Alexandria University
Faculty of Engineering

Master of Computer Science program

Modern Control Systems

State Feedback Control for Intelligent Traffic Light Systems

Project Report
Spring 2021

Yousef Mohamed Zook

Contents

Introduction

Traffic is an important factor affecting human life quality in crowded cities. Increasing population and increasing individual vehicle ownership lead to an increase in traffic density. This leads to an increase in the time lost for travelers and pollution. In this project, we are simulating State feedback control system to solve this kind of problems to help reduce the traffic jam.

As the paper ***'State Feedback Control for Intelligent Traffic Light Systems'*** wrote, it's approach is to reduce this using state feedback control system and comparing it with other techniques. We have implemented the paper to show the results.

Problem

Intersections are areas that cause more traffic density because of design making hurdles. Traffic jam can be reduced by adapting the traffic light control that can be adapted to the traffic density change at the junctions. In this study, the State Feedback Controller is proposed for the traffic light system control at the four-leg intersection. Simulation of this control system is made and handled using Simulation of Urban Mobility (SUMO) in the paper, however I have used Python for implementation and simulation with python turtle and python Freegames.

Results are compared for the proposed types of Traffic Light Control Systems. It was observed that the State Feedback traffic light controller, which is the recommended method from the simulation results, gives better results than both the Fuzzy Logic Control (FLC) and the Fixed Time traffic light controller.

Intelligent Transportation Systems (ITSs)

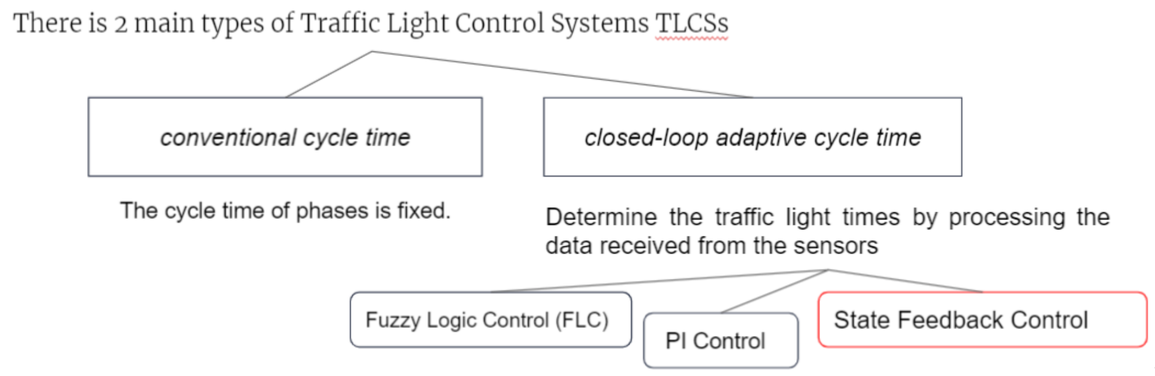
The intelligent transportation systems are systems that helps in reducing the traffic jams using new technologies and hardware. It has been developed in order to regulate the elements that cause traffic density. It determines the optimum green light duration using modern modeling systems with tools like sensors, cameras and more. Solutions proposed by ITS can also be used in Traffic Light Control Systems TLCSs.

Traffic Light Control Systems (TLCSs)

There are 2 main types of TLCSs: conventional cycle time and closed-loop adaptive cycle time. Closed-loop adaptive Traffic Light control Systems determine the traffic light times by processing the data received from the sensors. The cycle time of phases in conventional TLCSs is fixed. As a result, traffic density can increase because of variant traffic conditions. This drawback effects emission and waiting time. On the other hand, traffic lights can be regulated with the green light time, the cycle time of traffic light and lost time of traffic light in closed-loop adaptive TLCSs. The cycle time is a complete sequence of traffic signal lights: green light, red light, and yellow light.

Lost time is the time interval between changing lights. The lost time and the cycle time play a great role in TLCSs. Furthermore, the green light time needs to be adapted. To choose the efficient green light period, several control strategies have been proposed including Fuzzy Logic Control (FLC) and PI control. The green light time determination is a trade-off because not only using a short period but also using a long period for a green light period can cause traffic density. To choose the efficient green light period, several control strategies have been proposed including Fuzzy Logic Control (FLC). TLCS with FLC structure have been shown to reduce waiting times and offer significant advantages.

In this project we are discussing the State feedback control system to solve this problem and measure the best green light time for an intersection.


There is 2 main types of Traffic Light Control Systems TLCSs
conventional cycle time
closed-loop adaptive cycle time
The cycle time of phases is fixed.
Determine the traffic light times by processing the data received from the sensors
Fuzzy Logic Control (FLC)
PI Control
State Feedback Control

Traffic Light Control System

In this part we will discuss the fuzzy control system and the state feedback control system. We will set the assumptions for the system and discuss how can we achieve the desired control using state feedback system.

