

## MIT Open Access Articles

*Hop timing recovery algorithms*

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

**Citation:** Yazdani, Navid et al. "Hop Timing Recovery Algorithms." Military Communications Conference, 2009. MILCOM 2009. IEEE, 2009. 1–7. © Copyright 2012 IEEE

**As Published:** <http://dx.doi.org/10.1109/MILCOM.2009.5379799>

**Publisher:** Institute of Electrical and Electronics Engineers (IEEE)

**Persistent URL:** <http://hdl.handle.net/1721.1/74256>

**Version:** Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

**Terms of Use:** Article is made available in accordance with the publisher's policy and may be subject to US copyright law. Please refer to the publisher's site for terms of use.



# HOP TIMING RECOVERY ALGORITHMS

Navid Yazdani, Nancy List, Dwight Hutchenson, Thomas Royster, and Ryan Shoup  
MIT Lincoln Laboratory  
Lexington, MA 02420

**Abstract**—Communication systems must maintain tight timing synchronization between the transmitter and receiver. Systems typically pull-in timing from a large timing offset and then track timing once it has been sufficiently pulled in. When timing errors are large, additional reference symbols are needed to pull-in time. In some frequency hopping systems, the time pull-in step must be repeated for every hop and could occur many times a second. It then becomes important to develop rapid time pull-in algorithms that can operate on a limited number of data and reference symbols. This paper presents two algorithms for rapid time pull-in. The performance as a function of the number of data and reference symbols is shown.

## I. INTRODUCTION

Frequency-hop communication is common in military systems due to its inherent protection against jamming and interference from other frequency-hop users (i.e., frequency-hop multiple-access interference). Such performance is obtained by pseudorandomly changing the center frequency of the transmitted signal from time to time. The sequence of center frequencies is determined by a hopping pattern that is known to both the transmitter and receiver.

Time synchronization of such systems is very important, and synchronization must occur at several stages to enable successful communication. First, a coarse-time synchronization stage [1] is employed to reduce the timing error to a fraction of a hop. This can be accomplished by periodically transmitting time probe sequences, which allows the receiver to determine whether timing is early or late and send the result to the transmitter [2]. Next, a fine-time synchronization stage (e.g., [3]) reduces the timing error to within a symbol period. The fine-time synchronization stage may be used over several hops to iteratively converge to correct timing, at which point the fine-time synchronization step ends and the resulting time estimate is used for the remainder of the transmission.

This work was sponsored by the Department of the Air Force under Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

For many frequency-hop systems, the combination of coarse-time synchronization and fine-time synchronization is sufficient. However, the system that we consider requires a third timing recovery stage that is used for each hop throughout the transmission. For example, transmitter or receiver mobility can produce time-varying delays. It is also possible for the receiver's processing chain to have delays that vary across the total hopping bandwidth [4]. We refer to this third stage of synchronization as hop-timing recovery (HTR).

In this paper, we describe and investigate two HTR algorithms, a data-assisted algorithm and a non-data-assisted algorithm. To evaluate performance, we consider the residual timing error at the output of each algorithm. In addition, we also investigate the system-wide impacts of each algorithm by evaluating the resulting packet error rate performance when demodulation and decoding is added to the system.

## II. SYSTEM DESCRIPTION

The transmitter and receiver employ frequency-hop communication with random hopping patterns. The system under consideration is assumed to employ standard methods for coarse-time and fine-time synchronization. Because of the timing errors that are still present after fine-time synchronization, it is possible that the received signal does not get sampled at the optimum times. If the time error is large enough, the sampled signal may be offset from the ideal sampled signal by one or more samples.

To account for synthesizer settling time, the system employs *dead time* at the beginning and end of each hop. During the dead times, the transmitter does not transmit a meaningful signal. As long as the receiver's synthesizer is set properly before the dead time ends, the entire transmitted signal can be received.

