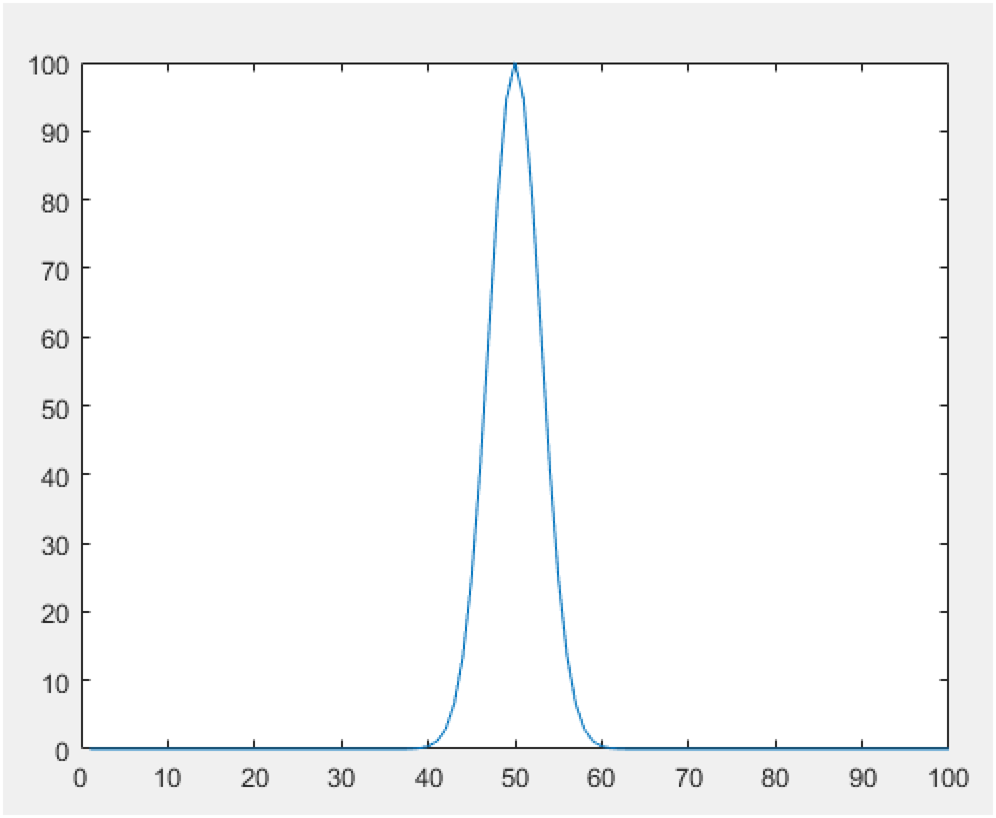
Fall 2017 - Mini project on population decoding

Mu: tuning curve center
Sigma: tuning curve width

Model the “clean” responses

Matlab code:

```
Time = 0.1; %converted from ms to s
maxspike = 100;
sigma = 3;
stim = 50;
index_neuron = 1:100;
fstim = maxspike* exp(-((stim-index_neuron).^2/(2*sigma^2)));
plot(1:100,fstim);
```

