EQ2341 - Assignment1

Matay Mayrany - mayrany@kth.se

April 2022

1 Question 1

To verify that P(Sj = 1) is equal for all t, we can multiply the initial state distribution q with A, the transition matrix.

$$q = [0.75, 0.25]$$

$$A = [[0.99, 0.01], [0.03, 0.97]]$$

$$p1 = q * A = [0.75, 0.25]$$

Similarly p2, p3, p4... will result in the constant distribution. Since the result are equal to the initial state distribution then we know that q is also the stationary distribution and that P(Sj = 1) is equivalent for all t.

2 Question 2

The frequencies of the elements generated from the random walk in the python code match the distribution that we confirmed earlier and can be seen below.

```
0: 0.7488
1: 0.2512
```

3 Question 3

Theoretically calculated mean and variance:

$$M_1 = 0, std_1 = 1, P(S1) = 0.75, M_2 = 3, std_2 = 2, P(S2) = 0.25$$

$$E[X] = E_S[E_X[X|S]]$$

$$E[X] = P(S1) * M_1 + P(S2) * M_2$$

$$E[X] = 0.75 * 0 + 0.25 * 3$$

$$E[X] = 0.75$$

$$var[X] = E_S[var_X[X|S]] + var_S[E_X[X|S]]$$

$$var[X] = (P(S1)*std_1^2 + P(S2)*std_2^2) + (P(S1)*(M-M_1)^2 + P(S2)*(M-M_2)^2)$$

$$var[X] = 0.75 * 1^2 + 0.25 * 2^2 + 0.75 * (0.75-0)^2 + 0.25 * (0.75-3)^2$$

$$var[X] = 0.75 + 1 + 0.75^3 + 0.25 * (2.25)^2$$

$$var[X] = 3,4375$$

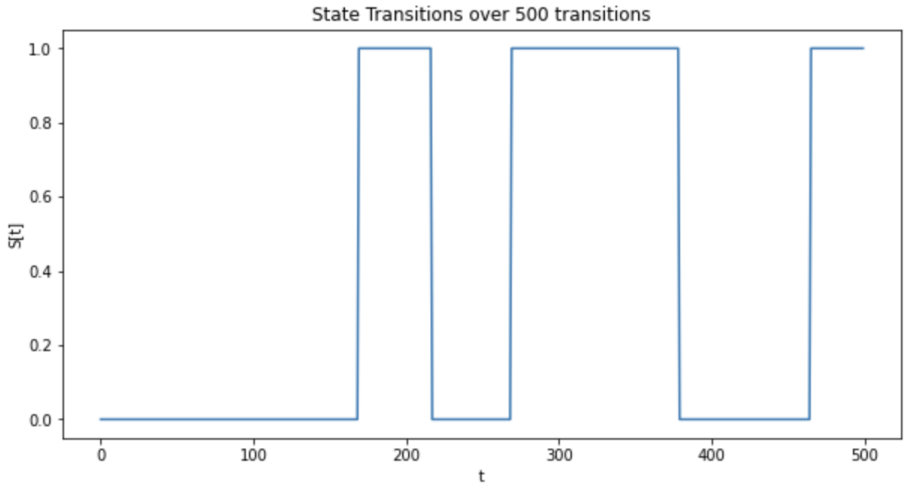
Practically calculated mean and variance:

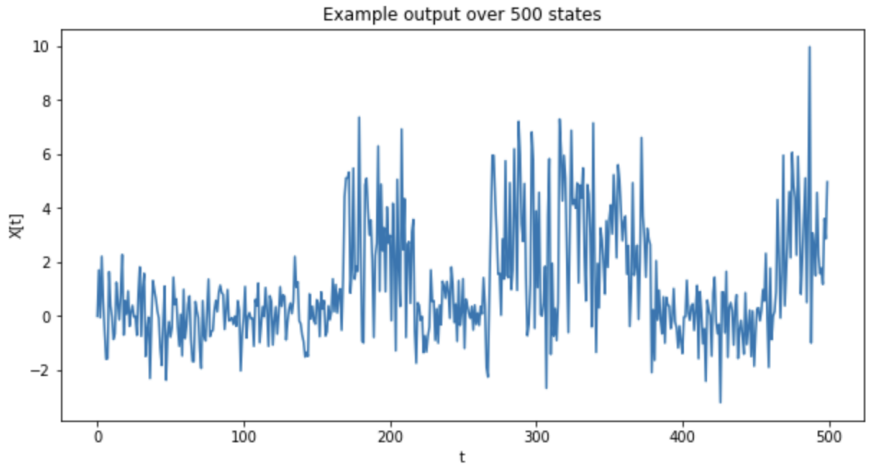
```
var:  3.4428407378854686
mean:  0.7425795036903823
```

Results from the program match the calculated values.

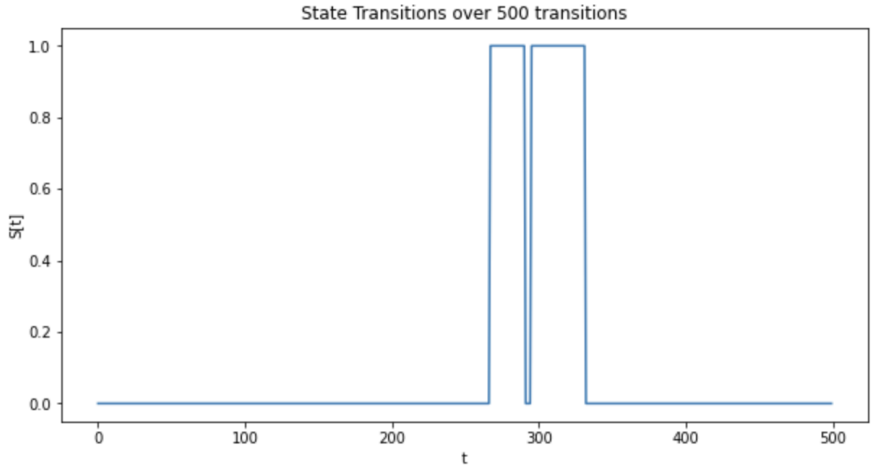
4 Question 4

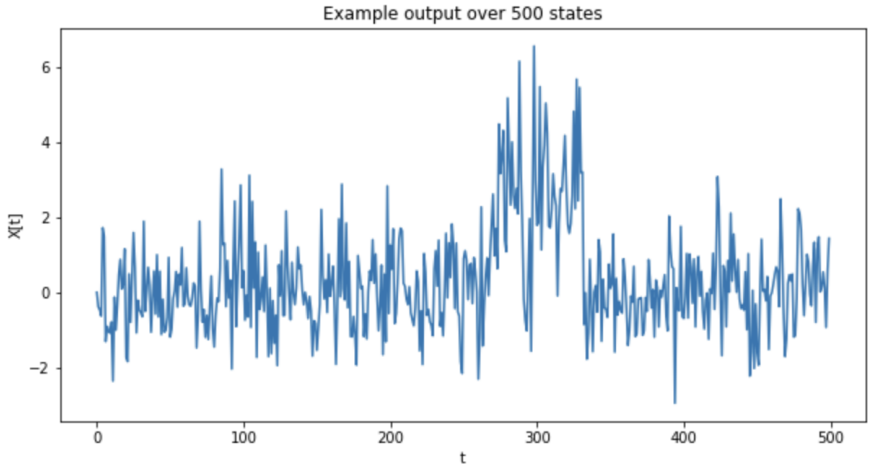
The output of the HMM is shifted upwards and more spaced out when we are in the second state, since that state has higher mean and std values. The graphs below from an example run demonstrate this behavior.