The system is designed such that each hop contains multiple *reference symbols* to aid in timing recovery. Reference symbols are modulation symbols whose value and position are known to both the transmitter and receiver. The receiver can thus search the received signal for the reference symbols and attempt to correct the timing. It is assumed that the search window is large enough to cover

Burst Rate	$N_r$	$N_d$
1	4	80
2	16	320
3	64	1280
4	256	5120
5	1024	20480

TABLE I  
THE FIVE BURST RATES EMPLOYED IN OUR SYSTEM.

the range of possible time offsets in the received signal. Although the addition of reference symbols reduces the data rate, the potentially large variations in timing error that are present in our system warrants their use.

We assume that phase recovery is not done prior to time recovery. Thus, we investigate HTR algorithms that can be performed noncoherently. For example, the data-assisted algorithm employs energy detection on the received samples. Both algorithms use the reference symbols, though in different ways, to ensure that the receiver is synchronized to the correct symbol time.

### III. SYSTEM PARAMETERS

The system can employ multiple data rates, and the data rate can be changed by varying the *burst rate*, which is related to the number of symbols per hop. Specifically, each of the burst rates consist of  $N_r$  reference symbols and  $N_d$  data symbols, and they are sometimes denoted as the  $N_r + N_d$  *burst rate*. The five different burst rates employed by our system are listed in Table I.

A rate-1/2 serially concatenated convolutional code (SCCC) is used by the system. After encoding a message, the codewords are modulated using either binary phase shift key (BPSK), quaternary phase shift key (QPSK), or 16-ary quadrature amplitude modulation (16-QAM). The data symbols are then interleaved using an  $N_d \times 576$  block interleaver, to distribute the symbols over 576 hops. The  $N_d$  data symbols are appended to  $N_r$  reference symbols and the hop is transmitted.

Regardless of the modulation or the values of  $N_r$  and  $N_d$ , the received signal is sampled at a fixed sampling rate. Then, the signal is downsampled by different amounts depending on the burst rate, and it is downsampled to a sampling rate that provides two samples per data symbol to the hop timing recovery algorithms.

At the output of the receiver's analog-to-digital converter, the dead time is assumed to be 512 samples at the beginning of the transmitted signal and 512 samples at the end. This allows there to be two samples at the beginning and two at the end if the signal is eventually downsampled by a factor of 256 for the 4+80 burst rate.

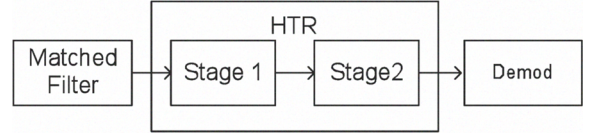


Fig. 1. DA algorithm location in the processing chain.

The reference symbol patterns are derived from binary complementary sequences [5]. Such sequences exist for all powers of two (e.g., [6]). One sequence of the pair is transmitted on the inphase branch and denoted  $I_{\text{ref}}$ , and the other sequence is transmitted simultaneously on the quadrature branch and denoted  $Q_{\text{ref}}$ . The reference sequence is transmitted as one contiguous block at the beginning of each hop. This creates a QPSK reference pattern which is used even if the data symbols that follow use a different modulation (e.g., 16-QAM). Complementary pairs are such that the component-wise sum of their respective aperiodic autocorrelation functions is zero everywhere except for a zero time offset. This property is not used directly in our implementations. However, one consequence of the complementary property is that each of the individual autocorrelation functions have relatively low sidelobes and are thus desirable for such an application. Thus, if desired,  $I_{\text{ref}}$  or  $Q_{\text{ref}}$  can be used as a single BPSK reference sequence pattern.

### IV. THE DATA-ASSISTED ALGORITHM

Hop timing error can be measured as consisting of an integer symbol offset and a fractional symbol offset. Correspondingly the data-assisted (DA) algorithm is performed in two stages. The first stage detects and corrects the fractional symbol offset. The second stage detects and corrects the integer symbol offset. This process is shown in Figure 1.

The first stage works as an energy detector which determines which fractional hop timing hypothesis contains the greatest energy. Given enough symbols, the correct timing hypothesis will statistically contain the greatest energy. The second stage works by aligning to the autocorrelation peak of the known reference symbol pattern.

