For a given network that is seen in Figure 1, initial kinesis control the activity of three other kinases with concentrations [Y1], [Y2] and Y[3] through weights w_1, w_2, w_3, w_4, w_5, w_6. And, [Y1], [Y2] and Y[3] control the activity of Z through weights w_7, w_8, w_9.

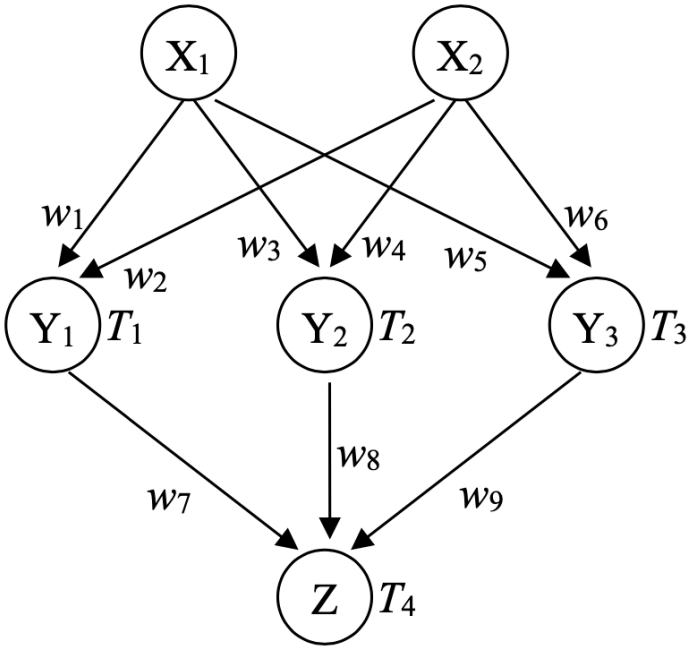

Figure 1: The given network

Firstly, the randomized network parameters, weights and tresholds, are initialized. In addition, the multi-layer perceptron network is defined.

```
from matplotlib import colors
import matplotlib.pyplot as plt
import matplotlib
import numpy as np
from random import random

# generates 101-element array from 0 to 1.01 in 0.01 increments
x = np.arange(0, 1.01, 0.01)

# creates 2-D coordinate arrays, given one-dimensional coordinate arrays x1, x2
X1, X2 = np.meshgrid(x, x)

# random weights and tresholds
w = np.random.normal(loc=0.0, scale=1.0, size=9)
T = np.random.normal(loc=0.0, scale=1.0, size=4)

# multi-layer perceptron network
Y1=  (w[0] * X1 + w[1] * X2 >= T[0])
Y2 = (w[2] * X1 + w[3] * X2 >= T[1])
Y3 = (w[4] * X1 + w[5] * X2 >= T[2])
```

```
Z = (w[6] * Y1 + w[7] * Y2 + w[8] * Y3 >= T[3])
```

And X1 and X2 can be seen in Figure 2.

```
fig=plt.figure(figsize=(20,6))
ax=fig.add_subplot(1,2,1)
h=ax.imshow(X1,origin='lower',extent=[0,1,0,1],cmap=matplotlib.cm.jet)
ax.set_xlabel('$[X_1]$')
ax.set_ylabel('$[X_2]$')
ax.set_title('$[X_1]$')
fig.colorbar(h)


ax=fig.add_subplot(1,2,2)
h=ax.imshow(X2,origin='lower',extent=[0,1,0,1],cmap=matplotlib.cm.jet)
ax.set_xlabel('$[X_1]$')
ax.set_ylabel('$[X_2]$')
ax.set_title('$[X_2]$')
fig.colorbar(h)
```

