Watermark-Based Code Construction for Finite-State Markov Channel with Synchronisation Errors

Shamin Achari^a, Ling Cheng^{a,*}

^aSchool of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, 2000, Gauteng, South Africa

Abstract

With advancements in telecommunications, data transmission over increasingly harsher channels that produce synchronisation errors is inevitable. Coding schemes for such channels are available through techniques such as the Davey-MacKay watermark coding; however, this is limited to memoryless channel estimates. Memory must be accounted for to ensure a realistic channel approximation - similar to a Finite State Markov Chain or Fritchman Model. A novel code construction and decoder are developed to correct synchronisation errors while considering the channel's correlated memory effects by incorporating ideas from the watermark scheme and memory modelling. Simulation results show that the proposed code construction and decoder rival the first and second-order Davey-MacKay type watermark decoder and even perform slightly better when the inner-channel capacity is higher than 0.9. The proposed system and decoder may prove helpful in fields such as free-space optics and possibly molecular communication, where harsh channels are used for communication.

Keywords: Finite-State Markov Channel, Insertion Deletion Correction, Synchronisation Channel Modelling, Synchronisation Decoding.

1. Introduction

As communication systems and technology advance, we inevitably begin to transmit data over increasingly harsher channels, and the need for digital signal processing techniques to improve the reliability of communications, as always, plays a vital role. These harsh channels naturally produce insertions and deletions (synchronisation errors) in addition to the standard substitution error. This impact is already witnessed in prevailing real-world communication systems that use visible light or free-space optics where synchronisation between the receiver and transmitter is not easily maintained and as a result, these systems suffer from synchronisation errors [1, 2]. The concern of synchronisation and its related errors are even found in unconventional fields such as DNA sequencing [3, 4] to even more intricate specialisations such as molecular communications [5, 6]. The latter has gained popularity in recent years as a potential for data communication in nano and micro systems, where open problems in the field contemplate the effects of synchronisation and memory [5, 6]. Davey and MacKay developed a watermark coding scheme in [7, 8] to deal with such insertion, deletion and substitution (IDS) errors. While this scheme proves useful in many fields, the main shortfall is that the channel used is a memoryless approximation, the Davey-MacKay (DM) channel

model, and many practical scenarios are hardly ever memoryless or independent and identically distributed (IID) in nature [9]. A recent paper by Achari et al. describes the modelling of an IDS channel that incorporates the effects of memory [2]. Here the channel is based on a Finite-State Markov Channel (FSMC), which allows the system to transition between various states (transmission, substitution, deletion and insertion) based on different probabilities and the given current state. This provides a valuable method to model correlated synchronisation channels and also provides a method to simulate various scenarios without the need for actual transceivers. Again, while the model and method presented are helpful, it is not without drawbacks. As the model is a general method of simulating such channels, it lacks any steadfast rules and limitations, which causes challenges in producing an effective encoding/decoding algorithm to protect against and correct communication errors.

In this paper, the IDS memory model presented by Achari et al. is adapted and integrated with the DM watermark scheme to create a more encompassing channel and code construction that is more indicative of realistic communication channels while still being simple enough to provide an effective error-correction and resynchronisation scheme. The main contributions of this paper are threefold. Firstly, the new proposed channel model, which combines ideas from the DM and FSMC models, is presented. Secondly, the code construction and decoding, which are based on the watermark codes, are described for the proposed channel. Lastly, we provide the details of extending

^{*}Corresponding Author: Ling Cheng

Email address: Ling.Cheng@wits.ac.za (Ling Cheng)

This work has been submitted to the Elsevier DSP for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible

the first-order decoding for the DM watermark scheme to a second-order system. While the last contribution has been alluded to in [7, 8], to the authors' knowledge, no explicit formulation is found in the literature. This second-order IID decoding will be used as one of the comparative benchmarks to test the proposed model and decoding algorithm against.

The rest of the paper is structured as follows. The DM channel and watermark code construction and the IDS FSMC are further detailed in Section 2. This is followed by Section 3 where the novel adapted channel model is described along with the new code construction. Section 4 then discusses the evaluation metrics and compares the simulations' various tests and results with relevant analysis. Conclusions are finally drawn in Section 5.

2. Background and Literature Review

2.1. Synchronisation Channels and Coding

Synchronisation channels remain a relatively unstudied and open topic in information theory and are generally more complex to analyse than their traditional substitution channel counterparts [10]. Mitzenmacher provides an early survey of these synchronisation channels in [11] and appropriately draws comparisons in the analysis of synchronisation channels to traditional channels by comparing them to constructs of Levenshtein Distance over the more straightforward Hamming distance. With recent advancements in DNA-type storage, a recent surge in research regarding insertion and deletion channels has since been prompted. More recently, [10] and [12] provide detailed surveys of the progress made in the investigation of synchronisation channels. Cheraghchi et al., in particular, survey work regarding the capacity of such channels [10].

2.2. Davey-Mackay Synchronisation Channel and Watermark Code

The DM watermark code provides a way to ensure reliable communication across a channel that produces IDS errors. A detailed description of the code construction can be found in [7, 8] and readers are encouraged to survey these texts for an in-depth explanation and analysis. What follows is the basic overview of watermark codes in order to distinguish the novelty of the proposed channel and code construction.

In the DM construction a message string, \mathbf{m} , consisting of q-ary symbols, are encoded using LDPC, which produces a corresponding binary sequence, \mathbf{d} . This sequence \mathbf{d} is then processed through a sparsifier which seeks to create a sparse sequence using a codebook or lookup table entry for each input symbol (group of q bits). The mean density of the sparse vectors is denoted by f, and the output of the sparsifer is the sparse binary string, \mathbf{s} . Here \mathbf{s} is defined as sparse if the hamming weight of the sequence divided by the length is less than 0.5. This sparse sequence is then modulo-two added with a watermark sequence, \mathbf{w} , to produce \mathbf{t} , which will be the sequence transmitted over the channel. The watermark sequence is one of the main components that allow the system to regain synchronisation during the decoding process. It can essentially be seen as a timing sequence as it is known to both the receiver and transmitter. The watermark sequence is generally run-length limited or random. [13] Provides a method to create the watermark sequence based on a proposed probability metric. For this paper, all watermark sequences are considered random. The sequence \mathbf{t} is then queued to be sent over the DM IDS channel.

Bits queued for transmission across the DM channel may undergo one of three transitions. 1) With a probability of P_i , a bit may randomly be inserted into the received sequence. For m consecutive insertions the probability is given as P_i^m . Following an insertion or multiple insertions, a transmission or deletion must follow to proceed to the next time step. Since, in theory, an unlimited number of consecutive insertions may occur, a maximum number of insertions, I_m , is imposed on the system for simplification. 2) A bit may be deleted from the sequence and will not appear in the received string with a probability P_d . 3) A bit may be transmitted with a probability P_t , where $P_t = 1 - P_i - P_d$ [7, 8, 14]. Note that when I_m insertions occur, a bit is transmitted with probability $\hat{P}_t = 1 - P_d$ [15]. If a bit is transmitted, it may undergo a substitution with a probability P_s . The DM channel model is better illustrated in Figure 1 [16] where τ_n and τ_{n+1} indicates the time at n and n+1 respectively.



