Lab Session #4

Computational Neurophysiology [E010620A]

Dept of Electronics and Informatics (VUB) and Dept of Information Technology (UGent)

Jorne Laton, Matthias Inghels, Talis Vertriest, Jeroen Van Schependom, Sarah Verhulst

Student names and IDs: Constantijn Coppers (02010771)
Academic Year: 2023-2024

Spiking Stochastics - Decoder and Encoder models

This exercise is adapted from the examples provided in the textbook "Case Studies in Neural Data Analysis", by Mark Kramer and Uri Eden (2016, MIT Press) and the 2020 eLife-LABs publication by Emily Schlafly, Anthea Cheung, Samantha Michalka, Paul Lipton, Caroline Moore-Kochlacs, Jason Bohland, Uri Eden, Mark Kramer. The outline of this exercise follows the theory presented in chapters 7 & 9 of "Neuronal Dynamics" by Gerstner, Kistler, Naud, Paninski (2014, Cambridge University Press). It also uses the principles outlined in Jackson and Carney, 2005 (publication provided with this exercise).

Aim

Here, we go deeper into understanding the stochastics behind neuronal action potential (spiking) generation. We will study the Poisson and Gaussian models and explore ways in which we can fit these point-process models to recorded datasets. In the second part of the exercise, we will consider encoder models that are based on leaky integrate-and-fire models with noisy inputs (8.1 in Gerstner et al.) and apply these models to test a specific research hypothesis related to the spontanous firing rate observed in cat auditory-nerve fibers.

Part 1 (Q1-Q4)

Data: Spontaneous spiking activity from a retinal neuron in culture, exposed to low-light and high-light environments. We continue with the dataset from the previous notebook.

Goal: Build simple models of interspike interval distributions as a function of the ambient light level.

Tools: Interspike interval (ISI) histograms, firing rate, maximum likelihood estimation, Kolmogorov-Smirnov plots, Poisson and Gaussian distributions.

Part 2 (Q5-Q7)

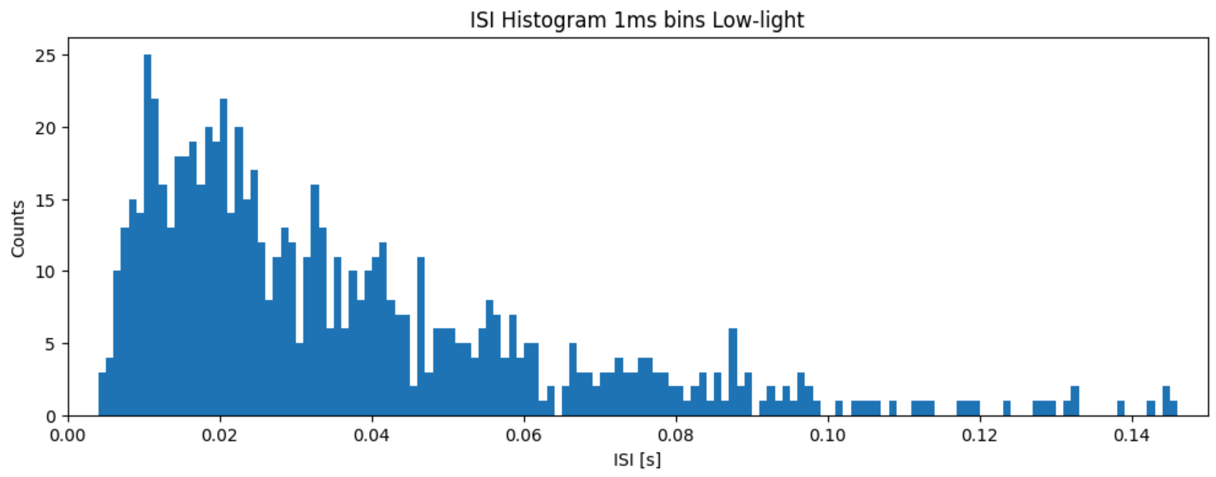
Data: Auditory-nerve spike patterns recorded in the absence of external stimulation from two different fiber types; a low-spontaneous rate fiber, and a high-spontaneous rate fiber.

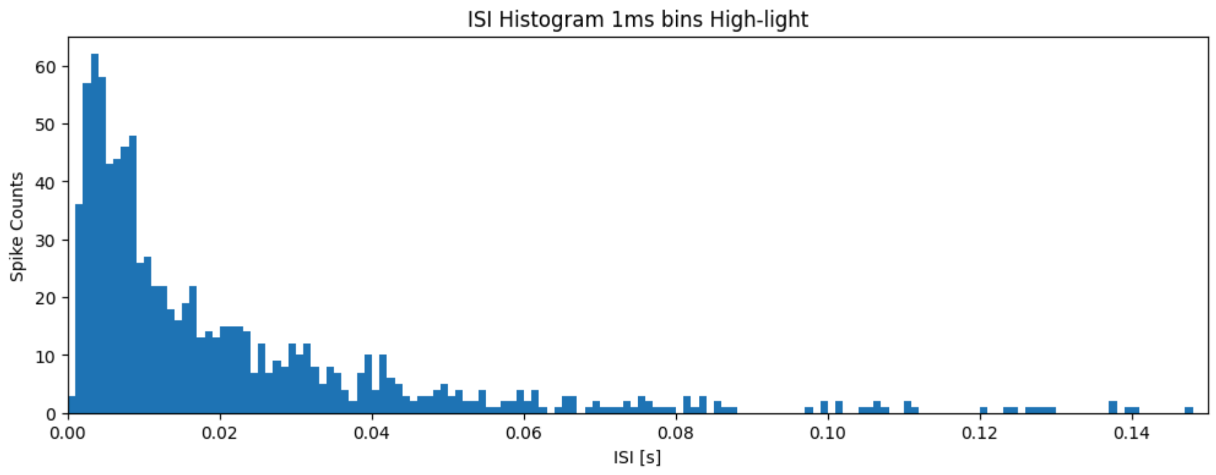
Goal: Build decoder models with the same stochastics as observed experimentally.

Tools: Encoder models, ISI histograms, autocorrelation functions, integrate-and-fire models.

In [1]:
```
# Run this cell to make sure to have the preambles/data loaded
from pylab import *
import scipy.io as sio
import numpy as np
import random
import warnings
from ffGn_module import ffGn, inhomPP
%matplotlib inline
rcParams['figure.figsize'] = (12, 4)  # Change the default figure size

warnings.simplefilter(action = 'ignore', category = FutureWarning)

data = sio.loadmat('matfiles/08_spikes-1.mat')  # Load the spike train data
print(data.keys())

spikes_low = data['SpikesLow'][0]
spikes_high = data['SpikesHigh'][0]

ISI_low = np.diff(spikes_low)
ISI_high = np.diff(spikes_high)

bin_edges = arange(0, 0.501, 0.001)  # Define the bins for the histogram

plt.hist(ISI_low, bin_edges)        # Plot the histogram of the low-light ISI data
xlim([0, 0.15])             # ... focus on ISIs from 0 to 150 ms
ylabel('Counts')
xlabel('ISI [s]')# ... label the y-axis
title('ISI Histogram 1ms bins Low-light')             # ... give the plot a title
plt.show()

plt.hist(ISI_high, bin_edges)       # Plot the histogram of the high-light ISI data
xlim([0, 0.15])             # ... focus on ISIs from 0 to 150 ms
xlabel('ISI [s]')               # ... label the x-axis
ylabel('Spike Counts')              # ... and the y-axis
title('ISI Histogram 1ms bins High-light')            # ... give the plot a title
plt.show()
```

```
dict_keys(['__header__', '__version__', '__globals__', 'SpikesLow', 'SpikesHigh'])
```





PART 1

Q1: Statistical Models

We now consider another powerful technique to understand spiking data, by constructing a ***statistical model*** of the data. This model will capture important features of the data but does not consist of explicit biophysical components, such as (for example) in the Hodgkin-Huxley equations. Statistical models are thus ***phenomenological*** in nature and not biophysically realistic.

To construct a statistical model, we typically assume that the ISIs are indendent samples from an unknown distribution. We then consider a class of statistical distributions that might fit the original data and identify which of these distributions (and its parameters) that provide the best fit to the observed data.

The most basic model we could fit is the Poisson distribution (assumption: the number of spikes in any bin is independent of all previous and future spikes) with an unknown but constant firing rate λ (v in the lecture slides). The probability P of observing k spikes in any time bin is given by the Poisson distribution,

$$P(k) = \frac{\lambda^k e^{-\lambda}}{k!},$$

where $k!$ is the factorial of k. Under this model, the distribution for the number of spikes in a bin is Poisson. An important question for our data-set is what the distribution of the waiting times between spikes (i.e., ISIs) looks like? It can be shown mathematically that for any Poisson process with a constant firing rate, the ISIs have an exponential distribution [Kass, Eden & Brown, 2014]. Mathematically, the probability density function for any ISI taking on a value x is

$$f(x) = \lambda \exp(-\lambda x),$$

where λ is the rate parameter for the Poisson process.

Alert 1! This is a common point of confusion. The increments of a Poisson process have a Poisson distribution, and the ISIs have an exponential distribution. The Poisson distribution takes on non-negative integer values {0,1,...,∞}, which make it appropriate for counting the number of spikes in an interval. The Poisson distribution does not make sense to describe the waiting time between spikes, since this typically takes on a continuous value in [0, ∞].

Our goal is to find a good value of λ so that our statistical model

$$f(x) = \lambda \exp(-\lambda x),$$

matches the observed ISI distributions.

Alert 2! The histogram ISI values are in the scale of frequencies, not probabilities, and moreover also binned,thus not taking into account its density. This means that we have to scale the histogram data twice. Once from frequencies to probability and once more to scale it from binned values to preveal the density values.

- Q1: First make a histogram of the actual data, bin the ISI's in 1ms bins. Then scale your data as described in alert 2 to be able to fit a probability density function. (Control that the total area of your histogram approaches the value of 1). Plot the scaled data together with the exponential probability density function, where you guess a value of λ . Now try different values of λ and guess which λ provides the best visual match. Do these steps for both the low and high light conditions.

- Fill in answer here

Q2: Likelihood functions

The process of guessing values of λ and comparing the model to the empirical ISI distribution can be automated by determining the ***likelihood*** function of the joint probability distribution of the data. The goal is to find the value of λ that maximizes the likelihood of the data given the statistical model; this value of λ will be the best fit of the model to the data.

To implement this procedure, first consider the probability density of observing a sequence of ISIs, $x_1, x_2, \ldots, x_n$. If we assume that the ISIs are independent, then the probability density is:

$$\begin{aligned} f(x_1, x_2, \ldots, x_n) &= f(x_1)f(x_2)\ldots f(x_n) \\ &= \lambda\exp(-\lambda x_1)\lambda\exp(-\lambda x_2)\ldots\lambda\exp(-\lambda x_n) \\ &= \lambda^n\exp(-\lambda\sum_{i=1}^{n} x_i). \end{aligned}$$

We call this expression the joint probability distribution of the observed data. In the first equality, we separate the joint probability distribution $f(x_1, x_2, \ldots, x_n)$ into a product of probability distributions of each event (i.e., $f(x_1)$, the probability of the first ISI equaling x_1; multiplied by $f(x_2)$, the probability of the second ISI equaling x_2; multiplied by $f(x_3)$, the probability of the third ISI equaling x_3; and so on). This partitioning of the joint probability is valid here because we assume the ISIs are independent. In the second equality, we replace each probability distribution with the exponential distribution we expect for the ISIs of a Poisson process. In the last equality, we rewrite the expression as a single exponential. Notice that this last expression is a function of the unknown rate parameter, λ.

When considered as a function of the unknown parameters, the joint distribution of the data is also called the *likelihood*. In this case, we write

$$L(\lambda) = \lambda^n e^{-\lambda(x_1+x_2+\ldots+x_n)},$$

to indicate that the likelihood L is a function of λ.

- Q2.1: Plot the likelihood function $L(\lambda)$ for each light condition and consider a range of λ values. Evaluate the maxima carefully, does the result make sense when considering the equation? If the result doesn't make sense at first, do not panic. It is a feature, not a bug.
- Q2.2: Plot the log of the likelihood instead, and identify which $L(\lambda)$s are a good fit to the low and high-light conditions. The λ, where the log likelihood is maximized, is called the **maximum likelihood estimate** of λ: $\hat{\lambda}$. The log-likelihood function is:

$$l(\lambda) = n\ln(\lambda) - \lambda(x_1 + x_2 + \ldots + x_n),$$

- Q2.3: Is the difference in the Poisson rate parameter between the low-and high-light conditions statistically significant? To address this last question, we can use a bootstrap analysis. Combine all the ISIs from both conditions into one pool, sample many (1000) new datasets with values randomly selected from that pool. Plot the distribution of the differences accross the 1000 samples as a histogram, together with the actual difference in both original rate parameters. Compare the results, what is your conclusion?

- Fill in answer here

Q3: Goodness of Fit

We assumed that the spikes were generated according to a poisson process, but is the exponential pdf that we created a good fit for the ISI?

- Q3.1: To answer this, first compare the model fits and the scaled data visually. Therefore plot again the exponential pdf (with the optimal λ) together with the scaled histogram values. Compare the expected proportion of ISIs for a Poisson process to the ISI histograms we actually observe in each condition. What is your conclusion?

