**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# Отчёт по лабораторной работе №2 по дисциплине "Основы искусственного интеллекта"

**Тема** Нечёткая лабораторная работа

**Студент** Варламова Е. А.

**Группа** ИУ7-13М

**Оценка (баллы)** _____

**Преподаватели** Строганов Ю.В.

Москва — 2023 г.

# СОДЕРЖАНИЕ

**ВВЕДЕНИЕ** **4**

**1 Аналитическая часть** **5**
1.1 Содержательная постановка задачи . . . . . 5
1.2 Формализация задачи . . . . . 5
1.3 Общие этапы нечёткого логического вывода . . . . . 6
1.4 Алгоритм нечёткого вывода Цукомото . . . . . 7
1.5 Алгоритмы дефаззификации . . . . . 7
1.5.1 Алгоритм центра тяжести . . . . . 7
1.5.2 Алгоритм среднего . . . . . 7
Вывод . . . . . 8

**2 Конструкторская часть** **9**
2.1 Формирование базы правил . . . . . 9
2.1.1 Описание переменных . . . . . 9
2.1.2 Функции принадлежности . . . . . 9
2.1.3 Правила для нечёткого логического вывода . . . . . 12
2.2 Алгоритм автопилота . . . . . 13
Вывод . . . . . 13

**3 Технологическая часть** **14**
3.1 Выбор средств разработки . . . . . 14
3.2 Листинг ПО . . . . . 14
3.3 Пример работы с постоянной скоростью лидера . . . . . 18
3.4 Пример работы с изменяющейся по sin скоростью лидера . . . . . 19
Вывод . . . . . 20

**4 Исследовательская часть** **21**
4.1 Исследование зависимости результатов работы ПО от моделируемого времени работы системы . . . . . 21

Вывод . . . . . 22

**ЗАКЛЮЧЕНИЕ** **23**

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** **24**

# Введение

Нечеткая логика – это раздел математики, который был разработан для описания нечетких и неопределенных понятий в реальном мире. Она была создана в 1965 году Лотфи Заде, который предложил новый подход к моделированию нечетких систем. Нечеткая логика нашла широкое применение в различных областях, таких как управление, искусственный интеллект, экономика, медицина и другие. Ее основное преимущество заключается в возможности работать с нечеткими и неопределенными данными, которые не могут быть точно определены с помощью классической логики.

Целью данной лабораторной работы является создание системы для следования при отсутствии данных о скорости «лидера», но с известным расстоянием до него с помощью аппарата нечёткой логики. Для этого необходимо решить следующие задачи:

— привести содержательную постановку задачи;

— описать общие этапы нечёткого логического вывода;

— описать алгоритм вывода Цукомото;

— описать алгоритмы дефаззификации;

— предложить функции принадлежности и правила для нечёткого логического вывода;

— разработать алгоритм работы автопилота;

— привести особенности реализации ПО, решающего поставленную задачу;

— провести исследование зависимости результатов работы ПО (среднеквадратичная ошибка, время вычислений) от моделируемого времени работы системы.

# 1 Аналитическая часть

## 1.1 Содержательная постановка задачи

Есть два автомобиля в двумерном пространстве: один под управлением пользователя («лидер»), другой – автопилота. Пользователь может изменять параметр скорости своего автомобиля, автопилот должен следовать за "лидером"строго на заданной дистанции.

Необходимо создать систему для следования, при отсутствии данных о скорости "лидера но с известным расстоянием до него.

## 1.2 Формализация задачи

Формализация задачи в виде IDEF0-диаграммы изображена на рисунке 1.1.

