- [2] B. Hollifield, "A High Performance HMI. Better Graphics for Operations Effectiveness", 2012 ISA Water & Wastewater and Automatic Controls Symposium, Orlando, 2012, pp. 1-27.
- [3] Situational Awareness – The Next Leap in Industrial Human Machine Interface Design. Whitepaper. Schneider Electric Software, LLC., 2015. [Online]. Available: <https://www.controlglobal.com/assets/15WPpdf/150813-Schneider-SituationalAwareness.pdf>. Accessed on: Nov. 19, 2020.
- [4] Next-generation SCADA high performance human-machine interfaces. Presentation. Arcadis, 2018. [Online]. Available: https://onewaterohio.org/docs/NextGenSCADA_HPHMI_Final_020817_-_OneWater.pdf. Accessed on: Nov. 11, 2020.
- [5] D. Reising, and P. Bullemer, "Operator situation awareness & delivering ASM-compliant high performance HMIs: Failure modes and success factors", American Fuel & Petrochemical Manufacturers Annual Meeting, San Antonio, 2015, pp. 1-10.
- [6] ISA101: What It Is and Why It Is Needed – Overview and Current Activities. Presentation. The International Society of Automation, 2017. [Online]. Available: <https://isaneworleans.files.wordpress.com/2017/08/isa-new-orleans-sept-2017-meeting-isa101-presentation.pdf>. Accessed on: Nov. 14, 2020.
- [7] Using ISA-101 & High Performance HMIs for More Effective Operations. Presentation. WEAO Intelligent Wastewater Systems Seminar, 2017. [Online]. Available: http://www.grahamnasby.com/files_publications/NasbyG_2017_High_PerformanceHMIs_IntelligentWastewaterSeminar_WEAO_sept14-2017_slides-public.pdf. Accessed on: Nov. 11, 2020.
- [8] ISA-101.01-2015 - Human Machine Interfaces for Process Automation Systems. Presentation. The International Society of Automation, 2019. [Online]. Available: https://www.instm.org/Portals/0/ISA101%202019%20Tech%20Lecture%20%281%29_1_1.pdf. Accessed on: Nov. 18, 2020.
- [9] ISA-101 Standard Addresses the HMI Lifecycle. Advisory group blog of industry trends. The International Society of Automation, 2018. [Online]. Available: <https://www.arcweb.com/blog/isa-101-standard-addresses-hmi-lifecycle>. Accessed on: Nov. 16, 2020.
- [10] ISA-101: Toward a More Effective HMI Strategy. A subsidiary of the International Society of Automation, 2015. [Online]. Available: <https://www.automation.com/en-us/articles/2015-1/isa-101-toward-a-more-effective-hmi-strategy>. Accessed on: Nov. 19, 2020.

УДК 681.51

РОЗРОБКА АВТОНОМНИХ WEB-СЕРВІСІВ ДЛЯ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ ДИНАМІЧНИХ СИСТЕМ В РЕАЛЬНОМУ ЧАСІ НА БАЗІ МОДЕЛЕЙ MATLAB SIMULINK

Стопакевич А. О.¹, Стопакевич О. А.², Тігарєв А. М.³, Тігарєва Т. Г.⁴

^{1,3} Одеська національна академія зв'язку імені О.С. Попова, м. Одеса, Україна

² Одеський національний політехнічний університет, м. Одеса, Україна

⁴ Одеська державна академія будівництва та архітектури, м. Одеса, Україна

ORCID: ¹<https://orcid.org/0000-0003-1719-9071>, ²<https://orcid.org/0000-0002-8318-6853>,

³<https://orcid.org/0000-0002-3394-2526>, ⁴<https://orcid.org/0000-0003-4453-279X>

E-mail: ¹stopakevich@gmail.com

Copyright © 2020 by author and the journal "Automation of technological and business – processes".

This work is licensed under the Creative Commons Attribution International License (CC BY).

<http://creativecommons.org/licenses/by/4.0>

DOI: <https://doi.org/10.15673/atbp.v12i4.1931>

Анотація. У статті розглянута задача розробки автономного web-сервісу на базі Simulink моделей. Запропоновано технологію, яка полягає в розробці комплексу з двох пов'язаних окремих додатків: консольного застосування на мові Cі на базі коду, який генерує Simulink Coder і додатки на мові Python, яке реалізує функціонал web-сервісу. Принциповим недоліком вже запропонованих рішень по розробці web-сервісів і web-додатків на базі Simulink моделей є те, що Simulink модель виконується за допомогою Matlab. Через це виникають нерозв'язні проблеми, а саме: надмірність і ресурсовитратність, ненадійність виконання моделі Simulink в реальному часі, складність взаємодії з моделлю під час виконання, фактична однопоточність Matlab. Останнє означає, що в один момент часу в пакеті може виконуватися тільки одна модель Simulink. Головною відмінністю запропонованої

