ECE-493
FINAL REPORT

FIR Channel Blind Equalization

Abstract

Multipath interference and symbol detection are crucial problems in communication systems. In an ideal situation, the signal is transmitted through a single transmission path to the receiver and the receiver should receive the same signal, provided that there is no noise and attenuation in the channel. Realistically, the signal propagates through different transmission paths due to the presence of physical medium. The receiver will then receive the delayed version of the signal multiple times, which cause distortion and make it difficult to recognize the original signal. This is known as multipath interference. Our project is a solution which performs both the reversal of the distortion caused by the multipath interference as well as the symbol detection. In this report we first give the problem analysis and description of the solution. Then, we provide the system models and system design along with the mathematical background. Next, we demonstrate our experiments and results. Finally, we give insight to other issues. Lastly, we include our project progress and what we learned from this project.

Date of Submission: 12/07/2015
Faculty Supervisor: Dr. Yariv Ephraim

Team Members: Qing Chen
Xiaolong Fang
Yian Hu
Stephen Owusu
Liem Tran
Nguyen Tran

Executive Summary

Equalization and symbol detection are typical problems in the design of communication systems. Synchronization for the transmitter and the receiver is assumed in this project. In a communication system, when the analog signals passes through a channel, distortion will occur at the receiver side due to multipath interference. To overcome the multipath interference, an equalizer must to be designed to offset the distortion. In this project, we assume our signal pass through an unknown channel in the presence of additive Gaussian white noise. The main goal of this project is to study and test a Hidden Markov Model (HMM) method that is used for channel parameter estimation and information symbols detection in an unknown FIR channel. The method uses the Baum-Welch algorithm for the channel estimation. It also uses the maximum a-posterior (MAP) decision rule for symbol detection. The receiver model was first implemented in Matlab using known transmitted data to identify the channel. Then a communication system, with unknown data and parameters, was implemented. The two systems were studied and compared, in order to determine the relative accuracy of the system in which both transmitted data and channel impulse response are unknown. Both a linear and a non-linear channels are studied in our project. The approach will be tested primarily for a linear channel with memory length equals one. Then we extended the project to the case where the channel has larger memory.

Contents

1 Approach

1.1 Problem Analysis

Multipath interference and symbol detection are crucial problems in communication systems. A detailed description and modeling on both problems can be found in [1]. Ideally, in communication systems, the signal is transmitted through a single transmission path to the receiver and the receiver should receive the exact same signal, provided that there is no noise and attenuation in the channel. However, in reality, the signal propagates through different transmission path due to the presence of physical medium, atmospheric effects, and so on. This means the receiver can receive one signal delayed multiple times, which cause distortion and make it difficult to recognize the original signal in the receiver. In this case, what is received by the receiver can be modeled by a combination of differently delayed and scaled signal. When there is thermal noise at the receiver side, the receiver should make a decision on which of information symbols are transmitted at each sampling time instance.

Our project provided a solution which performs both the reversal of the distortion caused by the multipath interference as well as the symbol detection. In our project, the information symbols are transmitted through an FIR blind channel, and received by the receiver with additive white Gaussian noise (AWGN). The goal of the project is to estimate the channel parameters first, and then recover the information symbols. The project was inspired by two of Dr. Yariv Ephraims classes, Communication and Information Theory, and Introduction to Random Processes. The classes also provided the necessary background for the project. The project aims at performance evaluation, which is simulated in Matlab for both linear and non-linear channels.

1.2 Description of the Solution

The solution to the problem consists of two parts which are based on Hidden Markov Model (HMM). In the first part, we use the Baum-Welch algorithm to estimate the channel parameters iteratively until convergence occurs. In the second part, we use the estimated channel parameters to detect the transmitted information symbols using the maximum a-posteriori (MAP) decision rule. The two algorithms are simulated in Matlab for performance evaluation.

There is an alternative solution for the problem, which uses training sequence to detect channel parameters. The training sequence are determined and known to both the sender and the receiver in advance. However, this design requires extra bandwidth to send the training sequence, otherwise it cannot be used for a time-varying channel. This design is also simulated in Matlab. We make a comparison of this design and our original design.

1.3 Contributions

We have six members in our team. Each member made contributions to the project. Their specific contributions are shown as follows.

- Qing Chen is the technical leader of the project. He simulated and tested the designs, created the contents for each document, and educated team members on technical parts of the project. He also organized the schedule for each meeting and assigned task allocation for each member.
- Xiaolong Fang investigated the detailed design for the project. He designed the level-0, level-1, and level-2 schematics for the project. He also made the poster for the project.
- Yian Hu investigated the system model for the project. She studied the possibility of each transition branch and explained it to each team member in detailed.
- Stephen Owusu investigated the MAP decision rule. He studied the concepts of Bayes rule, and explained how to apply the MAP decision rule.

- Liem Tran is the project manager of the project. He proof-read reports from team members and integrated them in LaTex for documents submission. He also kept track of the progress made by each team member. As the project manager, he communicated with our supervisor and reserved rooms for our presentations.
- Nguyen Tran investigated the Hidden Markov Model (HMM). He studied the basic concepts of HMM and how to apply them to the project.

2 Technical Section

The detailed system model and algorithms can be found in [2]. In [3], a summary of the algorithms is provided. The basic concepts of HMM can be found in [4]. For all the discussions and demonstrations in this report, we define the power of information symbols over the noise variance as our signal to noise ratio (SNR). All the SNRs we use are in decibel (dB) unit.

2.1 System Models

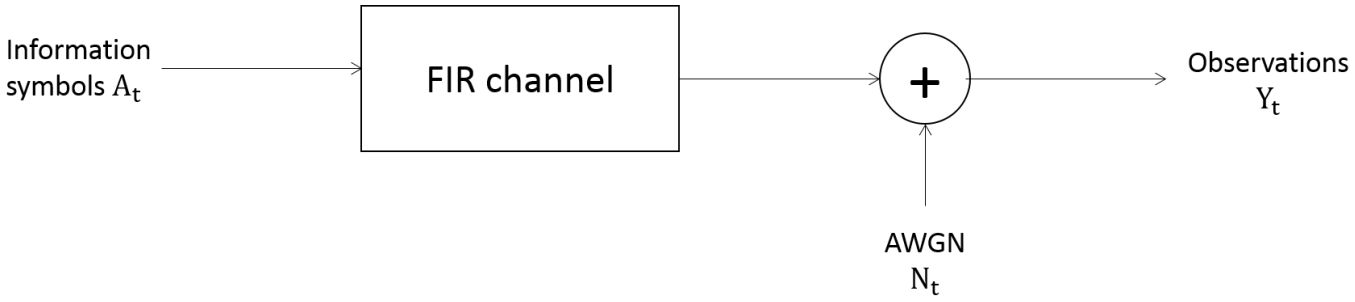


Figure 1: System model

The system model of this project is straightforward, as shown in Figure 1. A sequence of information symbols is transmitted through a FIR blind channel with additive white Gaussian noise (AWGN). The goal is to reveal the blind channel, and recover the information symbols at the receiver side based on the observations. From [2], we have the following formal definition for the system model:

1. $\mathcal{A}$: the sample space of transmitted information symbols.
2. $A_t : t = 1, 2, ...$: the transmitted information symbols which are independent and identically distributed random variables.
$$A_t \in \mathcal{A} = \{\alpha_m : m = 1, 2, ..., |\mathcal{A}|\},\ \alpha_m \in \mathcal{R} \tag{1}$$
3. $\mathcal{S}$: a state space where the state S_t at time t labels the channel memory $(A_{t-1}, A_{t-2}, ..., A_{t-L})$.
$$S_t \in \mathcal{S} = \{1, 2, ..., |\mathcal{S}|\},\ |\mathcal{S}| = |\mathcal{A}|^L \tag{2}$$
4. $\mathcal{X}$: the set of all possible transitions X_t from a state S_t to a state S_{t+1}.
$$X_t \in \mathcal{X} = \{\xi_n : n = 1, 2, ..., |\mathcal{X}|\},\ |\mathcal{X}| = |\mathcal{A}|^{L+1} \tag{3}$$
5. If the channel is linear, it can be determined by its discrete-time impulse response given by the vector
$$\mathrm{H} \triangleq (h_0, h_1, ...h_L)^{tr} \tag{4}$$
The output of the channel is then given by
$$\mathrm{h}(X_t) = \mathrm{H}^{tr} X_t = \sum_{i=0}^{L} h_i A_{t-i} \tag{5}$$
6. The data is processed in blocks. If the block size is small, the channel parameters can be regarded as constant. Therefore, with AWGN, the observation sequence of length T is given by.
$$Y_t = \mathrm{h}(X_t) + N_t, \quad t = 1, 2, ..., T \tag{6}$$
This sequence can be obtained by sampling at the Nyquist rate.

The Baum-Welch algorithm is an efficient algorithm that can estimate the parameters efficiently using the forward and backward recursion. From [2], we can summarize the contribution at time t of the past and of the future by the following definition:

$$\gamma_t(i) \triangleq \rho(y_1, y_2, ..., y_t, S_t = i) \tag{7}$$

$$\beta_{t+1}(j) \triangleq \rho(y_{t+1}, y_{t+2}, ..., y_T | S_{t+1} = j) \tag{8}$$

With $\gamma_1(i) = P[S_1 = i]$ and $\beta_T(j) = 1)$ for all i and j, we can deduce the following forward and backward recursions as shown by [2]:

$$\gamma_t(j) = \sum_{i \in S} \gamma_{t-1}(i) P[S_t = j | S_{t-1} = i] p(y_{t-1} | S_t = j, S_{t-1} = i) \tag{9}$$

$$\beta_t(i) = \sum_{j \in S} \beta_{t+1}(j) P[S_{t+1} = j | S_t = i] p(y_t | S_{t+1} = j, = S_t = i) \tag{10}$$

Then, the weighted conditional likelihood can be calculated as follows:

$$L(Y, X_t = \xi_n) = p(y_t | S_{t+1} = j, S_t = i) P[S_{t+1} = j | S_t = i] \gamma_t(i) \beta_t(j) \tag{11}$$

From [2], the Baum-Welch algorithm can be summarized as follows:

1. Use the forward recursion to calculate $\gamma_t(j)$.
2. Use the backward recursion to calculate $\beta_t(i)$.
3. Use $\gamma_t(j)$ and $\beta_t(i)$ to calculate $L(Y, X_t = \xi_n)$.
4. Use $L(Y, X_t = \xi_n)$ to calculate new re-estimations of the parameters $\widehat{\theta}$.
5. If $L(Y, \widehat{\theta})$ is greater than a predetermined threshold, then stop. Otherwise, go to step 1.

For this project, the re-estimation formulas, found in [2], are as follow:

$$h^{i+1}(\xi_n) = \frac{\sum_{t=1}^{T} L(Y, X_t = \xi_n; \theta^i) y_t}{\sum_{t=1}^{T} L(Y, X_t = \xi_n; \theta^i)} \tag{12}$$

$$(\sigma^2)^{i+1} = \frac{\sum_{t=1}^{T} \sum_{n=1}^{|X|} L(Y, X_t = \xi_n; \theta^i) |y_t - h^{i+1} \xi_n|^2}{\sum_{t=1}^{T} \sum_{n=1}^{|X|} L(Y, X_t = \xi_n; \theta^i)} \tag{13}$$

where $L(Y, X_t = \xi_n; \theta^i)$ is defined as in (11). For simplicity, the parameters θ will be omitted.
From [2] and [3], we can also calculate the estimate of the channel impulse response H^{i+1} by the following equation:

$$\{\sum_{t=1}^{T} \sum_{n=1}^{|X|} L(Y, X_t = \xi_n; \theta^i) \xi_n^* \xi_n^{tr}\} H^{i+1} = \sum_{t=1}^{T} \sum_{n=1}^{|X|} L(Y, X_t = \xi_n; \theta^i) y_t \xi_n^* \tag{14}$$

Suppose X is a discrete random variable where each realization represents the transmitted symbol of the communication system. X takes M value denoted by $\{, ..., M\}$. The value of X are transmitted over a channel whose output is Y. The MAP decision rule can be described as follows:

For a receive value y of Y, the optimal receiver calculates

$$L_m(y) := p_x(m) f_{Y|X}(y|m) \tag{15}$$

for every value of $m \in \{1, ...M\}$ and choose the value of m that maximizes $L_m(y)$ as the estimate of X:

$$\widehat{X} = argmax_{m \in \{1, ...M\}} \{\rho_x(m) f_{Y|X}(y|m)\} \tag{16}$$

This decision rule will be used for information symbol detection. [6]

In our project, the MAP decision rule is slightly different from equation (16) as we need to sum up all the branches that generate the same information symbol on a particular time instant t. From [2], the equation is given by

$$\widehat{A_t} = argmax_{\alpha_m \in A} L(Y, A_t = \alpha_m; \widehat{\theta}) \tag{17}$$

2.2 System Design

The system and algorithm simulated in Matlab is used for performance evaluation. Figure 2 shows the level-0 design of the Matlab simulation. There are five inputs and two outputs for the system.

The five inputs are:

1. Length of memory L
2. Block length T
3. Channel parameters for linear channel H or non-linear channel h
4. Initialization of the channel parameters H0 or h0
5. The power of information symbols to the noise variance ratio

And the two output are:

1. The true transmitted information symbols
2. The estimated information symbols

In addition, several plots are generated for the simulation, which are illustrated in section 3.

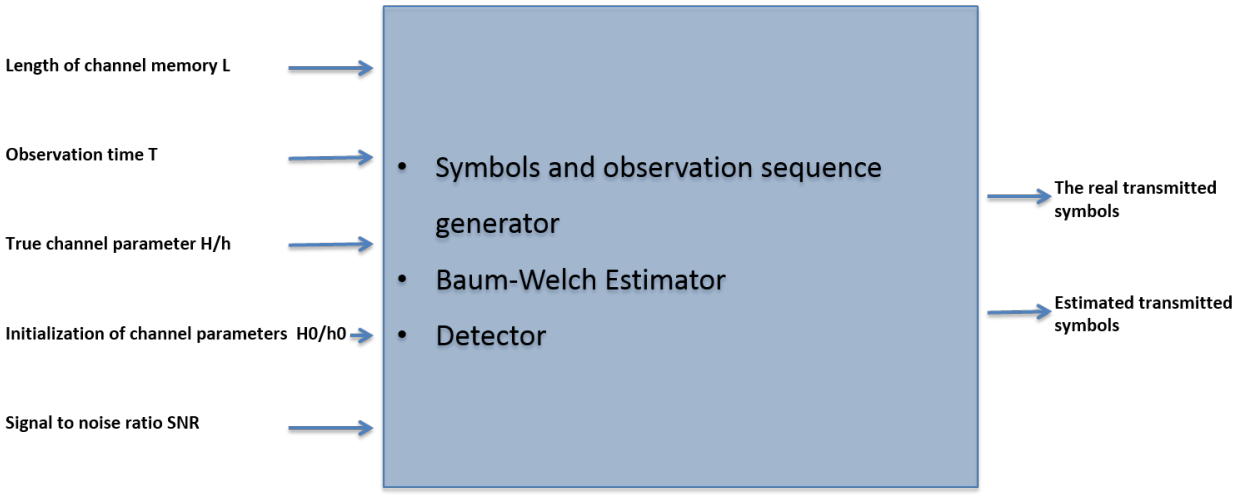


Figure 2: Level-0 design

As shown in Figure 3, the level-1 design of the system is straightforward. The observation sequence generator generates and outputs a random information symbols sequence for testing, and the related states information sequence and transition branches are generated. Then, given the specific channel parameters, the channel output and the observation sequences are generated accordingly to equation (5) and (6). After we get the observation sequence, it is used in the Baum-Welch estimator to estimate the channel parameters iteratively until convergence occurs. Then the converged parameters are used in the detector for symbol detection using MAP decision rule. Finally, the systems return the estimated transmitted symbols.

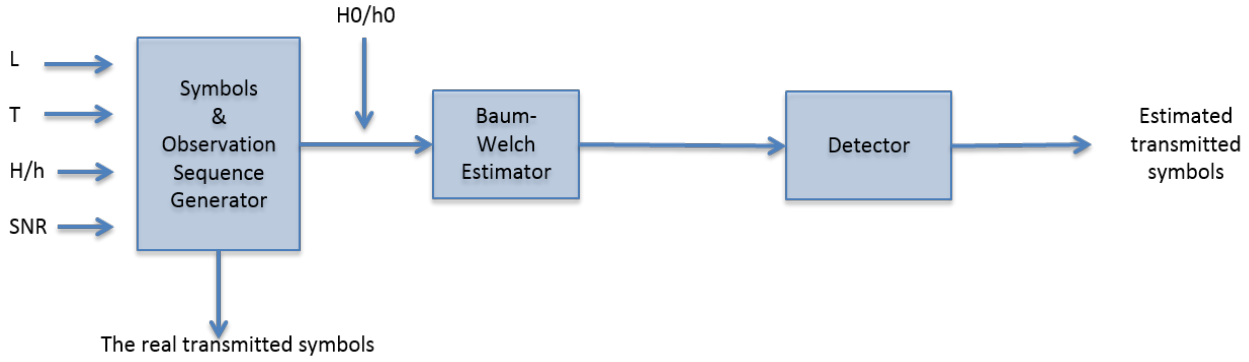


Figure 3: Level-1 design

Level-2 design reveals all the inner functions and their relations in the system, as illustrated in Figure 4. The observation sequence generator is the same as shown in the level-1 design. The Baum-Welch estimator consists five parts in the level-2 design. First, it uses the initialization of the channel parameters to generate the forward and backward functions according to equation (9) and (10), which can be used to calculate the weighted likelihood function in polynomial time using (11). Then the channel parameter estimator estimates a new parameter set using equation (12)-(14). If the parameter set converge, then it goes to the MAP decision rule function. Otherwise, the parameter set goes back to the forward and backward function blocks, and the same estimation process are performed again. Finally, the converged parameter set is used in the MAP decision rule detector, and the transmitted symbols are determined according to equation (17). Details of Baum-Welch algorithm and the MAP decision are demonstrated in section 2.1.

