Spectrum Access System: Design and Implementation of the Decision-Feedback Equalizer in Software

Heyi Sun
Anant Sahai, Ed.
John Wawrzynek, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-65
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-65.html

May 11, 2017

Acknowledgement

Professor Anant Sahai
Professor John Wawrzynek
Christopher Yarp
Colin de Vrieze

Spectrum Access System
Technical Contributions

Heyi Sun

1. Introduction and Background

1.1 Introduction to the Project

Distortion in data transmission can cause signal overlap making the original message indistinguishable. This loss of information, however, can be reduced by a well-designed equalizer (signal processing technique). This paper will cover the introduction of 2 equalization models and 4 adaptive algorithms, which will include a description of their software implementation as well as an assessments in performance and accuracy.

1.2 Multi-path Problem

There are many obstacles or objects (ex. buildings, people, plants and equipment in the lab) in the real world that reflect communication signals by creating additional paths for the signal in instead of a straight line of transmission. (As illustrated in Figure 1). These new paths will transmit the same data packet to the receiver but with a delay, resulting in multiple transmissions of the same signal at different times. The receiver unintelligently sums or overlaps all incoming signals including redundant communication which leads to a multi-path phenomenon and distortion of the original signal. The waveform of the distorted signal is illustrated in Figure 2. Equalization (Signal processing technique) is used to compensate for the distorted signal and the resulting loss in communication.

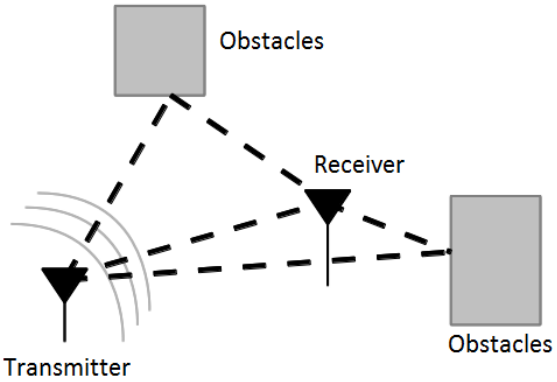


Figure 1. Illustration of Multi-Path

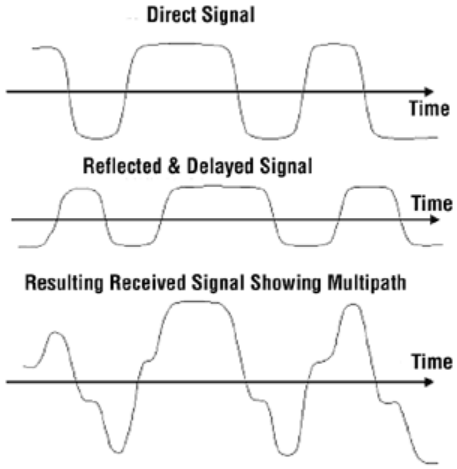


Figure 2. Waveform of direct sign vs. distorted signal

2. Theory of Equalization

2.1 Introduction to Equalization

Equalization is a signal processing technique that can reverse the distortion of the transmitted signal. The equalizer is composed of an equalization model and adaptive algorithms. This paper will cover two adaptive equalization models: Linear Equalization and Decision Feedback Equalization and four adaptive algorithms: Signed Least Mean Square (SLMS), Normalized LMS (Norm LMS) Variable-Stepped Least Mean Square and Recursive Least Square (RLS). In the real-world application, the adaptive algorithms are chosen based on tradeoffs present within its predicted environment (e.g. Noise level). In table I, I provide an overview of advantages and disadvantages of each adaptive algorithm with a specific focus on performance and accuracy.

Table 1. Advantages and Disadvantages Overview of Adaptive algorithms

	Advantage	Disadvantage
Signed LMS	● Low Computation Complexity ● Good fit for High-Speed Communication	● Relatively low accuracy
Normalized LMS	● Well adapt to Input vector with different magnitude and length	
Variable-Step LMS	● Good balance of convergence time and steady-state error ● Adaptive Step-size changing based on error	
RLS	● High accuracy	● Complexity grows exponentially within Taps ● Not ideal for high-speed communication

2.2 Introduction to Adaptive Equalization Model

The introduction of equalization models are adapted from professor Schober's lecture notes on Equalization of channels with ISI. [2]

2. Theory of Equalizer

An adaptive equalizer is the equalizer that adaptively updates its weights in response to the error.

2.2.1 Introduction to Linear Equalizer

Linear equalizer is relatively simple equalization model. The distortion is modeled by a channel transfer function H(z). In linear equalizer, the inverse function of channel transfer function F(z) is modeled by a linear filter. The linear filter inverts the channel transfer function without amplifying the noise by adaptively updating the weights based on the error (difference between the equalized signal and original signal). The block diagram of the linear equalizer is shown in Figure 3.

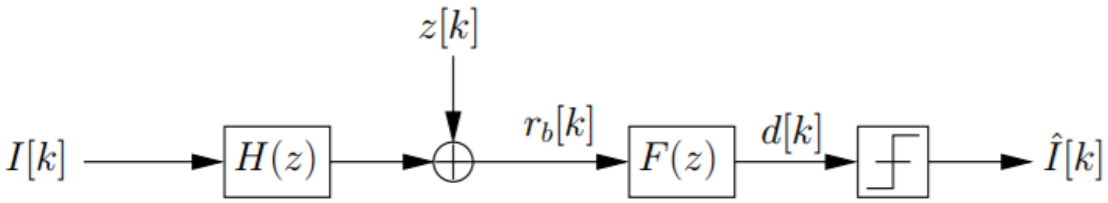


Figure 3. Block diagram of Linear Equalizer. I denote for input, H is channel transfer function and F is filter. I-hat is the output.

2.2.2 Introduction to Decision Feedback Equalizer

In linear equalizer, the linear filter may introduce additional noise variance at the output signal, this may led to poor performance. This drawback can be avoided in a Decision feedback equalizer at the expense of a more complex model. The decision feedback equalizer will predict the noise level of the channel through the noise predictor based on previous noise samples. Then predicted noise is subtracted from the input signal by using a feedback filter to reduce the noise level of the channel. The block diagram of decision feedback filter is shown in Figure 4.

