САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра системного программирования

РАЗРАБОТКА МОДУЛЯ ОЦЕНИВАНИЯ ПАРАМЕТРОВ ДЛЯ СИСТЕМ С РАНДОМИЗИРОВАННЫМИ УПРАВЛЕНИЯМИ

Дипломная работа студентки 545 группы

Филипповой Анастасии Валерьевны

Научный руководитель	…… / подпись /	д.ф.-м.н., проф. Граничин О.Н.
Рецензент	…… / подпись /	Бондарев А.В.
“Допустить к защите” заведующий кафедрой,	…… / подпись /	д.ф.-м.н., проф. Терехов А.Н.

Санкт-Петербург
2012

SAINT PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Software Engineering Chair

DEVELOPMENT OF PARAMETER ESTIMATOR FOR SYSTEMS WITH RANDOMIZED CONTROLS

by

Anastasia, Filippova

Graduate paper

Supervisor	……	Professor O. N. Granichin
Reviewer	……	A. V. Bondarev
"Approved by" Head of Department	……	Professor A. N. Terekhov

Saint Petersburg
2012

Оглавление

1. Введение

Сложность окружающего мира не позволяет человеку охватить его в полной мере, что приводит к необходимости рассмотрения отдельных его аспектов и упрощённого их описания с помощью моделей. Никакая модель не даёт полноценного представления описываемой ею системы. Обычно в модель вводится понятие «помехи», в которое закладывается влияние на систему всех прочих, не рассматриваемых моделью, аспектов.

Каждая модель реальной системы имеет свои границы применимости. Использование модели в этих границах позволяет осознанно влиять на описываемую систему, управлять ею. Принципами и методами управления различными системами занимается наука «Теория управления».

Процесс управления можно представить как подачу на вход динамической системе (объекту управления) некоторых управляющих воздействий (входных сигналов) для получения требуемых выходных сигналов, то есть достижения цели управления.

Каждая модель конкретной системы обладает множеством параметров. Для эффективного управления системой необходимо знать их значения. Как правило, некоторые или все из них неизвестны. В таком случае ищут приближённые значения для неизвестных параметров, проводя так называемую идентификацию системы. Идентификация осуществляется путём подачи пробных, тестовых сигналов на вход системе и анализа получаемых от неё выходных сигналов. Оценивание неизвестных значений параметров обеспечивается свойствами пробного сигнала. Недостаточная вариативность этого сигнала сильно усложняет процесс идентификации. С другой стороны, подача на вход «хорошего» сигнала позволяет успешно идентифицировать параметры системы при любой ограниченной помехе, даже не обладающей полезными статистическими свойствами или вообще не случайной.

Можно проводить идентификацию системы в процессе управления ею, путём добавления пробного сигнала к эффективному управляющему воздействию. В таких случаях говорят о процессе адаптивного управления.

На данный момент существует множество теоретических работ по управлению, идентификации параметров динамических систем и адаптивному управлению.

Практическое решение задачи управления сводится к построению регулятора – устройства, генерирующего управляющие сигналы для объекта управления на основе сведений о его входах и выходах в предыдущие моменты времени. Большинство

регуляторов строятся по принципу обратной связи, реже используется прямая связь. Важным свойством алгоритмов управления является возможность корректной работы с нестабильными системами.

Задача идентификации параметров динамической системы решается по-разному. Некоторые алгоритмы находят конкретное приближённое значение. Такие алгоритмы в большинстве своём являются итеративными, то есть позволяют улучшать оценку с каждым шагом. Важным свойством таких алгоритмов является сходимость последовательности оценок в том или ином смысле к точному значению оцениваемого параметра при устремлении количества итераций к бесконечности.

Другим классом алгоритмов идентификации параметров являются алгоритмы построения доверительных множеств. Такие алгоритмы находят не конкретное приближённое значение, а интервал, в котором с некоторой задаваемой вероятностью находится точное значение оцениваемого параметра.

Задачи адаптивного управления решаются путём смешивания различных алгоритмов управления, идентификации параметров и построения доверительных множеств. Часто используется сразу несколько алгоритмов из каждой категории, которые работают поочерёдно в зависимости от ряда условий, настраиваемых под конкретную задачу.

Теория управления активно развивается, что приводит к постоянному увеличению количества теоретических работ и затруднениям при выборе подходящих алгоритмов для решения конкретной задачи. Алгоритмы данной предметной области зависят от большого количества настраиваемых параметров, грамотный выбор которых существенно влияет на корректность их работы. Ситуация усугубляется ещё и тем, что в практических задачах алгоритмы приходится совмещать в различных комбинациях.

Существуют специализированные математические пакеты, такие как MathWorks MATLAB [13], Waterloo Maple [19] и Wolfram Mathematica [21], позволяющие проводить моделирование, анализ и проектирование систем автоматического управления. Однако, из-за их громоздкости, данные решения больше подходят для теоретических изысканий.

В дипломной работе рассматривается процесс разработки модуля для подбора (оценивания) параметров алгоритмов, позволяющего удобно и наглядно проводить тестирование и сравнение их работы.

1. Постановка задачи

Рассмотрим задачу адаптивного управления некоторым объектом управления (ОУ) со скалярными входами и выходами, описываемым как дискретная динамическая система, с аддитивно действующей на неё ограниченной помехой:

$$y_t = G(\lambda)u_t + v_t. \tag{2.1}$$

где y_t – выход ОУ, u_t – вход ОУ (управляющее воздействие), v_t – ограниченная помеха (возмущающее воздействие): $|v_t| \le C_v$, λ – оператор сдвига на такт назад: $\lambda u_t = u_{t-1}$, $G(\lambda)$ – передаточная функция ОУ, которая отражает зависимость между входами и выходами.

Будем рассматривать класс линейных передаточных функций:

$$G(\lambda) = \frac{b(\lambda,\tau)}{a(\lambda,\tau)}. \tag{2.2}$$

где $a(\lambda,\tau)$ и $b(\lambda,\tau)$ – многочлены от λ с коэффициентами из вектора параметров τ:

$$a(\lambda,\tau) = 1 + \lambda a_1 + \lambda^2 a_2 + \cdots + \lambda^{p_a} a_{p_a}, \tag{2.3}$$

$$b(\lambda,\tau) = \lambda b_l + \lambda^2 b_{l+1} + \cdots + \lambda^{p_b} b_{l+p_b-1,} \tag{2.4}$$

$$\tau = \begin{pmatrix} a_1 \\ \vdots \\ a_{p_a} \\ b_l \\ \vdots \\ b_{l+p_b-1} \end{pmatrix}. \tag{2.5}$$

Здесь a_i и b_j – параметры ОУ, значения некоторых из которых неизвестны, p_a – порядок ОУ по выходам, p_b – порядок ОУ по входам, l – запаздывание по управлению (т.е. выход в момент времени t формируется исходя из управления, поданного l и более тактов назад).

