СБОРНИК **НАУЧНЫХ ТРУДОВ НГТУ. – 2020. – № 1–2 (97).** – 40–54

УДК 681.5.01 DOI: 10.17212/2307-6879-2020-1-2-40-54

ПРОГРАММНОЕ ОПИСАНИЕ АЛГОРИТМА РАБОТЫ ПИД-РЕГУЛЯТОРА НА ЯЗЫКЕ C# ДЛЯ ПРИМЕНЕНИЯ В ЗАМКНУТОЙ СИСТЕМЕ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ*

Д.Л. ПИОТРОВСКИЙ[1], С.А. ПОДГОРНЫЙ[2], А.А. КУКОЛЕВ[3]

[1] *9900, ТРСК, Никосия, ул. Сержанта Юсуфа Эчвета, Средиземноморский университет Карпассии, доктор технических наук, заведующий кафедрой пилотирования. E-mail: Dmitrii.piotrovskii@akun.edu.tr*
[2] *350072, РФ, г. Краснодар, ул. Московская, 2, Кубанский государственный технологический университет, доктор технических наук, профессор кафедры автоматизации производственных процессов. E-mail: saptich@rambler.ru*
[3] *350072, РФ, г. Краснодар, ул. Московская, 2, Кубанский государственный технологический университет, аспирант кафедры автоматизации производственных процессов. E-mail: sashanius@yandex.ru*

На сегодняшний день автоматическое управление становится всё сильнее связано с возможностями электронно-вычислительной техники. Если на этапе зарождения теории управления регуляторы представляли собой преимущественно механические устройства с простейшей кинематикой, то сейчас большинство регуляторов на производстве являются электронно-вычислительными устройствами, содержащими центральный процессор, аналого-цифровые преобразователи, усилители, вспомогательную коммутационную аппаратуру, периферийные устройства в составе человеко-машинного интерфейса. Ответственность за корректную и безаварийную работу и эксплуатацию такой аппаратуры возложена на программное обеспечение, представляющее собой программный код, написанный на определенном языке программирования (Java, Python, C#, Pascal). Те или иные языки адаптированы под конкретные цели, определяющие степень их распространенности. Разработанный в 1998–2001 годах специалистами компании Microsoft объектно-ориентированный язык C# к настоящему времени обрел внушительную популярность благодаря выразительности синтаксиса и простоте изучения. Синтаксис языка стремится нивелировать сложности C#, предоставляя такие внушительные возможности, как использование лямбда-выражений, делегатов, а также предоставление прямого доступа к памяти. Количество написанных к настоящему времени приложений, написанных на C#, не поддается исчислению ввиду кроссплатформенности ядра .NET Core. По этой причине авторами была предпринята попытка рассмотрения возможности написания программного алгоритма классического ПИД-регулятора в цепи простейше-

* Статья получена 07 мая 2020 г.

го апериодического звена первого порядка с использованием именно этого языка программирования.

Ключевые слова: программное обеспечение, автоматическое управление, электронно-вычислительная техника, язык программирования, Visual C#, регулятор, приложение, устойчивость, переходной процесс

ВВЕДЕНИЕ

По различным оценкам, более 90 % реально используемых в системах автоматического регулирования регуляторов – это классические ПИД-регуляторы, синтезированные на основе традиционных инженерных методов [1]. Причиной этого служит тот факт, что к настоящему времени накоплен огромный опыт эксплуатации таких регуляторов, разработаны и апробированы различные методы синтеза и настройки параметров, а также простота построения и промышленного использования, ясность функционирования, пригодность для решения большинства промышленных задач [11]. Однако каждый из известных в настоящее время методов имеет определенные достоинства и недостатки, свои ограничения и область применения. Но внимание к методам синтеза и анализа систем, построенных под управлением таких устройств, не ослабевает и в последние годы. В этой связи можно упомянуть конференцию, проведенную 8 и 9 октября 2019 года в Нью-Йорке, США, по управлению, нечетким регуляторам и ПИД-регуляторам [2], на которой были представлены самые передовые решения в области управления технологическими процессами. Интерес к данной области не ослабевает и по причине разнообразия программных средств для разработки, отладки и моделирования таких устройств. К настоящему времени разработано множество программных комплексов типа Matlab, Simulink, Maple, VisSim. Каждый из них обладает определенными преимуществами и недостатками, предлагая научному сообществу самые разные пути решения прикладных задач. В настоящей работе рассматривается один из способов программного моделирования работы ПИД-регулятора в цепи с апериодическими звеньями. Программный код показан на примере языка C#, набравшего в последнее время определенную популярность ввиду простоты освоения и большого количества справочных материалов. Написание кода реализовано в программном пакете Visual Studio 2017.