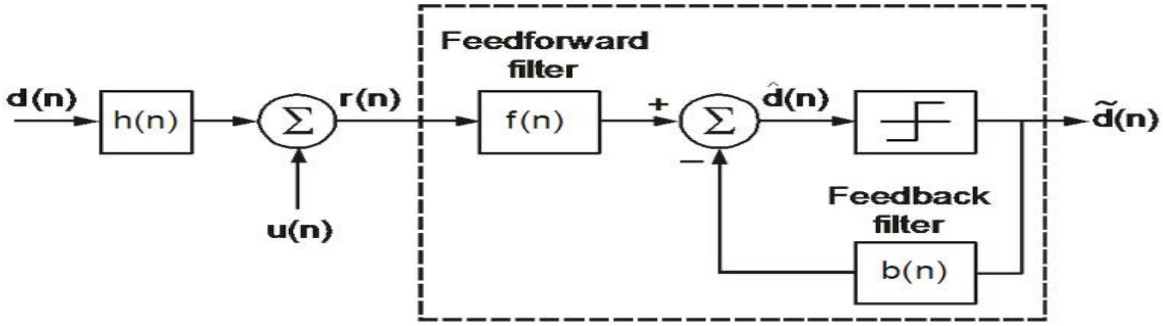


Figure 4. Block diagram of Decision Feedback Equalizer, where f(n) is feedforward

2.3 Introduction to Adaptive Algorithms

The introduction of adaptive algorithms is adapted from Professor Schober's lecture notes of Equalization of channels with ISI. [2]

In order to optimize the performance of the equalizer, the adaptive algorithms are introduced to update the weight taps.

2.3.1 Updating Weight Factor through Least Mean Square

Least Mean Square is the most common fundamental adaptive algorithm. It is a gradient descent algorithm that updates its weight taps based on the magnitude of the error. The pace of updating is controlled by the step size. The larger step size will speed up the training process but may come with low accuracy. The smaller step size can lead to higher accuracy but at the expense of longer training time.

$$f(\mathbf{u}(n), e(n), \mu) = \mu e(n)\mathbf{u}^*(n)$$

2.3.2 Updating Weight Factor through Signed Least Mean Square

Signed Least Mean Squares (SLMS) is a variant of the LMS algorithm that aims at speeding up computations and adapts to high speed communication. The difference between the SLMS and LMS is that SLMS only takes the sign of errors instead of the value of errors. If the step size chosen is the multiple of 2, then only shifting and adding operations are needed for computation. This can significantly reduce the complexity of the computational costs from weight taps updating processes. However, the SLMS speeds up the computation process at the expense of the accuracy. Since only the sign of error is taken, it can perceived as a rough quantization of gradient estimates. The drawbacks are the longer convergence time and increasing steady state error.

$$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{u}(n)\ sgn(e(n))$$

2.3.3 Updating Weight Factor through Normalized Least Mean Square

Normalized Least Mean Squares (NormLMS) is a variant of LMS algorithm that designed to compensate for the effect of large fluctuations in the power level of the input signal. NormLMS's step size is data-dependent or input dependent. The NormLMS compensates both the input vector with large value as well as large input vector length by reducing step size accordingly. The step size is inversely proportional to the modulation of the input vector. The disturbance caused by the difference of input signal is therefore minimized. The NormLMS significantly promotes the accuracy and stability of the system at the expense of a little increasing computation complexity.

$$w_j(n+1) = w_j(n) + \frac{\tilde{\mu}}{a + \|\underline{u}(n)\|^2} e(n)u(n-j)$$

2.3.4 Updating Weight Factor through Variable-Step Least Mean Square

Variable-Step Least Mean Square (VSLMS) is an adaptive algorithm that effectively resolves the inherent contradiction between convergence time and steady-state error rate in LMS model. In LMS model, The smaller step size leads to lower steady-state error but longer convergence time while the larger step size leads to faster convergence rate but higher steady-state error. VSLMS resolves the contradiction by changing the step size dynamically while

updating weight factor of the equalizer. The formula shown below is an example of VSLMS that based on sigmoid function. u(n) stands for step size.

$$\mu(n) = \beta[\frac{1}{1+e^{-\alpha|e(n)|}} - 0.5]$$

2.3.5 Updating Weight Factor through Recursive Least Square

The theory part of Kalman filter is adapted from professor Faragher's lecture note.[8]

Recursive Least Square (RLS) is an adaptive algorithm that explicitly solves for weight factor recursively using Kalman filter. Kalman filter is the optimal estimator for one-dimensional linear system with Gaussian noise. Kalman filter estimates the parameter of interest of current state based on the previous state and inverse correlation matrix. The terms laid on the the main-diagonal represent the variance of each terms in the state vector(input vector). The off-diagonal terms stand for co-variance within the terms of the state vector. Unlike the stochastic algorithms like SLMS and VSLMS, RLS solves for weight factor explicitly which gives it extremely fast convergence rate at the cost of high computational complexity.

$$K = \frac{Pu}{ForgetFactor + u^H Pu}$$

K stands for kalman gain vector, u is the input vector, P is the inverse correlation matrix and forget factor is introduced to penalized the weight of the older data.

3. Software Implementation

The signed Least Mean Square, Normalized LMS, Variable-Step LMS and Recursive Least Square adaptive algorithms are implemented in Python to extend their applications into more telecommunication-related software (ex. GNU Radio).

3.1 Implementing LMS in GNU Radio

The three variants of LMS' (Signed LMS, Normalized LMS and Variable-Steped LMS) implementation are referred to the mechanism described in Muhammad Wasimuddin and Navarun Gupta's paper [4] about the design of Least Mean Square adaptive filter and the Matlab's documentations [5] about LMS filter system. The Python function initializes the adaptive filter by creating weight taps based on the number of taps user entered and then sets the value to zero. The step size is also set to the value specified by the user.

```
def __init__(self,N,u):
    self.w= np.zeros((N,1),dtype=np.cfloat)
    self.alpha = u
```

N is the number of Taps, w is the weight factor and alpha is the step size

Let's denote the number of weight taps *len(w)*. In this specific implementation, all the predicative output before *len(w)* will omitted and it starts compute the predictive output *(y(n))* from *(len(w)+1)th* element. The predicative output is computed based on the formula on the left, which extracts the slice of the input signal which is equal to the number of the weight taps then performs an element wise multiplication with weight taps.

$$y(n) = \sum_{k=0}^{M-1} w_k u(n-k)$$

```
xt=x[(i-lenw):i]
y = self.w.T.dot(xt)
e = d[i-1]-self.w.T.dot(xt)
```

Xt is sliced input vector with the same length of the weight factor. The sum of element wise multiplication is computed by the dot product. The error of the signal is computed next based on the formula *e(n) = d(n) - y(n)* where *d(n)* is desired output or reference signal.

