Decision Making

This experiment models the decision-making stage by a hypothetical decision neuron accumulating the spikes retrieved from 100 sensory neurons.

$$d = \sum_{t=1}^{T} w^T r(t) = \sum_{t=1}^{T} \sum_{i=1}^{N} w_i r_i(t)$$

In this experiment, I fixed the two stimuli to be discriminated to be ϕ_1 = 45°, ϕ_2 = -45°. Assume independent sensory neurons. When the decision variable reaches a bound B, or when the stimulus is over, the corresponding decision is made. The time when |d| reaches B is the reaction time T. If |d| never reaches B, then T = 1. I used weights w proportional to f(ϕ_1) – f(ϕ_2). d > 0 means that the neuron choses ϕ_1, d < 0 otherwise. The time limit is set to 1 second throughout the experiment. I used 10ms time bins in this experiment.

PART I: Population Response and Read-Out Weights

The followings are the functions I used to calculate the population responses as in the Decoding experiment. Assume independent, uncorrelated, Poisson-like response variability.

```
function fi = getfi(inputOrient, orienti, c, k, fmax)
% Get the populatiton response for specific neuron orientation(orienti) over input
% orientation(inputOrient)
% inputOrient in degrees, orient for specific neuron in radians
    inputOrient = deg2rad(inputOrient);
    f0 = 5;
    fi = f0 + c * fmax * exp(k * (cos(2 * (inputOrient - orienti)) - 1));
end

function f = getf(inputOrient, c, n, k, fmax)
% Population response for euispaced neurons from 0 to pi
% over input orientation(inputOrient)
% inputOrient in degrees, orient for specific neuron in radians
    f = zeros(1, n);
    neuron = (0 : pi / n : pi - pi / n);    % Equispaced neurons of size n
    for i = 1 : n
        f(1, i) = getfi(inputOrient, neuron(1, i), c, k, fmax);
    end
end

function r = getResponse(inputOrient, c, n, k, fmax, time)
% Get the poisson noised population response
% Simulate in given time with each trial equal to 10ms
    tnum = time * 100;      % 100 trials in 1 second
    f = getf(inputOrient, c, n, k, fmax);
    r = poissrnd(repmat(f, tnum, 1), [tnum, n]);
end
```

The followings are the functions to calculate the covariance matrix and the read-out weights, as in the Decoding experiment.

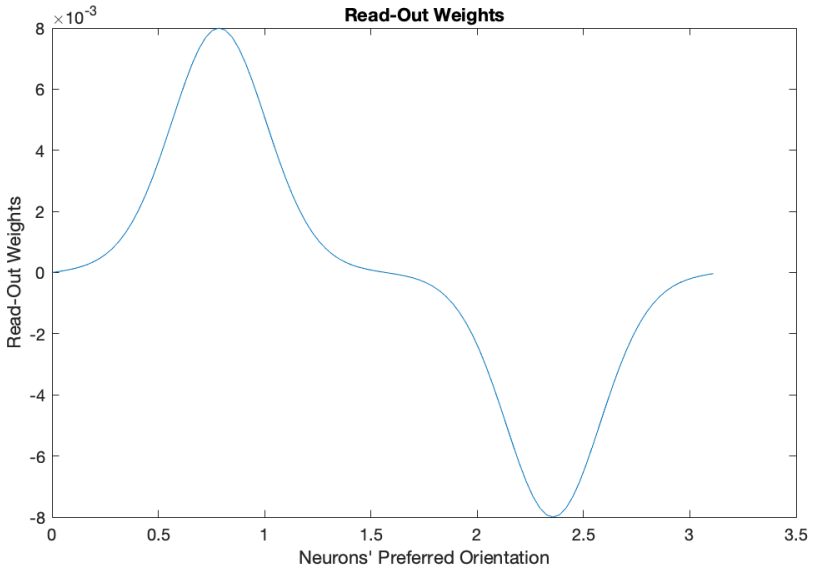
```
function cov = getCov(c, n, k, fmax)
% Get the covariance matrix
% No correlation noise taken into account (independent neurons)
% Diagonal Poisson variance
    cov = zeros(n, n);
    for i = 1 : n
        orienti = (i - 1) * pi / n;
        cov(i, i) = getfi(0, orienti, c, k, fmax);
    end
end

function w = getWeights(orient1, orient2, c, n, k, fmax)
% Get the read-out weights
% No correlation noise taken into account (independent neurons)
% Diagonal Poisson variance on the covariance matrix
    w = zeros(1, n);
    neuron = (0 : pi / n : pi - pi / n);    % Equispaced neurons of size n
    cov = getCov(c, n, k, fmax);
    for i = 1 : n
        fprime = getfi(orient1, neuron(1, i), c, k, fmax) - getfi(orient2, neuron(1,
i), c, k, fmax);
        w(1, i) = 1 / cov(i, i) * fprime;
    end
end
```

