LAPORAN TUGAS UAS  
PEMODELAN & SIMULASI IF-42Gab03

**Pemodelan dan Simulasi Simulasi untuk Traffic Flow Menggunakan Nagel-Schreckenberg Model**



Anggota Kelompok 9 :

Moh. Adi ikfini (1301194160)

Retno Diah Ayu N (1301194460)

Windy Ramadhanti (1301194002)

**IF 42 Gab03**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS INFORMATIKA**

**UNIVERSITAS TELKOM**

**BANDUNG**

**2022**

**Daftar ISI**

[1.](#_heading=h.gjdgxs) Deskripsi Permasalahan 3

[2.](#_heading=h.30j0zll) Pemodelan Matematika 3

[3.](#_heading=h.tyjcwt) Dokumentasi 5

[3.1](#_heading=h.3dy6vkm) Algoritma 5

[3.2](#_heading=h.1fob9te) Screenshot Program 5

[3.3](#_heading=h.3znysh7) Hasil Simulasi 6

[4.](#_heading=h.2et92p0) Kesimpulan 7

# Deskripsi Permasalahan

1. Buatlah simulasi untuk sistem tersebut dengan parameter tetap M = 100, p = 0.3, v0 = 0, d = 2, jumlah kendaraan N = 20, tmax = 1000, vmax = 5
2. Dari simulasi tersebut tentukan dan jelaskan:

(a) Gambarkan apa yang terjadi setelah sistem berjalan selama tmax.

(b) Kepadatan per satuan waktu, kendaraan di interval [x80, x90].

(c) Kepadatan maksimum per satuan waktu, kendaraan di setiap interval dengan panjang 5 unit posisi.

(d) Waktu rata-rata mobil kembali ke posisi awal

# Pemodelan Matematika

1. Dikarenakan jalur berbentuk siklis, maka setelah satu putaran, mobil akan kembali ke titik awal. Lihat ilustrasi pada Figure 1. Bayangkan sebuah karet gelang yang diputus, sehingga dapat dibentangkan menjadi sebuah garis horizontal, dimana titik-titik ujung tadinya bersatu. Kemudian jalur horizontal kita partisi menjadi M subinterval, sehingga posisi mobil dalam jalur siklis adalah dan untuk k = 1,2,... merupakan jumlah putaran.

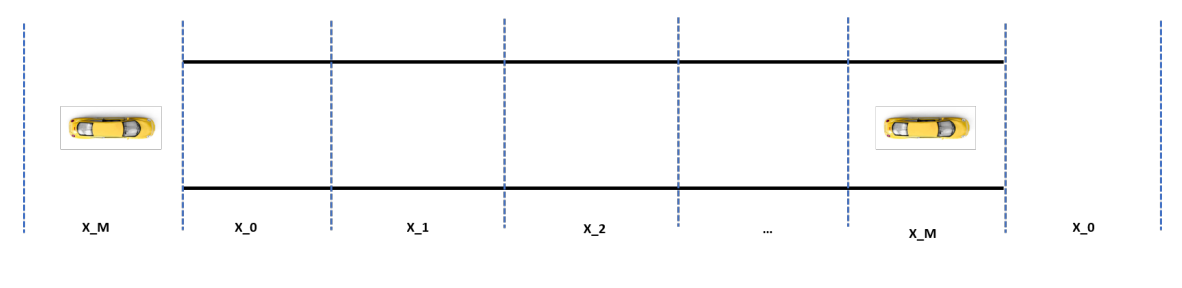
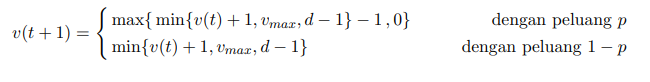


Figure 1: ilustrasi jalur siklis dibentangkan menjadi jalur horizontal. Setelah melewati posisi pengendara akan kembali ke posi