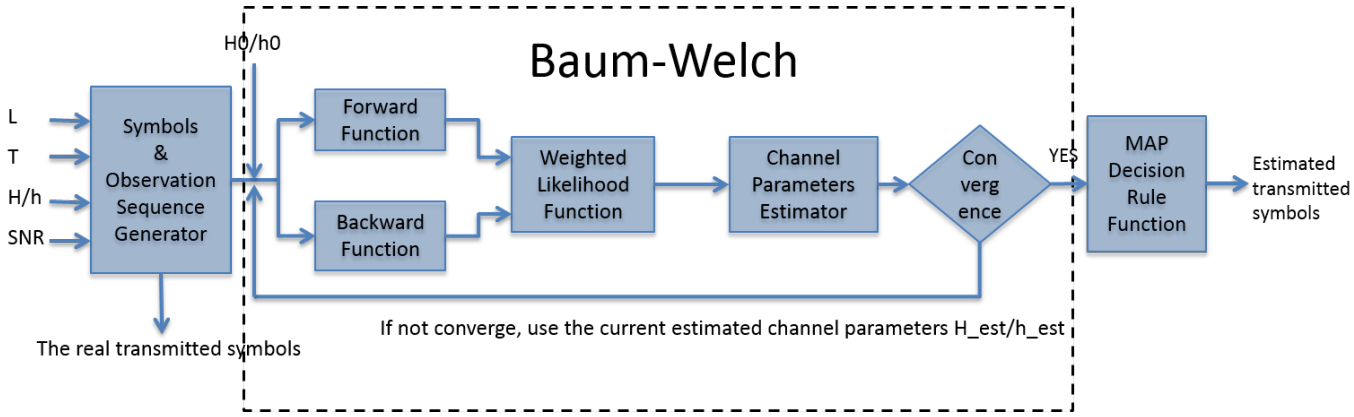


Figure 4: Level-2 design

3 Experiment

In our experiments, we tested the algorithms for both linear and non-linear channels. The algorithm is simulated for the binary information symbols in -1, 1. For simplicity, we assume the noise variance is known. The length of channel memory is L = 2. For the linear channel, we investigated the results when SNR = 8dB and 4dB. For the non-linear channel, we investigated the effects of different initializations of channel parameters under the same SNR of 8dB. We also compared the performance of the algorithm with the one using training sequence.

3.1 Experimentation for Linear Channel

For the linear channel, the impulse response we tested is composed of the same three coefficients from [2]: 0.5, 0.7, and 0.5. The block length is T = 100. We choose the initialization of the channel also the same as that of [2] parameters as 0, 0.1, and 0.

Figure 5 shows the observation sequence received by the receiver. The horizontal dash lines represent the true channel outputs of the system. It can be observed that the deviations between data points and those horizontal dash lines are not large when SNR = 8dB. Figure 6 shows the evolution of the channel parameters and also the weighted likelihood function. In this figure, the horizontal dash lines represent the true channel parameters. As can be seen in the figure, convergence occurs very quickly.

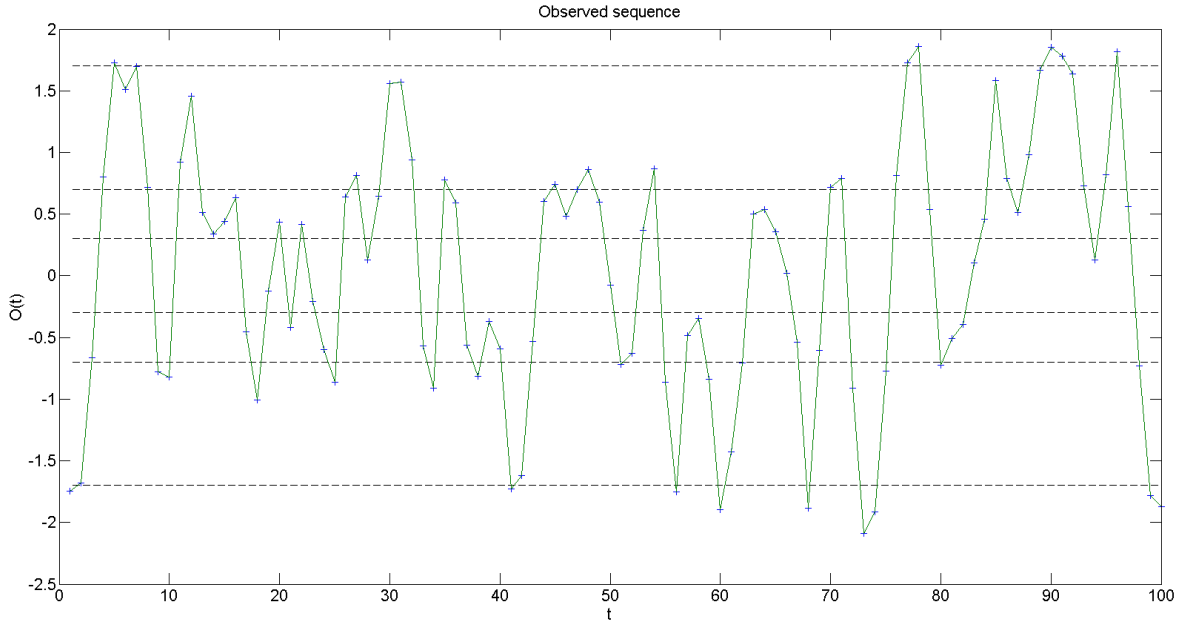


Figure 5: Observation sequence, linear channel, SNR = 8dB

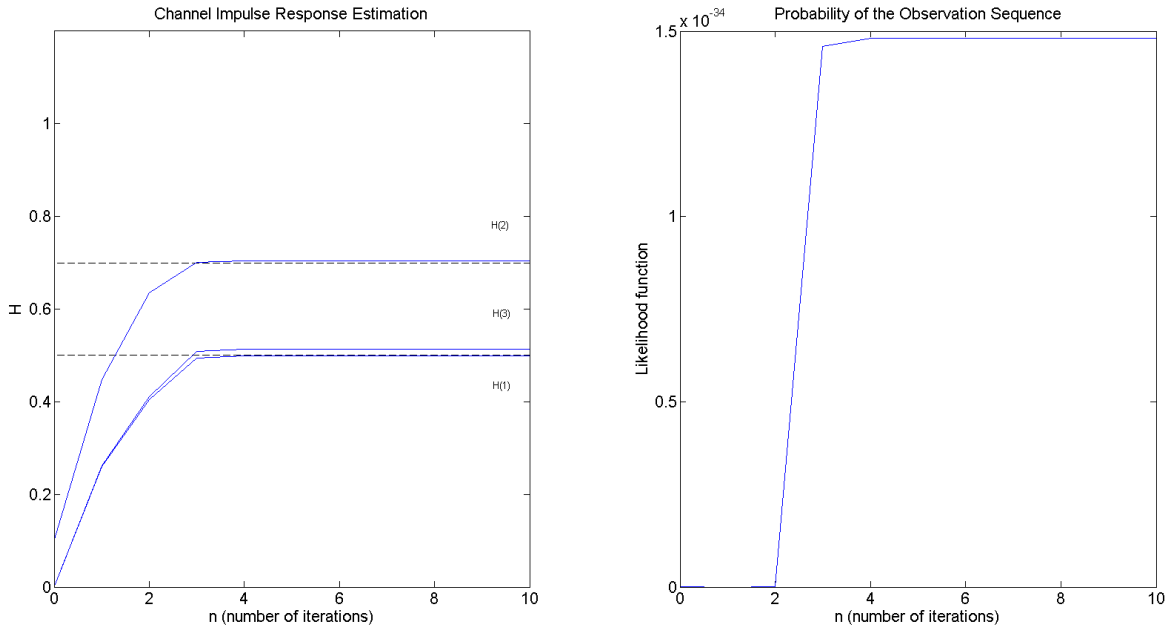


Figure 6: Evolution of channel parameters, linear channel, SNR = 8dB

From Figures 7-8, we can see that the accuracy of detection of information symbols is 100% even in 100 runs of the simulation when SNR = 8dB. This means that the algorithm can be very reliable if we put sufficient power into the system.

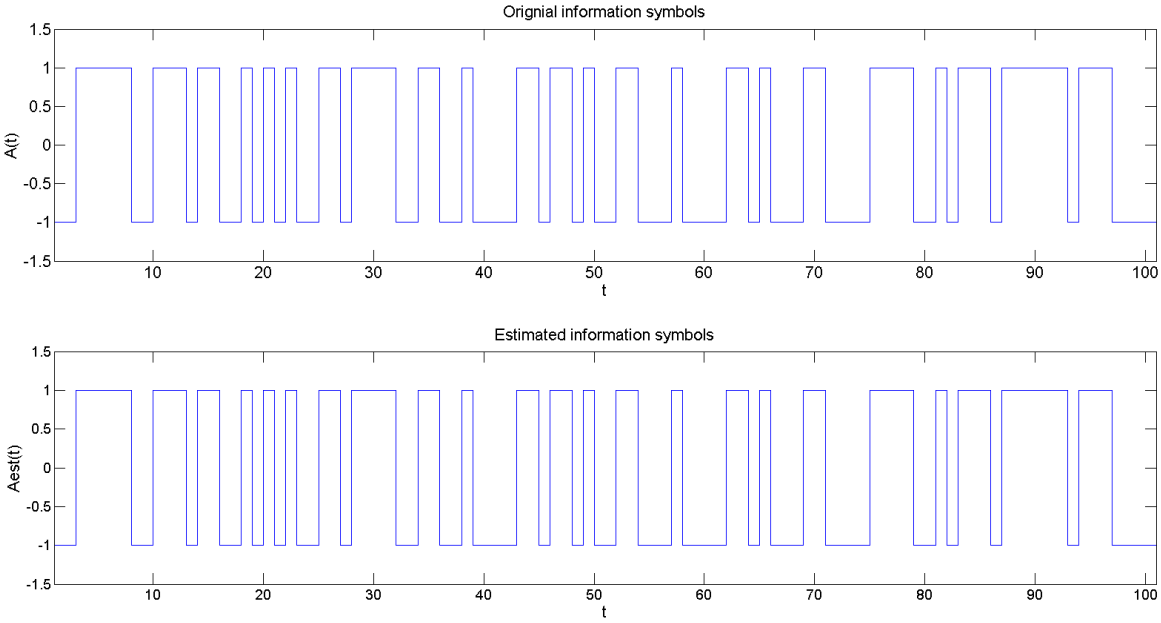


Figure 7: True information symbols vs Estimated information symbols, linear channel, SNR = 8dB

```
>> [accuracy, accuracy_mean, H_est] = run_program2_SNR_accuracy(T, H, SNR, H0, n)

accuracy =

  Columns 1 through 26

     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1

  Columns 27 through 52

     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1

  Columns 53 through 78

     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1

  Columns 79 through 100

     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1


accuracy_mean =

     1


H_est =

    0.4985
    0.7038
    0.5140
```

Figure 8: Average accuracy (100 runs), linear channel, SNR = 8dB

However, if we change our SNR from 8dB to 4dB, the performance becomes much worse, as shown in Figures 9-12. In Figure 10, we can observe that the convergence of parameters is incorrect this time. In Figure 11, even in a single run of the simulation, there are errors in the symbol detection. The average accuracy of the system is lowered to only about 90% in this case. This means that the system can be unreliable if we do not have sufficient power in the system to reach a high SNR.

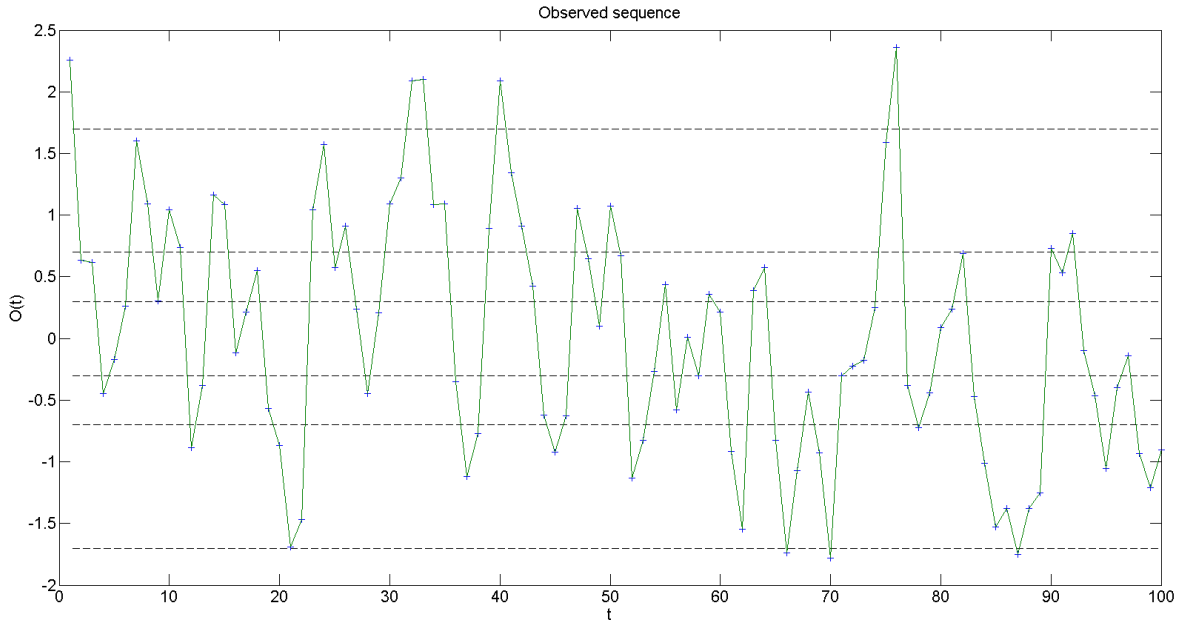


Figure 9: OBservation sequence, linear channel, SNR = 4dB

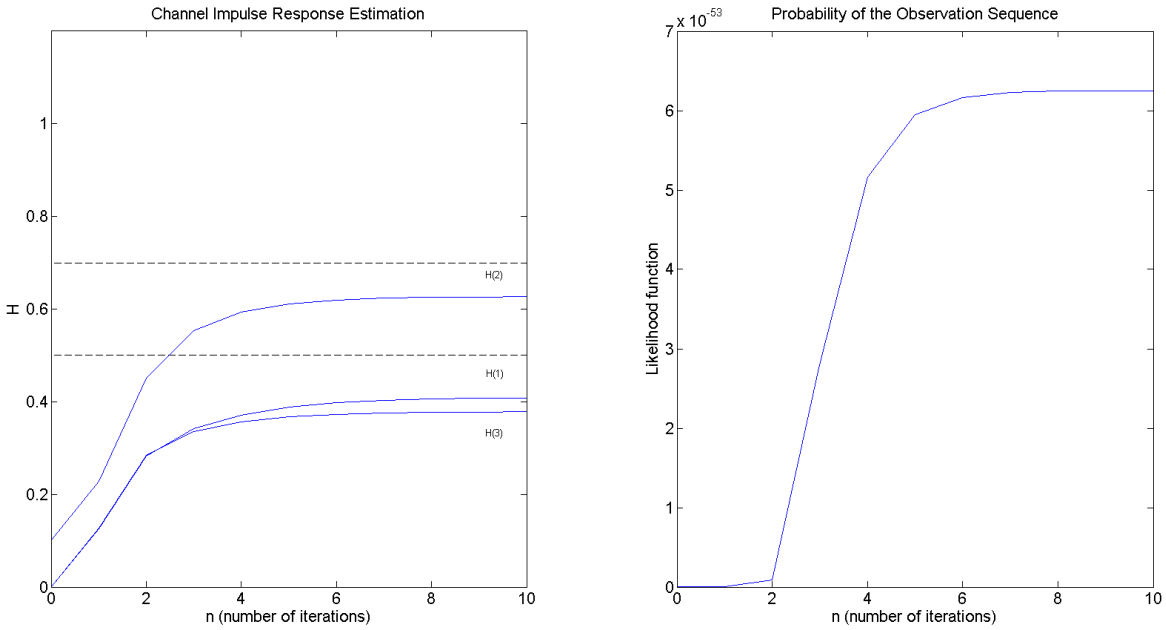


Figure 10: Evolution of channel parameters, linear channel, SNR = 4dB

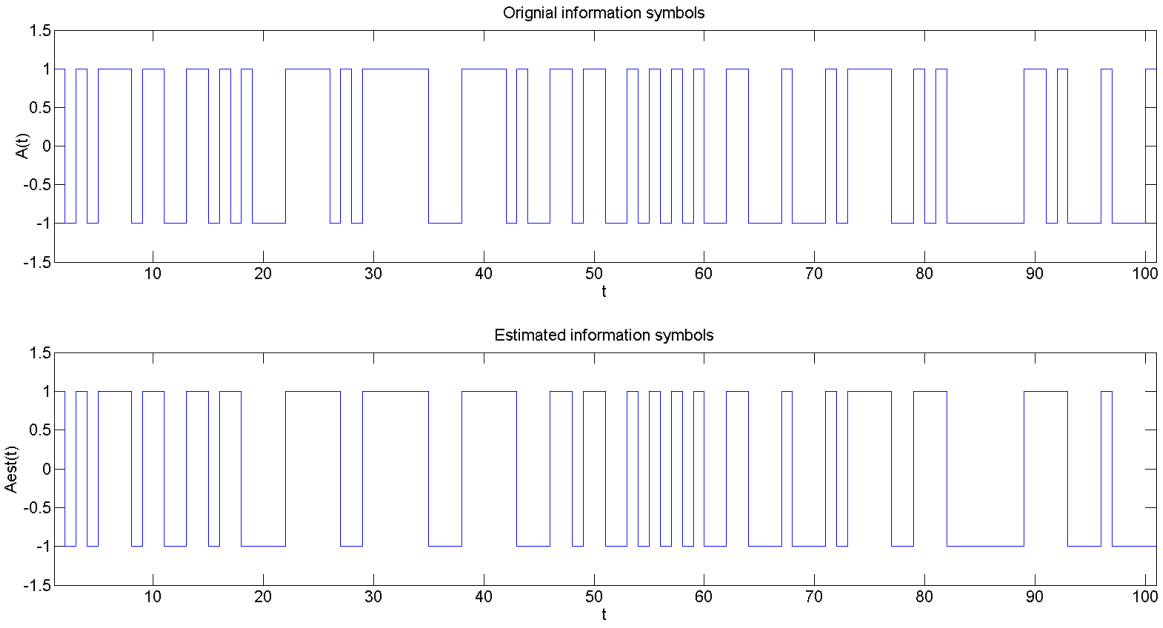


Figure 11: True information symbols vs Estimated information symbols, linear channel, SNR = 4dB

```
>> [accuracy, accuracy_mean, H_est] = run_program2_SNR_accuracy(T, H, SNR, H0, n)

accuracy =

  Columns 1 through 15

    0.9800    0.9300    0.9300    0.8900    0.9400    0.8600    0.9100    0.9100    0.9500    0.8600    0.8800    0.8500    0.8200    0.9600    0.9100

  Columns 16 through 30

    0.9700    0.9700    0.9400    0.9500    0.9000    0.9400    0.9100    0.9800    0.9300    0.8200    0.9600    0.9200    0.9500    0.8400    0.8800

  Columns 31 through 45

    0.9100    0.9100    0.8500    0.9400    0.8900    0.8500    0.9500    0.8800    0.9300    0.9400    0.8800    0.8500    0.9100    0.9600    0.9200

  Columns 46 through 60

    0.8700    0.9200    0.9100    0.9200    0.9300    0.9500    0.9500    0.8600    0.8400    0.8800    0.8800    0.8800    0.8900    0.9500    0.8200

  Columns 61 through 75

    0.9000    0.9700    0.8900    0.9300    0.9500    0.8800    0.8600    0.9300    0.9100    0.8900    0.9600    0.8800    0.9200    0.8900    0.9300

  Columns 76 through 90

    0.9400    0.8500    0.9100    0.8900    0.9100    0.8600    0.8200    0.8500    0.9500    0.9200    0.8900    1.0000    0.7800    0.8200    0.9500

  Columns 91 through 100

    0.7900    0.9500    0.8600    0.9400    0.8900    0.9200    0.7700    0.8800    0.8900    0.9100


accuracy_mean =

    0.9029


H_est =

    0.4084
    0.6268
    0.3782
```

Figure 12: Average accuracy (100 runs), linear channel, SNR = 4dB

Figure 13 shows the comparison of the performances between the algorithm and the one using training sequence. The red line represents the performance of our algorithm, while the blue line represents the one using training sequence. It can be observed that the performance of the algorithm in our project is about the same as the other one, just a little bit worse. However, as we mentioned in section 1.2, the algorithm using training sequence requires extra overhead to send training data, otherwise it is not suitable for time-varying channel. Therefore, our algorithm is better for time-varying channel.