Then the weight factors are updated based on the formula w(n+1) = w(n) + step size*u(n)*e(n).

```
self.w+=self.alpha*(e*(xt.conjugate()))
```

The weight factor will be updated iteratively until the window is moved to the end of the input signal. The overall process of the algorithm can be visualized as the diagram below. After the weight taps are trained by the training signal, the equalizer is ready to use. After the input signal passed through the equalizer the output signal will then be mapped to constellation points in order to restore the original signal. The block diagram of first iteration of LMS is shown in Figure 5.

3. Software Implementation

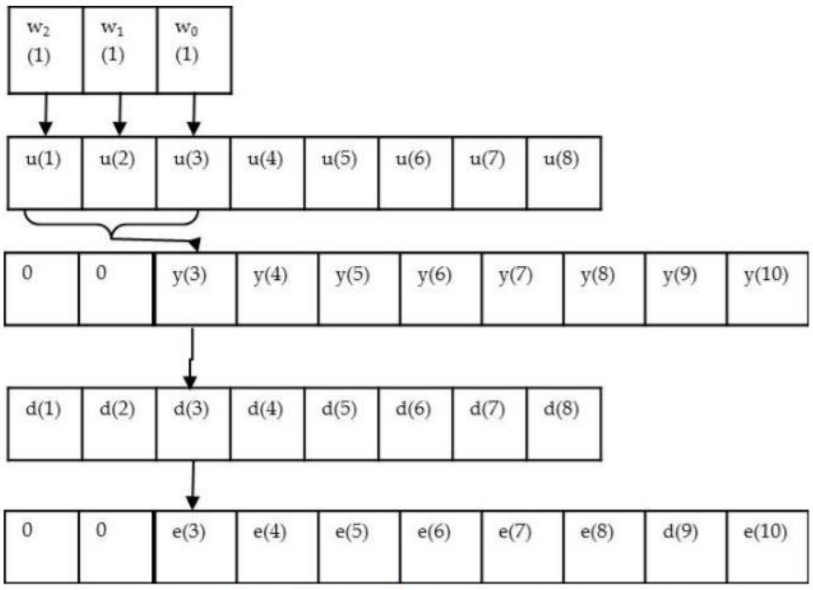


Figure 5. LMS Block Diagram First Iteration

$$\begin{bmatrix} w_0(2) \\ w_1(2) \\ w_2(2) \end{bmatrix} = \begin{bmatrix} w_0(1) \\ w_1(1) \\ w_2(1) \end{bmatrix} + \mu e(3) \begin{bmatrix} u(3) \\ u(2) \\ u(1) \end{bmatrix}$$

Figure 5. LMS Block Diagram First Iteration

3.2 Implementing Signed LMS in GNU Radio

Signed LMS is a variant of LMS that is aimed at speed up the computation. Instead of updating the weight factor based on the magnitude of the error, only the sign of the error is taken.

```
e_sign = np.sign(e.real)+np.sign(e.imag)i
```

In order to optimize the performance of the Signed LMS GNU Radio function block, the function block will round the step size to the nearest number that is the multiple of 2 implicitly in the case that the step size entered by user is not the multiple of 2. The function block will round the step size based on the formula: $\#\ Bit\ shift\ = Round(\frac{1}{Step\ size})$, then perform binary shifting.

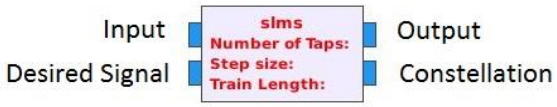


Figure 6. The Signed LMS function block (GNU Radio)

Figure 6 is the screenshot of the GNU function block of Signed LMS implemented. The top-left pin accepts the input signal to be equalized. The bottom-left pin accepts the desired signal to train the weight factor of the Signed LMS equalizer. The top-right pin outputs the equalized

signal and the bottom-right pin outputs the output signal that mapped to constellation point. The length of weight factor (Number of Taps), step size and train length can be specified by click on the block and then fill the corresponding field.

3.3 Implementing Normalized LMS in GNU Radio

For the Normalized LMS, if the length of the input vector is too large, then the denominator of the weight factor update function will dominate and cause the weight factor stick with the same value no matter how big the error is. There is a minor modification made to the updating function to fix this problem neatly. The term ||u(n)||^2 is divided by the length of weight factor in order to better accommodate to different setting of the length of the weight factors. The update function now turns out to $w(n+1) = w(n) + \frac{(\boldsymbol{step\ size})\,\boldsymbol{u}^*(\boldsymbol{n})\boldsymbol{e}(\boldsymbol{n})}{||\boldsymbol{u}(\boldsymbol{n})||^2/\boldsymbol{len}(\boldsymbol{w}) + \boldsymbol{Bias}}$. The step size will now remain stable for lengthy input vector.

```
self.w+=self.alpha*(xt.conjugate()*ec)/((xt.T.conjugate().dot(xt))/lenw+self.a)
```

Figure 7. The Normalized LMS function block (GNU Radio)

Figure 7 is the screenshot of the GNU function block of Normalized LMS implemented. All the four pins function exactly same as the Signed LMS function block illustrated above. The Normalized LMS equalizer has one more hyper parameter to tune compared to Signed LMS which is the bias. Users can tune the bias based on the characteristics of input vector to optimize the performance of the equalizer.

3.4 Implementing Variable-Step LMS in GNU Radio

For the Variable-Step function block of GNU Radio, the step size is updated based on the formula: $\mu(\boldsymbol{n}+\boldsymbol{1}) = \boldsymbol{\mu}(\boldsymbol{n}) + (\boldsymbol{IncStep})\boldsymbol{Real}(\boldsymbol{g} * \boldsymbol{g}_{\boldsymbol{prev}})$. *IncStep* denotes an increment step, *Real* denotes taking the real part of the input, $\boldsymbol{g} = \boldsymbol{u}(\boldsymbol{n})\,\boldsymbol{e}^*(\boldsymbol{n})$ and gprev denotes the value of *g* in the previous iteration. The Real(g*gprev) maps the magnitude of error to step size. There will be a large step size in the case that error is large and smaller step size while the error gets smaller.

```
g = xt.dot(ec.conjugate())
self.alpha+=self.incstep*np.real(g.T.dot(gprev))
```

In order to handle the edge case of overflow of the step size value, the maximum step size and minimum step size is set beforehand to refrain the step size from getting too large. If the step size is greater than the predefined maximum step size, the step size is set as maximum step size.