PART II: Decision Making Analysis

Fix $\kappa = 5$ and $f_{max} = 20$ for the rest of the experiment.

a. Read-Out Weights Plot

The following is the plot of w when contrast c = 0.002 between the 2 stimuli. The weights become 0 at multiples of $\pi/2$ and changes sign at each multiple.



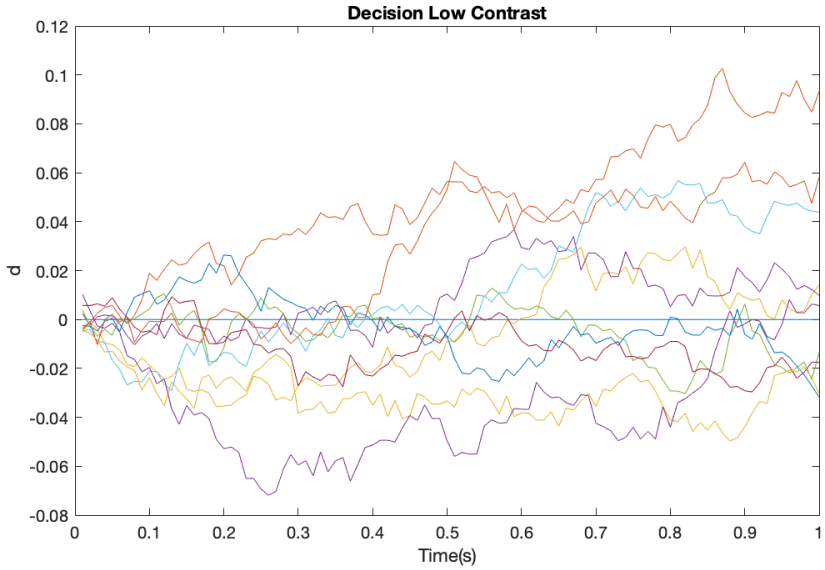
b. d vs. Time with Low Contrast and No Bound

