Санкт-Петербургский Государственный Университет

Фундаметальная информатика и информационные технологии
Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Дерюгин Денис Евгеньевич

# Применение мультиагентных технологий для управления группой роботизированных устройств

Выпускная квалификационная работа

Научный руководитель:
д. ф.-м. н., профессор Граничин О. Н.

Рецензент:
к. ф.-м. н., научный сотрудник ИПМаш РАН Иванский Ю. В.

Санкт-Петербург
2017

Saint Petersburg State University

Fundamental Informatics and Information Technology
Mathematical and Software Support for Computers, Computer Systems and Networks

Denis Deryugin

# Applying multi-agent technologies to control a group of robotic devices

Graduation Thesis

Scientific supervisor:
professor Oleg Granichin

Reviewer:
Ph. D. Yury Ivanskiy

Saint Petersburg
2017

# Оглавление

# Введение

В настоящее время исполнительные механизмы становятся всё более точными, а вычислительные устройства приобретают всё меньший размер. Это открывает новые возможности в интеллектуальном управлении сложными системами в тех случаях, когда стандартные математические модели в силу тех или иных причин оказываются неприменимыми.

Одним из ключевых недостатков традиционных методов анализа и управления системами является предположение о том, что модель описания окружающей среды является заранее известной и точной. На практике часто приходится сталкиваться с различными сложностями. Например, в ряде случаев приходится иметь дело с приближёнными моделями (причина может быть в том, что точной модели просто не существует, либо в том, она неудобна для вычислений) и искажёнными входными данными (для большинства измерений характерно наличие помех и "выбросов"). Более того, существует возможность технических сбоев, частично или полностью выводящих из строя различные компоненты системы. К тому же, в ряде случаев может меняться и структура пространства состояний.

Ранее вопросы адаптивного управления в условиях неопределённости не получали достаточного внимания со стороны исследователей в связи с техническими ограничениями по реализации механических систем. Однако, современная техническая база позволяет применять новые математические модели для решения подобных проблем. В частности, стал возможен сбор более детальной информации об окружающей среде — различные датчики становятся не только более точными, но и более компактными.

Как было показано в [5, 7, 18, 19], ряд природных явлений стоит рассматривать как процессы с переменной структурой пространства состояний. Изменение структуры пространства состояний приводит к изменению внутренней структуры системы, а взаимодействие элементов системы, в свою очередь, приводит к изменению системы в целом [30].

При рассмотрении турбулентности в жидкостях, газах, мультифазовых и пластических потоках окружение может рассматриваться в качестве подобной системы[8, 10]. Примером таких систем может служить практически любая биологическая система. О динамическом формировании структур можно говорить и в рамках социологии, психологии и экономики [2].

Из-за изменения числа степеней свободы даже фиксированный набор переменных для построения математической модели неравновесного процесса, как строго доказано в неравновесной статистической механике [27], никогда не будет полным. Следовательно, неравновесные системы в природе не могут быть описаны в полной мере с помощью традиционных дифференциальных моделей для динамических систем. Для описания таких переходных процессов следует использовать более гибкие математические модели. Эти модели должны уметь приспосабливаться к изменению внешней среды, например, с помощью механизма внутренней обратной связи [6].

Мультиагентные системы могут быть использованы для эффективного решения широкого спектра проблем, связанных с возмущениями и нестационарными системами [15]. Эффективность решения достигается заменой большой и сложной модели набором простых локальных моделей. Согласованное поведение некоторой подгруппы агентов приводит к уменьшению размерности пространства состояний [14, 26]. Возмущения внешней среды могут приводить к нарушению согласованности в поведении некоторых групп агентов, и это будет соответствовать увеличению размерности пространства состояний.

В последнее десятилетие проблемы взаимодействия в распределённых системах управления всё больше привлекает внимание исследователей [3, 9, 12, 13]. Этот интерес связан с растущим количеством прикладных областей, связанных с управлением в распределённых электрических сетях, межпроцессорным взаимодействием, беспроводными, транспортными и промышленными сетями, сетями датчиков, БПЛА, координацией мобильных роботов и так далее.

В [6] предлагалось использовать мультиагентную систему для ре-

шения проблем, связанных с движением в турбулентном потоке. Эта задача не имеет аналитического решения, и в настоящее время используются различные упрощённые модели, а значит, использование мультиагентного подхода может оказаться эффективным.

# 1. Постановка задачи

Целью работы является исследование применимости мультиагентных технологий для управления агентами в условиях неопределённости. Для достижения этой цели были сформулированы задачи, представленные ниже.

1. Разработка алгоритма управления агентами в условиях неопределённости с возможностью собирать, обрабатывать и визуализировать данные о поведении мультиагентной системы.

2. Моделирование работы алгоритма управления для упрощённой математической модели окружающей среды.

3. Разработка стенда для проверки применимости алгоритма управления в физических экспериментах.

4. Исследование влияния различных параметров на поведение алгоритма управления.

# 2. Обзор

## 2.1. Мультиагентные технологии

На практике часто возникают задачи, при которых требуется параллельно выполнять множество схожих заданий. Это справедливо как для вычислительных сетей (от межпроцессорного взаимодействия до распределённых вычислений), так и для промышленных, транспортных и прочих систем, где присутствуют естественные ограничения на задержки при коммуникации и пропускную способность каналов связи. Это выводит на первый план вопросы распределения нагрузки и децентрализованного управления, где классические планирование и управление становятся неэффективными.

В основе мультиагентного подхода лежит понятие *агента*, который представляет из себя некоторую сущность, способную собирать данные об окружающей среде и самостоятельно принимать какие-либо решения. Группа взаимодействующих агентов, имеющих общую цель, называется *мультиагентной системой (МАС)*. Наибольший интерес вызывает рассмотрение МАС, состоящих из *интеллектуальных агентов*, которые обладают следующими дополнительными свойствами.

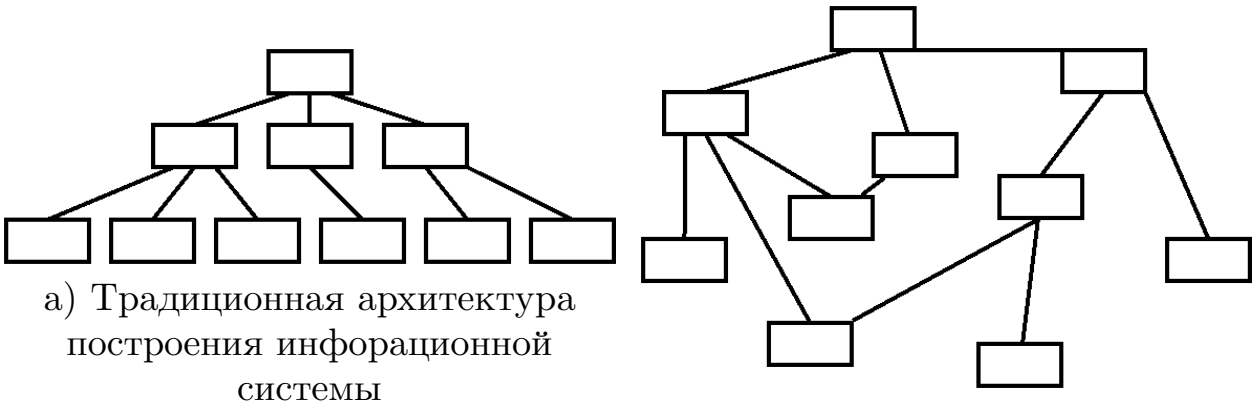
1. *Социальность* — интеллектуальный агент способен взаимодействовать с другими агентами, делая достижение цели более эффективным. Следует отметить, что социальность подразумевает не только обмен данными, но и способность к локальной самоорганизации.

