# Introduction

In these functions we cancel narrowband interference using a linear time domain approaches. The first approach is based on MMSE equalization applied in the time domain. Two adaptive algorithms are implemented including Least Mean Square (LMS) and Recursive Least Square (RLS). The technique is somewhat limited in that it applied linear cancellation, but can effectively cancel interference provided that the time-domain correlation is stationary over the length of the equalizing filter. LMS is a simpler update commonly used in adaptive cancellation, because it directly uses the error signal to approximate the gradient and to calculate the updated filter coefficients. The RLS algorithm is known for having faster convergence time and good performance when working in time varying environments but at the cost of an increased computational complexity. Note that ideally, the filter length should be at least equal to the pulse shaping length.

The second approach applies adaptive notch filtering to the signal in order to eliminate narrowband interference.

## Features

* Accepts a complex vector of input samples (ideally oversampled) and outputs – if oversampling isn’t used, symbol timing could be an issue for the equalization approach. Timing isn’t an issue for the notch filter approach.
* Equalization approach assumes the use of a training sequence followed by decision directed equalization.
* Notch filter approach is both adaptive (automatically determines the notch frequency and coefficients) and blind (doesn’t require knowledge of the signal symbols).
* Equalization approach implements either RLS or LMS adaptive algorithms

# Interface Description

## Generics

The function calls for the equalization approach are:

[y,Weights] = LMS(x,Weights,delta, TrainingSequence)

[y,Weights, P] = RLS(x, Weights, P, lambda\_inv, TrainingSequence)

The function call for the notch filter approach is:

y = NotchFilter(x, r)

## Inputs

The function inputs to the Equalization approach are defined in Table 1.

Table 1: Function Inputs

|  |  |  |
| --- | --- | --- |
| Input Name | Type | Description |
|  |  |  |
| x |  | Vector of received samples – equal to the filter legnth |
| *Weights* |  | Initial filter weights. |
| *delta* |  | An update factor for LMS Algorithm which should be a small negative value. |
| *P* |  | A matrix that is used in the RLS update. It will be updated and returned every time the function is called. |
| *Lambda\_ inv* |  | An update factor for the RLS Algorithm which should be positive and close to one. |
| *TrainingSequence* |  | A training (known data) value. If in decision directed mode, the previous weights should be used to determine the current symbol estimate and the estimated symbol should be used as training for weight update. |

The function inputs to the notch filter approach are defined in Table 2.

Table 2: Function Inputs

|  |  |  |
| --- | --- | --- |
| Input Name | Type | Description |
|  |  |  |
| x |  | Vector of received samples |
| r |  | Pole radius for notch filter (should be close to one) |

## Outputs

The signal outputs from the equalization approach are defined in Table 3.

Table 3: Equalization NBI Cancellation Output Signals

|  |  |  |
| --- | --- | --- |
| Output Name | Type | Description |
| y |  | The output decision statistic determined using the input weights |
| weights |  | The updated weights using either the LMS or RLS adaptive algorithm |
| P |  | The updated P matrix used in the RLS algorithm. |
|  |  |  |

The signal outputs from the equalization approach are defined in Table 4.

Table 4: Notch Filter Output Signals

|  |  |  |
| --- | --- | --- |
| Output Name | Type | Description |
| y |  | The output signal with interference removed |

## File List

Table 5 lists the files provided with the functions.

Table 5: Time Domain NBI Cancellation Source File List

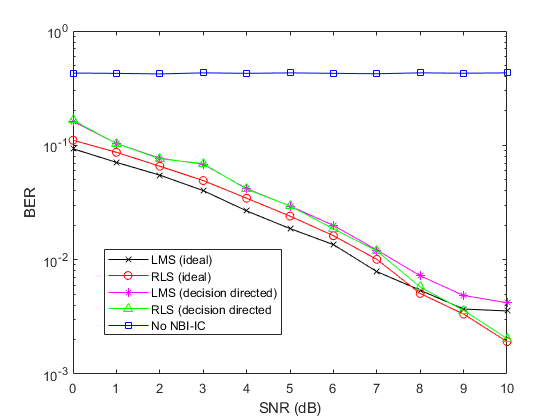
|  |  |
| --- | --- |
| File Name | Description |
| LMS.m | -This function performs a LMS update to a set of weights for MMSE-based equalization and combines the received signal using those weights to obtain the decision statistic for the current symbol. |
| RLS.m | This function performs an RLS update to a set of weights for MMSE-based equalization and combines the received signal using those weights to obtain the decision statistic for the current symbol. |
| TestNbiLinearCancel.m | This script simulates narrowband tone interference and applies the linear MMSE equalizer to cancel the interference using either an LMS adaptive update and an RLS adaptive update. The script also provides an example of how to call the functions LMS and RLS. |
| PulseShape.m | This function provides pulse shaping and is used by TestNbiLinearCancel.m in order to test the linear cancellation technique. |
| NotchFilter.m |  |
| TestNotchFilter.m | This script simulates narrowband tone interference and applies the notch filter to cancel the interference using either NotchFilter. The script also provides an example of how to call the function as well as allowing the system designer to test the function based on various interference paramters. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Functional Description – Equalization-based Cancellation