Assumptions

The TLC at the intersection is designed according to the following assumptions and limitations:

1. Right and left turns are not allowed at the intersection
2. The minimum time for the green light in both directions is 6 seconds.
3. Maximum green light duration for FLC and State Feedback Traffic Light Controllers is 100 seconds.
4. No amber time
5. Starting Vehicles number [enter + exit] are random.
6. North-South green light is on initially

The red points are assumptions set by me while implementing the project.

Traffic Light Design with Fuzzy Logic Controller

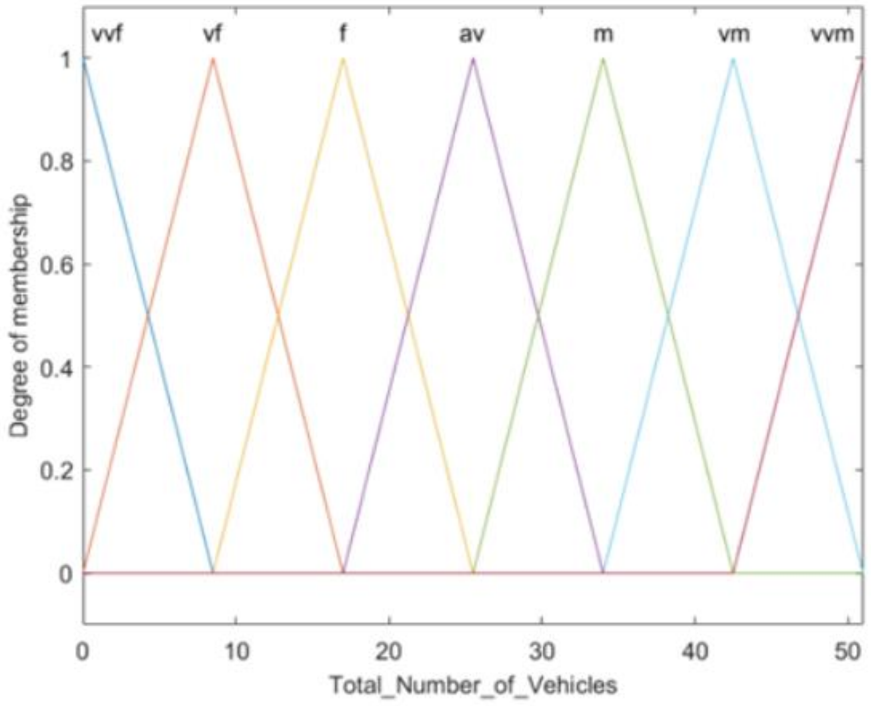
FLC works similarly to what people think. With the rules used, FLC can be used close to human thought. This method is also very useful for traffic light control. If you can accurately express the amount of traffic flow with the rules, you can get proper results.

Two fuzzy input variables have been selected in the traffic lights controller. The first is the total number of vehicles (TNV) at the intersection. The other variable is the difference between the number of vehicles (VND) coming from the east and west and the total number of vehicles coming from the north and south. The fuzzy controller has 7 membership functions which are very very few (vvf), very few (vf), few (f), average (av), much (m), very much (vm) and very very much (vvm).

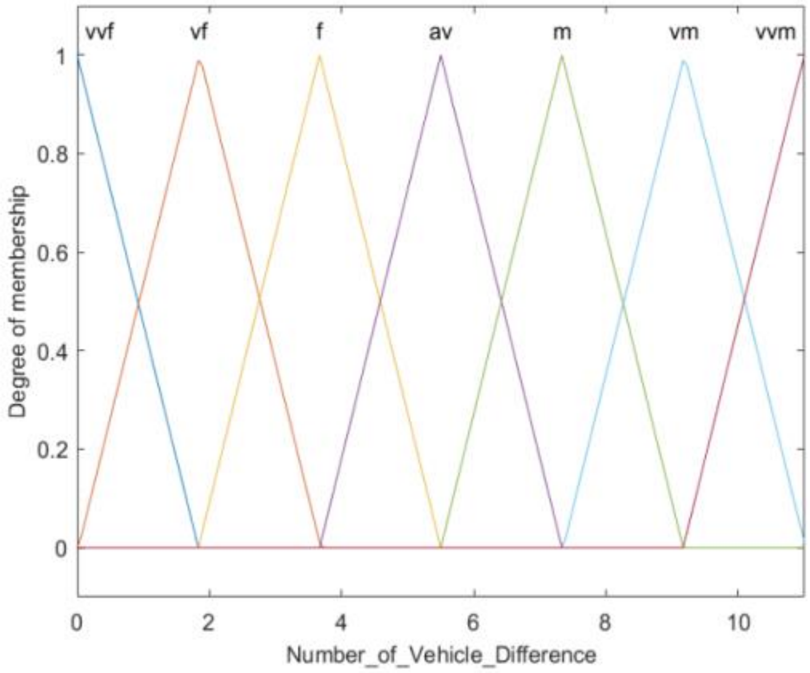
The following is the Rule Table for fuzzy logic:

TNV / VND	vvf	vf	f	av	m	vm	vvm
vvf	vvf	vvf	vf	vf	f	f	av
vf	vvf	vf	vf	f	f	av	m
f	vf	vf	f	f	av	m	m
av	vf	f	f	av	m	m	vm
m	f	f	av	m	m	vm	vm
vm	f	av	m	m	vm	vm	vvm
vvm	av	m	m	vm	vm	vvm	vvm