Figure 1: Davey-Mackay synchronisation channel model

The received sequence \mathbf{r} is produced at the channel's output. Since the channel produces a mixture of insertions and deletions, the length of \mathbf{r} is not necessarily equal to the length of \mathbf{t} . The received sequence is then processed through an inner decoder where the goal is to produce a symbol-by-symbol likelihood function: $P_n(a) = P(\mathbf{r}|\mathbf{d_n})$ $a, P_i, P_d, P_s, P_t, f, \mathbf{w}$ [15]. This involves making use of the Forward-Backward (FB) algorithm to infer the hidden states, which in this case is the synchronisation drift or the number of insertions minus the number of deletions the channel has made from the beginning of the channel use till time τ_n where the *n*th bit is queued to enter the channel. This FB algorithm is run firstly at a bit level, and a final forward pass is rerun at a symbol level. This produces a symbol-by-symbol likelihood which can be used as the input into the LDPC decoder. The LDPC decoder is a probabilistic iterative decoder that seeks to determine the

marginal posterior probabilities for the codeword symbols.

The DM code construction has been successfully used in many applications ranging from the barcoding of DNA in [3, 4] to speech watermarking in [15]. There has also been much work on improving the code construction and decoding, such as [17, 18, 19]. These references still, however, establish the core channel model on the memoryless interpretation.

2.3. FSMC Synchronisation Channel

Achari et al. describe a method of creating a channel model for an IDS channel that contains memory and thus has correlations between errors [2]. This contrasts with the DM synchronisation channel, which allows insertions, deletions and substitution errors to occur in an IID (essentially memoryless) manner. The model presented in [2] is based on a FSMC where the states of the model are Transmission (T), Substitution (S), Deletion (D) and Insertion (I). Here, one may move between any two states, including self-transitions, with some probability defined in a corresponding transition matrix. The model is better illustrated in Figure 2. In [2] the receiver is assumed to have full knowledge of the transmitted data. This allows the use of the Levenshtein Distance (LD or edit distance) algorithm to find the most likely series of events (transmission, substitution, deletion or insertion) that occurred to produce the final received string. Since this sequence is essentially the hidden states of the channel, the system is reduced to a Markov chain and the probabilities of the transition matrix are determined by using the Baum-Welch algorithm. The emission matrix, in this case, is reduced to an identity matrix with the dimensions equivalent to the number of states. The power of this modelling methodology is then shown by utilising data from a realworld visible light communication system and producing the corresponding channel model. As stated previously, this approach is extremely useful in creating the channel model and parameters. However, providing an encoding and decoding scheme to prevent such errors and resynchronise the data proves somewhat complicated.



Figure 2: Four-state FSMC for IDS channel[2]

3. Proposed Memory Synchronisation System

3.1. Proposed Memory Synchronisation Channel

This paper draws inspiration significantly from DM watermark codes [7, 8] and adapts the FSMC in [2] accordingly. In fact, this new proposed model can be viewed

as an amalgamation of the DM model and the FSMC. Firstly, since the main timing and resynchronisation capabilities rely heavily on the watermark sequence, we need to ensure that when the system transitions from time τ_n to τ_{n+1} that a bit queued on the transmitter end actually enters the channel. This differs from the FSMC as time progresses to the next time step even if no transmitted bit is sent across the channel, e.g. insertion to another insertion will cause a time increment for the FSMC. While the new proposed model allows for multiple insertions, before moving to the next time step, either a deletion of the actual transmitted bit or transmission must occur, which is similar to the DM channel model. Another alteration from the FSMC is that a substitution may only occur if a transition occurs. Again, this is similar to the DM channel, and the probability of substitution is treated as IID. The new model is adapted only to include the transmission, deletion and insertion states and a corresponding memoryless probability of substitution. We can obtain this new three-state FSMC from the original four-state FSMC. Firstly, the stationary distributions of the four-state transition matrix is determined which provides the IID probabilities for P_t , P_s , P_d and P_i . These probabilities are used to create a corresponding DM code construction and watermark decoding for comparative purposes and benchmarking. The value of P_s is used in both the DM construction and as the corresponding memoryless substitution probability for the new proposed model. After obtaining the stationary distribution values, the columns and rows corresponding to substitutions in the four-state transition matrix are removed and normalised across the rows to obtain the threestate transition matrix. To simplify the situation further, a limit of I_m maximum consecutive insertions is imposed on the channel - as with the DM model. If the maximum number of insertions is reached, the system must undergo a transition or a deletion and move to the next time step. This requires the value of insertion to insertion in the transition matrix to become zero (and normalise across the row) for this step. In all further simulations within this paper, the number of maximum insertions is capped to 1, which greatly simplifies the scenario. However, in theory, this may be restricted to any number - restricting this to zero produces a deletion channel. The proposed channel model is depicted in Figure 3 and the corresponding transition matrix, A, is given in Equation 1. Accompanying the transition matrix is the initial distribution vector, $\Pi = \{\pi_{-x_{max}}, \pi_{-x_{max+1}}, ..., \pi_{x_{max}}\}, \text{ which represents the}$ probabilities of starting in a particular state where x_{max} is the maximum offset considered.

$$A = \begin{bmatrix} a_{TT} & a_{TD} & a_{TI} \\ a_{DT} & a_{DD} & a_{DI} \\ a_{IT} & a_{ID} & a_{II} \end{bmatrix}$$
(1)

3.2. Proposed Code Construction

The code construction for this system model once again draws substantially from DM watermark codes construc-



Figure 3: Proposed three-state Markov model for synchronisation channel

tion. In the proposed system, the main idea of the watermark scheme is used to regain synchronisation - the primary differences occur at the FSCM channel and Inner Decoder blocks in Figure 4 which illustrates the block diagram of the systems code construction. For this paper, the scope remains entirely on the inner decoder at a bit level to regain synchronisation. This is an added benefit of the proposed methodology and construction as the system is entirely independent of the outer encoder and decoder. As such, any substitution error correction code may be used as the outer code in conjunction with the presented inner code construction, and the system will not be restricted to using LDPC as in the original DM watermark scheme. Here the bitstream block represents the data in binary or the message, d. This data sequence is then processed through a sparsifier. As in the DM code construction, the sparsifier provides the decoder with its capabilities as only information bits and errors will likely produce a 1 in the resulting received sequence. For simulations in this paper, a 4-to-5 sparsifier is used where the output code bits correspond to the sparsest permutations possible. The density of this output codebook and consequently the mean density of the output sequence of the sparsifier is again denoted by f. This sparse string, s, of length Γ is then modulo two added to a watermark vector, w, (also of length Γ) to produce an encoded bitstream, t, ready to be sent over the channel. The channel's output is the received sequence, \hat{t} . This received sequence then passes through the (inner) decoder, which seeks to resynchronise the bitstream and remove any insertion or deletion errors from \hat{t} . The process and workings of the inner decoder are further detailed in Section 3.3. The resynchronised string, denoted by r, then has the watermark sequence removed from it to produce \hat{s} which is the sparse decoded sequence. The sparse decoded sequence is then fed into a desparsifier which reverses the operations of the sparsifer to produce d, which is the received data bitstream. It is worth noting that the resynchronisation process ultimately introduces additional substitution errors into the sequence that were not caused by the channel. Such a system will undoubtedly benefit from an (outer) encoder and decoder.



