CodingLab4

May 18, 2025

Neural Data Science

Lecturer: Dr. Jan Lause, Prof. Dr. Philipp Berens

Tutors: Jonas Beck, Fabio Seel, Julius Würzler

Summer term 2025

Student names: Aakarsh Nair, Andreas Kotzur, Ahmed Eldably

LLM Disclaimer: *Did you use an LLM to solve this exercise? If yes, which one and where did you use it? Chat GPT 4o, Google Gemini - Task Tracking, Template Code, Background Knowledge, Plotting

1 Coding Lab 4

In this notebook you will work with preprocessed 2 photon calcium recordings, that have already been converted into spike counts for a population of cells from the Macaque V1. During the experiment the animal has been presented with several drifting grating stimuli, in response to which the neural activity was recorded. In this exercise sheet we will study how you can visualize the activity of multiple neural spike trains and assess whether a neuron is selective to a specific stimulus type.

Download the data files `nds_cl_4_*.csv` from ILIAS and save it in the subfolder `../data/`. We recommend you to use a subset of the data for testing and debugging, ideally focus on a single cell (e.g. cell number x). The spike times and stimulus conditions are read in as pandas data frames. You can solve the exercise by making heavy use of that, allowing for many quite compact computations. See documentation and several good tutorials on how to do this. Of course, converting the data into classical numpy arrays is also valid.

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
import scipy.optimize as opt

from scipy import signal as signal
from typing import Tuple

import itertools
```

```
import logging

%matplotlib inline

%load_ext jupyter_black

%load_ext watermark
%watermark --time --date --timezone --updated --python --iversions --watermark␣
 ↪-p sklearn
```

```
Last updated: 2025-05-18 18:35:48CEST

Python implementation: CPython
Python version       : 3.11.11
IPython version      : 9.2.0

sklearn: not installed

pandas    : 2.2.3
logging   : 0.5.1.2
scipy     : 1.15.2
seaborn   : 0.13.2
matplotlib: 3.9.4
numpy     : 1.26.4

Watermark: 2.5.0
```

[2]:
```
# %%
logging.basicConfig(
    format="%(asctime)s - %(levelname)s - %(message)s",
    level=logging.DEBUG,
)
logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)
logger.info("Starting the script...")


# --- Add these lines to suppress Matplotlib's DEBUG messages ---
mpl_logger = logging.getLogger("matplotlib")
mpl_logger.setLevel(logging.WARNING)  # Or logging.INFO, logging.ERROR
# You can also target specific noisy submodules if needed:
# logging.getLogger('matplotlib.font_manager').setLevel(logging.WARNING)
# logging.getLogger('matplotlib.pyplot').setLevel(logging.WARNING)
# --- End of added lines ---
```

```
INFO:__main__:Starting the script…
```

[3]:
```
plt.style.use("../matplotlib_style.txt")
```

1.1 Load data

[4]:
```
spikes = pd.read_csv("../data/nds_cl_4_spiketimes.csv")  # neuron id, spike time
stims = pd.read_csv("../data/nds_cl_4_stimulus.csv")  # stimulus onset in ms,␣
 ↪direction

stimDur = 2000.0  # in ms
nTrials = 11  # number of trials per condition
nDirs = 16  # number of conditions
deltaDir = 22.5  # difference between conditions

stims["StimOffset"] = stims["StimOnset"] + stimDur
```

We require some more information about the spikes for the plots and analyses we intend to make later. With a solution based on dataframes, it is natural to compute this information here and add it as additional columns to the `spikes` dataframe by combining it with the `stims` dataframe. We later need to know which condition (`Dir`) and trial (`Trial`) a spike was recorded in, the relative spike times compared to stimulus onset of the stimulus it was recorded in (`relTime`) and whether a spike was during the stimulation period (`stimPeriod`). But there are many options how to solve this exercise and you are free to choose any of them.

[5]:
```
# you may add computations as specified above
spikes["Dir"] = np.nan
spikes["relTime"] = np.nan
spikes["Trial"] = np.nan
spikes["stimPeriod"] = np.nan

dirs = np.unique(stims["Dir"])
trialcounter = np.zeros_like(dirs)

for i, row in stims.iterrows():
    trialcounter[dirs == row["Dir"]] += 1

    i0 = spikes["SpikeTimes"] > row["StimOnset"]
    i1 = spikes["SpikeTimes"] < row["StimOffset"]

    select = i0.values & i1.values

    spikes.loc[select, "Dir"] = row["Dir"]
    spikes.loc[select, "Trial"] = trialcounter[dirs == row["Dir"]][0]
    spikes.loc[select, "relTime"] = spikes.loc[select, "SpikeTimes"] -␣
 ↪row["StimOnset"]
    spikes.loc[select, "stimPeriod"] = True

spikes = spikes.dropna()
```

```
/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/4012369867.py:2
1: FutureWarning: Setting an item of incompatible dtype is deprecated and will
raise an error in a future version of pandas. Value 'True' has dtype
incompatible with float64, please explicitly cast to a compatible dtype first.
  spikes.loc[select, "stimPeriod"] = True
```

[6]:
```
spikes.head()
```

[6]:

	Neuron	SpikeTimes	Dir	relTime	Trial	stimPeriod
514	1	15739.000000	270.0	169.000000	1.0	True
515	1	15776.566667	270.0	206.566667	1.0	True
516	1	15808.466667	270.0	238.466667	1.0	True
517	1	15821.900000	270.0	251.900000	1.0	True
518	1	15842.966667	270.0	272.966667	1.0	True

[7]:
```
stims.head()
```

[7]:

	StimOnset	Dir	StimOffset
0	15570	270.0	17570.0
1	19022	45.0	21022.0
2	22592	112.5	24592.0
3	26095	225.0	28095.0
4	29431	180.0	31431.0

1.2 Task 1: Plot spike rasters

In a raster plot, each spike is shown by a small tick at the time it occurs relative to stimulus onset. Implement a function `plotRaster()` that plots the spikes of one cell as one trial per row, sorted by conditions (similar to what you saw in the lecture). Why are there no spikes in some conditions and many in others?

If you opt for a solution without a dataframe, you need to change the interface of the function.

Grading: 3 pts

[8]:
```
def plotRaster(spikes: pd.DataFrame, neuron: int):
    """plot spike rasters for a single neuron sorted by condition

    Parameters
    ----------

    spikes: pd.DataFrame
        Pandas DataFrame with columns
            Neuron | SpikeTimes | Dir | relTime | Trial | stimPeriod

    neuron: int
        Neuron ID
```

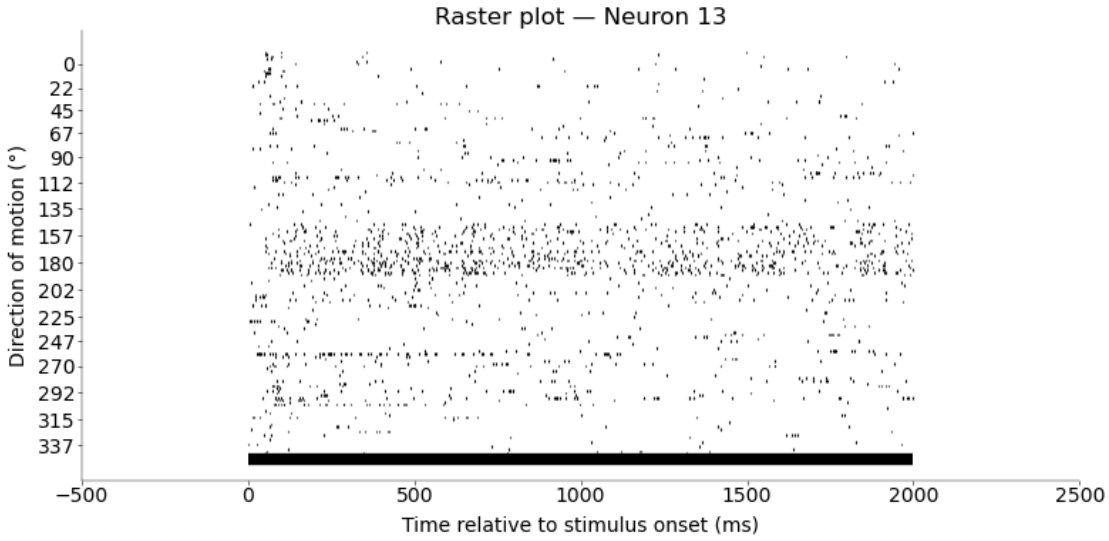
```
    Note
    ----

    this function does not return anything, it just creates a plot!
    """

    # ------------------------------------------------
    # Write a raster plot function for the data (2 pts)
    # ------------------------------------------------
    df = spikes[(spikes["Neuron"] == neuron) & (spikes["stimPeriod"])]
    directions = np.sort(df["Dir"].unique())

    fig, ax = plt.subplots(figsize=(8, 4))
    y = 0
    yticks, ylabels = [], []

    for d in directions:
        sub = df[df["Dir"] == d]
        trials = np.sort(sub["Trial"].unique())
        # remember the center for this direction block
        yticks.append(y + len(trials) / 2)
        ylabels.append(int(d))
        for t in trials:
            times = sub[sub["Trial"] == t]["relTime"]
            ax.vlines(times, y, y + 1, linewidth=0.8, color="k")
            y += 1
        # small gap between direction blocks
        y += 0.5

    # stimulus-on bar
    ax.hlines(y + 0.5, xmin=0, xmax=stimDur, linewidth=6, color="k")

    ax.set_yticks(yticks)
    ax.set_yticklabels(ylabels)
    ax.set_xlabel("Time relative to stimulus onset (ms)")
    ax.set_ylabel("Direction of motion (°)")
    ax.set_title(f"Raster plot - Neuron {neuron}")

    # show some pre- and post-stimulus window
    ax.set_xlim(-500, stimDur + 500)

    # if you want 0° at bottom, 360° at top:
    ax.invert_yaxis()

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

[ ]:
```
# ---------------------------------
# Find and explain examples? (1 pt)
# ---------------------------------
# average over the full 0-stimDur window, per trial
rates = (
    spikes.query("stimPeriod")
    .groupby(["Neuron", "Dir", "Trial"])
    .size()  # spike count
    .reset_index(name="count")
    .assign(rate=lambda df: df["count"] / (stimDur / 1000))
    .groupby(["Neuron", "Dir"])["rate"]
    .mean()  # mean across trials
    .unstack()  # rows=Neuron, cols=Dir
)
dsis = {}
osis = {}
for nm, row in rates.iterrows():
    pref = row.max()
    pref_dir = row.idxmax()
    null_dir = (pref_dir + 180) % 360
    perp_dirs = [(pref_dir + 90) % 360, (pref_dir - 90) % 360]

    R_null = row[null_dir]
    R_perp = row[perp_dirs].mean()

    dsis[nm] = (pref - R_null) / (pref + R_null + 1e-6)
    osis[nm] = (pref - R_perp) / (pref + R_perp + 1e-6)

# then:
# Using hueristic thresholds, permutation tests used later provide more robust␣
 ↪and accurate results
# to determine the selectivity of the neuron
dir_sel = [n for n in dsis if dsis[n] > 0.5]  # [9, 13, 27]
ori_sel = [n for n in osis if osis[n] > 0.5 and dsis[n] < 0.2]  # [6, 7, 12]
non_sel = [n for n in dsis if abs(dsis[n]) < 0.1 and abs(osis[n]) < 0.1]  # [1,␣
 ↪4]

print("Direction-selective examples:", dir_sel)
print("Orientation-selective examples:", ori_sel)
print("Non-selective examples:", non_sel)

for neuron in [dir_sel[1], ori_sel[3], non_sel[0]]:
    print(f"\nNeuron {neuron}:\n")
    plotRaster(spikes, neuron)
```

```
Direction-selective examples: [9, 13, 27, 29, 32, 37, 38]
Orientation-selective examples: [6, 7, 12, 16, 17, 21]
Non-selective examples: [1, 4]
```

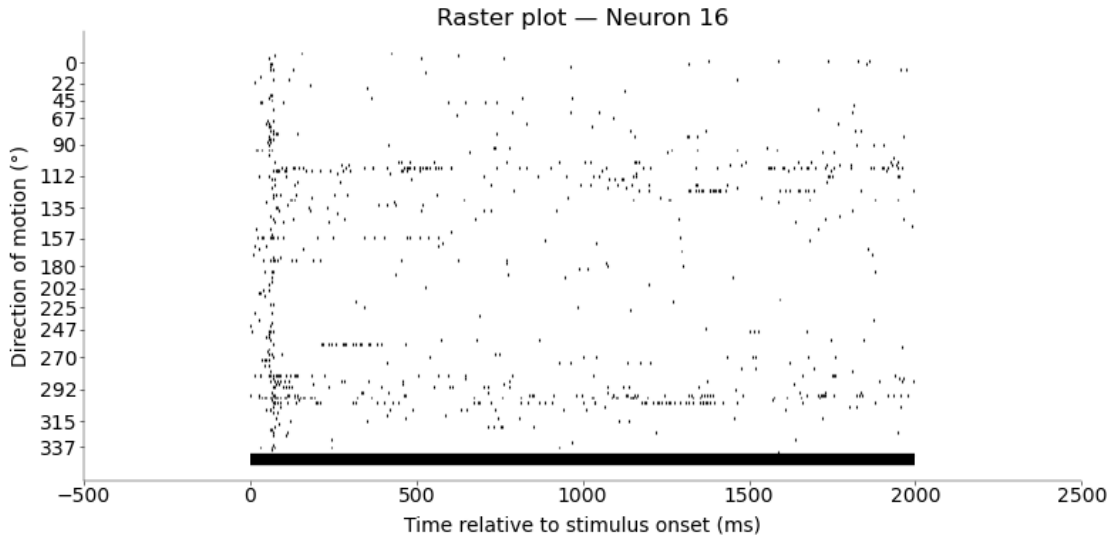
Neuron 13:

```
/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/224280136.py:59
: UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```



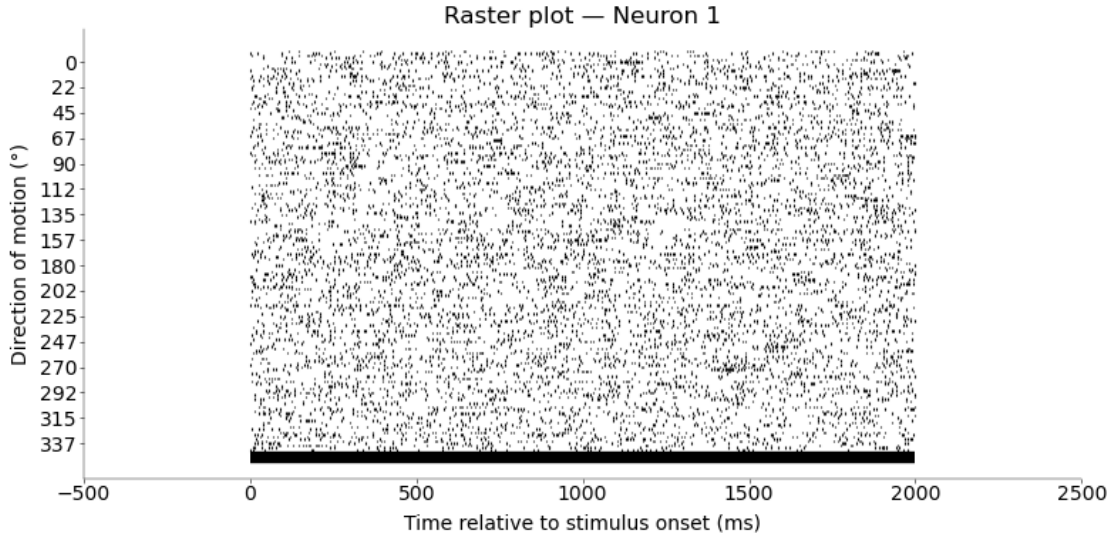
Neuron 16:

```
/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/224280136.py:59
: UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```



Neuron 1:

```
/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/224280136.py:59
: UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```



Why are there no spikes in some conditions and many in others? Answer: The primary reason why an individual neuron shows many spikes in some conditions and few or no spikes in

others is because neurons themselves are intrinsically selective for specific stimulus features, such as particular orientations and directions of motion.

Each neuron in a sensory area like V1 is often 'tuned' to respond most strongly to its preferred stimulus. When a presented stimulus aligns with a neuron's preference, it fires vigorously (many spikes). When the stimulus does not match its preference, it fires weakly or not at all (few or no spikes).

In the experiment we are sub-sampling a population, and the specific neurons we happen to record will determine the range of preferences we observe in our dataset.

Find examples and and explain your choices:

1. A direction selective neuron:

Answer: We chose Neuron 13 as a possible example of a direction selective neuron. Observing the firing rate we find that there is a band between 140-180 degrees where the neuron is firing at a higher rate without a corresponding second direction to which it is selective.

2. An orientation selective neuron

Answer: We chose Neuron 16 as a possible example of orientation selective neuron. Observing that it fires for two directions roughly 180 degrees apart.

3. neither

Answer: We chose Neuron 1 as a possible example of a neuron which is not tuned to direction or orientation. It fires equally for all angles following stimulus.

1.3 Task 2: Plot spike density functions

Compute an estimate of the spike rate against time relative to stimulus onset. There are two ways: * Discretize time: Decide on a bin size, count the spikes in each bin and average across trials. * Directly estimate the probability of spiking using a density estimator with specified kernel width.

For full points, the optimal kernel- or bin-width needs to be computed.

Implement one of them in the function `plotPSTH()`. If you dont use a dataframe you may need to change the interface of the function.

Grading: 4 pts

[10]:
```
from scipy.stats import norm

nTrials = 11  # number of trials per direction


def select_optimal_binwidth(
    rel_times, T, n_trials, delta_min=1.0, delta_max=None, n_deltas=200
):
    """
    Shimazaki & Shinomoto (2007) algorithm to pick histogram bin width Δ,
    with log-spaced search capped at T/10 and a Freedman-Diaconis fallback.
```

```
    Parameters
    ----------
    rel_times : array-like (ms)
        all spike times within [0, T]
    T : float (ms)
        total observation window
    n_trials : int
        number of repeated trials
    delta_min : float (ms)
        smallest bin width to try
    delta_max : float (ms), optional
        largest bin width to try; defaults to T/10
    n_deltas : int
        number of candidate widths in the log-spaced grid

    Returns
    -------
    Δ_opt_ms : float
        optimal bin width (ms)
    """
    # convert to seconds
    rel_s = np.asarray(rel_times) / 1000.0
    T_s = T / 1000.0

    # cap delta_max at T/10 if not provided
    if delta_max is None:
        delta_max = T / 20.0
    else:
        delta_max = min(delta_max, T / 20.0)

    # endpoints in seconds
    del_min_s = delta_min / 1000.0
    del_max_s = delta_max / 1000.0

    # log-spaced candidate Δ_s
    deltas_s = np.logspace(np.log10(del_min_s), np.log10(del_max_s), n_deltas)

    # compute Shimazaki-Shinomoto cost for each Δ_s
    costs = np.zeros_like(deltas_s)
    for i, Δ_s in enumerate(deltas_s):
        edges = np.arange(0, T_s + Δ_s, Δ_s)
        kis, _ = np.histogram(rel_s, bins=edges)
        kis = kis.astype(float) / n_trials
        k_bar = kis.mean()
        v = ((kis - k_bar) ** 2).mean()
        costs[i] = (2 * k_bar - v) / (Δ_s**2)
```

```
    # pick the Δ_s that minimizes the cost
    idx = np.argmin(costs)
    Δ_opt_ms = deltas_s[idx] * 1000.0

    # if the minimum is at the boundary, fallback to Freedman-Diaconis
    if idx == 0 or idx == len(deltas_s) - 1:
        x = np.asarray(rel_times)
        iqr = np.subtract(*np.percentile(x, [75, 25]))
        Δ_fd = 2 * iqr * (x.size ** (-1 / 3))
        Δ_opt_ms = np.clip(Δ_fd, delta_min, delta_max)

    return Δ_opt_ms


def plotPSTH(spikes: pd.DataFrame, neuron: int, stimDur=2000.0, n_trials=11):
    """Plot PSTH for a single neuron sorted by condition

    Parameters
    ----------

    spikes: pd.DataFrame
        Pandas DataFrame with columns
            Neuron | SpikeTimes | Dir | relTime | Trial | stimPeriod

    neuron: int
        Neuron ID


    Note
    ----

    this function does not return anything, it just creates a plot!
    """

    # -------------------------------------------------
    # Implement one of the spike rate estimates (3 pts)
    # -------------------------------------------------
    # filter data
    df = spikes[(spikes.Neuron == neuron) & spikes.stimPeriod]
    dirs = np.sort(df.Dir.unique())
    n_trials = df.groupby("Dir").Trial.nunique().iloc[0]

    # compute optimal PSTH bin width Δ*
    Δ = select_optimal_binwidth(df.relTime.values, stimDur, n_trials)

    # PSTH edges & centers
```

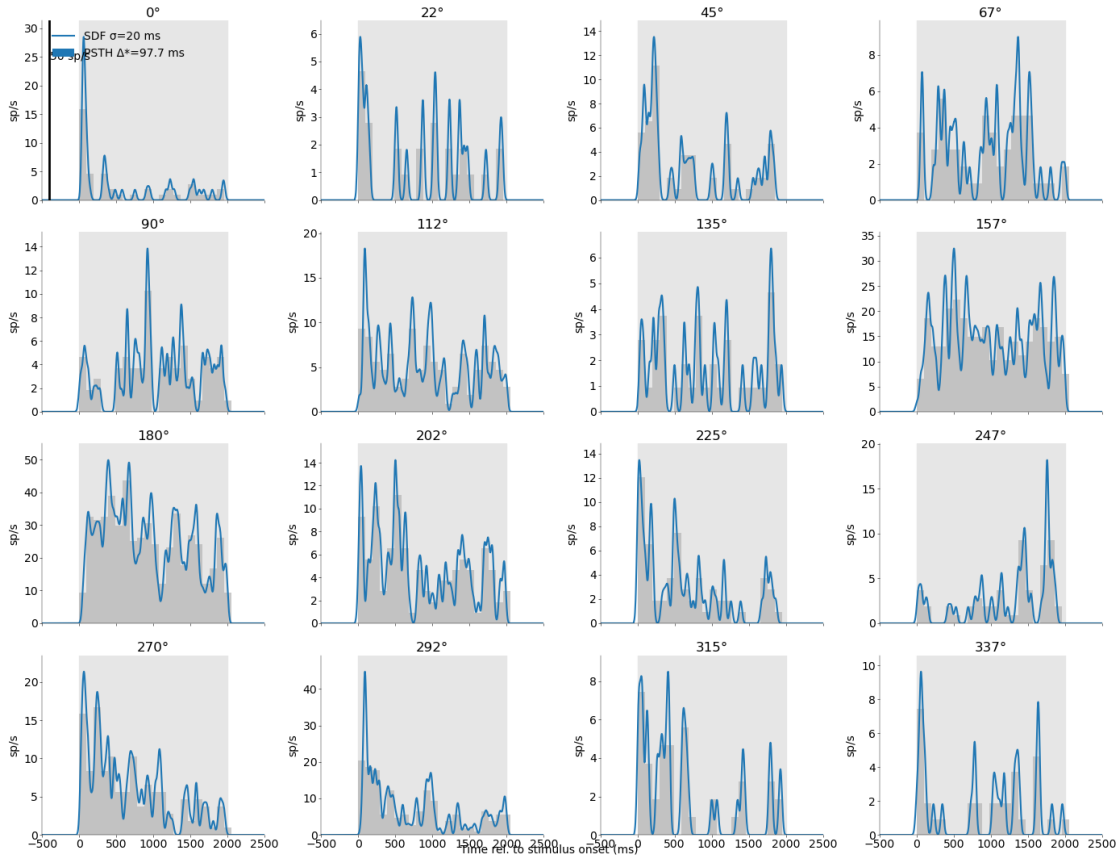
```
    psth_edges = np.arange(0, stimDur + Δ, Δ)
    psth_centers = psth_edges[:-1] + Δ / 2

    # time vector for SDF (±500 ms)
    bg, ag = 500, 500
    tvec = np.arange(-bg, stimDur + ag + 1)

    # Gaussian kernel  =20 ms
    sigma = 20.0
    kt = np.arange(-3 * sigma, 3 * sigma + 1)
    gauss = norm.pdf(kt, 0, sigma)
    gauss /= gauss.sum()

    # 3) set up 4×4 grid
    ncols = 4
    nrows = int(np.ceil(len(dirs) / ncols))
    fig, axes = plt.subplots(nrows, ncols, sharex=True, figsize=(14, 3 * nrows))
    axes = axes.flatten()

    for ax, d in zip(axes, dirs):
        sub = df[df.Dir == d]

        # PSTH → spikes/sec
        trial_counts = []
        for t in range(1, n_trials + 1):
            st = sub[sub.Trial == t].relTime.values
            cnts, _ = np.histogram(st, bins=psth_edges)
            trial_counts.append(cnts)
        rate_hist = np.mean(trial_counts, axis=0) / (Δ / 1000.0)

        # SDF → all-trial hist → sp/s → smooth
        hist1, _ = np.histogram(sub.relTime.values, bins=np.append(tvec,␣
↪tvec[-1] + 1))
        rate1 = hist1 / n_trials * 1000.0
        sdf = np.convolve(rate1, gauss, mode="same")