2. *Реактивность* — изменения во внешней среде должны приводить к реакции агента на эти взаимодействия.

3. *Проактивность* — агент должен не только реагировать на воздействия среды, но и брать на себя инициативу.

Перечень этих свойств может быть расширен, но перечисленные три требования к интеллектуальным агентам являются ключевыми.

В роли таких интелектуальных агентов могут выступать, к примеру, сотрудники какого-либо предприятия, но если речь идёт о применении мультиагентных технологий в информационных системах, под агентом понимается *программный агент*, который представляет из себя некоторый фрагмент кода, выполняющий соотвествующие действия для взаимодействия с окружающей средой. Программные агенты могут исполняться как на различных физических узлах, так и на одном вычислительном устройстве в виртуальном окружении.



а) Традиционная архитектура построения инфорационной системы

б) Мультиагентная архитектура

Рис. 1: Сравнение традиционной и мультиагентной архитектур.

Главная разница между традиционными информационными системами и МАС в том, что отсутствует строгая иерархия элементов системы (см. рисунок 1). Из этого вытекают следующие преимущества мультиагентного подхода.

- Децентрализованность и, как следствие, отказоустойчивость. Рассматриваются системы, в которых агенты могут присоединяться к системе и покидать её, также каналы связи могут появляться и исчезать. Основным преимуществом такой архитектуры является отказоустойчивость. В случае отказа главного узла в традиционной информационной системе произойдёт сбой всей системы, в то время как МАС сможет продолжить в работу при выходе из строя даже нескольких узлов.

- Эмерджентный интеллект (“интеллект роя”). При определённых

условиях даже простые агенты могут приводить к появлению сложных решений.

- Адаптивность. Изменения внешней среды приводят к изменению поведения системы.
- Автономность. В случае потери связи с остальной системой агент может продолжать действовать водиночку.

В области мультиагеных технологий до сих пор остаётся множество нерешённых задач. Публикации по теме MAC включают следующие области исследований.

- Кооперация и самоорганизация.
- Распределённые вычисления.
- Отказоустойчивость.
- Мультиагентное обучение.
- Синхронизация и достижение консенсуса.
- Управления группами роботов.

Так как именно последние два пункта из этого списка являются предметом этой работы, они будут рассмотрены более подробно.

## 2.2. Мультиагентные алгоритмы синхронизации

Когда речь идёт об управлении в мультиагентной системе, под синхронизацией понимается либо согласованное изменение, либо совпадение, либо приближение значений какого-либо параметра для некоторого числа агентов. Часто такая цель формулируется как достижение консенсуса или согласование некоторых характеристик.

На практике алгоритмы синхронизации могут использоваться в таких областях, как биология, социлогия, параллельные вычисления, системы энергоснабжения, системы датчиков и робототехника [4].

Топология взаимодействия агентов представляется с помощью ориентированного графа $G = (V, E)$, где $V = [1 : n]$ — это множество вершин, а $E \subset V \times V$ — множество рёбер, $N_k = \{i \in V : \exists (i, k) \in E, k \in V\}$ — множество соседей узла $k$.

В [12] было показано, что следующая простая линейная система приходит к консенсусу:

$$\dot{x}_k(t) = \sum_{i \in N_k} (x_i(t) - x_k(t)), x_k(0) = z_k \in \mathbb{R}. \quad (1)$$

Было доказано, что для связных графов существует единственная точка равновесия $x^* = (\alpha, ...\alpha)$ [11]. Более того, значение $\alpha$ зависит только от начальных значений $z_k$ и является их средним арифметическим, то есть, $\alpha = \frac{1}{n} \sum_{k \in V} z_k$. Может показаться, что нахождение среднего арифметического значения в связном графе является тривиальной задачей, но ситуация осложняется, если речь идёт о сетях с миллионами узлов и небольшими количеством соседей у каждого узла.

Так как класические вычислительные устройства работают не непрерывно, а с некоторой частотой, для практического использования алгоритма 1 в некоторой вычислительной системе необходимо провести дискретезацию (например, с шагом $h$), и тогда получится следующий закон управления:

$$x_k(t + 1) = x_k(t) + h \sum_{i \in N_k} (x_i(t) - x_k(t)), x_k(0) = z_k \in \mathbb{R}, h \in \mathbb{R}^+. \quad (2)$$

Ряд других особенностей практического применения ведёт к необходимости вносить в этот алгоритм другие изменения, например, анализ поведения системы при наличии центрированных помех и задержек связи.

## 2.3. О применении мультиагентных систем для управления группой роботизированных устройств

Роботизированные устройства представляют из себя один из наиболее интересных примеров возможного применения мультиагентных технологий, так как способны не только производить некоторые вычисления и обмениваться данными с другими агентами, но и с взаимодействовать с окружающим миром непосредственно.

Традиционно, сложные роботические устройства выпускались небольшими сериями и при этом имели весьма ограниченную автономность, так как автоматические алгоритмы управления и взаимодействия до сих пор в ряде случаев проигрывают схеме, при которой роботом управляет оператор, а само устройство выступает в роли инструмента, а не в качестве интеллектуального агента. Тем не менее, в ряде аспектов такой подход может быть попросту неприменим. Прежде всего, речь идёт о ситуациях, когда связь с центром управления обрывается, либо возникают слишком большие задержки (как, к примеру, при взаимодействии с удалёнными космическими аппаратами).

Более того, сегодня развитие технической базы привело к тому, что роботы могут обладать различными сенсорами (видеокамеры, акселерометры, датчики давления, датчики температуры и так далее), большим набором исполнительных механизмов, а также достаточно мощными вычислительными устройствами для сбора информации и управления механизмами.

И хотя сегодня речь не идёт о мультиагентных системах космических аппаратов, мультиагетные системы БПЛА уже находят своё применение. Например, мультиагентное управление группой беспилотников позволяет решать такие задачи, как организация распределённого RAID-массива в системе с ненадёжными каналами связи, позволяющего восстанавливать часть данных воздушной съёмки при выходе из строя некоторых устройств [31].

Есть два наиболее распространённых подхода к распределённому управлению группой роботов: алгоритмы, основанные на “рыночных от-

ношениях", и алгоритмы коллективного управления [17]. Первая группа алгоритмов позволяет распределять задания с помощью согласования условных "цен" за их выполнение, и в этом случае каждый робот пытается решить свою собственную задачу. Для второй группы алгоримтов характерна общая цель для всех роботов, которые согласуют свои действия между собой.

## 2.4. О реализации мультиагентных систем

Несмотря на разнообразие областей применения мультиагентных технологий, у сообщества разработчиков есть естественное стремление к стандартизации данной области. Существует ряд международных проектов по стандартизации мультиагетных систем. Наиболее известны среди них следующие.

- Спецификации FIPA (Foundations for Intelligent Physical Agenots).
- MASIF (Mobile Agent System Interoperability Facility) — стандарт, разрабатываемый консорциумом Object Management Group.
- Стандарты, разработанные DARPA (The Defense Advanced Research Projects Agency).

Данные спецификации и стандарты с разной степенью детализации определяют типы и способы именования агентов, протоколы коммуникации и структуру сообщений, методы миграции агентов между интеллектуальными системами и так далее. Также предлагаются специализированные языки программирования для описания поведения агентов (CLAIM, совместимый с MASIF; FIPA-ACL, совместимый с FIPA). Полные перечни спецификаций и стандартов доступны на сайтах соответствующих организаций [20, 22]

Из этой тройки проектов наибольшее распространым стал проект FIPA. Данные стандарты поддерживаются рядом платформ: ZEUS [25], Comtec Agent Platform [21], Jade [24], The April Agent Platform, FIPA-OS [23].

