Question 1. Linear Neural Field.

1.1 We are given the following linear neural field

$$\tau\dot{u}(x,t) = -u(x,t) + \int_{-\infty}^{\infty} w(x-x')u(x',t)dx' + s(x,t) \tag{1}$$

We assume that the input signal is constant over time and is given by

$$s(x) = \exp\left(-\frac{x^2}{4d^2}\right)/(2d\sqrt{\pi}) \tag{2}$$

and that the interaction kernel is given by the Gabor funciton

$$w(x) = a\left(\exp\left(-\frac{x^2}{4b^2}\right)\cos(k_0x)\right)/(b\sqrt{\pi}) \tag{3}$$

Note that equation 1 is a partial integro-differential equation. To solve it, we begin by transforming it in x to the frequency domain using the Fourier Transform $\mathcal{F}$

$$\tau\frac{d\widetilde{u}(\omega,t)}{dt} = -\widetilde{u}(\omega,t) + \widetilde{w}(\omega)\widetilde{u}(\omega,t) + \widetilde{s}(\omega)$$

and after rearranging we get

$$\tau\frac{d\widetilde{u}(\omega,t)}{dt} = (-1+\widetilde{w}(\omega))\widetilde{u}(\omega,t) + \widetilde{s}(\omega)$$

This is now a linear inhomogeneous ODE in the Fourier domain. If we further assume that the system has a stable solution that does not depend on time, we get

$$\frac{d\widetilde{u}(\omega)}{dt} = 0 \implies (-1+\widetilde{w}(\omega))\widetilde{u}(\omega) + \widetilde{s}(\omega) = 0$$

Therefore, we obtain the solution in the Fourier domain

$$\widetilde{u}(\omega,\infty) = \frac{\widetilde{s}(\omega)}{1-\widetilde{w}(\omega)}$$

1.2 Note that both the interaction kernel and input terms are Gaussians, and hence their Fourier Transforms are

$$\widetilde{s}(\omega) = \mathcal{F}[s](\omega) = \frac{\exp(-1/4d^2\omega^2)}{2\sqrt{2\pi}}$$

$$\begin{aligned}\widetilde{w}(\omega) = \mathcal{F}[w](\omega) &= \frac{a}{b\sqrt{\pi}}\int_{-\infty}^{\infty}\exp\left(-\frac{x^2}{4b^2}\right)\cos(k_0x)e^{-i\omega x}dx \\ &= \frac{a}{2b\sqrt{\pi}}\int_{-\infty}^{\infty}\exp\left(-\frac{x^2}{4b^2}\right)(e^{ik_0x}+e^{-ik_0x})e^{-i\omega x}dx \\ &= \frac{a}{2b\sqrt{\pi}}\left(\exp\left(-\frac{(x-k_0)^2}{4b^2}\right)+\exp\left(-\frac{(x+k_0)^2}{4b^2}\right)\right)\end{aligned}$$

1.3 As can be seen for the kernel in the frequency domain, parameter a controls the power at frequency k_0. Thus, in the original spatial domain, the a parameter controls the interaction strength. Plots of the kernel both in spatial and frequency domains are shown below.

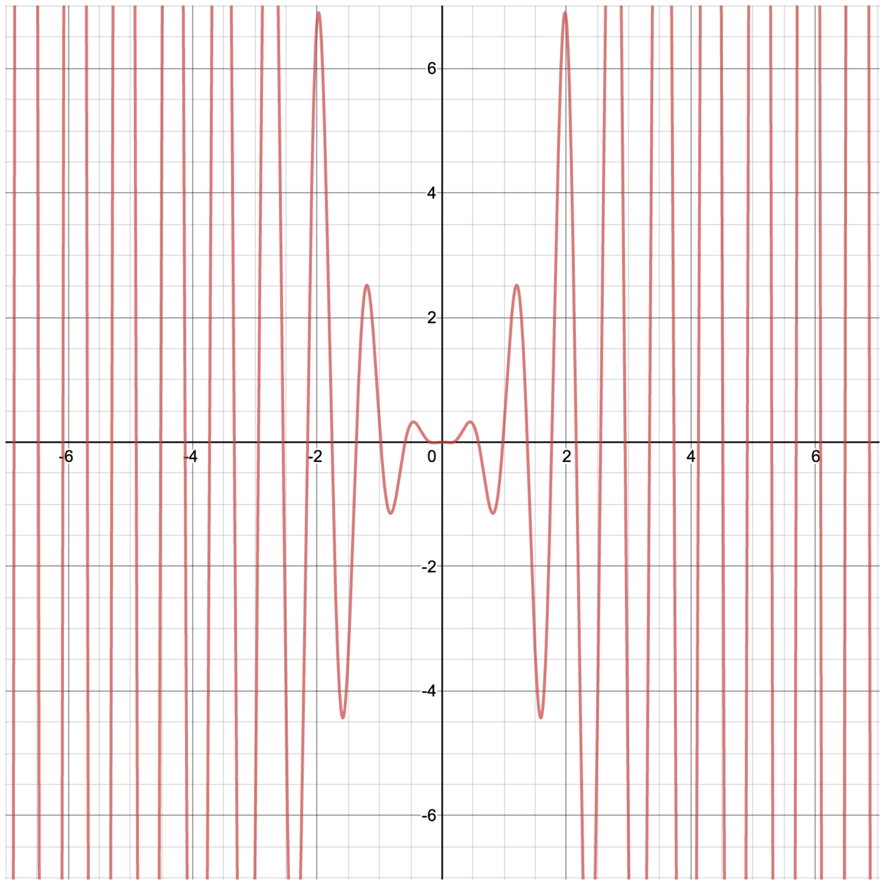

Figure 1: Interaction kernel in the spatial domain for $k = 8$ and $a = 1$.

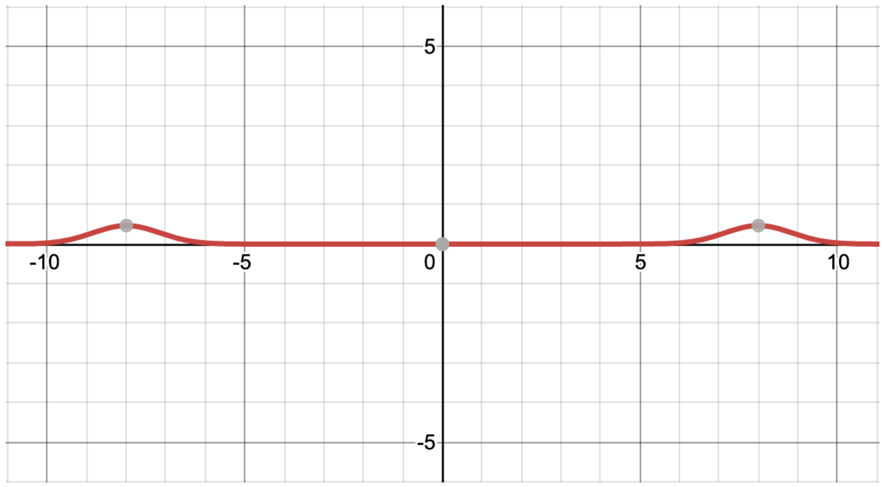

Figure 2: Interaction kernel in the frequency domain for $k = 8$ and $a = 1$.

