Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS

Исследование алгоритмов интеллектуального управления двигателями переменного тока

Обучающийся / Student Любичев Ефим Денисович
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет систем управления и робототехники
Группа/Group R34372
Направление подготовки/ Subject area 15.03.06 Мехатроника и робототехника
Образовательная программа / Educational program Робототехника 2020
Язык реализации ОП / Language of the educational program Русский
Квалификация/ Degree level Бакалавр
Руководитель ВКР/ Thesis supervisor Голубев Антон Кириллович, Университет ИТМО, факультет систем управления и робототехники, ассистент (квалификационная категория "ассистент")

Обучающийся/Student

Документ подписан
Любичев Ефим Денисович
13.05.2024

(эл. подпись/ signature)

Любичев Ефим Денисович
(Фамилия И.О./ name and surname)

Руководитель ВКР/ Thesis supervisor

Документ подписан
Голубев Антон Кириллович
13.05.2024

(эл. подпись/ signature)

Голубев Антон Кириллович
(Фамилия И.О./ name and surname)

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ / OBJECTIVES FOR A GRADUATION THESIS

Обучающийся / Student Любичев Ефим Денисович
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет систем управления и робототехники
Группа/Group R34372
Направление подготовки/ Subject area 15.03.06 Мехатроника и робототехника
Образовательная программа / Educational program Робототехника 2020
Язык реализации ОП / Language of the educational program Русский
Квалификация/ Degree level Бакалавр
Тема ВКР/ Thesis topic Исследование алгоритмов интеллектуального управления двигателями переменного тока
Руководитель ВКР/ Thesis supervisor Голубев Антон Кириллович, Университет ИТМО, факультет систем управления и робототехники, ассистент (квалификационная категория "ассистент")

Характеристика темы ВКР / Description of thesis subject (topic)

Тема в области фундаментальных исследований / Subject of fundamental research: нет / not
Тема в области прикладных исследований / Subject of applied research: да / yes

Основные вопросы, подлежащие разработке / Key issues to be analyzed
Целью выпускной квалификационной работы является: исследование интеллектуальных алгоритмов управления двигателями переменного тока.
Задачи выпускной квалификационной работы:
1. Рассмотреть принципы и задачи различных методов интеллектуального управления двигателями переменного тока.
2. Провести аналитический обзор существующих алгоритмов интеллектуальному управлению двигателями переменного тока.
3. Выполнить компьютерное моделирование выбранных алгоритмов управления.
В работе рассматривается модель трехфазного асинхронного двигателя. Должно быть рассмотрено не менее двух методов интеллектуального управления. Моделирование проводиться при различных сценариях, возмущающих воздействиях и неопределенностях. По результатам моделирование формируется иллюстративный материал, характеризующий сравнительный анализ эффективности выбранных методов. Показателями эффективности выбрать: время переходного процесса, перерегулирование, ошибка регулирования.

Дата выдачи задания / Assignment issued on: 13.02.2024

Срок представления готовой ВКР / Deadline for final edition of the thesis 20.05.2024

СОГЛАСОВАНО / AGREED:

Руководитель ВКР/ Thesis supervisor

Документ подписан
Голубев Антон Кириллович
16.02.2024

(эл. подпись)

Голубев Антон Кириллович

Задание принял к исполнению/ Objectives assumed BY

Документ подписан
Любичев Ефим Денисович
27.02.2024

(эл. подпись)

Любичев Ефим Денисович

Руководитель ОП/ Head of educational program

Документ подписан
Бобцов Алексей Алексеевич
06.05.2024

(эл. подпись)

Бобцов Алексей Алексеевич

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

АННОТАЦИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ SUMMARY OF A GRADUATION THESIS

Обучающийся / Student Любичев Ефим Денисович
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет систем управления и робототехники
Группа/Group R34372
Направление подготовки/ Subject area 15.03.06 Мехатроника и робототехника
Образовательная программа / Educational program Робототехника 2020
Язык реализации ОП / Language of the educational program Русский
Квалификация/ Degree level Бакалавр
Тема ВКР/ Thesis topic Исследование алгоритмов интеллектуального управления двигателями переменного тока
Руководитель ВКР/ Thesis supervisor Голубев Антон Кириллович, Университет ИТМО, факультет систем управления и робототехники, ассистент (квалификационная категория "ассистент")

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ DESCRIPTION OF THE GRADUATION THESIS

Цель исследования / Research goal
Исследовать алгоритмы интеллектуального управления двигателями переменного тока
Задачи, решаемые в ВКР / Research tasks
1. Рассмотреть принципы и задачи различных методов интеллектуального управления двигателями переменного тока. 2. Провести аналитический обзор существующих алгоритмов интеллектуальному управлению двигателями переменного тока. 3. Выполнить компьютерное моделирование выбранных алгоритмов управления.
Краткая характеристика полученных результатов / Short summary of results/findings
1. Рассмотрены принципы и задачи методов интеллектуального управления асинхронными двигателями. 2. Поведен аналитический обзор существующих интеллектуальных алгоритмов управления асинхронными двигателями. По результатам которого, в качестве исследуемых алгоритмов были выбраны: нейросетевой регулятор и регулятор с нечеткой логикой. 3. Выполнено компьютерное моделирование выбранных алгоритмов.

Наличие публикаций по теме выпускной работы / Publications on the topic of the thesis

1. Кузнецов В.Д., Голубев А.К., Федоров Н.А., Любичев Е.Д. ИССЛЕДОВАНИЕ АЛГОРИТМОВ РОБАСТНОГО УПРАВЛЕНИЯ ДВИГАТЕЛЯМИ ПЕРЕМЕННОГО ТОКА - 2024 (Тезисы)

Обучающийся/Student

Документ подписан
Любичев Ефим Денисович
13.05.2024

(эл. подпись/ signature)

Любичев Ефим Денисович

(Фамилия И.О./ name and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан
Голубев Антон Кириллович
13.05.2024

(эл. подпись/ signature)

Голубев Антон Кириллович

(Фамилия И.О./ name and surname)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время электрические двигатели используются повсеместно, что подтверждает их большой спрос на рынке. Особенно часто электродвигатели переменного тока используются в промышленности, их выбирают за свою надежность, простоту в обслуживании и эксплуатации, а самое главное - цену. Помимо промышленности, электродвигатели активно захватывают новые рынки. Так, все больше набирают популярность электромобили, в состав которых входят электродвигатели переменного тока. С растущим спросом и расширением сфер применения, к электродвигателям выдвигаются все более сложные задачи, а значит и системы управления электродвигателями должны развиваться, чтобы выполнять необходимые задачи эффективнее.

На тему управления электродвигателей есть множество исследований, но все они основаны на классических подходах теории управления [1]. Одной из основных проблем, связанных с управлением электроприводами, является необходимость учета различных факторов, таких как: изменение нагрузки, изменение параметров из-за износа или дефекта, появление помех. В связи с этим, традиционные методы управления не всегда обеспечивают требуемую точность и надежность. Для решения этой проблемы предлагается использовать интеллектуальные методы управления, основанные на применении алгоритмов машинного обучения, нейронных сетей и других методов искусственного интеллекта. Такие методы позволяют автоматически адаптироваться к изменяющимся условиям работы системы, улучшать ее производительность и снижать энергопотребление. С развитием вычислительных технологий, методы искусственного интеллекта быстро шагают вперед, а цена на вычислительные мощности падает.

Данная работа посвящена исследованию интеллектуальных методов и алгоритмов управления электроприводами, которые позволяют повысить эффективность и надежность работы электромеханических систем в условиях неопределенностей.

Целью данной работы является: исследование методов интеллектуального управления электроприводом в условиях неопределенности. В рамках работы будут проведены теоретические исследования и анализ полученных результатов.

Для достижения цели ставятся ряд задач:

1. Рассмотреть принципы и задачи различных методов интеллектуального управления двигателями переменного тока.

2. Провести аналитический обзор существующих алгоритмов интеллектуальному управлению двигателями переменного тока.

3. Выполнить компьютерное моделирование выбранных алгоритмов управления.

В качестве практической значимости исследования можно выделить улучшение основных показателей регулирования скорости вращения электродвигателя, а именно: перерегулирование, время переходного процесса и ошибка регулирования. Улучшение этих параметров позволит достигать поставленных целей с помощью электрического привода точнее, быстрее и эффективнее.

1 Принципы управления асинхронным двигателем и его математическая модель

1.1 Схема замещения асинхронного двигателя

Для описания математической модели асинхронного двигателя рассмотрим его схему замещения. Чаще всего потерями намагничивания, по сравнению с активными потерями в обмотках статора можно пренебречь из-за особенностей проектирования асинхронных двигателей (АД) [2]. В этом случае в качестве модели для иллюстрации процессов, происходящих в двигателе используется Т-образная схема замещения (рисунок 1.1).

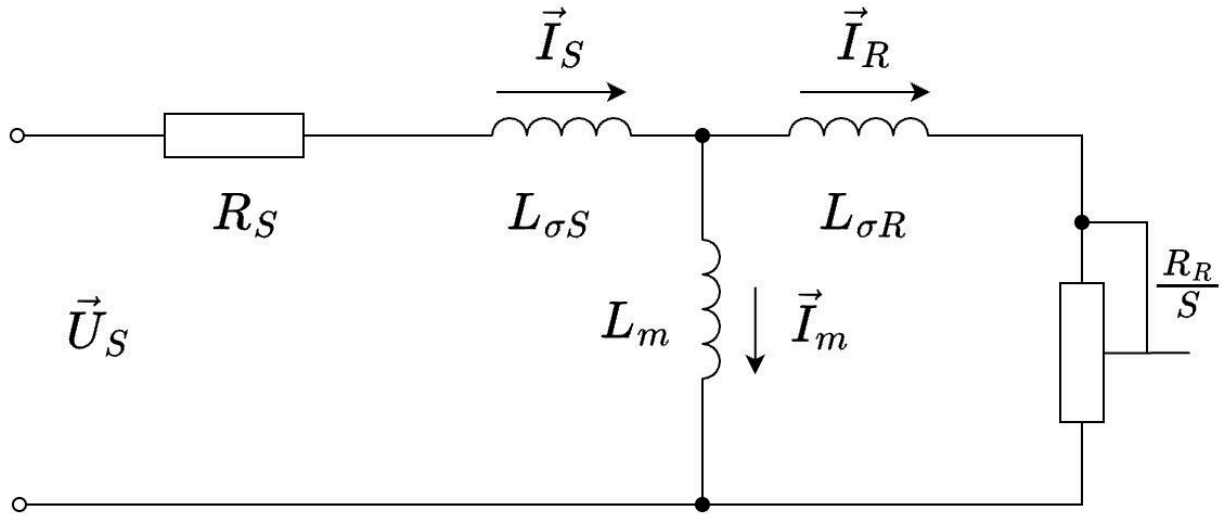

Рисунок 1.1 – Т-образная схема замещения асинхронного двигателя

Где на схеме замещения:

$\vec{U}_S$ – обобщенный вектор входного напряжения,

$\vec{I}_S$ – обобщенный вектор силы тока статора.

Вектор тока статора разбивается на:

$\vec{I}_m$ – обобщенный вектор тока намагничивания,

$\vec{I}_R$ – обобщенный вектор тока ротора.

Параметры схемы замещения:

R_s – сопротивление статора, которое характеризует активное сопротивление в статорных обмотках и равно сопротивлению фазной обмотки статора,

$L_{\sigma S}$ – индуктивность рассеяния обмотки статора, часть потока статора, которая не сцеплена с ротором и не принимает участие в создании момента,

$L_{\sigma R}$ – индуктивность рассеяния ротора, часть потока ротора, которая не сцеплена со статором и не принимает участие в создании момента,

L_m – главная индуктивность, характеризует часть потока, что сцеплена со статором и принимает участие в создании момента.

$\frac{R_R}{S}$ – активные потери в роторе, где:

R_R – активное сопротивление ротора в режиме короткого замыкания, когда вал двигателя не вращается.

$S = \frac{n_1 - n_2}{n_1}$ – скольжение, где n_1 – скорость вращения магнитного поля, n_2 – скорость вращения ротора.

1.2 Координатные преобразования

Для начала рассмотрим трехфазную систему координат, в которой легко представить электродвигатель. Она имеет три оси, расположенных под углом $120°$ по отношению друг у другу.

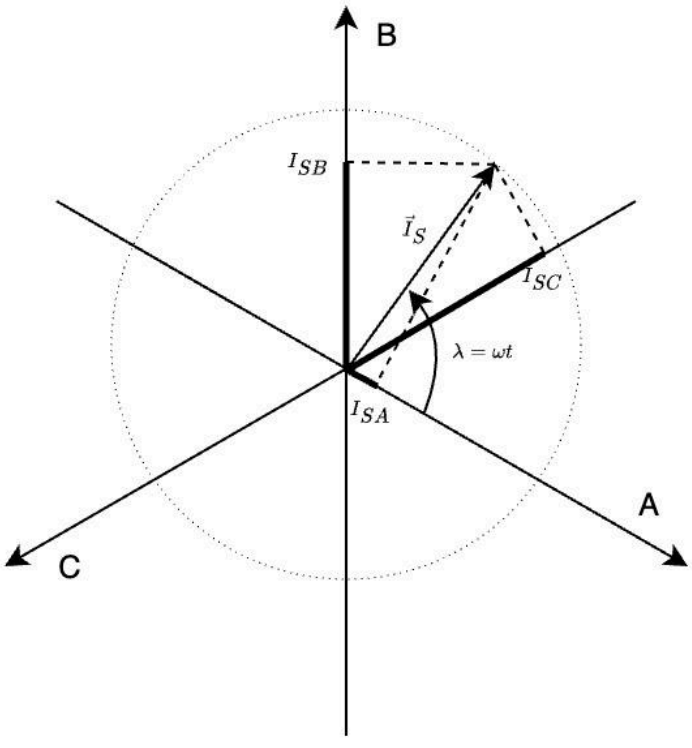

Рисунок 1.2 – Неподвижная трехфазная система координат

На рисунке 1.2 представлен обобщенный вектор тока, который связывает три скалярные величины, вращаясь в трехмерной системе координат. Проекции этого вектора на соответствующие оси будут представлять собой скалярные величины. Эта система обеспечивает условие симметрии трехфазной системы:

$$I_a + I_b + I_c = 0.$$

Получается, что данная неподвижная трехфазная система координат сцеплена со статором, а ее оси совпадают с электрическими осями обмоток двигателя.

Для упрощения системы координат и формирования управляющих воздействий пользуются переходом от реальной трехфазной машины к обобщенной двухфазной, описываемой декартовой системой координат с перпендикулярными неподвижными осями α и β. Обычно оси выбирают так, чтобы одна из них совпадала с трехфазной системой, а начала координат обеих систем тоже совпадали.

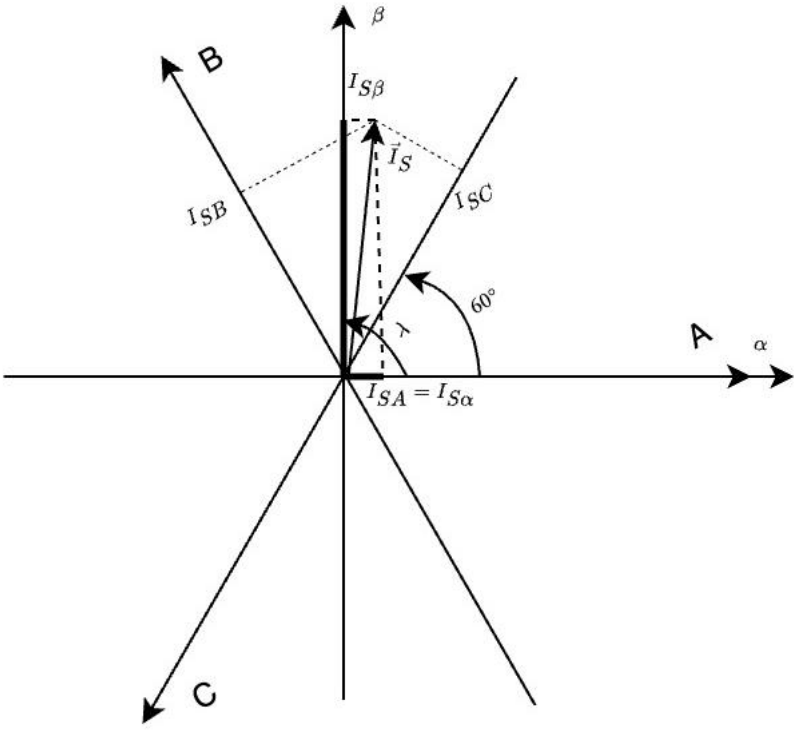

Рисунок 1.3 – Неподвижная двухфазная система координат

Для перехода к системе координат $\alpha\beta$ применяются преобразования Кларк:

$$\begin{cases} I_{S\alpha} = I_{SA}, \\ \mathrm{I}_{S\beta} = \dfrac{1}{\sqrt{3}} I_{SA} + \dfrac{2}{\sqrt{3}} I_{SA}. \end{cases} \quad (1.1)$$

Соответственно формулы обратного преобразования Кларк:

$$\begin{cases} I_{SA} = I_{S\alpha}, \\ \mathrm{I}_{SB} = \dfrac{\sqrt{3}}{2} I_{S\beta} - \dfrac{1}{2} I_{S\alpha}, \\ \mathrm{I}_{SC} = -\dfrac{1}{2} I_{S\alpha} - \dfrac{\sqrt{3}}{2} I_{S\beta}. \end{cases} \quad (1.2)$$

Для еще большего упрощения прибегают к вращающейся системе координат, так как дифференциальные уравнения в ней принимают простейший вид. Система имеет две перпендикулярные оси d и q, начало координат совпадает с остальными системами, но оси вращаются с произвольной скорость ω.