Одним из наиболее распространённых фреймворков для разработки мультиагентных систем является Java Agent Development Framwork [24] (сокр. JADE, не следует путать с Jade — препроцессором HTML). Как понятно из названия, данный фреймворк позволяет разрабатывать алгоритмы для взаимодействия интеллектуальных агентов на языке Java.

JADE состоит из трёх компонент:

- Окружение для работы агентов JADE.
- Библиотки классов для создания агентов при помощи наследования и переопределения их поведения.
- Средство графического мониторинга и управления платформой интеллектуальных агентов.

Сама платформа является распределённой, то есть, агенты могут исполняться на разных компьютерах и общаться по TCP/IP. Имеется полная совместимость со стандартами FIPA (Foundation for Intelligent Physical Agents) и поддержка специализированного язык разработки FIPA-ACL. Имеется достаточно много проектов, использующих JADE, а также ряд литературы, раскрывающей различные особенности работы с фреймворком [1]. Основной проблемой данного фреймворка является необходимость поддержки JVM, что делает его неприменимым на устройствах с ограниченными ресурсами (например, на платах Arduino и STM32).

Помимо упомянутых выше, есть ряд менее популярных средств разработки мультиагентных систем, не в полной мере поддерживающих перечисленные стандарты.

1. JaCaMo — фреймворк, состоящий из трёх модулей: Jason, использующийся для разработки автономных агентов, Cartago, нужный для работы с окружающей средой, Moise, служащий для организации мультиагентной системы.
2. JACK Intelligent Agents — ещё одно средство разработки мультиагентных систем. Оно также реалиовано на языке JAVA.

Несмотря на наличие всех этих инструментов, часто исследователи и разработчики создают собственные небольшие фреймворки для конкретных целей. Вероятно, это должно рано или поздно привести к переосмыслению того, что из себя должна представлять среда разработки ПО для мультиагентных систем.

# 3. Мультиагентный алгоритм управления группой роботизированных устройств

Прежде чем говорить об алгоритме управления агентами, необходимо определить, какая именно цель должна достигаться. Иными словами, необходимо ввести некоторую метрику, отражающую, насколько хорошим является то или иное состояние системы. Естественно предположить, что должна произойти синхронизация некоторого параметра, и, таким образом, определённый показатель для соседних агентов должен иметь близкое значение. Тогда функционал качества будет иметь следующий вид:

$$Q(X) = \sum_{k \in X} \sum_{i \in N_k} ||p_k - p_i||^2, \tag{3}$$

где $X$ — состояние системы, индекс $k \in [1..n]$ — номер агента, $N_k$ — множество соседей агента с номером $k$, $p_k$ — оцениваемый параметр.

Как уже упоминалось во введении, в [6] было предложено использовать алгоритм локального голосования для выравнивания давления воздуха на разные участки поверности.

Алгоритм локального голосования имеет следующий вид:

$$x_k^{t+1} = x_k^t + \gamma \sum_{i \in N_k} b_{ik}(x_i^t - x_k^t), \tag{4}$$

где индекс $t \in \mathbb{N}$ означает момент времени, индекс $k \in [1..n]$ — номер агента, $x_k \in \mathbb{R}$ — значение нагрузки у агента с номером $k$, $\gamma$ — некоторая константа, $N_k$ — множество соседей агента с номером $k$, а $b_{ij}$ — элементы некоторой стохастической матрицы, соответствующей топологии системы.

В результате применения алгоритма локального голосования значения нагрузки во всех узлах приходят к консенсусу при выполнении ряда условий (необходима связность графа, константа $\gamma$ должна выбираться достаточо малой в зависимости от матрицы смежности и т. п.). Есть строгое математическое доказательство работоспособности алго-

ритма, имеется оценка скорости сходимости алгоритма в зависимости от различных параметров[6, 16]. Таким образом, в качестве параметра $p_k$ в функционале качества (3) берётся давление на поверхность элемента $k$.

Несмотря на свою эффективность при решении задачи достижения консенсуса, протокол локального голосования имеет ряд сильных недостатков. Во-первых, неявно предполагается, что известна зависимость между управлением агента и его нагрузкой. Во-вторых, нет ограничений ни на состояние агента (в принципе, возможно наличие сколько угодно большой или даже отрицательной нагрузки на один узел), ни на управление (то есть, за один такт работы состояние может меняться на произвольную величину).

При работе с реальной физической средой, скорее всего, эти условия выполнены не будут. В случае с задачей распределения нагрузки эти ограничения накладываются неявно самой задачей: с достаточно малыми коэффициентами нагрузка узла никогда не станет отрицательной, а пропускная способность сети позволит передавать соответствующее количество данных между соседями. Также связь между управлением и состоянием может быть прямой в ряде приложений. Например, нагрузкой на узел может быть количество некоторых пакетов заданий, и тогда, передав набор из $m$ заданий другому узлу, мы уменьшим количество своих заданий на $m$ и увеличим количество заданий у другого узла на то же самое значение. К сожалению, для ряда физических явлений эти условия могут не выполняться даже при удачных коэффициентах.

Для устранения этих недостатков предполагается следующее. Во-первых, в случае с неочевидной связью между управлением и изменением воздействия внешней среды, имеет смысл рассматривать тот параметр, на который мы воздействуем непосредственно. В случае с давлением воздуха и изменением угла пластины, это будет, непосредственно, сам угол $\alpha$. Из-за технических ограничений, он имеет ограничение на минимальное и максимальное значения (для плоского крыла, в лучшем случае, это будет $[0; \pi]$). Обозначим эти значения как $\alpha_k^-$ и $\alpha_k^+$ соответственно. Отметим, что данный интервал может быть разным у

разных элементов из-за неплоской формы крыла либо из-за каких-либо технических ограничений.

Таким образом, если $u_k^t$ — это управление узла $k$ в момент времени $t$, значение угла наклона на следующем такте будет выглядеть следующем образом: $\alpha_k^{t+1} = max\{\alpha_k^-, min\{\alpha_k^+, \alpha_k^t + u_k^t\}\} := P_{[\alpha^-;\alpha^+]}(\alpha_k^t + u_k^t)$.

В работе [6] предлагается в качестве $p_k$ рассматривать вектор из вертикальной и горизонтальной проекций силы давения воздуха на пластины:

$$Q(X) = \frac{1}{2}\sum_{k=1}^{n}\sum_{i\in N^k} b_{ki}\left\|\begin{pmatrix}\Delta_1^i\\ \Delta_2^i\end{pmatrix} - \begin{pmatrix}\Delta_1^k\\ \Delta_2^k\end{pmatrix}\right\|^2. \tag{5}$$

где $\Delta_1^k = z^k - z_0^k = f^k\cos(\alpha^i) - f_0^k\cos(\alpha_0^k)$, $\Delta_2^k = x^k - x_0^k = f^k\sin(\alpha^i)\cos(\beta^k) - f_0^k\sin(\alpha_0^k)\cos(\beta_0^k)$, $f_k$ — сила давления воздуха на пластину, $\alpha_k$ — угол наклона пластины, $\beta_k$ — угол поворота пластины элемента с номером $k$. Также предлагается закон управления, для которого строго доказана сходимость функционала качества (5) к нулю.