Figure 4: Block diagram for system code construction

3.3. Inner Decoder

The purpose of the inner decoder is to regain synchronisation by accounting for the errors caused by insertions and deletions. A probabilistic approach is used to determine the most likely hidden states the channel traversed to produce the given received string. This decoding is based on Hidden Markov Model (HMM) decoding, and in particular, we use the FB algorithm on a bit level. For more information regarding standard HMMs and their respective decoding, readers are encouraged to review [20, 21]. What we present is not an HMM in the traditional sense, and as indicated in [7, 8], there are some subtle variations. Firstly the hidden states of the HMM are not the same as the states of the transition matrix. Here the hidden state is defined as the synchronisation offset or synchronisation drift (number of insertions minus number of deletions) at a given time instance. As in the DM code construction, we also restrict the maximum offset to x_{max} to simplify the calculations as our number of states ranges from $-x_{max}$ to x_{max} . In our case, x_{max} is equal to five times the absolute final offset. For example, if 600 bits were transmitted and only 595 bits were received, the final offset, Ψ , at the final time, Γ , would be -5 and the range of possible hidden states would be -25 to 25. The exception is when Ψ is zero (equal number of insertions and deletions occurring during data transmission), then x_{max} is set to 5. Increasing this range allows for a truer representation of the channel and could potentially increase accuracy, but this comes with a trade-off of increasing the number of calculations. The second alteration is that the transition matrix may not be used directly in the FB calculations, and the transitions are instead based on channel events. To better illustrate the transition between each time step, we use the number of bits output by the channel at a given time interval. Since two time intervals (three time steps) are used, we indicate the offset at times τ_{n-2} , τ_{n-1} and τ_n as $\psi_{n-2} = i$, $\psi_{n-1} = j$ and $\psi_n = k$ respectively. At each time interval $(\tau_{n-2} \text{ to } \tau_{n-1} \text{ or } \tau_{n-1} \text{ to } \tau_n)$ the channel may output from zero to $I_m + 1$ bits depending on the events of the channel. This bit output and corresponding channel events are better illustrated in Table 1.

From the number of bits output by the channel, and consequently the events that produced such an output, various transition probabilities that are based on the threestate transition matrix can be derived. Table 2 illustrates the probabilities used for the various bit emissions over two-time intervals. It is worth noting that values in this

Bits Output by Channel	Channel Event
0	Deletion
	Transmission
1	OR
	(Insertion and Deletion)
	(Insertion and Transmission)
2	OR
	(Two Insertions and Deletion)
	(Two Insertions and Transmission)
3	OR
	(Three Insertions and Deletion)
$I_m + 1$	$(I_m$ Insertions and Transmission)

Table 1: Bits Output by Channel and Corresponding Channel $\operatorname{Event}(\mathbf{s})$

Table 2: Bits Output by Channel and Corresponding Probabilities

Bits Output	Bits Output	Probability
$(\tau_{n-2} \text{ to } \tau_{n-1})$	$(\tau_{n-1} \text{ to } \tau_n)$	(P_{ijk})
0	0	a_{DD}
0	1	$a_{DT} + a_{DI} \frac{a_{ID}}{2}$
0	2	$a_{DI} \frac{a_{IT}}{2}$
1	0	$a_{TD} + a_{ID}a_{DD}$
1	1	$a_{TT} + a_{TI} \frac{a_{ID}}{2} + a_{ID} a_{DT} + a_{ID} a_{DI} \frac{a_{ID}}{2}$
1	2	$a_{TI} \frac{a_{IT}}{2} + a_{ID} a_{DI} \frac{a_{IT}}{2}$
2	0	$a_{IT}a_{TD}$
2	1	$a_{IT}a_{TT} + a_{IT}a_{TI}\frac{a_{ID}}{2}$
2	2	$a_{IT}a_{TI}\frac{a_{IT}}{2}$

table correspond to $I_m = 1$ and the table will differ depending on this parameter. The number of possibilities also increases as I_m increases as more potential events are now possible. Additionally, when the final event contains an insertion we scale the value over all possible bits i.e 2^m where m once again corresponds to m consecutive insertions. Note that the above notation is for the forward pass of the FB algorithm and the corresponding notation for the backward pass describes $\psi_n = k$, $\psi_{n+1} = j$ and $\psi_{n+2} = i$ as the synchronisation offset at τ_n , τ_{n+1} and τ_{n+2} respectively. Additionally, the time interval from τ_{n-2} to τ_{n-1} in Table 2 corresponds to time interval from τ_{n+1} to τ_{n+2} for the backward pass and similarly time interval from τ_{n-1} to τ_n will correspond to time interval from τ_n to τ_{n+1} . Likewise P_{ijk} in Table 2 corresponds to P_{kji} for the backward pass.

The final alteration is that there is no emission matrix for an observation given a current state. Rather, we use the watermark sequence and compare corresponding received bits to determine if a transmission or substitution has occurred. Again this shows the importance of the watermark sequence and the density of the sparse sequence. An altered FB algorithm is used to determine the likely states that the channel transitioned through. The forward probabilities at time n and state k is defined as $F_n(k) = P(\hat{t}_1, \hat{t}_2, ..., \hat{t}_{n-1+k}, \psi_n = k | A, P_s, w, \Pi)$ [15]. Similarly, the backward probabilities at time n for state k is defined as $B_n(k) = P(\hat{t}_{n+k}, \hat{t}_{n+k+1}, ..., \hat{t}_{\Gamma}, \psi_n = k | A, P_s, w)$