Fix c = 0.0001 and B = ∞. The following code calculates the decision variable d using the accumulation model. The function also retrieves T and whether the decision is correct for the trial.

```
function [d, T, correct] = getDecision(orient1, orient2, c, time, B)
% Evidence accumulation model
% Simulate in given time with each trial equal to 10ms
% Get the decision variable d sequences corresponding to orient1
% Orientation in degrees
% No correlation noise taken into account (independent neurons)
% Diagonal Poisson variance on the covariance matrix
% T is the time simulated untill decision is made
% B is the bound for the max evidence (d value) we can reach
% If reaches the bound then make the decision immediately
% Else continue until T reaches time limit
% if final d > 0 then the decision is correct (orient1 chosen)

    n = 100;
    k = 5;
    fmax = 20;

    bin = 1 / 100;  % 10ms time bins
    tnum = time / bin;
    w = getWeights(orient1, orient2, c, n, k, fmax);
    r = getResponse(orient1, c, n, k, fmax, time);

    d = zeros(1, tnum);
    d(1, 1) = w * r(1, :).';
    T = 0.01;       % 10ms after one trial
```

```
    for t = 2 : tnum
       d(1, t) = d(1, t - 1) + w * r(t, :).';
       T = T + bin;
       if T >= 1 || abs(d(1, t)) >= B
           break
       end
    end

    if d(1, t) >= B         % Fill in the remaining d data with B if reached
        for t = t : tnum
            d(1, t) = B;
        end
    elseif d(1, t) <= -B     % Same for -B if reached before time limit
        for t = t : tnum
            d(1, t) = -B;
        end
    end

    if d(1, tnum) >= 0       % Get the decision
        correct = 1;
    else
        correct = 0;
    end

end

d = zeros(10, 100);
t = [0.01 : 0.01 : 1];
plot(t, zeros(1, 100));
hold on
for i = 1 : 10
    [d(i, :), ~, ~] = getDecision(45, -45, 0.002, 1, inf);
    plot(t, d(i, :));
end
```

The plot for 10 example runs is the following. We can see that we have 6 correct decisions and 4 wrong decisions. When the contrast is low, the model has low accuracy in discrimination.



c. Deciding c^*

Now we want to find a value of contrast c^* such that 2/3 of the trials have d > 0 when T = 1, B = ∞, which means that the percent correct is about 67%. The next I have a function to calculate percent correct based on the performance of 1000 trials.

```
function p = percentCorrect(orient1, orient2, c, time, B, trialNum)
% Return the percent correct of the decision made in given number of trials
    correctNum = 0;
    for i = 1 : trialNum
        [~, ~, correct] = getDecision(orient1, orient2, c, time, B);
        if correct > 0
            correctNum = correctNum + 1;
        end
    end
    p = correctNum / trialNum;
end

p = percentCorrect(45, -45, c, 1, inf, 1000);
```

Gradually increasing c, I found c^* = 0.02 that creates a percent correct of about 67%. Next, we have the plot of 10 example runs with c^* as contrast. We got 3 wrong decisions among these 10 runs.

d. Psychometric Function (Infinite B)

Psychometric function is the percent correct as a function of contrast c as the following.

```
function p = p_vs_c(orient1, orient2, c, time, B, trialNum)
% Get the percent correct sequence base on the given contrast sequence(c)
% Fix time limit(time) and bound(B)
    p = zeros(1, length(c));
    for i = 1 : length(c)
        p(1, i) = percentCorrect(orient1, orient2, c(1, i), time, B, trialNum);
    end
end
```

I also have a function to plot the 1σ error bars (including 67% of the data), where

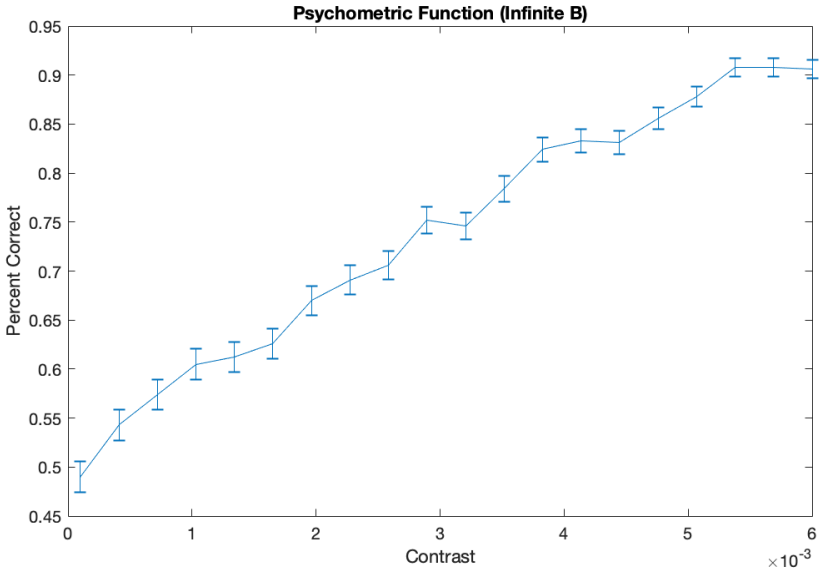
$$\sigma = \sqrt{p(1-p)/trialNum}$$

```
function err = getError(p, trialNum)
% Get 1 * variance error sequence according to the p sequence
% (Includes 67 percent of the distribution)
    err = zeros(1, length(p));
    for i = 1 : length(p)
        err(1, i) = sqrt(p(1, i) * (1 - p(1, i)) / trialNum);
    end
end
```

