**Assignment 3**

**Experiment 6 and 7**

Title: Learning Rules for single layer single continuous perceptron

Name of Student: Sangeet Agrawal PRN No. 21070122140

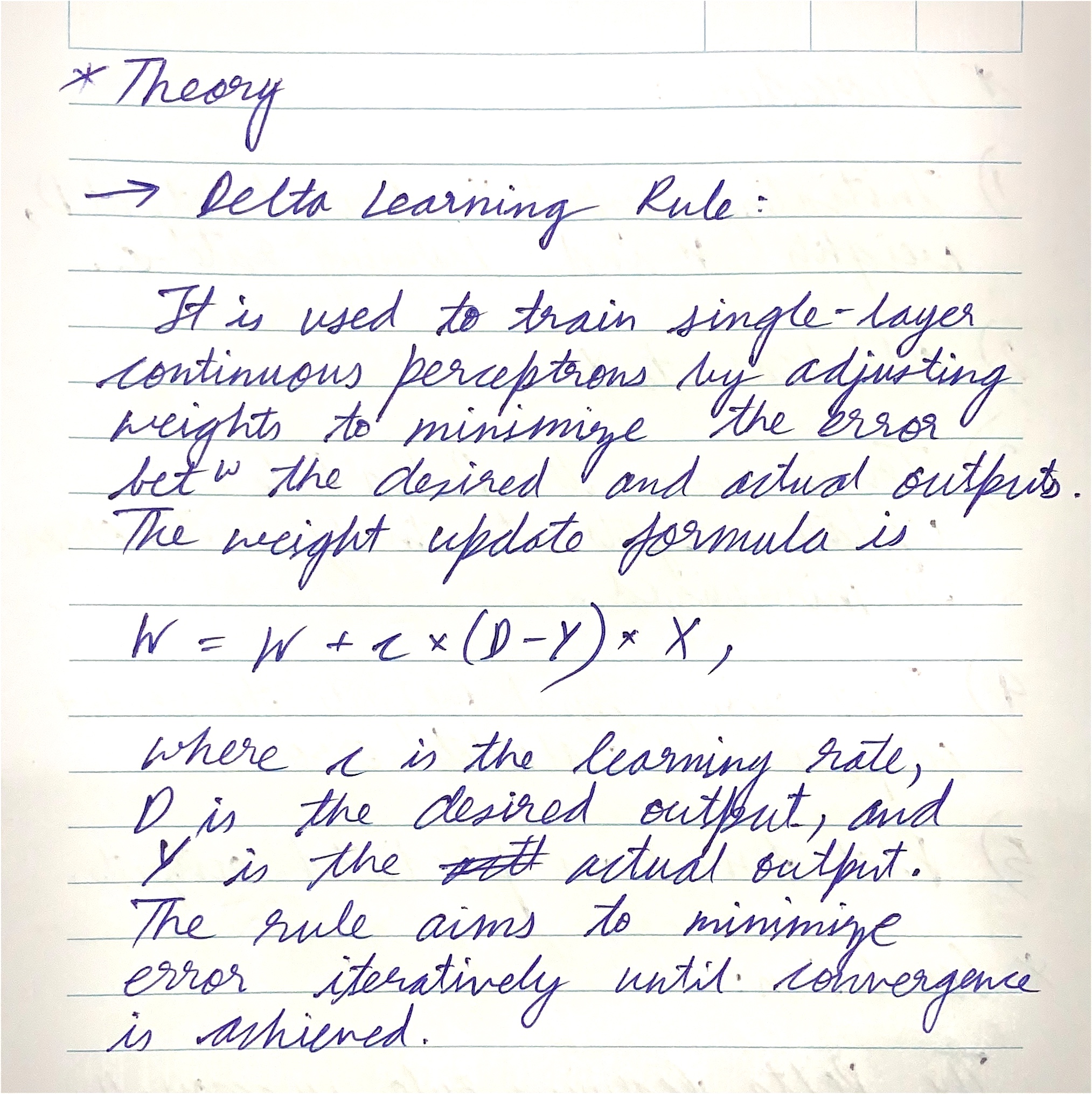
DoP6: 26 Aug

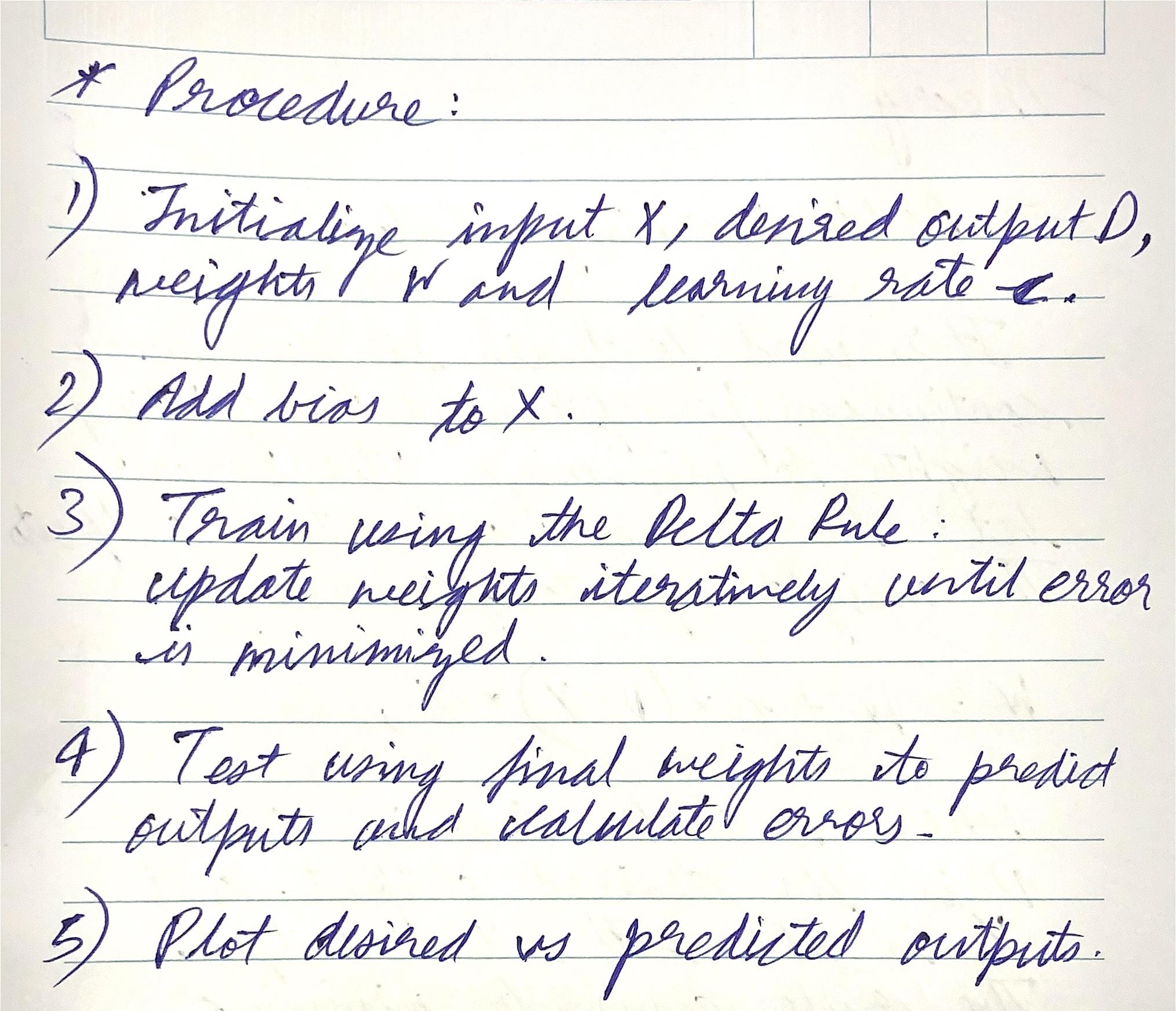
DoP7: 2 Sep DoS: 10 Sep

**Experiment 6**

**Title:**Implement Delta learning rule for single layer