[15]. As mentioned previously, an efficient method to calculate these probabilities is with the recursive FB algorithm. The equations for the forward pass are shown in Equations (2) to (4) and the backward pass formulae are shown in Equations (5) to (7). Additionally, Equation (2)is used for the initialisation step and Equation (3) is used for the recursion in the first-order memoryless DM decoding. Similarly, Equation (5) is used for the initialisation step and Equation (6) is used for the recursion in the backward pass for the first-order memoryless DM decoder. To better illustrate the processes described, the algorithm for the FB passes for the proposed decoding scheme is outlined in Algorithm 2 in Appendix B. The probability of being in a given state at a given time, or the posterior state probabilities, is equal to the product of the forward and backward values at the same corresponding state and time. From these posterior state probabilities, the most likely channel path sequence is determined by finding the state with the maximum probability for each time index constrained within a range of $s_p - 1$ to $s_p + I_m$ where s_p is the most likely state at the previous time index.

for n = 1

$$F_1(k) = \pi_k \tag{2}$$

for n = 2

$$F_{2}(k) = \sum_{j=k-I}^{k+1} F_{1}(j) \left(\alpha_{jk} + \beta_{jk} \zeta_{k}^{1} \right)$$
(3)

for $3 \le n \le \Gamma$

$$F_n(k) = \sum_{i=j-I}^{j+1} \sum_{j=k-I}^{k+1} F_{n-1}(j) P_{ijk} \xi_k^{n-1}$$
(4)

for $n = \Gamma$

$$B_{\Gamma}(k) = \begin{cases} 1 & k = \rho \\ 0 & otherwise \end{cases}$$
(5)

for $n = \Gamma - 1$

$$B_{\Gamma-1}(k) = \sum_{j=k-1}^{k+I} B_{\Gamma}(j) \left(\alpha_{kj} + \beta_{kj} \zeta_k^{\Gamma-1} \right)$$
(6)

for $\Gamma - 2 \ge n \ge 1$

$$B_n(k) = \sum_{i=j-1}^{j+I} \sum_{j=k-1}^{k+I} B_{n+1}(j) P_{kji} \xi_k^n$$
(7)

Here α_{jk} , β_{jk} and ζ_k^n are further elaborated in Equations (8),(9) and (10) respectively [15]. ξ_k^n in Equations (4) and (7) are equivalent to ζ_k^n in Equation (10), however, it only affects the probability in question if the second time interval ends in a Transmission. For example, the calculation for $\psi_{n-2} = 1$, $\psi_{n-1} = 0$ and $\psi_n = 0$, the channel

emits 0 bits for the time interval τ_{n-2} to τ_{n-1} and 1 bit for τ_{n-1} to τ_n . This means that the overall $P_{ijk}\xi_k^{n-1}$ for this case would be $a_{DT}\xi_0^n + a_{DI}\frac{a_{ID}}{2}$. In other words the value of ξ_k^{n-1} , which checks if the received bit matches the corresponding watermark bit, is only used if there was in fact a transmission event as the final respective occurrence.

$$\alpha_{jk} = \begin{cases} \frac{P_i^{k-j+1}P_d}{2^{k-j+1}} & -1 \le k-j < I\\ 0 & k-j < -1, k-j \ge I \end{cases}$$
(8)

$$\beta_{jk} = \begin{cases} \frac{P_i^{k-j}P_t}{2^{k-j}} & 0 \le k-j < I\\ \frac{P_i^I \hat{P}_t}{2^{k-j}} & k-j = I\\ 0 & k-j \le -1, k-j > I \end{cases}$$
(9)

$$\zeta_k^n = \begin{cases} 1 - P_f & \hat{t}_{n+k} = w_n \\ P_f & \hat{t}_{n+k} = w_n \oplus 1 \end{cases}$$
(10)

4. Simulations and Results

As previously mentioned, all simulation results from the proposed model and code construction are benchmarked against the first and second-order memoryless DM decoding. Algorithm 1 in Appendix B shows the pseudo-code for the first-order DM FB algorithm. We expand the firstorder decoder to a second-order, providing a more objective comparison to the proposed scheme as two-time intervals are now used to evaluate the forward and backward values. Equation (11) provides the memoryless secondorder recursive forward pass while Equation (12) describes the memoryless second-order recursive backwards pass. The initialisation steps follow the same process as the proposed FSMC decoder. The second-order DM FB algorithm pseudo-code is given in Algorithm 3 in Appendix B.

$$F_{n}(k) = \sum_{i=j-I}^{j+1} \sum_{j=k-I}^{k+1} F_{n-1}(j) \left(\alpha_{ij} + \beta_{ij}\right) \left(\alpha_{jk} + \beta_{jk} \zeta_{k}^{n-1}\right)$$
(11)

$$B_{n}(k) = \sum_{i=j-1}^{j+I} \sum_{j=k-1}^{k+I} B_{n+1}(j) \left(\alpha_{ji} + \beta_{ji}\right) \left(\alpha_{kj} + \beta_{kj} \zeta_{k}^{n}\right)$$
(12)

4.1. Analysis Metrics

Three evaluation metrics are used in the analysis of the simulation results. The first is the commonly used BER values for a given system entropy. While this provides a good insight into the system's performance, there are some minor drawbacks. The BER is calculated at the final output of the system; in other words, \hat{d} is compared against d to determine the equivalent BER. This requires a complete resynchronisation of the received sequence. Consequently,

if a deletion is detected during the inner decoding process, a bit is inserted into the received sequence at the corresponding time index to rectify this influence. In real applications, the rectified bit is random and may sometimes produce the correct bit that was deleted. Other times this may be the incorrect bit, which will alter the BER values and thus produce varying results for the different algorithms. For the analysis in this paper, the bit inserted to undo a possible deletion is always a '0' to ensure a fair comparison of the BER analysis for the different decoders. The bias caused is still, however, noted in this resynchronisation step which ultimately biases the algorithms BER performance. The second evaluation metric used is the number of positions that the derived hidden states disagree with the actual states that the channel traversed - we call this the NIIS or Number of Incorrectly Identified States. This indicates how well the respective FB algorithms could accurately deduce the correct hidden states. However, it suffers from the fact that the correct transitions between states may be correct, but the actual state (offset) is incorrect. Here a lower number indicates a more accurate representation of what truly transpired in the channel. For the results presented in this paper, the NIIS is normalised over the length of the transmitted sequence in order to provide a more general metric. Finally, an evaluation metric known as the Sum of Absolute Offset (SAO), which builds onto the idea of NIIS, is introduced. This metric is derived again at the inner decoder and looks at how the likely hidden state path produced from the algorithms differs from the actual path the channel traversed. Instead of looking at the number of individual states where the algorithm path matches the actual path (binary classification), we look at the difference between the actual path and algorithm-derived path at corresponding time indexes. This gives a more general overview and a value that better shows the performance of the associated algorithm where, again, the lower the SAO, the more aligned to the actual channel path. The SAO is calculated by Equation (13) where $P_{Actual}(\tau)$ and $P_{Alq}(\tau)$ are the actual state path produced by the channel and the derived state path inferred from the decoding algorithms respectively. In this paper, the results for the SAO plots are kept as absolute values as they closely correlate to what is observed in the NIIS, and this allows us to have a different perspective when viewing the results. It is noted that the SAO can, however, still be normalised across all possible offsets and the length of the transmitted sequence if needed.

