CS6786: Computational Cognitive Science Assignment 1

AMRIT SINGHAL
150092
amrits@iitk.ac.in

January 29, 2019

Solution 1

Code sumbitted as: `q1.py`
Run as: `python q1.py`

(a) The function `take_func_input()` can be used to take any boolean function as input from the user. The function is not being used in the code right now, as I am generating multiple boolean function randomly to verify the ANN model and training it on all of them sequentially. But it can directly be used in the `main()` function if the user needs to provide his own input.

(b) The function `create_dataset()` can be used to generate training examples from this boolean function. We can use the `size_multiplier` parameter to obtain what multiple of theinput do we want as our batch size. This prepares the required batch for training input.

(c) The class `ANN` defines our neural network architecture used to learn the boolean function. It contains all the definitions, forward propagation and back-propagation steps involved in the learning algorithm. The neural network finally used consists of a singel hidden layer with 15 nodes. I have used the sigmoid activation function on the hidden and the output layers.

(d) The function `generate_bool_func` can be used to generate a boolean function randomly. Rather than explicitly stating five fixed different boolean function, I generate five (or more) boolean functions randomly and use them to verify.

Solution 2

Code sumbitted as: `q2.py`
Run as: `python q2.py <N> <M>`

(a) The class `frozen_lake` takes the parameters `N` and `M` and initialises to a random frozen lake grid of size NxN with M holes.

(b) The functions `q_learn()`, `update_q()`, `get_next()`, `select_action_epsilon_greedy()` and `select_action_softmax()` together implement the Q-learning algorithm for the frozen lake scenario. I have implemented both the ϵ-greedy and the softmax approaches for action selection, and I have finally used ϵ-greedy approach.

Figure 1 is the graph obtained for "total reward per episode vs. episode number" when the q-learning algorithm is run on a 40×40 grid with 160 holes, having the ϵ-greedy action selection policy with initial $\epsilon = 0.5$ and ϵ-DecayFactor $= 0.99$. Each hole was assigned a reward of -1 and the goal was assigned a reward of 1. The lake which was initialised for the above setting and the path that was obtained is shown in Figure 2.

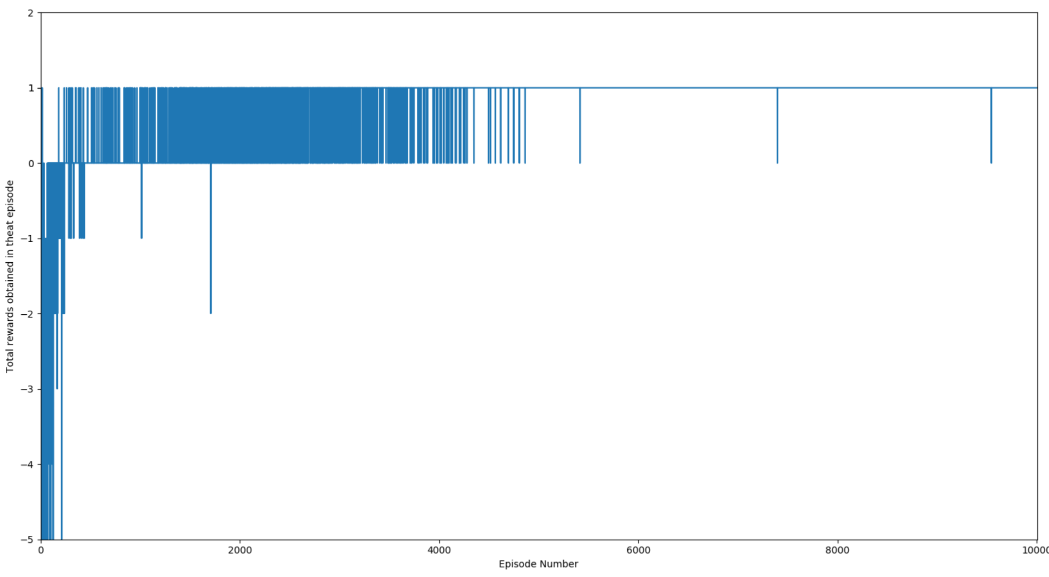
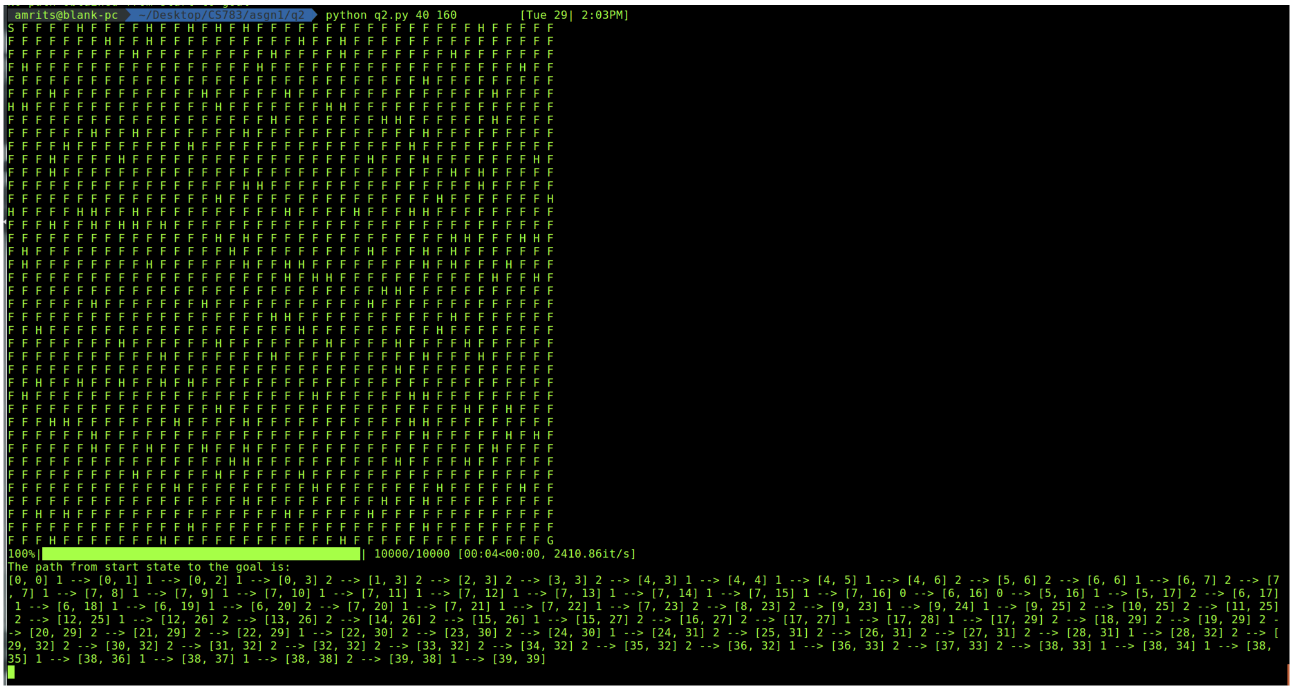