Перепишем формулу (2.1) в виде:

$$y_t + \sum_{i=1}^{p_a} a_i y_{t-i} = \sum_{j=l}^{l+p_b-1} b_j u_{t-j} + v_t, t = 1,2, \ldots, \tag{2.6}$$

Назовём порядком объекта управления $\max(p_a, p_b)$.

Ограничимся объектами управления, описываемыми уравнениями не выше второго порядка:

$$y_t + a_1 y_{t-1} + a_2 y_{t-2} = b_l u_{t-l} + b_{l+1} u_{t-l-1} + v_t, t = 1,2, \ldots \tag{2.7}$$

1.1. Формальная постановка задачи

1. Реализовать симулятор с возможностью выбора параметров объекта управления и типов неопределённостей (помех)
2. Спроектировать расширяемый модуль адаптивного управления с возможностью выбора используемых алгоритмов и задания их начальных параметров
3. Реализовать модуль адаптивного управления с возможностью встраивания в реальную физическую систему путём замены симулятора на данные датчиков
4. Реализовать несколько демонстрационных алгоритмов для использования в модуле адаптивного управления

2. Обзор существующих решений

Идея использования информационных технологий в задачах идентификации и управления появилась давно.

Рассмотрим существующие средства, с помощью которых обычно производится моделирование различных систем управления. Первым и наиболее известным является инструмент для моделирования Simulink [17], интегрированный в пакет MATLAB [13]. Работа с Simulink состоит в визуальном построении схемы взаимодействующих блоков, их настройки и дальнейшей симуляции полученной системы.

Simulink состоит из множества модулей. Наиболее известным модулем для систем управления является Control System Toolbox [10], предоставляющий коллекцию алгоритмов для моделирования и анализа таких систем. В пакете представлена возможность работы с наиболее традиционными типами передаточных функций. Пакет позволяет моделировать и анализировать как непрерывные, так и дискретные системы.

Другим известным модулем является Robust Control Toolbox [15]. Он позволяет проектировать и анализировать многопараметрические системы управления, для которых важна устойчивость. Возможно решение задач адаптивного управления при работе с моделями, содержащими ошибки, а также моделями, динамика которых известна не полностью и/или изменяется с течением времени. Модуль является надстройкой над уже рассмотренным пакетом Control System Toolbox, предоставляющей усовершенствованный набор алгоритмов для проектирования систем управления.

Также существует аналог с открытым исходным кодом для инструмента Simulink - Scicos [16] для пакета GNU Octave [11], который в свою очередь представляет собой открытый аналог для пакета MATLAB. Аналогом модуля Control System Toolbox является Octave Control Systems Toolbox (OCST).

3. Реализация

3.1. Используемые технологии

Первым вопросом на стадии реализации предложенного модуля оценивания параметров стал выбор используемых технологий.

3.1.1. Язык программирования

В качестве языка программирования была возможность выбора между применением некоторого специализированного языка для математических расчётов (например, MATLAB [13] или GNU Octave [11]) и использованием классических языков программирования общего назначения. Классические языки программирования значительно более распространены в мире и, таким образом, предоставляют более благоприятные условия для расширяемости создаваемого модуля. Преимуществом специализированных языков является возможность использования огромного количества встроенных в язык математических алгоритмов, формул, преобразований и т.д. Это может быть полезно для применения модуля со сложными алгоритмами, использующими обширный математический аппарат. Однако существуют способы получить все необходимые математические средства и в классических языках, используя сторонние математические библиотеки. Также возможна генерация кода на классических языках из специализированных математических пакетов (MATLAB).

После изучения существующих решений был сделан вывод, что для написания встраиваемого модуля специализированные математические пакеты не подходят. В случае пакета MATLAB существует возможность генерация кода на C/C++ под некоторые аппаратные платформы, но код при этом получается слишком громоздким. К тому же данная возможность доступна далеко не для всех платформ. В случае же открытого аналога GNU Octave возможность генерации кода отсутствует.

Среди классических языков была возможность выбора между управляемыми языками, интерпретируемыми виртуальной машиной, (языки платформы Microsoft.NET [8], Java [9]), и языками, компилируемыми в машинные коды. Управляемые языки традиционно считаются более медленными, по сравнению с компилируемыми, в связи с

накладными расходами на интерпретацию байт-кода. Также на быстродействии управляемых языков сказывается повсеместное использование в них сборщиков мусора для управления памятью, которое приводит к снижению производительности в моменты их вызова. Стоит отметить, что современное положение дел всё же позволяет во многих практических ситуациях говорить о почти одинаковом быстродействии с компилируемыми языками. Управляемые языки стали значительно быстрее за счёт активного использования различных оптимизаций и, в частности, JIT (Just-In-Time)-компиляции [12]. Однако проблема падения производительности во время сборки мусора до сих пор остаётся актуальной, в связи с чем, крайне не желательно использование управляемых языков в ситуациях, где требуется хорошее время отклика, а тем более в системах реального времени, где они вообще не применимы. Основным преимуществом управляемых языков является более быстрый, удобный и надёжный процесс разработки. К квалификации программиста, использующего управляемые языки, предъявляются значительно меньшие требования. Таким образом, использование подобного языка благоприятно сказалось бы на расширяемости модуля.

После рассмотрения и анализа всех преимуществ и недостатков управляемых и компилируемых языков программирования, для написания модуля было решено использовать компилируемый язык, обеспечивающий большие возможности для применения во встраиваемых устройствах, пускай и ценой увеличения затрат на разработку модуля и его расширение дополнительными алгоритмами.

Среди компилируемых языков был выбран самый распространённый язык программирования – C++. Была также рассмотрена возможность использования более низкоуровневого языка C, однако реализация на нём расширяемой системы значительно затруднена в силу его процедурной природы. При реализации самого модуля были приложены усилия для минимизации количества используемых сторонних библиотек во избежание проблем с его применением во встраиваемых системах. Конкретные алгоритмы, как уже отмечалось выше, могут зависеть от дополнительных библиотек, например для получения необходимых математических средств. Возможность использования таких алгоритмов во встраиваемых системах должна рассматриваться отдельно.

3.1.2. Интерфейс пользователя

Для использования модуля как средства проведения наглядного тестирования и сравнения различных алгоритмов необходимо было разработать удобный пользовательский интерфейс.

