CSC247/547 Assignment 2

Andrii Osipa

October 2017

Problem 1

Binomial distribution of spikes. For this case we have probability of spike p for each time bin. We have 1Δ time bins for 1s and therefore if we want to achieve spike rate r then expected value of $\mathbb{B}(\frac{1}{\Delta}, p)$ be equal r. And from this we can easily compute that $p = \Delta r$.

Exponential distribution for ISI. PDF for exponential distribution is $f(x) = \lambda \exp(-\lambda x)$. Mean for this distribution is λ^{-1}. We want to simulate ISI that correspond to some fixed spiking rate r. As $r = \frac{1}{ISI}$, then we may assume that mean spiking rate for this distribution is $\frac{1}{\lambda^{-1}} = \lambda$. Therefore if we want to simulate ISI for given spiking rate we take $\lambda = r$.

Poisson distribution. Assuming that we want to get a random value that correspond to the number of spikes during 1s then parameter for Poisson distribution will be $\lambda = r$.

Comparing results. Lets fix the following parameters for the simulation: $r = 20$ $spikes/s$ and time period of $30000s$. Those parameters will give good and continuous result as number of trials is quite big.

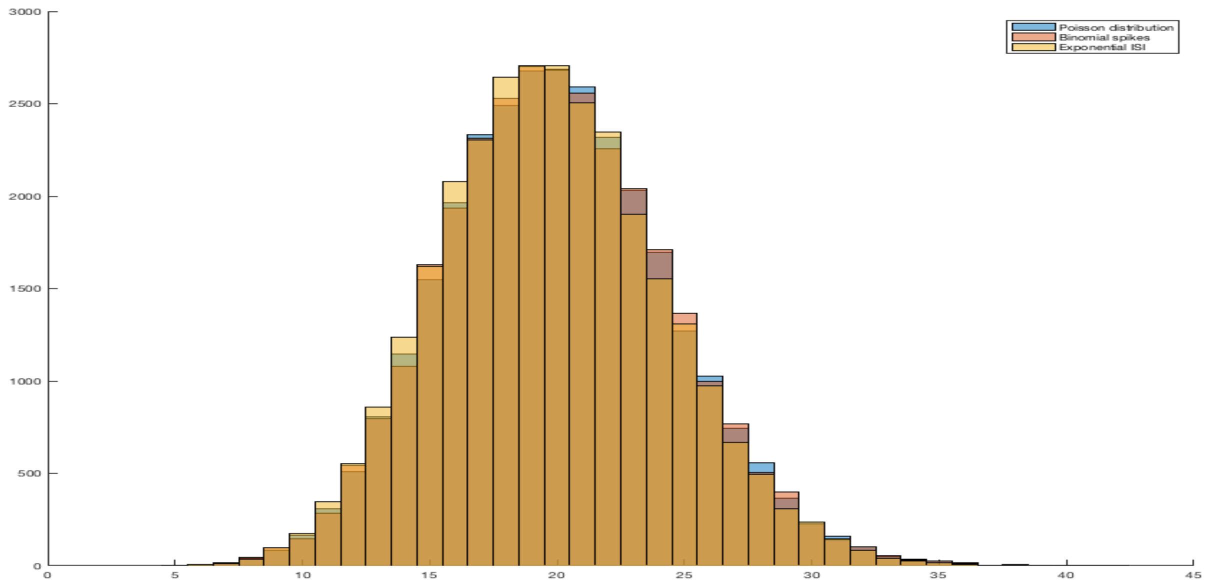


Figure 1: Histogram of spike counts for all cases

As from this plot we can see that all these ways of simulation of spike trail give almost same results. This is easier to see from density plot on the Figure 2. Therefore we may assume that all these ways to simulate spike counts are equivalent. If simulation is done for smaller time period, results have more difference, which is logical.

Code

Code for exponential ISI model:

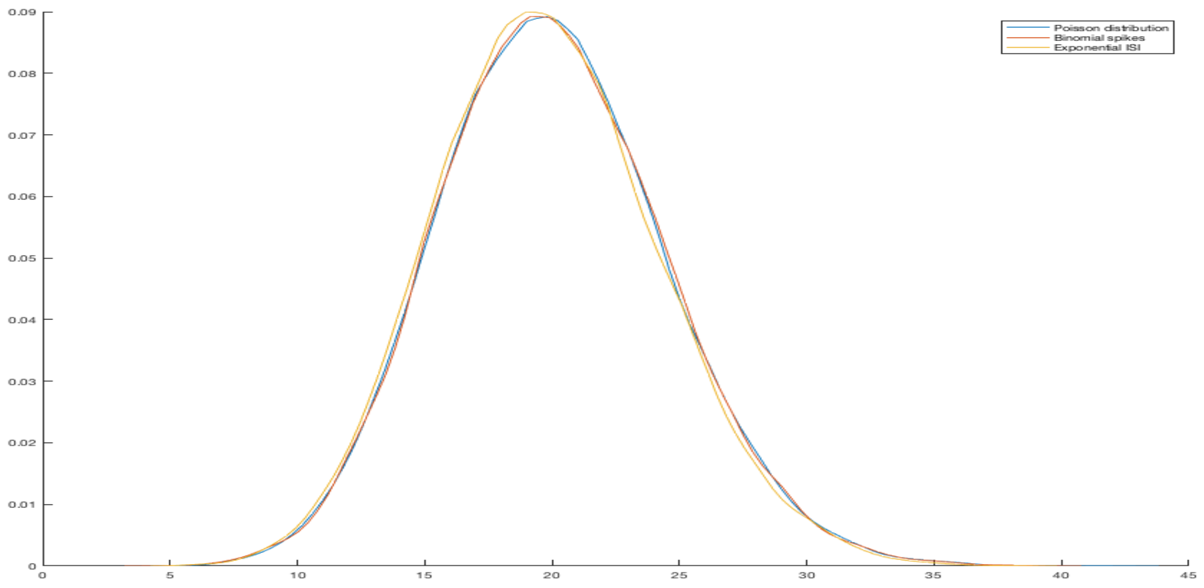


Figure 2: Density of spike counts for all cases

```
function [ spikes ] = exp_isi( time_limit , sp_rate )
% genrate spikes with bernoulli distribution with
% given spiking rate for given time interval.

lambda = 1/sp_rate; % mean ISI
time = 0;
delta = 0.001;
spikes = zeros(1, length(0:delta:time_limit));
while time < time_limit
    isi = exprnd(lambda,1,1); % generate next ISI
    time = time + isi;
    if time < time_limit
        spikes(ceil(time/delta)) = 1; % fill appropriate bin
    end
end
end
```

Code for binomial spikes model:

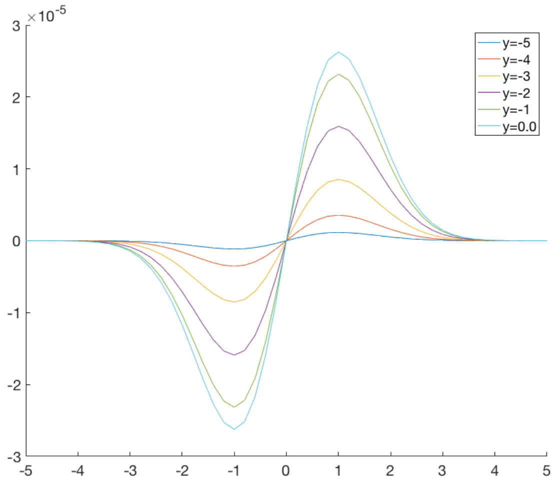
```
function [ spikes ] = bernoulli_spikes( time_limit , sp_rate )
delta = 0.001;
N = time_limit/delta;
P = sp_rate * delta; %probability of spike to occur in any time bin
spikes = binornd(1, P, 1, N);
end
```

Code for comparison of the two models with Poisson model:

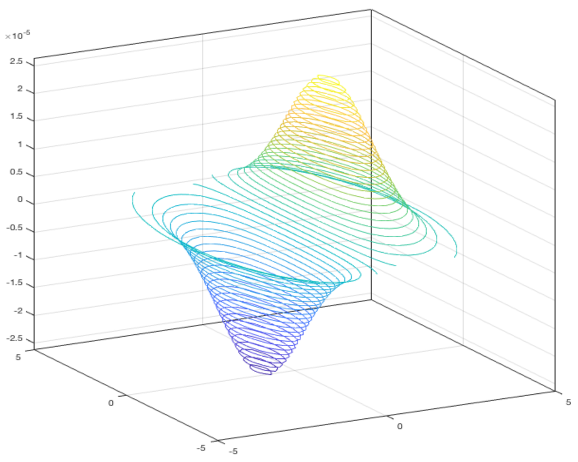
```
time_bound = 30000;
delta = 0.001;
sp_rate = 20;
b_spikes = bernoulli_spikes(time_bound, sp_rate);
e_spikes = exp_isi(time_bound, sp_rate);
b_sp_counts = zeros(1, time_bound);
e_sp_counts = zeros(1, time_bound);

disp(length(b_spikes));
disp(length(e_spikes));

for i=1:time_bound
   for idx = (1+(i-1)/delta) : (i/delta)
        if b_spikes(idx) == 1
            b_sp_counts(i) = b_sp_counts(i) + 1;
        end
        if e_spikes(idx) == 1
            e_sp_counts(i) = e_sp_counts(i) + 1;
        end
   end
end

% generate spikes from Possion model, taking it as
% union of many 1s trials.
p_sp_counts = poissrnd(sp_rate, 1, time_bound);

figure
hold on
ksdensity(p_sp_counts);
ksdensity(b_sp_counts);
ksdensity(e_sp_counts);
legend("Poisson distribution", "Binomial spikes", "Exponential ISI");
hold off

figure
hold on
histogram(p_sp_counts, 'FaceAlpha', 0.5);
histogram(b_sp_counts,'FaceAlpha', 0.5);
histogram(e_sp_counts, 'FaceAlpha', 0.5);
legend("Poisson distribution", "Binomial spikes", "Exponential ISI");
hold off
```

Problem 2

(a) For the following $RF(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}\right) cos(kx - \phi)$ with $phi = \pi/2$ and $k = \frac{1}{0.56} * \frac{180}{\pi}$ we have plot:



(a) RF slices for fixed xs



(b) 3D plot for RF

Note: RFs were scaled in a simple cell by a constant $c = \left(\sum_{x,y} |RF(x,y)|\right)^{-1}$. This is done to bake them comparable, as values of RF in case $\phi = 0$ and $\phi = \pi/2$ are very different. This scaling seems reasonable, as with it maximum response of the cell on image with values in $[-1, 1]$ will be same and will not depend on RF parameter ϕ. This scaling does not affects anything at all, as there many other constants appear later and number of constant may be reduced if needed.

Code:

```
function value = RF( x, y, sx, sy, phi, k )
% k in rad
value = (2*pi*sx*sy)^(-1)*exp(-x.^2/(2*sx^2)-y.^2/(2*sy^2))*cos(k.*x./180.*pi-phi);
end
```

(b) Consider simple neuron model such that $out(\text{image}) = c * \text{Poisson}(\max(0, \sum_{x,y} RF(x,y) * \text{image}(x,y))^2)$, where c is some constant to be established later. Then mean spike rate for each image is $c * \max(0, \sum_{x,y} RF(x,y) * \text{image}(x,y))^2)$. Plot for the response to $\text{image}(x,y) = sin(kx - \alpha)$ in on the Figure 4. Constant c was fitted accordingly.