технології є те, що розроблений web-сервіс, на відміну від раніше запропонованих рішень, є автономним, тобто не вимагає для своєї роботи Matlab. Це дозволяє досягти суттєвої економії ресурсів і підвищує надійність роботи в цілому. Переваги пропонованої технології: швидка розробка; перенесення, тобто можливість працювати як в операційних системах загального призначення, реальному часу і в операційних системах для одноплатних комп'ютерів; низькі вимоги до ресурсів; стандартний програмний інтерфейс віддаленого доступу. Сфера застосування технології: розробка «цифрових двійників» для налагодження компонентів систем автоматизації, швидка розробка регуляторів складної структури (за умови, що вона ефективно може бути представлена у вигляді окремих блоків), дистанційне навчання. Подальший напрямок досліджень планується в області удосконалення запропонованого підходу в області створення цифрових двійників для складних технологічних об'єктів, розробки регуляторів з неklasичними підходами в області відновлення змінних стану для об'єктів з істотними перехресними зв'язками, а також в галузі дистанційного навчання в рамках LMS платформи.

Abstract. The article is devoted to the task of developing an autonomous web-service based on Simulink models. The technology, which consists in the development of a complex of two linked separate applications: console application in C language on the basis of code, which generates Simulink Coder and applications in Python language, which implements the functionality of web-service, is proposed. The principal disadvantage of the already proposed solutions for the development of web services and web applications based on Simulink models is that the Simulink model is performed using Matlab. Because of this there are unsolvable problems, namely: redundancy and resource consumption, unreliability of Simulink model execution in real time, complexity of interaction with the model at runtime, the actual single threading Matlab. The latter means that only one Simulink model can run in a package at a time. The main difference of the proposed technology is that the developed web-service, in contrast to previously proposed solutions, is standalone, which means it does not require Matlab for its work. This allows the achievement of significant savings in resources and increases the reliability of the system as a whole. Advantages of the proposed technology: 1) fast development; 2) portability, i.e. possibility to work both in general-purpose OS, and in real time OS and in OS for single board computers; 3) low requirements to resources; 4) standard software interface for remote access. Scope of application of the technology: development of "digital twins" for debugging of automation system components, fast development of complex structure regulators (assuming that it can be effectively represented as separate blocks), distance learning.

Ключові слова: Matlab, Simulink, web-сервіс, автономність, цифровий двійник, відладка, регулятор, блок, мова C, Python

Keywords: Matlab, Simulink, web-service, autonomous, digital twin, debugging, controller, block, C language, Python

Введение

Simulink – один из самых известных пакетов Matlab, который позволяет представлять динамические системы в удобной для восприятия человека форме и проводить их имитационное моделирование. Кроме того, пакет имеет и дополнительные функции, среди которых обратим внимание на функцию генерации кода для выполнения моделируемой системы в среде реального времени. Проблемой программ, которые генерируются Matlab Simulink Coder из моделей Simulink является то, что они требуют значительной доработки для обеспечения возможности взаимодействия с ними в реальном времени с помощью стандартных программных интерфейсов. Программы, которые генерируются Matlab Compiler для использования на Desktop или с пакетом Web App для реализации web-приложений частично лишены указанных недостатков, но, по сути, требуют для своего выполнения Matlab (или его ограниченную версию, поставляемую в бесплатном Runtime). Такие программы требуют избыточное количество ресурсов, долго загружаются и могут быть запущены в ограниченном количестве программных окружений.

Указанные обстоятельства вынуждают разработчиков искать пути выхода из ситуации. В работе предлагается технология, которая позволяет быстро преобразовать модель Simulink в автономную динамическую систему, работающую в реальном времени в качестве web-сервиса в большом количестве программных окружений, в т.ч. в окружениях с ограниченными ресурсами. Такие системы полезны для отладки систем автоматизации в реальном времени как аналоги реальных объектов управления (т.н. цифровые двойники), для реализации управляющих алгоритмов, а также в образовательных целях – для реализации лабораторных работ с возможностью параллельного выполнения их студентами удаленно.

Анализ литературных источников

Задача разработки удаленного web-интерфейса для моделирования динамических систем при проведении лабораторных работ рассмотрена в работе [1]. Студент выбирает среди списка модель, которая реализует управляющее устройство, вводит его параметры и запускает на управление реальным двигателем в реальном времени. С помощью камеры студент может наблюдать за поведением объекта управления. После выполнения работы строится график. Технически это реализуется так: на сервере с платой АЦП устанавливается Matlab и web-сервер Apache. Для каждой модели из списка разрабатывается отдельный скрипт для ввода данных. Задача скрипта – сформировать mat файл и передать его по стандартным интерфейсам межпрограммного взаимодействия в Matlab, который вызовет функцию sim для одной из известных моделей с данными из сформированного файла. После выполнения модели в течении заданного времени моделирования результаты возвращаются PHP программе и отображаются пользователю.