1. ПОСТАНОВКА ЗАДАЧИ

С точки зрения программной реализации описание алгоритма работы ПИД-регулятора не представляет большой сложности, за исключением дифференциальной составляющей, которая и определяет один из его недостатков.

Так как дифференциальная составляющая пропорциональна темпу изменения отклонения регулируемой величины, то значительные отклонения в быстротекущих процессах способны вызвать значительные ее скачки, например, при внезапных амплитудных изменениях регулируемой величины по причине изменения нагрузки объекта:

$$U_d(t) = K_d \frac{de}{dt},$$

где $U_d(t)$ – величина дифференциальной составляющей; K_d – коэффициент усиления дифференцирующего звена; $\frac{de}{dt}$ – производная изменения невязки по времени.

В этом случае качество управления резко снижается из-за изменения времени переходного процесса или принципиальной невозможности поддержания заданного значения регулируемой величины (рис. 1).

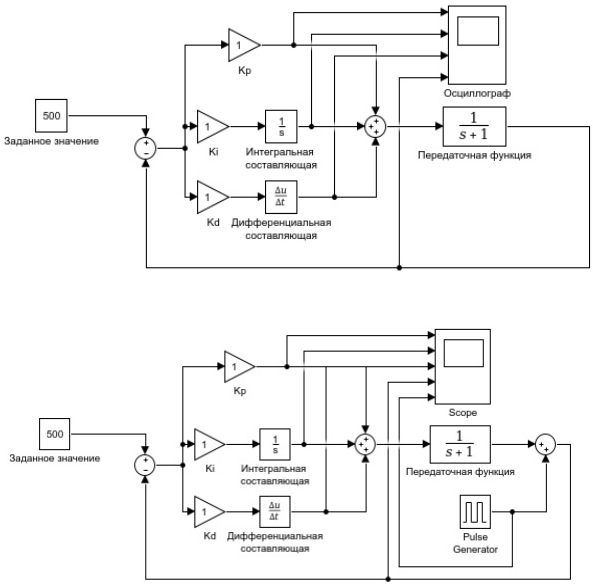

Рис. 1. Динамика изменения дифференциальной составляющей при апериодическом изменении нагрузки объекта

(окончание см. на с. 43)

