# Adaline Network

```
% ADALINE with Bipolar Activation Function
clc;
clear;
% Input and target output
X = [-1, -1, 1, 1; -1, 1, -1, 1]; % Input matrix (2x4)
T = [-1, 1, 1, 1];              % Target output
% Parameters
[features, samples] = size(X);
learning_rate = 0.01;           % Learning rate
epochs = 100;                   % Number of iterations
bias = 1;                       % Bias term
W = rand(1, features);          % Initialize random weights
% Add bias to input
X = [ones(1, samples); X];      % Augment input matrix with bias term
W = [rand(1), W];               % Initialize bias weight
% Training loop
for epoch = 1:epochs
   for i = 1:samples
      % Compute the net input (weighted sum)
      net_input = W * X(:, i);
      % Bipolar activation function (hard limit function)
      if net_input >= 0
         output = 1;
      else
         output = -1;
      end
```

```
        % Calculate error
        error = T(i) - output
        % Update weights using LMS rule (delta rule)
        W = W + learning_rate * error * X(:, i)';
    end
    % Display weights for each epoch (optional)
    fprintf('Epoch %d: Weights: %s\n', epoch, mat2str(W));
end
% Final weights
disp('Final weights:');
disp(W);
% Testing the network on the same inputs
for i = 1:samples
    net_input = W * X(:, i);
    % Bipolar activation function
    if net_input >= 0
        output = 1;
    else
        output = -1;
    end
    fprintf('Input: %s -> Predicted Output: %d\n', mat2str(X(2:end, i)), output);
end
```