Intermediate Report Nr. 9

Bernhard Geiger

July 19, 2009

Contents

1 The Sliding Algorithm

The idea behind the sliding algorithm is already explained in IR7, so I won't go into too much detail again. I just want to mention once again that the resolution of the sliding algorithm is both depending on the integration time $T_i$and the chip period T_C, using the following relationship:

$$T_r = \text{GCD}\left\{T_C, T_i\right\} \tag{1}$$

In fact, the property of the sliding algorithm is only achieved by performing this operation continuously over a set of symbol repetitions. Due to the non-integer fraction between $T_i$and T_C, each symbol contains a non-integer number of integration periods. Thus, the first integration period fully lying in the next symbol is delayed with respect to the real symbol starting instant by the following interval:

$$\Delta = \left((T_{sym})\right)_{T_i}, \tag{2}$$

where T_{sym} is the symbol duration. This shift Δ now leads to the resolution T_r[1]. At least for T_i= 2.25 ns it can be shown that $\Delta = T_r = 0.25$ ns.

Again, the sliding algorithm's output was constructed by upsampling of the ED outputs to the rate of the resolution (i.e. T_r) with an upsampling factor M

$$M = \frac{T_i}{T_r}. \tag{3}$$

As it can be seen from Tab. 1 all relevant values of M are odd[2]. Instead of filtering this upsampled version with an M-tap MA filter (as it was described in IR7), upsampling is just done by inserting $\frac{M-1}{2}$ zeros *before and after* each tap, which is equivalent to representing each integration interval with

[1] ...and without proof (so far) I'm bold to say that $\Delta = T_r$.

[2] Provide a proof for that!

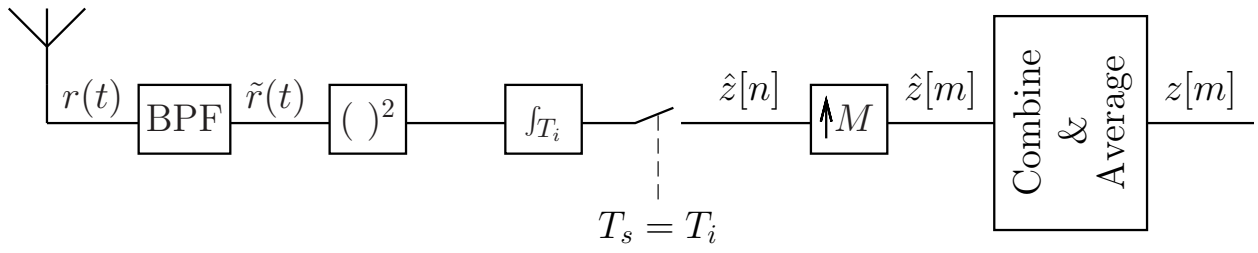


Figure 1: Sliding ED Structure

a weighted pulse placed at the center of it (which is possible since M is odd). Thus, using the notation from Fig. 1 we can say

$$\hat{z}[n] = \int_{nT_i-\frac{T_i}{2}}^{nT_i+\frac{T_i}{2}} \tilde{r}^2(t)dt \quad (4)$$

$$\hat{z}[m] = \hat{z}[nM] \quad (5)$$

This $\hat{z}[m]$ is sparse, i.e. only every m-th element is non-zero. In order to fill the other elements of this vector, we can combine the results from different symbols knowing that they are resulting from delayed versions of the received signal. For n being in an interval so that nT_i is located somewhere in the k-th symbol, we get

$$\hat{z}[n] = \int_{nT_i-\frac{T_i}{2}}^{nT_i+\frac{T_i}{2}} \tilde{r}^2\left(t-k\Delta\right)dt = \int_{nT_i-\frac{T_i}{2}}^{nT_i+\frac{T_i}{2}} \tilde{r}^2\left(t-kT_r\right)dt \quad (6)$$

$$= \int_{nT_i-\frac{T_i}{2}+kT_r}^{nT_i+\frac{T_i}{2}+kT_r} \tilde{r}^2(t)dt \quad (7)$$