To go beyond visual inspection and quantify the goodness of fit, we compare the cumulative distributions computed from the scaled data and the model. The ***cumulative distribution function (CDF)***, $F(x)$, is the probability that a random variable will take on a value less than or equal to x. For the exponential ISI model with rate parameter λ, the model CDF is

$$\begin{aligned} F_{mod}(x) = & \quad \Pr\,(\mathrm{ISI} \leq x) \\ = & \quad \int_0^x \lambda e^{-\lambda t} dt \\ = & \quad 1 - e^{-\lambda x}. \end{aligned}$$

We compare this to the empirical CDF of the data, $F_{emp}(x)$, which is defined as the proportion of observations less than or equal to x. In other words, this is the cumulative sum of all previous areas of the histogram, not just cumulative sum of the scaled probability density values. Scale your empirical data correctly, the last value should approach 1.

- Q3.2: Compute and plot the CDF function together with the empirical values. Evaluate how well the model fits the empirical data. Do this for both conditions and compare the CDF models according to its light condition.

- Q3.3: Another common way to visualize the difference between the model and empirical distributions is a ***Kolmogorov-Smirnov(KS)*** plot. This is a plot of the empirical CDF against the model CDF directly. Since the KS plot compares CDFs, both the x-axis and y-axis range from 0 to 1. A perfect fit between the model and empirical CDFs would look like a straight, 45-degree line between the points (0,0) and (1,1). Any deviation from this line represents deviation between the observed and model distributions. One nice result for comparing CDFs is that with enough data, the maximum difference between the model and empirical CDFs has a known asymptotic distribution, which can be used to put confidence bounds about the KS plot [Kass, Eden & Brown, 2014]. For 95% confidence bounds, a well-fit model should stay within $\pm 1.36/\sqrt{N}$ of the 45-degree line, where N is the number of ISIs observed. Let's place these confidence bounds on the KS plot and determine whether your model fits the data well.

- Fill in answer here

Q4: More advanced, inverse Gaussian probability models.

So far, we investigated how well one class of models, the exponential distribution function, fits the observed ISI distributions. However, this type of statistical model is not always sufficient to mimic the observed data. There are many other choices for statistical models; and here we'll try one other class of models: the inverse Gaussian probability model. This model has previously been used successfully to describe ISI structures [Iyengar & Liao, 1997]) and the mathematical expression for the inverse Gaussian probability density is,

$$f(x) = \sqrt{\frac{\lambda}{2\pi x^3}}\exp\left(\frac{-\lambda(x-\mu)^2}{2x\mu^2}\right).$$

The inverse Gaussian distribution has two parameters that determine its shape: μ, which determines the mean of the distribution; and λ, which is called the shape parameter. At $x = 0$, the inverse Gaussian has a probability density equal to zero, which suggests it could capture some of the refractoriness seen in the data.

If we again assume that the ISIs are independent of each other, then the likelihood of observing the sequence of ISIs, $x_1, x_2, \ldots, x_n$, is the product of the probabilities of each ISI,

$$L(\mu,\lambda) = f(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{N}\sqrt{\frac{\lambda}{2\pi x_i^3}}\exp\left(\frac{-\lambda(x_i-\mu)^2}{2x_i\mu^2}\right)$$

The log likelihood is then

$$\log\big(L(\mu,\lambda)\big) = \frac{N}{2}\log\frac{\lambda}{2\pi} - \frac{3}{2}\sum_{i=1}^{N}\log x_i - \sum_{i=1}^{N}\frac{\lambda(x_i-\mu)^2}{2x_i\mu^2}$$

Since this distribution has two parameters, the maximum likelihood solution for this model is the pair of parameter estimates $\hat{\mu}, \hat{\lambda}$ that maximizes the likelihood of the data. We can solve for the maximum likelihood estimate analytically by taking the derivative with respect to both parameters, setting these equal to zero, and solving the resulting set of equations. In this case, the maximum likelihood estimators are

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N}x_i$$

and

$$\hat{\lambda} = \left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{x_i} - \frac{1}{\hat{\mu}}\right)\right)^{-1}$$

- Q4.1 : Use this expression to fit an inverse Gaussian model to the scaled data in each condition. Again, make sure you scale the empirical data correctly to the model and plot them together.
- Q4.2: Consider the goodness-of-fit of the model in the two conditions. Does the inverse Gaussian model provide a good fit to the ISIs, and motivate why (or why not) this model is better fit? (calculate the CDF of both the inverse gaussian and the empirical data. Again, use the Ks plot to evaluate.)
- Q4.3: Compare the μ and λ estimates between the two conditions. What can you conclude about the differences between the low- and high-light conditions?
- Fill in answer here

PART 2

Q5: From Decoding to Encoding models

Identifying the best fitting model and its parameters helps us to understand and describe spiking statistics of a variety of neurons, and this process is called ***Decoding***. However, when developing neuronal models or mimicking physiological processes in machines (e.g. robots) one would like to follow an ***Encoding*** approach in which we provide a stimulus to a model of the neuron and predict a spiking pattern that mimics that of the biophysical system. Well-designed encoding models can generate spiking patterns to any stimulus (e.g. Hodgkin-Huxley type of neuronal models with escape noise), but for simplicity, we will focuss on simulating the spiking statistics of auditory-nerve models when spiking at their spontaneous rate (i.e. without external simululation applied). Indeed, spiking can occur in the absence of stimuluation because there is always some sort of neuronal/background noise present in biological systems that can drive the neuron temporarily above its firing threshold and generate sporadic spikes. Neurons or nerve fibers are therefore often characterised by their ***Spontaneous Rate (SR)*** i.e. the number of spikes/s (firing rate) generated in the absence of external stimulation.

Much of what we know about the auditory-nerve fibers (i.e. connection between the sensory cochlear inner-hair-cells and the spiral ganglion cells and auditory nerve bundle) stems from the characterisation of single-unit neuronal recordings in cats made by [Nelson Kiang, 1965] and [Charles M Liberman, 1978]. The below picture provides a good overview of the distribution of different spontaneous rates that were found from a large number fo single-unit of recordings (in the absence of external stimulation). Taken from [Jackson and Carney, 2005]

No description has been provided for this image

The authors of the last paper pose that the distribution of observed SRs fibers can be generated using ***Poisson-equivalent integrate-and-fire models*** for two or three fixed SR values. In other words, the stochastics of the underlying spike generation process is such that the observed SR distribution can be observed even when only two/three types of SR-AN fibers exist. Here, you will implement the proposed encoder model and compare it to a standard poisson-based encoder model.

- Q5.1 : First, determine (decode) the model parameters given recordings of two different AN fibers. Load in the data, and determine the $\hat{\lambda}$ and $\hat{\mu}$ parameters for both low and high-spontaneous rate AN fibers using an inverse Gaussian model. Plot again as done in Q4
- Fill in answer here

Q6: Inhomogeneous integrate-and-fire encoder model

Next, you can implement the encoding principle as outlined in [Jackson and Carney, 2005].

The authors suggest that a poisson process (i.e., each spike is generated independently from the previous spike times) cannot account for the observed behavior, instead the encoder model should include a ***long-range temporal dependence (LRD)*** which is characterised by the ***Hurst exponent (H)***. The hurst index is a measure of the long-term memory of a time series and relates to the rate at which the autocorrelation of a time series decreases as the lag between pairs of spike times increases. It quantifies the relative tendency of a time series to either regress strongly to the mean or to cluster in a specific direction.

- A value of H=0.5 reflects an exponential decay and describes a completely uncorrelated series (i.e. Poisson process). For this series, the absolute values of the autocorrelation function decays exponentially quickly to zero.
- Differently, for values of H: 0.5 - 1, there is LRD and the time serie has a long-term positive autocorrelation, i.e. the autocorrelations decay so slowly that their sum diverges to infinity. This means that a late spike in the series will probably be followed by a spike after a long ISI, and that the late spikes will all be characterised by long ISIs.

The authors suggest that the spontaneous rate histogram can be created by a ***fractal-Gaussian-noise-driven inhomogeneous poisson process (fGnDP)***, a type of escape noise model (chapter 9.4.1 in Gerstner) that consists of a doubly stochastic Poisson process, wherein the stochastic process that serves as an input to a poisson-equivalent integrate-fire model is a fractal-Gaussian noise (fGn) that can be made to contain LRD by modifying the Hurst index.

- Q6.1: Before we generate the spikes, let's first generate the escape noise (fGn). Generate a noisy output signal of 30-s length (FS = 1000 Hz) that has a mean rate equal to the SR of the fiber and that either follows a poisson process (H=0.5) or has LRD (H=0.95). Consider the SR 80 spikes/s. Note that even if H=0.5, the underlying SR distribution follows an inhomogeneous poisson process, where the rate varies due to pure white noise which is gaussian distributed around the mean SR. In the case of LRD (H=0.95), the distribution of the SR still follows an inhomogeneous poisson process, but the rate now varies due to white noise that also contains the effect of LRD. With the `ffGn(N, dt, H, SR)` function you can create both SR signals over time, one time for H=0.5 and one time for H=0.95. Note that in both cases (H=0.5 and H=0.95) the SR signal returned, has a mean of 0, to which you need to add the SR. Plot the time series of both noise signals, do you notice differences? This signal is equivalent to the $\Lambda(t)$ signal in the paper. Next, plot the autocorrelation function of the two noises (ACF, see previous exercise) and describe + interpret the differences between the two curves.
- Q6.2: In the second phase, feed the noise signal returned by the ffGn function as `in` to an inhomogenous integrate-and-fire neuron, which will output a list of spike times, given the noisy SR time-domain signal. The function works as follows: `spiketimes, PSref = inhomPP(in,dt)`. `PSref` is a control value that gives the poisson spike times that were on the basis of the inhomogeneous integrate-and-fire calculation. Compare the characteristics (ISI's) of the following models: a) inhomPP with H=0.5 input, b) inhomPP with H=0.95 input and c) the PSref basis spike times (those take a rate of 1 spike/s). Plot the three model's ISI's through a histogram with 1ms bins. Do your results align with expectations, why/why not?
- Fill in answer here

Q7: Investigate the research hypothesis

In the last step of this exercise, you should repeat the numerical experiment that Jackson and Carney performed to conclude that an inhomogeneous poisson process with 2/3 SR and includes LRD, is able to replicate the histogram characteristics (ISI's) of the reference AN SR distribution. You should also compare your outcomes with the figure when using Poisson-based (H=0.5) fGn as input to the inhomogeneous integrate-and-fire model.

- Q7.1: To obtain the histogram, you should compute the mean SR (spikes/s) in a repeated experiment, where you simulate the spike times in 30-s windows (using your encoder of choise) and repeat this experiment 150 times. You will need to use a for-loop for this. Consider the SRs of -20, 10 and 80 spikes/s and use FS = 1000 Hz. Compare your histograms with the reference figure and between Poisson, LRD models. Was their hypothesis sensible?

- Fill in answer here

Answers

A1: Statistical Models

- Go back to Q1

In [2]:
```
# use the following variables:
# ISI_low (ISI's for low light)
# ISI_high (ISI's for high light)
# prob_density_low (y axis values of the scaled histogram of ISI in 1ms bins, low l
# prob_density_high (y axis values of the scaled histogram of ISI in 1ms bins, high
# bin_edges
# bin_mids
# N_low (total number of ISI's, low light)
# N_high (total number of ISI's, high light)
# exp_pdf_low (exponential pdf, low light)
# exp_pdf_high (exponential pdf, high light)
```

In [3]:
```
# Q1
# load the data
data = sio.loadmat('matfiles/08_spikes-1.mat')  # Load the spike train data
# print(data.keys())


spikes_low = data['SpikesLow'][0]
spikes_high = data['SpikesHigh'][0]
```

In [4]:
```
# calculate the ISI's for both conditions
ISI_low = np.diff(spikes_low)
ISI_high = np.diff(spikes_high)
```

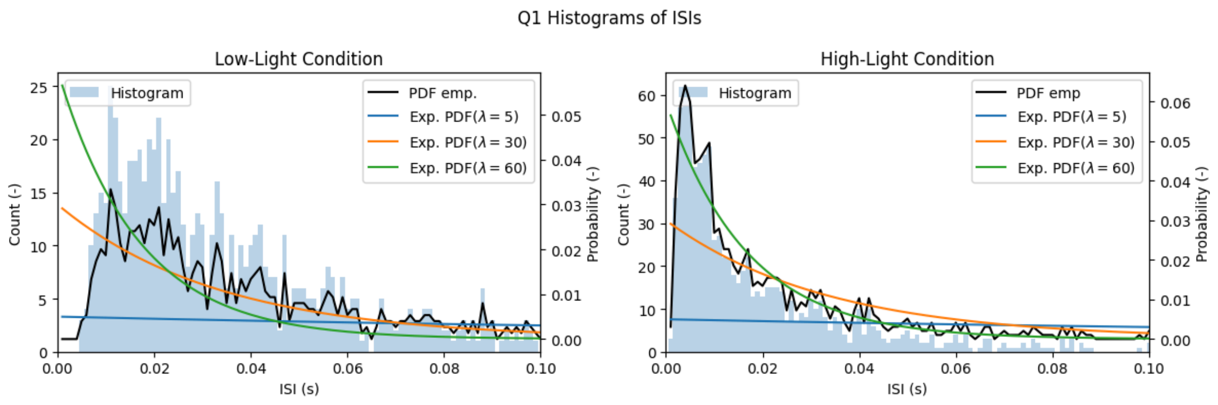
In [5]:
```
# Make a histogram of the actual data
dt = 1e-3
bin_edges_low = np.arange(0, ISI_low.max() + dt, dt)
```

```
bin_edges_high = np.arange(0, ISI_high.max() + dt, dt)

bin_mids_low = bin_edges_low[:-1] + np.diff(bin_edges_low)
bin_mids_high = bin_edges_high[:-1] + np.diff(bin_edges_high)

# get the histogram
ISI_hist_low, _ = np.histogram(ISI_low, bins = bin_edges_low)
ISI_hist_high, _ = np.histogram(ISI_high, bins = bin_edges_high)

# convert frequencies to probabilities
prob_density_low = ISI_hist_low / (np.sum(ISI_hist_low) * dt)
prob_density_high = ISI_hist_high / (np.sum(ISI_hist_high) * dt)

# check sum of probabilities
print('Sum of Probabilities High-Light condition:', np.sum(prob_density_high * dt))
print('Sum of Probabilities Low-Light condition:', np.sum(prob_density_low * dt))
```

```
Sum of Probabilities High-Light condition: 1.0
Sum of Probabilities Low-Light condition: 1.0000000000000002
```

In [6]:
```
# Fit an exponential pdf to the data
exp_pdf = lambda lmd, t: lmd * np.exp(-lmd * t)

lmds = [5, 30, 60]

exp_fits_low = [exp_pdf(lmd, bin_mids_low) for lmd in lmds]
exp_fits_high = [exp_pdf(lmd, bin_mids_high) for lmd in lmds]
```

In [7]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize =  (12, 4))

plt.suptitle('Q1 Histograms of ISIs')

# histogram low-light condition
ax1.set_title('Low-Light Condition')
ax1.bar(bin_mids_low, ISI_hist_low, alpha = 0.3, width = dt, label = 'Histogram')

# PDF low-light condition
ax12 = ax1.twinx()
ax12.plot(bin_mids_low, prob_density_low * dt, color = 'black', label = 'PDF emp.')