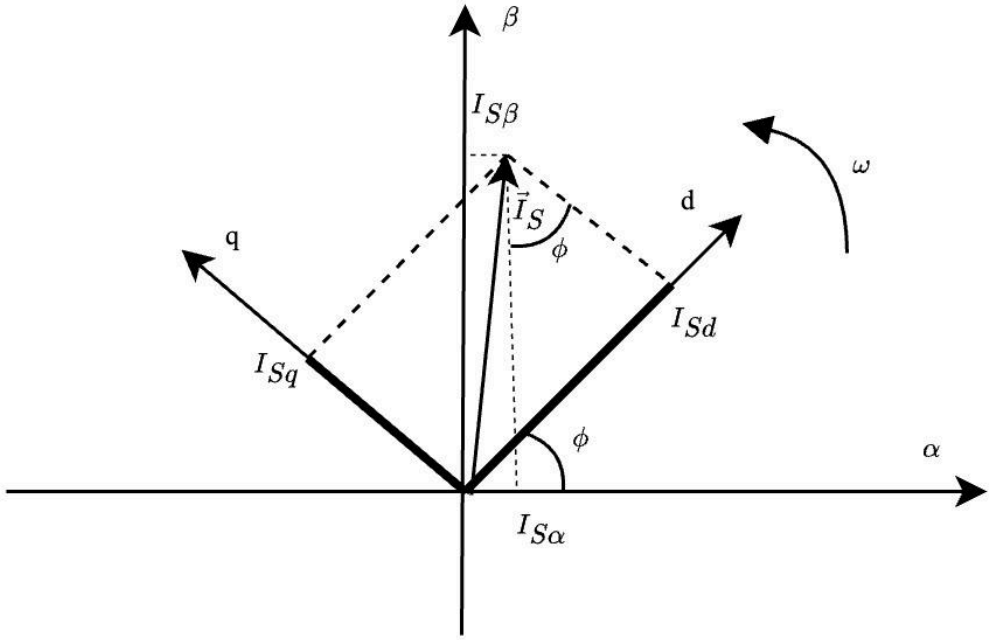

Рисунок 1.4 – Вращающаяся двухфазная система координат

Для перехода к системе dq применяются преобразования Парка:

$$\begin{cases} I_{Sd} = I_{S\beta} sin\phi + I_{S\alpha} cos\phi, \\ \mathrm{I}_{Sq} = I_{S\beta} cos\phi - I_{S\alpha} sin\phi. \end{cases} \quad (1.3)$$

Соответственно формулы обратного преобразования Парка:

$$\begin{cases} I_{S\alpha} = I_{Sd} cos\phi - I_{Sq} sin\phi, \\ \mathrm{I}_{S\beta} = I_{Sd} sin\phi + I_{Sq} cos\phi. \end{cases} \quad (1.4)$$

Наконец запишем соотношения, связывающие неподвижную трехфазную систем координат ABC с двухфазной вращающейся системой координат dq:

$$\begin{cases} I_{Sd} = \dfrac{I_{SB} - I_{SC}}{\sqrt{3}} sin\phi + I_{SA} cos\phi \,, \\ \mathrm{I}_{Sq} = \dfrac{I_{SB} - I_{SC}}{\sqrt{3}} cos\phi - I_{SA} sin\phi \,. \end{cases} \tag{1.5}$$

И обратное преобразование из вращающейся двухфазной в неподвижную трехфазную систему координат:

$$\begin{cases} I_{SA} = I_{Sd} cos\phi - I_{Sq} sin\phi \,, \\ \mathrm{I}_{SB} = \dfrac{\sqrt{3}}{2}(I_{Sd} sin\phi + I_{Sq} cos\phi) - \dfrac{1}{2} I_{SA} \,, \\ \mathrm{I}_{SC} = -\dfrac{\sqrt{3}}{2}(I_{Sd} sin\phi + I_{Sq} cos\phi) - \dfrac{1}{2} I_{SA} \,. \end{cases} \tag{1.6}$$

1.3 Математическая модель асинхронного двигателя

Рассмотрим векторную диаграмму АД, демонстрирующую взаимосвязь векторов состояний двигателя:

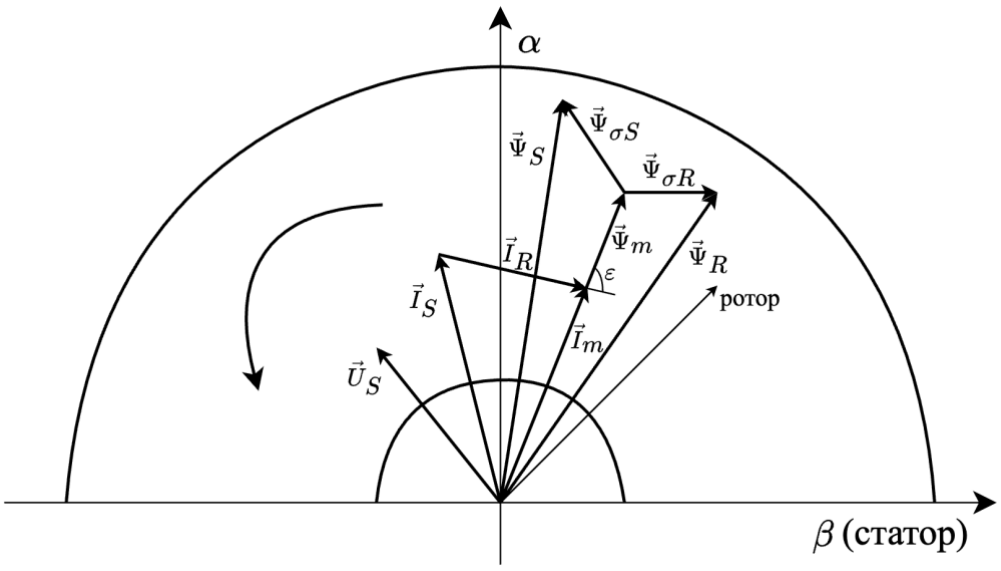

Рисунок 1.5 – Векторная диаграмма асинхронного двигателя

Во время работы все вектора вращаются вокруг оси вращения ротора. Запишем векторные уравнения на основе векторной диаграммы АД (рисунок 1.5).

Вектор тока намагничивания есть сумма векторов тока ротора и статора:

$$\vec{I}_m = \vec{I}_S + \vec{I}_R \,. \tag{1.7}$$

Вектор потокосцепления статора есть сумма векторов потоков намагничивания и рассеяния статора:

$$\vec{\Psi}_S = \vec{\Psi}_m + \vec{\Psi}_{\sigma S} \,. \tag{1.8}$$

Поток рассеяния статора равен произведению тока статора на индуктивность рассеяния ротора:

$$\vec{\Psi}_{\sigma S} = L_{\sigma S}\vec{I}_S \,. \tag{1.9}$$

Аналогично вектор потокосцепления ротора есть сумма векторов потоков намагничивания и рассеяния ротора:

$$\vec{\Psi}_R = \vec{\Psi}_m + \vec{\Psi}_{\sigma R} \,. \tag{1.10}$$

Поток рассеяния статора равен произведению тока статора на индуктивность рассеяния ротора:

$$\vec{\Psi}_{\sigma R} = L_{\sigma R}\vec{I}_R \,. \tag{1.11}$$

Запишем поток намагничивания как произведение индуктивности намагничивания двигателя на ток его намагничивания:

$$\vec{\Psi}_m = L_m\vec{I}_m \,. \tag{1.12}$$

При помощи уравнений (1.7 - 1.12) потоки выражаются через токи и индуктивности статора и ротора соответственно:

$$\vec{\Psi}_m = L_m\vec{I}_S + L_m\vec{I}_R \,. \tag{1.13}$$

Аналогично статора и ротора

$$\vec{\Psi}_S = L_S\vec{I}_S + L_m\vec{I}_R \,, \tag{1.14}$$

$$\vec{\Psi}_R = L_R\vec{I}_R + L_m\vec{I}_S \,, \tag{1.15}$$

где $\vec{L}_S = L_m + L_{\sigma S}$ – индуктивность статора, $\vec{L}_R = L_m + L_{\sigma R}$ – индуктивность ротора.

Запишем уравнение динамического равновесия статора и ротора:

$$\vec{U}_S = \frac{d\vec{\Psi}_S}{dt} + \vec{I}_S R_s \,. \tag{1.16}$$

При рассмотрении ротора с короткозамкнутой обмоткой, напряжение на которой равно нулю, получим следующие уравнение

$$0 = \frac{d\vec{\Psi}_S}{dt} + \vec{I}_S R_s \,. \tag{1.17}$$

В неподвижной системе координат $\alpha\beta$, которая связана со статором, уравнение динамического равновесия переписывается в виде:

$$\begin{cases} \mathrm{U}_{\mathrm{S}\alpha} = \dfrac{d\Psi_{S\alpha}}{dt} + I_{S\alpha} R_s \,, \\ \mathrm{U}_{\mathrm{S}\beta} = \dfrac{d\Psi_{S\beta}}{dt} + I_{S\beta} R_s \,. \end{cases} \tag{1.18}$$

Запишем скорость скольжения используя скорость поля ω_e и скорость ротора ω_R:

$$\Delta\omega = \omega_e - Z_p \omega_R \,. \tag{1.19}$$

Перепишем уравнения статора во вращающейся системе координат dq, где ось d ориентирована по потоку ротора, а система вращается в пространстве со скоростью поля ω_e:

$$\begin{cases} \mathrm{U}_{\mathrm{S}d} = \dfrac{d\Psi_{Sd}}{dt} + I_{Sd} R_s - \omega_e \Psi_{Sq} \,, \\ \mathrm{U}_{\mathrm{S}q} = \dfrac{d\Psi_{Sq}}{dt} + I_{Sq} R_s + \omega_e \Psi_{Sd} \,. \end{cases} \tag{1.20}$$

И уравнения ротора соответственно:

$$\begin{cases} 0 = \dfrac{d\Psi_{Rd}}{dt} + I_{Rd} R_R \,, \\ 0 = I_{Rq} R_R + \Delta\omega \Psi_{Rd} \,. \end{cases} \tag{1.21}$$

Запишем электромагнитный момент двигателя в системе координат dq:

$$M_{em} = \frac{3}{2} Z_p \left(I_{Sq} \Psi_{Sd} - I_{Sd} \Psi_{Sq} \right). \tag{1.22}$$

Запишем в проекциях на оси dq поток статора:

$$\begin{cases} \Psi_{Sd} = (L_s - L_m K_R) I_{Sd} + K_R \Psi_R \,, \\ \Psi_{Sq} = (L_s - L_m K_R) I_{Sq} \,. \end{cases} \tag{1.23}$$

где $K_R = \frac{L_m}{L_R}$.

Подставив в уравнение равновесия статора уравнения потока статора:

$$\begin{cases} pI_{Sd} = \dfrac{1}{L_s - L_m K_R}(U_{Sd} - I_{Sd}R_S - K_R \mathrm{p}\Psi_R) + \omega_\mathrm{e} I_{Sq}\,, \\ \mathrm{pI}_{Sq} = \dfrac{1}{L_s - L_m K_R}\left(U_{Sq} - I_{Sq}R_S - K_R \omega_\mathrm{e}\Psi_R\right) - \omega_\mathrm{e} I_{Sd}\,. \end{cases} \quad (1.24)$$

Запишем модель ротора:

$$p\Psi_\mathrm{R} = \frac{1}{T_R}(L_m I_{Sd} - \Psi_R)\,, \quad (1.25)$$

где $T_R = \frac{L_R}{R_R}$.

Рассчитаем скорость поля как:

$$\omega_e = Z_p\omega_R + \Delta\omega = Z_p\omega_R + \frac{L_m}{\Psi_R T_R} I_{Sq}\,. \quad (1.26)$$

Подставив в формулу (1.22) уравнения потока статора (1.23), получим электромагнитный момент ротора:

$$M_{em} = \frac{3Z_p K_R}{2}\Psi_R \times I_{Sq}\,. \quad (1.27)$$

Объединив уравнения в систему, получим описание электромагнитных процессов в АД:

$$\begin{cases} pI_{Sd} = \dfrac{1}{L_s - L_m K_R}(U_{Sd} - I_{Sd}R_S - K_R p\Psi_R) + \omega_e I_{Sq}\,, \\ pI_{Sq} = \dfrac{1}{L_s - L_m K_R}\left(U_{Sq} - I_{Sq}R_S - K_R \omega_e\Psi_R\right) - \omega_e I_{Sd}\,, \\ p\Psi_R = \dfrac{1}{T_R}(L_m I_{Sd} - \Psi_R)\,, \\ \omega_e = Z_p\omega_R + \dfrac{L_m}{\Psi_R T_R} I_{Sq}\,, \\ M_{em} = \dfrac{3Z_p K_R}{2}\Psi_R I_{Sq}\,. \end{cases} \quad (1.28)$$

Для вычисления скорости вращения ротора двигателя можно использовать следующую формулу:

$$\omega_R = \frac{1}{J}\int (M_{em} - M_c)dt\,, \quad (1.29)$$

где J – момент инерции, а M_c – момент сопротивления, приложенный к ротору.

1.4 Скалярное управление

Скалярное управление – это метод управления двигателем переменного тока, который заключается в поддержании постоянного отношения напряжения к частоте (В/Гц) во всем диапазоне рабочих скоростей с контролем только величины и частоты питающего напряжения:

$$u = \begin{bmatrix} V_S \\ f \end{bmatrix} = \begin{bmatrix} const \cdot f^* \\ f^* \end{bmatrix}, \tag{1.30}$$

где V_S – напряжение и f – частота питающего напряжения.

Отношение $\frac{V_S}{f}$ поддерживается константным, чтобы обеспечить постоянное потокосцепление статора.

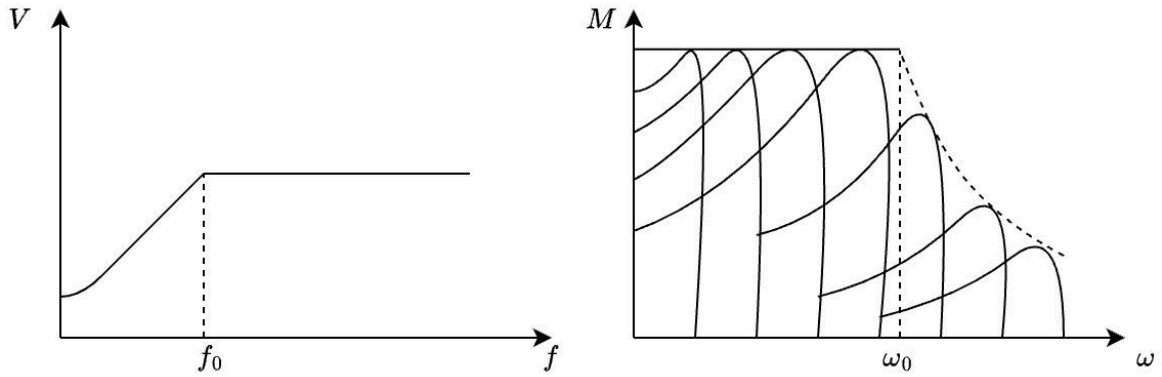

Рисунок 1.6 – Графики скалярного управления и семейство механических характеристик

Как можно заметить на Рисунке 1.6 при превышении синхронной частоты вращения момент начинает падать. Так же стоит отметить, что при низких частотах стоит поддерживать более высокое напряжение, так как двигатель может остановиться из-за потерь в обмотках статора.