В каждый момент времени с системой может работать только один студент. Задача составления схемы модели для удаленного моделирования рассматривается в работе [2]. Редактор с web-интерфейсом позволяет составить схему моделирования, которую можно затем экспортировать в форматы схем Simulink и Xcos. Указанные две идеи и разработка редактора схем с web-интерфейсом, который генерирует Simulink модели и выполняет их в Matlab, объединены в работе [3]. В качестве серверного языка используется PHP. Задача взаимодействия с выполняемой Simulink-моделью в реальном времени решается в работе [4]. Поскольку получить состояние модели при её выполнении в Matlab без остановки моделирования невозможно, то разработан специальный MEX блок для Simulink, который реализует информационный обмен в реальном времени. Для информационного обмена используются механизмы сокетов и технология построения XML web-сервисов ASP.NET на базе Microsoft WCF. Такой же подход предложен в более современной работе [5], однако используется язык Python и web-сервисы JSON. Web-сервер реализуется прямо в Matlab в работе [6]. Web-сервер приостанавливает моделирование, обновляет параметры модели при получении данных от удаленного пользователя и запускает моделирование снова, что позволяет избежать необходимости разрабатывать отдельный блок для коммуникации модели, однако является более ресурсозатратным.

Все указанные работы ориентируются на то, что Simulink модель выполняется с помощью Matlab. Из-за этого возникают неразрешимые проблемы, а именно: избыточность и ресурсозатратность, ненадежность выполнения модели Simulink в реальном времени, сложность взаимодействия с моделью во время выполнения, принципиальная однопоточность Matlab. Последнее означает, что в один момент времени в пакете может выполняться только одна модель Simulink.

Таким образом, реализовать полноценную многопользовательскую клиент-серверную систему имитационного моделирования динамических систем в реальном времени с таким подходом невозможно. Кроме того, идея разработки альтернативного редактора для моделей Simulink в предложенных решениях сводится к тому, что генерируется файл Simulink-модели программой, а не средствами Matlab. Практически, это может приводить к проблемам совместимости и ошибкам, т.к. форматы Simulink файлов меняются каждый выпуск Matlab. Также не рассматривается задача отображения в web-интерфейсе состояния блоков в реальном времени, что во многом нивелирует преимущество наглядности схематического моделирования перед моделированием программным.

Целью статьи является разработка технологии проектирования автономного web-сервиса, который не требует для своей работы Matlab. Кроме того, программное решение должно соответствовать следующим требованиям: 1) быстрая разработка; 2) переносимость, т.е. возможность работать как в ОС общего назначения, в ОС реального времени и в ОС для одноплатных компьютеров; 3) низкие требования к ресурсам; 4) стандартный программный интерфейс удаленного доступа.

Выбор средств и стратегии для реализации требований

С помощью инструментов Matlab возможно преобразовать большинство Simulink-моделей в код на языках Си и Си++. Собственно, сам Simulink этим активно пользуется, применяя генерацию и компиляцию кода для отдельных блоков при симуляции. Более того, режим ускорения является, по сути, выполнением скомпилированного кода программы на языке Си, сгенерированного по модели. Поэтому первый этап состоит в генерации кода на одном из этих языков. Хотя языки Си с Си++ формально являются кроссплатформенными, на практике программы, которые реализуют работу с потоками, сетью, драйверами будут сильно отличаться по причине несовместимых программных интерфейсов различных ОС. Поэтому, доработка сгенерированного кода программы до полноценного web-сервиса на языках Си/Си++, является задачей достаточно затратной, следовательно, не соответствует требованию 1. Возможно также решение в виде реализации дополнительной библиотеки или модуля к web-серверам Apache или nginx, что является более простым, но не соответствует требованию 2. Кроме того, программные интерфейсы web-серверов постоянно меняются и в них выявляются ошибки уязвимости, что вынудит постоянно обновлять код программы. Поэтому предлагается для реализации функционала web-сервиса написать дополнительную программу на языке программирования, который оптимизирован для этой цели

Среди языков, оптимизированных для разработки web-сервисов, следует выделить C#, Java, Go и Python. Первые два языка требуют наличия виртуальной машины, которая не доступна для всех платформ, поэтому их использование не соответствует требованию 2. По большому счету соответствию требованию 1 язык Python выигрывает, т.к. сам по себе язык Python более простой чем Go, функциональный и расширяемый за счет большого количества готовых библиотек. Кроме того, программы на Python не требуют компиляции для выполнения. С другой стороны язык Go, который по сути является более развитым языком Си, оптимизированным не только для системного программирования, а и для web-приложений выигрывает по производительности полученных скомпилированных программ. Хотя разница по скорости между программами, написанными на Python и Go, может отличаться в десятки раз, в рассмотренном случае «узким местом» является программа, которая выполняет модель в реальном времени. Поэтому при больших нагрузках система будет отказывать отнюдь не из-за медленного web-сервиса, а из-за невозможности параллельного выполнения множества динамических моделей в реальном времени. Следовательно, остановим свой выбор на Python.