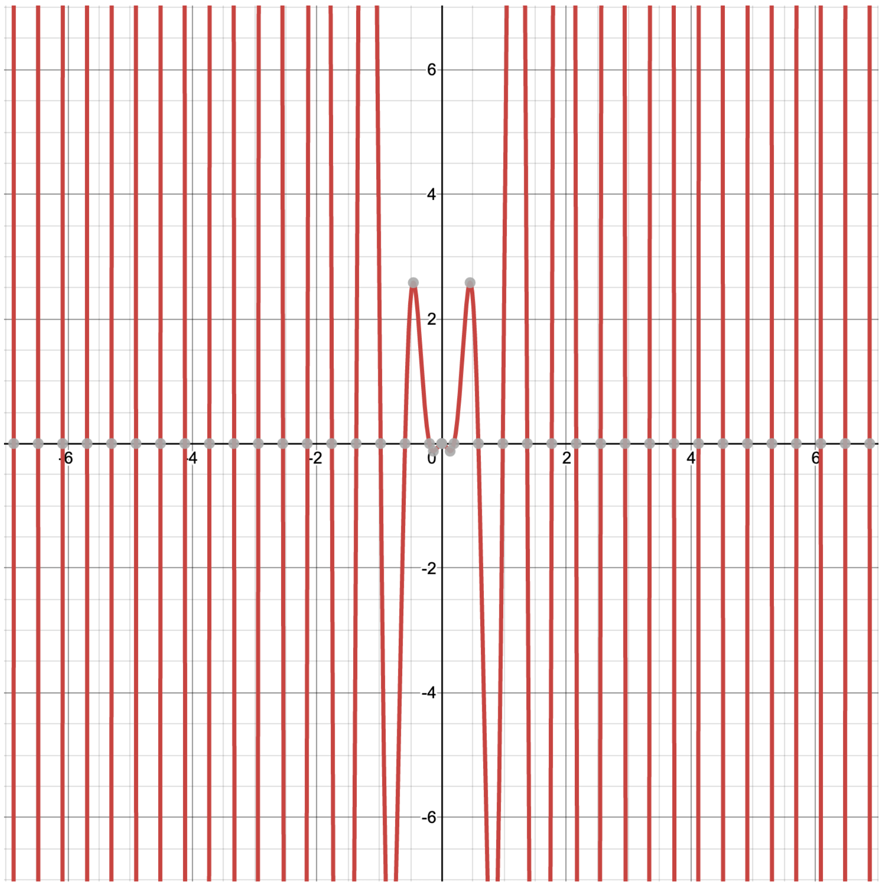

Figure 3: Interaction kernel in the spatial domain for $k = 8$ and $a = 8$.

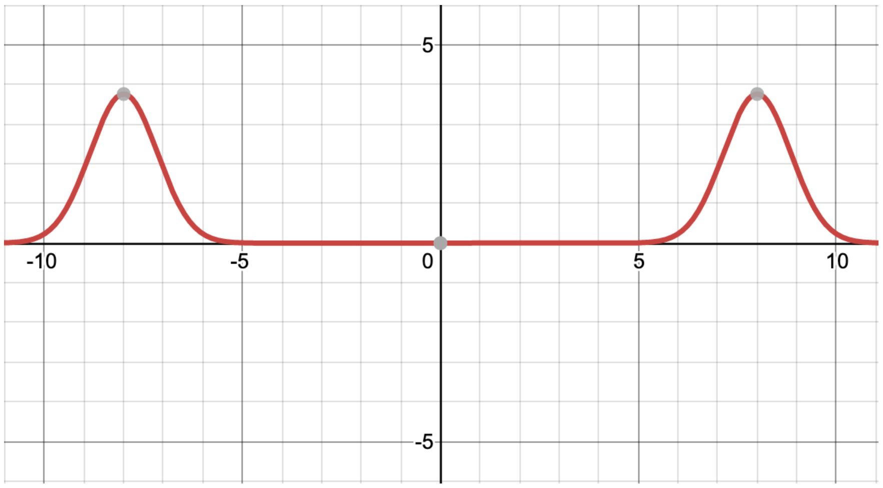

Figure 4: Interaction kernel in the frequency domain for $k = 8$ and $a = 8$.

Therefore, for a approaching 1 from below we would expect an increasing effect from nearby neurons in the field.

1.4 Now, we simulate the neural field equation by approximating the integral as a Riemann sum

$$\int_A^B f(x)dx \approx \sum_{i=0}^{N} f(x_i)\Delta x$$

with $x_i = A + i\Delta x$ and $\Delta x = (B - A)/N$. Thus, we now have

$$\tau\dot{u}(x,t) = -u(x,t) + \sum_{i=0}^{N} \big[w(x - x_i)u(x_i,t)\Delta x\big] + s(x,t)$$

We also need to make use of the Forward Euler method to approximate the derivative. Therefore, we finally obtain

$$u(x, t + \Delta t) = u(x,t) + \frac{\Delta t}{\tau}\left(-u(x,t) + \sum_{i=0}^{N} \big[w(x - x_i)u(x_i,t)\Delta x\big] + s(x,t)\right)$$

The simulation parameters are as follows

- $A = 10$
- $B = -10$
- $N \geq 200$
- $\tau = 10$
- $a = 1$
- $b = 0.6$
- $d = 2$
- $k_0 = 4$

The result of the simulation appears in Figure 5. We can see a stationary solution emerging as a localised peak of activity centred at $x = 0$.

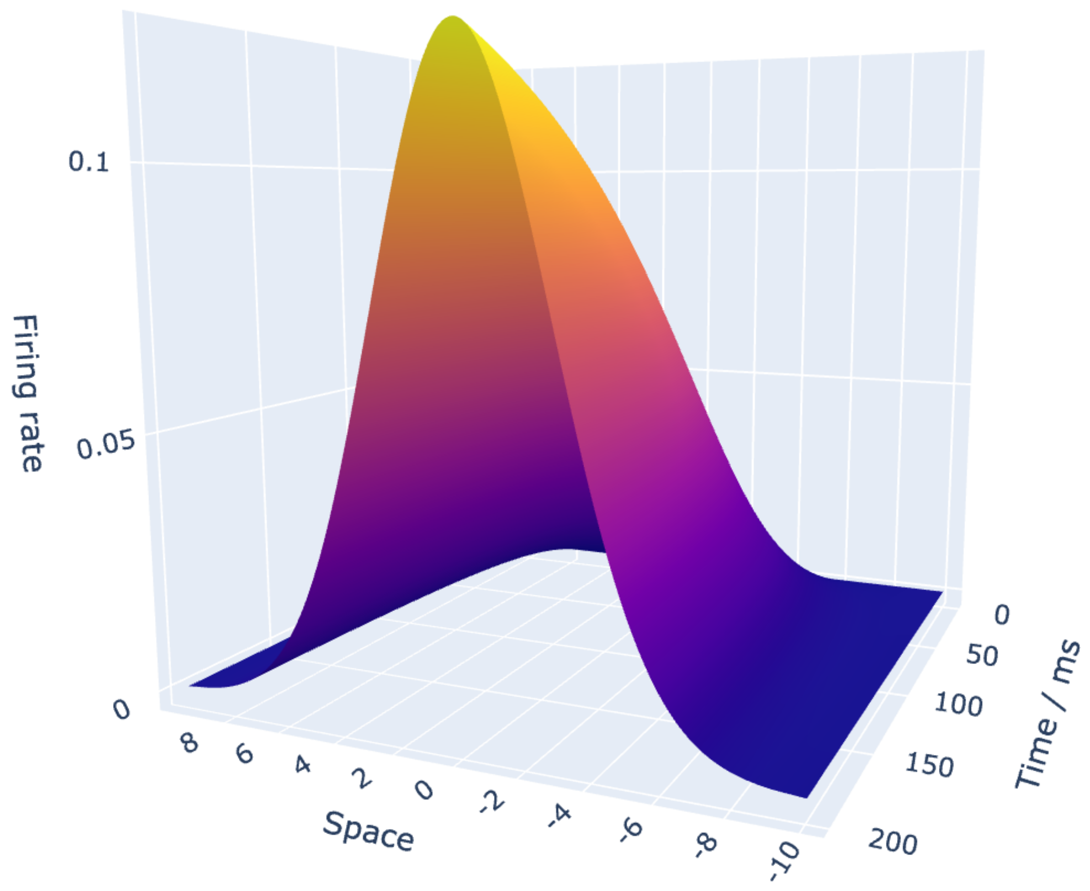

Figure 5: Dynamics of the linear neural field defined by equation 1.

We can also add a small amount of Gaussian noise to the input signal. This result is shown in Figure 6 for $\sigma^2(s(x)) = 0.01^2$. With the noise added, the solution remains stable albeit with minor perturbations.