Stage 1 works in two passes. In the first pass, it examines eight timing hypotheses to narrow down the fractional symbol timing correction. Based on the result of the first pass, the second pass examines eight more timing hypotheses in determining the final fractional symbol timing correction. This two-pass approach is effectively a binary search which calculates timing hypotheses down

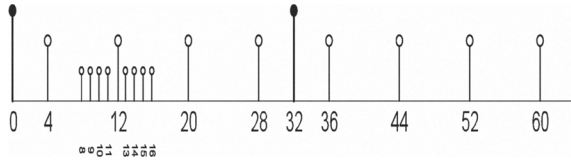


Fig. 2. Interpolated timing hypothesis locations.

to  $\frac{1}{64}$  of a symbol while calculating only 16 rather than 64 timing hypotheses.

Stage 1 operates after the matched filter at two samples per symbol plus additional deadtime samples on each end of the hop. The Stage 1 algorithm will use two samples per symbol to interpolate time hypothesis down to  $\frac{1}{64}$  of a symbol and ultimately return one sample per symbol plus the additional deadtime samples on each end of the hop.

The location of the interpolated timing hypothesis is shown in Figure 2. The diagram corresponds to one symbol period (note: this symbol period does not perfectly align with any particular symbol). The  $x$ -axis denotes time in  $64^{\text{th}}$  of a symbol (e.g., 12 corresponds to  $\frac{12}{64}$  of a symbol). During this symbol period, there are two measured samples taken at 0 and 32. In the first pass, the stage 1 algorithm interpolates the signal at 4, 12, 20, 28, 36, 44, 52, and 60. It then determines the interpolated hypothesis that contains the greatest energy. In the second pass, the algorithm interpolates  $\pm 4$  hypothesis at  $\frac{1}{64}$  spacing around the strongest estimate from the first pass. For example, in Figure 2, the timing hypothesis at 12 contains the most energy, so the second pass interpolates the signal at 8, 9, 10, 11, 13, 14, 15, and 16. The hypothesis containing the greatest energy (including the original hypothesis at 12) is returned as the resulting one-sample-per-symbol solution.

The block diagram in Figure 3 provides an illustration of the first pass of the Stage 1 algorithm. The basis of the stage 1 algorithm is that timing is first interpolated to a hypothesis, which is then squared and summed to obtain phase-independent energy. The hypothesis with the greatest energy is selected. At the output of this step, a timing hypothesis is selected in a resolution of  $\frac{1}{8}$  symbol. The index of the hypothesis with the greatest energy is taken as the output of the first pass.

The second pass of the Stage 1 algorithm is illustrated in Figure 4. At the output of this step, the fractional timing error is corrected with a resolution of  $\frac{1}{64}$  symbol. Depending on the index calculated in the first pass, different filter coefficients are used. The filter path for

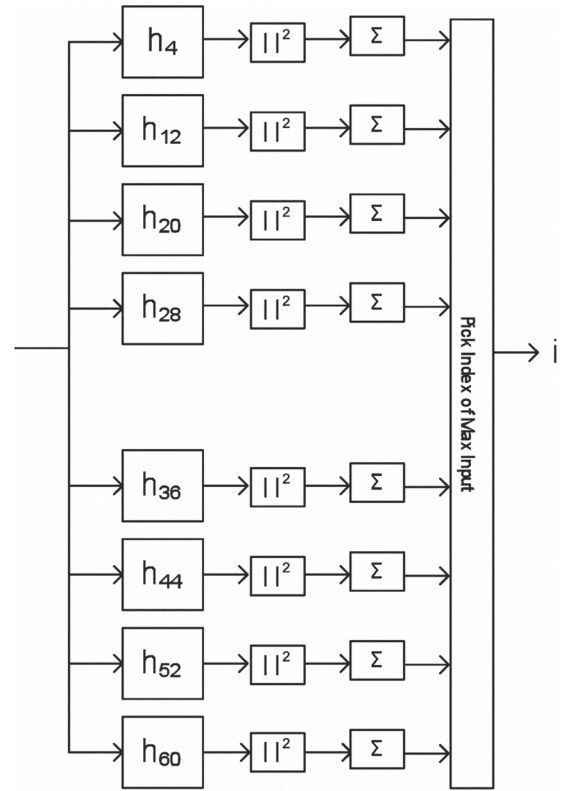


Fig. 3. First pass block diagram (Stage 1).