В качестве механизма взаимодействия между программой на Python и программой на Си/Си++ можно рассмотреть следующие механизмы: 1 – реализации библиотеки на Си/Си++ с последующим вызовом её из программы на Python; 2 – обмена с помощью сокетов между приложениями; 3 – одностороннего информационного обмена с помощью интерфейса командной строки. С точки зрения требования 2 стоит выбрать интерфейс командной строки. Хотя

первый механизм в общем случае является наиболее корректным, однако для Python он нецелесообразен по причине использования языком Python блокировок GIL в многопоточных программах, которая вызывает проблемы со скоростью и надежностью работы программы. Второй механизм делает программу на Си сильно зависимой от платформы. Поэтому, следует выбрать третий механизм, т.к. Python имеет библиотеки для exрест-подобного управления интерактивными приложениями, работающими в режиме командной строки, как для POSIX-совместимых систем, так и для Windows (для Go, например, на данный момент такой библиотеки нет).

Поскольку объектно-ориентированная функциональность в данном случае не требуется и снижает скорость работы, то для генерации кода выберем язык Си. Последующая доработка кода будет вестись в соответствии со спецификацией языка C99.

Существуют две основные спецификации web-сервисов: REST и SOAP. Первая более популярна в последнее время и больше соответствует требованиям 3 и 4. В качестве языка сообщений в RESTful сервисах используются JSON-схемы данных, передающиеся по протоколу HTTP.

Структурная схема предлагаемого программного решения для реализации автономного web-сервиса показана на рисунке 1.

Рис. 1 – Структурная схема предлагаемого программного решения
Fig. 1 – Schematic diagram of the proposed program solution

Этапы разработки автономного web-сервиса

Предлагаемый нами алгоритм выполняется в 6 этапов:

- 1) Преобразование имеющейся математической модели в соответствии с требованиями Simulink Coder;
- 2) Генерация или ручная сборка Simulink модели;
- 3) Реализация решателя в реальном времени в виде консольного приложения на базе программного кода, сгенерированного по Simulink-модели;
- 4) Реализация RESTful web-сервиса на языке Python;
- 5) Реализация интерфейса для удаленной работы с web-сервисом;
- 6) Интеграция RESTful web-сервиса в имеющееся программно-техническое окружение.

Пример для иллюстрации

В качестве иллюстративного примера рассмотрим задачу моделирования объекта управления в программном обеспечении системы автоматизации по имеющейся линейной математической модели динамики.

Модель динамики объекта управления зададим в виде матрицы передаточных функций:

$$W(s) = \begin{bmatrix} W_{11}(s) & W_{12}(s) \\ W_{21}(s) & W_{22}(s) \end{bmatrix} = \begin{bmatrix} \frac{11}{11s+1}e^{-100s} & \frac{12}{33s+1}e^{-200s} \\ \frac{21}{22s+1}e^{-300s} & \frac{22}{44s+1}e^{-400s} \end{bmatrix},$$

где s – оператор Лапласа.

Этап 1 – Адаптация к требованиям Simulink Coder. Требования к моделям в Simulink Coder следующие:

1. Модель должна быть реализована с одним из решателей в дискретном времени. Это означает что ряд блоков не применим, в частности все блоки пакета Simscape. В документации по каждому пакету Matlab есть сводная таблица совместимости.

2. Модель может содержать Matlab код только из подмножества Embedded Matlab.

3. Модель не должна содержать МЕХ и прочие платформенно зависимые блоки, в том числе, отвечающие за доступ к основным программным интерфейсам.

Все параметры модели, которые пользователь должен менять или за которыми следует наблюдать надо четко выделить, поскольку при генерации кода Matlab склонен сокращать код, отвечающий за расчеты.

Также, по возможности, с целью увеличения наглядности модели желательно заменять блоки с комплексом параметров на элементарные блоки Simulink.

С целью упрощения представим передаточные функции (ПФ) в виде дифференциальных уравнений вида:

$$\dot{x}(t) = \frac{-1}{T} \cdot x(t) + \frac{k}{T} \cdot u(t - \tau),$$

где x – переменная состояния, T – постоянная времени звена ПФ, k – коэффициент передачи звена ПФ, τ – время запаздывания.

Кроме того, пользователь должен видеть значения выходов системы управления (y_1, y_2), а также выходы основных блоков модели (интеграторов, запаздываний и блоков вычисления отношения k/T)

В качестве решателя выберем ode5 – дискретный аналог используемого Matlab по умолчанию решателя ode45. Шаг дискретности выберем равным – 0,2.

Этап 2 – Генерация или ручная сборка Simulink модели. Стандартным способом разработки схемы математической модели в Simulink является её ручная сборка. Для последующей генерации каждому блоку необходимо дать понятный идентификатор латинскими буквами с целью последующей локализации отвечающего за него кода.

Задача генерации математической модели стандартными средствами Matlab решена на данный момент неудовлетворительно. Для добавления блоков на схему с заданными параметрами используется функция `add_block`, а для связывания входов между блоками – `add_line`. Функционал связанный с автоматическим размещением блоков фактически отсутствует, что требует при генерации вручную вводить координаты блоков. Это сильно усложняет проблему.