single continuous perceptron

**Aim:**  To design and simulate delta learning rule for single layer single continuous perceptron using MATLAB. To simulate testing (Recall) delta learning rule for single layer single continuous perceptron



****

**Input:**

X=[10 2; 2 -5; 5 5 ]

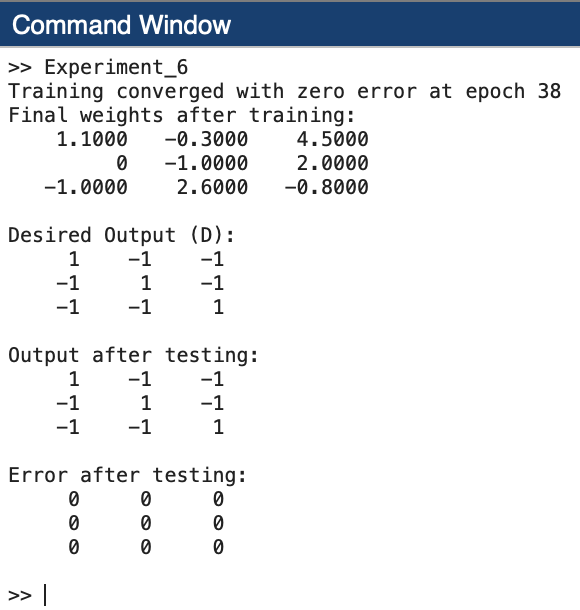
D = [1 -1 -1 ; -1 1 -1 ; -1 -1 1]

