Decoding

This experiment consists of 3 parts. They are all performing analysis of decision percent correct (or d') in responding to the change of different variables such as the number of neurons (n), κ, and the variance of κ or f_{max}. The neurons are equispaced between [0…I'm using von-Mises function-shaped tuning curves with offset f_0 = 5:

$$f_i(c, \phi) = f_0 + c f_{max} \exp[\kappa(\cos(2(\phi - \phi_i)) - 1)]$$

where $0 \leq c \leq 1$ is the contrast.

PART I: Analysis with No Correlation Noise

In this part I fixed f_{max} = 20 and κ = 1.

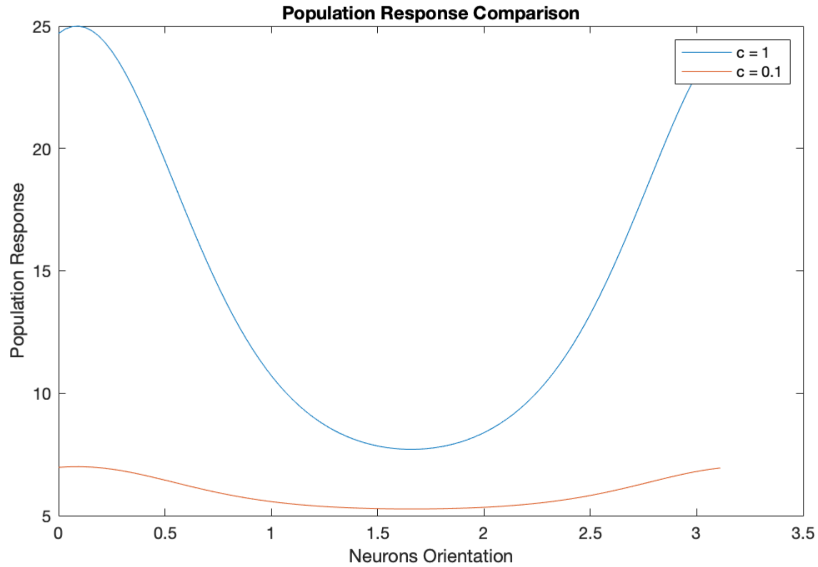
1. Population Response

The followings are the functions that generate the population response, implementing the above equation.

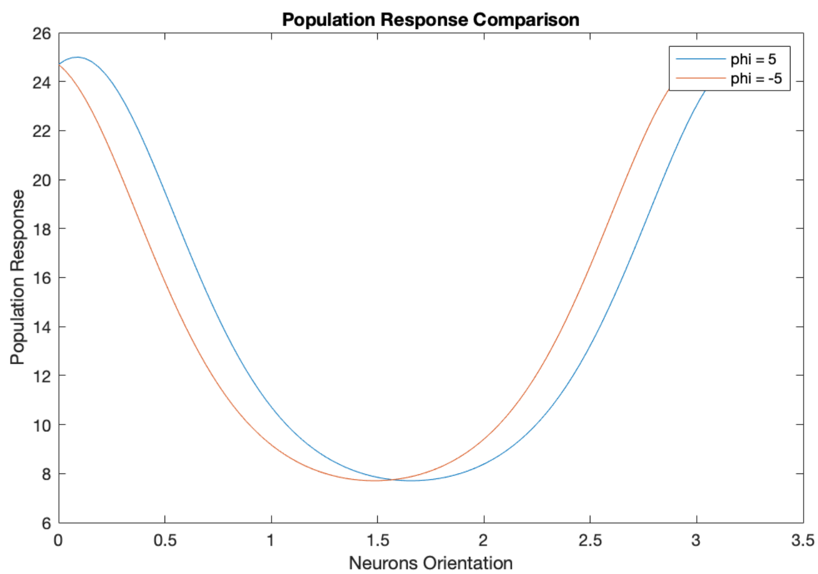
```
function fi = getfi(inputOrient, orienti, c, k, fmax)
% Get the populatiton response for specific neuron orientation(orienti) over input
% orientation(inputOrient)
% inputOrient in degrees, orient for specific neuron in radians
    inputOrient = deg2rad(inputOrient);
    f0 = 5;
    fi = f0 + c *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

function f = getf(inputOrient, c, n, k, fmax)
% Population response sequence for n equispaced 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
```

1st Comparison: ϕ = 5°, c = 1 and c = 0.1:

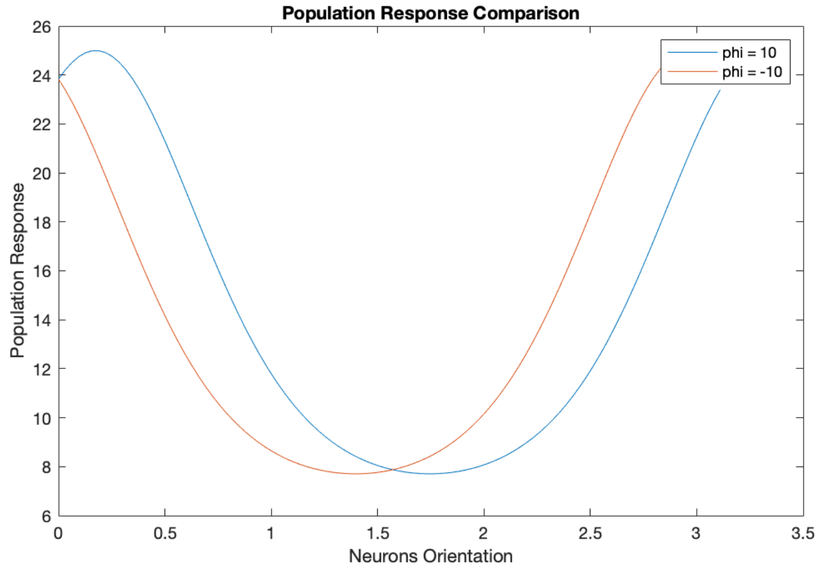
The slope of the curve becomes more steep as c become larger.



