|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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: «Специальное машиностроение»

Кафедра: «Робототехнические системы и мехатроника»

**Лабораторная работа № 7**

по курсу «Теория автоматического управления»

Вариант 8

Выполнил: Ионин Даниил

Группа: СМ11-61Б

Проверил(а):

Москва, 2024 г.

ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc167124447)

[АНАЛиз неизменяемой части 3](#_Toc167124448)

[Моделирование наблюдателя Люенбергера 4](#_Toc167124449)

[Моделирование линейно-квадратичного управления 7](#_Toc167124450)

[Моделирование системы с нечеткой логикой 9](#_Toc167124451)

АНАЛиз неизменяемой части

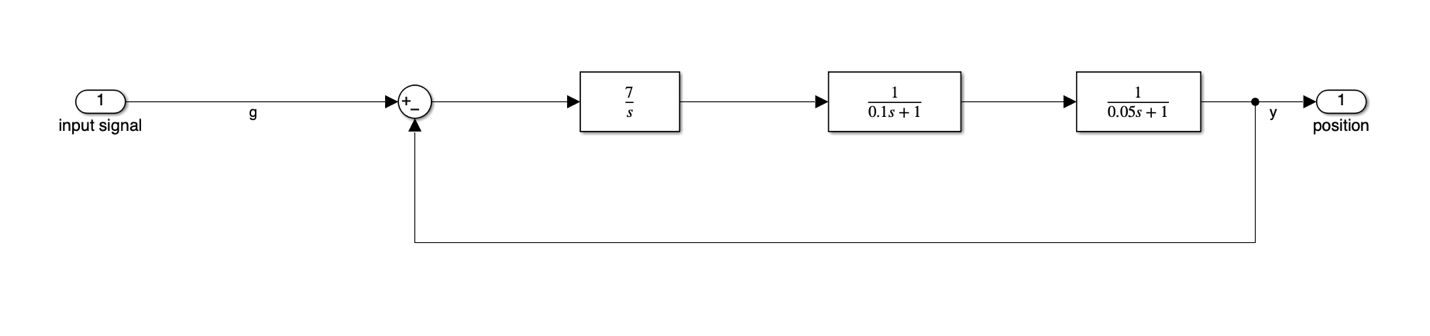


Рисунок 1 – линейная часть системы из первого д/з

Передаточная функция линейной части:

Передаточная функция замкнутой системы:

Моделирование наблюдателя Люенбергера

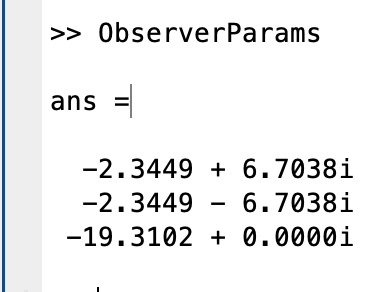
Для того чтобы реализовать наблюдатель Люенбергера, необходимо представить имеющуюся систему в виде:

где L - это некоторая матрица констант.

- состояние системы, которое мы предполагаем равным настоящему состоянию системы.

Скрипт ObserverParams, в котором записаны матрицы заданные выше:

|  |
| --- |
| %задаем матрицы наблюдатель Люенбергера  lab7.Observer.A = [0 1 0; 0 0 1; -1167 -167 -25];  lab7.Observer.B = [0; 0; 1167];  lab7.Observer.C = [0, 0, 7];  lab7.Observer.D = [0, 0, 0];  lab7.Observer.L = [-1;-1;-1];  % eig(lab7.Observer.A - [lab7. Observer.L, zeros(3, 2)]) |



