УДК 512.6, 517.9, 519.6

КВАНТОВЫЙ СИМУЛЯТОР. Ч. 1: МОДЕЛИРОВАНИЕ СИСТЕМЫ «КАРЕТКА – МАЯТНИК» НА ОСНОВЕ КВАНТОВОГО ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Ульянов Сергей Викторович[1], Решетников Андрей Геннадьевич[2], Рябов Никита Владимирович[3]

[1]Доктор физико-математических наук, профессор;
ГБОУ ВО МО «Университет «Дубна»;
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: ulyanovsv@mail.ru.

[2]Кандидат технических наук, доцент;
ГБОУ ВО МО «Университет «Дубна»;
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: agreshetnikov@gmail.com.

[3]Аспирант;
ГБОУ ВО МО «Университет «Дубна»;
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: ryabov_nv95@mail.ru.

Статья открывает цикл работ по разработке квантового симулятора на примере системы «каретка – маятник» на основе квантового генетического алгоритма. Особое внимание уделено актуальности разработки симуляторов в целом и конкретно для образовательного процесса в интеллектуальной робототехнике. В статье описывается выбор средств разработки, процесс проектирования симулятора и параллельной частичной реализации одного из алгоритмов.

Ключевые слова: квантовые вычисления, квантовый генетический алгоритм, симулятор, *Python*, *Django*.

QUANTUM SIMULATOR. PT. 1: MODELING OF THE “CART - PENDULUM” SYSTEM BASED ON QUANTUM GENETIC ALGORITHM

Ulyanov Sergey[1], Reshetnikov Andrey[2], Ryabov Nikita[3]

[1]Doctor of Science in Physics and Mathematics, professor;
Dubna State University;
Institute of the system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: ulyanovsv@mail.ru.

[2]PhD, associate professor;
Dubna State University;
Institute of the system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: agreshetnikov@gmail.com.

[3]PhD student;
Dubna State University;
Institute of the system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: ryabov_nv95@mail.ru.

The article open a series of papers on the development of a quantum simulator using the example of "cart – pendulum" system based on a quantum genetic algorithm. The introduction part focuses on the relevance of the development of simulators in general and specifically for the educational process in the intelligent robotics. The article describes the choice of development tools, the simulation design process and the parallel partial implementation of one of the algorithms.

Keywords: quantum computing, quantum genetic algorithm, simulator, python, Django.

Введение

Интеллектуальная робототехника – одно из самых важных направлений в ИТ. Все сильнее, от прогресса в этой области зависит дальнейшее качество жизни и экономической устойчивости государства. Именно достижения, произведённые в этой области, позволяют лучше исследовать космос, люди, потерявшие конечности могут вести прежний образ жизни благодаря большому прогрессу в протезировании, разрабатываются когнитивно-интеллектуальные системы диагностики, обучения и адаптации детей-аутистов [1]. Перечислить все направления, на которые повлияла интеллектуальная робототехника, невозможно, их огромное множество, но также многие изобретения ещё не сделаны. Квантовый процессор и квантовые вычисления, являются одной из перспективнейшей сквозной квантовой технологией, которая окажет значительное влияние на данную область. Поэтому очень важно подготавливать хорошо квалифицированных специалистов в этой области знаний, заинтересовывать студентов и школьников связать свою дальнейшую жизнь с робототехникой.

Роботы становятся сложнее, цена оборудования высокая, цена ошибки и последующее повреждение могут обойтись в огромную сумму, которая ощутима не только для обычного любителя, но и даже для крупных компаний. Если ошибка при написании программного приложения выявится при помощи тестирования и не приведёт ни к каким проблемам, то логическая ошибка в программном обеспечении робота может привести к повреждениям техники. Кроме финансовой стороны на решение подобных проблем может уйти не малое количество человеко-часов для диагностирования поломки, разборки робота, сборке, заказу или производству новых деталей.

Необходимость серьезного оборудования в робототехнике – основная причина невысокой популярности данного направления у новичков. Не все школы и университеты могут позволить закупить себе оборудование для экспериментов или заключить соглашения с компаниями, которые могут его предоставить.