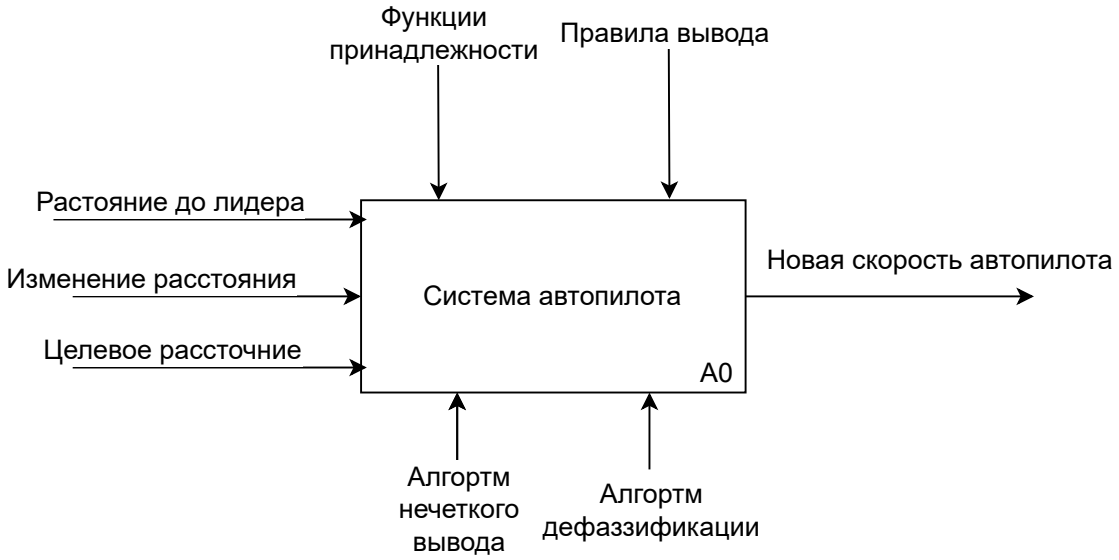


Рис. 1.1: IDEF0-диаграмма задачи

## 1.3 Общие этапы нечёткого логического вывода

1. сформировать на естественном языке импликативные правила, отражающие зависимости, которые действуют в предметной области;
2. выделить из правил лингвистические переменные;
3. определить их значения;
4. сопоставить каждой переменной нечёткое множество значений;
5. формализовать правила из п. 1 с помощью переменных из п. 2;
6. проверить базу правил на полноту, то есть проверить, что:
   - для каждого значения выходных переменных должно быть хотя бы одно правило;
   - для каждой входной лингвистической переменной и каждого ее значения должно быть хотя бы одно правило, в котором это значение стоит в основании импликации.
7. оценить степень принадлежности каждого входного значения к каждому нечеткому множеству (фаззификация);
8. применить нечеткие правила для определения степени принадлежности каждого выходного значения к каждому нечеткому множеству (фаззификация);
9. агрегировать степени принадлежности выходных значений для получения одного значения для каждой выходной переменной (агрегация);
10. применить операции дефаззификации для получения конкретных значений выходных переменных (дефаззификация).

Таким образом, общая последовательность логического вывода на основе сформированной базы правил состоит из 3 этапов:

- фаззификация;
- агрегация;
- дефаззификация.

## 1.4 Алгоритм нечёткого вывода Цукомото

Алгоритм состоит из следующих этапов [1]:

1. введение нечёткости;
2. агрегирование степеней истинности предпосылок по каждому из правил;
3. активизация заключений по каждому из правил – в результате находятся четкие значения выходных переменных в каждом из заключений правил;
4. этап аккумулирования активизированных заключений правил в данном алгоритме отсутствует вследствие четких значений выходных переменных;
5. в качестве метода дефаззификации в алгоритме Цукамото используется разновидность метода центра тяжести для одноточечных множеств, позволяющий осуществить приведение к четкости выходной переменной без предварительного аккумулирования активизированных заключений отдельных правил.

## 1.5 Алгоритмы дефаззификации

### 1.5.1 Алгоритм центра тяжести

Дефаззификация алгоритмом центра тяжести осуществляется по формуле 1.1.

$$y = \frac{\int_{min}^{max} x \cdot \mu(x) dx}{\int_{min}^{max} \mu(x) dx}, \tag{1.1}$$

где $y$ – результат дефазификации, $x$ – переменная, соответствующая выходной лингвистической переменной, $\mu(x)$ – функция принадлежности нечеткого множества, соответствующего выходной переменной после этапа аккумуляции, $min$ и $max$ – левая и правая точки интервала носителя нечеткого множества рассматриваемой выходной переменной.

### 1.5.2 Алгоритм среднего

Дефазификация алгоритмом среднего осуществляется по формуле 1.2.

$$y = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{1.2}$$

где $n$ – количество точек выходной переменной, $\mu(x)$ – функция принадлежности нечеткого множества, соответствующего выходной переменной после этапа аккумуляции.

## Вывод

В данном разделе была приведена содержательная постановка задачи и ее формализация, были описаны общие этапы нечёткого логического вывода, а также приведены 2 алгоритма дефаззификации и алгоритм нечёткого вывода Цукомото.

# 2 | Конструкторская часть

## 2.1 Формирование базы правил

### 2.1.1 Описание переменных

Для решения поставленной задачи предлагается ввести следующие лингвистические переменные:

— входная переменная – расстояние между лидером и пилотируемым транспортным средством;

— входная переменная – изменение расстояния между лидером и пилотируемым траспортным средством;

— выходная переменная – значение изменения скорости пиотируемого траспортного средства.

### 2.1.2 Функции принадлежности

Опишем функции принадлежности каждой из переменных.

#### Переменная расстояния между лидером и пилотируемым транспортным средством

Требуемым расстоянием называется расстояние, которое необходимо соблюдать между пилотируемым транспортным средством и лидером. В данной работе оно принято за 100.

Под функцией Gaussian(a, b) понимается функция Гаусса, где a – математическое ожидание, b – среднеквадратичное отклонение.

Переменной дано имя distance. Функция принадлежности для переменной distance представлена в таблице 2.1. Визуализация функции принадлежности для переменной distance представлена на рисунке 2.1.