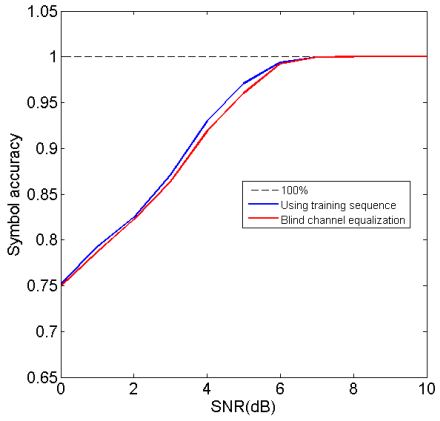


Figure 13: Average accuracy (100 runs) vs. SNR comparison, linear channel

We also performed a test with ASCII conversion. In the experiment, the sender transmits some texts to the receiver. The texts are converted to binary streams, and the binary streams are converted to information symbols. After the symbols are transmitted through the system, the inverse conversions are performed. As

shown in Figure 14 and Figure 15, surprisingly, we cannot recognize the original text even when we have a 93% accuracy of the binary streams. Therefore, the detection of the information symbols has to be very high in order to make successful communications.

```
>> accuracy = run_program4_ASCII_COMM(H, H0, SNR, n)
What is the original text? Welcome to the final presentation of ECE 493.
Welcome to the final presentation of ECE 493.

accuracy =

     1
```

Figure 14: ASCII test, linear channel, SNR = 4dB, 100% accuracy

```
>> accuracy = run_program4_ASCII_COMM(H, H0, SNR, n)
What is the original text? Welcome to the final presentation of ECE 493.
gelc me pw thE ganal pqgcen|at1on of ECF 48s.

accuracy =

    0.9333
```

Figure 15: ASCII test, linear channel, SNR = 4dB, 93% accuracy

3.2 Experimentation for Non-linear Channel

In our test for the non-linear channel, the block length is T = 100, and the channel output is composed of eight coefficients: -1.7, -0.7, -0.3, 0.7, -0.7, 0.3, 0.7, and 1.7, as proposed in [2]. We found that the results are similar to those in the linear channel case. For simplicity, we only show the evolution of channel parameters under different initializations. The SNR is 8dB, and we tested two different initialization of channel parameters of [-1, -1, 1, 1, -1, -1, 1, 1] and [-1, -1, -1, -1, 1, 1, 1, 1]. We also show the comparison of performance using the two different algorithms.

As shown in Figure 16, the convergence occurs quickly, when we use the first parameters initialization, similar to the results in Figure 6. However, as illustrated in Figure 17, once we use the other initialization choice, the performance becomes much worse even at the same SNR. Therefore, the initialization is also a crucial factor when we use this algorithm.

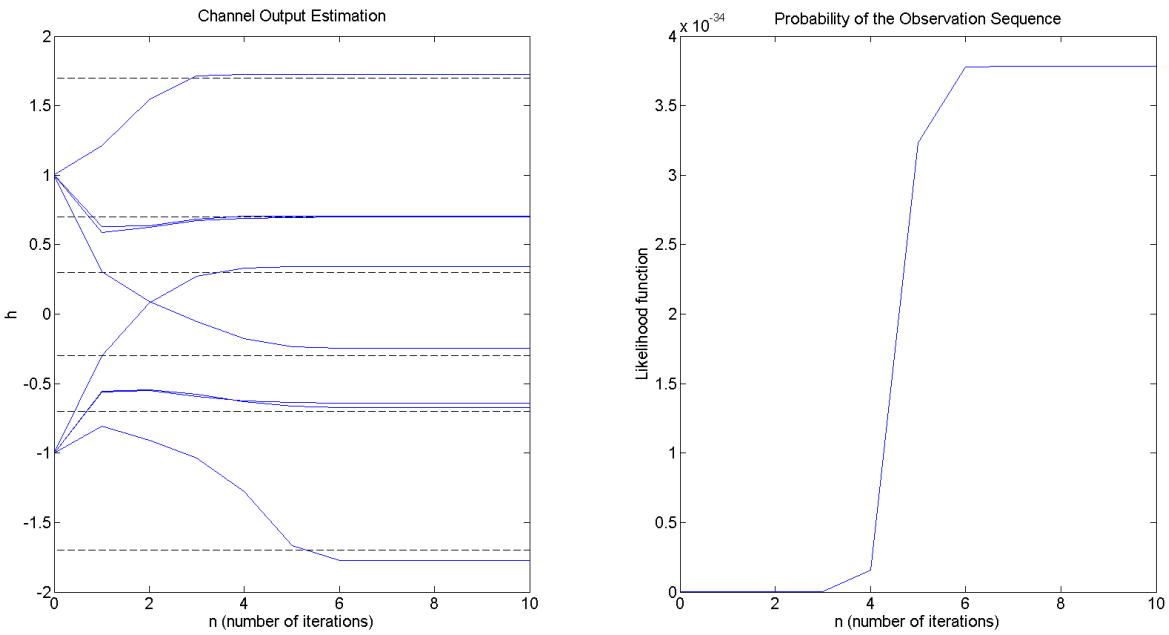


Figure 16: Evolution of channel parameters, non-linear channel ,SNR = 8dB, initialization 1

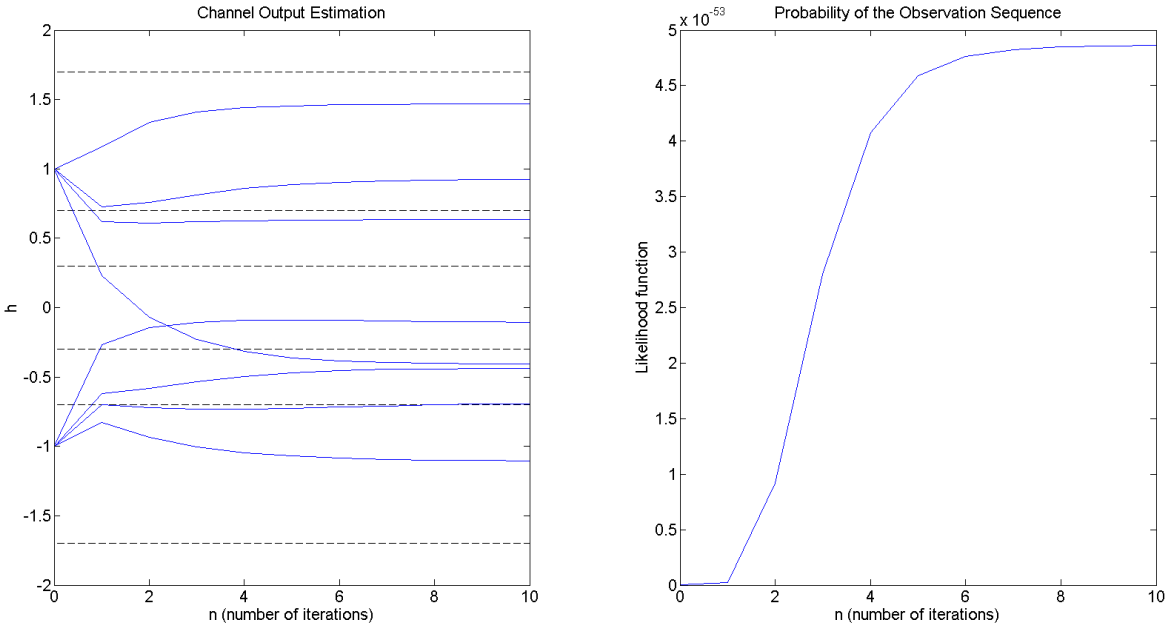


Figure 17: Evolution of channel parameters, non-linear channel ,SNR = 8dB, initialization 2

Figure 18 shows the comparison of performance under different SNRs. The red line represents the algorithm in our project, while the blue line represents the one using training sequence. Similar to Figure 13, the performance of our algorithm is a little bit worse. For this test, we use the first initialization choice as discussed above.

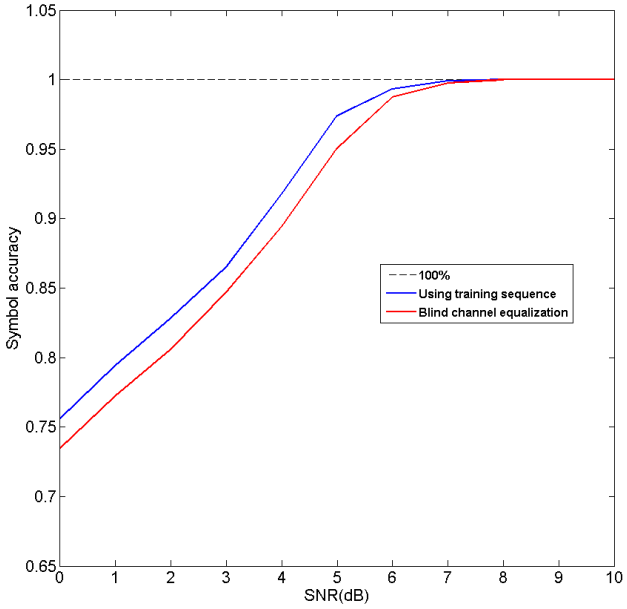


Figure 18: Average accuracy vs. SNR comparison, non-linear channel

4 Experimentation Validation Using Evaluation Criteria

In these experiments, we have met the following requirements from our proposal in the section 2 of Appendix A in this paper:

- The approach is tested in Matlab for both linear and non-linear channels, at different noise levels and different filter-length (for simplicity we only show the results of filter-length equals 3).
- For a given channel, a random vector with length of 100 samples (T = 100) is generated with binary information symbols.
- The simulation results has showed the fast convergence of the iteratively estimated channel parameters.
- The average accuracy of estimation in all tests are calculated.

All the processed experimental data are demonstrated in Figures 5-18. In conclusion, to make the system reliable, sufficient SNR and proper choice of initialization are required in order to get very high accuracy to assure effective communications.

5 Other Issues

The reason for this project is that we intend to investigate a statistical model that could be used for solving communication problems. To be more specific, we want to know how HMM can be applied to FIR blind channel equalization. Everyone would benefit from this project, as long as they use any type of wireless communication systems. If the algorithm can tolerate different choices of the initialization of parameters, it can be used to effectively save the bandwidth used for training sequence. In [4], a particular method regarding segmentation to properly choose the initialization of the channel parameters is discussed. If the block length is small enough, the system could be potentially be used for real-time communication, since the number of iterations are relatively small, as illustrated in Figure 6 and Figure 16.

In addition, the project could lead to one important component of a marketable product, which is the software-defined radio. In a software-defined radio (SDR), many traditional hardware components used for communication systems can be replaced by software on PCs or embedded systems. Any companies or anyone who are interested in the future radio systems might be the users of this product. The product could be applicable to the global range of users.

For our project, we only consider the operation time of the program and the hours we spent on the designs and implementations. The average time cost of the programming on blind channel equalization and also the algorithm using training sequence are shown in Figure 19 (assume that the block length T = 100). The hours we spent on the designs and implementations are shown in Figure 20.

	Number of iterations	Our algorithm	Algorithm with training sequence
Linear Channel	n = 10	148.4ms	10.4ms
	n = 5	77.8ms	10.4ms
Non-linear channel	n = 10	113.2ms	8.7ms
	n = 5	60.9ms	8.7ms

Figure 19: Average time of implementation on each algorithm

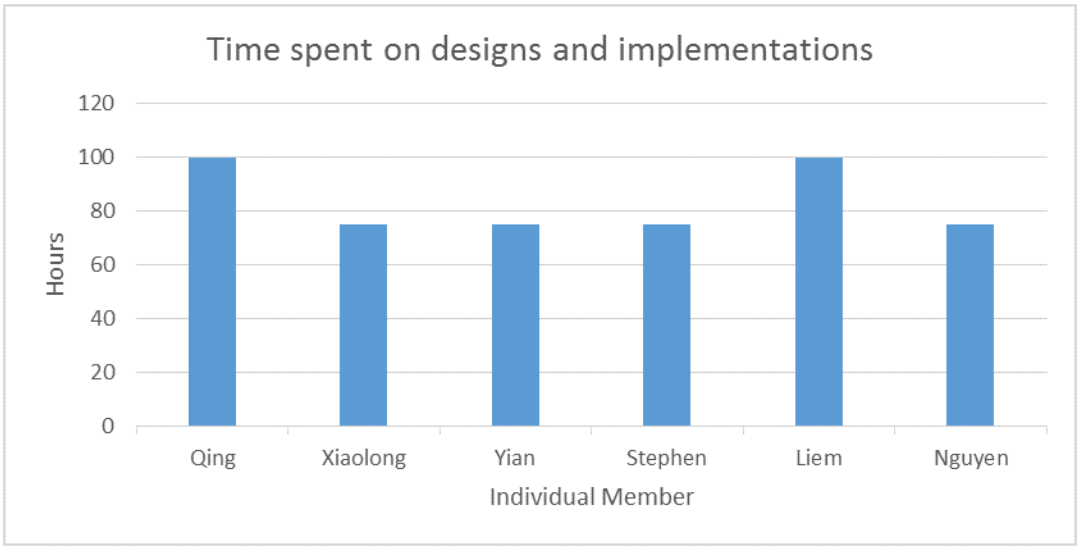


Figure 20: Time spent on designs and implementations

If the channel does not change abruptly, a hybrid system can be used to optimize our design. Basically, the hybrid system can be used in the following way:

1. Use training sequence to estimate the channel parameters, which can be treated as the initialized parameters of the Baum-Welch algorithm.

2. Set the timer, and use our Baum-Welch estimator to estimate the channel parameters iteratively with the initialization of parameters.

3. If a time-out occurs, go back to step 1, otherwise input the converged parameters from step 2 as the next initialized parameters.

This hybrid design could potentially solve the initialization problem, although we have not fully tested it.

Since the algorithm is implemented in software, it should not have any big negative impacts to the environment. It can be maintained as long as we have the supported software. The algorithm itself may be upgraded to a stronger one in future research.

6 Administrative Part

6.1 Project Progress

The overall project progress is shown in Figure 21. We first implemented the test for linear channel with memory length = 1 using training sequence, which is the simplest case in the project. Then we gradually extended the simulation to the case for non-linear channel, with memory length = 2, and without the use of training sequence. We kept analyzing and correcting the problems we met while we completed each task. As it can be seen from Figure 21, we have successfully completed all of the tasks, although we changed some of our schedules, which are discussed in section 6.2.

6.2 Changes

There are several changes in our project, compared with our initial design. The first change is that we used SNR instead of $\frac{E_b}{N_0}$ to test the algorithms. The reason is that $\frac{E_b}{N_0}$ is not well defined in [2], and we do not know which modulation scheme used in the paper. Therefore, we choose SNR, which has been defined in section 2 to evaluate the algorithms. The second change is that we simplified our level-1 and level-2 designs because they were actually not that complicated as we thought previously. The new designs are clearer than the old ones.

We also had to carry out some extra activities. Originally, we did not plan to perform the ASCII test. However, in order to demonstrate our project more clearly and easily, we thought we should simulated a real world communication between two users. Text messaging is the easiest one that we can demonstrate. Therefore, we did some extra coding in our project to perform the ASCII simulation.

6.3 Funds Spent and Man-hours

Since we simulated our project in Matlab, which is free for students and provided in computer labs by George Mason University, we did not spend any funds on our project. We spent about 900 hours on our project in total. Each member spent 5-7 hours per week on the project. Figure 22 and Figure 23 show the approximate time spent by each team member on the project.

Task	Completion rate	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1. Known-data estimation (memory length = 1, linear channel)															
1.1 Generate training data	100%														
1.2 Detect the parameters using training data	100%														
1.3 Generate input and information	100%														
1.4 Calculate the weighted likelihood	100%														
1.5 Detect the input using MAP decision rule	100%														
1.6 Evaluate the estimation	100%														
2. Known-data estimation (memory length = 1, non-linear channel)															
2.1 Repeat 1.1-1.6 properly with related change	100%														
3. Known-data estimation (memory length = 2, linear channel)															
3.1 Modify the I/O generator with related change	100%														
3.2 Update the model of each function with related change	100%														
3.3 Parameters and inputs estimation	100%														
4. Known-data estimation (memory length = 2, non-linear channel)															
4.1 Combine step 2 and 3 properly with related change	100%														
5. Unknown-data estimation (memory length = 1, non-linear channel)															
5.1 Repeat the Matlab design properly with related change	100%														
6. Unknown-data estimation (memory length = 2, linear channel)															
6.1 Modify the I/O generator with related change	100%														
6.2 Update the model of each function with related change	100%														
6.3 Repeat the design in the prototyping properly with related change	100%														
7. Unknown-data estimation (memory length = 2, non-linear channel)															
7.1 Combine step 5 and 6 properly	100%														
8. Problem analysis and correction															
8.1 Collect all problems	100%														
8.2 Problem analysis and resources reading	100%														
9. Generic design(optional)															
9.1 Generic design for all function units															
10. Testing and evaluation															
10.1 Case 1: SNR = 4dB	100%														
10.2 Case 2: SNR = 8dB	100%														
11. Report presentation and document															
11.1 In-progress and final report															
11.2 In-progress to FS and final oral presentation															
11.3 Final report and poster															

Figure 21: Project progress table

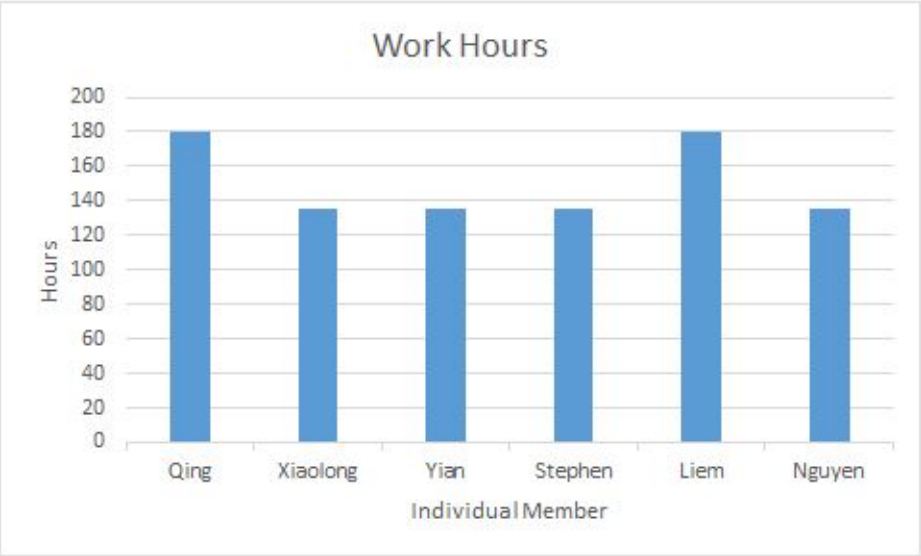


Figure 22: Man-hours column chart

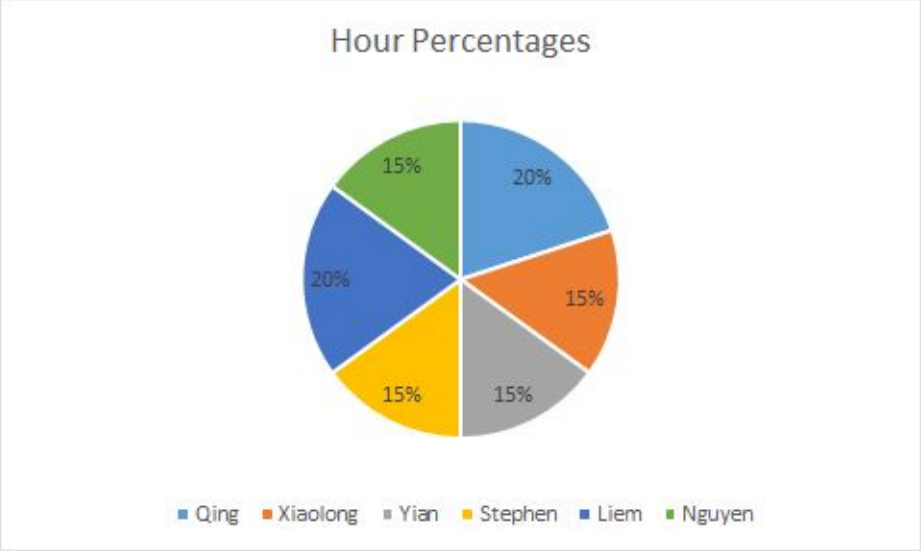


Figure 23: Time percentages pie chart

7 Lesson Learned

From this project, we learned much additional knowledge in programming. Although the algorithm is relatively straightforward, to successfully generate the simulation is another story. We encountered many programming and machine problems. First of all, we had to consider the data structure of the information symbols and states in Matlab. Other issues we encountered include indexing problem, overflow and underflow, and scaling.