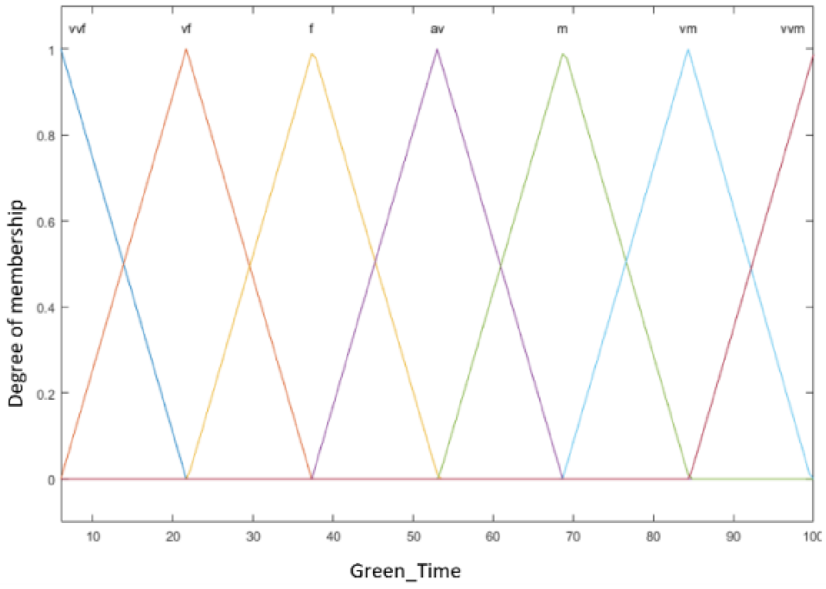
The first input [Total number of vehicles] Membership function is shown in the following plot:



The second input [Number of vehicle difference] Membership function is shown in the following plot:



The green phase is recalculated every second, so the green phase times change dynamically. The green phase duration, which is the FLC output value, is calculated according to the FLC input values. A graphical representation of the membership functions of the output value is given in the following figure:



State Feedback for intelligent traffic light systems

In this section we will discuss the state feedback model for the intelligent traffic light system.

Inputs

The inputs for the system are the number of vehicles coming from each direction [q].

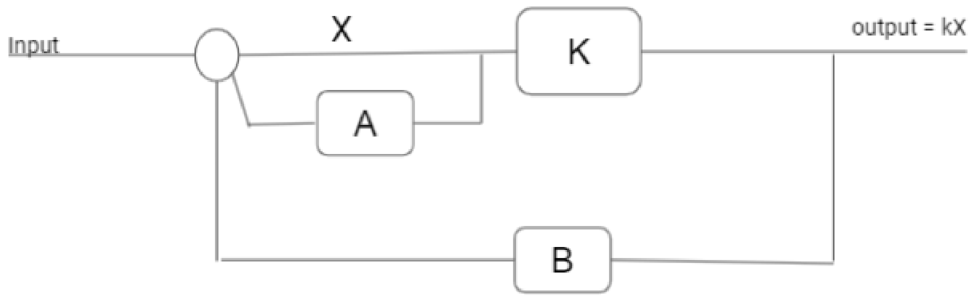
And also, the number of vehicles leaving the intersection from each direction [d].

Outputs

The output is a vector [S] which indicates at which direction should be allowed to pass, e.g., green light should be on in this direction, and which is not.

System Model Diagram

The model block diagram can be shown in the following diagram:



As shown the input are the

System State Equations

Dynamic equations of the system are derived in discrete time. The system equations are given using two state variables named as Qi and Wi. Q1 is the sum of vehicle numbers coming from east. Q2 is the sum of vehicle numbers coming from west. Similarly, Q3 and Q4 are the sum of vehicles from the north and the south, respectively. T is the sampling time. The state equations are given as follows:

$$Q_i(n+1) = Q_i(n) + q_i(n) - d_i(n)S_i(n) \qquad (1)$$

$$W_i(n+1) = W_i(n) + TQ_i(n) + \frac{1}{2}Tq_i(n) - \frac{1}{2}Td_i(n)S_i(n) \qquad (2)$$

The other state variable W represents the waiting times of the vehicles for each direction. That is W1 and W2 are the sum of waiting times of vehicles coming from the east and the west, respectively. On the other hand, W3 and W4 are the sum of waiting times of vehicles coming from the north and the south, respectively. In equation (1) and equation (2), di is the number of vehicles leaving the intersection, qi is the number of vehicles entering the intersection.