Таблица 2.1: Переменная distance

| название | описание | функция |
|---|---|---|
| less | расстояние меньше требуемого | Gaussian(25, 30) |
| normal | расстояние в пределах требуемого | Gaussian(100, 15) |
| more | расстояние больше требуемого | Gaussian(175, 30) |

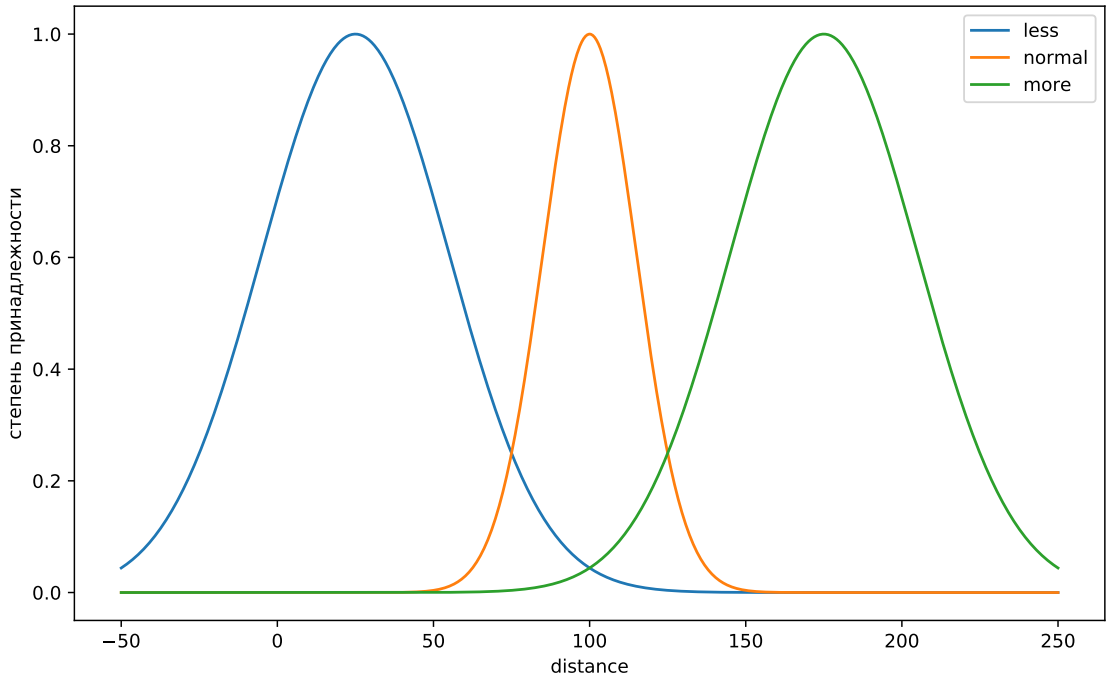


Рис. 2.1: Функция принадлежности для переменной distance

**Переменная изменения скорости пиотируемого траспортного средства**

Под функцией Gaussian(a, b) понимается функция Гаусса, где a – математическое ожидание, b – среднеквадратичное отклонение.

Переменной дано имя distance_change. Функция принадлежности для переменной distance_change представлена в таблице 2.2.

Визуализация функции принадлежности для переменной distance_change представлена на рисунке 2.2.

Таблица 2.2: Переменная distance_change

| название | описание | функция |
|---|---|---|
| to_leader | направление изменения расстояния – к лидеру | Gaussian(-50, 20) |
| same | расстояние не меняется | Gaussian(0, 10) |
| from_leader | направление изменения расстояния – от лидера | Gaussian(50,20) |

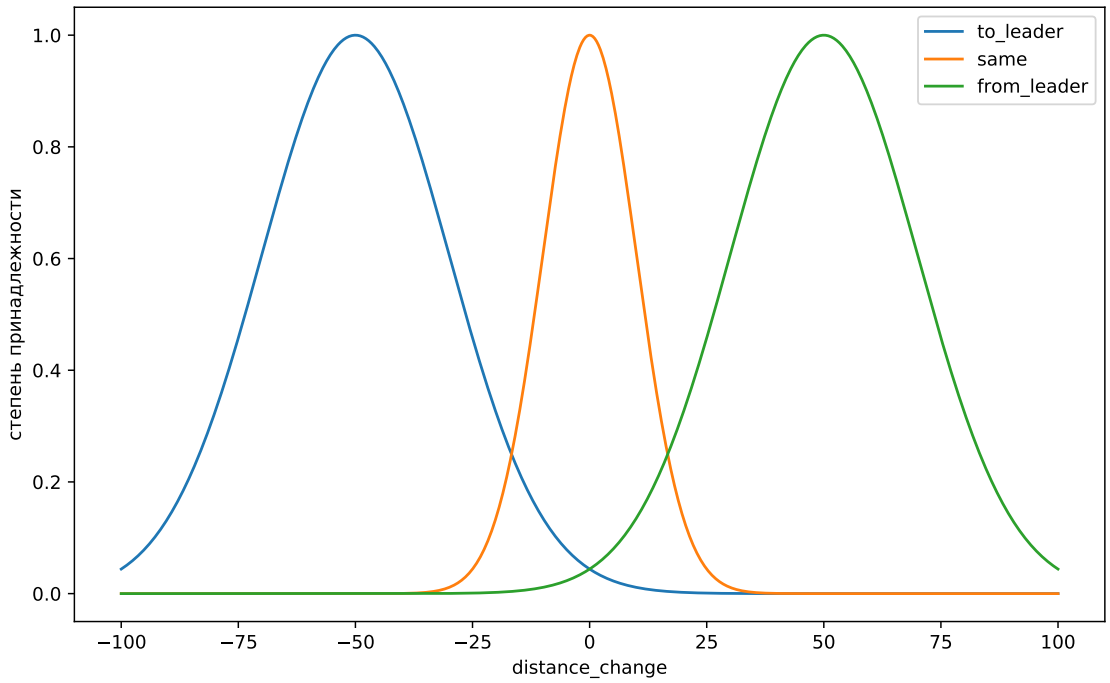


Рис. 2.2: Функция принадлежности для переменной distance_change

**Переменная изменения расстояния между лидером и пилотируемым траспортным средством**

Под функцией Polygon(a, b, c) понимается функция треугольного вида (состоящая из двух прямых), где a, b, c – точки, определяющие эти прямые.