        # plot
        ax.axvspan(0, stimDur, color="0.9", zorder=-1)
        ax.bar(
            psth_centers, rate_hist, width=Δ, color="0.7", edgecolor="none",␣
↪alpha=0.7
        )
        ax.plot(tvec, sdf, color="C0", lw=1.5)

        # format
        ax.set_xlim(-bg, stimDur + ag)
        ax.set_ylim(0, max(rate_hist.max(), sdf.max()) * 1.1)
```

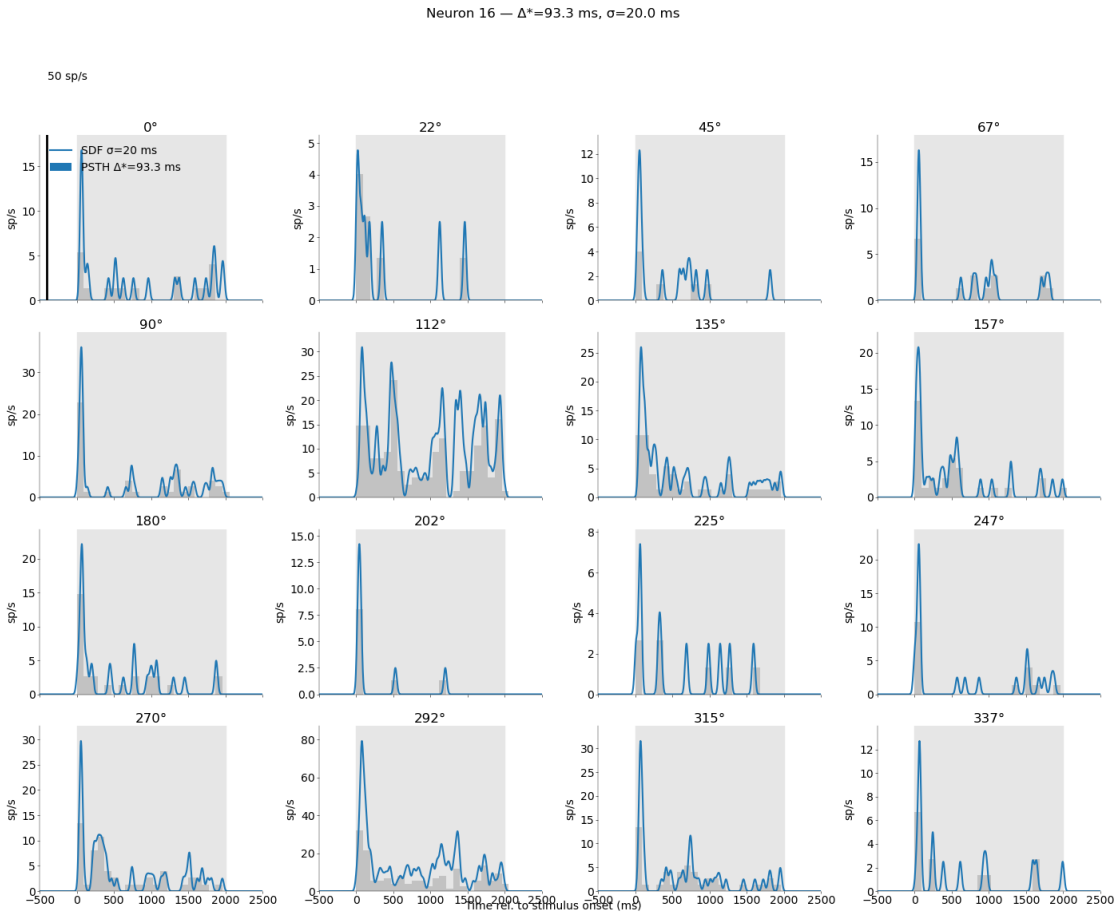
```
        ax.set_title(f"{int(d)}°")
        ax.set_ylabel("sp/s")

    # turn off any unused axes
    for ax in axes[len(dirs) :]:
        ax.axis("off")

    # shared legend & scale bar
    axes[0].bar([], [], color="0.7", alpha=0.7, label=f"PSTH Δ*={Δ:.1f} ms")
    axes[0].plot([], [], color="C0", lw=1.5, label="SDF  =20 ms")
    axes[0].legend(loc="upper left")
    axes[0].plot([-400, -400], [0, 50], "k", lw=2)
    axes[0].text(-390, 25, "50 sp/s", va="center")

    # global labels & title
    fig.text(0.5, 0.04, "Time rel. to stimulus onset (ms)", ha="center")
    fig.suptitle(f"Neuron {neuron} - Δ*={Δ:.1f} ms,  ={sigma:.1f} ms", y=0.98)

    plt.tight_layout(rect=[0, 0.03, 1, 0.95])
    plt.show()
```

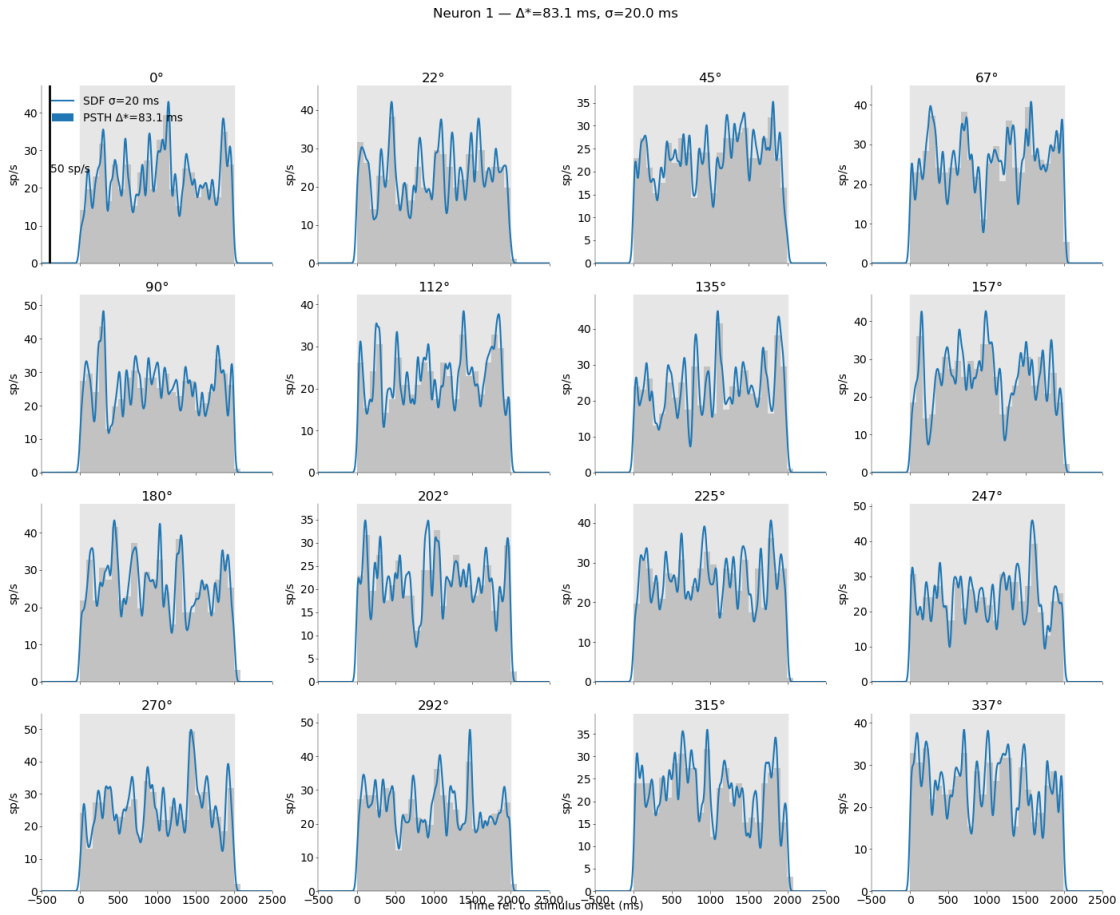
Plot the same 3 examples you selected in Task 1

[11]:
```
for neuron in [dir_sel[1], ori_sel[3], non_sel[0]]:
    plotPSTH(spikes, neuron)
```

```
/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/24118823.py:169
: UserWarning: The figure layout has changed to tight
  plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```


Neuron 13 — Δ*=97.7 ms, σ=20.0 ms
0°
22°
45°
67°
90°
112°
135°
157°
180°
202°
225°
247°
270°
292°
315°
337°
SDF σ=20 ms
PSTH Δ*=97.7 ms
sp/s
Time rel. to stimulus onset (ms)


Neuron 16 — Δ*=93.3 ms, σ=20.0 ms
50 sp/s
0°
22°
45°
67°
90°
112°
135°
157°
180°
202°
225°
247°
270°
292°
315°
337°
SDF σ=20 ms
PSTH Δ*=93.3 ms
sp/s
Time rel. to stimulus onset (ms)



1.4 Task 3: Fit and plot tuning functions

The goal is to visualize the activity of each neuron as a function of stimulus direction. First, compute the spike counts of each neuron for each direction of motion and trial. The result should be a matrix `x`, where x_{jk} represents the spike count of the j-th response to the k-th direction of motion (i.e. each column contains the spike counts for all trials with one direction of motion). If you used dataframes above, the `groupby()` function allows to implement this very compactly. Make sure you don't loose trials with zero spikes though. Again, other implementations are completely fine.

Fit the tuning curve, i.e. the average spike count per direction, using a von Mises model. To capture the non-linearity and direction selectivity of the neurons, we will fit a modified von Mises function:

$$f(\theta) = \exp(\alpha + \kappa(\cos(2 * (\theta - \phi)) - 1) + \nu(\cos(\theta - \phi) - 1))$$

Here, θ is the stimulus direction. Implement the von Mises function in `vonMises()` and plot it to understand how to interpret its parameters ϕ, κ, ν, α. Perform a non-linear least squares fit using a package/function of your choice. Implement the fitting in `tuningCurve()`.

Plot the average number of spikes per direction, the spike counts from individual trials as well as your optimal fit.

Select two cells that show nice tuning to test your code.

Grading: 5 pts

[12]:
```
def vonMises( : np.ndarray,  : float,  : float,  : float,  : float) -> np.ndarray:
    """Evaluate the parametric von Mises tuning curve
        with parameters p at locations theta.

    Parameters
    ----------

     : np.array, shape=(N, )
        Locations. The input unit is degree.

     ,  ,  ,   : float
        Function parameters

    Return
    ------
    f: np.array, shape=(N, )
        Tuning curve.
    """

    # ------------------------------------
    # Implement the Mises model (0.5 pts)
    # ------------------------------------
    return np.exp(  +   * (np.cos(2 * (  -  )) - 1) +   * (np.cos(  -  ) - 1))
```

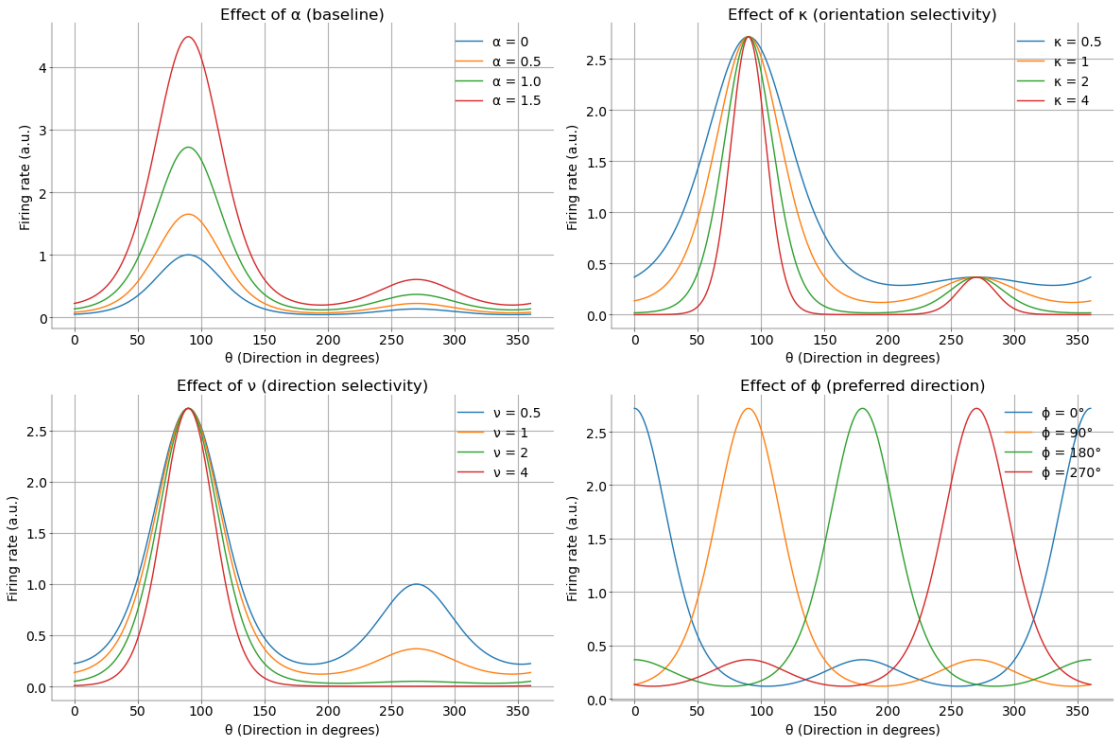
Plot the von Mises function while varying the parameters systematically.

[13]:
```
# ------------------------------------------------------------------------------
# plot von Mises curves with varying parameters and explain what they do (2 pts)
# ------------------------------------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt


def plot_vonMises_param_effects():
      = np.linspace(0, 360, 500)

    # Baseline parameters
      = 1.0
      = 1.0
      = 1.0
      = 90.0
```

```
    theta_rad = np.deg2rad( )  # convert degrees to radians
    phi_rad = np.deg2rad( )  # convert degrees to radians

    fig, axs = plt.subplots(2, 2, figsize=(12, 8))
    axs = axs.ravel()

    # Vary   (baseline)
    for a in [0, 0.5, 1.0, 1.5]:
        f = vonMises(theta_rad,  =a,  = ,  = ,  =phi_rad)
        axs[0].plot( , f, label=f"  = {a}")
    axs[0].set_title("Effect of   (baseline)")
    axs[0].legend()

    # Vary   (orientation tuning sharpness)
    for k in [0.5, 1, 2, 4]:
        f = vonMises(theta_rad,  = ,  =k,  = ,  =phi_rad)
        axs[1].plot( , f, label=f"  = {k}")
    axs[1].set_title("Effect of   (orientation selectivity)")
    axs[1].legend()

    # Vary   (direction tuning sharpness)
    for n in [0.5, 1, 2, 4]:
        f = vonMises(theta_rad,  = ,  = ,  =n,  =phi_rad)
        axs[2].plot( , f, label=f"  = {n}")
    axs[2].set_title("Effect of   (direction selectivity)")
    axs[2].legend()

    # Vary   (preferred direction)
    for phi in [0, 90, 180, 270]:
         _rad = np.deg2rad(phi)
        f = vonMises(theta_rad,  = ,  = ,  = ,  = _rad)
        axs[3].plot( , f, label=f"  = {phi}°")
    axs[3].set_title("Effect of   (preferred direction)")
    axs[3].legend()

    for ax in axs:
        ax.set_xlabel("  (Direction in degrees)")
        ax.set_ylabel("Firing rate (a.u.)")
        ax.grid(True)

    plt.tight_layout()
    plt.show()


plot_vonMises_param_effects()
```

```
/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/2235227151.py:5
7: UserWarning: The figure layout has changed to tight
```

```
plt.tight_layout()
```



Plot the von Mises function while varying the parameters systematically and explain what they do. The von Mises function used for fitting tuning curves is:

$$f(\theta) = \exp(\alpha + \kappa(\cos(2(\theta - \phi)) - 1) + \nu(\cos(\theta - \phi) - 1))$$

The parameters each distinctly describe the shape of the distribution: * α - Baseline/Amplitude - is related to the neuron's baseline firing rate or the overall responsiveness. A larger alpha generally means the neuron fires more across all tested directions, or has a higher peak response. We see in the plot that going from a range of $\alpha = 0$ to $\alpha = 1.5$ all peaks show respective increase. * κ - Orientation Selectivity / Bimodal Component Strength - this term creates a response pattern that is 180° periodic (i.e., it has two peaks in a 360° range). kappa controls the strength and sharpness of this bimodal component. * ν - Direction Selectivity - Controls the strength and sharpness of this unimodal component. We see that $\nu = 4$ has a flat bimodal peak. * ϕ - Preferred Direction/Orientation - ϕ is the neuron's preferred direction (if ν is dominant) or preferred orientation axis (if κ is dominant). It shifts the entire tuning curve horizontally along the direction axis. We note the shift in peak as we vary the ϕ above.

[14]:
```
def compute_spike_count_matrix(counts: np.ndarray, dirs: np.ndarray) -> np.
↪ndarray:
    """Compute the spike count matrix from the counts and dirs.

    Parameters
```

```
    ----------
    counts: np.ndarray
        The spike counts for each trial.

    dirs: np.ndarray
        The stimulus directions for each trial.

    Returns
    -------
    spike_count_matrix: np.ndarray
        The computed spike count matrix.
    """
    unique_stim_directions_deg = np.unique(dirs)  # Shape: (nDirs,)
    num_unique_directions = len(unique_stim_directions_deg)
    logger.debug(f"Unique stimulus directions: {unique_stim_directions_deg}")
    logger.debug(f"Number of unique stimulus directions:␣
↪{num_unique_directions}")

    # Get the unique stimulus directions, sorted. These will be the columns of␣
↪our matrix.
    unique_stim_directions_deg = np.unique(dirs)  # Shape: (nDirs,)
    num_unique_directions = len(unique_stim_directions_deg)
    # Initialize the spike count matrix `x` with shape (nTrials, nDirs)
    # x_jk: j-th trial (row), k-th direction (column)
    spike_count_matrix_x = np.zeros((nTrials, num_unique_directions))
    # Populate the matrix
    for k_idx, direction_value in enumerate(unique_stim_directions_deg):
        # Extract all spike counts from the 1D 'counts' array that correspond␣
↪to the current 'direction_value'
        counts_for_this_direction = counts[dirs == direction_value]
        # The get_data function should ensure that 'counts_for_this_direction'
        # has exactly 'nTrials' elements.
        if len(counts_for_this_direction) == nTrials:
            spike_count_matrix_x[:, k_idx] = counts_for_this_direction
        else:
            # This part handles unexpected lengths, though get_data should␣
↪prevent this.
            # If fewer than nTrials, pad with zeros (already done by get_data,␣
↪but good for robustness).
            # If more than nTrials (unlikely), take the first nTrials.
            actual_trials_found = len(counts_for_this_direction)
            if actual_trials_found >= nTrials:
                spike_count_matrix_x[:, k_idx] = counts_for_this_direction[:
↪nTrials]
            else:  # actual_trials_found < nTrials
                spike_count_matrix_x[:actual_trials_found, k_idx] = (
```

```
                    counts_for_this_direction
                )
                # The remaining (nTrials - actual_trials_found) elements will␣
 ↪stay zero
                # due to initialization with np.zeros.
    return spike_count_matrix_x
```

[15]:
```
def inital_von_mises_params(
    mean_counts_to_fit: np.ndarray, unique_dirs_rad: np.ndarray
) -> tuple:
    """Initial guess for the von Mises parameters based on mean counts.
    Parameters
    ----------
    mean_counts_to_fit: np.ndarray
        The mean counts for each direction.
    unique_dirs_rad: np.ndarray
        The unique directions in radians.
    Returns
    -------
    tuple: (alpha_guess, kappa_guess, nu_guess, phi_guess_rad)
        Initial guesses for the parameters of the von Mises function.
    """
    # Robust initial guesses:
    if not np.any(
        mean_counts_to_fit > 1e-9
    ):  # Handle cases where all mean counts are zero or tiny
        alpha_guess = np.log(1e-6)  # A small baseline
        phi_guess_rad = 0.0  # Default preferred direction
    else:
        # For alpha, use log of mean of positive counts, or log of max if all␣
 ↪else fails
        positive_mean_counts = mean_counts_to_fit[mean_counts_to_fit > 1e-9]
        if len(positive_mean_counts) > 0:
            alpha_guess = np.log(np.maximum(1e-6, np.
 ↪mean(positive_mean_counts)))
        else:  # Should be caught by the outer if, but as a fallback
            alpha_guess = np.log(np.maximum(1e-6, np.max(mean_counts_to_fit)))

        phi_guess_rad = unique_dirs_rad[np.argmax(mean_counts_to_fit)]

    kappa_guess = 1.0  # Initial guess for bimodal strength
    nu_guess = 1.0  # Initial guess for unimodal strength
    return alpha_guess, kappa_guess, nu_guess, phi_guess_rad
```

[16]:
```
def plot_tuning_curve_fit(
    neuron_id: int,
    counts: np.ndarray,
```

```
    dirs: np.ndarray,
    mean_counts_to_fit: np.ndarray,
    p_opt: np.ndarray,
    r_squared: float = None,
) -> None:
    """Plot the tuning curve fit for a single neuron.
    Parameters
    ----------
    neuron_id: int
        The ID of the neuron being plotted.
    counts: np.ndarray
        The spike counts for each trial.
    dirs: np.ndarray
        The stimulus directions for each trial.
    mean_counts_to_fit: np.ndarray
        The mean counts for each direction.
    p_opt: np.ndarray
        The optimized parameters from the von Mises fit.
    r_squared: float, optional
        The R-squared value of the fit, if available.
    """
    plt.figure(figsize=(10, 7))  # Adjusted figure size for better layout
    # 1. Plot individual trial spike counts (scatter plot)
    # Add some jitter to x-values for better visibility if many trials per␣
↪direction
    unique_stim_directions_deg = np.unique(dirs)
    unique_dir_vals_plot = np.unique(dirs)
    dir_spacing = (
        np.min(np.diff(unique_dir_vals_plot)) if len(unique_dir_vals_plot) > 1␣
↪else 22.5
    )  # Default spacing
    jitter_strength = dir_spacing * 0.05  # Small jitter

    # Create a mapping from original direction to jittered direction for␣
↪consistency
    # Only apply jitter if there are multiple trials for a direction, otherwise␣
↪center it.
    jittered_dirs = np.copy(dirs).astype(float)
    for ud in unique_dir_vals_plot:
        trials_for_dir_mask = dirs == ud
        num_trials_this_dir = np.sum(trials_for_dir_mask)
        if num_trials_this_dir > 1:
            jittered_dirs[trials_for_dir_mask] += np.random.uniform(
                -jitter_strength, jitter_strength, num_trials_this_dir
            )

    plt.scatter(
```

```
        jittered_dirs,
        counts,
        alpha=0.3,
        s=25,
        color="darkgray",
        label="Individual Trial Counts",
        zorder=1,
    )

    # 2. Plot average spike counts per direction
    # Calculate standard error of the mean (SEM) for error bars if desired
    # sem_counts = np.std(spike_count_matrix_x, axis=0) / np.
↪sqrt(spike_count_matrix_x.shape[0])
    # plt.errorbar(unique_stim_directions_deg, mean_counts_to_fit,␣
↪yerr=sem_counts,
    #              fmt='o', color='dodgerblue', markersize=8, capsize=5,
    #              label='Mean Counts (±SEM)', zorder=2)
    # Or simpler, just plot mean points:
    plt.plot(
        unique_stim_directions_deg,
        mean_counts_to_fit,
        "o",
        color="dodgerblue",
        markersize=8,
        label="Mean Spike Count",
        zorder=2,
    )

    # 3. Plot the fitted von Mises curve (if fit was successful)
    if p_opt is not None:
        theta_plot_deg = np.linspace(0, 360, 361)  # Smooth range of angles in␣
↪degrees
        theta_plot_rad = np.deg2rad(
            theta_plot_deg
        )  # Convert to radians for vonMises function

        fitted_values = vonMises(theta_plot_rad, *p_opt)

        plt.plot(
            theta_plot_deg,
            fitted_values,
            "-",
            color="red",
            linewidth=2.5,
            label=(
                f"Von Mises Fit (R²={r_squared:.2f})"
                if r_squared is not None
```

```
                else "Von Mises Fit"
            ),
            zorder=3,
        )
    else:
        logger.info(f"Neuron {neuron_id if neuron_id else 'Unknown'}: No fit to␣
↪plot.")

    plt.xlabel("Direction of motion (degrees)")
    plt.ylabel("Spike count")
    title_parts = ["Tuning Curve"]
    if neuron_id is not None:
        title_parts.append(f"Neuron {neuron_id}")
    if p_opt is None:
        title_parts.append("(Fit Failed)")

    plt.title(" - ".join(title_parts))
    plt.xticks(np.arange(0, 361, 45))
    plt.xlim(-10, 370)  # Give a bit of space around 0 and 360

    # Adjust y-limits dynamically
    min_y_val = 0
    if len(counts) > 0:
        min_y_val = min(0, np.min(counts) - 1)
        max_y_val = np.max(counts) + max(
            1, np.std(counts) * 0.5
        )  # Ensure some space above max count
        if p_opt is not None:  # Also consider fitted curve for ymax
            max_y_val = max(max_y_val, np.max(fitted_values) * 1.1)
        plt.ylim(bottom=min_y_val, top=max_y_val)
    else:
        plt.ylim(bottom=0, top=1)  # Default if no counts

    plt.legend(loc="upper right")
    plt.grid(True, linestyle="--", alpha=0.6)
    plt.tight_layout()
    plt.show()
```

