CUED - Engineering Tripos Part IIB 2023-2024 **Module Coursework**

Module	4G10	Title of report	Coursework 1 – Rotational Dynamics in the Motor Cortex
Date submitted: 17/11/2023			Assessment for this module is ☑ 100% / ☐ 25% coursework of which this assignment forms __50_ %

UNDERGRADUATE and POST GRADUATE STUDENTS

Candidate number:	5553C	☑ Undergraduate ☐ Post graduate

Feedback to the student ☐ **See also comments in the text**		Very good	**Good**	Needs improvmt
CONTENT	**Completeness, quantity of content:** Has the report covered all aspects of the lab? Has the analysis been carried out thoroughly?			
	Correctness, quality of content Is the data correct? Is the analysis of the data correct? Are the conclusions correct?			
	Depth of understanding, quality of discussion Does the report show a good technical understanding? Have all the relevant conclusions been drawn?			
	Comments:			
PR	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			

1 Plotting raw PSTHs

The original dataset gives the PSTHs (peristimulus time histogram) of 182 different neurons under 108 experimental conditions, against time relative to the onset of hand movement. PSTHs show us the average number of spikes for a given time bin, in this case $10ms$ long.

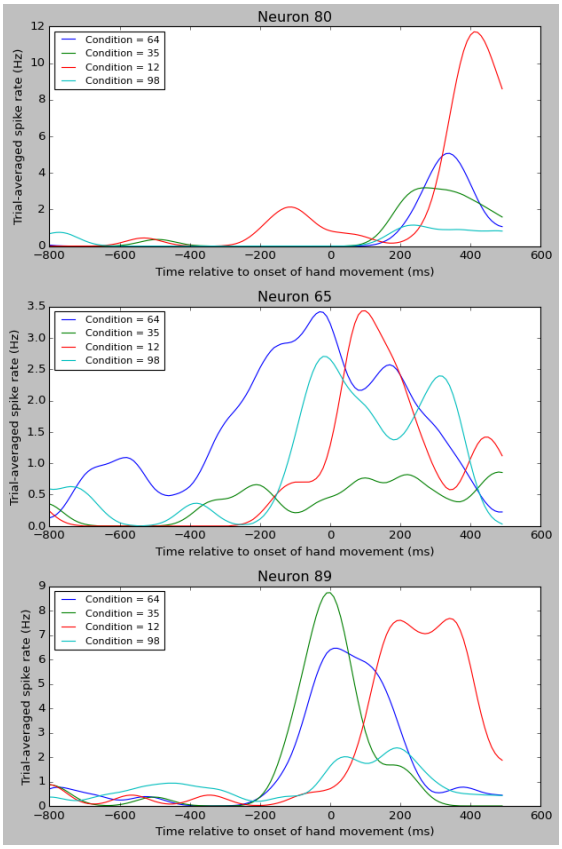

Figure 1: Plot of average spike count against time relative to the onset of movement for 3 sample neurons under 4 sample conditions.

A small sample of this data is shown in Figure 1.

Note that in the pre-movement period of Figure 1, the average spike count in neurons in different conditions tends to be low. Just before the onset of hand movement, we begin to observe a sharp rise in the average spike count. Around $0ms$ the average spike count peaks and then begins to fall again, but not quite down to the pre-movement period levels. These trends are true for all neurons as shown by Figure 2.

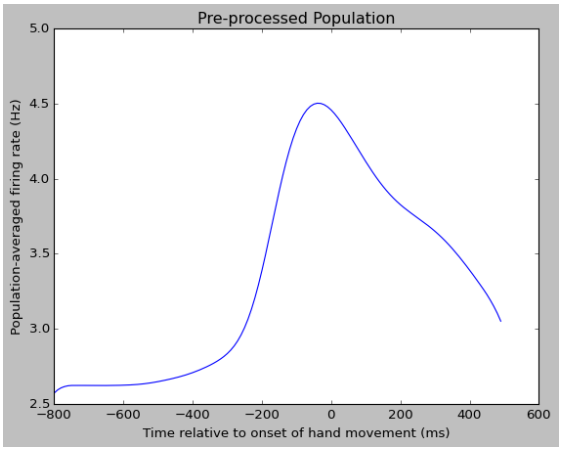

Figure 2: Original population-averaged firing rate plotted against time relative to the onset of hand movement. Averaged across all conditions and neurons.

Interestingly, the sharp rise in the average spike counts begins at approximately $-300ms$ relative to the onset of movement. This is likely due to the subject readying itself for the task, hence the neurons fire indicating the brain has planned the trajectory and is waiting for the go cue. This implies that the "target onset" (the time at which the target appears) stage occurs approximately at $-300ms$.

2 Pre-processing

As shown in Figure 3, certain neurons exhibit a higher average spike count. These neurons will hence give rise to large eigenvalues, whilst small ones are still present. This is undesirable as it means the condition number (ratio between the largest and smallest eigenvalues) will be large, implying the matrix is poorly conditioned and hence unstable (e.g. more likely to have numerical overflows) when plugged into linear solvers. Secondly, and more importantly for our data, if the variance of our neurons is non-uniform, the PCA (principal component analysis) will be biased towards the neurons with a larger variance. The neurons with larger maximum values will hence dominate the formation of the principal components.