Интересное решение указанной проблемы найдено в библиотеке Matsim [7]. Эта библиотека имеет две уникальные возможности: построение системы Simulink из уравнений и автоматическое размещение элементов схемы. Последнее реализуется следующим образом: измеряются размеры блоков Simulink схемы, формируется задача на языке DOT, которая решается путем поиска оптимального размещения блоков с минимумом пересечений с учетом их порядка и приоритета, решение этой задачи переводится в координаты блоков Simulink. Схематическое представление решения задачи для элемента (1,1) матрицы передаточной функции проведенное с помощью интерпретатора Graphviz языка DOT, показано на рисунке 2.

Рис. 2 – Оптимальное расположение блоков, рассчитанное пакетом Graphviz

Fig. 2 – Optimal location of blocks, calculated by Graphviz package

При представлении результата схематически каждый блок содержит три столбца. Первый содержит номер(а) входов, второй – уникальный числовой идентификатор блока, третий – номер(а) выходов. Для входных блоков первый столбец опускается, для выходных опускается последний столбец.

Сигнал управления (блок 8) суммируется (блок 2) с сигналом возмущения ко входу (блок 9). Далее величина умножается на результат вычисления $\frac{k}{T}$ (блок 5). Результат вычисления блока 5 суммируется в блоке 1 с результатом

вычисления блока 1 ($-\frac{1}{T}$). Далее сигнал проходит через интегратор (блок 6), звено запаздывания (блок 7), суммируется в блоке (4) и записывается в выходную переменную y_1 (блок 10).

Этап 3 – Реализация решателя в реальном времени в виде консольного приложения на базе программного кода, сгенерированного по Simulink-модели. Для генерации кода в Matlab необходимо выбрать цель, язык программирования и компилятор. В предлагаемом алгоритме следует выбрать цель ert (встраиваемые системы) и язык Си.

Необходимые настройки: генерировать только код, динамическое размещение памяти для инициализации модели, представлять входы и выходы корневого уровня в качестве структуры данных модели, поведение параметров – настраиваемые, оставлять порядок операндов в выражении неизменным. Рекомендуется также задать цели: эффективность и возможность трассировки.

После генерации Simulink Coder формирует папку с программой, которая проводит бесконечный вычислительный цикл без таймера. В отличие от программ, сгенерированных обычным Matlab Coder, программа содержит обработчики цикла, что позволяет модифицировать код для симуляции в реальном времени.

Ключевой файл сгенерированного кода – это файл с именем модели Simulink и с расширением .c. В этом файле (для примера он составляет порядка 30 кб кода) реализуется численное дифференцирование с помощью алгоритма ode5 и он не нуждается в модификации. Файл с окончанием _data.c содержит в себе структуру данных, которая содержит параметры модели, которые возможно модифицировать во время выполнения программы.

Файл, который необходимо модифицировать называется ert_main.c. В этом файле необходимо реализовать две задачи: 1) возможность модификации параметров модели 2) выполнение в реальном времени или с заданным ускорением.

Для решения первой задачи наиболее простым и переносимым является реализация интерфейса командной строки, рассмотренная, например в [8]. Минимальный интерфейс командной строки должен иметь следующие команды: 1) start – запуск симуляции 2) pause – пауза 3) restart – сброс настроек и запуск симуляции 4) time – вывод текущего времени симуляции / переход к заданному времени 5) acc – коэффициент ускорения симуляции 6) set – установка параметра 7) get – получение значения параметра, группы параметров или всех блоков 8) status – текущее состояние (пауза, выполнение, ошибка) 9) stepsize – получение шага моделирования 10) quit – прекращение работы программы. При запуске программы возникает приглашение командной строки, после ввода команды надо нажать ENTER и результат выведется. Для повышения надежности работы с экранным началом и концом результата выполнения команды рекомендуется выделять специальными ASCII символами (№2 и 3 в десятичной системе). Кроме того, в результате должен быть символ, который указывает на ошибку (21) и успешный результат выполнения (6).

Решение второй задачи во многом зависит от возможностей платформы. Для кроссплатформенного решения нужно реализовать как-минимум работу с помощью потоков POSIX, так и работу с помощью потоков Windows. Поскольку задача жесткого реального времени не стоит, а в Python программе и не может стоять (в т.ч. из-за наличия сборщика мусора), то это не является сложной задачей.

Этап 4 – Реализация RESTful web-сервиса на языке Python

4.1 Определение программного интерфейса. В первую очередь для реализации любого web-сервиса необходимо определить программный интерфейс. В нашем случае имеет смысл использовать только 2 класса команд: GET (считывание параметров) и PUT (изменение параметров). Сериализация данных проходит по правилам JSON.