Then I varied contrast from 0.0001 to $3c^*$ = 0.006 for 20 values and simulated 1000 trials for each contrast. The following is the script and the resulting plot. We can see that the percent correct increases from 50% as contrast increases and reaches a limit at about 90%.

```
c = [0.0001 : (0.006 - 0.0001) / 19 : 0.006];
```

```
p = p_vs_c(45, -45, c, 1, inf, 1000);
err = getError(p, 1000);
errorbar(c, p, err);
```



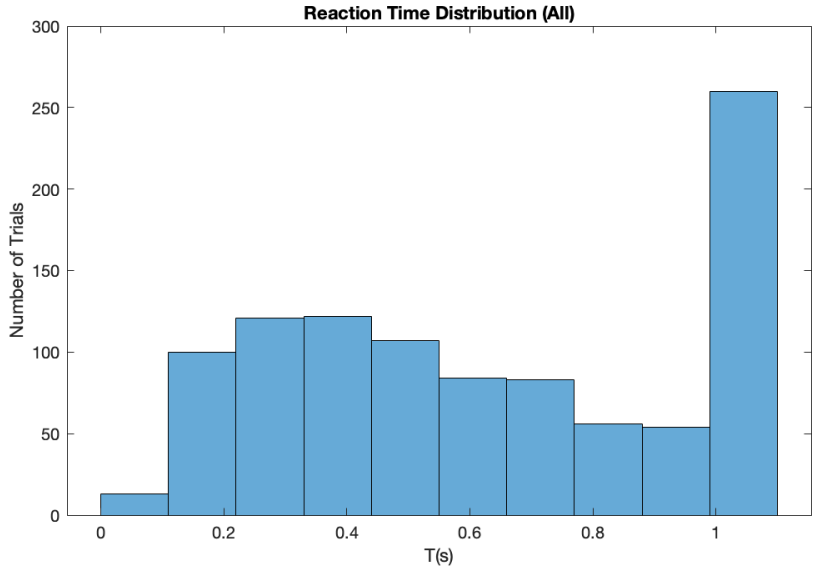
e. Deciding B

Now we need to set a bound for out model. The following function helps us get the percent of fast reaction, namely when T < 1. Also, it records the sequences of all reaction times in each trial. It also records the reaction times of each group (correct decision vs. wrong decision).

```
function [pfast, Ttotal, Tcorrect, Twrong] = reactTime(orient1, orient2, c, time, B,
trialNum)
% pfast is the percentage of fast react
% Ttotal is the sequence of reaction times for all trials
% Tcorrect is for trials that have made the right decisions
% T wrong is for trials that have made the wrong decisions

    fastReact = 0;      % Count fast reaction numbers
    Ttotal = zeros(1, trialNum);

    correctNum = 0;     % Count correct decision trial numbers
    Tcorrect_temp = zeros(1, trialNum);
    Twrong_temp = zeros(1, trialNum);

    for i = 1 : trialNum

        [~, T, correct] = getDecision(orient1, orient2, c, time, B);
        if T < 1
            fastReact = fastReact + 1;
        end

        if correct > 0
            correctNum = correctNum + 1;
```