Симулятор – это имитация работы реального процесса. Первый симулятор появился в 1947 году. Его разработчиками были Томас Голдсмит и Эстле Рэй Манн. Это была небольшая игра, которая симулировала попадание ракеты в цель [2]. С развитием компьютерных технологий разработка симулятора даёт огромные возможности. Они применяются во многих сферах. В большинстве случае человек, обучающийся вождению, проведёт свои первые часы управления автомобилем именно на симуляторе. Пилоты Формулы-1 тренируются на симуляторах. Это позволяет им попробовать какие-то опасные манёвры без риска разбить машину. Хирурги проводят виртуальные операции для обучения.

В последние годы стали гораздо доступнее технологии виртуальной реальности и дополненной реальности. За дополненной реальностью большое будущее. Данная технология применяется как для развлекательных целей – на новых купюрах 200 и 2000 рублей, в зоопарке – для визуализации животного, которое обитает в данном вольере, так и для серьезных задач. Данная технология применяется в образовании – для визуализации солнечной системы, в медицине – визуальная помощь врачам прямо во время операции, в авиакомпания — для тренировок пилотов.

В России не так давно проводился конкурс в нескольких номинациях по управлению роботом Фёдором [3]. Это отличный пример применения симулятора. Реальная модель стоит огромные деньги, а благодаря симулятору студенты различных вузов смогли поучаствовать в конкурсе и получили хороший опыт разработки алгоритмов управления подобными системами. А организаторы конкурса получили свежие взгляды на решение различных проблем.

Технологически развитые страны увеличивают количество исследований в области квантовых вычислениях. Соединенные Штаты создают национальный подход к исследованиям и разработке

квантовой информации [4] под единым брендом. Так, например, планируют приложить наибольшие усилия на следующих направлениях:

- выявление и решение мегазадач: проблемах, решения которых позволят преобразовать научный и промышленных комплекс;
- обучение персонала;
- поощрение исследований частного сектора и подготовка механизмов связи между государственными и частными секторами;
- обеспечение инфраструктуры, необходимой для реализации научно-технических возможностей;
- продолжать развивать международное сотрудничество.

При разработке квантового генетического алгоритма в данной работе на макете перевернутого маятника [5] была обнаружены несколько проблем. Во-первых, тестирование написанного алгоритма на роботе занимает очень много времени. Во-вторых, можно столкнуться с неправильно работающим железом, при этом, довольно непросто выявить саму неисправность. В-третьих, генетический алгоритм – это подбор параметров, которые работают в конкретной ситуации лучше всего, но вполне распространённая ситуация, что эти параметры были очень плохие, что приводит к сложности настройки динамически неустойчивого объекта. Само падение может привести к повреждению различных частей. Если страховать реальный объект – значит влиять на среду функционирования и генетический алгоритм не справится с задачей адаптации. В программе будет создаваться впечатление, что робот ведёт себя хорошо, а значит это хорошая популяция и генетический алгоритм не будет отбрасывать проверяемое решение, а, наоборот, на основе него создавать новые решения. Это ведёт к тому, что нахождение самых оптимальных параметров наступает гораздо дольше, либо вообще не наступает. А релевантность этих данных может быть очень низкая.

Решение вышеперечисленных проблем является очень востребованной задачей. Поэтому было решено разработать симулятор перевёрнутого маятника.

Описание задачи

Первым делом, было необходимо понять в каком виде должен быть представлен симулятор, на каком языке лучше всего написать, как визуализировать полученные данные.

Основная цель разработки симулятора – образовательные цели. Студенты должны иметь возможность провести лабораторную работу, наблюдая поведение маятника при применении различных алгоритмов интеллектуального управления с разными параметрами: использование только *PID*-регулятора, добавление нечеткого регулятора в интеллектуальную систему управления (ИСУ), использование генетического алгоритма и нейронной сети, использование квантового генетического алгоритма. Особо заинтересованные студенты должны иметь возможность углубиться в техническую сторону реализации симулятора для реализации собственных алгоритмов, изучения уже реализованных или внесения различных улучшений в проект.

Симулятор интересен тем, что покрывает много областей, требуемых для его реализации. Также есть много различных путей развития: усовершенствование 2*d* модели или вообще реализация в 3*d*, управление маятником в режиме реального времени (изменение параметров маятника, добавление различных шумов), сделать симулятор более универсальным для простого создания симуляции других задач на основе подготовленного проекта.

Симулятор охватывает разные направления и может применяться не только на курсе робототехники.

Выбор средств разработки

Доступ к симулятору должен быть максимально простой, поэтому было решено разработать веб-приложение – клиент-серверная программа, в которой пользователь взаимодействует с сервером при помощи браузера. Оно состоит из нескольких частей: клиентская (или *frontend*) и серверная часть (*backend*).