# Fitted exp-models
for i, exp_fit in enumerate(exp_fits_low):
    ax12.plot(bin_mids_low, exp_fit * dt, label = 'Exp. PDF' + r'($\lambda ' + f'=

# histogram high-light condition
ax2.set_title('High-Light Condition')
ax2.bar(bin_mids_high, ISI_hist_high, alpha = 0.3, width = dt, label = 'Histogram')

# PDF high-light condition
ax22 = ax2.twinx()
ax22.plot(bin_mids_high, prob_density_high * dt, color = 'black', label = 'PDF emp'

# Fitted exp-models
for i, exp_fit in enumerate(exp_fits_high):
    ax22.plot(bin_mids_high, exp_fit * dt, label = 'Exp. PDF' + r'($\lambda ' + f'=
for ax in (ax12, ax22):
    ax.set_ylabel('Probability (-)')
    ax.legend(loc = 'upper right')

for ax in (ax1, ax2):
    ax.set_xlabel('ISI (s)')
    ax.set_ylabel('Count (-)')
    ax.set_xlim((0, 0.1))
    ax.legend(loc = 'upper left')

plt.tight_layout()
plt.show()
```



A1 conclusion

- The probability density function (PDF) for inter-spike intervals (ISIs) in both low- and high-light conditions was estimated based on histogram data, with each ISI categorized into $\Delta t = 1$ ms bins. Each bin k represents a range of ISIs:

$$k\Delta t \leq \text{ISI} \leq (k+1)\Delta t, \qquad k \in \{0, 1, \ldots, M\},$$

where $M \in \mathrm{N}$ is approprialy chosen such that the maximum ISI value belongs to bin M. The PDF $\hat{f}(k)$ for bin k is calculated as

$$\hat{f}(k) = \frac{1}{\Delta t} p_k = \frac{c_k}{N\Delta t},$$

where $p_k = \frac{c_k}{N}$ is the probability of an ISI falling into bin k, c_k is the count of ISIs in bin k, and N is the total sample size (= `len(ISI_X`).

- The normalization of the PDF ensures that the area under the curve sums to 1, indicating the total probability:

$$\int_{-\infty}^{+\infty} f(x)\mathrm{d}x \approx \sum_{k=0}^{M} \hat{f}(k)\Delta t = \Delta t \sum_{k=0}^{M} \frac{1}{N\Delta t} c_k = \frac{1}{N} \sum_{k=0}^{M} c_k = \frac{1}{N} \cdot N = 1, \qquad x = \text{ISI}.$$

This was verified by summing the probabilities over all bins, yielding a value of 1.

- Histograms depict ISI counts within each bin on the left vertical axis, while PDFs are represented by curves, with corresponding probabilities shown on the right axis.

- Attempts were made to fit exponential curves to the data using various λ values ($\lambda \in \{5, 30, 60\}$) to visually align with the distribution. However, this approach yielded unsatisfactory results, indicating the limitations of the exponential model in capturing phenomena such as the refractory period. The exponential fit tends to overestimate probabilities for shorter ISIs, failing to account for the low counts in bins corresponding to refractory periods. Despite this, the exponential model better fits (but also not optimally) the right tail of the observed right-skewed distribution for higher ISI values.

A2: Likelihood functions

- Go back to Q2

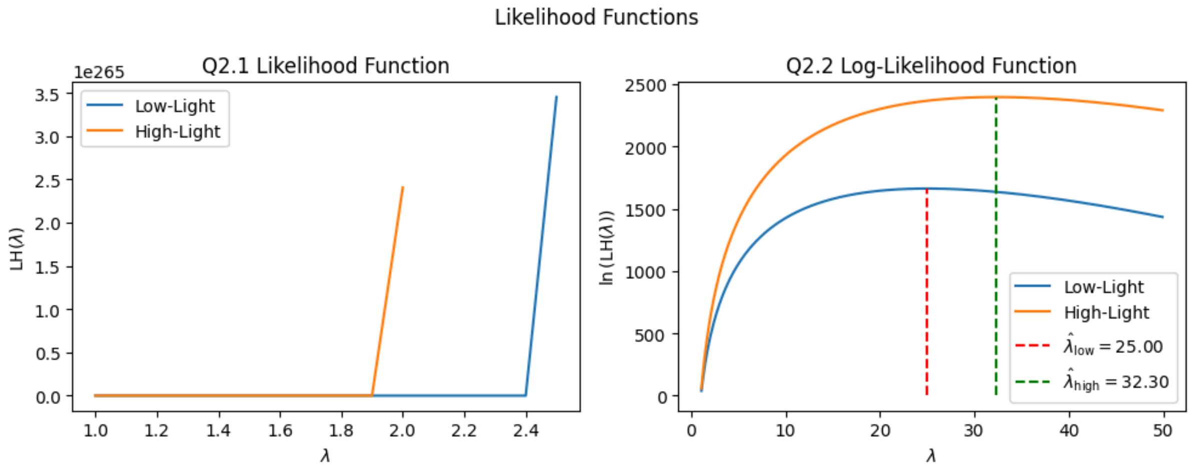
In [8]:
```
# hint, exclude lambda = 0 in your range of lambdas.

# Q2.1: use the following variables:
# likelihood_low      (for your likelihood model, low light)
# likelihood_high      (for your likelihood model, high light)

# Q2.2: use the following variables:

# log_likelihood_low (for your log_likelihood model, low light)
# log_likelihood_high (for your log_likelihood model, high light)

# max_likelihood_lambda_low (for the optimal lambda value, low light)
# max_likelihood_lambda_high (for the optimal lambda value, high light)

# Q2.3

# use the following parameters:
# N_all (N_low + N_high)
# mean_lambda_diff (1 digit value)
# sample_diff (array with 1000 difference values)
```

In [9]:
```
# Q2.1 & Q2.2

# amount of samples
N_low, N_high = len(ISI_low), len(ISI_high)

# LH functions
likelihood_low = lambda l: (l)**N_low * np.exp(-l*np.sum(ISI_low))
likelihood_high = lambda l: (l)**N_high * np.exp(-l*np.sum(ISI_high))

# take the log-TF of LH functions
log_likelihood_low = lambda l: N_low * np.log(l) - l*np.sum(ISI_low)
log_likelihood_high = lambda l: N_high * np.log(l) - l*np.sum(ISI_high)

# range of lambdas
```

```
lambdas = np.arange(1.0, 50.0, 0.1)

# get the maximal log-likelihood
max_likelihood_low = np.max(log_likelihood_low(lambdas[1:]))
max_likelihood_high = np.max(log_likelihood_high(lambdas[1:]))

# get the maximum likelihood estimate
max_likelihood_index_low = np.argmax(log_likelihood_low(lambdas[1:]))
max_likelihood_lambda_low = lambdas[max_likelihood_index_low + 1]

max_likelihood_index_high = np.argmax(log_likelihood_high(lambdas[1:]))
max_likelihood_lambda_high = lambdas[max_likelihood_index_high + 1]

# get the mean values of ISIs
mean_ISI_low = ISI_low.mean()
mean_ISI_high = ISI_high.mean()
```

In [10]:
```
# visualize the LH functions
fig, (ax1, ax2) = plt.subplots(1, 2, figsize =  (10, 4))

plt.suptitle('Likelihood Functions')

ax1.set_title('Q2.1 Likelihood Function')
ax1.plot(lambdas, likelihood_low(lambdas), label = 'Low-Light')
ax1.plot(lambdas, likelihood_high(lambdas), label = 'High-Light')

ax1.set_xlabel(r'$\lambda$')
ax1.set_ylabel(r'LH($\lambda$)')
ax1.legend()

ax2.set_title('Q2.2 Log-Likelihood Function')
#ax2.axhline(0, color = 'black', ls = '-')

ax2.plot(lambdas[1:], log_likelihood_low(lambdas[1:]), label = 'Low-Light')   # ex
ax2.plot(lambdas[1:], log_likelihood_high(lambdas[1:]), label = 'High-Light') # ex

ax2.vlines(x = max_likelihood_lambda_low, ymin = 0, ymax = max_likelihood_low, col
ax2.vlines(x = max_likelihood_lambda_high, ymin = 0, ymax = max_likelihood_high, c


ax2.set_xlabel(r'$\lambda$')
ax2.set_ylabel(r'$\ln\left(\mathrm{LH}(\lambda)\right)$')

ax2.legend()

plt.tight_layout()
plt.show()
```

```
C:\Users\coppe\AppData\Local\Temp\ipykernel_10864\4079048045.py:7: RuntimeWarning:
overflow encountered in power
  likelihood_low = lambda l: (l)**N_low * np.exp(-l*np.sum(ISI_low))
C:\Users\coppe\AppData\Local\Temp\ipykernel_10864\4079048045.py:7: RuntimeWarning:
invalid value encountered in multiply
  likelihood_low = lambda l: (l)**N_low * np.exp(-l*np.sum(ISI_low))
C:\Users\coppe\AppData\Local\Temp\ipykernel_10864\4079048045.py:8: RuntimeWarning:
overflow encountered in power
  likelihood_high = lambda l: (l)**N_high * np.exp(-l*np.sum(ISI_high))
C:\Users\coppe\AppData\Local\Temp\ipykernel_10864\4079048045.py:8: RuntimeWarning:
invalid value encountered in multiply
  likelihood_high = lambda l: (l)**N_high * np.exp(-l*np.sum(ISI_high))
```



In [11]:
```
print(f'Maximal Log-Likelihood of Low-Light condition: {max_likelihood_low:.2f}')
print(f'Maximal Log-Likelihood of High-Light condition: {max_likelihood_high:.2f}
```

```
Maximal Log-Likelihood of Low-Light condition: 1662.16
Maximal Log-Likelihood of High-Light condition: 2396.42
```

In [12]:
```
c = np.log(10)
ML_low, ML_high = max_likelihood_low / c, max_likelihood_high / c
LH_low, LH_high = ML_low - int(ML_low), ML_high - int(ML_high)

print(f'Maximal Likelihood of Low-Light condition: {10**(LH_low):.2f} * 10^{int(ML
print(f'Maximal Likelihood of High-Light condition: {10**(LH_high):.2f} * 10^{int(
```

```
Maximal Likelihood of Low-Light condition: 7.33 * 10^721
Maximal Likelihood of High-Light condition: 5.65 * 10^1040
```

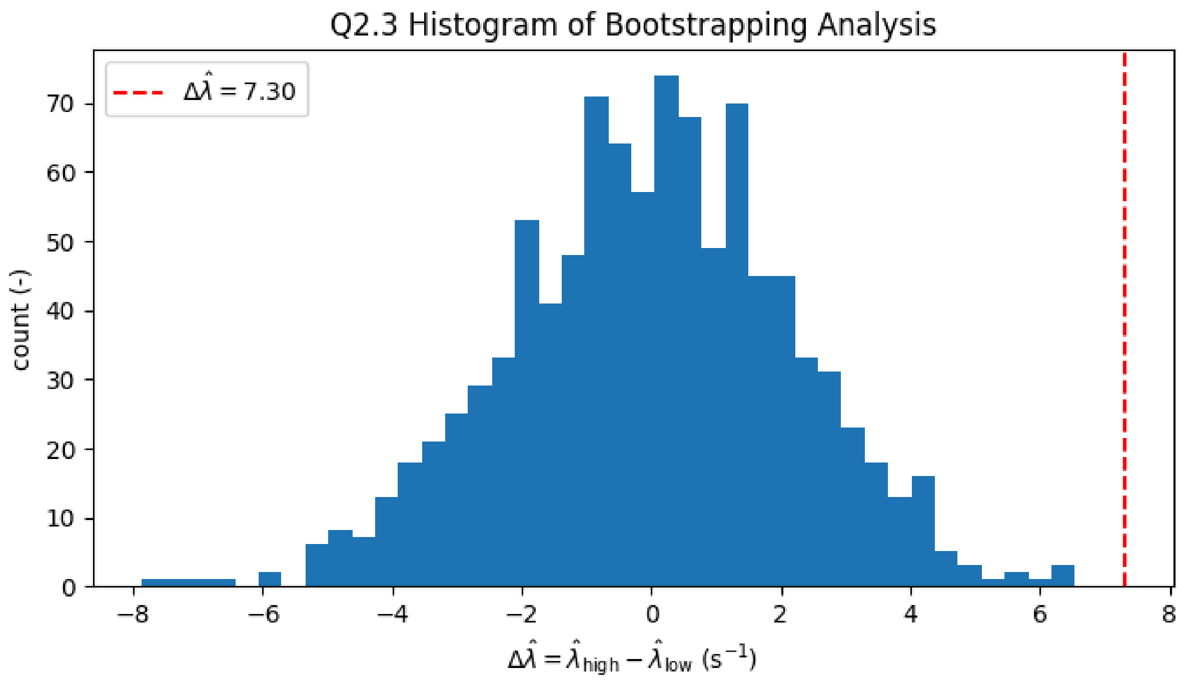
In [13]:
```
print(f'Maximum Likelihood Estimate of Low-Light condition: {max_likelihood_lambda
print(f'Maximum Likelihood Estimate of High-Light condition: {max_likelihood_lamb
```

