{per} and Y_k^{per} given by

$$Cov\left(Y_0^{per}, Y_1^{per}\right) = N_0 T_c \sum_{i=0}^{n-1}\Big[\left(c_{q-n+i-1} - c_{q-n+i+1} + 2c_{q-n+i-2} + 2c_{q-n+i-3} + 2c_{q-n+i-4}\right) \times\left(c_{q-n+i-3} - c_{q-n+i-1} + 2c_{q-n+i-4} - 2c_{q-n+i}\right)\Big], \quad (3.48)$$

$$Cov\left(Y_0^{per}, Y_2^{per}\right) = N_0 T_c \sum_{i=0}^{n-1}\Big[\left(c_{q-n+i-1} - c_{q-n+i+1} + 2c_{q-n+i-2} + 2c_{q-n+i-3} + 2c_{q-n+i-4}\right) \times\left(c_{q-n+i-5} - c_{q-n+i-3} - 2c_{q-n+i-2} - 2c_{q-n+i-1} - 2c_{q-n+i}\right)\Big], \quad (3.49)$$

$$Cov\left(Y_1^{per}, Y_2^{per}\right) = N_0 T_c \sum_{i=0}^{n-1}\Big[\left(c_{q-n+i-3} - c_{q-n+i-1} + 2c_{q-n+i-4} - 2c_{q-n+i}\right) \times\left(c_{q-n+i-5} - c_{q-n+i-3} - 2c_{q-n+i-2} - 2c_{q-n+i-1} - 2c_{q-n+i}\right)\Big], \quad (3.50)$$

For convenience, we define the normalized covariance matrix as

$$\mathbf{K}_{nor} = \frac{1}{N_0 T_c}\mathbf{K}. \quad (3.51)$$

Given hypothesis H_0, H_1, H_2, and H_3, which are defined by (3.9), the correct branch is 0, 1, 1, and 2, respectively. The probability of correct selection given H_β can be obtained, as derived in Appendix I. Similarly, the probability of correct selection given n and N can be calculated by averaging over H_β, γ, and q

$$P_{ave}^{per}\left(\text{correct selection}\middle|n,N\right) = \int_0^1 \sum_{q=0}^{N-1} \frac{1}{N}\Big[\Pr(H_0)\Pr\left(\left|Y_0^{per}\right| < \left|Y_1^{per}\right|, \left|Y_2^{per}\right| \middle| 0,\gamma,q,n,N\right) + \Pr(H_1)\Pr\left(\left|Y_1^{per}\right| < \left|Y_0^{per}\right|, \left|Y_2^{per}\right| \middle| 1,\gamma,q,n,N\right) + \Pr(H_2)\Pr\left(\left|Y_1^{per}\right| < \left|Y_0^{per}\right|, \left|Y_2^{per}\right| \middle| 2,\gamma,q,n,N\right) + \Pr(H_3)\Pr\left(\left|Y_2^{per}\right| < \left|Y_0^{per}\right|, \left|Y_1^{per}\right| \middle| 3,\gamma,q,n,N\right)\Big]. \quad (3.52)$$

3.4 Performance Evaluation Results

In this section, we show and compare the probability of correct selection under various parameters. Only SelDLL and PelDLL schemes are numerically evaluated. Since numerical evaluation of PerDLL is very time consuming due to the triple integration and joint pdf as shown in Appendix I, the probability of correct selection of PerDLL is evaluated using simulation. An m-sequence with a period of 63 chips is used here.

Figure 3.10 shows the probability of correct selection versus τ at SNR= −5 dB for various n. Analytical results are obtained from (D.4)-(D.7) for SelDLL and (G.1)-(G.4) for PelDLL. Results of SelDLL and PelDLL with different δ are compared. The pattern of probability versus τ is directly due to the discriminator characteristic. All of them peak at the middle between integer values of τ and lowest at integer values of τ. The peak value can be improved by increasing n. However, the lowest value seems to be constant for all n. As seen, PelDLL improves the probability around integer values of τ, compared to SelDLL. With a proper choice of δ, PelDLL is better than SelDLL for all values of τ.

Figures 3.11 and 3.12 show similar results for SNR=−10 and −15 dB, respectively. Note that the probability of correct selection is reduced for lower SNR albeit n is the same since the effective SNR decreases. At SNR=−5 dB, the effective SNR for n=200 is

$10\times\log(200\times0.316)$=18dB, while at SNR=−15dB, the effective SNR for the same n is $10\times\log(200\times0.0316)$=8dB.