Команды GET – команды для считывания данных:

1. /get/acc – текущий коэффициент ускорения симуляции. Пример вывода: 1.000000.
2. /get/blocks – текущие состояния блоков. Пример вывода: {"A11": -0.045103, "A12": -0.0, "A21": -0.024769, "A22": -0.0, "Add": 1.0, "Add1": 0.0, "Add4": 0.053996, "Add5": 0.069825, "Add6": 0.0, "Add7": 0.0, "G11": 0.099099, "G12": 0.0, "G21": 0.094595, "G22": 0.0, "Time": "67.800000", "W11": 5.006425, "W11delay": 0.0, "W12": 0.0, "W12delay": 0.0, "W21": 5.498778, "W21delay": 0.0, "W22": 0.0, "W22delay": 0.0}.
3. /get/model/[k, T или tau] – текущие коэффициенты блоков. Пример вывода для k: {"k11": 11.0, "k12": 12.0, "k21": 21.0, "k22": 22.0}.
4. /get/status – состояние симуляции. Примеры вывода: RUNNING, PAUSED, ERROR.
5. /get/stepsize – шаг моделирования. Пример вывода: 0.2.
6. /get/time – время моделирования. Пример вывода: 789.200000.
7. /get/variable/[u, f или y] – текущие значения основных переменных с указанием времени моделирования. Пример вывода для y: {"y1": 10.990272, "y2": 19.569536, "Time": "880.400000"}.

Команды PUT – команды для считывания данных:

1. /set/acc – новые коэффициент ускорения симуляции. Пример запроса: {"new_acc": 10}
2. /set/model – новые параметры модели. Пример запроса: {"k11": 2, "T12": 4}
3. /set/status – изменение состояния. Пример запроса: {"new_status": "pause"}
4. /set/time – ускоренно промоделировать до нового времени. Пример запроса: {"new_time": "500"}
5. /set/time – новые переменные. Пример запроса {"u1": 1, "f2": 5}

4.2 Определение формата файла конфигурации. Конфигурацию предлагается хранить в виде текстового файла формате INI. Этот формат оптимален для хранения небольшого количества настроек и удобен для ручного редактирования пользователем. Конфигурация должна содержать следующие параметры: 1) сетевые – сетевой адрес и порт, 2) имя и кодировка исполняемого файла программы-решателя 3) режим запуска по умолчанию – фоновая программа или программа с интерфейсом командной строки, требование автоматического запуска моделирования 4)

параметры безопасности (наличие гостевого доступа, необходимость авторизации для записи данных, разрешение на удаленную перезагрузку и приостановку моделирования) 5) параметры отчетности – имя файла для отчета, степень детальности отчета (отладка, нормальный, только ошибки) 6) параметры модели, вводимые перед началом моделирования (для того, чтобы не менять исполняемый файл решателя).

4.3 Выбор программных библиотек. Язык Python – лидер по количеству качественных и бесплатных библиотек для многих типовых задач программирования.

Наиболее популярными библиотеками для реализации web-сервисов и приложений на Python являются Django и Flask. Последняя библиотека минималистична при достаточном для поставленной задачи функционале, поэтому следует использовать именно Flask. Для реализации публичного программного интерфейса необходима библиотека для управления доступом с других ресурсов – Swagger и flask-cors. Также рекомендуется использовать для реализации функционала авторизации – flask-login и для кеширования – lru-cache. Поскольку информация имеет дискретный характер, то кеширование GET запросов может значительно снизить нагрузку.

Для взаимодействия с исполняемой программой решателя необходимо использовать библиотеку wxpext для Windows и rpxext для POSIX-совместимых платформ. Программный интерфейс у этих библиотек практически одинаков, однако внутренняя реализация полностью отлична.

Для реализации многопоточности необходимо использовать функционал библиотек os, threading, signal.

Для формирования приоритетной очереди рекомендуется использовать функционал, реализуемый с помощью библиотек queue, itertools и TypeVar.

Для обработки конфигурационных файлов рекомендуется использовать библиотеку configparser (преимущество этой библиотеки также в обеспечении возможности менять параметры конфигурации с помощью передачи аргументов командной строки программе), для ведения отчетов – logger, для записи текущей даты и времени в отчет по формату – datetime.

4.4 Программная реализация. Предлагаемая структура программы на Python запускает четыре процесса: процесс основной программы, процесс сервера Flask, процессы AppThread и TimerThread.

Процесс основной программы проводит запуск остальных процессов, реализует работу с системой ввода-вывода I/O и обработку стандартных сигналов ОС (SIGTERM, SIGINT и т.п.).

Процесс сервера Flask запускается в многопоточном режиме и обрабатывает адреса (т. н. маршруты, routes) API, описанные в разделе 4.1. С помощью декоратора, который содержит шаблон адреса, к адресу привязывается функция-обработчик. В случае GET запроса в функциях-обработчиках должны возвращаться данные из кэша, реализуемого с помощью функций с декоратором lru_cache с указанием периода обновления кэша. В случае PUT запроса входные данные проверяются и при их корректности добавляются в очередь с приоритетом 1.

Процесс, реализуемый с помощью класса TimerThread периодически добавляет в очередь команды обновления переменных (приоритет 2) и обновления параметров блоков (приоритет 3). Кроме того, в процессе адаптируется время цикла добавления в очередь по результатам отслеживания времени возвращения результата выполнения команд от решателя с целью недопущения переполнения очереди при высокой загрузке компьютера.

Процесс, реализуемый с помощью класса AppThread обеспечивает постоянную работу с программой-решателем с помощью epxext-подобного управления: выбирает команду из очереди, посылает команды программе-решателю (sendline), дожидается возврата результата (expect) и считывает результаты (readline). Структура и основные алгоритмы класса представлены на рисунке 3.