[17]:
```
def tuningCurve(
    counts: np.ndarray, dirs: np.ndarray, show: bool = True, neuron_id=None
) -> np.ndarray:
    """Fit a von Mises tuning curve to the spike counts in count with
    direction dir using a **least-squares fit**.

    Parameters
    ----------
```

```
    counts: np.array, shape=(total_n_trials, )
        the spike count during the stimulation period

    dirs: np.array, shape=(total_n_trials, )
        the stimulus direction in degrees

    show: bool, default=True
        Plot or not.
    neuron_id: int, optional
        The ID of the neuron being fitted. Used for logging and plotting.
        If None, will be set to 'Unknown' in logs and plots.

    Return
    ------
    p: np.array or list, (4,)
        parameter vector of tuning curve function
    """
    # ----------------------------------------
    # Compute the spike count matrix (0.5 pts)
    # ----------------------------------------
    logger.info("Fitting tuning curve...")
    logger.info(f"Counts: {counts.shape}")
    logger.info(f"Dirs: {dirs.shape}")

    spike_count_matrix_x = compute_spike_count_matrix(counts, dirs)

    logger.info(f"Spike count matrix shape: {spike_count_matrix_x.shape}")
    logger.info(f"Spike count matrix: {spike_count_matrix_x}")

    # ------------------------------------------------------------
    # fit the von Mises tuning curve to the spike counts (0.5 pts)
    # ------------------------------------------------------------

    # 1. Calculate mean spike counts per direction
    mean_counts_to_fit = np.mean(spike_count_matrix_x, axis=0)
    logger.info(f"Mean counts to fit: {mean_counts_to_fit}")

    # 2. Get unique directions (degrees) and convert to radians
    # 'dirs' is the original 1D array of directions for all trials passed to␣
↪tuningCurve
    unique_stim_directions_deg = np.unique(dirs)
    unique_dirs_rad = np.deg2rad(unique_stim_directions_deg)
    logger.info(f"Unique directions (radians) for fitting: {unique_dirs_rad}")

    # Check if there's enough data to fit (at least as many points as␣
↪parameters)
    if len(unique_dirs_rad) < 4:  # vonMises has 4 parameters
```

```
        logger.warning(
            f"Not enough unique directions ({len(unique_dirs_rad)}) to fit the␣
↪von Mises model. Need at least 4. Skipping fit."
        )
        p_opt = None  # Indicate fit failed
    else:
        alpha_guess, kappa_guess, nu_guess, phi_guess_rad =␣
↪inital_von_mises_params(
            mean_counts_to_fit, unique_dirs_rad
        )
        p0 = [alpha_guess, kappa_guess, nu_guess, phi_guess_rad]
        logger.info(f"Initial parameter guesses (p0): {p0}")
        # Bounds: alpha, kappa>=0, nu>=0, phi in [0, 2*pi]
        bounds = ([-np.inf, 0, 0, 0], [np.inf, np.inf, np.inf, 2 * np.pi])
        # Fit the von Mises function to the mean counts
        # 4. Perform the non-linear least squares fit
        r_squared = None
        try:
            p_opt, p_cov = opt.curve_fit(
                f=vonMises,  # Your vonMises function (make sure it's defined␣
↪and accessible)
                xdata=unique_dirs_rad,
                ydata=mean_counts_to_fit,
                p0=p0,
                bounds=bounds,
                maxfev=5000,  # Maximum number of function evaluations
            )
            residuals = mean_counts_to_fit - vonMises(unique_dirs_rad, *p_opt)
            ss_res = np.sum(residuals**2)
            ss_tot = np.sum((mean_counts_to_fit - np.mean(mean_counts_to_fit))␣
↪** 2)
            if ss_tot == 0:  # Avoid division by zero if all mean_counts are␣
↪the same
                r_squared = 1.0 if ss_res < 1e-9 else 0.0
            else:
                r_squared = 1 - (ss_res / ss_tot)
            logger.info(
                f"Neuron {neuron_id if neuron_id else 'Unknown'}: R-squared =␣
↪{r_squared:.3f}"
            )
            logger.info(f"Optimized parameters (p_opt): {p_opt}")
        except RuntimeError:
            logger.warning(
                "RuntimeError: Optimal parameters not found during curve_fit.␣
↪Fit failed."
            )
```

```
            p_opt = (
                None  # Or assign np.full(4, np.nan) if you prefer NaNs for␣
↪failed fits
            )
        except ValueError as e:
            logger.warning(f"ValueError during curve_fit: {e}. Fit failed.")
            p_opt = None

        if show:
            # -----------------------------------------
            # plot the data and fitted tuning curve (1 pt)
            # -----------------------------------------
            plot_tuning_curve_fit(
                neuron_id=neuron_id,
                counts=counts,  # spike_count_matrix_x.flatten(),
                dirs=dirs,
                mean_counts_to_fit=mean_counts_to_fit,
                p_opt=p_opt,
                r_squared=r_squared,  # Replace with actual R-squared value if␣
↪available
                # r_squared=1 - (np.sum((mean_counts_to_fit -␣
↪vonMises(unique_dirs_rad, *p_opt))**2) / np.sum((mean_counts_to_fit - np.
↪mean(mean_counts_to_fit))**2))
            )
        return p_opt
```

Plot tuning curve and fit for different neurons. Good candidates to check are 28, 29 or 37.

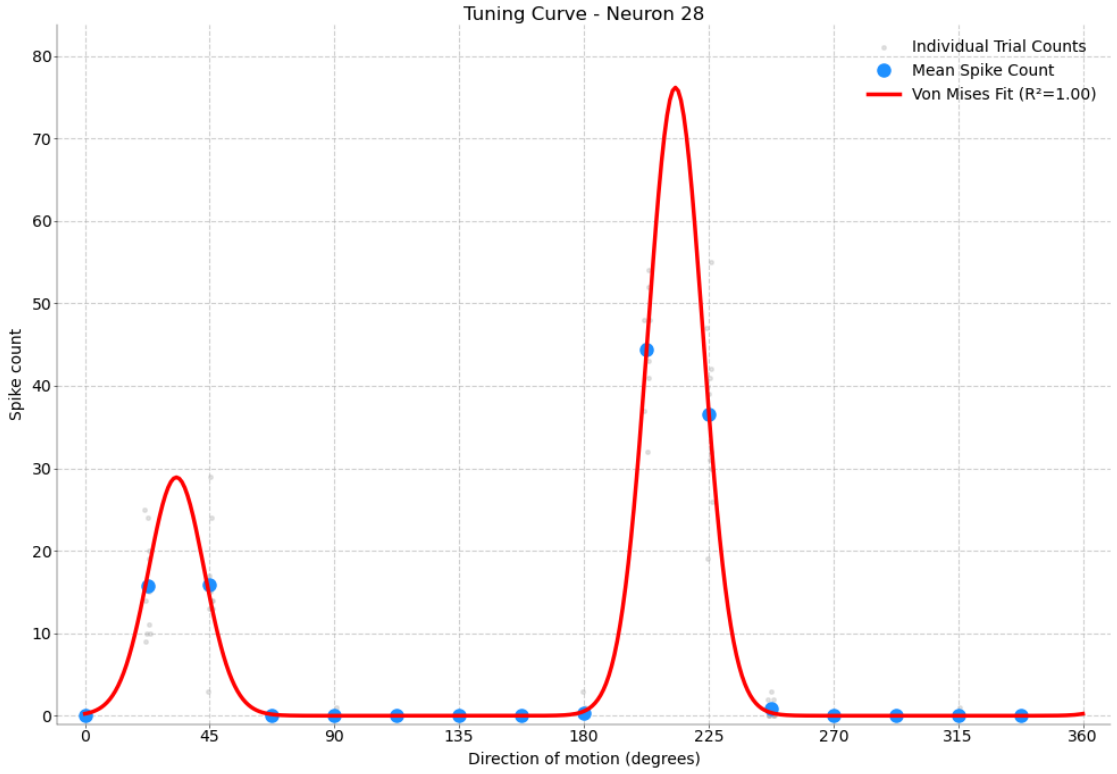
[18]:
```
def get_data(spikes, neuron):
    spk_by_dir = (
        spikes[spikes["Neuron"] == neuron]
        .groupby(["Dir", "Trial"])["stimPeriod"]
        .sum()
        .astype(int)
        .reset_index()
    )

    dirs = spk_by_dir["Dir"].values
    counts = spk_by_dir["stimPeriod"].values

    # because we count spikes only when they are present, some zero entries in␣
↪the count vector are missing
    for i, Dir in enumerate(np.unique(spikes["Dir"])):
        m = nTrials - np.sum(dirs == Dir)
        if m > 0:
            dirs = np.concatenate((dirs, np.ones(m) * Dir))
            counts = np.concatenate((counts, np.zeros(m)))
```

```
    idx = np.argsort(dirs)
    dirs_sorted = dirs[idx]  # sorted dirs
    counts_sorted = counts[idx]

    return dirs_sorted, counts_sorted
```

[19]:
```
# ------------------------------------------------------------
# plot the average number of spikes per direction, the spike
# counts from individual trials as well as your optimal fit
# for different neurons (0.5 pts)
# ------------------------------------------------------------
neurons_to_plot = [28, 29, 37]
for neuron in neurons_to_plot:
    dirs_sorted, counts_sorted = get_data(spikes, neuron)
    result = tuningCurve(counts_sorted, dirs_sorted, show=True,␣
 ↪neuron_id=neuron)
    if result is not None:
        print(
            f"Neuron {neuron}: dirs_sorted.shape = {dirs_sorted.shape},␣
 ↪counts_sorted.shape = {counts_sorted.shape}"
        )
    else:
        print(f"Neuron {neuron}: No result from tuningCurve()")
```

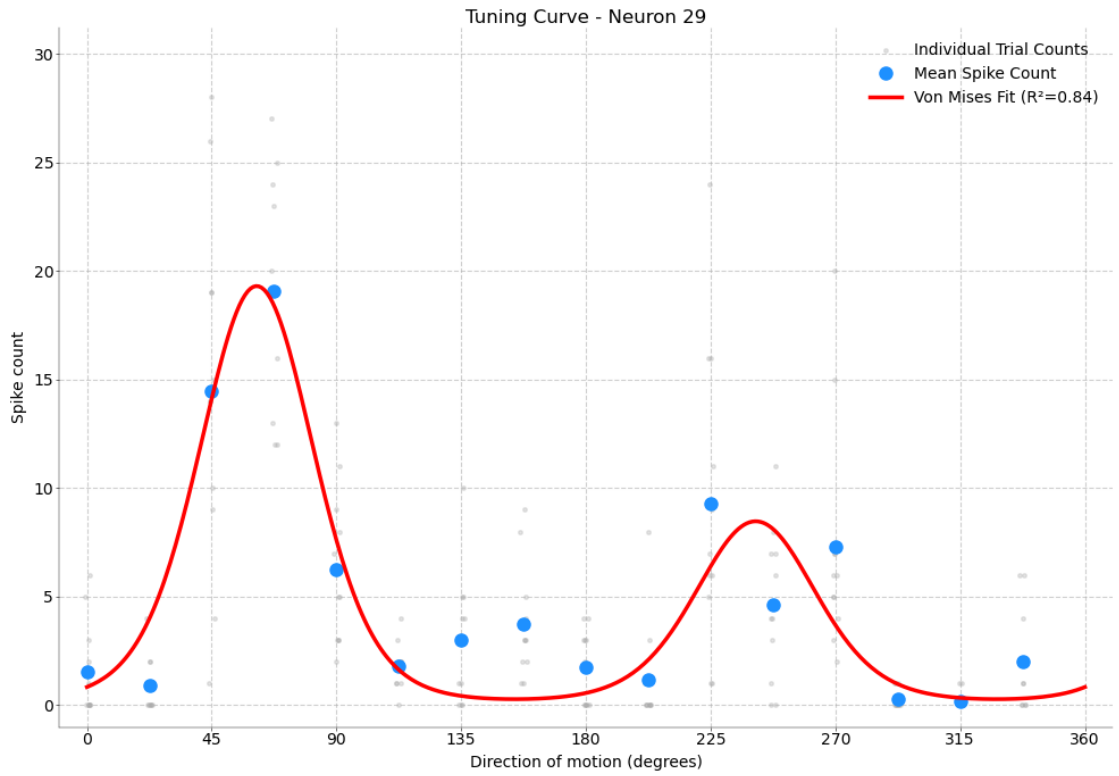
```
INFO:__main__:Fitting tuning curve…

INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 0. 16. 14.  0.  0.  0.  0.  0.  0. 32. 31.
0.  0.  0.  0.  0.]
 [ 0. 10. 13.  0.  0.  0.  0.  0.  0. 44. 19.  0.  0.  0.  0.  0.]
 [ 0. 20. 14.  0.  0.  0.  0.  0.  0. 41. 42.  0.  0.  0.  0.  0.]
 [ 0. 10. 16.  0.  0.  0.  0.  0.  0. 54. 32.  0.  0.  0.  0.  0.]
 [ 0. 24. 16.  0.  0.  0.  0.  0.  0. 43. 55.  0.  0.  0.  0.  0.]
 [ 0.  9. 13.  0.  0.  0.  0.  0.  0. 52. 41.  0.  0.  0.  0.  0.]
 [ 0. 19. 15.  0.  0.  0.  0.  0.  0. 44. 30.  3.  0.  0.  0.  0.]
 [ 0. 15.  3.  0.  0.  0.  0.  0.  0. 48. 47.  1.  0.  0.  0.  0.]
 [ 0. 25. 24.  0.  0.  0.  0.  0.  3. 45. 26.  1.  0.  0.  0.  0.]
 [ 0. 11. 29.  0.  1.  0.  0.  0.  1. 37. 40.  2.  0.  0.  0.  0.]
 [ 0. 14. 17.  0.  0.  0.  0.  0.  0. 48. 39.  2.  0.  0.  1.  0.]]
INFO:__main__:Mean counts to fit: [ 0.         15.72727273 15.81818182  0.
0.09090909  0.
  0.          0.          0.36363636 44.36363636 36.54545455  0.81818182
  0.          0.          0.09090909  0.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
```

```
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.6551607371818373, 1.0, 1.0,
3.5342917352885173]
INFO:__main__:Neuron 28: R-squared = 0.999
INFO:__main__:Optimized parameters (p_opt): [4.33282731 8.18559981 0.48435863
3.71739285]
/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/4021924317.py:1
25: UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```

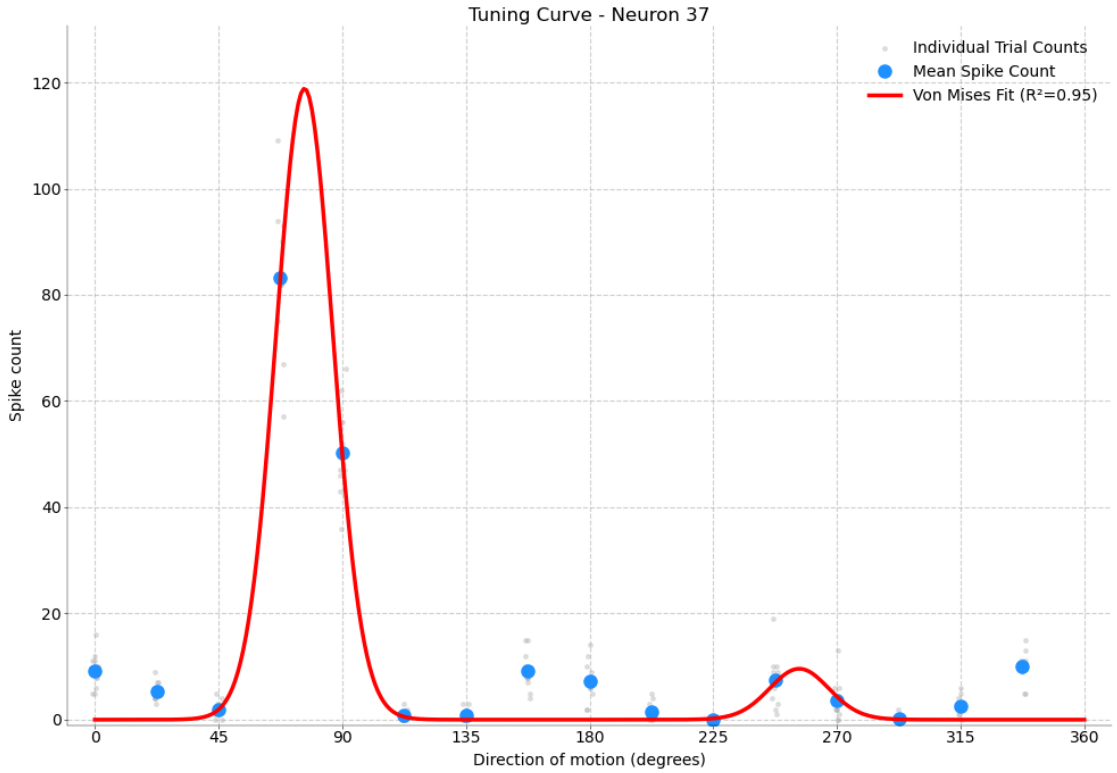



```
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 5.  2.  4. 13.  2.  0.  0.  9.  0.  1. 24.
0.  2.  0.  1.  1.]
 [ 0.  0. 19. 20.  8.  3.  0.  3.  2.  0.  9.  8.  4.  0.  0.  2.]
 [ 0.  0.  9. 19. 13.  1.  0.  4.  0.  0.  6.  0. 15.  0.  0.  6.]
 [ 0.  0.  1. 16.  3.  1.  5.  2.  0.  1.  7. 11.  7.  0.  0.  6.]
 [ 0.  0. 28. 12.  3.  1. 10.  1.  0.  0.  1.  7.  6.  0.  0.  1.]
 [ 0.  0. 14. 25.  7.  4.  1.  8.  0.  0.  1.  4.  5.  0.  0.  0.]
 [ 0.  0. 15. 12.  9.  2.  1.  3.  3.  0. 11.  3.  7.  0.  0.  2.]
```

```
 [ 6.  4. 19. 27.  5.  2.  4.  5.  4.  0. 16.  6.  6.  1.  0.  0.]
 [ 3.  2. 14. 23.  5.  2.  3.  2.  4.  0. 16.  4.  5.  1.  0.  0.]
 [ 2.  1. 10. 19. 11.  2.  5.  1.  3.  8.  5.  7.  3.  0.  0.  4.]
 [ 1.  1. 26. 24.  3.  2.  4.  3.  3.  3.  6.  1. 20.  1.  1.  0.]]
INFO:__main__:Mean counts to fit: [ 1.54545455  0.90909091 14.45454545
19.09090909  6.27272727  1.81818182
  3.          3.72727273  1.72727273  1.18181818  9.27272727  4.63636364
  7.27272727  0.27272727  0.18181818  2.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [1.5759281335352222, 1.0, 1.0,
1.1780972450961724]
INFO:__main__:Neuron 29: R-squared = 0.841
INFO:__main__:Optimized parameters (p_opt): [2.96043674 1.90155694 0.41176043
1.06863547]

Neuron 28: dirs_sorted.shape = (176,), counts_sorted.shape = (176,)

/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/4021924317.py:1
25: UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```



```
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 10.   4.   1.  82.  66.   0.   0.  15.   5.
0.   0.   3.   6.   2.
    0.  11.]
 [ 11.   9.   0.  84.  62.   0.   0.   8.  12.   0.   0.   9.   4.   0.
    2.   5.]
 [  5.   5.   1.  82.  52.   0.   0.  15.   2.   0.   0.   2.   4.   0.
    1.  10.]
 [  8.   5.   2.  82.  43.   0.   0.   4.   7.   3.   0.  19.   2.   0.
    6.  13.]
 [ 10.   4.   5.  94.  48.   0.   1.   9.   9.   1.   0.   9.   1.   0.
    2.   9.]
 [ 11.   6.   0.  57.  47.   3.   2.   8.  10.   1.   0.   6.  13.   0.
    2.  11.]
 [  5.   3.   2. 109.  47.   2.   3.  10.   6.   5.   0.  10.   2.   0.
    3.  10.]
 [  6.   4.   3.  67.  50.   1.   1.   8.   2.   1.   0.   8.   2.   0.
    5.   5.]
 [ 12.   5.   1.  75.  46.   1.   3.   5.   7.   1.   0.  10.   0.   0.
    1.  11.]
 [ 16.   7.   4.  90.  56.   1.   0.   7.   6.   1.   0.   4.   0.   0.
    3.  10.]
 [  8.   7.   1.  93.  36.   1.   0.  12.  14.   4.   0.   1.   6.   0.
    4.  15.]]
INFO:__main__:Mean counts to fit: [ 9.27272727  5.36363636  1.81818182
83.18181818 50.27272727  0.81818182
  0.90909091  9.18181818  7.27272727  1.54545455  0.          7.36363636
  3.63636364  0.18181818  2.63636364 10.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.5569923765609546, 1.0, 1.0,
1.1780972450961724]
INFO:__main__:Neuron 37: R-squared = 0.952
INFO:__main__:Optimized parameters (p_opt): [4.77770423 7.33355117 1.25860608
1.33130282]

Neuron 29: dirs_sorted.shape = (176,), counts_sorted.shape = (176,)

/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/4021924317.py:1
25: UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```




```
Neuron 37: dirs_sorted.shape = (176,), counts_sorted.shape = (176,)
```

1.4.1 Tuning Curves for neurons visualized

[20]:
```
all_neuron_ids = spikes["Neuron"].unique()  # Or a subset you want to plot
num_neurons = len(all_neuron_ids)
ncols = 4  # Example: 4 plots per row
nrows = int(np.ceil(num_neurons / ncols))

fig, axes = plt.subplots(
    nrows, ncols, figsize=(ncols * 5, nrows * 4), sharex=True, sharey="row"
)  # Adjust sharey as needed
axes = axes.flatten()  # To easily iterate through axes


for i, neuron_id in enumerate(all_neuron_ids):
    if i >= len(axes):  # Stop if we run out of subplots
        break
    ax = axes[i]
    dirs_sorted, counts_sorted = get_data(spikes, neuron_id)

    # Get fit parameters *without* showing individual plot
```

```
    p_opt = None
    r_sq = None
    try:
        p_opt = tuningCurve(counts_sorted, dirs_sorted, neuron_id=neuron_id,␣
↪show=False)
    except Exception as e:
        logger.warning(f"Error fitting tuning curve for neuron {neuron_id}:␣
↪{e}")
        p_opt = None
    # You'd also need mean_counts_to_fit and unique_stim_directions_deg here
    spike_count_matrix_x = compute_spike_count_matrix(counts_sorted,␣
↪dirs_sorted)
    mean_counts_to_fit = np.mean(spike_count_matrix_x, axis=0)
    unique_stim_directions_deg = np.unique(dirs_sorted)

    ax.plot(unique_stim_directions_deg, mean_counts_to_fit, "o-")
    if p_opt is not None:
        theta_plot_deg = np.linspace(0, 360, 100)
        fitted_values = vonMises(np.deg2rad(theta_plot_deg), *p_opt)
        ax.plot(theta_plot_deg, fitted_values, "r-")
    ax.set_title(f"Neuron {neuron_id}")
    ax.set_xticks(np.arange(0, 361, 90))  # Fewer ticks for subplots

# Hide any unused subplots
for j in range(i + 1, nrows * ncols):
    if j < len(axes):  # Check if axes[j] exists
        fig.delaxes(axes[j])