2^{nd} Comparison: ϕ = 5° and ϕ = -5°, c = 1:
The two curves are slightly out of sync.



3^{rd} Comparison: ϕ = 10° and ϕ = -10°, c = 1:
The curves become more and more out of sync as the difference of the input orientations become larger.



2. Covariance Matrix

Uses the value of population response at the decision boundary ($\phi = 0$) to compute:

$$C_{ii} = f_i(c, 0)$$

```
function cov = getCov(c, n, k, fmax)
% Get the covariance matrix
% No correlation noise taken into account
% 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
```

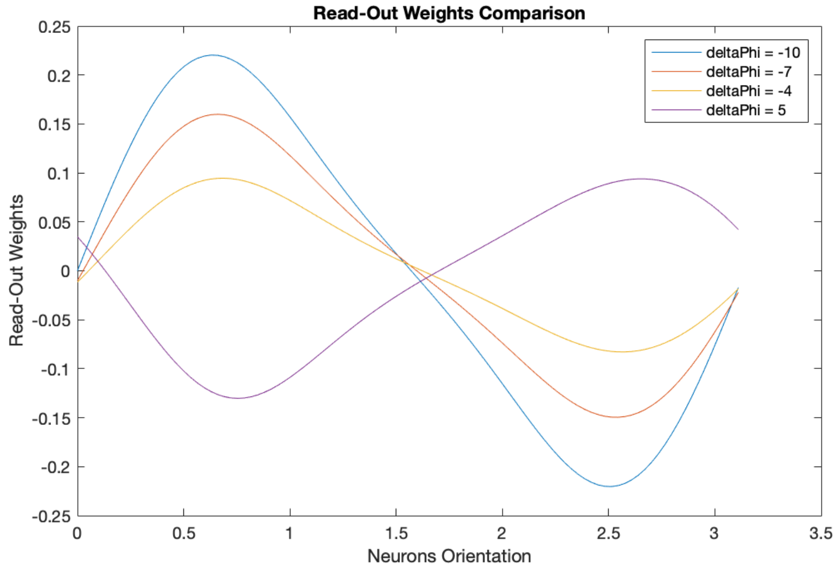
3. Read-Out Weights

I use the following equation to calculate the read-out weights vector:

$$w = C^{-1}[f_i(c, \phi_1) - f_i(c, \phi_2)]$$

```
function w = getW(orient1, orient2, c, n, k, fmax)
% Get the read-out weights
% No correlation noise taken into account
% 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
```

Now I fix c = 1, vary $\Delta\phi$ and plot w. From the following graph we see that the |w| become smaller as $|\Delta\phi|$ becomes smaller. As $\Delta\phi$ changes sign the weights also change their signs.



4. Decision Variable

The following function generates the sequence of d1's and d2's on some given number of trials, where $d = w^T r$ and r is the Poisson noised spike rates based on the population responses.

```
function [d1, d2] = getD(orient1, orient2, c, n, k, fmax, tnum)
% Simulates for given number of trials(tnum)
% Get the decision variables d1 and d2 sequences corresponding to the 2 given
orientations
% No correlation noise taken into account
% Diagonal Poisson variance on the covariance matrix
    d1 = zeros(1, tnum);
    d2 = zeros(1, tnum);
    f1 = getf(orient1, c, n, k, fmax);
    r1 = poissrnd(repmat(f1, tnum, 1), [tnum, n]);
    f2 = getf(orient2, c, n, k, fmax);
    r2 = poissrnd(repmat(f2, tnum, 1), [tnum, n]);

    w = getW(orient1, orient2 - orient1, c, n, k, fmax);

    for i = 1 : tnum
        d1(1, i) = w * r1(i, :).';
        d2(1, i) = w * r2(i, :).';
    end
end
```

5. Computing d'

Fix $\phi_1 = 2^o$ and $\phi_2 = -2^o$, c = 1, simulate 1000 trials, and I got 883 d1's that are positive (correct). The rate is about 88.3%. Now the following is the function to calculate d' given by:

$$d' = \frac{\overline{d_1} - \overline{d_2}}{\sqrt{var d_1 + var d_2}}$$

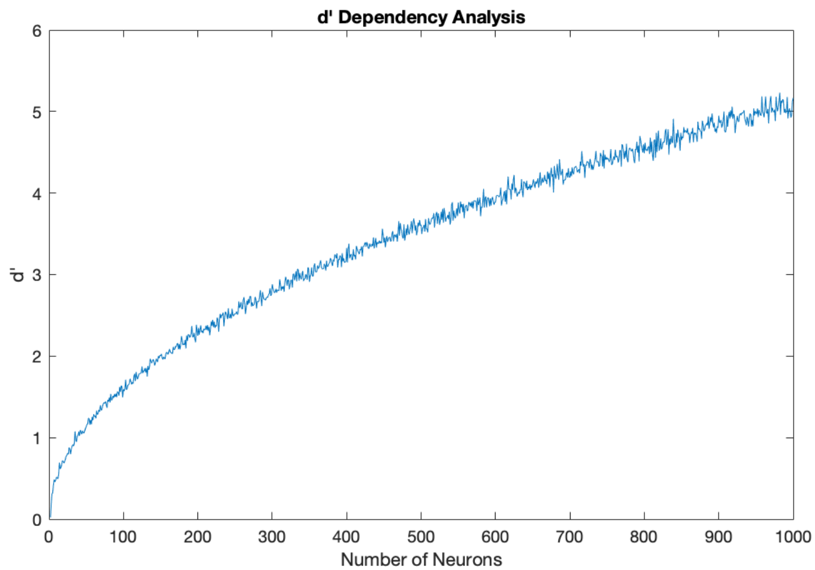
```
function dp = getDp(orient1, orient2, c, n, k, fmax, cmax, tnum, dnum)
% Get the percent correct(d') from given number of trials(tnum)
% dnum stands for the weights used
% dnum = 0 for no correlation noise
% dnum = 1 for correlation noised, diagonally noised covariance
% else we use the optimal weights
    if dnum == 0
        [d1, d2] = getD(orient1, orient2, c, n, k, fmax, tnum);
    elseif dnum == 1
        [d1, d2] = getDnoised(orient1, orient2, c, n, k, fmax, cmax, tnum);
    else
        [d1, d2] = getDopt(orient1, orient2, c, n, k, fmax, cmax, tnum);
    end
    dp = (mean(d1) - mean(d2)) / sqrt(var(d1) + var(d2));
end
```

6. d’ vs n

In this section I want to find the relationship between d’ and n by calculating d’ and varying n from 2 to 1000.

```
function dp = dp_vs_n(orient1, orient2, c, n, cmax, dnum, vark, varfmax)
% Get the sequence of d' based on the given parameters
% Vary n
% dnum stands for the weights used
% dnum = 0 for no correlation noise
% dnum = 1 for correlation noised, diagonally noised covariance
% else we use the optimal weights
    meank = 1;
    meanfmax = 20;
    if vark == 0
        k = ones(1, length(n)) * meank;
    else
        fanok = vark / meank;
        k = gamrnd(meank / fanok, fanok, [1, length(n)]);
    end
    if varfmax == 0
        fmax = ones(1, length(n)) * meanfmax;
    else
        fanofmax = varfmax / meanfmax;
        fmax = gamrnd(meanfmax / fanofmax, fanofmax, [1, length(n)]);
    end

    tnum = 1000;        % # of trials
    dp = zeros(1, length(n));
    for i = 1 : length(n)
        dp(1, i) = getDp(orient1, orient2, c, n(1, i), k(1, i), fmax(1, i), cmax, 
tnum, dnum);
    end
end
```

In this case I fix the variance of both f_{max} and κ 0, and dnum = 0. I also fix $\phi_1 = 2^o$ and $\phi_2 = -2^o$, c = 1, and simulate 1000 trials. From the plot we can see that d' increases as the number of neurons increases.

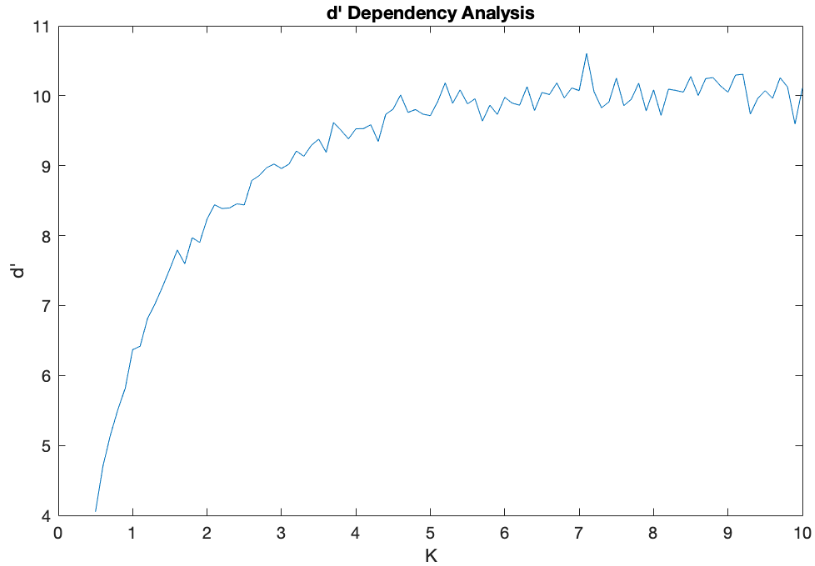


7. d' vs κ

In this section I want to find the relationship between d' and κ by calculating d' and varying κ from 0.5 to 10.

```
function dp = dp_vs_k(orient1, orient2, k)
% Get the sequence of d' based on input sequence k
% No correlation noise taken into account
% Diagonally Poisson noised covariance
    fmax = 20;
    n = 100;
    c = 1;
    tnum = 1000;
    dnum = 0;
    cmax = 0;
    dp = zeros(1, length(k));
    for i = 1 : length(k)
        dp(1, i) = getDp(orient1, orient2, c, n, k(1, i), fmax, cmax, tnum, dnum);
    end
end
```

I fixed n = 100, $f_{max} = 20$, c = 1, $\phi_1 = 8^o$, $\phi_2 = -8^o$, and simulated for 1000 trials for each κ. We see that d' becomes asymptotic as κ increases.



PART II: Limited-Range Noise Correlation

In this part I repeat the analysis including limited-range noise correlations such that two neurons are correlated depending on the difference in their preferred orientations:

$$c_{ij} = c_{max} \exp\left(\frac{-|\phi_i - \phi_j|}{\tau}\right)$$

where $\tau = 0.5$. I also keep $\kappa = 1$ and $f_{max} = 20$ fixed.