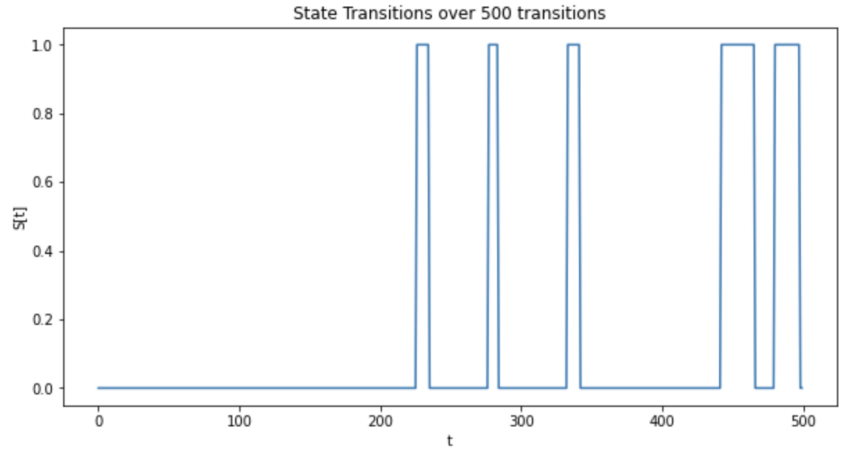
In some runs, one state is more prevalent than the other since the Markov Chain heavily favors returning from one state to itself rather than transitioning to the other. Hence, if the first state is more prevalent then a larger part of the graph is more compact. This is shown in the graphs below from an example run.