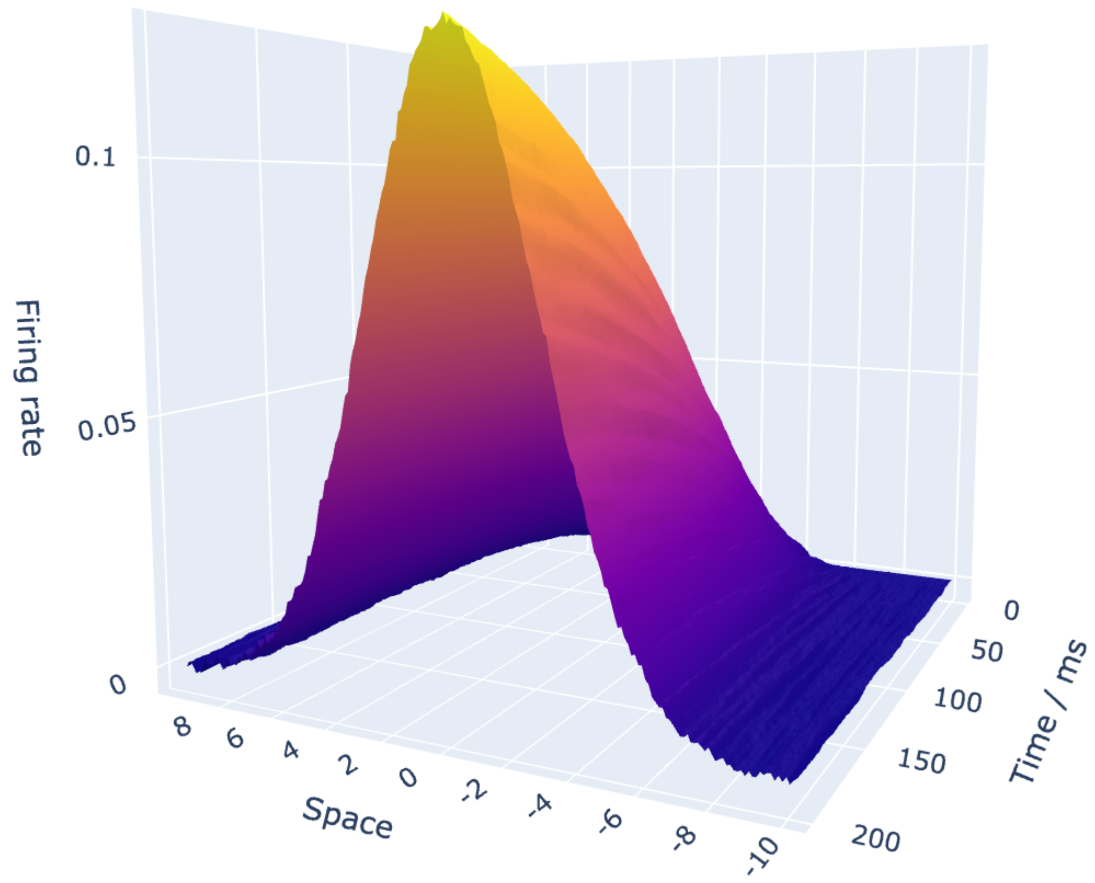

Figure 6: Same as in Fig. 1 but with added Gaussian noise.

Now, let us change the interaction kernel parameter a keeping the same noise level as before. The results for $a = 0.7$ and $a = 1.5$ appear in Figures 7 & 8, repsectively. It can be clearly seen that the a parameter influences the system's sensitivity to noise. With a higher value of a, the solution looks noisier.

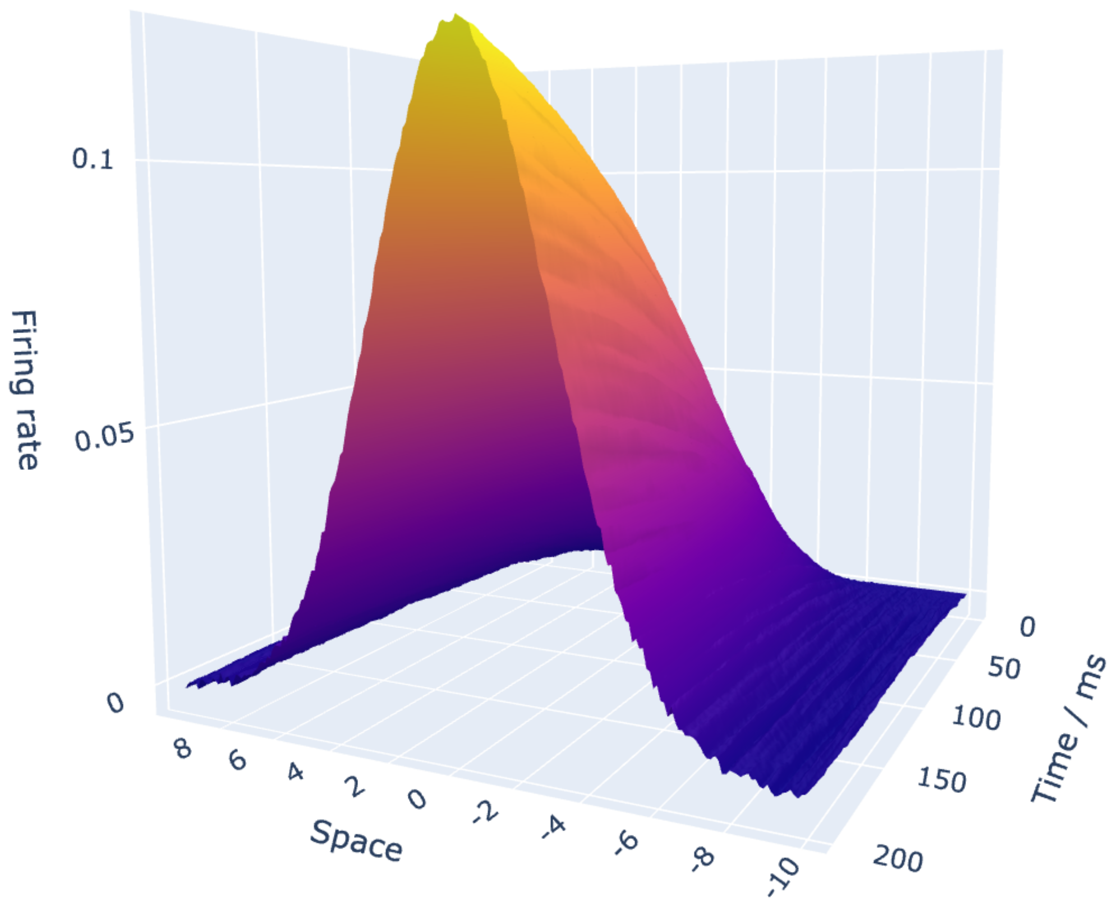

Figure 7: Same as in Fig. 1 but with added Gaussian noise and $a = 0.7$.

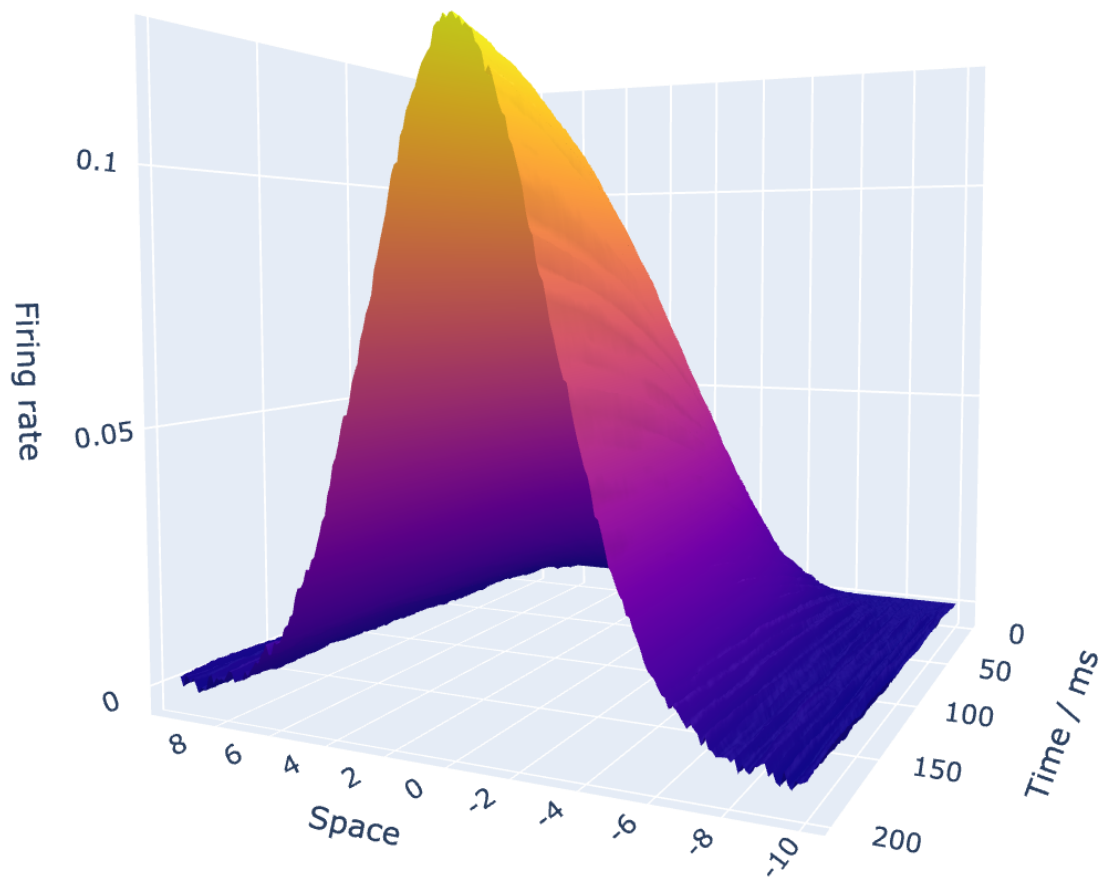

Figure 8: Same as in Fig. 1 but with added Gaussian noise and $a = 1.5$.

