Table of Contents

In [1]:
```
# Default imports
import numpy as np
import pickle
import matplotlib
import matplotlib.pyplot as plt
from scipy import special
```

Directions for homework submission

Submit each of your homework to canvas as the pdf output of the jupyter notebook, and the jupyter notebook (.ipynb). Name your files starting in the format of "Last_name_First_name_File_name" separated by underscores.

For example, Jieyu submits two files for her homework this week:

1. Zheng_Jieyu_HW1.pdf (the pdf output of the jupyter notebook)

   If you have problems rendering your notebook into pdf, you can open your notebook in a browser and print -> save as pdf.
2. Zheng_Jieyu_HW1.ipynb

Please make sure your notebook can be run without errors within the cns187 virtual environment. Any file that fails to be executed on TA's end will be considered as late submissions.

Caltech Honor code: Searching for the solutions online is strictly prohibited. You should refer to the textbooks and lecture slides. If you are citing any external sources online, please include a list of references.

Collaboration on homework assignments is encouraged. However, **you cannot show each other the numerical answers or codes**.

All the mathematics should be typed in Latex format. You may work on a piece of paper and then type it into the notebook. Here is a useful cheat sheet (http://users.dickinson.edu/~richesod/latex/latexcheatsheet.pdf). Please do not submit pictures of handwritten maths.

For the schematic and drawings to be submitted, please display the images in markdown cells in your homework amd make sure they show up in your pdf rendering.

Please make sure that all your plots include a title and axis labels with units. One point will be deducted for each missing element.

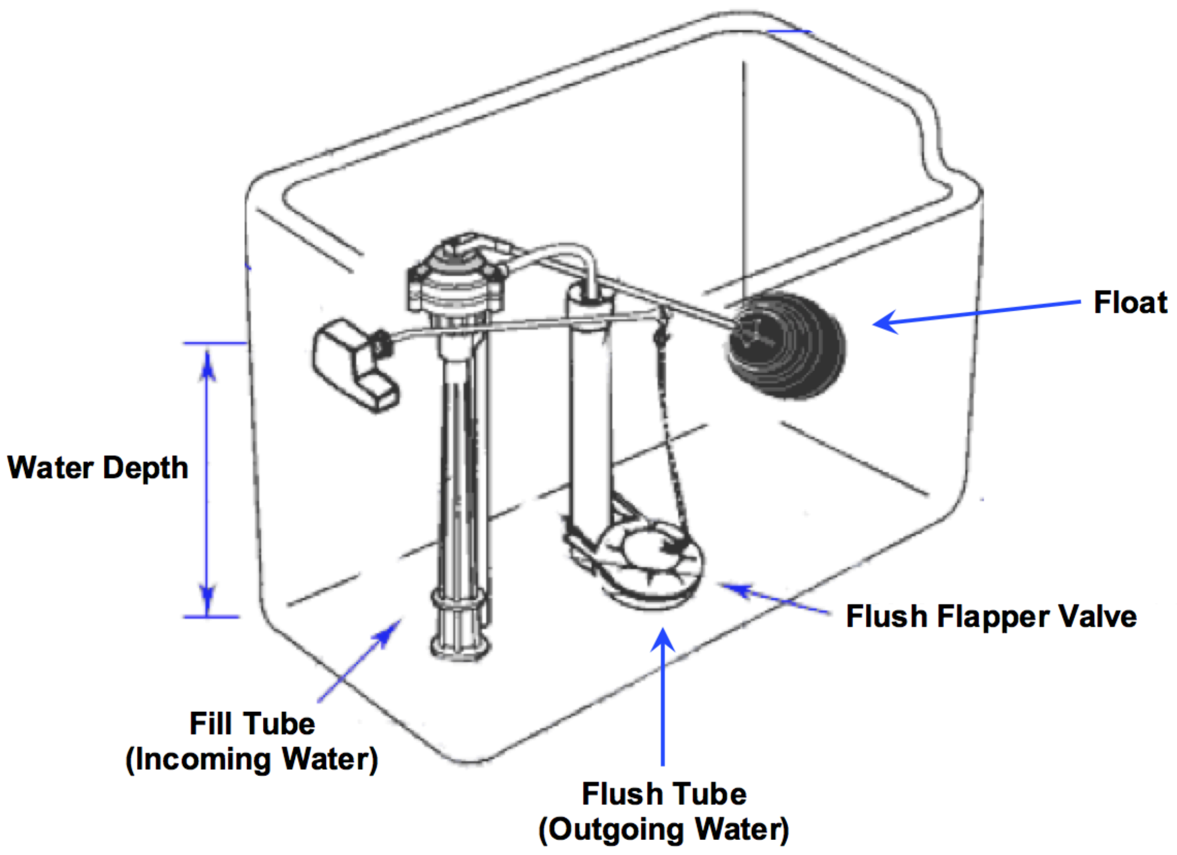
Order of Magnitude Neuroscience: Brain wiring (5 pts)

At the hotel bar someone offers you a bet that the brain contains more than 100,000 km of wiring. Seems like a large number, and you're inclined to bet against it. But should you? It would be poor form to whip out your cell phone, so you have to go with what you know. How to decide whether that claim is reasonable?

One could consider that there are 86 billion neurons in the brain, and looking at the size of a head, knowing connections often move from all over, you can say in the brain a neuron might be around a 15 cm length. If this is the case, then there is more than 100000 km of wiring, although it would be on the same order of magnitude.