Под функцией Trapezoid(a, b, c, d) понимается трапецевидная функция (состоящая из трех прямых), где a, b, c, d – точки, определяющие эти прямые.

Переменной дано имя autopilot_control. Функция принадлежности для переменной autopilot_control представлена в таблице 2.4.

Визуализация функции принадлежности для переменной autopilot_control представлена на рисунке 2.3.

Таблица 2.3: Переменная autopilot_control

| название | описание | функция |
|---|---|---|
| slowdown | уменьшить скорость | Polygon((-50,1), (-25,1), (-5,0)) |
| maintain_speed | оставить скорость | Trapezoid((-10,0), (-5,1), (5,1), (10,0)) |
| speed_up | увеличить скорость | Polygon((5,0), (25,1), (50,1)) |

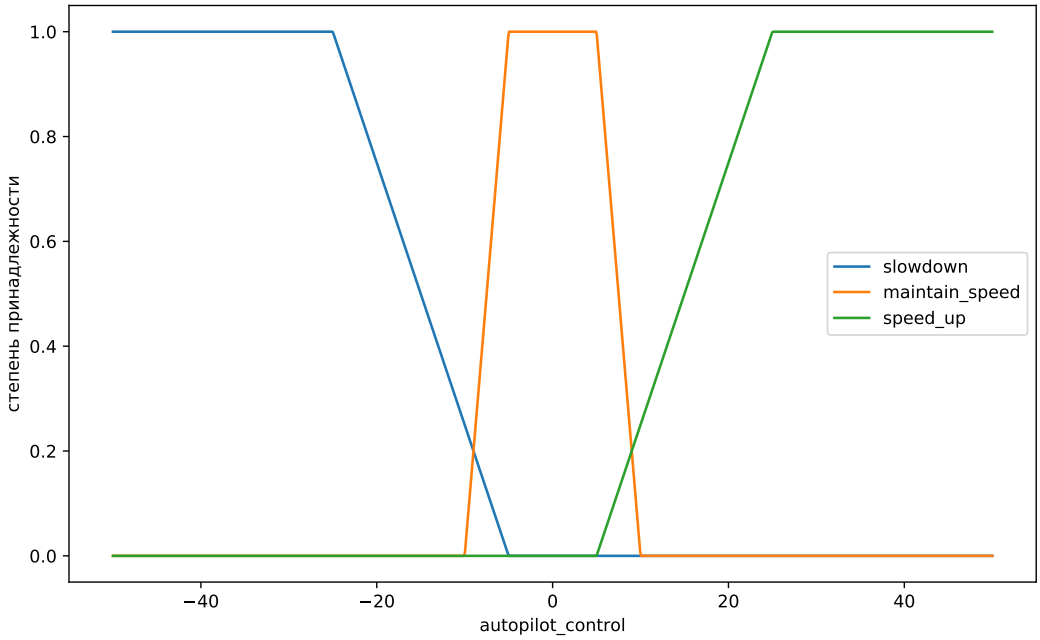


Рис. 2.3: Функция принадлежности для переменной autopilot_control

### 2.1.3 Правила для нечёткого логического вывода

Для нечёткого логического вывода предлагаются следующие правила, отвечающие специфике предметной области:

Таблица 2.4: Правила вывода

| если | то |
|---|---|
| distance = less | autopilot_control = slowdown |
| distance = normal and distance_change = to_leader | autopilot_control = slowdown |
| distance = normal and distance_change = same | autopilot_control = maintain_speed |
| distance = normal and distance_change = from_leader | autopilot_control = speed_up |
| distance = more | autopilot_control = speed_up |

## 2.2 Алгоритм автопилота

Разработанный алгоритм автопилота, использующий нечеткую систему для определения изменения скорости ТС, представлен на рисунке 2.4.