To calculate the forward, backward, and weighted likelihood function, we need to know the Gaussian probability density function (pdf) given the observation and output of channel at each time instant t. If the state transition is given by index number for each state, we need to index the corresponding channel output properly. In addition, some transition branches are not applicable for the systems. We had to find a systematic way to properly index each state transition.

When the initialization of the channel parameters are far away from the real ones, the Gaussian pdf is likely to be very small, even smaller than the double precision range of a classical computer machine. To solve this problem, we need to take the logarithm of each term in calculation of the forward, backward, and weighted likelihood function, and set the lower bound and the upper bound for underflow and overflow, respectively.

When the time duration is large, such as T = 1500, the forward and backward function would be very small due to large number of iterations. We need to do the scaling for forward and backward function to limit their ranges according to [4].

By doing this project, we learned and solved these problems successfully, which highly improved our programming skills. We also learned that HMM is a useful stochastic model that can be applied to many areas, such as speech recognition, cryptanalysis, and gene prediction. In recent cognitive radio research, it has been extended to a Hidden Bivariate Markov Model (HBMM), which can be used for temporal spectrum sensing.

We also learned that team work is very important. Each team member must contribute in the areas that they are good at. Since the theory behind this project is somewhat difficult, it is unrealistic to have everyone understand the model.

8 References

[1] C.R. Johnson *et al.*, *Software Receiver Design.* Cambridge: Cambridge University Press, 2011.

[2] G. K. Kaleh and R. Vallet, Joint parameter estimation and symbol detection for linear or nonlinear unknown channels, IEEE Trans. Commun., vol. 42, pp. 2406-2413, July 1994.

[3] Y. Ephraim and N. Merhav. Hidden Markov processes. IEEE Transactions on Information Theory, 48:6 (2002), 1518-1569.

[4] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE, vol. 77, pp.257-286, Feb. 1989.

[5] D. P. Bertsekas and J. N. Tsitsiklis. Markov Chains, in Introduction to Probability, 2nd ed. Nashua, NH: Athena Scientific, 2008, pp. 339-405.

[6] Y. Ephraim. Digital Communication over AWGN, ECE 460 Lecture Notes: Communication and Information Theory. Fairfax, VA: George Mason University, 2015, pp. 81-107.

9 Appendix A: Proposal

ECE-492
Proposal

FIR Channel Blind Equalization

Date of Submission: 03/16/2015
Faculty Supervisor: Dr. Yariv Ephraim

Team Members: Qing Chen
Xiaolong Fang
Yian Hu
Stephen Owusu
Liem Tran
Nguyen Tran

1 Executive Summary

The purpose of this project is to study and design a receiver to decode signals, utilizing Markov properties, over an unknown channel. The receiver model will initially be simulated in Matlab with a simple communication system, with known parameters, in order to check for the performance and reliability of the model. A more complicated communication system, with unknown parameters, will then be implemented. The result of this system will then be compare to a communication system with known training data at the transmitter and receiver, in order to determine the receiver model's efficiency between when the parameters are known and when they are not.

2 Problem Statement

Symbol detection is a typical problem in signal communication. The goal is to recover the information symbols after they are passed through an unknown channel in the presence of additive noise. The common approach is to identify the channel first using known symbols, and then determine the unknown information symbols using the identified channel. The problem of this approach is that it limits the system's bit-rate. This project proposes an alternative approach that can improve the system throughput, by reducing the transmission of known symbols used in channel identification. Communication performance will also be enhanced by minimizing the symbol error probability. [1]

The objective of this project is to design and simulate a more efficient receiver model, utilizing its Markov properties. Our motivation here is to apply our knowledge in signal processing to design a receiver model that can reduce the probability of error, for finite blocks of symbols, when the cost of getting signals from the actual source is high.

3 Approach

3.1 Problem Analysis

Consider an input signal passing through a memoryless FIR channel in the presence of Gaussian white noise. The problem is how to design an optimal receiver that can perform blind equalization to determine the input of the system [2]. The problem is formulated as a Hidden Markov Process (HMP). The approach to this problem requires two steps: 1) estimate the channel parameters and noise variation using the Baum-Welch algorithm, 2) and information symbols detection using estimated parameters and maximum a-posteriori (MAP) decision rule.

3.2 Requirement Specification

The following are requirements for the expected simulation results under certain setting for parameters of the system:

- The approach should be tested in Matlab for both linear and nonlinear channels, different noise levels and different filter-length.
- For a given channel, a random vector with at least length of 100 samples (T = 100) should be generated with binary information symbols
- The simulation results should show the convergence of the iteratively estimated channel parameters.
- The average accuracy of estimation in all tests should be calculated.

3.3 Markov Chain and Hidden Markov Model

In a first order Markov chain model, the future states are independent of the past states given the current state. Here we only consider the discrete-time Markov chains in which the states change at certain discrete instances, indexed by an integer variable n. At time t, the state of the chain is denoted by S_t, and it belongs to a

state space S. Markov chain is described by its transition probabilities $\rho_{ij} = P(S_{t+1} = j|S_t = i),\ i, j \in S$. [4]

Also from [4], we have the following specifications of a Markov model:

1. The set of states $S = \{1, ..., m\}$.
2. The set of possible transitions for pairs of i and j. The transition probabilities, ρ_{ij}, cannot be negative.
3. Markov property:

$$P(S_{t+1}|S_t = i, S_{t-1} = i_{t-1}, ..., S_0 = i_0) = P(S_{t+1} = j|S_t = i) \tag{1}$$

   for all times t, all states S, and all possible sequences $i_{t-1}, ..., i$ of earlier states.

A hidden Markov model(HMM) is a statistical Markov model with hidden states. From [3], it has the following parameters:

1. State transition probabilities, $A = \{a_{ij}\}$, where $a_{ij} = P(q_{t+1} = S_j|q_t = S_i)$.
2. Observation probability distribution, $B = \{b_i(v)\}$, where $b_i(v) = P(O_t = v|q_t = S_i)$.
3. Initial state distribution, $\pi = \{\pi_i\}$, where $\pi_i = P(q_1 = S_i)$.
4. Model, λ, where $\lambda = (A, B, \pi)$.

There are three fundamental problems in a HMM. They are evaluation, decoding, and learning problems. In the evaluation problem, we are to find $P(O|\lambda)$, given O and λ. In the decoding problem, we are to find the optimal state sequence Q that maximizes $P(O|\lambda)$, given O and λ. Lastly, for the learning problem, we are to find the model, λ, that maximizes $P(O|\lambda)$, given O.
The evaluation problem be be used for isolated recognition, such as word. The decoding problem can be used for continuous recognition as well as segmentation. The learning problem must be solved if we want to train an HMM for subsequent use of recognition tasks.

3.4 Baum-Welch Algorithm

The Baum-Welch algorithm is an efficient algorithm that can calculate the parameter and the probability of observation iteratively using the forward-backward algorithm.

From [1] and [3], we can summarize the contribution at a time t of the past and of the future by the following definitions:

$$\gamma_t(i) \triangleq \rho(y_1, y_2, ..., y_t, S_t = i) \tag{2}$$

$$\beta_{t+1}(j) \triangleq \rho(y_{t+1}, y_{t+2}, ..., y_T|S_{t+1} = j) \tag{3}$$

with $\gamma_1(i) = P[S_1 = i]$ and $\beta_T(j) = 1$ for all i and j, we can deduce the following forward and backward recursion as shown by [1]:

$$\gamma_t(j) = \sum_{i \in S} \gamma_{t-1}(i) P[S_t = j|S_{t-1} = i] p(y_{t-1}|S_t = j, S_{t-1} = i) \tag{4}$$

$$\beta_t(i) = \sum_{j \in S} \beta_{t+1}(j) P[S_{t+1} = j|S_t = i] p(y_t|S_{t+1} = j, = S_t = i) \tag{5}$$

3.5 MAP Decision Rule

Suppose X is a discrete random variable where each realization represents the transmitted symbol of the communication system. X takes M value denoted by $\{1, ..., M\}$. The value of X are transmitted over a channel whose output is Y. The MAP decision rule can be described as follows:

For a receive value y of Y, the optimal receiver calculates

$$L_m(y) := p_x(m) f_{Y|X}(y|m) \tag{6}$$

for every value of $m \in \{1, ...M\}$ and choose the value of m that maximizes $L_m(y)$ as the estimate of X:

$$\widehat{X} = argmax_{m \in \{1,...M\}} \{\rho_x(m) f_{Y|X}(y|m)\} \tag{7}$$

This decision rule will be used for information symbol detection. [5]

4 Preliminary Design

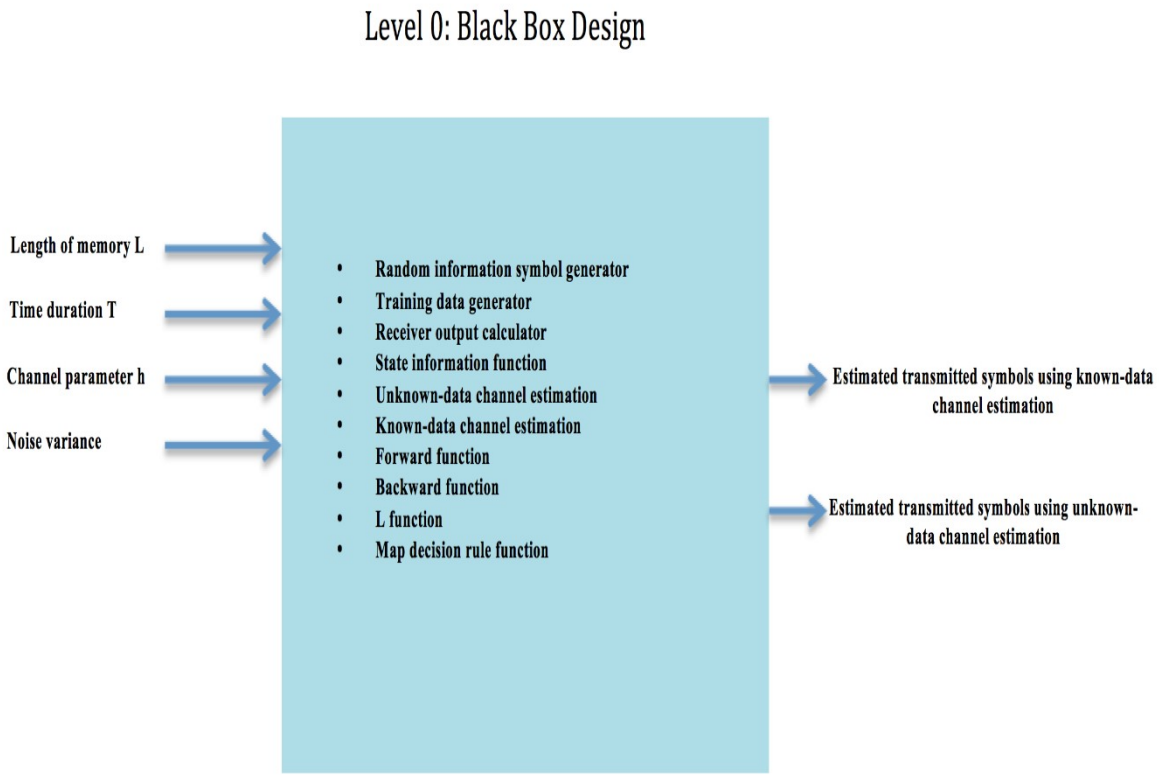


Figure 1: Level-0 Design

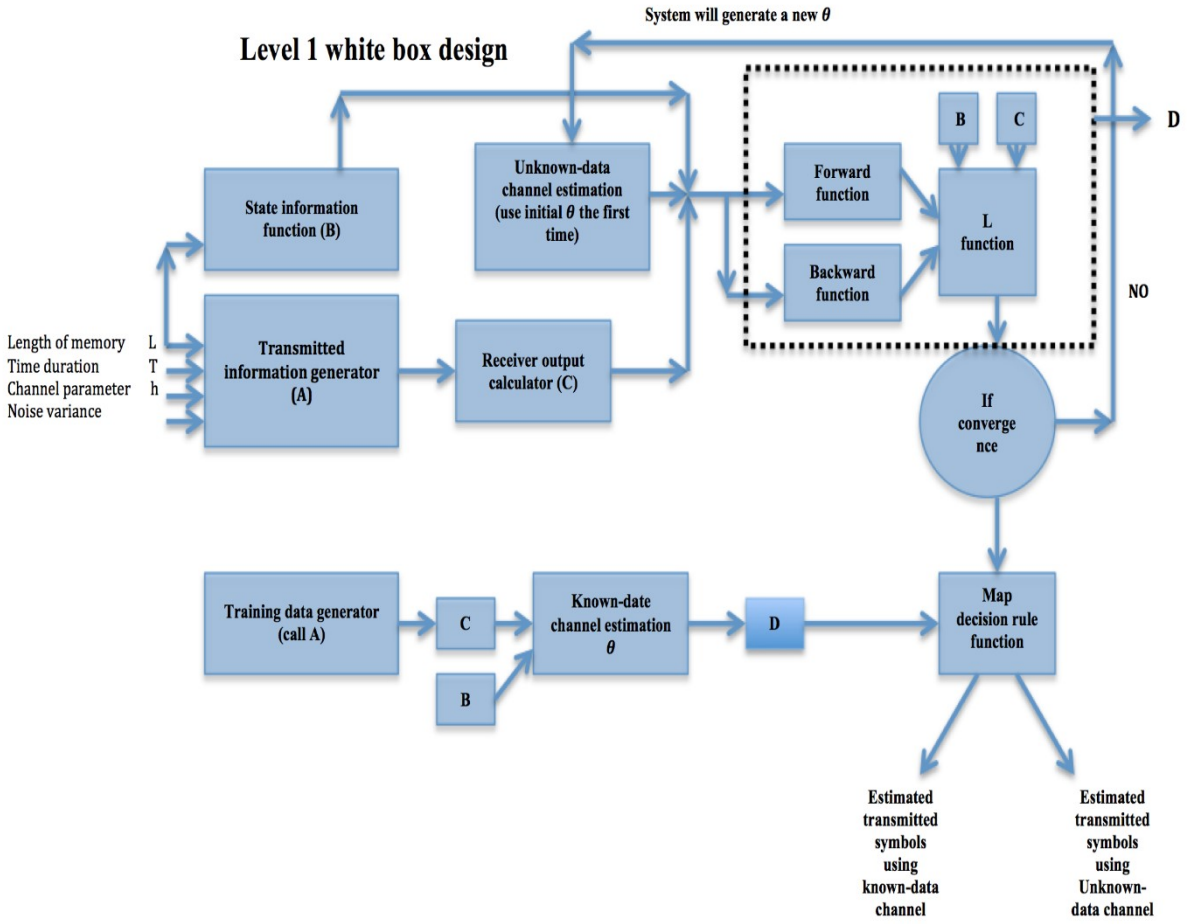


Figure 2: Level-1 Design

5 Preliminary Experimentation Plan

1. Evaluate the receiver using known symbols for transmission first.
2. Evaluate the receiver using a simple simulation with binary input and small length of vectors.
3. Compare performance of the information symbol detection with the known symbol channel detection based case.

6 Preliminary List of Tasks and Allocation of Responsibilities

- Liem Tran is the Project Manager.
- The comprehension of the key algorithms is assigned to Qing Chen and Liem Tran.
- The investigation of the background and supporting knowledge or material of the project is assigned to Stephen Owusu and Nguyen Tran.
- The realization of the algorithm for known symbols channel detection is assigned to Yian Hu and Xiaolong Fang.
- The realization of the iterative algorithm for joint parameters estimation on Matlab is assigned to Qing Chen, Yian Hu and Xiaolong Fang.
- The design of sequence detector in Matlab is assigned to Qing Chen, Stephen Owusu and Nguyen Tran.
- The final test and evaluation for the simulation model is assigned to Qing Chen and Liem Tran.

References

[1] G. K. Kaleh and R. Vallet, Joint parameter estimation and symbol detection for linear or nonlinear unknown channels, IEEE Trans. Commun., vol. 42, pp. 2406-2413, July 1994.

[2] Y. Ephraim and N. Merhav. Hidden Markov processes. IEEE Transactions on Information Theory, 48:6 (2002), 1518-1569.

[3] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE, vol. 77, pp.257-286, Feb. 1989.

[4] D. P. Bertsekas and J. N. Tsitsiklis. Markov Chains, in Introduction to Probability, 2nd ed. Nashua, NH: Athena Scientific, 2008, pp. 339-405..

[5] Y. Ephraim. Digital Communication over AWGN, ECE 460 Lecture Notes: Communication and Information Theory. Fairfax, VA: George Mason University, 2015, pp. 81-107.