Similarly, if the step size computed is less than minimum step size, the step size is set as minimum step size.

```
if self.alpha>self.maxstep:
    self.alpha=self.maxstep
elif self.alpha<self.minstep:
    self.alpha=self.minstep
```

The update function maps the magnitude of error to magnitude of step size to fasten the process of training weight factor without sacrificing the steady-state error.

Figure 8. The Variable Step LMS function block(GNU Radio)

Figure 8 is the screenshot of the GNU function block of Variable Step LMS implemented. All four pins function exactly the same as the Normalized LMS and Signed LMS. There are four more hyper parameters to tune in Variable-Step LMS function block: the initial step, the incremental step, the minimum step and the maximum step.

3.5 Implementing RLS in GNU Radio

For the RLS function block of GNU Radio implemented, the inverse correlation matrix will be initialized at first by setting the value of diagonal elements to the value user specified.

```
self.P = np.identity(N,dtype=np.cfloat)*InvCorrInit
```

All the matrixes are initialized and their dimensions are confined to make the utilization of resources more efficient. The Kalman gain vector and the inverse correlation matrix are computed alternatively throughout the training process. The Kalman gain vector with dimension N by 1 is computed based on the formula $K = \frac{Pu}{ForgetFactor+u^H Pu}$, where the forget factor is used to reduce the weight of older data, the superscript H denotes for Hermitian Transpose and P denotes for inverse correlation matrix with dimension of N by N. After that, the inverse correlation matrix will be updated based on the formula $P = \frac{P - Ku^H P}{ForgetFactor}$.

Eventually, the weight factor will updated based on the formula: $w(n+1) = w(n) + K^* e(n)$, where the * denotes for complex conjugate.

```
self.w+=K.conjugate()*ec
```

Figure 9. The Recursive Least Square function block (GNU Radio)

The Figure 9 is the screenshot of the GNU function block of RLS implemented. All four pins function exactly same as all the function blocks mentioned above. However, the hyper parameters of RLS are very different to the function block mentioned above. The forget factor and the initial coefficient need to be specified by click on the block and fill in corresponding field before running the function block.

4. Performance Assessment

4.1 TestBench for the Equalizers

QPSK (Quadrature Phase-Shift Keying) modulation scheme is deployed in assessment. QPSK uses 4 points in the constellation diagram, which means it can transmit two bits per symbol (which takes value from 0 to 3). The data stream with 100 thousand symbols that is randomly generated is used to make the result of test more generalizable. The distortion caused by multi-path is simulated in different channel models. (802.11a, 802.11b, 802.11g and a FIR filter with 512 taps.) The noise in wireless transmission is characterized by gaussian noise with mean of zero and standard deviation of 0.01. The different combinations of equalizer models and adaptive algorithms are used to equalize the distorted signal. Eventually, the equalized signal is demodulated and then compared with the original data packet. The performance of the equalizer is evaluated based on the symbol error rate of the equalized signal. The symbol error rate (SER) is computed by the formula:$SER = \frac{\textbf{\textit{Number of symbols with errors}}}{\textbf{\textit{Total number of the symbol}}}$. The test for each combination is repeated for 20 times to ensure the generalizability of the result. The test of SER for Decision Feedback Equalizers is conducted within the equalizer object in Communications System Toolbox of the Matlab, while the SER of Linear equalizers is conducted within self-implemented GNU Radio function blocks.

4.2 Symbol Error Rate of the Equalizers

Table 2. Symbol error rate(SER) of equalizers. Where SER = Number of symbol with errors / Total Number of the symbol

	Signed LMS	Normalized LMS	Variable-Step LMS	RLS
Linear Equalizer	3.9%	1.3%	1.0%	0.2%
Decision Feedback Equalizer	3%	0%	0%	0%

From Table II, it is apparent that the decision feedback equalizers give lower steady-state error than linear equalizers. This coincides with the initial expectation, since the feedback filter reduces the noise in the incoming signal, the overall performance of the equalizer benefits from the lower noise level. For the adaptive algorithms, the RLS gives the lowest steady-state error under the linear equalization model. The performance of Normalized LMS and Variable-Step LMS almost tied. For the decision feedback equalizer, Normalized LMS, Variable-Step Least Mean Square and Recursive Least Square all give zero steady-state error rate.

4.3 Convergence Time of the Equalizers

4. Performance Assessment

The convergence time is the time it takes for the error of the system to converge. As the data stream is transmitted at a constant speed, the time to converge is proportional to the number of symbols transmitted. Convergence time is measured by the number of symbols it takes for the error to converge. The criteria of convergence in this test is the timestamp that the average error for 20 symbols is below 0.2. The test of convergence time is conducted within self-implemented GNU Radio function blocks.

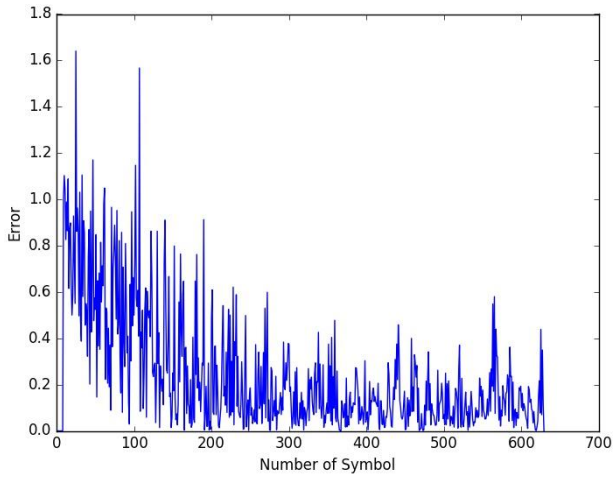


Figure 10. The plot of Error vs. Number of Symbols of Signed LMS

As Figure 10 shows, the error rate of Signed LMS converged after 231 symbols. The plot also looks “noisy”, because Signed LMS updates the weight factor at exactly the same stride for every iteration. The stride only depends on predefined step size instead of the magnitude of error. Thus, it turns out that the Signed LMS cannot make a more precise adjustment on weight factor once the error gets smaller.