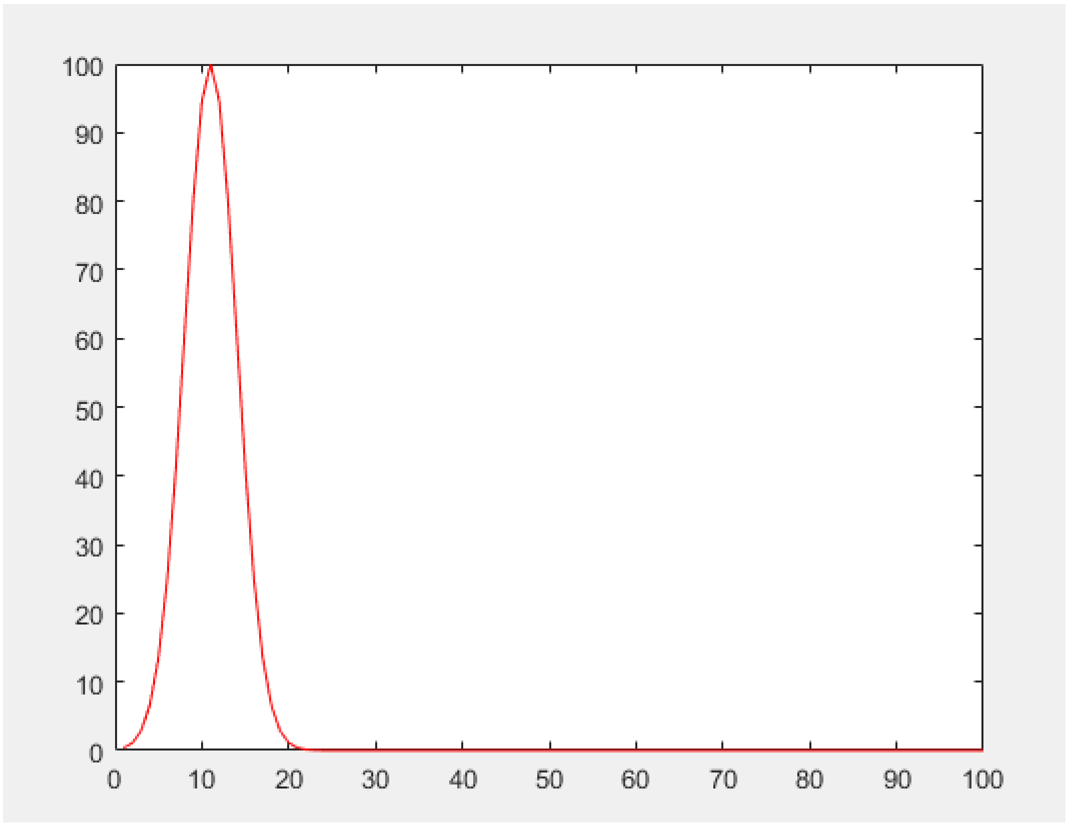


In the above figure a bell-shaped bump on the x axis is centered wherever the stimulus was. In this case the stimulus is at 50

To get different values of stimuli we use rand function of matlab along with this and store them all in an array arraystim.

Matlab code:

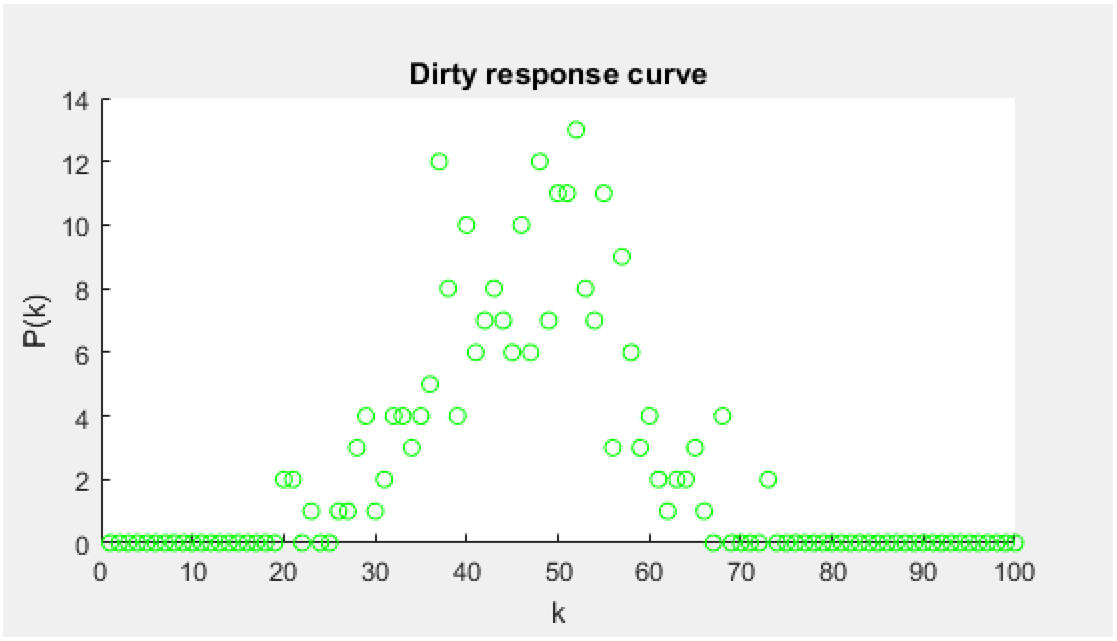
```
Time = 0.1;
maxspike = 100;
sigma = 3;
index_neuron = 1:100;
arraystim = zeros(2,100);

for i = 1:100
    stim = 100*rand;
    arraystim(1,i) = stim;
    fstim = maxspike* exp(-((stim-index_neuron).^2/(2*sigma^2)));
   plot(1:100,fstim,'r');
end
```