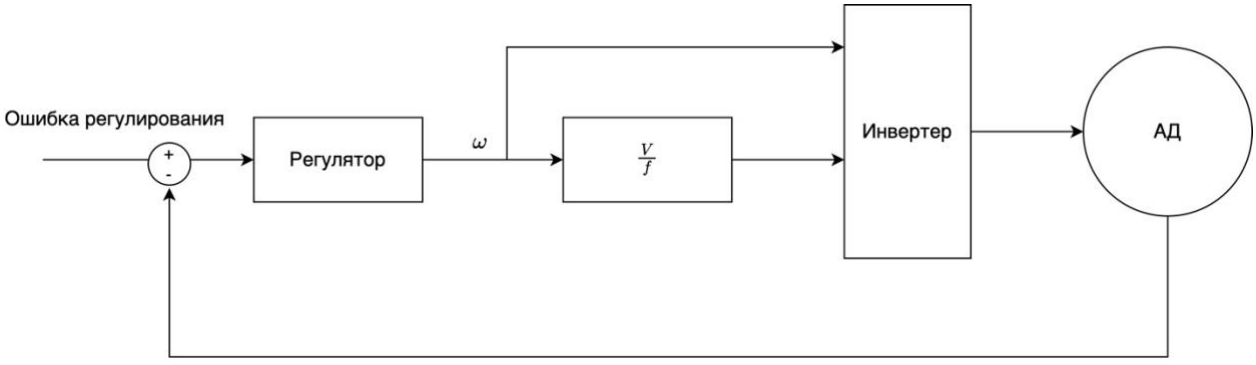

Рисунок 1.7 – Функциональная схема скалярного управления

На рисунке 1.7 представлена функциональная схема скалярного управления. Этот метод позволяет контролировать лишь скорость АД, причем обязательно для этого в системе должна быть обратная связь, иначе при изменении нагрузки появится установившаяся ошибка. Математической основой этого метода являются дифференциальные уравнения, описывающие установившиеся режимы работы двигателя, когда скорость и момент являются постоянными величинами.

1.5 Векторное управление

Векторное управление – метод управления двигателем переменного тока, основанный на управлении, не только формирующим гармонические токи фаз, но и обеспечивающим управление магнитным потоком ротора. Математической основой векторного управления являются дифференциальные уравнения, описывающие электродвигатель в динамических и статических режимах одинаково. Существует множество подвидов этого способа управления, но все они имеют сложную структуру, поэтому в данной работе этот метод не подлежит рассмотрению.

1.6 Выводы по главе 1

Была описана математическая система и координатные преобразования, позволяющие моделировать поведение АД. Для управления АД был выбран скалярный метод управления, так как он широко распространен, прост в реализации, но зачастую в нем применяется ПИД-регулятор, что вызывает сложность в настройке и дает не очень хорошие характеристики переходного процесса. Так же скалярный метод управления был подробно описан для его дальнейшей реализации.

2 Принципы и задачи интеллектуальных методов управления

2.1 Интеллектуальное управление

Термин «интеллектуальное» управление получил широкое распространение по мере расширения возможностей систем управления за пределы заранее заданных правил. Эта эволюция включает в себя переход от управления, основанного на правилах, к управлению, основанному на обучении, позволяющему системе приспосабливаться и улучшать свою работу на основе опыта и приобретенной информации [3].

Интеллектуальное управление — это новый вид метода управления в области автоматического управления, который включает в себя экспертную систему управления, нечетко-логическое управление, нейросетевое управление и генетические алгоритмы. Оно основано не только на теории искусственного интеллекта, но и на традиционной теории управления. Следовательно, новые методы управления могут быть разработаны путем комбинации этих способов управления. Интеллектуальное управление не опирается на математическую модель объекта управления, в которой целью управления является фактический результат управления. С другой стороны, интеллектуальное управление является крайне нелинейным. Например, нечеткая логика и ее правила являются одним из видов отображения, что отражает нелинейную природу системы.

Проблема управления АД привлекает внимание исследователей на протяжении многих лет. Большинство ранних исследований основано на классической теории управления и теории электрических машин, использующих точные математические модели асинхронного двигателя.

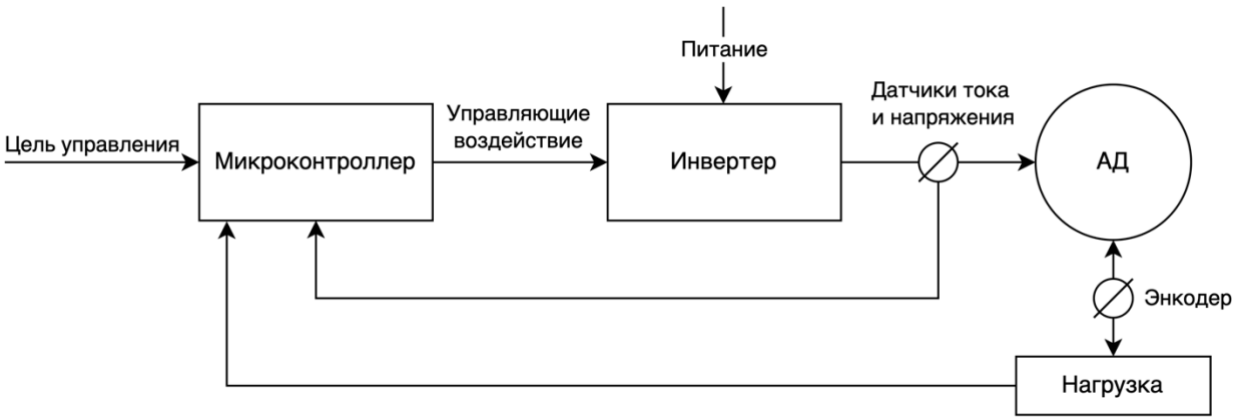

Рисунок 2.1 – Типовая схема управления асинхронного двигателя

Как показано на рисунке 2.1, система управления асинхронным двигателем состоит из микроконтроллера, датчиков, инвертора и асинхронного двигателя. Можно заметить, что изучение управления асинхронным двигателем затрагивает три основные области электротехники: управление, силовая электроника и электрические машины.

Асинхронный двигатель может быть описан нелинейными дифференциальными уравнениями. Задача управления усложняется тем, что асинхронный двигатель подвержен непредсказуемым возмущениям (таким как шум и изменение нагрузки), а также неопределенности в параметрах машины, возникающих, в частности, из-за нагрева. Управление асинхронным двигателем представляет собой теоретически интересный и практически важный класс нелинейных систем и превращается в эталонный пример нелинейного управления.

2.2 Нечеткая логика

Нечеткая логика – это одна из форм искусственного интеллекта, но ее история и применение более поздние, чем у экспертных систем. Утверждается, что человеческое мышление не всегда следует четкой логике "да-нет", а часто бывает расплывчатым, неопределенным, нерешительным или нечетким [5]. Основываясь на этом, Лофти Заде, ученый-информатик, в 1965 году представил "нечеткую логику" или теорию нечетких множеств, которая постепенно сформировалась как дисциплина в ИИ. Основная особенность

метода нечеткой логики заключается в использовании нечетких наборов правил и лингвистического представления знаний человека для описания объекта управления и построения нечеткого регулятора. Нечетко-логический регулятор состоит из фаззификации, нечеткого вывода с базой правил, а также дефаззификации (рисунок 2.2).

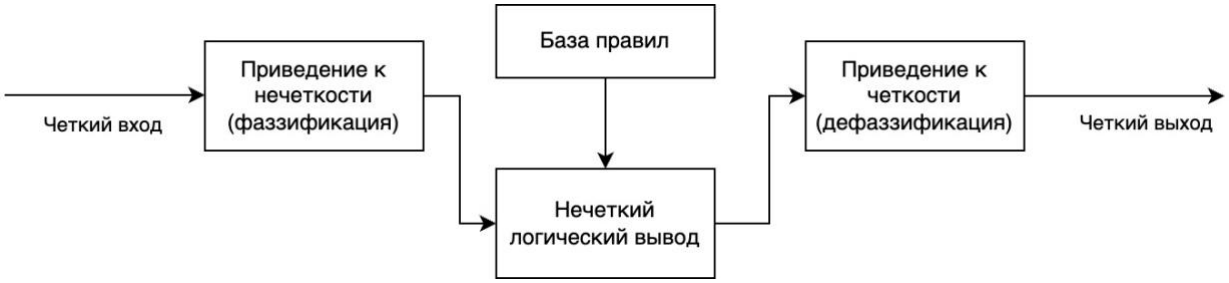

Рисунок 2.2 – Принцип работы нечеткой логики

Операция фаззификации реализует процесс преобразования четких входных значений в нечеткие множества. Нечеткое множество состоит из элементов, каждый из которых имеет степень принадлежности и связан с лингвистическими значениями.

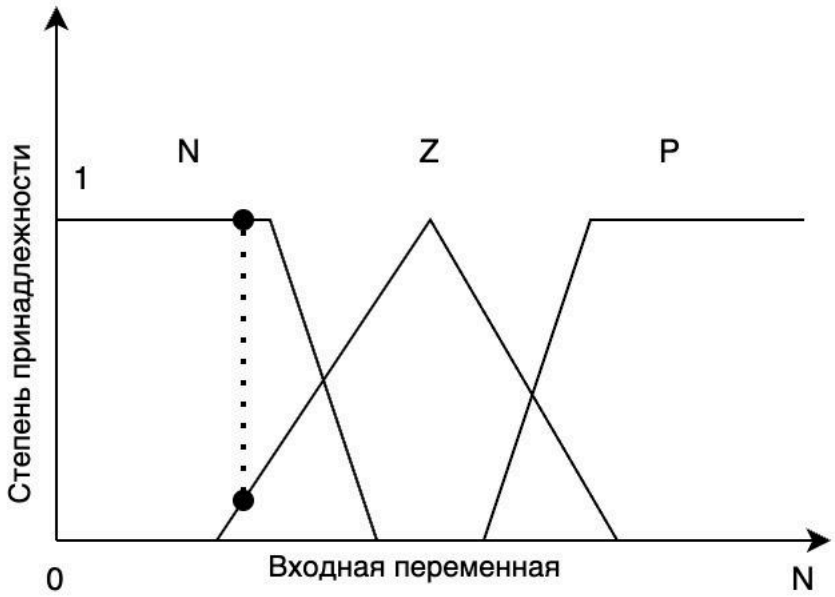

Рисунок 2.3 – Диаграмма фаззификации

На Рисунке 2.3 представлена диаграмма фаззификации, выполненная из трех функций принадлежности: треугольной и трапециевидных. Существует большое количество функций принадлежности, которые и позволяют определить принадлежность значения к нечеткому множеству. На диаграмме

по оси абсцисс представлена входная переменная, а по оси ординат мы получаем степень принадлежности к нечеткому множеству: $[1,\ 0.1, 0]$.

Затем идет операция применения базы правил к нечеткому входу и получения нечеткого выхода:

$$if\ in_1\ is\ N\ and\ in_2\ is\ N\ then\ out\ is\ PL$$
$$\dots$$
$$if\ in_1\ is\ P\ and\ in_2\ is\ P\ then\ out\ is\ NL$$

Логические операции для нечетких множеств могут быть реализованы несколькими способами, но самый простой из них это замена логической операции OR на математическую max, а операции AND на min. Операцию NOT можно представить как 1-N.

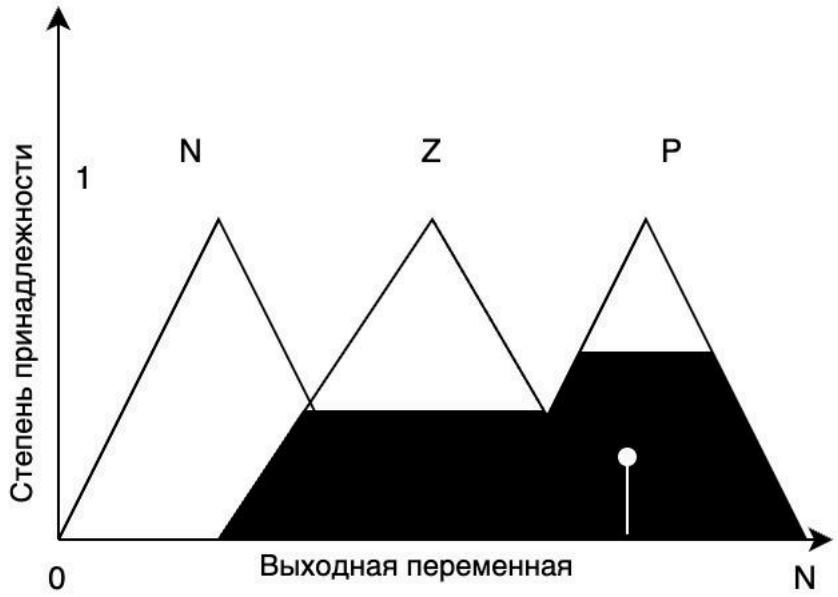

Рисунок 2.4 – Диаграмма дефаззификации

Операция дефаззификации – это процесс определения наилучшего числового значения для представления данного нечеткого множества, например это можно сделать, представив нечеткое множество как фигуру, а абсцисса центроида как раз и будет четким выводом. База правил содержит необходимые лингвистические управляющие правила для нечеткого вывода.

Соответственно для синтеза регулятора, основанного на нечеткой логике, нужно выбрать функции принадлежности для фаззификации и задать размеры пространства входных переменных. Затем нужно составить базу

правил, по которой будет обработано нечеткое множество. И наконец составить с помощью функций принадлежности дефаззификатор, задать ему размеры пространства выходных переменных и выбрать функцию деффазификации.

Некоторые нечетко-логические регуляторы уже были разработаны для управления асинхронными двигателями, например, полеориентированное управление с нечетким оптимизатором эффективности и прямое управление моментом на основе нечеткой логики. Так же регулятор на нечеткой логике часто используется в скалярном управлении. В системах управления с искусственным интеллектом нечеткая логика применяется для моделирования и управления сложной, неопределенной и нелинейной динамикой, позволяя создавать более адаптивные и гибкие стратегии управления.

2.3 Нейронные сети

Нейронная сеть – наиболее общая форма искусственного интеллекта для эмуляции человеческого мышления по сравнению с экспертными системами и нечеткой логикой [5]. В 1943 году Маккаллох и Питтс впервые предложили сеть, состоящую из бинарных искусственных нейронов, которые были способны выполнять простые пороговые логические вычисления. Современная эра нейронных сетей с возрождением исследований практически началась в 1982 году, когда Хопфилд представил свое изобретение. С тех пор было представлено множество моделей сетей и правил обучения. Нейронная сеть известна своей способностью к обучению и произвольной аппроксимации любой непрерывной функции. Нейронная сеть была создана на основе структуры и функциональности человеческого мозга.

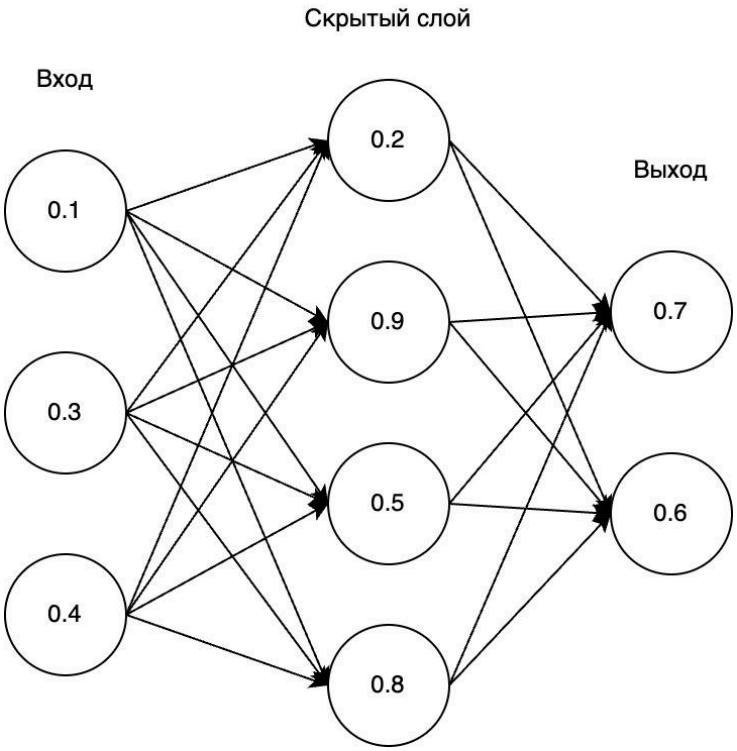

Рисунок 2.5 – Пример структуры простой нейронной сети

Также известная как искусственная нейронная сеть, она состоит из взаимосвязанных узлов, или "нейронов" как на рисунке 2.5, которые обрабатывают и передают информацию. Благодаря слоям взаимосвязанных нейронов нейронная сеть может научиться распознавать закономерности и структуры, делать прогнозы и выполнять сложные задачи по принятию решений. Исследования нейросетевого нелинейного динамического управления ведутся с 1988 года.