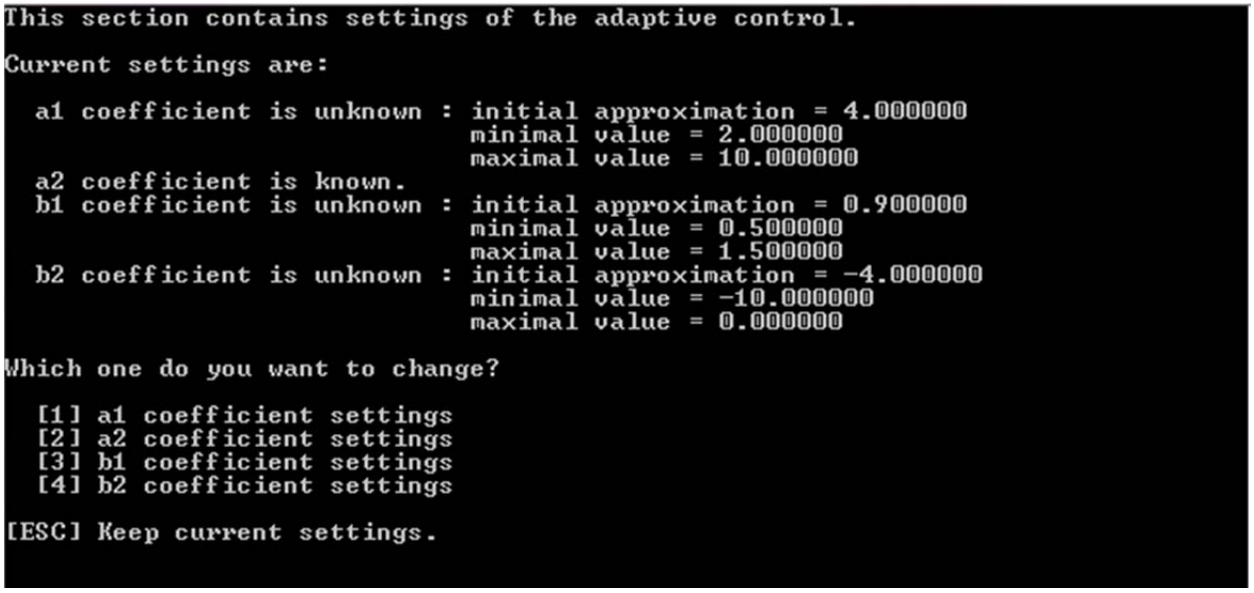

Рис. 1. Задание оценок для параметров ОУ

Первоначально для отладки работы модуля был создан консольный интерфейс, позволяющий задавать истинные значения параметров симулируемого объекта управления, указывать тип помех и их характеристики, выбирать множество используемых алгоритмов с заданием значений для всех их параметров (Рис. 1) и отображать в виде таблицы результаты симуляции, а именно значения входов, выходов и оценок для всех параметров ОУ (Рис. 2).

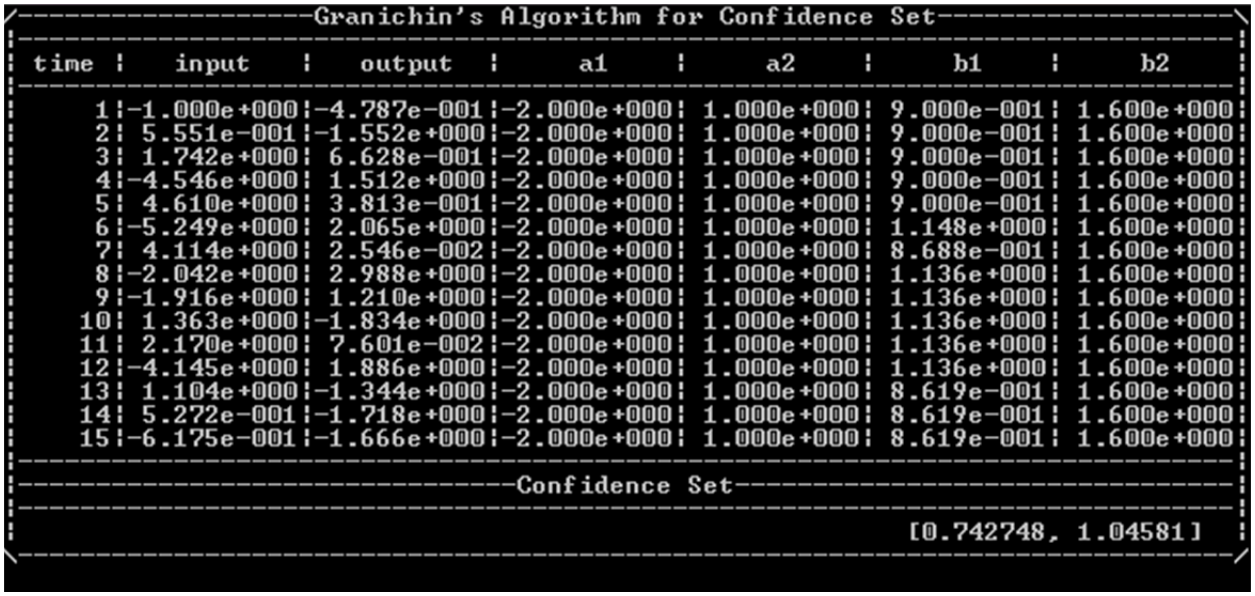

Рис. 2. Построение доверительного интервала

В дальнейшем был разработан графический интерфейс на базе Windows API [20], обладающий тем же функционалом, но предоставляющий значительно более наглядный способ выбора алгоритмов и возможность изменения всевозможных параметров «на лету» при помощи графических элементов управления.

3.2. Архитектура модуля

Описание архитектуры модуля можно разбить на пять частей: описание ядра модуля, объектов управления, типов помех, алгоритмов и вспомогательных классов.

На Рис. 3 представлена диаграмма классов созданного модуля и набора демонстрационных алгоритмов.