When the green light is on in the east and west directions, the input signal S1 is applied. The S2 input signal is applied for the north and south direction.

$$\mathrm{X}(n+1) = AX(n) + B(n)S(n) + C(n) \qquad (3)$$

$$\mathrm{X}(n) = \left[Q_{12}(\mathrm{n}) - \mathrm{Q}_{34}(n) \quad \mathrm{W}_{12}(n) - \mathrm{W}_{34}(n)\right] \qquad (4)$$

$$\begin{aligned} Q_{12}(\mathrm{n}) &= Q_1(\mathrm{n}) + \mathrm{Q}_2(n) \\ Q_{34}(\mathrm{n}) &= \mathrm{Q}_3(\mathrm{n}) + \mathrm{Q}_4(n) \end{aligned} \qquad (5)$$

$$\begin{aligned} \mathrm{W}_{12}(n) &= \mathrm{W}_1(n) + \mathrm{W}_2(n) \\ \mathrm{W}_{34}(n) &= \mathrm{W}_3(n) + \mathrm{W}_4(n) \end{aligned} \qquad (6)$$

$$\mathrm{S}(n) = \left[S_1(n) \quad \mathrm{S}_2(n)\right]^T \qquad (7)$$

$$\begin{aligned} S_1 &= [1 \quad 0]^T \\ S_2 &= [0 \quad 1]^T \end{aligned} \qquad (8)$$

$$A = \begin{bmatrix} 1 & 0 \\ T & 1 \end{bmatrix} \qquad (9)$$

$$B = \begin{bmatrix} -d_1 - d_2 & -d_3 - d_4 \\ -\frac{1}{2}Td_1 - \frac{1}{2}Td_2 & -\frac{1}{2}Td_3 - \frac{1}{2}Td_4 \end{bmatrix} \quad (10)$$

$$C = [1 \quad 1] \qquad (11)$$

$$C(n) = [Tq_{12} \quad Tq_{34}] \qquad (12)$$

$$\begin{aligned} q_{12} &= q_1 + q_2 \\ q_{34} &= q_3 + q_4 \end{aligned} \qquad (13)$$

$$S_1(n) = KX(n)$$

Ackermann’s Formula

To determine the gain K, Ackermann’s formula is used. The Ackermann formula is a very useful method for controlling systems with state space models especially in highgrade systems. When the desired poles are known, z=λ1, z=λ2,… z=λn for an n-th order system, the characteristic equation,

$$\alpha_c(z) = (z-\lambda_1)(z-\lambda_2)\ldots(z-\lambda_n) \qquad (14)$$

$$\alpha_c(z) = z^n + \alpha_{n-1}z^{n-1} + \alpha_{n-2}z^{n-2}\ldots\alpha_1(z) + \alpha_0 = 0 \quad (15)$$

Ackermann's formula for the gain matrix K is given by

$$K = [0 \quad 1][B \quad AB]^{-1}\,\alpha(A) \qquad (17)$$

K is a row vector of n elements. In Eq. (17), α(A) is a matrix polynomial with coefficients determined by the desired closed loop system characteristic polynomial as follows.

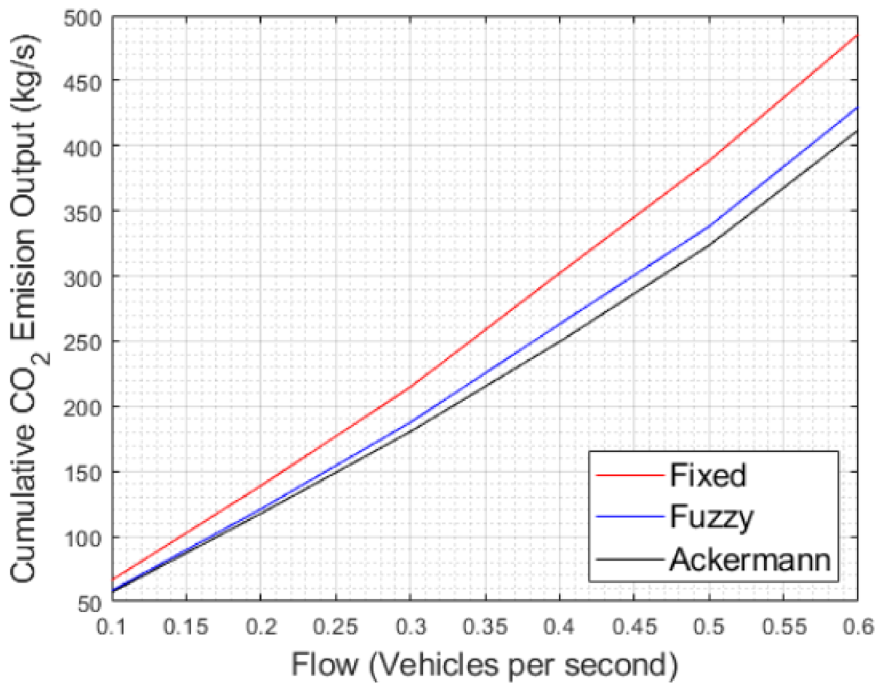
$$\alpha_c(A) = A^n + \alpha_{n-1}A^{n-1} + \alpha_{n-2}A^{n-2}\ldots\alpha_1 A + \alpha_0 \quad (18)$$