$$SAO = \sum_{n=1}^{\Gamma} |P_{Actual}(n) - P_{Alg}(n)|$$
(13)

These evaluation metrics are plotted against a range of entropy values in the results. Entropy is used as a measure of how uncertain the communication channel is. In other words, we use entropy to quantify the harshness of the channel as it is a good indicator of channel capacity [9]. The average entropy is calculated by Equation (14) where the entropy of state i is H_i and is calculated by Equation (15) [22, 2]. The value of ρ_i in Equation 14 is the stationary distribution of been in that specific state. As different combinations of P_t , P_s , P_d , P_i and A may produce similar entropies, the justification of using entropy in this regard is to limit the number of these varying parameters in the analysis, which seeks to simplify a rather complex phenomena. As can be seen in the overall results, having varying parameters within certain ranges reliably reproduces relatively constant entropies. In this paper, three error ranges are used for the generation of the transition matrix entries, and this, in turn, corresponds to three different entropy ranges. These ranges are shown in Table A.1 in Appendix A along with the outline and method for creating the transition matrices. We, however, reiterate that the objective of this paper is to give a comparison of the decoding algorithms. While this matter is briefly discussed, the manner in which the different parameters contribute to the entropy is beyond the scope of this paper.

$$\overline{H} = \sum_{i=1}^{N} \rho_i H_i \quad bits/symbol \tag{14}$$

$$H_i = -\sum_{j=1}^{N} a_{ij} \log_2(a_{ij}) \quad bits/symbol \tag{15}$$

4.2. Overall Results

The following results are obtained to show the general trends in performance of the three decoding algorithms across various entropy values. To ensure the generality and demonstrate the practicality of the decoders, for these tests, multiple iterations are run on varying transition matrices that produce an entropy within 0.001 from the desired entropy value. For example, if the desired entropy is 0.01, transition matrices that produce entropies between 0.009 to 0.011 are considered. Twenty different transition matrices are generated for each entropy value from 0.01 to 0.3, according to the method outlined in Appendix A. For each of these matrices, communication across the channel is simulated 100 times with varying message data, and from this, the decoding performances are computed. The results are then averaged across all iterations to produce the following graphs and outcomes.

As shown in Figure 5, all the evaluation metrics follow similar trends for the three decoding algorithms. As the entropy values increases so too does the BER, NIIS and *SAO* which is shown in Figures 5a, 5b and 5c respectively. This is an expected outcome as an increase in entropy increases the uncertainty of what is occurring within the communication channel, and as a result, more errors are produced. These results agree with the notion of memory decreasing entropy and uncertainty and, consequently, increasing channel capacity [9]. At lower entropy values, the memory FSMC outperforms its memory-



(c) SAO performance for decoding algorithms

Figure 5: BER, NIIS and SAO performance for the DM, FSMC and second-order DM decoding algorithms across various entropy values using varying transition matrices for each of the entropy values

less counterpart. As the entropy increases, the memoryless algorithms perform slightly better. For almost all the simulations, the first- and second-order memoryless DM algorithms perform identically, but the second-order decoder performance is slightly better at the much higher entropies. What is advantageous from these results is that we see entropy is indeed a good indicator of the channel's performance. Even though different transition matrices that produce a given entropy are used, the results obtained are generally independent of the individual entries of A and the stationary distribution values. This is welcomed to demonstrate the real-world application of such a system as the results are not constrained to specific entries of Abut can rather be overviewed by an encompassing metric like entropy. We also note jumps in the trajectory of the plots in Figure 5a occurring at entropy values of 0.1 and 0.2. These discontinuities are attributed to the different error ranges outlined in Table A.1 in Appendix A. We propose that using a smaller resolution of changing error values will decrease this discontinuity, but this is left as a future exercise.



(b) NIIS performance for decoding algorithms

Figure 6: BER, NIIS and SAO performance for the DM, FSMC and second-order DM decoding algorithms across various entropy values



Figure 6: BER, NIIS and SAO performance for the DM, FSMC and second-order DM decoding algorithms across various entropy values (cont.)

4.3. Constant Entropy Results

The following results are obtained by running the three decoding algorithms on data passed through the channel for specific entropy values. Here a constant transition matrix and consequently constant entropy value within the desired range is used. This reduces the number of changing parameters, allowing a more thorough analysis of P_s , P_d and P_i values. For each entropy value, there are 5000 iterations of the channel use with random data bits for each run (all three algorithms are run using the same data and channel output for a true representation for the given iteration). For all entropies and iterations, the same watermark string is used to reduce the number of varying parameters. Figure 6a, Figure 6b and Figure 6c show the results of these tests for the BER, NNA and SAO respectively. The results obtained using set transition matrices for a given entropy range are in direct comparison to those obtained previously, where the transition matrix was varied for an entropy range. Additionally, it is noted that the results from the BER, NIIS and SAO are highly correlated and in agreement with one another and as such, the focus can remain on the NIIS plots for further analysis. Figure 7 shows specific zoomed-in areas of the NIIS plots to illustrate further the results obtained. From Figure 7 it can be seen that using an entropy range from 0 to 0.1, the FSMC memory decoding appears to have slightly better performance than both the DM and second-order DM decoding. From an entropy range of 0.11 to 0.2, it is evident that all of the decoders have similar performance and achieve almost identical results. Lastly, for entropies ranging from 0.2 to 0.3, it is seen that both the DM decoders perform better than the FSMC in most cases. We also see the second-order DM decoding slightly outperforming the first-order DM algorithm at these higher entropy values. It is also worth noting that while the proposed model and scheme perform only slightly better, within lower entropy

ranges, than the memoryless counterparts, there is almost no increase in the algorithm's complexity. Additionally, using the more sophisticated FSMC memory model allows for more realistic channels to be simulated and coded for. The emphasis is once again placed on providing an effective model and code construction to correct and protect against synchronisation errors in a channel that contains memory. It is easily seen that when the proposed scheme does not perform better than the existing approximations, it is at the very least comparable to them.