Симулятор – это не типичное веб-приложение. Большая часть работы на серверной части – с математикой. Необходимо по множеству заданных и вычисляемых параметров рассчитывать положение каретки, угол наклона маятника в пространстве. По этому критерию выделяется один язык - *Python* – это лучший язык для выполнения каких-то научных работ. Содержит огромное количество библиотек, в т.ч. *NumPy* – фундаментальный пакет для научных исследований, который содержит в себе линейную алгебру, преобразование Фурье и многое другое [6], также понадобится функция линейного квадратичного регулятора, вычисляющего оптимальное управление, которое минимизирует квадратичную ошибку [7].

Все сгенерированные значения должны быть сохранены. Для этого понадобится СУБД. Поначалу приложение будет генерировать небольшое количество значений, но на парах, во время проведения лабораторных работ нагрузка может серьезно возрастать. СУБД для проекта выбрана *MySQL*, однако при реализации симулятора в режиме реального времени возможно скорости работы *MySQL* будет не хватать и для такой симуляции будет добавлена *Redis.*

Веб-фреймворк содержит набор библиотек, и обработчик при помощи которого можно создавать пользовательский код для реализации веб-приложения. Существуют несколько распространённых фреймворков: *Django, Flask, Tornado, Pyramid*. *Django* самый крупный веб-фреймворк, предоставляет множество пакетов, шаблонов, утилит из коробки, стремится помогать быстро помогать строить сложные веб-приложения. Имеет большое активное сообщество. Большинство веб-приложений, написанных на *python* используют *Django*. *Django* – самый универсальный вариант, который и будет применяться в этом проекте. В официальной документации есть огромное количество примеров, найти ответ можно практически на любой вопрос. У веб-фреймворка большое сообщество, где можно с уверенностью получить ответ на любой вопрос.

Django реализует подход *model-view-controller (MVC)* (или больше *MVT – model-view-template*) [8]. *Model*: содержит объектно-ориентированное отображение (*object relational mapping – ORM*), которое взаимодействует между моделями, описанными как класс в *python* и реляционной базой данных. *View*: система обработки *HTTP*-запросов с системой веб-шаблонов. *Template*: отображает данные в *HTML*.

В последующих частях будет ещё уделено внимание развёртыванию приложения на веб-сервере.

Проектирование симулятора перевёрнутого маятника

Проектирование приложение очень важная часть разработки. Именно на этом этапе формируется полная картина возможностей системы.

Для пользователя должен иметься доступ к созданию маятника, а также просмотру истории созданных маятников. Для создания маятника необходимо указать все параметры, используемые для работы алгоритма симулирования системы. Для просмотра истории маятников необходимо хранить сгенерированные координаты в базе данных.

Согласно методологии *MVT*, используемой в ранее выбранном веб-фреймворке *Django*, пользователь взаимодействует с шаблонами, которые связаны с представлениями, а те, в свою очередь, связаны с моделью. В симуляторе должно быть 5 страниц:

- главная страница с описанием проекта и наличием навигационного меню к остальным разделам;
- страница истории проведённых симуляций с поиском по различным критериями;
- страница создания новых симуляций с формой заполнения параметров;
- страница с описанием выполненных и будущих шагов развития проекта;
- страница просмотра самой симуляции.

Демонстрация пошаговой работы системы изображена на диаграмме последовательности (см. рис. 1)

Пользователь передаёт через шаблон параметры симулятора, которые получает представление. Представление записывает заданные параметры в БД и начинает генерировать координаты по полученным параметрами симулятора. Эти точки также сохраняются в БД, а

затем передаются шаблону. Визуализация симуляция интеллектуальной системы управления перевёрнутым маятником осуществляется при помощи *JavaScript*.