10 Appendix B: Design Document

ECE-492
DESIGN DOCUMENT

FIR Channel Blind Equalization

Abstract

The main goal of this project is to study and test a Hidden Markov Model(HMM) method that is used for information symbols detection in an unknown FIR channel. The method uses the Baum-Welch algorithm for channel detection. It also uses the the maximum a-posteriori(MAP) decision rule for symbol detection. We first introduce the analysis and specifications of the problem. Second, we demonstrate the architecture of the system such as the level-0, level-1, and level-2 designs based on Matlab simulation. Then, we introduce the background, methodology, and detailed design with data flow diagram. We also show our prototyping activities for unknown data channel estimation on memoryless and linear case. The codes, included in the appendix, were implemented in Matlab. We briefly mention our attempt for unknown data channel estimation on linear channel with memory. We also provided details about the problems we encountered. Our simulation results show the that method works for simple cases even though not all the problems were solved entirely. Lastly, we propose a schedule for future development of the project.

Date of Submission: 04/29/2015
Faculty Supervisor: Dr. Yariv Ephraim

Team Members: Qing Chen
Xiaolong Fang
Yian Hu
Stephen Owusu
Liem Tran
Nguyen Tran

1 Introduction and Problem Statement

Symbol detection is a typical problem in signal communication. The goal is to recover the information symbols after they are passed through an unknown channel in the presence of additive noise. The common approach is to identify the channel first, using known symbols, and then determine the unknown information symbols using the identified channel. The problem of this approach is that it limits the system's bit-rate. This project proposes an alternative approach that can improve the system throughput, by reducing the transmission of known symbols used in channel identification. Communication performance will also be enhanced by minimizing the symbol error probability as shown in [1].

Our motivation for this project is to apply our knowledge in signal processing and random process to design a receiver model that can reduce the probability of error for finite blocks of symbols when the code of getting signals from the actual source is high.

To approach this project, we begin with a problem analysis. Consider an input signal passing through an unknown FIR channel in the presence of Gaussian white noise, [2]. The problem is how to design an optimal receiver that can perform blind equalization to determine the input of he system. The problem is formulated as a hidden Markov process (HMP). The approach to this problem requires two steps: 1) estimate the channel parameters and noise variation using the Baum-Welch algorithm, and 2) information symbols detection using estimated parameters and MAP decision rule.

As shown in [1], the receiver proposed here has the following characteristics: it performs blind channel identification, no symbol is lost, and it delivers optimal symbol-by-symbol decisions based on MAP decision rule and channel parameter estimation.

The purpose of this project is to study, design, and test a receiver to decode signals, utilizing Markov properties, over an unknown channel.

The receiver model will be simulated in Matlab starting with a simple communication system, with known parameters, in order to check the performance and reliability of the model.

A more complicated communication system, with unknown parameters, will then be implemented. The result of this system will then be compare to a communication system with known training data at the transmitter and receiver, in order to determine the receiver model's efficiency between when the parameters are known and when they are not.

2 Requirement Specifications

- The approach should be tested in Matlab for both linear and non-linear channels, different noise levels and different filter-length.
- For a given channel, a random vector with at least length of 100 samples($T = 100$) should be generated with binary information symbols.
- The simulation results should show the convergence for the iteratively estimated channel parameters.
- The average accuracy of estimation in all tests should be calculated.

3 System Design and Architecture

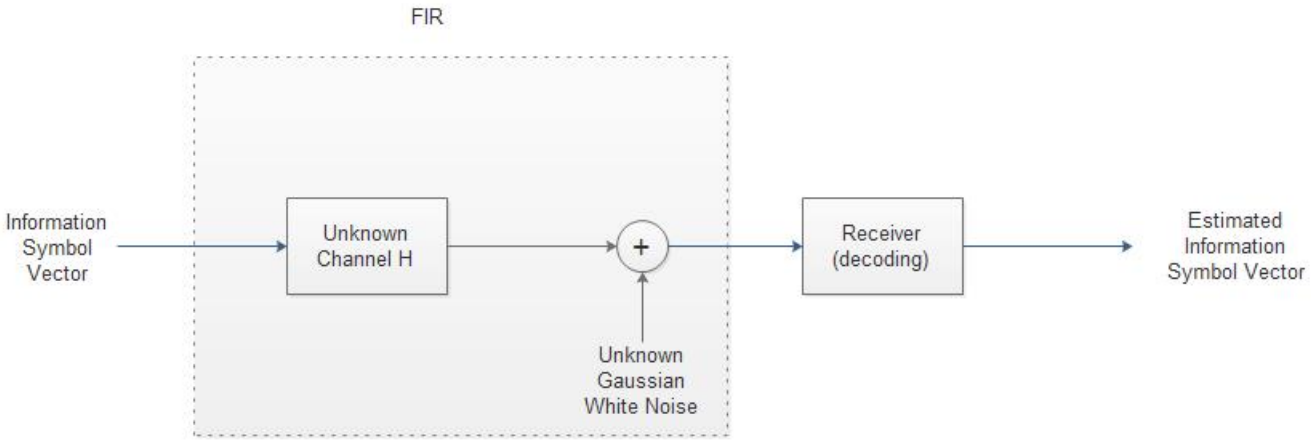


Figure 1: System architecture

The system architecture of this project is straightforward, as shown in figure 1. We have a sequence of the information symbols which are transmitted at the sender side. If we look at each input element of the sequence at the receiver side, they have a predetermined sample space for symbol detection. For example, the binary case where the inputs can only be 0 or 1. The unknown channel in the system can be either memoryless or with memory, and linear or non-linear. The input has additive Gaussian white noise as it passes through the unknown channel. The output of the FIR channel, which is a sequence that has the same length as the input sequence, will be observed by the receiver. The receiver will then determine the parameters in the FIR channel and estimates the input information symbols.

The system and algorithms in our project will be simulated in Matlab. Figure 1 shows the level-0 design of the Matlab simulation. There are four inputs and two outputs for the system design. The four inputs are: 1) length of memory L, 2) time duration/length of the sequence T, 3)channel parameter h, and 4) noise variance σ^2. And the two outputs are: 1) estimated input based on training data channel estimation, and 2) estimated input based on unknown channel estimation.

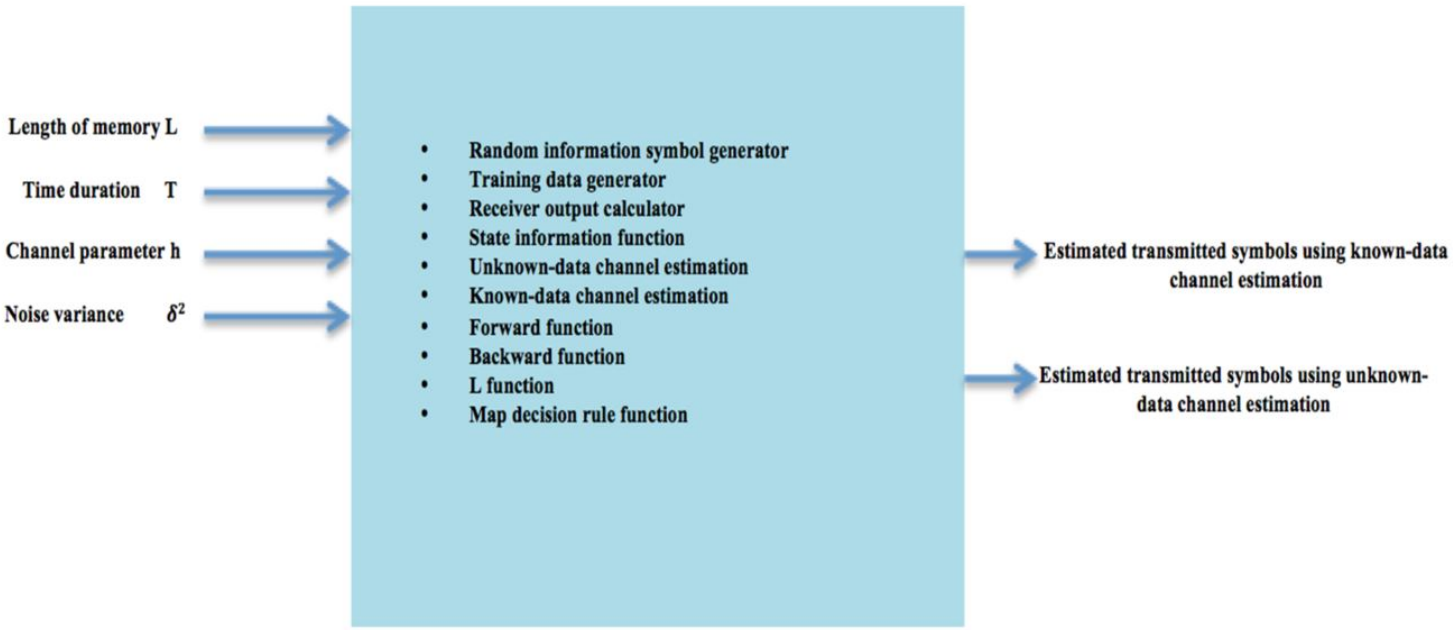


Figure 2: Level-0(Black box)

In level-1 design we build the function blocks we illustrate in the black box of level-0, as shown in figure 3. We have the random information symbol generator, state information function, and observed sequence(received output) calculator, which are used for testing the algorithm for the simulation. For unknown data channel estimation we use the forward, backward, and likelihood function units to calculate the variables we need for both re-estimation and receiver decision on the input. The re-estimation unit will re-estimate the parameters iteratively until the convergences of those parameters occur. The MAP decision rule unit will estimate the input using estimated parameters, For the known data channel estimation, most of the functions units are the same as the other estimation method except that we use the symbol generator to generate the training data first, and there is only a one-time estimation function unit rather than the re-estimation unit.

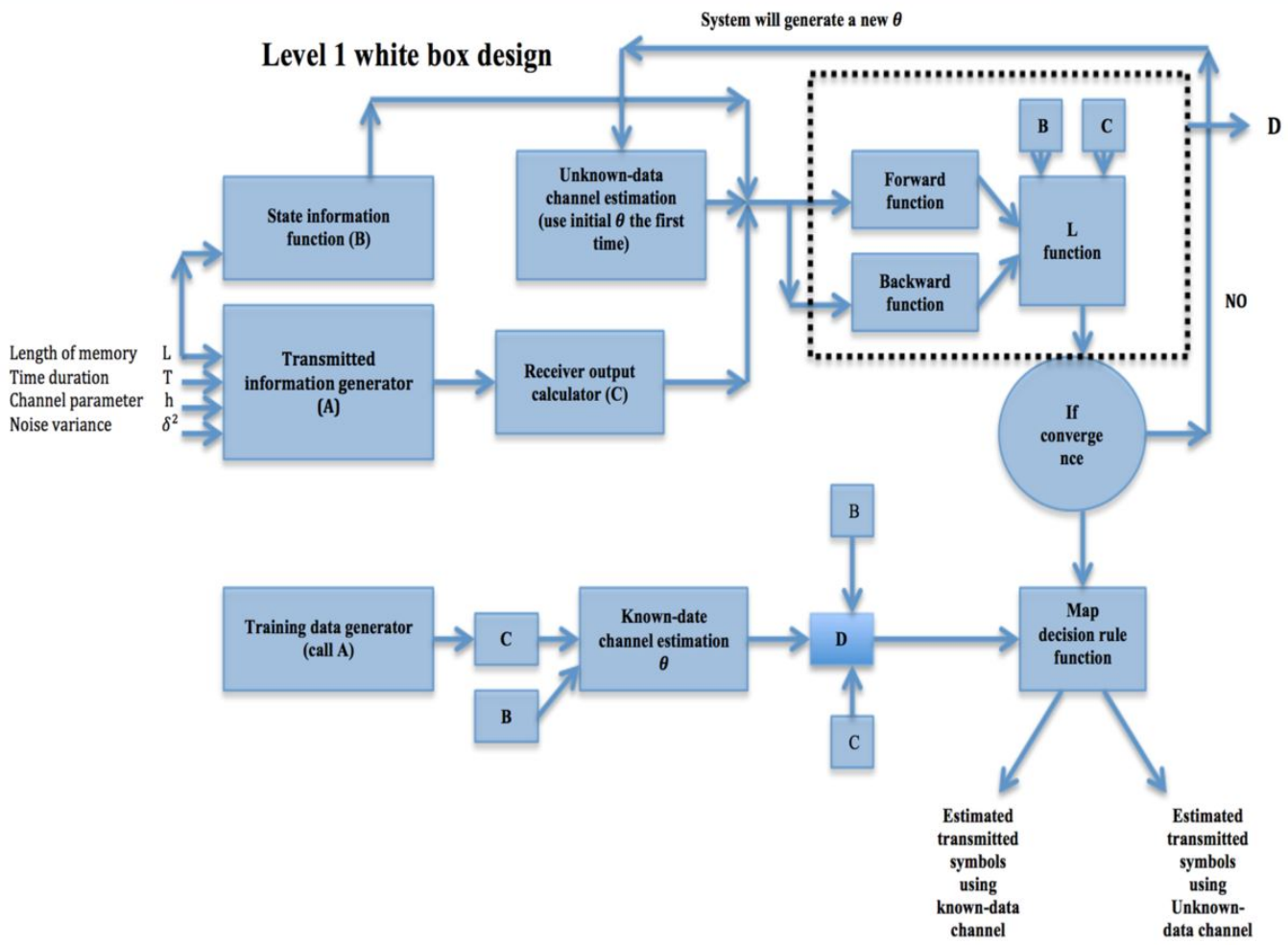


Figure 3: Level-1(White box)

For the level-2 design, in order to explain it in detail, we split it into four parts, as shown in figures 4-7. For part 1, three function blocks are used for generation the tested input, which can possibly be combined into one function unit when doing the Matlab design. The information symbols, state information, and the transition branches, can be generated by (2)-(4), in section 4. For part 2, the re-estimation process for unknown data channel estimation, the forward function, backward function, and likelihood function can be generated from (11)-(13). For part 3, the re-estimation process can be generated by to (14) and (15) if the channel is non-linear. For linear case, (16) can be used instead of (14). The parameters should be initialized and the convergence of parameters will be checked before it goes to the next step. For known data channel estimation, the likelihood function unit in part 2 will be only used to determine the input. The one-time estimation of known data channel case can be generated by (5)-(8).

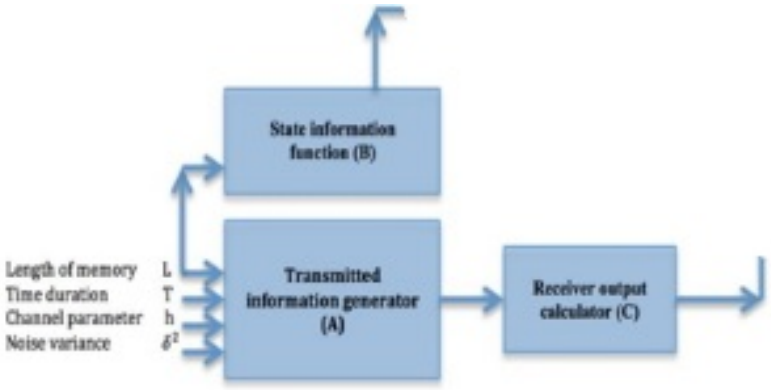


Figure 4: Level-2 part 1

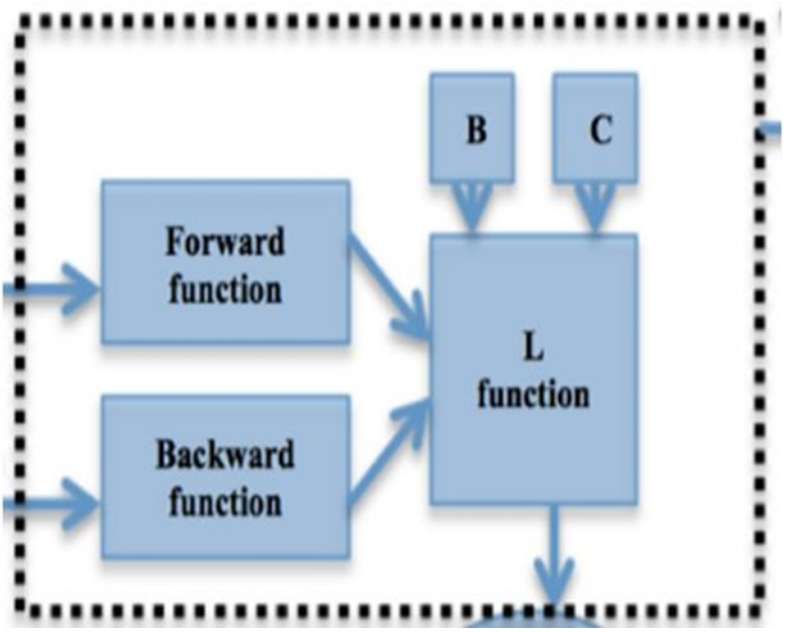


Figure 5: Level-2 part 2

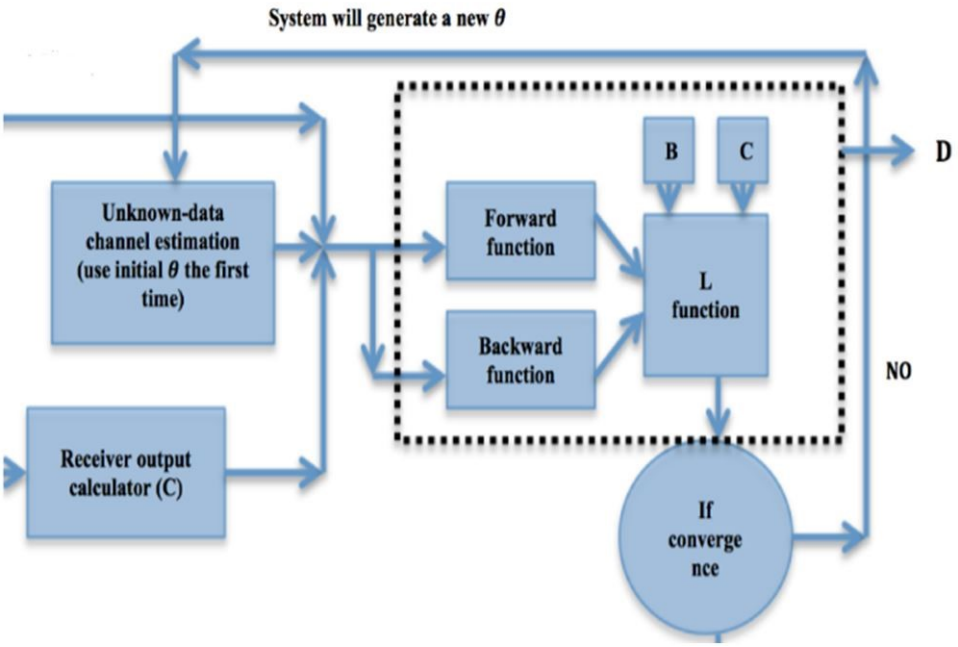


Figure 6: Level-2 part 3

When the convergence occurs and the re-estimation process is done, it will go to part 4 to estimate the input sequence, as shown in figure 7. The decision is making based on (18) and the specific form we use is explained in detail in [1]. After the sequence of information symbols is estimated, they should be compared with the original input generated by the generator in part 1 for both estimation methods.