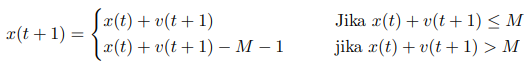
1. Terlebih dahulu kita sederhanakan sistem dengan model posisi dan yang diskrit. dan waktu diskrit Perilaku pengendara disederhanakan menjadi seperti berikut:
2. Penyederhanaan Rule-1: Penambahan 1 unit satuan kecepatan untuk kecepatan setiap mobil per satu unit satuan waktu, dan kecepatan setiap mobil tidak boleh melebihi batas kecepatan yang dibolehkan Artinya
3. Penyederhanaan Rule-2: Misalkan jarak aman antar dua mobil beriringan adalah maka kecepatan setiap mobil harus memenuhi
4. Pengemudi memiliki kecenderungan untuk mengerem dengan peluang sebesar p, yang mengakibatkan mobil mengalami perlambatan sebesar 1 satuan unit kecepatan, sehingga jika menginjak rem kecepatan menjadi (karena mobil tidak dibolehkan mundur).

Dengan kata lain melalui penyederhanaan ini, setiap kendaraan akan mengalami update kecepatan dan posisi sebagai berikut:

**update kecepatan:**



**kemudian update posisi:**



Pemisalan diatas tidak membatasi kalian, jika dapat memikirkan cara yang lebih realistik dan mungkin untuk dilakukan. Hint: Persoalan mendeteksi traffic flow seperti digambarkan diatas, dapat kalian gali lebih dalam dari sumber-sumber di internet dengan kata kunci: Nagel-Schreckenberg Model.

# ALGORITMA

<deskripsikan pseudocode / algoritma yang digunakan.>

## Algoritma Program

# Tubes Pemodelan dan Simulasi

# Moh. Adi ikfini (1301194160)

# Retno Diah Ayu N (1301194460)

# Windy Ramadhanti (1301194002)

import matplotlib.pyplot as plt

import numpy as np

import numpy.random as random

from matplotlib import animation

import operator as op

def car\_model(car):

# Inisiasi variabel untuk pemodelan

M = 100

p = float(0.3)

v = 0

N = 20

tmax = 1000

vmax = 5

D = 2

car\_in\_circle = []

c\_antri = [i for i in range(N)]

time = 0

c\_dens = {}

c\_Max = [0 for i in range(tmax)]

avg0 = 0

# Proses iterasi pemodelan

for x in range(tmax):

x\_row = []

iterasi\_car = 0

for i in c\_antri:

s\_car = car[i]

next\_c = car[i+1 if i+1 < N else 0]

#Mengupdate Kecepatan

v = np.min([v+1, vmax])

# Mengambil Jarak Antar Mobil

if (i == 0):

d = D

elif (next\_c[0] < s\_car[0]):

d = M - next\_c[0]

else:

d = (next\_c[0]-s\_car[0])

# Peluang Kecenderungan pengemudi mengerem

rem = random.rand()

if(rem < p):

v = np.max([np.min([v+1, vmax, d-1]) - 1, 0])

else :

v = np.min([v+1, vmax, d-1])

# Update Posisi

x = s\_car[0] # Mengambil nilai x dari mobil ke - i

x = x + v

# Pengecekan jika nilai x lebih dari M, maka kembali ke awal

if (x >= M):

temp = []

for i in range(N):

order = c\_antri[i] + N-1

if (order + N-1 > N):

order = order - N

temp.append(order)

c\_antri = temp

x = x - M

car[i][2] += 1

x\_row.append([x,s\_car[1],car[i][2]])

# Update Posisi

if x >= 80 and x <= 90:

iterasi\_car += 1

# Menghitung kepadatan lalulintas

c\_dens[time] = ((iterasi\_car/10)\*100)

time += 1

# Menyimpan dan menampung posisi terbaru dari mobil

car = x\_row

car\_in\_circle.append(x\_row)