Также в [6] предлагается следующий закон управления:

$$\alpha_m^i = Pr_{[\alpha_k^-,\alpha_k^+]}\left(\alpha_{m-1}^i + \gamma D_\alpha^i\right) \tag{6}$$

$$D_\alpha^i = \frac{\sum_{j\in N^i} b^{i,j}(\bar{\Delta}_m^j - \bar{\Delta}_m^i) - tg(\alpha_{m-1}^i)(\ln f_m^i - \ln f_{m-1}^i)}{f_{m-1}^i cos(\alpha_{m-1}^i)} \tag{7}$$

Для этого алгоритма доказана сходимость к нулю соответствующего функционала качества (5).

Недостатком данного алгоритма является тот факт, что он не делает различий между различными элементами. То есть, давление на центр поверхности крыла и на его край в равной степени влияет и на функционал качества, и на поведение элементов. В то же время ясно, что если речь идёт об устойчивости аппарата при полёте в воздушном потоке, из-за большего плеча элементы на краях крыла будут иметь большее значение. В соответствии с этим, есть смысл добавить в функционал качества длину плеча элемента, равного расстоянию от элемента до оси движения.

Если добавить в функционал качества (5) длину рычага и оставить

только одну проекцию (так как мы рассматриваем ситуацию, когда мы меняем только угол наклона, то есть $\beta \equiv 0$), получится следующий функционал качества:

$$Q(X) = \sum_{k \in X} \sum_{i \in N_k} ||m_k - m_i||^2, m_k = r_k \Delta_k cos\alpha_k. \tag{8}$$

Дополнив закон управления 6, получим следующий алгоритм:

$$\alpha_m^i = Pr_{[\alpha_k^-, \alpha_k^+]} \left( \alpha_{m-1}^i + \gamma D_\alpha^i \right) \tag{9}$$

$$D_\alpha^i = \frac{\sum_{j \in N^i} b^{i,j} (\bar{\Delta}_m^j - \bar{\Delta}_m^i) - tg(\alpha_{m-1}^i)(\ln f_m^i - \ln f_{m-1}^i)}{r^i f_{m-1}^i cos(\alpha_{m-1}^i)} \tag{10}$$

## 3.1. Моделирование работы алгоритма

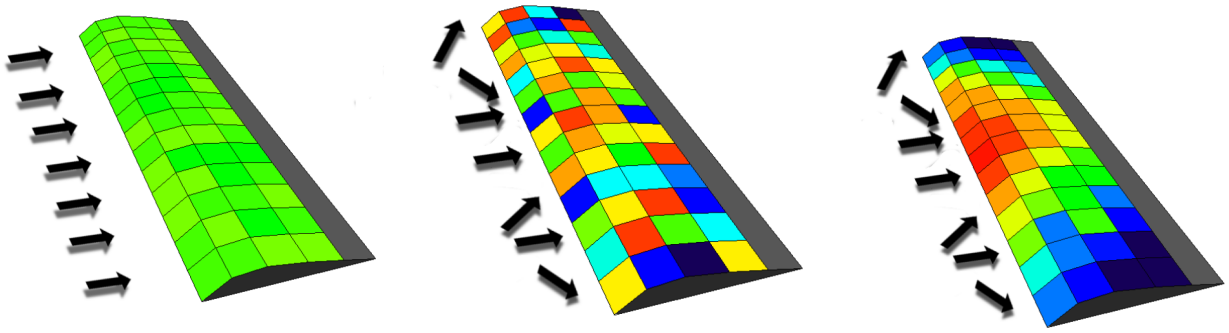
### 3.1.1. Упрощённая математическая модель турбулентого потока

Турбулентный поток действует не постоянно, а возникает в некоторый момент времени. Таким образом, давление воздуха на различные участки поверхности описывается следующим образом. До момента времени $t_1$ на поверхность воздействует ламинарный поток, т.к. $p_k = w_k$, где $w_k$ — центрированные помехи, соответствующие неточному измерению давления. В момент времени $t_1$ возникает возмущение, которое действует до момента времени $t_2$, после которого может либо вернуться ламинарный поток воздуха, либо появиться турбулентный поток с новыми параметрами (Рис. 2).

Можно предположить, что турбулентный поток формирует конечное число возмущений, каждое из которых может быть представлено как нормально распределённое с некоторыми коэффициентами.

Таким образом, давление в точке $x$ в момент времени $t$ может быть выражено как

$$\Delta(x,t) = \sum_{i=1}^{I} \Delta^i(t) \exp\{-\frac{1}{2}(x - r^i(t))^{\mathrm{T}} D^i(t)^{-1} (x - r^i(t))\} \tag{11}$$



а) Движение в ламинарном потоке ($t < t_0$).

б) Воздействие турбулентного потока ($t = t_0$).

в) Балансирвка нагрузки ($t_0 < t < t_1$).

Рис. 2: Математическая модель воздействия турбулентного потока на поверхность крыла.

$$P_{laminar} = (const, 0, 0) + w \tag{12}$$

$$P_{turbulent}(x, t) = P_{laminar} + \Delta(x, t) \tag{13}$$

### 3.1.2. Программная реализация фреймворка для работы с МАС

Для работы с данной математической моделью и взаимодействия со стендом для проведения экспериментов было необходимо разработать соответствующий фреймворк, позволяющий обрабатывать данные о работе системы, строить графики и визуализировать состояние системы в реальном времени. Также нужно было реализовать модуль, моделирующий окружающую среду и её воздействие на МАС.

Был разработан соответствующий фреймворк (см. рисунок 3) на языке Python, состоящий из двух основных модулей.

- *Симулятор* моделирует поведение внешней окружающей среды и агентов мультиагентной системы, при этом можно добавлять новые модели окружающей среды и новые алгоритмы управления, а также задавать топологию МАС. В рамках данной работы были реализованы две модели окружающей системы.

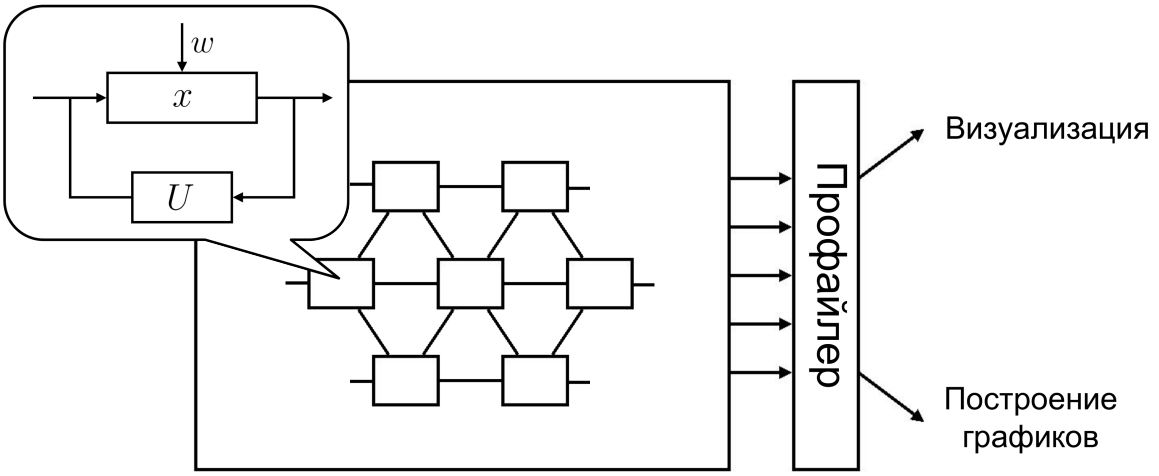



Рис. 3: Фрейморк для работы с мультиагентными системами.