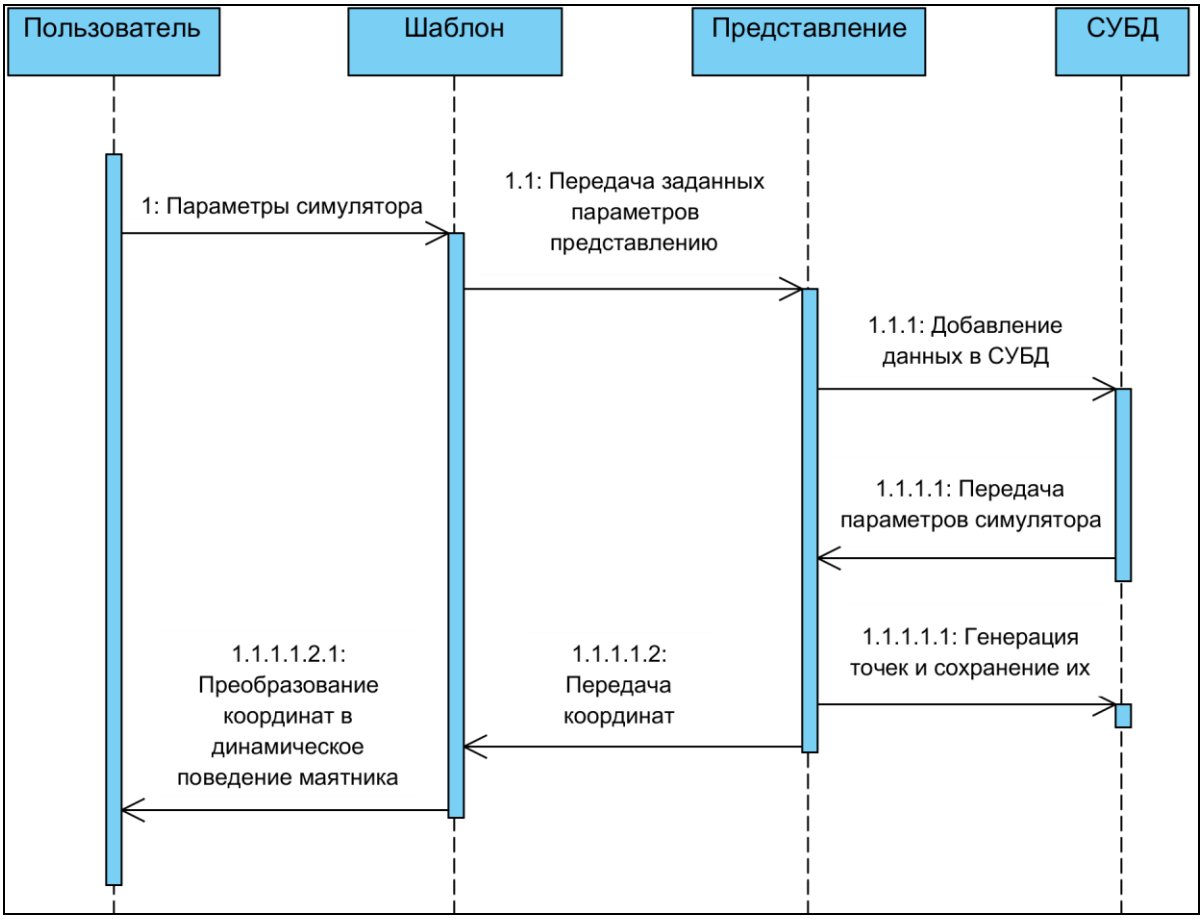

Рис. 1. Диаграмма последовательности работы системы

Веб-приложение, написанное на *Django,* состоит из нескольких частей:

Виртуальная окружающая среда. *Python* импортирует различные библиотеки, которые хранятся в папке, в которую он установлен. Но существует соглашение не устанавливать туда все библиотеки, т.к. могут возникать различные конфликты между разными приложениями, которые требуют, например, разные версии библиотеки. Чтобы решить эту проблему внутри каждого проекта необходимо создать *virtualenv*, куда и будут устанавливаться все библиотеки, использующиеся в текущем проекте.

Главный файл всего проекта – *manage.py*, который импортирует все настройки проекта, проверяет установлен ли *Django*.

Главная папка проекта – содержит файл с импортированием модуля *WSGI*. Файл с описание всех ссылок. В нём описываются именно ссылки на сами приложения. В нашем случае по умолчанию ссылка на *admin* и, добавленная разработчиком, *pendulums* – созданное приложение. Также содержит все настройки проекта:

- установленные приложение (туда входит уже готовые в веб-фреймворке приложение админ панели, аутентификации, сессии, сообщений, статический файлов), а также туда записываем, если разработчиком было создано своё приложение (в данном случае – *pendulums*);
- используемые шаблоны. По умолчанию используется *DjangoTemplates*, но в случае необходимости могут быть использовано, например, *Jinja2;*
- путь к файлу с *route* (ссылками приложения);
- путь к файлу с *WSGI*;