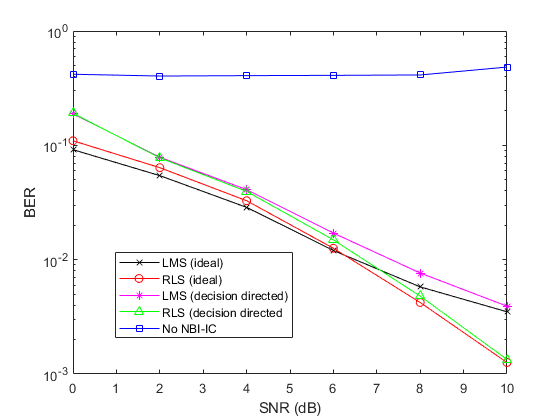
This technique is based on classic linear equalization. Specifically, linear equalization applies a weight vector to a vector of received samples in order to cancel the intersymbol (or other) interference. A common solution for choosing the weights is the MMSE solution which minimizes the mean square error between a known symbol sequence and the received vector of samples which is contaminated by ISI and other interference.

The MMSE weights are found using one of two classic adaptive algorithms – the Least Mean Squares (LMS) algorithm or the Recursive Least Squares (RLS) algorithm. The former algorithm is simpler to implement but well-known to converge slowly. The update parameter dictates the speed of convergence but also the final convergence error. Smaller values of the update parameter allow for better final performance, but slower convergence. The RLS algorithm converges significantly faster than the LMS algorithm but at the cost of increased complexity.

Example Performance:



**Figure 1 - Performance of the Time-Domain NBI Cancellation with Tone Interferer (fc = 0.01, SIR = -10dB, filter length = 10, Training Sequence Length = 10, delta = -0.0001, lambda = 0.9999, BPSK modulation)**

****

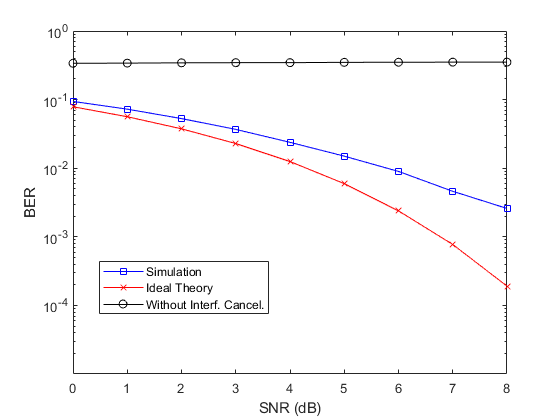
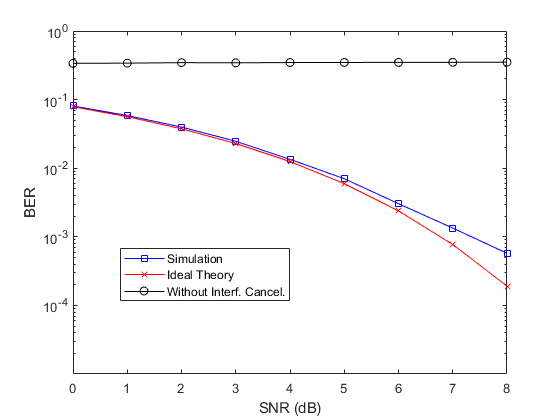
**Figure 2 - Performance of the Time-Domain NBI Cancellation with Tone Interferer (fc = 0.1, SIR = -10dB, filter length = 10, Training Sequence Length = 10, delta = -0.0001, lambda = 0.9999, BPSK modulation)**

# Functional Description – Notch Filter

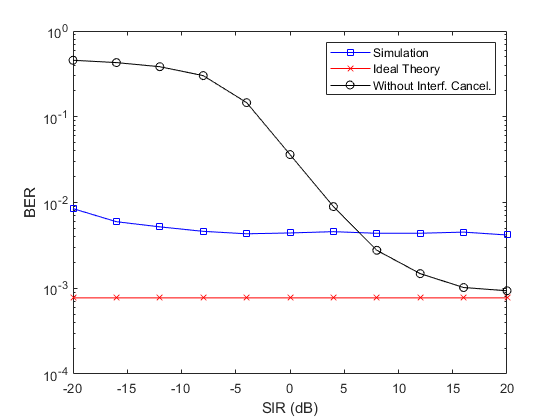
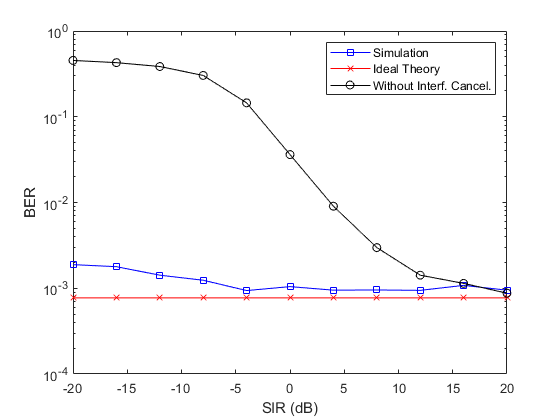
This technique is based on classic notch filtering with an adaptive technique to determine the filter coefficients. Specifically, we implement a notch filter with the following transfer function:

where *k1* and *k2* are related to the pole radius *r,* and digital notch frequency  as

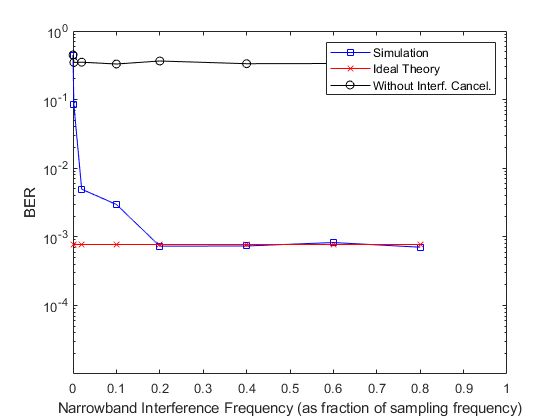
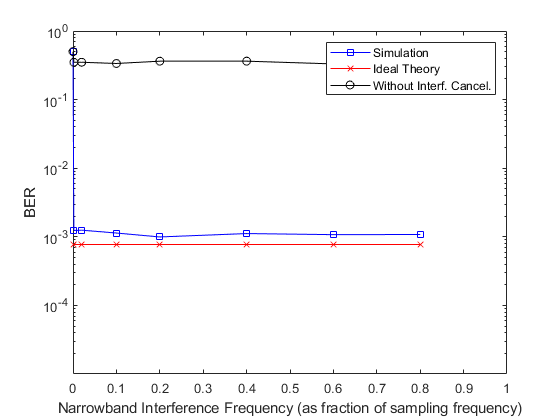
The function requires the designer to determine the pole radius *r*¸ while the functions determine the notch frequency  and the coefficients adaptively.



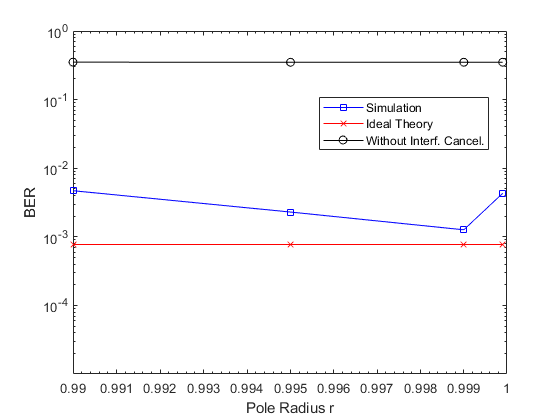
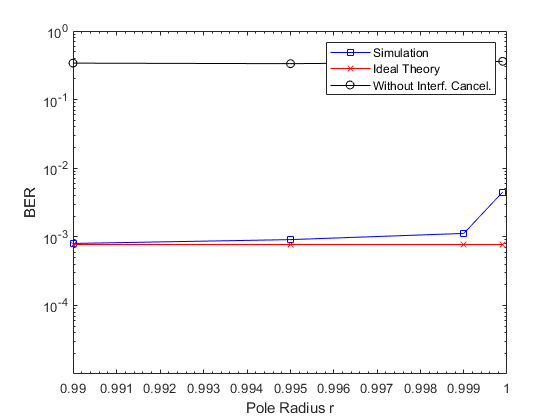
**Figure 3 – Performance of the Notch Filter Approach vs. SNR for SIR = -10dB and r = 0.999 (left) and r = 0.99 (right), using 20000 signal symbols, fint / fs = 0.0225.**

****

**Figure 4 – Performance of the Notch Filter Approach vs. SIR for SNR = 7dB and r = 0.999 (left) and r = 0.99 (right), using 20000 signal symbols, fint / fs = 0.0225.**

****

**Figure 5 – Performance of the Notch Filter Approach vs. Interference Frequency for SNR = 7dB, SIR = -10dB and r = 0.999 (left) and r = 0.99 (right), using 20000 signal symbols.**

**** ****

**Figure 6 – Performance of the Notch Filter Approach vs. Pole Radius for SNR = 7dB, SIR = -10dB, using 20000 signal symbols, Interference Frequency = 0.0225 (left) and 0.2 (right)**