Perception Assignment 2

Table of Contents

Name: Rachel Chen

Question 1

Simulate a two-alternative, forced-choice experiment. Each trial consists of one interval with stimulus A or stimulus B. In each 100 ms, the observer collects evidence for whether the stimulus is A or B. The evidence in each 100 ms time step consists of a single number. The expected value of that number is +2 * c for an A stimulus, and -2 * c for a B stimulus, where c is the stimulus contrast. However, that number is noisy, perturbed by independent random draws each time step from a standard Gaussian distribution (mean zero, SD = 1).

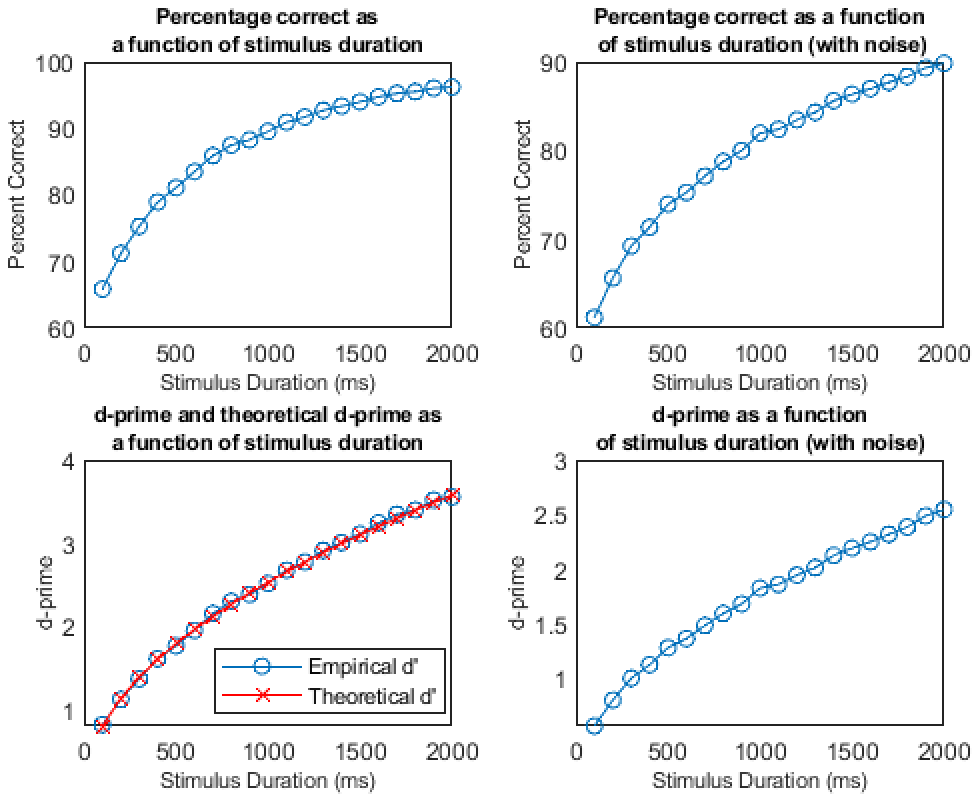
(a.1)