Figure 1: Total reward per episode vs. episode number in q-learning



```
 amrits@blank-pc  ~/Desktop/CS783/asgn1/q2  python q2.py 40 160          [Tue 29| 2:03PM]
S F F F F H F F F F H F F H F H F H F F F F F F F F F F F F F F F F H F F F F F
F F F F F F F H F F H F F F F F F F F F F H F F H F F F F F F F F F F F F F F F
F F F F F F F F F H F F F F F F F F F H F F F F H F F F F F F F H F F F F F F F
F H F F F F F F F F F F F F F F F F H F F F F F F F F F F F F F F F F F F H F F
F F F F F F F F F F F F F F F F F F F F F F F F F F F F F F H F F F F F F F F F
F F F H F F F F F F F F F F H F F F F F H F F F F F F F F F F F F F F H F F F F
H H F F F F F F F F F F F F F H F F F F F F F H H F F F F F F F F F F F F F F F
F F F F F F F F F F F F F F F F F F F H F F F F F F F H H F F F F F F H F F F F
F F F F F F H F F H F F F F F F F H F F F F F F F F F F F F H F F F F F F F F F
F F F F H F F F F F F F F H F F F F F F F F F F F F F F F H F F F F F F F F F F
F F F H F F F F H F F F F F F F F F F F F F F F F F H F F F H F F F F F F F H F
F F F H F F F F F F F F F F F F F F F F F F F F F F F F F F F F H F H F F F F F
F F F F F F F F F F F F F F F F F H H F F F F F F F F F F F F F F F H F F F F F
F F F F F F F F F F F F F F F H F F F F F F F F F F F F F F F H F F F F F F F H
H F F F F H H F F H F F F F F F F F F F H F F F F H F F F H H F F F F F F F F F
F F F H F F H F H H F H F F F F F F F F F F F F F F F F F F F F F F F F F F F F
F F F F F F F F F F F F F F F H F H F F F F F F F F F F F F F F H H F F F H H F
F H F F F F F F F F F F F F F F H F F F F F F F F F H F F F H F H F F F F F F F
F H F F F F F F F F H F F F F F F H F F H H F F F F F F F F H F H F F F H F F F
F F F F F F F F F F F F F F F F F F F F H F H H F F F F F F F F F F F H F F H F
F F F F F F F F F F F F F F F F F F F F F F F F F F F H H F F F F F F F F F F F
F F F F F F H F F F F F F F H F F F F F F F F F F F H F F F F F F F F F F F F F
F F F F F F F F F F F F F F F F F F F H H F F F F F F F F F F F H F F F F F F F
F F H F F F F F F F F F F F F F F F F F F H F F F F F F F F F H F F F F F F F F
F F F F F F F F H F F F F F F H F F F F F F F H F F F F H F F F F H F F F F F F
F F F F F F F F F F F H F F F F F F F H F F F F F F F F F F H F F F H F F F F F
F F F F F F F F F F F F F F F F F F F F F F F F F F F F H F F F F F F F F F F F
F F H F F H F F H F F H F H F F F F F F F F F F F F F F F F F F F F F F F F F F
F H F F F F F F F F F F F F F F F F F F F F H F F F F F F H H F F F F F F F F F
F F F F F F F F F F F F F F F H F F F F F F F F F F F F F F F F F H F F H F F F
F F F H H F F F F F F F H F F F F H F F F F F F F F F F F H H F F F F F F F F F
F F F F F F H F F F F F F F F F F F F F F F F F F F F F F F H F F F F F H F H F
F F F F F F H F F F H F F F H F F H F F F F F F F F F F F F F F F F F H F F F F
F F F F F F F F F F F F F F F F H H F F F F F F F F F F H F F F F H F F F F F F
F F F F F F F F F H F F F F F H F F F F F H F F F F F F F F F F F F F F F F F F
F F F F F F F F F F F F H F F F F F F F F F H F F F F F F F F H F F F F F H F F
F F F F F F F F F F F F F F F F F H F F F F F F F F F H F F H F F F F F F F F F
F F H F H F F F F F F F F F F F F F F F H F F F F F H F F F F F F F F F F F F F
F F F F F F F F F F F F F H F F F F F F F F F F F F F F F F H F F F F F F F F F
F F F H F F F F F F F H F F F F F F F F F F F F H F F F F F F F F F F F F F F G
100%|████████████████████████████████████████| 10000/10000 [00:04<00:00, 2410.86it/s]
The path from start state to the goal is:
[0, 0] 1 --> [0, 1] 1 --> [0, 2] 1 --> [0, 3] 2 --> [1, 3] 2 --> [2, 3] 2 --> [3, 3] 2 --> [4, 3] 1 --> [4, 4] 1 --> [4, 5] 1 --> [4, 6] 2 --> [5, 6] 2 --> [6, 6] 1 --> [6, 7] 2 --> [7
, 7] 1 --> [7, 8] 1 --> [7, 9] 1 --> [7, 10] 1 --> [7, 11] 1 --> [7, 12] 1 --> [7, 13] 1 --> [7, 14] 1 --> [7, 15] 1 --> [7, 16] 0 --> [6, 16] 0 --> [5, 16] 1 --> [5, 17] 2 --> [6, 17]
 1 --> [6, 18] 1 --> [6, 19] 1 --> [6, 20] 2 --> [7, 20] 1 --> [7, 21] 1 --> [7, 22] 1 --> [7, 23] 2 --> [8, 23] 2 --> [9, 23] 1 --> [9, 24] 1 --> [9, 25] 2 --> [10, 25] 2 --> [11, 25]
 2 --> [12, 25] 1 --> [12, 26] 2 --> [13, 26] 2 --> [14, 26] 2 --> [15, 26] 1 --> [15, 27] 2 --> [16, 27] 2 --> [17, 27] 1 --> [17, 28] 1 --> [17, 29] 2 --> [18, 29] 2 --> [19, 29] 2 -
-> [20, 29] 2 --> [21, 29] 2 --> [22, 29] 1 --> [22, 30] 2 --> [23, 30] 2 --> [24, 30] 1 --> [24, 31] 2 --> [25, 31] 2 --> [26, 31] 2 --> [27, 31] 2 --> [28, 31] 1 --> [28, 32] 2 --> [
29, 32] 2 --> [30, 32] 2 --> [31, 32] 2 --> [32, 32] 2 --> [33, 32] 2 --> [34, 32] 2 --> [35, 32] 2 --> [36, 32] 1 --> [36, 33] 2 --> [37, 33] 2 --> [38, 33] 1 --> [38, 34] 1 --> [38,
35] 1 --> [38, 36] 1 --> [38, 37] 1 --> [38, 38] 2 --> [39, 38] 1 --> [39, 39]
```

Figure 2: An instance of initialised lake and obtained path after q-learning

(c) **<u>Variation with α</u>**

Figure 3 shows results of the Q-learning algorithm for various values for α in the range (0.0, 1.0), keeping all other parameters the same.