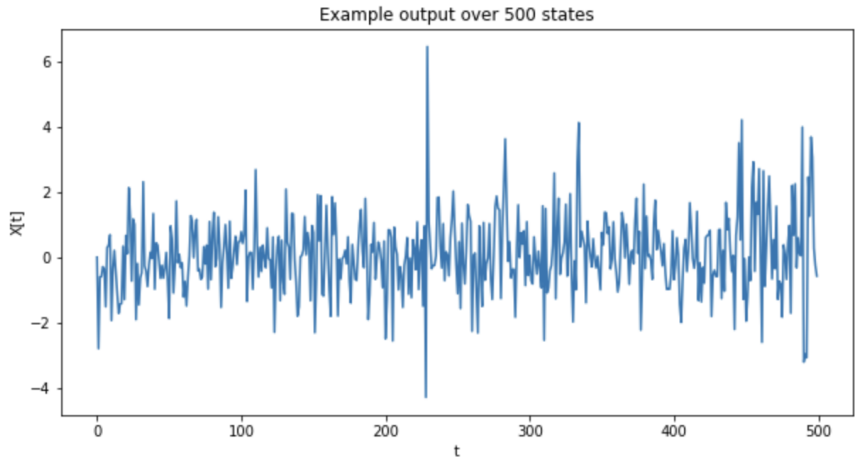




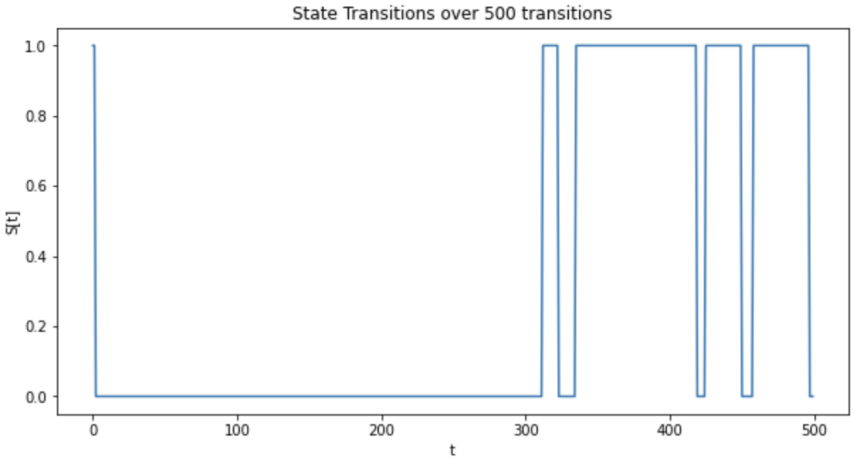
5 Question 5

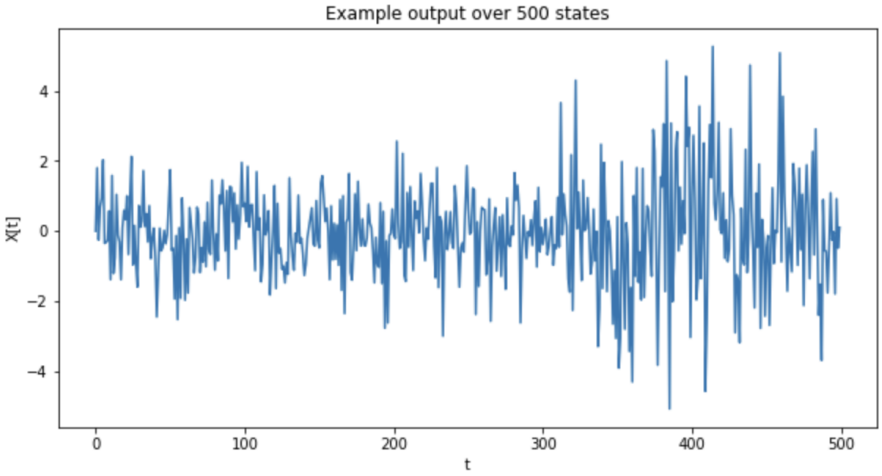
The output of the HMM is no longer shifted upwards since the means are the same now. However there is still a spread that can be used to identify the occurrence of the second state when bigger spikes occur, since that state has a higher standard deviation. The graphs below from an example run demonstrate this behavior.





However it is harder to distinguish the states from each other from the outputs with this model, specially in the cases where the changes are rapid and a state can go unnoticed, or falsely classified. The graphs below from an example run demonstrate this behavior.