Проверка найденной матрицы L

Вывод: действительно корни находятся в левой части комплексной плоскости

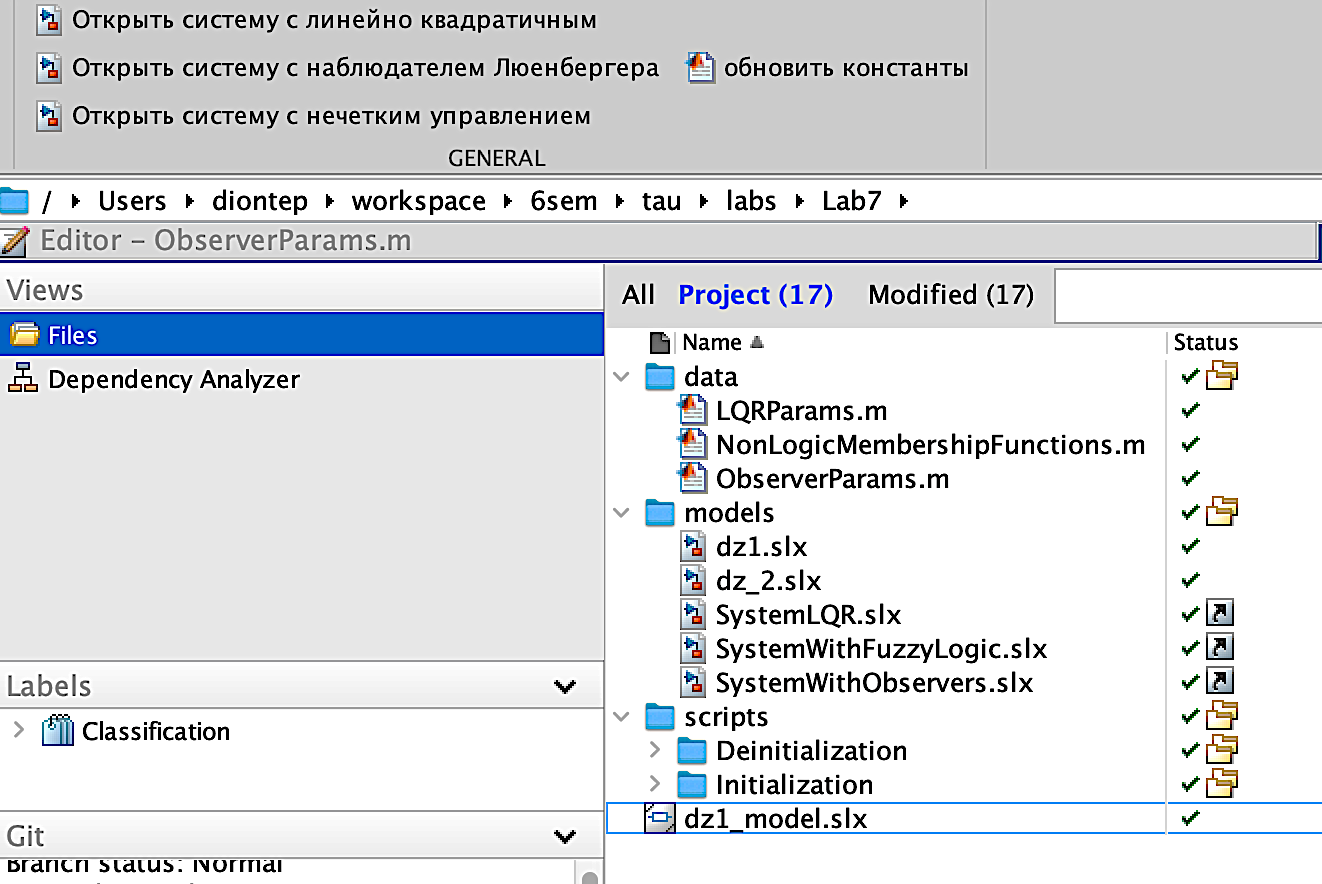


Рисунок 2 – структура проекта, решающего задачу.