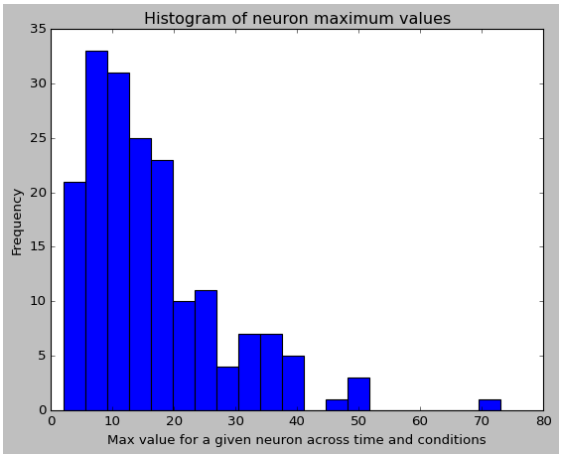

Figure 3: Maximum average spike count of neurons across all times and conditions.

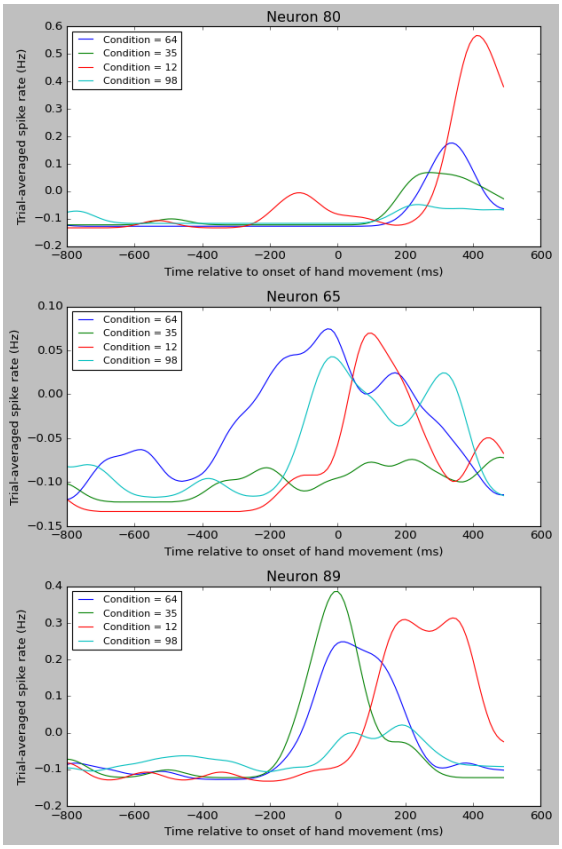

Figure 4: Pre-processed neuron average spike count plotted against time relative to the onset of hand movement. The raw version is shown in Figure 1.

By normalising and subtracting the mean (standardising), the data has no loss of information and will now perform better in linear solvers as well as providing an unbiased PCA.

The result of such processing can be seen in Figure 4 and compared to the unprocessed, shown in Figure 1.

3 Plotting PC space trajectories

Using the pre-processed data, we are able to use PCA to reduce the first dimension of the data from 182 (the total number of neurons) to 12. Figure 5 shows the plot of the first two principal components; these are the components which explain the most and second most, respectively, amount of variance in the PSTHs.

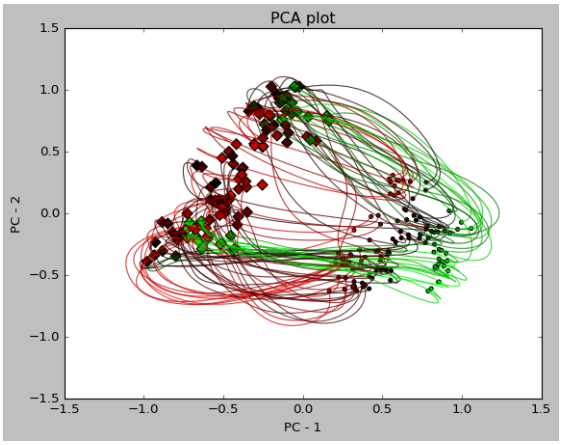

Figure 5: Plot of the first two principal components of the neural data. Each line represents one of the conditions with a round marker denoting the first time, through to the last time denoted by the diamond marker. Red implies the maximum-spread direction is positive, green for negative and black for close to zero.

With the number of dimensions reduced and centred around the origin, we can begin proving our hypothesis that the data exhibits time-invariant rotational dynamics.

4 Maximum-likelihood estimate of A

$$\Delta z_{t+1} = A z_t + \sigma \epsilon_t \tag{1}$$

With $\sigma = 1$ and $\epsilon_t = N(0, I_{12})$ we are able to rewrite our time-invariant rotational dynamic model in Equation 1 as Equation 2.

$$\Delta z_{t+1} = \mathcal{N}(Az_t, I_{12}) \quad (2)$$

This implies that the likelihood of the sequence is given by Equation 3.

$$P(\Delta z_{1:T}|z_{0:T-1}) = \prod_{t=0}^{T} \mathcal{N}(Az_t, I_{12}) \quad (3)$$

Taking the log of this, subbing in the terms for the multivariate Gaussian and converting to matrix form, yields Equation 4. Note all terms independent of A are dropped. Also note that when solving, we obtain the required ΔZ using $\Delta Z_{t+1} = Z_{t+1} - Z_t$; that is why in Equation 3 the $\Delta z_{1:T}$ is shifted one ahead of $z_{0:T-1}$.

$$log(P(\Delta Z|Z)) \approx -Z^T A^T AZ + 2\Delta Z^T AZ \quad (4)$$

We then differentiate Equation 4 in order to obtain Equation 5. Setting this equal to zero would allow us to estimate A. However, the solution provided here doesn't constrain A to be an anti-symmetric matrix.

$$\frac{d}{dA}(log(P(\Delta Z|Z))) \propto -AZZ^T + \Delta ZZ^T \quad (5)$$

Therefore, to avoid constrained optimisation, we use the anti-symmetric relations to construct the matrix A from a row vector β. By doing this we are able to find an estimate of A by differentiating the log-likelihood w.r.t β.