As we can see from the graphs, the number of steps required to converge decently first decreases with an increase in the value of α as it is increased from 0.2 to 0.5, but a further increase in the value of α leads to a increase in the number of steps required to converge. This increase in steps required to converge with increase in α beyond 0.5 is lesser as compared to the increase in steps with decrease in α from 0.5.

<u>Variation with λ</u>

Figure 4 shows the results of the Q-learning algorithm for various values for λ, keeping all other parameters the same.

It is clearly visible from the graphs that there seems to be a threshold value of λ below which the Q-learning algorithm fails to learn the path to the goal. Notably, it still learns to stay away from the holes, getting its total reward to 0 or 1, depending on whether it could reach the goal without any holes, or could reach the goal through one hole, or could reach the goal at all, and just went oscillating on the frozen parts of the lake. But, once above the threshold value for λ, all values of λ seem to do a good job at learning a path to reach the goal to get a reward of 1.

(d) **<u>Variation with N</u>**

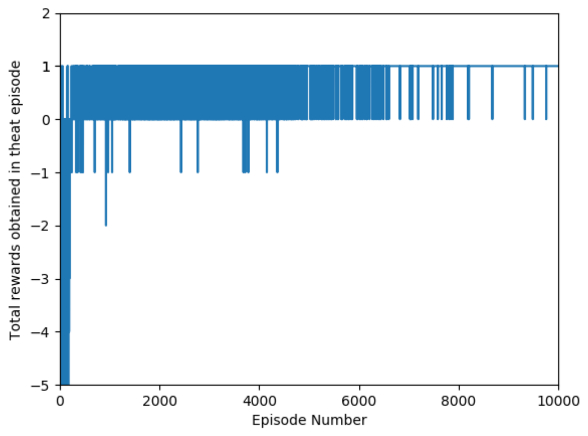
As the size of the lake increases, the number of steps required to converge increases. This can be empirically seen from the graphs of the Q-learning task at different values of N, as shown in Figure 5 keeping all other parameters constant.

This can be attributed to the fact that with an increase in the value of N, the total number of parameters to explore and learn grows in $O(N^2)$. So, the total number of iterations required to learn them all increases.

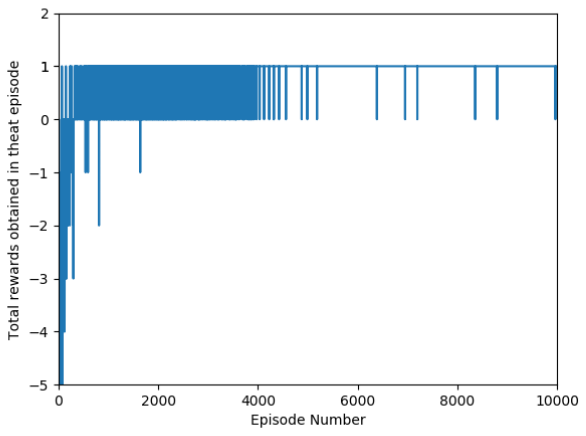
<u>Variation with M</u>

As the number of holes in the lake increases, we see that the learning time of the algorithm increases, as well a gentle increase in the number of steps required for convergence. The graphs shown in Figure 6 affirm this empirically, where we show the learning performance of three Q-learning scenarios, with different values of M, but keep all other parameters constant. The increase in learning time was also observed during training.

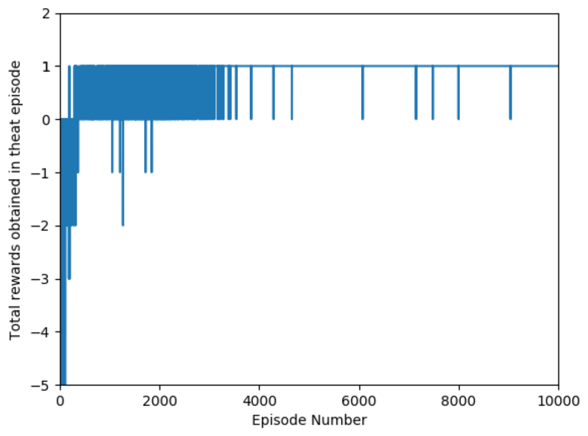
As we increase the number of holes in the matrix, we have more and more steps in the iterations that now possibly lead to a negatively rewarding condition and will be avoided later. Thus, more paths have to be found to obtain a rewarding path. So, the number of episodes required to converge increases. Also, since finding the goal becomes harder in the presence of more holes, thus more number of iteration per episode are required in general before we can reach our goal, or till the episode reaches max-iterations. Thus, the run time of the learning seems to increase.