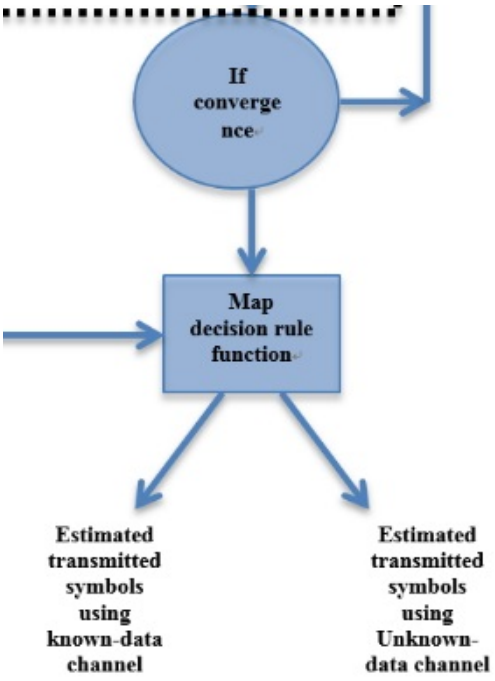


Figure 7: Level-2 part 4

4 Background Knowledge

4.1 Markov Chain and Hidden Markov Model

In a first order Markov chain model, the future states are independent of the past states given the current state. Here we only consider the discrete-time Markov chains in which the states change at certain discrete instances, indexed by an integer variable n. At time t, the state of the chain is denoted by S_t, and it belongs to a state space S. Markov chain is described by its transition probabilities, $\rho_{ij} = P(S_{t+1} = j|S_t = i),\ i, j \in S$. [4]

Also from [4], we have the following specifications of a Markov model:

1. The set of states $S = \{1, ..., m\}$.
2. The set of possible transitions for pairs of i and j. The transition probabilities, ρ_{ij}, cannot be negative.
3. Markov property:

$$P(S_{t+1}|S_t = i, S_{t-1} = i_{t-1}, ..., S_0 = i_0) = P(S_{t+1} = j|S_t = i) \tag{1}$$

   for all times t, all states S, and all possible sequences $i_{t-1}, ..., i$ of earlier states.

A hidden Markov model(HMM) is a statistical Markov model with hidden states. From [3], it has the following parameters:

1. State transition probabilities, $A = \{a_{ij}\}$, where $a_{ij} = P(q_{t+1} = S_j|q_t = S_i)$.
2. Observation probability distribution, $B = \{b_i(v)\}$, where $b_i(v) = P(O_t = v|q_t = S_i)$.
3. Initial state distribution, $\pi = \{\pi_i\}$, where $\pi_i = P(q_1 = S_i)$.
4. Model, λ, where $\lambda = (A, B, \pi)$.

There are three fundamental problems in a HMM. They are evaluation, decoding, and learning problems. In the evaluation problem, we are to find $P(O|\lambda)$, given O and λ. In the decoding problem, we are to find the optimal state sequence Q that maximizes $P(O|\lambda)$, given O and λ. Lastly, for the learning problem, we are to find the model, λ, that maximizes $P(O|\lambda)$, given O.

The evaluation problem be be used for isolated recognition, such as word. The decoding problem can be used for continuous recognition as well as segmentation. The learning problem must be solved if we want to train an HMM for subsequent use of recognition tasks.

For this project, as proposed in [1], we have the following definition:

1. $\mathcal{A}$: the sample space of transmitted information symbols.
2. $A_t : t = 1, 2, ...$: the transmitted information symbols which are independent and identically distributed random variables.

$$A_t \in \mathcal{A} = \{\alpha_m : m = 1, 2, ..., |\mathcal{A}|\},\ \alpha_m \in \mathcal{R} \tag{2}$$

3. $\mathcal{S}$: a state space where the state S_t at time t labels the channel memory $(A_{t-1}, A_{t-2}, ..., A_{t-L})$.

$$S_t \in \mathcal{S} = \{1, 2, ..., |\mathcal{S}|\},\ |\mathcal{S}| = |\mathcal{A}|^L \tag{3}$$

4. $\mathcal{X}$: the set of all possible transitions X_t from a state S_t to a state S_{t+1}.

$$X_t \in \mathcal{X} = \{\xi_n : n = 1, 2, ..., |\mathcal{X}|\},\ |\mathcal{X}| = |\mathcal{A}|^{L+1} \tag{4}$$

4.2 Estimation of Parameters on Known Transmitted Sequence

Kaleh and Vallet shows the estimation method of the parameters in [1] when the transmitted sequence is given. It can be summarized from [1] that if a T-sequence of branches X and an observed T-sequence Y are given, the maximum likelihood estimate estimate, $\widehat{\theta}$, of θ would then be the value of θ which maximizes the conditional likelihood $p(Y;\theta|X$. The estimate of $h(\xi_n)$ would be given by the following definition and equations:

$$\delta(x_t,\xi) = \begin{cases} 1 & X_t = \xi_n \\ 0 & otherwise \end{cases} \tag{5}$$

$$\widehat{h}(\xi_n) = \frac{\sum_{t=1}^{T} y_t \delta(X_t,\xi_n)}{\sum_{t=1}^{T} \delta(X_t,\xi_n} \tag{6}$$

$$\widehat{\sigma}^2 = \frac{1}{T}\sum_{t=1}^{T} |y_t - \widehat{h}(\xi_n)|^2 \tag{7}$$

$$\{\sum_{t=1}^{T} x_t^* x_t^{tr}\}\widehat{H} = \sum_{t=1}^{T} y_t \xi_n^* \tag{8}$$

4.3 Baum-Welch Algorithm

The Baum-Welch algorithm is an efficient algorithm that can calculate the parameter and the probability of observation iteratively using the forward-backward algorithm.

From [1] and [3], we can summarize the contribution at a time t of the past and of the future by the following definitions:

$$\gamma_t(i) \triangleq \rho(y_1, y_2, ..., y_t, S_t = i) \tag{9}$$

$$\beta_{t+1}(j) \triangleq \rho(y_{t+1}, y_{t+2}, ..., y_T | S_{t+1} = j) \tag{10}$$

with $\gamma_1(i) = P[S_1 = i]$ and $\beta_T(j) = 1$ for all i and j, we can deduce the following forward and backward recursion as shown by [1]:

$$\gamma_t(j) = \sum_{i\in S} \gamma_{t-1}(i) P[S_t = j|S_{t-1} = i] p(y_{t-1}|S_t = j, S_{t-1} = i) \tag{11}$$

$$\beta_t(i) = \sum_{j\in S} \beta_{t+1}(j) P[S_{t+1} = j|S_t = i] p(y_t|S_{t+1} = j, = S_t = i) \tag{12}$$

Then, the weighted conditional likelihood can be calculated as follows:

$$L(Y, X_t = \xi_n) = p(y_t|S_{t+1} = j, S_t = i) P[S_{t+1} = j|S_t = i] \gamma_t(i) \beta_t(j) \tag{13}$$

From [1], the Baum-Welch algorithm can be summarized as follows:

1. Use the forward recursion to calculate $\gamma_t(j)$.
2. Use the backward recursion to calculate $\beta_t(i)$.
3. Use $\gamma_t(j)$ and $\beta_t(i)$ to calculate $L(Y, X_t = \xi_n)$.
4. Use $L(Y, X_t = \xi_n)$ to calculate new re-estimations of the parameters $\widehat{\theta}$.
5. If $L(Y, \widehat{\theta})$ is greater than a predetermined threshold, then stop. Otherwise, go to step 1.

For this project, the re-estimation formulas, found in [1], are as follow:

$$h^{i+1}(\xi_n) = \frac{\sum_{t=1}^{T} L(Y, X_t = \xi_n; \theta^i) y_t}{\sum_{t=1}^{T} L(Y, X_t = \xi_n; \theta^i)} \tag{14}$$

$$(\sigma^2)^{i+1} = \frac{\sum_{t=1}^{T} \sum_{n=1}^{|X|} L(Y, X_t = \xi_n; \theta^i) |y_t - h^{i+1}\xi_n|^2}{\sum_{t=1}^{T} \sum_{n=1}^{|X|} L(Y, X_t = \xi_n; \theta^i)} \tag{15}$$

where $L(Y, X_t = \xi_n; \theta^i)$ is defined as in (13). For simplicity, the parameters θ will be omitted.

From [1] and [2], we can also calculate the estimate of the channel impulse response H^{i+1} by the following equation:

$$\{\sum_{t=1}^{T} \sum_{n=1}^{|X|} L(Y, X_t = \xi_n; \theta^i) \xi_n^* \xi_n^{tr}\} H^{i+1} = \sum_{t=1}^{T} \sum_{n=1}^{|X|} L(Y, X_t = \xi_n; \theta^i) y_t \xi_n^* \tag{16}$$

4.4 MAP Decision Rule

Suppose X is a discrete random variable where each realization represents the transmitted symbol of the communication system. X takes M value denoted by $\{1, ..., M\}$. The value of X are transmitted over a channel whose output is Y. The MAP decision rule can be described as follows:

For a receive value y of Y, the optimal receiver calculates

$$L_m(y) := p_x(m) f_{Y|X}(y|m) \tag{17}$$

for every value of $m \in \{1, ...M\}$ and choose the value of m that maximizes $L_m(y)$ as the estimate of X:

$$\widehat{X} = argmax_{m \in \{1,...M\}} \{\rho_x(m) f_{Y|X}(y|m)\} \tag{18}$$

This decision rule will be used for information symbol detection. [5]

In our project, the MAP, decision rule will be slightly modified as we need to sum up all branches that generate the same information symbol on a specific time index. The derivation and corresponding equation can be found in [1].

4.5 Scaling

As shown in [1], and underflow may arise when calculating the weighted conditional likelihoods, (13), due to too many product terms as time duration is large or due to the exponential terms in the Gaussian pdf. From [3], a method for scaling these computations are found as follow:

$$\alpha_t(i) = \sum_{j=1}^{N} \widehat{\alpha}_{t-1}(j) a_{ij} b_j(O_t) \tag{19}$$

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)} \tag{20}$$

$$\widehat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \widehat{\alpha}_{t-1}(j) a_{ij} b_j(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \widehat{\alpha}_{t-1}(j) a_{ij} b_j(O_t)} \tag{21}$$

5 Detail Design

Figure 8 shows the flow chart of the algorithms for both estimation methods. The difference in term of the function blocks in these two method is shown earlier in section 3. It should be simple to understand the algorithm and flow chart of the known data channel estimation. Also, the flow chart should be clear as we explained each function units of the unknown data channel estimation in the level-2 design. The output Y can be calculated by $Y_t = h(X_t) + N_t)$ where N_t is the zero mean Gaussian white noise.

The data flow diagram is shown in figure 9. WE also specify the type and size of each input, output, and intermediate variables in the diagram. For example, if we transmit a T-sequence binary information symbol with a memoryless channel($L = 1$), the state information should be stored in a T^*1 vector since there is no memory of the previous information. The transition branches should be store in a T^*2 matrix since the length of each state information is only one.

From the data flow diagram, we can see the data flow from part 1 through part 4 of the level-2 design. For part 1, we input L, T, h, and σ^2, and output state information S, transition branches X, and observed sequence Y. the transition probability matrix T_r can be calculated by predetermined distribution of each information symbol. For part 2, we input the initialized channel parameters and noise variance with observed sequence to get the Gaussian probability density function(pdf), and calculate the forward, backward, and likelihood function step by step as outputs. For part 3, we input the weighted likelihood function and previous inputs to estimate the parameters iteratively until it converges. After that, we output the converged parameters. For part 4, we input the estimated parameters and use the weighted likelihood and MAP decision rule to determine the input at each time index.

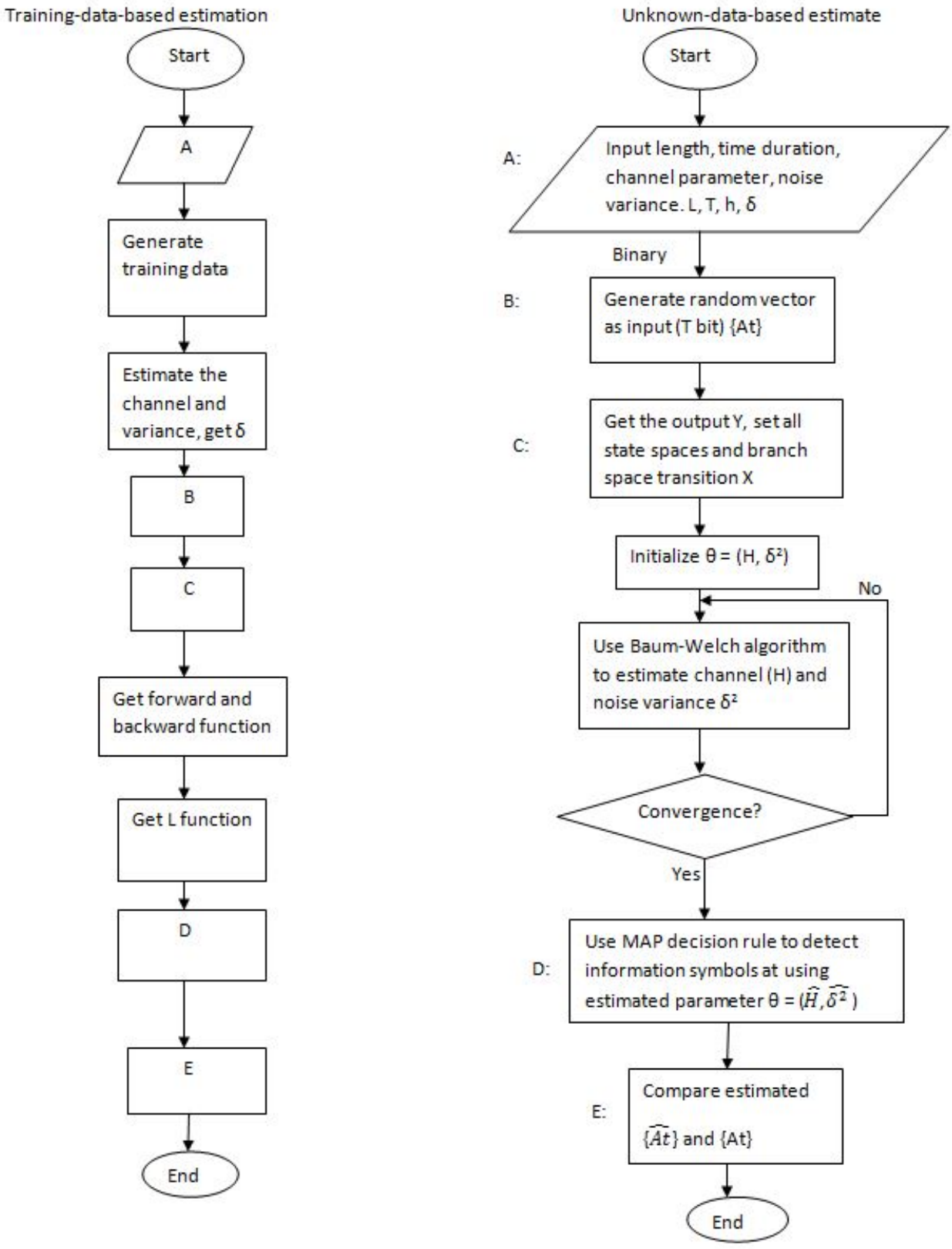


Figure 8: Flow chart

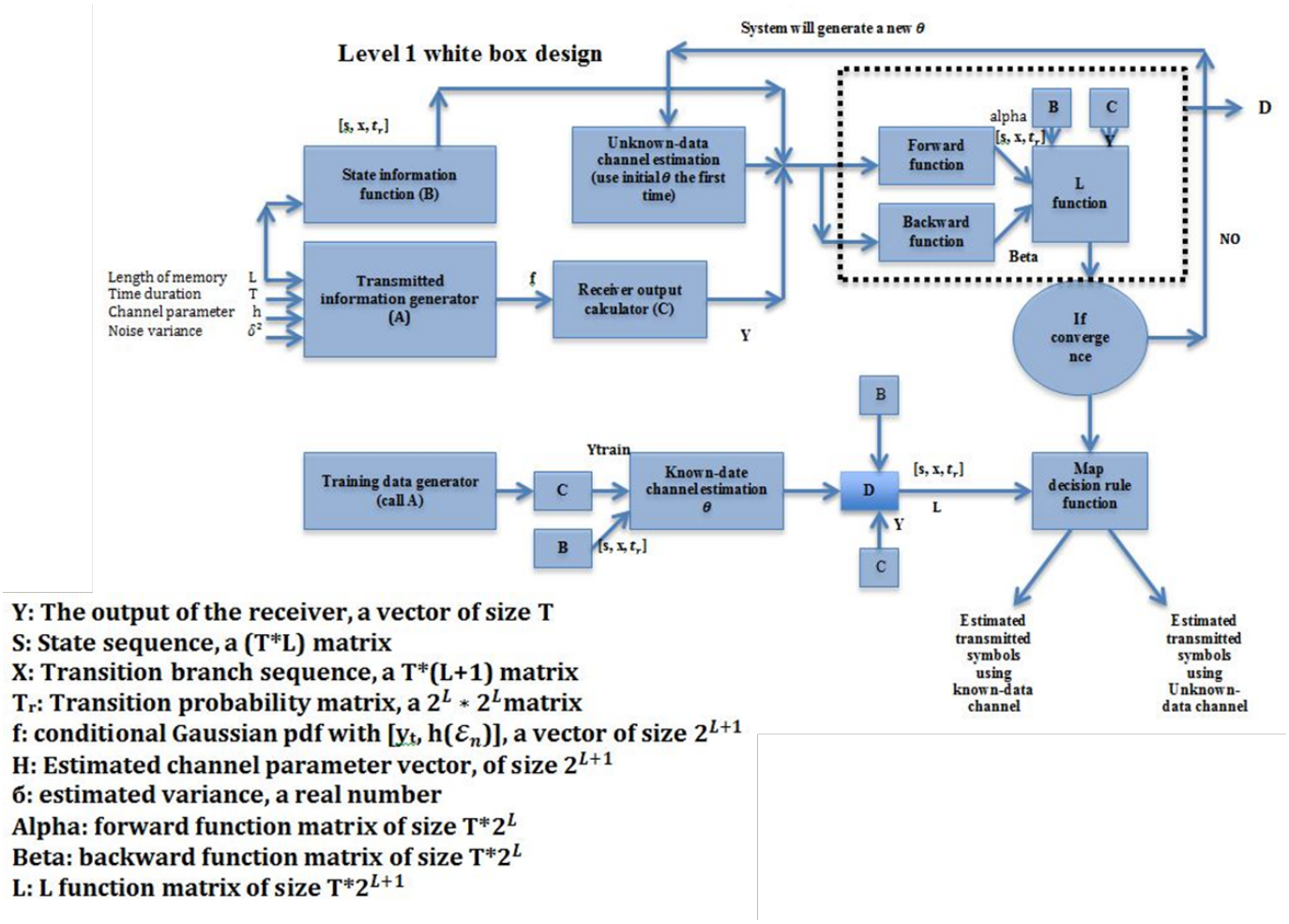


Figure 9: Flow diagram

6 Prototyping Progress Report

6.1 Simulation and Results

We designed and tested a relatively complex design on unknown symbol channel estimation. The channel is linear and memoryless, with channel length $L = 1$. The re-estimation is only for channel's parameters as we fixed the noise variance. To implement this design, we needed the following 9 function units:

1. Simulation I/O generator.
2. Forward function unit.
3. Scaling forward function unit.
4. Scaling backward function unit.
5. Scaling weighted likelihood function unit.
6. Channel parameters re-estimation unit.
7. MAP decision rule unit.
8. Plot unit.