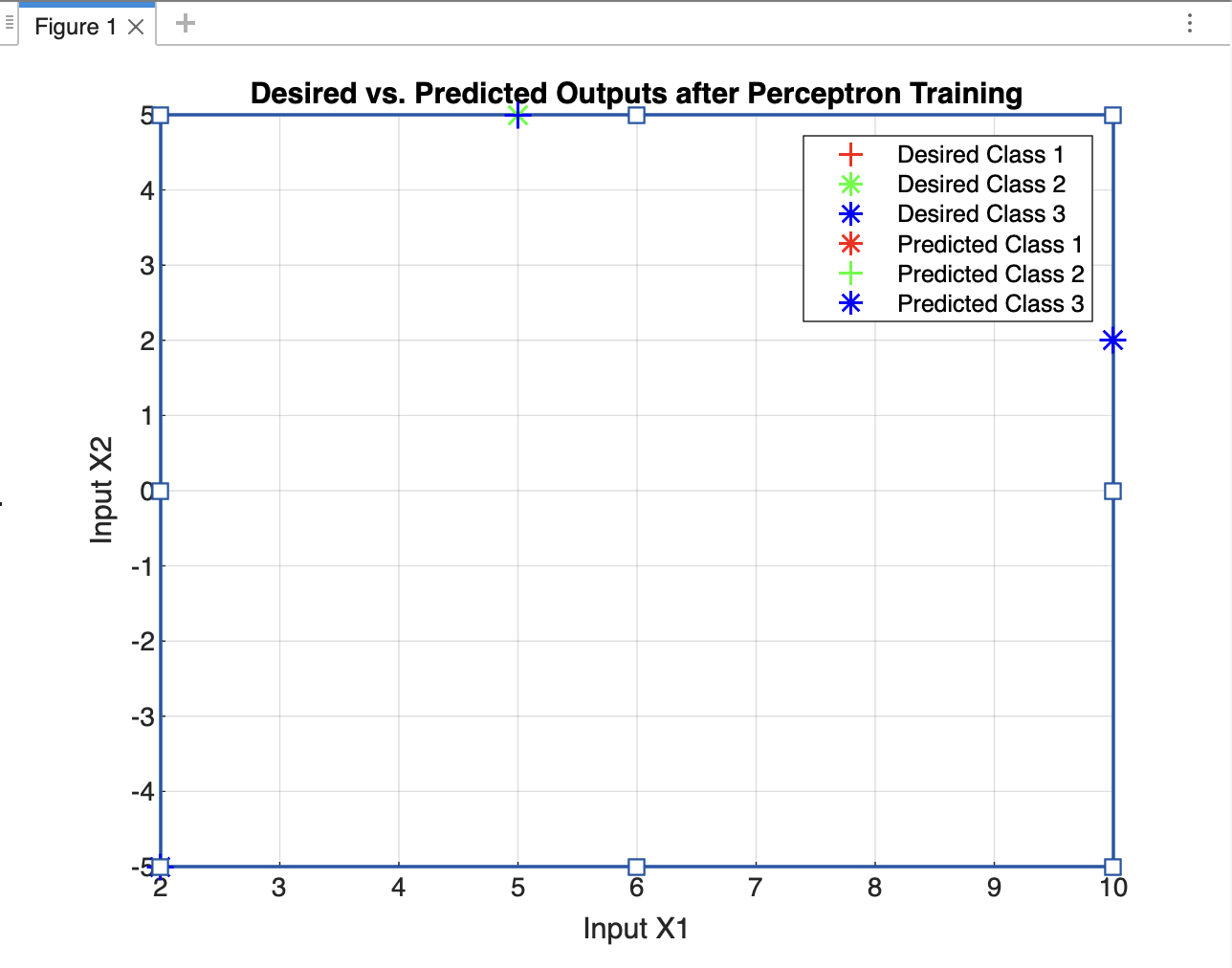
W=[1 -2 0; 0 -1 2; 1 3 -1] , c = 0.1

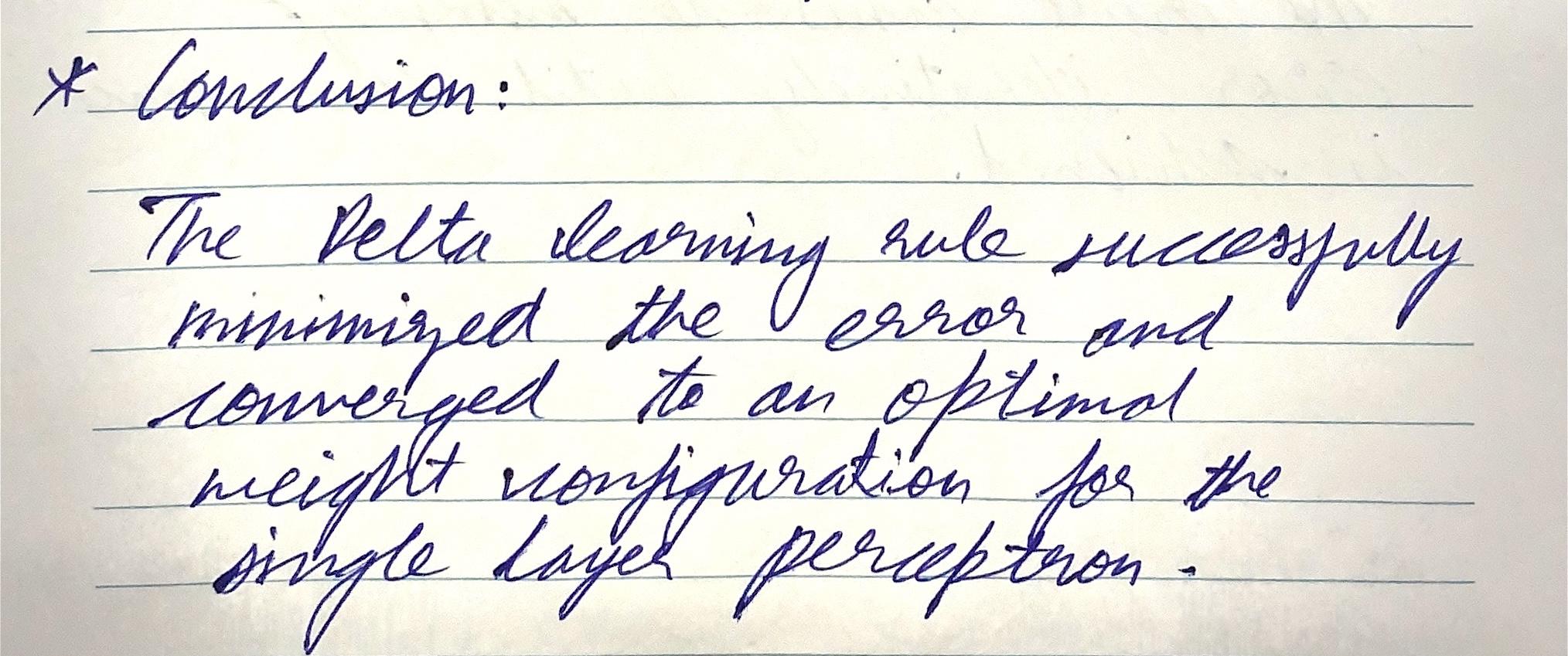
**Code:**

| X = [10 2; 2 -5; 5 5]; % Inputs D = [1 -1 -1; -1 1 -1; -1 -1 1]; % Desired outputs W = [1 -2 0; 0 -1 2; 1 3 -1]; % Initial weights c = 0.1; % Learning rate (η) % Add bias input to X X = [X, -1\*ones(size(X,1), 1)]; epochs = 38; % Training phase using Delta Learning Rule for i = 1:epochs  % Calculate the net input and apply the activation function (sign function)  Y = sign(W \* X');    % Update weights using the delta rule  W = W + c \* (D - Y) \* X;    % Calculate the total error to check for convergence  total\_error = sum(sum(abs(D - Y)));    % Stop training if error reaches zero  if total\_error == 0  fprintf('Training converged with zero error at epoch %d\n', i);  break;  end end disp('Final weights after training:'); disp(W); % Testing phase (Recall) Y\_test = sign(W \* X'); error = D - Y\_test; disp('Desired Output (D):'); disp(D); disp('Output after testing:'); disp(Y\_test); disp('Error after testing:'); disp(error); % Plotting the results figure; hold on; colors = {'r', 'g', 'b'}; for i = 1:size(X, 1)  for j = 1:size(Y\_test, 1)  if Y\_test(i, j) == 1  plot(X(i, 1), X(i, 2), '+', 'Color', colors{j}, 'MarkerSize', 10, 'LineWidth', 1); % '+' for predicted output  else  plot(X(i, 1), X(i, 2), '\*', 'Color', colors{j}, 'MarkerSize', 10, 'LineWidth', 1); % '\*' for predicted output  end  end end xlabel('Input X1'); ylabel('Input X2'); title('Desired vs. Predicted Outputs after Perceptron Training'); legend('Desired Class 1', 'Desired Class 2', 'Desired Class 3', ...  'Predicted Class 1', 'Predicted Class 2', 'Predicted Class 3'); hold off; grid on; |
| --- |