The Integrate-and-fire Model (25 pts total)

Consider the main elements of the flush toilet (http://en.wikipedia.org/wiki/Flush_toilet). Briefly, water enters the tank through the tank fill tube. The float shuts off the incoming water when it reaches a certain level. When the toilet is flushed, the flush flapper valve is opened and the water in the tank exits through the flush tube. The flush flapper valve then closes, starting the process over again.



Toilet vs Neuron (7 pts)

If the water in the tank corresponds to electric charge in the integrate and fire circuit, describe the correspondence between the flush toilet elements and the elements in the integrate and fire circuit. (You only need to write down the corresponding toilet component for each sub-question, no need to explain. #1 is an example with given answer.)

1. Electric charge: water
2. Voltage: Water depth
3. Ground: Flush Tube
4. Voltmeter: Float (hint: think about what keeps track of the voltage and check if it reaches the threshold)
5. Capacitor: Tank
6. Capacitance: Area of the base of the tank
7. Input current: Fill Tube

Improvement (3 pts)

Compared to a neuron, a toilet based solely on the elements above cannot "fire" on its own, why? What elements are missing?

The toilet above would have to flush on its own when the float reaches its threshold given that this is when action potentials occur for integrate-and-fire neurons

Simple simulation (15 pts)

Consider the equation given below with the one in the lecture slide (simplest version):

$$\tau\frac{dV}{dt} + (V - E_L) = I \cdot R$$

If the membrane resistance R is 15 MΩ, Resting potential E_L is -70 mV, time constant τ is 10 ms, $V_{threshold}$ is -50 mV, V_{reset} is -65 mV, refractory period is 5 ms, what would be the minimal constant current for a neuron to fire? What current would make the firing rate saturate? Explain using the equation above.

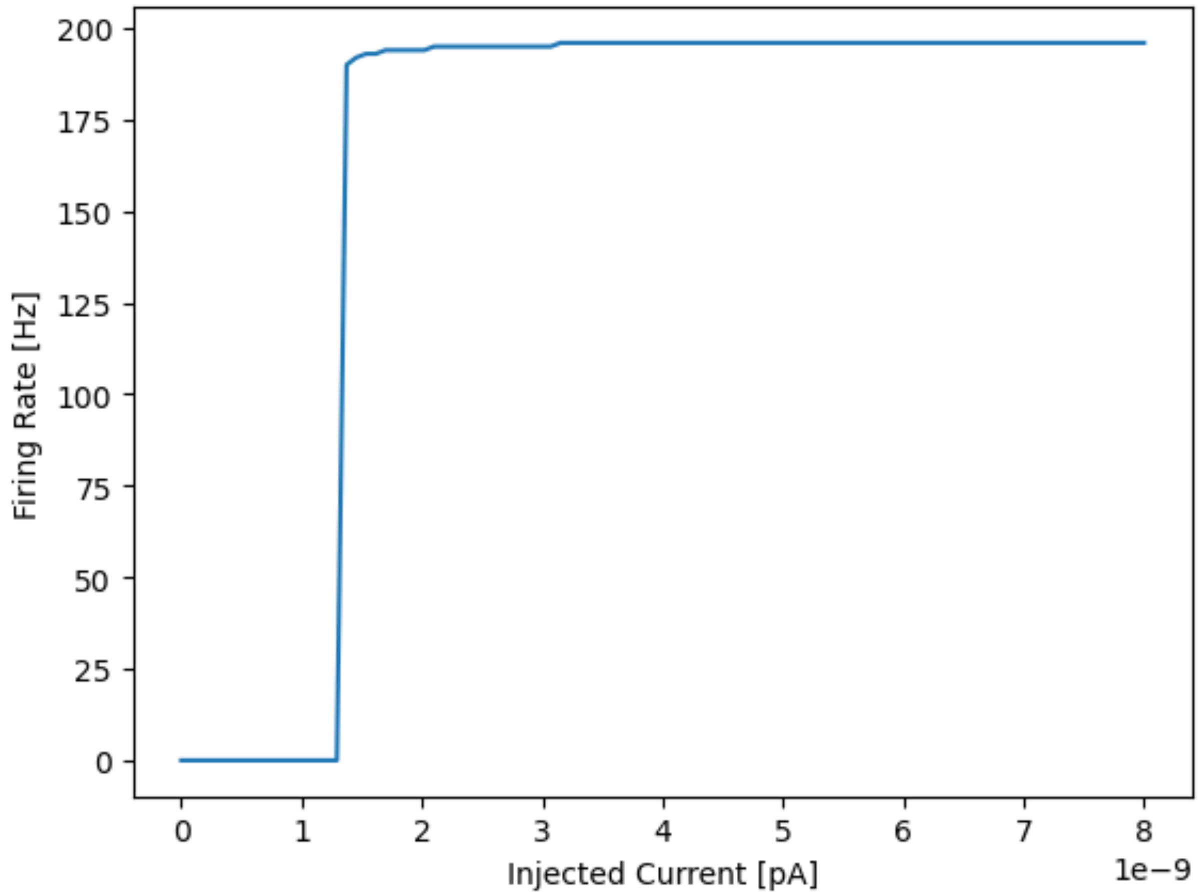
After doing the calculation above, take a look at the code section below. This is a partially finished simulation of a single neuron, with a stimulation duration of 1s, under constant current injection Iinj. The output of the simulation is an array of spike train.

Please finish the code section according to instruction below, then use the simulation function to validate your calculation result.