$h_i$  does not need to be recalculated, as this was done in the first pass. Otherwise, the operation in the second pass is identical to the first pass. The index of the largest hypothesis from the second pass is used to select the correct interpolated time hypothesis and is the output of the stage 1 algorithm. Note that the stage 1 algorithm uses the reference and data symbols to detect energy but does not use the known reference symbol pattern. Stage 1 performance could be further improved by additional processing incorporating knowledge of the known reference symbol pattern.

The Stage 2 algorithm then correlates the Stage 1 output with the known reference symbol pattern over the time uncertainty window. The crosscorrelation with the largest value is then used to shift the Stage 1 output by an integer number of samples. Therefore only reference symbols are used to perform the Stage 2 integer time shift. Finally the additional deadtime samples can be removed and the output of the algorithm is exactly one sample per symbol with full timing correction applied.

## V. THE NON-DATA-ASSISTED ALGORITHM

The non-data-assisted (NDA) algorithm uses only the reference symbols transmitted with the received signal

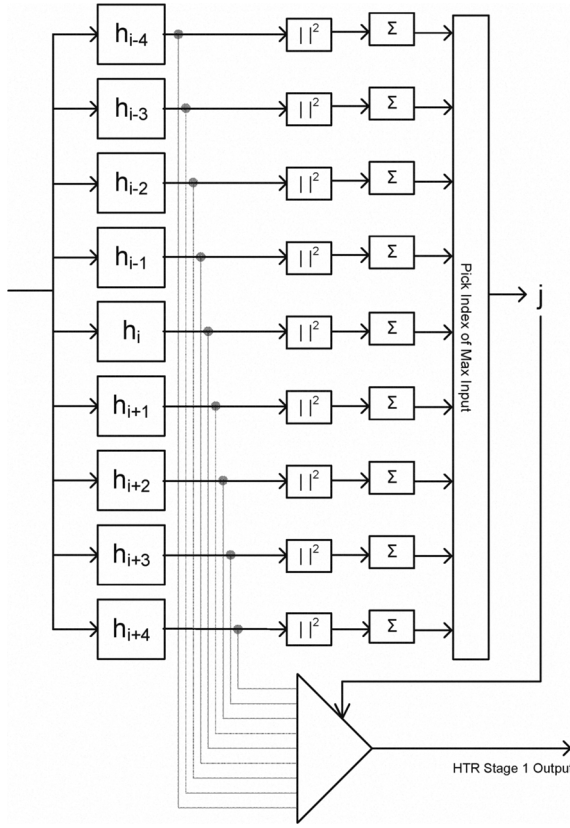


Fig. 4. Second pass block diagram (Stage 1).

to estimate the time error in the received signal. Correlation with only the reference symbols provides enough information to accurately estimate the time error in the received signal for many of the scenarios examined in this paper.

Let  $I_s$  and  $Q_s$  be the inphase and quadrature samples, respectively, of the reference symbols plus samples that cover the uncertainty window of time before and after expected transmission time of the reference symbols. In the first stage of estimating the time error in the NDA algorithm,  $I_s$  and  $Q_s$  are convolved with the stored reference symbols  $I_{\text{ref}}$  and  $Q_{\text{ref}}$  as follows:

$$R_{\text{ref}} = [I_s \star I_{\text{ref}} + Q_s \star Q_{\text{ref}}]^2 + [I_s \star Q_{\text{ref}} - Q_s \star I_{\text{ref}}]^2,$$

where  $A \star B$  denotes the convolution of  $A$  and  $B$ . This method of correlating the received reference symbols with the stored reference symbols effectively takes the magnitude of the projection of the received sequence,  $I_s + jQ_s$ , onto the in-phase reference sequence,  $I_{\text{ref}} + jQ_{\text{ref}}$ , and adds it to the magnitude of the projection of the received sequence onto the quadrature reference sequence,  $I_{\text{ref}} - jQ_{\text{ref}}$ . Correlating in this manner allows

the estimate of time offset to be made on a phase-incoherent received sequence.