If we now, after upsampling, combine these energy blocks in a way that blocks from M consecutive symbols collapse into one symbol in such a way that the samples from the k-th symbol fill the elements $nM+k$, we get the following relationship after combination:

$$z[m] = z[nM+k] = \int_{nT_i-\frac{T_i}{2}+kT_r}^{nT_i+\frac{T_i}{2}+kT_r} \tilde{r}^2(t)dt \quad (8)$$

$$= \int_{nMT_r-\frac{T_i}{2}+kT_r}^{nMT_r+\frac{T_i}{2}+kT_r} \tilde{r}^2(t)dt = \int_{(nM+k)T_r-\frac{T_i}{2}}^{(nM+k)T_r+\frac{T_i}{2}} \tilde{r}^2(t)dt \quad (9)$$

$$= \int_{mT_r-\frac{T_i}{2}}^{mT_r+\frac{T_i}{2}} \tilde{r}^2(t)dt \quad (10)$$

Using that result, which is obtained by sliding over M (or an integer multiple of M) symbols, we can now provide a...

1.1 Proof, that the sliding structure over M symbols is identical to the direct structure with a MA filter

The direct structure as shown in Fig. 2 is operates at an integration and sampling period of $T_r$and provides the ED output samples $\tilde{z}[m]$ according to the formula

$$\tilde{z}[m] = \int_{mT_r-\frac{T_r}{2}}^{mT_r+\frac{T_r}{2}} \tilde{r}^2\left(t-\Delta\right)dt, \quad (11)$$

where Δ is an arbitrary shift of the initial integration instant relative to the sliding structure. Having an M-tap MA filter with an impulse response of $h[m] = \sum_{k=\frac{M-1}{2}}^{\frac{M-1}{2}} \delta[m-k]$, we get the filter output according to

$$z[m] = \sum_{k=\frac{M-1}{2}}^{\frac{M-1}{2}} \tilde{z}[m-k] = \sum_{k=\frac{M-1}{2}}^{\frac{M-1}{2}} \int_{mT_r-\frac{T_r}{2}}^{mT_r+\frac{T_r}{2}} \tilde{r}^2\left(t-\Delta-kT_r\right) dt \tag{12}$$

$$= \sum_{k=\frac{M-1}{2}}^{\frac{M-1}{2}} \int_{(m-k)T_r-\frac{T_r}{2}}^{(m-k)T_r+\frac{T_r}{2}} \tilde{r}^2\left(t-\Delta\right) dt \tag{13}$$

Since these integration intervals are not overlapping and since the upper integration bound for each

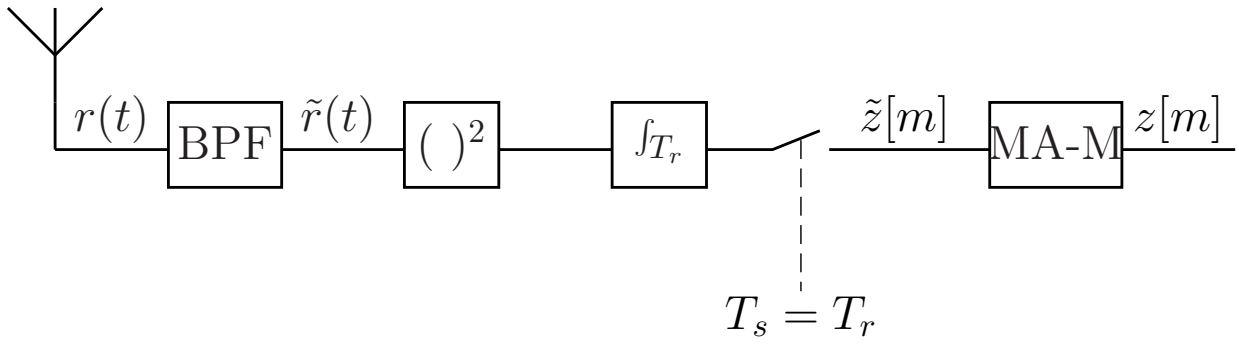


Figure 2: Direct Structure with MA filtering

index of the sum is identical to the lower integration bound of the following index, one can simplify the sum according to