- Модель случайных воздействий в каждой точке пространства.
- Упрощённая модель турбулентного потока (13).

Также были реализованы дискретные модификации следующих алгоритмов управления.

- Протокол локального голосования.
- Алгоритм для выравнивания давления, предложенный в [6].
- Модификация предыдущего алгоритма, учитывающая длину плеча для каждого элемента (9).

Моделирование происходит итеративно, шаг за шагом обновляя данные об окружающем мире, затем вычисляя воздействие воздушных потоков на систему и осуществляя управление элементами системы.

- *Профайлер* представляет из себя демон, исполняющийся на некотором хосте, позволяющий получать данные либо из симулятора,

либо из стенда для проведения экспериментов. Профайлер может передавать информацию о поведении системы в реальном времени на несколько разных хостов. Также возможна запись этих данных для дальнейшего построения графиков и так далее.

Исходные тексты всех программ доступны в открытом репозитории на GitHub [28].

Из соображений кросс-платформенности было решено использовать веб-приложения для визуализации состояния системы. На языке JavaScript было реализовано приложение, обменивающееся данными с python-сервером через WebSocket. Отображение производится при помощи библиотеки ThreeJS, являющимся обёрткой над WebGL. Цветом выделяется сила давления: красный — сильное давление, зелёный — среднее, синий — низкое (см. рисунок 4).

Для того, чтобы "развернуть" сервер, нужно запустить следующую команду:

```
./websocketd --port=9876 ./server.py
```

После этого можно открыть соответствующую страницу с веб-приложением для отображения информации о работе системы, информация о состоянии системы будет обвновляться в реальном времени.

## 3.2. Тестовый стенд

Так как моделирование показало применимость алгоритма балансировки нагрузки на поверхность поверхности крыла, необходимо опробовать этот алгоритм на практике. Для этого, прежде всего, необходимо было выбрать элементную базу будущего стенда.

В качестве аппаратной платформы была выбрана плата STM32 F3 Discovery (рис. 5).

Использование прямой "прошивки" (без операционной системы) не желательно, так как в этом случае необходимо реализовать ряд дополнительного функционала (работа с прерываниями и таймерами, некотоые системные вызовы), поэтому было решено использовать одну из

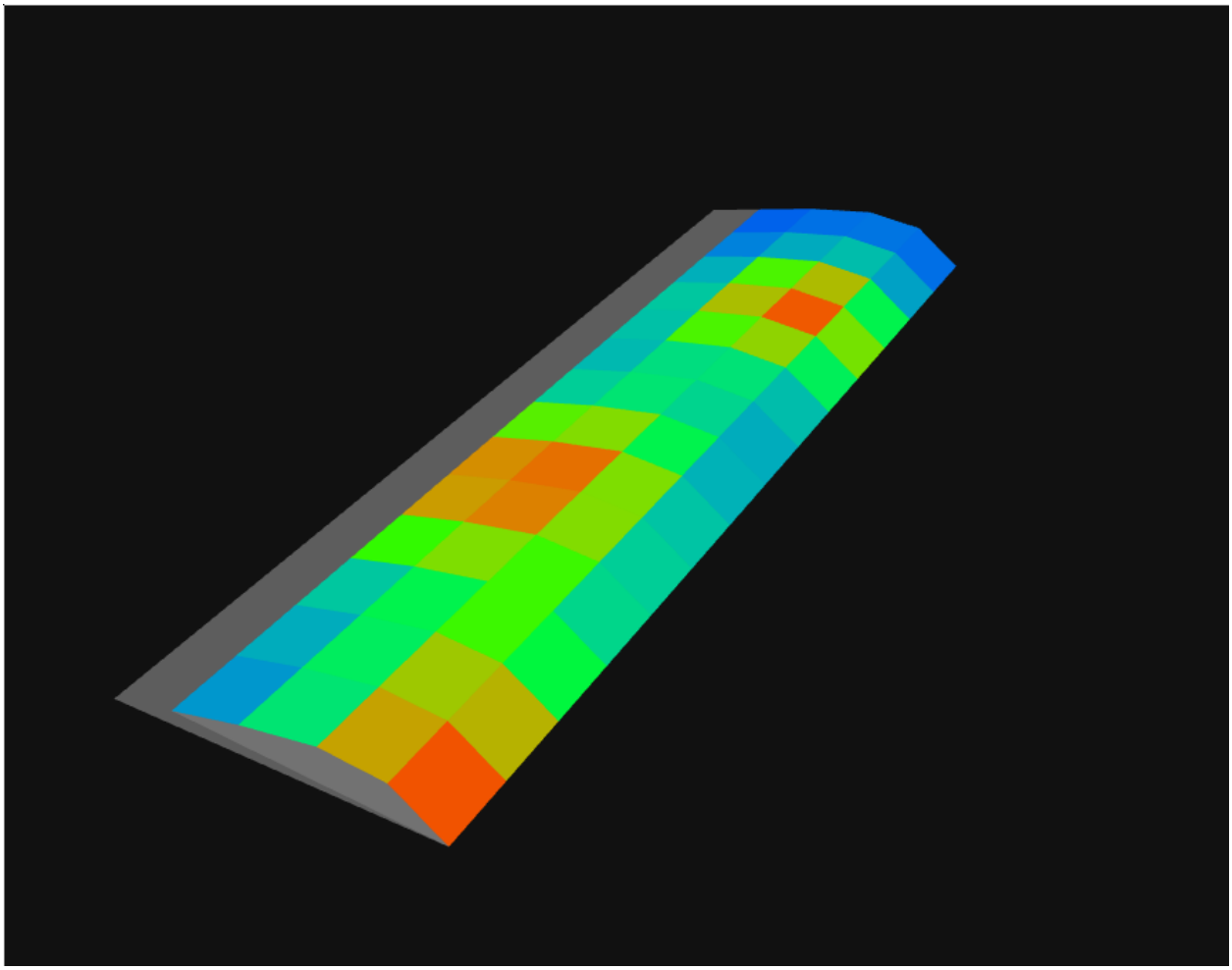


Рис. 4: Скриншот веб-приложения для визуализации состояния системы.

встроенных операционных систем с открытым исходным кодом. В качестве такой системы была выбрана ОС Embox [29]. Эта операционная система разрабатывается при поддержке кафедры системного программирования, и на момент разработки тестового стенда у меня уже был опыт работы с этой ОС и имелась возможность коммуникации с разработчиками этого проекта.