The average probabilities of correct selection defined in (3.15) and (3.27) are shown in Figure 3.13. They are plotted versus n. PelDLL always provides better probability than SelDLL. Probabilities are obtained for PelDLL with different δ. The performance with δ=0.1 is worse than the others. There is optimum δ that provides the best average probability, for each n. It is observed that with the optimum δ, the average probability improves over that of the SelDLL by about 0.15, 0.16, 0.17 at SNR=−5, −10, −15 dB, respectively. Note that the performances of PelDLL with δ in the range 0.2 to 0.5 differ only slightly.

The probability of correct estimate of PerDLL scheme versus τ at SNR=−5dB and −15dB, n=63 and 189 is shown in Figure 3.14. We also provide the probability of correct selection of PelDLL with δ=0.3 for comparison. Both are obtained from simulation with 10000 trials. Observe that there are only two values of τ that give the lowest probability of correct selection, i.e. at τ=1 and 3. For high effective SNR, Figures 3.14(a) and (b), the peak values are at τ=0, 2, 4. However, there is only one peak value (at τ=2) for low effective SNR. When compare the probability of correct selection of PerDLL with that of PelDLL, we see that PerDLL is better for range of τ, while PelDLL is better for other values of τ.

Figure 3.15 shows the average probability of correct estimation versus n of both PerDLL and PelDLL. These are obtained by using simulation with 400000 trials. The values at SNR=−5dB and −15dB are plotted. We see that PerDLL provides higher average probability of correct selection than PelDLL when the SNR is −15dB, while PelDLL is better when the SNR is −5dB. This suggests that some algorithm which is the hybrid of PerDLL and PelDLL can outperform both schemes.

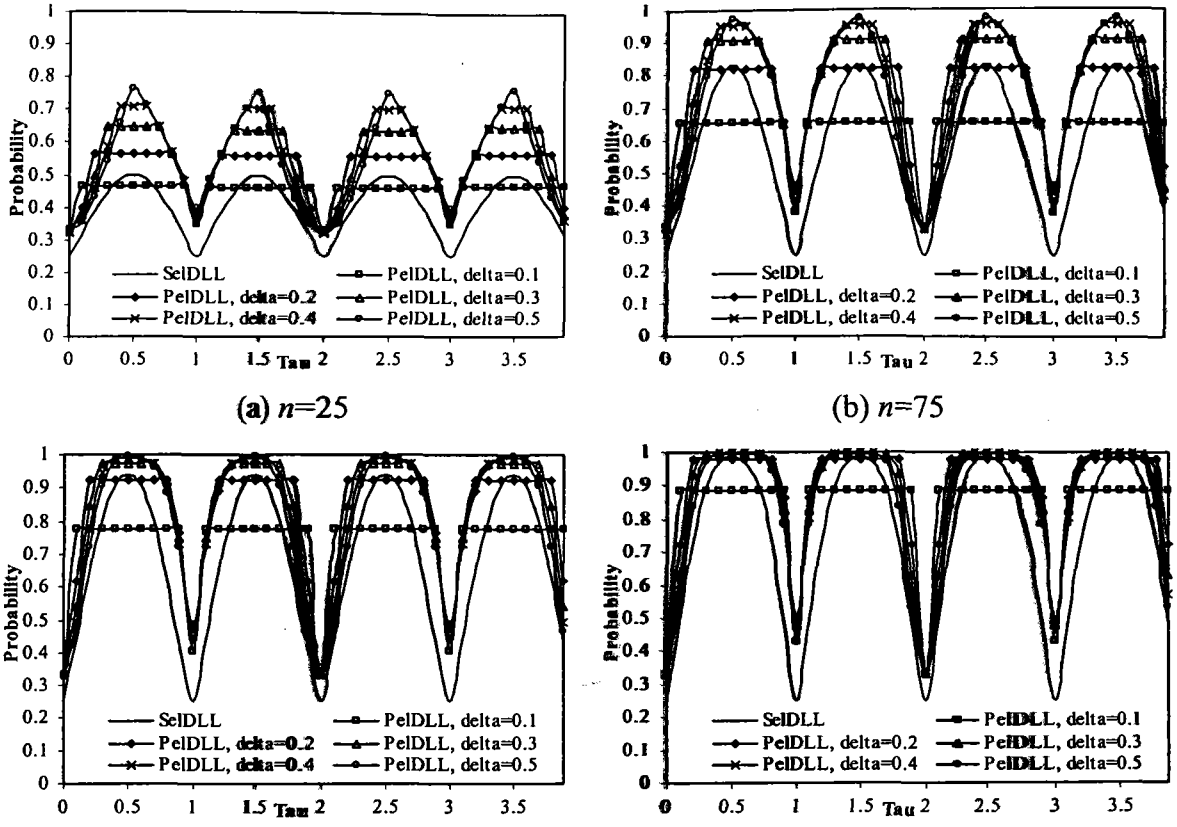
The effect of imperfect estimated SNR (ESNR) on the probability of correct selection of PerDLL scheme is considered in Figure 3.16. These values are obtained by using simulation with 400000 trials. We consider the cases that ESNR is less than the actual SNR by 10%, 20% and ESNR larger than the actual SNR by 10%, 20%. Results show that performance of PerDLL is only slightly affected by the imperfect ESNR. Observe that, in Figure 3.16, PerDLL with ESNR larger than SNR 20% gives the best probability of correct selection, while PerDLL with ESNR less than SNR 20% gives the worst probability of correct selection. However, it may affect the jitter noise.