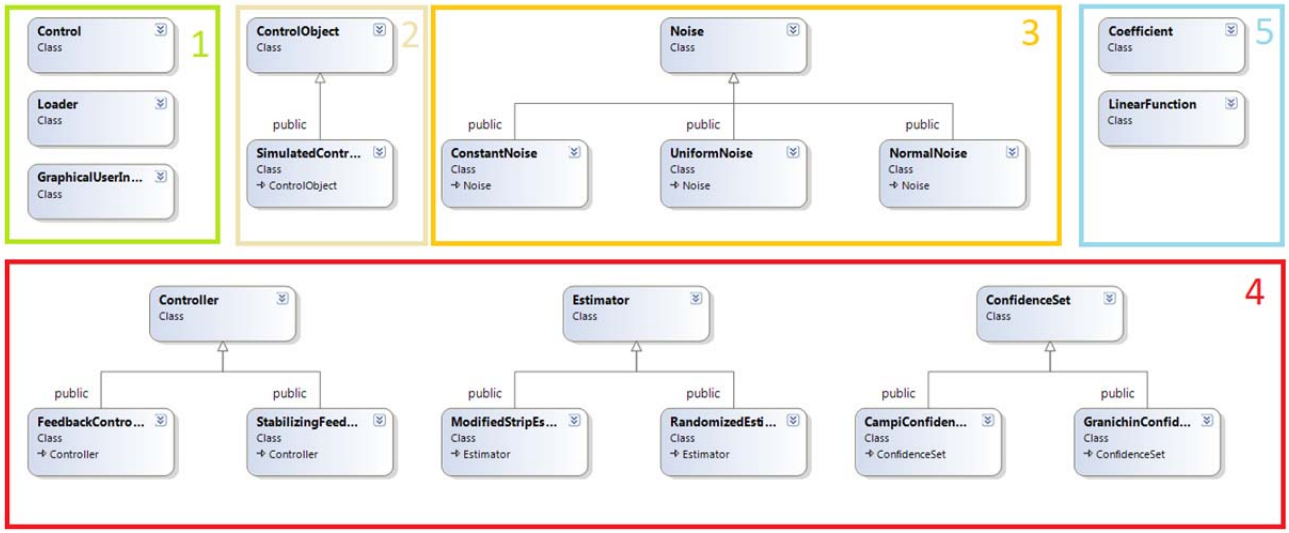

Рис. 3. Диаграмма классов. 1 - Ядро модуля, 2- Объекты управления, 3 - Типы помех, 4 - Алгоритмы, 5 - Вспомогательные классы

3.2.1. Ядро модуля

Главным классом, составляющим ядро модуля оценивания параметров, является класс `Control`, задачей которого является управление заданным ОУ с использованием предоставленного набора алгоритмов. В конструктор класса передаётся объект, представляющий конкретный ОУ, которым необходимо управлять, а также массив алгоритмов, которые необходимо использовать для управления, и значения их начальных параметров.

Также ядро содержит класс `Loader`, обеспечивающий подгрузку динамических библиотек для расширения модуля дополнительными объектами управления, алгоритмами и типами помех.

Графический интерфейс пользователя также отнесён к ядру системы и представлен классом `GraphicalUserInterface`, хотя является отключаемым, что позволяет встраивать модуль и использовать его для непосредственно управления физическими системами, которые не имеют средств для вывода графической информации.

3.2.2. Объекты управления

Произвольный объект управления представляется с использованием абстрактного (или pure virtual в терминологии языка C++ [14]) класса `ControlObject` и некоторого конкретного класса, от него наследующегося. В классе `ControlObject` объявлена функция `makeStep`, предназначенная для преобразования подаваемого в неё входного значения в выходное. Классы наследники определяют эту функцию в соответствии с тем, как функционирует конкретный описываемый объект управления. Базовый класс `ControlObject` также обеспечивает сохранение истории входов и выходов на задаваемое в конструкторе количество шагов в прошлое. История необходима для функционирования большинства алгоритмов, работающих над объектами управления. Функции `getInput` и `getOutput` позволяет получить доступ к этим входам и выходам соответственно или возвращают задаваемое в одном из их аргументов значение по-умолчанию.

Важным классом наследником является класс `SimulatedControlObject`, функционал которого обеспечивает возможность симуляции процесса управления некоторым линейным объектом. Конструктор класса принимает массивы коэффициентов, определяющих передаточную функцию объекта управления, объект, описывающий действующую на ОУ помеху, и величину запаздывания ОУ по управлению. Возможна симуляция объектов управления произвольного порядка. Для упрощения работы с частным случаем ОУ не выше второго порядка (с которым работают демонстрационные алгоритмы) реализованы перегруженные конструкторы, принимающие конкретные коэффициенты вместо массивов.

3.2.3. Типы помех

Для проведения симуляции реализован ряд классов, описывающих различные типы помех (неопределённостей), влияющих на симулируемый объект управления. Все помехи наследуются от абстрактного класса `Noise`. В классе `Noise` объявлена функция next, возвращающая очередное значение помехи.

В демонстрационных целях реализованы следующие типы помех: постоянная помеха - `ConstNoise`, равномерно-распределённая на задаваемом интервале помеха – `UniformNoise` и нормально-распределённая помеха с задаваемыми средним значением и среднеквадратичным отклонением - `NormalNoise`.

3.2.4. Алгоритмы

Модуль предполагает использование алгоритмов трёх различных типов: регуляторы, алгоритмы идентификации и алгоритмы построения доверительного множества. Каждый из типов алгоритмов представлен своим абстрактным базовым классом - `Controller`, `Estimator` и `ConfidenceSet` соответственно. Конкретные алгоритмы наследуются от этих базовых классов.

Для демонстрации работы модуля были выбраны и реализованы следующие алгоритмы.

Регуляторы представлены двумя классами: `FeedbackController` и `StabilizingFeedbackController`, реализующими регулятор типа обратной связи и стабилизирующий регулятор типа обратной связи соответственно.

В классе `RandomizedEstimator` реализован рандомизированный алгоритм идентификации типа стохастической аппроксимации (РСА). В классе `ModifiedStripEstimator` – стабилизирующий алгоритм «модифицированная полоска».

Также в работе представлено два алгоритма построения доверительного множества: алгоритм Campi M.C. (класс `CampiConfidenceSet`) и алгоритм Граничина О.Н. (класс `GranichinConfidenceSet`).

Данные, полученные в ходе тестирования алгоритмов, проведённого на представленных в оригинальных работах авторов [1, 2, 3, 5] примерах объектов управления и начальных параметров, совпадают с опубликованными результатами.

Подробно реализованные алгоритмы будут рассмотрены ниже в соответствующем разделе.

3.2.5. Вспомогательные классы

В процесс реализации модуля возникла потребность в некоторых вспомогательных классах, не представляющих особого интереса по отдельности, но являющихся, тем не менее, неотъемлемой частью архитектуры системы.

Класс `Coefficient` описывает состояние некоторого коэффициента объекта управления. Возможными состояниями являются состояние точного известного значения и состояние некоторого приближённого значения (оценки). Для коэффициентов в состоянии оценки хранится текущий интервал, в котором предполагается нахождение точного значения. Объекты этого класса активно используются в алгоритмах идентификации и построения доверительного множества.

Класс `LinearFunction` описывает линейную функцию многих аргументов. Такие функции активно используются в алгоритмах построения доверительного множества. В классе определены методы для сложения линейных функций и умножения их на скаляр.

4. Обзор реализованных алгоритмов

Созданный модуль предполагает использование трёх различных классов алгоритмов: регуляторы, алгоритмы идентификации и алгоритмы построения доверительного множества.