This is done using Equation 6, where H is a tensor that has dimensions $K \times M \times M$. This tensor uses β to construct a $M \times M$ anti-symmetric matrix.

$$A_{i,j} = \sum_{a=1}^{K} \beta_a H_{a,i,j} \quad (6)$$

K is the number of unconstrained and non-redundant parameters within A. We know that the diagonal must be 0 due to the rotations being small; hence K is equal to the number of entries above the diagonal. We don't require the lower entries as the matrix is anti-symmetric so this can be deduced from the upper part. K is given by Equation 7. For our case $M = 12$ we have $K = 66$.

$$K = \frac{M^2 - M}{2} \quad (7)$$

Using Equation 6 and $W_{a,i,n} = \sum_{j=1}^{M} H_{a,i,j} Z_{j,n}$, we can write get $AZ = \sum_{a=1}^{K} \beta_a \sum_{j=1}^{M} H_{a,i,j} Z_{j,n}$. Summing this over j and subbing into Equation 4, we obtain Equation 8, the log-likelihood in terms of β.

$$log(P(\Delta Z|Z)) \approx -(\beta W)^T(\beta W) + 2\Delta Z^T(\beta W) \quad (8)$$

Differentiating this now with respect to β gives us Equation 9; setting this to zero allows us to solve for β, given by $\beta = bQ^{-1}$ where $Q = W^T W$ and $b = \Delta Z^T W$.

$$\frac{d}{d\beta}(log(P(\Delta Z|Z))) \approx -2\beta W^T W + 2\Delta Z^T W \quad (9)$$

Then using Equation 6 we construct the estimate for the anti-symmetric matrix A. The visualisation of the estimate is shown in Figure 6. The largest error of any of the elements was 2.69×10^{-12} (3 S.F.).

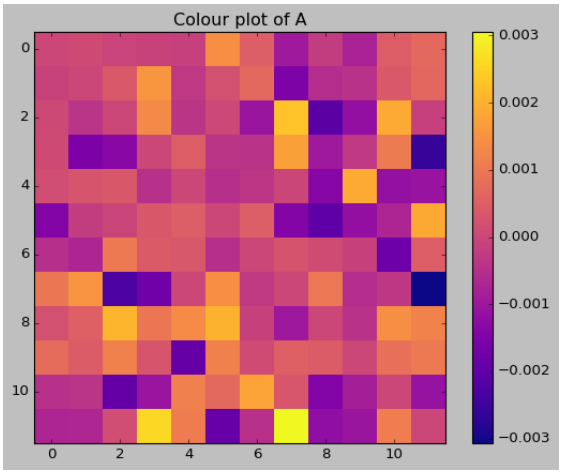

Figure 6: Colour plot of the estimate for matrix A.

5 2D projections with rotational dynamics

We are now able to further decompose the matrix A into rotations in $\frac{M}{2} = 6$ orthogonal

planes. These are formed by taking the real and imaginary parts of eigenvectors, producing two vectors which span this plane. Note this only has to be done for every other eigenvector, as the eigenvalues come in conjugate pairs. Furthermore, we are able to identify the angular velocity of the rotations via the imaginary part of the eigenvalues.

Neural data projected onto the plane with the highest angular velocity is shown in Figure 7. We observe that the lines all roughly start in the same area and then follow smooth arcing trajectories to the endpoints in another separate area; showing that the data is being projected onto a rotating plane. This is also seen in Figure 8. Comparing this to Figure 5, we see that here the trajectories are far more complex; instead of the smooth arc present in Figure 7, we now observe lines that transition between segments that have sharp bends and segments that are only slightly curved.

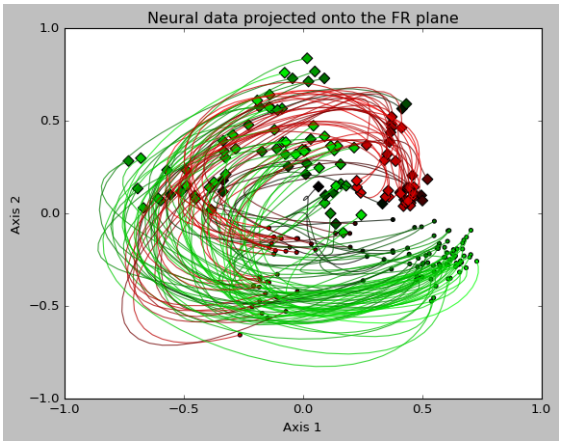

Figure 7: Projection of neural data (-150ms to 200ms) on to the plane with the fastest rotations. Each line represents one of the conditions with a round marker denoting the first time, through to the last time denoted by the diamond marker. Red implies the maximum-spread direction is positive, green is negative and black for close to zero.

From Figure 8, we observe that the neural data projected onto the second fastest plane offers smooth arcing trajectories, like Figure 7, with clear zones for starting and ending points. The location of these points and the direction of the rotation differ, as this is the projection onto a plane created from different eigenvalues.