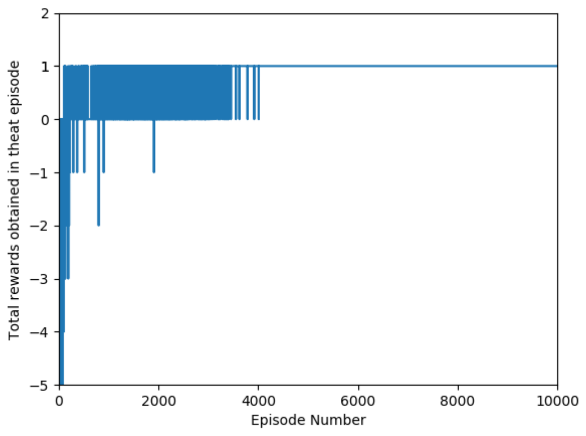
(a) $\alpha = 0.2$



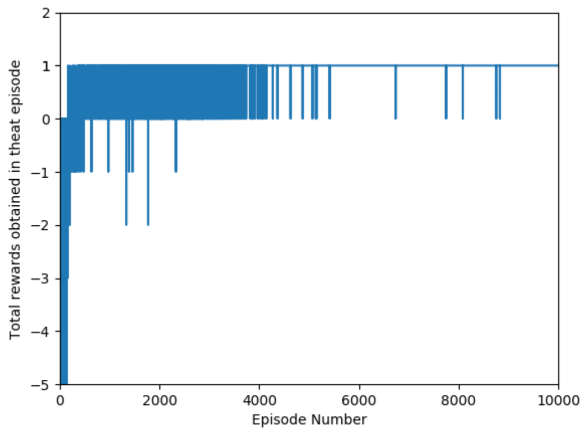
(b) $\alpha = 0.4$



(c) $\alpha = 0.5$

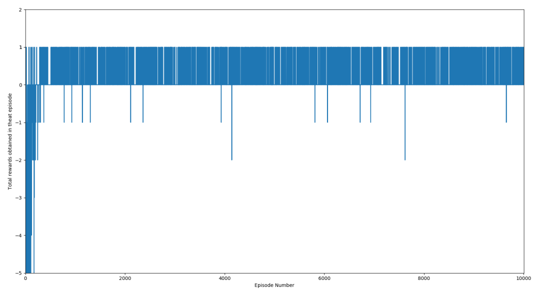


(d) $\alpha = 0.6$

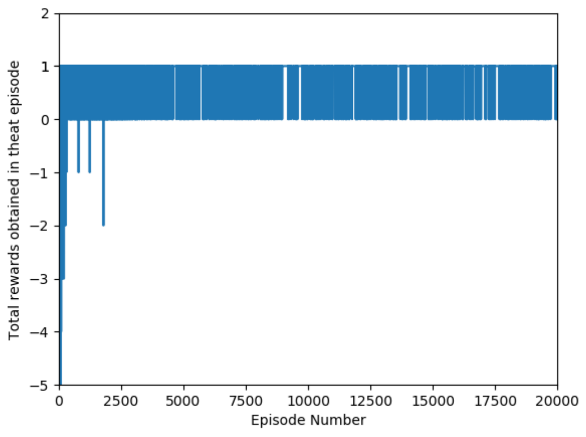


(e) $\alpha = 0.8$

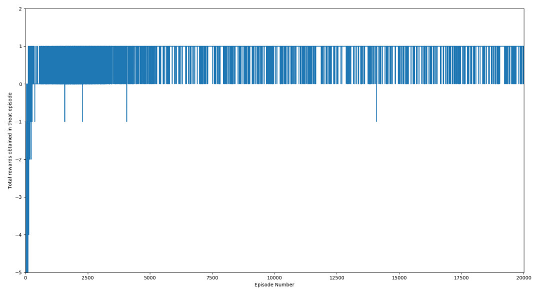
Figure 3: Variation in learning performance with α