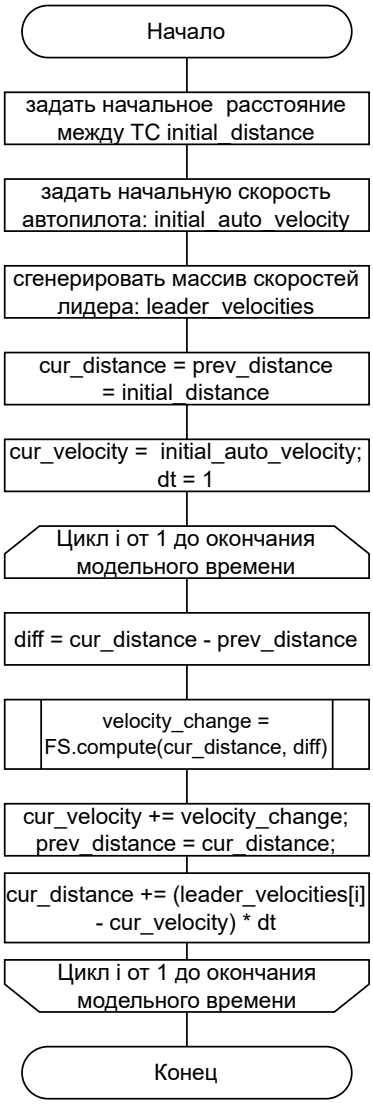


Рис. 2.4: Алгоритм автопилота

## Вывод

В данном разделе были описаны переменные нечёткой системы, используемой для решения задачи, были предложены их функции принадлежности, а также разработаны на основе знаний о предметной области нечёткие правила вывода. Кроме того, был разработан алгоритм автопилота, использующий нечёткую систему для определения изменения скорости ТС.

# 3 | Технологическая часть

## 3.1 Выбор средств разработки

В качестве языка программирования был использован язык Python, поскольку этот язык кроссплатформенный и для него разработано огромное количество библиотек и модулей, решающих разнообразные задачи.

В частности, имеются библиотеки, включающие в себя алгоритмы для нечётких систем. В данной работе была использована библиотека «fuzzypython» [2], поскольку в отличие от многих других библиотек, имеющих схожую функциональность, в «fuzzypython» реализован алгоритм нечёткого вывода Цукомото.

Для создания графиков была выбрана библиотека matplotlib [3], доступная на языке Python, так как она предоставляет удобный интерфейс для работы с данными и их визуализации.

## 3.2 Листинг ПО

На основе алгоритма автопилота, разработанного в конструкторской части, а также нечёткой системы из «fuzzypython», детали формирования базы правил для которой были описаны в конструкторской части, было разработано ПО, решающее поставленную задачу. В листинге 3.1 представлена реализация решения поставленной задачи.