$$z[m] = \int_{(m-(\frac{M-1}{2}))T_r-\frac{T_r}{2}}^{(m-(-\frac{M-1}{2}))T_r+\frac{T_r}{2}} \tilde{r}^2\left(t-\Delta\right) dt = \int_{mT_r-\frac{MT_r}{2}}^{mT_r+\frac{MT_r}{2}} \tilde{r}^2\left(t-\Delta\right) dt \tag{14}$$

$$= \int_{mT_r-\frac{T_i}{2}}^{mT_r+\frac{T_i}{2}} \tilde{r}^2\left(t-\Delta\right) dt \tag{15}$$

Accordingly, the output of the sliding structure is identical to the output of the direct structure followed by an M-tap MA filter if the sliding structure is operated over $N_{sync} = kM$ symbols, with $k \in Z$. A shifting is not necessary ($\Delta = 0$).

If, however, the number of symbol repetitions is not identical to an integer multiple of M, this MA identity does not hold. In fact, it is quite difficult to compute the output after averaging and correlation: The reference symbol for correlation is created on a chip-spaced basis, that is the pulses are located at integer multiples of T_C. Due to arbitrary values of N_{sync} additional peaks in the averaged (and combined) energy blocks are integration period spaced, i.e. located at integer multiples of T_i. Consequently, these peaks are located at different positions within each chip interval, and after correlation they are somewhat *distributed* among the area of an overall integration interval (?)[3].

[3]Maybe I will have a closer look at these distributions, and maybe I can even derive an expression for it, which eventually will help in equalizing even these irregularities.

1.2 Ranging with the sliding algorithm

The problem of the sliding algorithm, applied to ranging, is based on the fact that the number of symbol repetitions has to be an integer multiple of the MA factor M. Thus, for different integration periods $T_i$we get the following resolutions $T_r$and number of required symbol repetitions as shown ín Tab. 1. This now has the consequence, that for a given number of symbol repetitions (as they are specified in the standard) only a portion can be used by the sliding algorithm, if one desires to relate this algorithm to MA filtering[4]. This, in turn reduces the performance in medium SNR regions due to reduced processing gain, which furthermore explains the shifts in the curves in Fig. 3.

T_i	M	T_r	N_{sync}	N_{sync}
ns	-	ns	16	64
1.75	7	0.25	14	63
2.25	9	0.25	9	63
2.5	5	0.5	15	60
2.75	11	0.25	11	55
3	3	1	15	63
3.75	15	0.25	15	60
4.25	17	0.25	–	51
4.5	9	0.5	9	63
4.75	19	0.25	–	57
5	5	1	15	60