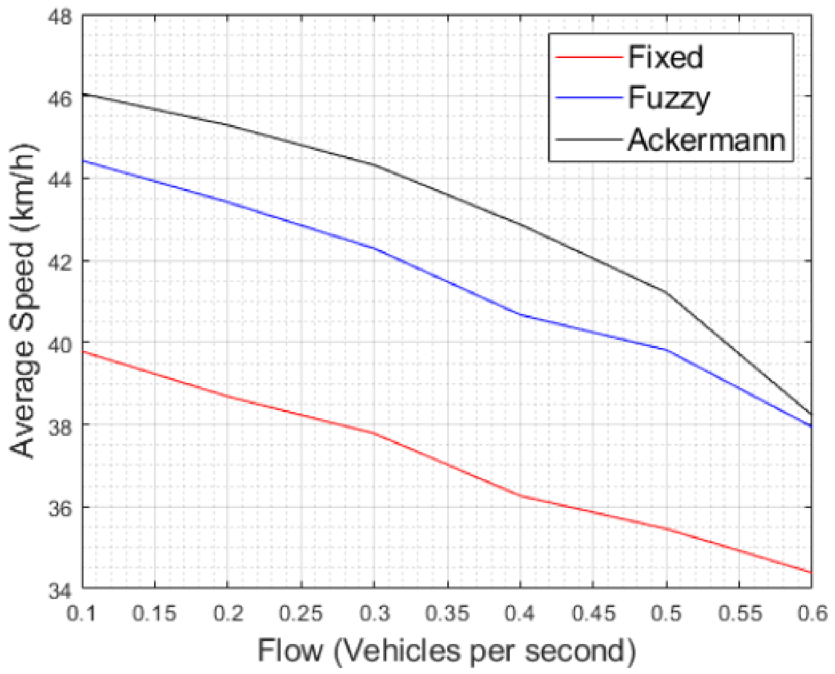
Results

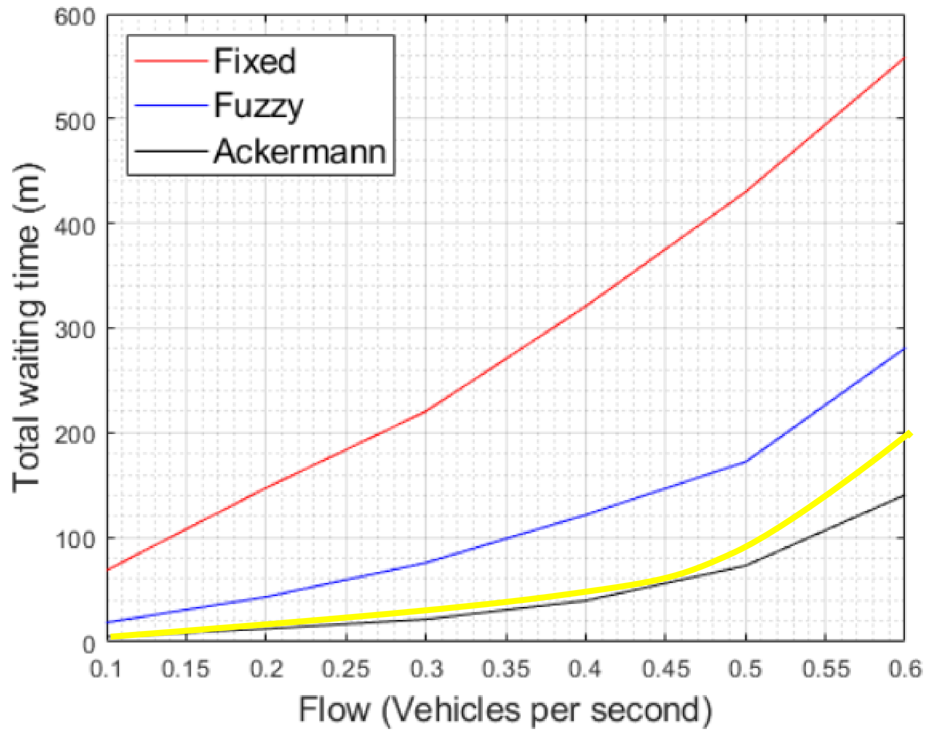
The paper simulation environment is designed and implemented using SUMO. The CO2 emission values, the average speed values of the vehicles and total waiting time are taken directly from the SUMO. Simultaneous vehicles produced for 1 hour during the simulation. In the simulation, the ratio of the number of vehicles coming from east-west direction to the number of vehicles coming from north-south direction is 1.5. Moreover, each vehicle crosses the intersection once. In the simulation, 0.1, 0.2, 0.3,0.4, 0.5 and 0.6 vehicles are produced per second to determine the vehicle density. Therefore, during simulation, 360, 720,1080, 1440, 1800 and 2160 vehicles are produced for vehicle densities of 0.1, 0.2, 0.3,0.4, 0.5 and 0.6, respectively.

Our simulation is implemented using Python and tested for 200 seconds to with random inputs between 0 to 5 for the first 80 Seconds and to 20 for the remainder.