**Output:**

****

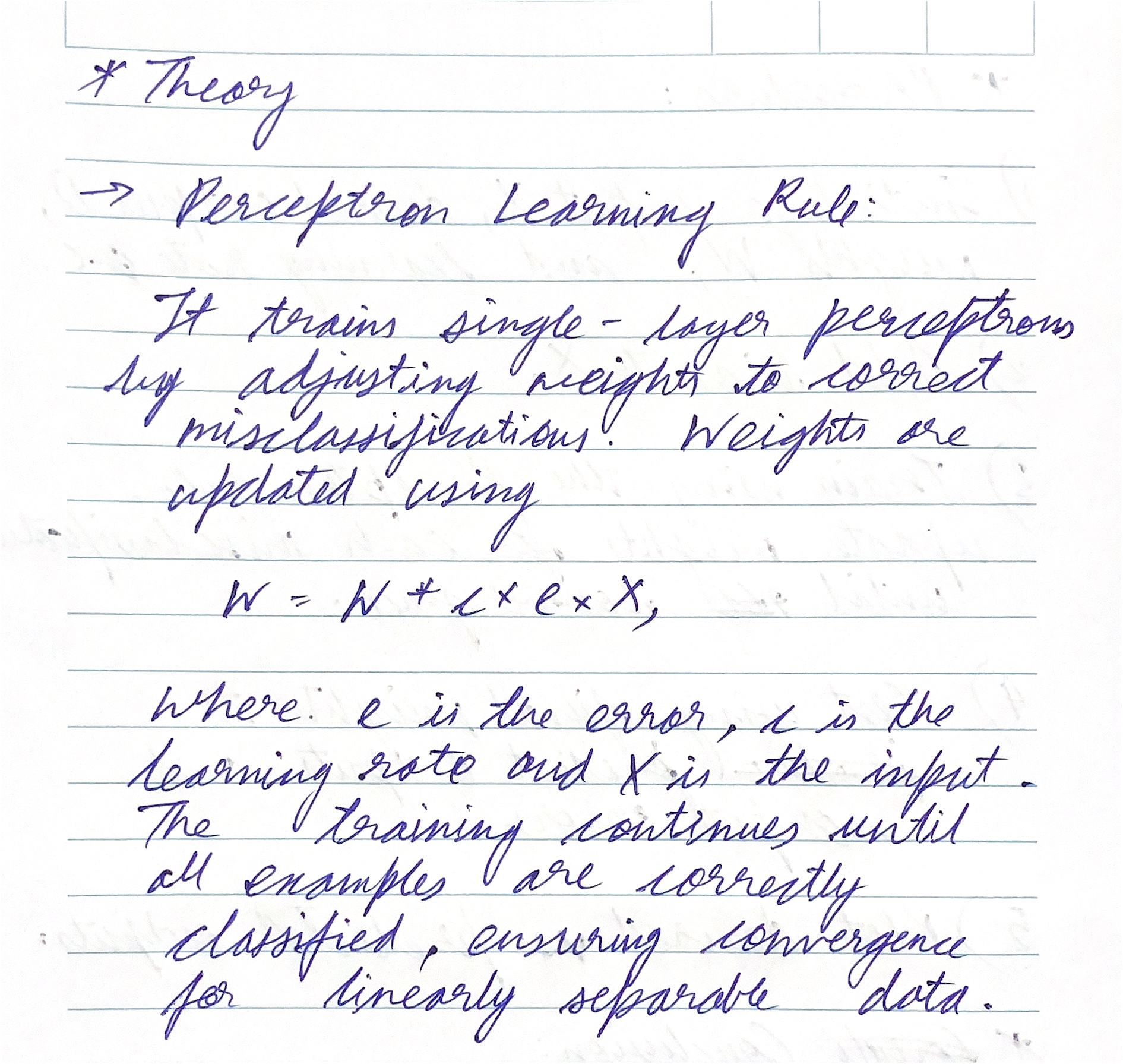
****

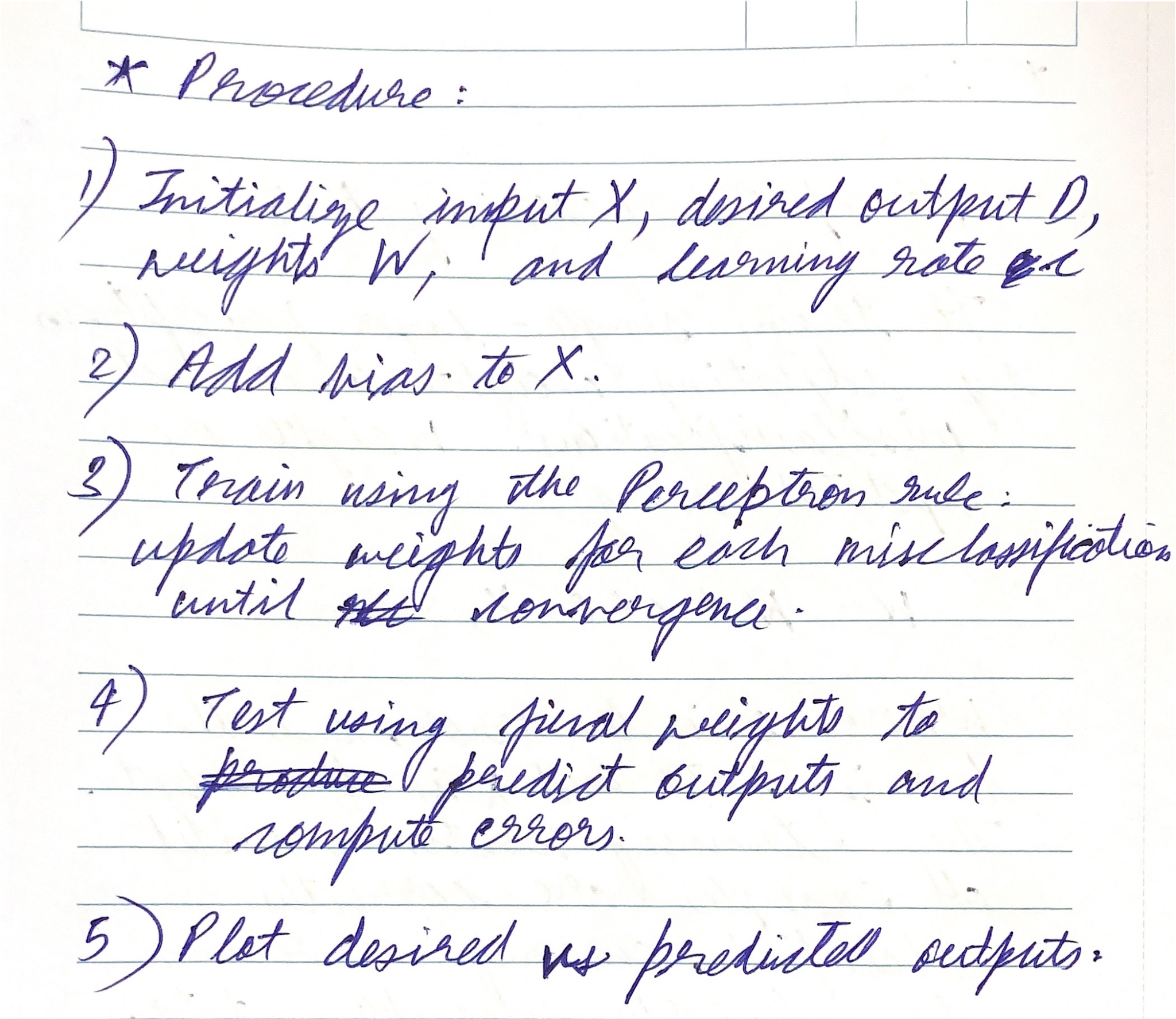
****

**Experiment 7**

**Title:** Perceptron learning rule for single layer single continuous perceptron

**Aim:** To design and simulate perceptron learning rule