- подключаемый движок базы данных (в зависимости от того, какая СУБД используется в проекте), а также *host, port, user, database name* для подключения к СУБД;

- список различных проверок паролей;

- язык проекта;

- часовой пояс;

- путь к статическим файлам;

- различные пользовательские настройки по мере необходимости

Созданное пользовательское приложение (*pendulums*) содержит следующее:

- файл с описанием тестов проекта;

- файл со списком ссылок данного приложения. В данном случае по ссылке */pendulums* будет располагаться весь список уже созданных маятников с указанием их параметров, на любой из которых можно нажать и посмотреть его работу. По ссылке *pendulums/create* можно будет создать новый маятник, после создания пользователя отправит на страницу с визуализацией работы. По ссылке *pendulums/<int:pk>*, где *<int:pk>* – уникальный идентификатор маятника, по которому будет отправлен запрос к БД, из которой необходимо получить координаты маятника с таким *ID*;

- файл с описание форм проекта;

- файл с конфигурацией приложения, по умолчанию туда входит только название приложения;

- файл, в котором необходимо указать, какие модели будут доступны для изменения в административной панели;

- папка, содержащая статические файлы проекта. Это могут быть как стили страниц, так и различные изображения, подключаемые библиотеки;

- папка миграций. Миграция – это изменение БД. Если разработчик вносит изменения в модели, то после этого необходимо провести миграцию. При помощи встроенных средств изменить структуру СУБД, чтобы она стала одинаковая со структурой модели. В папке содержатся все изменения БД, что очень удобно, похоже на систему контроля версий;

- файл с описанием моделей проекта;

- файл с представлениями;

- папка с шаблонами (о последних 3-х пунктах более подробно будет рассказано дальше, т.к. они содержат всю основную логику приложения).

Файл с описанием моделей *models.py* содержит объектно-реляционное отображение базы данных в виде классов. В приложении это класс *Pendulums*, который содержит все изменяемые параметры маятника и несколько параметров окружающей среды:

- шаг времени (*dt*) – при рисовании от этого показателя зависит количество точек в секунду. Отметим, что при очень большом количестве точек ожидание после создания маятника может несколько затянуться. Выставлять значения меньше 0.01 не имеет смысла, т.к. это уже будет 100 *fps*, увеличение его ещё больше незаметно на мониторе, а после 200 *fps* это влияет только на скорость вывода графического изображения (она будет гораздо ниже);

- время моделирования (*time*) – сколько в секундах будет длиться моделирование положения маятника. Без сильных шумов вероятность того, что маятник перестанет работать нулевая, поэтому нет необходимости моделировать более 30 сек. С добавлением симуляции в реальном времени моделирование может стать бесконечным по времени, однако в такой ситуации, при сохранении возможности просматривать историю БД может очень быстро увеличиваться в размерах;

- масса каретки и маятника (*M*);

- масса маятника (*m*);

- постоянная момента двигателя (*Km*);

- передаточное число (*Kg*);
- сопротивление (*R*);
- радиус движения (*r*);
- высота маятника (*l*);
- инерция (*I*);
- гравитация (*g*);
- напряжение (*Vsat*);
- алгоритм управления (*PID*, генетический алгоритм, квантовый генетический алгоритм);
- дата создания.

Класс *Coordinates*, который включает атрибут положения центра каретки, а также угла отклонения маятника, из которого при помощи *sin* и *cos* можно получить в градусах отклонение по оси *X* и *Y*. Этот класс связан с классом *Pendulums* при помощи внешнего ключа *Pendulums.id = Coordinates.pendulum_id*.

В файле с описание представлений *view.py* происходит вся основная логика программы. В нём происходит получение параметров от пользователя, вычисления поведения маятника по этим параметрам, запись их в БД и передача этих параметров шаблонам. Шаблон в свою очередь выводит переданные представлением параметры на экран в определённом формате. В файле представлений было описано несколько классов.

IndexView, который сработает при запросе от пользователя вернуть данные по ссылке */pendulums*. Класс возвращает 10 последних смоделированных маятников, передаёт эти данные шаблону с названием *index.html*. В шаблоне в виде табличке выводятся все данные, полученные от представления (см. рис. 2.).