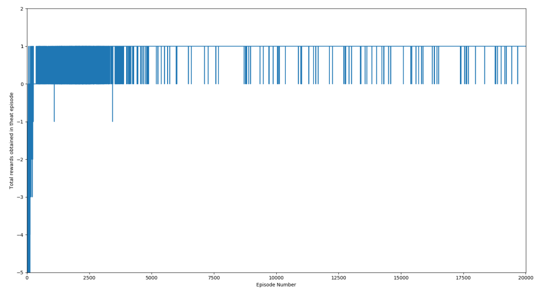
(a) $\lambda = 0.35$



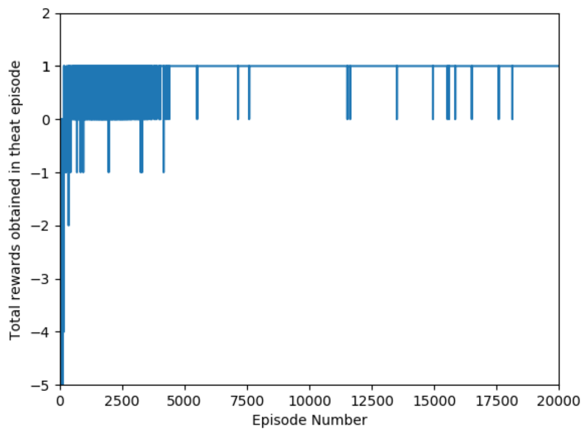
(b) $\lambda = 0.81$



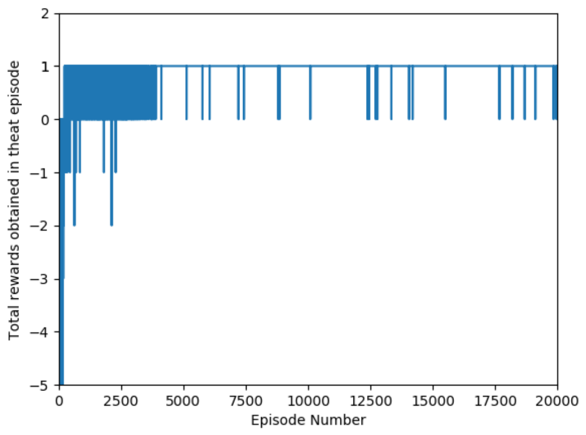
(c) $\lambda = 0.82$



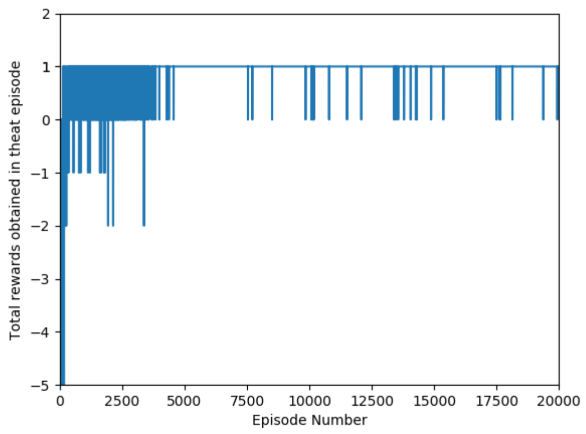
(d) $\lambda = 0.83$



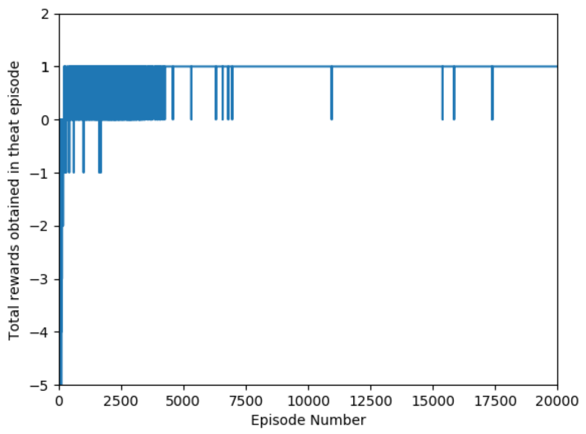
(e) $\lambda = 0.84$



(f) $\lambda = 0.85$

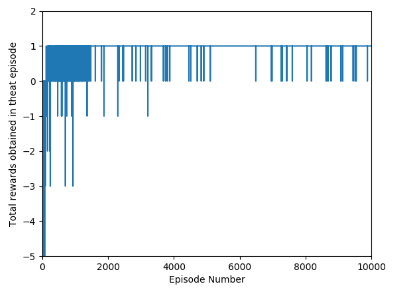


(g) $\lambda = 0.9$

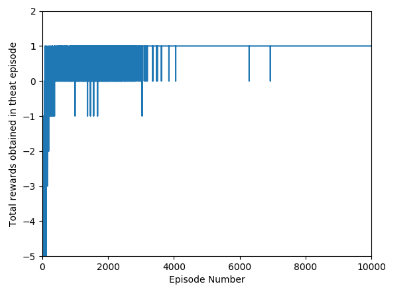


(h) $\lambda = 1.0$

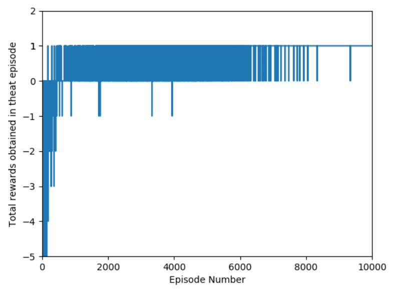
Figure 4: Variation in learning performance with λ



(a) $N = 30$

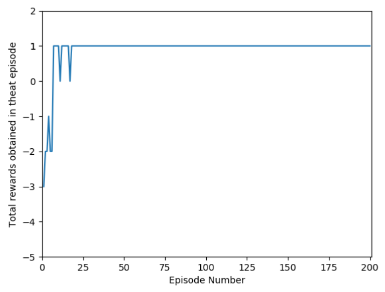


(b) $N = 40$

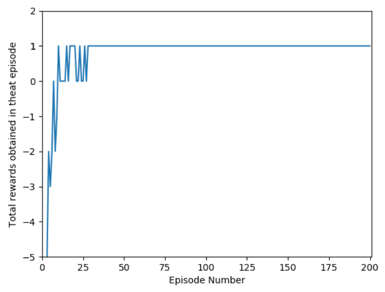


(c) $N = 50$

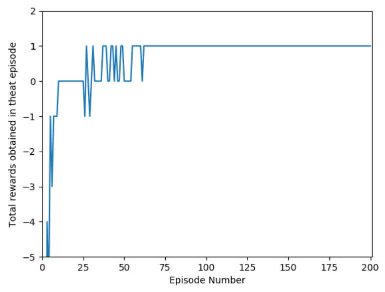
Figure 5: Variation in learning performance with N



(a) $M = 5$



(b) $M = 15$



(c) $M = 20$

Figure 6: Variation in learning performance with M

Solution 3

Code sumbitted as: `q3.py`
Run as: `python q3.py <S,T,B>`

Demonstrating the various neuronal activity phases using Rulkov Map

The following are the parameter values, along with the obtained spiking graphs obtained using those parameter values in the Rulkov map, for the various phases of neuronal activity.

1. **Silence Periods**

$$\boldsymbol{\alpha = 4,\ \mu = 0.001,\ \sigma = -0.01,\ x_0 = 0.5,\ y_0 = -2.8}$$