4.1. Регуляторы

Регулятор предназначен для построения такой последовательности управляющих воздействий на объект управления, которая позволяет достичь цели управления, то есть получить на выходе последовательность сигналов, близкую к требуемой. Для построения регулятора необходимо иметь представление о том, как работает объект управления, то есть нужна модель объекта управления и точные или приближённые значения определяющих её параметров.

Рассмотрим алгоритм работы регулятора обратной связи на примере линейной дискретной системы с ограниченной аддитивной помехой [3]. Пусть целью управления является последовательность выходов y_t^*. Необходимо построить такую функцию $U(y_t, y_{t-1}, \dots, y_{t-p_a+1}, u_{t-l}, u_{t-l-1}, .., u_{t-l-p_b+2})$, что при подаче на вход объекту управления последовательности $u_t = U(y_t, y_{t-1}, \dots, y_{t-p_a+1}, u_{t-l}, u_{t-l-1}, .., u_{t-l-p_b+2})$, получаемые выходы будут максимально близки к требуемым, то есть: $|y_t - y_t^*| \to min$. При этом не имеется никаких сведений о типе и величине помехи v_t, за исключением её ограниченности некоторым значением C_v.

Путём сдвига по времени на l шагов назад, уравнение объекта управления может быть записано в виде:

$$y_{t+l} + \sum_{i=1}^{p_a} a_i y_{t-i+l} = \sum_{j=l}^{l+p_b-1} b_j u_{t-j+l} + v_{t+l}, t = -l+1, -l+2, \dots \quad (5.1)$$

Отсюда легко выражается формула для входов u_t:

$$u_t = \frac{y_{t+l}}{b_l} + \sum_{i=1}^{p_a} \frac{a_i}{b_l} y_{t-i+l} - \sum_{j=l+1}^{l+p_b-1} \frac{b_j}{b_l} u_{t-j+l} - \frac{v_{t+l}}{b_l}. \quad (5.2)$$

Подставив вместо $y_{t+l}, y_{t+l-1}, \dots, y_{t+1}$ желаемые значения $y_{t+l}^*, y_{t+l-1}^*, \dots, y_{t+1}^*$ и опустив неизвестное значение помехи v_{t+l}, получаем формулу для функции U:

$$U(y_t, y_{t-1}, \dots, y_{t-p_a+1}, u_{t-l}, u_{t-l-1}, .., u_{t-l-p_b+2}) = \\ = \frac{y^*_{t+l}}{b_l} + \sum_{i=1}^{l-1} \frac{a_i}{b_l} y^*_{t-i+l} + \sum_{i=l}^{p_a} \frac{a_i}{b_l} y_{t-i+l} - \sum_{j=l+1}^{l+p_b-1} \frac{b_j}{b_l} u_{t-j+l}. \quad (5.3)$$

Формула (5.3) описывает регулятор обратной связи, который является оптимальным для устойчивого по управлению ОУ. В рассматриваемом нами случае объектов управления не выше второго порядка, формула (5.3) принимает вид:

$$U(y_t, y_{t-1}, u_{t-1}) = \frac{a_1}{b_1} y_t + \frac{a_2}{b_1} y_{t-1} - \frac{b_2}{b_1} u_{t-1}. \quad (5.4)$$

Для неустойчивых по управлению систем, можно построить лишь субоптимальный регулятор обратной связи. Однако в случае ОУ не выше второго порядка, такой регулятор является оптимальным [3] и имеет вид:

$$U(y_t, y_{t-1}, u_{t-1}) = \frac{a^*_1}{b^*_1} y_t + \frac{a^*_2}{b^*_1} y_{t-1} - \frac{b^*_2}{b^*_1} u_{t-1}, \quad (5.5)$$

$$a^*_1 = a_2 b_2 + a_1 a_2 b_1 - a_1^2 b_2, \quad (5.6)$$

$$a^*_2 = a_2^2 b_1 - a_1 a_2 b_2, \quad (5.7)$$

$$b^*_1 = a_2 b_1^2 + b_2^2 - a_1 b_1 b_2, \quad (5.8)$$

$$b^*_2 = a_2 b_1 b_2 - a_1 b_2^2. \quad (5.9)$$

4.2. Идентификация параметров

Алгоритмы идентификации предназначены для нахождения приближённых значений неизвестных параметров некоторого динамического объекта. Заранее предполагается сделанным выбор модели, описывающей данный объект. В большинстве своём такие алгоритмы являются итеративными. Полученное приближённое значение всегда можно улучшить, добавив к рассмотрению дополнительные значения входов и выходов, пересчёт будет проводиться с использованием лишь этих новых значений и предыдущего приближения. В теоретических работах [3, 4] доказывается, что последовательность оценок, получаемых с использованием алгоритмов этого класса, сходится в том или ином смысле к точным значениям оцениваемых параметров при устремлении количества итераций к бесконечности.

Идентификация неизвестных параметров осуществляется за счёт подачи некоторых пробных сигналов на вход динамического объекта и анализа получаемых выходных значений. Успешность процесса идентификации сильно зависит от вариативности этих

входных сигналов. Довольно успешным подходом к решению данной проблемы является использование рандомизированных пробных сигналов. Такие сигналы являются случайными величинами, обладающими некоторой заранее известной функцией распределения. Статические свойства помех, с другой стороны, обычно не известны, или сами помехи даже не являются случайными величинами. Рандомизированные сигналы позволяют получить достаточную вариативность управляющий воздействий, и в то же время не наносят вреда в силу известности их статистических параметров. В задачах адаптивного управления, когда процессы идентификации и непосредственно управления происходят одновременно, рандомизированные сигналы прибавляются к полезному управляющему воздействию, получаемому из регулятора, и эта сумма подаётся на вход объекту управления.

4.2.1. Рандомизированный алгоритм идентификации

Основной идеей, используемой в этом алгоритме, является приведение модели авторегрессии скользящего среднего, описывающей динамическую систему, к более простой модели линейной регрессии и применение к ней рандомизированного алгоритма типа стохастической аппроксимации (РСА) для оценивания вектора неизвестных параметров.

Пусть объект управления описывается моделью авторегрессии скользящего среднего:

$$a(\lambda,\tau)y_t = b(\lambda,\tau)u_t + v_t, t = l, l+1, \dots \quad (5.10)$$

Выбирается некоторое натуральное число $s > p_a + p_b - l$. Пробное возмущение (рандомизированный сигнал) Δ_n добавляется на каждом s-ом шаге.