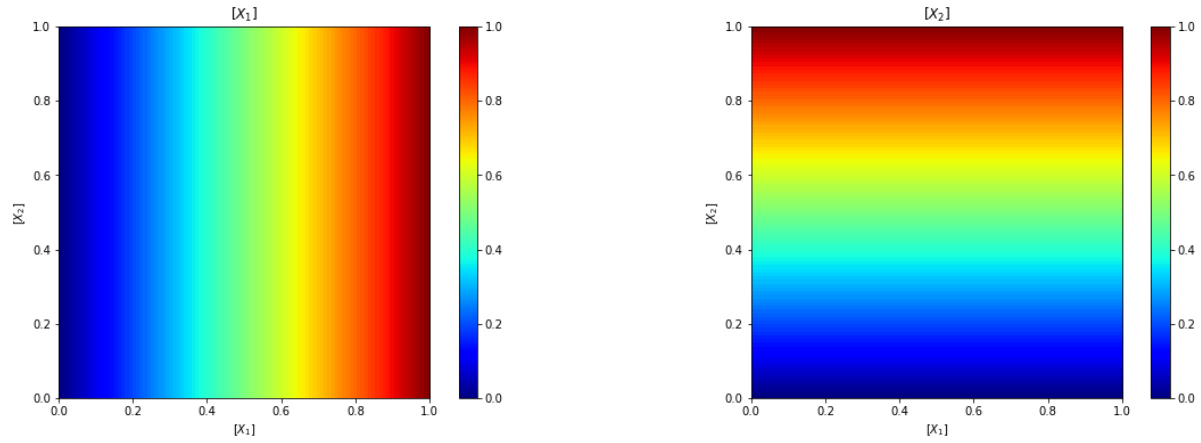

Figure 2:X1 and X2

Besides, the desired response of this network is as shown in Figure 2.

```
# desired response
border =
np.logical_or(np.logical_and(X1>=0.3,X1<=0.6),np.logical_and(X2>=0.2,X2
<=0.7))

fig, ax = plt.subplots()
plt.imshow(border,cmap='OrRd',interpolation='None',extent=[0,1,0,1])
plt.xlabel('x1')
plt.ylabel('x2')
plt.grid(linestyle='--')
plt.show()
```

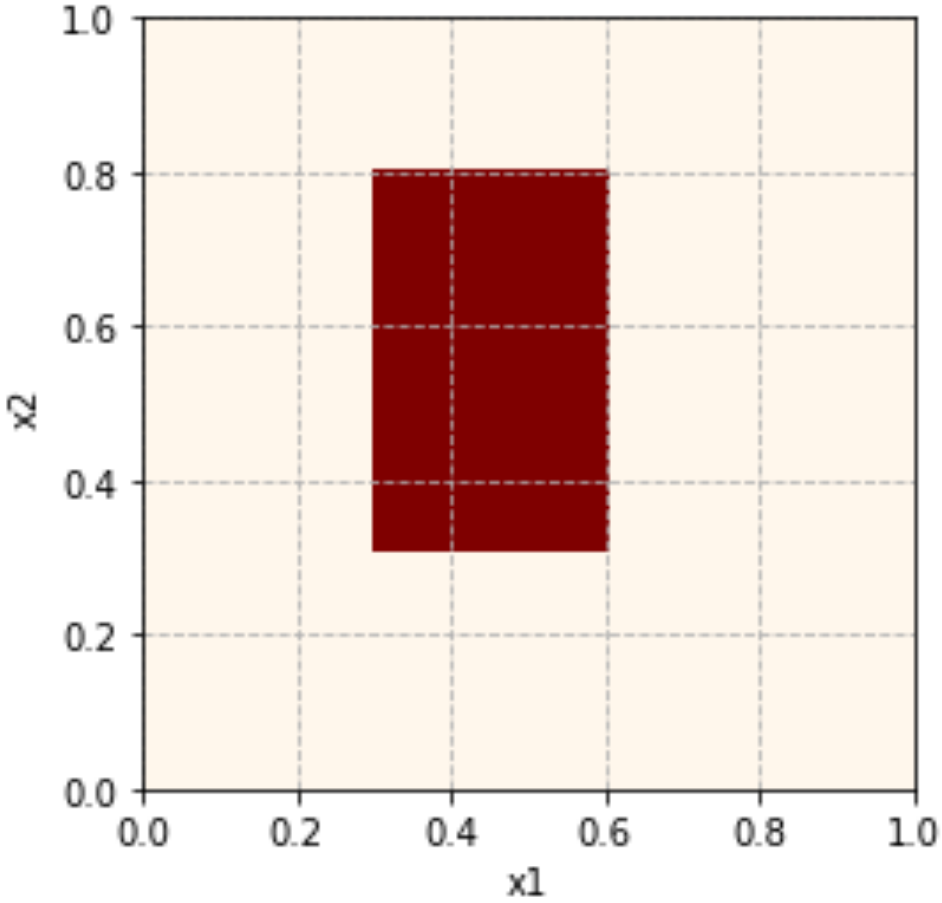

Figure 3: Desired response of the network

And the initial response characteristic of the system can be obtained by getting the difference between current response and desired respones.

```
# initial response characteristic of the system
err=np.sum(border != Z)
print(100*err/(len(border)**2))
```

The error rate is approximately 23.3% for a sample given in Figure 3.