В последнее время нейронные сети используются для идентификации параметров и оценки состояния систем привода асинхронных двигателей. Так же нейронные сети широко используются для раннего обнаружения неисправностей двигателей. Гибридный нечеткий и нейронный регулятор (также называемый нейро-нечетким регулятором) был разработан для управления асинхронным двигателем мощностью 100 кВт. Нейронная сеть, обладающая преимуществом параллельных вычислений, может быть использована для уменьшения временных задержек контроллера, вызванных сложными вычислениями.

Нейронные сети являются частью машинного обучения, которое в свою очередь разделяется на три большие категории:

1. Обучение с учителем – позволяет найти скрытые структуры и закономерности в данных.

2. Обучение без учителя – решает задачу разметки данных, то есть кластеризации и поиска аномалий в данных.

3. Обучение с подкреплением – поиск лучшей последовательности событий, чтобы получить лучший результат в контексте награды.

Первые две категории не совсем подходят для синтеза регулятора, а вот обучение с подкреплением является задачей оптимизации и может решать задачи управления π^*:

$$\pi^* = \arg\max_{\pi} J(\pi) \ . \tag{2.1}$$

С помощью обучения с подкреплением можно синтезировать регулятор, сходный по характеристикам с линейно-квадратичным регулятором, но без глубокого знания о системе управления.

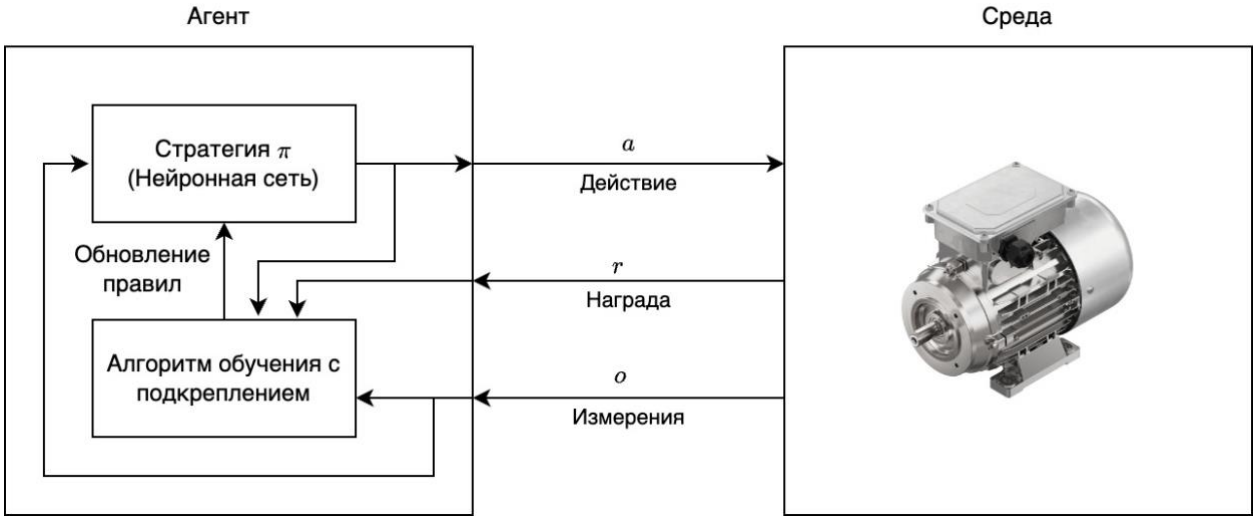

Рисунок 2.6 – Принцип обучения с подкреплением

Для синтеза регулятора с нейронной сетью, обученной с помощью метода обучения с подкреплением (рисунок 2.6), нужно совершить несколько шагов. Для начала нужно описать среду, в которой будет действовать агент, то есть наш регулятор. В нашем случае средой является асинхронный двигатель. Затем нужно выбрать такую функцию награды, чтобы агент научился решать

поставленную задачу. Выбрать структуру стратегии (нейронная сеть, метод машинного обучения или просто таблица значений) и алгоритм обучения. После этих шагов можно обучать агента и смотреть на метрики его эффективности, изменяя гиперпараметры при необходимости.

Существует множество вариантов стратегии поведения агента, которые зависят от алгоритма обучения. Алгоритмы обучения делятся на две большие группы: дискретные и непрерывные. В нашем случае модель является непрерывной, поэтому будет использован непрерывный алгоритм обучения. Оптимальным выбором является алгоритм Deep Deterministic Policy Gradients (DDPG), так как его легко настраивать и обучать, но при этом он хорошо решает поставленные задачи. DDPG – это алгоритм, который одновременно обучает Q-функцию и стратегию [7]. Он использует данные измерений и уравнение Беллмана для обучения Q-функции, а далее использует Q-функцию для обучения правил поведения. Q-функция – это функция полезности действий, то есть математическое ожидание отдачи при следовании стратегии (рисунок 2.7).

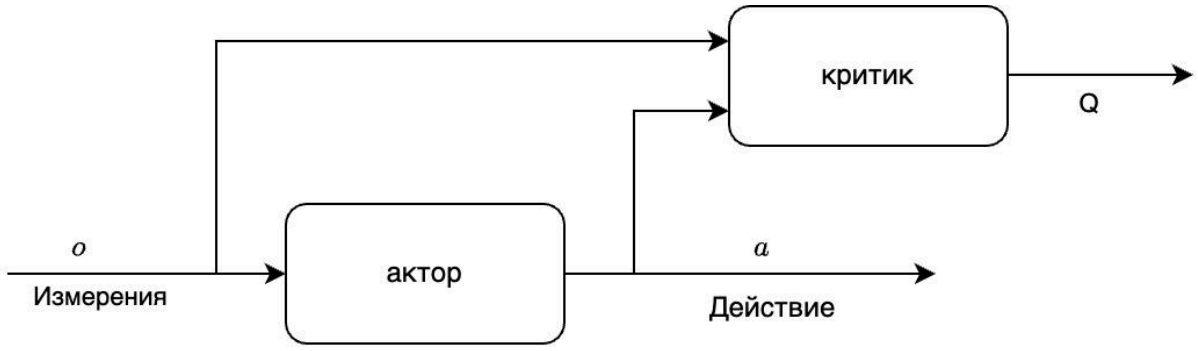

Рисунок 2.7 – Модель обучения актор-критик

Когда существует конечное число дискретных действий, максимальное значение не представляет проблемы, потому что мы можем просто вычислить Q-значения для каждого действия отдельно и напрямую сравнить их. Но когда пространство действий непрерывно, мы не можем исчерпывающе оценить это пространство, и решение проблемы оптимизации становится весьма нетривиальным. Использование обычного алгоритма оптимизации превратило бы вычисление $max_a Q^*(s,a)$ в дорогую задачу. А поскольку ее нужно будет

запускать каждый раз, когда агент захочет совершить какое-либо действие в окружении, это неприемлемо. Для упрощения этой задачи используется уравнение Беллмана:

$$Q^*(s,a) = Q^*(s,a) + \alpha[r(s,a) + \gamma \max \mathrm{Q}'(\mathrm{s}', \mathrm{a}') - Q(s,a)] \ , \qquad (2.2)$$

где s – это состояния объекта, a – это действия, α – коэффициент скорости обучения, r – это награда за текущее действие, γ – коэффициент дисконтирования. С помощью динамического программирования веса нейронной сети обновляются на каждой итерации обучения.

В этом алгоритме при обучении так же используется шум, который добавляется к стратегии, для того чтобы происходило исследование среды, так как стратегия детерминирована:

$$A_t = \pi^{\eta}(\mathrm{S_t}) + \mathrm{Noise} \ . \qquad (2.3)$$

2.4 Выводы по главе 2

Интеллектуальные методы управления являются очень перспективным направлением исследования. Были подробно рассмотрены и математически описаны различные методы интеллектуального управления, а именно нечеткая логика и нейронные сети. Так же были описаны шаги для синтеза алгоритмов управления для этих методов.

3 Моделирование и сравнение интеллектуальных алгоритмов управления асинхронным двигателем

3.1 Моделирование асинхронного двигателя

Формулы координатных преобразований 1.5, 1.6 были описаны с помощью стандартных блоков Simulink. Так же была описана и система уравнений 1.28 электромагнитных процессов асинхронного двигателя. К электромагнитному моменту двигателя была добавлена некоторая механическая нагрузка, а затем вычислена скорость по формуле 1.29. В результате получилась следующая модель Simulink (рисунок 3.1).

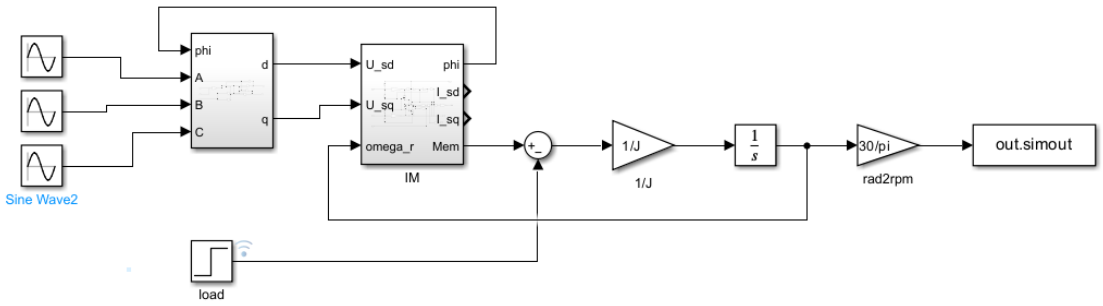

Рисунок 3.1 – Модель Simulink асинхронного двигателя

На вход этой модели поступает три синусоиды с амплитудой 220 Вольт, частотой 50 герц и фазовым сдвигом 120 градусов друг относительно друга. Для моделирования самого асинхронного двигателя используются его параметры, которые приведены в таблице 1.

Таблица 1 – Параметры асинхронного двигателя

Параметр	Значение
Сопротивление ротора R_r, Ом	8.3
Индуктивность ротора L_r, Гн	0.638
Сопротивление статора R_s, Ом	11.2
Индуктивность статора L_s, Гн	0.6155
Общая индуктивность мотора L_m, Гн	0.57
Количество пар полюсов Z_p	2
Момент инерции двигателя J, кг · м^2	0.00214

В результате моделирования описанной выше системы получили следующий график скорости АД (рисунок 3.2).

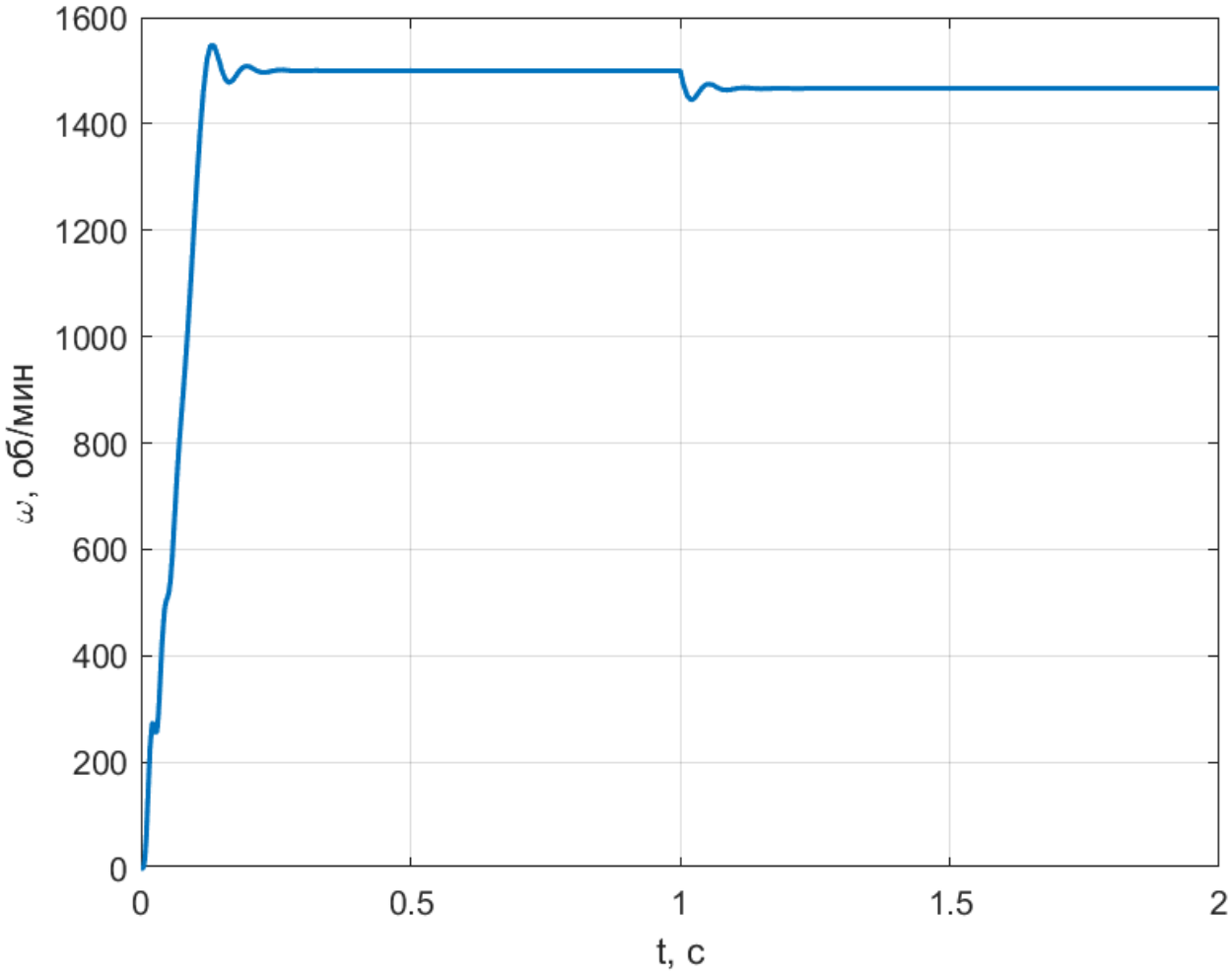

Рисунок 3.2 – График моделирования скорости асинхронного двигателя

Как видно на рисунке 3.2 двигатель постепенно разгоняется до скорости своего холостого хода, затем во времени 1 секунды появляется механическая нагрузка 1 Н · м и из-за этого увеличивается скольжение.

3.2 ПИ регулятор

Был синтезирован и настроен ПИ регулятор АД основанный на скалярном управлении из главы 1. Согласно рисунку 1.7 была построена схема Simulink, но в нее еще была добавлена положительная обратная связь (рисунок 3.3).

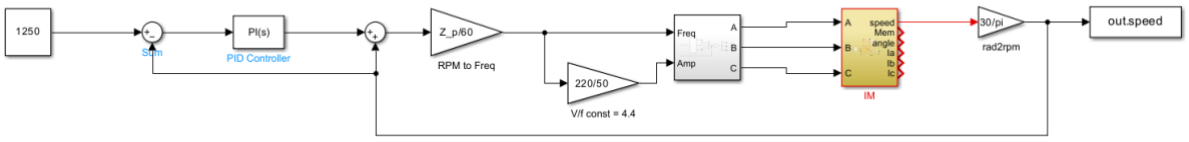

Рисунок 3.3 – Модель Simulink скалярного управления скоростью асинхронного двигателя с ПИ-регулятором

Для регулятора были экспериментально подобранны коэффициенты $k_p = 0.42$ и $k_i = 0.87$. В результате получился следующий график моделирования (рисунок 3.4).