Type *Markdown* and LaTeX: α^2

In [2]:
```
def run_LIF(Iinj):
    """
    Simulate the LIF dynamics with external input current

    Args:
    Iinj: input current [pA]. The injected current here can be a value
                 or an array

    To be specified in the section below:
      V_threshold : threshold of action potential for the neuron, in m
      V_reset: post AP reset potential, in mV
      R: membrane resistance, in mega ohm
      E_resting: resting membrane potential, in mV
      tau_rc: membrane RC time constant, in ms
      t_ref: refractory period time length, in ms
    Returns:
    rec_sp     : array of time points at which a spike is fired, in ms
                 firing rate, note that the simulation is set for a to
    """

    # Set parameters
    V_threshold = -50
    V_reset = -65
    R = 15e9
    E_resting = -70
    tau_rc =  10
    tref = 5
    #some preset values
    # timestep size, in msec
    dt = 0.1  #if it is taking too long to finish simulation, consider

    # range of timesteps covered, in unit of dt, set to total length o
    range_t =  np.linspace(0, 1000, int((1000-0)/dt) + 1)

    # total number of timesteps
    Lt = range_t.size

    #initialize membrane potential at resting potential
    V_init = E_resting

    # Initialize voltage
    v = np.zeros(Lt)
    v[0] = V_init

    # Set current time course
    Iinj = Iinj * np.ones(Lt)

    # Loop over time
    rec_spikes = []  # record spike times
    tr = 0.  # the count for refractory duration

    for it in range(Lt - 1):
        if tr > 0:  # check if in refractory period
            v[it] = V_reset  # set voltage to reset
            tr = tr - 1 # reduce running counter of refractory period
        else:
            dv = (-(v[it] - E_resting) + Iinj[it]*R) / tau_rc
            v[it + 1] = v[it] + dt*dv

            if v[it + 1] >= V_threshold:  # if voltage over threshold
                rec_spikes.append(it)  # record spike event
                v[it + 1] = V_reset  # reset voltage
                tr = tref / dt  # set refractory time

    # Get spike times in ms
    rec_spikes = np.array(rec_spikes) * dt

    return rec_spikes
```

In [3]:
```
Iinj_vals = np.linspace(0, 8e-9, 100)

fire_rates = np.zeros(len(Iinj_vals))

for i, Iinj in enumerate(Iinj_vals):
    sp = run_LIF(Iinj)
    fr = len(sp) / 1.0
    fire_rates[i] = fr

plt.plot(Iinj_vals, fire_rates)
plt.xlabel('Injected Current [pA]')
plt.ylabel('Firing Rate [Hz]')
plt.show()
```



Neuronal Tuning Properties (30 pts total)

Estimating a Tuning Curve (6 pts)

Consider a pyramidal neuron in primary visual cortex. Describe how you would go about measuring the tuning properties (classical tuning curves) of that neuron with respect to:

1. Orientation Selectivity
2. Spatial Frequency Selectivity
3. Retinotopic Receptive Field Location

Be specific about the kinds of stimuli you would use and how they are parametrized. (Hint: Averages of repeated presentations!)

To measure the tuning properties of a pyramidal neuron in primary visual cortex, we would typically use electrophysiological recording techniques. Specifically with respect to the three parameters aforementioned:

1. Orientation Selectivity: Orientation selectivity of the neuron can be measured by presenting visual stimuli that vary in orientation and recording the neuron's response to each stimulus. A visual stimuli of the same form could be presented continually at different angles, such as a some bar of light that is rotated. While this is being done, a neuron's response can be recorded and tracked. Then, using plots and other forms of visualizations, the neuron's response can be modeled as a function of stimulus orientation to generate a tuning curve for orientation selectivity.
2. Spatial Frequency Selectivity: Spatial frequency selectivity of the neuron can be measured by presenting visual stimuli that vary in spatial frequency and recording the neuron's response to each stimulus. Similar to orientation, this task would require presenting stimuli of similar form from low frequency to higher frequency, such as groups of moving dots as seen in lecture. While this is being done, a neuron's response can be recorded and tracked. Then, using plots and other forms of visualizations, the neuron's response can be modeled as a function of spatial frequency to generate a tuning curve for spatial frequency selectivity.
3. Retinotopic Receptive Field Location: Retinotopic receptive field location of the neuron can be found by presenting visual stimuli at different locations in the visual field and recording the neuron's response to each stimulus. This can be done by presenting a targeted stimulus that moves across the field of vision, such as a spot of light, recording and tracking the response of the neuron as different positions are reached. Then, using plots and other forms of visualizations, the neuron's response can be modeled as a function of stimulus location to generate a tuning curve for retinotopic receptive field location.

In all experiments the presentation of each stimulus is repeated multiple times to obtain an average response and ensure that the neuron's response is consistent across presentations. The resulting tuning curves will then provide information necessary to relate real world occurrences to the processing in the brain.

Spike Triggered Average (4 pts)

Given the same neuron above, describe the procedure you would use for computing the spike-triggered average. Also contrast the two approaches. What information does each give you that the other does not? When is each approach appropriate to use?

Procedure:

1. Record neuronal activity while presenting a visual stimulus that produces reliable responses from the neuron.
2. Extract the spike times of the neuron during the stimulus presentation.
3. Define a time window preceding each spike time.
4. For each spike, keep track the visual stimulus within the time window preceding the spike.
5. Average the preceding visual stimuli of the time windows across all spikes.

The spike-triggered average (STA) is a way to measure the relationship between a neuron's spiking activity and its input as a whole. It represents the average of the visual stimuli that preceded the neuron's spikes. The STA can provide insights into the neuron's receptive field properties and can help us understand which parameters of the visual stimulus express the greatest effectiveness in driving a neuron's spiking activity.