Table 1: Number of usable symbol repetitions

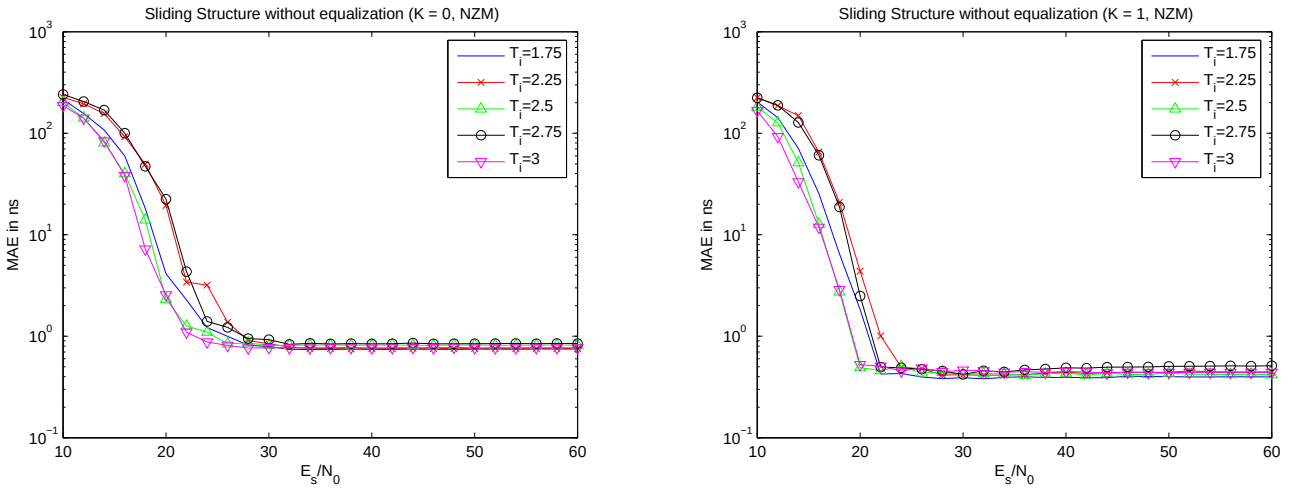
Moreover, it can be seen in the figures that higher integration times outperform shorter integration times, which is explained by the fact that longer integration windows lead to a better noise averaging – the noise standard deviation reduces relative to the signal, and it is therefore easier to detect the signal. In other words, it is less likely that noise-only blocks exceed the threshold optimal for that specific SNR region[5]. This assumption can also be verified by looking at Fig. 7. It can further be seen in these tables, that the minimum mean absolute error is not limited to $\frac{T_i}{4}$.

1.3 Equalization

Ranging Similar to ranging with the parallel algorithm, also ranging with the sliding algorithm can be equalized based on the fact that both algorithms represent some MA filtering. In this case, however, equalization may be a bit more difficult since not only a two-tap MA filter has to be equalized, but an M-tap filter, with M specified by $T_i$according to Tab. 1. Furthermore, since I did not want to design an equalizer with M poles close to the unit circle I decided to automatically design a 40-tap MMSE equalizer, where I set the artificial noise variance to 1% of the signal power. The consequence was, that (at least for the noise free case), equalization improved channel estimation and ranging accuracy significantly, as it can be seen in Figures 5 and 6. Unfortunately, this method tends to oscillate at lower SNR values, which highly decreases performance. For example, as it can be seen in Fig. 4, the

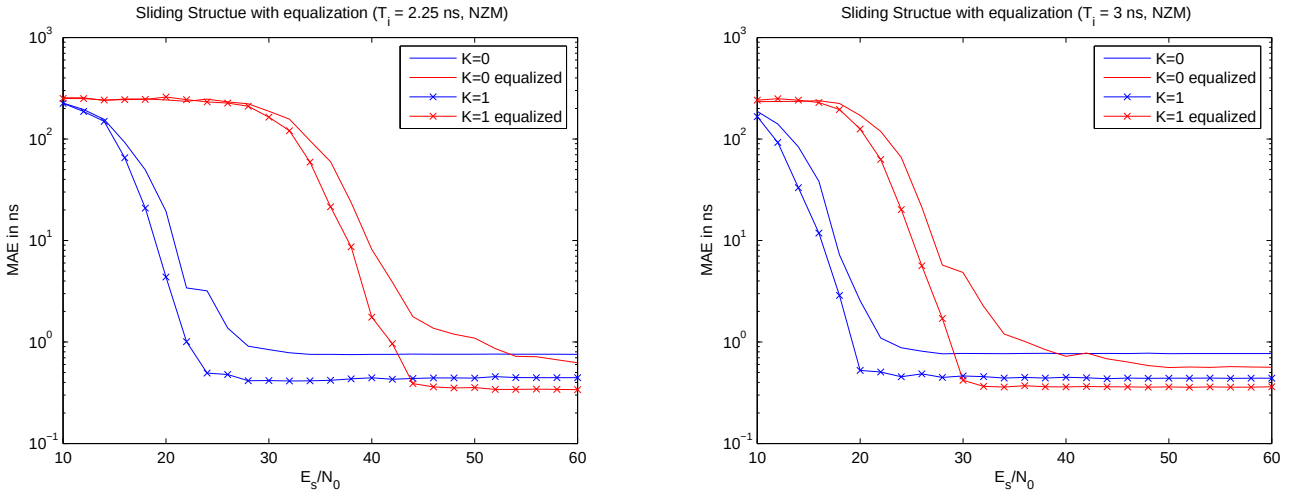
[4]I am not sure, if the use of all possible symbol repetitions could improve performance in terms of ranging accuracy – it is something which might be interesting to analyze in the future.