Управление u_t формируется по принципу:

$$u_{sn+i} = \begin{cases} \bar{u}_{sn} + R_n\Delta_n, & \text{при } i = 0, \\ \bar{u}_{sn+i}, & \text{при } i = 1,2,\dots,s-1 \end{cases} \quad (5.11)$$

где

$$R_n = C_R\left(1 + \sum_{j=1}^{p_a}\left|y_{sn+l-j}\right| + \sum_{j=1}^{p_b-l}\left|\bar{u}_{sn-j}\right|\right). \quad (5.12)$$

Используя перепараметризацию, можно привести уравнение (5.10) к виду:

$$\bar{y}_{t+l} = \sum_{i=0}^{r-1}\theta_{i+1}\varphi_{t-i} + v_{t+l}, t = 0,1,\dots \quad (5.13)$$

где $\bar{y}_t = y_t/R_n$, $\varphi_t = u_t/R_n$, $\theta = A^{-1}B$ - параметр оценивания в полученной модели скользящего среднего, $r \le s$ – размерность вектора θ, которая выбирается из условия однозначности обратной замены.

$$A = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ a_1 & 1 & \dots & 0 & 0 \\ a_2 & a_1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{p_a} & \dots & a_1 & 1 \end{pmatrix}, B = \begin{pmatrix} b_l \\ \vdots \\ b_{p_b} \\ \vdots \\ 0 \end{pmatrix}.$$

Для перехода к модели линейной регрессии рассматриваются интервалы длины s, в каждом из которых на каждом шаге сумма из формулы (5.13) заменяется одним из её слагаемых, а остальные заносятся в помеху ξ.

$$\bar{y}_{sn+i+l} = \varphi_{sn}\theta_{i+1} + \xi_{n,i} \ , \tag{5.14}$$

$$\xi_{n,i} = \varphi_{sn+i}\theta_1 + \dots + \varphi_{sn+1}\theta_i + \varphi_{sn-1}\theta_{i+2} + \dots + \varphi_{sn+i-r+1}\theta_r + v_{sn+i+l} \, . \tag{5.15}$$

Несмотря на то, что эта новая помеха коррелирует со входами, в теоретических работах [3] доказывается, что алгоритмы для оценивания вектора неизвестных параметров можно использовать и в этом случае.

Рандомизированный алгоритм типа стохастической аппроксимации (РСА) для модели линейной регрессии со скалярными входами φ_n , выходами y_n , неизвестным параметром θ и помехой v_n

$$y_n = \varphi_n\theta + v_n, n = 1,2, \dots \tag{5.16}$$

в общем случае имеет вид:

$$\hat{\theta}_n = \hat{\theta}_{n-1} - \alpha_n \Gamma \Delta_n \left(\varphi_n \hat{\theta}_{n-1} - y_n\right). \tag{5.17}$$

где $\hat{\theta}_n$ – последовательность получаемых оценок, $\alpha_n \ge 0$ – неслучайная последовательность, определяющая шаг алгоритма, и Г – некоторая положительно определённая матрица. В качестве входов φ_n подаётся некоторая последовательность независимых случайных величин с известными математическими ожиданиями (пробное возмущение). Случайные величины $\Delta_n = \varphi_n - E(\varphi_n)$ получаются путём центрирования входов φ_n ($E(\cdot)$ обозначает математическое ожидание).

В теоретических работах [3] доказывается, что при выполнении некоторых ограничений на входы и помехи, предложенный алгоритм является состоятельным в том или ином смысле. Например, доказана сходимость последовательности оценок к точному значению неизвестного параметра с вероятностью единица, а также сходимость к нулю среднеквадратичной ошибки последовательности оценок. Более того, предлагаются оценки скорости сходимости алгоритма и доказывается его оптимальность при определённом выборе последовательности α_n и матрицы Г.

Применительно к задаче идентификации динамического объекта, формула (5.17) принимает вид:

$$\hat{\theta}_n = \hat{\theta}_{n-1} - \Gamma \frac{\beta_n^{-1}}{n} \Delta_n \left(\Delta_n \hat{\theta}_{n-1} - \frac{Y^n - \psi^n(\hat{\theta}_{n-1})}{R_n} \right). \quad (5.18)$$

4.3. Идентификация в задачах управления

Использование алгоритмов идентификации в задачах управления (в этом случае говорят об «адаптивном управлении») связано с некоторыми трудностями. Успешная идентификация системы осуществляется за счёт «раскачивания» управляющего воздействия, например, с использованием рандомизированных пробных сигналов. Однако регуляторы в задачах управления наоборот стараются погасить сторонние колебания (помехи) для достижения цели управления. Таким образом, задачи управления и идентификации являются противоположными по своей природе. На практике приходится некоторым образом разрешать данный конфликт.

Пусть необходимо решать задачу управления в условиях неизвестности точных значений параметров объекта управления. Можно некоторым образом получить оценку для значений этих параметров и затем использовать рассмотренные ранее регуляторы. Однако в данном случае использование даже стабилизирующего регулятора не гарантирует стабилизации процесса управления. А именно, последовательности входов и выходов могут расходиться на бесконечности из-за использования неточной оценки. В таком случае необходимо использовать некоторый дополнительный алгоритм для изменения оценок параметров так, чтобы гарантировать стабильность процесса управления. Такие алгоритмы называются алгоритмами стабилизации. Для практического применения алгоритм стабилизации должен за конечное число шагов обеспечить ограниченность входного и выходного сигнала заданным уровнем.

В задачах адаптивного управления алгоритм стабилизации запускается тогда, когда входы и выходы объекта управления за последние несколько шагов в целом превышают некоторое заданное значение (порог). Алгоритм стабилизации выполняет пересчёт оценок до возвращения значений входов и выходов в установленные рамки, а затем оценивание параметров снова начинает выполняться алгоритмом идентификации.

4.3.1. Алгоритм «модифицированная полоска»

В данной работе, в качестве примера, для стабилизации процесса адаптивного управления используется алгоритм «модифицированная полоска».

Введём обозначение:

$$\varphi_t = \begin{pmatrix} -y_{t-1} \\ \vdots \\ -y_{t-p_a} \\ u_{t-l} \\ \vdots \\ u_{t-p_b} \end{pmatrix}. \tag{5.19}$$

Тогда уравнение объекта управления (2.1) принимает вид:

$$y_t = \varphi_t^T \tau + v_t. \tag{5.20}$$

Алгоритм производит пересчёт оценок таким образом, чтобы за конечное число шагов обеспечить выполнение следующего неравенства:

$$|y_t - \varphi_t^T \hat{\tau}_t| \le 2C_v + \delta\|\varphi_t\|\,. \tag{5.21}$$

На каждом шаге алгоритма, вычисляется выражение под модулем в левой части неравенства с имеющейся оценкой $\hat{\tau}_{t-1}$:

$$\eta_t = y_t - \varphi_t^T \hat{\tau}_{t-1}\,. \tag{5.22}$$

Если для текущей оценки выполняется неравенство:

$$|\eta_t| > 2C_v + \delta\|\varphi_t\|\,, \tag{5.23}$$

то пересчёт оценок $\hat{\tau}_t$ производится по формуле:

$$\hat{\tau}_t = \hat{\tau}_{t-1} - \frac{\eta_t}{\|\varphi_t\|^2}\varphi_t \tag{5.24}$$

иначе, оценки не пересчитываются.