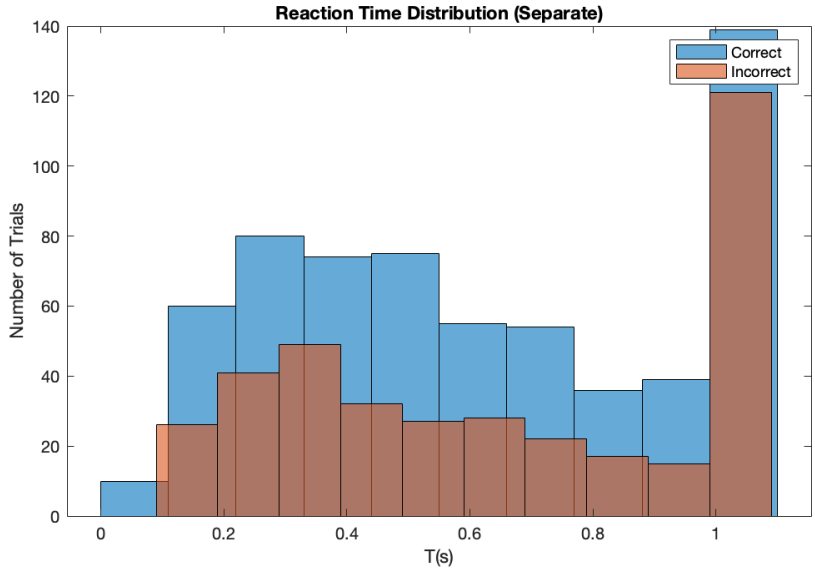
```
            Tcorrect_temp(1, i) = T;
        else
            Twrong_temp(1, i) = T;
        end

        Ttotal(1, i) = T;
    end

    pfast = fastReact / trialNum;

    Tcorrect = zeros(1, correctNum);
    Twrong = zeros(1, trialNum - correctNum);
    countc = 0;
    countw = 0;                     % Fill in reaction time values for each group
    for i = 1 : trialNum
        if Tcorrect_temp(1, i) ~= 0
            countc = countc + 1;
            Tcorrect(1, countc) = Tcorrect_temp(1, i);
        end
        if Twrong_temp(1, i) ~= 0
            countw = countw + 1;
            Twrong(1, countw) = Twrong_temp(1, i);
        end
    end

end

[pfast, Ttotal, Tcorrect, Twrong] = reactTime(45, -45, 0.002, 1, 0.76, 1000);
histogram(Ttotal, 10);
```

I still use c^* = 0.002 as contrast. With the above function, I was able to find B = 0.76 where the percentage of fast reaction is about 75%. The following is the distribution of reaction times for 1000 trials when B = 0.76. We see that in all fast reaction trials, the mean is at about 0.3s, and the distribution is skewed to the right.



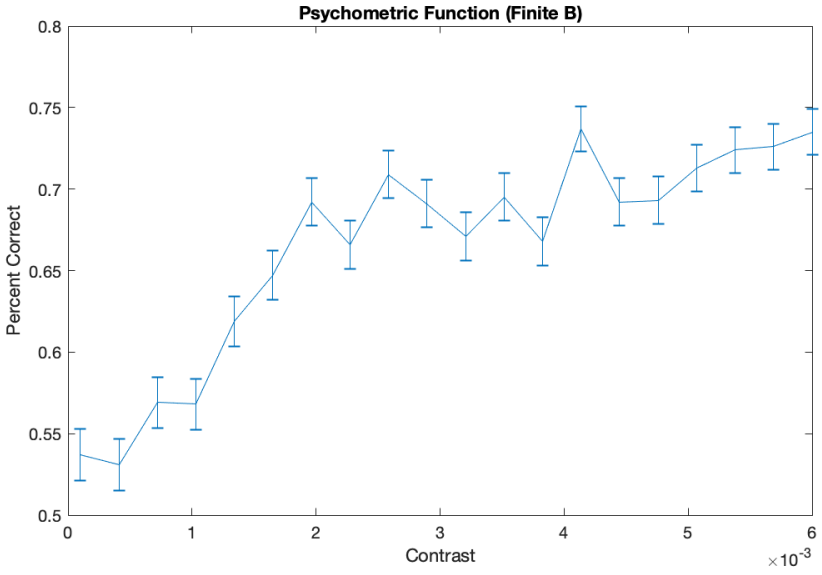
f. Reaction Time (Correct vs. Incorrect Trials)