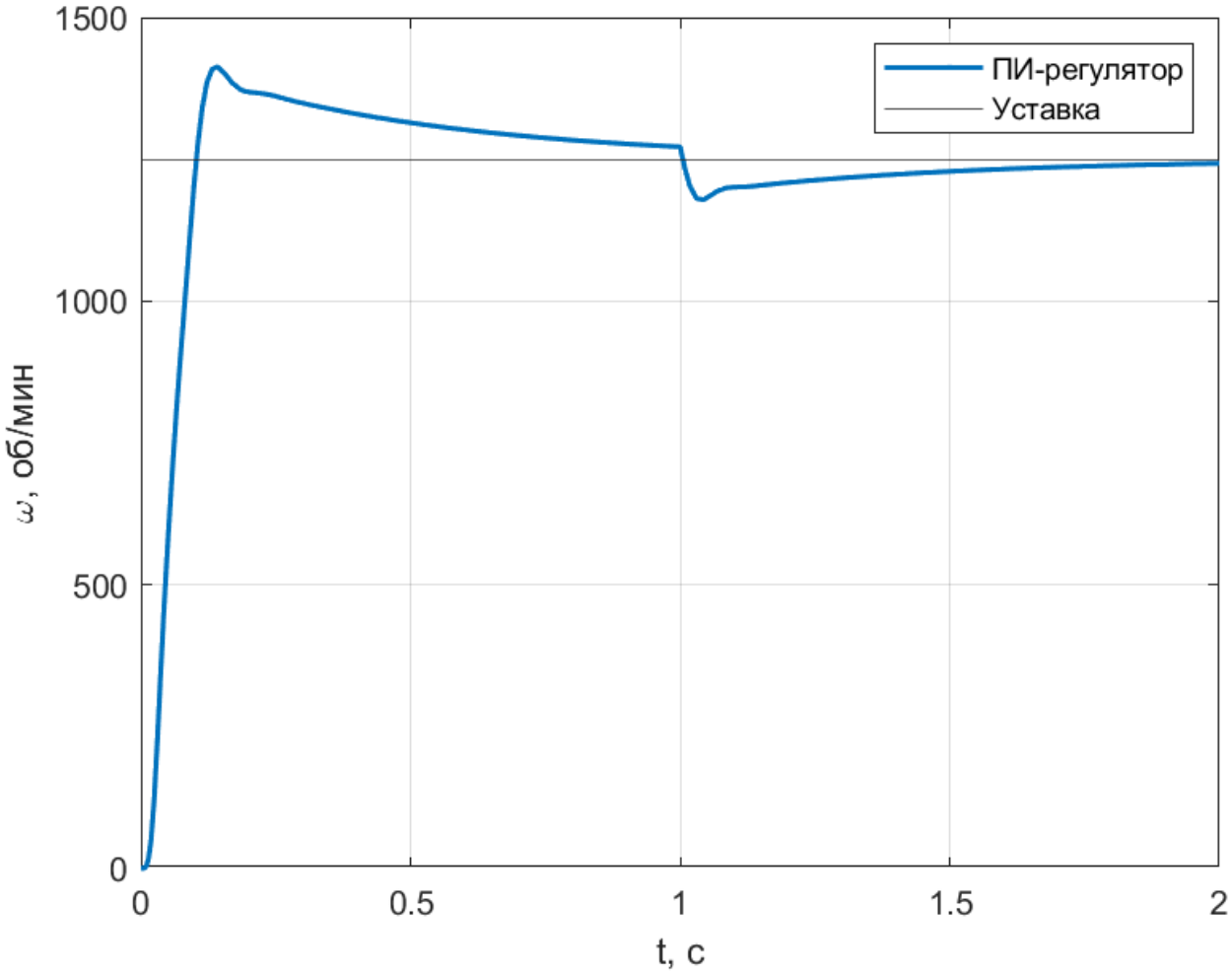

Рисунок 3.4 – График моделирования скорости асинхронного двигателя замкнутого ПИ-регулятором

Как видно на рисунке 3.4 у системы большое перерегулирование. Время переходного процесса составляет примерно 1 секунду. Так же за счет интегральной составляющей удается компенсировать внешнее возмущение и поддержать скорость на заданном уровне.

3.3 Регулятор с нечеткой логикой

Перейдем к синтезу регулятора с нечеткой логикой с двумя входами и одним выходом, для этого заменим блок ПИ-регулятора в модели из предыдущего пункта на fuzzy logic controller, а также добавим на вход приращение ошибки с помощью блока unit delay (рисунок 3.5).

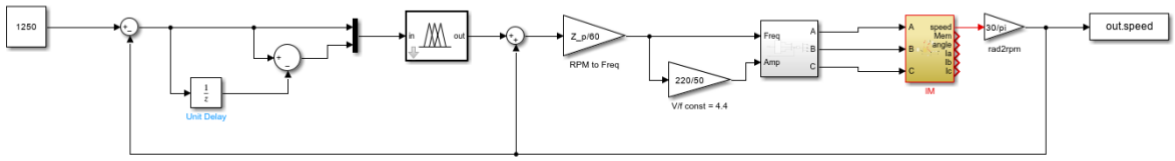

Рисунок 3.5 – Модель Simulink скалярного управления скоростью асинхронного двигателя с нечетким регулятором

Далее выполним команду в MATLAB fuzzyLogicDesigner и запустим редактор нечеткого регулятора [6]. Создадим новый нечеткий регулятор Mamdani Type-1 и перейдем к его настройке. Для начала добавим еще один вход и перейдем к конфигурации функций принадлежности для операции фаззификации. Возьмем самую простую конфигурацию системы, где у нас будет 3 нечетких множества: N (Negative) – для отрицательных значений, Z (Zero) – для значений близким к нулю и P (Positive) – для положительных значений [4]. В нашем случае у двигателя две пары полюсов, а значит его синхронная частота вращения около 1500 оборотов в минуту. Поэтому установим диапазон принимаемых значений от -2000 до 2000. Далее будем использовать простую логику: значения нуля множество будет принимать около нуля, поэтому для этого достаточно выбрать треугольную функцию принадлежности. А для положительных и отрицательных значений используем трапециевидную функцию принадлежности, где вершина трапеции будет начинаться рядом с синхронной скоростью АД. Получим следующую функцию принадлежности (рисунок 3.6).

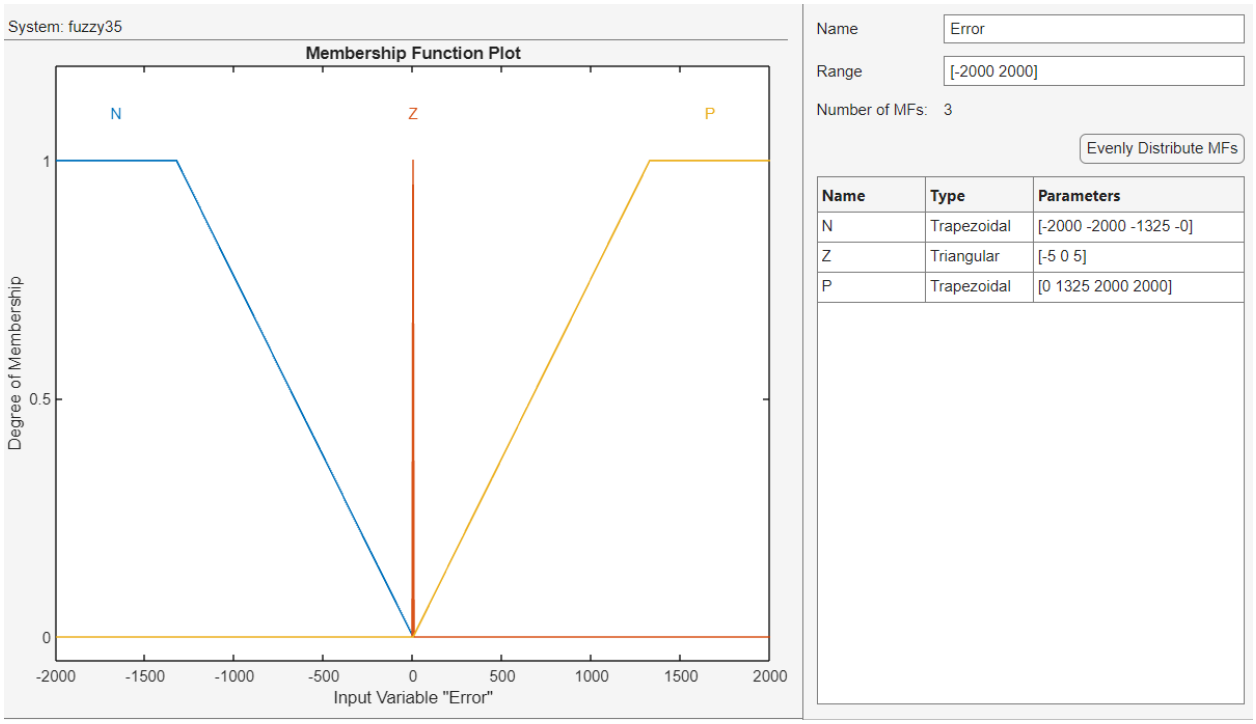

Рисунок 3.6 – Функция принадлежности для ошибки регулирования скорости

Аналогичным образом сделаем функцию принадлежности для приращения ошибки регулирования скорости. Только при этом немного увеличим диапазон для значений около нуля, так как приращение принимает большие значения.

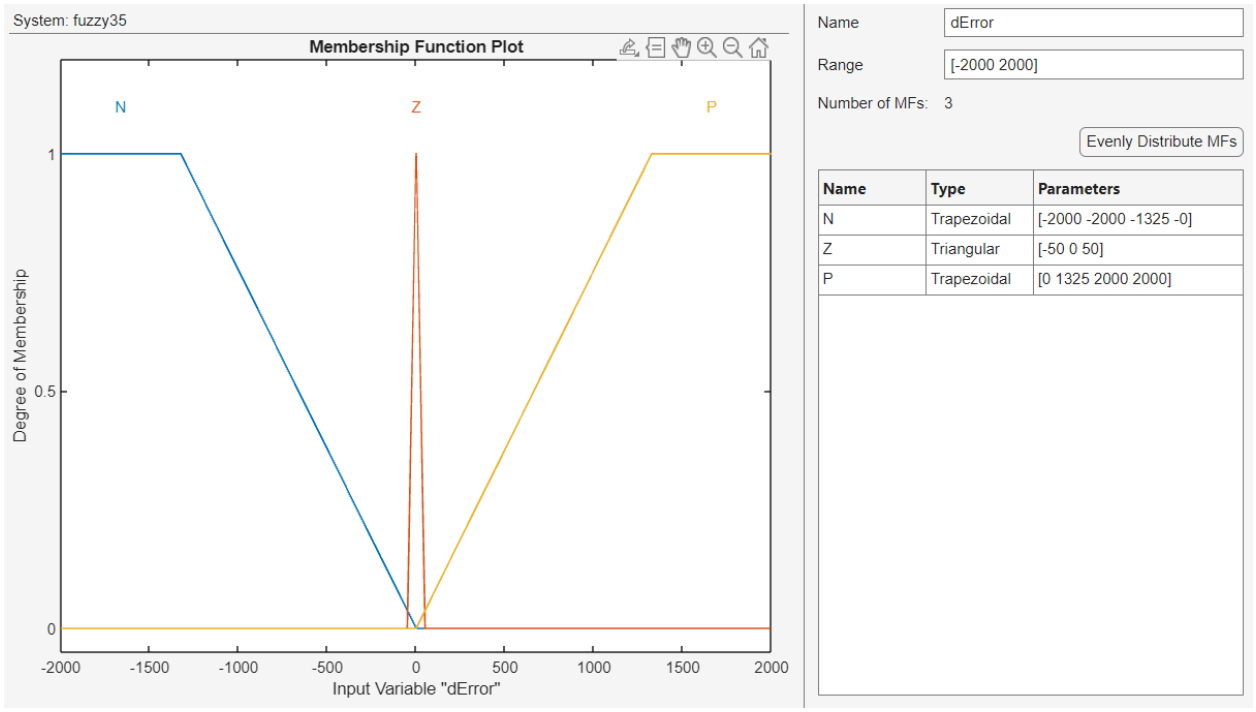

Рисунок 3.7 – Функция принадлежности для приращения ошибки регулирования скорости

Далее зададим нечеткие множества для операции деффазификации. Используем уже не 3, а 5 множеств, так как управляющее воздействие требует подробное описание: NL (Negative Large) – для больших отрицательных значений, NS (Negative Small) – для маленьких отрицательных значений, ZE (Zero) – для значений близким к нулю, PS (Positive Small) – для маленьких положительных значений, PL (Positive Large) – для больших положительных значений. Так как в системе присутствует положительная связь, то диапазон принимаемых значений уменьшается от -1000 до 1000.

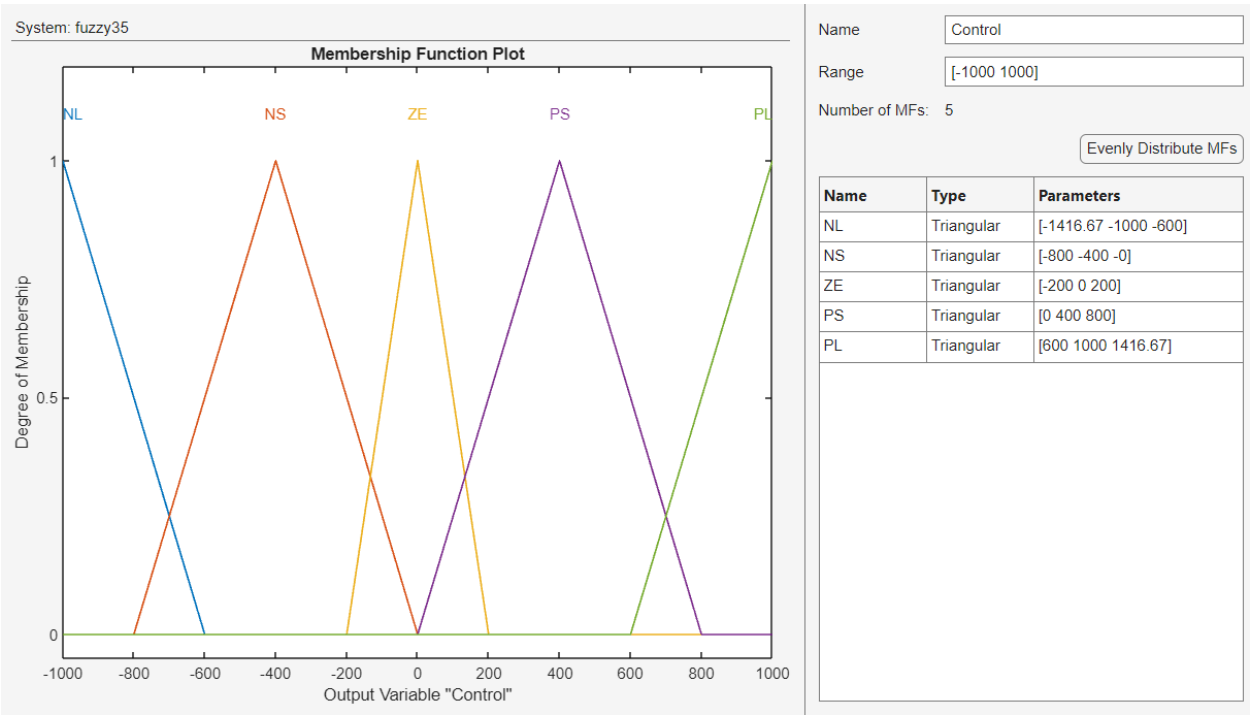

Рисунок 3.8 – Функция принадлежности для операции дефаззификации

Остается только описать нечеткие правила, по которым будет действовать регулятор. Количество правил определяет количество входов и количество функций принадлежности в них. Обычно это их декартово произведение. Запишем правила в виде таблицы (таблица 2), пользуясь логикой, что основой регулирования является ошибка, а приращение ошибки говорит о том, как быстро мы приближаемся к заданному значению.

Таблица 2 – База правил нечеткой логики

E / dE	N	Z	P
N	NL	NS	PS
Z	NL	ZE	PL
P	PS	PS	PL

Регулятор готов, а это значит, что можно построить поверхность управления для наглядной демонстрации синтезированного алгоритма (рисунок 3.9).