При выполнении предположений, описанных в [3], алгоритм «модифицированная полоска» сходится за конечное число шагов к одному из некоторого множества значений.

Однако такой алгоритм может уводить оценки далеко от истинного значения неизвестных параметров, что сильно вредит процессу идентификации. Можно улучшить взаимодействие алгоритмов идентификации и стабилизации путём задания множества допустимых значений неизвестных параметров. Стабилизирующий алгоритм, в таком случае, будет проектировать на него получаемые оценки. Тогда «модифицированная полоска» принимает вид:

$$\hat{\tau}_t = P_T\left(\hat{\tau}_{t-1} - \frac{\eta_t}{\|\varphi_t\|^2}\varphi_t\right), \tag{5.25}$$

где P_T – проектор в множество T допустимых значений вектора неизвестных параметров τ, сопоставляющий произвольному вектору из $\mathbb{R}^{p_a+p_b-l+1}$ ближайший к нему вектор из T.

Такая модификация не препятствует работе стабилизирующего алгоритма до тех пор, пока в множестве T находится истинное значение вектора неизвестных параметров. Это происходит из-за того, что оно, в частности, является одним из значений, к которому за конечное число шагов сходится стабилизирующий алгоритм.

4.4. Построение доверительных множеств

Алгоритмы построения доверительных множеств предназначены для нахождения множеств, в которых с некоторой заданной вероятностью находится точное значение оцениваемых параметров. Возможность построения таких множеств может быть использована для повышения качества управления и идентификации в задачах адаптивного управления. Доверительное множество можно использовать, например, в качестве множества T в алгоритме «модифицированная полоска».

Традиционно задача построения доверительных множеств решается применением асимптотических алгоритмов идентификации. Однако такие алгоритмы могут приводить к неверным результатам даже при большом количестве итераций, а при малом количестве асимптотический подход вообще не применим.

В данной работе представлены рандомизированные не асимптотические алгоритмы построения доверительных множеств, базирующиеся на общей схеме «исключения областей знакодоминирующих корреляций» (LSCR, Leave-out Sign-dominant Correlation Regions) [1, 2, 5]. Одним из параметров таких алгоритмов является количество шагов (интервал времени). Доверительное множество строится по заданному количеству входов. Из-за рандомизированности алгоритмов даже на одних и тех же входных данных при повторных запусках получаются разные результаты. Гарантируется работа алгоритмов при ограниченных, а в остальном произвольных, помехах. При этом само значение границы помех в алгоритмах не используется.

Основной идеей является построение некоторого семейства рандомизированных функций. Теоретически доказано, что маловероятно доминирование определённого знака на подмножествах области определения этих функций. Такие подмножества исключаются

из рассмотрения, а оставшиеся составляют доверительное множество для неизвестных параметров объекта управления.

4.4.1. Алгоритм Campi M.C.

Пусть N – количество входов, по которым строится доверительное множество.

Для каждого момента времени $t = 1, \dots, N$ сгенерируем пробные возмущения Δ_t, равные 1 или -1 с одинаковой вероятностью ½. Сложив пробное возмущение и управляющее воздействие u_t и подав сумму на вход объекта управления, получим для каждого момента времени t выходы y_t.

Будем рассматривать функции, предсказывающие значения выходов в зависимости от вектора значений параметров объекта управления:

$$\hat{y}_t(\theta) = \varphi_t^T \theta = \sum_{j=l}^{p_b+l-1} b_j u_{t-j} - \sum_{i=1}^{p_a} a_i y_{t-i}, t = 1, \dots, N \tag{5.26}$$

Определим ошибки такого предсказания как функции:

$$\epsilon_t(\theta) = y_t - \hat{y}_t(\theta), t = 1, \dots, N \tag{5.27}$$

Построим вектор-функции:

$$f_t(\theta) = \xi_t \epsilon_t(\theta), t = 1, \dots, N \tag{5.28}$$

где $p = p_a + p_b$,

$$\xi_t = (\Delta_{t-1} \quad \dots \quad \Delta_{t-p})^T . \tag{5.29}$$

Выберем натуральные числа $M > 2p$ и $q \in \left[1, \frac{M}{2p}\right]$. Данный выбор определяет вероятность P, с которой построенное доверительное множество будет содержать истинное значение вектора параметров объекта управления:

$$P = 1 - \frac{2q}{M}. \tag{5.30}$$

Сгенерируем набор из $M - 1$ различной бинарной стохастической строки длины N. Каждый элемент $h_{i,j}$ каждой строки $(h_{i,1}, \dots, h_{i,N})$ равен 0 или 1 с равной вероятностью ½.

Сформируем рандомизированный набор из $M - 1$ вектор-функции:

$$g_i(\theta) = \sum_{t=1}^{N} h_{i,t} \cdot f_t(\theta). \tag{5.31}$$

Обозначим за $g_i^k(\theta)$ k-ую компоненту вектор-функции $g_i(\theta)$.

Для каждого $k = 1, \dots, p$ построим область Θ^k, так что в ней по крайней мере q из функций $g_i^k(\theta)$ строго больше 0 и по крайней мере q из них строго меньше 0. Доверительным множеством Θ является пересечений всех областей Θ^k:

$$\Theta = \bigcap_{k=1}^{p} \Theta^k . \tag{5.32}$$

4.4.2. Алгоритм Граничина О.Н.

Этот алгоритм является модификацией рассмотренного выше алгоритма Campi M.C. [5]. Основным отличием является использование перепараметризации (5.13), которая позволяет уменьшить необходимый объём вычислений за счёт уменьшения размерности доверительной области [1, 2].

Выберем натуральное число $s \le p - l + 1$. В большинстве случаев выбирается s равное количеству неизвестных параметров объекта управления.

Пусть N – количество входов, по которым строится доверительное множество, и $N = s \cdot N_\Delta$.

Для каждого $n = 0, \dots, N_\Delta - 1$ сгенерируем пробные возмущения Δ_n, равные 1 или -1 с одинаковой вероятностью ½. Будем вносить пробное возмущение на каждом s-ом шаге, прибавляя его к управляющему воздействию u_t:

$$\bar{u}_{sn+i-l} = \begin{cases} u_{sn-l} + \Delta_n & \text{при } i = 0, \\ u_{sn+i-l} & \text{при } i = 1,2, \dots, s-1 \end{cases} \tag{5.33}$$

Подадим полученное управление $\bar{u}_t$ на вход объекту управления и получим для каждого момента времени t выходы y_t.