Finally, let us discard the noise and keep $a = 1$ but now we set $k = 8$. This result is shown in figure 9. A higher value of k stretches the solution upwards. With even increasing k, the peak becomes higher and boundaries get narrower. This is because k controls the kernel width.

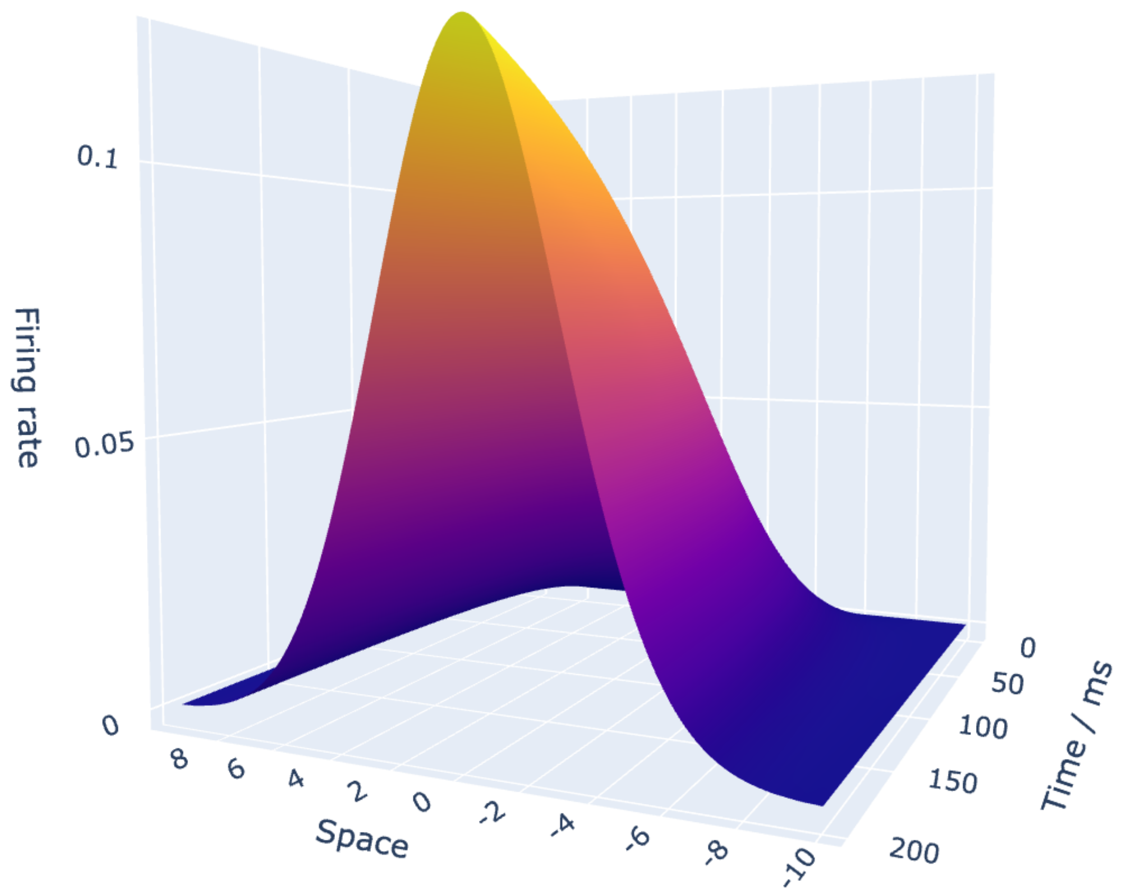

Figure 9: Same as in Fig. 1 but with $k = 8$.

1.5 We can define a Green's function $g(x,t)$ for the neural field that describes its response to a delta input signal of the form $s(x,t) = \delta(x)\delta(t)$. If the function is know, then the response of the filed to $s(x,t)$ is characterised by

$$u(x,t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x-x', t-t')s(x',t')dx'dt'$$

Note that we can apply a 2D Fourier Transform to the above equation (i.e. transform both space and time) to obtain $\mathcal{F}[g](k,\omega)$. First, we transorm the spatial domain (using the convolution theorem)

$$\widetilde{u}(k,t) = \int_{-\infty}^{\infty} \widetilde{g}(k,t-t')\widetilde{s}(k,t')dt'$$

Now, we transform the temporal domain, again applying the convolution theorem

$$\mathcal{F}[\tilde{u}](k,\omega) = \mathcal{F}[\tilde{g}](k,\omega)\mathcal{F}[\tilde{s}](k,\omega)$$

Hence, the 2D Fourier Transform of the Green's function of our neural field is

$$\mathcal{F}[\tilde{g}](k,\omega) = \frac{\mathcal{F}[\tilde{u}](k,\omega)}{\mathcal{F}[\tilde{s}](k,\omega)}$$

1.6 We now assume a different interaction kernel $w(x)$ given by

$$w(x) = e^{-c|x|}\text{sign}(x)$$

where $c > 0$ and a time-dependent stimulus of the form

$$s(x,t) = c\exp\left(-\frac{(x-vt)^2}{4d_1^2}\right)/(2d_1\sqrt{\pi})$$

where v is the stimulus peak travelling speed. We run the simulation with the following parameters

- $c = 1$
- $d_1 = 0.5$
- $v = 0.1$

The result is reported in Figure 10.