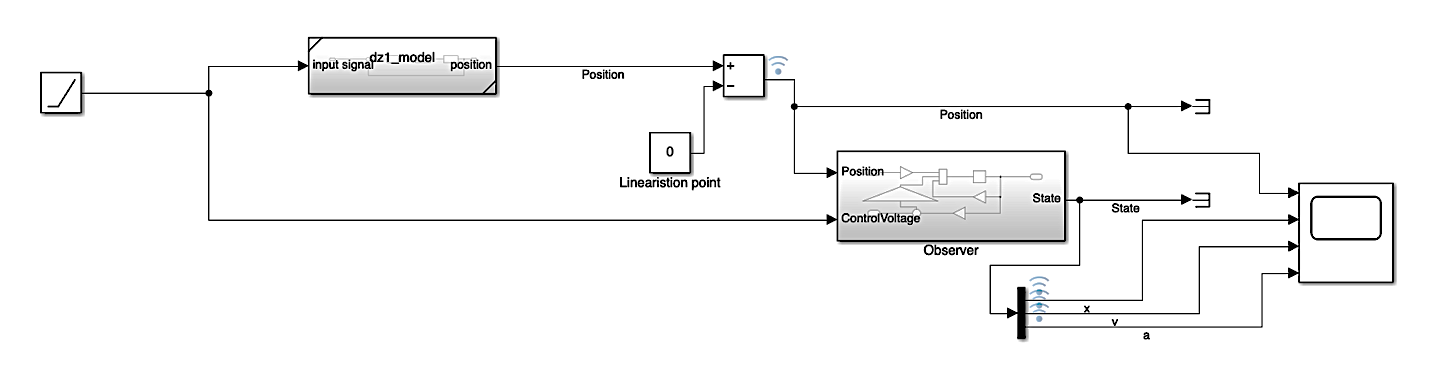


Рисунок 3 – модель с домашним заданием и наблюдателем

На вход подается сигнал g = 5t

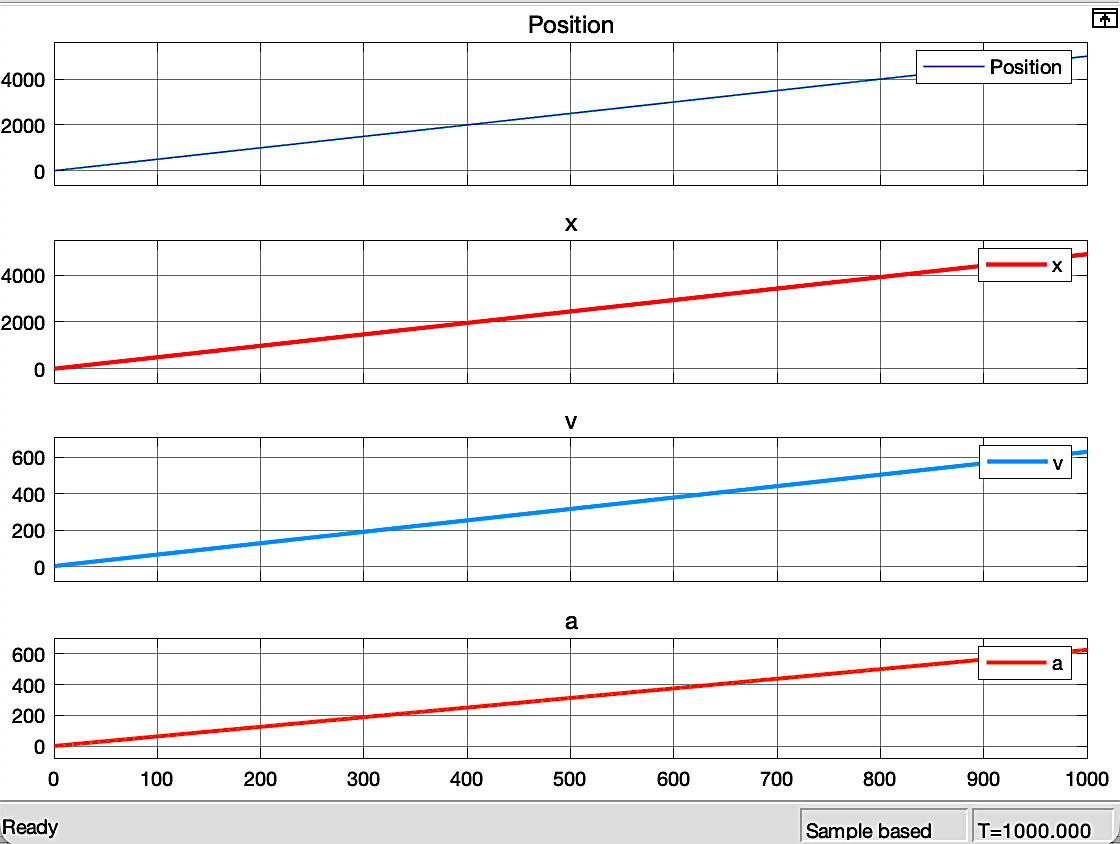


Рисунок 4 – результаты наблюдения

Вывод: система реализующая наблюдатель Люенбергера построена

Моделирование линейно-квадратичного управления

Для задания линейно-квадратичного управления необходимо получить

А также задать матрицы Q, R, P и K

Если первые матрица мы уже нашли, вторые найдем решив алгебраического уравнения Риккати:

Которое при *N*=0 (6) принимает вид

В процессе решения определяется матрица 𝑃 и оптимальный

регулятор задается выражением

или

при *N*=0

Данные матрицы задаются в системе при помощи скрипта LQRParams:

|  |
| --- |
| %задаем матрицы линейно-квадратичного управления  lab7.LQRParams.A = [0 1 0; 0 0 1; -1167 -167 -25];  lab7.LQRParams.B = [0; 0; 1167];  lab7.LQRParams.C = [0, 0, 7];  lab7.LQRParams.D = [0, 0, 0];  lab7.LQRParams.Q = 25\*lab7.LQRParams.C'\*lab7.LQRParams.C;  lab7.LQRParams.R = 1;  [lab7.LQRParams.K,lab7.LQRParams.S,lab7.LQRParams.P] = lqr(lab7.LQRParams.A,lab7.LQRParams.B,lab7.  ... LQRParams.Q,lab7.LQRParams.R); |

При запуске соответствующего шортката, (которые есть в предыдущей главе, открывается система, показанная на рис. 5:

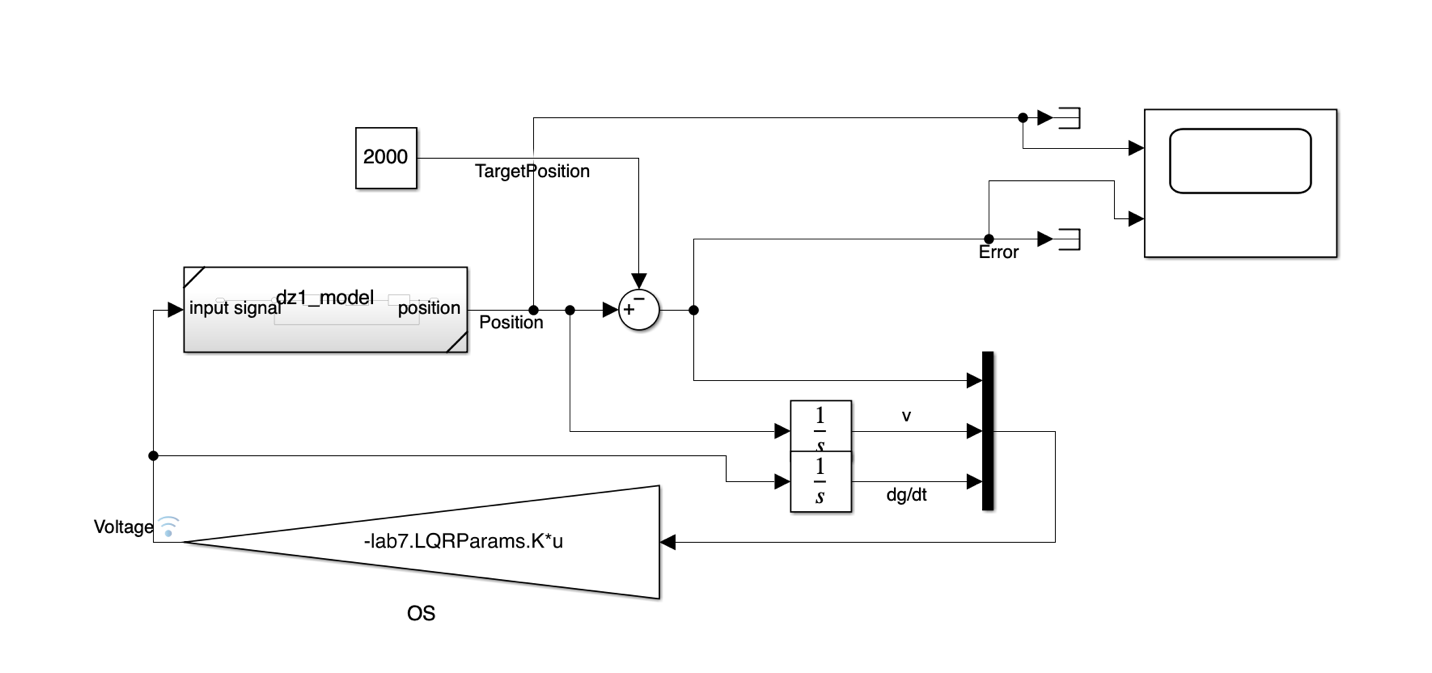


Рисунок 5 – система с линейно-квадратичным управлением

В качестве примера я задал слежение за позицией 2000.