9. Program "wrapper" unit.

The Matlab code for each function unit of this design can be found in the appendix section. All the codes are of our own design. The two main Matlab built-in functions we use are normrnd() and normpdf(), which give us the respective random Gaussian noise given the mean and variance, and Gaussian pdf given the output with estimated mean and estimated variance.

The tested value we chose are as follow:

1. Sample space for transmitted information symbols $As = [-1, 1]$ with equal probability.
2. Time duration $T = 100$.
3. Channel impulse response $H = [0.4; 0.8]$ as a column vector.
4. Gaussian noise variance $Nvar = 0.1$.
5. Number of iterations $n = 15$.
6. Initial value $H0 = [0.5; 0.6]$ and $Nvar0 = 0.1$.

Figure 10 shows the observation sequence waveform. We can see that each point in the sequence is close to one of the horizontal dash lines, which are the outputs of the channel. They are also the means of the Gaussian distribution. It should be intuitive to see that the Gaussian white noise is added successfully. In figure 11, the weighted likelihood increases along n monotonically and the channel's parameters reach convergence quickly. In this particular case, the MAP decision rule-based detector shows that the accuracy of input estimation is 100% since the noise variance is small.

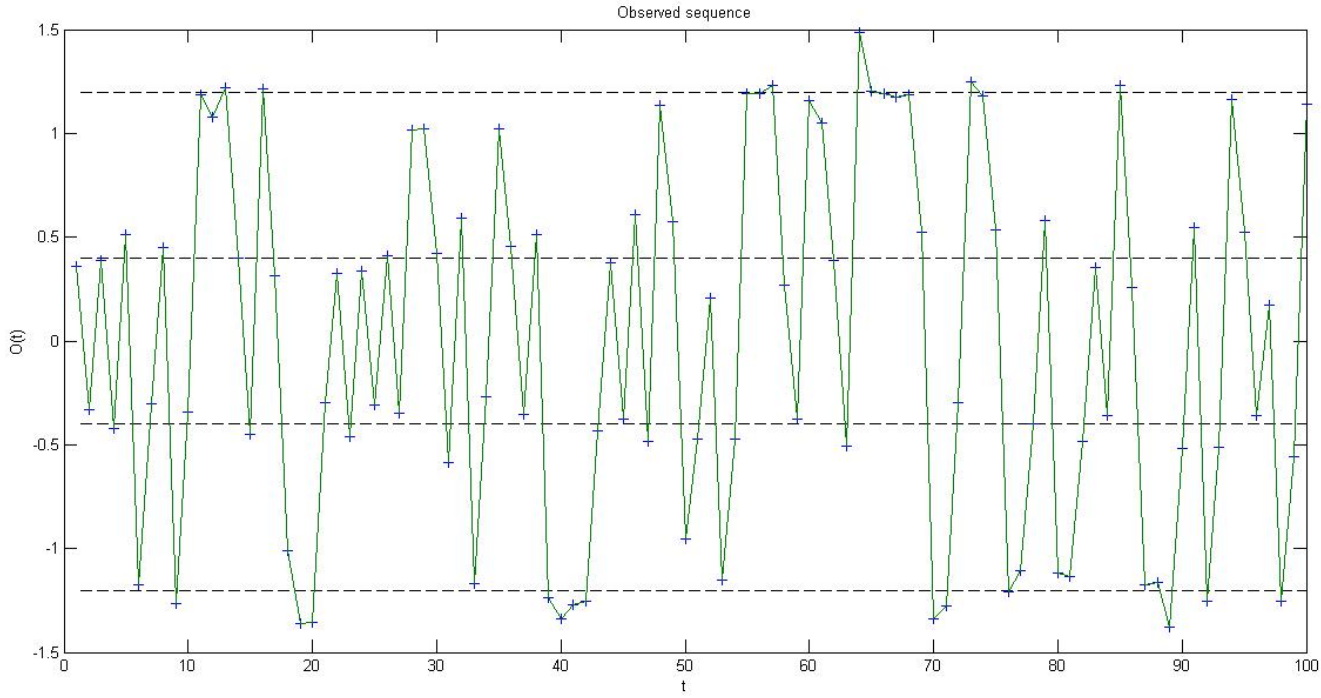
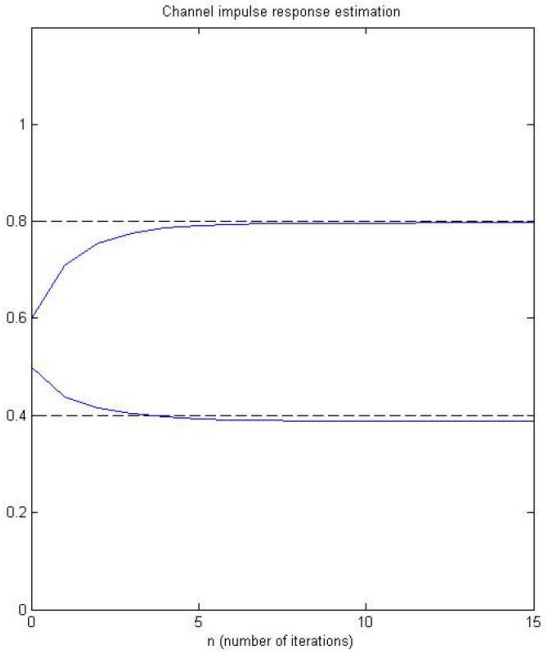
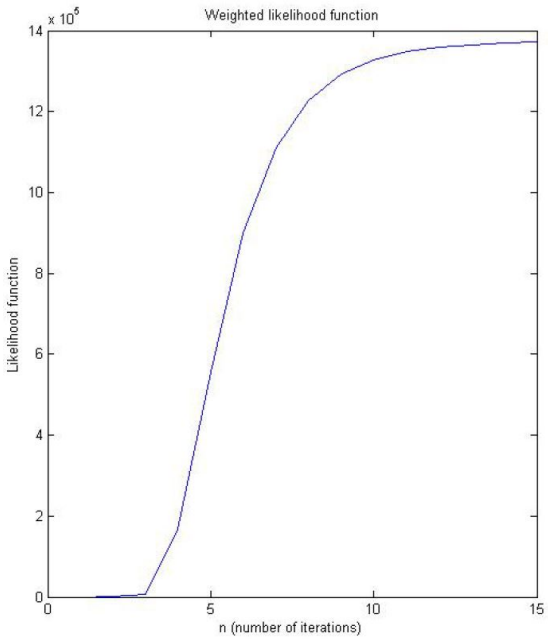


Figure 10: Observed sequence given $Nvar = 0.1$



(a) Channel's parameter re-estimation



(b) Likelihood function

Figure 11: Channel's parameters re-estimation and likelihood function given given $Nvar = 0.1$

We then changed the noise variance to $Nvar = 0.2$ and keep all other variables the same. Figure 12 shows the waveform of the output received by the receiver. In figure 13, when the noise level is increased from 0.1 to 0.2, the re-estimation performance decreased slightly. Figure 14 shows that the accuracy for estimation of the input sequence is reduced from 100% to 99%.

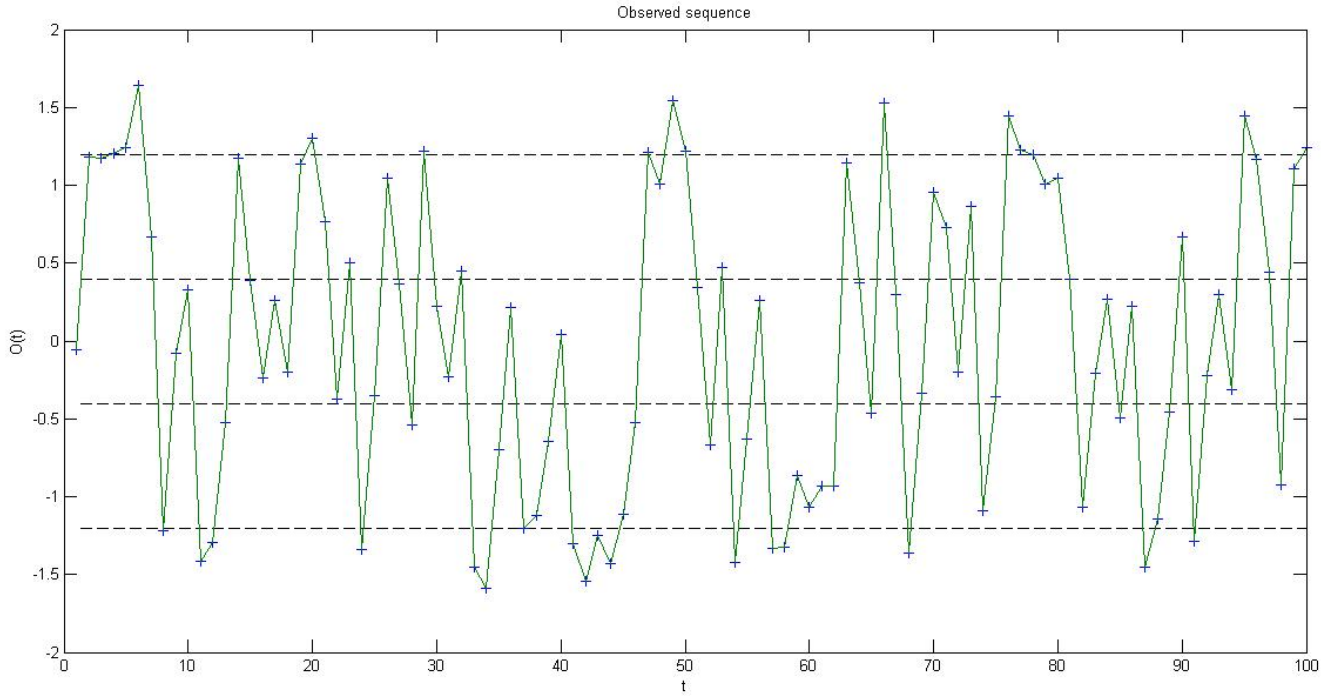
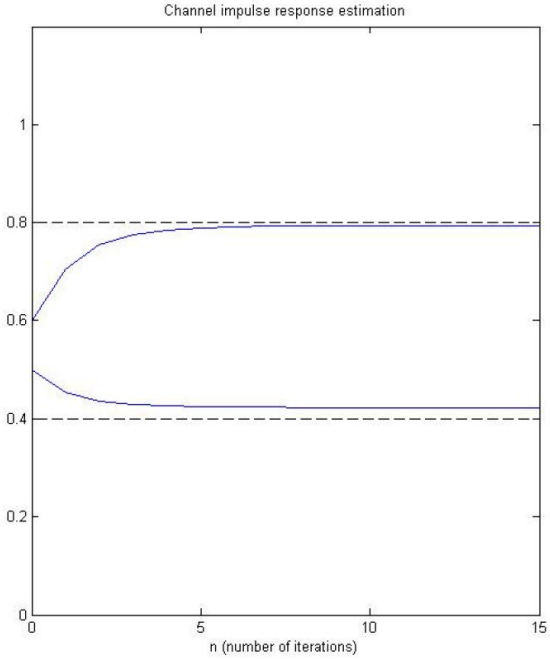
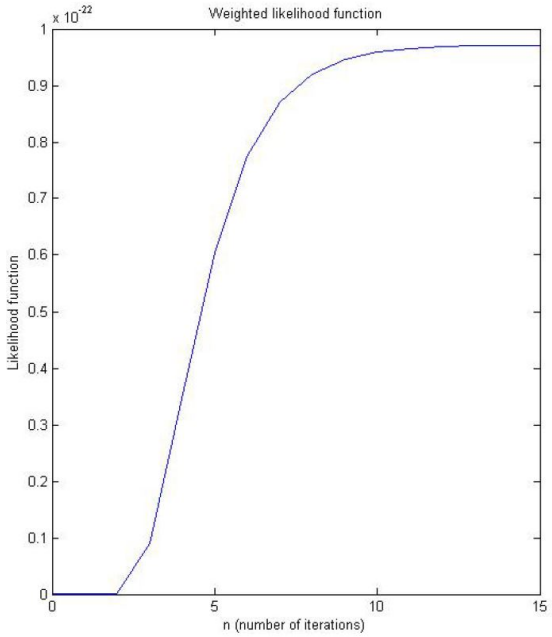


Figure 12: Observed sequence given $Nvar = 0.2$



(a) Channel's parameter re-estimation



(b) Likelihood function

Figure 13: Channel's parameters re-estimation and likelihood function given given $Nvar = 0.2$

```
     1     1     1     1     1     1    -1    -1     1    -1    -1    -1     1     1    -1     1    -1     1

  Columns 19 through 36

     1     1    -1     1    -1    -1     1     1    -1     1     1    -1     1    -1    -1    -1     1    -1

  Columns 37 through 54

    -1    -1     1    -1    -1    -1    -1    -1    -1     1     1     1     1     1    -1     1    -1    -1

  Columns 55 through 72

     1    -1    -1    -1    -1    -1    -1     1     1    -1     1     1    -1    -1     1     1    -1     1

  Columns 73 through 90

     1    -1     1     1     1     1     1     1    -1    -1     1    -1     1    -1    -1    -1     1    -1

  Columns 91 through 100

    -1     1    -1     1     1     1    -1     1     1     1

accuracy =

    0.9900

fx >>
```

Figure 14: Accuracy of estimation given $Nvar = 0.2$

The performance decreases further when we set $Nvar = 0.4$. The result can be seen in figures 15, 16, and 17. The accuracy in this case is lowered to 94%. Notice that in this case, the likelihood function is still monotonic.

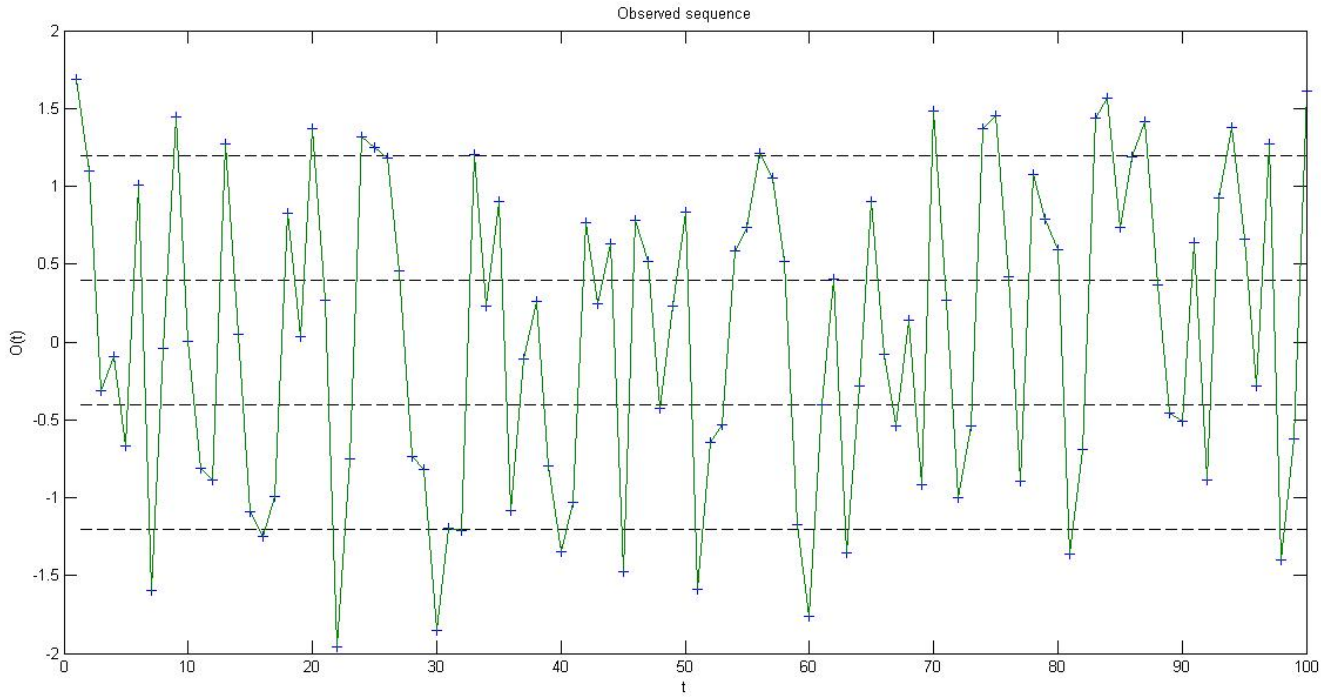
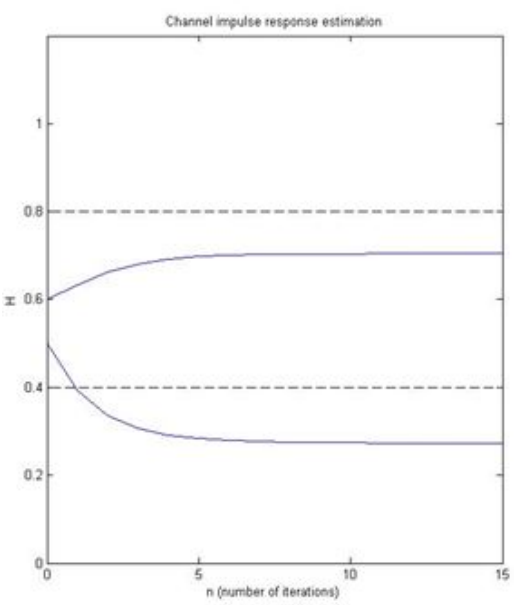
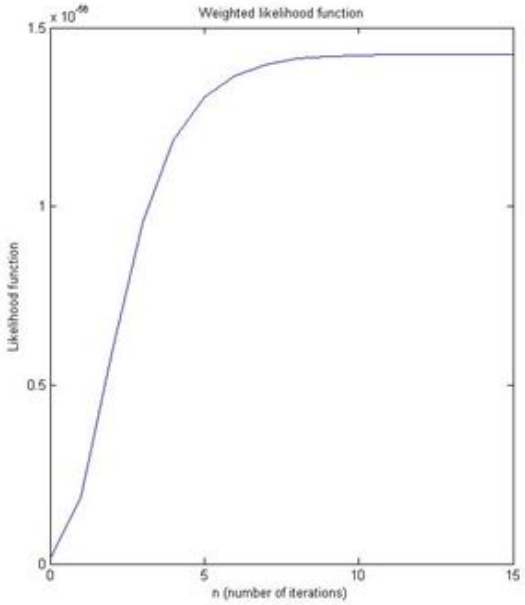


Figure 15: Observed sequence given $Nvar = 0.4$



(a) Channel's parameter re-estimation



(b) Likelihood function

Figure 16: Channel's parameters re-estimation and likelihood function given given $Nvar = 0.4$

```
    1    -1     1    -1     1    -1    -1     1     1    -1    -1     1     1    -1    -1    -1     1    -1

  Columns 19 through 36

    1     1    -1    -1     1     1     1     1    -1    -1    -1    -1    -1     1     1     1    -1    -1

  Columns 37 through 54

    1    -1    -1    -1     1     1     1    -1     1     1    -1    -1     1    -1    -1    -1     1     1

  Columns 55 through 72

    1     1     1    -1    -1    -1     1    -1    -1     1     1    -1     1    -1     1     1    -1    -1

  Columns 73 through 90

    1     1     1    -1     1     1     1    -1    -1     1     1     1     1     1     1    -1    -1     1

  Columns 91 through 100

   -1     1     1     1    -1     1    -1    -1     1     1

accuracy =

    0.9400

fx >>
```

Figure 17: Accuracy of estimation given $Nvar = 0.4$

We can deduce from these results that the performance of the algorithm goes down as the noise variance, or noise power, goes up while fixing other variables. When the channel's outputs, or the means of the Gaussian distribution, are the same and we increase the noise variance, the overlaps of the Gaussian distribution curves between different decisions becomes larger, and hence increase the probability of error.