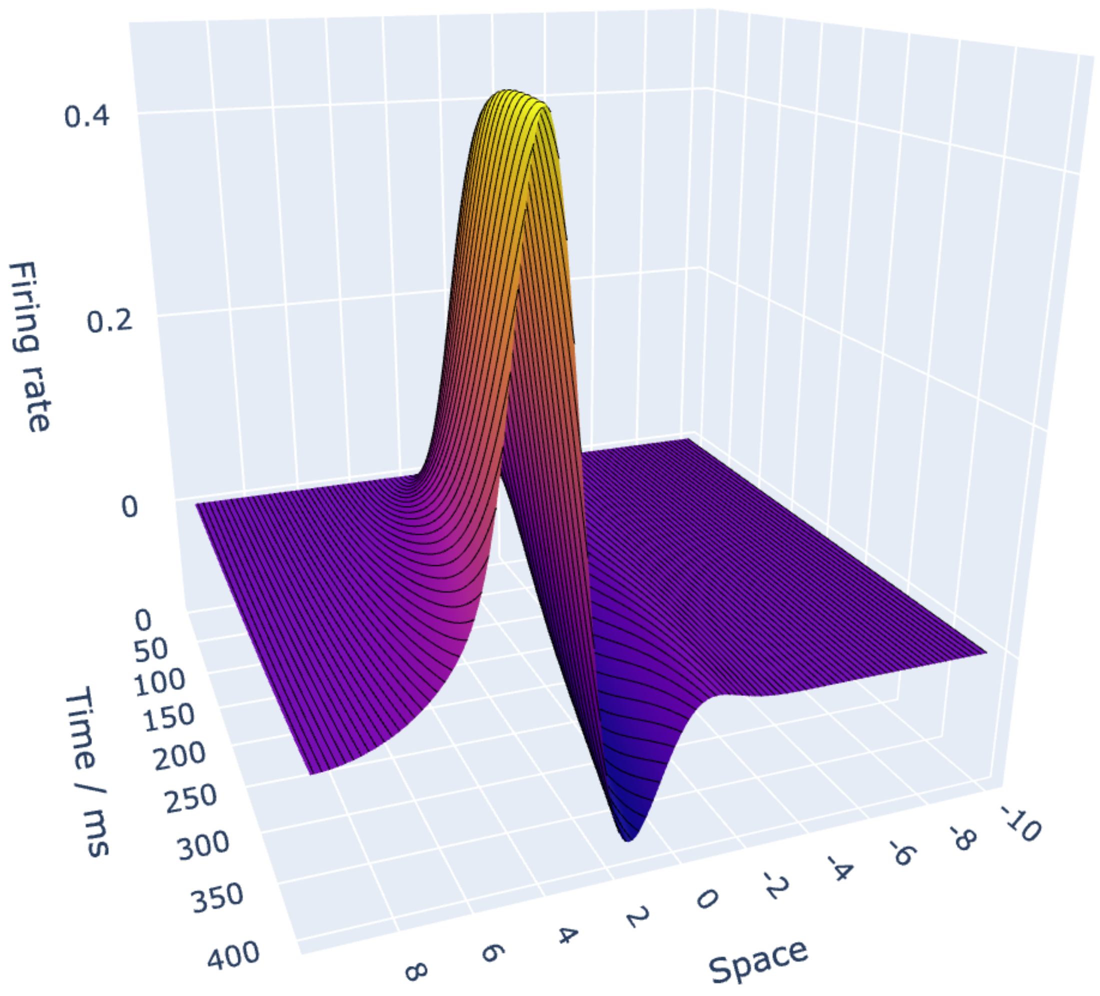

Figure 10: Linear neural field dynamics with different interaction kernel and input.

Now, if we change the stimulus travelling speed to $v = -0.1$, we yeild what is shown in Figure 11.

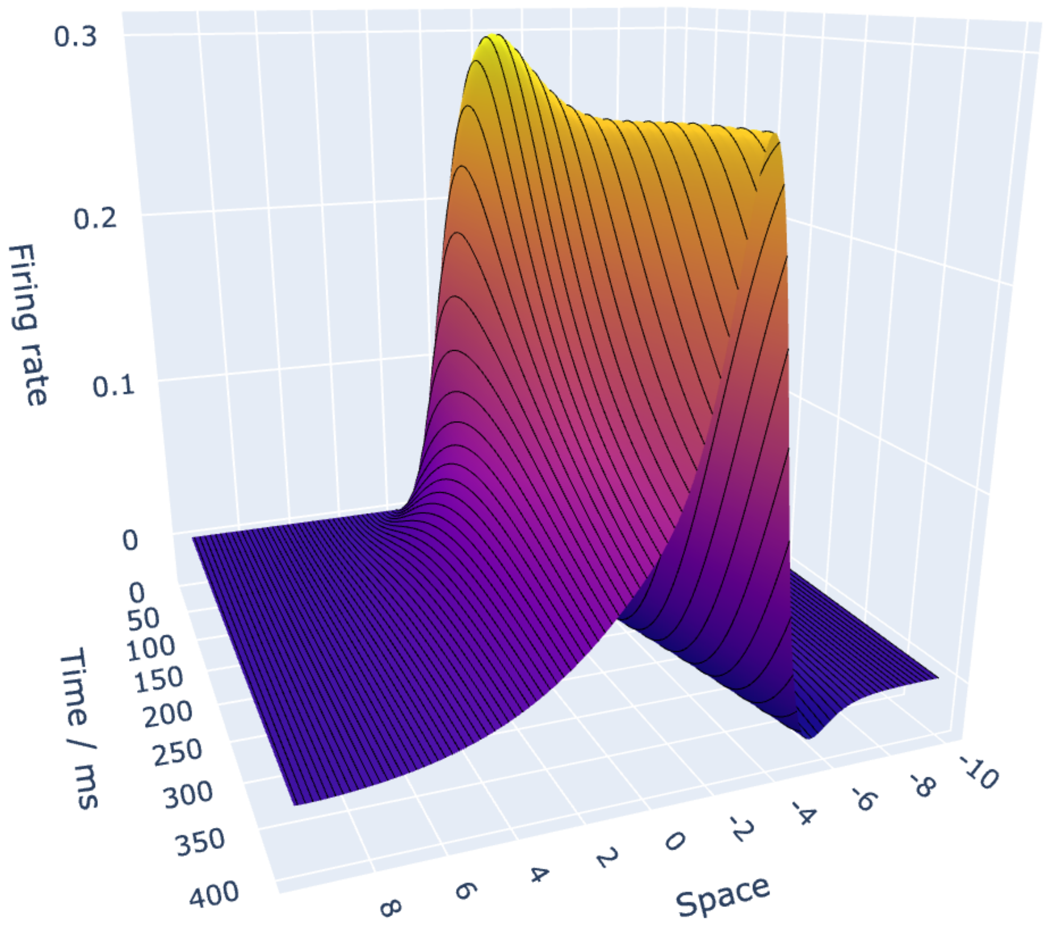

Figure 11: Same as in Fig. 6 but a negative stimulus speed.

The solution turns, or changes it's direction depending on the input velocity. Hence, this implies potential direction selectivity for the neural field computations.

1.7 We are required to compute the optimal input speed that leads to the highest output amplitude in the field assuming $|c| << k$. Recall that our solution in the Fourier domain is

$$\widetilde{u}(\omega, \infty) = \frac{\widetilde{s}(\omega)}{1 - \widetilde{w}(\omega)}$$

This time, we also have

$$\widetilde{w}(\omega) = \mathcal{F}[w](\omega) = \frac{2i\omega}{c^2 + \omega^2}$$

Question 2. Noninear Neural Field.

2.1 We are given a neural field of Amari type which is described by the following equation

$$\tau \dot{u}(x,t) = -u(x,t) + \int_{-\infty}^{\infty} w(x-x')\mathbb{1}[u(x',t)]dx' + s(x,t) - h \tag{4}$$

We assume the resting level parameter, $h = 1$, and that both the input and recurrent interaction kernel are given by the following piecewise linear functions

$$s(x) = \begin{cases} C(1-|x|/d), & |x| \leq d \\ 0, & \text{otherwise} \end{cases}$$

$$w(x) = \begin{cases} A, & |x| \leq a \\ -B, & a \leq |x| \leq b \\ 0, & \text{otherwise} \end{cases}$$

To simulate this neural field, similarly to Question 1, we use the Forward Euler approximation and we also express the convolution integral as a Riemann sum

$$\dot{u}(x,t+\Delta t) = u(x,t) + \frac{\Delta t}{\tau}\left(-u(x,t) + \sum_{i=0}^{N}\big[w(x-x_i)\mathbb{1}[u(x_i,t)]\Delta x\big] + s(x,t) - h\right)$$

Initially, we use the following parameters

- $A = 3$
- $B = 2$
- $C = 0.6$
- $a = 1$
- $b = 3$
- $d = 4$
- $h = 1$

The result of the simulation with the initial condition $u(x,0) = -h$ is shown in Figure 12.