1. Correlation Matrix

```
function cor = getCor(n, cmax)
% Get the correlation matrix of n neurons
    tao = 0.5;
    neuron = (0 : pi / n : pi - pi / n);
    cor = zeros(n, n);
    for i = 1 : n
        for j = 1 : n
            cor(i, j) = cmax * exp(-abs(neuron(i) - neuron(j)) / tao);
        end
    end
end
```

2. Covariance Matrix

I calculate the covariance matrix using:

$$C_{ij} = c_{ij}\sqrt{f_i(c,0) f_j(c,0)}$$

I have the following 2 functions to compute the covariance matrices including noise correlations, the first one only contains diagonal elements while the second one is the full covariance matrix.

```
function cov = getCovNoised(cmax, c, n, k, fmax)
% Get the covariance matrix
% Limited-range noise correlations
% Diagonal Poisson variance
```

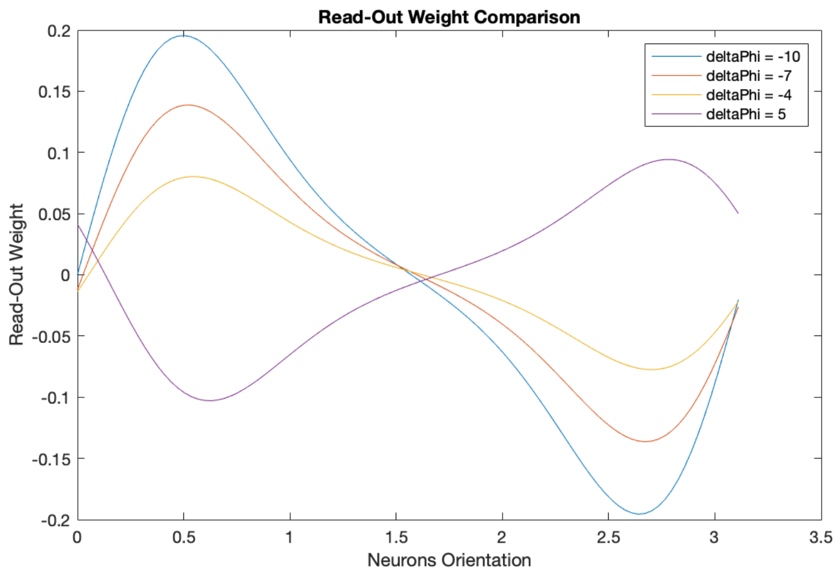
```
    cov = zeros(n, n);
    cor = getCor(n, cmax);
    f = getf(0, c, n, k, fmax);
    for i = 1 : n
        cov(i, i) = cor(i, i) * sqrt(f(1, i) * f(1, i));
    end
end

function cov = getCovOpt(cmax, c, n, k, fmax)
% Get the covariance matrix
% Limited-range noise correlations
% Optimal covariance, poisson noised
    cov = zeros(n, n);
    cor = getCor(n, cmax);
    f = getf(0, c, n, k, fmax);
    for i = 1 : n
        for j = 1 : n
            cov(i, j) = cor(i, j) * sqrt(f(1, i) * f(1, j));
        end
    end
end
```

3. Read-Out Weights

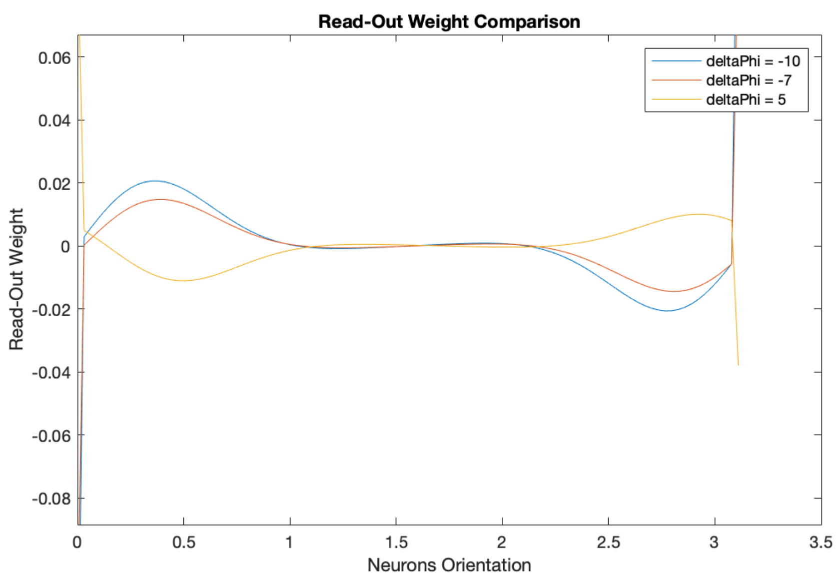
With the above 2 covariance matrices, we get the following 2 functions to calculate the corresponding read-out weights.

```
function w = getWnoised(orient1, orient2, c, n, k, fmax, cmax)
% Get the read-out weights
% Limited-range noise correlations
% Diagonal Poisson variance on the covariance matrix
    w = zeros(1, n);
    neuron = (0 : pi / n : pi - pi / n);    % Equispaced neurons of size n
    cov = getCovNoised(cmax, 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

function wopt = getWopt(orient1, orient2, c, n, k, fmax, cmax)
% Get the read-out weights
% Limited-range noise correlations
% Optimal covariance, poisson noised
    cov = getCovOpt(cmax, c, n, k, fmax);
    f1 = getf(orient1, c, n, k, fmax);
    f2 = getf(orient2, c, n, k, fmax);
    diff = f1 - f2;
    wopt = cov \ diff.';      % inv(cov) * diff
    wopt = wopt.';
end
```

Now I fix c = 0.1, c_{max} = 0.3, then vary $\Delta\phi$ and plot w as in Part I. First we have the weights using only diagonal covariance matrix, including noise correlations. We see that the behavior of w is about the same as in Part I.