Tuning curves, however, are generated by presenting a range of stimuli that vary in a particular parameter, and measuring the neuron's response to each stimulus. By plotting the neuron's response as a function of the parameter, we can generate a tuning curve that describes the neuron's sensitivity on average to that parameter.

The information provided by STA and tuning curves can be used in cohesion to fully model a neuron's response properties. STA reveals which features of the visual stimulus are most effective in driving the neuron's spiking activity, while the tuning curves reveal how the neuron's response varies with different intensities of the parameters composing the visual stimulus. The choice of approach should be guided by the specific research question and the properties of the neuron under study, and oftentimes it is useful to use both in order to gain different insights about a neuron.

Computing an STA: theory (10 pts)

Fill in the method below to properly compute a 1D STA for the specified inputs.

Hints:

```
a) No need to do anything fancy for this one. Just apply the d
efinition of "Spike-triggered average".

b) if you're computing an STA with k time points and you have
spikes that happen very early in the recording where you don't
have at least k stimulus samples yet, you can discard those.
```

In [4]:
```
def compute_sta(X, y, k):
    ''' Compute the spike triggered average from a stimulus and a spik

    Args:
    X: vector of stimulus time series (1D)
    y: spike train time-series
    k: time window for STA

    Return:
    vector of length k that is the STA'''

    sta = np.zeros(k)
    for i in range(k, len(y)):
        if y[i] == 1:
            sta += X[i - k:i]

    sta /= np.sum(y)
    return sta
```

Computing an STA: application (10 pts)

Download the data pickle and import it using the code shown below. This is real data recorded from the H1 neuron of a fly. This neuron receives input from the fly retina, and its purpose appears to be the detection of full field of view horizontal motion. Detecting horizontal motion is particularly useful in flight when the fly needs to control its yaw in order to maintain flight heading, or alternatively if it needs to turn in the XY plane. More information about the H1 neuron can be found here: https://en.wikipedia.org/wiki/H1_neuron (https://en.wikipedia.org/wiki/H1_neuron).

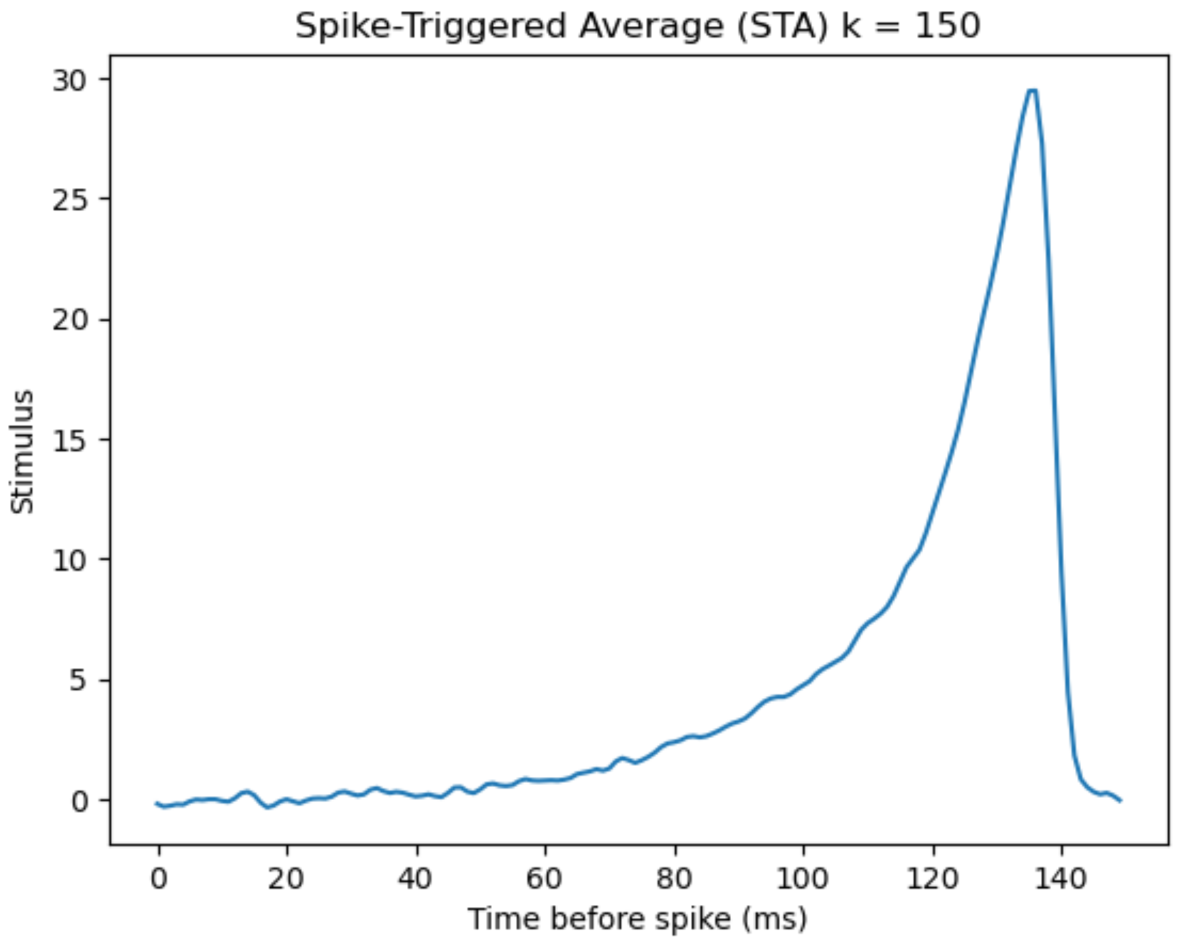
For the question, we have a 1D stimulus (the horizontal visual pattern experienced by the fly).