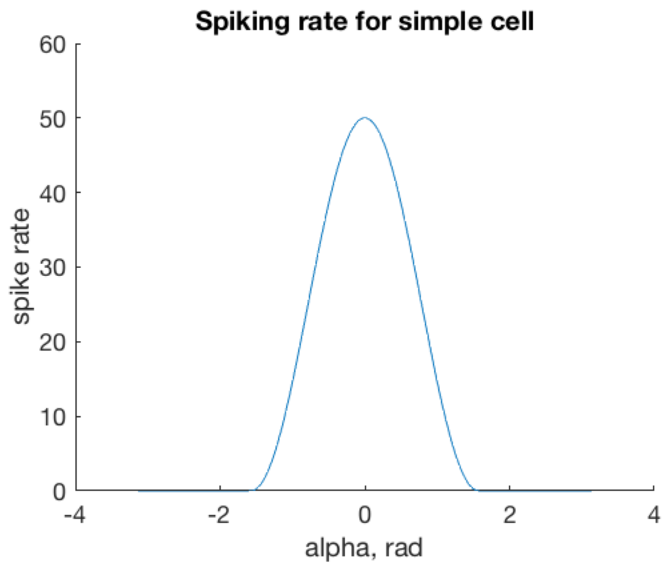


Figure 4: Mean spiking rate for simple cell

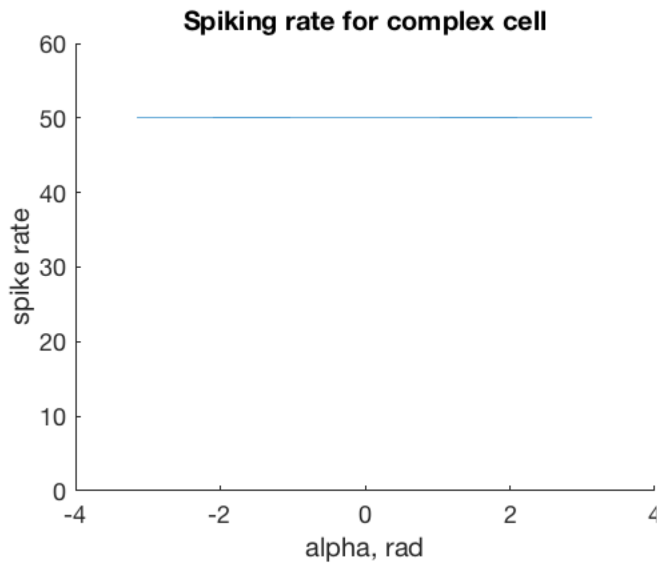
Code:

```
function [ out ] = simple_cell( image, range, phi, nonlinearity, const )
%Assuming range is values of X(Y has same)
%Nonlinearity is some function, e.g. x^2

sx = 1;
sy = 2;
k = 1/0.56/180*pi; %convert k to rad

[X,Y] = meshgrid(range);
rf_values = zeros(length(range));
%evaluate RF:
for ix=1:length(range)
    for iy=1:length(range)
        rf_values(ix,iy) = RF( X(ix, iy), Y(ix, iy), sx, sy, phi, k);
    end
end
%adj constant, based on the values of RF.
adj = sum(sum(abs(rf_values)));

rf_values = rf_values ./ adj;

out = const * nonlinearity(sum(sum(rf_values .* image)));

end
```

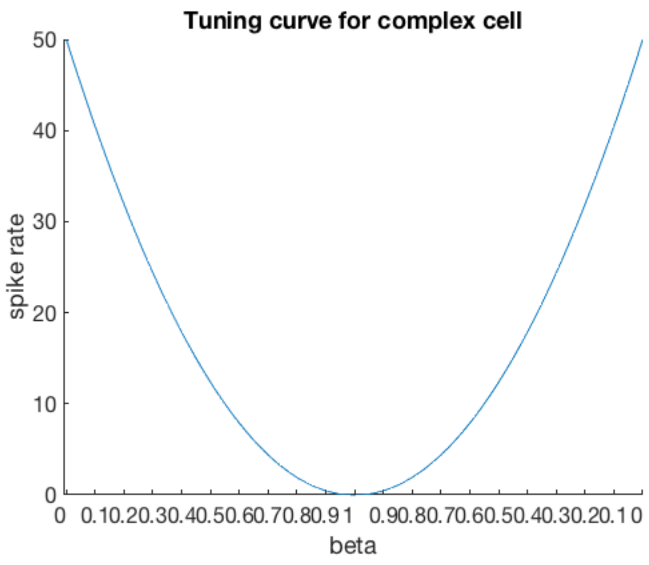
(c) Consider complex cell that has as inputs two simple cells $S_1(\phi = \pi/2)$ and $S_2(\phi = 0)$. Both cells are "fitted", so there exists an image from $I(x, y) = sin(kx - \alpha)$ for some value of parameter α such that mean response of cell on it is 50Hz. Then mean spiking rate for the complex cell is described in the Fig.5(a).

This result is understandable as we just have shifted waves for each simple cell that are similar to Fig.4, but have more then 1 wave, as have different non-linearity function. As shift is exactly $\pi/2$ the sum is almost constant function. Therefore complex cell gives stable firing rate for any vertical grating image.

Now consider image defined with the function $I(x, y) = sin(k((1 - \beta)x + \beta y) - \alpha)$, $\beta \in [0, 1]$. This image is just rotated grating. For $\beta = 0$ it is vertical grating and for $\beta = 1$ it is horizontal grating. Then we have the following response mean spiking rate from complex cell if $\alpha = 0$ and β is changed $0 \to 1 \to 0$.



(a) Mean spiking rate for complex cell



(b) Tuning curve for complex cell

Figure 5: Complex cell results

From the Fig.5(b) we see that complex cell detects vertical grating itself, does not depend on the phase, but it does not fires on horizontal grating.

Code:

```
function [ out ] = complex_cell( image, range )
phi1 = pi/2;
phi2 = 0;

nonlinearity = @(x) x^2;

adj_1 = 50/4.6790e-07;
adj_2 = 50/0.9999;

out1 = simple_cell(image, range, phi1, nonlinearity, adj_1);
out2 = simple_cell(image, range, phi2, nonlinearity, adj_2);

out = out1 + out2;

end
```

(d) Let's take 10000 random noise images, where noise is from Normal distribution with 0 mean and variance 1. The responses aster scaling were as on the Fig.6.