Now we plot the optimal weights. The behavior is rather different than the previous two. |w| are scaled to much smaller values mostly. At the endpoints (0 and π) it becomes large and at around $\pi/2$ it becomes very close to 0. As $\Delta\phi$ changes sign the weights also change their signs, similar to the previous two.



4. Decision Variable

Similar as in Part I we compute the decision variable but this time we draw r from a correlated Gaussian distribution. Similarly, we have the following 2 functions.

```
function [d1, d2] = getDnoised(orient1, orient2, c, n, k, fmax, cmax, tnum)
% Simulates for given number of trials(tnum)
% Get the decision variables d1 and d2 sequences corresponding to the 2 given
orientations
% Limited-range noise correlations
```

```
% Diagonal Poisson variance on the covariance matrix
    d1 = zeros(1, tnum);
    d2 = zeros(1, tnum);

    f1 = getf(orient1, c, n, k, fmax);
    f2 = getf(orient2, c, n, k, fmax);
    w = getWnoised(orient1, orient2, c, n, k, fmax, cmax);
    cov = getCovNoised(cmax, c, n, k, fmax);
    r1 = mvnrnd(repmat(f1, tnum, 1), cov);
    r2 = mvnrnd(repmat(f2, tnum, 1), cov);

    for i = 1 : tnum
        d1(1, i) = w * r1(i, :).';
        d2(1, i) = w * r2(i, :).';
    end
end

function [d1, d2] = getDopt(orient1, orient2, c, n, k, fmax, cmax, tnum)
% Simulates for given number of trials(tnum)
% Get the decision variables d1 and d2 sequences corresponding to the 2 given
orientations
% Limited-range noise correlations
% Optimal covariance, poisson noised
    d1 = zeros(1, tnum);
    d2 = zeros(1, tnum);

    f1 = getf(orient1, c, n, k, fmax);
    f2 = getf(orient2, c, n, k, fmax);
    w = getWopt(orient1, orient2, c, n, k, fmax, cmax);
    cov = getCovOpt(cmax, c, n, k, fmax);
    r1 = mvnrnd(repmat(f1, tnum, 1), cov);
    r2 = mvnrnd(repmat(f2, tnum, 1), cov);

    for i = 1 : tnum
        d1(1, i) = w * r1(i, :).';
        d2(1, i) = w * r2(i, :).';
    end
end
```

5. Computing d'

Fix $\phi_1 = 2^o$ and $\phi_2 = -2^o$, c = 0.1, simulate 1000 trials. Using diagonal covariance matrix, I got 642 d1's that are positive (64.2%). Using optimal covariance matrix, I got 456 positive d1's (45.6%). Now I use the same function as Part I to calculate d'. We only need to change the parameter dnum to include the noise correlation.

```
function dp = getDp(orient1, orient2, c, n, k, fmax, cmax, tnum, dnum)
% Get the percent correct(d') from given number of trials(tnum)
% dnum stands for the weights used
% dnum = 0 for no correlation noise
% dnum = 1 for correlation noised, diagonally noised covariance
% else we use the optimal weights
    if dnum == 0
        [d1, d2] = getD(orient1, orient2, c, n, k, fmax, tnum);
    elseif dnum == 1
```

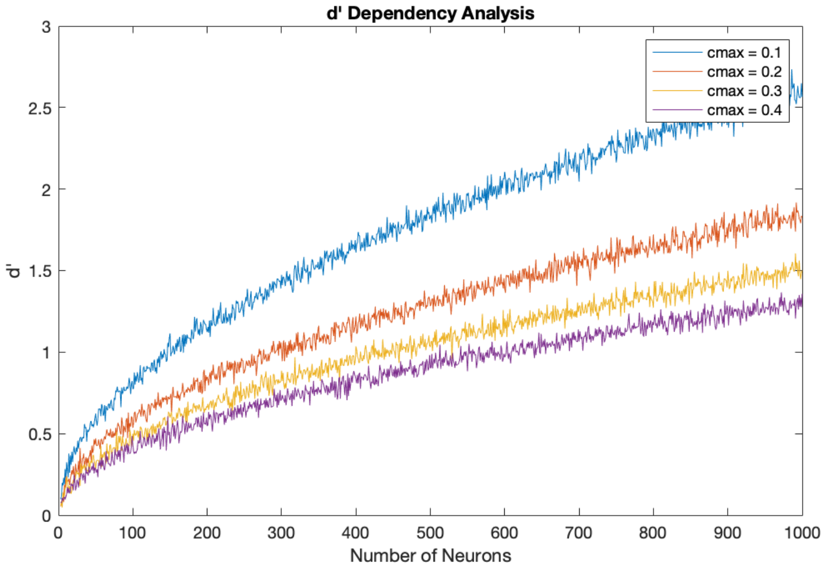
```
        [d1, d2] = getDnoised(orient1, orient2, c, n, k, fmax, cmax, tnum);
    else
        [d1, d2] = getDopt(orient1, orient2, c, n, k, fmax, cmax, tnum);
    end
    dp = (mean(d1) - mean(d2)) / sqrt(var(d1) + var(d2));
end
```

6. d' vs. n Using Diagonal Covariance

I use the exact same function as in Part I.