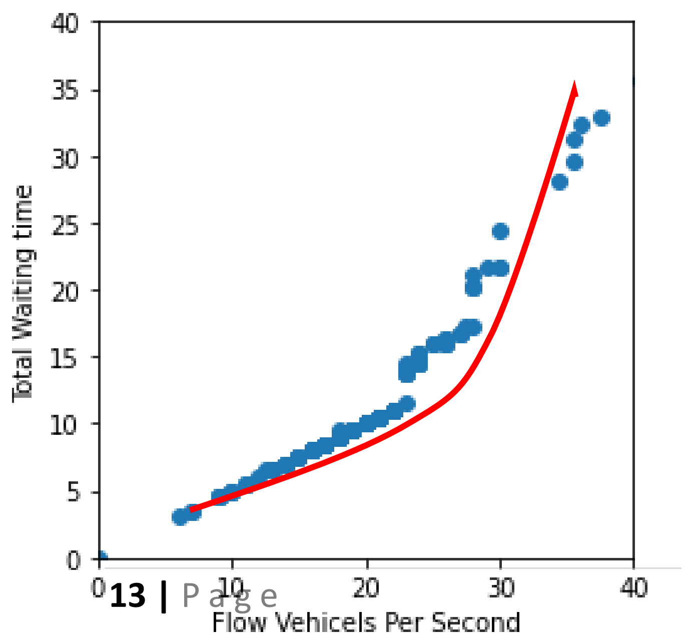
The results shown is as follow:

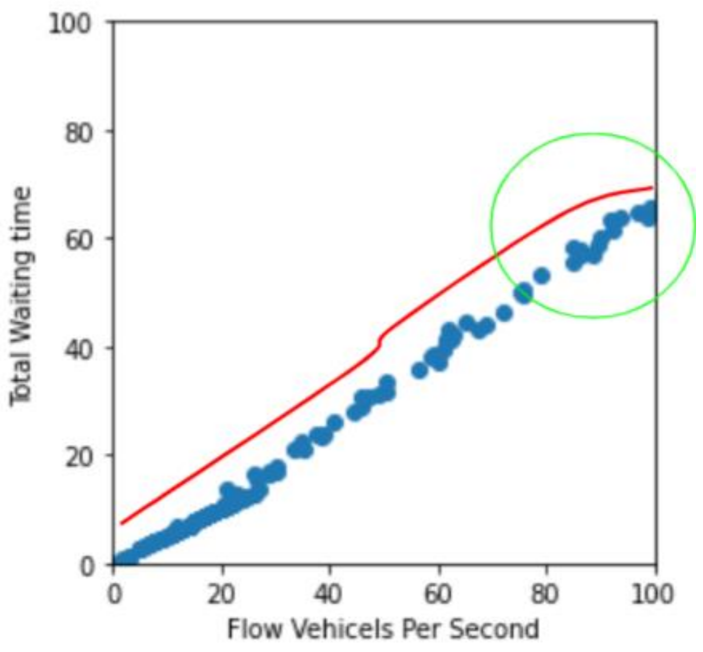






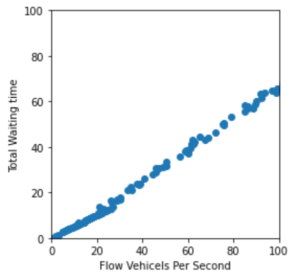
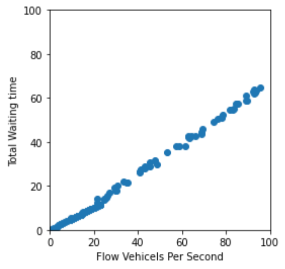
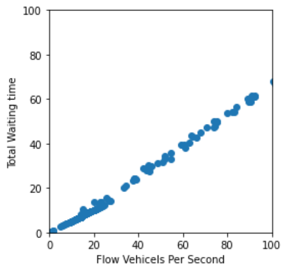
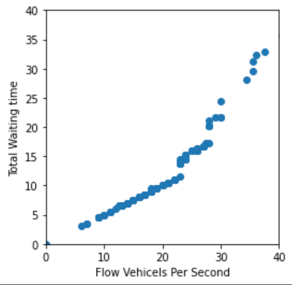
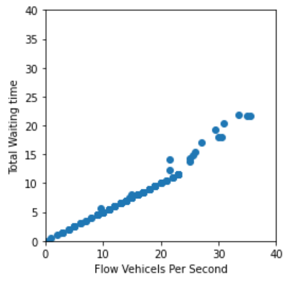
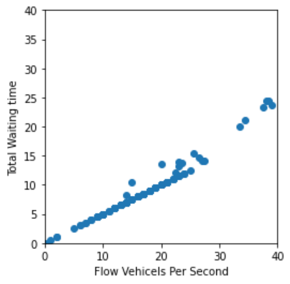
As shown above the yellow curve is our results from the simulation very near to the paper curve, below in the python simulation, when the flow increases the waiting time increases but not linearly at the end, which is indicator that the system can adapt to enhance the flow and reduce the traffic jam.