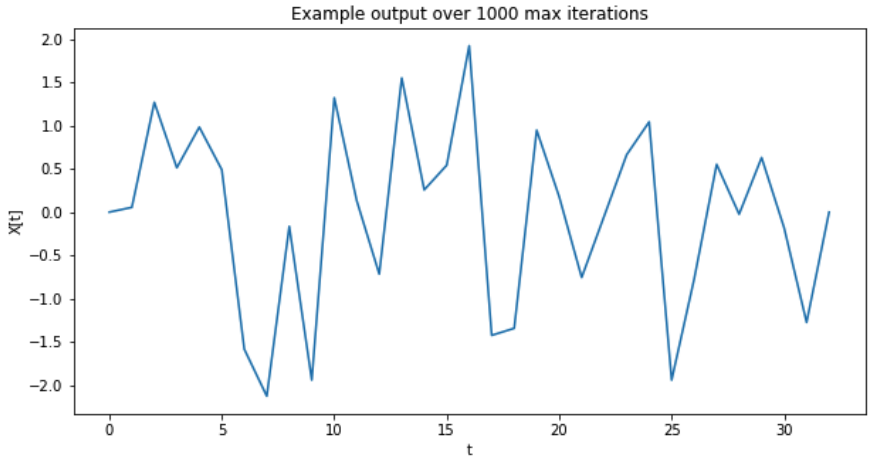


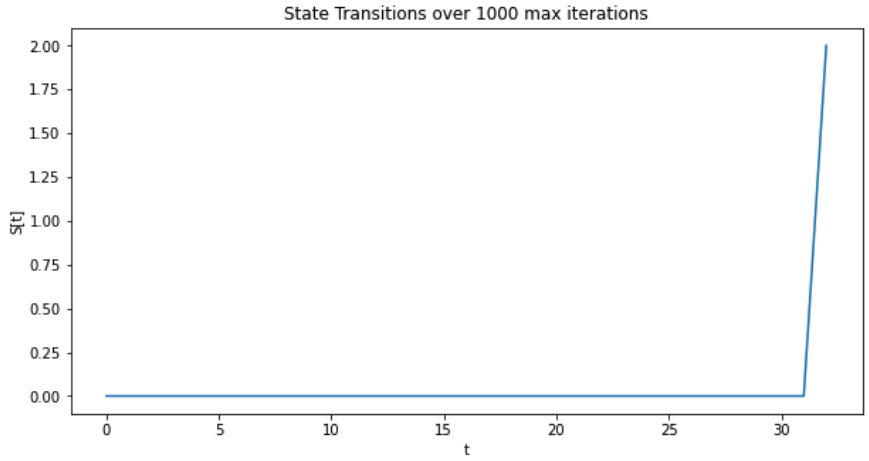


6 Question 6

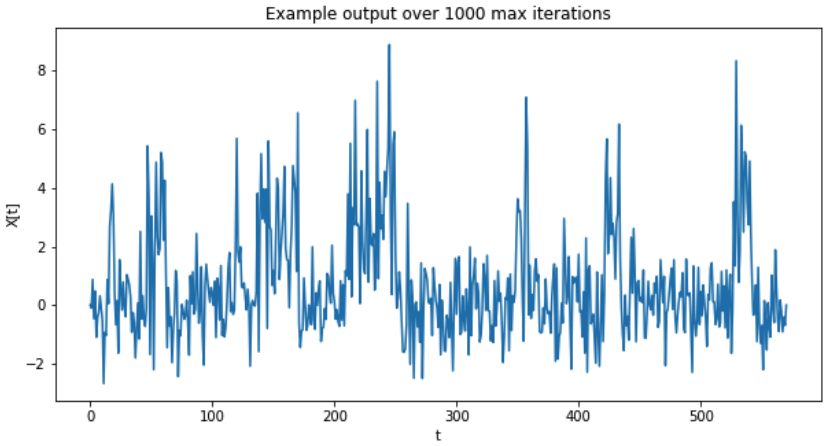
To test whether the HMM and MarkovChain implementations work with finite duration scenarios we can add another state to the chain that does not have any transition probabilities to other states, making it the end state. The value in the initial probability distribution for this state will be set to 0 so that we do not start with it, and the transition probability to the state from the other two states will be low, so that is unlikely to reach the state very quickly and we can observe the behavior better.

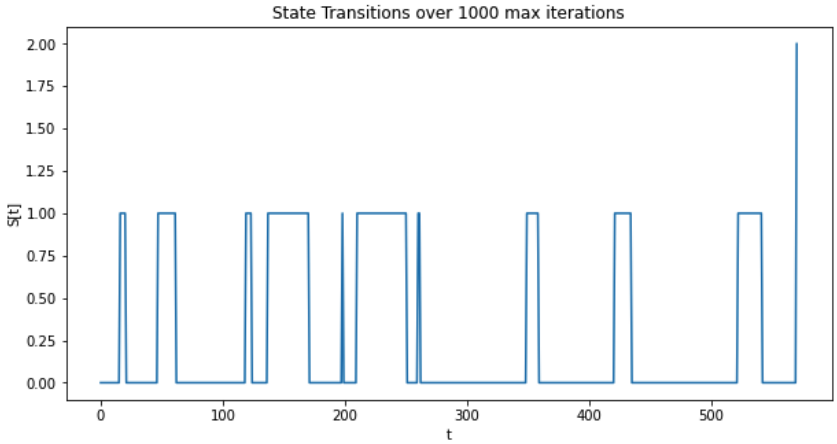
The results make sense and show that the implementation works with finite duration chains as we see the number outputs from the HMM stop once we reach the end state 2. This can be seen in the figures below, which were created by example runs where one run was stopped quickly after starting. Here we start with state 0 and stay there until we switch to state 2, hence we reached the end state quickly (33/1000 iterations). The other run took more time to reach the final state and also stopped outputting values when state 2 was reached (571/1000 iterations). The output values make sense corresponding to the states where the output matches the description discussed in the previous section with regards to state 0 and state 1, and no output is produced for state 2.






```
maximum iteration:  1000
total number of states reached before exiting: 33
```






```
maximum iteration:  1000
total number of states reached before exiting: 571
```