3.5 Summary

In this chapter, we proposed two schemes of modified extended range DLL by applying two selection algorithms, i.e., PelDLL and PerDLL. The original idea called SelDLL was proposed in [36]. However, after we analyze its probability of correct branch selection, numerical results of SelDLL show the low probability of correct selection due to its selection algorithm and discriminator characteristic, especially at integer values of the incoming code phase. By contrast, our proposed scheme PelDLL provides considerably improvement of the probability of correct decision. The improvement depends on the value of δ. The optimum δ that provides the best probability of correct selection is found between δ=0.2 and 0.5.

For PerDLL, the second proposed scheme, we also show the analysis of its probability of correct branch selection. Since numerical evaluation of analytical expressions is time consuming, we evaluate its performance by using simulation. Results show that when compared with PelDLL, PerDLL provides better probability of correct

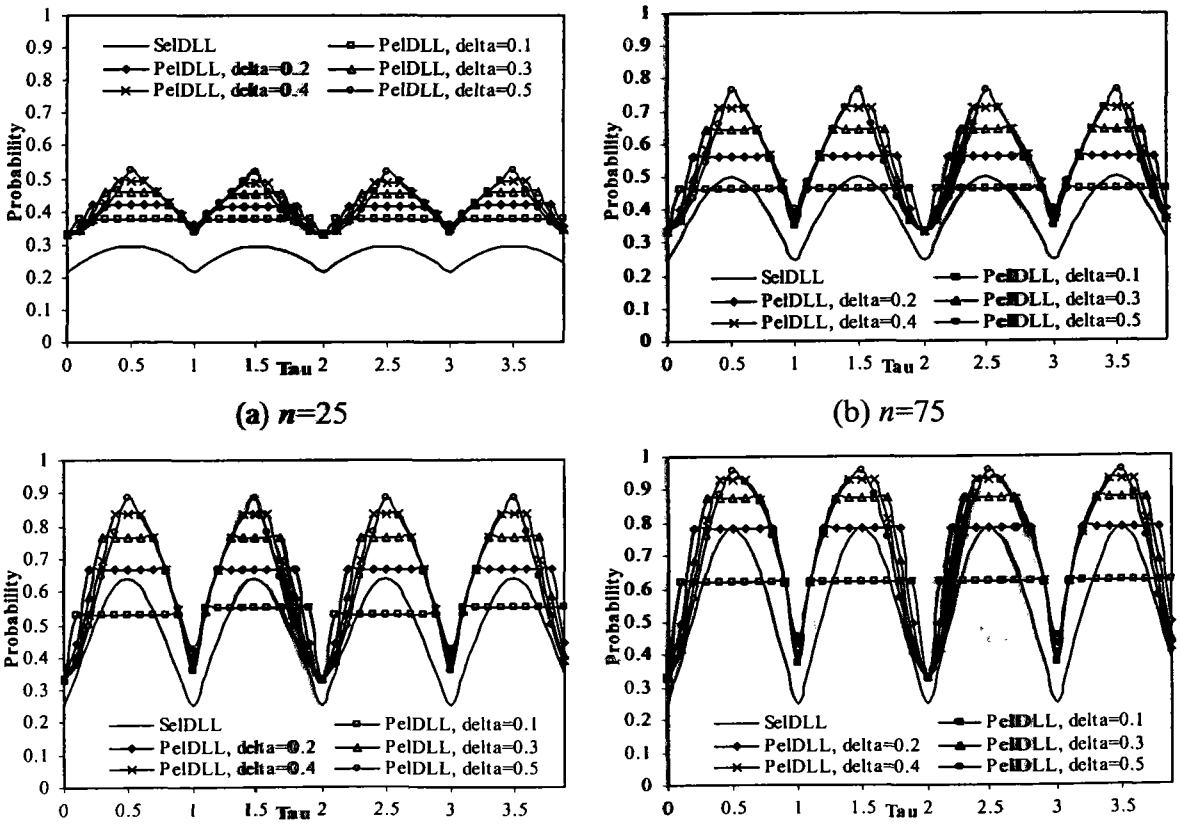
selection at low effective SNR. Furthermore, the probability of correct selection of PerDLL is only slightly affected by imperfect estimation of the SNR.