for single layer single continuous perceptron using MATLAB. To simulate testing (Recall) delta learning rule for single layer single continuous perceptron





**Input:**

X=[10 2; 2 -5; 5 5 ]

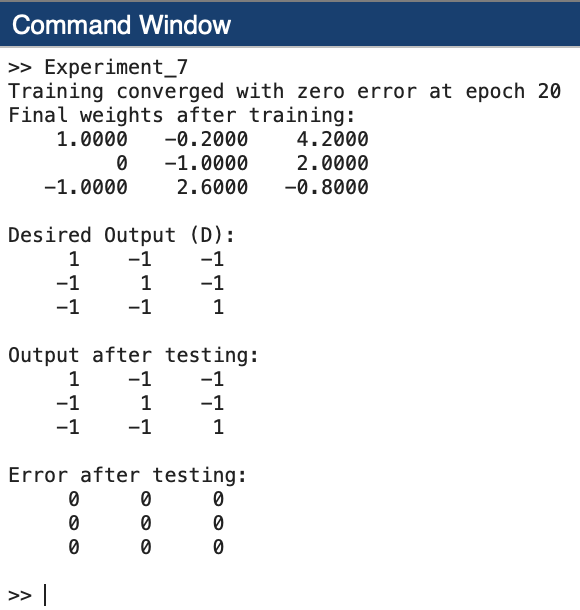
D = [1 -1 -1 ; -1 1 -1 ; -1 -1 1]