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

```
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[37. 32. 39. 41. 61. 50. 42. 48. 48. 39. 36.
37. 39. 25. 43. 48.]
 [55. 71. 39. 38. 42. 54. 31. 40. 42. 27. 58. 61. 58. 35. 39. 84.]
 [50. 31. 39. 62. 54. 38. 52. 49. 35. 47. 49. 38. 35. 47. 47. 58.]
 [41. 38. 32. 56. 23. 30. 43. 44. 37. 38. 56. 39. 48. 55. 37. 53.]
 [47. 53. 48. 50. 52. 48. 35. 76. 58. 37. 43. 53. 50. 58. 34. 49.]
 [56. 34. 45. 48. 41. 46. 37. 24. 36. 43. 45. 50. 45. 41. 36. 52.]
 [41. 36. 40. 56. 51. 42. 60. 40. 68. 39. 58. 27. 67. 54. 60. 46.]
 [44. 40. 57. 78. 52. 54. 56. 58. 63. 55. 42. 51. 55. 45. 49. 46.]
 [49. 62. 86. 66. 49. 43. 62. 63. 56. 51. 52. 47. 51. 48. 51. 55.]
 [53. 79. 62. 55. 81. 68. 61. 65. 67. 50. 58. 74. 62. 79. 64. 44.]
 [36. 48. 27. 43. 71. 34. 53. 50. 75. 63. 79. 58. 66. 63. 41. 31.]]
```

```
INFO:__main__:Mean counts to fit: [46.27272727 47.63636364 46.72727273
53.90909091 52.45454545 46.09090909
 48.36363636 50.63636364 53.18181818 44.45454545 52.36363636 48.63636364
 52.36363636 50.         45.54545455 51.45454545]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [3.8995592913993082, 1.0, 1.0,
1.1780972450961724]
INFO:__main__:Neuron 1: R-squared = 0.073
INFO:__main__:Optimized parameters (p_opt): [3.92255983e+00 2.31375385e-02
3.85252615e-17 1.41425082e+00]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[34. 43. 39.  9. 33. 38. 43. 23. 32. 43. 35.
12. 15. 46. 33. 36.]
 [48. 42. 45. 12. 33. 42. 31. 39. 51. 50. 39. 25. 26. 42. 35. 30.]
 [42. 42. 37. 32. 19. 25. 43. 43. 26. 32. 35. 16. 28. 46. 22. 31.]
 [44. 45. 14. 15. 17. 32. 42. 45. 29. 31. 43. 15. 34. 53. 41. 28.]
 [37. 48. 33. 21. 35. 39. 35. 55. 31. 33. 39.  7. 34. 31. 44. 29.]
 [58. 66. 12. 18. 19. 51. 35. 23. 50. 47. 34.  7. 33. 58. 36. 34.]
 [38. 52. 16. 25.  9. 25. 36. 46. 47. 39. 45.  7. 24. 20. 43. 36.]
 [47. 52. 11. 20. 27. 20. 33. 38. 47. 47. 34. 16. 28. 52. 28. 26.]
 [49. 32. 35. 14. 26. 12. 30. 45. 46. 37. 37. 17. 17. 47. 40. 43.]
 [35. 34. 26. 12. 23. 37. 47. 48. 35. 31. 42. 26. 19. 44. 43. 34.]
 [34. 76. 15. 19. 25. 49. 54. 17. 29. 28. 68. 22. 37. 46. 37. 38.]]
INFO:__main__:Mean counts to fit: [42.36363636 48.36363636 25.72727273
17.90909091 24.18181818 33.63636364
 39.         38.36363636 38.45454545 38.         41.         15.45454545
 26.81818182 44.09090909 36.54545455 33.18181818]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [3.524688003737903, 1.0, 1.0,
0.39269908169872414]
INFO:__main__:Neuron 2: R-squared = 0.363
INFO:__main__:Optimized parameters (p_opt): [3.74579482e+00 2.27283702e-01
6.38265043e-03 1.92532328e-27]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[36. 13. 26. 13. 15. 49. 25. 12. 22. 22. 18.
6. 32. 17. 33. 37.]
```

```
 [23. 22. 19. 18. 13. 27. 19. 28. 20. 12. 26. 26. 28. 20. 36. 16.]
 [19. 26. 27. 31. 28.  9. 45. 19. 22. 16. 29. 20. 25. 27. 21. 33.]
 [35.  8.  8. 25. 25. 20. 19. 16.  7.  8. 30. 27. 21. 28. 31. 30.]
 [42. 16. 23. 27. 26. 30. 28. 28. 24. 18. 32.  9. 18. 23. 21. 14.]
 [43. 27. 12. 21. 18. 35. 21. 14. 20. 11. 22. 21. 22. 41. 19. 25.]
 [22. 18. 14. 18. 20. 24. 25. 15. 31. 16. 29.  6. 32. 32. 30. 31.]
 [20. 14. 17. 23. 15. 35. 17.  6. 30. 25. 20. 17. 17. 29. 19. 19.]
 [44. 21. 14. 23. 19. 18. 18. 21. 16. 15. 29. 14. 24. 28. 31. 28.]
 [26. 19. 18. 27. 14. 28. 28. 26. 28. 15. 25. 15. 18. 31. 36. 35.]
 [37. 28.  9. 15. 42. 24. 38. 28. 31. 19. 32. 29. 42. 28. 31.  7.]]
INFO:__main__:Mean counts to fit: [31.54545455 19.27272727 17.
21.90909091 21.36363636 27.18181818
 25.72727273 19.36363636 22.81818182 16.09090909 26.54545455 17.27272727
 25.36363636 27.63636364 28.         25.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [3.146549481454251, 1.0, 1.0, 0.0]
INFO:__main__:Neuron 3: R-squared = 0.074
INFO:__main__:Optimized parameters (p_opt): [3.21831441e+00 2.06931258e-13
7.30914676e-02 2.46783589e-36]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[34. 38. 33. 38. 51. 26. 28. 16. 53. 36. 27.
39. 51. 33. 40. 34.]
 [33. 53. 27. 37. 56. 55. 24. 34. 26. 18. 32. 40. 27. 46. 41. 44.]
 [28. 28. 43. 44. 39. 41. 56. 37. 26. 30. 39. 43. 35. 36. 36. 30.]
 [38. 35. 27. 36. 35. 34. 37. 34. 29. 27. 44. 37. 50. 52. 44. 38.]
 [30. 46. 56. 33. 47. 27. 32. 51. 52. 28. 28. 33. 51. 39. 39. 49.]
 [31. 31. 32. 46. 32. 38. 31. 41. 29. 43. 44. 45. 40. 40. 43. 28.]
 [22. 39. 25. 46. 40. 36. 34. 34. 47. 30. 63. 31. 54. 47. 49. 34.]
 [43. 31. 26. 50. 39. 47. 36. 44. 57. 56. 23. 32. 36. 34. 43. 39.]
 [48. 31. 59. 44. 29. 25. 44. 29. 43. 37. 29. 40. 43. 39. 31. 48.]
 [35. 44. 44. 36. 47. 53. 27. 34. 45. 23. 37. 57. 38. 45. 34. 39.]
 [33. 39. 33. 36. 47. 33. 54. 45. 36. 27. 39. 32. 49. 63. 23. 24.]]
INFO:__main__:Mean counts to fit: [34.09090909 37.72727273 36.81818182
40.54545455 42.         37.72727273
 36.63636364 36.27272727 40.27272727 32.27272727 36.81818182 39.
 43.09090909 43.09090909 38.45454545 37.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [3.6438464276006215, 1.0, 1.0,
4.71238898038469]
```

```
INFO:__main__:Neuron 4: R-squared = 0.524
INFO:__main__:Optimized parameters (p_opt): [3.73465045 0.07740786 0.01498081
4.81697768]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 1.  0.  2.  0.  0.  3.  0.  0.  0.  2.  0.
0.  0.  0.  0.  0.]
 [ 2.  0.  0.  0.  1.  1.  7.  0.  1.  1.  2.  0.  1.  0.  0.  5.]
 [ 2.  0.  0.  0.  3.  1.  4.  2.  2.  1.  0.  0.  1.  0.  5.  2.]
 [ 1.  0.  0.  0.  7.  3.  4.  3.  4.  2.  4.  0.  4.  0.  0.  5.]
 [ 1.  1.  0.  9.  1.  0.  2.  1.  6.  0.  6.  1.  3.  0.  1.  1.]
 [ 5.  1.  0.  4.  2.  0.  7.  2.  1.  0.  1.  3.  0.  2.  1.  1.]
 [ 8.  4.  1.  2.  0.  0.  4.  2.  0.  0.  0.  1.  0.  1.  1.  1.]
 [ 0.  2.  5.  2.  1.  2.  2.  1.  5.  0.  1.  0.  0.  1.  3.  3.]
 [ 0.  1.  4.  2.  0. 14.  1.  2.  0.  0.  0.  1.  0.  2.  0.  2.]
 [ 0.  6.  1.  2.  0.  1.  9. 11.  0.  0.  0.  1.  0.  1.  0.  0.]
 [ 0.  4.  0.  1.  0.  0.  5.  1.  1.  3.  0.  2.  0.  2.  5.  0.]]
INFO:__main__:Mean counts to fit: [1.81818182 1.72727273 1.18181818 2.
1.36363636 2.27272727
 4.09090909 2.27272727 1.81818182 0.81818182 1.27272727 0.81818182
 0.81818182 0.81818182 1.45454545 1.81818182]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [0.49939692794236806, 1.0, 1.0,
2.356194490192345]
INFO:__main__:Neuron 5: R-squared = 0.604
INFO:__main__:Optimized parameters (p_opt): [1.14069805 0.38917036 0.33004495
2.40863861]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[43. 22. 16.  6. 11. 10.  6. 13. 18. 38.  7.
1.  2.  5. 10. 15.]
 [20. 26. 10.  6.  6. 12.  6. 21. 11. 37. 12. 11.  8. 14. 11. 24.]
 [15. 51. 15.  4.  6.  7.  9.  7. 13. 26.  5.  9.  2.  4. 27. 13.]
 [32. 21.  2.  5.  9.  4.  9. 12. 20. 21.  7.  4.  6. 10. 11. 11.]
 [16. 52.  7.  7. 11. 13.  7. 10. 11. 26. 29.  2.  6.  2. 26. 14.]
 [39. 31.  2.  4. 11.  6.  2. 12. 63. 24.  4.  4. 16. 18. 16. 24.]
 [23. 33.  4.  4.  3.  2.  5.  8. 27. 23.  7.  1.  1. 12.  3. 13.]
 [21. 36.  4.  5.  2. 15.  8. 11. 27. 30.  6.  3.  6. 11.  3.  7.]
 [30. 15. 21.  3. 16.  1.  4. 29. 14. 14.  6.  3.  5.  4. 11. 15.]
 [45. 22.  6.  2.  5.  5. 10. 16. 21. 15. 11.  4.  6.  5. 16. 13.]
 [19. 45. 11. 10.  6.  2. 10.  9. 20. 15. 13.  4.  6. 14. 15. 19.]]
```

```
INFO:__main__:Mean counts to fit: [27.54545455 32.18181818  8.90909091
5.09090909  7.81818182  7.
  6.90909091 13.45454545 22.27272727 24.45454545  9.72727273  4.18181818
  5.81818182  9.         13.54545455 15.27272727]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.589556685842228, 1.0, 1.0,
0.39269908169872414]
INFO:__main__:Neuron 6: R-squared = 0.851
INFO:__main__:Optimized parameters (p_opt): [3.41179424 0.93832691 0.11870433
0.13948939]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 8.  0.  0.  0.  0.  6.  1.  1.  0.  4.  0.
0.  0.  0.  0.  0.]
 [11.  0.  0.  0.  0.  0.  3.  1.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 5.  5.  0.  0.  0.  1.  1.  1.  1.  2.  0.  0.  0.  0.  0.  0.]
 [ 3.  2.  0.  0.  0.  2.  1.  0.  4.  1.  0.  0.  0.  0.  0.  0.]
 [ 4.  3.  0.  0.  0.  0.  2.  0.  2.  3.  0.  0.  0.  0.  0.  2.]
 [13.  0.  0.  0.  0.  1.  0.  0.  1.  2.  0.  0.  0.  0.  1.  0.]
 [ 3.  7.  0.  0.  0.  0.  0.  0. 21.  0.  0.  0.  0.  0.  3.  0.]
 [ 9.  1.  0.  0.  1.  0.  0.  0.  6.  0.  0.  0.  0.  1.  1.  0.]
 [ 9.  1.  0.  0.  1.  0.  0.  0.  2.  3.  0.  0.  2.  0.  1.  1.]
 [11.  1.  1.  0.  0.  0.  0.  0.  4.  1.  0.  0.  0.  1.  1.  0.]
 [ 3.  3.  0.  2.  1.  0.  0.  0.  4.  0.  1.  0.  1.  1.  1.  0.]]
INFO:__main__:Mean counts to fit: [7.18181818 2.09090909 0.09090909 0.18181818
0.27272727 0.90909091
 0.72727273 0.27272727 4.09090909 1.54545455 0.09090909 0.
 0.27272727 0.27272727 0.72727273 0.27272727]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [0.23638877806423053, 1.0, 1.0,
0.0]
INFO:__main__:Neuron 7: R-squared = 0.959
INFO:__main__:Optimized parameters (p_opt): [2.08811474 7.2557649  0.27104786
0.09155932]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[26. 43. 11. 21. 24. 38. 49. 29. 19. 24. 18.
21. 17. 24. 25. 24.]
```

```
 [17. 15. 37. 22. 10. 49. 44. 36. 25. 18. 25.  1.  5. 21. 35. 16.]
 [14. 15. 27. 33. 15. 40. 44. 44. 14. 30. 23. 12. 35. 28. 44. 56.]
 [21. 22. 27.  6. 32. 10. 24. 25. 21. 23. 41. 20. 28. 72. 25. 22.]
 [28. 30.  8. 12. 16. 43. 47. 34. 22.  8. 22.  5. 26. 30. 24. 34.]
 [42. 38.  5.  7. 35. 33. 34. 32. 12. 31. 23. 16. 24. 49. 23. 27.]
 [34. 18. 10. 21.  3. 35. 30. 39. 60. 28.  9.  9. 13. 30. 34. 41.]
 [16. 23.  6. 22. 31.  6. 21. 33. 34.  3. 22. 34. 29. 63. 29. 37.]
 [26. 12.  6.  7. 30. 22. 34. 48. 11. 23. 40. 12. 12. 27. 53. 35.]
 [22.  1. 29. 11. 25. 43. 28. 25.  4. 38. 24.  3. 16. 51. 56. 34.]
 [ 5.  6. 18. 32. 19. 42. 51. 26. 23. 10. 10. 32. 71. 39. 39. 89.]]
INFO:__main__:Mean counts to fit: [22.81818182 20.27272727 16.72727273
17.63636364 21.81818182 32.81818182
 36.90909091 33.72727273 22.27272727 21.45454545 23.36363636 15.
 25.09090909 39.45454545 35.18181818 37.72727273]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [3.2730626562097873, 1.0, 1.0,
5.105088062083414]
INFO:__main__:Neuron 8: R-squared = 0.858
INFO:__main__:Optimized parameters (p_opt): [3.67003479 0.40119258 0.03687164
5.50779121]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 1.  0.  0.  0.  0.  0.  0.  4.  0. 14.  1.
0.  1.  1.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0. 17.  0.  0.  1.  0. 24.  1.  0.  0.]
 [ 0.  0. 12.  0.  0.  0. 25.  0.  0.  0.  0.  0.  1.  1.  1.  0.]
 [ 0.  0. 10.  0.  0.  0.  5. 12.  4.  0.  0.  0.  2.  8.  3.  0.]
 [ 3.  0.  1. 19.  0.  3.  1.  0.  2.  0.  0.  0.  2.  2.  4.  1.]
 [ 0.  0.  0. 18.  0.  4.  4.  1.  1.  2.  0.  2.  2.  0.  4.  0.]
 [ 2.  0.  0.  1.  0.  2.  6.  0.  1.  1.  0.  2.  2.  0.  3.  0.]
 [ 1.  0.  0.  4.  2.  0.  0.  7.  1.  1.  0.  1.  0.  0.  5.  5.]
 [15.  1.  0.  2.  2.  3.  3. 19.  6.  1.  0.  1.  0.  0. 10.  0.]
 [ 1.  3.  0.  1.  2.  1.  4.  8.  1.  5.  0.  5.  2.  2.  4.  0.]
 [ 3.  0.  0.  0. 16.  8.  8.  0.  0.  0. 15.  0.  1.  1.  3.  0.]]
INFO:__main__:Mean counts to fit: [2.36363636 0.36363636 2.18181818 4.09090909
2.         1.90909091
 5.09090909 6.18181818 1.45454545 2.18181818 1.54545455 1.
 3.36363636 1.45454545 3.36363636 0.54545455]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [0.8933012136494562, 1.0, 1.0,
```

```
2.748893571891069]
INFO:__main__:Neuron 9: R-squared = 0.351
INFO:__main__:Optimized parameters (p_opt): [1.53712907 0.36893088 0.35499363
2.45850171]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[15. 17.  0.  0.  0.  0.  4.  3.  4. 15.  0.
0.  3.  0.  0. 10.]
 [ 0.  6.  0.  0.  8.  1.  0.  4.  1. 21.  0.  0.  1.  3.  0.  8.]
 [ 8.  7.  0. 13.  2.  2.  5.  5. 18. 21.  0.  5. 15. 13.  7. 14.]
 [ 8.  6.  1. 11.  3.  4.  5. 16. 18. 11.  1. 17.  1. 10.  1.  4.]
 [10.  0.  1.  9. 13.  4. 10.  6. 29. 13.  0.  2.  8.  6.  2. 14.]
 [ 0. 21.  1.  6.  4.  1.  1. 12.  4. 15.  0.  3. 13.  0.  7.  5.]
 [16.  4.  2.  8.  1.  4.  6. 15. 34. 10.  0.  2.  1.  0.  1.  3.]
 [15.  6.  0.  4. 15.  1.  4. 23.  2. 20.  1.  3.  0.  2.  2.  3.]
 [12.  1.  6. 19.  2.  4.  5.  9.  3. 13.  3.  9.  0.  0.  1.  3.]
 [ 2.  1. 10.  2.  4.  4. 16.  1.  0.  1.  0. 10. 35.  0. 18.  2.]
 [28. 13.  1.  1. 17.  1.  2. 17.  1. 11.  0. 15.  2.  4.  0.  3.]]
INFO:__main__:Mean counts to fit: [10.36363636  7.45454545  2.
6.63636364  6.27272727  2.36363636
  5.27272727 10.09090909 10.36363636 13.72727273  0.45454545  6.
  7.18181818  3.45454545  3.54545455  6.27272727]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [1.8470221479031044, 1.0, 1.0,
3.5342917352885173]
INFO:__main__:Neuron 10: R-squared = 0.421
INFO:__main__:Optimized parameters (p_opt): [2.43748777 0.53169515 0.14873161
3.169115  ]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 3.  0.  0.  3.  0.  3.  3.  1.  0.  5.  0.
0.  0.  5.  0.  0.]
 [ 1.  0.  0.  0.  0.  5.  4.  0.  0.  4.  0.  0. 10.  0.  0.  0.]
 [ 2.  0.  0.  0.  0.  3.  7.  0.  0.  0.  0.  0.  2.  0.  1.  0.]
 [ 0.  0.  0.  0.  3.  0.  0.  0.  0.  0.  0.  0. 10.  1.  4.  0.]
 [ 0.  0.  4.  1.  2.  0.  0.  0.  0.  0.  0.  0. 15.  0.  7.  0.]
 [ 0.  0. 12.  0.  2.  0.  0.  0.  0.  0.  0.  0.  7.  0.  2.  0.]
 [ 0.  2.  7.  0. 14.  0.  0.  0.  0.  2.  0.  8.  1.  0.  0.  1.]
 [ 0.  1.  6.  0.  3.  0.  0.  7.  6.  1. 12.  3.  0. 12. 15.  2.]
 [ 0.  4. 13.  0.  7.  2.  2.  0.  8.  2. 21.  1.  0.  1.  5.  4.]
 [ 0. 11.  0.  6.  1.  0.  0.  6.  6.  0.  8.  1.  0.  0.  0.  0.]
```

```
 [ 0.  1.  0.  5.  2.  1.  0.  0.  0.  1.  2.  2.  0.  0.  0.  0.]]
INFO:__main__:Mean counts to fit: [0.54545455 1.72727273 3.81818182 1.36363636
3.09090909 1.27272727
 1.45454545 1.27272727 1.81818182 1.36363636 3.90909091 1.36363636
 4.09090909 1.72727273 3.09090909 0.63636364]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [0.7100489913625485, 1.0, 1.0,
4.71238898038469]
INFO:__main__:Neuron 11: R-squared = 0.266
INFO:__main__:Optimized parameters (p_opt): [1.14125477 0.36398983 0.09853366
4.39490938]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 4.  0.  0.  9.  0.  4.  0.  1.  0.  0.  0.
9.  7.  0.  1.  0.]
 [ 0.  0.  0.  9.  3.  0.  0.  1.  1.  0.  0.  4.  1.  0.  5.  5.]
 [ 0.  0.  1.  9. 15.  0.  0.  3.  2.  0.  0.  8.  1.  0.  6.  3.]
 [ 3.  0.  1. 15.  4.  4.  4.  2.  1.  0.  1.  8.  0.  0.  1.  3.]
 [ 3.  0.  0. 21.  3.  5.  1.  5.  1.  0.  0.  7.  0.  0.  4.  1.]
 [ 0.  0.  0.  8.  2.  1.  6.  4.  6.  0.  0.  9.  0.  2.  0.  1.]
 [ 2.  0.  0.  7.  3.  2.  0.  1.  0.  0.  0.  7.  0.  2.  0.  3.]
 [ 2.  0.  0.  8.  2.  1.  2.  1.  0.  0.  0. 11.  0.  1.  0.  1.]
 [ 1.  0.  0.  9.  4.  1.  2.  2.  0.  1.  0.  6.  0.  2.  0.  1.]
 [ 1.  1.  0. 10.  0.  0.  1.  3.  1.  1.  0.  7.  5.  3.  2.  0.]
 [ 2.  1.  0. 10.  4.  0.  0.  0.  2.  1.  0.  8.  2.  1.  6.  0.]]
INFO:__main__:Mean counts to fit: [ 1.63636364  0.18181818  0.18181818
10.45454545  3.63636364  1.63636364
  1.45454545  2.09090909  1.27272727  0.27272727  0.09090909  7.63636364
  1.45454545  1.          2.27272727  1.63636364]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [0.835869164563581, 1.0, 1.0,
1.1780972450961724]
INFO:__main__:Neuron 12: R-squared = 0.806
INFO:__main__:Optimized parameters (p_opt): [2.55500547 8.75758898 0.17955287
1.28329267]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 4.  0.  0.  0.  4. 10.  7. 22. 34. 13.  7.
```

```
0.  6. 33.  8.  7.]
 [ 3.  0.  0.  0.  2.  4.  2. 29. 62. 11.  4. 15.  1. 39.  2.  3.]
 [ 6.  2.  7.  5. 18.  2.  2. 33. 59. 18.  2.  3.  7.  9.  1.  6.]
 [ 3.  2. 14.  2. 13.  7.  3. 28. 63.  8. 11.  2.  1.  3.  1.  3.]
 [ 3.  1.  1.  2.  9. 20.  3. 21. 56.  8. 15.  3. 16. 22.  3.  2.]
 [ 4.  4.  1. 10.  1. 27.  3. 23. 66.  2.  7.  9. 15.  4.  4.  3.]
 [ 1.  1.  2.  5.  5.  4.  2. 39. 35.  8.  3.  2.  9. 14. 13.  3.]
 [10. 11.  9. 11.  6. 18.  6. 38. 28. 13.  2.  7.  5.  5.  1.  1.]
 [ 4.  1. 11.  2.  4.  0.  5. 36. 43.  4.  7.  5.  9. 18.  1.  2.]
 [ 4.  1.  3.  3.  3. 20.  2. 29. 63.  9.  3.  0. 62. 13.  1.  4.]
 [ 3.  2.  6. 17. 11.  6.  2. 32. 57. 15.  1. 11.  4.  3.  7.  0.]]
INFO:__main__:Mean counts to fit: [ 4.09090909  2.27272727  4.90909091
5.18181818  6.90909091 10.72727273
  3.36363636 30.         51.45454545  9.90909091  5.63636364  5.18181818
 12.27272727 14.81818182  3.81818182  3.09090909]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.3843745260025235, 1.0, 1.0,
3.141592653589793]
INFO:__main__:Neuron 13: R-squared = 0.741
INFO:__main__:Optimized parameters (p_opt): [3.98478291 2.99343621 1.17192119
3.04379669]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 2.  1.  2.  0.  4.  0.  0.  0.  0.  1.  9.
4. 24.  1.  1.  0.]
 [ 0.  1.  3.  1.  0.  0.  0.  0.  0.  7.  4.  3.  7.  4.  0.  0.]
 [ 0.  3.  3.  3.  0.  1.  0.  0.  6.  1.  3.  1. 14.  2.  0.  0.]
 [ 0.  0. 12.  4.  1.  4.  0.  0.  0.  1.  9.  2. 30.  7.  3.  0.]
 [ 0.  0. 12.  5.  4.  6.  9.  0.  0.  3.  3.  4. 10.  6.  2.  2.]
 [ 0.  3.  6.  7.  3.  3.  0.  1.  0.  7.  3.  2.  0.  3.  0.  2.]
 [ 0.  1. 21. 20. 15.  5.  3.  0.  0.  1.  5.  8.  0. 12.  3.  1.]
 [ 0.  4.  6.  6.  4.  8.  1.  0.  0.  9.  7.  6.  0.  3.  3.  1.]
 [ 2.  5. 10.  4.  1.  6.  2.  0.  0.  1.  0.  1. 12.  1.  0.  1.]
 [ 2.  1.  6.  5.  7.  7.  5.  0.  1.  0.  4.  4.  8.  4.  0.  0.]
 [ 2.  1. 10.  6.  3.  8.  2.  0.  0.  0.  1.  8.  6.  1.  2.  0.]]
INFO:__main__:Mean counts to fit: [ 0.72727273  1.81818182  8.27272727
5.54545455  3.81818182  4.36363636
  2.          0.09090909  0.63636364  2.81818182  4.36363636  3.90909091
 10.09090909  4.          1.27272727  0.63636364]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
```

```
INFO:__main__:Initial parameter guesses (p0): [1.22310675891248, 1.0, 1.0,
4.71238898038469]
INFO:__main__:Neuron 14: R-squared = 0.595
INFO:__main__:Optimized parameters (p_opt): [1.97382927 0.83415982 0.05913131
4.47410656]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 8.  2.  1.  0. 10. 20. 23.  9. 20.  9. 11.
0.  4. 30. 14. 17.]
 [ 6. 19. 23.  6.  7.  8. 39. 17. 32.  3. 14.  0.  6. 28. 13. 14.]
 [ 3.  5. 44.  5.  4. 48. 36. 24. 18.  3. 29. 43.  1. 20. 31.  2.]
 [ 1.  4. 15.  9.  6. 14. 11. 17. 15. 17.  9. 10.  1. 28. 20. 15.]
 [13.  6. 25. 15.  7.  6. 21. 13. 15. 19. 25.  0. 20. 39. 40.  3.]
 [14.  9.  3. 10.  9.  6. 46. 19. 10. 11. 25. 10. 14. 20. 12. 20.]
 [ 9. 22. 15. 25.  2. 19. 28. 14. 24. 39. 35.  4.  6. 27. 31. 15.]
 [21. 17. 11.  6.  6. 21. 30. 21. 24.  4. 13. 15.  7. 16. 30. 20.]
 [29. 10. 27.  6.  4.  9. 48. 28. 31. 16.  1.  3. 22. 12. 20.  8.]
 [ 7. 20. 27.  6. 13. 29. 31. 23.  6. 17. 16.  5.  7. 34. 31.  8.]
 [ 7. 33.  3.  6.  5. 17. 22. 20. 15. 34.  5.  4.  6. 55.  9. 21.]]
INFO:__main__:Mean counts to fit: [10.72727273 13.36363636 17.63636364
8.54545455  6.63636364 17.90909091
 30.45454545 18.63636364 19.09090909 15.63636364 16.63636364  8.54545455
  8.54545455 28.09090909 22.81818182 13.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.7736534960759607, 1.0, 1.0,
2.356194490192345]
INFO:__main__:Neuron 15: R-squared = 0.379
INFO:__main__:Optimized parameters (p_opt): [3.18508046 0.38039439 0.07213656
2.44967559]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 2.  1.  0.  0.  1. 28.  2.  2.  2.  2.  0.
8.  9. 18.  6.  2.]
 [ 0.  1.  0.  0.  6.  4.  7.  2.  2.  1.  0.  6.  7.  1. 11.  0.]
 [ 0.  1.  0.  0.  5. 11.  7. 19. 12.  0.  0.  2.  4.  4.  2.  1.]
 [ 1.  1.  0.  2.  6.  3.  5.  3.  2.  1.  0.  1.  5. 15.  2.  1.]
 [ 5.  0.  0.  4.  3.  7. 10.  1.  0.  2.  2.  1.  2.  6.  6.  5.]
 [ 0.  0.  1.  1.  2. 15.  3.  3.  0.  2.  1.  2.  1.  9.  4.  1.]
 [ 4.  0.  7.  1.  5.  3.  5.  0.  2.  1.  2.  2.  1. 45.  6.  1.]
 [ 3.  0.  1.  2.  4. 21.  4.  1.  2.  0.  2.  0. 18. 42.  2.  4.]
 [ 2.  0.  4.  4.  4. 59.  7.  3.  4.  0.  1.  0.  3.  3.  4.  0.]
```

```
 [ 2.  3.  2.  3.  9.  9.  1.  1.  2.  0.  2.  0.  1. 54.  2.  0.]
 [ 5.  1.  0.  0.  1. 17.  4.  1.  3.  0.  1.  1.  2. 28.  1.  0.]]
INFO:__main__:Mean counts to fit: [ 2.18181818  0.72727273  1.36363636
1.54545455  4.18181818 16.09090909
  5.          3.27272727  2.81818182  0.81818182  1.          2.09090909
  4.81818182 20.45454545  4.18181818  1.36363636]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [1.502813972729502, 1.0, 1.0,
5.105088062083414]
INFO:__main__:Neuron 16: R-squared = 0.926
INFO:__main__:Optimized parameters (p_opt): [3.00210939 4.5750625  0.10633022
5.10570081]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 5.  9. 11. 12.  7. 15. 45. 21. 44. 19. 13.
16. 13. 21. 13. 44.]
 [11. 15. 27.  5.  8. 31. 15. 25. 36. 27. 12. 23.  8. 31. 30. 27.]
 [14.  8. 17.  6. 11. 28. 18. 47. 27. 16. 13. 10.  8. 24. 42. 28.]
 [21. 22. 20.  1.  9. 15. 30. 17. 35. 17. 19. 15. 20. 30. 28. 29.]
 [36. 29.  7. 10.  4. 28. 34. 30. 42. 26. 20.  7. 14. 25. 21. 33.]
 [27. 19.  4.  7.  8. 29. 34. 24. 17. 38. 14.  8. 23. 20. 41. 21.]
 [29. 21.  7. 14.  6. 32. 29. 20. 50. 16. 29. 11. 20. 12. 12. 33.]
 [22. 11. 11.  3.  7. 24. 25. 27. 37. 15. 15.  9. 10. 15. 26. 43.]
 [16. 16. 11.  3.  7.  7. 37. 33. 30. 19. 14. 10. 20. 21. 35. 22.]
 [26.  9.  7.  2.  4. 22. 14. 16. 11. 21. 14. 11. 11. 31. 24. 22.]
 [23. 11. 14.  8.  4. 25. 18. 21. 11. 24. 11.  7.  3. 42. 23. 43.]]
INFO:__main__:Mean counts to fit: [20.90909091 15.45454545 12.36363636
6.45454545  6.81818182 23.27272727
 27.18181818 25.54545455 30.90909091 21.63636364 15.81818182 11.54545455
 13.63636364 24.72727273 26.81818182 31.36363636]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.978250813899017, 1.0, 1.0,
5.8904862254808625]
INFO:__main__:Neuron 17: R-squared = 0.778
INFO:__main__:Optimized parameters (p_opt): [3.41347446e+00 4.91256948e-01
2.30711219e-15 5.83069626e+00]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
```

```
INFO:__main__:Spike count matrix: [[ 7.  0.  2.  0.  1. 13.  3.  0.  3.  1.  0.
0. 10.  2. 11. 10.]
 [ 8.  2.  0.  0.  0.  3.  4.  3.  1.  1. 11.  1.  5. 10. 11. 10.]
 [ 6.  3.  4.  0.  4.  5. 19.  4.  1.  0.  7.  0.  1.  4.  1. 11.]
 [ 6.  1.  9.  2.  2. 12.  5.  4.  7.  2.  0.  5.  1. 10.  8. 10.]
 [ 1.  4.  1.  1.  4.  1.  3.  1.  5.  2.  4.  2.  1. 11.  8.  3.]
 [ 4.  2.  1.  4.  1.  9.  5.  6.  1.  0.  5.  1.  4. 11.  6.  4.]
 [ 4.  9.  1.  2.  2. 19.  2.  1.  3.  0.  6.  2.  3.  4.  3.  1.]
 [ 3.  8.  1. 11.  8.  4.  6.  3.  2.  8.  2.  1.  1.  5. 26.  3.]
 [12.  6.  2.  2.  5.  4. 10.  5.  2.  3.  7.  1.  1.  4.  3.  7.]
 [ 3.  4. 17.  3.  3.  9.  2.  6.  3.  2.  9.  2.  3.  6.  7.  1.]
 [ 7.  5.  2.  1.  1.  8.  2.  5.  6.  1.  1. 16.  6. 12.  3.  0.]]
INFO:__main__:Mean counts to fit: [5.54545455 4.         3.63636364 2.36363636
2.81818182 7.90909091
 5.54545455 3.45454545 3.09090909 1.81818182 4.72727273 2.81818182
 3.27272727 7.18181818 7.90909091 5.45454545]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [1.4977442533792515, 1.0, 1.0,
1.9634954084936207]
INFO:__main__:Neuron 18: R-squared = 0.511
INFO:__main__:Optimized parameters (p_opt): [1.90044272e+00 4.60238797e-01
1.53559830e-16 2.26945131e+00]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[17. 15.  7.  2.  9. 19. 21. 11. 20.  4.  1.
25. 15.  6.  9. 15.]