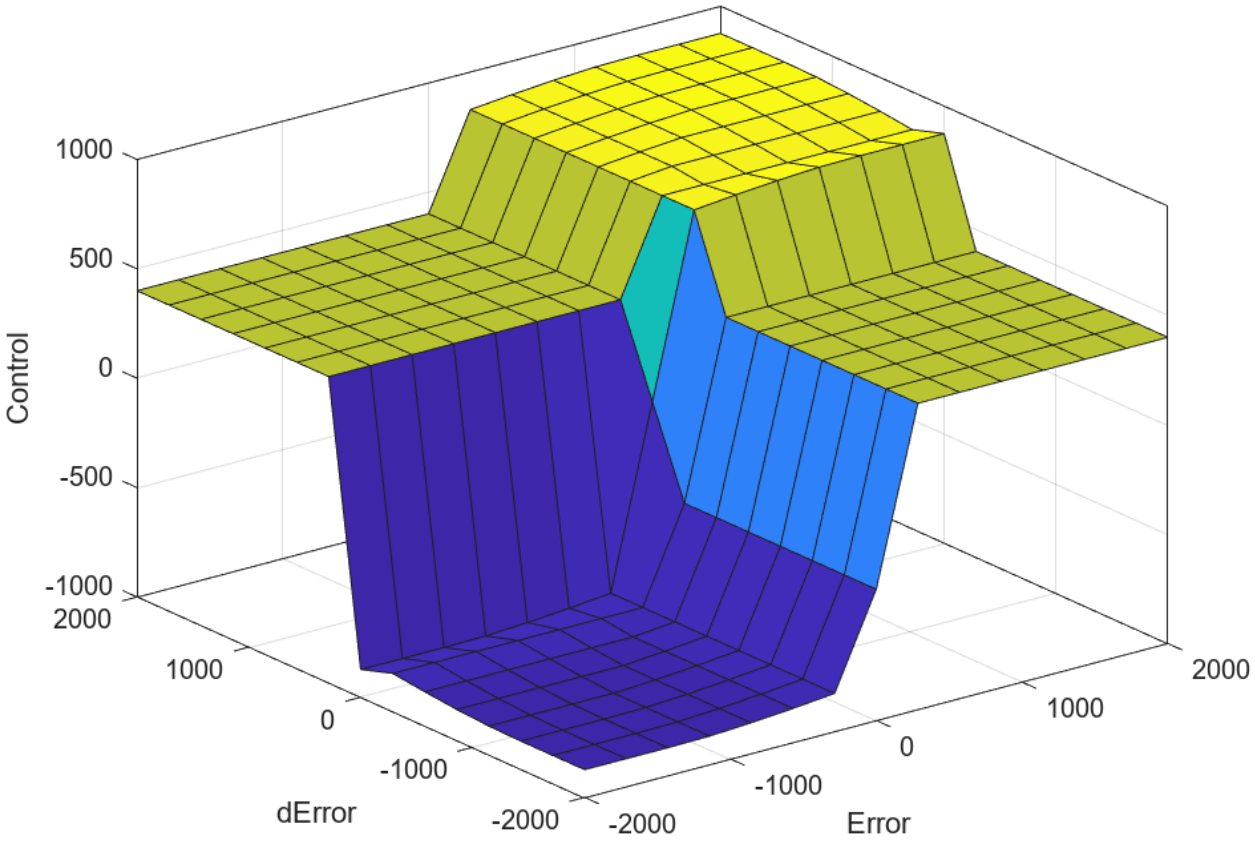

Рисунок 3.9 – Поверхность управления построенного регулятора

Запустим моделирование с построенным регулятором и получим следующий результат (рисунок 3.10).

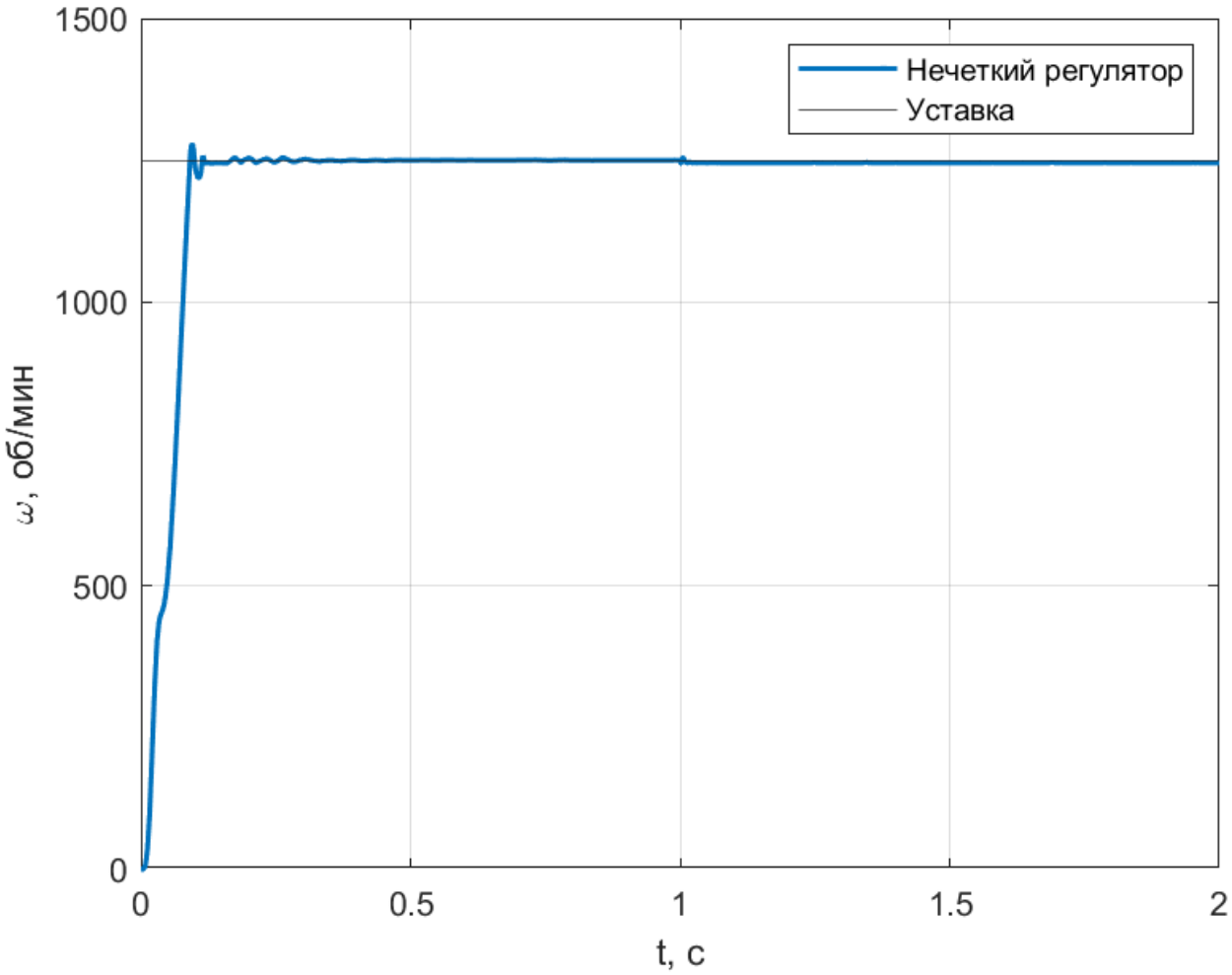

Рисунок 3.10 – График моделирования скорости асинхронного двигателя замкнутого нечетким регулятором

На рисунке 3.10 отражено перерегулирование порядка 30 об/мин, быстрое время переходного процесса равное 0.25 секунд, сначала установившаяся ошибка отсутствует, но при появлении нагрузки на 1 секунде появляется небольшая установившаяся ошибка.

3.4 Регулятор с применением нейронной сети

Основываясь на информации из главы 2.3, приступим к синтезу нейросетевого регулятора. Среда была описана в главе 3.1, поэтому позаимствуем ее там. Далее определим функцию награды для обучения стратегии. Измерениями регулятора будут: текущая скорость, ошибка регулирования скорости, интеграл ошибки регулярования скорости и приращение ошибки регулирования скорости. Подбор правильной награды

сильно влияет на качество и результат обучения с подкреплением. После множества экспериментальных итераций была выбрана следующая функция:

$$r = -0.1|e| + |e| \leq 1 - 0.05|e_i| - 0.05|e_d| \ , \qquad (3.1)$$

где e – ошибка регулирования скорости, e_i – интеграл ошибки регулирования скорости, e_d – приращение ошибки скорости.

В итоге получилась следующая модель Simulink для обучения и моделирования асинхронного двигателя, замкнутого нейросетевым регулятором (рисунок 3.11).

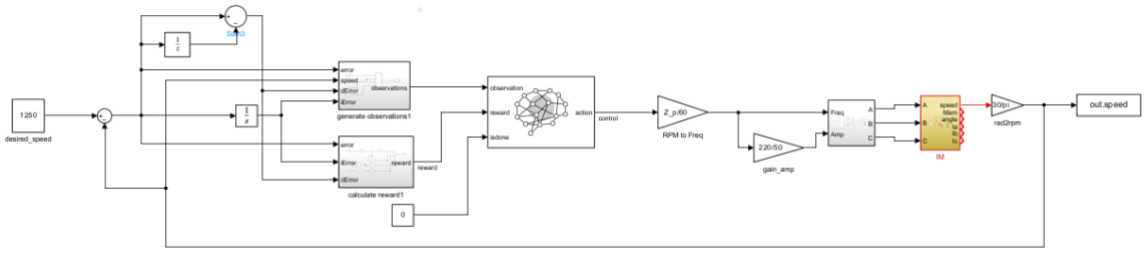

Рисунок 3.11 – Модель Simulink скалярного управления скоростью асинхронного двигателя с нейросетевым регулятором

Стоит отметить, что в данной конфигурации системы управления у регулятора исключена положительная обратная связь, так как она усложняет динамику системы и делает процесс обучения более долгим.

Далее с помощью кода в среде MATLAB создадим окружение (рисунок 3.12).

Определим объекты для действий и измерений.

```
% Observation info
obsInfo = rlNumericSpec([4 1],...
    LowerLimit=[-2000 -2000 -2000 -2000]',...
    UpperLimit=[2000 2000 2000 2000]');
obsInfo.Name = "observations";
obsInfo.Description = "error, d_error, i_error, speed";

% Action info
actInfo = rlNumericSpec([1 1], LowerLimit=-2000, UpperLimit=2000);
actInfo.Name = "control";
```

Создадим объект окружения.

```
env = rlSimulinkEnv("vf_nn", "vf_nn/RL Agent", obsInfo, actInfo);
```

Установим окружению функцию: которая будет случайно устанавливать начальные условия и уставку при обучении.

```
env.ResetFcn = @(in)localResetFcn(in);
```

Установим время моделирования `Tf` и шаг моделирования `Ts` в секундах.

```
Ts = 0.005;
Tf = 3;
```

Рисунок 3.12 – Листинг кода в программе MATLAB для создания окружения

Для окружения отдельно определим функцию задания начальных условий и уставки (рисунок 3.13).

```
function in = localResetFcn(in)

% Randomize reference signal
blk = sprintf("vf_nn/desired_speed");
speed = 500*randn + 1000;
while speed <= 800 || speed >= 1500
    speed = 500*randn + 1000;
end
in = setBlockParameter(in,blk,Value=num2str(speed));

% Randomize initial speed
speed = 1000*randn;
while speed < 0 || speed >= 1500
    speed = 1000*randn;
end
speed = speed * pi/30;
blk = "vf_nn/IM/spdi";
in = setBlockParameter(in,blk,InitialCondition=num2str(speed));

end
```

Рисунок 3.13 – Функция задания начальных условий окружения и уставки

После инициализации окружения для обучения перейдем к созданию нейронной сети критика. Определим ее так же, с помощью кода в MATLAB (рисунок 3.14).

```
% Observation path
obsPath = [
    featureInputLayer(obsInfo.Dimension(1),Name="obsInLyr")
    fullyConnectedLayer(64)
    reluLayer
    fullyConnectedLayer(64,Name="obsPathOutLyr")
    ];

% Action path
actPath = [
    featureInputLayer(actInfo.Dimension(1),Name="actInLyr")
    fullyConnectedLayer(64,Name="actPathOutLyr")
    ];

% Common path
commonPath = [
    additionLayer(2,Name="add")
    reluLayer
    fullyConnectedLayer(64)
    reluLayer
    fullyConnectedLayer(1,Name="QValue")
    ];

criticNetwork = layerGraph();
criticNetwork = addLayers(criticNetwork,obsPath);
criticNetwork = addLayers(criticNetwork,actPath);
criticNetwork = addLayers(criticNetwork,commonPath);

criticNetwork = connectLayers(criticNetwork, ...
    "obsPathOutLyr","add/in1");
criticNetwork = connectLayers(criticNetwork, ...
    "actPathOutLyr","add/in2");

criticNetwork = dlnetwork(criticNetwork);
critic = rlQValueFunction(criticNetwork, ...
    obsInfo,actInfo, ...
    ObservationInputNames="obsInLyr", ...
    ActionInputNames="actInLyr");
```

Рисунок 3.14 – Листинг кода в программе MATLAB для создания критика

У получившейся нейронной сети 8.8 тысяч нейронов для обучения. На 1 вход подается 4 признака измерений, а на второй вход подается выход актора. У нейронной сети есть несколько скрытых слоев и добавочный слой (рисунок 3.15).

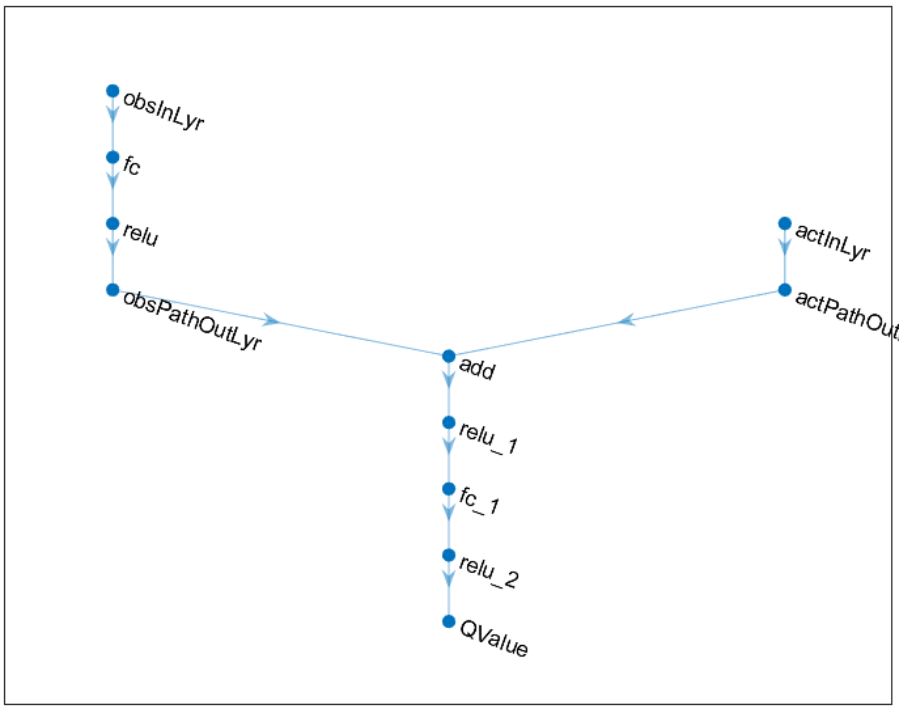

Рисунок 3.15 – Схема нейронной сети критика

Аналогичным образом определим нейронную сеть актора (рисунок 3.16).

```
actorNet = [
    featureInputLayer(obsInfo.Dimension(1))
    fullyConnectedLayer(64)
    reluLayer
    fullyConnectedLayer(64)
    reluLayer
    fullyConnectedLayer(actInfo.Dimension(1))
    ];
actorNet = dlnetwork(actorNet);
actor = rlContinuousDeterministicActor(actorNet,obsInfo,actInfo);
```

Рисунок 3.16 – Листинг кода в программе MATLAB для создания актора

У получившейся нейронной сети 4.5 тысяч нейронов для обучения и 1 вход с 4 признаками измерений (рисунок 3.17).

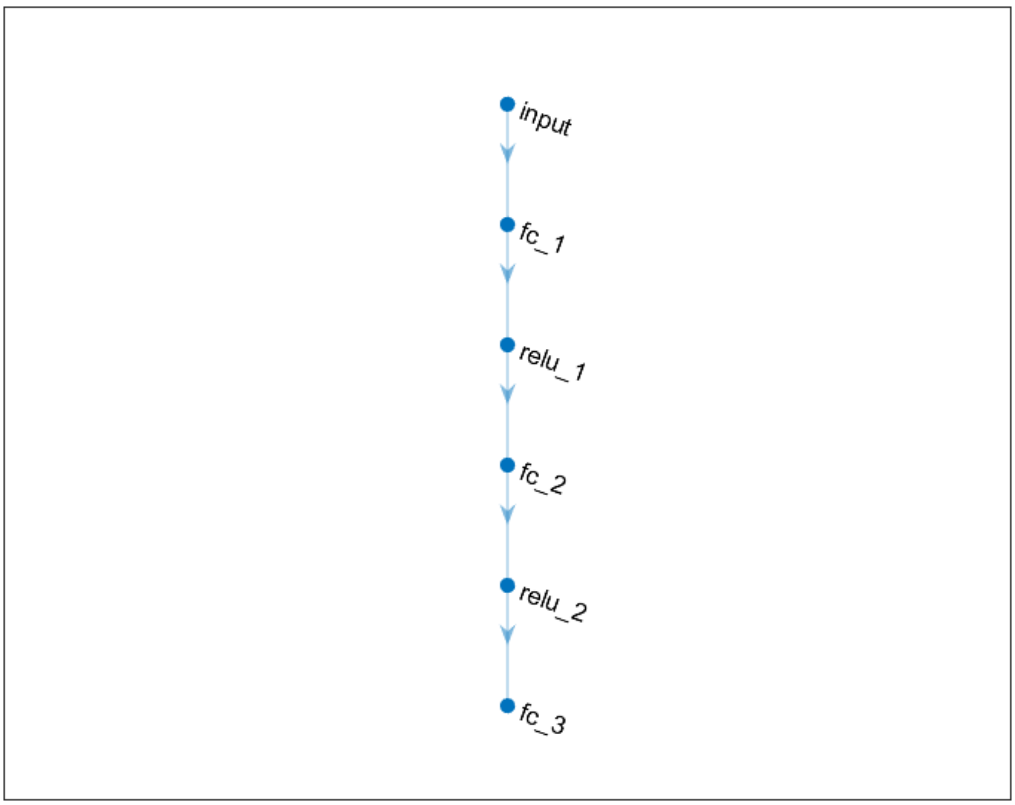

Рисунок 3.17 – Схема нейронной сети актора

Теперь перейдем к созданию DDPG агента и настройке гиперпараметорв (рисунок 3.18).