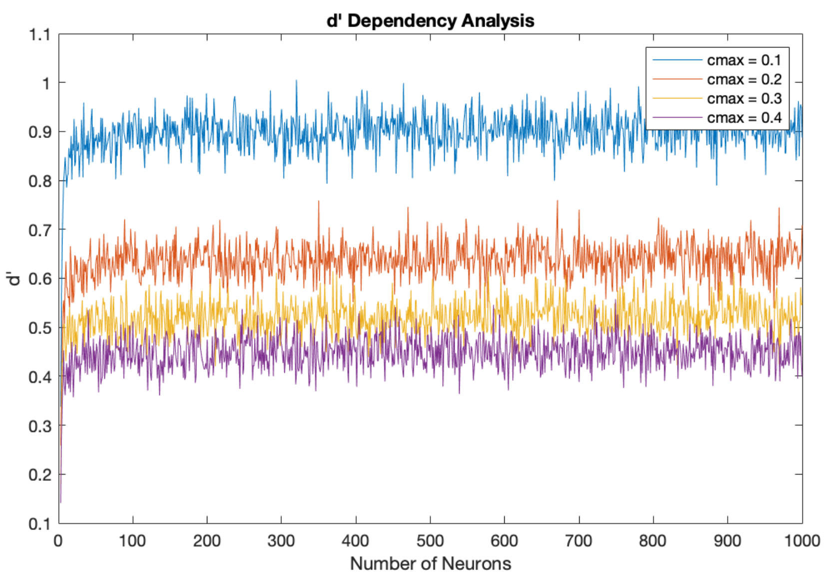
```
function dp = dp_vs_n(orient1, orient2, c, n, cmax, dnum, vark, varfmax)
% Get the sequence of d' based on the given parameters
% Vary n
% dnum stands for the weights used
% dnum = 0 for no correlation noise
% dnum = 1 for correlation noised, diagonally noised covariance
% else we use the optimal weights
    meank = 1;
    meanfmax = 20;
    if vark == 0
        k = ones(1, length(n)) * meank;
    else
        fanok = vark / meank;
        k = gamrnd(meank / fanok, fanok, [1, length(n)]);
    end
    if varfmax == 0
        fmax = ones(1, length(n)) * meanfmax;
    else
        fanofmax = varfmax / meanfmax;
        fmax = gamrnd(meanfmax / fanofmax, fanofmax, [1, length(n)]);
    end

    tnum = 1000;        % # of trials
    dp = zeros(1, length(n));
    for i = 1 : length(n)
        dp(1, i) = getDp(orient1, orient2, c, n(1, i), k(1, i), fmax(1, i), cmax,
tnum, dnum);
    end
end
```

Fix $\phi_1 = 2^o$ and $\phi_2 = -2^o$, c = 0.1, vary n from 2 to 1000 with 4 different values of c_{max} (0.1, 0.2, 0.3, 0.4). We see that the value of d' decreases as c_{max} increases. But they all tend to be asymptotic, unlike when there's no noise correlation in Part I.



7. d' vs. n Using Optimal Weights/Covariance

I use the exact same function as above. Fix $\phi_1 = 8^o$ and $\phi_2 = -8^o$, c = 0.1, vary n from 2 to 1000 with 4 different values of c_{max} (0.1, 0.2, 0.3, 0.4). Using the optimal weights, we are able to control d' such that it does not increase as n increases, but rather stays around a specific value. Also, as c_{max} increases, |d'| becomes smaller.



PART III: κ and f_{max} from Gamma Distribution

In this part I draw κ and f_{max} from 'reasonable' distributions and repeat the analysis. I also study how d' depend on the variances of these distributions.

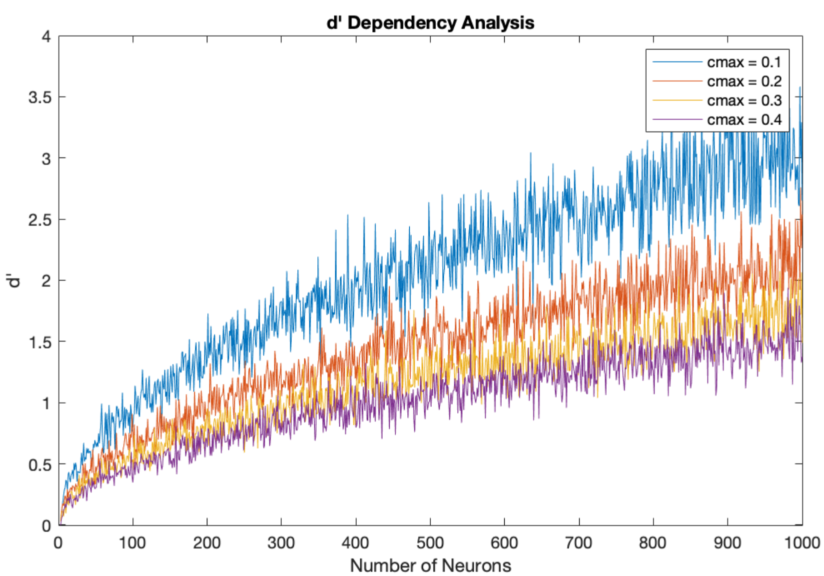
1. d' vs n Using Diagonal Covariance

I use the same function as above. Simply modify the mean of κ to 1.5.

```
function dp = dp_vs_n(orient1, orient2, c, n, cmax, dnum, vark, varfmax)
% Get the sequence of d' based on the given parameters
```