Model the "dirty" response

```
for i = 1:100
    stim = 100*rand;
    arraystim(1,i) = stim;
    fstim = maxspike* exp(-((stim-index_neuron).^2/(2*sigma^2)));

    lambda = fstim*Time;
    dirtyarray = poissrnd(lambda);
```

```
end
figure;
subplot(2,2,1);
plot(1:100,fstim,'r');
xlabel('stim')
ylabel('Firing rates')
title({['Clean tuning curve']})



subplot(2,2,2);
scatter(1:100,dirtyarray,'g');
xlabel('k')
ylabel('P(k)')
title({['Dirty response curve']})
```



We have used a scatter plot here. We can see the spikes when we use plot function.

Basically, neurons generate spikes randomly at the rate f according to a Poisson process, and we count the number of spikes over some time interval.

Lamda is the number of events expected to be measure in a given time interval T.
Where Lamda= f(stim) * T.
The Poisson distribution is the probability distribution over the number of events k (i.e. spikes) we will measure over that time interval in repeated trials given the value of lamda.

If lamda=10, you will get 10 spikes on average, but we could possibly get k=0, 1, 2, … to an arbitrarily large number of spikes.
This is the digression from the actual stimulus we observe later.

Decode : Step 4 and 5

Matlab code including all the steps:

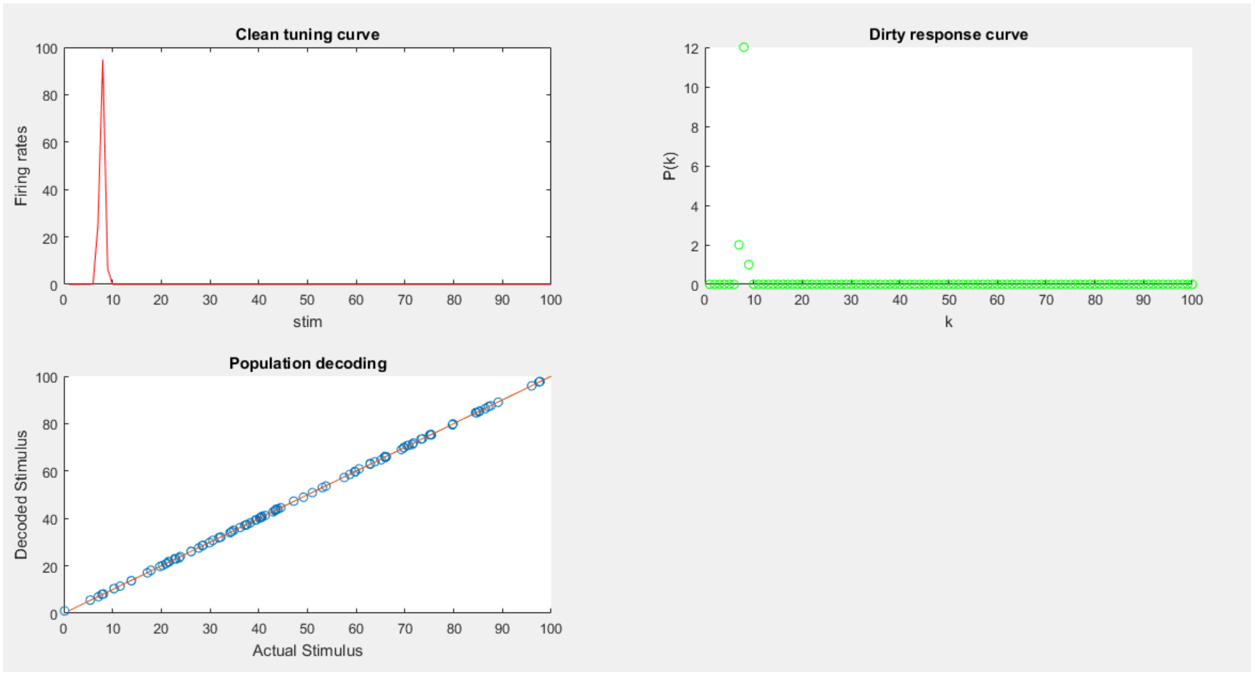
```
Time = 0.1;
maxspike = 100;
sigma = 8;

index_neuron = 1:100;

arraystim = zeros(2,100);
weighted_avg_array= zeros(1,100);

count=0;
squarederror=0;



for i = 1:100
    stim = 100*rand;
    arraystim(1,i) = stim;
    fstim = maxspike* exp(-((stim-index_neuron).^2/(2*sigma^2)));
   plot(1:100,fstim,'r');
    lambda = fstim*Time;
    dirtyarray = poissrnd(lambda);

    %Decode
    weightedavg = sum(dirtyarray.*index_neuron)/sum(dirtyarray);
    weighted_avg_array = weightedavg;
    arraystim(2,i)=weighted_avg_array;

if(stim > 40 && stim<60)
    squarederror= squarederror+ (stim-weightedavg)^2;
    count=count +1;
end
meansquarederror= squarederror/count;
end
figure;
subplot(2,2,1);
plot(1:100,fstim,'r');
xlabel('stim')
```

```
ylabel('Firing rates')
title({['Clean tuning curve']})



subplot(2,2,2);
scatter(1:100,dirtyarray,'g');
xlabel('k')
ylabel('P(k)')
title({['Dirty response curve']})

subplot(2,2,3);
scatter(arraystim(1,:),arraystim(2,:));
hold on
plot([1,100],[1,100]);
ylabel('Decoded Stimulus')
xlabel('Actual Stimulus')
title({['Population decoding']})
```