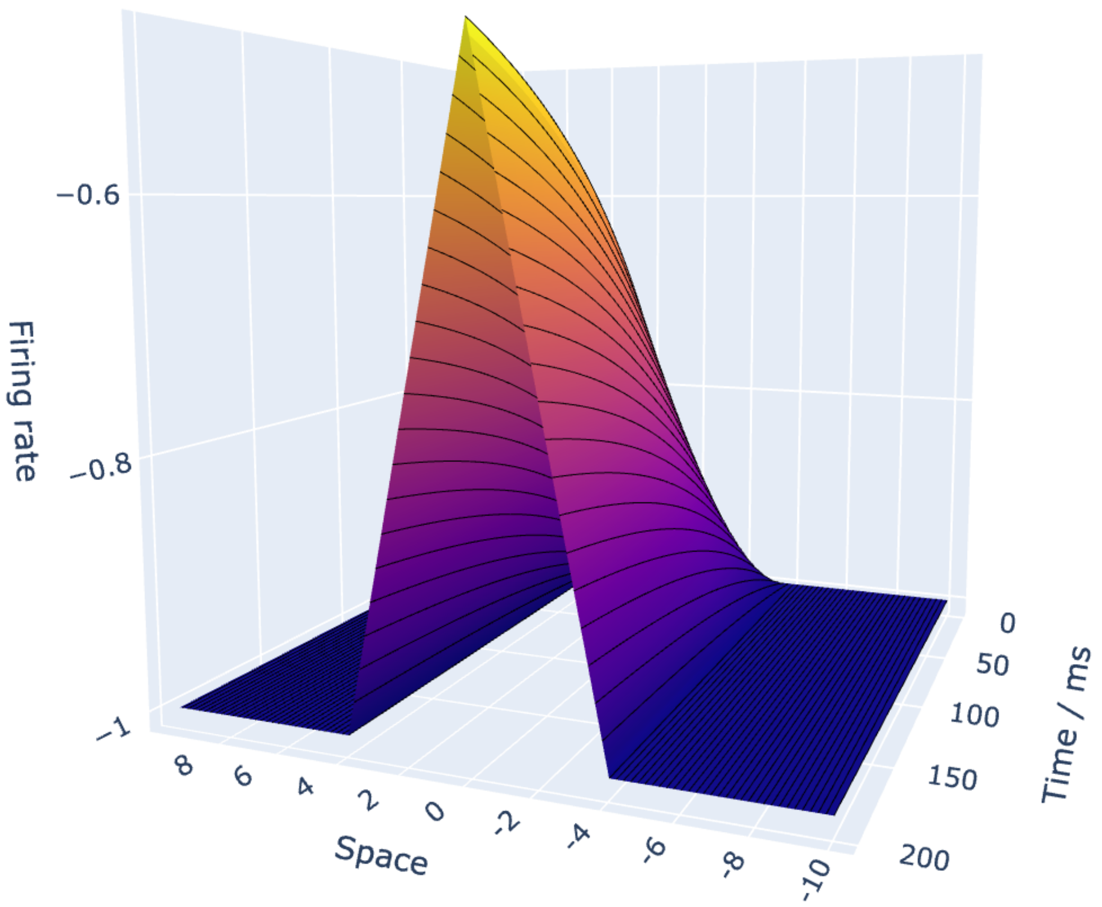

Figure 12: Dynamics of the nonlinear neural field defined by equation 4.

Now, we run another simulation but this time with a different initial condition, $u(x, 0) = 3.3s(x) - h$. This simulation appears in Figure 9. The apparent change can be explained by the fact that now our kernel starts to 'act' in all 3 modes given by the definition. Hence, we see 3 stable solutions.

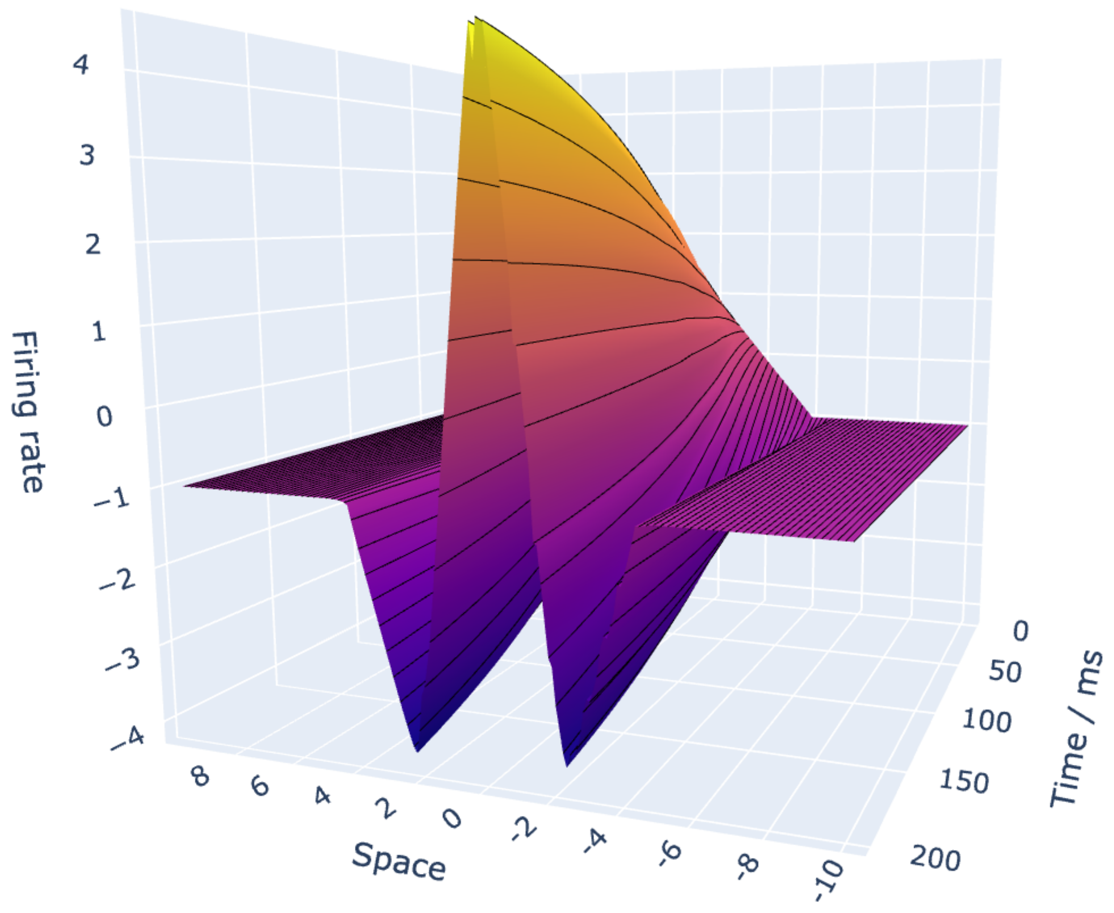

Figure 13: Same as in Figure 12 but with the initial condition $u(x, 0) = 3.3s(x) - h$.

Finally, we reduce the strength of the interaction kernel by one order of magnitude and run the simulation again with the remaining parameters as before. This simulation result is shown in Figure 14.

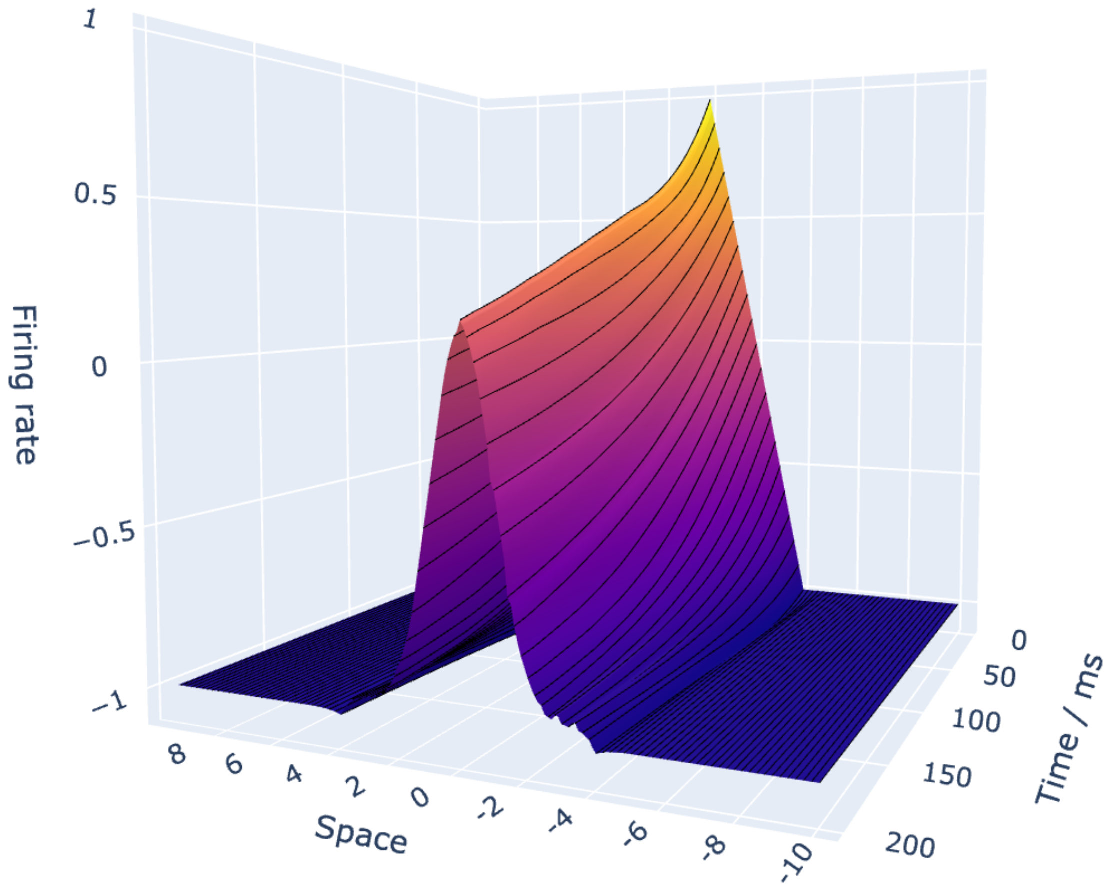

Figure 14: Same as in Figure 13 but with the interaction kernel $w(x)$ weakend by one order of magnitude.