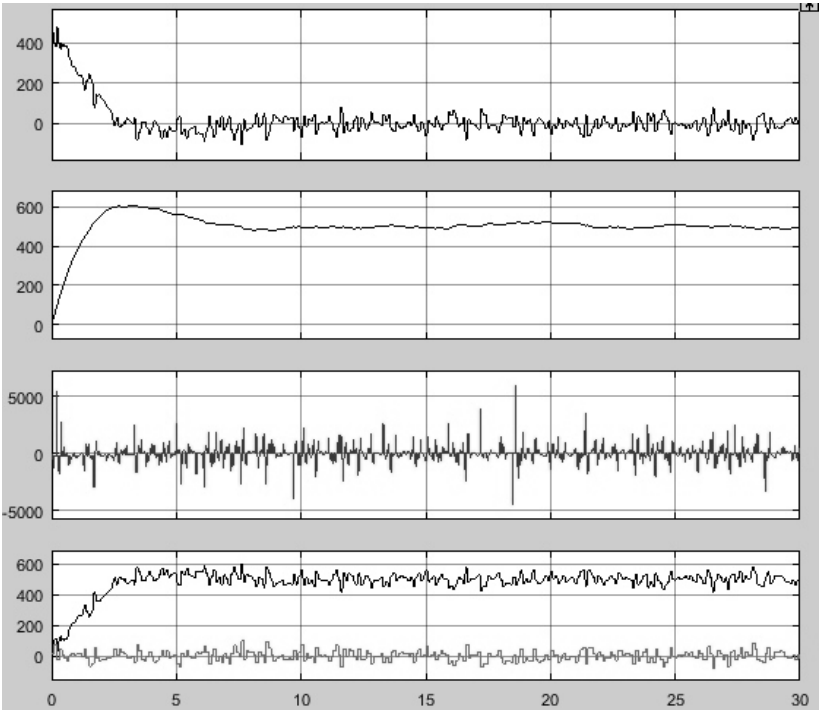

Рис. 1. Окончание

В связи с этим в системах автоматического управления для защиты объектов от ненормальных режимов работы под управлением ПИД-регуляторов вводят дополнительные программные и аппаратные решения, ограничивающие значения регулирующих воздействий на объекты управления.

По этой же причине программное описание ПИД-закона имеет ряд характерных особенностей.

2. ОСОБЕННОСТИ ПРОГРАММНОЙ РЕАЛИЗАЦИИ АЛГОРИТМА РАБОТЫ РЕГУЛЯТОРА

Для написания и компиляции программного кода авторами использовался программный пакет Microsoft Visual Studio 2017. Простейшая программа, реализованная в Windows Forms, включает в себя несложный интерфейс и встроенный таймер, при помощи событий которого и производится инициализация очередного цикла расчета переходного процесса.

Рассмотрим реализацию расчета значения сигнала воздействия регулятора в следующей функции встроенного таймера программы:

```
public void calculation()
        {

1       err = stpnt – proc_val;
        #region integral
2       integral = integral + (err * t);

3       if (integral > 300000)
            integral = 300000;
        if (integral < -300000)
            integral = -300000;
        #endregion

        #region дифференциальная составляющая
4       derivative = (err - preErr) / t;
5       if (derivative > 300000)
            derivative = 300000;
        if (derivative < -300000)
            derivative = -300000;

        #endregion

        #region output
6       output = Math.Round((Kp * error) + (Ki * integral) + (Kd *
derivative),
                    round_value);
7       if (output > 300000)
            output = 300000;
        if (output < -300000)
            output = -300000;
        #endregion
8       preErr = err;

        }
```

Данная функция вызывается событием срабатывания таймера и в каждый момент осуществляет вычисление одного значения, соответствующего

предыдущему результату регулируемой величины. Определение ошибки регулирования (строка 1 программы) осуществляется в первой строке функции посредством вычисления разности заданного значения регулируемой величины (*stpnt*) и ее действительного значения (*proc_val – process value*). Интегральная составляющая (*integral*, строка 2 программы) определяется как сумма предыдущего значения integral с ошибкой регулирования, умноженной на шаг времени. Строка 3 служит для ограничения значений интегральной составляющей диапазоном –300 000…300 000 относительных единиц. Дифференциальная составляющая (*derivative,* строка 4) определяется как изменение ошибки регулирования за шаг времени. В строке 5 производится ограничение дифференцирующей составляющей, а в строках 6 и 7 – расчет выходного сигнала регулятора и его ограничение. Строка 8 присваивает значение ошибки регулирования переменной предыдущего значения ошибки для последующего расчета. Как можно заметить, алгоритм работы программы не представляет сложности. Поэтому целесообразно произвести сравнение результатов работы данного алгоритма с результатами, полученными в программном пакете *Matlab Simulink*. Как видно из рис. 2, при шаге моделирования 1 мс графики переходных процессов совпадают. В связи с этим можно судить о справедливости вышеприведенных рассуждений.