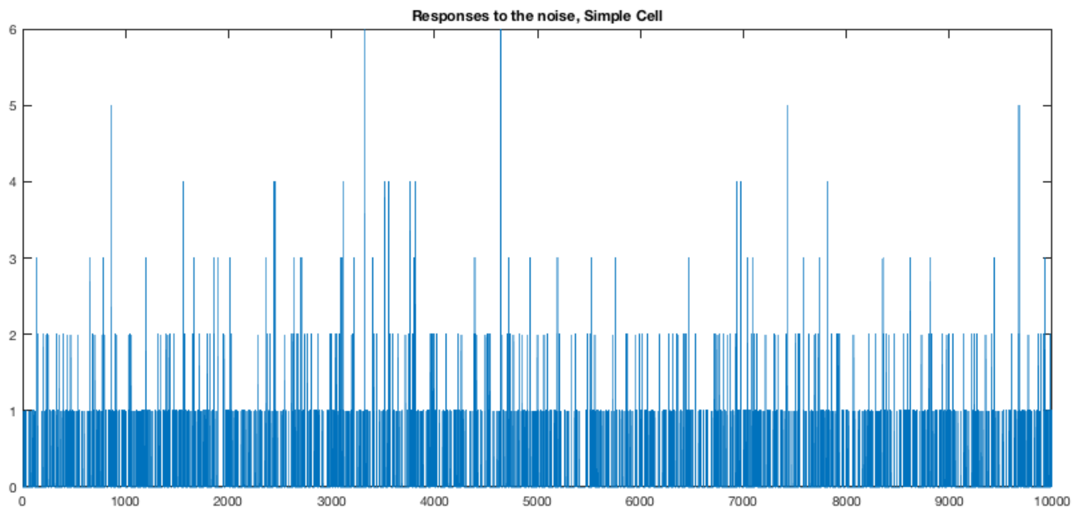


Figure 6: Responses to the noise from SC

After computing average from every image with 1 or more spikes we have the following estimation which has same shape as real RF. Correlation coefficient between real and estimation is 0.73. Therefore estimation is quite good.

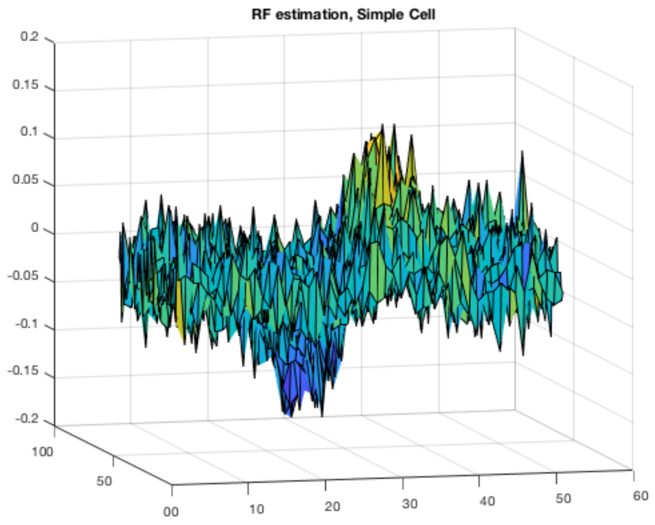


Figure 7: Estimation of RF for SC

Doing same for complex cell does not give any results for RF estimation, despite pattern of responses seems very similar to the one of the simple cell. This is understandable as from the nature of the output of the complex cell there is no unique image that will maximize response. Computing average image gives no significant result, image is still seems as a random one. This just confirmed known fact about the RF of the complex cell - it is not obvious as for the simple cell. Also nature of the feature this cell "detects" confirms that RF has different meaning compared to the case of simple cell. The results for reverse correlation are in Fig.8 and Fig.9.