Для ОС Embox на языке C были реализованы драйвер управления сервоприводами (с использованием ШИМ), драйвер для работы с несколькими UART-портами для коммуникации между устройствами и драйвер сбора информации с датчиков давления (с помощью АЦП). Была также реализована библиотека, позволяющая собирать данные с соседних устройств, и приложение, которое при помощи данной биб-

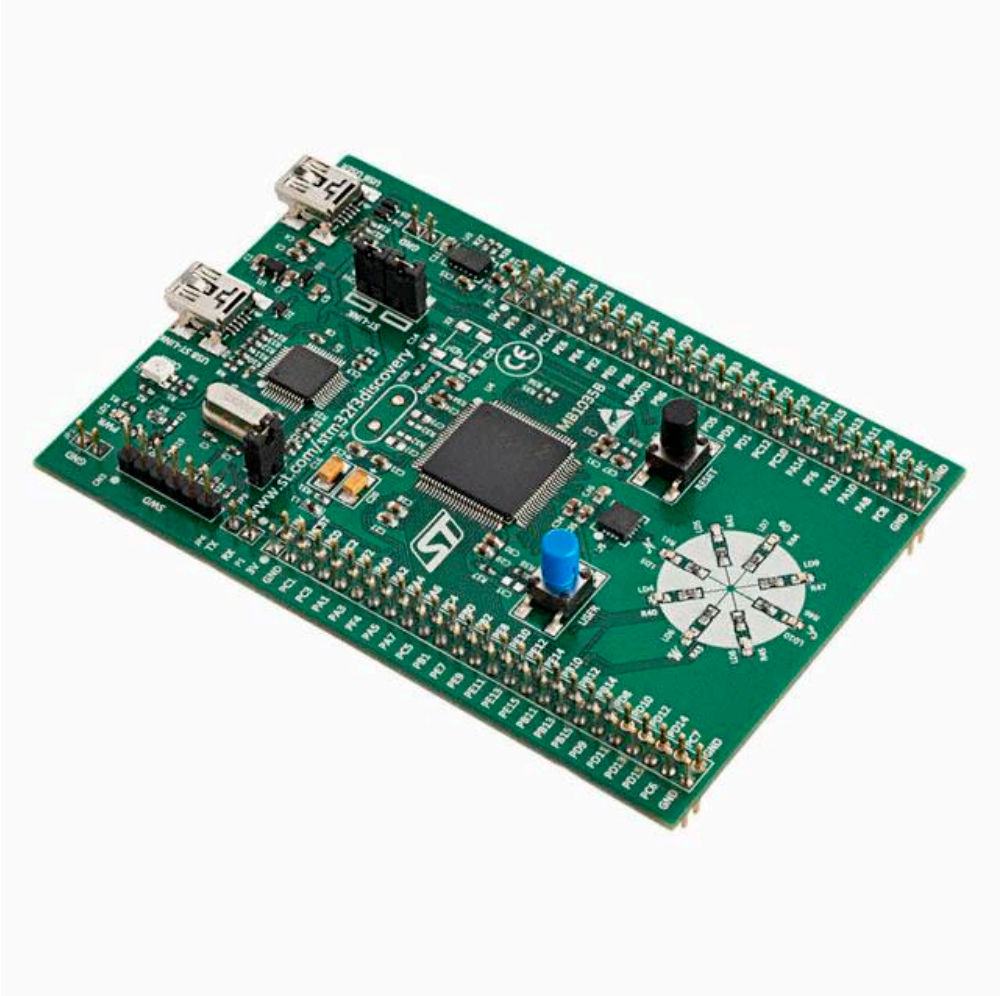


Рис. 5: Внешний вид платы STM32F3Discovery.

лиотеки осуществляет управление сервоприводом при помощи алгоирт-ма (9).

Соответствующий исходный код доступен в репозитории проекта Embox [29] в директории `platform/arm/stm32f3_agents`.

На тестовом макете из двух устройств было показано, что необходимое ПО действительно функционирует, а данные передаются по UART на хост в профайлер (рис. 6).

Сборка полноценного стенда для проведения физических экспериментов (с определённым профилем крыла, с реализацией всех электросхем и тому подобным) не входит в данную работу, но должна была быть сделана в рамках проекта. К сожалению, на текущий момент стенд всё ещё находится в процессе разработки, поэтому испытать фреймворк на большом количестве устройств не предоставляется возможным. В

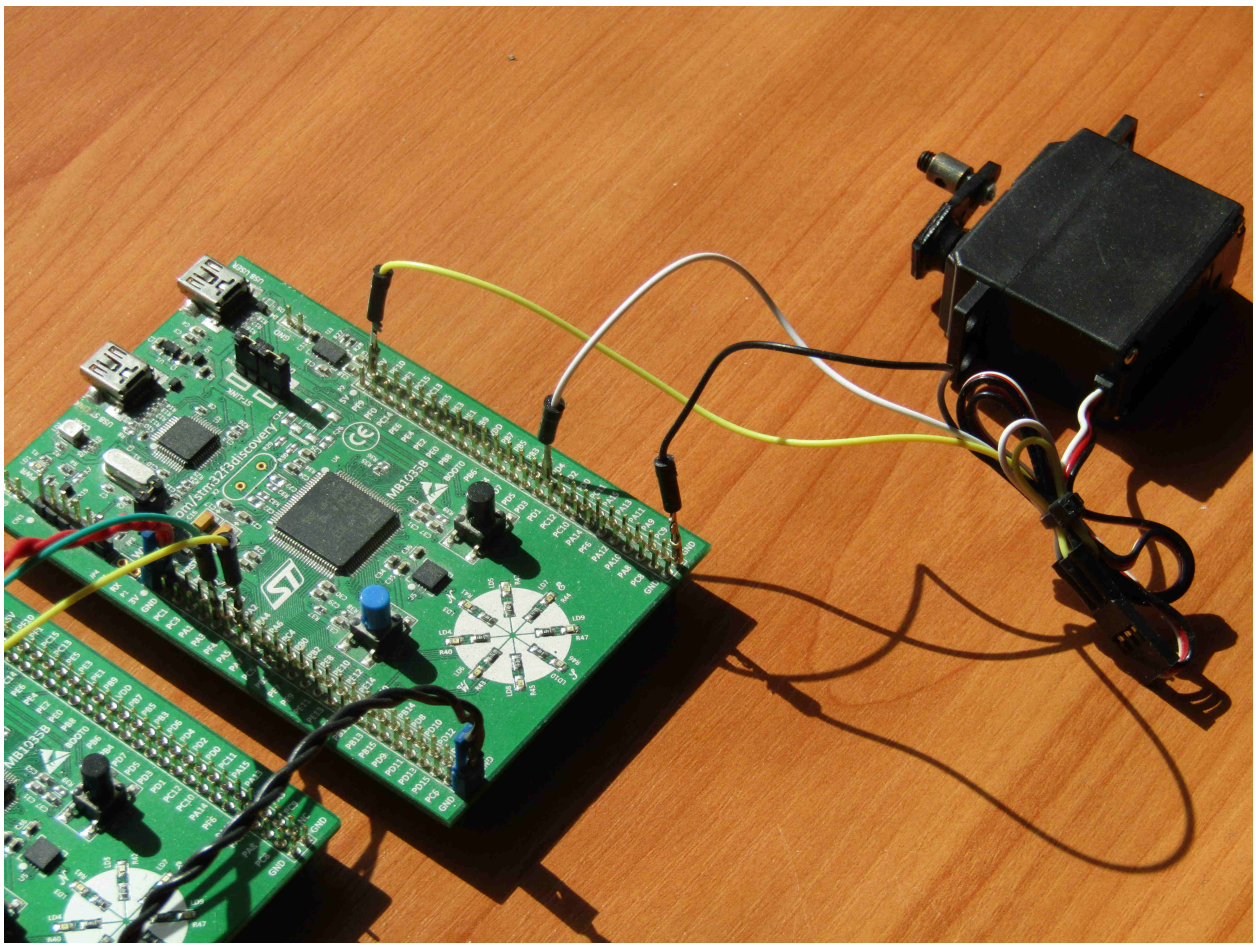


Рис. 6: Макет с двумя платами.