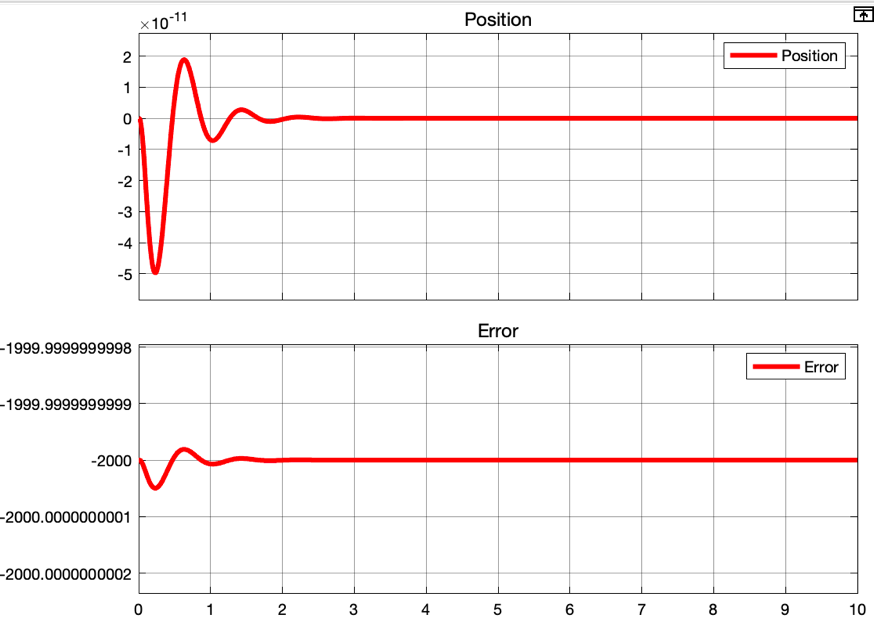


Рисунок 6 - Результат моделирования

Вывод: система с линейно-квадратичным управлением построена

Моделирование системы с нечеткой логикой

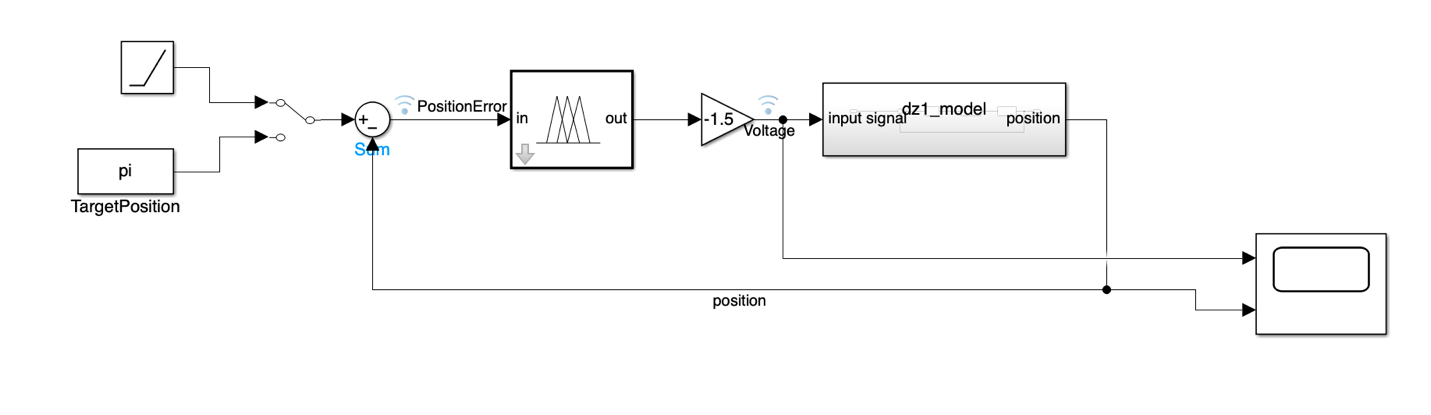


Рисунок 7 – стандартная схема с нечеткой логикой.

Из схемы понятно, что для построения системы с управлением с помощью нечеткой логики необходиио и достаточно построить нечеткую логику.(Как бы странно это не звучало).

Нечеткая логика – концепция, позволяющая работать с данными, которые не являются строго истинными или ложными, а представляют промежуточные состояния. Такой подход позволяет моделировать неопределённость и приближение, характерные для естественного языка и человеческого мышления

Полученная модель с нечеткой логикой, с помощью программного пакета FuzzyLogicToolbox:

|  |
| --- |
| % Создание нечёткой системы управления  fuzzy\_lab7 = mamfis('Name', 'fuzzy\_lab7');  % Добавление входных и выходных переменных к системе  fuzzy\_lab7 = addInput(fuzzy\_lab7, [-10 10], 'Name', 'PositionError');  fuzzy\_lab7 = addOutput(fuzzy\_lab7, [-12 12], 'Name', 'ControlVoltage');  % Функции принадлежности для ошибки по положению  fuzzy\_lab7 = addMF(fuzzy\_lab7, 'PositionError', 'trapmf', [-10 -10 -pi -0.5\*pi], 'Name', 'BigNegativeError');  fuzzy\_lab7 = addMF(fuzzy\_lab7, 'PositionError', 'trimf', [-pi -0.1\*pi 0], 'Name', 'NegativeError');  