 [12. 13. 21.  0.  3. 11.  6. 16. 10.  4. 14. 11.  8.  5. 10.  6.]
 [14.  7. 14.  9. 17. 23. 17.  8. 16.  2.  5.  4. 10. 10. 17. 11.]
 [10. 13.  4.  4. 13.  7. 10.  8. 10. 11.  4.  8. 25. 20.  9. 28.]
 [12. 20.  7.  8.  9. 17. 17.  9.  5.  4.  6. 14. 11. 18. 17.  8.]
 [12. 17.  3.  0.  8. 17. 14. 14.  9.  7.  5.  2. 11.  2. 13.  8.]
 [16. 16. 13.  3.  1. 13. 27. 12. 12.  7.  9. 13. 18. 26. 15.  8.]
 [22. 13. 10. 14.  8. 15. 17.  5. 28.  3. 10.  6. 41.  9. 21. 11.]
 [29.  9. 21. 13.  5.  6. 21. 14. 23.  5. 11. 12. 15.  8.  6. 19.]
 [12. 15. 18. 18. 20. 15.  9. 13.  8.  8. 21. 15.  9. 10. 10. 19.]
 [11. 12. 24. 11. 10.  2. 16.  9. 11.  3. 14.  8. 17.  2. 11.  9.]]
INFO:__main__:Mean counts to fit: [15.18181818 13.63636364 12.90909091
7.45454545  9.36363636 13.18181818
 15.90909091 10.81818182 13.81818182  5.27272727  9.09090909 10.72727273
 16.36363636 10.54545455 12.54545455 12.90909091]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
```

```
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.4729989120390488, 1.0, 1.0,
4.71238898038469]
INFO:__main__:Neuron 19: R-squared = 0.211
INFO:__main__:Optimized parameters (p_opt): [2.66883308 0.13949486 0.06166185
5.69028405]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 1.  0.  0.  0.  1.  2.  0.  0.  0.  1.  0.
4.  0.  0.  2.  0.]
 [ 0.  0.  0.  0.  3.  3.  0.  0.  0.  0.  0.  4.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  2.  6.  1.  0.  0.  1.  8.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  2. 10.  2.  0.  2.  0.  1. 12.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  0.  2.  1.  0.  0.  2.  9.  0.  2.  0.  0.]
 [ 0.  0.  0.  0.  2.  1.  0.  0.  0.  0.  1. 15.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  3.  5.  0.  0.  0.  0.  1. 15.  0.  2.  0.  0.]
 [ 0.  0.  0.  0.  0.  2.  0.  0.  0.  0.  1. 11.  0.  1.  0.  1.]
 [ 0.  0.  0.  0.  0. 11.  0.  0.  0.  0.  2. 11.  1.  1.  0.  2.]
 [ 0.  0.  0.  0.  0. 10.  0.  0.  0.  0.  4. 14.  1.  2.  0.  0.]
 [ 0.  0.  0.  0.  1.  1.  0.  0.  0.  0.  3. 10.  5.  1.  0.  0.]]
INFO:__main__:Mean counts to fit: [ 0.09090909  0.          0.          0.
1.27272727  4.27272727
  0.90909091  0.18181818  0.18181818  0.09090909  1.45454545 10.27272727
  0.63636364  0.81818182  0.18181818  0.27272727]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [0.46210538722149547, 1.0, 1.0,
4.319689898685966]
INFO:__main__:Neuron 20: R-squared = 0.791
INFO:__main__:Optimized parameters (p_opt): [2.34850484 7.44105887 2.26932096
4.2850757 ]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 26.  25.  46. 120.  85.  33.  40.  48.  19.
23.  32.  94. 146.  38.
   18.  25.]
 [ 15.   6.  27. 100. 105.  52.  40.  32.  23.  22.   6.  82. 135.  94.
   33.  12.]
 [  9.  16.  27.  68.  48.  22.  29.  43.  41.  11.  13.  80. 103.  61.
   27.  15.]
 [ 21.  20.  19.  66. 145.  41.  33.  32.  15.   6.  20.  69.  89.  44.
   24.  13.]
```

```
 [ 25.  27.  10.  83.  86.  96.  29.  27.  13.  16.  50. 105.  55.  21.
   63.  26.]
 [ 35.  23.  14.  41. 127.  53.  32.  13.  12.  15.   6.  63. 119.  81.
   24.  25.]
 [ 36.  28.  24. 104.  57.  55.  32.  17.  13.  11.  23.  76.  69.  33.
   26.  13.]
 [ 14.  26.   7.  49.  48.  87.  64.  21.  16.  28.  35.  64.  95.  64.
   21.  36.]
 [ 20.  11.  35.  41. 120.  42.  24.  51.  10.   7.  20.  74. 116.  29.
   40.  31.]
 [ 44.   4.  23. 124.  85.  40.  32.  20.   4.  14.  20.  54. 112.  60.
   34.  12.]
 [  8.  31.  32. 120.  88.  41.  51.   8.  17.   7.  31.  90. 168.  49.
   30.  63.]]
INFO:__main__:Mean counts to fit: [ 23.          19.72727273  24.
83.27272727  90.36363636
  51.09090909  36.90909091  28.36363636  16.63636364  14.54545455
  23.27272727  77.36363636 109.72727273  52.18181818  30.90909091
  24.63636364]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [3.787026515253461, 1.0, 1.0,
4.71238898038469]
INFO:__main__:Neuron 21: R-squared = 0.892
INFO:__main__:Optimized parameters (p_opt): [4.59161637 1.06601956 0.03431355
4.65379577]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[17. 30. 29. 17. 41. 29. 34. 20.  8. 11.  2.
13. 34. 22. 20. 16.]
 [13.  8.  7. 26. 32. 36. 32. 11. 16. 19. 16.  4. 27. 21. 27.  2.]
 [12. 16.  9. 15. 30. 37. 17. 15. 24.  4. 12. 13. 29. 28. 15. 31.]
 [11. 21. 33.  8. 31. 16. 33. 18. 17. 11. 17. 13. 27. 33. 25.  9.]
 [21. 15. 11. 13. 28. 23. 30.  9. 11.  8. 13. 18. 10. 16.  8.  9.]
 [22. 14.  5. 28. 35. 32. 47. 18.  5.  7.  6.  9. 27. 32. 15. 10.]
 [14. 21. 15. 17. 30. 37. 38. 16. 17. 16.  7. 11. 20. 24. 27. 16.]
 [13. 33.  8. 11. 39. 30. 34. 19. 16. 14.  9. 11. 25. 38. 31. 13.]
 [ 6. 21. 10. 16. 36. 32. 13. 17. 12. 14. 15. 14. 31. 46. 31.  9.]
 [12. 16. 12. 20. 25. 39. 39. 16.  3. 20. 30.  9. 21. 29. 52. 14.]
 [ 7. 16.  5. 23. 29. 29. 25.  9. 33. 14. 14. 17. 31. 28. 32. 31.]]
INFO:__main__:Mean counts to fit: [13.45454545 19.18181818 13.09090909
17.63636364 32.36363636 30.90909091
 31.09090909 15.27272727 14.72727273 12.54545455 12.81818182 12.
 25.63636364 28.81818182 25.72727273 14.54545455]
```

```
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.9951639302593516, 1.0, 1.0,
1.5707963267948966]
INFO:__main__:Neuron 22: R-squared = 0.821
INFO:__main__:Optimized parameters (p_opt): [3.50804692 0.48808919 0.08940405
1.95805888]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 0.  0.  1.  1.  3.  6.  8.  0.  1.  0.  0.
1.  0.  0.  0.  0.]
 [ 0.  0.  0.  2.  1.  1. 11.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  2. 11.  1.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  4.  8.  3.  1.  0.  0.  0.  1.  4.  2.  0.  0.]
 [ 0.  0.  0.  1.  3.  0.  2.  1.  0.  0.  0.  1.  7.  7.  0.  0.]
 [ 0.  0.  0.  2.  4.  0.  1.  1.  0.  0.  0.  0.  2.  5.  0.  0.]
 [ 0.  0.  0.  3.  7.  0.  0.  1.  0.  0.  0.  0.  1.  9.  0.  0.]
 [ 0.  2.  0.  0.  1.  0.  0.  1.  0.  0.  0.  0.  1.  1.  4.  2.]
 [ 0.  0.  0.  0.  0.  5.  0.  0.  0.  0.  0.  0.  4.  2.  2.  2.]
 [ 0.  1.  0.  0.  3.  3.  0.  8.  1.  0.  0.  1.  2.  3.  1.  0.]
 [ 0.  2.  0.  0.  5. 10.  0.  1.  0.  0.  0.  0.  1.  3.  0.  0.]]
INFO:__main__:Mean counts to fit: [0.         0.45454545 0.09090909 1.
3.         4.
 2.36363636 1.27272727 0.18181818 0.         0.         0.54545455
 2.         2.90909091 0.63636364 0.36363636]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [0.3698741630054618, 1.0, 1.0,
1.9634954084936207]
INFO:__main__:Neuron 23: R-squared = 0.940
INFO:__main__:Optimized parameters (p_opt): [1.4412205  1.62891924 0.24237924
1.89367802]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[28. 16. 10. 33.  7. 22. 23. 50. 29. 29. 23.
38. 16. 20. 14. 31.]
 [26. 19. 35. 26. 14. 40. 35. 48. 37. 28. 22. 23. 14. 21. 13. 25.]
 [ 8. 17. 15. 13. 15. 23. 34. 27. 49. 21. 10. 25.  6. 12. 17. 22.]
 [24. 21. 13. 24. 22. 12. 26. 30. 29. 17. 19. 15.  8. 16.  9. 19.]
 [28. 42. 12.  6. 10. 33. 33. 29. 38. 32. 25. 23.  9. 12. 22. 18.]
```

```
 [39. 12. 21. 12. 18. 24. 28. 21. 55. 31. 20. 17. 20. 18. 25. 26.]
 [51. 22.  6. 14. 11. 12. 24. 37. 44. 18. 24. 22. 20. 11. 20. 20.]
 [18. 16. 11. 11.  6. 13. 30. 34. 25. 11. 21. 19. 16.  9.  4. 24.]
 [35. 16. 12. 12. 15. 12. 23. 42. 23.  8. 18. 18. 10.  6. 11. 26.]
 [46. 18.  7. 18.  7.  9. 28. 27. 22. 16. 14. 13.  3.  8. 13. 19.]
 [24. 22. 18. 21. 10. 15. 27. 27. 23.  6.  8. 17.  4. 21. 22. 31.]]
INFO:__main__:Mean counts to fit: [29.72727273 20.09090909 14.54545455
17.27272727 12.27272727 19.54545455
 28.27272727 33.81818182 34.         19.72727273 18.54545455 20.90909091
 11.45454545 14.         15.45454545 23.72727273]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [3.036645173033174, 1.0, 1.0,
3.141592653589793]
INFO:__main__:Neuron 24: R-squared = 0.782
INFO:__main__:Optimized parameters (p_opt): [3.52852833 0.38797273 0.14932384
2.93634046]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[24. 19. 26.  1. 17.  8. 39. 16. 51. 28. 17.
10.  6. 39. 27. 13.]
 [19.  6. 34.  4.  4. 37. 11. 31. 23. 13. 22. 56. 10. 32. 15. 38.]
 [21. 15. 31.  7.  1. 14. 36. 49. 18. 12. 17.  6. 13. 29. 24. 23.]
 [10. 10. 13. 18.  8. 20. 26. 33. 35. 10. 24. 12.  7. 17. 17. 20.]
 [16. 15. 25.  9.  4. 16. 30. 48. 14. 13. 12. 11.  9.  5. 12. 22.]
 [28. 27. 16. 21.  7. 13. 18. 44. 29. 27.  8. 18.  0. 36.  8. 17.]
 [ 8. 21. 20. 14.  2. 27. 29. 22. 31. 23. 34.  6.  1. 21. 25. 16.]
 [57. 14. 16. 15.  5. 37. 43. 37. 50. 37. 16. 22.  9. 21. 37. 22.]
 [34. 24. 40. 11.  3. 10. 25. 41. 51. 29. 30. 19. 12. 28. 23. 36.]
 [21. 34. 27. 10. 18. 53. 40. 44. 40. 37. 21. 15. 49. 11. 28. 21.]
 [24. 14. 27. 23.  9. 16. 43. 33. 49. 28. 53. 16.  5. 28. 26. 19.]]
INFO:__main__:Mean counts to fit: [23.81818182 18.09090909 25.
12.09090909  7.09090909 22.81818182
 30.90909091 36.18181818 35.54545455 23.36363636 23.09090909 17.36363636
 11.         24.27272727 22.         22.45454545]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [3.0997851163984724, 1.0, 1.0,
2.748893571891069]
INFO:__main__:Neuron 25: R-squared = 0.712
INFO:__main__:Optimized parameters (p_opt): [3.59394213 0.35462887 0.18019148
2.89127112]
```

```
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[20. 10. 15. 14. 19. 20. 10.  0.  0.  9. 10.
18. 11.  9.  0.  6.]
 [ 6.  4.  0.  6. 14. 22. 16. 11.  5. 12. 10. 14.  1. 12. 22.  6.]
 [ 1. 12.  7.  2.  2. 28. 13. 21. 10.  9.  5. 17.  2. 32.  8. 10.]
 [ 5. 14. 20.  8. 14. 13. 11.  2.  6.  4. 16.  4. 22. 12. 13.  2.]
 [ 7.  2.  3.  4.  5. 24. 21. 17.  1. 13.  6.  5. 17.  2.  3.  9.]
 [13.  8.  7. 10. 19. 22. 12.  3.  1. 14. 13.  2.  5. 16. 14. 24.]
 [12. 12.  4. 14.  5. 15. 19.  1.  2.  9.  7. 13.  7. 13. 13.  9.]
 [14. 11.  8. 13. 16. 19. 17.  6. 11.  1.  9. 12.  5. 29. 14.  3.]
 [16.  4.  2.  3. 14. 16. 18. 15. 16.  8. 12.  6.  5.  6. 26.  6.]
 [18.  2.  8.  4.  1. 25. 25. 14.  3.  0. 24. 11.  3. 16. 17. 16.]
 [ 0. 10. 13. 16.  7. 12.  5. 13.  6. 15.  6.  4.  3. 24. 14.  0.]]
INFO:__main__:Mean counts to fit: [10.18181818  8.09090909  7.90909091
8.54545455 10.54545455 19.63636364
 15.18181818  9.36363636  5.54545455  8.54545455 10.72727273  9.63636364
  7.36363636 15.54545455 13.09090909  8.27272727]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.352456923034219, 1.0, 1.0,
1.9634954084936207]
INFO:__main__:Neuron 26: R-squared = 0.569
INFO:__main__:Optimized parameters (p_opt): [2.76783732 0.37766858 0.07983682
2.06495944]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 0. 27. 19.  1.  0.  0.  0.  0.  0.  1.  2.
0.  0.  0.  0.  0.]
 [ 0. 13. 23.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0.  7. 40.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  9. 33.  0.  0.  0.  0.  0.  0.  0.  2.  0.  0.  0.  0.  0.]
 [ 0. 15.  8.  0.  0.  0.  0.  0.  0.  0.  2.  0.  0.  0.  0.  0.]
 [ 0. 12. 25.  4.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0. 11.  1.  3.  0.  0.  0.  0.  0.  0.  5.  0.  0.  0.  0.  0.]
 [ 0.  8. 33.  2.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0. 10. 20.  5.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 15.  7.  3.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0.  7. 11.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]]
INFO:__main__:Mean counts to fit: [ 0.         12.18181818 20.
1.63636364  0.09090909  0.
  0.          0.          0.          0.09090909  1.27272727  0.09090909
```

```
  0.          0.          0.          0.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [1.6197739217647624, 1.0, 1.0,
0.7853981633974483]
INFO:__main__:Neuron 27: R-squared = 0.999
INFO:__main__:Optimized parameters (p_opt): [3.19840347 5.06790213 1.54450164
0.64932497]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 0. 16. 14.  0.  0.  0.  0.  0.  0. 32. 31.
0.  0.  0.  0.  0.]
 [ 0. 10. 13.  0.  0.  0.  0.  0.  0. 44. 19.  0.  0.  0.  0.  0.]
 [ 0. 20. 14.  0.  0.  0.  0.  0.  0. 41. 42.  0.  0.  0.  0.  0.]
 [ 0. 10. 16.  0.  0.  0.  0.  0.  0. 54. 32.  0.  0.  0.  0.  0.]
 [ 0. 24. 16.  0.  0.  0.  0.  0.  0. 43. 55.  0.  0.  0.  0.  0.]
 [ 0.  9. 13.  0.  0.  0.  0.  0.  0. 52. 41.  0.  0.  0.  0.  0.]
 [ 0. 19. 15.  0.  0.  0.  0.  0.  0. 44. 30.  3.  0.  0.  0.  0.]
 [ 0. 15.  3.  0.  0.  0.  0.  0.  0. 48. 47.  1.  0.  0.  0.  0.]
 [ 0. 25. 24.  0.  0.  0.  0.  0.  3. 45. 26.  1.  0.  0.  0.  0.]
 [ 0. 11. 29.  0.  1.  0.  0.  0.  1. 37. 40.  2.  0.  0.  0.  0.]
 [ 0. 14. 17.  0.  0.  0.  0.  0.  0. 48. 39.  2.  0.  0.  1.  0.]]
INFO:__main__:Mean counts to fit: [ 0.         15.72727273 15.81818182  0.
0.09090909  0.
  0.          0.          0.36363636 44.36363636 36.54545455  0.81818182
  0.          0.          0.09090909  0.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.6551607371818373, 1.0, 1.0,
3.5342917352885173]
INFO:__main__:Neuron 28: R-squared = 0.999
INFO:__main__:Optimized parameters (p_opt): [4.33282731 8.18559981 0.48435863
3.71739285]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 5.  2.  4. 13.  2.  0.  0.  9.  0.  1. 24.
0.  2.  0.  1.  1.]
 [ 0.  0. 19. 20.  8.  3.  0.  3.  2.  0.  9.  8.  4.  0.  0.  2.]
 [ 0.  0.  9. 19. 13.  1.  0.  4.  0.  0.  6.  0. 15.  0.  0.  6.]
 [ 0.  0.  1. 16.  3.  1.  5.  2.  0.  1.  7. 11.  7.  0.  0.  6.]
```

```
 [ 0.  0. 28. 12.  3.  1. 10.  1.  0.  0.  1.  7.  6.  0.  0.  1.]
 [ 0.  0. 14. 25.  7.  4.  1.  8.  0.  0.  1.  4.  5.  0.  0.  0.]
 [ 0.  0. 15. 12.  9.  2.  1.  3.  3.  0. 11.  3.  7.  0.  0.  2.]
 [ 6.  4. 19. 27.  5.  2.  4.  5.  4.  0. 16.  6.  6.  1.  0.  0.]
 [ 3.  2. 14. 23.  5.  2.  3.  2.  4.  0. 16.  4.  5.  1.  0.  0.]
 [ 2.  1. 10. 19. 11.  2.  5.  1.  3.  8.  5.  7.  3.  0.  0.  4.]
 [ 1.  1. 26. 24.  3.  2.  4.  3.  3.  3.  6.  1. 20.  1.  1.  0.]]
INFO:__main__:Mean counts to fit: [ 1.54545455  0.90909091 14.45454545
19.09090909  6.27272727  1.81818182
  3.          3.72727273  1.72727273  1.18181818  9.27272727  4.63636364
  7.27272727  0.27272727  0.18181818  2.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [1.5759281335352222, 1.0, 1.0,
1.1780972450961724]
INFO:__main__:Neuron 29: R-squared = 0.841
INFO:__main__:Optimized parameters (p_opt): [2.96043674 1.90155694 0.41176043
1.06863547]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 65.  35.  45.  70.  89.  52. 104.  95.  81.
25.  64.  63.  60.  80.
   71.  91.]
 [ 80.  40.  55.  55.  72.  85.  84. 102.  65.  30.  57.  52.  83. 100.
   82. 139.]
 [ 45.  34.  54.  77.  76.  62. 136. 127.  56.  31.  46.  46.  68.  89.
   98. 109.]
 [ 39.  32.  53.  79.  70.  43.  79.  89.  74.  21.  74.  67.  73.  95.
   70. 103.]
 [ 58.  42.  56.  56.  81.  74.  90. 114.  68.  36.  44.  54.  67.  72.
   89. 101.]
 [ 66.  25.  46.  63.  65.  75.  93.  84.  46.  25.  69.  54.  59.  81.
   87. 127.]
 [ 62.  33.  50.  61.  50.  65.  97. 102.  48.  38.  40.  38.  68.  57.
   82. 115.]
 [ 88.  29.  50.  57.  66.  72.  96. 107.  64.  43.  49.  51.  59.  68.
   64.  92.]
 [ 75.  38.  56.  40.  67.  45. 121. 149.  77.  32.  77.  52.  65.  63.
   85. 100.]
 [ 69.  49.  72.  52.  64.  60.  95. 114.  58.  35.  55.  54.  58.  81.
   85. 109.]
 [ 58.  35.  44.  53.  73.  68. 109. 103.  65.  40.  45.  59.  92.  65.
   71.  94.]]
INFO:__main__:Mean counts to fit: [ 64.09090909  35.63636364  52.81818182
```

```
60.27272727  70.27272727
  63.72727273 100.36363636 107.81818182  63.81818182  32.36363636
  56.36363636  53.63636364  68.36363636  77.36363636  80.36363636
 107.27272727]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [4.22550572382466, 1.0, 1.0,
2.748893571891069]
INFO:__main__:Neuron 30: R-squared = 0.665
INFO:__main__:Optimized parameters (p_opt): [4.58295291 0.37397861 0.01945008
2.43326854]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 48.  77.  80.  17.  29.  20.  29.  25.  38.
45.  61.  53.  47.  25.
   19.  41.]
 [ 70.  98.  91.  12.  16.  43.  21.  38.  40.  67.  57. 118.  53.  53.
   35.  40.]
 [ 39. 100.  65.  25.  15.  29.  25.  27.  61.  29.  47.  43.  26.  64.
   38.  33.]
 [ 39. 101.  26.  10.  25.  31.  27.  31.  18.  39.  70.  61.  36.  50.
   38.  41.]
 [ 47.  84.  75.  24.  33.  34.  24.  44.  21.  42.  90.  33.  45.  26.
   31.  40.]
 [ 46.  94.  33.  33.  19.  34.  20.  24.  14.  60.  34.  57.  56.  49.
   36.  36.]
 [ 53.  87.  46.  39.  13.  44.  23.   9.  31.  25.  84.  35.  53.  27.
   34.  45.]
 [ 68.  72.  57.  23.  16.  60.  27.  26.  45.  63.  62.  63.  44.  38.
   28.  24.]
 [ 77.  72.  81.  22.  10.  20.  27.  50.  37.  39.  35.  61.  43.  35.
   32.  42.]
 [ 65.  91.  69.  19.  23.  34.  23.  31.  28.  23.  73.  93.  56.  45.
   51.  51.]
 [ 39. 105.  52.  20.  28.  21.  45.  24.  38.  35.  98.  64.  56.  43.
   27.  45.]]
INFO:__main__:Mean counts to fit: [53.72727273 89.18181818 61.36363636
22.18181818 20.63636364 33.63636364
 26.45454545 29.90909091 33.72727273 42.45454545 64.63636364 61.90909091
 46.81818182 41.36363636 33.54545455 39.81818182]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
```

```
INFO:__main__:Initial parameter guesses (p0): [3.7804377697591094, 1.0, 1.0,
0.39269908169872414]
INFO:__main__:Neuron 31: R-squared = 0.454
INFO:__main__:Optimized parameters (p_opt): [4.22169665 0.40337152 0.0849954
0.5215131 ]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[22.  6.  2.  1.  0.  0.  4.  2.  6.  0.  0.
1.  1.  2.  1.  2.]
 [50. 20.  0.  2.  5.  0.  7.  2.  3.  0. 10.  1.  1.  0.  2.  4.]
 [38. 12.  0.  1.  1.  0.  0.  3.  5.  3.  2.  6.  4.  0.  2.  2.]
 [60.  8.  1.  2.  2.  2.  2.  6.  3.  4.  6.  1.  5.  2.  6.  0.]
 [48. 21. 10.  1.  2.  3.  8.  3.  8.  4.  4.  1.  1.  3.  5.  3.]
 [19. 10.  0.  2.  5.  3.  2.  4.  2.  4.  4.  2.  0.  3.  8.  4.]
 [33. 12.  1.  0.  3.  5.  5.  5.  5.  1.  2.  2.  4.  1.  4.  2.]
 [32. 26.  1.  0.  1.  6.  3.  2.  9.  4.  1.  0.  0.  4.  6.  3.]
 [42. 40.  2.  0.  5.  2.  3.  4.  1.  2.  0.  0.  3.  4.  2.  1.]
 [32. 16.  3.  3.  8.  1. 11.  0.  8.  3.  3.  0.  6.  1.  5.  3.]
 [30.  8.  1.  2.  6.  5.  3.  0.  0.  1.  8.  0.  2.  3. 13.  0.]]
INFO:__main__:Mean counts to fit: [36.90909091 16.27272727  1.90909091
1.27272727  3.45454545  2.45454545
  4.36363636  2.81818182  4.54545455  2.36363636  3.63636364  1.27272727
  2.45454545  2.09090909  4.90909091  2.18181818]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [1.759032775725498, 1.0, 1.0, 0.0]
INFO:__main__:Neuron 32: R-squared = 0.922
INFO:__main__:Optimized parameters (p_opt): [3.72861588 5.46999002 1.01984578
0.10399382]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[28. 40. 38. 48. 64. 30. 36. 37. 44. 38. 20.
69. 62. 38. 31. 31.]
 [18. 30. 41. 48. 72. 56. 23. 27. 22. 40. 50. 46. 55. 57. 30. 21.]
 [20. 21. 44. 32. 46. 44. 43. 32. 32. 27. 32. 47. 48. 36. 39. 18.]
 [22. 30. 22. 43. 63. 36. 26. 27. 31. 38. 35. 45. 57. 46. 26. 16.]
 [14. 61. 42. 43. 55. 34. 22. 75. 31. 38. 34. 57. 62. 54. 40. 38.]
 [52. 29. 31. 42. 51. 32. 27. 40. 32. 26. 40. 56. 59. 35. 35. 30.]
 [25. 33. 24. 26. 61. 32. 28. 25. 51. 33. 48. 44. 63. 46. 30. 25.]
 [43. 22. 28. 46. 47. 19. 14. 19. 47. 44. 24. 50. 42. 25. 25. 21.]
 [45. 23. 45. 32. 41. 39. 25. 26. 33. 22. 22. 61. 50. 30. 19. 36.]
 [29. 39. 29. 44. 52. 43. 18. 36. 23. 24. 15. 33. 48. 43. 19. 23.]
```

```
 [40. 34. 39. 27. 58. 35. 32. 43. 25. 41. 36. 35. 59. 44. 24. 28.]]
INFO:__main__:Mean counts to fit: [30.54545455 32.90909091 34.81818182
39.18181818 55.45454545 36.36363636
 26.72727273 35.18181818 33.72727273 33.72727273 32.36363636 49.36363636
 55.         41.27272727 28.90909091 26.09090909]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [3.610303473300777, 1.0, 1.0,
1.5707963267948966]
INFO:__main__:Neuron 33: R-squared = 0.643
INFO:__main__:Optimized parameters (p_opt): [3.87198863e+00 2.83077586e-01
6.87706849e-16 1.43885454e+00]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 7.  0.  0.  2.  0.  0.  0.  2. 17.  0.  0.
0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.  1.  0.  0.  8.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.  1.  0.  1.  4.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.  1.  0.  3. 15.  0.  0.  0.  0.  0.  0.  0.]
 [ 5.  0.  0.  0.  0.  3.  0.  0.  3.  0.  0.  0.  0.  0.  0.  2.]
 [ 5.  0.  0.  0.  0.  2.  0.  2.  4.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.  4.  0.  0.  9.  0.  0.  0.  0.  1.  0.  1.]
 [ 2.  0.  0.  0.  0.  0.  0.  4. 10.  0.  0.  0.  0.  1.  0.  0.]
 [ 2.  0.  0.  0.  0.  0.  0.  0.  2.  0.  0.  0.  1.  1.  0.  1.]
 [ 8.  0.  0.  0.  0.  0.  0.  1.  9.  0.  1.  2.  3.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.  0.  1.  0.  1.  0.  1.  0.  0.]]
INFO:__main__:Mean counts to fit: [3.         0.         0.         0.18181818
0.         1.09090909
 0.         1.27272727 7.36363636 0.09090909 0.09090909 0.27272727
 0.36363636 0.45454545 0.         0.36363636]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [0.2793832696370859, 1.0, 1.0,
3.141592653589793]
INFO:__main__:Neuron 34: R-squared = 0.969
INFO:__main__:Optimized parameters (p_opt): [ 2.17960392 11.07758037  0.45549026
3.05136035]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 3.  0.  0.  0.  0.  0.  0.  2.  2.  6.  1.
```

```
2. 12.  2.  2.  1.]
 [ 2.  0.  0.  0.  0.  0.  2.  2.  3.  6.  1.  0.  2.  2.  1.  1.]
 [ 7.  7.  0.  1.  0.  0.  9.  1.  3.  7.  2.  0.  5.  1.  1.  1.]
 [ 3.  7.  2.  2.  3.  0.  1.  1.  7.  5.  1.  0.  3.  6.  0.  1.]
 [ 1.  0.  3.  1.  3.  2.  1.  4.  2.  1.  4.  0.  1.  0.  2.  1.]
 [ 0.  5.  5.  2.  1.  7.  1.  3.  6.  3.  3.  0.  3.  4.  0.  2.]
 [ 4.  2.  1.  1. 13.  1.  2.  2.  2.  4.  3.  6.  0.  8.  2.  1.]
 [ 1.  2.  3.  4.  1.  2.  0.  3.  3.  6.  1.  2.  2.  2.  3.  6.]
 [ 3.  2.  1.  3.  4.  5.  0.  1.  0.  5.  1.  2.  2.  0.  2.  3.]
 [ 2.  1.  2.  2.  2.  1.  0.  1.  2.  6.  2.  1.  5.  0.  1.  0.]
 [ 3.  1.  2.  2.  2.  3.  6.  2.  6.  6.  1.  2.  0.  4.  0.  0.]]
INFO:__main__:Mean counts to fit: [2.63636364 2.45454545 1.72727273 1.63636364
2.63636364 1.90909091
 2.         2.         3.27272727 5.         1.81818182 1.36363636
 3.18181818 2.63636364 1.27272727 1.54545455]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [0.8407831793660098, 1.0, 1.0,
3.5342917352885173]
INFO:__main__:Neuron 35: R-squared = 0.271
INFO:__main__:Optimized parameters (p_opt): [1.22702831 0.21572188 0.19612322
3.42858521]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 88.  80.  49.  36. 109.  71.  71.  75. 130.
50. 102.  77. 126. 145.
   33. 127.]
 [123.  72.  77.  83. 107. 121.  87. 147. 150.  78.  77. 100. 168. 124.
   95. 130.]
 [ 66.  95.  49.  96. 129. 121. 115.  99. 113.  55. 108.  43. 132. 120.
  112.  90.]
 [ 60.  91.  62.  24. 104.  79. 128. 105. 104.  53.  45.  98. 157. 151.
   88. 123.]
 [ 89.  52.  84.  67.  87. 108.  91. 149.  90.  46.  48.  50. 155. 110.
   90. 141.]
 [103.  67.  53.  61. 136.  70. 109. 107. 153.  72.  28.  46. 152. 114.
   96. 101.]
 [ 76.  63.  52.  55. 132. 104. 105. 110. 111.  61.  98.  75. 142. 124.
   62.  92.]
 [125.  48.  36.  31.  92. 119.  98.  95. 129.  70.  59.  41. 167. 137.
   48. 107.]
 [111.  47.  33.  33. 116.  58. 131. 142. 138.  49.  54.  59. 133. 136.
   69. 133.]
 [109.  93.  15.  62.  85.  55. 104. 155. 117.  49.  83. 102. 150. 112.
```