4.5 Распространение. Исполняемый файл программы-решателя при сборке с помощью компилятора MINGW64 занимает ~1.4 МБ места на диске, в рабочем режиме процесс занимает ~14 МБ + память библиотек. Переносимая сборка программы на Python под разные платформы может быть реализована с помощью утилиты cx_Freeze. Под Windows путем тонкой настройки удалось добиться размера исполняемого файла ~13 МБ + ~24 МБ библиотек. Альтернативная возможность – использование такого инструмента как Docker. Кроме того, можно удовлетвориться стандартной установкой Python с установкой при необходимости требуемых библиотек в виртуальном окружении.

Этап 5 – Реализация интерфейса для удаленной работы с web-сервисом. Наиболее простым средством общения с RESTful web-сервисом является консольный клиент curl. Для ряда задач такой интерфейс может быть достаточен.

Типовым пользовательским интерфейсом для web-сервисов является web-интерфейс. Технология разработки такого интерфейса сводится к следующему 1) С помощью команды Matlab print модель Simulink экспортируется в формат SVG. 2) По рассчитанным с помощью языка DOT координатам на схему накладываются необходимые текстовые элементы. 3) Надписи связываются по таймеру с функцией периодического обмена с web-сервисом (при использовании библиотеки JQuery, например, обмен сообщениями реализуется с помощью функции \$.getJSON) 4) создаются функциональные элементы для изменения параметров с помощью вызовов put (в JQuery функция \$.ajax). 5) При необходимости строятся графики.

Пример работающего интерфейса показан на рисунке 4. На рисунке показан момент после изменения значения с u1 с 3 на 2. При нажатии на параметр, который можно менять, в интерфейсе возникает диалог с текстовым полем. При подтверждении данные передаются web-сервису. Построение графика в реальном времени производится с помощью библиотеки CanvasJs. В желтых прямоугольных рамках отображаются выходные сигналы блоков.

Структура класса

Алгоритм метода run()

Рис. 3 – Структура и основные алгоритмы класса AppThread

Fig. 3 – Structure and main algorithms of AppThread class

Этап 6 – Интеграция RESTful web-сервиса в имеющееся программно-техническое окружение. Этот этап сложно формализовать, поскольку многое зависит от задачи. Для интеграции в типовую современную промышленную систему автоматизации необходимо использовать шлюз JSON-OPC UA. Такой шлюз возможно как разработать отдельно, так и использовать стандартный, например, реализованный в пакете OPC Router. Для интеграции с MQTT возможно использовать функционал IoT платформ.

Для образовательных целей целесообразна разработка дополнительной программы, которая будет авторизовать пользователей на выполнение заданной работы. После авторизации на сервере должен запускаться для конкретного пользователя отдельный web-сервис с выделенным для него сетевым портом. После выполнения работы сервис должен прекращать работу и освобождать порт при сохранении отчета о действиях пользователя.

Выводы

Разработка web-сервисов на базе Simulink моделей является удобным инструментом для реализации т.н. «цифровых двойников» с целью отладки работы систем автоматизации, для реализации систем регулирования и для дистанционного обучения. Анализ литературных источников показал, что предлагаемые решения ориентируются на выполнение Simulink моделей с помощью Matlab, создавая удаленный интерфейс к Matlab, что ресурсозатратно и в отдельных случаях небезопасно.

Show API

Рис. 4 – Пример работающего web-интерфейса
Fig. 4 – Example of a working web-interface

В статье предложена технология проектирования автономного web-сервиса, который реализует Simulink модель в реальном времени и не требует для своей работы Matlab. Технология заключается в разработке комплекса из двух связанных отдельных приложений: консольного приложения на языке Си на базе кода, который генерирует Simulink Coder и приложения на языке Python, которое реализует функционал web-сервиса.

Технология разработки программного решения исходит из соответствия следующим требованиям:

1) быстрая разработка; 2) переносимость, т.е. возможность работать как в ОС общего назначения, реального времени и в ОС для одноплатных компьютеров; 3) низкие требования к ресурсам; 4) стандартный программный интерфейс удаленного доступа.

Рассмотрены этапы проектирования на примере разработки автономного web-сервиса для простого объекта, описываемого матрицей передаточных функций 2×2 .

Результаты исследования рекомендуется использовать для: 1) разработки «цифровых двойников» для отладки компонентов систем автоматизации; 2) быстрой разработки регуляторов сложной структуры (при условии, что она эффективно может быть представлена в виде отдельных блоков); 3) дистанционного обучения (для удаленного многопоточного доступа студентов к Simulink моделям).

Дальнейшее направление исследований планируется в области усовершенствования предложенного подхода в области создания цифровых двойников для сложных технологических объектов, таких как спиртовая БРУ [9], разработки регуляторов с неклассическими подходами в области восстановления переменных состояния для объектов с существенными перекрестными связями, таких как атмосферная нефтяная ректификационная колонна [10, а также в области дистанционного обучения в рамках LMS платформы.