# Iterasi untuk Mencari nilai kepadatan pada tiap internal waktu, dengan ketentuan 5 unit posisi

for i in range(len(car\_in\_circle)):

for j in range(len(car\_in\_circle[0])):

if(j < N-1):

select = car\_in\_circle[i][j][0]

next = car\_in\_circle[i][j+1][0]

if(next - select <= 5 and next - select >= 0):

c\_Max[i] += 1

# Menghitung nilai Rata-Rata mobil kembali ke posisi awal

sum = 0

for i in range(len(car)):

sum += car[i][2]

avg0 = sum/N/tmax\*100

return car\_in\_circle, c\_dens, avg0, c\_Max

#Fungsi render

def animate(i):

cars\_p = car\_Move[i]

car\_marker.set\_offsets(cars\_p)

return car\_marker

#main program

if \_\_name\_\_ == "\_\_main\_\_":

#inisialisasi variabel pemodelan

M = 100

p = float(0.3)

v = 0

N = 20

tmax = 1000

vmax = 5

d = 2

#Set Visualisasi Jalan

road\_fig = np.array( [ [[0,M+0.5], [3,3]], [[0,M+0.5], [7,7]] ] )

#Random Posisi Mobil

car = np.array([[random.randint(1,M), 5, 0] for i in range(1,N+1)])

#Sorting Posisi mobil

car = np.array(sorted(car, key=op.itemgetter(0)))

car\_Move, c\_dens, avg, c\_Max = car\_model(car)

# Menampilkan kepadatan di posisi 80-90

print(' Kepadatan Kendaraan di posisi X80 ~ X90:')

for x in c\_dens:

print(f'Kepadatan Kendaraan di waktu ke-{x+1} adalah sebesar {c\_dens[x]}')

# Menampilkan Figure kepadatan di posisi 80-90

plt.figure(1, figsize= (14,6))

plt.bar(range(len(c\_dens)), list(c\_dens.values()), width=0.6)

plt.title("Kepadatan Kedaraan di posisi X80 ~ X90")

plt.xlabel("Satuan Waktu")

plt.ylabel("Banyak Kendaraan")

# Menampilkan kepadatan Maksimum tiap internal dengan ketentuan 5 unit posisi

print("kepadatan Maksimum ketentuan 5 Unit Posisi")

for x in c\_Max:

print(f'Kepadatan Kendaraan Maks di waktu ke-{x+1} adalah sebesar {c\_Max[x]}')

# Menampilkan Figure kepadatan di posisi 80-90

plt.figure(2, figsize= (14,6))

plt.bar(range(len(c\_Max)), c\_Max, width=0.6)

plt.title("kepadatan Maksimum ketentuan 5 Unit Posisi")

plt.xlabel("Satuan Waktu")

plt.ylabel("Banyak Kendaraan")

# Menampilkan Nilai Rata rata Mobil kembali ke posisi awal

print('Waktu Rata Rata Kendaraan:')

print('Waktu rata-rata mobil kembali ke posisi awal adalah', format(avg, '.3f'))

# Figure Titik Semua Mobil

a = np.zeros(shape=(tmax,M))

for i in range(tmax):

index = 0

for j in range(M):

temp = np.array(sorted(car\_Move[i], key=op.itemgetter(0)))

if(j == temp[index][0] and index < N-1):

index +=1

a[i,j] = 0

else :

a[i,j] = -1

plt.figure(3, figsize= (15,25))

plt.xlabel("Satuan Poisis")

plt.ylabel("Satuan Waktu")

plt.imshow(a, cmap="Greys", interpolation="nearest")

# Menampilkan Simulasi

fig = plt.figure(4)

plot\_axes = plt.axes(ylim=(0,10), xlim=(0,M+0.5))

plt.plot([90,90],[3,7],color='blue')

plt.plot([80,80],[3,7],color='blue')

plt.title("Simulasi Traffic Flow")