fuzzy\_lab7 = addMF(fuzzy\_lab7, 'PositionError', 'trimf', [-0.25\*pi 0 0.25\*pi], 'Name', 'Zero');  fuzzy\_lab7 = addMF(fuzzy\_lab7, 'PositionError', 'trimf', [0 0.1\*pi pi], 'Name', 'PositiveError');  fuzzy\_lab7 = addMF(fuzzy\_lab7, 'PositionError', 'trapmf', [0.5\*pi pi 10 10], 'Name', 'BigPositiveError');  % Функции принадлежности для выходного напряжения  fuzzy\_lab7 = addMF(fuzzy\_lab7, 'ControlVoltage', 'trimf', [-12 -12 -9], 'Name', 'HighNegativeVoltage');  fuzzy\_lab7 = addMF(fuzzy\_lab7, 'ControlVoltage', 'trimf', [-9 -8 -5], 'Name', 'NegativeVoltage');  fuzzy\_lab7 = addMF(fuzzy\_lab7, 'ControlVoltage', 'trimf', [-3.3 0 3.3], 'Name', 'Zero');  fuzzy\_lab7 = addMF(fuzzy\_lab7, 'ControlVoltage', 'trimf', [5 8 9], 'Name', 'PositiveVoltage');  fuzzy\_lab7 = addMF(fuzzy\_lab7, 'ControlVoltage', 'trimf', [9 12 12], 'Name', 'HighPositiveVoltage');  % Создание правил управления и добавление их к системе  fuzzy\_lab7 = addRule(fuzzy\_lab7, ["If (PositionError is BigNegativeError) then (ControlVoltage is HighPositiveVoltage)", ...  "If (PositionError is NegativeError) then (ControlVoltage is PositiveVoltage)", ...  "If (PositionError is Zero) then (ControlVoltage is Zero)", ...  "If (PositionError is PositiveError) then (ControlVoltage is NegativeVoltage)", ...  "If (PositionError is BigPositiveError) then (ControlVoltage is HighNegativeVoltage)"]); |