T varying

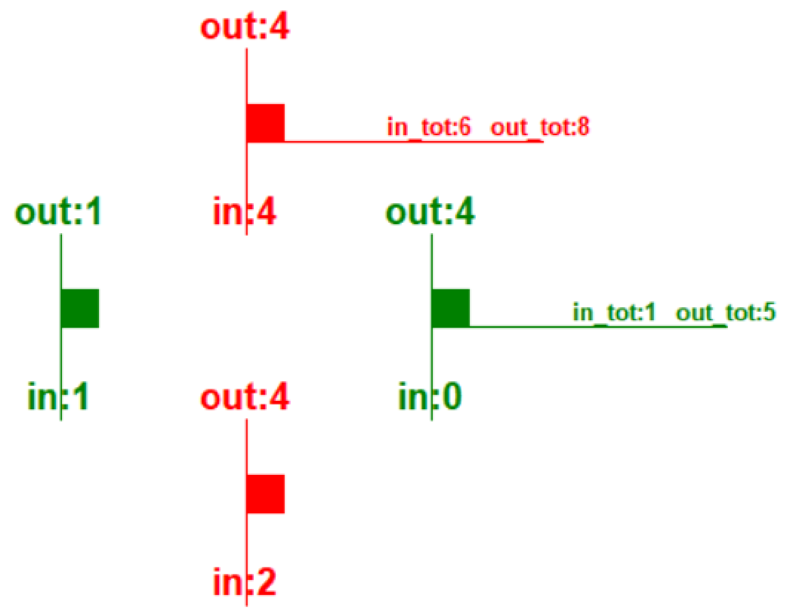
When varying the value of T, the following results are obtained:

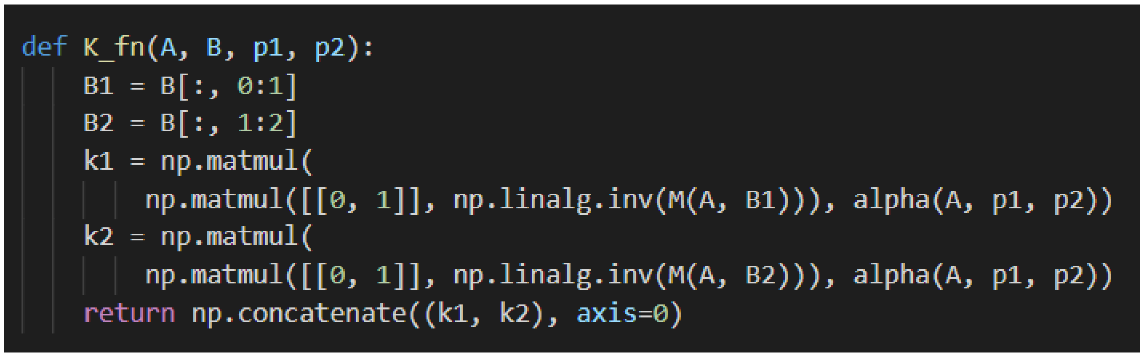
Time	1	1.5	2
100% scale			
First 40% scale			

As shown the system output is varying according to the given time sample T.