(a) n=25 (b) n=75

(c) n=125 (d) n=200

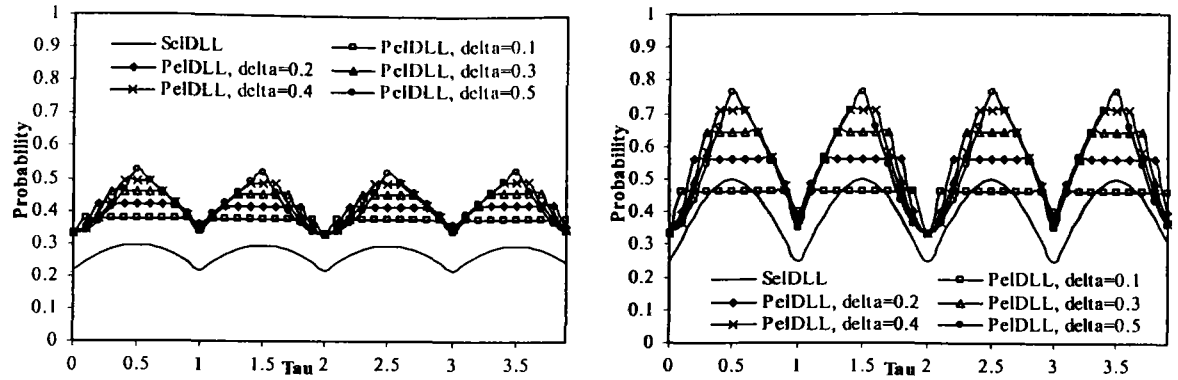
Figure 3.10. Probability of correct selection versus τ at SNR=−5 dB for various n.



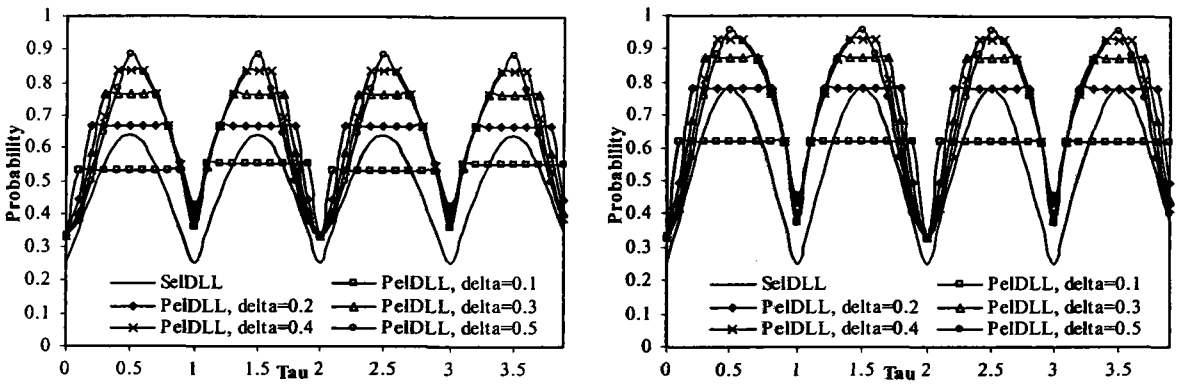
(a) n=25 (b) n=75

(c) n=125 (d) n=200

Figure 3.11. Probability of correct selection versus τ at SNR=−10 dB for various n.