Stimulus: A fixed shape rotates around the fly over time and creates motion input that varies in one dimension (left-right around the fly). The stimulus information is contained in the variable `data['stim']` .

Spikes: The spikes discharged from the H1 neuron of the fly are stored in a vector with binary values, where 1 indicates spiking at that time point, and 0 indicates no spiking. The spiking information is contained in the variable "rho".

Compute and plot the STA of the fly H1 neuron using the method you implemented above. Please select and report an appropriate value of K for the STA.

In [5]:
```
file_name = 'HW2_data.pickle'

with open(file_name, 'rb') as f:
    data = pickle.load(f)

stim = data['stim']
rho  = data['rho']

#Compute and plot the STA!
k = 150
sta = compute_sta(stim, rho, k)

plt.plot(sta)
plt.xlabel('Time before spike (ms)')
plt.ylabel('Stimulus')
plt.title(f'Spike-Triggered Average (STA) k = {k}')
plt.show()
```



The Signal Detection Theory (40 pts total)

Suppose that the probabilities that a neuron responds with a firing rate between r and $r + \Delta r$ to two stimuli labeled plus and minus are $p[r|\pm]\,\Delta r$ where

$$p[r|\pm] = \frac{1}{\sqrt{2\pi}\sigma_r}\exp\left(-\frac{1}{2}\left(\frac{r-\langle r\rangle_\pm}{\sigma_r}\right)^2\right)$$

Assume $\langle r\rangle_- = 10\text{Hz}, \langle \text{r}\rangle_+ = 20\text{Hz}, \sigma_\text{r} = 5\text{Hz}$, and that hat these distributions produce negative rates rarely enough that we can integrate over r values over the entire range $-\infty < r < +\infty$

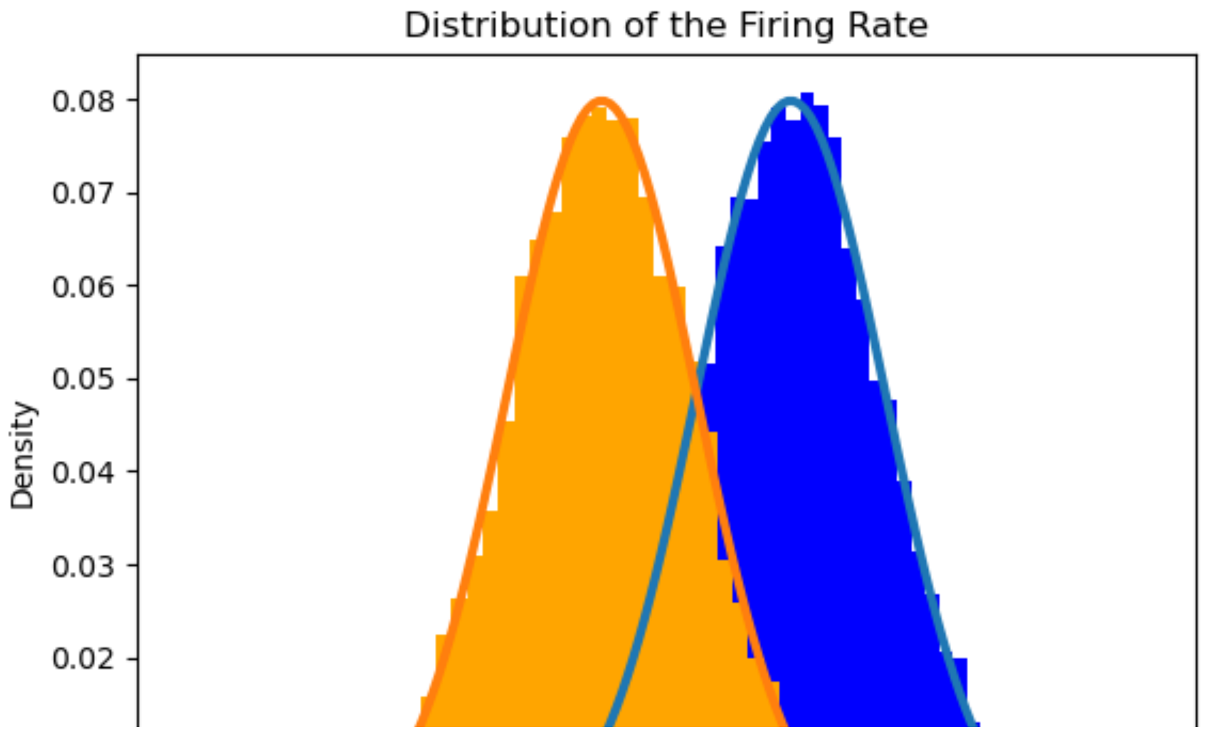
Distribution of the firing rate (10 pts)

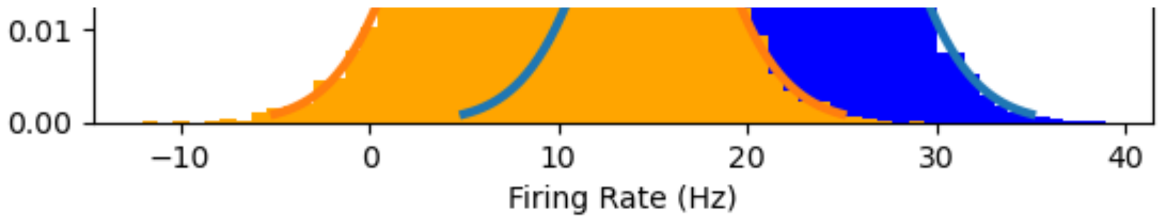
Plot the probability distributions of the firing rate of this neuron to the plus stimulus and minus stimulus on the same plot.