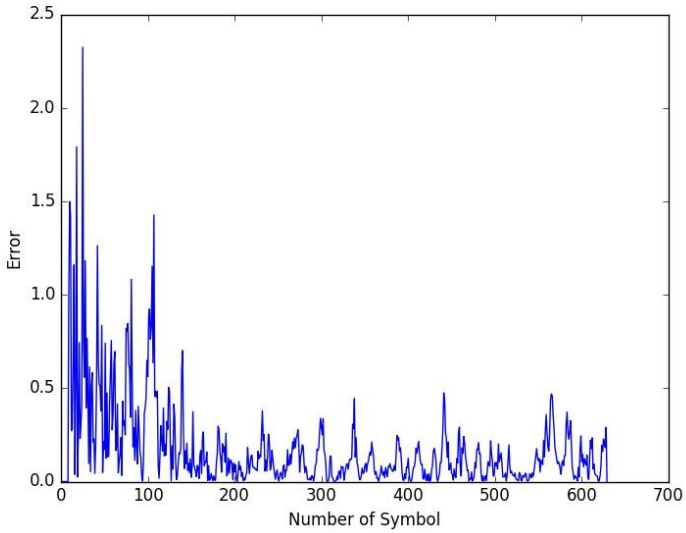


Figure 11. The plot of Error vs. Number of Symbols of Normalized LMS

4. Performance Assessment

As shown in Figure 11, the error rate of Normalized LMS converged after 147 symbols. It is apparently faster than Signed LMS, since the weight factor was updated adaptively based on the magnitude of error. The plot also looks much smoother compared to Signed LMS, since the Normalized LMS compensates for the fluctuation in magnitude of input vector by changing the step size accordingly.

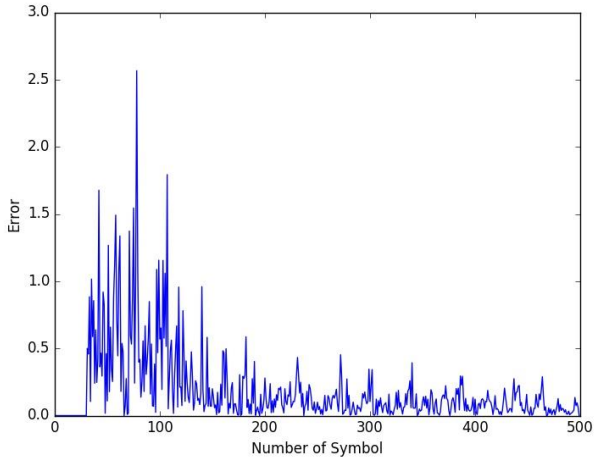


Figure 12. The plot of Error vs. Number of Symbols of Variable-Step LMS

As shown in Figure 12, the error rate of Variable-Step LMS converged after 146 symbols. Benefiting from the precise adjustment of step size based on the error, the variable-step LMS achieves lower steady-state error within shorter convergence time compared to signed LMS. The variable-step LMS picks relatively large step size at beginning in order to get closer to optimal point at shorter time which lead to relative large fluctuation in magnitude of error at beginning.

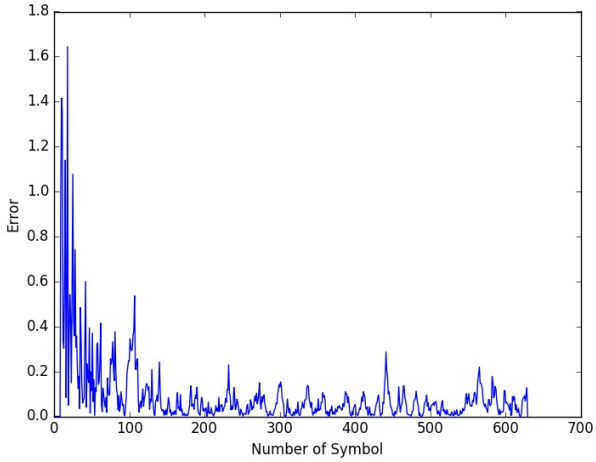


Figure 13. The plot of Error vs. Number of Symbols of Recursive Least Square

4. Performance Assessment

As shown in Figure 13, the error rate of Recursive Least Square converged after 101 symbols. As the most complex model among the models in this performance assessment, the RLS gives out lowest steady-state error. Furthermore, the error rate drops down drastically right after a few iterations which is much faster compared to the other models.

The convergence time of adaptive algorithms under different channel are summarized in Table III and visualized in Figure 14.

Table 3. Table of Convergence time within adaptive algorithms

	Signed LMS	Normalized LMS	Variable-Step LMS	RLS
802.11a	417	168	146	101
802.11b	111	48	103	101
802.11g	148	98	124	124
FIR with 512 taps	231	147	146	101

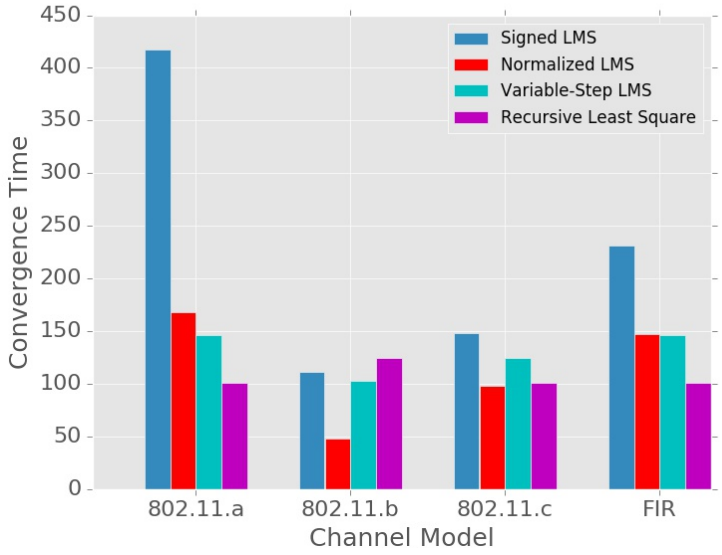


Figure 14. Convergence time of adaptive algorithms

4.4 Time Complexity of the Equalizers

The computation time of Signed LMS grows linearly with respect to the number of weights taps (***O(N)***). Since weight factors of Signed LMS can be computed by binary shift

instead of floating point computation, its constant coefficient is much smaller than that of the other three algorithms which should give it remarkably short running time in theory.