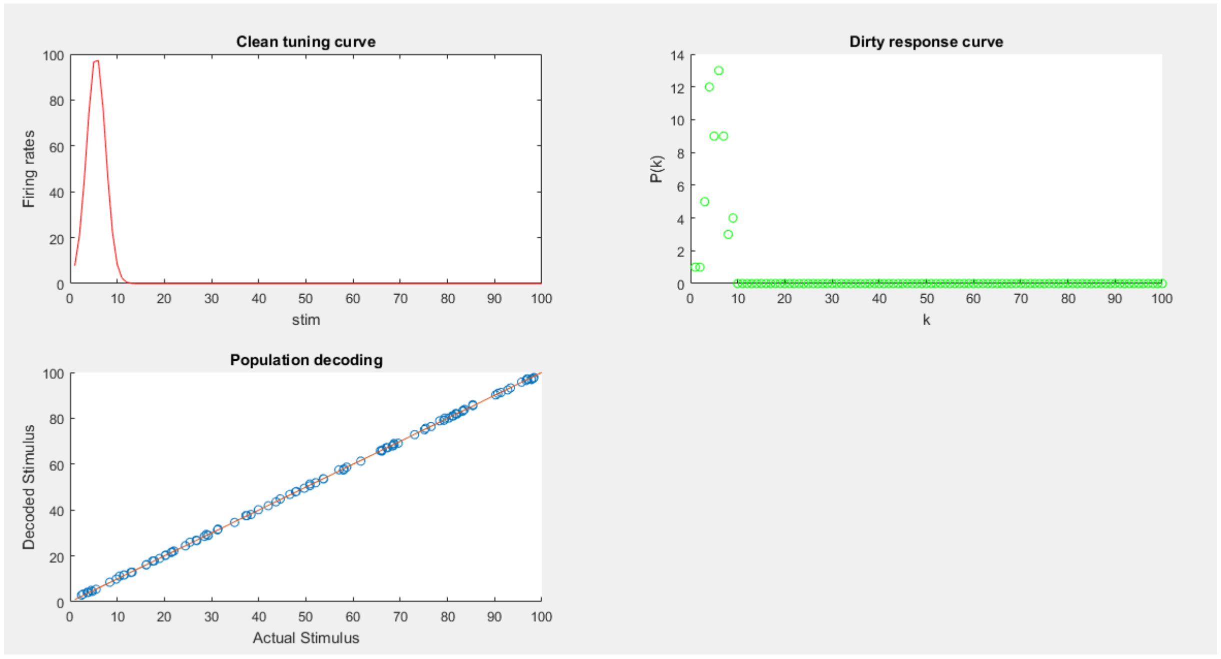
We consider a minimum value of 40 and maximum value of 60 while calculating the MSE and the ends of the array where other systematic biases have nothing to do with noises and will simply be zero.

<u>To determine experimentally whether it's better to have narrow or wide tuning curves to maximize decoding accuracy</u>.

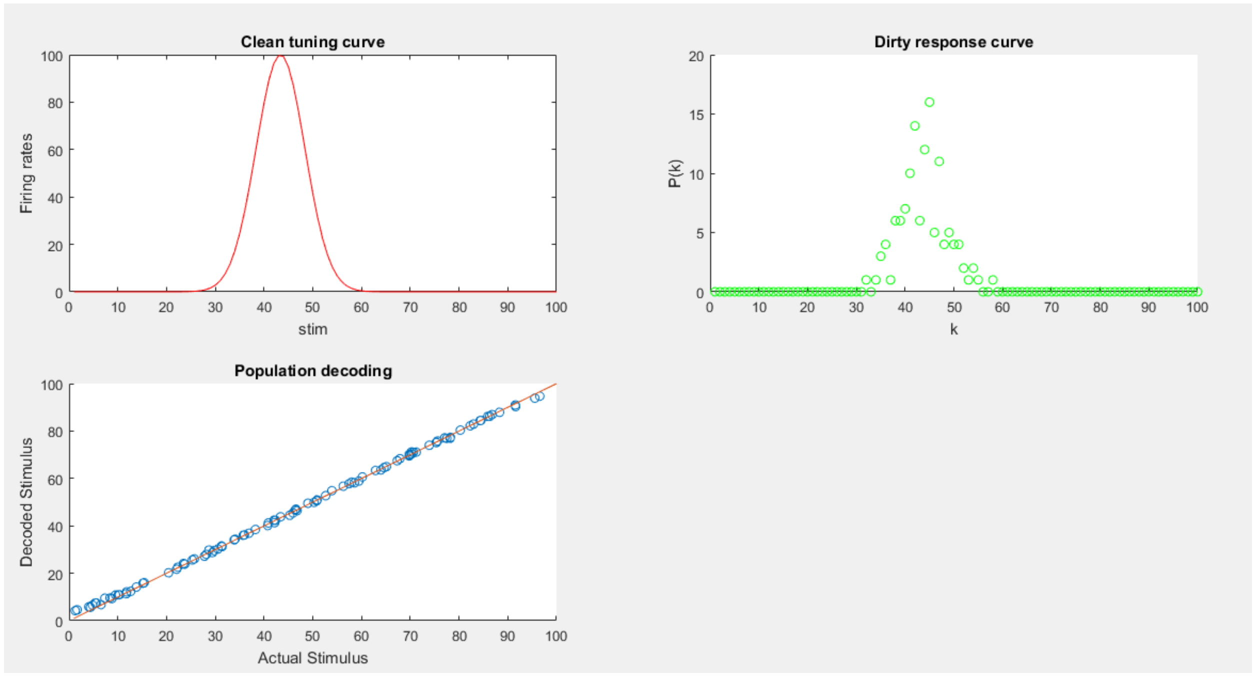
<u>Sigma=0.5, meansquarederror =0.0208</u>