```
Maximum Likelihood Estimate of Low-Light condition: 25.00 Hz
Maximum Likelihood Estimate of High-Light condition: 32.30 Hz
```

In [14]:
```
print(f'Inverse mean of ISI Low-Light condition: {1/mean_ISI_low:.2f} Hz')
print(f'Inverse mean of ISI High-Light condition: {1/mean_ISI_high:.2f} Hz')
```

```
Inverse mean of ISI Low-Light condition: 25.01 Hz
Inverse mean of ISI High-Light condition: 32.32 Hz
```

In [15]:
```
# Q2.3 Bootstrap Analysis
# 1. Pool the ISIs
ISI_pooled = np.concatenate((ISI_low, ISI_high))

# 2. Select datasets
n = 1000
N_high, N_low, N_all = len(ISI_high), len(ISI_low), len(ISI_pooled)
sample_diff = [1 / np.mean(ISI_pooled[np.random.randint(N_all, size = N_high)]) -

# 3. actual difference
mean_lambda_diff = max_likelihood_lambda_high - max_likelihood_lambda_low
#print(f'Mean difference of lambda: {mean_lambda_diff:.2f}')

# 4. plot the histogram
fig, ax = plt.subplots(1, 1, figsize = (8, 4))

ax.set_title('Q2.3 Histogram of Bootstrapping Analysis')

ax.hist(sample_diff, bins = 40)
ax.axvline(x = mean_lambda_diff, ls = '--', color = 'red', label  = r'$\Delta\hat{

ax.set_xlabel(r'$\Delta\hat{\lambda} = \hat{\lambda}_{\mathrm{high}} - \hat{\lambd
ax.set_ylabel('count (-)')

ax.legend()
plt.show()
```



A2.1 conclusion

The outcome of the likelihood function $L(\lambda)$ raises concerns due to a `RuntimeWarning`, indicating limitations in the computational capability of my system. The calculation involves exponentiating λ to high powers (`len(ISI_low)` = 749), surpassing the numerical precision limits. As a result, the accuracy of the likelihood function assessment is compromised. The solution to this is taking the log-transform of our likelihood function.

A2.2 conclusion (Optional)

The obtained maximum likelihood estimates (MLE) are

$$\lambda_{\text{low}} = 25.00 \text{ Hz}, \qquad \lambda_{\text{high}} = 32.30 \text{ Hz}.$$

This result is logic: they correspond to the firing rates calculated in Q1 of Ex. 1 (module 2).

A2.3 conclusion

The bootstrap analysis reveals a non-significant difference in the Poisson rate parameter between the low- and high-light conditions, as indicated by the mean difference of λ lying outside the bootstrapped histogram. Consistent with expectations, the Poisson rate parameter of the high-light condition is found to be significantly higher than that of the low-light condition. This implies a higher spike rate in the high-light condition.

A3: Goodness of Fit

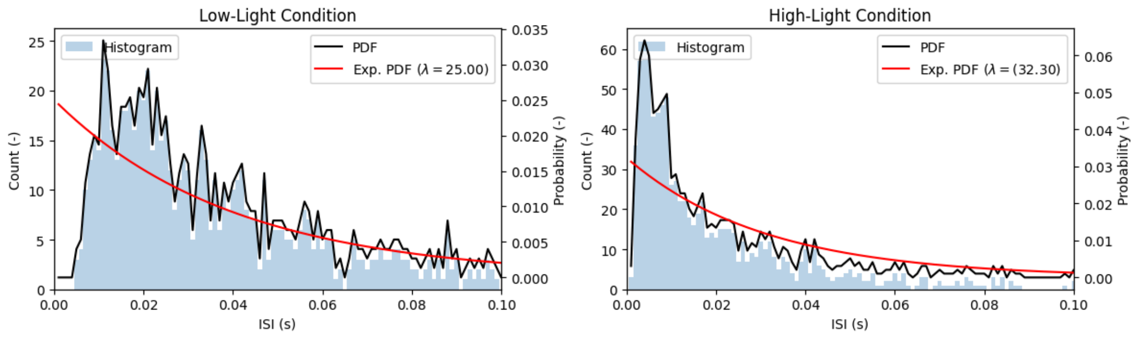
- Go back to Q3

In [16]:
```
# use the following parameters
# ISI_low (ISI's for low light)
# ISI_high (ISI's for high light)
# prob_density_low (y axis values of the scaled histogram of ISI in 1ms bins,
# prob_density_high (y axis values of the scaled histogram of ISI in 1ms bins,
# bin_edges
# bin_mids
# N_low (total number of ISI's, low light)
# N_high (total number of ISI's, high light)
# exp_pdf_low (exponential pdf, low light)
# exp_pdf_high (exponential pdf, high light)
# CDF_exp_low (the CDF function of the exponential pdf, low light)
# CDF_exp_high (the CDF function of the exponential pdf, high light)
# CDF_emp_low (empirical values, low light)
# CDF_emp_high (empirical values, high light)

# Check the integral (sum of all y-values * dt = 1)
```

In [17]:
```
# Q3.1
fig, (ax1, ax2) = plt.subplots(1, 2, figsize =  (12, 4))

plt.suptitle('Q3.1 Histograms of ISIs with MLE fitted exp. distribtion')
```

```
# histogram low-light condition
ax1.set_title('Low-Light Condition')
ax1.bar(bin_mids_low, ISI_hist_low, alpha = 0.3, width = dt, label = 'Histogra

# PDF low-light condition
ax12 = ax1.twinx()
ax12.plot(bin_mids_low, prob_density_low * dt, color = 'black', label = 'PDF'

# Fitted exp-models
ax12.plot(bin_mids_low, exp_pdf(max_likelihood_lambda_low, bin_mids_low)* dt,

# histogram high-light condition
ax2.set_title('High-Light Condition')
ax2.bar(bin_mids_high, ISI_hist_high, alpha = 0.3, width = dt, label = 'Histog

# PDF high-light condition
ax22 = ax2.twinx()
ax22.plot(bin_mids_high, prob_density_high * dt, color = 'black', label = 'PD

# Fitted exp-models
ax22.plot(bin_mids_high, exp_pdf(max_likelihood_lambda_high, bin_mids_high)* 

for ax in (ax12, ax22):
    ax.set_ylabel('Probability (-)')
    ax.legend(loc = 'upper right')

for ax in (ax1, ax2):
    ax.set_xlabel('ISI (s)')
    ax.set_ylabel('Count (-)')
    ax.set_xlim((0, 0.1))
    ax.legend(loc = 'upper left')

plt.tight_layout()
plt.show()
```

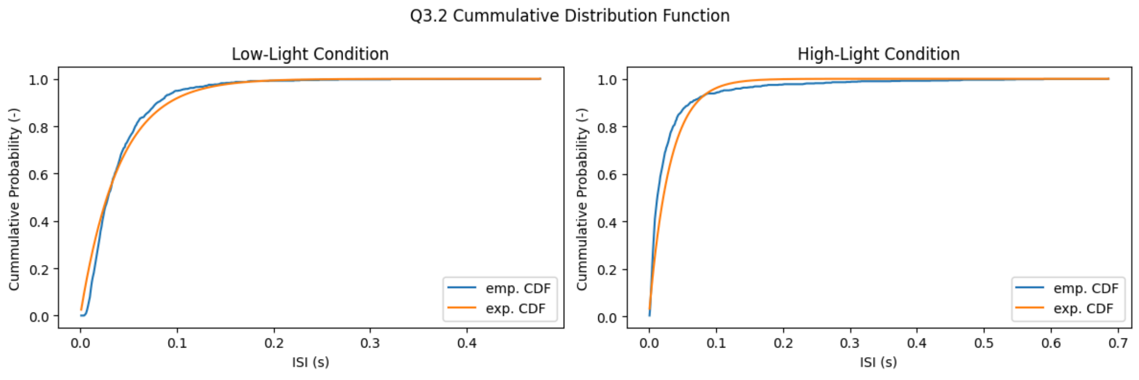
Q3.1 Histograms of ISIs with MLE fitted exp. distribtion



```
In [18]: # check normality
print('Normality Exp. fit Low:', np.sum(exp_pdf(max_likelihood_lambda_low, bi
print('Normality Exp. fit High:', np.sum(exp_pdf(max_likelihood_lambda_high, 
```

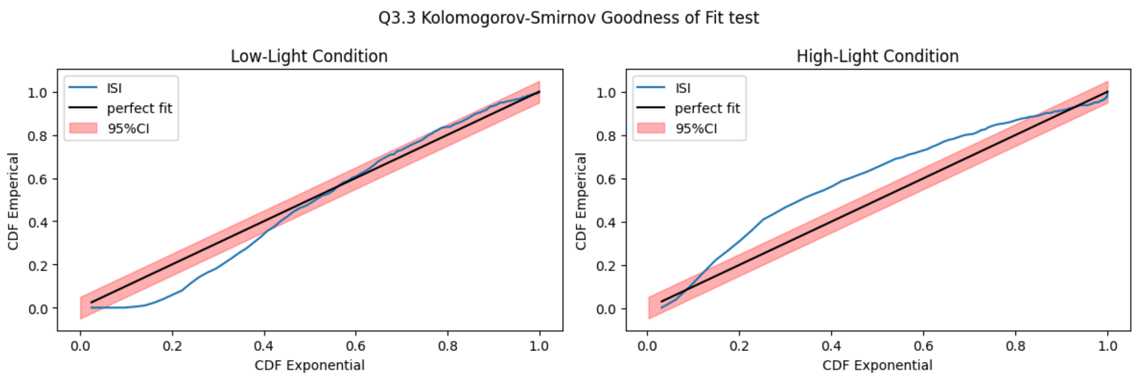
```
Normality Exp. fit Low: 0.9875453769123962
Normality Exp. fit High: 0.9839369390872297
```

In [19]:
```python
# Q3.2
# exp. model
exp_cdf = lambda l, x: 1 - np.exp(-l*x)
exp_cdf_low = exp_cdf(max_likelihood_lambda_low, bin_mids_low)
exp_cdf_high = exp_cdf(max_likelihood_lambda_high, bin_mids_high)

# emperical
cdf_low = np.cumsum(prob_density_low * dt)
cdf_high = np.cumsum(prob_density_high * dt)

# make a plot
fig, (ax1, ax2) = plt.subplots(1, 2)

plt.suptitle('Q3.2 Cummulative Distribution Function')

ax1.set_title('Low-Light Condition')
ax1.plot(bin_mids_low, cdf_low, label = 'emp. CDF')
ax1.plot(bin_mids_low, exp_cdf_low, label = 'exp. CDF')

ax2.set_title('High-Light Condition')
ax2.plot(bin_mids_high, cdf_high, label = 'emp. CDF')
ax2.plot(bin_mids_high, exp_cdf_high, label = 'exp. CDF')

for ax in (ax1, ax2):
    ax.set_xlabel('ISI (s)')
    ax.set_ylabel('Cummulative Probability (-)')
    ax.legend()

plt.tight_layout()
plt.show()
```



In [20]:
```python
# Q3.3
# make a plot
fig, (ax1, ax2) = plt.subplots(1, 2)

plt.suptitle('Q3.3 Kolomogorov-Smirnov Goodness of Fit test')

ax1.set_title('Low-Light Condition')
ax1.plot(exp_cdf_low, cdf_low, label = 'ISI')
ax1.plot(exp_cdf_low, exp_cdf_low, color = 'black', label = 'perfect fit')
ax1.fill_between(cdf_low, cdf_low + 1.36/np.sqrt(N_low), cdf_low - 1.36/np.sq

ax2.set_title('High-Light Condition')
```

```
ax2.plot(exp_cdf_high, cdf_high, label = 'ISI')
ax2.plot(exp_cdf_high, exp_cdf_high, color = 'black', label = 'perfect fit')
ax2.fill_between(cdf_high, cdf_high + 1.36 / np.sqrt(N_low), cdf_high - 1.36 

for ax in (ax1, ax2):
    ax.set_xlabel('CDF Exponential')
    ax.set_ylabel('CDF Emperical')
    ax.legend()