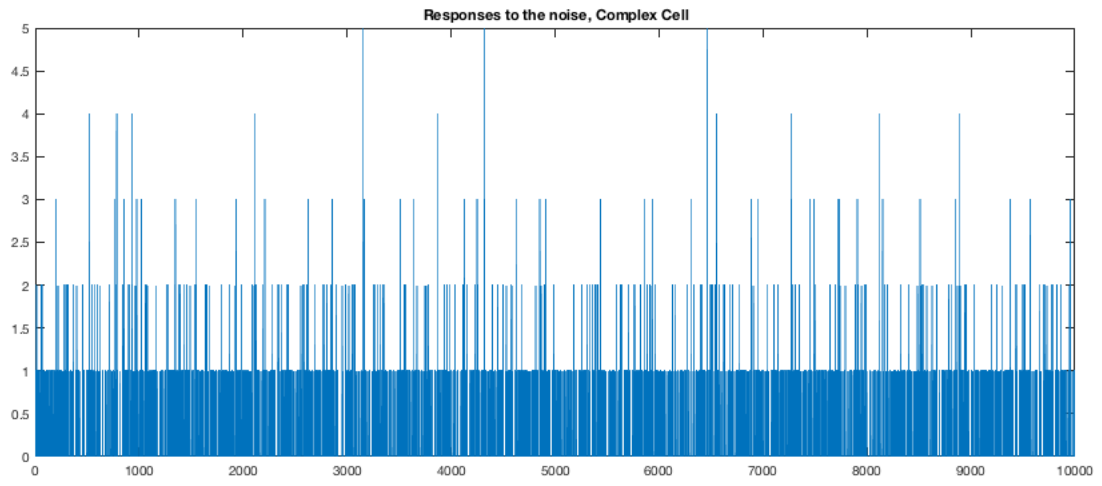


Figure 8: Responses to the noise from CC

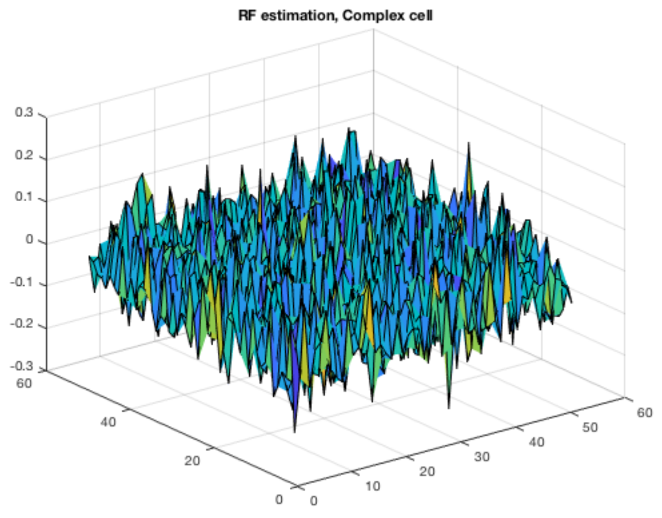
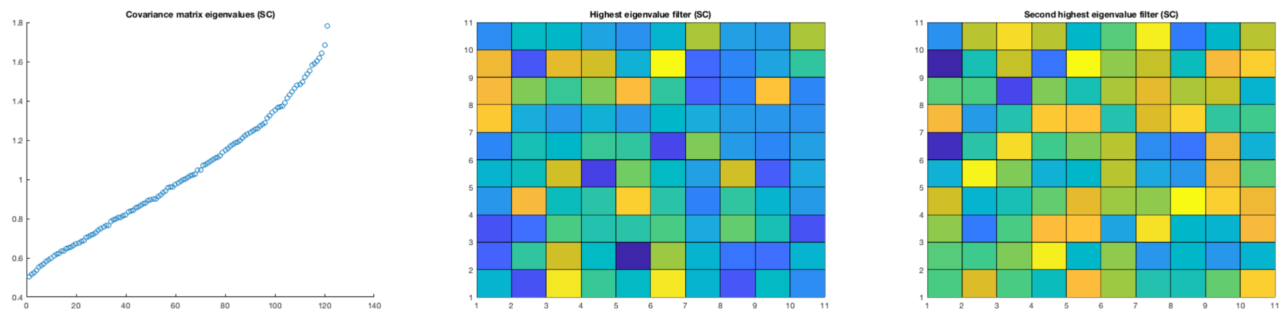


Figure 9: Estimation of RF for CC

(e) STC done with set of 10000 random white noise images of size 11x11 pixels, response to the them is scaled, so the average is 0.2.