Figure 7: NIIS performance for decoding algorithms with zoomed areas

4.4. Error Probability Level Results

We can further our analysis of the decoding algorithms by taking a closer look at the parameters that contribute to the entropy values and thus affect decoder performance. For the following results, we use the constant entropy data to keep the relevant stationary distributions for a given transition matrix constant. Figure 8 shows the stationary distribution values in yellow against the actual respective frequencies of P_d , P_i and P_s for given entropies. Note that the stationary distribution error probability is scaled to match the maximum frequency of the DM or FSCM plots but has no actual frequency per se. We limit the analysis to certain selected entropy values (H = 0.014,(0.074, 0.182, 0.292) to illustrate the points. This analysis, however, can be extended to any of the entropy values discussed. From these results, we identify general themes and trends on how the P_d , P_i and P_s values affect the decoder performance but note that this is a high-level analysis, and further investigation is required to better identify the prevailing effects of the parameters. In each of the subplots shown in Figure 8, graphs of the number of times the first-order DM algorithm performs better (blue), the FSCM decoding performs better (red), and the times both the first-order DM and FSCM decoding have identical decoding performance (purple) are plotted. This analysis is

based in terms of the NIIS performance metric. From Figure 8a, it can be seen that for the majority of cases in the simulated channel use, no errors - be it deletion, insertion or substitution - are witnessed, and it is shown that both the DM and FSCM perform equally for the majority of the simulation runs. There are cases where the DM decoder outperforms the FSCM as it is better equipped to handle no error situations. This is because the FSCM model accounts for memory in the channel and almost always allows for some transition to an error state and consequently has higher probabilities of detecting an error even if none are present. It is worth noting that this does not imply the FSCM is a bad decoder but rather suggests its usefulness is better suited to cases where errors are, in fact, present. In contrast, the DM decoder should be used if the channel is relatively stable and causes little to no synchronisation errors. It can also be argued that a backwards error correction technique such as automatic repeat request would be better suited in these low-error cases than trying to correct these minimal errors. Figure 8a shows the outperformance frequency plots at an entropy of H = 0.074. It is evident that most error probabilities produced by the channel are scattered around the relevant stationary distribution values. When the respective channel error is less than the stationary distribution, it is observed that the DM decoder tends to outperform the FSCM decoder. However, as the probability of the error in question surpasses its corresponding stationary distribution, we notice that the FSCM starts to have a larger number of occurrences where it performs better than the DM decoder. Similar trends are noticed at entropies of 0.182 and 0.292, especially concerning the insertion probability graphs. The plots for the entropies of 0.182 and 0.292 are shown in Figure 8c and 8d respectively. It is also noticed that at these higher entropy values, the probability of deletion and substitution caused by the channel spans a larger range of values. In contrast, the range of the insertion probabilities remains relatively constant. Also noted at the higher entropy values is that the decoders are very seldom in agreement with one another, and one decoding algorithm generally outperforms the other at these points. It is still observed that a fair majority of the error probabilities produced by the channel are generally centred around the relevant stationary distribution error probability. For these results and analysis, the second-order DM decoding has been omitted as it closely follows the performance of the first-order decoding for the cases in question.

4.5. Effects of P_s on simulations

As outlined in the proposed system, the entropy values calculated are based on the converted 3-state FSMC and not the original 4-state. In other words, the substitution state is omitted from this calculation as it is converted to a memoryless error value for the discussed system. The following test provides a way to determine the effect the value of P_s has on the system. In these tests, the same



Figure 8: Frequency decoders outperform each other at various channel error probabilities

transition matrix is used for a given entropy value; however, the value of P_s is varied, and the NIIS values are recorded. The tests show results in an entropy range from 0.01 to 0.1 using P_s values of 0.0017, 0.0033, 0.005 and 0.0067 which corresponds to 1, 2, 3 or 4 substitution errors in the given system. It is worth noting that this is a general excerpt of the results, and higher entropy values will tend to see higher P_s values. Additionally, for given parameters, a certain P_s value may occur more often than others in actual channel usage, as is shown in Figure 8. As such, the results for this set of tests may contain slight biases. However, these values are normalised according to the number of times a certain P_s value occurred as this tries to minimise the bias. Figure 9 shows that the value of P_s has a rather random effect on the NIIS, with the system performing better for different cases of P_s and entropy values. However, the normalised NIIS will generally be in a similar region with slight differences for varying P_s values at a set entropy. There are, of course, outliers

which are evident at an entropy of 0.0737 where the NIIS associated with the higher P_s value of 0.005 has a better performance. However, this is due to the aforementioned bias in the number of occurrences of that specific P_s value. In this case there are 10 counts of $P_s = 0.005$, 3693 counts of $P_s = 0.1135$ counts of $P_s = 0.0017$ and 162 counts of $P_s = 0.0033$.

5. Conclusion

The DM synchronisation coding and decoding scheme are discussed along with the FSMC synchronisation channel, which can model an IDS channel while incorporating memory aspects of the channel. By integrating ideas from both schemes and channels, a novel memory synchronisation system is developed and introduced, which is capable of correcting insertion and deletion errors. Various tests and simulations are done using a variety of evaluation metrics, and it is shown that the new proposed decoder performs similar to that of the original first-order and second-



Figure 9: Effect of P_s on NIIS

order DM decoder. It is even shown that for specific entropy values (0.01 to 0.1), the proposed decoder performs slightly better than the DM counterparts. A further analysis that delves into the individual error probabilities and their effect on decoder performance is conducted, which reveals a generally better performance from the FSMC proposed decoding when error values exceed that of the corresponding stationary distribution error values. While the coding gain may be insubstantial, the proposed channel is more indicative of real-world communication channels, especially in harsher environments, as errors are more likely to be correlated. Thus the code construction and decoding will likely prove useful for such applications.

CRediT authorship contribution statement

Shamin Achari: Conceptualisation, Methodology, Software, Formal analysis, Investigation and Writing - Original Draft. **Ling Cheng**: Supervision, Conceptualisation and Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Numbers: 148765, 132651 & 129311).

Appendix A. Generation of Transition Matrices

This appendix outlines the procedure to generate various transition matrices which correspond to different ranges of entropy values. The stationary distributions of the fourstate matrix, as well as the actual transition matrix entries, affect the channel entropy. The matrix created is the four-state version which is then converted to a corresponding three-state matrix using the procedure outlined in the main document. In order to create the four-state matrix, different transition probabilities are set up. This is broken into two large groups, firstly a range of probabilities corresponding to a transition from a Transmission to an Error where an error constitutes either substitution, insertion or deletion states. The second range of probabilities is that of Error to Error. Table A.1 shows the different error ranges used to generate the matrix entries and the related range of entropy values it produces. Each entry in the matrix is randomly generated within these predefined error ranges and finally, for each row, the entry corresponding to a final Transmission state is calculated by taking 1 minus the other entries in the row. This is done to ensure all possible transitions in a given row equal unity. Once the matrix is created, its equivalent 3-state form is determined and the entropy of the 3-state model is calculated and used in accordance with the various simulations and tests. As is evident from the values in Table A.1, the higher the probability of staying in a transmission state (lower Transmission to Error and lower Error to Error probability), the lower the entropy. This again intuitively makes sense as the lower the entropy, the less uncertainty is found in the channel. As the entries in the transition matrix become more alike to each other, the entropy increases and it becomes more difficult to determine the correct state as the uncertainty of the channel state is much higher.