Simulate this experiment for stimulus durations ranging from 100 ms to 2 s and a contrast of 0.2. Assume A and B are equally likely, payoffs are symmetric and an optimal criterion. Thus, for example, for a 500 ms stimulus there are 5 time steps. And, if the stimulus has contrast 0.2, that means each time step is a random draw from a Gaussian with mean 0.4 and variance 1.0. The observer will optimally combine those bits of evidence by summing the 5 numbers from the 5 time steps. Thus, for an A stimulus, the total evidence is expected to be 5*.4 = 2, on average, and -2 for a B stimulus. What is the optimal criterion? Plot your results as percentage correct as a function of stimulus duration. Replot the results as d’ as a function of duration. Compare your simulation results with theoretical predictions.

```
clear all; close all; clc;
% Setting up parameters
c = 0.2; %contrast level
durations = 100:100:2000;
ntrials = 20000;

% Empty space for storing
pcorrect = zeros(length(durations), 1); % percentage correctness
pcorrectnoise= zeros(length(durations), 1);
d_prime = zeros(length(durations), 1);
d_prime_noise = zeros(length(durations), 1);
thero_dprime=zeros(length(durations), 1);

for i = 1:length(durations)
    duration = durations(i);
    num_steps = duration / 100;
    expectA = 2 * c; % output value
    expectB = -2 * c; % output value
    mu_signal=expectA* num_steps; % mean of signal distribution for each
 duration
```

```
    mu_noise =expectB* num_steps;  % mean of noise distribution for each
 duration
    sigma=sqrt(num_steps);
    thero_dprime(i) = (mu_signal - mu_noise) / sigma; %calculating theoretical
 d-prime


    % Simulate the trials
    actual = rand(ntrials, 1) > 0.5; % 1 for stimulus A, 0 for stimulus B
    response = zeros(ntrials, 1);
    responsenoise=zeros(ntrials, 1);

    for trial = 1:ntrials % randomly generate A or B
        noise=generate_noisy_number(num_steps);
        if actual(trial)
            evidence = sum(normrnd(expectA, 1, [1, num_steps])); % responseA
            evidencenoise=sum((normrnd(expectA, 1, [1, num_steps]))+noise);
        else
            evidence = sum(normrnd(expectB, 1, [1, num_steps])); % responseB
            evidencenoise=sum((normrnd(expectB, 1, [1, num_steps]))+noise);
        end

        response(trial) = evidence > 0; % 1 for guessing A, 0 for guessing B
        responsenoise(trial)=evidencenoise > 0;

    end

    correct_responses = actual == response;
    correct_responses_noise = actual == responsenoise;
    pcorrect(i) = mean(correct_responses) * 100;  % compute percentage correct
    pcorrectnoise(i) = mean(correct_responses_noise) * 100;

    % Compute d'
    hits = sum(actual & response) / sum(actual);
    correctRejections = sum(~actual & ~response) / sum(~actual);
    fas = sum(~actual & response) / sum(~actual);
    d_prime(i) = norminv(hits) - norminv(fas);

    %compute d' under noise
    hitsnoise = sum(actual & responsenoise) / sum(actual);
    fasnoise = sum(~actual & responsenoise) / sum(~actual);
    d_prime_noise(i) = norminv(hitsnoise) - norminv(fasnoise);

end

figure;
subplot(2,2,1);
plot(durations, pcorrect, '-o');
xlabel('Stimulus Duration (ms)','FontSize', 8);
ylabel('Percent Correct','FontSize', 8);
title({'Percentage correct as', 'a function of stimulus
 duration'}, 'FontSize',8);

subplot(2,2,3);
```

```
plot(durations, d_prime, '-o','DisplayName', 'Empirical d'''); hold on
plot(durations, thero_dprime, '-x','Color','r','DisplayName', 'Theoretical
 d''')
xlabel('Stimulus Duration (ms)','FontSize', 8);
ylabel('d-prime','FontSize', 8);
legend('Location', 'best');  % Display legend in the best location to avoid
 obscuring data
title({'d-prime and theoretical d-prime as',  'a function of stimulus
 duration'}, 'FontSize', 8);

subplot(2,2,2);
plot(durations, pcorrectnoise, '-o');
xlabel('Stimulus Duration (ms)','FontSize', 8);
ylabel('Percent Correct','FontSize', 8);
title({'Percentage correct as a function', ' of stimulus duration (with
 noise)'}, 'FontSize',8);

subplot(2,2,4);
plot(durations, d_prime_noise, '-o');
xlabel('Stimulus Duration (ms)','FontSize', 8);
ylabel('d-prime', 'FontSize', 8);
title({'d-prime as a function',  'of stimulus duration (with
 noise)'}, 'FontSize', 8);

%I simulated to study the effect of stimulus duration on the performance of an
 observer in
% distinguishing between two stimuli (A and B) under different conditions
% of noise. I setted up parameters according to instruction and tried to
% generate the evidence from a Gaussian distribution in each time of a
% trial. For stimulus A distribution, the mean is +2c; and for stimulus B,
% the mean is -2c; they all have variance of 1. I used the 'normrnd'
% function do that. To compute d-prime, I used d'=Z(hit rate)-Z(false alarm
% rate), where Z is the inverse of the cumulative Gaussian distribution
% function. To calculate the theoretical d-prime, I used the equation
 thero_dprime(i) = (mu_signal - mu_noise) / sigma;
% the former parts are means of the signal and noise distributions.  and # is
 the standard deviation,
% which is the square root of the number of time steps due to the accumulation
 of evidence.
% For interpretation, the plotted graphs showed how the percentage of
% correct respoinse and d-prime value vary with the stimulus duration. When
% the duration time increase, subjects have more time to gather evidence,
% thus leading to higher percentage of correctness and higher d' values. In
% this case, the discriminability between two stimuli is better.
% Comparing the theoretical d' and true d' can provide insight into the
% difference between the simulated behavior and predictions based on
 computations.
% Since we are simulating and setting parameters, the difference between
% computation and simulation is not much.
```



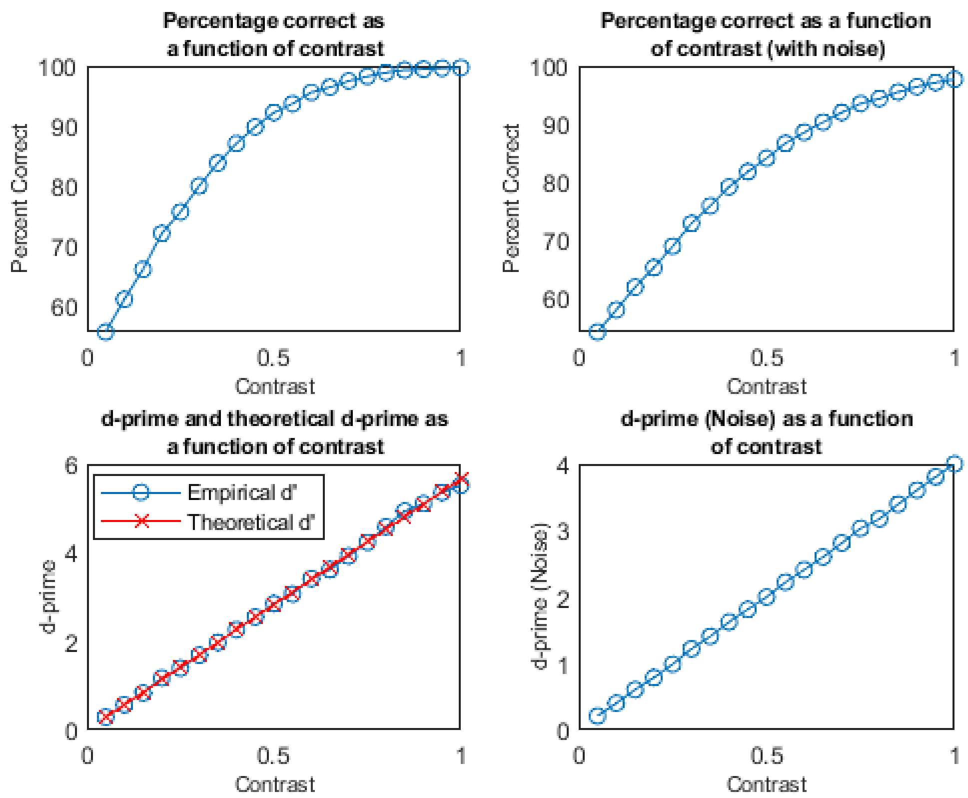
(a.2)