Листинг 3.1: Реализация решения поставленной задачи

```
from adjective import Adjective
from fsets.polygon import Polygon
from fsets.gaussian import Gaussian
from fsets.trapezoid import Trapezoid
from ruleblock import RuleBlock
from systems.tsukamoto import TsukamotoSystem
from variable import Variable
import time
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
import random
```

```
13 def DrawVarFuncs(fs, interval, labs, title):
14     fig = plt.figure(figsize=(10, 6))
15     x = np.linspace(interval[0], interval[1], 1000)
16     for i in range(len(labs)):
17 
18         y = [fs[i](j) for j in x]
19         plt.plot(x, y, label=labs[i])
20     plt.legend()
21     plt.xlabel(title)
22     plt.ylabel("                                                ")
23     pdf = PdfPages('img/variable_' + title + ".pdf")
24     pdf.savefig(fig)
25     pdf.close()
26     plt.clf()
27 def DrawResult(n_it, auto_velocities, leader_velocities):
28     fig = plt.figure(figsize=(10, 6))
29     x = [i for i in range(n_it)]
30 
31     plt.plot(x, auto_velocities, label='                    ')
32     plt.plot(x, leader_velocities, label='          ')
33 
34     plt.grid()
35     plt.xlabel("                              ")
36     plt.ylabel("                ,   /  ")
37 
38     plt.legend()
39     pdf = PdfPages('img/' + "result_" + str(n_it) + "_iters.pdf")
40     pdf.savefig(fig)
41     pdf.close()
42     plt.clf()
43 def PrintResultTable(iters, times, stds):
44     for i in range(len(iters)):
45         print("\hline {} & {:.2f} & {:.2f} \\\\".format(iters[i], times[i],
              stds[i]))
46 def CreateSystem():
47     #
48     ## distance
49     less_func = Gaussian(25, 30)
50     less = Adjective('less', less_func )
51     normal_func = Gaussian(100, 15)
52     normal = Adjective('normal', normal_func )
53     more_func = Gaussian(175, 30)
54     more = Adjective('more', more_func )
55 
56     distance = Variable('distance', 'km', less, normal, more)
57     DrawVarFuncs([less_func, normal_func, more_func],
58                  [-50, 250],
59                  ['less', 'normal', 'more'],
60                  'distance')
61 
```

```
62      ## distance_change
63      to_leader_func = Gaussian(-50, 20)
64      to_leader = Adjective('to_leader', to_leader_func )
65      same_func = Gaussian(0, 10)
66      same = Adjective('same', same_func )
67      from_leader_func = Gaussian(50,20)
68      from_leader = Adjective('from_leader', from_leader_func )
69
70      distance_change = Variable('distance_change', 'km', to_leader, same,
            from_leader )
71
72      DrawVarFuncs([to_leader_func, same_func, from_leader_func],
73                   [-100, 100],
74                   ['to_leader', 'same', 'from_leader'],
75                   'distance_change')
76
77
78      ## autopilot_control
79      slowdown_func = Polygon((-50,1), (-25,1), (-5,0))
80      slowdown = Adjective('slowdown', slowdown_func )
81      maintain_speed_func = Trapezoid((-10,0), (-5,1), (5,1), (10,0))
82      maintain_speed = Adjective('maintain_speed',  maintain_speed_func)
83      speed_up_func = Polygon((5,0), (25,1), (50,1))
84      speed_up = Adjective('speed_up',speed_up_func)
85
86      autopilot_control = Variable('autopilot_control', 'km', slowdown,
            maintain_speed, speed_up)
87      DrawVarFuncs([slowdown_func, maintain_speed_func, speed_up_func],
88                   [-50, 50],
89                   ['slowdown', 'maintain_speed', 'speed_up'],
90                   'autopilot_control')
91      scope = locals()
92
93      #
94      block = RuleBlock('autopilot_control', operators=('MIN', 'MAX', 'ZADEH')
            , activation='MAX')
95
96      r1 = 't:if distance is less then autopilot_control is slowdown'
97      r2 = 't:if distance_change is to_leader and distance is normal then
            autopilot_control is slowdown'
98      r3 = 't:if distance_change is same and distance is normal then
            autopilot_control is maintain_speed'
99      r4 = 't:if distance_change is from_leader and distance is normal then
            autopilot_control is speed_up'
100     r5 = 't:if distance is more then autopilot_control is speed_up'
101
102     block.add_rules(r1, r2, r3, r4, r5, scope = scope)
103     #
104     tsukamoto = TsukamotoSystem('auto_pilot_system', block)
105     return tsukamoto
```

```
106
107
108 #n_it_arr = list(map(int, np.linspace(20, 500, 50)))
109
110 n_it_arr = [10, 100,200, 500, 1000]
111 time_work = []
112 std_arr = []
113
114 dt = 1
115 initial_distance = 50
116 initial_auto_velocity = 20
117 initial_leader_velocity = 50
118
119 leader_velocities = []
120 auto_velocities = []
121 system = CreateSystem()
122 for n_it in n_it_arr:
123     time_start = time.time()
124     leader_velocities = initial_leader_velocity + 20 * np.sin(np.linspace(0,
          10, n_it))
125     #leader_velocities = [initial_leader_velocity] * n_it
126
127     auto_velocities = []
128
129     prev_distance = cur_distance = initial_distance
130     cur_velocity = initial_auto_velocity
131
132     for i in range(n_it):
133         auto_velocities.append(cur_velocity)
134         velocity_change = system.compute({'distance': cur_distance, '
              distance_change': cur_distance - prev_distance})['
              autopilot_control']
135
136         cur_velocity += velocity_change
137
138         prev_distance = cur_distance
139         cur_distance += (leader_velocities[i] - cur_velocity) * dt
140
141     time_work.append(time.time() - time_start)
142     std_arr.append(np.sqrt(np.std(np.array(auto_velocities) - np.array(
          leader_velocities))))
143
144     DrawResult(n_it, auto_velocities, leader_velocities)
145 ##fig = plt.figure(figsize=(10, 6))
146 ##plt.plot(n_it_arr, time_work)
147 ##plt.xlabel("                                ")
148 ##plt.ylabel("                               (   )")
149 ##pdf = PdfPages('img/' + "result_time.pdf")
150 ##pdf.savefig(fig)
151 ##pdf.close()
```

```
##plt.clf()
##
##fig = plt.figure(figsize=(10, 6))
##plt.plot(n_it_arr, std_arr)
##plt.xlabel("                                    ")
##plt.ylabel("       ")
##pdf = PdfPages('img/' + "result_std.pdf")
##pdf.savefig(fig)
##pdf.close()
##plt.clf()
PrintResultTable(n_it_arr, time_work, std_arr)
```

## 3.3 Пример работы с постоянной скоростью лидера

Зададим:

— начальную скорость автопилота, равную 20 м/с;

— начальную скорость лидера, равную 50 м/с;

— начальную дистанцию, равную 50м.

Скорость лидера положим постоянной.

Результат работы за 100 единиц модельного времени представлен на рисунке 3.1.

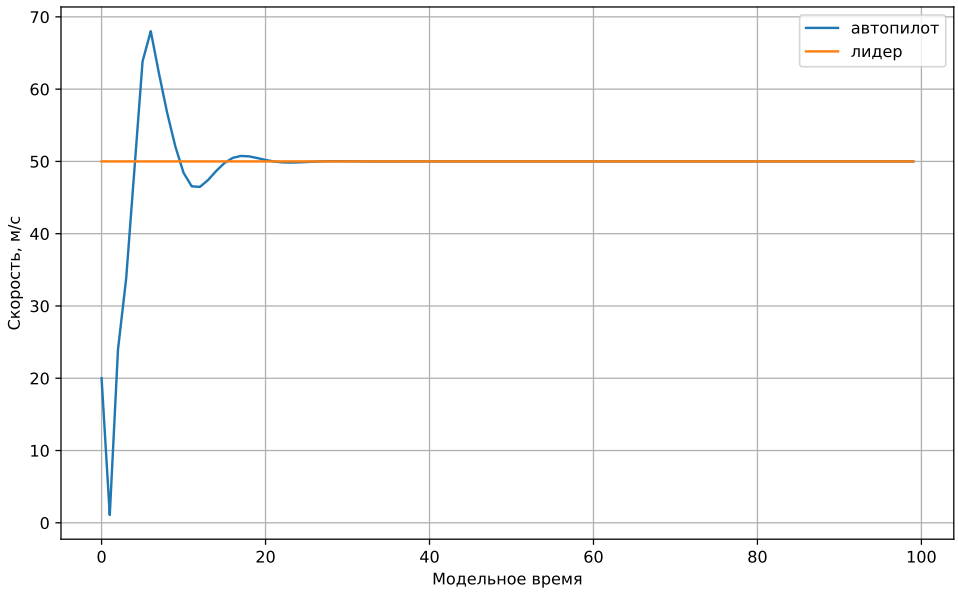


Рис. 3.1: Результат для постоянной скорости лидера

## 3.4 Пример работы с изменяющейся по sin скоростью лидера

Зададим:

— начальную скорость автопилота, равную 20 м/с;

— начальную скорость лидера, равную 50 м/с;

— начальную дистанцию, равную 50м.