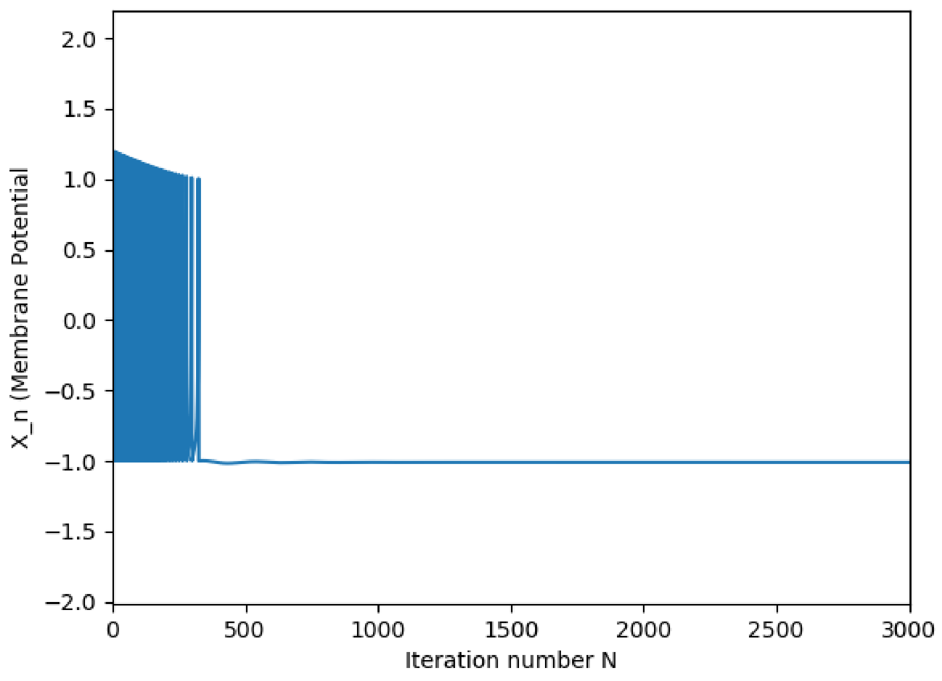


Figure 7: Silence Phase of neuronal activity

2. **Tonic Spiking**

$$\boldsymbol{\alpha = 4,\ \mu = 0.001,\ \sigma = 0.01,\ x_0 = 1,\ y_0 = -3}$$

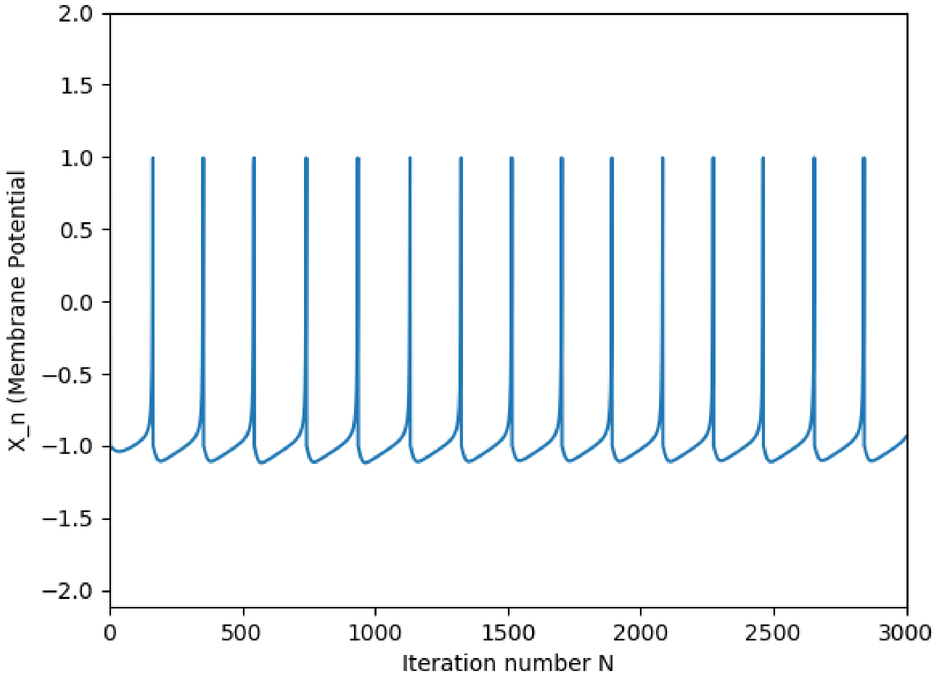


Figure 8: Tonic spiking phase of neuronal activity

3. **Burst spiking**

$$\boldsymbol{\alpha = 6,\ \mu = 0.001,\ \sigma = 0.14,\ x_0 = -1,\ y_0 = -4}$$

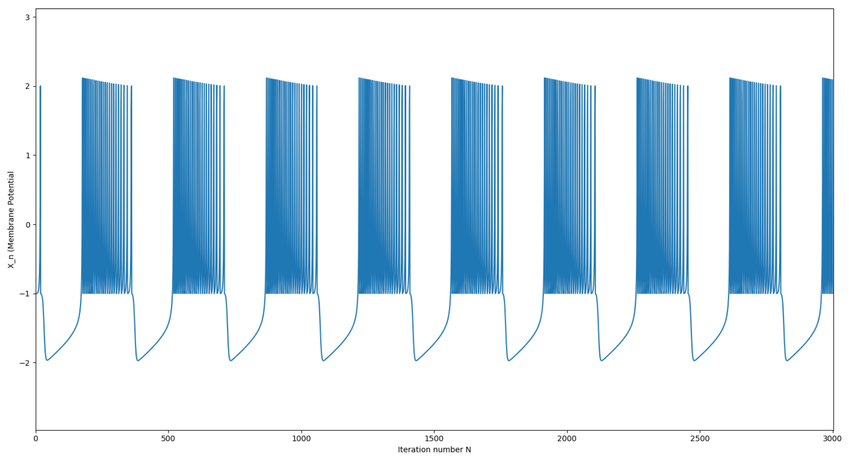


Figure 9: Burst spiking phase of neuronal activity

Emperical Demonstration of parameter ranges

Sl. No.	$\alpha - value$	$\sigma - value$	Type of Neuronal Activity	Demonstrating figure
1	2	0.55	Silence	12a
2	2	0.6	Tonic Spiking	12b
3	4	-0.01	Silence	7
4	4	0.01	Tonic Spiking	8
5	4.5	-0.5	Silence	12c
6	4.5	0	Burst Spiking	12d
7	4.5	0.5	Tonic Spiking	12e
8	6	-0.5	Silence	12f
9	6	-0.4	Burst Spiking	13a
10	6	0.14	Burst Spiking	9
11	6	0.3	Burst Spiking	13b
12	6	0.4	Tonic Spiking	13c
13	8	-0.85	Silence	13d
14	8	-0.8	Burst Spiking	13e
15	8	0	Burst Spiking	13f
16	8	0.6	Burst Spiking	13g
17	8	0.7	Tonic Spiking	13h

On plotting these points on the two dimensional parameter plane of α vs σ, we get the plot as shown in Figure 10.

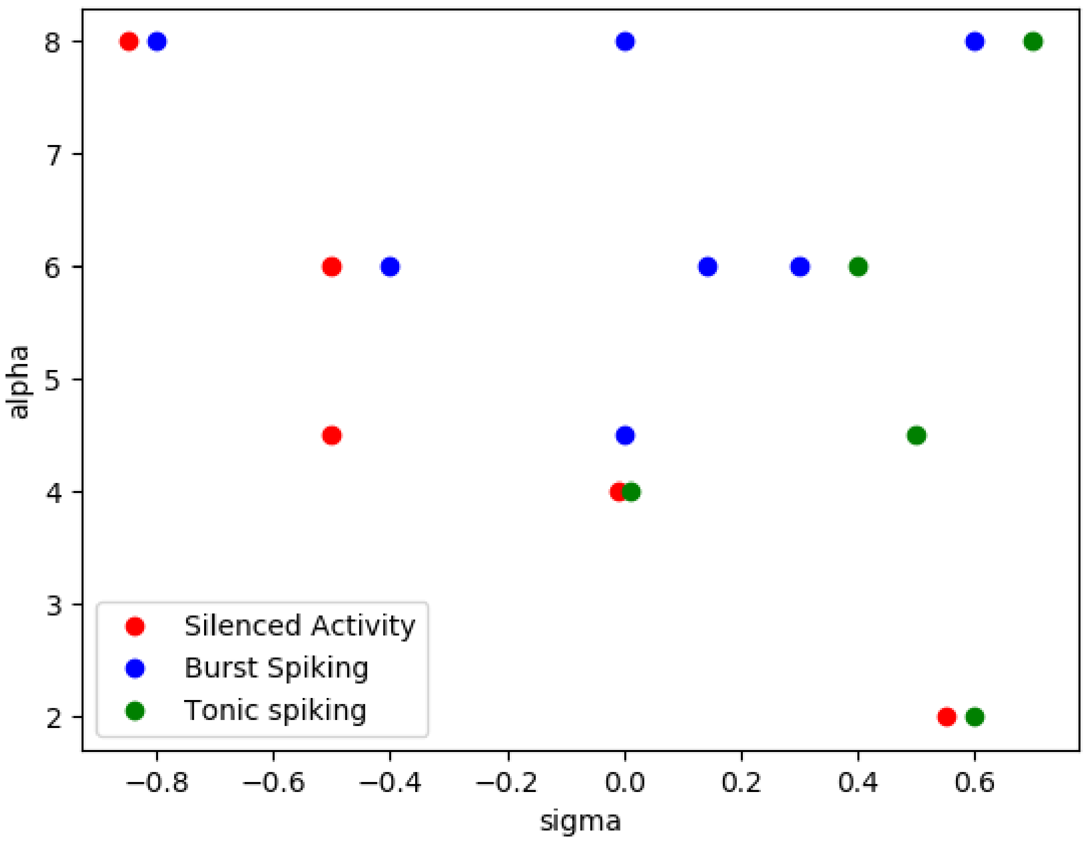


Figure 10: Emperically obtained neuronal activity for various parameter pair values of (σ, α)