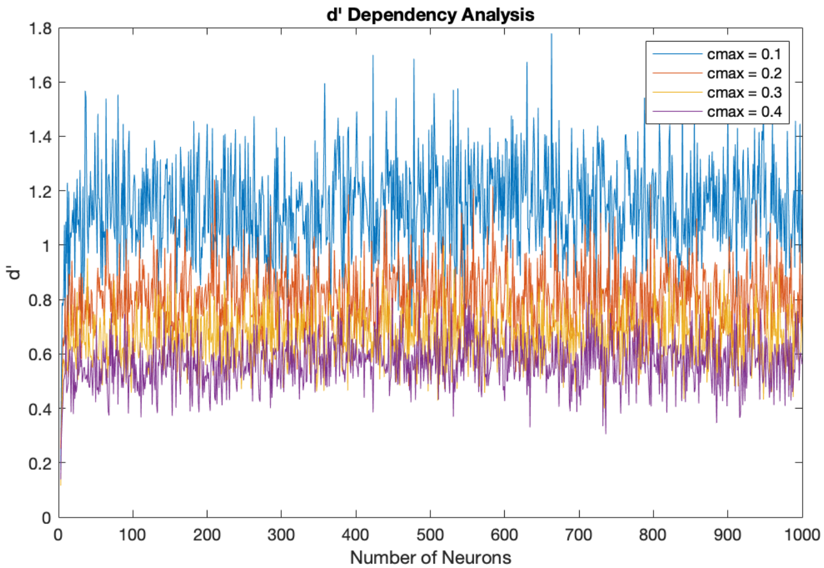
```
% Vary n
% dnum stands for the weights used
% dnum = 0 for no correlation noise
% dnum = 1 for correlation noised, diagonally noised covariance
% else we use the optimal weights
    meank = 1.5;
    meanfmax = 20;
    if vark == 0
        k = ones(1, length(n)) * meank;
    else
        fanok = vark / meank;
        k = gamrnd(meank / fanok, fanok, [1, length(n)]);
    end
    if varfmax == 0
        fmax = ones(1, length(n)) * meanfmax;
    else
        fanofmax = varfmax / meanfmax;
        fmax = gamrnd(meanfmax / fanofmax, fanofmax, [1, length(n)]);
    end

    tnum = 1000;        % # of trials
    dp = zeros(1, length(n));
    for i = 1 : length(n)
        dp(1, i) = getDp(orient1, orient2, c, n(1, i), k(1, i), fmax(1, i), cmax,
tnum, dnum);
    end
end
```

Fix $\phi_1 = 2^o$ and $\phi_2 = -2^o$, c = 0.1, dnum = 1, vary n from 2 to 1000 with 4 different values of c_{max} (0.1, 0.2, 0.3, 0.4). I draw κ from a Gamma distribution with mean = 1.5 and variance = 0.1. Similarly, I draw f_{max} from a Gamma distribution with mean = 20 and variance = 2. From the plot we can see that the behavior is the same as in Part II, but more sparse (noisy) as expected.



2. d' vs n Using Optimal Weights

I use the exact same function as above. Fix $\phi_1 = 8^o$ and $\phi_2 = -8^o$, c = 0.1, vary n from 2 to 1000 with 4 different values of c_{max} (0.1, 0.2, 0.3, 0.4). I draw κ from a Gamma distribution with mean = 1.5 and variance = 0.1. Similarly, I draw f_{max} from a Gamma distribution with mean = 20 and variance = 2. The behavior of the plot, as expected, becomes noisier, but nothing more different than the plot in Part II.