The computation time of Normalized LMS also stays within linear time (***O(N)***).However, its computation process incorporates floating-point computation and matrix multiplication that turns out its constant coefficient to be greater than that of Signed LMS.

Same as Signed LMS and Normalized LMS, Variable-Step LMS's computation time is within linear time(***O(N)***). The computation complexity of Varaible-Step LMS is basically the same as the Normalized LMS except the threshold check takes extra-time to execute. So its constant coefficient is slightly greater than that of Normalized LMS.

RLS's time complexity is proportional to the square of the number of weight taps which is the highest among the four adaptive algorithms(***O(N^2)***).

The actual runtime test is conducted within GNU Radio function block within 1500 symbols input datastream while set the number of weight taps to 100.

The runtime analysis and actual runtime of adaptive algorithms are summarized in table IV.

Table 4. Table of running time of adaptive algorithms

	Signed LMS	Normalized LMS	Variable-Step LMS	RLS
Time Complexity Analysis	***O(N)***	***O(N)***	***O(N)***	***O(N^2)***
Actual Running Time	0.01557s	0.01559s	0.01560s	0.468s

In the actual runtime measurement, the Signed LMS did not significantly reduce the runtime as expected. There are several reasons for this. First of all, there are multiple steps for computation while updating weight factor is only the last step, it turns out the binary shift does not have much influence on the overall runtime. Secondly, Python is a weakly typed programming language; this means that extra time must be spent on checking the data type of the operand; in turn, this reduces the weight of the time saved by binary shift reflects on the total runtime. As what is expected, the RLS takes much longer time (Quadratics) than that of LMS algorithms.

4.4 Space Complexity of the Equalizers

"Space complexity" describes computer resources allocated to the function (ex. register, memory space). The space complexity of Signed LMS and Normalized LMS are linearly proportional to the length of weight taps(***O(N)***). Since the all the variables take space at most of

N (ex. Input vector). The Variable-Step LMS still takes linear space(***O(N)***). However, there are more variables, for instance g (which maps error to step size) and gprev (g in previous iteration). The constant coefficient of Variable-Step LMS is larger than that of Signed LMS and Normalized LMS. Unlike the three variant of LMS adaptive algorithms, the RLS takes quadratic space(***O(N^2)***). For instance, the inverse correlation matrix has dimension of N by N, the Kalman gain vector also takes space of N. Sum up the space to store all the variables, the space complexity of RLS is much larger than the other three adaptive algorithms.

The space complexity of adaptive algorithms are summarized in table V.

Table 5. Table of space Complexity of adaptive algorithms

Signed LMS	Normalized LMS	Variable-Step LMS	RLS
O(N)	*O(N)*	*O(N)*	*O(N^2)*

5. Neural Network

5.1 Introduction to Neural Network Design

The linear equalizer is not efficacious for severely distorted channel or nonlinear channel. This problem can be addressed by replacing the linear filter by neural networks, considering the nonlinear processing capacity of neural networks. Three-layer perceptron is proposed to balance the performance and computation time. There are two configurations for modifying decision feedback equalizer: (1) the regular decision feedback equalizer is modified by replacing feedforward filter with neural network while retaining the linear feedback filter; (2) the regular decision feedback equalizer is modified by replacing both feedforward and feedback filters with neural networks.

5.2 Structure of Neural Network

As shown in Figure 15, three-layer perceptron is composed of an input layer, a hidden layer and an output layer. The hidden layer is composed of a linear combiner and an activation function. The output of the previous layer forms the input of next layer. The number of input node is set to 6. The hidden layer has exactly same numbers of neurons as the input layer. There is two node in output layer, one for real part and one for imaginary part.

$$y = \sum_{j=1}^{N} W_j x_j + b,$$
$$X = f(y)$$

The linear combiner and activation function

The loss function is $loss = (prediction - y)^2$, where y is the label or desired output. The neural networks is trained by the gradient descent optimizer.

5. Neural Network

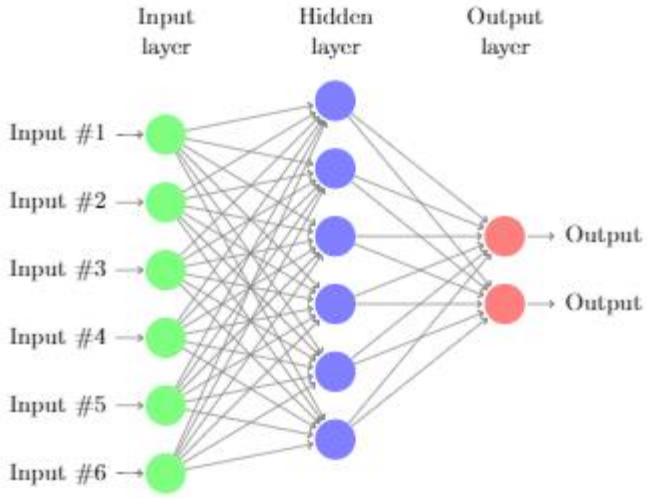


Figure 15. The structure of neural network

5.3 Preprocessing of the Input Signal

Since the deep learning frameworks (ex. TensorFlow) do not support complex computation, special preprocessing is needed to address this problem. The N by 1 complex input vector is separated into two arrays (one for real part and the other for imaginary part). Then the imaginary array is appended to end of real part array to form 2N by 1 vector.

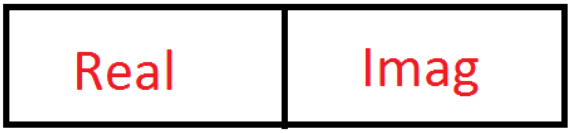


Figure 16. The format of input vector

5.4 Convergence Time of Neural Network

The testBench is same as the before except the signal is modulated by BPSK modulation scheme. The criteria of convergence (the timestamp that the average error for 20 symbols is below 0.2) is exactly same as that of linear equalizer mentioned above. The most common activation functions for neural network are examined based on their convergence time.

5.4.1 Perfomance of Neural Net with Rectifer (Relu) Activation function

Relu (x) = max(0,x). The Relu activation function ignores all the negative data points and retains only positive data points. The rectifier function can be illustrated by the graph below.