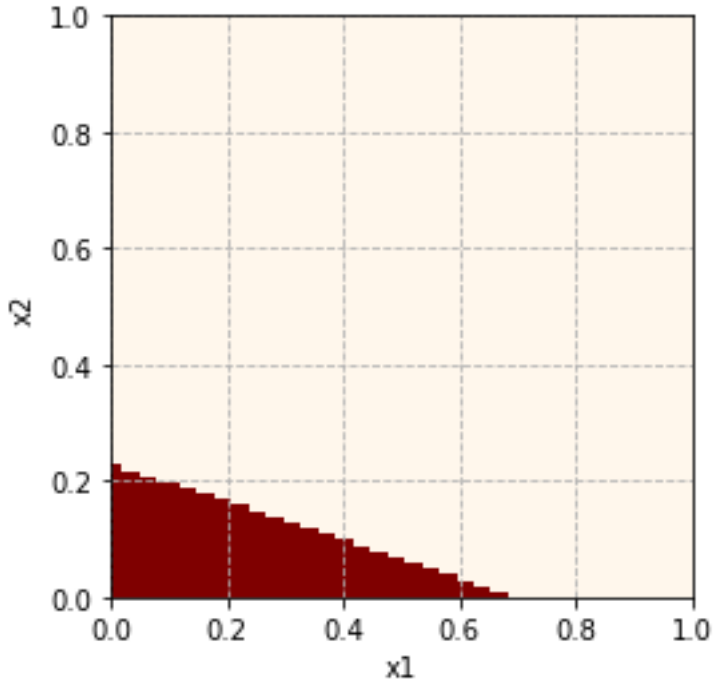

Figure 4:Initial response of the network

As requested, iteration is applied for the best response.

```
#initialization for the good variables
w_new=w
T_new=T
Z_new=Z

i =0

#iterate for the best scenario
while i<100000:

    w =np.random.normal(loc=w_new,scale=1.0,size=9)
    T =np.random.normal(loc=T_new,scale=1.0,size=4)
    Y1=(w[0] * X1 + w[1] * X2 >= T[0])
    Y2=(w[2] * X1 + w[3] * X2 >= T[1])
    Y3=(w[4] * X1 + w[5] * X2 >= T[2])
    Z=(w[6] * Y1 + w[7] * Y2 + w[8] * Y3 >= T[3])

    #find the difference
    err_best=np.sum(border != Z)

    #change the condition
    if err_best < err:
        w_new=w
```

```
            T_new=T
            Z_new=Z
            err=err_best

        i=i+1

fig, ax = plt.subplots()
plt.imshow(border != Z_new,extent=[0,1,0,1], cmap='OrRd')
plt.title('', fontsize=8)
plt.xlabel('x1')
plt.ylabel('x2')
plt.grid(linestyle='--')
plt.show()

print(100*err/(len(border)**2))
```

The difference between desired and the best response for a sample is given in Figure 4.

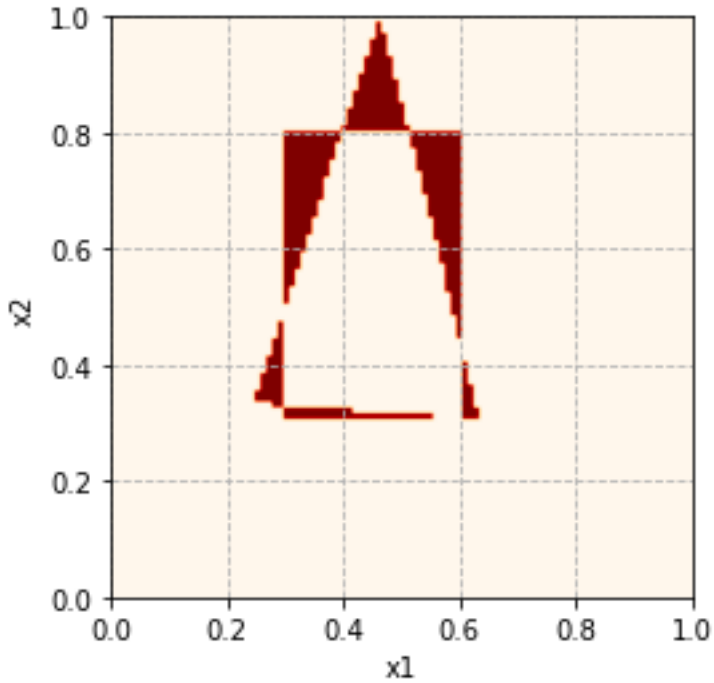

Figure 5: The difference between desired and the best response

The improved rate is approximately 5.38% for the parameters below.

T1	-3.06793
T2	4.27503
T3	-2.7551
T4	2.05969
W1	0.340394
W2	-4.71902
W3	9.20858
W4	-2.26693
W5	-5.98004
W6	-1.94117
W7	2.41218
W8	-5.64457
W9	-4.45549

The whole code can be accessed in Colab Notebook[1].

[1] https://colab.research.google.com/drive/1k4pYShZHtdzolLPCoG1Gf_EFlC06KcsE#scrollTo=b98JLk0merAQ