Создадим DDPG стратегию с помощью ранее определенных объектов актора и критика.

```
agent = rlDDPGAgent(actor,critic);
```

Настроим стратегию, актора и критика.

```
agent.SampleTime = Ts;

agent.AgentOptions.TargetSmoothFactor = 1e-3;
agent.AgentOptions.DiscountFactor = 0.9;
agent.AgentOptions.MiniBatchSize = 128;
agent.AgentOptions.ExperienceBufferLength = 1e6;

agent.AgentOptions.NoiseOptions.StandardDeviation = 80;

agent.AgentOptions.CriticOptimizerOptions.LearnRate = 1e-03;
agent.AgentOptions.CriticOptimizerOptions.GradientThreshold = 1;

agent.AgentOptions.ActorOptimizerOptions.LearnRate = 1e-04;
agent.AgentOptions.ActorOptimizerOptions.GradientThreshold = 1;
```

Рисунок 3.18 – Листинг кода в программе MATLAB для настройки агента DDPG

Важно отметить, что эти параметры тоже сильно влияют на процесс и результат обучения. Коэффициент скорости обучения для критика должен быть выше, для более быстрой его сходимости. Так же стандартное отклонение у шума нужно установить в пределах 1–5% от рабочего диапазона регулятора, для корректного исследования среды.

Агент и среда настроены, можно запустить обучение с помощью кода MATLAB (рисунок 3.19).

```
trainOpts = rlTrainingOptions(...
    MaxEpisodes=5000, ...
    MaxStepsPerEpisode=ceil(Tf/Ts), ...
    ScoreAveragingWindowLength=20, ...
    Verbose=false, ...
    Plots="training-progress",...
    StopTrainingCriteria="AverageReward",...
    StopTrainingValue=800);
trainingStats = train(agent,env,trainOpts);
```

Рисунок 3.19 – Листинг кода в программе MATLAB для обучения с подкреплением

Получим следующий график обучения нашей модели (рисунок 3.20).

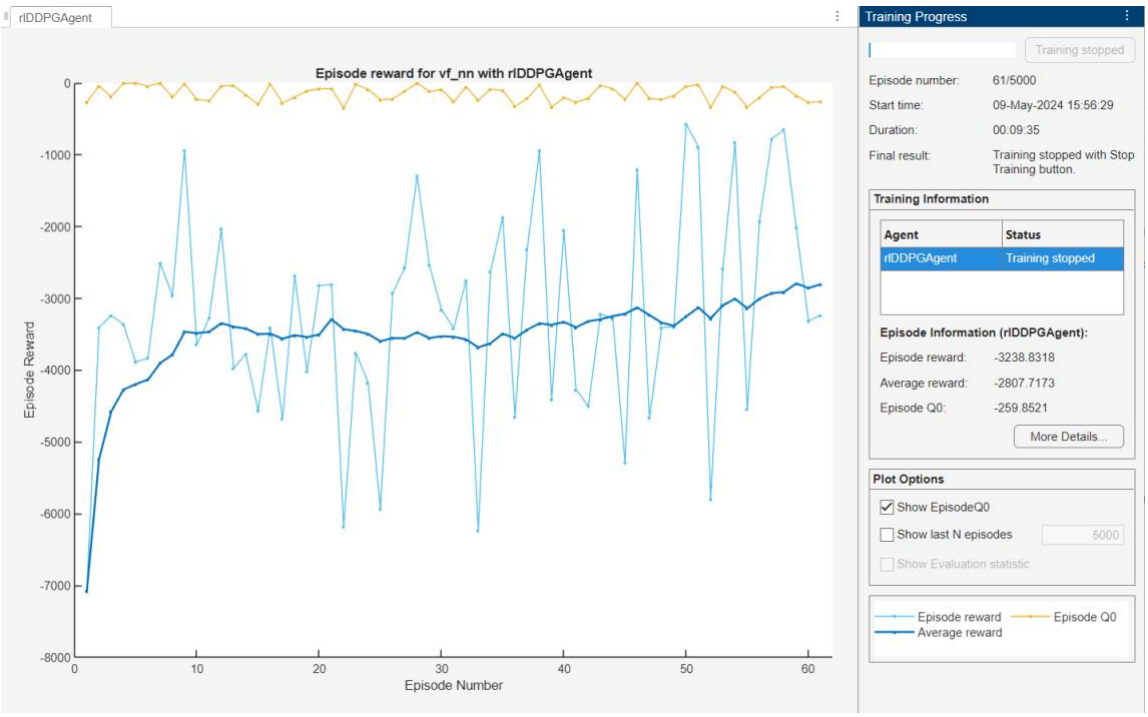

Рисунок 3.20 – График обучения с подкреплением модели DDPG

Полученную модель сохраним для дальнейшего использования и моделирования. С помощью обученной модели получим следующий график переходного процесса (рисунок 3.21).

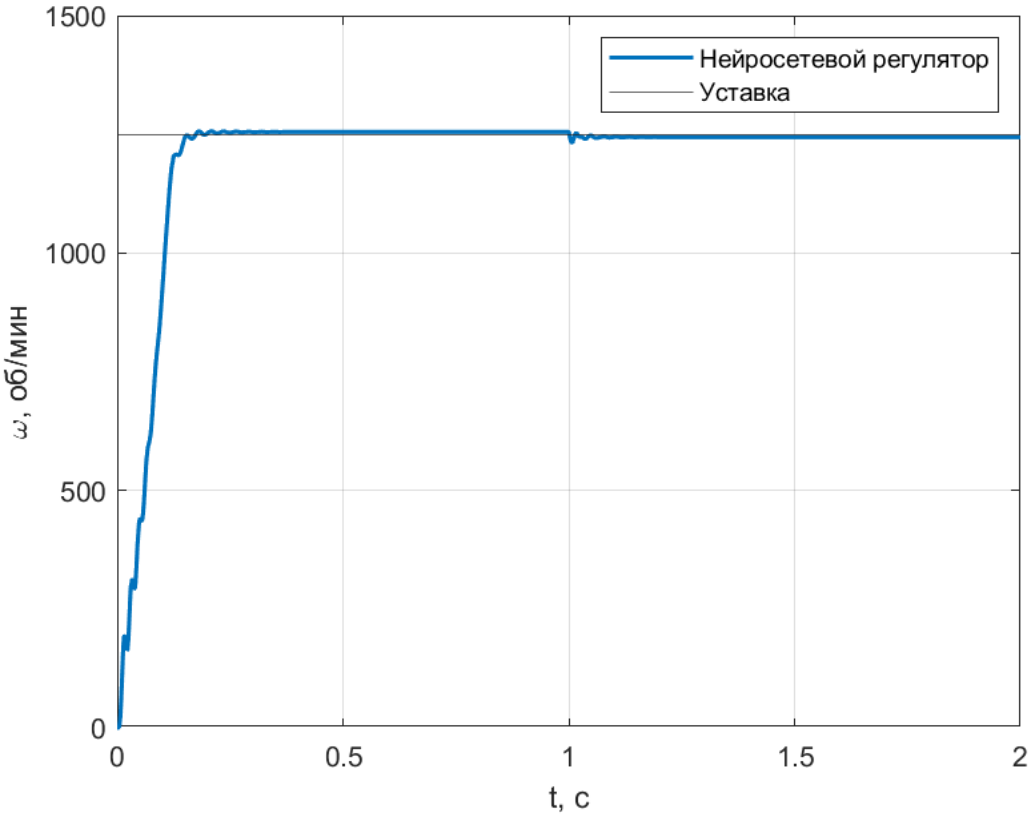

Рисунок 3.21 – График моделирования скорости асинхронного двигателя замкнутого нейросетевым регулятором

Как видно на рисунке 3.21, регулятор справляется с поставленной задачей, но имеет небольшую установившуюся ошибку при появлении механической нагрузки на валу электродвигателя.

3.5 Выводы по главе 3

Построим график, где случайно будем задавать уставку и менять ее через каждые две секунды. Моделирование будем вести 8 секунд, а на 5-й секунде добавим механическую нагрузку 1 Н · м. Получим следующий график (рисунок 3.22).

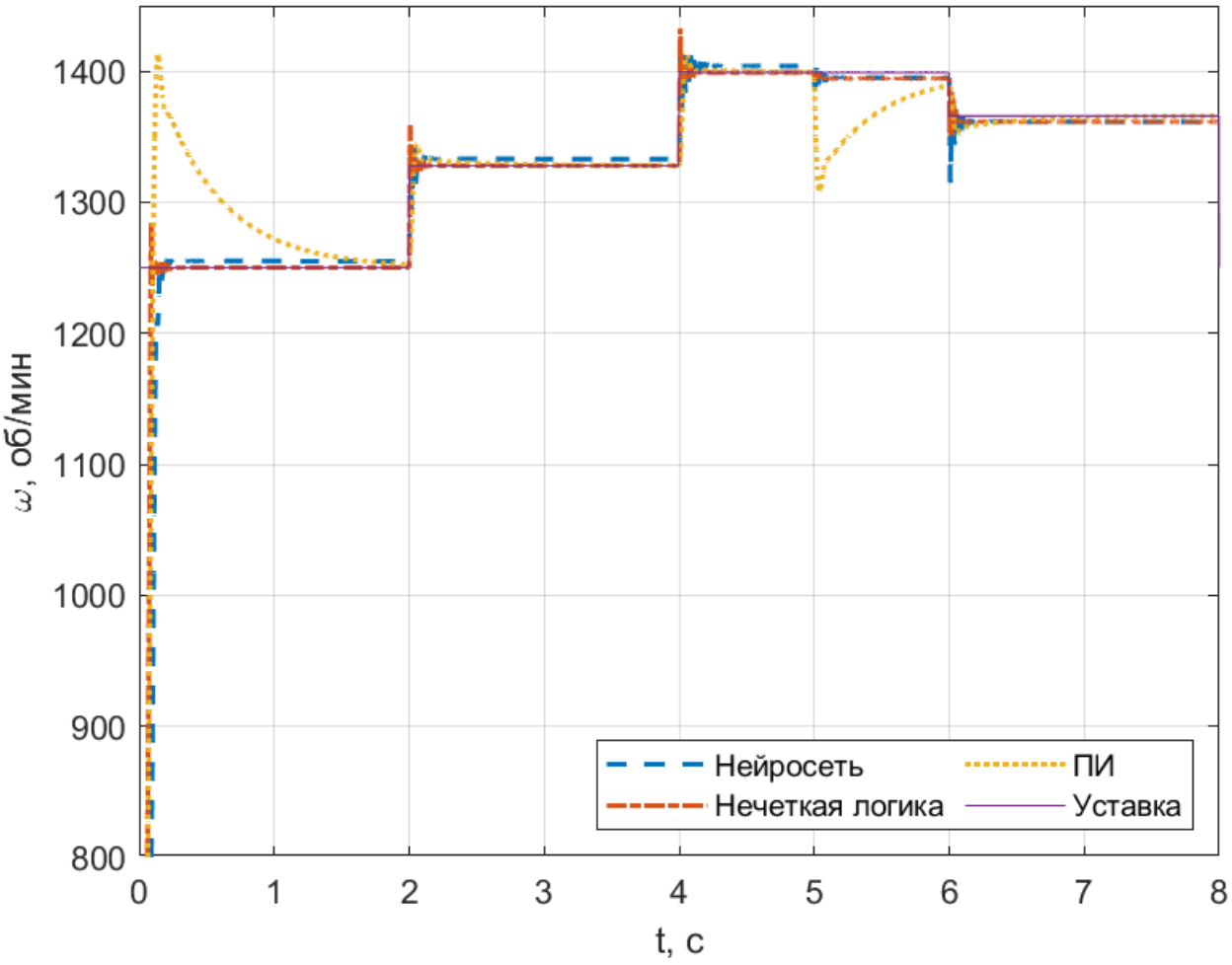

Рисунок 3.22 – График моделирования скорости асинхронного двигателя замкнутого разными регуляторами

Как видно по графику на рисунке 3.22, самое большое перерегулирование у ПИ-регулятора. Но у нейросетевого и регулятора на

нечеткой логике есть небольшая установившаяся ошибка, которая уходит со временем у ПИ-регулятора.

Теперь добавим в датчик измерения скорости белый шум с амплитудой по модулю 10 и шагом обновления 0.1 секунды (рисунок 3.23).

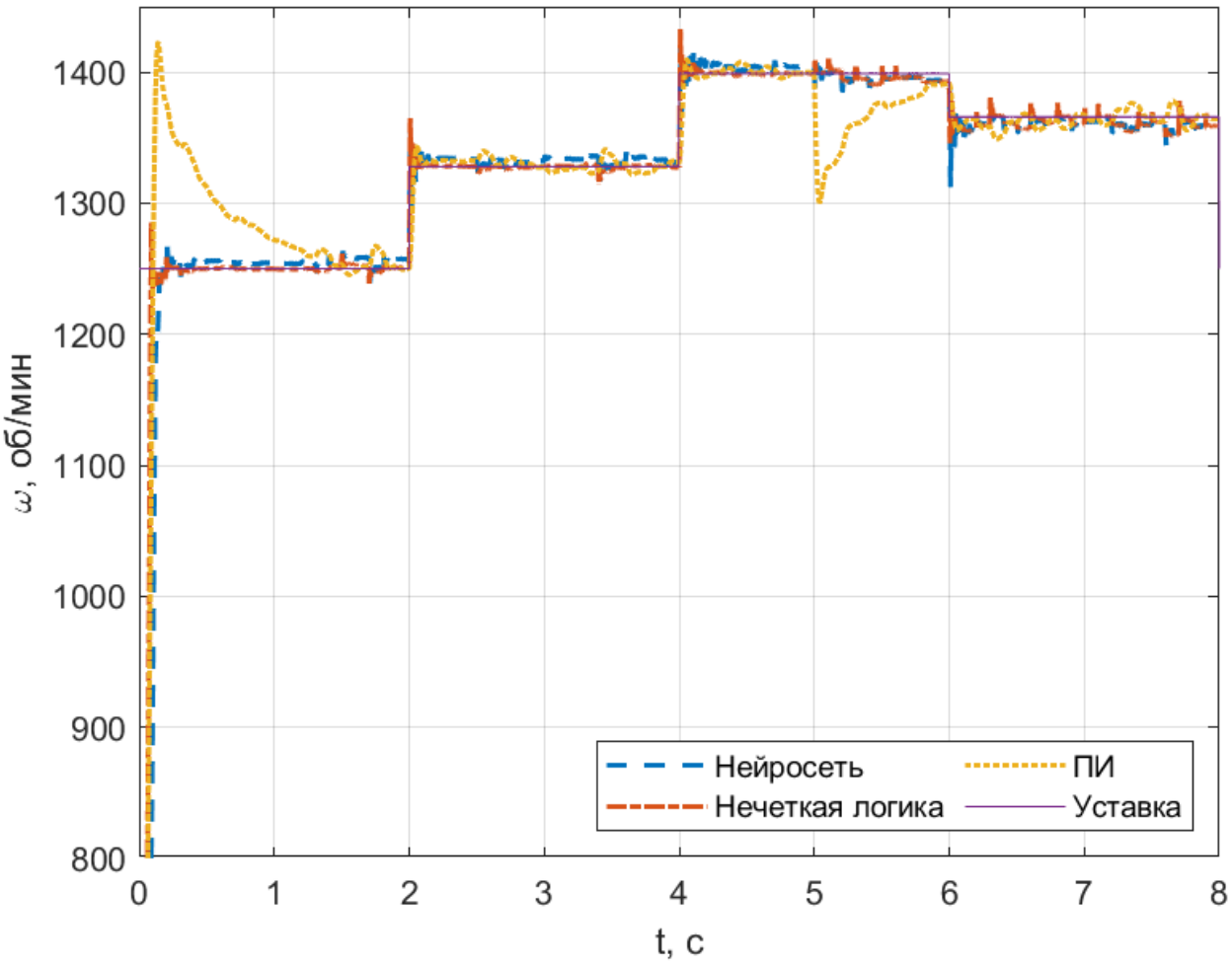

Рисунок 3.23 – График моделирования скорости асинхронного двигателя замкнутого разными регуляторами с шумом

Как видно по рисунку 3.23, добавления шума в датчик не сильно влияет на поведение регуляторов, у них лишь появляется небольшая ошибка.