Скорость лидера будем изменять по закону синуса.

Результат работы за 100 единиц модельного времени представлен на рисунке 3.2.

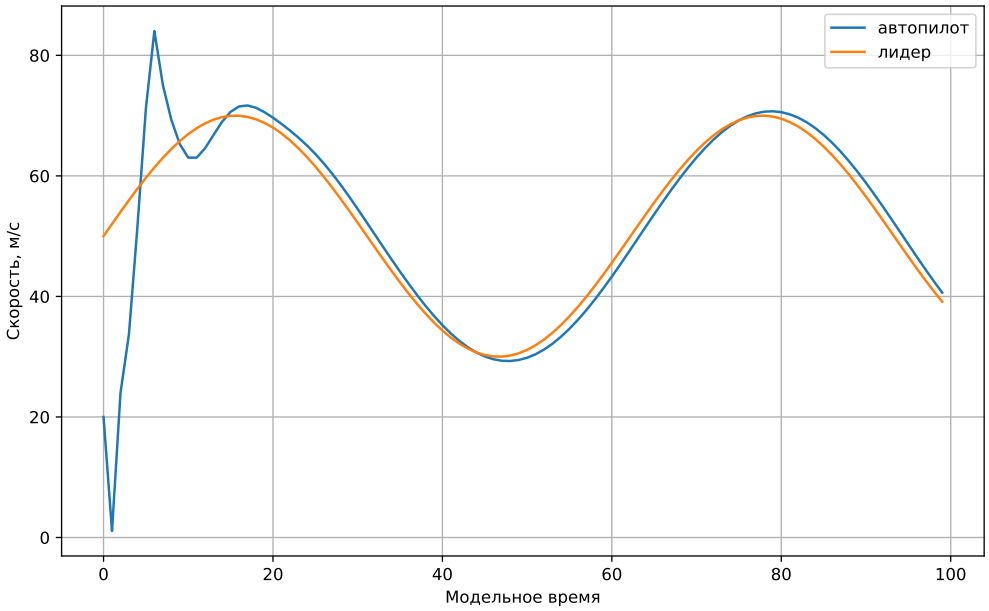


Рис. 3.2: Результат с изменяющейся по sin скоростью лидера (100с)

Результат работы за 1000 единиц модельного времени представлен на рисунке 3.3.

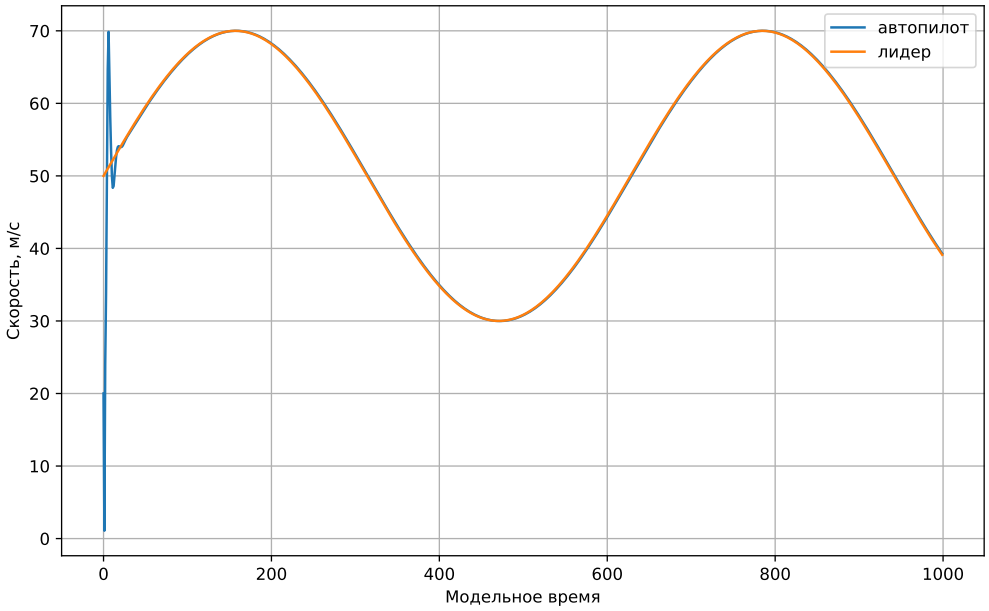


Рис. 3.3: Результат с изменяющейся по sin скоростью лидера (1000с)

## Вывод

В данном разделе были обоснованы средства реализации ПО, реализовано ПО, решающее поставленную задачу, и приведены примеры его работы.

