```
% MATLAB Program for OR function using Adaline

% Bipolar inputs and target for OR function
X = [-1 -1; -1 1; 1 -1; 1 1];  % Input matrix (4x2) for X1 and X2
Y = [-1; 1; 1; 1];             % Target output vector (4x1)

% Parameters
[num_samples, num_features] = size(X);  % Number of training samples and features
learning_rate = 0.1;                    % Learning rate
epochs = 100;                           % Number of iterations/epochs

% Initialize weights and bias
weights = rand(1, num_features);        % Weights for inputs (randomly initialized)
bias = rand();                          % Bias (randomly initialized)

% Training process
for epoch = 1:epochs
    for i = 1:num_samples
        % Calculate net input
        net_input = sum(weights .* X(i, :)) + bias;

        % Activation function (Linear for Adaline)
        output = net_input;

        % Calculate error
        error = Y(i) - output;

        % Update weights and bias using Adaline rule
        weights = weights + learning_rate * error * X(i, :);
        bias = bias + learning_rate * error;
    end

    % Optionally, calculate the sum squared error (SSE) for monitoring
    SSE = sum((Y - (X * weights' + bias)).^2);
    fprintf('Epoch: %d, SSE: %.4f\n', epoch, SSE);

    % Stop training if error is sufficiently small (optional)
    if SSE < 0.01
        break;
    end
end

% Final weights and bias
fprintf('Final weights: %.4f %.4f\n', weights);
fprintf('Final bias: %.4f\n', bias);

% Testing the Adaline on the OR function
for i = 1:num_samples
    net_input = sum(weights .* X(i, :)) + bias;
    output = net_input;

    % Since this is Adaline, we'll use the sign function for bipolar outputs
    if output >= 0
        predicted_output = 1;
    else
        predicted_output = -1;
    end

    fprintf('Input: [%d %d], Predicted Output: %d, Expected Output: %d\n', X(i,1), X(i,2), predicted_output, Y(i));
end
```