(Hint: You should recognize the distribution from the mathematical expression and use exsiting modules/functions from numpy to create your plots.)

In [6]:

```
# Set the mean and standard deviation of the normal distribution
mu_plus = 20
mu_minus = 10
sigma = 5

# Generate a random sample from the normal distribution
sample_plus = np.random.normal(mu_plus, sigma, 10000)
sample_minus = np.random.normal(mu_minus, sigma, 10000)

# Plot a histogram of the sample using Matplotlib
plt.hist(sample_plus, bins=50, density=True, color="blue")
plt.hist(sample_minus, bins=50, density=True, color="orange")

# Plot the normal distribution curve
x_plus = np.linspace(mu_plus - 3*sigma, mu_plus + 3*sigma, 100)
y_plus = 1/(sigma * np.sqrt(2 * np.pi)) * np.exp(-(x_plus - mu_plus)**
x_minus = np.linspace(mu_minus - 3*sigma, mu_minus + 3*sigma, 100)
y_minus = 1/(sigma * np.sqrt(2 * np.pi)) * np.exp(-(x_minus - mu_minus
plt.plot(x_plus, y_plus, linewidth=3)
plt.plot(x_minus, y_minus, linewidth=3)

# Add labels and a title to the plot
plt.xlabel('Firing Rate (Hz)')
plt.ylabel('Density')
plt.title('Distribution of the Firing Rate')

# Show the plot
plt.show()
```





False alarm and hit rate (10 pts)

Suppose that you base discrimination of the plus and minus stimuli on whether the evoked firing rate is greater or less than a threshold z. Show that the size or false alarm rate $\alpha(z)$ and power or hit rate $\beta(z)$, of this test are given by

$$\alpha(z) = \frac{1}{2}\mathrm{erfc}\left(\frac{\mathrm{z} - \langle \mathrm{r}\rangle_-}{\sqrt{2}\sigma_\mathrm{r}}\right)$$

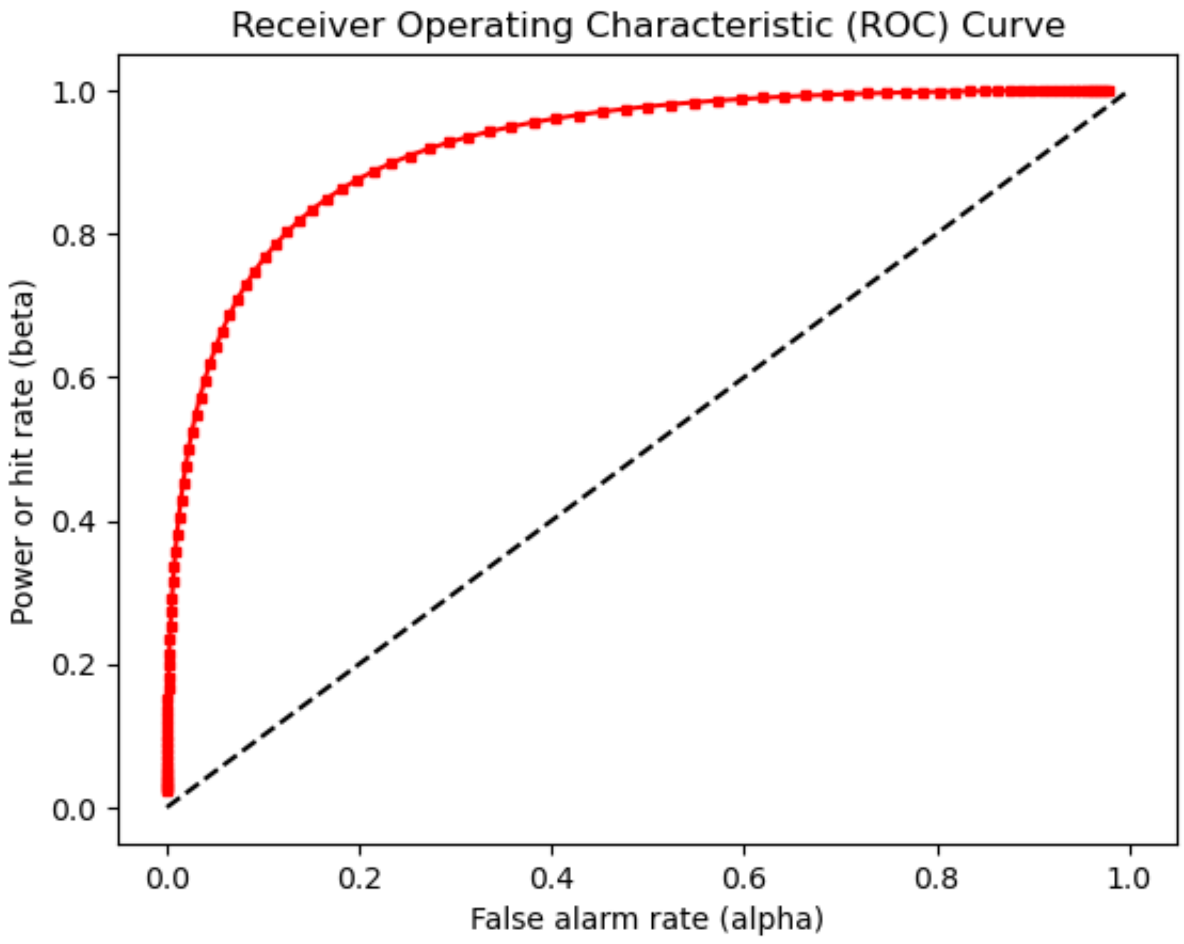
and

$$\beta(z) = \frac{1}{2}\mathrm{erfc}\left(\frac{\mathrm{z} - \langle \mathrm{r}\rangle_+}{\sqrt{2}\sigma_\mathrm{r}}\right)$$

Where $\mathrm{erfc(x)}$ is the complementary error function (https://en.wikipedia.org/wiki/Error_function).