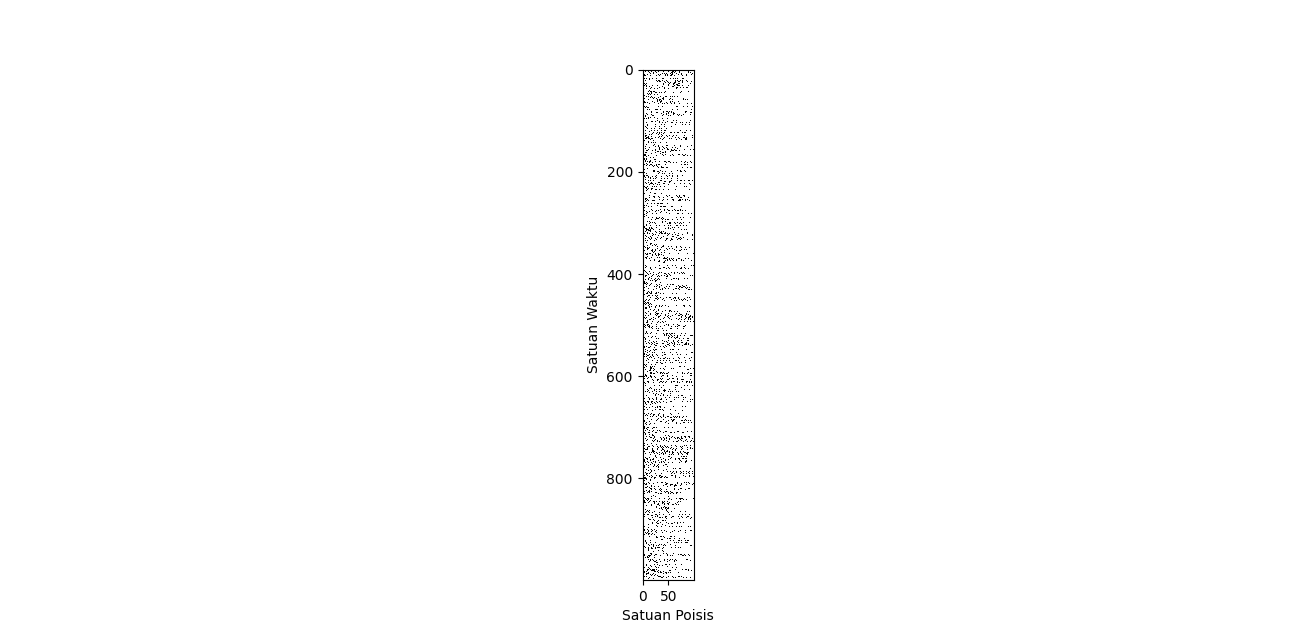
for road in road\_fig:

plt.plot(road[0], road[1], color="black")