Notice the clear starting and end zones, as well as the smoothness of the trajectories, are not so well defined in the third fastest plane of rotation. This may well be due to the diminishing value of the rotational speed. For the third fastest plane, the angular velocity is approximately half that of the fastest plane. This large decrease in the magnitude of the eigenvalue indicates that the corresponding eigenvector does worse in capturing and explaining the data as rotational. With a lower variance along this, the rotational plane that is constructed from it is unable to reproduce the same results as shown by the fastest and second fastest planes of rotation.

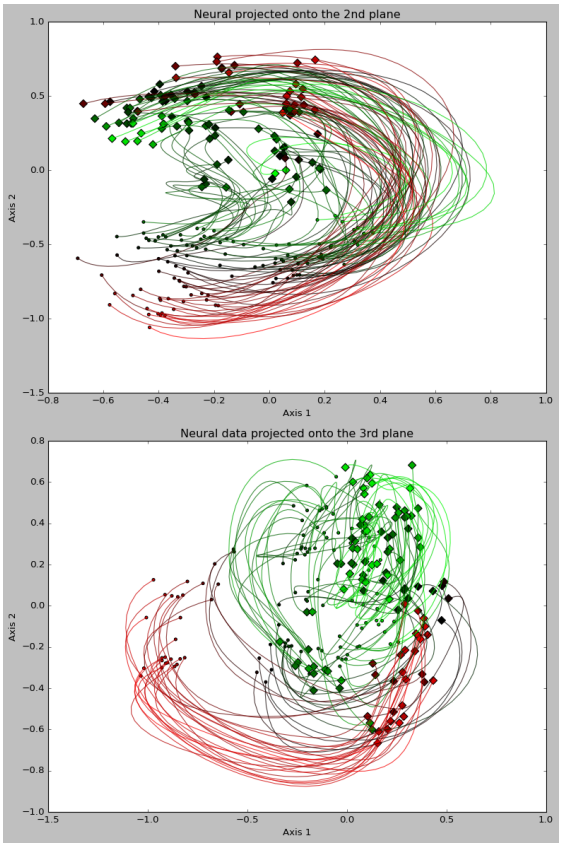

Figure 8: Projection of neural data (-150ms to 200ms) on to the plane with the 2nd (top) and 3rd (bottom) fastest rotations. Each line represents one of the conditions with a round marker denoting the first time, through to the last time denoted by the diamond marker. Red implies the maximum-spread direction is positive, green for negative and black for close to zero.

6 Pre-movement period

Figure 9 plots a greater range of time stamps for the projection of the neural data onto the fastest plane of rotation. Here two sets of colours are used to help us identify the projections from the pre-movement period. The interweaving of cyan and magenta lines clearly indicates that there is little to no rotational data for the projected pre-movement period.

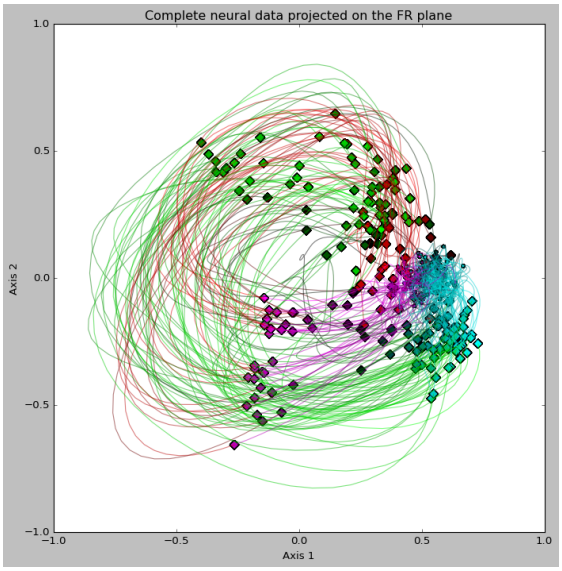

Figure 9: Projection of pre-movement neural data (-800ms to -150ms) shown in cyan and magenta onto the FR plane, along with target onset and movement neural data (-150ms to 300ms) shown in green and red. Each line represents one of the conditions with a round marker denoting the first time, through to the last time denoted by the diamond marker. Red and magenta imply the maximum-spread direction is positive, green and cyan for negative and black for close to zero.

As mentioned in Section 1, in the pre-movement period, the average firing rate of each of the neurons tends to be lower and mostly at the baseline level. Hence it is not surprising that no rotational dynamics are observed in this epoch, as the neurons aren't firing a substantial amount. In fact, this is expected from our model; we describe the animal motor behaviours as inherently repetitive and hence periodic, therefore we model the system's dynamics to be rotational. If our participant is in the rest state, with no periodic motor behaviours, then the data in this period will not possess any rotational dynamics. This explains why the data in this epoch is all 'bundled' tightly in one area. Then once the monkey observes the target, and begins activating neurons as it plans its trajectories, as discussed in Section 1, the neural data represents a movement which is generally repetitive and therefore periodic, resulting in the presence of rotational dynamics in the data, which is shown clearly in Figure 9.

7 Control analysis

It is possible the model may 'over-fit' and exhibit rotational dynamics that aren't actually present in the neural data.