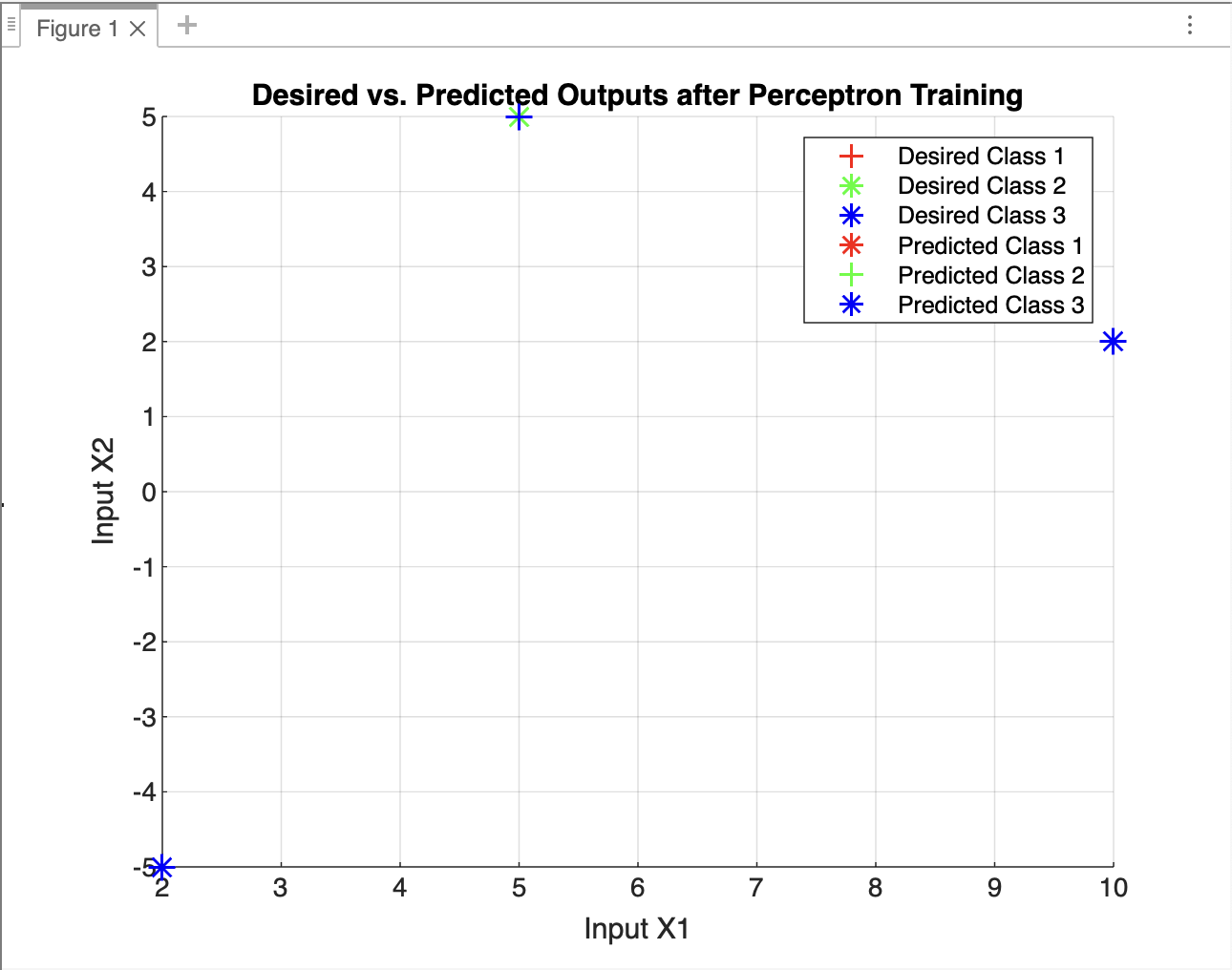
W=[1 -2 0; 0 -1 2; 1 3 -1] , c = 0.1

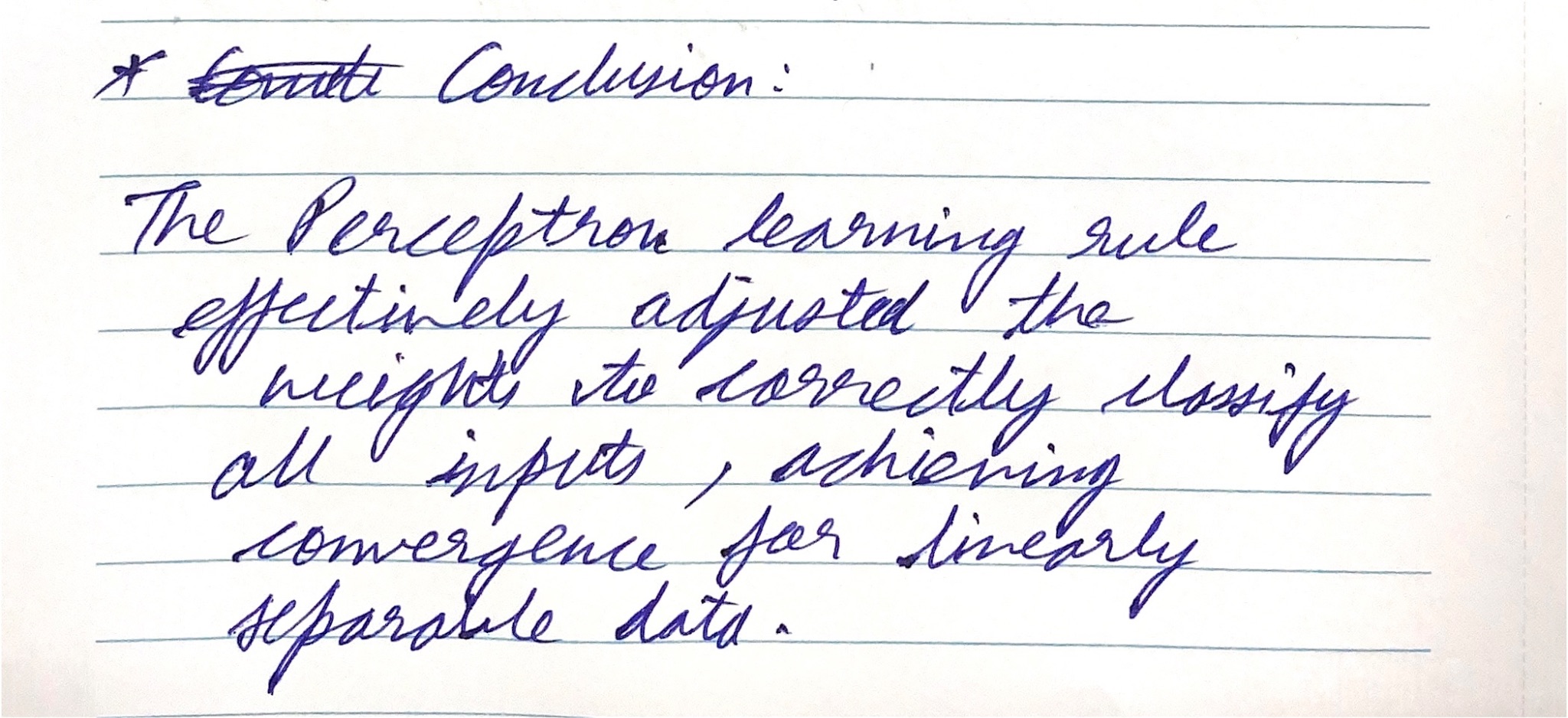
**Code:**

| X = [10 2; 2 -5; 5 5]; % Inputs D = [1 -1 -1; -1 1 -1; -1 -1 1]; % Desired outputs W = [1 -2 0; 0 -1 2; 1 3 -1]; % Initial weights c = 0.1; % Learning rate % Add bias input to X X = [X, -1\*ones(size(X,1), 1)]; epochs = 20;  % Training phase using Perceptron Learning Rule for i = 1:epochs  total\_error = 0; % Initialize total error for the epoch    % Iterate over each input vector  for j = 1:size(X, 1)  % Calculate the net input and apply the activation function (sign function)  Y = sign(W \* X(j, :)');    % Calculate the error between desired and predicted output  e = D(:, j) - Y;    % Update weights if there is any error (misclassification)  if any(e) % If there is a misclassification  W = W + c \* e \* X(j, :); % Update weights  total\_error = total\_error + sum(abs(e)); % Update total error  end  end    % Stop training if total error is zero (i.e., no misclassification)  if total\_error == 0  fprintf('Training converged with zero error at epoch %d\n', i);  break;  end end disp('Final weights after training:'); disp(W); % Testing phase (Recall) Y\_test = sign(W \* X'); % Testing with the same input data error = D - Y\_test; disp('Desired Output (D):'); disp(D); disp('Output after testing:'); disp(Y\_test); disp('Error after testing:'); disp(error); % Plotting the results figure; hold on; colors = {'r', 'g', 'b'};  for i = 1:size(X, 1)  for j = 1:size(Y\_test, 1)  if Y\_test(j, i) == 1  plot(X(i, 1), X(i, 2), '+', 'Color', colors{j}, 'MarkerSize', 10, 'LineWidth', 1); % '+' for predicted output  else  plot(X(i, 1), X(i, 2), '\*', 'Color', colors{j}, 'MarkerSize', 10, 'LineWidth', 1); % '\*' for predicted output  end  end end xlabel('Input X1'); ylabel('Input X2'); title('Desired vs. Predicted Outputs after Perceptron Training'); legend('Desired Class 1', 'Desired Class 2', 'Desired Class 3', ...  'Predicted Class 1', 'Predicted Class 2', 'Predicted Class 3'); hold off; grid on; |
| --- |

**Output:**

****

****

****