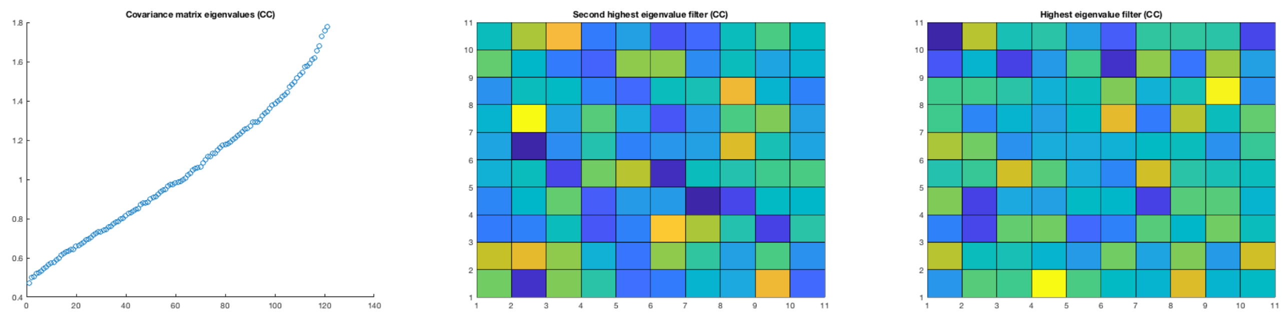
Simple Cell: In most simulations there were a few largest eigenvalues with significant gap between them. Plot of the sorted eigenvalues and the filters corresponding to them are in the Fig.10. In this case, it's often happens that one filter is similar to diagonal grating, the second one has not very "understandable" shape.

Complex Cell: In this case there is no significant gap between biggest eigenvalues, but still there is some gap. Plot of the sorted eigenvalues and the filters corresponding to them are in the Fig.12 and Fig.13. Filters are not very informative in this case. This makes sense, as even RF for complex cell is not "nicely" defined thing.