Picking the peak value of the convolution of the received reference symbols with the stored reference symbols will usually allow the time offset to be estimated to within  $\frac{1}{2}$  a sample (in this case  $\frac{1}{4}$  symbol). Having more samples of reference symbols to correlate against obviously reduces the likelihood of picking the wrong sample time offset at this stage because the energy in the received samples of the reference symbols is increased relative to the noise added to those reference symbol samples with increasing sampling rate.

Less obviously, having more reference symbols (rather than having more samples of reference symbols) also reduces the likelihood of picking the wrong sample time offset because the data symbols that fall within the time uncertainty window change the shape of the received reference symbol sequence. This affects the shape of the correlation curve and can sometimes cause false peaks by increasing the sidelobes of the correlation curves. The fewer the total number of reference symbols, the greater the effect of the data symbols that occur immediately after the reference symbols have on the shape of the correlation curves.

In the second stage of the NDA algorithm, the peak correlation value from the convolution of the received reference symbols with the stored reference symbols is used to more finely estimate the time offset within the range of  $\pm \frac{1}{2}$  sample time. This is done by interpolating a normalized version of the peak correlation value off of a normalized reference correlation curve.

Since the correlation curve is symmetric about zero, the sample-offset-corrected received sequence is correlated with the stored reference symbols shifted one sample early ( $I_e + jQ_e$ ) and the stored reference symbols shifted one sample late ( $I_l + jQ_l$ ) to resolve the ambiguity in whether the peak value falls on the early or late side of the reference correlation curve. We have as the early and late correlations, respectively,

$$R_e = [I_s \star I_e + Q_s \star Q_e]^2 + [I_s \star Q_e - Q_s \star I_e]^2$$

$$R_l = [I_s \star I_l + Q_s \star Q_l]^2 + [I_s \star Q_l - Q_s \star I_l]^2$$

The peak value of  $R_{\text{ref}}$  occurs at the center of that sequence when the sequence  $I_{\text{ref}} + jQ_{\text{ref}}$  is received with a timing error of less than one sample. Otherwise, the peak value occurs at some offset from the center of the sequence which corresponds to the bulk time error which is corrected by either delaying or advancing the sequence by the integer number of samples corresponding to the



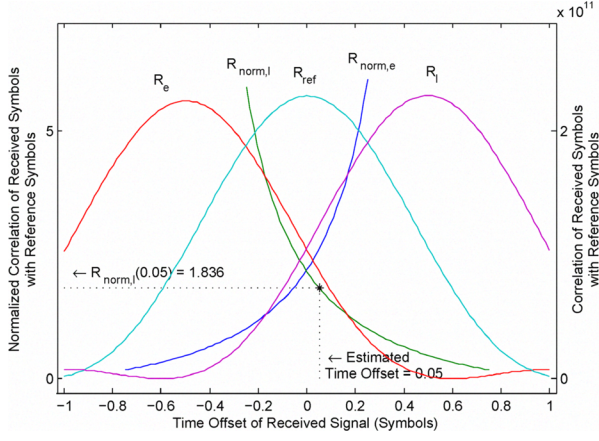


Fig. 5. Correlation curves (reference, early and late) and normalized correlation curves (reference/early and reference/late)

offset. Denote the value of a sequence  $S$  at position  $k$  as  $S(k)$ . If the peak value of the sequence  $R_{\text{ref}}$  is  $R_{\text{ref}}(n)$ , then  $n$  is called the *integer sample time estimate*. As will be discussed, the integer sample time estimate is used to decide whether timing is early or late.