Теперь попробуем в качестве уставки задать синусоидальный сигнал, время моделирования зададим 10 секунд и механическую нагрузку оставим прежней (рисунок 3.24).

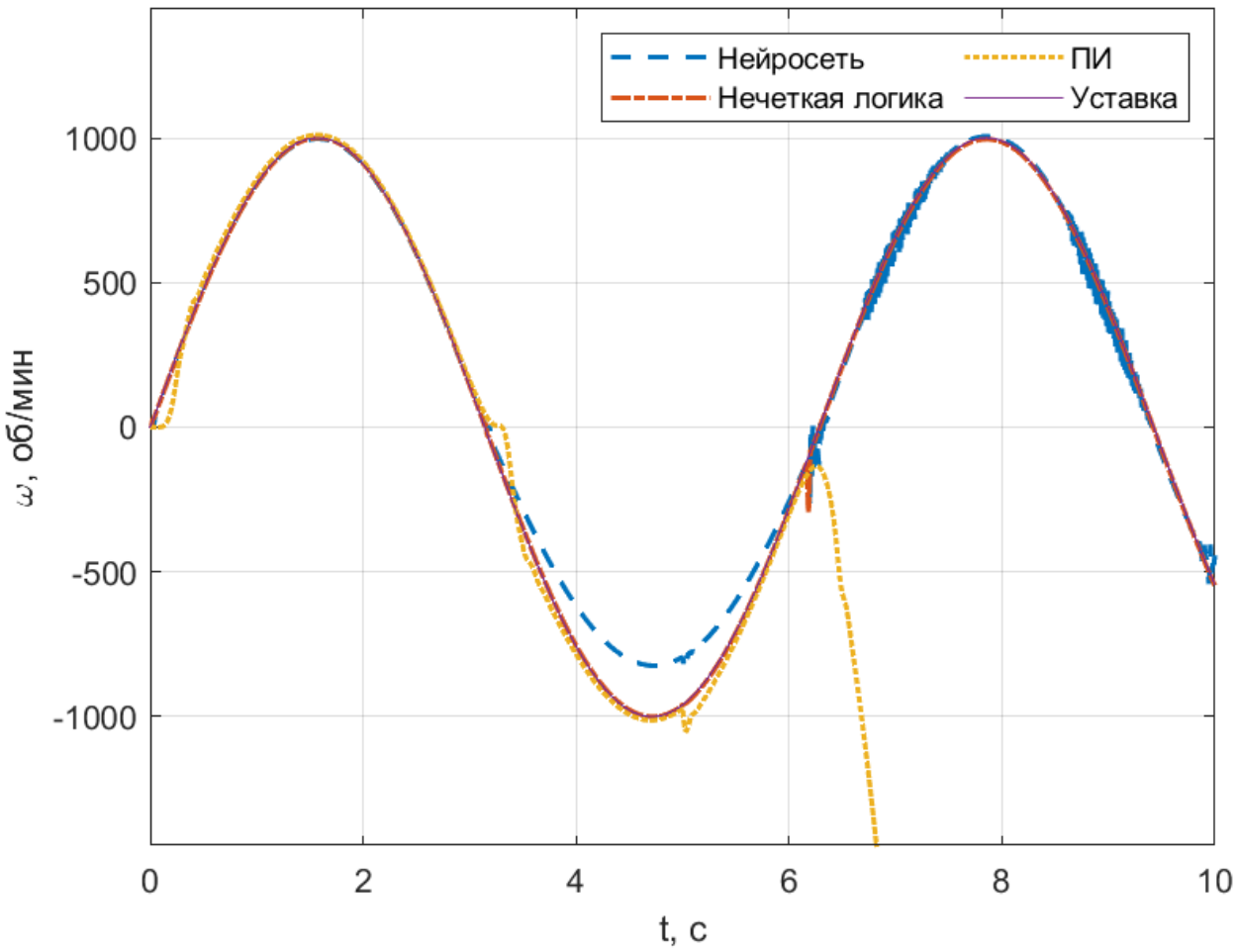

Рисунок 3.24 – График моделирования скорости асинхронного двигателя замкнутого разными регуляторами

Как видно на рисунке 3.24, ПИ-регулятор на 6-ой секунде потерял устойчивость, а на 5-ой секунде нейросетевой регулятор имел большую ошибку. Стоит отметить, что обучение нейросетевого регулятора происходило на линейной уставке. Зато регулятор на нечеткой логике оказался универсальным и имеет малую ошибку регулирования до 20 об/мин.

Наконец подробно рассмотрим характеристики переходного процесса, выбрав уставку скорости 1250 оборотов в минуту, время моделирования 2 секунды, двигатель разгоняется с нулевой скорости.

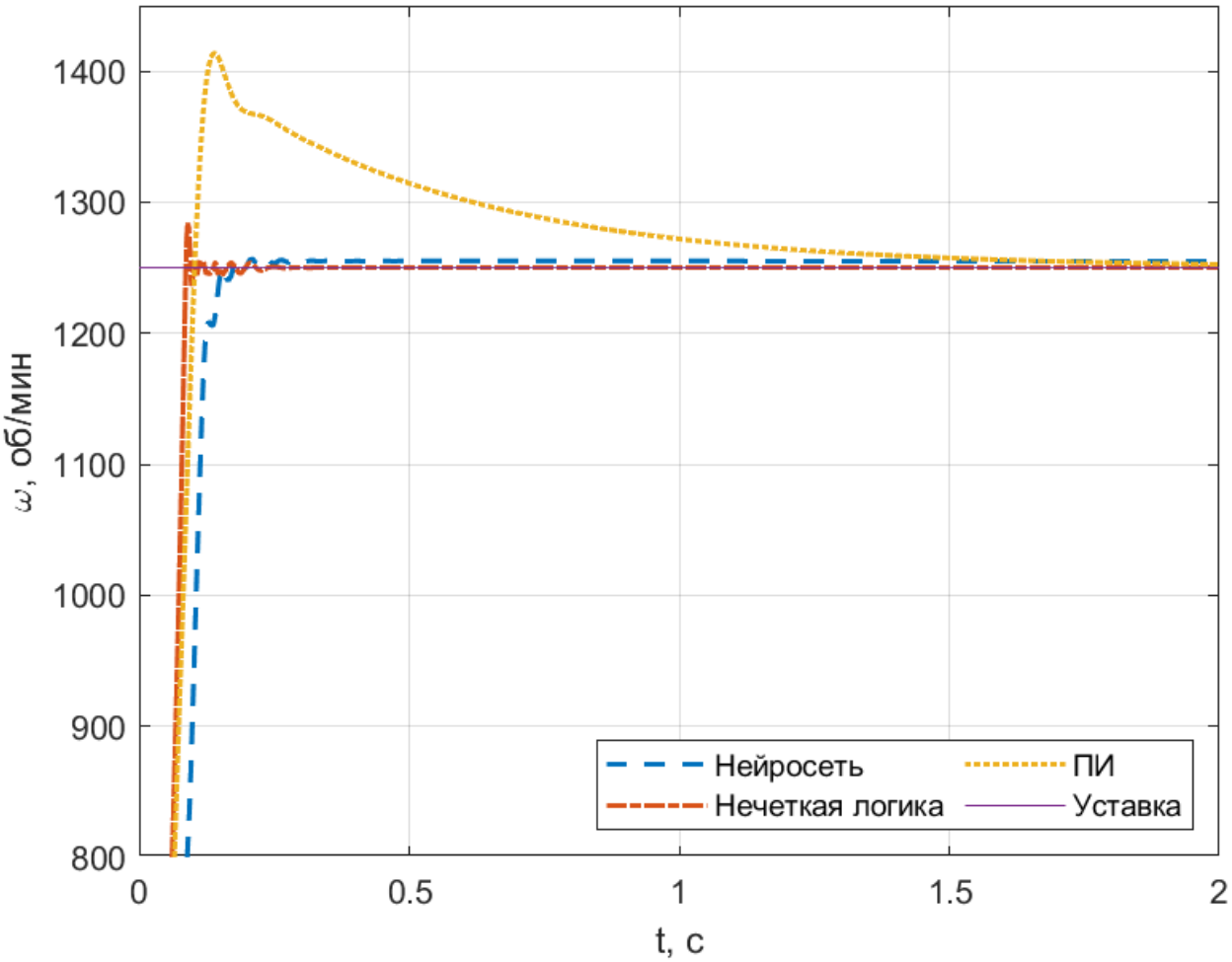

Рисунок 3.25 – График моделирования скорости асинхронного двигателя замкнутого разными регуляторами

По рисунку 3.25 составим таблицу характеристик каждого из регуляторов:

Таблица 3 – Характеристика синтезированных регуляторов

Характеристика Регулятор	Перерегулирование, об/мин	Установившаяся ошибка, об/мин	Время переходного процесса, с
ПИ-регулятор	163	1	2
Нечеткая логика	32	2	0.2
Нейросеть	8	5	0.25

На основе полученных характеристик из таблицы 3 можно сделать вывод о том, что синтезированные алгоритмы управления не уступают классическому ПИ-регулятору.

Так же рассмотрим случай, когда двигатель работает в постоянном режиме и на валу появляется механическая нагрузка $1\ Н \cdot м$ в момент 1 секунды (рисунок 3.26).

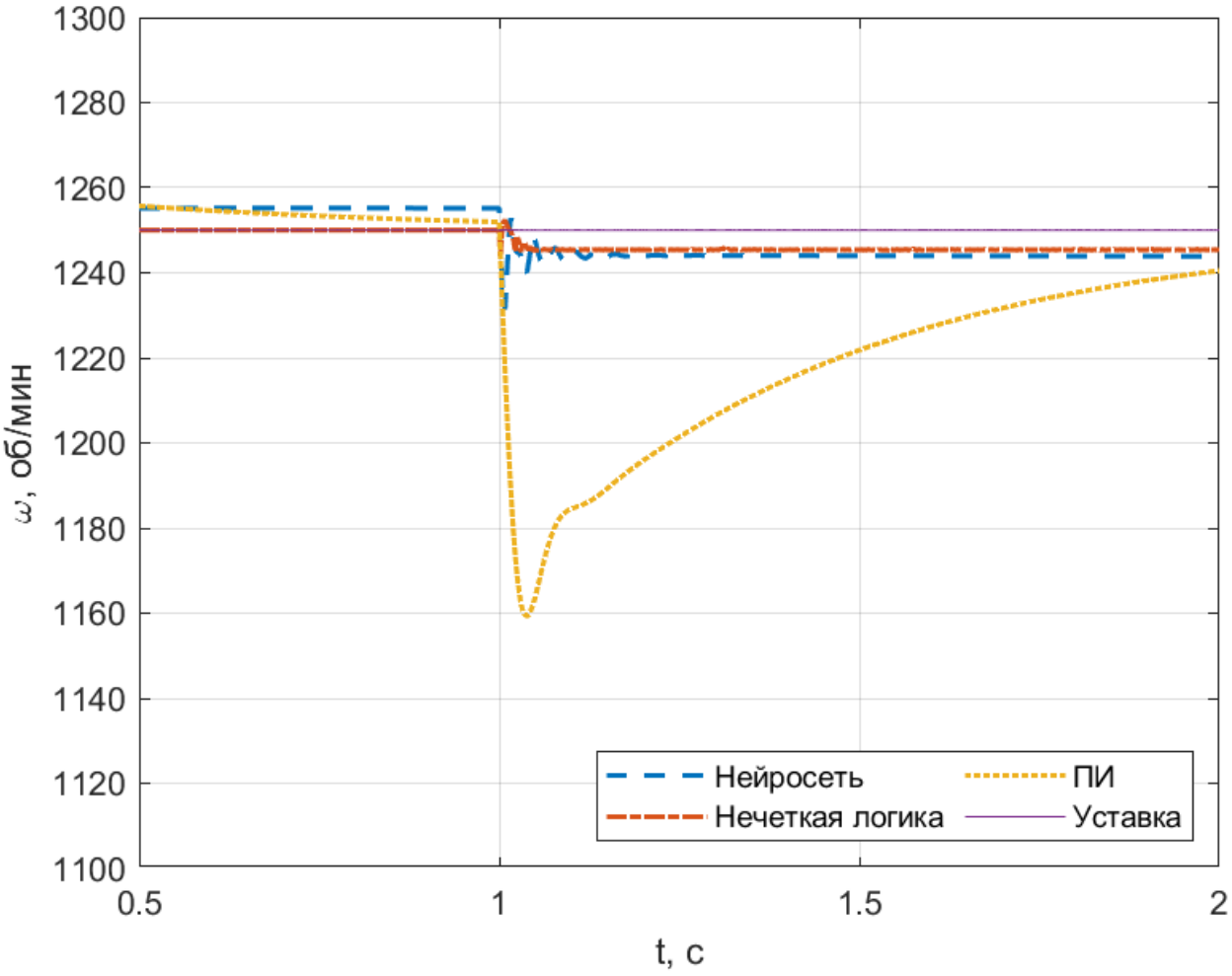

Рисунок 3.26 – График моделирования скорости асинхронного двигателя замкнутого разными регуляторами

Как видно на рисунке 3.26 самая большая потеря в скорости была у ПИ-регулятора, порядка 70 об/мин, у нейросетевого 15 об/мин и у нечеткой логики 6 об/мин. У нечеткой логики появилась установившаяся ошибка порядка 4 об/мин, у ПИ-регулятора она постепенно сходится к 0, а у нейросетевого регулятора установившаяся ошибка составила 6 об/мин.

ЗАКЛЮЧЕНИЕ

Была описана математическая система и координатные преобразования, позволяющие моделировать поведение АД. Затем создана модель АД в программе Simulink для моделирования. Для управления АД был выбран скалярный метод управления, так как он широко распространен и прост в реализации. Так же скалярный метод управления был подробно описан и реализован в схеме Simulink с ПД-регулятором.

Затем были рассмотрены и математически описаны различные методы интеллектуального управления, а именно нечеткая логика и нейронные сети. После рассмотрения этих методов они были реализованы и промоделированы в различных сценариях.

Были получены следующие результаты: графики моделирования переходного процесса в разных сценариях и сводная таблица характеристик синтезированных регуляторов, из которой можно сделать вывод, что полученные регуляторы оказались лучше общепринятого ПИ-регулятора.

В рамках дальнейшей работы над рассматриваемыми алгоритмами стоит провести эксперимент на реальной системе, а также применить интеллектуальные методы с векторным способом управления асинхронным двигателем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Andrzej M. Trzynadlowski Control of Induction Motors. - 1 изд. - San Diego: Academic press, 2001. - 225 с.
2. Калачев Ю.Н. «SimInTech»: моделирование в электроприводе. - 1 изд. - Москва: ДМК Пресс, 2019. - 105 с.
3. Tze-Fun Chan, Keli Shi Applied Intelligent Control of Induction Motor Drives. - 1 изд. - Singapore: John Wiley & Sons (Asia) Pte Ltd, 2011. - 417 с.
4. Г.Л. Демидова, Д.В. Лукичев РЕГУЛЯТОРЫ НА ОСНОВЕ НЕЧЕТКОЙ ЛОГИКИ В СИСТЕМАХ УПРАВЛЕНИЯ ТЕХНИЧЕСКИМИ ОБЪЕКТАМИ. - 1 изд. - Санкт-Петербург: Университет ИТМО, 2017. - 78 с.
5. Lark Topics: AI Glossary // Lark URL: https://www.larksuite.com/en_us/topics/ai-glossary (дата обращения: 10.04.2024).
6. Fuzzy Logic Toolbox // mathworks URL: https://www.mathworks.com/help/fuzzy/ (дата обращения: 15.04.2024).
7. Reinforcement Learning // mathworks URL: https://www.mathworks.com/help/deeplearning/reinforcement-learning.html (дата обращения: 15.04.2024).