The stationary solution appears at a firing rate of ≈ 0.3, which is $0.1w(x)$ for the initial condition used in this simulation.

2.2 As can be clearly seen from Figure 12, we have a localised activation peak

$$u^{\star}(x) = \begin{cases} \geq -h, & x_1 \leq x \leq x_2 \\ -h, & \text{otherwise} \end{cases}$$

Hence, we can express the field dynamics within the peak by

$$\tau \frac{\partial u(x,t)}{\partial t} = -u(x,t) + \int_{x_1(t)}^{x_2(t)} w(x-x')dx' + s(x) - h$$

where $x_1(t)$ and $x_2(t)$ are time-dependent boundaries of the peak. We can further simplify this to

$$\tau \frac{\partial u(x,t)}{\partial t} = -u(x,t) - W(x - x_2(t)) + W(x - x_1(t)) + s(x) - h$$

where

$$W(x) = \int_0^x w(x')dx'$$

Thus, our stationary solution is

$$u^\star(x) = W(x - x_1^\star(t)) - W(x - x_2^\star(t)) + s(x) - h$$

At the boundaries of our excited region, we have $u^\star(x) = -1$. Hence, this implies the following system with $a = x_2 - x_1$

$$\begin{aligned} &- W(x_1^\star(t) - x_1^\star(t)) + W(x_1^\star(t) - x_2^\star(t)) = W(-a^\star) = s(x_1^\star) - h \\ &- W(x_2^\star(t) - x_1^\star(t)) + W(x_2^\star(t) - x_2^\star(t)) = -W(a^\star) = s(x_2^\star) - h \end{aligned}$$

For spatially constant input, the above equations are satisfied for the symmetric kernel which fulfills

$$W(a^\star) + s - h = 0$$

Now, let us us employ the level-set method to analyse stability at the boundary points

$$u(x_i, t) = -1 \implies \frac{dx_i}{dt} = -\frac{\partial u(x_i, t)}{\partial t} / \frac{\partial u(x_i, t)}{\partial x}$$

for $i \in \{1, 2\}$. Let us define

$$\gamma_i \doteq \frac{\partial}{\partial x} u(x_i, t)$$

Substituting into the above we get

$$\begin{aligned} \frac{dx_1}{dt} &= -\frac{1}{\tau\gamma_1}[-W(x_1(t) - x_2(t)) + s - h] \\ \frac{dx_2}{dt} &= -\frac{1}{\tau\gamma_2}[W(x_2(t) - x_1(t)) + s - h] \end{aligned}$$

Note that if we subtract second equation from the first, we get

$$\frac{da}{dt} = \frac{1}{\tau}\left(\frac{1}{\gamma_1} - \frac{1}{\gamma_2}\right)[W(a(t)) + s - h]$$

Since the gamma-tau term is non-negative, this implies asymptotic stability for

$$\left.\frac{dW}{da}\right|_{a^\star} = w(a^\star) < 0$$

2.4 Now we simulate the same neural field with the initial condition $u(x, 0) = 3.3s(x) - h$ and the initial interaction kernel; however, we set the input current to zero (but not for the initial condition). The result is shown in Figure 15.

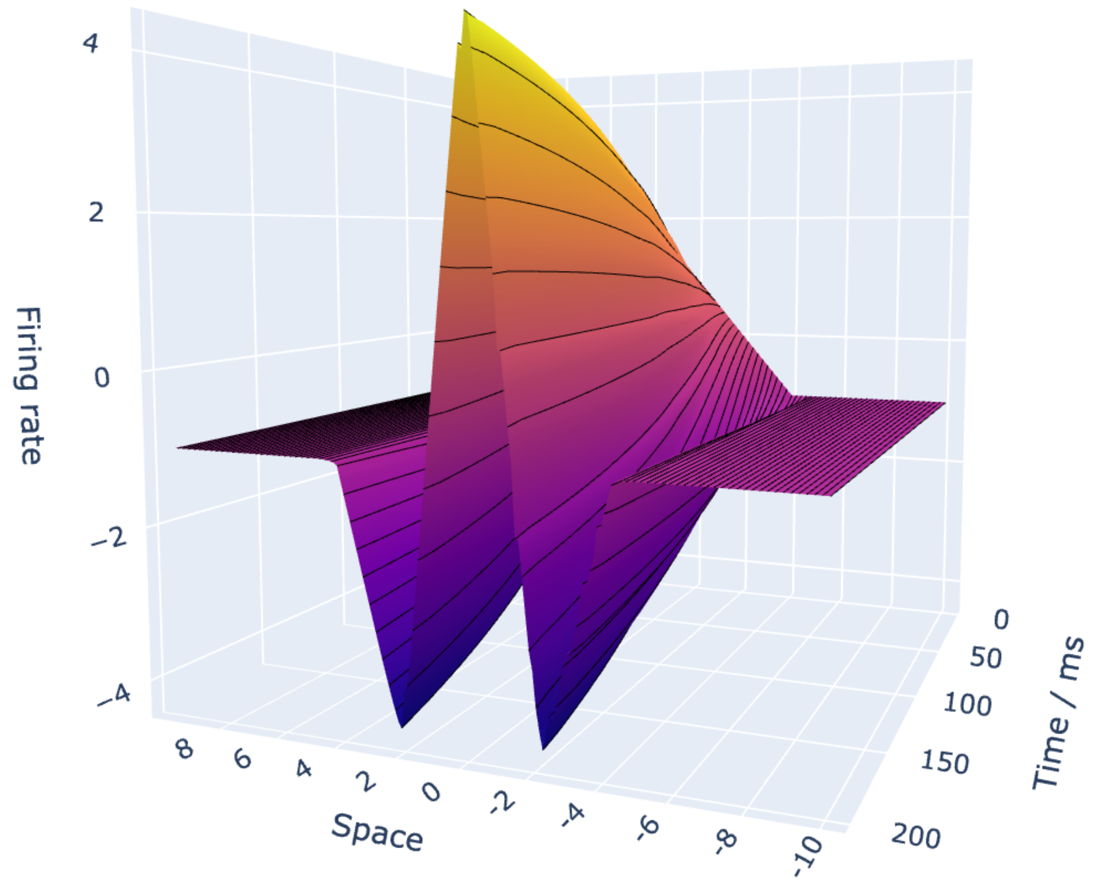

Figure 15: Nonlinear neural field dynamics with the initial condition $u(x,0) = 3.3s(x) - h$ and the initial interaction kernel, but no input current.

2.5 Finally, we run the simulatiom with the same conditions as above, except that now we add a non-symmetric interaction kernel to our previous kernel, i.e. effectively we have $w(x) + w_u(x)$, where

$$w_u(x) = \begin{cases} 0.8x, & |x| \leq b \\ 0, & \text{otherwise} \end{cases}$$

The emerging solution appears in Figure 16.