The late sequence  $R_l$  and early sequence  $R_e$  are each used to normalize the reference correlation sequence,  $R_{\text{ref}}$ , as follows:

$$R_{\text{norm},l}(n) = \frac{R_{\text{ref}}(n)}{R_l(n)}, \quad \forall n$$

$$R_{\text{norm},e}(n) = \frac{R_{\text{ref}}(n)}{R_e(n)}, \quad \forall n.$$

$R_{\text{norm},l}$  is the normalized late sequence and  $R_{\text{norm},e}$  is the normalized early sequence. Interpolation off a reference curve is then done using the normalized peak correlation value to estimate the fractional sample offset of the received signal. Figure 5 shows  $R_e$ ,  $R_{\text{ref}}$ , and  $R_l$  correlation curves superimposed with  $R_{\text{norm},e}$  and  $R_{\text{norm},l}$  correlation curves. As an example, the interpolation of  $R_{\text{norm},l}$  value of 1.836 to estimate a fine time offset of 0.05 symbols is shown in Figure 5. This value could have been used to interpolate off the  $R_{\text{norm},e}$  curve. The correct normalized correlation curve to use for interpolation is chosen by comparing the magnitude of  $R_e(n)$  with  $R_l(n)$ . If  $R_e(n) > R_l(n)$ , then interpolate off the  $R_{\text{norm},e}$  normalized correlation curve. If  $R_l(n) > R_e(n)$ , then interpolate off  $R_{\text{norm},l}$  normalized correlation curve.

The estimated fine time offset is corrected by time-shift interpolation filter at the last stage of the NDA algorithm.

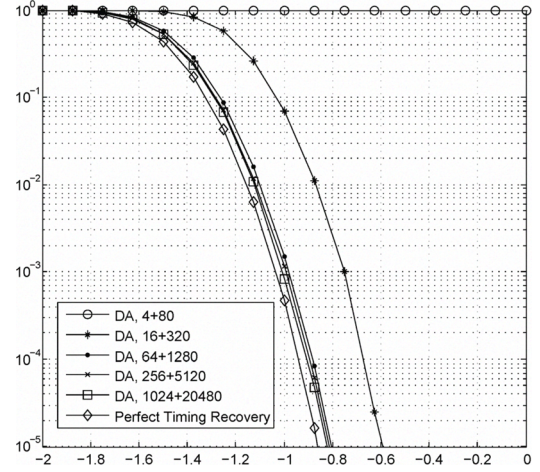


Fig. 6. Simulation results for all burst rates considered, for BPSK and the DA algorithm.

## VI. RESULTS

Figure 6 contains simulation results for all of the burst rates considered for BPSK with the DA algorithm, along with results for perfect timing recovery. Note that perfect timing recovery assumes perfect knowledge of the integer and fractional time offsets that are computed by the HTR algorithms and uses those values to adjust time in the received sequence using a combination of sample delays and interpolating filters. Therefore, signal distortion induced by the interpolating filter is present in the results shown for perfect timing recovery. Performance is very poor for the 4 + 80 burst rate on the range of  $E_s/N_0$  shown, but increasing the burst rate to 16 + 320 yields results approximately 0.3 dB worse than perfect timing recovery at a block error rate (BLER) of  $10^{-4}$ . For burst rates of 64 + 1280 and higher, the DA algorithm achieves performance within 0.06 dB of perfect timing recovery at this BLER.

Figure 7 contains simulation results for all of the burst rates considered for BPSK with the NDA algorithm along with results for perfect timing recovery. As is the case with the DA algorithm, performance is very poor for the 4 + 80 burst rate on the range of  $E_s/N_0$  shown. For the burst rates 16 + 320 and 64 + 1280, performance is approximately 0.6 dB and 0.1 dB worse than perfect timing recovery, respectively, at BLER= $10^{-4}$ . For each of the burst rates 256 + 5120 and 1024 + 20480, the NDA algorithm achieves performance within 0.05 dB of perfect timing recovery at BLER= $10^{-4}$ .