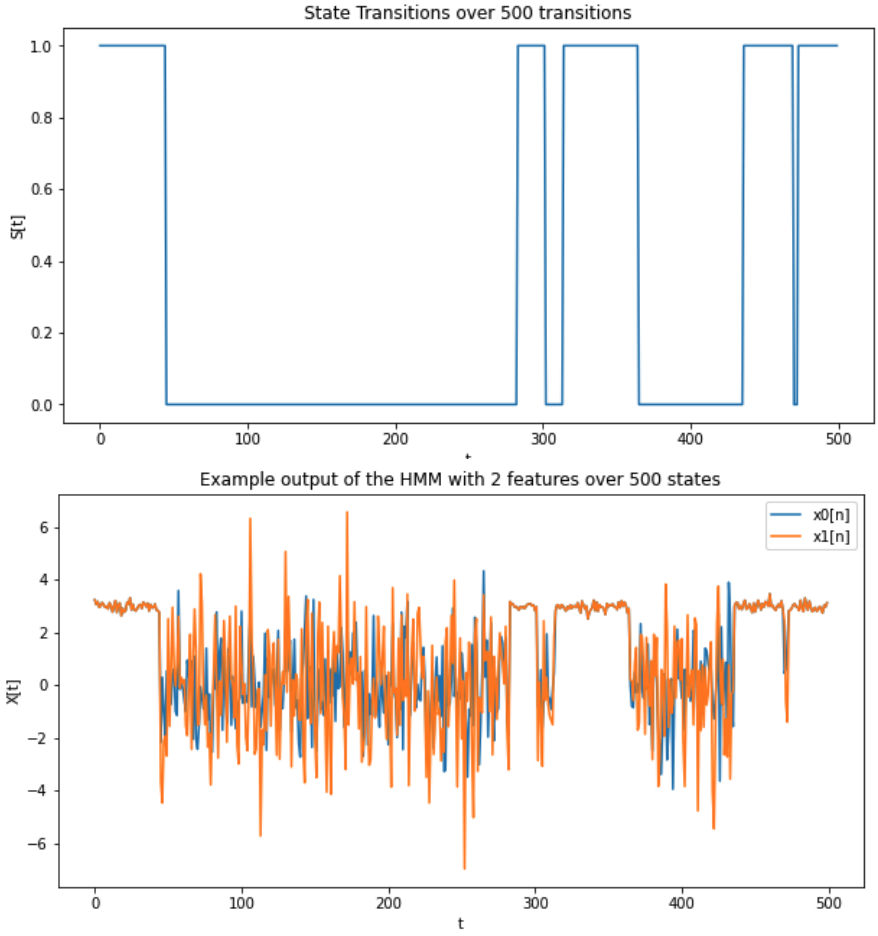
I also ran this 100 times, each with 1000 iterations and took the average cutoff point from the runs. The results showed that the average cutoff point was around 600/1000 iterations and can be seen below:

```
average cut off over 100 runs:  628.46
```

7 Question 7

To confirm that the implementation works with vector outputs as well as scalar outputs I created a new HMM instance with two vector Gaussian distribution outputs, each representing a feature that can be outputted from the HMM. one of them has a diagonal covariance matrix and the other a non diagonal one, where it uses the one described in Question. The results show that the

implementation works for this scenario and 2 features are outputted where the outputs match their respective distribution based on the state. Results can be seen below:



8 Notes

Code used to answer all questions can be found in its respective section in the jupyter notebook.