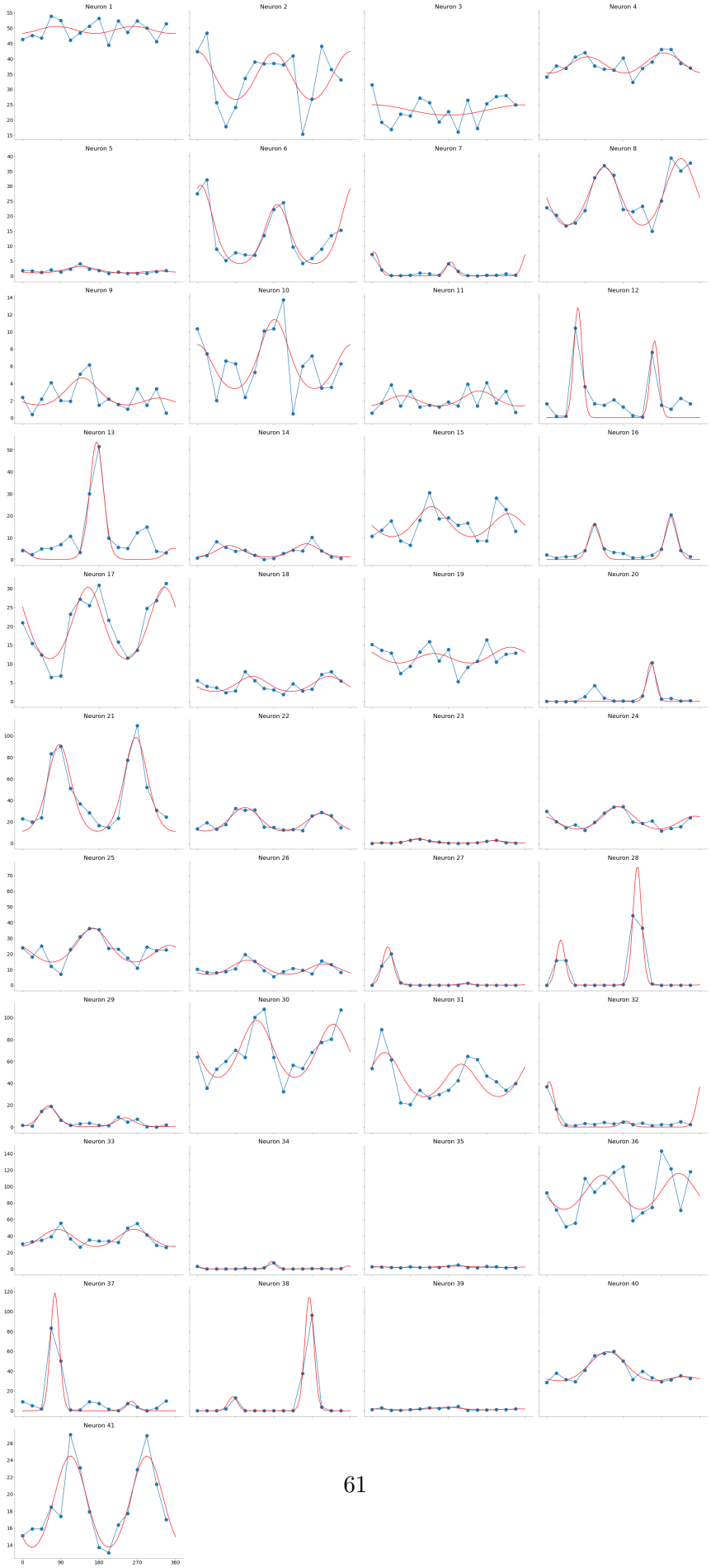
```
   51. 121.]
 [ 69.  81.  52.  64. 111. 119. 107. 104. 131.  60.  49. 129.  93.  66.
   39. 132.]]
INFO:__main__:Mean counts to fit: [ 92.63636364  71.72727273  51.09090909
55.63636364 109.81818182
  93.18181818 104.18181818 117.09090909 124.18181818  58.45454545
  68.27272727  74.54545455 143.18181818 121.72727273  71.18181818
 117.90909091]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [4.523701272372247, 1.0, 1.0,
4.71238898038469]
INFO:__main__:Neuron 36: R-squared = 0.336
INFO:__main__:Optimized parameters (p_opt): [4.75356597 0.23053633 0.01085589
5.42413262]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 10.   4.   1.  82.  66.   0.   0.  15.   5.
0.   0.   3.   6.   2.
    0.  11.]
 [ 11.   9.   0.  84.  62.   0.   0.   8.  12.   0.   0.   9.   4.   0.
    2.   5.]
 [  5.   5.   1.  82.  52.   0.   0.  15.   2.   0.   0.   2.   4.   0.
    1.  10.]
 [  8.   5.   2.  82.  43.   0.   0.   4.   7.   3.   0.  19.   2.   0.
    6.  13.]
 [ 10.   4.   5.  94.  48.   0.   1.   9.   9.   1.   0.   9.   1.   0.
    2.   9.]
 [ 11.   6.   0.  57.  47.   3.   2.   8.  10.   1.   0.   6.  13.   0.
    2.  11.]
 [  5.   3.   2. 109.  47.   2.   3.  10.   6.   5.   0.  10.   2.   0.
    3.  10.]
 [  6.   4.   3.  67.  50.   1.   1.   8.   2.   1.   0.   8.   2.   0.
    5.   5.]
 [ 12.   5.   1.  75.  46.   1.   3.   5.   7.   1.   0.  10.   0.   0.
    1.  11.]
 [ 16.   7.   4.  90.  56.   1.   0.   7.   6.   1.   0.   4.   0.   0.
    3.  10.]
 [  8.   7.   1.  93.  36.   1.   0.  12.  14.   4.   0.   1.   6.   0.
    4.  15.]]
INFO:__main__:Mean counts to fit: [ 9.27272727  5.36363636  1.81818182
83.18181818 50.27272727  0.81818182
  0.90909091  9.18181818  7.27272727  1.54545455  0.          7.36363636
  3.63636364  0.18181818  2.63636364 10.        ]
```

```
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.5569923765609546, 1.0, 1.0,
1.1780972450961724]
INFO:__main__:Neuron 37: R-squared = 0.952
INFO:__main__:Optimized parameters (p_opt): [4.77770423 7.33355117 1.25860608
1.33130282]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[  0.   0.   0.   0.  24.   0.   0.   0.   0.
0.   0.  24. 109.   0.
    0.   0.]
 [  0.   0.   0.   2.  14.   0.   0.   0.   0.   0.   0.  31. 118.   0.
    0.   0.]
 [  0.   0.   0.   5.  12.   0.   0.   0.   0.   0.   0.  46.  90.   0.
    0.   0.]
 [  0.   0.   0.   1.   3.   0.   0.   0.   0.   0.   0.  53. 109.   0.
    0.   0.]
 [  0.   0.   0.   5.  15.   0.   0.   0.   0.   0.   0.  36.  90.   1.
    0.   0.]
 [  0.   0.   0.   1.   4.   0.   0.   0.   0.   0.   0.  45.  69.   4.
    0.   0.]
 [  0.   0.   0.   2.  14.   0.   0.   0.   0.   0.   0.  41.  74.   9.
    0.   0.]
 [  0.   0.   0.   1.  24.   0.   0.   0.   0.   0.   0.  33. 124.  10.
    0.   0.]
 [  0.   0.   0.   0.   7.   0.   0.   0.   0.   0.   0.  36.  82.   6.
    0.   0.]
 [  0.   0.   0.   5.   6.   0.   0.   0.   0.   0.   0.  65.  90.   9.
    0.   0.]
 [  0.   0.   0.   0.  21.   2.   0.   0.   0.   0.   0.   2. 103.   4.
    0.   0.]]
INFO:__main__:Mean counts to fit: [ 0.          0.          0.          2.
13.09090909  0.18181818
  0.          0.          0.          0.          0.         37.45454545
 96.18181818  3.90909091  0.          0.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [3.23748939138219, 1.0, 1.0,
4.71238898038469]
INFO:__main__:Neuron 38: R-squared = 0.999
INFO:__main__:Optimized parameters (p_opt): [4.7480149  7.07698427 1.05170566
```

```
4.60134752]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 5.  2.  0.  0.  0.  4.  1.  8.  0.  4.  0.
0.  0.  0.  0.  1.]
 [ 0.  4.  0.  0.  0.  1.  0.  1.  0. 11.  2.  0.  2.  0.  0.  1.]
 [ 0.  6.  0.  0.  0.  1.  3.  2.  0.  6.  1.  1.  1.  0.  0.  1.]
 [ 0.  2.  0.  0.  0.  2.  3.  5.  1.  3.  0.  1.  1.  1.  1.  3.]
 [ 2.  2.  0.  0.  1.  1.  2.  3.  3.  7.  0.  2.  1.  1.  3.  1.]
 [ 0. 10.  0.  0.  5.  1.  1.  2.  7.  3.  1.  2.  3.  1.  5.  5.]
 [ 2.  3.  0.  1.  1.  0.  7.  1.  1.  5.  0.  0.  0.  3.  3.  1.]
 [ 2.  1.  0.  1.  3.  0.  1.  0.  0.  1.  0.  0.  0.  1.  2.  4.]
 [ 1.  0.  0.  2.  1.  0.  5.  0. 16.  5.  0.  0.  0.  1.  0.  1.]
 [ 2.  1.  4.  0.  1.  4. 12.  1.  3.  4.  0.  0.  0.  6.  0.  2.]
 [ 1.  3.  1.  0.  1.  8.  0.  2.  2.  0.  0.  3.  1.  1.  0.  0.]]
INFO:__main__:Mean counts to fit: [1.36363636 3.09090909 0.45454545 0.36363636
1.18181818 2.
 3.18181818 2.27272727 3.         4.45454545 0.36363636 0.81818182
 0.81818182 1.36363636 1.27272727 1.81818182]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [0.553101106914229, 1.0, 1.0,
3.5342917352885173]
INFO:__main__:Neuron 39: R-squared = 0.481
INFO:__main__:Optimized parameters (p_opt): [1.25402552 0.48614771 0.28888653
3.03735677]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 39.  72.  38.  36.  32.  56.  71.  49.  49.
26.  61.  45.  24.  25.
   14.  22.]
 [ 44.  18.  53.  37.  21.  96.  38.  58.  43.  36.  64.  27.  31.  21.
   47.  42.]
 [ 30.  29.  20.  24.  31.  29.  35.  69.  69.  17.  33.  17.  27.  22.
   49.  44.]
 [ 37.  35.  57.  31.  45.  45.  32.  48.  17.  31.  44.  22.  32.  42.
   43.  23.]
 [ 24.  27.  23.  23.  37.  53.  28.  66.  63.  18.  30.   7.  21.  24.
   23.  28.]
 [ 19.  52.  31.  32.  30.  28. 128.  27.  30.  38.  29.  34.  28.  44.
   31.  23.]
 [ 20.  19.  34.  23.  27.  79.  50.  55.  69.  28.  35.   4.  31.  29.
```

```
   20.  37.]
 [ 26.  42.  30.  39.  53.  25.  68.  75.  79.  30.  19.  50.  21.  23.
   12.  53.]
 [ 28.  37.  24.  27.  76.  28.  78.  61.  39.  19.  49.  66.  54.  39.
   55.  23.]
 [ 24.  62.  19.  29.  40.  72.  37.  79.  37.  44.  41.  45.  36.  52.
   48.  32.]
 [ 24.  26.  17.  22.  59. 102.  71.  70.  58.  61.  34.  50.  17.  23.
   47.  31.]]
INFO:__main__:Mean counts to fit: [28.63636364 38.09090909 31.45454545
29.36363636 41.         55.72727273
 57.81818182 59.72727273 50.27272727 31.63636364 39.90909091 33.36363636
 29.27272727 31.27272727 35.36363636 32.54545455]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [3.6658899358892376, 1.0, 1.0,
2.748893571891069]
INFO:__main__:Neuron 40: R-squared = 0.856
INFO:__main__:Optimized parameters (p_opt): [4.08781324 0.17668345 0.27410818
2.49707122]
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[10. 13. 15. 18. 12. 24. 14. 21. 23. 19. 15.
11. 28. 21. 15. 17.]
 [13. 14. 12. 31. 29. 34. 18. 17.  5. 15. 17. 33. 16. 42. 20. 19.]
 [ 6. 14. 16. 20. 13. 23. 31. 25.  8. 10. 13. 18. 24. 23. 33. 15.]
 [16.  8. 20. 16. 30. 25. 15. 14. 15. 14. 14. 13. 32. 29. 28. 19.]
 [16. 30. 24.  8. 14. 35. 33. 13. 20. 14. 21. 14. 14. 20. 25. 16.]
 [15. 14. 13. 18. 22. 25. 17. 18.  7. 22. 12. 18. 27. 31. 20. 12.]
 [22. 14.  4. 23. 14. 24. 20. 14. 23. 11. 18. 11. 22. 22. 14. 17.]
 [35. 12. 13. 16.  6. 20. 21. 13. 18. 12. 23. 11. 17. 35. 20. 17.]
 [15. 16. 16. 16. 23. 25. 28. 26. 13.  9. 16. 20. 16. 23. 16. 21.]
 [14. 21. 23. 20. 17. 25. 23. 23. 11.  7. 15. 19. 22. 18. 27. 22.]
 [ 4. 19. 19. 17. 11. 37. 34. 13.  8. 11. 16. 27. 34. 32. 15. 12.]]
INFO:__main__:Mean counts to fit: [15.09090909 15.90909091 15.90909091
18.45454545 17.36363636 27.
 23.09090909 17.90909091 13.72727273 13.09090909 16.36363636 17.72727273
 22.90909091 26.90909091 21.18181818 17.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.9299808959912106, 1.0, 1.0,
1.9634954084936207]
```

```
INFO:__main__:Neuron 41: R-squared = 0.809
INFO:__main__:Optimized parameters (p_opt): [3.19773604e+00 2.89233330e-01
1.27612652e-17 1.96575830e+00]
/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/214422327.py:44
: UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```


Neuron 1
Neuron 2
Neuron 3
Neuron 4
Neuron 5
Neuron 6
Neuron 7
Neuron 8
Neuron 9
Neuron 10
Neuron 11
Neuron 12
Neuron 13
Neuron 14
Neuron 15
Neuron 16
Neuron 17
Neuron 18
Neuron 19
Neuron 20
Neuron 21
Neuron 22
Neuron 23
Neuron 24
Neuron 25
Neuron 26
Neuron 27
Neuron 28
Neuron 29
Neuron 30
Neuron 31
Neuron 32
Neuron 33
Neuron 34
Neuron 35
Neuron 36
Neuron 37
Neuron 38
Neuron 39
Neuron 40
Neuron 41
0
90
180
270
360

1.5 Task 4: Permutation test for direction tuning

Implement a permutation test to quantitatively assess whether a neuron is direction/orientation selective. To do so, project the vector of average spike counts, $m_k = \frac{1}{N}\sum_j x_{jk}$ on a complex exponential with two cycles, $v_k = \exp(\psi i \theta_k)$, where θ_k is the k-th direction of motion in radians and $\psi \in 1, 2$ is the fourier component to test (1: direction, 2: orientation). Denote the projection by $q = m^T v$. The magnitude $|q|$ tells you how much power there is in the ψ-th fourier component.

Estimate the distribution of |q| under the null hypothesis that the neuron fires randomly across directions by running 1000 iterations where you repeat the same calculation as above but on a random permutation of the trials (that is, randomly shuffle the entries in the spike count matrix x). The fraction of iterations for which you obtain a value more extreme than what you observed in the data is your p-value. Implement this procedure in the function `testTuning()`.

Illustrate the test procedure for one of the cells from above. Plot the sampling distribution of |q| and indicate the value observed in the real data in your plot.

How many cells are tuned at $p < 0.01$?

Grading: 3 pts

[21]:
```python
def compute_null_distribution(
    counts: np.ndarray,
    dirs: np.ndarray,
    v_k: np.ndarray,
    niters: int,
    rng: np.random.Generator,
) -> np.ndarray:
    """Compute the null distribution of |q| under the null hypothesis.