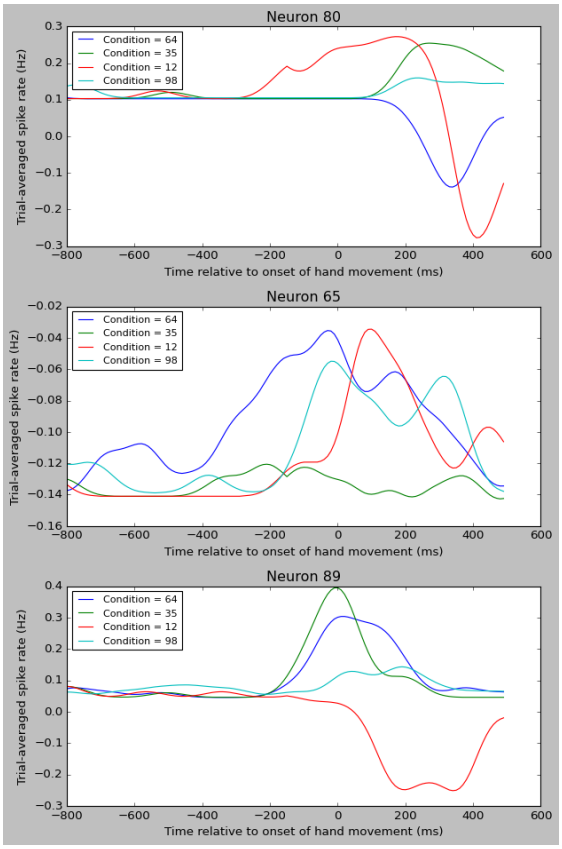

Figure 10: Distorted Neuron firing rate plotted against time relative to the onset of hand movement (distorted version of Fig 4). Averaged across all conditions and neurons.

In order to test whether this is happening, we distort the neural data pass and proceed

with the same steps as we did with the original data. The distortion carried out used $x[t_{0:T}] = 2 \times x[t_0] - x[t_{0:T}]$ where t_0 represents the $-150ms$ bin. This is done for a random selection of half the conditions for all the neurons independently. A sample of pre-processed neurons under a sample of conditions is given in Figure 10.

After distorting the data we expect the data to still be smooth, as well as its derivatives, as we are not shuffling the time bins, which would allow sharp, instantaneous jumps in the average spike count to be present. After pre-processing the mean and range of values will be the same, due to the common standardisation procedure, although the principal components that explain the most variance will be different, and explain less due to the random shuffling. Hence, we expect the characterising feature of autonomous dynamical systems, that is the instantaneous state of change is always the same, to be broken as we have now fundamentally deformed the neural activity. Therefore we cannot expect the data to follow the same dynamics. So, if the model is not 'over-fitting' we expect the rotational dynamics to not be present; if 'over-fitting' is present, then the data will still display rotational dynamics on the rotational plane projection plots, despite the data containing no inherent rational dynamics.

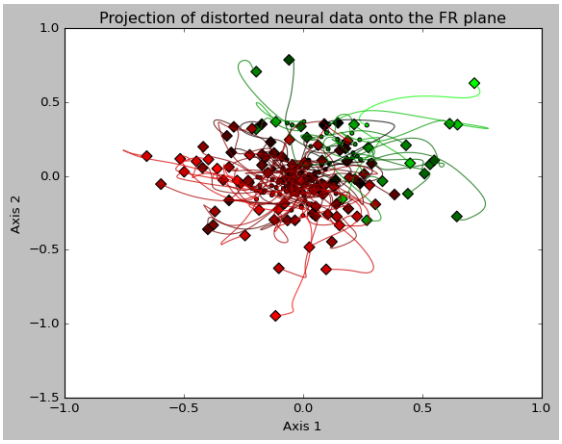

Figure 11: Projection of distorted neural data (-150ms to 200ms) on to the plane with the fastest rotations. Each line represents one of the conditions with a round marker denoting the first time, through to the last time denoted by the diamond marker. Red implies the maximum-spread direction is positive, green for negative, and black for close to zero.

As we can see in Figure 11, no clear arcing trajectory is displayed between two starting and ending zones, unlike what is seen in Figure 7. This shows that indeed no rotational dynamics are present within this distorted dataset, implying that the model is not 'over-fitting'. Moving forward we can be confident that any rotational dynamics presented by our projections are certainly present within the data and not a result of 'over-fitting' by the model.