# 4 | Исследовательская часть

## 4.1 Исследование зависимости результатов работы ПО от моделируемого времени работы системы

Под моделируемым временем работы системы имеется в виду количество модельных секунд, в течение которых моделируемая автопилотируемая система принимала решения об изменении скорости транспортного средства. Предполагается, что система принимает решения каждую секунду.

Под результатами работы системы имеются в виду следующие характеристики:

— время работы – реальное время работы ПО;

— среднеквадратичное отклонение скорости автопилота от скорости лидера за всё время работы системы.

Для исследования были использованы те же начальные условия, что в примере работы с изменяющейся по sin скоростью лидера в технологической части:

— начальную скорость автопилота, равная 20 м/с;

— начальную скорость лидера, равная 50 м/с;

— начальную дистанцию, равная 50м.

Результаты исследования приведены на рисунках 4.1 и 4.2 .

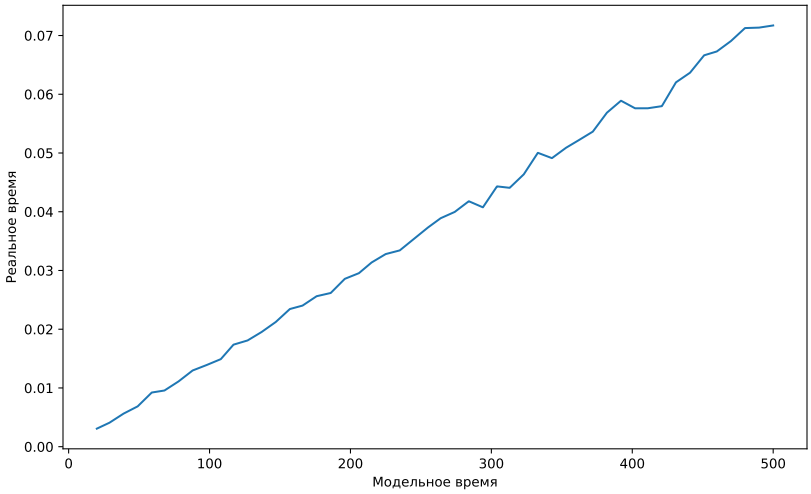


Рис. 4.1: Результат по времени работы

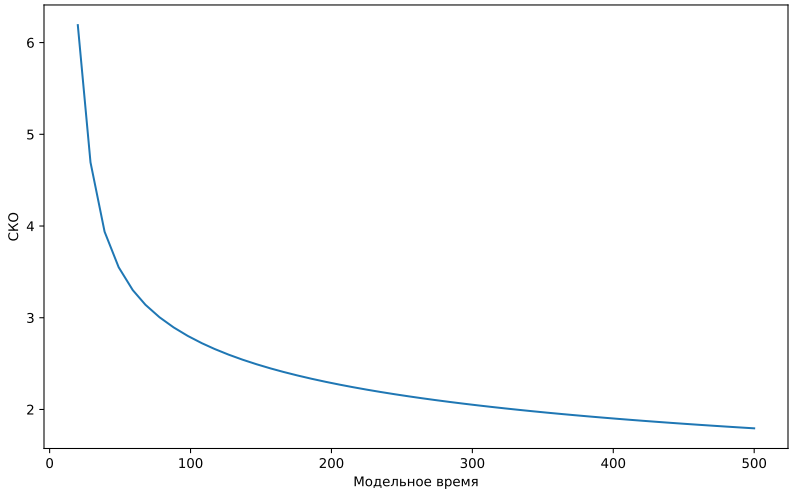


Рис. 4.2: Результат по СКО

# Вывод

В данном разделе было проведено исследование зависимости результатов работы ПО от моделируемого времени работы системы. Как и ожидалось, время работы растёт с увеличением моделируемого времени работы. При этом СКО с увеличением моделируемого времени уменьшается.

# ЗАКЛЮЧЕНИЕ

Целью данной лабораторной работы являлось создание системы для следования при отсутствии данных о скорости «лидера», но с известным расстоянием до него с помощью аппарата нечёткой логики. Цель была достигнута, при этом были решены следующие задачи:

— приведена содержательную постановку задачи;

— описаны общие этапы нечёткого логического вывода;

— описан алгоритм вывода Цукомото;

— описан алгоритмы дефаззификации;

— предложены функции принадлежности и правила для нечёткого логического вывода;

— разработан алгоритм работы автопилота;

— приведены особенности реализации ПО, решающего поставленную задачу;

— проведено исследование зависимости результатов работы ПО (среднеквадратичная ошибка, время вычислений) от моделируемого времени работы системы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Борисов, В. В. Нечеткие модели и сети : учебное пособие / В. В. Борисов, В. В. Круглов, А. С. Федулов. [и др.] // Москва : Горячая линия-Телеком. — 2018. — С. 284.
2. Библиотека с алгоритмами нечёткой логики fuzzython [Электронный ресурс]. — Режим доступа: URL: https://github.com/yudivian/fuzzython (дата обращения: 13.12.2023).
3. Библиотека визуализации данных matplotlib [Электронный ресурс]. — Режим доступа: URL: https://matplotlib.org (дата обращения: 13.12.2023).