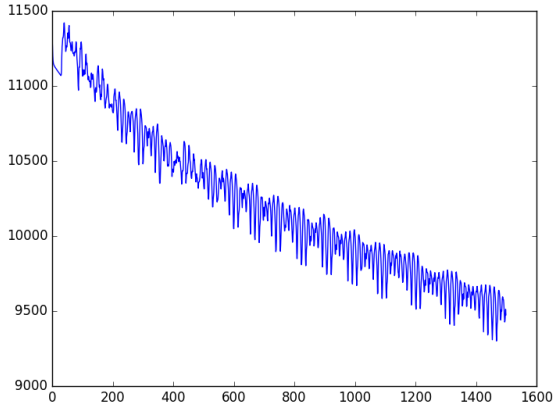
связи с этим исследование влияния параметров алгоритма ограничивается проведением моделирования алгоритма (9) для упрощённой математической модели турбулентного потока воздуха.

## 3.3. Исследование алгоритма

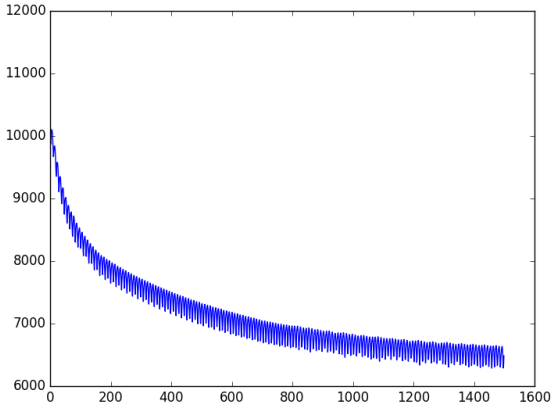
По виду алгоритма (9) видно, что для изменения поведения системы можно изменять следующие параметры.

- Размер шага $\gamma$.
- Отрезок $[\alpha_k^-; \alpha_k^+]$, в пределах которого изменяется $\alpha$.
- Топология системы (т.е. матрица связей $B$).
- Соотношение сил ламинарного и турбулентного потоков.

Если для алгоритма локального голосования 4 имеются строгие математические доказательства того, что система будет сходиться с определённой скоростью в зависимости от скорости шага и топологии, в нашем случае вывод подобных формул затруднён рядом сложностей. В данной работе не содержится формальных доказательств, но на ряде примеров можно видеть, как меняется функционал качества (8).




$\gamma = 0.01, \alpha^+ = \frac{\pi}{4}$.



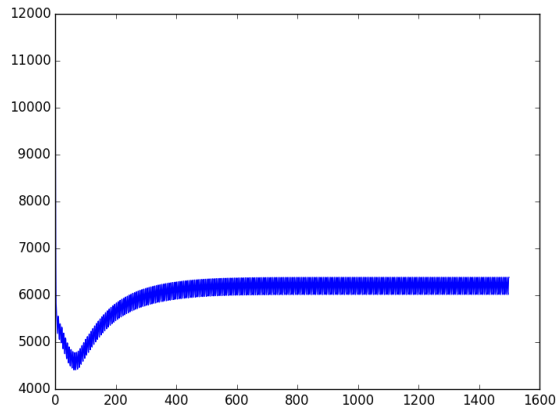

$\gamma = 0.1, \alpha^+ = \frac{\pi}{4}$.

Рис. 7: Различная скорость сходимости при разных коэффициентах $\gamma$.

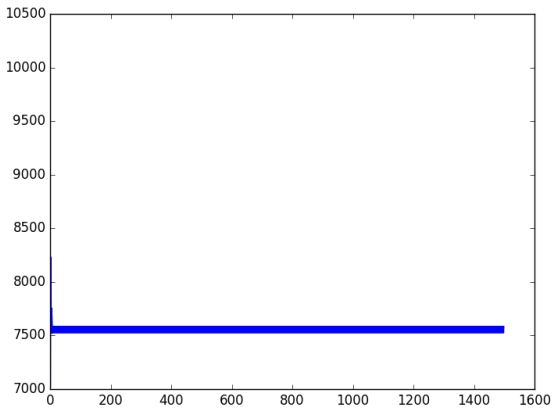
Как видно на рисунке 7, при меньших $\gamma$ функционал уменьшается медленнее, имея примерно тот же предел и порождая более мелкое "дрожание".

Вместе с тем, увеличение размера шага на несколько порядков (например, для $\gamma = 10, \gamma = 100$) не наблюдается явного расхождения алгоритма. Он продолжает сходиться, но к неоптимальному значению (рисунок 8).

Одной из интересных особенностей функционала качества при работе алгоритма (9) является "дрожание", которое видно на рисунках выше. Из-за этого эффекта, во-первых, даже при неизменных внешних условиях роботизированные устройства продолжают периодически менять своё положение, а во-вторых, уже нельзя говорить о строгой математической сходимости алгоритма. При уменьшени шага пропадает, но время сходимости алгоритма вырастает пропорционально. Было сделано предположение, что это происходит из-за центральных элементов на



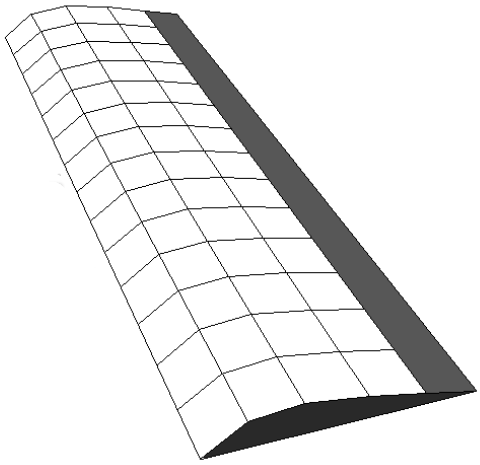

$\gamma = 10, \alpha^+ = \frac{\pi}{4}$.



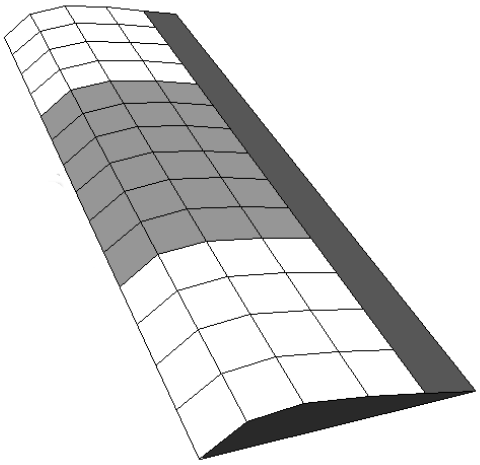

$\gamma = 100, \alpha^+ = \frac{\pi}{4}$.

Рис. 8: Различная скорость сходимости при разных коэффициентах $\gamma$.

крыле: из-за малого значения плеча возрастает управление $u_k^t$ (так как длина плеча $r_k$ находится в знаменателе). Изменение структуры крыла с регулярной на два региона с механизмами по краям крыла (см. рисунок 9) привело к стабилизации (см. рисунок 10).

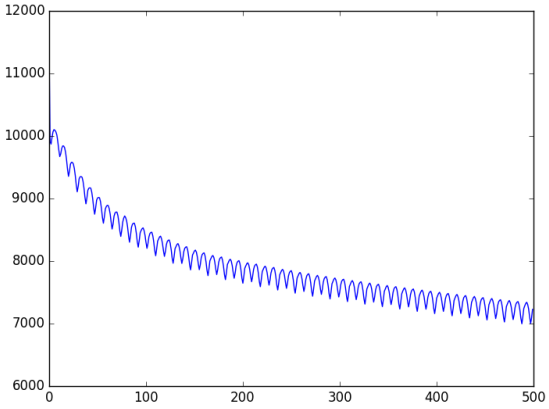


а) Использование всей поверхности крыла.