8 Appendix

The full code, excluding the module for colouring the plots, for each of the above sections, is given below as a PDF export of a Python Jupyter Notebook.

```
#%%
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
style.use('classic')
from sklearn.decomposition import PCA
from cond_color import get_colors, plot_start, plot_end
#%%
np.random.seed(41)
#%% md
Loading Data
#%%
data_orig = np.load('psths.npz')
X, times = data_orig['X'], data_orig['times']
#%% md
# Exercise 1: Plotting raw PSTHs
#%%
def plot_psths(data, timeintervals, n_rows, n_cols, n_cond):
    np.random.seed(41)
    cond = np.random.randint(low=data.shape[1], size=n_cond)

    fig, ax = plt.subplots(n_rows, n_cols, tight_layout=True, figsize=(8,12))
    for row in ax:
        n = np.random.randint(low=data.shape[0], size=1)[0]
        for c in cond:
            row.plot(timeintervals, data[n, c,:], label = 'Condition = '+str(c))
            row.set_title(f'Neuron {n}')
            row.set_xlabel('Time relative to onset of hand movement')
            row.set_ylabel('Trial-averaged spike rate (Hz)')
            row.legend(loc='upper left', prop={'size': 10})
    return
#%%
```

```
plot_psths(X, times, 3, 1, 4)
#%%
pop_mean = data_orig['X'].mean(axis=(0,1))
plt.plot(times, pop_mean)
plt.title('Original Population')
plt.xlabel('Time relative to onset of hand movement (ms)')
plt.ylabel('Population-averaged firing rate (Hz)')

plt.show()
#%% md
# Exercise 2: Pre-processing
#%% md
## Normalisation
#%%
def plot_max_hist(data):
    plt.hist(data.max(axis=(1,2)), bins=20)
    plt.title('Histogram of neuron maximum values')
    plt.ylabel('Frequency')
    plt.xlabel('Max value for a given neuron across time and conditions')

    plt.show()
    return
#%%
plot_max_hist(X)
#%%
def norm_data(data):
    a, b = data.max(axis=(1,2)), data.min(axis=(1,2))
    a, b = a.reshape(data.shape[0],1,1), b.reshape(182,1,1)
    data_normed = (data - b) / (a - b + 5 )

    return data_normed
#%%
X_norm = norm_data(X)
#%%
plot_psths(X_norm, times, 2, 1, 4)
#%% md
## Mean centering
#%%
```

```
 71 def center_data(data):
 72     mean = data.mean(axis=(0,2))
 73     mean = mean.reshape(1, 108, 1)
 74     data_centered = data - mean
 75     return data_centered
 76 #%%
 77 X_norm_mean = center_data(X_norm)
 78 #%%
 79 plot_psths(X_norm_mean, times, 3, 1, 4)
 80 #%%
 81 pop_mean = X_norm_mean.mean(axis=(0,1))
 82 plt.plot(times, pop_mean)
 83 plt.title('Pre-processed Population')
 84 plt.xlabel('Time relative to onset of hand movement
     (ms)')
 85 plt.ylabel('Population-averaged firing rate (Hz)')
 86
 87 plt.show()
 88 #%% md
 89 ## Dimensionality reduction by PCA
 90 #%%
 91 # times = times[65:111]
 92 #%%
 93 X_trunc = X_norm_mean[:,:,65:111]
 94 X = X_trunc.reshape(X_trunc.shape[0], X_trunc.shape[
    1]*X_trunc.shape[2])
 95 #%%
 96 pca = PCA(n_components=12)
 97 pca.fit(X.T)
 98 Z = pca.transform(X.T).T
 99 #%% md
100 # Plotting PC space trajectorie
101 #%%
102 Z = Z.reshape(12, X_trunc.shape[1], X_trunc.shape[2
    ])
103 #%%
104 def plot_pca_psths(data1, pca_axis_1, pca_axis_2,
    data2=None, alpha=1):
105     np.random.seed(41)
106     pca_axis_1 -= 1
107     pca_axis_2 -= 1
```

```
    alt_colors = [False, True]

    for index, data in enumerate([data1, data2]):
        if data is not None:
            colors = get_colors(data[pca_axis_1, :, 0], data[pca_axis_2, :, 0], alt_colors[index//2])
            for cond in range(0, data.shape[1]):
                plt.plot(data[pca_axis_1, cond, :], data[pca_axis_2, cond, :], label = 'C = '+str(cond), color=colors[cond], alpha=alpha)
                plt.title(f'PCA plot')
                plt.xlabel(f'PC - {pca_axis_1 + 1}')
                plt.ylabel(f'PC - {pca_axis_2 + 1}')
            plot_start(data[pca_axis_1, :, 0], data[pca_axis_2, :, 0], colors, markersize=50, ax=None)
            plot_end(data[pca_axis_1, :, -1], data[pca_axis_2, :, -1], colors, markersize=50, ax=None)
        else:
            pass
    plt.show()
    return
#%%
plot_pca_psths(Z, 1, 2, alpha=0.75)
#%%
plot_pca_psths(Z, 1, 3, alpha=0.75)
#%% md
# Exercise 4: Finding the max. likelihood estimate for A
#%% md
## Log-likelihood and its (naive) gradient
#%% md
$\Delta z_{t+1} = Az_{t} + \sigma \epsilon_t$ where $\sigma=1$
#%% md
$\Delta z_{t+1} = N(Az_{t},I_{12\times12})$
#%% md
$P(\Delta z_{1:T-1} | z_{0:T-1}) = \prod^T_{t=0}N(Az_{t},I_{12\times12})$
#%% md
$log(P(\Delta z_{1:T} | z_{0:T-1})) = \sum^T_{t=0} log(N(Az_{t},I_{12\times12}))$
```

```
#%% md
$log(P(\Delta z_{1:T} | z_{0:T-1})) = \sum^T_{t=0} log(\exp(-(\Delta z_{t+1} - Az_{t})^TI_{12\times12}(\Delta z_{t+1} - Az_{t}))) + const$
#%% md
$log(P(\Delta z_{1:T} | z_{0:T-1})) = -\sum^T_{t=0} (\Delta z_{t+1} - Az_{t})^TI_{12\times12}(\Delta z_{t+1} - Az_{t}) + const$
#%% md
$log(P(\Delta Z | Z)) \approx -(\Delta Z - AZ)^T(\Delta Z - AZ) \approx  - Z^TA^TAZ + 2\Delta Z^T A Z$
#%% md
$\frac{d}{dA}(log(P(\Delta Z | Z))) = -2 AZZ^T + 2\Delta Z Z^T$
#%% md
##  Parametrising an antisymmetric
#%% md
K will equal 6, the number of matrix entries above the diagonal
#%% md
$M \times M = 2K + M$
#%% md
$K = \frac{M(M-1)}{2}$
#%%
beta = np.array([[0.0001, 1, 1, 0.0001, 1, 1]])

def create_h(m):
    k = int( (m**2 - m)/2)
    h = np.zeros((k, m, m))
    row, column = 0, 1
    for i in range(0, k):
        h[i][int(row)][int(column)], h[i][int(column)][int(row)] = 1, -1
        column += 1
        if column >= m:
            row += 1
            column = row + 1
    return h

H = create_h(4)
print(beta.shape)
```

```
print(H.shape)
print(H)

A = np.tensordot(beta, H, axes=1)
#%% md
# Gradient with respect to β
#%% md
$log(P(\Delta Z | Z)) \approx  - (\sum^K_{a=1}\
beta_a \sum^M_{j=1}H_{a,i,j}Z_{j,n})^T(\sum^K_{a=1}\
beta_a\sum^M_{j=1}H_{a,i,j}Z_{j,n}) + 2\Delta Z^T (\
sum^K_{a=1}\beta_a\sum^M_{j=1}H_{a,i,j}Z_{j,n})$
#%% md
$log(P(\Delta Z | Z)) \approx  - (\sum^K_{a=1}\
beta_a W_{a,i,n})^T(\sum^K_{a=1}\beta_aW_{a,i,n}) +
2\Delta Z^T (\sum^K_{a=1}\beta_aW_{a,i,n}) = - (\
beta W)^T(\beta W) + 2\Delta Z^T (\beta W)$
#%% md
$\frac{d}{d\beta}(log(P(\Delta Z | Z))) \approx - 2\
beta W^TW + 2\Delta Z^T W$
#%% md
$Q = W^TW$ and $b = \Delta Z^T W$
#%% md
## An antisymmetric estimate for A
#%% md
Solve $ \beta = b Q^{-1}$
#%%
def a_estimate(z):
    m = z.shape[0]
    z_plus, z_  = z[:,:,1:], z[:,:,:-1]
    z_plus = z_plus.reshape((m, z_plus.shape[1]*
z_plus.shape[2]))
    z_ = z_.reshape((m, z_.shape[1]*z_.shape[2]))

    h = create_h(z_.shape[0])
    w = np.tensordot(h, z_, axes=1)
    q = np.tensordot(w, w, axes=([1,2],[1,2]))

    delta_z = z_plus - z_
    b = np.tensordot(delta_z, w, axes=([0,1],[1,2]))
    b = b.reshape((1,66))