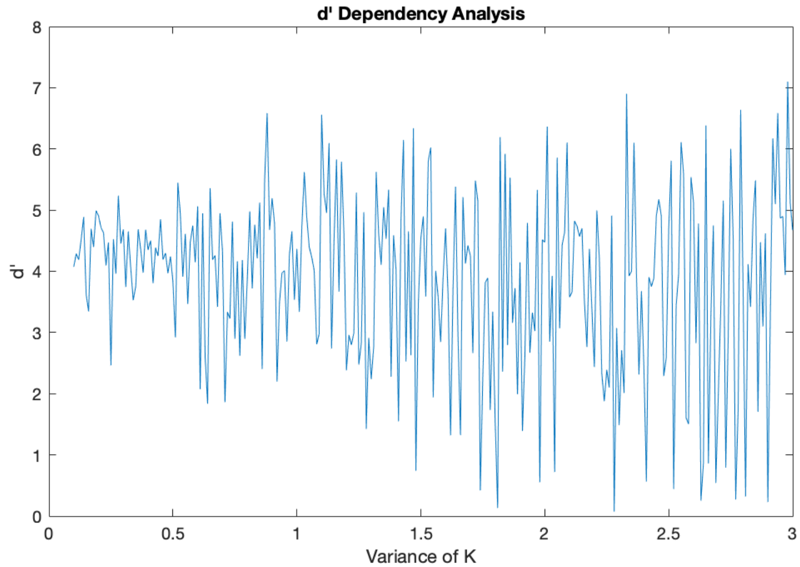


3. d' vs Variance of κ

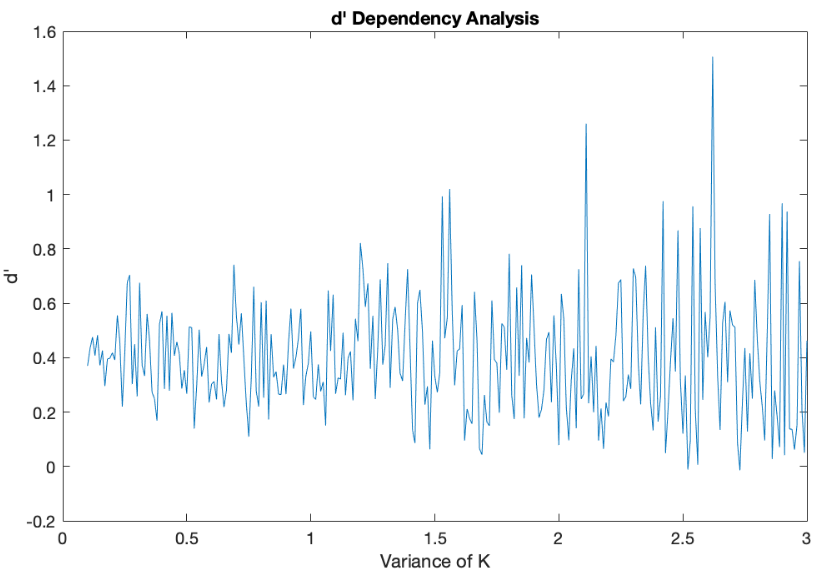
```
function dp = dp_vs_vark(orient1, orient2, vark, dnum)
% Get the sequence of d' based on the given variance sequence of k
% dnum stands for the weights used
% dnum = 0 for no correlation noise
% dnum = 1 for correlation noised, diagonally noised covariance
% else we use the optimal weights
    mean = ones(1, length(vark)) * 1.5;
    fano = vark ./ mean;
    k = gamrnd(mean ./ fano, fano);
    fmax = 20;
    n = 1000;
    c = 0.1;
    tnum = 1000;
    cmax = 0.3;
    dp = zeros(1, length(k));
    for i = 1 : length(k)
        dp(1, i) = getDp(orient1, orient2, c, n, k(1, i), fmax, cmax, tnum, dnum);
    end
end
```

In the following analysis I fix $\phi_1 = 5^o$ and $\phi_2 = -5^o$, c = 0.1, n = 1000, f_{max} = 20, c_{max} = 0.3, mean of κ = 1.5, and vary the variance of κ. I take the variance from 0.1 to 3 (FF from 0.06 to 2). The variance of d' increases as the variance of κ increases in both of the plots. However, using the optimal weights, d' varies much smaller than using diagonal covariance.

Diagonal Covariance



Optimal Covariance



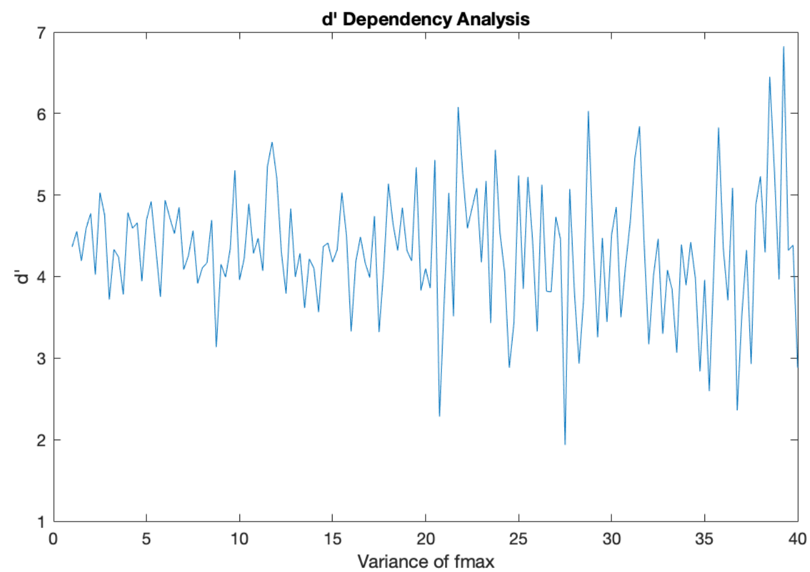
4. d' vs Variance of f_{max}

```
function dp = dp_vs_varfmax(orient1, orient2, varfmax, dnum)
% Get the sequence of d' based on the given variance sequence of fmax
% dnum stands for the weights used
% dnum = 0 for no correlation noise
% dnum = 1 for correlation noised, diagonally noised covariance
% else we use the optimal weights
    mean = ones(1, length(varfmax)) * 20;
    fano = varfmax ./ mean;
    fmax = gamrnd(mean ./ fano, fano);
    k = 1.5;
    n = 1000;
    c = 0.1;
```

```
    tnum = 1000;
    cmax = 0.3;
    dp = zeros(1, length(fmax));
    for i = 1 : length(fmax)
        dp(1, i) = getDp(orient1, orient2, c, n, k, fmax(1, i), cmax, tnum, dnum);
    end
end
```

In the following analysis I fix $\phi_1 = 5^o$ and $\phi_2 = -5^o$, c = 0.1, n = 1000, $\kappa = 1.5$, $c_{max} = 0.3$, mean of f_{max} = 20, and vary the variance of f_{max}. I take the variance from 1 to 40 (FF from 0.05 to 2). We can see that the variance of d' increases as the variance of f_{max} increases. However, similar as for κ, using the optimal weights yields a smaller variance of d' than using diagonal covariance. But the difference is not much seeing from the plots.

Diagonal Covariance



Optimal Covariance