Now, repeat the above (determine optimal criterion, simulate, plot results, compare to theoretical predictions) for an experiment in which the duration is fixed at 0.2 s and contrast ranges from 0.05 to 1.

```
durations = 200;  % Duration is fixed at 0.2s
contrasts = 0.05:0.05:1;  % Contrast ranges from 0.05 to 1
ntrials = 20000;
pcorrect = zeros(length(contrasts), 1);
pcorrectnoise= zeros(length(contrasts), 1);
d_prime = zeros(length(contrasts), 1);
d_prime_noise = zeros(length(contrasts), 1);
thero_dprime = zeros(length(contrasts), 1);

for i = 1:length(contrasts)
    c = contrasts(i);
    expectA = 2 * c;  % output value
    expectB = -2 * c;  % output value
    mu_signal = expectA * (durations / 100);  % mean of signal distribution
    mu_noise = expectB * (durations / 100);  % mean of noise distribution
    sigma = sqrt(durations / 100);  % standard deviation of sum of evidence
    thero_dprime(i) = (mu_signal - mu_noise) / sigma;  % calculating
 theoretical d-prime

    % Simulate the trials
    actual = rand(ntrials, 1) > 0.5; % 1 for stimulus A, 0 for stimulus B
```

```
    response = zeros(ntrials, 1);
    responsenoise = zeros(ntrials, 1);

    for trial = 1:ntrials % randomly generate A or B
        noise = generate_noisy_number(durations / 100);
        if actual(trial)
            evidence = sum(normrnd(expectA, 1, [1, durations / 100])); %
 responseA
            evidencenoise = sum((normrnd(expectA, 1, [1, durations / 100])) +
 noise);
        else
            evidence = sum(normrnd(expectB, 1, [1, durations / 100])); %
 responseB
            evidencenoise = sum((normrnd(expectB, 1, [1, durations / 100])) +
 noise);
        end

        response(trial) = evidence > 0; % 1 for guessing A, 0 for guessing B
        responsenoise(trial) = evidencenoise > 0;
    end

    correct_responses = actual == response;
    correct_responses_noise = actual == responsenoise;
    pcorrect(i) = mean(correct_responses) * 100;  % compute percentage correct
    pcorrectnoise(i) = mean(correct_responses_noise) * 100;

    % Compute d'
    hits = sum(actual & response) / sum(actual);
    fas = sum(~actual & response) / sum(~actual);
    d_prime(i) = norminv(hits) - norminv(fas);

    %compute d' under noise
    hitsnoise = sum(actual & responsenoise) / sum(actual);
    fasnoise = sum(~actual & responsenoise) / sum(~actual);
    d_prime_noise(i) = norminv(hitsnoise) - norminv(fasnoise);

end

% Plotting
figure;
subplot(2,2,1);
plot(contrasts, pcorrect, '-o');
xlabel('Contrast','FontSize', 8);
ylabel('Percent Correct','FontSize', 8);
title({'Percentage correct as', 'a function of contrast'}, 'FontSize',8);

subplot(2,2,3);
plot(contrasts, d_prime, '-o','DisplayName', 'Empirical d'''); hold on
plot(contrasts, thero_dprime, '-x','Color','r','DisplayName', 'Theoretical
 d''')
xlabel('Contrast','FontSize', 8);
ylabel('d-prime','FontSize', 8);
legend('Location', 'best');  % Display legend in the best location to avoid
 obscuring data
```

```
title({'d-prime and theoretical d-prime as',  'a function of
 contrast'}, 'FontSize', 8);

subplot(2,2,2);
plot(contrasts, pcorrectnoise, '-o');
xlabel('Contrast','FontSize', 8);
ylabel('Percent Correct','FontSize', 8);
title({'Percentage correct as a function', ' of contrast (with
 noise)'}, 'FontSize',8);

subplot(2,2,4);
plot(contrasts, d_prime_noise, '-o');
xlabel('Contrast','FontSize', 8);
ylabel('d-prime (Noise)', 'FontSize', 8);
title({'d-prime (Noise) as a function',  'of contrast'}, 'FontSize', 8);

% In this case, the setting up, parameters and equations are similar. The
% reuslts are plotted to present responses under the condition of fixed
% stimulus duration but varied contrast levels. For the percentage
% correctness, It's expected that as contrast increases, the percentage of
% correct responses would also increase because a higher contrast usually
 makes
% the stimuli easier to discriminate. d-prime should also increase with
% contrast, indicating better dsicriminability between two stimuli.
```



(b)