Both the DA and NDA algorithms perform quite poorly for the 4 + 80 burst rate, though the DA algo-

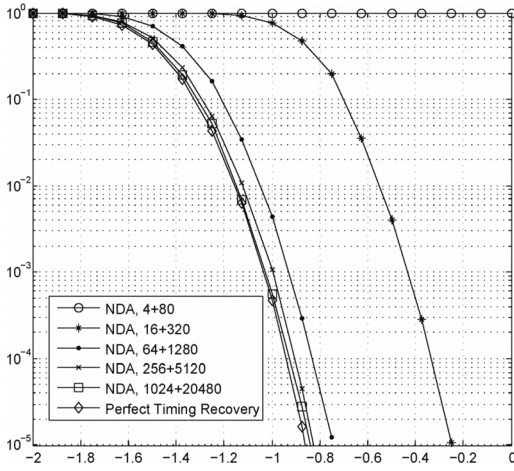


Fig. 7. Simulation results for all burst rates considered, for BPSK and the NDA algorithm.

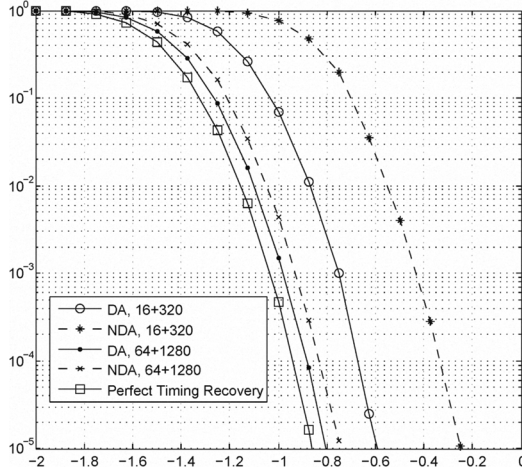


Fig. 8. Simulation results for BPSK with the both the DA and NDA algorithms, for burst rates 16 + 320 and 64 + 1280.

algorithm does offer about 1 dB improvement over the NDA algorithm at  $\text{BLER}=10^{-4}$ . Both algorithms have performance very close to that of perfect timing recovery for 256 + 5120 and 1024 + 20480. Figure 8 shows the results of each HTR algorithm for the remaining two burst rates. For 16 + 320 and 64 + 1280, the DA algorithm performs about 0.3 dB and 0.05 dB better than the NDA algorithm, respectively.

Let  $T$  represent a symbol duration at the burst rate 4 + 80. Up to this point, the timing error considered is within  $0.5T$ . However, performance is much better for the 4+80 burst rate if the timing uncertainty is decreased to  $0.25T$ . Results for BPSK are shown in Figure 9 with both

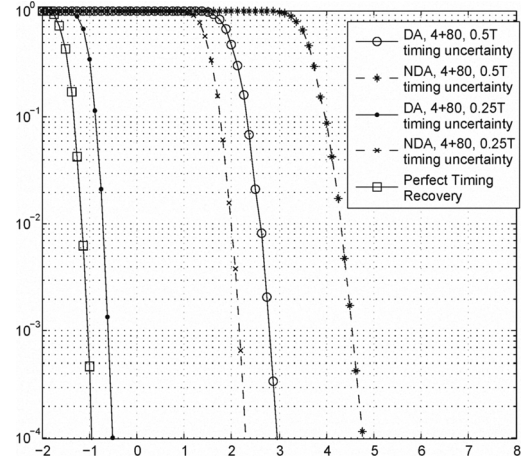


Fig. 9. Simulations results for BPSK and burst rate 4+80 for different timing uncertainties.

algorithms and timing uncertainties. At the decreased timing uncertainty, the DA algorithm performs over 3 dB better than the NDA algorithm at  $\text{BLER}=10^{-4}$ , and the DA algorithm is within 0.5 dB of perfect timing recovery at this BLER. Note that both algorithms are tuned to operate over this smaller timing uncertainty to achieve these results.