(a) Responses to the noise from SC (b) Filter for SC (c) Filter for SC

Figure 10: STC results for SC



(a) Responses to the noise from CC (b) Filter for CC (c) Filter for CC

Figure 11: STC results for CC

Code for all simulations for (a) - (e):

```
%Discovering RF
% given parameters
sx = 1;
sy = 2;
k = 1/0.56/180*pi;
phi = pi/2;

nonlinearity = @(x) max(0,x)^2;

range = -5:0.2:5;
[X,Y] = meshgrid(range);
rf_values = zeros(length(range));
for ix=1:length(range)
    for iy=1:length(range)
        rf_values(ix,iy) = RF( X(ix, iy), Y(ix, iy), sx, sy, phi, k);
    end
end
figure
contour3(X,Y,rf_values,20);
title("RF");



%Simple cell simulation

image_fun = @(x,y,alpha) sin(k.*x./180.*pi - alpha);

% number of different phases to simulate
```

```
n_alphas = 200;
out_rates = zeros(1,n_alphas);
alphas = linspace(-pi,pi,n_alphas);
%constant, fitted to have maen spiking rate of 50Hz
adj_c_s = 50/4.6778e-07;
for alpha_idx = 1:n_alphas
    image = zeros(length(range));
    for ix=1:length(range)
        for iy=1:length(range)
            image(ix,iy) = image_fun( X(ix, iy), Y(ix, iy), alphas(alpha_idx));
        end
    end
    out_rates(alpha_idx) = simple_cell(image, range, phi, nonlinearity, adj_c_s);
end

figure
hold on
plot(alphas, out_rates);
set(gca,'fontsize',18);
title("Spiking rate for simple cell");
xlabel("alpha, rad");
ylabel("spike rate");
hold off



%Complex cell simulation

n_alphas = 200;
out_rates = zeros(1,n_alphas);
alphas = linspace(-pi,pi,n_alphas);
for alpha_idx = 1:n_alphas
    image = zeros(length(range));
    for ix=1:length(range)
        for iy=1:length(range)
            image(ix,iy) = image_fun( X(ix, iy), Y(ix, iy), alphas(alpha_idx));
        end
    end
    out_rates(alpha_idx) = complex_cell(image, range);
end

figure
hold on
plot(alphas, out_rates);
ylim([0,60]);
set(gca,'fontsize',18);
xlabel("alpha, rad");
ylabel("spike rate");
title("Spiking rate for complex cell");
hold off



%Tuning curve for complex cell

image_fun_r = @(x,y,alpha,beta) sin(k.*((1-beta).*x./180.*pi + beta.*y/180*pi) - alpha);

n_betas = 200;
out_rates = zeros(1,n_betas);
% grating rotations
```

```
betas = [linspace(0,1,n_betas/2), linspace(1,0,n_betas/2)];
for beta_idx = 1:n_betas
    image = zeros(length(range));
    for ix=1:length(range)
        for iy=1:length(range)
            image(ix,iy) = image_fun_r( X(ix, iy), Y(ix, iy), 0, betas(beta_idx));
        end
    end
    out_rates(beta_idx) = complex_cell(image, range);
end

figure
hold on
plot(out_rates);
set(gca,'XTick',linspace(1, n_betas,21));
set(gca, 'XTickLabel', [linspace(0,1,11), linspace(0.9,0,10)]);
set(gca,'fontsize',16);
xlabel("beta");
ylabel("spike rate");
title("Tuning curve for complex cell");
hold off



%Reverse corellation for simple cell

n_images = 10000;
images = normrnd(0,1,51,51,n_images);
sp_counts_simple = zeros(1,n_images);
%constant to make spikes "noticable, despite smnall values.
adj_const = 1e+6;
for idx=1:n_images
    sp_counts_simple(idx) = poissrnd(adj_const * simple_cell(images(:,:,idx), range, phi, nc
end

figure
sp_counts_simple = round(0.2 .* sp_counts_simple ./ mean(sp_counts_simple));
disp("mean spiking rate SC: " + mean(sp_counts_simple));
plot(sp_counts_simple);
title("Responses to the noise, Simple Cell");

%compute avrage image
average = zeros(51);
count = 0;
for idx=1:n_images
   if sp_counts_simple(idx) >= 1
      average = average + images(:,:,idx);
      count = count + 1;
   end
end

average = average ./ count;
figure
surf(average);
title("RF estimation, Simple Cell");

disp("Correlation matrix with true RF");
disp(corrcoef(reshape(average, 1, 51^2), reshape(rf_values, 1, 51^2)));
```

```
%Reverse corellation for complex cell

n_images = 10000;
images = normrnd(0,3,51,51,n_images);
sp_counts_complex = zeros(1,n_images);
for idx=1:n_images
    sp_counts_complex(idx) = poissrnd( 1e-3 * complex_cell(images(:,:,idx), range));
end

figure
sp_counts_complex = round(0.3 .* sp_counts_complex ./ mean(sp_counts_complex));
disp("Mean spiking rate CC: " + mean(sp_counts_complex));
plot(sp_counts_complex);
title("Responses to the noise, Complex Cell");

%compute average
average = zeros(51);
count = 0;
for idx=1:n_images
   if sp_counts_complex(idx) >= 1
      average = average + images(:,:,idx);
      count = count + 1;
   end
end
average = average ./ count;
figure
surf(average);
title("RF estimation, Complex cell");



%STA simple cell
range = -5:1:5;
n_images = 10000;
images = normrnd(0,1,11,11,n_images);
out_rates = zeros(1,n_images);
for idx = 1:n_images
    out_rates(idx) = poissrnd(simple_cell(images(:,:,idx), range, phi, nonlinearity, adj_c_s
end
out_rates = round(0.2 .* out_rates ./ mean(out_rates));

% find all images that triggered spikes
count = 0;
triggers = [];
for idx=1:n_images
   if out_rates(idx) >= 1
      count = count + 1;
      triggers = [triggers, reshape(images(:,:,idx), 1, 11^2)];
   end
end

triggers = reshape(triggers, count, 11^2);
covar = cov(triggers);
[V,D] = eig(covar);
figure
scatter(1:11^2, sort(diag(D)));
title("Covariance matrix eigenvalues (SC)");
figure
surf(reshape(V(:,11^2),11,11))
```

```
title("Highest eigenvalue filter (SC)");
figure
surf(reshape(V(:,11^2 -1),11,11))
title("Second highest eigenvalue filter (SC)");



%STA complex cell
range = -5:1:5;
n_images = 10000;
images = normrnd(0,1,11,11,n_images);
out_rates = zeros(1,n_images);
for idx = 1:n_images
    out_rates(idx) = poissrnd(complex_cell(images(:,:,idx), range));
end
out_rates = round(0.2 .* out_rates ./ mean(out_rates));

count = 0;
triggers = [];
for idx=1:n_images
   if out_rates(idx) >= 1
      count = count + 1;
      triggers = [triggers, reshape(images(:,:,idx), 1, 11^2)];
   end
end

triggers = reshape(triggers, count, 11^2);
covar = cov(triggers);
[V,D] = eig(covar);
figure
scatter(1:11^2, sort(diag(D)));
title("Covariance matrix eigenvalues (CC)");
figure
surf(reshape(V(:,11^2),11,11))
title("Highest eigenvalue filter (CC)");
figure
surf(reshape(V(:,11^2 -1),11,11))
title("Second highest eigenvalue filter (CC)");
```