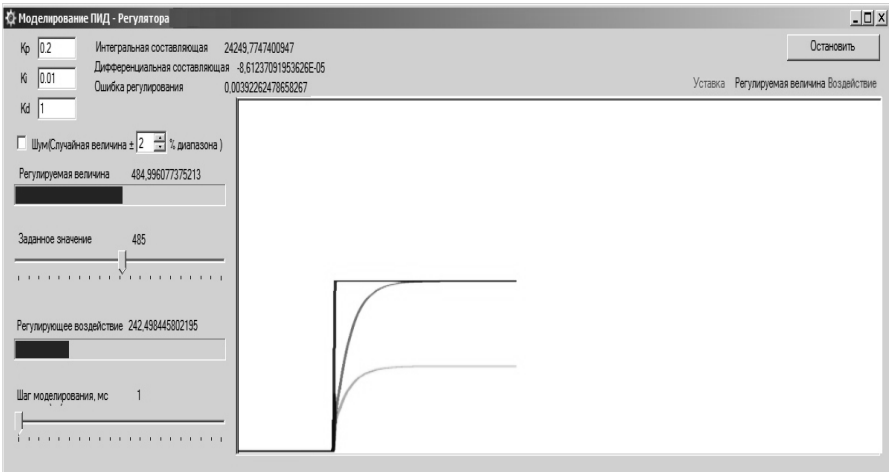

Рис. 2. Сравнение переходных процессов простейшего ПИД-регулятора в программном комплексе *Matlab Simulink* и разработанной программной модели (окончание см. на с. 46)

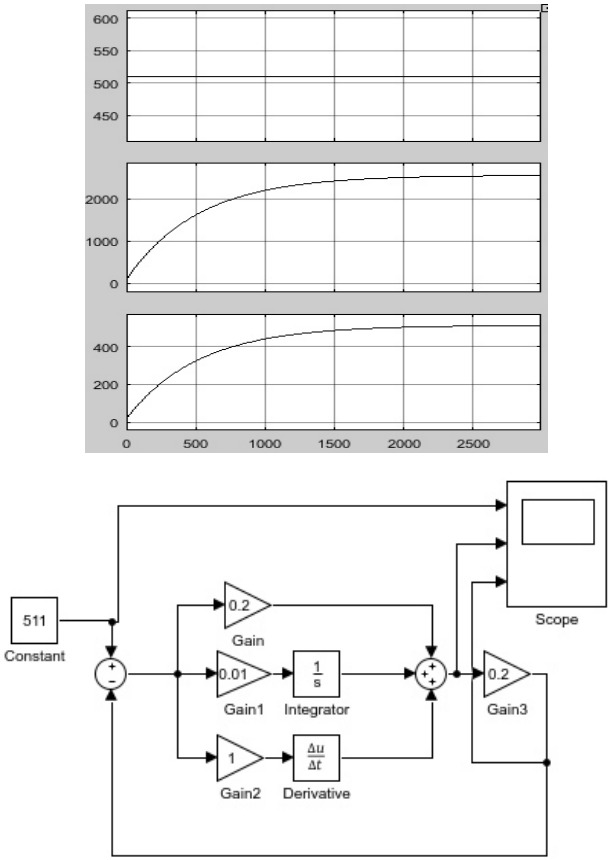

Рис. 2. Окончание

Большинство систем управления описываются нелинейными дифференциальными уравнениями. Во многих случаях их подвергают линеаризации, т. е. замене исходных нелинейных уравнений линейными [3, 9]. Впоследствии при помощи преобразования Лапласа дифференциальные уравнения сводятся к передаточным функциям звеньев – составляющих системы управления. Типовая непрерывная эволюционная система описывается системой обыкновенных дифференциальных уравнений, записанных в нормальной форме:

$$\dot{x}_i(t) = f_i(t,\, x_1, \ldots,\, x_n), \quad i = 1, \ldots,\, n;$$

$$x_i(t_0) = x_{i0}.$$

В данном случае t – независимая переменная; $x_1(t)$, $x_n(t)$ – искомые функции; t_0, x_{i0} – начальные заданные условия.

При рассмотрении процесса программной реализации передаточных функций актуальным вопросом становится выбор метода решения дифференциальных уравнений звеньев [4, 7, 8]. Для решения этой задачи существует ряд библиотек, позволяющих с той или иной степенью точности определять выходное воздействие звена как результат входного. На официальном сайте компании *Microsoft* 15 июля 2014 года была опубликована открытая библиотека *Open Solving Library for ODEs*. Компания *Microsoft* разработала эту библиотеку в сотрудничестве с научным сообществом Московского государственного университета с целью предоставления численных методов решений обыкновенных дифференциальных уравнений [5, 6]. В качестве методов решения дифференциальных уравнений библиотека использует метод Рунге–Кутты и формулы обратного дифференцирования [5]. Например, для решения модели Лотки–Вольтерра (хищник – жертва, в которой взаимодействия существ ведут к изменению во времени их популяций) вида