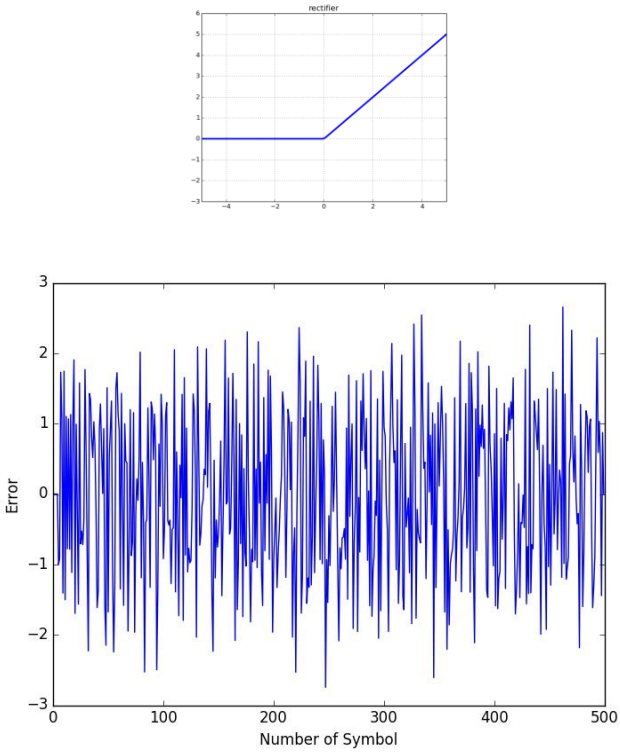
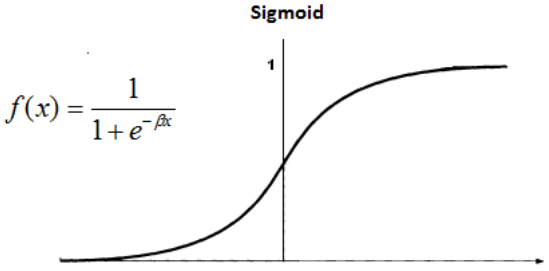


Figure 17. The plot of Error vs. Number of Symbols of Neural net with Relu activation function

Both neural networks with configuration (1) (replacing feedforward filter while retaining linear feedback) and configuration (2) (replacing both feedforward filter and feedback filter) with Relu activation function fails to converge.

5.4.2 Performance of Neural Net with Sigmoid Activation function

Sigmoid Activation function is a bounded differentiable real function with "S" shaped curve.



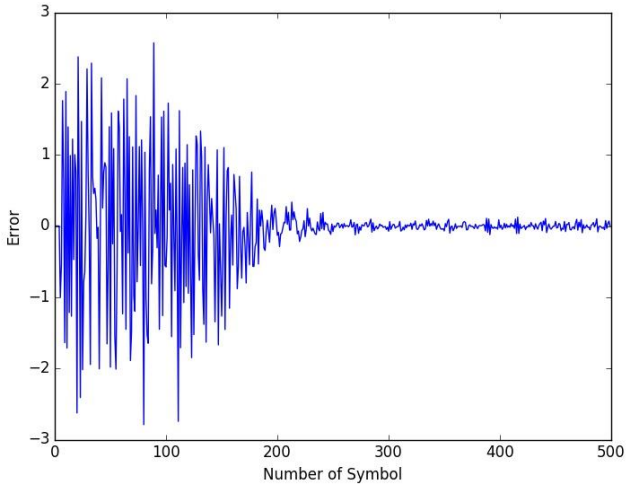


Figure 18. The plot of Error vs. Number of Symbols of Neural net of configuration (1) with sigmoid activation function

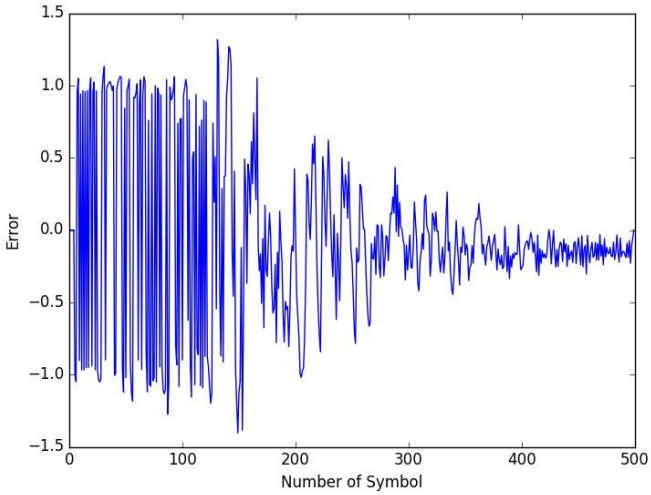


Figure 19. The plot of Error vs. Number of Symbols of Neural net of configuration (2) with sigmoid activation function

The error rate of neural network of configuration (1) with sigmoid activation function converge after 187 symbols. The error rate of neural network of configuration (2) with sigmoid activation function converge after 304 symbols. The plot looks noisy at beginning and gradually converge to zero.

5.4.3 Performance of Neural Net with Tanh Activation function

The tanh denotes for hyperbolic tangent function. It is a bounded differentiable real function.

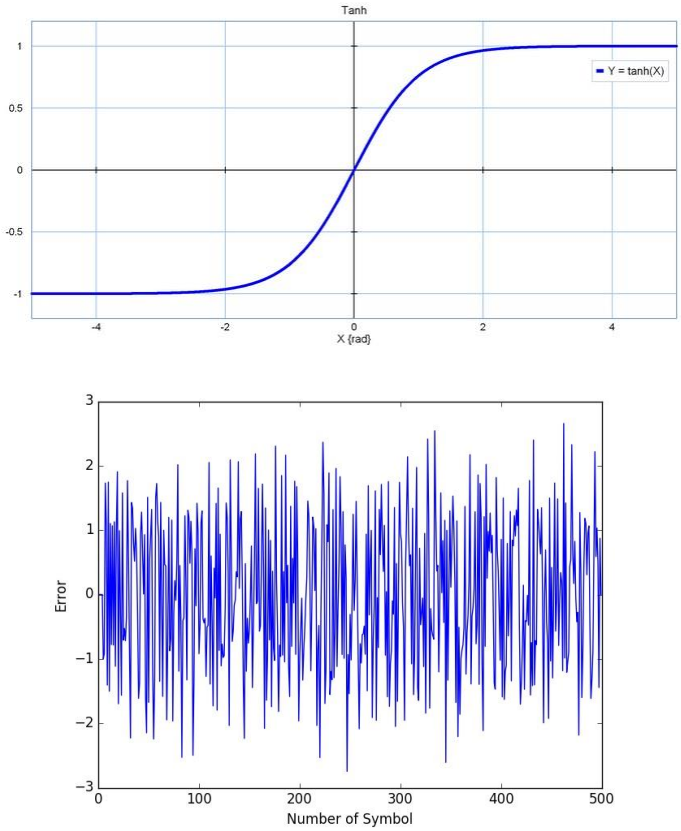


Figure 20. The plot of Error vs. Number of Symbols of Neural net with tanh activation function

Both neural networks of configuration (1) and configuration (2) with tanh activation function fails to converge.