[5]I also want to mention that these figures cannot be directly compared to the figures from IR7, because these figures were obtained using a different algorithm.



(a) K=0 (b) K=1

Figure 3: Ranging performance of the Sliding Algorithm



(a) (b)

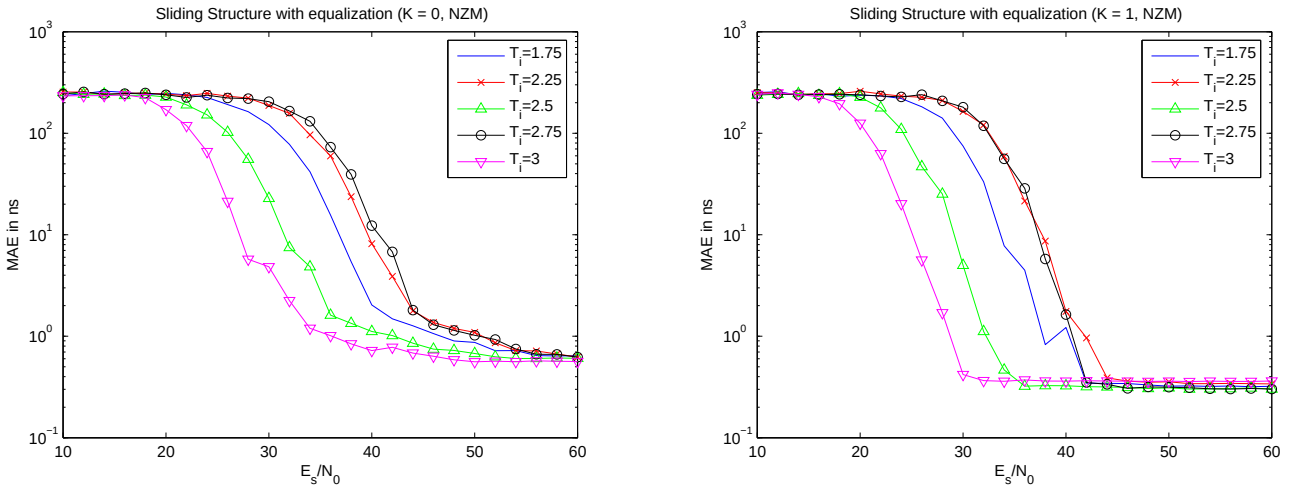
Figure 4: Comparison of pure and equalized sliding algorithm

advantage of equalization over direct sliding implementation can only be exploited at relatively high SNR values.

It might be necessary to take a closer look at the way the MMSE equalizer is designed – both the length of the impulse response and the design noise variance have to be optimized with respect to *both* accuracy and robustness[6]. Another way would be to design a ZF equalizer with a set of poles close to the unit circle. This way, only one design parameter, namely the magnitude of the poles, has to be determined (assuming that all poles lie on a circle). It is within the scope of future work.