Using the same function as above, I separated the reaction times into correct and incorrect trials (Tcorrect and Twrong). The following plot is their distribution. We see that they almost completely overlap, meaning the reaction time is not affected by any decision factors. The distributions are skewed to the right as expected.



g. Psychometric Function (Finite B)

Varying contrast from 0.0001 to 0.006 for 20 values and using B = 0.76, we can plot the psychometric function in the same way as above. We see that this time the percent correct is lower when we have a finite bound. It has a value of about 75% even when the contrast is 0.006.

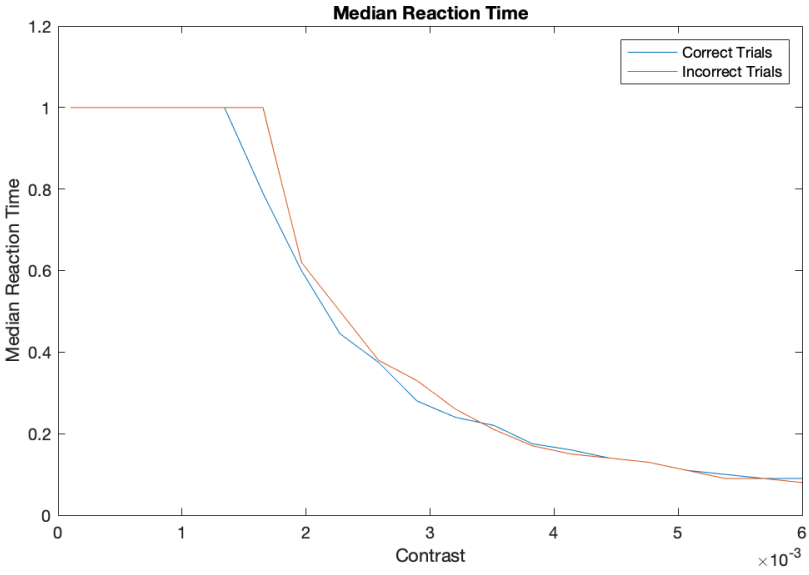


h. Median Reaction Time

I use the following function to find the median reaction time in response to the change of contrast.

```
function [medTc, medTw] = medT_vs_c(orient1, orient2, c, time, B, trialNum)
% mediTc is the sequence of median reaction time in which the decision is correct
% medTw is the sequence of median reaction time in which the decision is incorrect
% Simulate for each c and get one medTc and one medTw for each c
% For each simulation (each c) run given number of trials
    medTc = zeros(1, length(c));
    medTw = zeros(1, length(c));

    for i = 1 : length(c)
        [~, ~, Tcorrect, Twrong] = reactTime(orient1, orient2, c(1, i), time, B,
trialNum);
        medTc(1, i) = median(Tcorrect);
        medTw(1, i) = median(Twrong);
    end
end
```

Then I plotted the median reaction time as a function of contrast for the same range of contrast and same bound as in section d and g. The median reaction time remains 1s when the contrast is low, then starts to decrease as contrast increases for both correct and incorrect trials. The 2 groups again overlap mostly as in section f. But it seems that the correct trials react faster then incorrect trials at first.


Median Reaction Time
Correct Trials
Incorrect Trials
Median Reaction Time
Contrast
×10^-3
0
0.2
0.4
0.6
0.8
1
1.2
0
1
2
3
4
5
6