plt.tight_layout()
plt.show()
```



A3.1 conclusion

The fit between the ISI histogram and the exponential PDF is not optimal. While the exponential PDF exhibits an exponentially decreasing trend, it fails to accurately capture the characteristics of the ISI histogram. Specifically, the exponential fit tends to overestimate the occurrence of lower ISI values, indicating a discrepancy in capturing the refractory period. Conversely, the relationship between the model and observed data holds reasonably well for higher ISI values. However, for lower ISI values, the Poisson process underlying the exponential PDF underestimates the frequency of occurrence, failing to account for bursting phenomena. These limitations arise from the assumption of independence among ISI values inherent in a Poisson process, neglecting the influence of refractory periods and bursting behavior on spike generation.

A3.2 conclusion

The findings from the CDF plot validate our visual assessment. A closer alignment between the model and empirical data is observed for higher ISI values, indicating a relatively good fit. However, discrepancies become evident for lower ISI values, where the fit is less accurate. Notably, the low-light condition exhibits a comparatively better fit than the high-light condition. In the high-light condition, the CDF consistently maintains higher values (bursting), suggesting a less precise match between the model and observed data.

A3.3 conclusion

Based on the plots provided, it is evident that the fit for both - and high-light intensities is unsatisfactory. The KS curve deviates significantly from the confidence bounds in both conditions, indicating poor agreement between the model and empirical distributions. In the low-light condition, the deviation from the confidence boundary is notably smaller compared to the high-light condition. Despite this, neither condition demonstrates a sufficiently accurate fit based on the KS plot analysis.

A4: More advanced, inverse Gaussian probability models

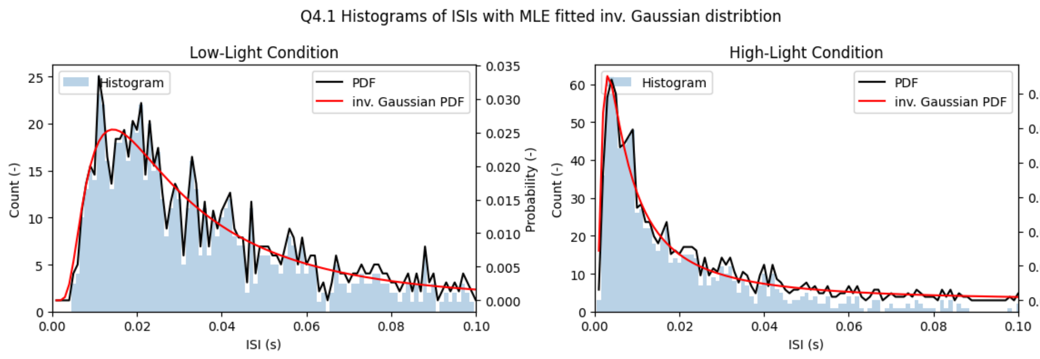
- Go back to Q4

In [21]:
```
# use the following parameters:
# mu_low (mean of the ISI's... the mu of the inverse gaussian, low light)
# mu_high (mean of the ISI's... the mu of the inverse gaussian, high ligh
# lambda_low (shape parameter of the inverse gaussion, low light)
# lambda_high (shape parameter of the inverse gaussion, low light)
# inv_gaussian_low (inverse gaussion model, low light)
# inv_gaussian_high (inverse gaussion model, high light)
# CDF_invgauss_low (CDF of inverse gaussian function, low light)
# CDF_invgauss_high (CDF of inverse gaussian function, high light)

def get_shape_parm(x):
    mu = x.mean()
    N = x.shape[0]
    return 1 / (np.sum(1/x - 1/mu) / N)

def fit_inv_gaussian(x, mu, l):
    return np.sqrt(l / (2*np.pi*x**3)) * np.exp(- (l*(x - mu)**2) / (2*x*
```

In [22]:
```
# Q4.1
# ML Estimates
## mean
mu_low, mu_high = ISI_low.mean(), ISI_high.mean()

## shape paramters
lambda_low, lambda_high = get_shape_parm(ISI_low), get_shape_parm(ISI_hig

# PDFs
inv_gaussian_low, inv_gaussian_high = fit_inv_gaussian(bin_mids_low, mu_l

# CDFs
CDF_invgauss_low, CDF_invgauss_high = np.cumsum(inv_gaussian_low * dt), n
```

In [23]:
```
# Q4.1
fig, (ax1, ax2) = plt.subplots(1, 2, figsize =  (12, 4))
```

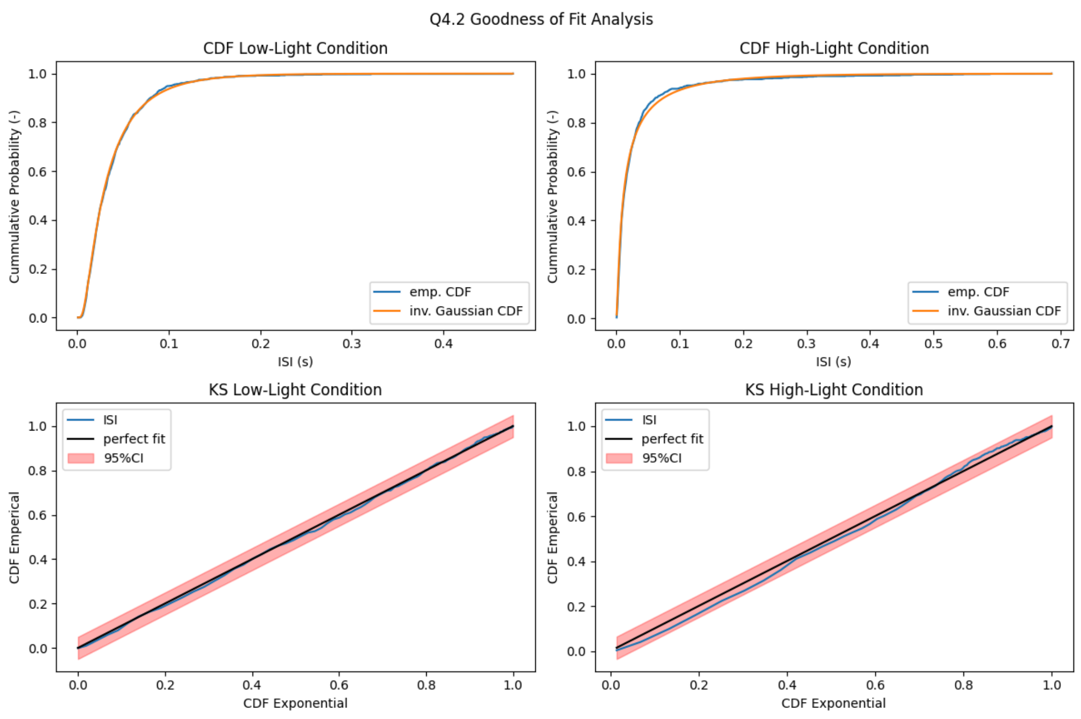
```
plt.suptitle('Q4.1 Histograms of ISIs with MLE fitted inv. Gaussian distr

# histogram low-light condition
ax1.set_title('Low-Light Condition')
ax1.bar(bin_mids_low, ISI_hist_low, alpha = 0.3, width = dt, label = 'His

# PDF low-light condition
ax12 = ax1.twinx()
ax12.plot(bin_mids_low, prob_density_low * dt, color = 'black', label = '

# Fitted inv Gaussian model
ax12.plot(bin_mids_low, inv_gaussian_low * dt, color = 'red', label = f'i

# histogram high-light condition
ax2.set_title('High-Light Condition')
ax2.bar(bin_mids_high, ISI_hist_high, alpha = 0.3, width = dt, label = 'H

# PDF high-light condition
ax22 = ax2.twinx()
ax22.plot(bin_mids_high, prob_density_high * dt, color = 'black', label =

# Fitted inv Gaussian model
ax22.plot(bin_mids_high, inv_gaussian_high * dt, color = 'red', label = f

for ax in (ax12, ax22):
    ax.set_ylabel('Probability (-)')
    ax.legend(loc = 'upper right')

for ax in (ax1, ax2):
    ax.set_xlabel('ISI (s)')
    ax.set_ylabel('Count (-)')
    ax.set_xlim((0, 0.1))
    ax.legend(loc = 'upper left')

plt.tight_layout()
plt.show()
```



In [24]:
```
# check normality
print('Normality inv. Gaussian Low:', np.sum(inv_gaussian_low * dt))
print('Normality inv. Gaussian High:', np.sum(inv_gaussian_high * dt))
```

```
Normality inv. Gaussian Low: 0.9999689476168446
Normality inv. Gaussian High: 0.9997049537230472
```

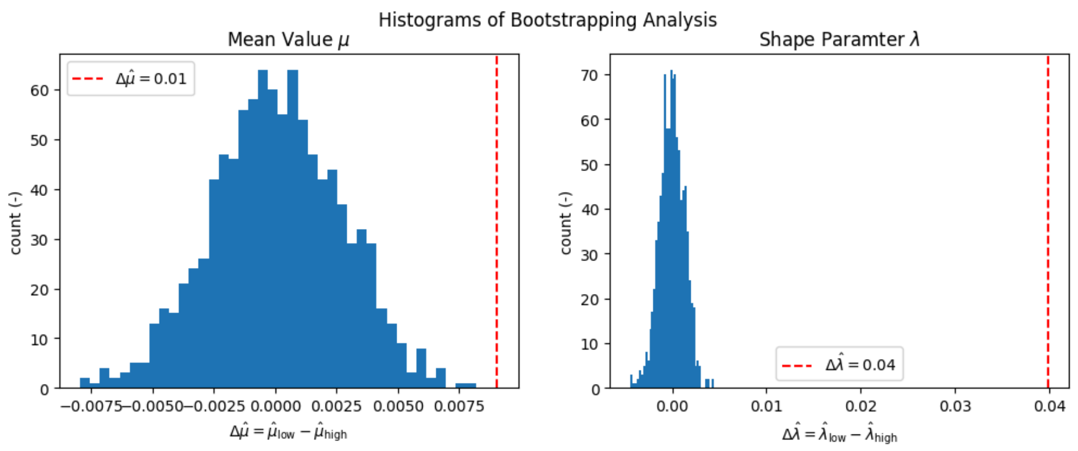
In [25]:
```
# make a plot
fig, (ax_1, ax_2) = plt.subplots(2, 2, figsize = (12, 4*2))

plt.suptitle('Q4.2 Goodness of Fit Analysis')

ax1 = ax_1[0]
ax1.set_title('CDF Low-Light Condition')
ax1.plot(bin_mids_low, cdf_low, label = 'emp. CDF')
ax1.plot(bin_mids_low, CDF_invgauss_low, label = 'inv. Gaussian CDF')

ax2 = ax_1[1]
ax2.set_title('CDF High-Light Condition')
ax2.plot(bin_mids_high, cdf_high, label = 'emp. CDF')
ax2.plot(bin_mids_high, CDF_invgauss_high, label = 'inv. Gaussian CDF')

for ax in (ax1, ax2):
    ax.set_xlabel('ISI (s)')
    ax.set_ylabel('Cummulative Probability (-)')
    ax.legend()

ax3 = ax_2[0]
ax3.set_title('KS Low-Light Condition')
ax3.plot(CDF_invgauss_low, cdf_low, label = 'ISI')
ax3.plot(CDF_invgauss_low, CDF_invgauss_low, color = 'black', label = 'pe
ax3.fill_between(CDF_invgauss_low, CDF_invgauss_low + 1.36/np.sqrt(N_low)

ax4 = ax_2[1]
ax4.set_title('KS High-Light Condition')
ax4.plot(CDF_invgauss_high, cdf_high, label = 'ISI')
ax4.plot(CDF_invgauss_high, CDF_invgauss_high, color = 'black', label = '
ax4.fill_between(CDF_invgauss_high, CDF_invgauss_high + 1.36 / np.sqrt(N_

for ax in (ax3, ax4):
    ax.set_xlabel('CDF Exponential')
    ax.set_ylabel('CDF Emperical')
    ax.legend()


plt.tight_layout()
plt.show()
```



In [26]:
```
# Q4.3
# 1. Pool the ISIs
ISI_pooled = np.concatenate((ISI_low, ISI_high))

# 2. parameters
n = 1000
N_high, N_low, N_all = len(ISI_high), len(ISI_low), len(ISI_pooled)

# 3. actual difference
mean_lambda_diff = lambda_low - lambda_high
mean_mu_diff = mu_low - mu_high

# 4. Bootstrapping
## of mu
mu_diff = [ISI_pooled[np.random.randint(N_all, size = N_low)].mean() - IS

## of lambda
lambda_diff = [get_shape_parm(ISI_pooled[np.random.randint(N_all, size =
```

In [27]:
```
# 5. plot the histogram
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (12, 4))

plt.suptitle('Histograms of Bootstrapping Analysis')


ax1.set_title(r'Mean Value $\mu$')
ax1.hist(mu_diff, bins = 40)
ax1.axvline(x = mean_mu_diff, ls = '--', color = 'red', label  = r'$\Delt

ax1.set_xlabel(r'$\Delta\hat{\mu} = \hat{\mu}_{\mathrm{low}} - \hat{\mu}_
ax1.set_ylabel('count (-)')
```

```
ax2.set_title(r'Shape Paramter $\lambda$')
ax2.hist(lambda_diff, bins = 40)
ax2.axvline(x = mean_lambda_diff, ls = '--', color = 'red', label  = r'$\

ax2.set_xlabel(r'$\Delta\hat{\lambda} = \hat{\lambda}_{\mathrm{low}} - \h
ax2.set_ylabel('count (-)')

for ax in (ax1, ax2):
    ax.legend()

plt.show()
```



In [28]:
```
print('Low-Light Condition')
print(f'Mean value: {mu_low}' + f'\tShape Parameter: {lambda_low}')

print('\nHigh-Light Condition')
print(f'Mean value: {mu_high}' + f'\tShape Parameter: {lambda_high}')
```

```
Low-Light Condition
Mean value: 0.039988397284383186        Shape Parameter: 0.049318167692539
32