Список використаних джерел

- [1] Puerto R., Jiménez L.M., Reinoso O. Remote control laboratory via Internet using Matlab and Simulink. *Computer Applications in Engineering Education*, 2010. T. 18, № 4. C. 694-702.
- [2] Janík Z., Žáková K. Online design of Matlab/Simulink and SciLab/Xcos block schemes, *14th International Conference on Interactive Collaborative Learning*. Piestany, 2011. C. 241-247.
- [3] Cirka L'uboš, Kalúz Martin. A Web-based Tool for Design of Simulink Models, *21st International Conference on Process Control*, 2017. C. 92-97.
- [4] Sysel Martin, Vaclavsky Michal. Remote Control and Monitoring in the Simulink. *International Journal of Circuits, Systems and Signal Processing*, 2013. T. 7. № 2. C. 83-90.

- [5] Bailey Bill. Connecting Simulink models to REST APIs. URL: <https://billbailey.io/2020/07/10/connecting-simulink-models-to-rest-apis/>.
- [6] Cirka L', Kal'uz M., Fikar M. On-line remote control of MATLAB simulations based on asynchronous communication model. *Matlab conference in Institute of Chemical Technology*, Prague, 2013. URL: http://dsp.vscht.cz/konference_matlab/MATLAB13/prispevky/014_cirka.pdf
- [7] Matsim: A sleek, intuitive interface for building Simulink models from a Matlab script. URL: <https://github.com/gave92/Matsim>.
- [8] Brennan Stephan. Write a Shell in C. URL: <https://brennan.io/2015/01/16/write-a-shell-in-c/>
- [9] Стопакевич А. А. Многомерная модель брагоректификационной установки косвенного действия. *Наукові праці ОНАХТ*, 2010. Т. 1., №38. С. 360-365.
- [10] Стопакевич А. А. Разработка робастной системы управления колонной атмосферной перегонки нефти *Восточно-европейский журнал передовых технологий*, 2015. Т. 5., №2 (77). С. 49-57.

References

- [1] R. Puerto, L.M. Jimenez and O. Reinoso, "Remote control laboratory via Internet using Matlab and Simulink," *Computer Applications in Engineering Education*, vol. 18, iss. 4, pp. 694-702, 2010.
- [2] Z. Janík and K. Zhakova, "Online design of Matlab/Simulink and SciLab/Xcos block schemes," *14th International Conference on Interactive Collaborative Learning*, Piastany, pp. 241-247, 2011.
- [3] L. Cirka and M. A. Kaluz, "Web-based Tool for Design of Simulink Models", *21st International Conference on Process Control*, pp. 92-97, 2017.
- [4] M. Sysel and M. Vaclavsky, "Remote Control and Monitoring in the Simulink," *International Journal of Circuits, Systems and Signal Processing*, vol. 7. iss. 2, pp. 83-90, 2013.
- [5] B. Bailey, *Connecting Simulink models to REST APIs*. URL: <https://billbailey.io/2020/07/10/connecting-simulink-models-to-rest-apis>.
- [6] L. Cirka, M. A. Kaluz and M. Fikar, "On-line remote control of MATLAB simulations based on asynchronous communication model," *Matlab conference in Institute of Chemical Technology*, 2013, Prague. URL: http://dsp.vscht.cz/konference_matlab/MATLAB13/prispevky/014_cirka.pdf
- [7] *Matsim: a sleek, intuitive interface for building Simulink models from a Matlab script*. URL: <https://github.com/gave92/Matsim>.
- [8] Stephan Brennan, *Write a Shell in C*. URL: <https://brennan.io/2015/01/16/write-a-shell-in-c/>
- [9] A. A. Stopakevich, "Mnogomernaja model' bragorektifikacionnoj ustanovki kosvennogo dejstvija," *Naukovi pratsi ONAKHT*, vol. 1, iss. 38, pp. 360-365, 2010.
- [10] A. A. Stopakevich, "Robust control system design of crude oil atmospheric distillation column," *Eastern-European Journal of Enterprise Technologies*, vol. 5., no. 2(77), pp. 49-57, 2015, DOI: <https://doi.org/10.15587/1729-4061.2015.50964>

УДК 004.42

ДОСЛІДЖЕННЯ МЕТОДІВ COMPUTER VISION ДЛЯ ВИКОРИСТАННЯ В АВТОМАТИЗОВАНИХ СИСТЕМАХ

Голубєв Л. П.¹, Ківа І. Л.², Матяш О. В.³

¹ Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», м. Київ, Україна

^{2,3} Київський національний університет технологій та дизайну, м. Київ, Україна

ORCID: ¹ <http://orcid.org/0000-0002-2980-8017>, ² <http://orcid.org/0000-0002-2920-5312>

E-mail: ¹golubev1@ukr.net, ²leonidovich1259@gmail.com, ³matasaleksandr97@gmail.com

Copyright © 2020 by author and the journal "Automation of technological and business – processes".

This work is licensed under the Creative Commons Attribution International License (CC BY).

<http://creativecommons.org/licenses/by/4.0>

DOI: <https://doi.org/10.15673/atbp.v12i4.1932>