We also expand our code to the case for an unknown linear channel with memory, where the length of memory equals one. The code is based on the code we provide in the Appendix, and is not included in this document. We tested the parameters simulated in [1]. We set $T = 64$, $H = [0.5; 0.7; 0.5]$, $H0 = [0; 0.1; 0]$, $Nvar = Nvar0 = 0.2$, and $n = 15$. The results are shown in figures 18-20. Figure 19 shows fast convergence. The accuracy in this case is 100%.

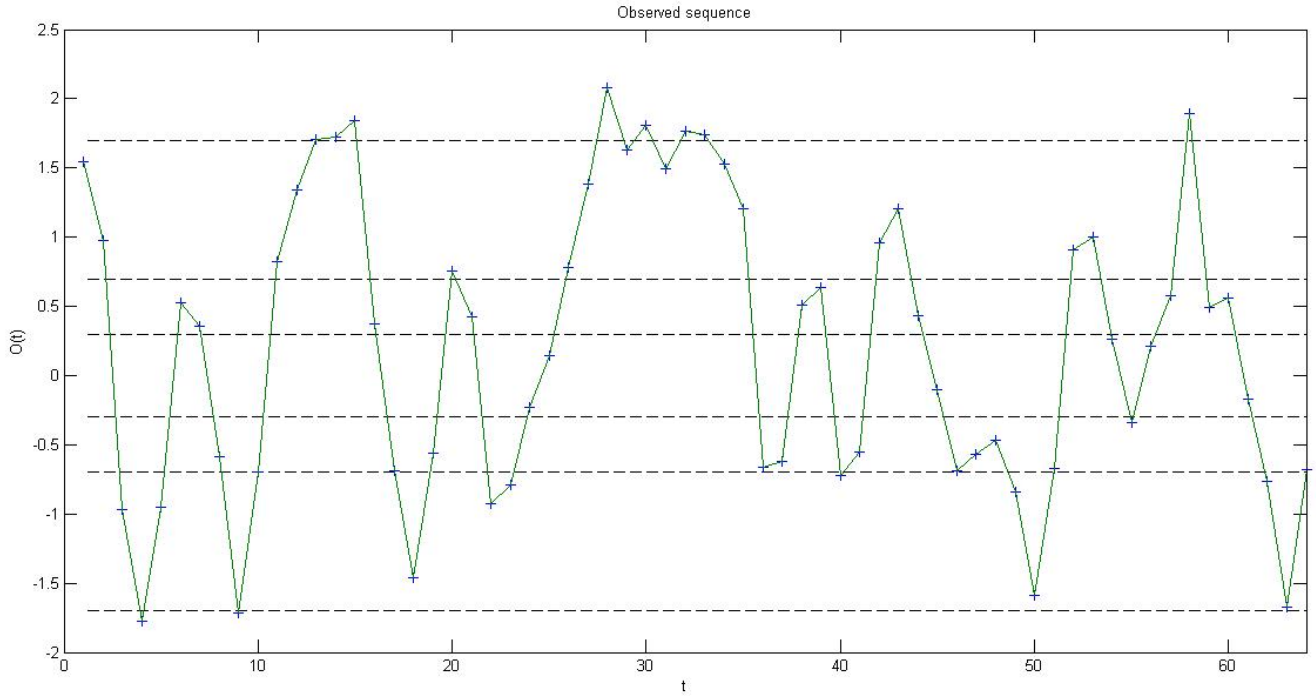


Figure 18: Observed sequence given $Nvar = 0.2$ and $H0 = [0; 0.1; 0]$

6.2 Issues Encountered

We encountered several problems and manged to solve some of them before arriving at the reasonable simulation results.

The first problem was how to number the state space and transition branch space properly in order to transform each one from the other. It was crucial because we need to know both of them when calculating the forward, backward, and weighted likelihood function.

The second problem was the scaling problem. Initially, without proper scaling, as time duration T becomes large, for example $T = 750$, the forward,backward, and likelihood functions will generate an underflow due to too many products of probability which is in the range from 0 to 1. To solve this problem, we used the scaling method shown with the formula(19)-(21). In this case, with a good initialization of channel's parameters, the system can handle the case as $T > 1000$. However, due to the exponential terms in the Gaussian pdf, an underflow may still occur if the noise variance is small even after we scale the forward and backward function as shown above. This can be solved by taking the logarithm on each term of the likelihood function. For Gaussian pdf, the natural logarithm can be calculated directly. If the logarithm result is out of the logarithm of the double precision range, we can round the likelihood to the minimal number that the double precision allows in the machine.

We also have one unsolved problem. When we set the noise variance to a very small number, the performance of the re-estimation on channel parameters becomes even worse than a relatively large noise variance. Occasionally, we would get a "matrix close to singular" warning form Matlab command window. This problem is probably a machine problem from Matlab. When trying to solve equation (16), the matrix on the left side is close to singular due to its many small elements. This means the determinant of the matrix is out of the double precision range. We have not found a good solution to solve this problem yet. This problem should only occur in linear channel case when we have to handle the matrix operation.

7 Testing Plan

The testing plan is based on Matlab code implementation and debugging. The following experiments will be done in order.

7.1 Generating The Observed Sequence

1. After we define the state space, the time duration T, the channel impulse response H, and the Gaussian noise variance $Nvar$, we will then try to generate a random input A.
2. We will generate the state information S and transition branches matrix X with the generated input A.
3. We will generate the observed sequence with the Gaussian function $N(h, Nvar)$ after we calculate the output of the channel h. The resulting figure will show whether it has additive Gaussian noise.

7.2 Likelihood Calculation and Re-estimation of Channel's Parameters with Known Gaussian Noise Variance

1. As we get the transition probability matrix and observed sequence with some initialization of the channel's parameters, we will calculate the forward and backward function. Failure on scaling will cause a warning on singular matrix and return a NaN for our forward and backward function.
2. As an extension from step 1, the scaling likelihood function is given by $\frac{E_b}{N_0} = 8dB$ and $T = 1000$ should not be a NaN matrix, caused by an underflow. To ensure a good re-estimation, the actual likelihood function should increase monotonically.

6.2 Issues Encountered

We encountered several problems and manged to solve some of them before arriving at the reasonable simulation results.

The first problem was how to number the state space and transition branch space properly in order to transform each one from the other. It was crucial because we need to know both of them when calculating the forward, backward, and weighted likelihood function.

The second problem was the scaling problem. Initially, without proper scaling, as time duration T becomes large, for example $T = 750$, the forward,backward, and likelihood functions will generate an underflow due to too many products of probability which is in the range from 0 to 1. To solve this problem, we used the scaling method shown with the formula(19)-(21). In this case, with a good initialization of channel's parameters, the system can handle the case as $T > 1000$. However, due to the exponential terms in the Gaussian pdf, an underflow may still occur if the noise variance is small even after we scale the forward and backward function as shown above. This can be solved by taking the logarithm on each term of the likelihood function. For Gaussian pdf, the natural logarithm can be calculated directly. If the logarithm result is out of the logarithm of the double precision range, we can round the likelihood to the minimal number that the double precision allows in the machine.

We also have one unsolved problem. When we set the noise variance to a very small number, the performance of the re-estimation on channel parameters becomes even worse than a relatively large noise variance. Occasionally, we would get a "matrix close to singular" warning form Matlab command window. This problem is probably a machine problem from Matlab. When trying to solve equation (16), the matrix on the left side is close to singular due to its many small elements. This means the determinant of the matrix is out of the double precision range. We have not found a good solution to solve this problem yet. This problem should only occur in linear channel case when we have to handle the matrix operation.

7 Testing Plan

The testing plan is based on Matlab code implementation and debugging. The following experiments will be done in order.

7.1 Generating The Observed Sequence

1. After we define the state space, the time duration T, the channel impulse response H, and the Gaussian noise variance $Nvar$, we will then try to generate a random input A.
2. We will generate the state information S and transition branches matrix X with the generated input A.
3. We will generate the observed sequence with the Gaussian function $N(h, Nvar)$ after we calculate the output of the channel h. The resulting figure will show whether it has additive Gaussian noise.

7.2 Likelihood Calculation and Re-estimation of Channel's Parameters with Known Gaussian Noise Variance

1. As we get the transition probability matrix and observed sequence with some initialization of the channel's parameters, we will calculate the forward and backward function. Failure on scaling will cause a warning on singular matrix and return a NaN for our forward and backward function.
2. As an extension from step 1, the scaling likelihood function is given by $\frac{E_b}{N_0} = 8dB$ and $T = 1000$ should not be a NaN matrix, caused by an underflow. To ensure a good re-estimation, the actual likelihood function should increase monotonically.

7.3 Convergence Monitoring and Accuracy Calculation

1. We will make fixed and horizontal lines for the simulated channel's parameters H, and see if and when the estimated channel's parameters reach convergence.
2. After we obtain a close estimation of the channel's parameters, we will calculate the scaling likelihood function with this estimation and use the MAP decision to determine the input symbols.

7.4 Evaluation Criteria

If the observed sequence is a signal with additive noise, the likelihood function in the figure tends to be monotonically increasing along with the number of iterations and the channel's parameters in the figure tend to converge. This will be considered as a success. Otherwise, a failure occurs with this estimate method at some point.

8 List and Description of Tasks

8.1 Known-data Channel(Memoryless and Linear) Estimation Design Using Training Data

1. Generate training data and its corresponding output, state information, and transition branch for time duration $T = 100$.
2. Design the function unit with training data to estimate the channel parameters and noise variance using equations (5), (7), and (8).
3. Generate input data and its corresponding output, state information and transition branch for time duration $T = 100$.
4. Use the scaling weighted likelihood function calculator unit to calculate the likelihood function which determines the joint probability of the observed sequence and each transition branch at each time index t.
5. Use the MAP decision rule-based detector with estimated parameters to estimate the input symbols.
6. Compare the estimated input with the original one and examine the accuracy of this method.

8.2 Known-data Channel(Memoryless and Non-linear) Estimation Design Using Training Data

1. Repeat the design in 8.1 but use formula (6) instead of (8).

8.3 Known-data Channel(Memory Length $L = 1$ and Linear)Estimation Design Using Training Data

1. Modify the simulation I/O generator unit with $L = 1$ on the state space, transition branch function matrix and output of the channel accordingly.
2. Modify the size of transition probability matrix and its elements respectively. Formulate the updated relation between state space index and transition branch index.
3. Repeat 8.1 with the new change.

8.4 Known-data Channel(Memory Length $L = 1$ and Non-linear)Estimation Design Using Training Data

1. Combine 8.2 and 8.3.

8.5 Unknown-data Channel(Memoryless and Non-linear) Estimation Design

1. Repeat our Matlab design in the appendix but use formula (14) this time.

8.6 Unknown-data Channel(Memory Length $L = 1$ and Linear) Estimation Design

1. Modify the simulation I/O generator unit with $L = 1$ on the state space, transition branch function matrix and output of the channel accordingly.
2. Modify the size of transition probability matrix and its elements respectively. Formulate the updated relation between the state space index and transition branch index used in all functions.
3. Repeat the correct approach we take in out prototyping activities.

8.7 Unknown-data Channel(Memory Length $L = 1$ and Non-linear) Estimation Design

1. Combine 8.5 and 8.6.

8.8 Problem Analysis and Correction(Issues occurred in our prototyping activities)

1. Collect all problems in the Matlab simulation for 8.1-8.7 such as singular matrix with bad scaling, failure on reaching convergence, failure on maintaining the monotonicity of the likelihood function, and initialization problems.
2. Problem analysis and solving using supporting resources.

8.9 Generic Design(Optional)

1. Design all the function units without giving any specific value on length of memory L, without specifying the channel linearity, with user's self-defied sample space for input information symbols and transition probability matrix.

8.10 Testing and Evaluation

1. Case 1: $\frac{E_b}{N_0} = 0dB$
2. Case 2: $\frac{E_b}{N_0} = 4dB$
3. Case 3: $\frac{E_b}{N_0} = 8dB$

8.11 Report Presentation and Document

1. In-progress and final report.
2. In-progress presentation to FS and final oral presentation.
3. Final report and poster.

8.12 Allocation of Responsibilities

Qing Chen: 8.4-8.11
Xiaolong Fang: 8.1-8.5, 8.10-8.11
Yian Hu: 8.1, 8.4-8.8, 8.10-8.11
Stephen Owusu: 8.1-8.5, 8.10-8.11
Liem Tran: 8.1, 8.4-8.8, 8.10-8.11
Nguyen Tran: 8.1-.5, 8.10-8.11

The allocation of responsibilities is just an initial distribution of work to each group member. Depend on the the progress of each part that each group member makes, the responsibilities of each member can be changed. Each group member should gain reasonable understanding for parts beyond their allocation.

The project is based on Matlab simulation. Each design will be made by at least two or three members as a group effort. The entire team will work together to optimize any of the designs in case there is an unsatisfying outcome or evaluation that occurs.

9 Schedule and Milestones

Figure 21 shows the milestones and schedule of the time we plan to spend on this project in the future. We have 14 weeks the next semester to complete the project. The week number is shown as the week number of the year, not as the week number after the project start, for example, week number 36 is the first week of the project.

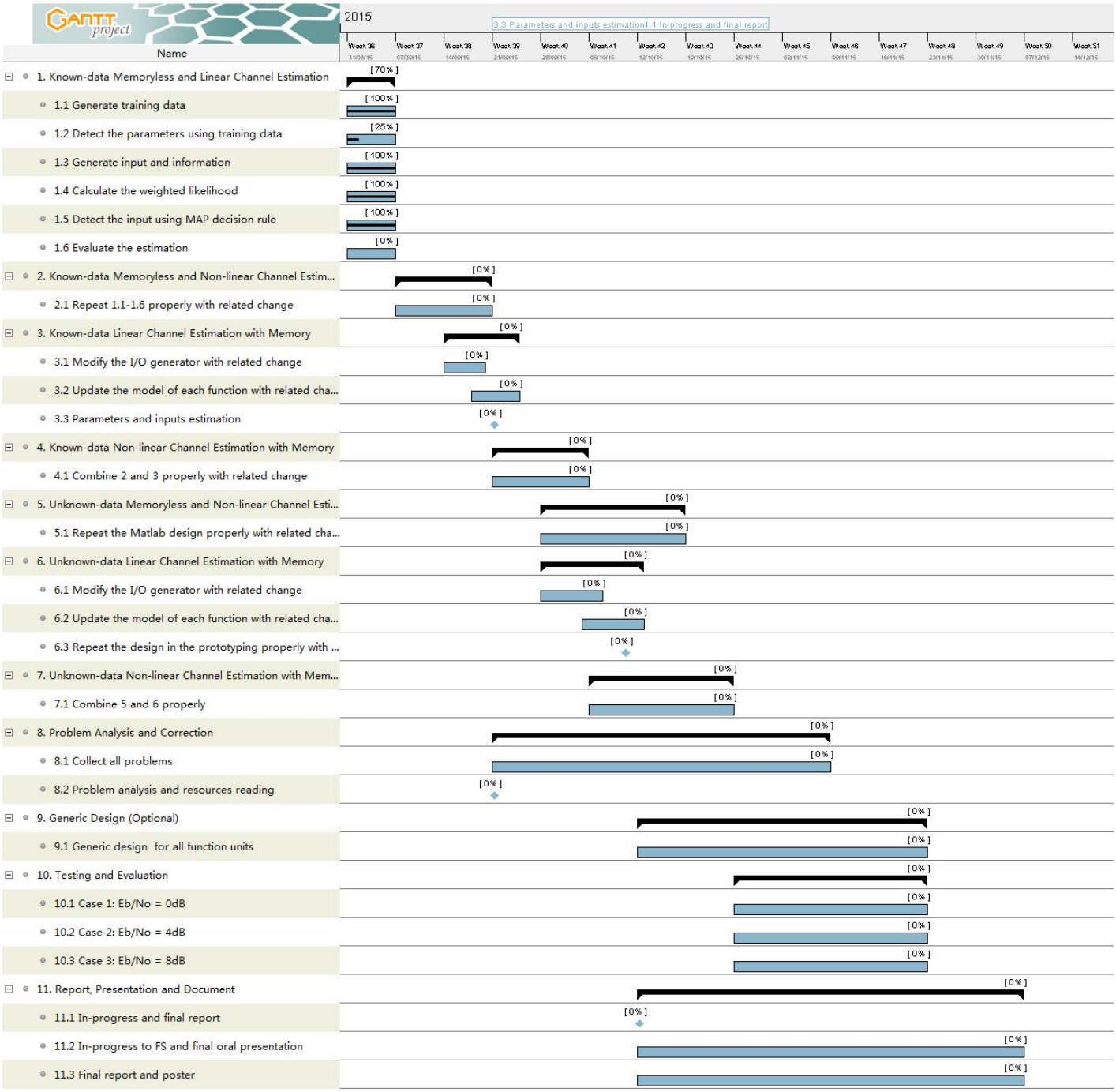


Figure 21: Schedule and milestone

References

[1] G. K. Kaleh and R. Vallet, Joint parameter estimation and symbol detection for linear or nonlinear unknown channels, IEEE Trans. Commun., vol. 42, pp. 2406-2413, July 1994.

[2] Y. Ephraim and N. Merhav. Hidden Markov processes. IEEE Transactions on Information Theory, 48:6 (2002), 1518-1569.

[3] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE, vol. 77, pp.257-286, Feb. 1989.

[4] D. P. Bertsekas and J. N. Tsitsiklis. Markov Chains, in Introduction to Probability, 2nd ed. Nashua, NH: Athena Scientific, 2008, pp. 339-405..

[5] Y. Ephraim. Digital Communication over AWGN, ECE 460 Lecture Notes: Communication and Information Theory. Fairfax, VA: George Mason University, 2015, pp. 81-107.