DetailView, который срабатывает при запросе от пользователя вернуть данные по ссылке */pendulums/id*, где *id* – уникальный идентификатор маятника. Представление делать запрос к БД, от которой получает список всей информации о маятнике с заданным *id*, в том числе и все данные из связанных таблиц, что очень удобно. Дальше эти данные получает шаблон *detail.html*, в котором создаётся *canvas*, в котором и осуществляется рисование. При помощи *JavaScript* происходит подготовка данных, полученных из БД, задаётся размер маятника и каретки, скорость рисования (*dt* * 1000). По данным из БД находится *sin* и *cos*, по которым определяются точки *x* и *y* в координатах *canvas*, которые и используются для рисования. Для рисования создана функция *draw*, которая выполняется столько раз, сколько точек имеет этот маятник и выполняется с заданной скоростью рисования. Алгоритм рисования следующий: из центральной каретки точки влево на заданное количество – первая координата, вправо – вторая. У маятника одна координата находится тоже в центре каретки, а вторая находится по данным из БД. В зависимости от заданных параметров веб-приложение генерирует от нескольких тысяч точек. Все они последовательно обрисовываются. Несколько положений маятника показано на рис. 3.

Create new pendulum

#	date	dt	time	mass	mass of pendulum	torgue constant	gear ratio	resistance	drive radius	length pendulum	inertia	gravity	voltage
43	29/05/2018 23:52	0.050	10.00	0.600	0.300	2.00	0.010	6.00	0.010	0.300	0.0060	9.81	20.00
42	20/05/2018 20:55	0.004	10.00	0.600	0.300	2.00	0.010	6.00	0.010	0.300	0.0060	9.81	20.00
41	20/05/2018 20:55	0.003	10.00	0.600	0.300	2.00	0.010	6.00	0.010	0.300	0.0060	9.81	20.00
40	20/05/2018 20:51	0.005	10.00	0.600	0.300	2.00	0.010	6.00	0.010	0.300	0.0060	9.81	20.00
39	20/05/2018 20:50	0.001	10.00	0.600	0.300	2.00	0.010	6.00	0.010	0.300	0.0060	9.81	20.00
38	19/05/2018 23:46	0.050	10.00	0.600	0.300	2.00	0.010	6.00	0.010	0.300	0.0060	9.81	20.00
37	19/05/2018 22:43	0.010	10.00	0.600	0.300	2.00	0.010	6.00	0.010	0.300	0.0060	9.81	20.00
36	19/05/2018 22:42	0.015	10.00	0.600	0.300	2.00	0.010	6.00	0.010	0.300	0.0060	9.81	20.00