Plot the ROC curve ($\beta(z)$ over $\alpha(z)$) and calculate the area under curve -- this is the probability of reporting the correct choice.

In [7]:
```
z = np.linspace(0, 30, 100)
alpha = 0.5 * special.erfc((z - mu_minus)/ (np.sqrt(2) * sigma))
beta = 0.5 * special.erfc((z - mu_plus)/ (np.sqrt(2) * sigma))
plt.plot(alpha, beta, marker="s", color="red", markersize=3)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False alarm rate (alpha)')
plt.ylabel('Power or hit rate (beta)')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.show()

alpha_sort = alpha.argsort()
alpha_sorted = alpha[alpha_sort]
beta_sorted = alpha[alpha_sort]
auc = np.trapz(beta_sorted, alpha_sorted)
print(auc)
```

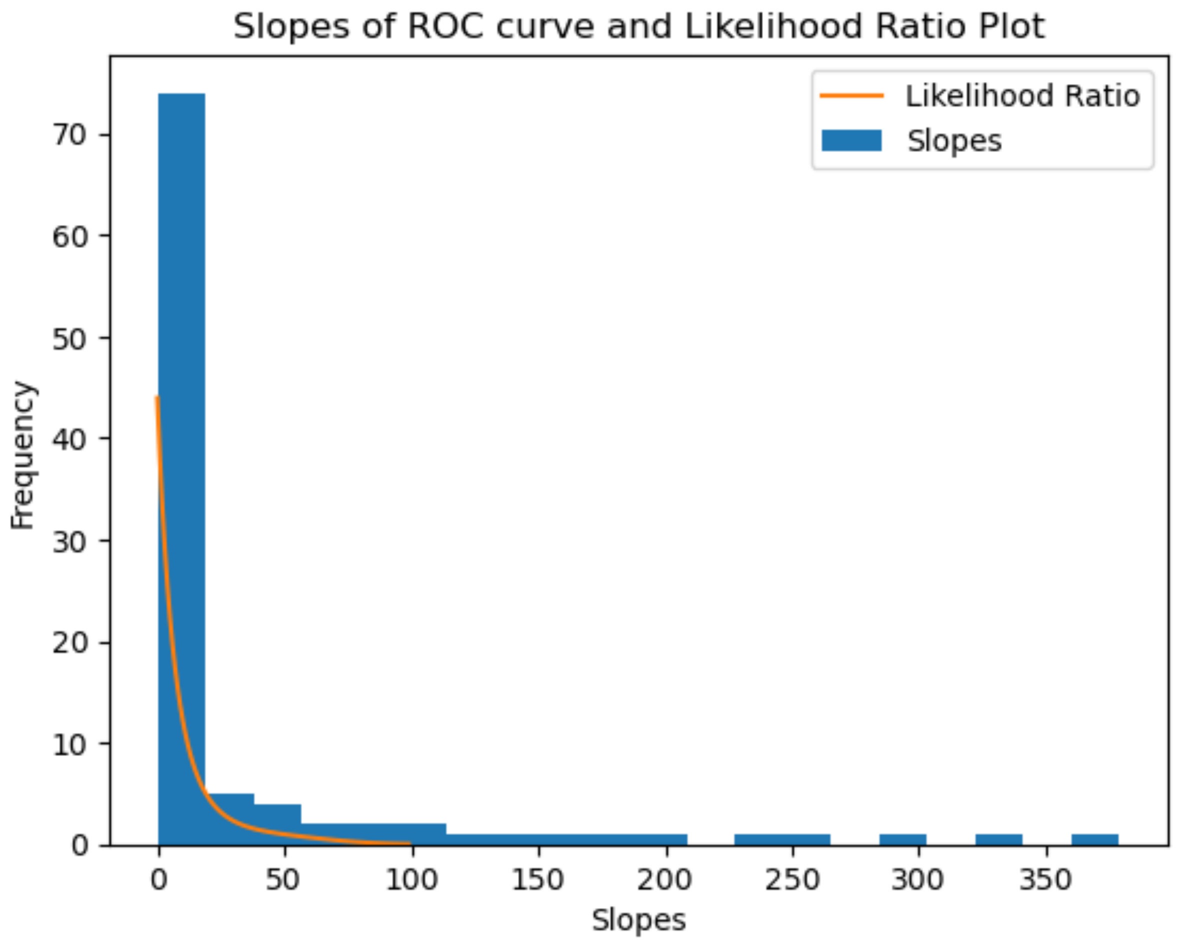


```
0.47750865180211677
```

Lieklihood ratio (10 pts)

Plot the histogram of slopes of the ROC curve; Calculate the likelihood ratio $l(r)$ and plot $l(r)$ on the same x-axis. What do you observe?

In [8]:
```
slopes = np.diff(beta) / np.diff(alpha)
lr = beta / (1 - alpha)
plt.hist(slopes, bins=20)
plt.plot(lr)
plt.legend(["Likelihood Ratio", "Slopes"])
plt.xlabel('Slopes')
plt.ylabel('Frequency')
plt.title('Slopes of ROC curve and Likelihood Ratio Plot')
plt.show()
```



It appears that the likelihood ratio plotted follows the distribution of the slopes of the ROC curve. Additionally, this appears to be a poisson distribution.