Table A.1: Entropy Ranges and Corresponding Transition Matrix Entries

Entropy Range	Transmission to Error Probabilty	Error to Error Probabilty
0.01 - 0.1	0.0001 - 0.005	0.001 - 0.05
0.1 - 0.2	0.001 - 0.05	0.01 - 0.05
0.2 - 0.3	0.01 - 0.05	0.001 - 0.05

Appendix B. Pseudocode for Decoders Forward-Backward Algorithms

This appendix outlines the various forward-backward algorithms used in the testing of the decoders. This allows us to easily illustrate and compare the similarities, and consequently, differences, between the various algorithms used. Algorithm 1 describes the forward-backward process for the bit level first-order Davey-MacKay decoder. Algorithm 2 shows the pseudo-code the forward-backward function for the finite state Markov channel decoder. Lastly, Algorithm 3 illustrates the forward-backward process for the bit level second-order Davey-MacKay decoder.

Algorithm 1 : First-Order DM Forward-Backward Algorithm

1:	function FORWARDBACKWARDDM(maximum offset
	x_{max} , maximum consecutive insertions I)
	$nStates = [-x_{max},, x_{max}]$
2:	$F[nStates, \Gamma],$
3:	$B[nStates, \Gamma],$
4:	$FB[nStates, \Gamma]$
5:	for all states $s \in nStates$ do
6:	$F[s,1] \leftarrow \pi[s] $ \triangleright Forward Initialisation
7:	end for
8:	for all time $\tau \in [2,, \Gamma]$ do \triangleright Forward Recursion
9:	for each current state $j \in nStates$ do
10:	for each previous state i from $j - I$ to $j + 1$
	do
11:	if $i \in nStates$ then
12:	$F[j,\tau] \leftarrow \sum F[i,\tau-1](\alpha_{ij}+\beta_{ij}\zeta_j^{\tau-1})$
13:	end if
14:	end for
15:	end for
16:	Normalise F across all rows at current τ
17:	end for
18:	$B[\rho,\Gamma] \leftarrow 1 \qquad \qquad \triangleright \text{ Backward Initialisation}$
19:	for all time $\tau \in [\Gamma - 1,, 1]$ do \triangleright Back Recursion
20:	for all current states $j \in nStates do$
21:	for each next state $i \in [j - 1, j + I]$ do
22:	if $i \in nStates$ then
23:	$B[j,\tau] \leftarrow \sum B[i,\tau+1](\alpha_{ji}+\beta_{ji}\zeta_j^{\tau})$
24:	end if
25:	end for
26:	end for
27:	Normalise B across all rows at current τ
28:	end for
29:	for all $s, \tau \in nStates$ do \triangleright FB Calculations
30:	$FB[s,\tau] \leftarrow F[s,\tau] * B[s,\tau]$
31:	end for
32:	Normalise FB across all rows at given τ
33:	return FB
34:	end function

References

- S. Achari, A. Y. Yang, J. Goodhead, B. Swanepoel, L. Cheng, Self-Synchronising On-Off-Keying Visible Light Communication System for Intra and Inter-Vehicle Data Transmission, arXiv preprint arXiv:2101.05126 (2021).
- [2] S. Achari, D. G. Holmes, L. Cheng, Symbol-Level Synchronisation Channel Modelling With Real-World Application: From Davey-Mackay, Fritchman to Markov, IEEE Access (2021).
- [3] D. Kracht, S. Schober, Using the Davey-Mackay code construction for barcodes in DNA sequencing, in: 2014 8th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), IEEE, 2014, pp. 142–146.
- [4] D. Kracht, S. Schober, Insertion and deletion correcting DNA barcodes based on watermarks, BMC bioinformatics 16 (1) (2015) 1–14.
- [5] M. C. Gursoy, M. Nasiri-Kenari, U. Mitra, Towards high datarate diffusive molecular communications: A review on performance enhancement strategies, Digital Signal Processing (2021) 103161.
- [6] J. T. Gómez, K. Pitke, L. Stratmann, F. Dressler, Age of information in molecular communication channels, Digital Signal Processing 124 (2022) 103108.
- [7] M. C. Davey, Error-correction using low-density parity-check codes, Ph.D. thesis, University of Cambridge (2000).
- [8] M. C. Davey, D. J. MacKay, Reliable communication over channels with insertions, deletions, and substitutions, IEEE Transactions on Information Theory 47 (2) (2001) 687–698.
- [9] L. N. Kanal, A. Sastry, Models for channels with memory and their applications to error control, Proceedings of the IEEE 66 (7) (1978) 724–744.
- [10] M. Cheraghchi, J. Ribeiro, An overview of capacity results for synchronization channels, IEEE Transactions on Information Theory 67 (6) (2020) 3207–3232.
- [11] M. Mitzenmacher, A survey of results for deletion channels and related synchronization channels, Probability Surveys 6 (2009) 1–33.
- [12] B. Haeupler, A. Shahrasbi, Synchronization strings and codes for insertions and deletions–a survey, IEEE Transactions on Information Theory (2021).
- [13] P.-M. Nguyen, M. A. Armand, T. Wu, On the watermark string in the Davey-Mackay construction, IEEE communications letters 17 (9) (2013) 1830–1833.
- [14] D. Leigh, Capacity of Insertion and Deletion channels, Project Report, available at http://www. inference. phy. cam. ac. uk/is/papers (2001).
- [15] D. J. Coumou, G. Sharma, Insertion, deletion codes with feature-based embedding: a new paradigm for watermark synchronization with applications to speech watermarking, IEEE Transactions on Information Forensics and Security 3 (2) (2008) 153–165.
- [16] F. Wang, Coding for Insertion/Deletion Channels, Ph.D. thesis, Arizona State University (2012).
- [17] J. A. Briffa, H. G. Schaathun, S. Wesemeyer, An improved decoding algorithm for the Davey-Mackay construction, in: 2010 IEEE International Conference on Communications, IEEE, 2010, pp. 1–5.
- [18] X. Jiao, M. A. Armand, Interleaved LDPC codes, reducedcomplexity inner decoder and an iterative decoder for the Davey-Mackay construction, in: 2011 IEEE International Symposium on Information Theory Proceedings, IEEE, 2011, pp. 742–746.
- [19] X. Jiao, M. A. Armand, Soft-input inner decoder for the Davey-Mackay construction, IEEE communications letters 16 (5) (2012) 722–725.
- [20] L. Rabiner, B. Juang, An introduction to hidden Markov models, ieee assp magazine 3 (1) (1986) 4–16.
- [21] D. Jurafsky, J. H. Martin, Speech and language processing, Vol. 3, Pearson London, 2014.
- [22] K. Shanmugam, Digital and analog communication systems, New York: Wiley, 1979.