High-Light Condition
Mean value: 0.030941974963219623        Shape Parameter: 0.009498135387175
857
```

A4 conclusion

In contrast to the limitations of the exponential fit in capturing refractoriness and bursting features in the ISI, the inverse Gaussian model demonstrates improved capability in capturing these phenomena. Therefore, the inverse Gaussian model provides a satisfactory fit for both low- and high-light conditions. Inspection of the KS plots confirms the adequacy of the fit for both intensity levels, as they lie comfortably within the confidence interval.

A significant discrepancy is observed in the shape parameter values λ between the low- and high-light conditions, with the latter exhibiting lower values. This discrepancy can be attributed to the prevalence of bursting patterns in the high-light condition, leading to shorter ISI intervals (and thus a lower $\lambda \rightarrow$ smaller shape). Additionally, there is a considerable difference in mean μ between the two conditions, with the low-light condition having a higher average mean due to more spread-out ISIs (less/no bursting).

A5: From Decoding to Encoding models

- Go back to Q5

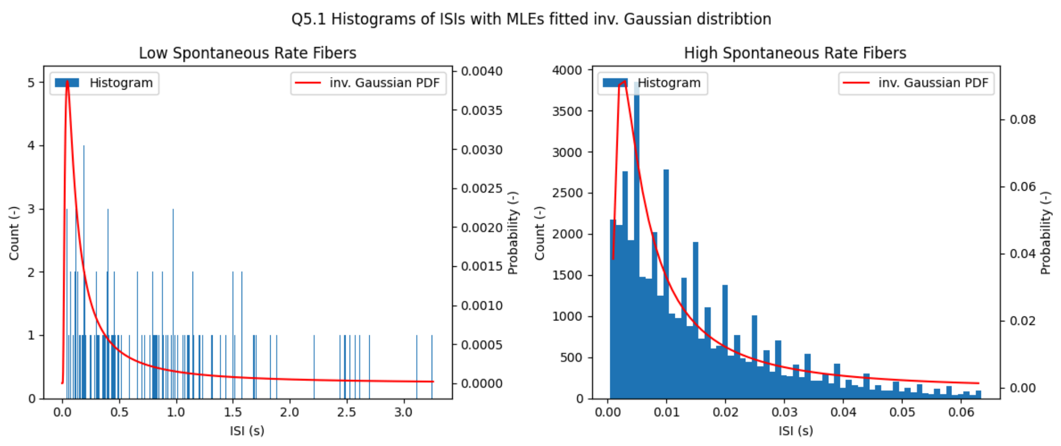
In [29]:
```
# Load the data
data = sio.loadmat('matfiles/ANspikes.mat')

fibers_low = data['LS'][0]
fibers_high = data['HS'][0]
```

In [30]:
```
# Q5.1
# get the ISIs
ISI_low = np.diff(fibers_low)
ISI_high = np.diff(fibers_high)

# Make a histogram of the actual data
dt_low = 1e-3 #* 0.5
dt_high = 1e-3 #* 0.5

#max_bin_low, max_bin_high = ISI_low.max(), ISI_high.max()
max_bin_low, max_bin_high = np.percentile(ISI_low, 97.5), np.percentile(
bin_edges_low = np.arange(0, max_bin_low + dt_low, dt_low)
bin_edges_high = np.arange(0, max_bin_high + dt_high, dt_high)

bin_mids_low = bin_edges_low[:-1] + np.diff(bin_edges_low)
bin_mids_high = bin_edges_high[:-1] + np.diff(bin_edges_high)

# get the histogram
ISI_hist_low, _ = np.histogram(ISI_low, bins = bin_edges_low)
ISI_hist_high, _ = np.histogram(ISI_high, bins = bin_edges_high)

# convert frequencies to probabilities
prob_density_low = ISI_hist_low / np.sum(ISI_hist_low)
prob_density_high = ISI_hist_high / np.sum(ISI_hist_high)

# check sum of probabilities
print('Sum of Probabilities High-Light condition:', np.sum(prob_density_
print('Sum of Probabilities Low-Light condition:', np.sum(prob_density_l
```

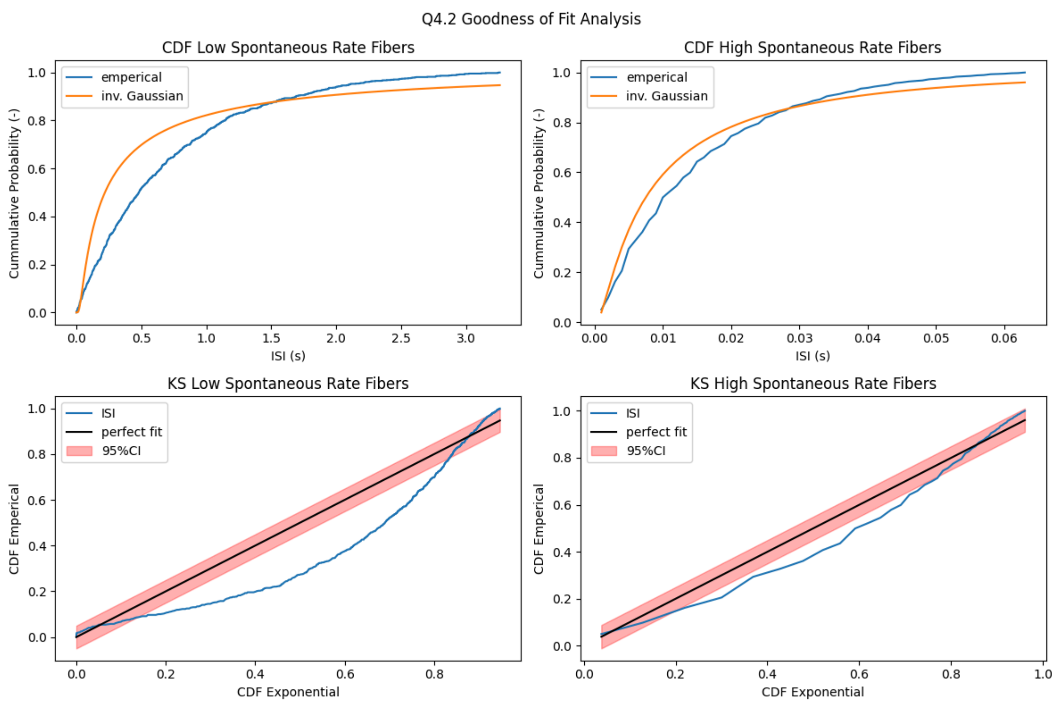
```
Sum of Probabilities High-Light condition: 0.9999999999999998
Sum of Probabilities Low-Light condition: 1.0
```

In [31]:
```
# fit an inv. Gaussian
# ML Estimates
## mean
mu_low, mu_high = ISI_low.mean(), ISI_high.mean()

## shape paramters
lambda_low, lambda_high = get_shape_parm(ISI_low), get_shape_parm(ISI_hi
# PDFs
inv_gaussian_low, inv_gaussian_high = fit_inv_gaussian(bin_mids_low, mu_

# CDFs
CDF_invgauss_low, CDF_invgauss_high = np.cumsum(inv_gaussian_low * dt_lo

# CDF emperical
cdf_low, cdf_high = np.cumsum(prob_density_low), np.cumsum(prob_density_
```

In [35]:
```
# Q4.1
fig, (ax1, ax2) = plt.subplots(1, 2, figsize =  (12, 5))

plt.suptitle('Q5.1 Histograms of ISIs with MLEs fitted inv. Gaussian dis

# histogram low-light condition
ax1.set_title('Low Spontaneous Rate Fibers')
ax1.bar(bin_mids_low, ISI_hist_low, width = dt_low, label = 'Histogram')

# PDF low-light condition
ax12 = ax1.twinx()
#ax121.plot(bin_mids_low, prob_density_low, color = 'black', label = 'PD

# Fitted inv Gaussian model
ax12.plot(bin_mids_low, inv_gaussian_low * dt_low, color = 'red', label

# histogram high-light condition
ax2.set_title('High Spontaneous Rate Fibers')
ax2.bar(bin_mids_high, ISI_hist_high, width = dt_high, label = 'Histogra

# PDF high-light condition
ax22 = ax2.twinx()
#ax221.plot(bin_mids_high, prob_density_high, color = 'black', label = '

# Fitted inv Gaussian model
ax22.plot(bin_mids_high, inv_gaussian_high * dt_high, color = 'red', lab

#ax1.set_xlim((0, 4))
for ax in (ax12, ax22):
    ax.set_ylabel('Probability (-)')
    ax.legend(loc = 'upper right')

for ax in (ax1, ax2):
    ax.set_xlabel('ISI (s)')
    ax.set_ylabel('Count (-)')
    #ax.set_xlim((0, 0.2))
    ax.legend(loc = 'upper left')

plt.tight_layout()
plt.show()
```



In [36]:
```python
# GoF
fig, (ax_1, ax_2) = plt.subplots(2, 2, figsize = (12, 4*2))

plt.suptitle('Q4.2 Goodness of Fit Analysis')

ax1 = ax_1[0]
ax1.set_title('CDF Low Spontaneous Rate Fibers')
ax1.plot(bin_mids_low, cdf_low, label = 'emperical')
ax1.plot(bin_mids_low, CDF_invgauss_low, label = 'inv. Gaussian')

ax2 = ax_1[1]
ax2.set_title('CDF High Spontaneous Rate Fibers')
ax2.plot(bin_mids_high, cdf_high, label = 'emperical')
ax2.plot(bin_mids_high, CDF_invgauss_high, label = 'inv. Gaussian')

for ax in (ax1, ax2):
    ax.set_xlabel('ISI (s)')
    ax.set_ylabel('Cummulative Probability (-)')
    ax.legend()

ax3 = ax_2[0]
ax3.set_title('KS Low Spontaneous Rate Fibers')
ax3.plot(CDF_invgauss_low, cdf_low, label = 'ISI')
ax3.plot(CDF_invgauss_low, CDF_invgauss_low, color = 'black', label = 'p
ax3.fill_between(CDF_invgauss_low, CDF_invgauss_low + 1.36/np.sqrt(N_low

ax4 = ax_2[1]
ax4.set_title('KS High Spontaneous Rate Fibers')
ax4.plot(CDF_invgauss_high, cdf_high, label = 'ISI')
ax4.plot(CDF_invgauss_high, CDF_invgauss_high, color = 'black', label =
ax4.fill_between(CDF_invgauss_high, CDF_invgauss_high + 1.36 / np.sqrt(N

for ax in (ax3, ax4):
    ax.set_xlabel('CDF Exponential')
    ax.set_ylabel('CDF Emperical')
    ax.legend()


plt.tight_layout()
plt.show()
```

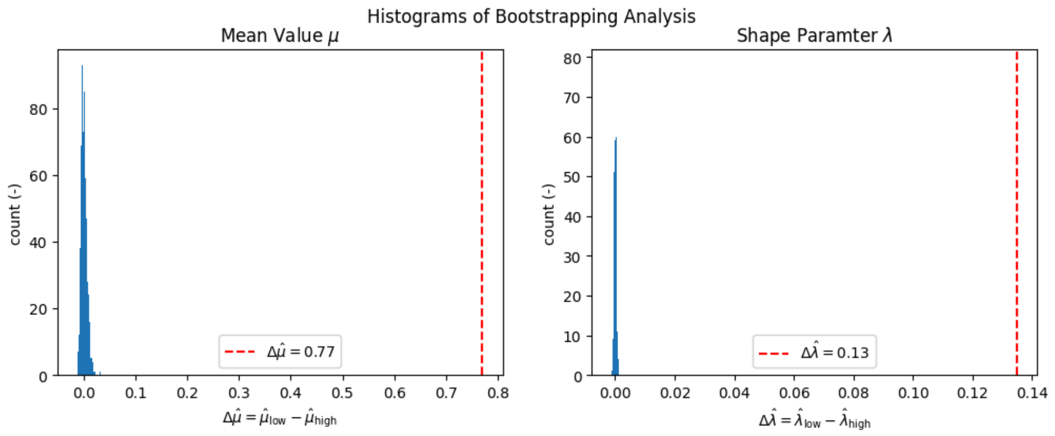


In [37]:
```
# Investigation of the distribution parametes
lambda_low, lambda_high = get_shape_parm(ISI_low), get_shape_parm(ISI_hi
mu_low, mu_high = ISI_low.mean(), ISI_high.mean()

print('LSR Fibers')
print(f'Mean: {mu_low:.2f}' + f'\tShape Parameter: {lambda_low:.2f}')

print('\nHSR Fibers')
print(f'Mean: {mu_high:.2f}' + f'\tShape Parameter: {lambda_high:.2f}')
```

```
LSR Fibers
Mean: 0.79      Shape Parameter: 0.14

HSR Fibers
Mean: 0.02      Shape Parameter: 0.01
```

In [38]:
```
# Q4.3
# 1. Pool the ISIs
ISI_pooled = np.concatenate((ISI_low, ISI_high))