Taking a closer look at the influence of the integration interval on the accuracy of the ranging algorithms, one can see in Fig. 6 that the influence is much more prominent for the pure algorithm

[6]For example, a next step could be to relate the design noise variance to the noise floor, i.e. the variance of the noise after correlation and averaging. This way, lower SNR regions would be designed with a higher artificial noise, and consequently the equalizer of constant order resembles less the ZF equalizer (which hopefully reduces oscillations) – I tried that once, but I was not successful, unfortunately. Maybe I will give it another try at a later time.



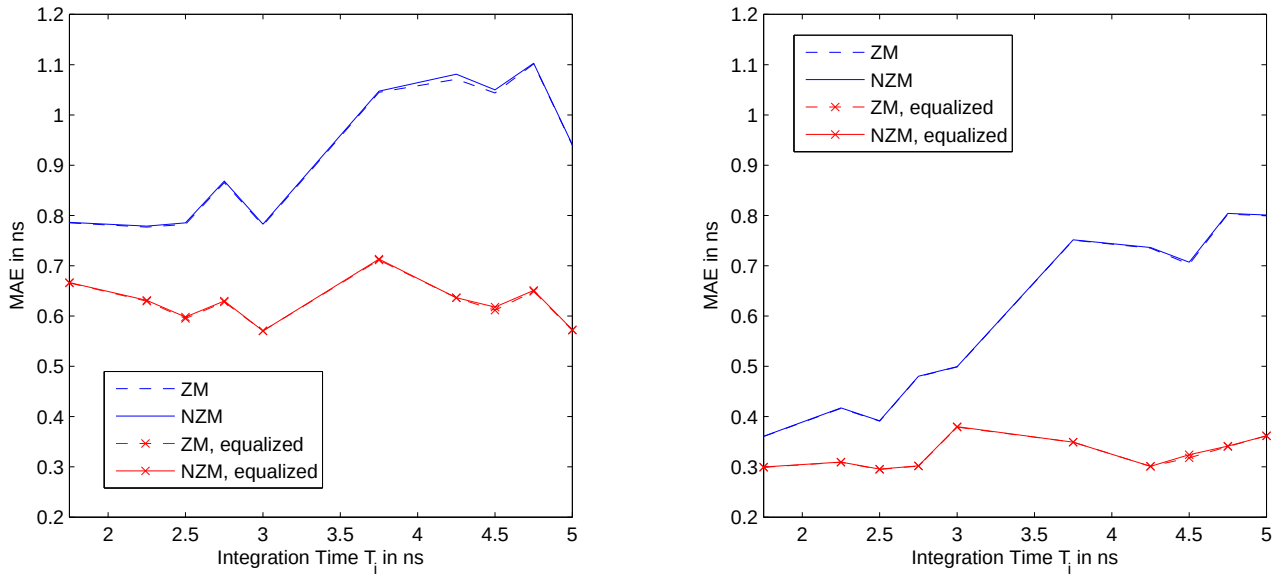
(a) K=0 (b) K=1

Figure 5: Ranging performance of the Equalized Sliding Algorithm

compared to the equalized algorithm. Moreover, it is more emphasized for the LOS (K=1) case. This may be explained by the fact that for channels with a strong leading edge long integration periods lead to high averaging, which in turn leads to an early detection of the leading edge (most errors are negative, cf. IR 10). Although a proper threshold selection may overcome that problem, it may be the case here that the resolution of the thresholds is not high enough.

Furthermore, a longer integration period of 3 ns seems to perform better, as also Fig. 6 suggests. The increased performance, however, is not directly related to the T_i, but to the temporal resolution T_r– by making the temporal resolution larger, the MA filter modeling the sliding algorithm is shorter and thus its equalizer more stable. This explains the local minima in Fig. 6 for integration values 2.5, 3, 4.5, and 5 ns. Another explanation for this phenomenon could be related to the fact that a coarser resolution does not require such a high resolution for the threshold values. However, this assumption would not explain why 3 ns and 2.5 ns show an increased robustness in Figure 5. In fact, the most influence will be the number of symbol repetitions which can be exploited for ranging, at least in terms of robustness.