Будем рассматривать функции, предсказывающие значения выходов в зависимости от вектора значений параметров объекта управления:

$$\hat{y}_{sn+k-1}(\theta) = \Delta_n \theta_k + \sum_{i=1}^{k-1} \theta_{k-i}\, u_{sn+k-l-i}, k = 1, \dots, s . \tag{5.34}$$

Определим ошибки такого предсказания как функции:

$$\epsilon_t(\theta) = y_t - \hat{y}_t(\theta), t = 1, \dots, N. \tag{5.35}$$

Построим функции:

$$f_{sn+k-1}(\theta) = \Delta_n \epsilon_{sn+k-1}(\theta), k = 1, \dots, s . \tag{5.36}$$

Выберем натуральные числа $M > 2s$ и $q \in \left[1, \frac{M}{2s}\right]$. Данный выбор определяет вероятность $P = 1 - 2q/M$, с которой построенное доверительное множество будет содержать истинное значение вектора параметров объекта управления.

Сгенерируем набор из $M - 1$ различной бинарной стохастической строки длины N. Каждый элемент $h_{i,j}$ каждой строки $(h_{i,1}, \dots, h_{i,N})$ равен 0 или 1 с равной вероятностью ½.

Для каждого $k = 1, \dots, s$ сформируем рандомизированный набор из $M - 1$ функции:

$$g_i^k(\theta) = \sum_{n=0}^{N_\Delta - 1} h_{i,ns+k} \cdot f_{ns+k-1}(\theta), i = 1, \dots, M - 1. \quad (5.37)$$

Построим области Θ^k, так что в каждой из них по крайней мере q из функций $g_i^k(\theta)$ строго больше 0 и по крайней мере q из них строго меньше 0. Доверительным множеством Θ является пересечений всех областей Θ^k:

$$\Theta = \bigcap_{k=1}^{p} \Theta^k. \quad (5.38)$$

Полученное доверительное множество построено для введённого вектора параметров θ. Для получения доверительного множества исходного вектора параметров объекта управления τ, необходимо выполнить обратную замену, которая для данной параметризации определяется однозначно.

С помощью созданного модуля оценивания параметров было произведено сравнение алгоритмов построения доверительного множества Campi M.C. и Граничина О.Н. Полученные результаты наглядно продемонстрировали, что использование перепараметризации в алгоритме Граничина О.Н. позволяет существенно уменьшить объём необходимых вычислений для построения доверительного множества за счёт уменьшения размерности.

5. Заключение

В работе получены следующие результаты.

Был спроектирован и реализован на языке C++ расширяемый модуль адаптивного управления с возможностью выбора набора используемых алгоритмов управления, идентификации параметров и построения доверительных множеств и задания их начальных параметров. Для тестирования и сравнения работы алгоритмов был реализован симулятор линейных объектов управления не выше второго порядка. В симуляторе предоставлена возможность выбора параметров симулируемого ОУ и действующих на него неопределённостей (помех). Также поддержана возможность встраивания модуля адаптивного управления в реальную физическую систему путём замены симулятора на данные датчиков.

В демонстрационных целях были реализованы следующие алгоритмы:

- Алгоритм регулятора обратной связи
- Алгоритм стабилизирующего регулятора обратной связи
- Рандомизированный алгоритм идентификации параметров
- Стабилизирующий алгоритм «модифицированная полоска»
- Алгоритм Campi M.C. для построения доверительного множества
- Алгоритм Граничина О.Н. для построения доверительного множества

В дальнейшем возможно расширение системы дополнительными алгоритмами, в том числе алгоритмами для работы с линейными объектами управления выше второго порядка. Возможна модификация модуля для работы с нелинейными ОУ. Также планируется практическое встраивание модуля для управления реальной физической системой, например беспилотным летательным аппаратом (БПЛА) в проекте SmartFly [18].

6. Список литературы

1. Граничин О.Н. Неасимптотическое доверительное множество для параметров линейного объекта управления при произвольном внешнем возмущении // Автоматика и телемеханика. 2012. № 1. С. 24-35. http://www.math.spbu.ru/user/gran/papers/ait11_2.pdf
2. Граничин О.Н. Новые рандомизированные алгоритмы в управлении и обработке данных // Стохастическая оптимизация в информатике. Том 7. Вып 1. СПб.: Изд-во СПбГУ. 2011. С. 3-68. http://www.math.spbu.ru/user/gran/soi7/granichin.pdf
3. Граничин О.Н., Поляк Б.Т. Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах. М.: Наука. 2003. 291 с. http://www.math.spbu.ru/user/gran/BOOK_WIN.pdf
4. Граничин О.Н. Стохастическая оптимизация и системное программирование // Стохастическая оптимизация в информатике. Том 6. Вып 1. СПб.: Изд-во СПбГУ. 2010. С. 3-44. http://www.math.spbu.ru/user/gran/soi6/granichin6.pdf
5. Campi M.C., Weyer E. Non-Asymptotic Confidence Sets for the Parameters of Linear Transfer Functions // IEEE Transactions on Automatic Control. Vol. 55. No. 12. December 2010. P. 2708-2720.
6. Granichin O., Granichina O., Amelin K., Amelina N. Combined Procedure with Randomized Controls for the Parameters' Confidence Region of Linear Plant under External Arbitrary Noise // Preprint submitted to 51st IEEE Conference on Decision and Control. Received March 7, 2012.
7. Granichin O., Granichina O., Amelin K., Amelina N. Randomized Control for Small UAV Under Unknown Arbitrary Wind Disturbances // Preprint submitted to 2012 International Conference on Unmanned Aircraft Systems. Received February 29, 2012.
8. Платформа .NET http://www.microsoft.com/net
9. Язык Java http://www.oracle.com/technetwork/java/index.html
10. Control System Toolbox http://www.mathworks.com/products/control/

11. GNU Octave
    http://www.gnu.org/software/octave/
12. JIT-компиляция
    http://en.wikipedia.org/wiki/Just-in-time_compilation
13. Mathworks MATLAB
    http://www.mathworks.com/products/matlab/
14. Pure virtual функции в C++
    http://www.cplusplus.com/doc/tutorial/polymorphism/
15. Robust Control Toolbox
    http://www.mathworks.com/products/robust/
16. Scicos
    http://www.scicos.org/
17. Simulink
    http://www.mathworks.com/products/simulink/
18. SmartFly LLC. Multi-agent group of UAVs
    https://sites.google.com/site/smartflyllc/
19. Waterloo Maple
    http://www.maplesoft.com/products/Maple/index.aspx
20. Windows API Reference
    http://msdn.microsoft.com/en-us/library/aa383749
21. Wolfram Mathematica
    http://www.wolfram.com/mathematica/