# 2. parameters
n = 1000
N_high, N_low, N_all = len(ISI_high), len(ISI_low), len(ISI_pooled)

# 3. actual difference
mean_lambda_diff = lambda_low - lambda_high
mean_mu_diff = mu_low - mu_high

# 4. Bootstrapping
## of mu
mu_diff = [ISI_pooled[np.random.randint(N_all, size = N_low)].mean() - I

## of lambda
lambda_diff = [get_shape_parm(ISI_pooled[np.random.randint(N_all, size =
```

In [39]:
```
# 5. plot the histogram
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (12, 4))

plt.suptitle('Histograms of Bootstrapping Analysis')


ax1.set_title(r'Mean Value $\mu$')
ax1.hist(mu_diff, bins = 40)
ax1.axvline(x = mean_mu_diff, ls = '--', color = 'red', label  = r'$\Del

ax1.set_xlabel(r'$\Delta\hat{\mu} = \hat{\mu}_{\mathrm{low}} - \hat{\mu}
ax1.set_ylabel('count (-)')


ax2.set_title(r'Shape Paramter $\lambda$')
ax2.hist(lambda_diff, bins = 40)
ax2.axvline(x = mean_lambda_diff, ls = '--', color = 'red', label  = r'$

ax2.set_xlabel(r'$\Delta\hat{\lambda} = \hat{\lambda}_{\mathrm{low}} - \
ax2.set_ylabel('count (-)')

for ax in (ax1, ax2):
    ax.legend()

plt.show()
```



A5 conclusion

At first glance, the fits on the histograms appear 'oke'. However, upon closer examination, there is an overestimation of the low ISI values for the low spontaneous rate (LSR) fibers condition.

After generating the cumulative distribution functions (CDFs) and plotting the Kolmogorov-Smirnov (KS) curves, it becomes evident that the fit is not adequate for both cases, although it does come close for the high spontaneous rate (HSR) fiber distribution.

Upon investigating the model parameters, it is observed that the mean ISI ($\hat{\mu}$) for the HSR fibers is lower compared to the LSR fibers. This aligns with expectations, as higher spiking rates are anticipated for HSR fibers, resulting in shorter mean ISIs.

Further examination of the shape parameter reveals that the ISI distribution is narrower for HSR fibers compared to LSR fibers. Consequently, determining a *single* average spike rate for LSR fibers with high reliability becomes challenging; the LSR fibers histogram show low count bins (max = 4) and a broad distribution up to ISI ≈ 3.3.

A6: Inhomogenous integrate-and-fire encoder model

- Go back to Q6

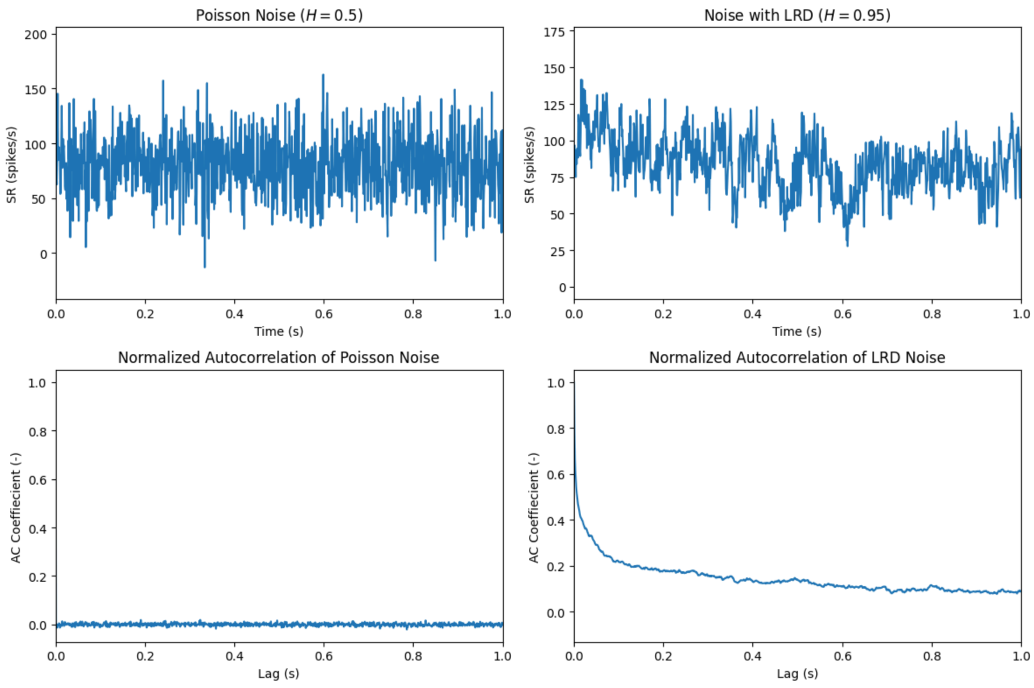
In [50]:
```python
# use the following parameters for Q6.1:

# rate_white (your rate over time signal with H=0.5 with an average of
# rate_LRD (your rate over time signal with H=0.95 with an average of 8
# ACF_white (autocorrelation of rate_white for the first 1000 data poir
# ACF_LRD (autocorrelation of rate_LRD for the first 1000 data points)

# plot the first 1000 data points, not all 30000
# similar for the autocorrelation, plot both ACF's in one figure

# use the following parameters for Q6.2:
# spiketimes_white (the first output parameter of the inhomPP function,
# spiketimes_LRD (the first output parameter of the inhomPP function, H

def autocorr(x):
    mean, var = x.mean(), x.var()
    ndata = x - mean
    x_corr = np.correlate(ndata, ndata, 'full')
    x_corr = x_corr[x_corr.size//2:]
    x_corr = x_corr / var / len(ndata)

    return x_corr
```

In [51]:
```python
# Q6.1
# Parameters
T = 30                  # s
FS = 1000               # Hz
SR = 80                 # spikes/s
H_Poisson = 0.5         # Hurst exponent for Poisson process
H_LRD = 0.95            # Hurst exponent for process with LRD

# Generate escape noise
## Poisson processs
noise_Poisson = ffGn(T * FS, 1 / FS, H_Poisson, SR) + SR

## process with LRD
noise_LRD = ffGn(T * FS, 1 / FS, H_LRD, SR) + SR
```

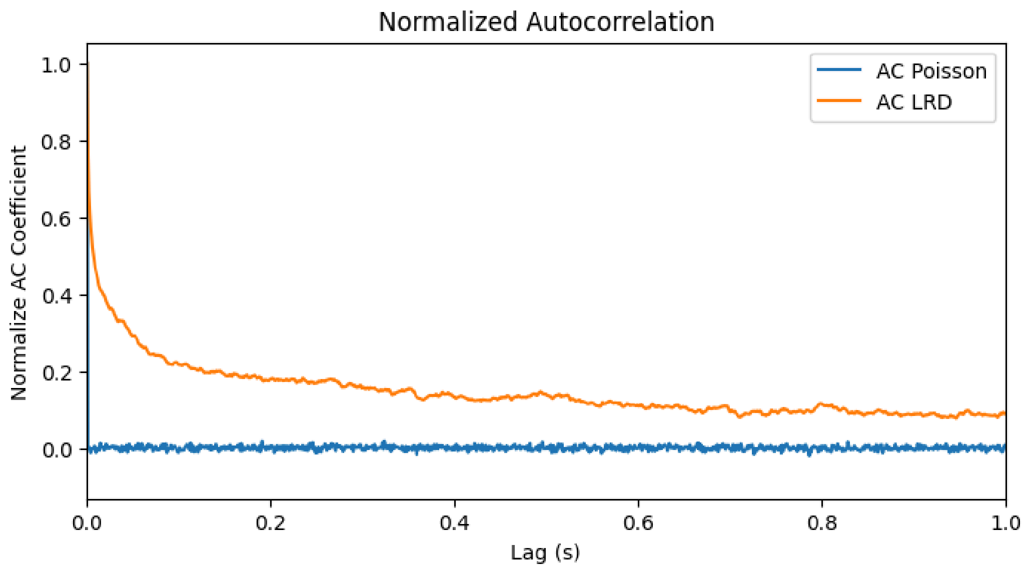
```
# time vector
t = np.arange(0, T, 1 / FS)

# get the autocorrelations
autocorr_Poisson = autocorr(noise_Poisson)
autocorr_LRD = autocorr(noise_LRD)
```

In [52]:

```
# Plot time series of both noise signals
t0, t1 = 0.0, 1.0

fig, (ax1, ax2) = plt.subplots(2, 2, figsize = (12, 4*2))

# Plot the noise signals
ax1[0].set_title('Poisson Noise ($H=0.5$)')
ax1[0].plot(t, noise_Poisson)

ax1[1].set_title('Noise with LRD ($H=0.95$)')
ax1[1].plot(t, noise_LRD)

for ax in (ax1[0], ax1[1]):
    ax.set_ylabel('SR (spikes/s)')
    ax.set_xlabel('Time (s)')
    ax.set_xlim((t0, t1))

# Plot the autocorrelation coeffecients

ax2[0].set_title('Normalized Autocorrelation of Poisson Noise')
ax2[0].plot(t, autocorr_Poisson)

ax2[1].set_title('Normalized Autocorrelation of LRD Noise')
ax2[1].plot(t, autocorr_LRD)

for ax in (ax2[0], ax2[1]):
    ax.set_ylabel('AC Coeffiecient (-)')
    ax.set_xlabel('Lag (s)')
    ax.set_xlim((t0, t1))
    #ax.set_ylim()

plt.tight_layout()
plt.show()
```



In [53]:
```
# autocorrelations in one figure
plt.figure(figsize = (8, 4))

plt.title('Normalized Autocorrelation')
plt.plot(t, autocorr_Poisson, label = 'AC Poisson')
plt.plot(t, autocorr_LRD, label = 'AC LRD')

plt.xlabel('Lag (s)')
plt.ylabel('Normalize AC Coefficient')

plt.xlim((t0, t1))
plt.legend()

plt.show()
```

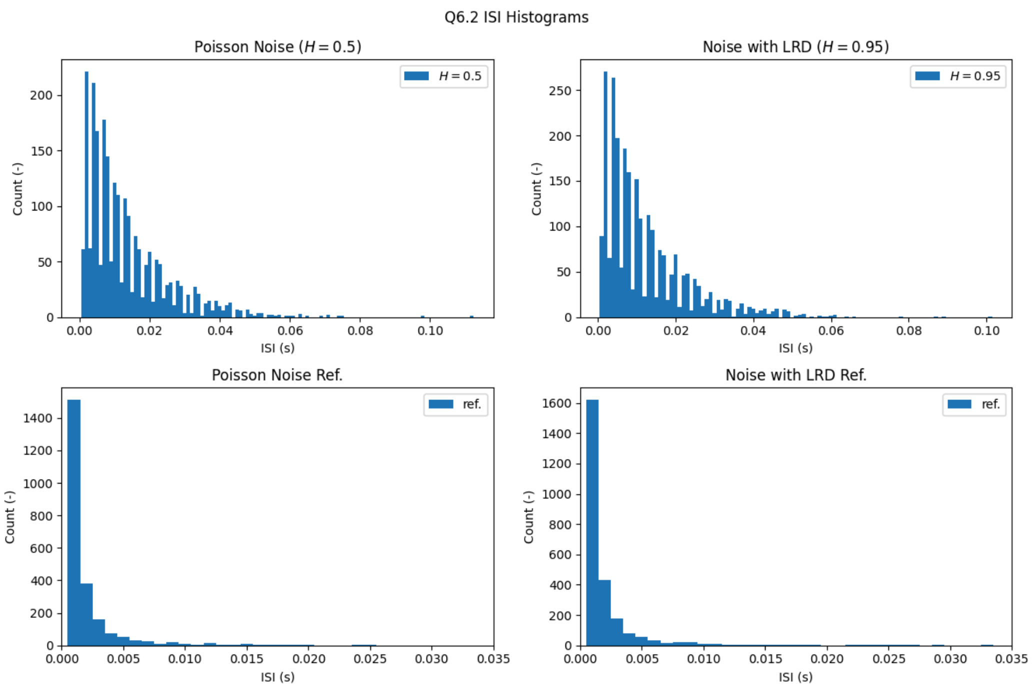


In [54]:
```
# Q6.2
# use the following parameters for Q6.2:
```

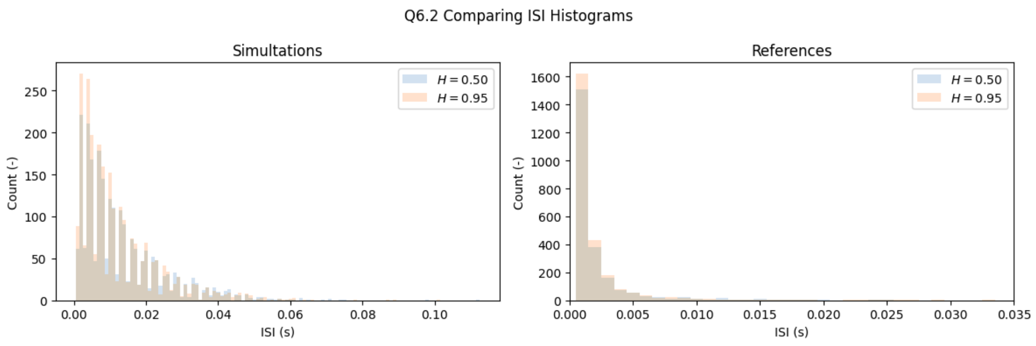
```
# spiketimes_white (the first output parameter of the inhomPP function,
# spiketimes_LRD (the first output parameter of the inhomPP function, l

def get_histogram(data, dt, perc = 100):
    data = np.asarray(data)
    max_bin = np.percentile(data, perc)
    bin_edges = np.arange(-0.0, max_bin + dt, dt)
    bin_mids = bin_edges[:-1] + np.diff(bin_edges)

    hist, _ = np.histogram(data, bins = bin_edges)

    return bin_mids, hist
```

In [55]:
```
dt = 1e-3 # s

# encoding
spiketimes_white, PSref_white = inhomPP(noise_Poisson, dt)
PSref_white = PSref_white

spiketimes_LRD, PSref_LRD = inhomPP(noise_LRD, dt)
PSref_LRD = PSref_LRD

# get the ISIs
ISI_white, ISI_white_ref = np.diff(spiketimes_white), np.diff(sorted(PS
ISI_LRD, ISI_LRD_ref = np.diff(spiketimes_LRD), np.diff(sorted(PSref_LI