```

```
206     beta = b @ np.linalg.inv(q)
207     a = np.tensordot(beta, h, axes=([1],[0]))
208     a = a.reshape((m, m))
209     return beta, a
210 #%%
211 beta, A = a_estimate(Z)
212 #%%
213 img = plt.imshow(A, interpolation ='nearest', cmap='
    plasma' )
214 plt.colorbar(img)
215 plt.title('Colour plot of A')
216 plt.show()
217 #%% md
218 ## Test
219 #%%
220 test = np.load('test.npz')
221 Z_test = test['Z_test']
222 A_test = test['A_test']
223 #%%
224 beta_test_estimated, A_test_estimated = a_estimate(
    Z_test)
225 #%%
226 A_inaccuracy = (A_test_estimated - A_test)
227 print(A_inaccuracy.max())
228 #%% md
229 # Exercise 5: 2D projections with rotational
    dynamics
230 #%% md
231 ## A) Eigenvalues and Eigenvectors of A
232 #%%
233 A_evalue, A_evector = np.linalg.eig(A)
234 print(A_evalue)
235 #%% md
236 ## B)
237 #%%
238 def get_p(eigen_vectors, plane):
239     evector_real = eigen_vectors[:,plane].real
240     evector_imag = eigen_vectors[:,plane].imag
241     p = np.zeros((2, 12))
242     p[0, :] = evector_real/np.linalg.norm(
    evector_real)
```

```
    p[1, :] = evector_imag/np.linalg.norm(evector_imag)
    print(p[0, :].T @ p[1, :])
    return p
#%%
P_FR = get_p(A_evector, 0)
#%% md
## C)
#%%
def plot_proj_psths(data1, title, data2=None, alpha=1):
    np.random.seed(41)
    alt_colors = [False, True]

    for index, data in enumerate([data1, data2]):
        if data is not None:
            colors = get_colors(data[0, :, 0], data[1, :, 0], alt_colors[index//2])
            for cond in range(0, data.shape[1]):
                plt.plot(data[0, cond, :], data[1, cond, :], label = 'C = '+str(cond), color=colors[cond], alpha=alpha)
                plt.title(title)
                plt.xlabel(f'Axis 1')
                plt.ylabel(f'Axis 2')
            plot_start(data[0, :, 0], data[1, :, 0], colors, markersize=50, ax=None)
            plot_end(data[0, :, -1], data[1, :, -1], colors, markersize=50, ax=None)
        else:
            pass
    plt.show()
    return
#%%
Projection_FR = np.tensordot(P_FR, Z, axes=([1],[0]))
#%%
plot_proj_psths(Projection_FR[:, :, 0:36], 'Z projected onto the FR plane', alpha=0.75)
#%% md
## D)
```

```
#%%
P_2 = get_p(A_evector, 3)
Projection_2 = np.tensordot(P_2, Z, axes=([1], [0]))
P_3 = get_p(A_evector, 5)
Projection_3 = np.tensordot(P_3, Z, axes=([1], [0]))
#%%
fig, ax = plt.subplots(2, 1, tight_layout=True, figsize=(10,15))
data1 = Projection_2[:, :, 0:36]
data2 = Projection_3[:, :, 0:36]
colors = get_colors(data1[0, :, 0], data1[1, :, 0])
for cond in range(0, data1.shape[1]):
    ax[0].plot(data1[0, cond, :], data1[1, cond, :], label = 'C = '+str(cond), color=colors[cond], alpha=0.75)
    ax[0].set_title('Z projected onto the 2nd plane')
    ax[0].set_xlabel(f'Axis 1')
    ax[0].set_ylabel(f'Axis 2')
plot_start(data1[0, :, 0], data1[1, :, 0], colors, markersize=50, ax=ax[0])
plot_end(data1[0, :, -1], data1[1, :, -1], colors, markersize=50, ax=ax[0])

colors = get_colors(data2[0, :, 0], data2[1, :, 0])
for cond in range(0, data2.shape[1]):
    ax[1].plot(data2[0, cond, :], data2[1, cond, :], label = 'C = '+str(cond), color=colors[cond], alpha=0.75)
    ax[1].set_title('Z projected onto the 3rd plane')
    ax[1].set_xlabel(f'Axis 1')
    ax[1].set_ylabel(f'Axis 2')
plot_start(data2[0, :, 0], data2[1, :, 0], colors, markersize=50, ax=ax[1])
plot_end(data2[0, :, -1], data2[1, :, -1], colors, markersize=50, ax=ax[1])
#%% md
# Exercise 6: Pre-movement period
#%%
X_pre = X_norm_mean[:, :, :66]
```

```
X = X_pre.reshape(X_pre.shape[0], X_pre.shape[1] * X_pre.shape[2])
Z_pre = pca.transform(X.T).T
Z_pre = Z_pre.reshape(12 , X_pre.shape[1], X_pre.shape[2])
Projection_FR_pre = np.tensordot(P_FR, Z_pre, axes=([1], [0]))
#%%
plt.figure(figsize=(10, 10))
data = Projection_FR
colors = get_colors(data[0, :, 0], data[1, :, 0], alt_colors=False)
for cond in range(0, data.shape[1]):
    plt.plot(data[0, cond, :], data[1, cond, :], label = 'C = '+str(cond), color=colors[cond], alpha=0.5)
    plot_start(data[0, :, 0], data[1, :, 0], colors, markersize=50, ax=None)
    plot_end(data[0, :, -1], data[1, :, -1], colors, markersize=50, ax=None)

data = Projection_FR_pre
colors = get_colors(data[0, :, -1], data[1, :, -1], alt_colors=True)
for cond in range(0, data.shape[1]):
    plt.plot(data[0, cond, :], data[1, cond, :], label = 'C = '+str(cond), color=colors[cond], alpha=0.5)
    plot_start(data[0, :, 0], data[1, :, 0], colors, markersize=50, ax=None)
    plot_end(data[0, :, -1], data[1, :, -1], colors, markersize=50, ax=None)

plt.title('Z (-800ms to 300ms) projected on the FR plane')
plt.xlabel(f'Axis 1')
plt.ylabel(f'Axis 2')

plt.show()
#%% md
# Exercise 7: Control Analysis
```

```
#%%
## Load data
X_seven, times = data_orig['X'], data_orig['times']
#%%
## Plot pre-distortion
plot_psths(X_seven, times, 2, 1, 4)
#%%
## Distortion
for N in range(X_seven.shape[0]):
    conditions = np.random.choice(108, (108//2,), replace=False)
    X_seven[N, conditions, 65:] = 2*X_seven[N, conditions, 65].reshape((54,1))- X_seven[N, conditions, 65:]
#%%
## Normalising
X_seven_norm = norm_data(X_seven)
#%%
## Mean centering
X_seven_norm_mean = center_data(X_seven_norm)
plot_psths(X_seven_norm_mean, times, 3, 1, 4)
#%%
## Dimensionality reduction by PCA
X_seven_trunc = X_seven_norm_mean[:, :, 65:111]
X_seven = X_seven_trunc.reshape(X_seven_trunc.shape[0], X_seven_trunc.shape[1] * X_seven_trunc.shape[2])
pca_seven = PCA(n_components=12)
pca_seven.fit(X_seven.T)
Z = pca_seven.transform(X_seven.T).T
#%%
Z = Z.reshape(12, X_trunc.shape[1], X_trunc.shape[2])
plot_pca_psths(Z, 1, 2)
#%%
# Exercise 4
beta, A = a_estimate(Z)
img = plt.imshow(A, interpolation ='nearest', cmap='plasma' )
plt.colorbar(img)
plt.title('Colour plot of A')
plt.show()
```

```
367 #%%
368 ## A)
369 A_evalue, A_evector = np.linalg.eig(A)
370 
371 ## B)
372 P_FR = get_p(A_evector, 0)
373 
374 ## C)
375 Projection_FR = np.tensordot(P_FR, Z, axes=([1], [0
    ]))
376 plot_proj_psths(Projection_FR[:, :, 0:36], '
    Projection of disorted Z onto the FR plane', alpha=0
    .75)
377 #%%
378 
```