    Parameters
    ----------
    counts: np.ndarray
        The spike counts for each trial.

    dirs: np.ndarray
        The stimulus directions for each trial.

    v_k: np.ndarray
        The complex exponential vector for the specified psi.

    niters: int
        Number of iterations for the permutation test.

    rng: np.random.Generator
        Random number generator for reproducibility.
    Returns
```

```
    -------
    q_distribution_null: np.ndarray
        The computed null distribution of |q|.

    """
    # Initialize an array to store the |q| values from each permutation
    q_distribution_null = np.zeros(niters)
    logger.info(f"Starting permutation test with {niters} iterations...")

    # Loop over the number of iterations
    for i in range(niters):
        # 1. Permute the data: Shuffle the spike counts randomly across all␣
↪trials.
        shuffled_trial_counts = rng.permutation(counts)

        # 2. Recalculate m_k (average spike count per direction) for this␣
↪permuted dataset.
        permuted_spike_matrix =␣
↪compute_spike_count_matrix(shuffled_trial_counts, dirs)
        m_k_permuted = np.mean(permuted_spike_matrix, axis=0)

        # 3. Recalculate |q| using the permuted m_k_permuted and the *original*␣
↪v_k.
        # v_k does not change because it depends on the stimulus directions and␣
↪psi,
        # which are fixed.
        q_complex_permuted = np.dot(m_k_permuted, v_k)
        q_magnitude_permuted = np.abs(q_complex_permuted)

        # 4. Store the magnitude from this permutation.
        q_distribution_null[i] = q_magnitude_permuted

        if (i + 1) % (niters // 10) == 0:  # Log progress every 10%
            logger.debug(f"Permutation iteration {i+1}/{niters} completed.")

    logger.info("Permutation test finished.")

    return q_distribution_null
```

[22]:
```
def plot_null_distribution(
    q_distribution_null: np.ndarray,
    q_observed_magnitude: float,
    niters: int,
    title: str = "Null Distribution of |q|",
):
    """Plot the null distribution of |q| and the observed |q|.
```

```
    Parameters
    ----------
    q_distribution_null: np.ndarray
        The computed null distribution of |q|.

    q_observed_magnitude: float
        The observed magnitude of |q| from the original data.

    niters: int
        Number of iterations for the permutation test.
    """
    plt.figure(figsize=(8, 6))

    # Plot the histogram of the null distribution
    plt.hist(
        q_distribution_null,
        bins=50,
        density=True,
        alpha=0.7,
        color="skyblue",
        label=f"Null Distribution of |q|\n({niters} permutations)",
    )

    # Add a vertical line for the observed |q|
    plt.axvline(
        x=q_observed_magnitude,
        color="red",
        linestyle="--",
        linewidth=2,
        label=f"Observed |q| = {q_observed_magnitude:.4f}",
    )

    plt.title(title)
    plt.xlabel("|q|")
    plt.ylabel("Density")
    plt.legend()
    plt.grid()
    plt.show()
```

[23]:
```
def testTuning(
    counts: np.ndarray,
    dirs: np.ndarray,
    psi: int = 1,
    niters: int = 1000,
    show: bool = False,
    random_seed: int = 2046,
    neuron: int = None,
```

```
) -> Tuple[float, float, np.ndarray]:
    """Plot the data if show is True, otherwise just return the fit.

    Parameters
    ----------

    counts: np.array, shape=(total_n_trials, )
        the spike count during the stimulation period

    dirs: np.array, shape=(total_n_trials, )
        the stimulus direction in degrees

    psi: int
        fourier component to test (1 = direction, 2 = orientation)

    niters: int
        Number of iterations / permutation

    show: bool
        Plot or not.

    random_seed: int
        Random seed for reproducibility.

    Returns
    -------
    p: float
        p-value
    q: float
        magnitude of second Fourier component

    qdistr: np.array
        sampling distribution of |q| under the null hypothesis

    """
    # --------------------------------
    # Calculate m, nu and q (0.5 pts)
    # --------------------------------
    # m - This is the vector of average spike counts for each unique stimulus␣
↪direction.
    spike_count_matrix = compute_spike_count_matrix(
        counts, dirs
    )  # 'counts' and 'dirs' are 1D arrays of all trials
    m_k = np.mean(spike_count_matrix, axis=0)  # Shape: (nUniqueDirs,)

    # Get unique directions and convert to radians for v_k
    unique_stim_directions_deg = np.unique(
```

```
        dirs
    )  # These are the directions corresponding to columns of spike_count_matrix
    theta_k_rad = np.deg2rad(unique_stim_directions_deg)  # Shape: (nUniqueDirs,)

    # v_k - This is the complex exponential vector for the specified psi.
    # It should be based on the unique radian directions theta_k_rad.
    v_k = np.exp(1j * psi * theta_k_rad)  # Corrected: Shape: (nUniqueDirs,)

    # q - This is the projection of m_k onto v_k.
    q_complex_observed = np.dot(m_k, v_k)

    # Magnitude of the projection for the observed data.
    # This is the 'q' to be returned and tested against null distribution.
    q_observed_magnitude = np.abs(q_complex_observed)

    logger.debug(f"Observed q magnitude: {q_observed_magnitude}")
    logger.debug(f"Observed q complex: {q_complex_observed}")
    logger.debug(f"Observed m_k: {m_k}")
    logger.debug(f"Observed v_k: {v_k}")

    # ------------------------------------------------------------------------
    # Estimate the distribution of q under the H0 and obtain the p value (1 pt)
    # ------------------------------------------------------------------------
    # Ensure reproducibility using a random number generator
    # Hint: Access random functions of this generator
    rng = np.random.default_rng(random_seed)

    # 1-4. Compute the null distribution of |q| under the null hypothesis
    #    by running niters iterations where you repeat the same calculation as above
    #    but on a random permutation of the trials (that is, randomly shuffle the entries in the spike count matrix x).
    #    This is done in the compute_null_distribution function.
    #    The q_distribution_null is the array of |q| values from the null distribution.
    #    The function compute_null_distribution is defined above.
    #    It takes the counts, dirs, v_k, niters, and rng as inputs.
    q_distribution_null = compute_null_distribution(
        counts=counts, dirs=dirs, v_k=v_k, niters=niters, rng=rng
    )

    # 5. Calculate the p-value.
    #    This is the proportion of permuted |q| values that are as extreme as,
    #    or more extreme than, the |q| observed from the original data.
    # We use smoothing to avoid p-values of 0 or 1.
```

```
    p_value = (np.sum(q_distribution_null >= q_observed_magnitude) + 1) /␣
↪(niters + 1)
    # p_value = np.sum(q_distribution_null >= q_observed_magnitude) / niters

    # For a slightly more robust p-value, especially if niters is not huge or␣
↪q_observed_magnitude is very extreme:
    # p_value_corrected = (np.sum(q_distribution_null >= q_observed_magnitude)␣
↪+ 1) / (niters + 1)
    # You can choose which one to use; the simpler one is fine for this lab␣
↪typically.
    logger.info(
        f"Observed |q|: {q_observed_magnitude:.4f}, Calculated p-value:␣
↪{p_value:.35f}"
    )

    if show:
        neuron_title_str = "" if neuron is None else f"Neuron: {neuron}"
        title = f"Null Distribution {neuron_title_str} $\Psi$ : {psi}  P Value:␣
↪{p_value:.05f}"
        plot_null_distribution(
            q_distribution_null, q_observed_magnitude, niters, title=title
        )
    # The array q_distribution_null is the 'qdistr' to be returned.
    qdistr = q_distribution_null
    return p_value, q_observed_magnitude, qdistr
```

[24]:
```
def test_neuron_tuning(all_neuron_ids, spikes, niters=1000):
    """Test the tuning of a neuron for direction and orientation."""
    tuning_results = []

    for i, neuron_id in enumerate(all_neuron_ids):
        logger.info(f"Processing neuron {i+1}/{len(all_neuron_ids)}: ID␣
↪{neuron_id}...")

        # Get data for the current neuron
        dirs_sorted, counts_sorted = get_data(spikes, neuron_id)

        # It's possible some neurons might have no spikes in the defined␣
↪periods.
        # get_data should produce empty or all-zero counts_sorted in such cases.
        # testTuning should ideally handle this (e.g., result in a␣
↪non-significant p-value).
        # For instance, if all counts are 0, m_k will be 0,␣
↪q_observed_magnitude will be 0,
        # and p_value will likely be 1.0, which is correct (not selective).
```

```
        # Test for direction tuning (psi=1)
        p_direction, q_direction, _ = testTuning(
            counts_sorted,
            dirs_sorted,
            psi=1,
            niters=niters,
            show=False,  # No plots for batch processing
            neuron=neuron_id,  # Pass neuron ID for logging
        )

        # Test for orientation tuning (psi=2)
        p_orientation, q_orientation, _ = testTuning(
            counts_sorted,
            dirs_sorted,
            psi=2,
            niters=niters,
            show=False,
            neuron=neuron_id,  # Pass neuron ID for logging
        )

        tuning_results.append(
            {
                "neuron_id": neuron_id,
                "p_direction": p_direction,
                "q_direction": q_direction,
                "p_orientation": p_orientation,
                "q_orientation": q_orientation,
            }
        )

    return pd.DataFrame(tuning_results)
```

[25]:
```
def filter_tuning_results(tuning_results_df, alpha_threshold=0.01):
    """Filter the tuning results based on the significance level."""
    logging.debug(f"\n--- Tuning Selectivity Results (p < {alpha_threshold})␣
↪---")

    direction_selective_neurons = tuning_results_df[
        tuning_results_df["p_direction"] < alpha_threshold
    ]
    orientation_selective_neurons = tuning_results_df[
        tuning_results_df["p_orientation"] < alpha_threshold
    ]

    logging.debug(f"\nDirection Selective Neurons (psi=1, p <␣
↪{alpha_threshold}):")
    if not direction_selective_neurons.empty:
```

```
        logging.debug(
            direction_selective_neurons[["neuron_id", "p_direction",␣
↪"q_direction"]]
        )
    else:
        logging.debug(
            "No neurons found to be significantly direction selective at this␣
↪threshold."
        )

    logging.debug(f"\nOrientation Selective Neurons (psi=2, p <␣
↪{alpha_threshold}):")
    if not orientation_selective_neurons.empty:
        logging.debug(
            orientation_selective_neurons[
                ["neuron_id", "p_orientation", "q_orientation"]
            ]
        )
    else:
        logging.debug(
            "No neurons found to be significantly orientation selective at this␣
↪threshold."
        )

    # You might also be interested in neurons that are BOTH or EXCLUSIVELY one␣
↪type
    both_selective = tuning_results_df[
        (tuning_results_df["p_direction"] < alpha_threshold)
        & (tuning_results_df["p_orientation"] < alpha_threshold)
    ]
    logging.debug(
        f"\nNeurons Selective for BOTH Direction and Orientation (p <␣
↪{alpha_threshold}):"
    )
    if not both_selective.empty:
        logging.debug(both_selective[["neuron_id", "p_direction",␣
↪"p_orientation"]])
    else:
        logging.debug(
            "No neurons found to be significantly selective for both at this␣
↪threshold."
        )

    # Example: Strictly direction selective (significant for direction, not for␣
↪orientation)
    strictly_direction_selective = tuning_results_df[
```

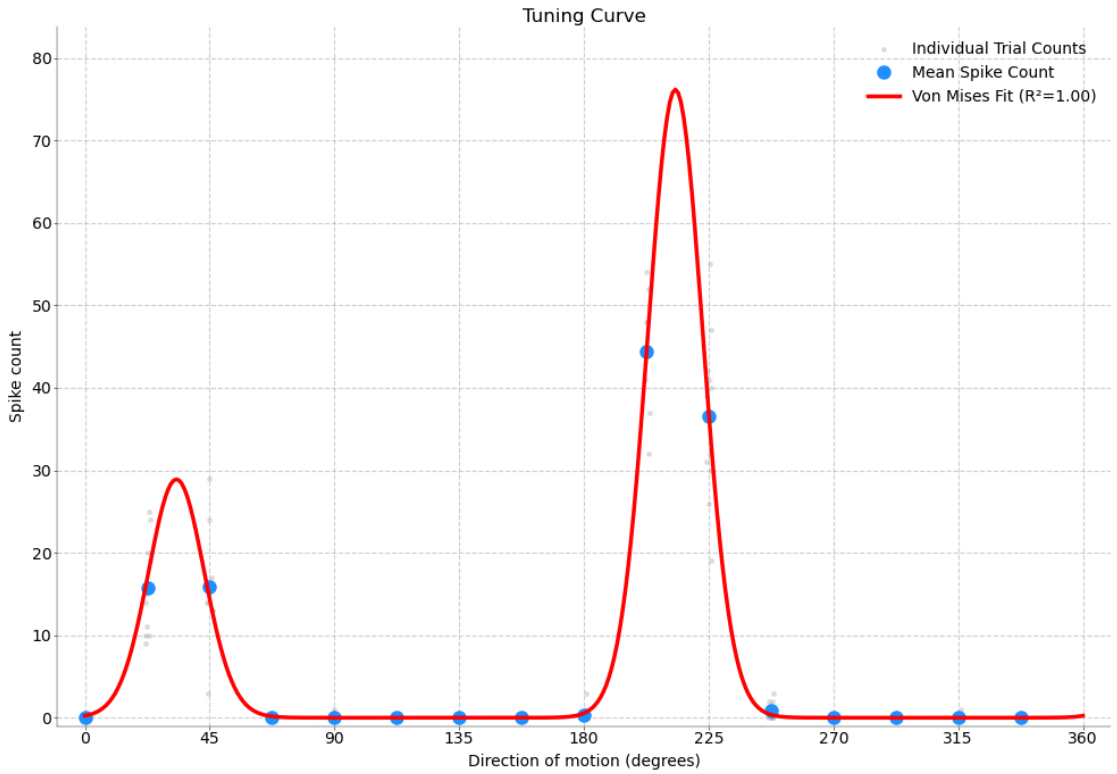
```
        (tuning_results_df["p_direction"] < alpha_threshold)
        & (tuning_results_df["p_orientation"] >= alpha_threshold)
    ]
    logging.debug(
        f"\nNeurons Strictly Direction Selective (p_dir < {alpha_threshold},␣
↪p_ori >= {alpha_threshold}):"
    )
    if not strictly_direction_selective.empty:
        logging.debug(
            strictly_direction_selective[["neuron_id", "p_direction",␣
↪"p_orientation"]]
        )
    else:
        logging.debug(
            "No neurons found to be strictly direction selective at this␣
↪threshold."
        )
    return (
        direction_selective_neurons,
        orientation_selective_neurons,
        both_selective,
        strictly_direction_selective,
    )
```

Show null distribution for the example cell:

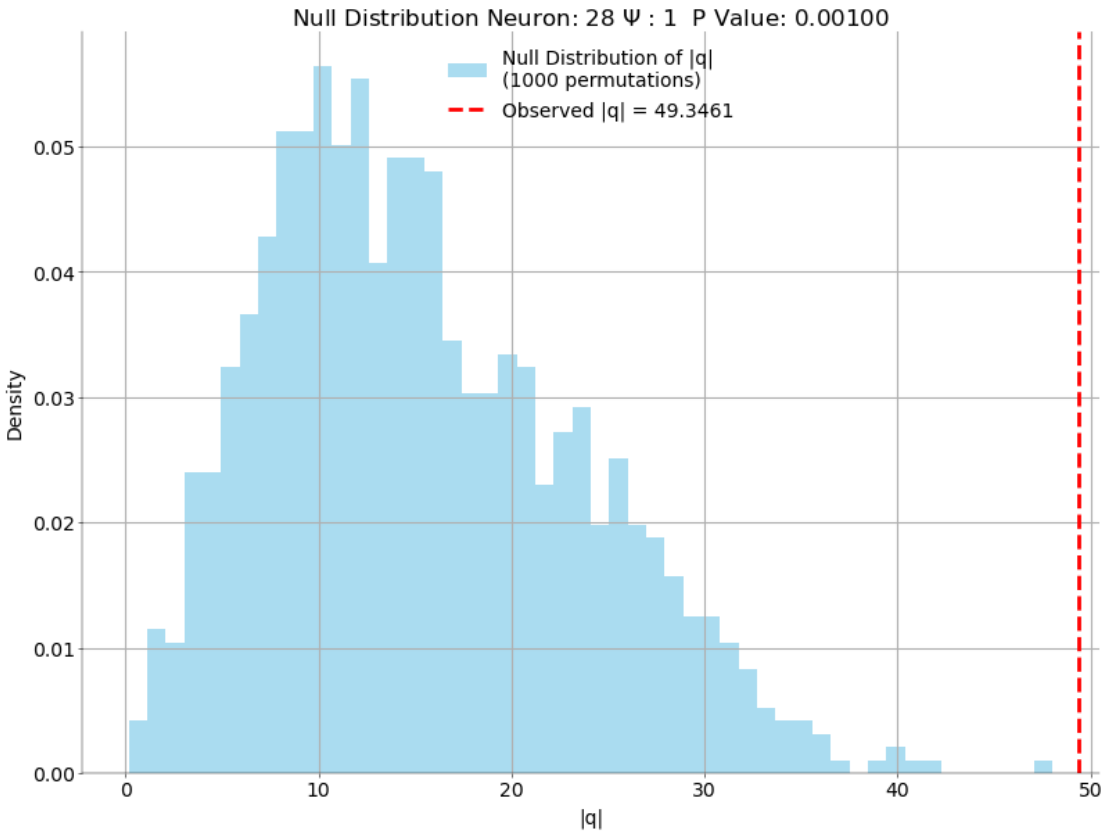
[26]:
```
# ------------------------------------------------------------
# Plot null distributions for example cells 28 & 29. (1 pt)
# ------------------------------------------------------------
neurons_to_plot = [28, 29]
for neuron in neurons_to_plot:
    dirs_sorted, counts_sorted = get_data(spikes, neuron)
    result = tuningCurve(counts_sorted, dirs_sorted, show=True)
    for psi in [1, 2]:
        p_value, q_observed_magnitude, qdistr = testTuning(
            counts_sorted, dirs_sorted, psi=psi, show=True, niters=1000,␣
↪neuron=neuron
        )
        logger.info(
            f"Neuron:{neuron} psi == {psi} ->  p-value: {p_value},␣
↪q_observed_magnitude: {q_observed_magnitude}"
        )
```