5.4.4 Performance of Neural Net with Softplus activation function

The softplus activation function is smooth approximation of rectifier (Relu).

$$\boldsymbol{Softplus}(\boldsymbol{x}) = \boldsymbol{log}(\boldsymbol{e^x} + \mathbf{1})$$

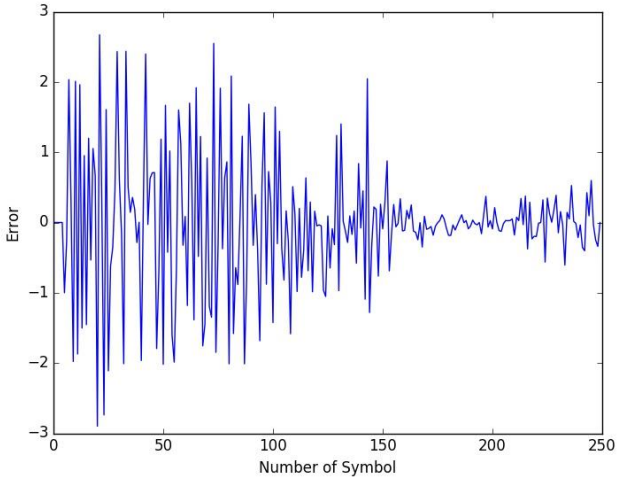


Figure 21. The plot of Error vs. Number of Symbols of Neural net of configuration (1) with softplus activation function

5. Neural Network

The neural network of configuration (1) with softplus activation function converge after 157 symbols while the neural network with configuration (2) fails to converge..

The convergence time of the common activation functions is summarized in table VI.

Table 6. Table of convergence time of activation functions

	Sigmoid	Relu	tanh	Softplus
Replace feedforward filter	187	Fails to Converge	Fails to Converge	157
Replace both feedforward and feedback filter	304	Fails to converge	Fails to converge	Fails to converge

6. Summary & Recommendation

This paper provided:

- an overview of an equalizer and four adaptive algorithms
- the software implementation of the equalizers
- the assessment for combinations of different adaptive equalization models and adaptive algorithms
- Replacing the linear filter of Decision Feedback Equalizer with neural network and performance assessment on Decision Feedback Equalizer with neural network

Equalization Models:

Equalization is introduced to reverse the distortion caused by multi-path in the signal transmission process without amplifying the noise.

- Decision feedback equalizer has a better performance over the linear equalizer insofar as it reduces the noise in the incoming signal through the feedback filter.
- However, Linear Equalizer has lower computational complexity.

Adaptive Algorithms:

Signed LMS's performance falls behind the Normalized LMS, Variable-Step Least Mean Square and Recursive Least Square algorithms. Although Signed LMS has lowest computational complexity in theory, it does not result in any notable reduction of runtime in runtime assessment. Therefore, Signed LMS is not recommended for real application in software implementation.

The Recursive Least Square outperformed the other three adaptive algorithms at the cost of remarkably higher computational complexity and space complexity(Quadratics). This implies that it is not a suitable choice for high-speed communication.

The performance of Normalized LMS and Variable-Step Least Mean Square falls slightly behind the Recursive Least Square. In the case of an incoming signal with a low noise level or used with a decision feedback equalization model, Normalized LMS and Variable-Step Least Mean Square perform as good as the Recursive Least Square with the significantly lower computational complexity than of the Recursive Least Square. Their well-balanced performance and computational complexity makes them a suitable choice for general application.

Neural Network:

6. Summary

The linear filter of equalizer is replaced by the neural networks. The structure of Neural network is composed of an input layer, a hidden layer and an output layer. The performance of common activation function is examined based on their convergence time. The Neural net of configuration (1) with sigmoid activation function takes 187 symbols to converge, while Neural net of configuration (1) with softplus function takes 157 symbols to converge. Both tanh and Relu fail to converge. For neural net of configuration (2), only sigmoid function converged after 304 symbols. The neural net of configuration (1) outperformed configuration (2) with shorter convergence time. Therefore, the configuration (1) is recommended for real world application. Considering the complexity of neural network, it takes longer time to converge compared to linear filter. Furthermore, configuration (2) takes even longer time.

Reference

[1]. GNU Radio. (2016, March 02).Tutorial: PSK Symbol Recovery. Retrieved November 11, 2016, from http://gnuradio.org/redmine/projects/gnuradio/wiki/Guided_Tutorial_PSK_Demodulation

[2]. Schober, R. (2010). Equalization of Channels with ISI. In Detection and Estimation of Signals in Noise. Vancouver.

[3]. Zhang, J. (2012, December). Variable Step Size LMS Algorithm. Retrieved March 9, 2017, from http://www.ijfcc.org/papers/104-F0012.pdf

[4]. Wasimuddin, M., & Gupta, N. (2014). Design and Implementation of Least Mean Square Adaptive Filter on Fetal Electrocardiography . Retrieved March 12, 2017, from http://www.asee.org/documents/zones/zone1/2014/Professional/PDFs/57.pdf

[5]. Mathworks, inc. (n.d.). Equalization. Retrieved March 12, 2017, from https://www.mathworks.com/help/comm/ug/equalization.html

[6]. Tabus, I. (2012). Lecture 10: Recursive Least Squares Estimation. Retrieved March 12, 2017, from http://www.cs.tut.fi/~tabus/course/ASP/LectureNew10.pdf

[7]. Mathworks. Inc. LMS Filter System. (2017). Retrieved April 3, 2017, from https://www.mathworks.com/help/dsp/ref/dsp.lmsfilter-class.html

[8]. Faragher, R. (n.d.). Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation. Retrieved May 1, 2017, from https://www.cl.cam.ac.uk/~rmf25/papers/Understanding%20the%20Basis%20of%20the%20Kalman%20Filter.pdf