```
% Initialize parameters
clc;
clear;

% Define hyperparameters
learning_rate = 0.5; % Learning rate
N = 2000; % Maximum number of iterations
soft_high = 1; % Softened value for 1
soft_low = 0;  % Softened value for 0

input_patterns = [soft_high, soft_low, soft_low, soft_low;
                  soft_low, soft_high, soft_low, soft_low;
                  soft_low, soft_low, soft_high, soft_low;
                  soft_low, soft_low, soft_low, soft_high];
targets = input_patterns;


% Initialize weights with bias terms
W1 = rand(5, 2) - 0.5; % Input-to-hidden weights (4 inputs + 1 bias)
W2 = rand(3, 4) - 0.5; % Hidden-to-output weights (2 hidden units + 1 bias)

% Define sigmoid activation function and its derivative
sigmoid = @(x) 1 ./ (1 + exp(-x)); % Sigmoid function
sigmoid_derivative = @(y) y .* (1 - y); % Derivative of sigmoid function


for n = 1:N
    total_error = 0;
    for d = 1:4
        % Forward propagation
        input = input_patterns(d, :)';
        target = targets(d, :)';

        % Add bias to input layer
        input_with_bias = [input; 1];

        % Compute hidden layer activation
        net_hidden = W1' * input_with_bias;
        output_hidden = sigmoid(net_hidden);

        % Add bias to hidden layer
        hidden_with_bias = [output_hidden; 1];

        % Compute output layer activation
        net_output = W2' * hidden_with_bias;
        output = sigmoid(net_output);

        % Compute error
        error = target - output;
        total_error = total_error + sum(error .^ 2);

        % Backpropagation for output layer
        delta_output = error .* sigmoid_derivative(output);

        % Backpropagation for hidden layer
        delta_hidden = (W2(1:end-1, :) * delta_output) .*
sigmoid_derivative(output_hidden);

        % Update weights for W2 (including bias weight)
        W2 = W2 + learning_rate * (hidden_with_bias * delta_output');

        % Update weights for W1 (including bias weight)
        W1 = W1 + learning_rate * (input_with_bias * delta_hidden');
    end
    errors(n) = total_error;
end

% Plot error curve
figure;
plot(1:N, errors);
xlabel('Number of Iterations');
ylabel('Total Error');
title('Error Curve with Bias Units');
```