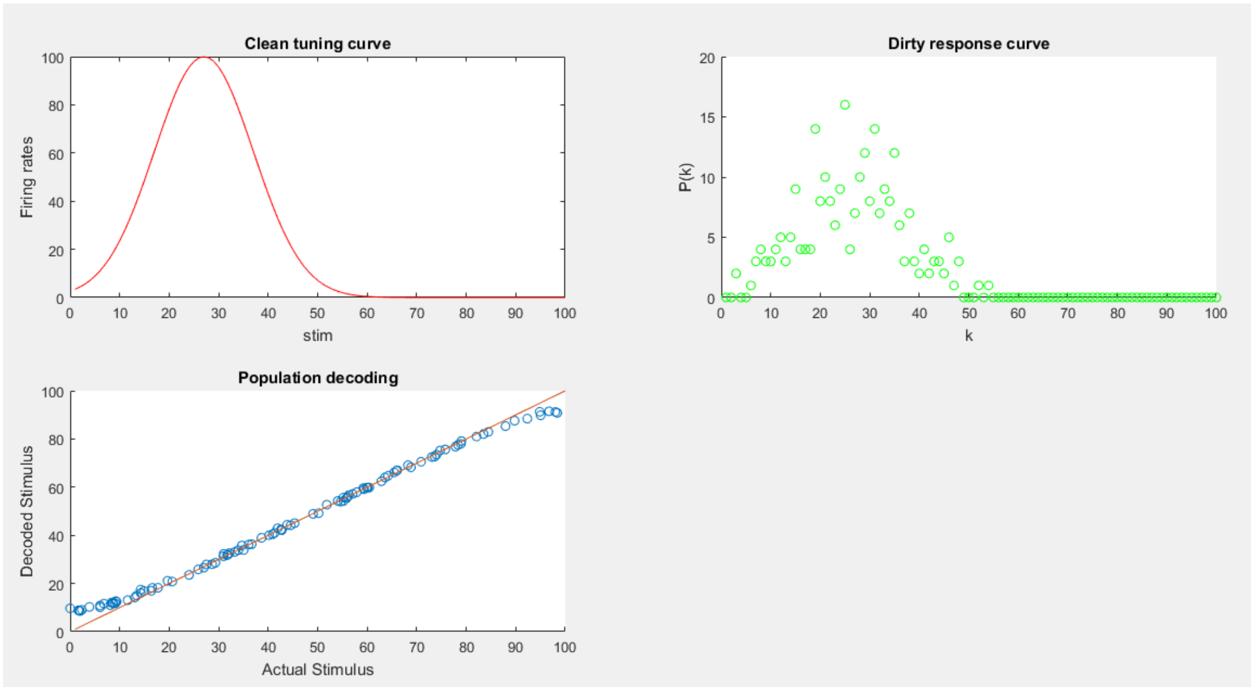
Sigma=2, meansquarederror =0.0627



Sigma=5, Mean squared error= 0.1305



Sigma=10,Mean squared error= 0.3673



Inference: Mean squared error increase with the increase in sigma values.Experimentally it's better to have narrow tuning curves to maximize decoding accuracy.

We can also figure this out by eye balling the population decoding curve. We can zoom in to see the digression of decoding stimulus from the actual stimulus increasing as the sigma values increase.

The best tuning curve width is that of a narrow curve where sigma value is low. This is inferred by examining by eye the decoding errors around the diagonal in decoded-vs-actual scatter plot, and also by the conclusion with MSE measure for every sigma value.