The relation between the parameter values and the different phases is summarised by Rulkov et al [1] as shown in Figure 11.

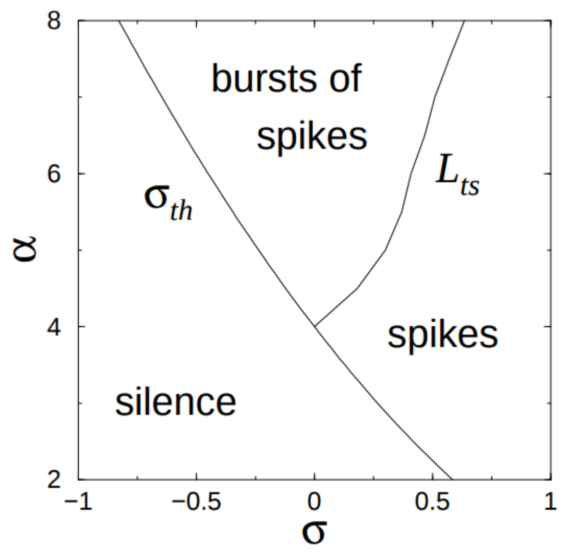


Figure 11: Bifurcation diagram on the parameter plane (σ, α) [1]

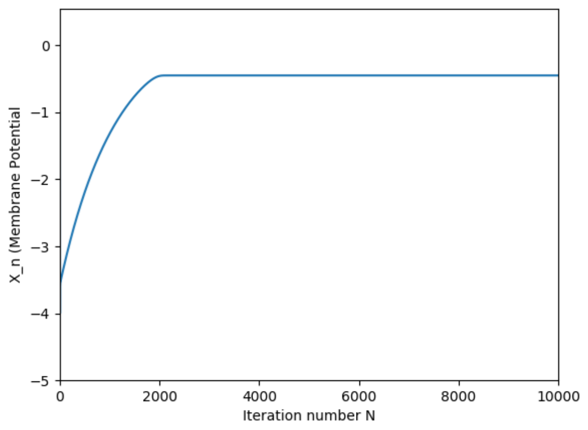
We can clearly see that the empirical results obtained are in accordance with the decision boundaries obtained by Rulkov et al [1] in their work.

References

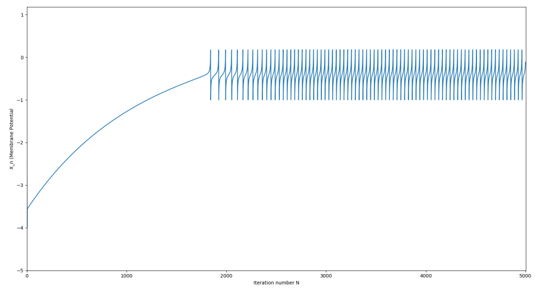
[1] N. F. Rulkov, "Modeling of spiking-bursting neural behavior using two-dimensional map.," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 65 4 Pt 1, p. 041922, 2002.

[1]Image Courtesy: N. F. Rulkov, "Modeling of spiking-bursting neural behavior using two-dimensional map."