# get histograms
(bin_mids_white, hist_white), (bin_mids_white_ref, hist_white_ref) = ge
(bin_mids_LRD, hist_LRD), (bin_mids_LRD_ref, hist_LRD_ref) = get_histog
```

In [56]:
```
# visualization of the plots
fig, (ax1, ax2) = plt.subplots(2, 2, figsize = (12, 4*2))

plt.suptitle('Q6.2 ISI Histograms')

ax1[0].set_title('Poisson Noise ($H = 0.5$)')
ax1[0].bar(bin_mids_white, hist_white, width = dt, label = '$H = 0.5$'
#ax1[0].bar(bin_mids_LRD, hist_LRD, alpha = 0.3, width = dt, label = '$
#ax1[0].set_ylim((0, 300))

ax2[0].set_title('Poisson Noise Ref.')
ax2[0].bar(bin_mids_white_ref, hist_white_ref, width = dt, label = 're

ax1[1].set_title('Noise with LRD ($H = 0.95$)')
ax1[1].bar(bin_mids_LRD, hist_LRD, width = dt, label = '$H = 0.95$')

ax2[1].set_title('Noise with LRD Ref.')
ax2[1].bar(bin_mids_LRD_ref, hist_LRD_ref, width = dt, label = 'ref.')

for ax in (ax2):
    ax.set_xlim((0, 0.035))

for axs in (ax1, ax2):
    for ax in axs:
        ax.set_ylabel('Count (-)')
        ax.set_xlabel('ISI (s)')
        ax.legend()

plt.tight_layout()
plt.show()
```



In [57]:
```
# visualization of the plots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (12, 4))

plt.suptitle('Q6.2 Comparing ISI Histograms')

ax1.set_title('Simultations')
ax1.bar(bin_mids_white, hist_white, width = dt, alpha = 0.2, label = '$
ax1.bar(bin_mids_LRD, hist_LRD, alpha = 0.2, width = dt, label = '$H=0


ax2.set_title('References')
ax2.bar(bin_mids_white_ref, hist_white_ref, alpha = 0.2, width = dt, la
ax2.bar(bin_mids_LRD_ref, hist_LRD_ref, alpha = 0.2, width = dt, label
ax2.set_xlim((0, 0.035))

for ax in (ax1, ax2):
    ax.set_ylabel('Count (-)')
    ax.set_xlabel('ISI (s)')
    ax.legend()

plt.tight_layout()
plt.show()
```



A6.1 conclusion

It's evident from the plotted signals that the noise signal with $H = 0.5$ (white noise) exhibits complete randomness, lacking any discernible pattern. Conversely, in the noise signal with $H = 0.95$ (LDR), some oscillatory behavior is noticeable, indicating a more structured (correlated) pattern.

The autocorrelation analysis reinforces these observations. For the white noise signal, the autocorrelation remains consistently at zero across all lag values. This aligns with expectations, as in a Poisson process, each sample (spike) is independent. However, for the noise signal with $H = 0.95$, the autocorrelation shows a slow decay with increasing temporal lag (as stated in Jackson et al.). This signifies a dependency between spikes, characteristic of LRD noise signals.

A6.2 conclusion

Upon initial observation, both ISI histograms derived from noise signals with Hurst exponents of $H = 0.5$ and $H = 0.95$ exhibit a similar pattern: alternating high and low count bins that align periodically. This synchronicity likely arises from the initial spikes aligning and the shared characteristics among neurons, leading to synchronized refractory periods.

However, upon closer examination, a noteworthy difference emerges: the histogram corresponding to $H = 0.95$ displays slightly larger counts for larger ISI values compared to the $H = 0.5$ histogram. This discrepancy, albeit subtle, aligns with the expectations outlined in the question. Higher Hurst exponents signify increased variability in the ISI histogram, indicating broader distributions beyond simple Poisson variability (i.e. there is another factor causing this variability). Notably, in **most** of the simulations, the ISI histogram for $H = 0.95$ also features bins with the highest counts, particularly at low ISI values.

Furthermore, it's worth highlighting that the control value `PSref` primarily exhibits high counts for small ISI values in both cases. This observation underscores the significant influence of input noise properties, as represented by the Hurst exponent, on the spike generation dynamics captured by the inhomogeneous integrate-and-fire neuron model.

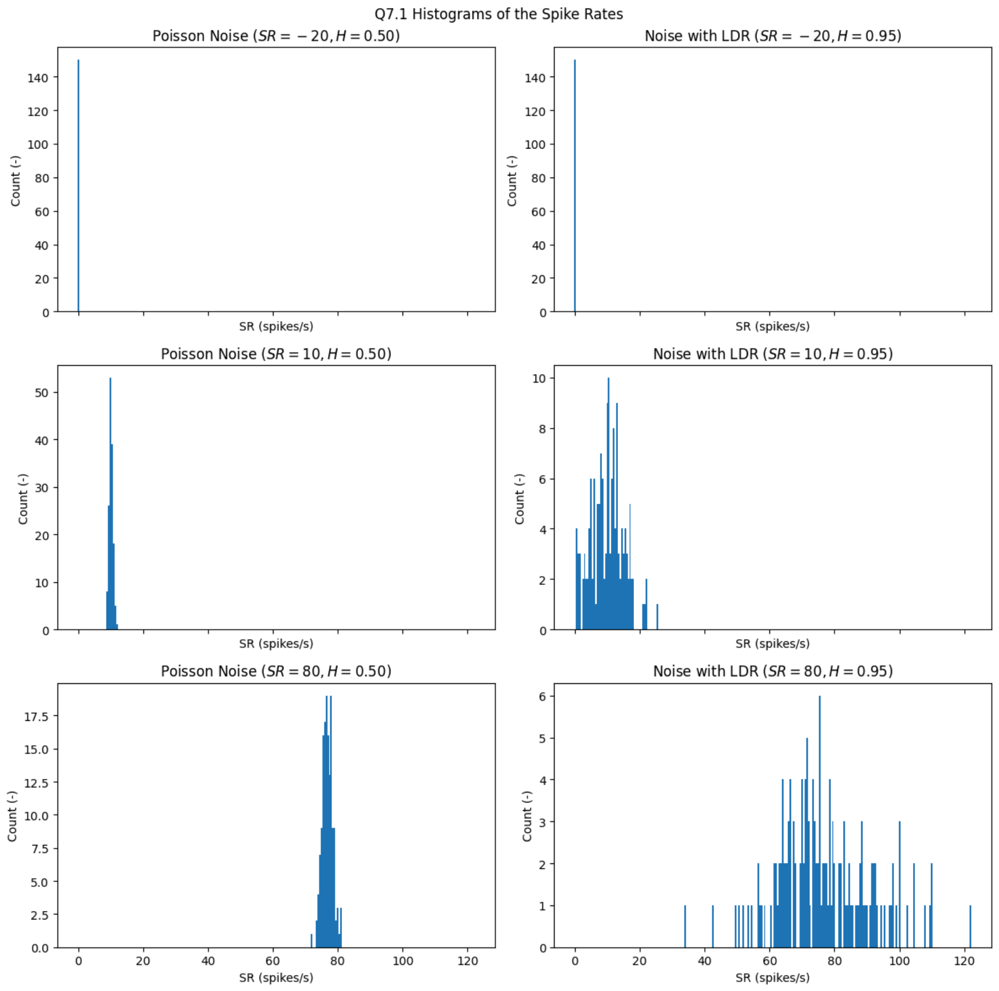
A7: Investigate the research hypothesis

- Go back to Q7

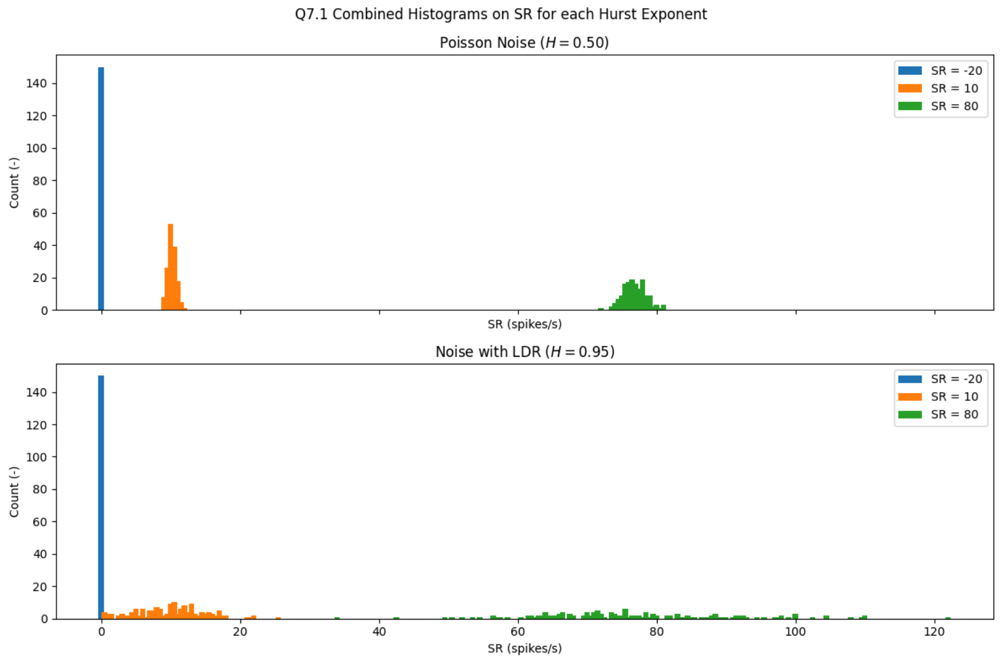
In [58]:
```
# Q7.1
def get_mean_SR(SR, H, n = 150, T = 30, FS = 1000):

    mean_SR = []

    for _ in range(n):

        # encoding
        rate = ffGn(T * FS, 1 / FS, H, SR) + SR
        spiketimes , PSref = inhomPP(rate, 1 / FS)

        # add to the list
        mean_SR.append(len(spiketimes) / T)

    return mean_SR
```

In [59]:
```
# simulations
# Hurst exponents
H_Poisson, H_LDR = 0.50, 0.95

# SR values
SRs = [-20, 10, 80]

# Encoding results
mean_SRs_LDR = {SR: get_mean_SR(SR, H_LDR) for SR in SRs}
mean_SRs_white = {SR: get_mean_SR(SR, H_Poisson) for SR in SRs}
```

In [60]:
```
# Make the figures here
dSR = 0.5

fig, ax = plt.subplots(3, 2, figsize = (12, 4*3), sharex = True)

plt.suptitle('Q7.1 Histograms of the Spike Rates')

for i, ((SR_w, mean_w), (SR_LDR, mean_LDR)) in enumerate(zip(mean_SF

    if SR_w != -20:
        bins_w, hist_w = get_histogram(mean_w, dt = dSR)
        bins_LDR, hist_LDR = get_histogram(mean_LDR, dt = dSR)
    else:
        bins_w, hist_w = [0], [len(mean_w)]
        bins_LDR, hist_LDR = [0], [len(mean_LDR)]

    ax[i, 0].set_title(f'Poisson Noise ($SR = {SR_w:.0f}, H = {H_Poi
    ax[i, 0].bar(bins_w, hist_w, width = dSR)

    ax[i, 1].set_title(f'Noise with LDR ($SR = {SR_LDR:.0f}, H = {H_
    ax[i, 1].bar(bins_LDR, hist_LDR, width = dSR)

    for axi in (ax[i, 0], ax[i, 1]):
        axi.set_xlabel('SR (spikes/s)')
        axi.set_ylabel('Count (-)')

plt.tight_layout()
plt.show()
```



In [61]:
```
# everything together
fig, (ax1, ax2) = plt.subplots(2, 1, figsize = (12, 4*2), sharex = T

plt.suptitle('Q7.1 Combined Histograms on SR for each Hurst Exponent

ax1.set_title(f'Poisson Noise ($H = {H_Poisson:.2f}$)')
ax2.set_title(f'Noise with LDR ($H = {H_LDR:.2f}$)')

for i, ((SR_w, mean_w), (SR_LDR, mean_LDR)) in enumerate(zip(mean_SR
    if SR_w != -20:
        bins_w, hist_w = get_histogram(mean_w, dt = dSR)
        bins_LDR, hist_LDR = get_histogram(mean_LDR, dt = dSR)

    else:
        bins_w, hist_w = [0], [len(mean_w)]
        bins_LDR, hist_LDR = [0], [len(mean_LDR)]

    ax1.bar(bins_w, hist_w, label = f'SR = {SR_w:.0f}')

    ax2.bar(bins_LDR, hist_LDR, label = f'SR = {SR_LDR:.0f}')

for axi in (ax1, ax2):
    axi.set_xlabel('SR (spikes/s)')
    axi.set_ylabel('Count (-)')
    axi.legend()

plt.tight_layout()
plt.show()
```



A7 conclusion

The generated histograms closely resemble the findings reported in the paper. In the simulations with white noise ($H = 0.50$), the spontaneous rate distribution appears narrow due to the absence of temporal correlation. Conversely, the histograms obtained from simulations incorporating LDR exhibit a broader distribution.

The presence of a negative SR may not reflect physiological reality, as it suggests a spiking system where the underlying process driving the action potential generator remains below a threshold.

By employing an inhomogeneous Poisson process with three different SR values and incorporating LRD, our results align with those reported in the paper; the obtained histogram is similar to the one in the paper (and provided in the notebook description). This reaffirms the sensibility of their hypothesis, indicating that the behavior of AN fibers can be effectively captured using this approach. The inclusion of LRD proves crucial, as it obviuously accounts for memory and long-range dependency in the spike generation process.

```
In [ ]:
```