Channel Estimation I also performed a set of simulations to assess the quality of channel estimation for both the sliding and the equalized sliding algorithm. Naturally, channel estimation is only sensible if the temporal resolution (T_r) is relatively high. Thus, I only compared integration periods $T_i$which lead to a resolution T_r= 0.25 ns (see. Tab. 1). This set of simulation proves that by using an equalizer, estimation errors can be reduced to a high degree (see Tab. 2). To further point out the benefits of the equalization with respect to channel estimation accuracy, I provide a few plot in Fig. 7. Here, the reader can see different NLOS channel responses ($K = 0$) and the outcome of the sliding algorithm and the equalized sliding algorithm, respectively. Obviously, with increasing $T_i$the dynamics of the output signal also reduces, unless the effect of averaging is mitigated via equalization. Moreover, one can observe that at the tail of the channel response oscillations in the equalized signal are visible – this is due to the fact that the equalizer, if it was a zero-forcing equalizer, would be an unstable system. Oscillations are still present at the MMSE equalizer, but they decay due to guaranteed stability (finite



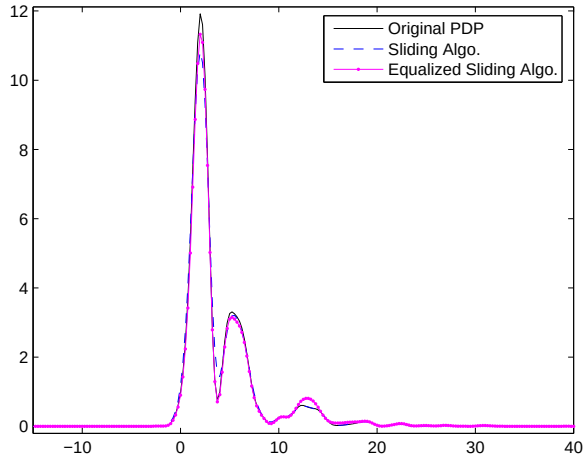
(a) K=0 (b) K=1

Figure 6: Minimum error (noise-free case) for sliding and equalized sliding under different channel conditions

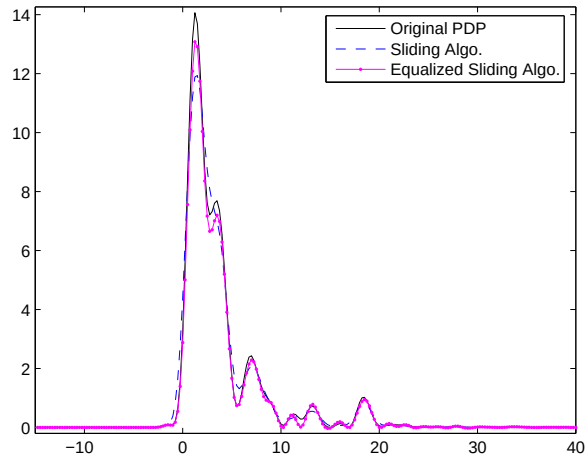
IR).

	K=0		K=1	
	Sliding	Equalized	Sliding	Equalized
T_i=0.75 ns	-14.48	-13.72	-13.06	-12.57
T_i=1.25 ns	-10.16	-12.11	-9.25	-11.32
T_i=1.75 ns	-7.56	-10.62	-6.89	-10.01
T_i=2.25 ns	-5.68	-7.14	-5.17	-6.49
T_i=2.75 ns	-4.71	-5.85	-3.99	-4.99
T_i=3.75 ns	-3.10	-4.26	-2.57	-3.28
T_i=4.25 ns	-2.71	-3.73	-2.00	-2.75
T_i=4.75 ns	-2.47	-3.43	-1.59	-2.16