$$\frac{dx}{dt} = x - xy;$$

$$\frac{dy}{dt} = -y + xy,$$

где $x(t)$ – популяция жертв, а $y(t)$ – популяция хищников с начальными условиями $x(0) = 5$, $y(0) = 1$. Используемый метод Рунге–Кутты (*RK547M*) принимает следующие переменные: время начала расчета, двумерный вектор состояния системы, лямбда-выражение, определяющее правую сторону уравнения, соответствующего заданной передаточной функции. Для модели Лотки–Вольтерра выражение имеет следующий синтаксис:

```
var sol = Ode.RK547M(0, new Vector(5.0, 1.0), (t, x) => new Vector( x[0] - x[0] * x[1], -x[1] + x[0] * x[1]));
```

Переменная sol после расчета представляет собой конечную последовательность точек – решений уравнения. Интегрирование же производится после оценки переменной sol при помощи метода SolveFromToStep (0, 20, 1), который формирует переменную, содержащую все численные решения уравнения, пригодные для дальнейшей обработки и визуализации (рис. 3).

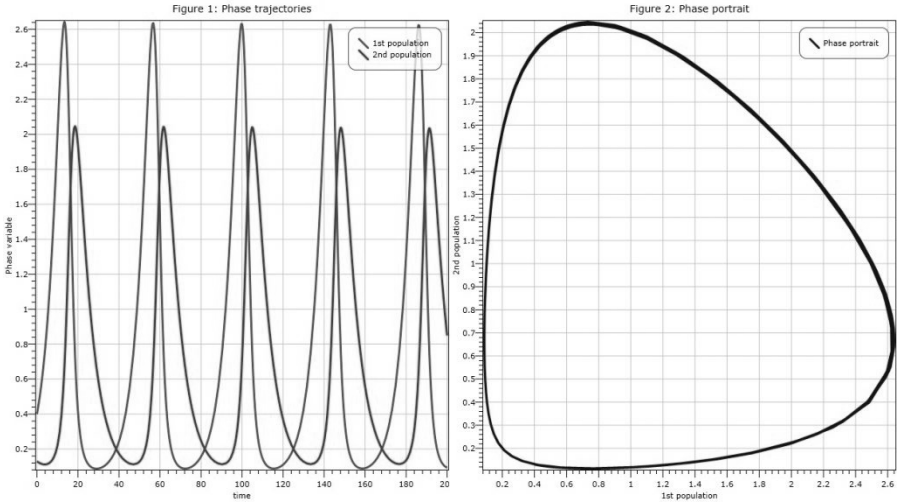

Рис. 3. Визуализация решения задачи Лотки–Вольтерра при помощи библиотеки *Open Solving Library for ODEs*

3. АЛГОРИТМ ПРОГРАММНОЙ РЕАЛИЗАЦИИ РАСЧЕТА

Для рассмотрения алгоритма программной реализации приведем пример определения реакции исполнительного механизма, описанного передаточной функцией *W(S)= 1/(10S+1),* для системы управления, изображенной на рис. 4:

#region определение реакции исполнительного механизма

```
1       sol_ME_calc = Math.Round(motor_reaction, round_value);

2       motor_k = Math.Round((gain * output), round_value);

3       var sol_motor = Microsoft.Research.Oslo.Ode.RK547M(solve_from, new
Vector(sol_ME_calc), (t, x) => ((motor_k - motor_b * x[0]) / motor_T),

4       new Options
        {
        AbsoluteTolerance = 1e-6,
        RelativeTolerance = 1e-6
        }
        );
```

```
5       foreach (var spm in sol_motor.SkipWhile(spm => spm.T < solve_from).TakeWhile(spm => spm.T <= solve_to))
        {

                motor_reaction = Math.Round(spm.X[0], round_value);
                Data.t_value = spm.T;
                if (motor_reaction > 300000)
                        motor_reaction = 300000;
                if (motor_reaction < -300000)
                        motor_reaction = -300000;
        }
        #endregion
```