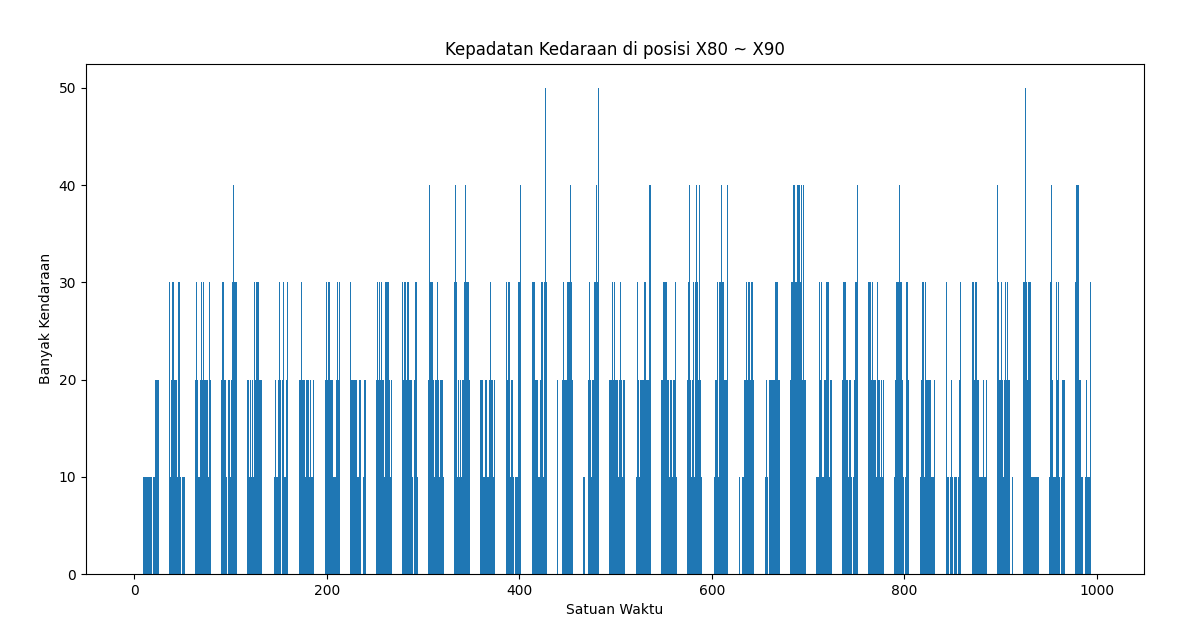
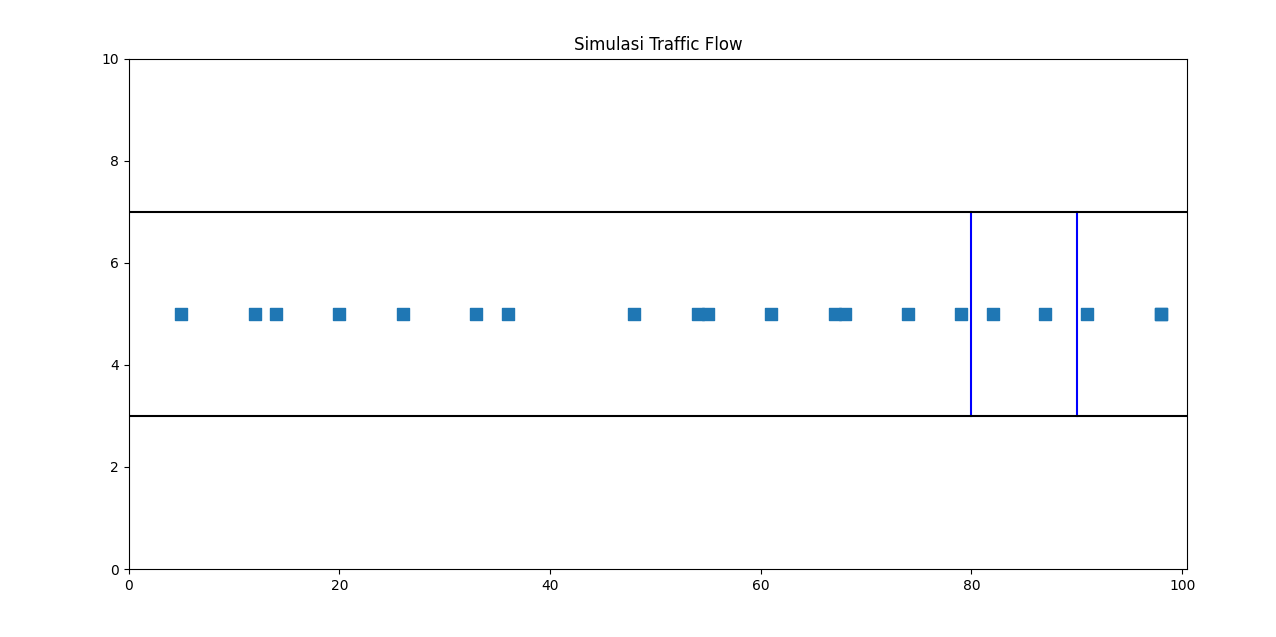
car\_marker = plot\_axes.scatter([], [], s=75, marker="s")

anim = animation.FuncAnimation(fig, animate, frames=len(car\_Move), interval=300, repeat=False)