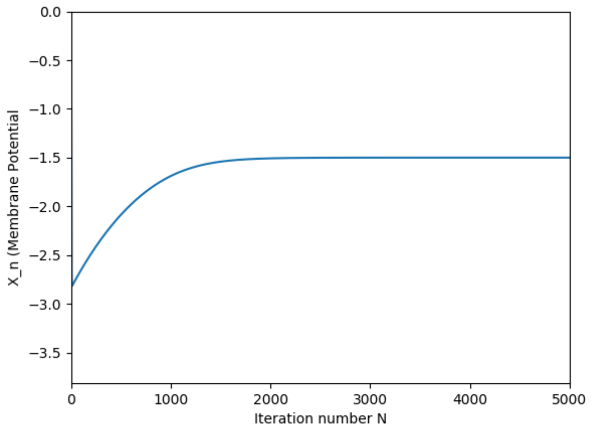
Supplementary Plots



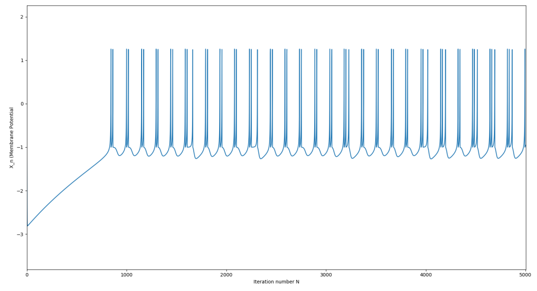
(a) $\alpha = 2,\ \sigma = 0.55$



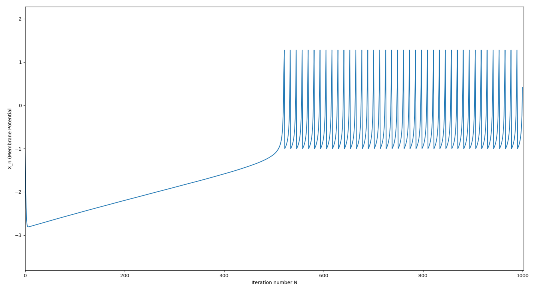
(b) $\alpha = 2,\ \sigma = 0.6$



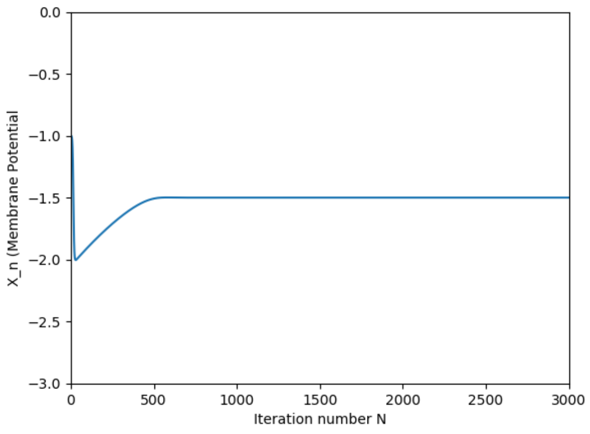
(c) $\alpha = 4.5,\ \sigma = -0.5$



(d) $\alpha = 4.5,\ \sigma = 0$

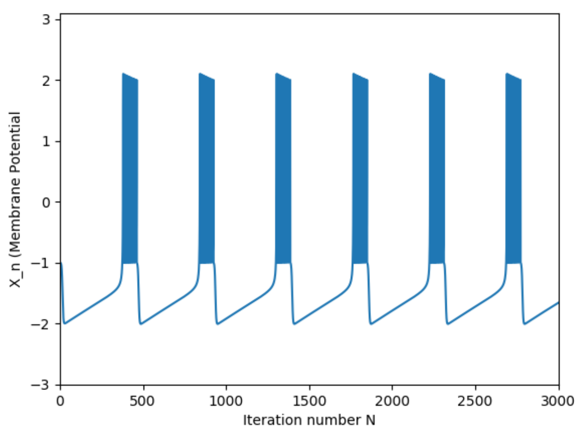


(e) $\alpha = 4.5,\ \sigma = 0.5$

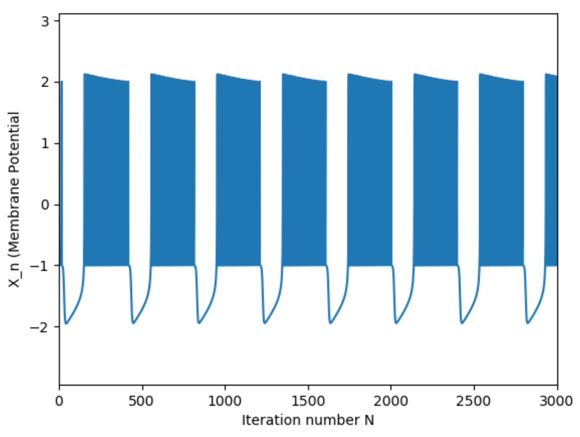


(f) $\alpha = 6,\ \sigma = -0.5$

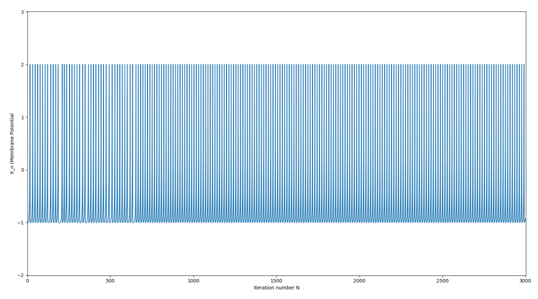
Figure 12



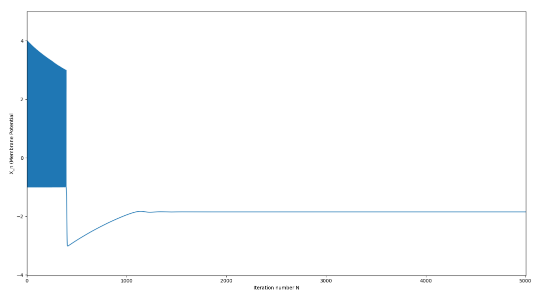
(a) $\alpha = 6,\ \sigma = -0.5$



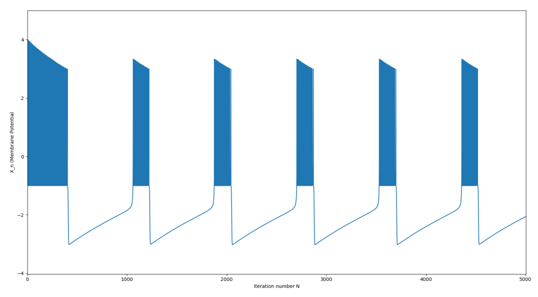
(b) $\alpha = 6,\ \sigma = 0.3$



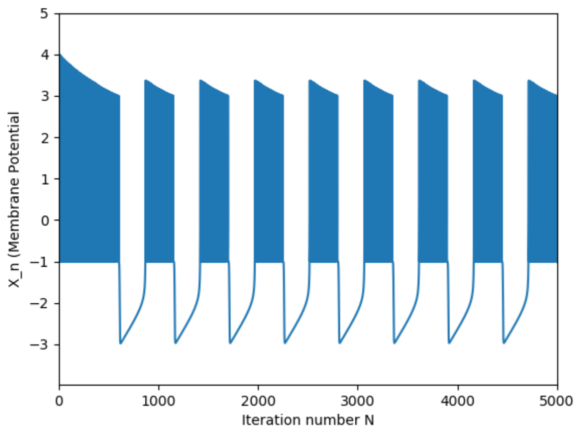
(c) $\alpha = 6,\ \sigma = 0.4$



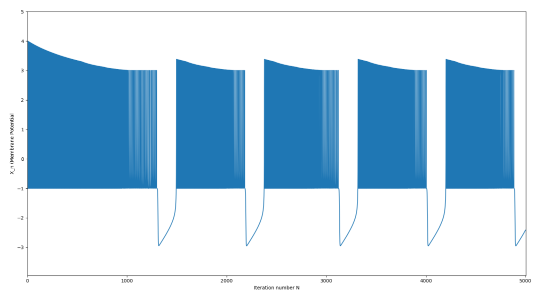
(d) $\alpha = 8,\ \sigma = -0.85$



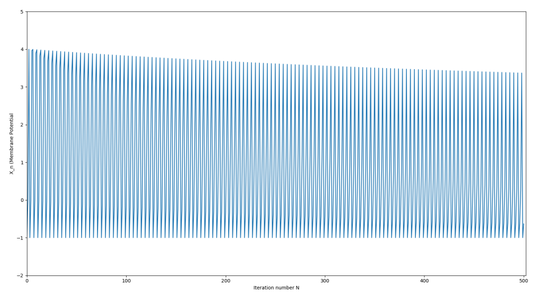
(e) $\alpha = 8,\ \sigma = -0.8$



(f) $\alpha = 8,\ \sigma = 0$



(g) $\alpha = 8,\ \sigma = 0.6$



(h) $\alpha = 8,\ \sigma = 0.7$

Figure 13