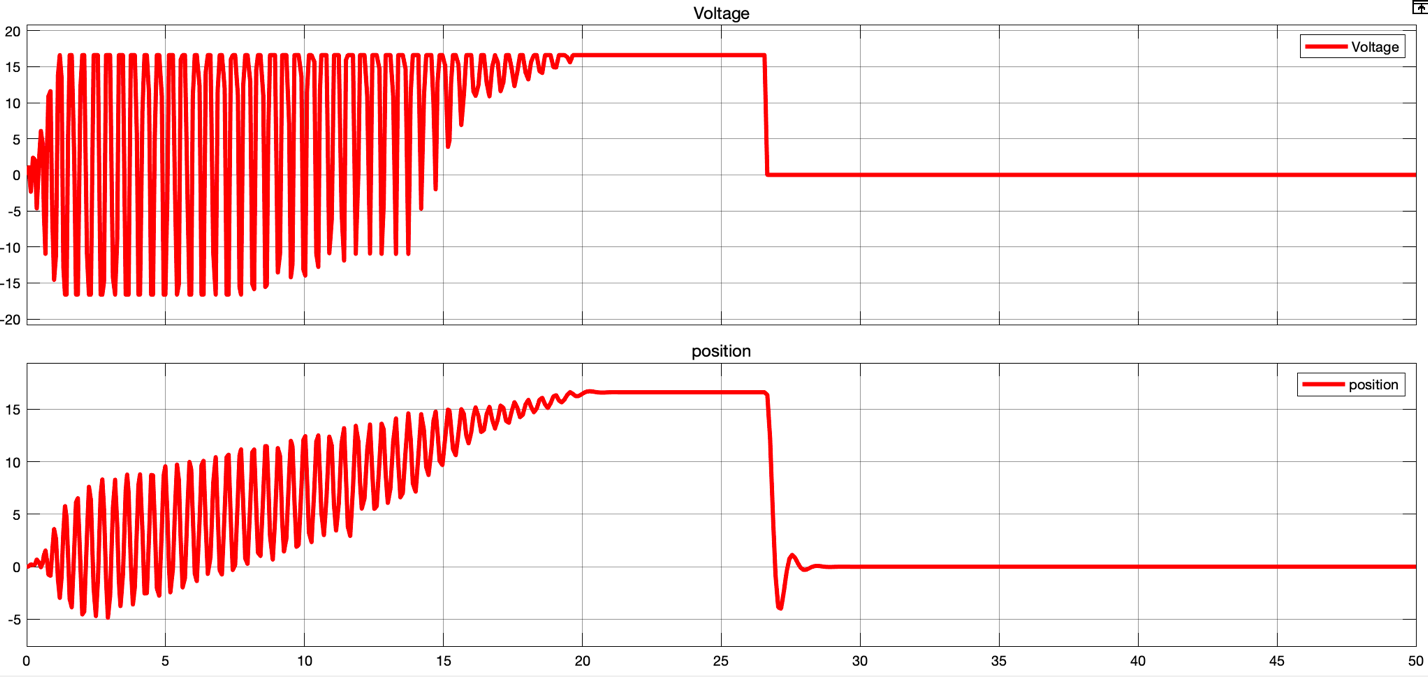


Рисунок 8 – результат моделирования системы с нечеткой логикой.

Вывод: система с нечеткой логикой построена.