# Introduction

In s Least Mean Square (LMS) and Recursive Least Square (RLS) are commonly used adaptive algorithms. The LMS is one of the easiest algorithms used in the adaptive noise cancellation, because it uses the error signal to calculate the filter coefficients. The RLS algorithm is known for its excellent performance when working in time varying environments but at the cost of an increased computational complexity and it also suffer with some stability problems

# Feature

* Adaptive filter
* Different complexity which means it should balance the cost and performance

## Algorithm

* Least Mean Square (LMS),
* Recursive Least Squares (RLS)
* Pulse Shaping

# Main Function

*[y, weights, p] = mmseBeamformingToolbox(x, Weights, delta, P, lambda\_inv, TrainingSequence, Algorithm)*

|  |  |
| --- | --- |
| Input |  |
| x | The received samples |
| *Weights* | Initial weights (or last weights). |
| *delta* | An update factor which should be negative for LMS Algorithm. If using other algorithm, please use 'Null' or '0' |
| *P* | a matrix that is used in the RLS update. It will be updated and returned every time the function is called. If using other algorithm, please use 'Null' or '0. |
| *Lambda\_ inv* | An update factor for RLS Algorithm which should be positive. If using other algorithm, please use 'Null' or '0' |
| *TrainingSequence* | A 1 x Nt vector of training (known data) values |
| *Algorithm* | ‘LMS’ and ‘RLS’ |

|  |  |
| --- | --- |
| Output |  |
| *y* | A 1 x Nb output vector of data values after combining |
| *weight* | The Nr x 1 vector of Weight values after processing the data and updating the weights. |
| *p* | The updated P matrix for RLS |

# Module

* LMS

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

|  |  |
| --- | --- |
| Input |  |
| x | The received samples |
| *Weights* | Initial weights (or last weights). |
| *delta* | An update factor which should be negative for LMS Algorithm. If using other algorithm, please use 'Null' or '0' |
| *TrainingSequence* | A 1 x Nt vector of training (known data) values |

* RLS

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

|  |  |
| --- | --- |
| Input |  |
| x | The received samples |
| *Weights* | Initial weights (or last weights). |
| *P* | a matrix that is used in the RLS update. It will be updated and returned every time the function is called. If using other algorithm, please use 'Null' or '0. |
| *Lambda\_ inv* | An update factor for RLS Algorithm which should be positive. If using other algorithm, please use 'Null' or '0' |
| *TrainingSequence* | A 1 x Nt vector of training (known data) values |

* PulseShape

*[y, pulse, Es] = PulseShape( x, PulseShape, Ns, N, roll\_off)*

|  |  |
| --- | --- |
| Input |  |
| x | The transmitted signal |
| *PulseShape* | 'SQAR', 'SINC', 'SRRC', 'RaCo' |
| *Ns* | The number of samples per pulse |
| *N* | The number of pulse durations when truncation is necessary. |
| *roll\_off* | The roll-off factor for raised cosine or square root raised cosine filters |

# Simulation