Рис. 2. История созданных симуляций

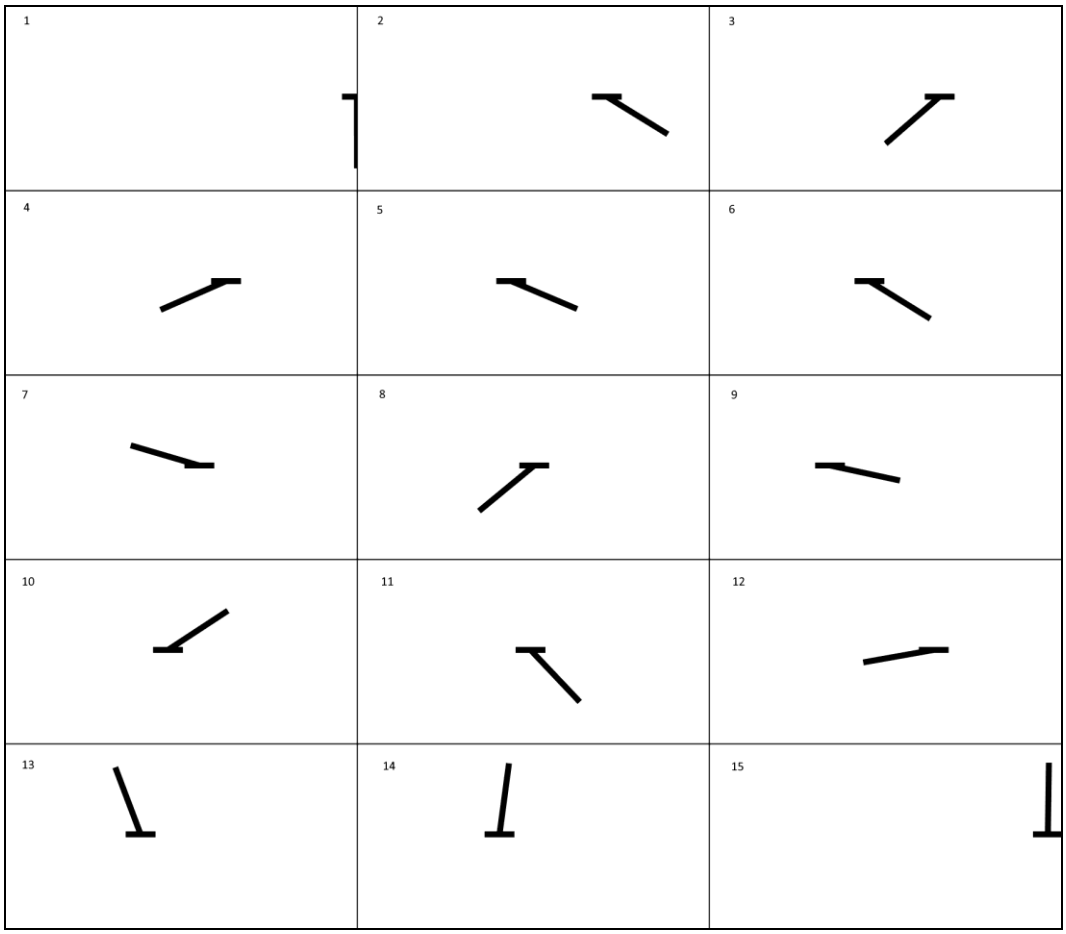

Рис. 3. Визуализация выполнения алгоритма

Каретка начинает раскачивать маятник и удерживает его в верхнем вертикальном положении.

PendulumView – класс, в которым находится вся основная логика программы. Шаблон создаёт форму, которую пользователь может изменить, чтобы смоделировать маятник с другими параметрами, после нажатия на кнопку сохранить данные получает этот класс. В нём создаётся объект класса *PendulumGen*, в котором и происходит генерирования точек расположения маятника в пространстве при помощи математических формул.

Заключение

Данная работа, представляет промежуточный результат, по созданию виртуальной лаборатории интеллектуальной робототехники. Описаны требования к интерфейсу обучающегося. Спроектирована и разработана инфраструктура для реализации алгоритмов интеллектуального управления. Проведена апробация результатов, на примере ПИД и ГА регулятора.

Результаты работы могут применяться при проведении семинарских и дистанционных занятий по различным курсам. Планируется реализация управления с применением нечеткой нейронной сети, квантового нечеткого вывода и квантового генетического алгоритма [9].

Список литературы

1. Reshetnikov A., Ulyanov S., Mamaeva A. Cognitive fuzzy control system: examples of intelligent control and therapy of the autism applications // Системный анализ в науке и образовании: сетевое научное издание. — 2017. — №4. — [Электронный ресурс]. URL: http://sanse.ru/download/303.
2. PONG-story. — 2013. — [Электронный ресурс]. URL: http://www.pong-story.com/intro.htm.
3. Открытый конкурс лучших решений по разработке программного обеспечения автономного управления антропоморфным роботом на основе функциональной 3*D*-модели в среде симулятора. — 2018. — [Электронный ресурс]. URL: http://fpi.gov.ru/activities/konkurs/robot.
4. Ted Wackler, Chloe Kontos. National strategic overview for quantum information science. September 2018.
5. Решетников А.Г., Керимов Т.А., Ульянов С.В. Применение технологии проектирования интеллектуальных систем управления на основе мягких вычислений (на примере перевернутого маятника) // Системный анализ в науке и образовании: сетевое научное издание. — 2015. — №1. — [Электронный ресурс]. URL: http://sanse.ru/download/231.
6. NumPy documentations. — 2018. — [Электронный ресурс]. URL: http://www.numpy.org.
7. Richard Murray, Rene van Paassen. Slycot library. — 2017. — [Электронный ресурс]. URL: https://github.com/python-control/Slycot.
8. Django documentation. — 2018. — [Электронный ресурс]. URL: https://docs.djangoproject.com/en/2.0/.
9. Ульянов С.В., Решетников А.Г., Рябов Н.В. Квантовый генетический алгоритм: выбор типа и вида корреляции в квантовом нечетком выводе // Системный анализ в науке и образовании: сетевое научное издание. — 2018. — №2. — [Электронный ресурс]. URL: http://sanse.ru/download/316.