Change the probability of an A stimulus to 0.75. Use a single fixed duration of 0.4 s and contrast of 0.1. What is the optimal criterion now? Simulate performance with that optimal criterion, and then simulate it with the criterion you used in part (a), comparing the performance across the two for 100 trials each. You should be able to compute the optimal criterion. However, you can also determine/estimate that criterion by doing large-scale simulations for a range of criteria (I'd prefer the closedform computation if you can manage it).

```
%To calculate optimal criterion, we can use the formula:
%log((exp((x-meanA)^2/-2*sigma^2))/(exp((x-meanB)^2/-2*sigma^2))) = ((1-pA)/
pA);
%And can thus obtain the value: optimal criterion is around -2.7465.

duration = 400;
num_steps = duration / 100;
contrast = 0.1;
ntrials = 100;
sigma=sqrt(num_steps);
pA = 0.75;
optCri = -2.7465;

expectA = 2 * contrast; % output value
meanA=expectA*num_steps;
expectB = -2 * contrast; % output value
meanB=expectB*num_steps;

%empty container
pcorrect_optimal = zeros(1, 1);
pcorrect_previous = zeros(1, 1);

%simulation
actual = rand(ntrials, 1) < pA;  % 1 for stimulus A, 0 for stimulus B
response_optimal = zeros(ntrials, 1);
response_previous = zeros(ntrials, 1);

for trial = 1:ntrials
    if actual(trial)
        evidence = sum(normrnd(expectA, 1, [1, num_steps]));  % responseA
    else
        evidence = sum(normrnd(expectB, 1, [1, num_steps]));  % responseB
    end


    % Decision making based on optimal criterion
    response_optimal(trial) = evidence > optCri;
    % Decision making based on previous criterion (0)
    response_previous(trial) = evidence > 0;

end

% Compute percentage correct for both criteria
pcorrect_optimal = mean(actual == response_optimal) * 100;
pcorrect_previous = mean(actual == response_previous) * 100;
```

```
% Display results
disp(['Optimal Criterion', num2str(optCri)])
disp(['Percentage correct with optimal criterion: ',
 num2str(pcorrect_optimal)]);
disp(['Percentage correct with previous criterion: ',
 num2str(pcorrect_previous)]);

%In this part,firstly calculated the optimal criterion manually, based on
%the Baye's Rue, and obtained the value that optimal criterion it's -2.71.
% for the decision makin procedure, the likelihood ratio should be bigger
% than the optimal criterion. The percentage of correct responses is computed
 for both
% criteria by comparing the responses to the actual stimuli.
%The likelihood ratio is used to make decisions based on the optimal
%criterion. For the performance comparison, it's expected that uisng the
%optimal criterion would yield  a higher percentage of correct responses
 compared to
% using the previous criterion of zero.
```

```
Optimal Criterion-2.7465
Percentage correct with optimal criterion: 81
Percentage correct with previous criterion: 63
```

Question 2

With the same setup as in (1), switch to a reaction-time experiment using the accumulated evidence values in a drift-diffusion framework. That is, accumulate the sum across time steps until the sum hits a bound representing a decision to respond “A” (with bound value +b) and another for response “B” (with value -b). Use simulations to characterize the reaction-time distributions for correct vs. error trials. Do this for contrasts of 0.1 and 0.5. For each contrast, run simulations for a near decision boundary (relatively small value of b), a distant decision boundary (large value of b), and an asymmetric pair of decision boundaries (+b and -d). What happens to the hit and false-alarm rates (treating stimulus B as “noise” and A as “signal”) and RT distributions with each manipulation of the model?

```
% The experiment runs for two contrast levels: 0.1 and 0.5.
contrasts = [0.1, 0.5];
bounds = [2, 2]; % Near and distant symmetric boundaries
boundsfar = [7, 7];
asym_bounds = [3, 4]; % Asymmetric boundaries
num_trials = 10000;
p_A = 0.5;

for i = 1:length(contrasts)
    contrast = contrasts(i);
    evidence_A = 2 * contrast;
    evidence_B = -2 * contrast;

    for b = 1:length(bounds)
        bound = bounds(b);
        RTs_correct = zeros(1, num_trials);
        RTs_error = zeros(1, num_trials);
        hits = 0;
        fas = 0;
```

```
        for trial = 1:num_trials
            is_A = rand() < p_A;
            evidence = 0;
            time_steps = 0;

            while evidence < bound && evidence > -bound
                time_steps = time_steps + 1;
                if is_A
                    evidence = evidence + normrnd(evidence_A, 1);
                else
                    evidence = evidence + normrnd(evidence_B, 1);
                end
            end

            if evidence >= bound
                if is_A
                    RTs_correct(trial) = time_steps;
                    hits = hits + 1;
                else
                    RTs_error(trial) = time_steps;
                    fas = fas + 1;
                end
            else
                if is_A
                    RTs_error(trial) = time_steps;
                else
                    RTs_correct(trial) = time_steps;
                end
            end
        end

        fprintf('Contrast: %.1f, Bound: %d\n', contrast, bound);
        fprintf('Hit rate: %.2f%%, False alarm rate: %.2f%%\n', hits/
num_trials*100, fas/num_trials*100);
        fprintf('Mean RT (correct): %.2f, Mean RT (error): %.2f\n\n',
 mean(RTs_correct(RTs_correct~=0)), mean(RTs_error(RTs_error~=0)));
    end

    % Plot RT distributions
        figure;

        histogram(RTs_correct, 'Normalization', 'probability');
        hold on;
        histogram(RTs_error, 'Normalization', 'probability');
        title(['Contrast: ', num2str(contrast), ', Boundaries: ',
 num2str(bounds)]);
        legend('Correct', 'Error');
        xlabel('Reaction Time (steps)');
        ylabel('Probability');



        %Plotting for far boundaries
        for bf = 1:length(boundsfar)
```