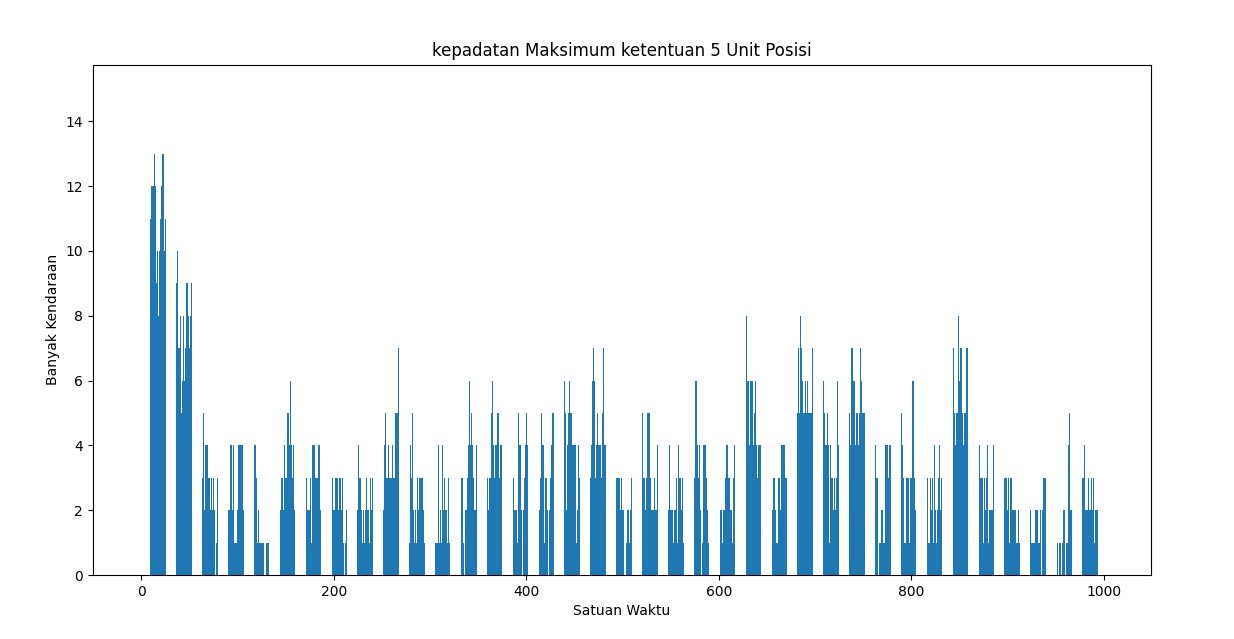
plt.show()

* 1. Hasil Simulasi

Bercak hitam dalam grafik merupakan simulasi suatu mobil pada interval suatu waktu dan suatu posis, kelompok kami menggunakan animation yang menampilkan gambar yang berganti sesuai dengan jumlah satuan waktu, pada gambar berikut adalah contoh tampilannya dimana kotak berwarna biru melambangkan mobil dan garis biru menggambarkan satuan posisi rentang 80-90.



hasil untuk nilai kepadatan di internal x80-x90, dimana pada label y menampilkan banyak kendaraan pada kepadatan x80-x90, dan pada label x menampilkan satuan waktu, bisa dilihat ada rentang satuan waktu 400 - 1000 mencapai 50 kendaraan.



## Berikut adalah tampilan kepadatan maksimum di setiap interval waktu, seperti ketentuan yang dibangun sesuai yang ada pada code, dimana program akan mengecek apakah kendaraan saat ini dan kendaraan di depannya memiliki jarak kurang dari sama dengan 5, untuk hasil yang didapatkan ini pada satuan waktu awal memiliki kepadatan yang lumayan cukup banyak.

## Berikut adalah hasil dari waktu rata-rata mobil kembali ke posisi awal, dimana rata-rata mobil kembali ke posisi awal adalah setelah 9 satuan waktu, dan mobil akan kembali ke posisi awal.

link

<https://drive.google.com/file/d/1mlfIEDOYtG1_AennaNK7ZmlKboHHQRRT/view?usp=sharing> << Codingan

Di windy << video

ini << laporan