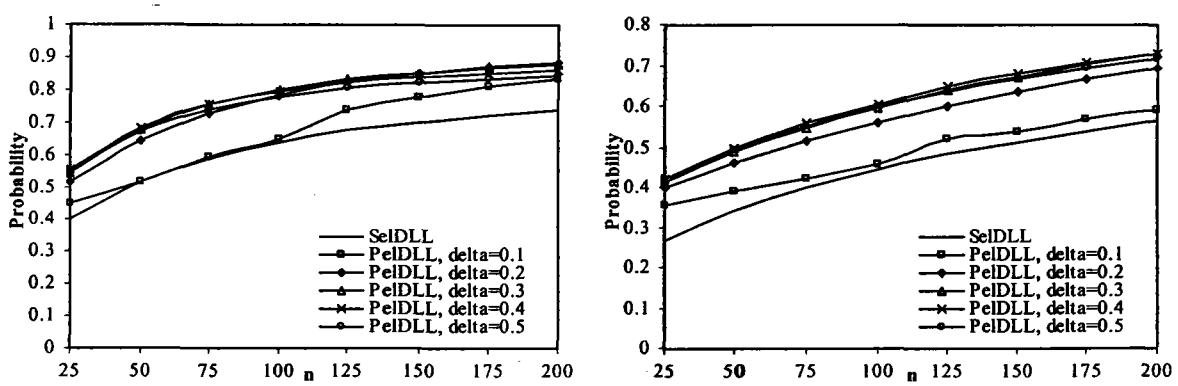


(a) n=25 (b) n=75

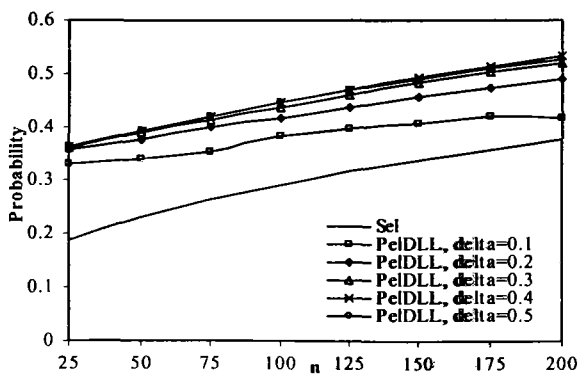


(c) n=125 (d) n=200

Figure 3.12. Probability of correct selection versus τ at SNR=−15 dB for various n.