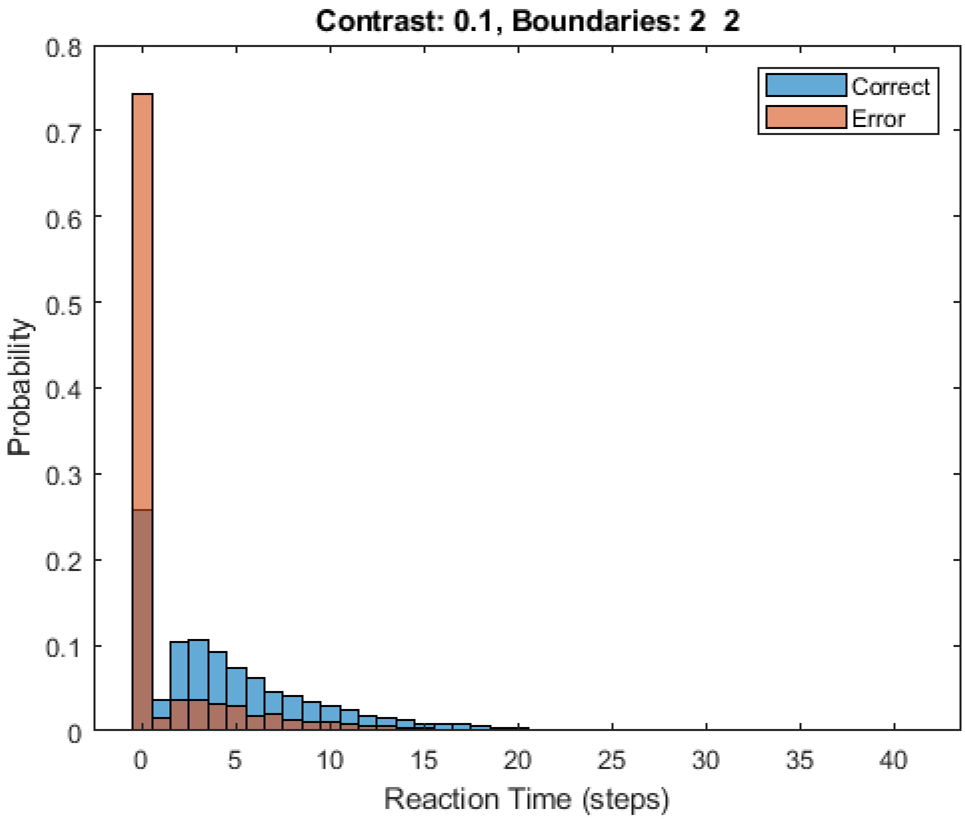
```
        boundfar = boundsfar(b);
        RTs_correct = zeros(1, num_trials);
        RTs_error = zeros(1, num_trials);
        hits = 0;
        fas = 0;

        for trial = 1:num_trials
            is_A = rand() < p_A;
            evidence = 0;
            time_steps = 0;

            while evidence < boundfar && evidence > -boundfar
                time_steps = time_steps + 1;
                if is_A
                    evidence = evidence + normrnd(evidence_A, 1);
                else
                    evidence = evidence + normrnd(evidence_B, 1);
                end
            end

            if evidence >= boundfar
                if is_A
                    RTs_correct(trial) = time_steps;
                    hits = hits + 1;
                else
                    RTs_error(trial) = time_steps;
                    fas = fas + 1;
                end
            else
                if is_A
                    RTs_error(trial) = time_steps;
                else
                    RTs_correct(trial) = time_steps;
                end
            end
        end

        fprintf('Contrast: %.1f, Bound: %d\n', contrast, boundfar);
        fprintf('Hit rate: %.2f%%, False alarm rate: %.2f%%\n', hits/
num_trials*100, fas/num_trials*100);
        fprintf('Mean RT (correct): %.2f, Mean RT (error): %.2f\n\n',
 mean(RTs_correct(RTs_correct~=0)), mean(RTs_error(RTs_error~=0)));
        end

    % Plot RT distributions
        figure;

        histogram(RTs_correct, 'Normalization', 'probability');
        hold on;
        histogram(RTs_error, 'Normalization', 'probability');
        title(['Contrast: ', num2str(contrast), ', Boundaries: ',
 num2str(boundsfar)]);
        legend('Correct', 'Error');
        xlabel('Reaction Time (steps)');
```