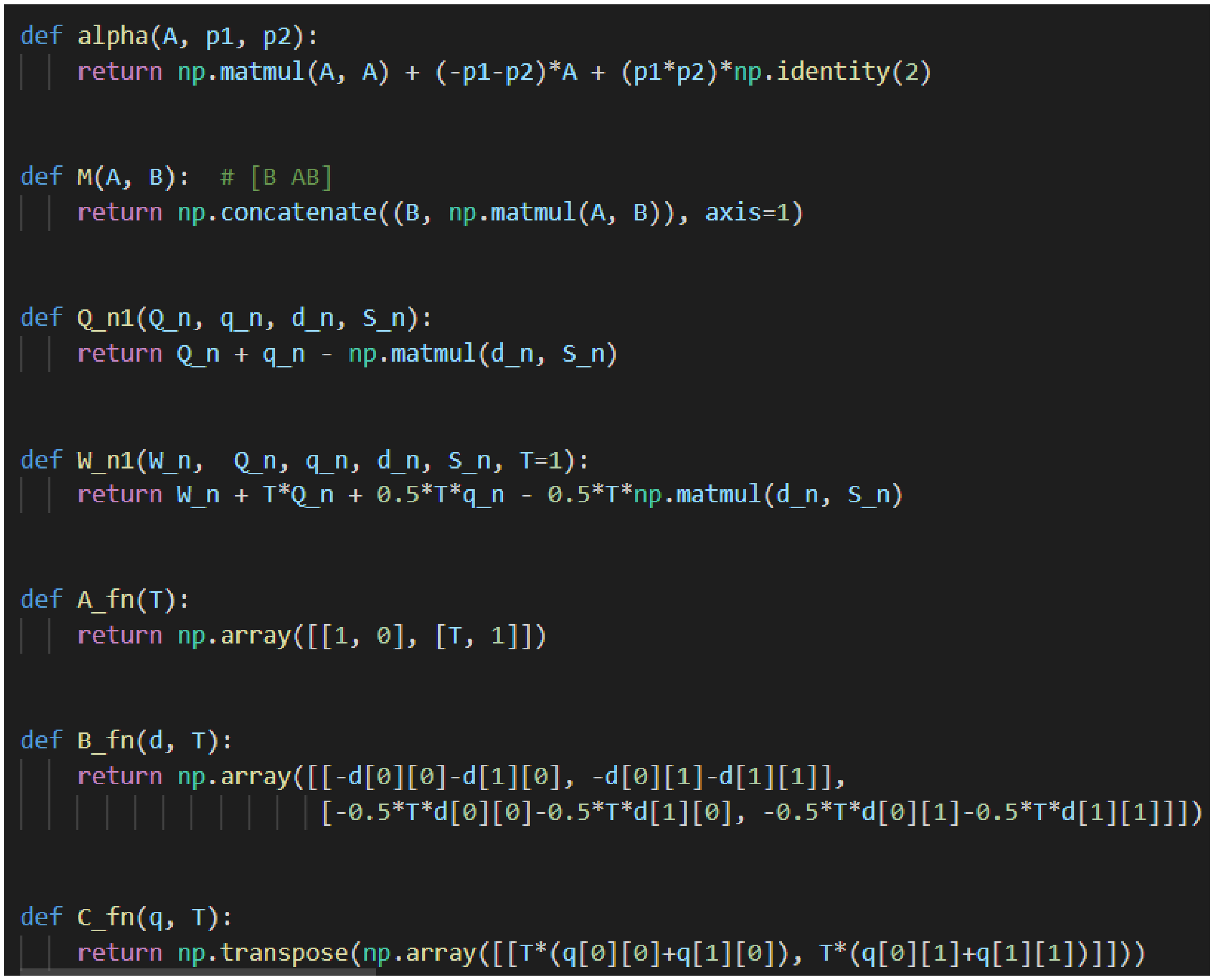
Simulation and Implementation

As mentioned, the python environment has been used for the simulation and implementation. You can find some samples of the code and a simulation picture for the model:

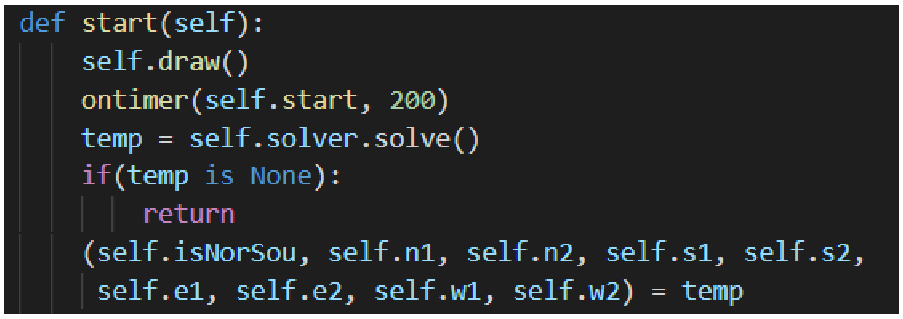




```python
def K_fn(A, B, p1, p2):
    B1 = B[:, 0:1]
    B2 = B[:, 1:2]
    k1 = np.matmul(
        np.matmul([[0, 1]], np.linalg.inv(M(A, B1))), alpha(A, p1, p2))
    k2 = np.matmul(
        np.matmul([[0, 1]], np.linalg.inv(M(A, B2))), alpha(A, p1, p2))
    return np.concatenate((k1, k2), axis=0)
```



```python
def alpha(A, p1, p2):
    return np.matmul(A, A) + (-p1-p2)*A + (p1*p2)*np.identity(2)


def M(A, B):  # [B AB]
    return np.concatenate((B, np.matmul(A, B)), axis=1)


def Q_n1(Q_n, q_n, d_n, S_n):
    return Q_n + q_n - np.matmul(d_n, S_n)


def W_n1(W_n,  Q_n, q_n, d_n, S_n, T=1):
    return W_n + T*Q_n + 0.5*T*q_n - 0.5*T*np.matmul(d_n, S_n)


def A_fn(T):
    return np.array([[1, 0], [T, 1]])


def B_fn(d, T):
    return np.array([[-d[0][0]-d[1][0], -d[0][1]-d[1][1]],
                     [-0.5*T*d[0][0]-0.5*T*d[1][0], -0.5*T*d[0][1]-0.5*T*d[1][1]]])


def C_fn(q, T):
    return np.transpose(np.array([[T*(q[0][0]+q[1][0]), T*(q[0][1]+q[1][1])]]))
```



```python
def start(self):
    self.draw()
    ontimer(self.start, 200)
    temp = self.solver.solve()
    if(temp is None):
        return
    (self.isNorSou, self.n1, self.n2, self.s1, self.s2,
     self.e1, self.e2, self.w1, self.w2) = temp
```

References:

State Feedback Control for Intelligent Traffic Light Systems | IEEE Conference Publication | IEEE Xplore

Ackermann's formula - Wikipedia

Python simulation Code:

https://drive.google.com/file/d/1pqMxcTsZDfSvgJd5BxPiAqfTiWdKChqj/view?usp=sharing

IPython solver notebook:

https://colab.research.google.com/drive/1qEkhwwdqYqAusLAYnFjLU74cjcVmL1R0?usp=sharing