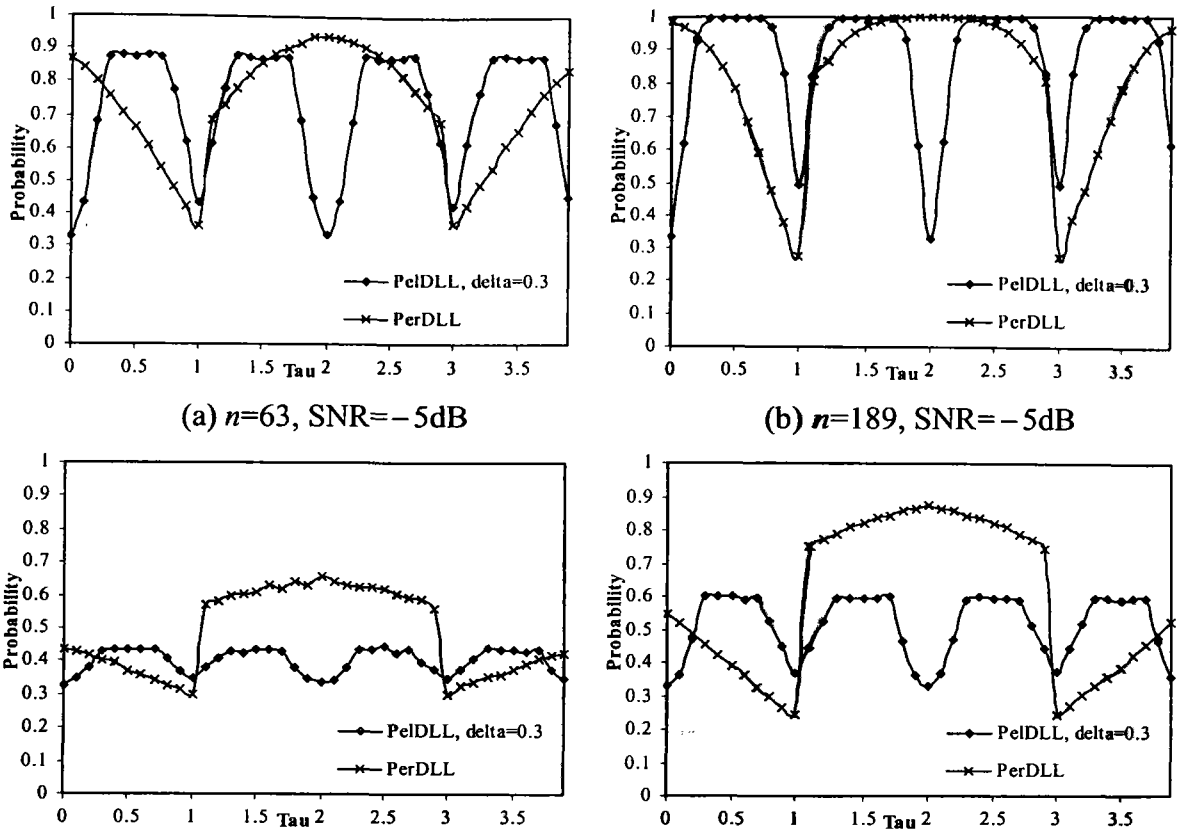


(a) SNR=−5 dB (b) SNR=−10 dB



(c) SNR=−15 dB

Figure 3.13. Average probability of correct selection versus n for various SNR.



(a) n=63, SNR=−5dB (b) n=189, SNR=−5dB

(c) n=63, SNR=−15dB (d) n=189, SNR=−15dB

Figure 3.14. Probability of correct selection of PelDLL and PerDLL versus τ.

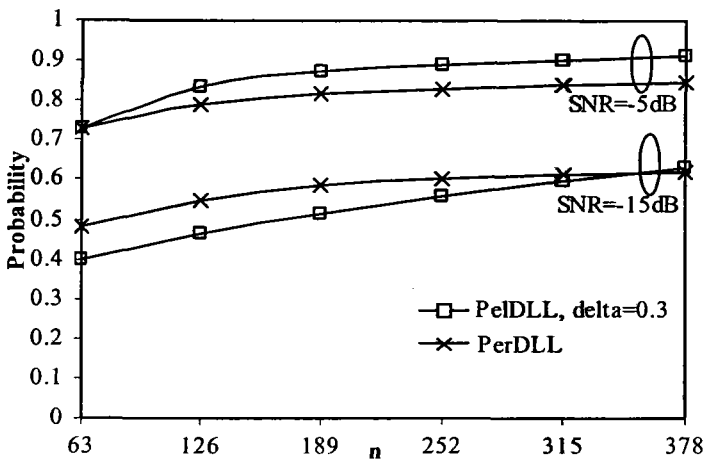
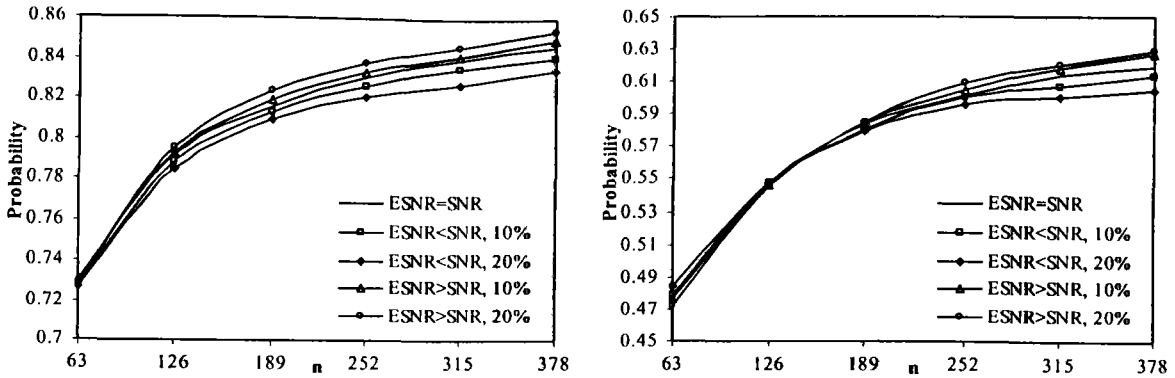


Figure 3.15. Probability of correct selection of PelDLL and PerDLL versus n.



(a) SNR=−5dB (b) SNR=−15dB

Figure 3.16. Effect of imperfect estimated SNR on performance of PerDLL.