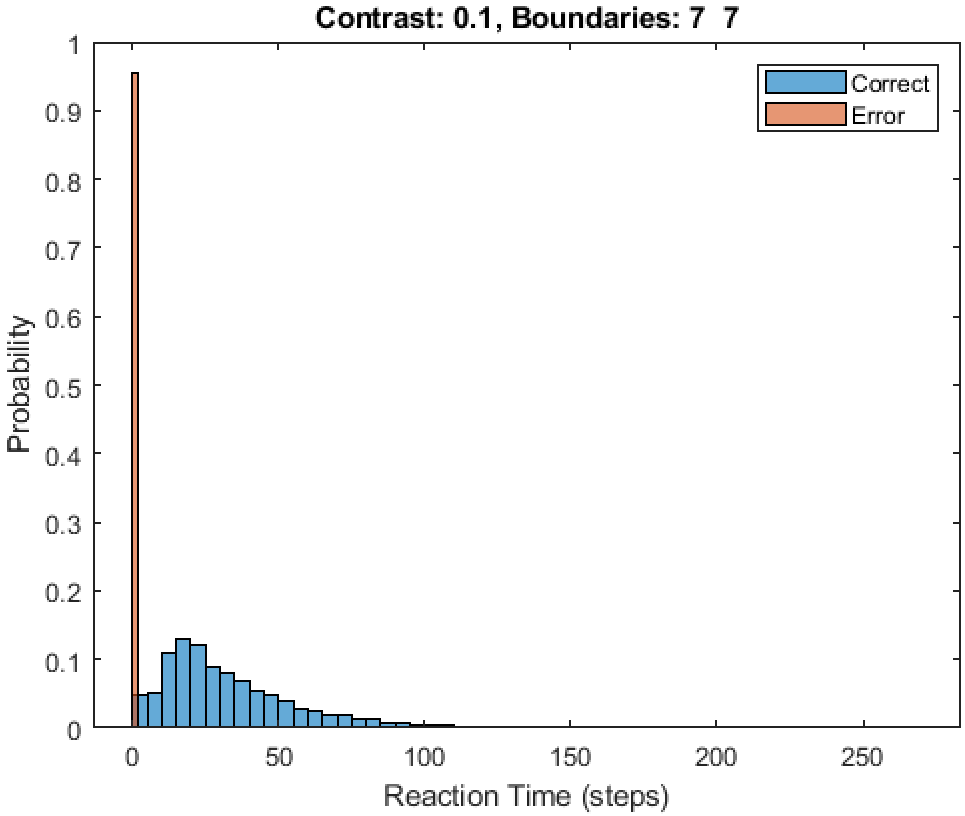
```
        ylabel('Probability');


    % Asymmetric boundaries
    RTs_correct = zeros(1, num_trials);
    RTs_error = zeros(1, num_trials);
    hits = 0;
    fas = 0;

    for trial = 1:num_trials
        is_A = rand() < p_A;
        evidence = 0;
        time_steps = 0;

        while evidence < asym_bounds(1) && evidence > -asym_bounds(2)
            time_steps = time_steps + 1;
            if is_A
                evidence = evidence + normrnd(evidence_A, 1);
            else
                evidence = evidence + normrnd(evidence_B, 1);
            end
        end

        if evidence >= asym_bounds(1)
            if is_A
                RTs_correct(trial) = time_steps;
                hits = hits + 1;
            else
                RTs_error(trial) = time_steps;
                fas = fas + 1;
            end
        else
            if is_A
                RTs_error(trial) = time_steps;
            else
                RTs_correct(trial) = time_steps;
            end
        end
    end

    fprintf('Contrast: %.1f, Asymmetric Bounds: +b = %d, d = %d\n', contrast,
 asym_bounds(1), asym_bounds(2));
    fprintf('Hit rate: %.2f%%, False alarm rate: %.2f%%\n', hits/
num_trials*100, fas/num_trials*100);
    fprintf('Mean RT (correct): %.2f, Mean RT (error): %.2f\n\n',
 mean(RTs_correct(RTs_correct~=0)), mean(RTs_error(RTs_error~=0)));

    % Plot RT distributions
    figure;

        histogram(RTs_correct, 'Normalization', 'probability');
        hold on;

        histogram(RTs_error, 'Normalization', 'probability');
```

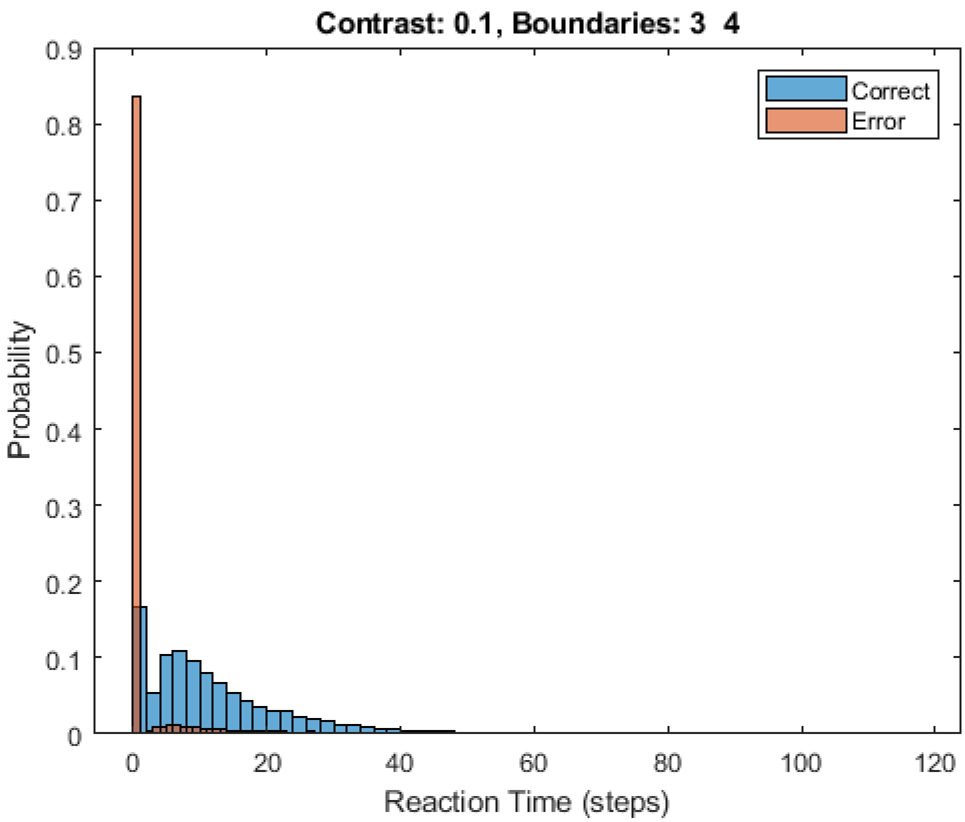
```
        title(['Contrast: ', num2str(contrast), ', Boundaries: ',
 num2str(asym_bounds)]);
        legend('Correct', 'Error');
        xlabel('Reaction Time (steps)');
        ylabel('Probability');
end

%In this question, I simulated a 2AFC task with Drift-diffusion model.
%For generating the evidence, they are generated by gaussian distribution
%with mean of expect value and standard deviation of 1.
%Decisition A is made is made if the accumulated evidence reaches or exceeds
 the positive boundary.
%Decision B is made if the accumulated evidence reaches or exceeds the
 negative boundary.
%For the results, The experiment is run for two contrast levels, 0.1 and 0.5.
 A higher
%contrast means the evidence for the corresponding stimulus (A or B) will be
 stronger,
% leading to faster and potentially more accurate decisions.  A higher
 contrast should generally
% lead to faster and more accurate decisions since the evidence is stronger.
%For reaction time, The code calculates the mean reaction time for both
 correct and error trials.
%A bimodal distribution might suggest two distinct processes or strategies at
 play.
% A shorter reaction time suggests faster decision-making, while a longer
 reaction time indicates more deliberation or uncertainty.
%For different boundaries, nearer boundaries might lead to faster decisions
 but with a higher
% chance of errors since less evidence is required to make a decision.
% Distanced boundaries would require more evidence accumulation,
% leading to slower but potentially more accurate decisions.
%Asymmetric boundaries introduce a bias in the decision-making process.
% If the boundary for deciding 'A' is closer than for 'B', decisions might be
 biased towards 'A'.
```