Probability of correct choice (10 pts)

Now plot a few different ROC curves with four different σ_r values of your choice (Be careful to not violate the assumption we made). Given the following equation:

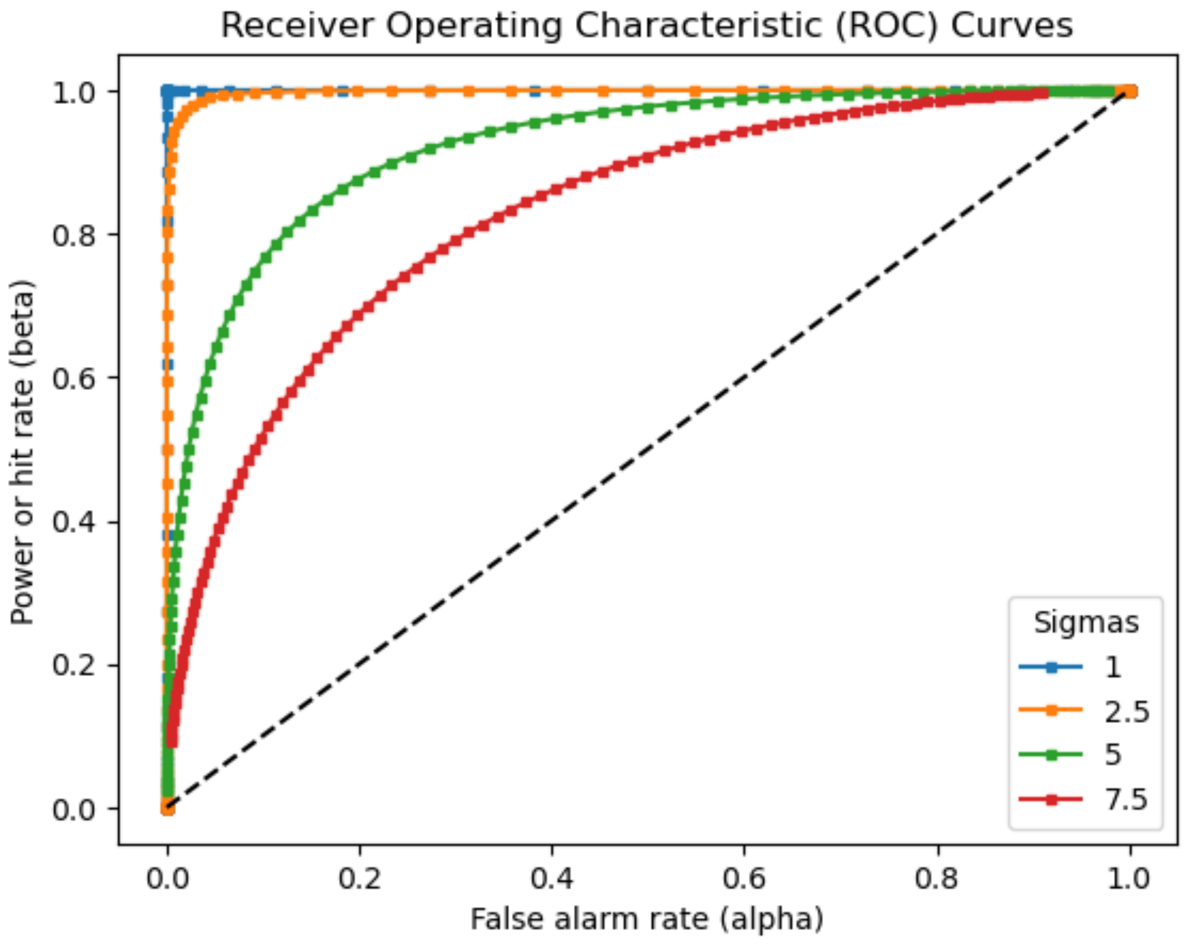
$$P[\text{correct}] = \frac{1}{2}\text{erfc}\left(-\frac{\mathrm{d}'}{2}\right)$$

where d' is the discriminability:

$$d' = \frac{\langle r \rangle_+ - \langle r \rangle_-}{\sigma_r}$$

Observe how the ROC curve change with d'. Explain qualitatively what happens.

In [9]:
```python
sigma_choices = [1, 2.5, 5, 7.5]
d_primes = [(mu_plus - mu_minus) / sig for sig in sigma_choices]

for sig in sigma_choices:
    alpha_sig = 0.5 * special.erfc((z - mu_minus)/ (np.sqrt(2) * sig))
    beta_sig = 0.5 * special.erfc((z - mu_plus)/ (np.sqrt(2) * sig))
    plt.plot(alpha_sig, beta_sig, marker="s", markersize=3)
    plt.legend(sigma_choices, title="Sigmas")
    plt.xlabel('False alarm rate (alpha)')
    plt.ylabel('Power or hit rate (beta)')
    plt.title('Receiver Operating Characteristic (ROC) Curves')

plt.plot([0, 1], [0, 1], 'k--')
plt.show()
print([0.5 * special.erfc(-0.5 * d) for d in d_primes])
```



```
[0.9999999999992313, 0.9976611325094764, 0.9213503964748574, 0.827110
7069244198]
```

It seems that higher discriminability means higher probability of being correct. One sees that as sigma increases, causing d' to decrease, the area under the curve of the ROC appears to become smaller indicating a lower probability of being correct.

In [ ]:

```
```