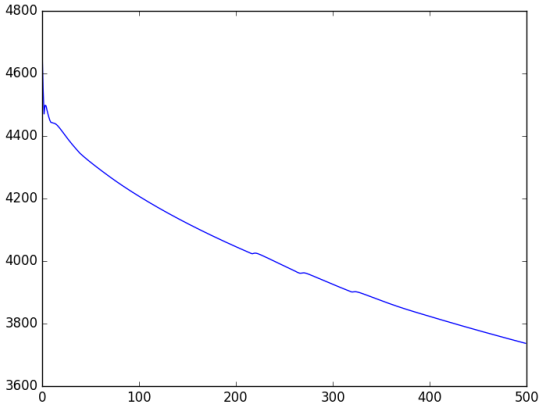


б) Использование двух регионов по краям крыла.

Рис. 9: Использование различных подмножеств массива датчиков (белым цветом выделены активные устройства, серым цветом выделены устройства, не участвующие в работе алгоритма).




а) Функционал качества для обычного крыла.




б) Функционал качества для крыла с двумя регионами датчиков.

Рис. 10: Поведение функционала качества при различных топологиях.

# Заключение

В ходе выполнения данной работы были получены следующие результаты.

1. Разработан фреймворк для работы с мультиагентными системами на языке Python, позволяющий собирать и обрабатывать данные о работе системы.

2. Выполнено моделирование работы алгоритма для упрощённой математической модели, показавшее применимость данного подхода.

3. Разработано ПО для макета для физических экспериментов на базе набора плат STM32F3Discovery, каждая из которых оснащена сервоприводом и датчиком давления.

4. Проведено исследование влияния различных параметров на поведение мультиагентной системы, использующей разработанный алгоритм. Показана зависимость скорости сходимости от данных параметров.

5. Результаты работы представлены на конференции СПИСОК-2017.

# Список литературы


[1] Bellifemine Fabio Luigi, Caire Giovanni, Greenwood Dominic. Developing multi-agent systems with JADE. — John Wiley & Sons, 2007. — Vol. 7.

[2] Boccaletti S. et al. Complex networks: Structure and dynamics // Physics reports. — 2006. — Vol. 424, no. 4. — P. 175–308.

[3] Chebotarev P. Yu., Agaev R. P. Coordination in multiagent systems and Laplacian spectra of digraphs // Automation and Remote Control. — 2009. — Vol. 70, no. 3. — P. 469–483.

[4] Chen Yao et al. Multi-agent systems with dynamical topologies: Consensus and applications // IEEE circuits and systems magazine. — 2013. — Vol. 13, no. 3. — P. 21–34.

[5] Granichin Oleg, Khantuleva Tatjana. Hybrid systems and randomized measuring in nonequilibrium processes // Differential Equations and Control Processes. — 2004. — no. 3. — P. 35–43.

[6] Granichin Oleg, Khantuleva Tatjana. Local voting protocol for the adaptation of airplane's "feathers" in a turbulence flow // In: Proc. of the 2017 American Control Conference, May 24-26. — 2017.

[7] Granichin Oleg, Khantuleva Tatjana, Amelina Natalia. Adaptation of Aircraft's Wings Elements in Turbulent Flows by Local Voting Protocol // IFAC Proceedings. — 2017.

[8] Khantuleva T.A., Meshcheryakov Yu.I. Nonequilibrium processes in condensed media. Part 2. Structural instability induced by shock loading // Physical Mesomechanics. — 2016. — Vol. 19(1). — P. 69–76.

[9] Lewis F.L. et al. Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches (Communications and Control Engineering). — Springer, 2014. — P. 307.

[10] Meshcheryakov Yu.I., Khantuleva T.A. Nonequilibrium processes in condensed media: Part 1. Experimental studies in light of nonlocal transport theory // Physical Mesomechanics. — 2015. — Vol. 18(3). — P. 228–243.

[11] Olfati-Saber R., Fax J. A., Murray R. M. Consensus and cooperation in networked multi-agent systems // Proceedings of the IEEE. — 2007. — Vol. 95, no. 1. — P. 215–233.

[12] Olfati-Saber R., Murray R.M. Consensus problems in networks of agents with switching topology and time-delays // Automatic Control, IEEE Transactions on. — 2004. — Vol. 49, no. 9. — P. 1520–1533.

[13] Ren W., Beard R.W., Atkins E.M. Information consensus in multivehicle cooperative control // Control Systems, IEEE. — 2007. — Vol. 27, no. 2. — P. 71–82.

[14] Utkin V.I. Sliding Modes in Control and Optimization // Springer-Verlag. — 1992.

[15] Yong-Zheng Sun, Jiong Ruan. Leader–follower consensus problems of multi-agent systems with noise perturbation and time delays // Chinese Physics Letters. — 2008. — Vol. 25, no. 9. — P. 3493.

[16] Амелина Н.О. Применение протокола локального голосования для децентрализованной балансировки загрузки сети с переменной топологией и помехами в измерениях // Вестник Санкт-Петербургского университета. Серия 1. Математика. Механика. Астрономия. — 2013. — Vol. 3. — P. 12–20.

[17] Амелина Н. О. Мультиагентные технологии, адаптация, самоорганизация, достижение консенсуса // Стохастическая оптимизация в информатике. — 2011. — Vol. 7. — P. 149–185.

[18] Граничин О. Н. Как действительно устроены сложные информационно-управляющие системы? // Стохастическая оптимизация в информатике. — 2016. — Vol. 12. — P. 3–19.

[19] Граничин О. Н., Хантулева Т.А. Адаптация элементов крыла (”перьев”) самолёта в турбулентном потоке в турбулентном потоке с помощью мультиагентного протокола // Автоматика и телемеханика (в печати). — 2017.

[20] Домашняя страница OMG MASIF // Open Media Group, Mobile Agent System Interoperability Facilities Specification. — Режим доступа: http://www.omg.org/cgi-bin/doc?orbos/97-10-05 (дата обращения: 20.04.2017).

[21] Домашняя страница проекта Comtec // Comtec Agent Platform. — Режим доступа: http://fipa.comtec.co.jp/glointe.htm (дата обращения: 20.04.2017).

[22] Домашняя страница проекта FIPA // The Foundation for Intelligent Physical Agents. — Режим доступа: http://http://fipa.org/ (дата обращения: 20.04.2017).

[23] Домашняя страница проекта FIPA-OS. — Режим доступа: http://fipa-os.sourceforge.net/index.htm (дата обращения: 20.04.2017).

[24] Домашняя страница проекта Jade // JAVA Agent Development Framework. — Режим доступа: http://jade.tilab.com/ (дата обращения: 20.04.2017).

[25] Домашняя страница проекта Zeus // Zeus Agent Toolkit. — Режим доступа: https://sourceforge.net/p/zeusagent/news/ (дата обращения: 20.04.2017).

[26] Емельянов С.В. Системы автоматического управления с переменной структурой. — Москва : Наука, 1967.

[27] Зубарев Д. Н. Неравновесная статистическая термодинамика. — Москва : Наука, 1971.

[28] Репозиторий на Github с исходными кодами фреймворка для работы с мультиагентыми системами. — Access mode: `https://github.com/Deryugin/visual-agents` (online; accessed: 4.05.2017).

[29] Репозиторий проекта Embox. — Access mode: `https://github.com/embox/embox` (online; accessed: 2.05.2017).

[30] Синай Я.Г. Построение кластерной динамики для динамических систем статистической механики // Вестник МГУ. — 1974. — Vol. 29(1). — P. 152–158.

[31] Тюшев К.И. Мультиагентные технологии для построения RAID-подобных распределенных систем хранения данных: дипломная работа. — 2014.