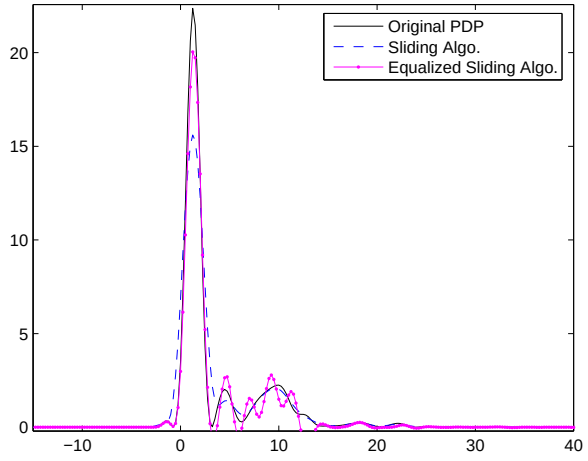
Table 2: Channel Estimation for Sliding Structures



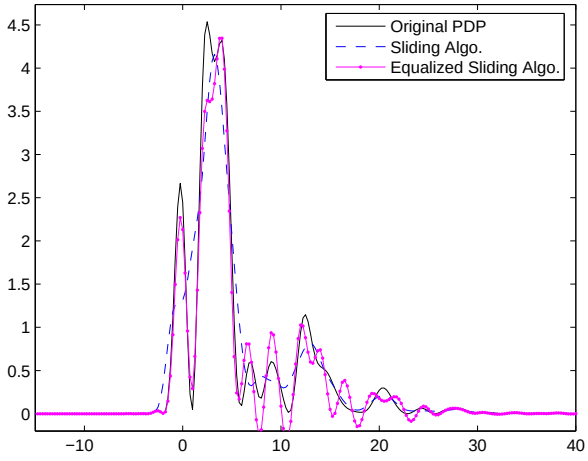
(a) T_i= 1.25 ns



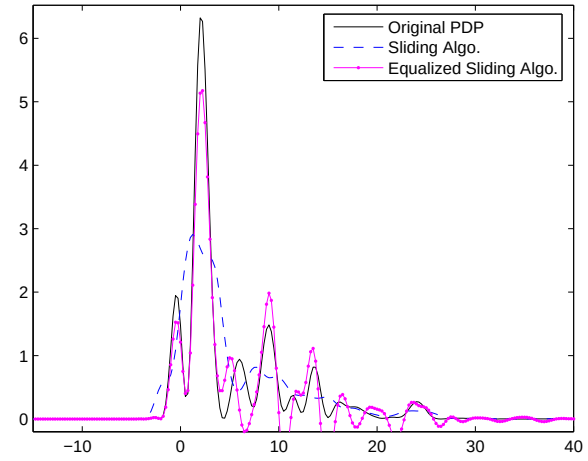
(b) T_i= 1.75 ns



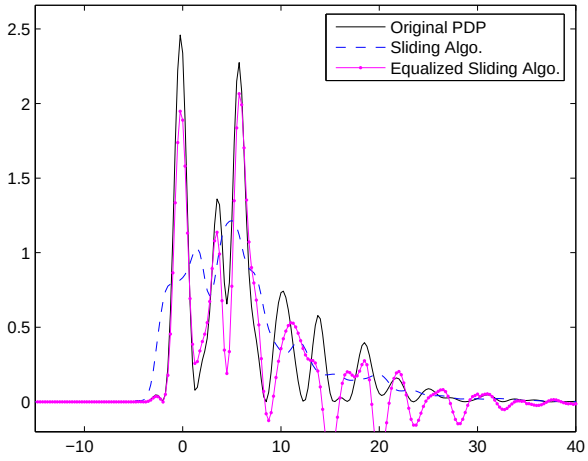
(c) T_i= 2.25 ns



(d) T_i= 2.75 ns



(e) T_i= 4.25 ns



(f) T_i= 4.75 ns

Figure 7: Equalized Sliding Algorithm for Channel Estimation