Algorithm 2 : FSMC Forward-Backward Algorithm

```
1: function FORWARDBACKWARDFSMC(maximum offset x_{max},
                                                                                     27:
                                                                                              end for
    maximum consecutive insertions I)
                                                                                     28:
                                                                                              B[\Psi, \Gamma] \leftarrow 1
                                                                                                                                       ▷ Backward Initialisation
         nStates = [-x_{max}, ..., x_{max}]
                                                                                     29:
                                                                                              for all current states j \in nSt states do
2:
         F[nStates, \Gamma],
                                                                                     30:
                                                                                                  for each next state i \in [j - 1, j + I] do
                                                                                                      if i \in nStates then

B[j, \Gamma-1] \leftarrow \sum B[i, \Gamma](\alpha_{ji} + \beta_{ji}\zeta_j^{\Gamma-1})
        B[nStates, \Gamma],
                                                                                     31:
3:
4
        FB[nStates, \Gamma]
                                                                                     32:
        for all states s \in nStates do
                                                                                     33:
                                                                                                      end if
5:
                                                                                                  end for
            F[s,1] \leftarrow \pi[s]
                                                   ▷ Forward Initialisation
                                                                                     34:
6:
        end for
7:
                                                                                     35:
                                                                                              end for
                                                                                              Normalise B across all rows at current \tau
8:
        for each current state j \in nStates do
                                                                                     36:
            for each previous state i from j-I to j+1~{\bf do}
                                                                                                                                                ▷ Back Recursion
9:
                                                                                     37:
                                                                                              for all time \tau \in [\Gamma - 2, ..., 1] do
10:
                if i \in nStates then
                                                                                     38:
                                                                                                  for all current states k \in nStates do
                    F[j,2] \leftarrow \sum F[i,1](\alpha_{ij} + \beta_{ij}\zeta_i^1)
                                                                                     39:
                                                                                                      for each \tau + 1 state j \in [k - 1, k + I] do
11:
                                                                                                          for each \tau + 2 state i \in [j - 1, j + I] do
                end if
12:
                                                                                     40:
13:
            end for
                                                                                     41:
                                                                                                              if i \& j \in nStates then
                                                                                                                  B[k,\tau] \leftarrow \sum B[j,\tau+1](Prob(k,j,i))
         end for
                                                                                     42:
14:
15:
         Normalise F across all rows at current \tau=2
                                                                                          Here Prob() is the relevant probabilities from Table 2
16:
         for all time \tau \in [3, ..., \Gamma] do
                                                       ▷ Forward Recursion
                                                                                     43:
                                                                                                              end if
            for each current state k \in nStates do
                                                                                                          end for
                                                                                     44:
17:
                for each \tau - 1 state j from k - I to k + 1 do
                                                                                     45:
                                                                                                      end for
18:
19:
                    for each \tau - 2 state i from j - I to j + 1 do
                                                                                                  end for
                                                                                     46:
                        if i \& j \in nStates then
20:
                                                                                     47:
                                                                                                  Normalise B across all rows at current \tau
                             F[k,\tau] \leftarrow \sum F[j,\tau-1](Prob(i,j,k))
21:
                                                                              ⊳
                                                                                     48:
                                                                                              end for
    Here Prob() is the relevant probabilities from Table 2
                                                                                     49:
                                                                                              for all s, \tau \in nStates do
                                                                                                                                               \triangleright FB Calculations
22:
                         end if
                                                                                     50:
                                                                                                  FB[s,\tau] \leftarrow F[s,\tau] * B[s,\tau]
23:
                    end for
                                                                                     51:
                                                                                              end for
                end for
                                                                                              Normalise FB across all rows at given \tau
24:
                                                                                     52:
25:
             end for
                                                                                     53:
                                                                                              \mathbf{return} \ \mathrm{FB}
            Normalise F across all rows at current \tau
                                                                                     54: end function
26:
```

Algorithm 3 : Second-Order DM Forward-Backward Algorithm

1:	function 2NDORDERFORWARDBACKWARDDM(maximum offset
	x_{max} , maximum consecutive insertions I)
	$nStates = [-x_{max},, x_{max}]$
2:	$F[nStates, \Gamma],$
3:	$B[nStates, \Gamma],$
4:	$FB[nStates, \Gamma]$
5:	for all states $s \in nStates$ do
6:	$F[s,1] \leftarrow \pi[s]$ \triangleright Forward Initialisation
7:	end for
8:	for each current state $j \in nStates$ do
9:	for each previous state i from $j - I$ to $j + 1$ do
10:	if $i \in nStates$ then
11:	$F[j,2] \leftarrow \sum F[i,1](\alpha_{ij} + \beta_{ij}\zeta_j^1)$
12:	end if
13:	end for
14:	end for
15:	Normalise F across all rows at current τ
16:	for all time $\tau \in [3,, \Gamma]$ do \triangleright Forward Recursion
17:	for each current state $k \in nStates$ do
18:	for each $\tau - 1$ state j from $k - I$ to $k + 1$ do
19:	for each $\tau - 2$ state <i>i</i> from $j - I$ to $j + 1$ do
20:	$\mathbf{if} i \& j \in nStates \mathbf{then}$
21:	$F[k,\tau] \leftarrow \sum F[j,\tau-1]((\alpha_{ij}+\beta_{ij})*\alpha_{jk}+$
	$\beta_{ik}\zeta_k^{\tau-1}$
22:	end if
23:	end for
24:	end for
25:	end for
26:	Normalise F across all rows at current τ

27:end for $B[\Psi,\Gamma] \leftarrow 1$ 28:▷ Backward Initialisation for each current state $j \in nStates$ do 29: 30: for each next state *i* from j - 1 to j + I do $\mathbf{if}~i \in nStates~\mathbf{then}$ 31: $B[j, \Gamma - 1] \leftarrow \sum [j, \Gamma](\alpha_{ji} + \beta_{ji}\zeta_i^{\Gamma - 1})$ 32: 33: end if 34:end for 35: end for Normalise B across all rows at current τ 36: 37: for all time $\tau \in [\Gamma - 2, ..., 1]$ do \triangleright Back Recursion 38: for all current states $k \in nSt$ states do for each $\tau+1$ state $j\in [k-1,k+I]$ do 39:for each $\tau + 2$ state $i \in [j - 1, j + I]$ do 40: 41: if $i \& j \in nStates$ then $B[k,\tau] \leftarrow \sum B[j,\tau+1]((\alpha_{ji}+\beta_{ji})*\alpha_{kj}+$ 42: $\beta_{kj}\zeta_k^{\tau}$) 43: end if end for 44: 45:end for end for 46: 47:Normalise B across all rows at current τ 48: end for for all $s, \tau \in nStates$ do \triangleright FB Calculations 49: 50: $FB[s,\tau] \leftarrow F[s,\tau] * B[s,\tau]$ 51:end for Normalise FB across all rows at given τ 52: return FB 53:54: end function

⊳