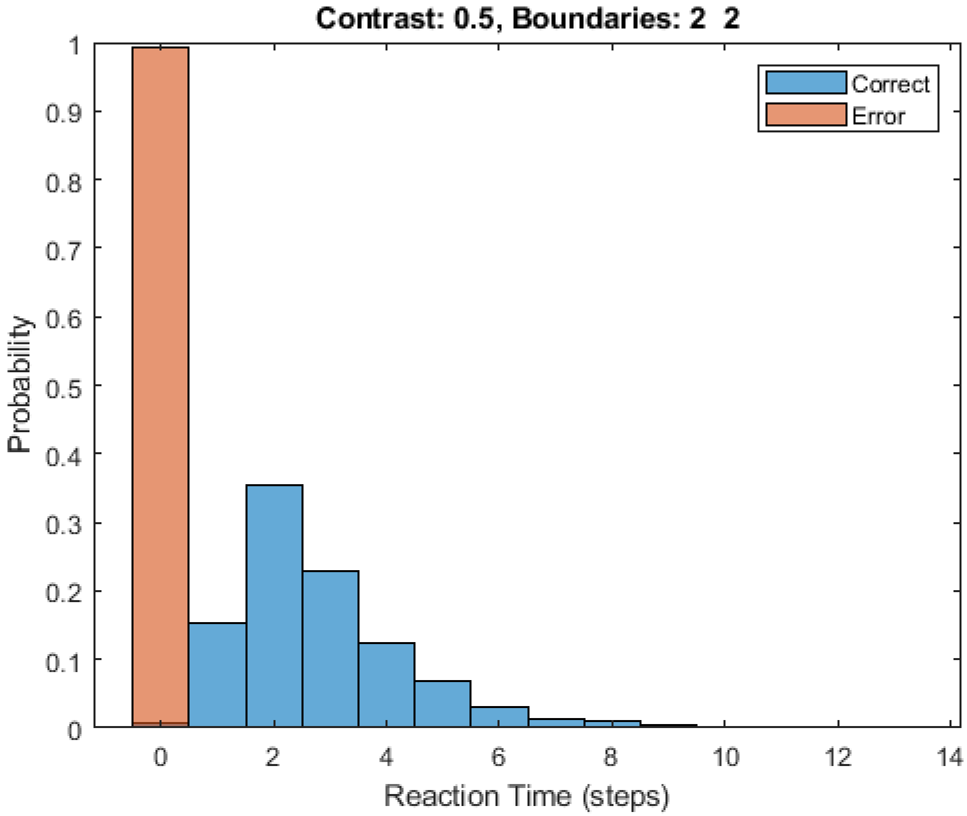
```
Contrast: 0.1, Bound: 2
Hit rate: 36.94%, False alarm rate: 13.15%
Mean RT (correct): 6.36, Mean RT (error): 6.26

Contrast: 0.1, Bound: 2
Hit rate: 37.63%, False alarm rate: 13.05%
Mean RT (correct): 6.36, Mean RT (error): 6.32

Contrast: 0.1, Bound: 7
Hit rate: 47.75%, False alarm rate: 2.43%
Mean RT (correct): 34.97, Mean RT (error): 35.17

Contrast: 0.1, Bound: 7
Hit rate: 47.37%, False alarm rate: 2.19%
Mean RT (correct): 35.16, Mean RT (error): 35.11

Contrast: 0.1, Asymmetric Bounds: +b = 3, d = 4
Hit rate: 43.26%, False alarm rate: 10.48%
```