```
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 0. 16. 14.  0.  0.  0.  0.  0.  0. 32. 31.
```

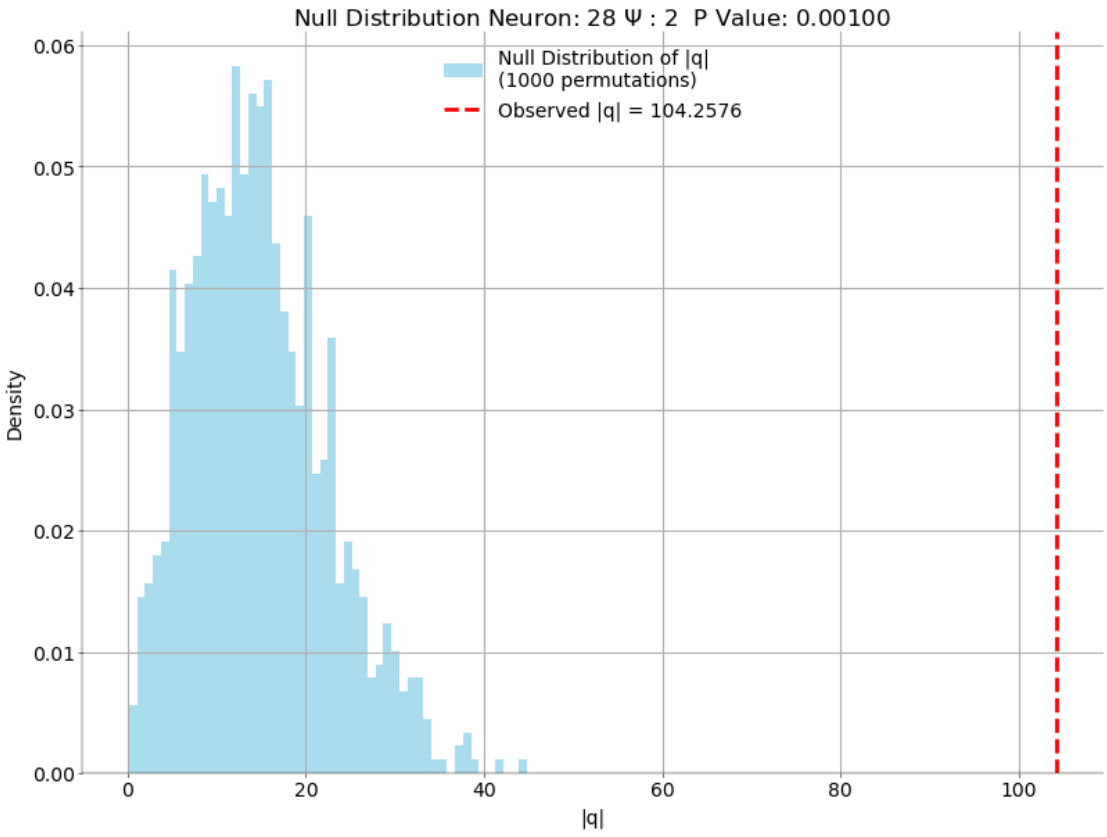
```
0.  0.  0.  0.  0.]
 [ 0. 10. 13.  0.  0.  0.  0.  0.  0. 44. 19.  0.  0.  0.  0.  0.]
 [ 0. 20. 14.  0.  0.  0.  0.  0.  0. 41. 42.  0.  0.  0.  0.  0.]
 [ 0. 10. 16.  0.  0.  0.  0.  0.  0. 54. 32.  0.  0.  0.  0.  0.]
 [ 0. 24. 16.  0.  0.  0.  0.  0.  0. 43. 55.  0.  0.  0.  0.  0.]
 [ 0.  9. 13.  0.  0.  0.  0.  0.  0. 52. 41.  0.  0.  0.  0.  0.]
 [ 0. 19. 15.  0.  0.  0.  0.  0.  0. 44. 30.  3.  0.  0.  0.  0.]
 [ 0. 15.  3.  0.  0.  0.  0.  0.  0. 48. 47.  1.  0.  0.  0.  0.]
 [ 0. 25. 24.  0.  0.  0.  0.  0.  3. 45. 26.  1.  0.  0.  0.  0.]
 [ 0. 11. 29.  0.  1.  0.  0.  0.  1. 37. 40.  2.  0.  0.  0.  0.]
 [ 0. 14. 17.  0.  0.  0.  0.  0.  0. 48. 39.  2.  0.  0.  1.  0.]]
INFO:__main__:Mean counts to fit: [ 0.         15.72727273 15.81818182  0.
0.09090909  0.
  0.          0.          0.36363636 44.36363636 36.54545455  0.81818182
  0.          0.          0.09090909  0.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [2.6551607371818373, 1.0, 1.0,
3.5342917352885173]
INFO:__main__:Neuron Unknown: R-squared = 0.999
INFO:__main__:Optimized parameters (p_opt): [4.33282731 8.18559981 0.48435863
3.71739285]
/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/4021924317.py:1
25: UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```




```
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 49.3461, Calculated p-value:
0.000999000999000999000013250575625534
```

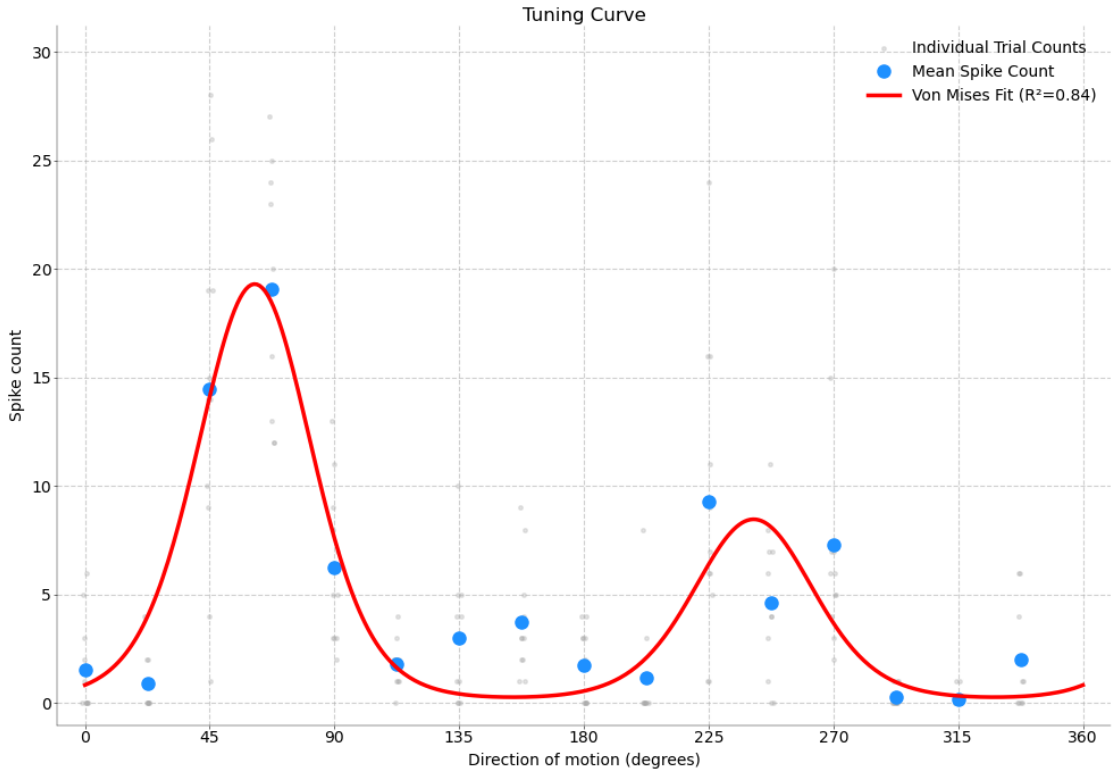



```
INFO:__main__:Neuron:28 psi == 1 ->  p-value: 0.000999000999000999,
q_observed_magnitude: 49.34605505462993
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 104.2576, Calculated p-value:
0.000999000999000999000013250575625534
```

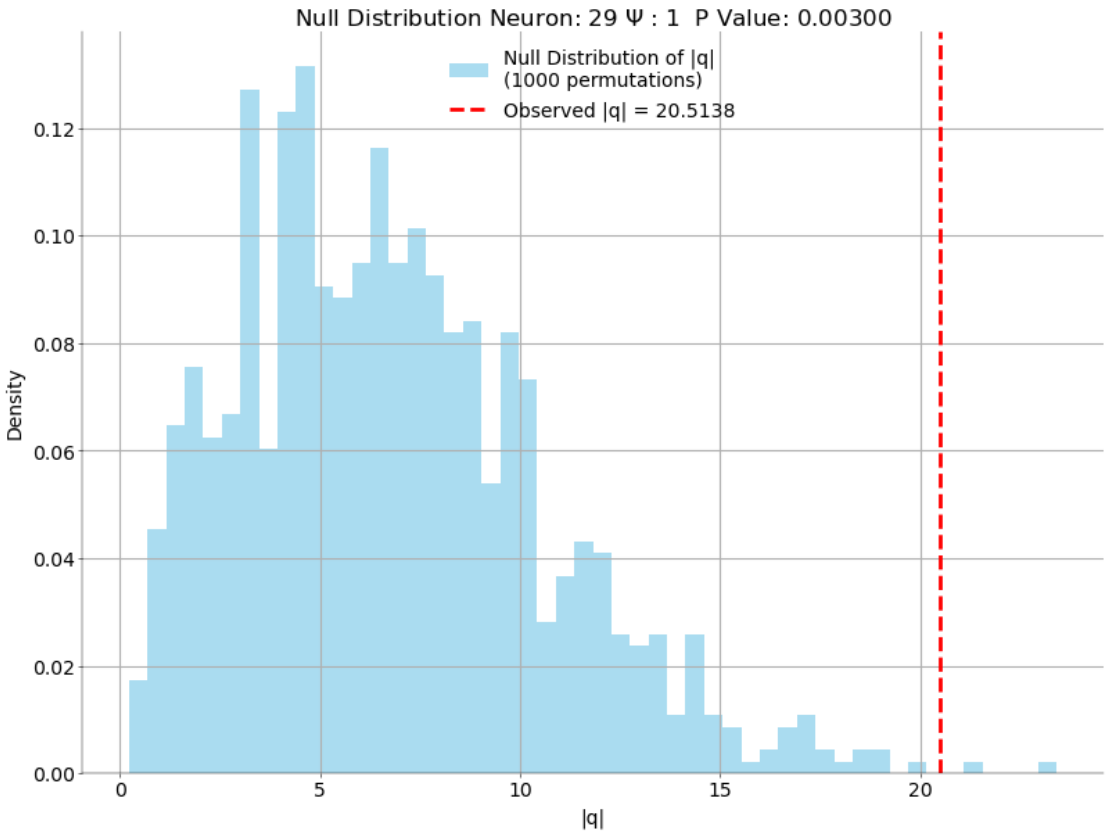



```
INFO:__main__:Neuron:28 psi == 2 ->  p-value: 0.000999000999000999,
q_observed_magnitude: 104.25762657685993
INFO:__main__:Fitting tuning curve…
INFO:__main__:Counts: (176,)
INFO:__main__:Dirs: (176,)
INFO:__main__:Spike count matrix shape: (11, 16)
INFO:__main__:Spike count matrix: [[ 5.  2.  4. 13.  2.  0.  0.  9.  0.  1. 24.
0.  2.  0.  1.  1.]
 [ 0.  0. 19. 20.  8.  3.  0.  3.  2.  0.  9.  8.  4.  0.  0.  2.]
 [ 0.  0.  9. 19. 13.  1.  0.  4.  0.  0.  6.  0. 15.  0.  0.  6.]
 [ 0.  0.  1. 16.  3.  1.  5.  2.  0.  1.  7. 11.  7.  0.  0.  6.]
 [ 0.  0. 28. 12.  3.  1. 10.  1.  0.  0.  1.  7.  6.  0.  0.  1.]
 [ 0.  0. 14. 25.  7.  4.  1.  8.  0.  0.  1.  4.  5.  0.  0.  0.]
 [ 0.  0. 15. 12.  9.  2.  1.  3.  3.  0. 11.  3.  7.  0.  0.  2.]
 [ 6.  4. 19. 27.  5.  2.  4.  5.  4.  0. 16.  6.  6.  1.  0.  0.]
 [ 3.  2. 14. 23.  5.  2.  3.  2.  4.  0. 16.  4.  5.  1.  0.  0.]
 [ 2.  1. 10. 19. 11.  2.  5.  1.  3.  8.  5.  7.  3.  0.  0.  4.]
 [ 1.  1. 26. 24.  3.  2.  4.  3.  3.  3.  6.  1. 20.  1.  1.  0.]]
INFO:__main__:Mean counts to fit: [ 1.54545455  0.90909091 14.45454545
19.09090909  6.27272727  1.81818182
  3.          3.72727273  1.72727273  1.18181818  9.27272727  4.63636364
```

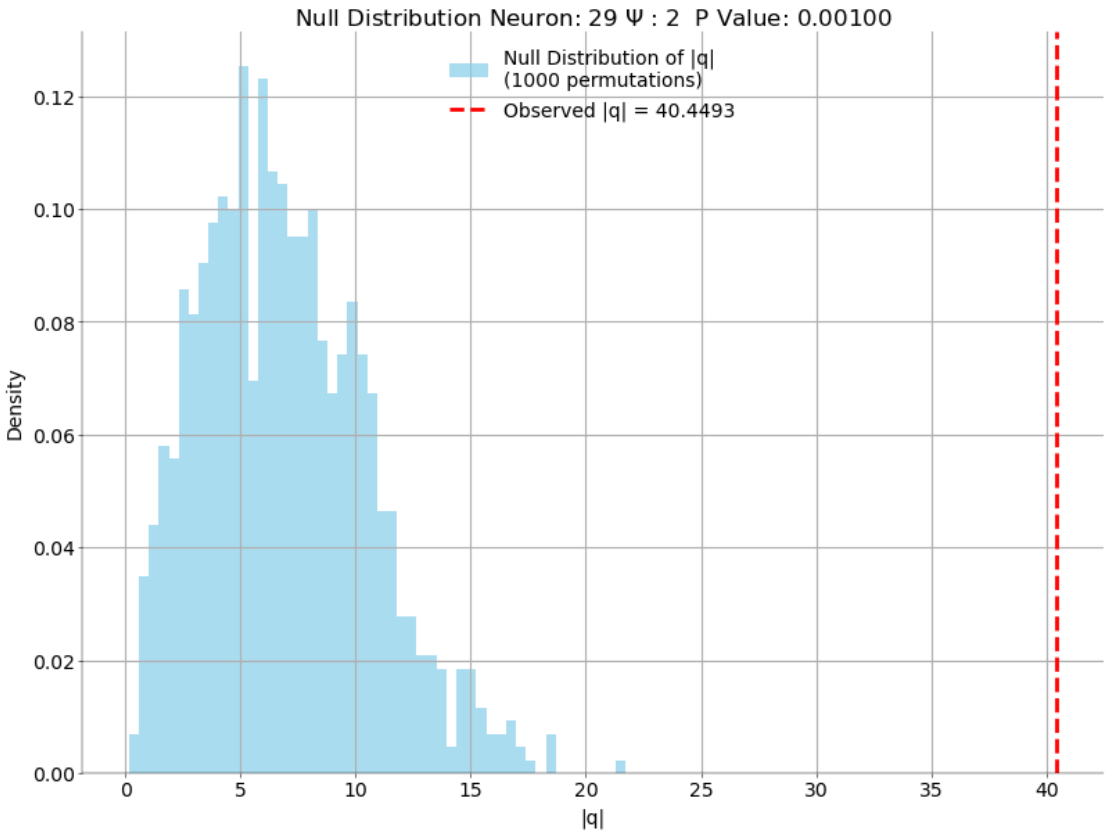
```
  7.27272727  0.27272727  0.18181818  2.        ]
INFO:__main__:Unique directions (radians) for fitting: [0.         0.39269908
0.78539816 1.17809725 1.57079633 1.96349541
 2.35619449 2.74889357 3.14159265 3.53429174 3.92699082 4.3196899
 4.71238898 5.10508806 5.49778714 5.89048623]
INFO:__main__:Initial parameter guesses (p0): [1.5759281335352222, 1.0, 1.0,
1.1780972450961724]
INFO:__main__:Neuron Unknown: R-squared = 0.841
INFO:__main__:Optimized parameters (p_opt): [2.96043674 1.90155694 0.41176043
1.06863547]
/var/folders/_q/9l1zk1853mv45phss2fsffd00000gn/T/ipykernel_70358/4021924317.py:1
25: UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```




```
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 20.5138, Calculated p-value:
0.002997002997002997000397517268766O2
```




```
INFO:__main__:Neuron:29 psi == 1 ->  p-value: 0.002997002997002997,
q_observed_magnitude: 20.51379441068113
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 40.4493, Calculated p-value:
0.000999000999000999000013250575625534
```




```
INFO:__main__:Neuron:29 psi == 2 ->  p-value: 0.000999000999000999,
q_observed_magnitude: 40.44931965569163
```

Test all cells for orientation and direction tuning

[27]:
```
# ------------------------------------------------
# Test all cells for orientation / direction tuning.
# Which ones are selective? (0.5 pts)
# ------------------------------------------------
# %%
neuron_tuning_results_df = test_neuron_tuning(
    np.unique(spikes["Neuron"]), spikes, niters=1000
)
# %% Filter the results based on the significance level
(
    direction_selective_neurons,
    orientation_selective_neurons,
    both_selective,
    strictly_direction_selective,
) = filter_tuning_results(neuron_tuning_results_df, alpha_threshold=0.01)
```

```
INFO:__main__:Processing neuron 1/41: ID 1…
```

```
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 5.6842, Calculated p-value:
0.878121878121878141065792533481726 42
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 8.9238, Calculated p-value:
0.692307692307692290611953467305283 99
INFO:__main__:Processing neuron 2/41: ID 2…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 13.1883, Calculated p-value:
0.476523476523476541277801743490272 20
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 69.5850, Calculated p-value:
0.000999000999000999000132505756255 34
INFO:__main__:Processing neuron 3/41: ID 3…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 16.6166, Calculated p-value:
0.063936063936063936008480368400341 83
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 27.7114, Calculated p-value:
0.001998001998001998000265011512510 68
INFO:__main__:Processing neuron 4/41: ID 4…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 5.3605, Calculated p-value:
0.793206793206793214778826950350776 31
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 23.2852, Calculated p-value:
0.013986013986013986001855080587574 78
INFO:__main__:Processing neuron 5/41: ID 5…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 5.5427, Calculated p-value:
0.013986013986013986001855080587574 78
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 4.7990, Calculated p-value:
0.043956043956043959475277205228849 20
INFO:__main__:Processing neuron 6/41: ID 6…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 19.4240, Calculated p-value:
```

```
0.12287712287712287961838342198461760
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 79.7655, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 7/41: ID 7…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 3.5543, Calculated p-value:
0.27772227722277699485843141254049260
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 12.7217, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 8/41: ID 8…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 12.4472, Calculated p-value:
0.60839160839160844052031507089850493
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 81.1477, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 9/41: ID 9…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 7.6420, Calculated p-value:
0.13686313686313686388551502659538528
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 6.5348, Calculated p-value:
0.26173826173826175711667474388377741
INFO:__main__:Processing neuron 10/41: ID 10…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 8.7905, Calculated p-value:
0.33966033966033964963671110126597341
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 20.8556, Calculated p-value:
0.0029970029970029970003975172687602
INFO:__main__:Processing neuron 11/41: ID 11…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 2.2733, Calculated p-value:
0.76723276723276723210176442080410197
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
```

```
INFO:__main__:Observed |q|: 6.5622, Calculated p-value:
0.10389610389610390295267450255778385
INFO:__main__:Processing neuron 12/41: ID 12…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 5.1742, Calculated p-value:
0.20679320679320678522117304964922369
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 14.8090, Calculated p-value:
0.00099900099900099900013250575625534
INFO:__main__:Processing neuron 13/41: ID 13…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 77.9503, Calculated p-value:
0.00099900099900099900013250575625534
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 48.3590, Calculated p-value:
0.00099900099900099900013250575625534
INFO:__main__:Processing neuron 14/41: ID 14…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 2.9696, Calculated p-value:
0.76723276723276723210176442080410197
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 24.9460, Calculated p-value:
0.00099900099900099900013250575625534
INFO:__main__:Processing neuron 15/41: ID 15…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 16.9275, Calculated p-value:
0.23976023976023977390958918931573862
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 43.2737, Calculated p-value:
0.00099900099900099900013250575625534
INFO:__main__:Processing neuron 16/41: ID 16…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 3.8806, Calculated p-value:
0.85914085914085913664450799842597917
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 42.7545, Calculated p-value:
0.00099900099900099900013250575625534
INFO:__main__:Processing neuron 17/41: ID 17…
```

```
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 24.3722, Calculated p-value:
0.025974025974025975738162862563944596
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 79.6499, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 18/41: ID 18…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 7.2394, Calculated p-value:
0.11688311688311688041341795951666427
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 14.3494, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 19/41: ID 19…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 9.0831, Calculated p-value:
0.26173826173826175711667474388377741
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 14.2935, Calculated p-value:
0.043956043956043959475277205228849 20
INFO:__main__:Processing neuron 20/41: ID 20…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 9.2978, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 12.6876, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 21/41: ID 21…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 9.4781, Calculated p-value:
0.955044955044955012901652935397578 40
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 285.4357, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 22/41: ID 22…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 20.7610, Calculated p-value:
```

```
0.07092907092907092814204617070572567
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 70.9471, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 23/41: ID 23…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 4.6848, Calculated p-value:
0.044955044955044952403877545066279 71
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 11.9849, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 24/41: ID 24…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 31.4009, Calculated p-value:
0.001998001998001998000265011512510 68
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 61.4374, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 25/41: ID 25…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 35.7000, Calculated p-value:
0.003996003996003996000530023025021 36
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 64.9735, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 26/41: ID 26…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 6.1104, Calculated p-value:
0.582417582417582457843252541351830 59
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 27.3141, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Processing neuron 27/41: ID 27…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 31.6401, Calculated p-value:
0.000999000999000999000013250575625534
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
```

```
INFO:__main__:Observed |q|: 32.0307, Calculated p-value:
0.000999000999000999900013250575625534
INFO:__main__:Processing neuron 28/41: ID 28…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 49.3461, Calculated p-value:
0.000999000999000999900013250575625534
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 104.2576, Calculated p-value:
0.000999000999000999900013250575625534
INFO:__main__:Processing neuron 29/41: ID 29…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 20.5138, Calculated p-value:
0.002997002997002997000397517268766O2
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 40.4493, Calculated p-value:
0.000999000999000999900013250575625534
INFO:__main__:Processing neuron 30/41: ID 30…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 10.4926, Calculated p-value:
0.881118881118881147607169168622931 47
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 194.9890, Calculated p-value:
0.000999000999000999900013250575625534
INFO:__main__:Processing neuron 31/41: ID 31…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 89.1256, Calculated p-value:
0.000999000999000999900013250575625534
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 126.4620, Calculated p-value:
0.000999000999000999900013250575625534
INFO:__main__:Processing neuron 32/41: ID 32…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 43.9712, Calculated p-value:
0.000999000999000999900013250575625534
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 47.4588, Calculated p-value:
0.000999000999000999900013250575625534
INFO:__main__:Processing neuron 33/41: ID 33…
```

```
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 15.0091, Calculated p-value:
0.399600399600399613930790110316593 20
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 76.7866, Calculated p-value:
0.000999000999000999000132505756255 34
INFO:__main__:Processing neuron 34/41: ID 34…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 5.6435, Calculated p-value:
0.010989010989010989868819301307212 30
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 9.9662, Calculated p-value:
0.000999000999000999000132505756255 34
INFO:__main__:Processing neuron 35/41: ID 35…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 3.8365, Calculated p-value:
0.151848151848151841081246971043583 47
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 3.1935, Calculated p-value:
0.285714285714285698425385362497763 71
INFO:__main__:Processing neuron 36/41: ID 36…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 79.2753, Calculated p-value:
0.024975024975024975870674381894787 07
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 191.2953, Calculated p-value:
0.000999000999000999000132505756255 34
INFO:__main__:Processing neuron 37/41: ID 37…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 124.2948, Calculated p-value:
0.000999000999000999000132505756255 34
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 98.9777, Calculated p-value:
0.000999000999000999000132505756255 34
INFO:__main__:Processing neuron 38/41: ID 38…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 119.9065, Calculated p-value:
```

```
0.0009990009990009990013250575625534
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 142.2787, Calculated p-value:
0.0009990009990009990013250575625534
INFO:__main__:Processing neuron 39/41: ID 39…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 5.2675, Calculated p-value:
0.043956043956043959475277205228849 20
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 7.8688, Calculated p-value:
0.0019980019980019980002650115125 1068
INFO:__main__:Processing neuron 40/41: ID 40…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 90.8734, Calculated p-value:
0.0009990009990009990013250575625534
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 57.5882, Calculated p-value:
0.0029970029970029970003975172687 6602
INFO:__main__:Processing neuron 41/41: ID 41…
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 2.8878, Calculated p-value:
0.9000999000999001520284537036786 7872
INFO:__main__:Starting permutation test with 1000 iterations…
INFO:__main__:Permutation test finished.
INFO:__main__:Observed |q|: 41.5034, Calculated p-value:
0.0009990009990009990013250575625534
```

[28]:
```
num_direction_tuned = len(direction_selective_neurons)
num_orientation_tuned = len(orientation_selective_neurons)
num_both_tuned = len(both_selective)  # 'both_selective' from your␣
 ↪filter_tuning_results

total_unique_tuned_cells = num_direction_tuned + num_orientation_tuned -␣
 ↪num_both_tuned

print(
    f"Total number of unique neurons showing any tuning (direction or␣
 ↪orientation, p < 0.01): {total_unique_tuned_cells}"
)
```

```
Total number of unique neurons showing any tuning (direction or orientation, p <
0.01): 34
```

Number of direction tuned neurons:

[29]:
```
print(f"Number of direction selective neurons:␣
 ↪{len(direction_selective_neurons)}")
```

```
Number of direction selective neurons: 12
```

Number of orientation tuned neurons:

[30]:
```
print(f"Number of orientation selective neurons:␣
 ↪{len(orientation_selective_neurons)}")
```

```
Number of orientation selective neurons: 34
```

[31]:
```
print(f"Number of neurons selective for both: {len(both_selective)}")
print(
    f"Number of strictly direction selective neurons:␣
 ↪{len(strictly_direction_selective)}"
)
```

```
Number of neurons selective for both: 12
Number of strictly direction selective neurons: 0
```