Finally, results are shown in Figure 10 for BPSK, QPSK, and 16-QAM, for both algorithms as well as perfect timing recovery. Though the DA algorithm is optimized for use with BPSK data symbols, its performance is very close to that of perfect timing recovery for all three modulations at a burst rate of 64 + 1280. The NDA algorithm also performs well for all of the modulations at this burst rate.

## VII. CONCLUSION

Rapid timing recovery is a necessary component in a variety of communication systems. The timing recovery algorithms must accurately pull-in time over a large range of time uncertainty with a limited number of reference and data symbols. Both the DA algorithm and the NDA algorithm described in this paper perform well under these conditions. When there are abundant reference (and data) symbols, both algorithms operate nearly flawlessly with less than 0.05 dB of implementation loss over arbitrarily large pull-in time ranges. As the number of reference symbols is decreased to 16, the implementation loss increases to several tenths of a dB and the DA algorithm shows some improvement over the NDA algorithm by using the more plentiful data symbols. When the

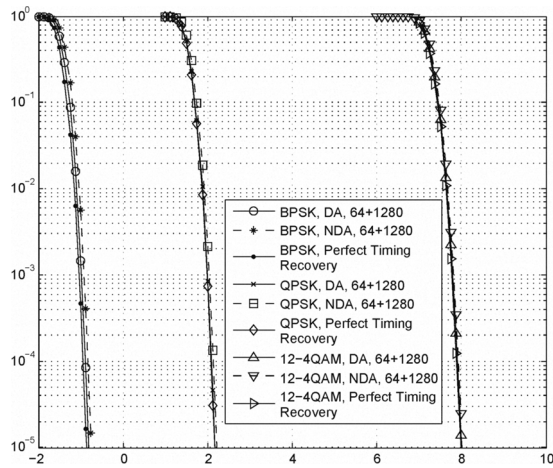


Fig. 10. Simulations results for the DA algorithm and perfect timing recovery for several modulations.

number of reference symbols is further reduced to 4, both algorithms are severely impacted. By reducing the time uncertainty to 25% of a symbol, the DA algorithm which can use the additional 80 data symbols can operate with only 0.5 dB implementation loss, while the NDA algorithm has a much larger implementation loss. Though the DA algorithm does show some performance improvement compared to the NDA algorithm when fewer reference symbols are available, it is also more computationally intensive. In cases where performance is comparable, it may be desirable to use the more computationally efficient NDA algorithm.

Fundamentally, these results show that rapid timing recovery can be achieved and how well it can be achieved. The performance is primarily driven by the number of available reference symbols and the results from this paper can be used when determining the number of reference symbols needed for future communication systems.

## REFERENCES

- [1] C. A. Putman, S. S. Rappaport, and D. L. Schilling, "A comparison of schemes for coarse acquisition of frequency-hopped spread-spectrum signals," *IEEE Transactions on Communications*, vol. COM-31, no. 2, pp. 183–189, February 1983.
- [2] M. R. Shane, H.-G. Yeh, and A. H. Yamada, "Uplink timing detection for frequency hopping communication," *Proceedings of the 1999 IEEE Military Communications Conference (MILCOM)*, vol. 2, pp. 860–864, October 1999.
- [3] L. J. Mason, "Estimation of fine-time synchronization error in FH FDMA satcom systems using the early-late filter technique," *IEEE Transactions on Communications*, vol. 42, no. 2/3/4, pp. 1254–1263, Feb./Mar./Apr. 1994.

- [4] N. B. List, "Effect of group delay variation on time tracking for frequency hopped satellite systems," *Proceedings of the 2006 IEEE Military Communications Conference (MILCOM)*, October 2006.
- [5] M. J. E. Golay, "Complementary series," *IRE Transactions on Information Theory*, vol. IT-7, pp. 82–87, April 1961.
- [6] R. J. Turyn, "Hadamard matrices, Baumert-Hall units, four symbol sequences, pulse compression and surface wave encoding," *J. Combin. Theory Ser. A*, 16, pp. 313–333, 1974.