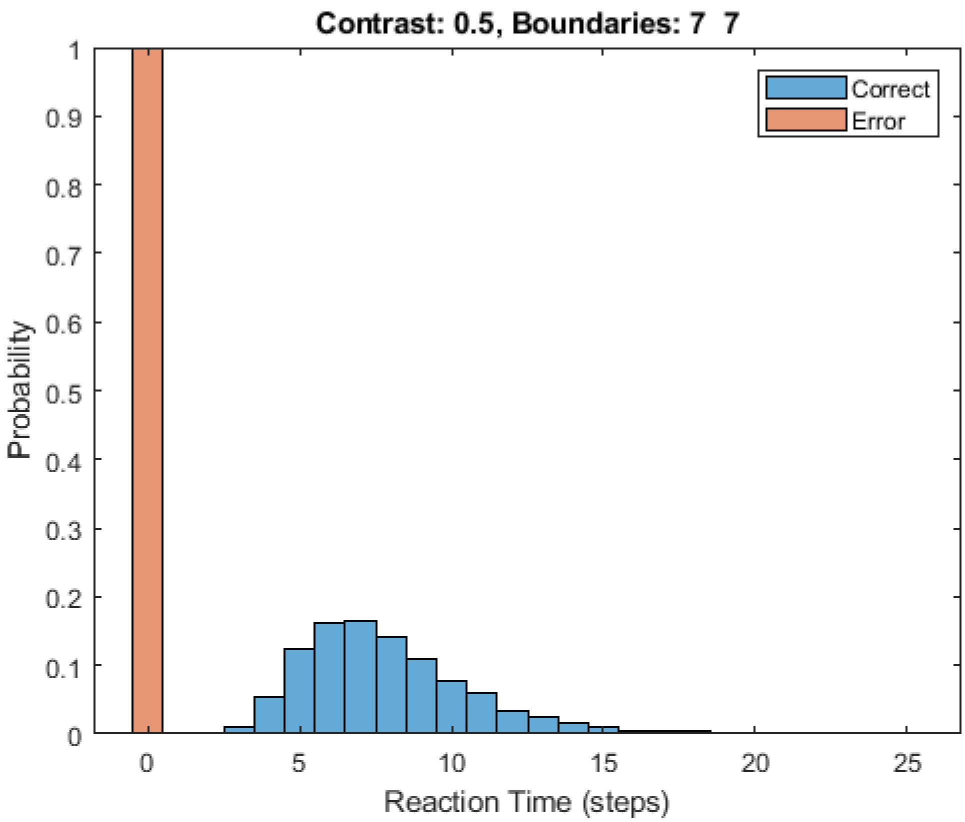
```
Mean RT (correct): 13.98, Mean RT (error): 13.87

Contrast: 0.5, Bound: 2
Hit rate: 49.35%, False alarm rate: 0.27%
Mean RT (correct): 2.85, Mean RT (error): 2.73

Contrast: 0.5, Bound: 2
Hit rate: 49.22%, False alarm rate: 0.36%
Mean RT (correct): 2.85, Mean RT (error): 2.55

Contrast: 0.5, Bound: 7
Hit rate: 49.25%, False alarm rate: 0.00%
Mean RT (correct): 7.87, Mean RT (error): NaN

Contrast: 0.5, Bound: 7
Hit rate: 49.73%, False alarm rate: 0.00%
Mean RT (correct): 7.85, Mean RT (error): NaN

Contrast: 0.5, Asymmetric Bounds: +b = 3, d = 4
Hit rate: 49.88%, False alarm rate: 0.07%
Mean RT (correct): 4.37, Mean RT (error): 4.14
```

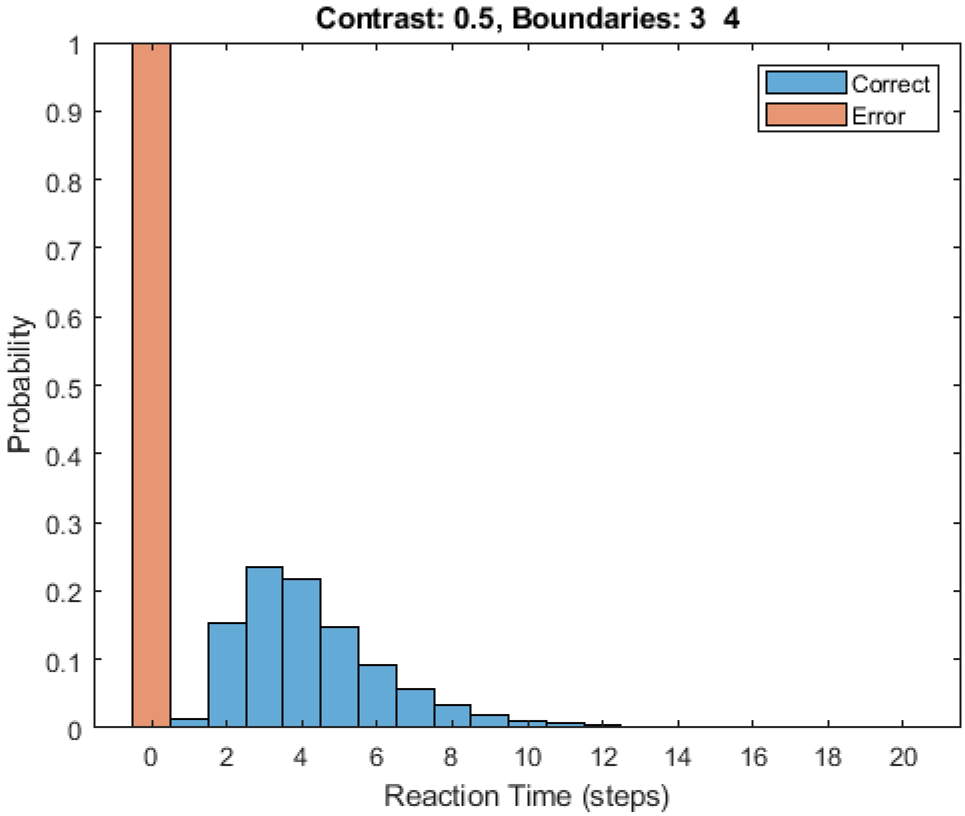



Contrast: 0.1, Boundaries: 7 7
Probability
Reaction Time (steps)
Correct
Error
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250


Contrast: 0.1, Boundaries: 3 4
Probability
Reaction Time (steps)
Correct
Error
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
20
40
60
80
100
120


Contrast: 0.5, Boundaries: 2 2
Correct
Error
Probability
Reaction Time (steps)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2
4
6
8
10
12
14


Contrast: 0.5, Boundaries: 7 7
Correct
Error
Probability
Reaction Time (steps)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
5
10
15
20
25



functions

```
function noise_number = generate_noisy_number(time_steps)
    % Generate a random number from a standard Gaussian distribution for each
 time step
    noise_number = randn(1, time_steps);
end
```

Published with MATLAB® R2022b