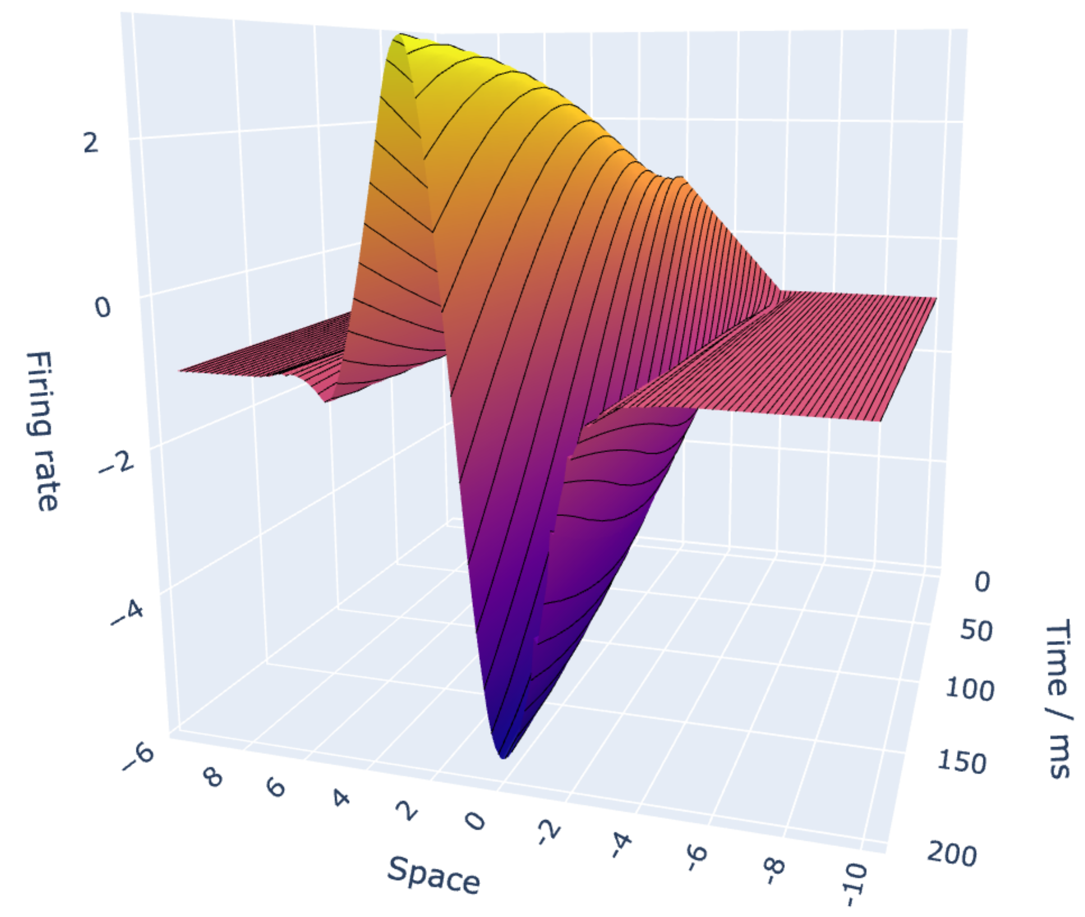

Figure 16: Nonlinear neural field dynamics with the initial condition $u(x, 0) = 3.3s(x) - h$ and the interaction kernel $w(x) + w_u(x)$.

NeuralField

January 29, 2020

```
[ ]: import matplotlib.pyplot as plt
     import plotly.graph_objects as go
     import numpy as np
     %matplotlib inline
```

```
[ ]: class NeuralField:

         def __init__(self, tau, a, b, d, k, c=None, v=None):

             self.tau = tau
             self.a = a
             self.b = b
             self.d = d
             self.k = k

             if c and v:
                 self.c = c
                 self.v = v

         def kernel(self, x):
             '''Interaction kernel, Gabor filter'''
             return self.a * (np.exp(-(x**2)/(4*self.b**2))*np.cos(self.k*x))/(self.
      ↪b*np.sqrt(np.pi))

         def kernel2(self, x):
             return np.exp(-self.c*abs(x))*np.sign(x)

         def current(self, f, A, dx, add_noise=False):
             '''Input stimulus'''
             r = np.zeros(len(f))
             for k in range(len(f)):
                 xk = A + k*dx
                 r[k] = np.exp(-(xk**2)/(4*self.d**2))/(2*self.d*np.sqrt(np.pi))
             if add_noise:
                 mu, sigma = 0, 0.01
                 noise = np.random.normal(mu, sigma, len(f))
                 r += noise
```

```
        return r

    def current2(self, f, A, dx, t, add_noise=False):
        r = np.zeros(len(f))
        for k in range(len(f)):
            xk = A + k*dx
            r[k] = self.c * np.exp(-((xk - self.v*t)**2)/(4*self.d**2))/(2*self.
↪d*np.sqrt(np.pi))
        if add_noise:
            mu, sigma = 0, 0.01
            noise = np.random.normal(mu, sigma, len(f))
            r += noise
        return r

    def convolve(self, f, A, dx):
        '''Linear 1D convolution'''
        r = np.zeros(len(f))
        for k in range(len(f)):
            xk = A + k*dx
            for i in range(len(f)):
                xi = A + i*dx
                r[k] += self.kernel(xk - xi) * f[i] * dx
        return r

    def simulate(self, A, B, dt, N):
        '''Simulate the Neural Field'''
        t = np.arange(0, 40, dt)
        dx = (B-A)/N
        du = np.zeros((len(t), N))

        for i in range(1, len(t)):
            this_convolution = self.convolve(du[(i-1), :], A, dx)
            this_current     = self.current(du[(i-1), :], A, dx,␣
↪add_noise=False)
            du[i, :] = du[(i-1), :] + (dt/self.tau)*(-du[(i-1), :] +␣
↪this_convolution + this_current)
        return du

class NonlinearNeuralField:

    def __init__(self, tau, a, b, d, h, A, B, C):

        self.tau = tau
        self.a = a
        self.b = b
        self.d = d
        self.h = h
        self.A = A
        self.B = B
        self.C = C

    def kernel(self, x):

        if abs(x) <= self.a:
            return self.A
        elif (abs(x) >= self.a) and (abs(x) <= self.b):
            return -self.B
        return 0

    def kernel2(self, x):
        if abs(x) <= self.b:
            return 0.8*x
        else:
            return 0

    def current(self, f, A, dx):
        '''Input stimulus'''
        r = np.zeros(len(f))
        for k in range(len(f)):
            xk = A + k*dx
            if abs(xk) <= self.d:
                r[k] = self.C*(1-(abs(xk)/self.d))
            else:
                r[k] = 0
        return r

    def convolve(self, f, A, dx):
        '''Linear 1D convolution'''
        r = np.zeros(len(f))
        for k in range(len(f)):
            xk = A + k*dx
            for i in range(len(f)):
                xi = A + i*dx

                r[k] += self.kernel(xk - xi) * (f[i]>0) * dx
        return r

    def simulate(self, A, B, dt, N):
        '''Simulate the Neural Field'''
        t = np.arange(A, B, dt)
        dx = (B-A)/N
        du = np.zeros((len(t), N))
        to_start = - self.h
        du[0, :] = to_start
        for i in range(1, len(t)):
            this_convolution = self.convolve(du[(i-1), :], A, dx)
            this_current     = self.current(du[(i-1), :], A, dx)
            du[i, :] = du[(i-1), :] + (dt/self.tau)*(-du[(i-1), :] +␣
 ↪this_convolution + this_current - self.h)
        return du
```

[ ]:
```
tau = 10
a = 1
b = 0.6
d = 2
k = 4
c = 1
v = 0.1
f = NeuralField(tau, a, b, d, k, c, v)
```

[ ]:
```
A = -10
B = 10
dt = 0.1
N = 200
du = f.simulate(A, B, dt, N)
```

[ ]:
```
vals = np.arange(0, 200, 20)
ticks = np.arange(-10, 10, 2)

layout = go.Layout(scene = dict(
                    xaxis = dict(
                        title = 'Space',
                        tickmode = 'array',
                        tickvals = vals,
                        ticktext = ticks),
                    yaxis_title='Time / ms',
                    zaxis_title='Firing rate'),
                    width=700,
                    margin=dict(r=20, b=10, l=10, t=10))

fig = go.Figure(data=go.Surface(z=du, contours = {
        "x": {"show": True, "start": 0, "end": 200, "size": 2, "color":
 ↪"black"}}), layout=layout)
fig.show()
```

[ ]:
```
tau = 10
a = 1
b = 3
d = 4
h = 1
```

```
A = 3
B = 2
C = 0.6
n = NonlinearNeuralField(tau, a, b, d, h, A, B, C)
```

[ ]:
```
A = -10
B = 10
dt = 0.1
N = 200
du = n.simulate(A, B, dt, N)
```

[ ]:
```
vals = np.arange(0, 200, 20)
ticks = np.arange(-10, 10, 2)

layout = go.Layout(scene = dict(
                    xaxis = dict(
                        title = 'Space',
                        tickmode = 'array',
                        tickvals = vals,
                        ticktext = ticks),
                    yaxis_title='Time / ms',
                    zaxis_title='Firing rate'),
                    width=700,
                    margin=dict(r=20, b=10, l=10, t=10))

fig = go.Figure(data=go.Surface(z=du, contours = {
        "x": {"show": True, "start": 0, "end": 200, "size": 2, "color":
 ↪"black"}}), layout=layout)
fig.show()
```

[ ]:
```
```