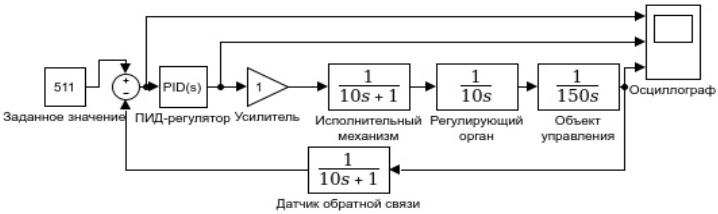

Рис. 4. Структурная схема системы управления

Код операции численного интегрирования в этом случае состоит из нескольких шагов. В первой строке кода определяется округленное значение реакции исполнительного механизма за предыдущий интервал времени. Вторая строка определяет величину числителя передаточной функции, помноженного на входное воздействие исполнительного механизма. Третья строка осуществляет решение дифференциального уравнения по методу Рунге–Кутты. Операторами выражения здесь являются: *solve_from* – начальное значение времени интегрирования; *new Vector(sol_ME_calc)* – двумерный вектор состояния системы; *(t, x) => ((motor_k - motor_b * x[0]) / motor_T)* – лямбда-выражение, определяющее правую сторону уравнения, соответствующего заданной передаточной функции. Четвертая строка определяет абсолютную и относительную точности интегрирования. Пятая строка осуществляет непосредственное интегрирование полученного в четвертой строке экземпляра коллекции *IEnumerable<SolPoint>* в заданных пределах – от точки *spm* до точки *solve_to*. Далее рассчитывается ограничение, и процесс повторяется для следующего состояния системы. В качестве события, воспроизводящего алгоритм расчета, авторами использовалось событие срабатывания встроенного программного таймера. Данная реализация алгоритма позволяет осуществить непрерывное моделирование переходного процесса либо моделирование до появления определенного состояния (например, выход регулируе-

мой величины в установившееся значение). Однако в данном случае может стать актуальным вопрос оптимизации ввиду большого количества вычислений, производимых в единицу времени [10].

Построив таким образом показанную на рис. 4 систему и сравнив ее с полученными переходными характеристиками схожей системы, построенной в *Matlab Simulink*, можно получить следующие данные (рис. 5).

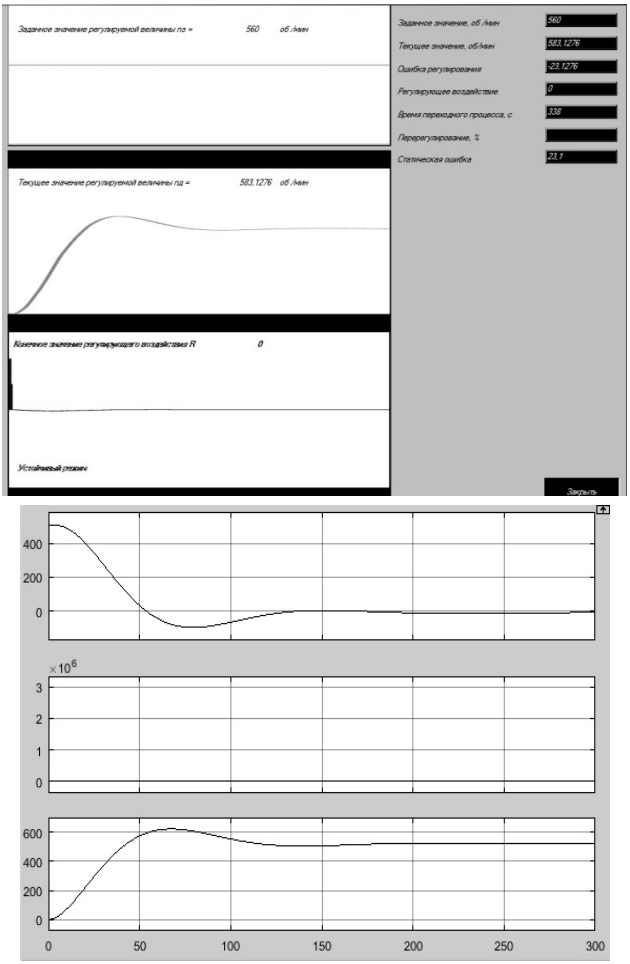

Рис. 5. Сравнение переходных процессов исследуемой САУ в разрабатываемом программном пакете и в *Matlab Simulink*

Анализ качества переходного процесса, проведенный в *Matlab Simulink*, предоставляет следующий результат.

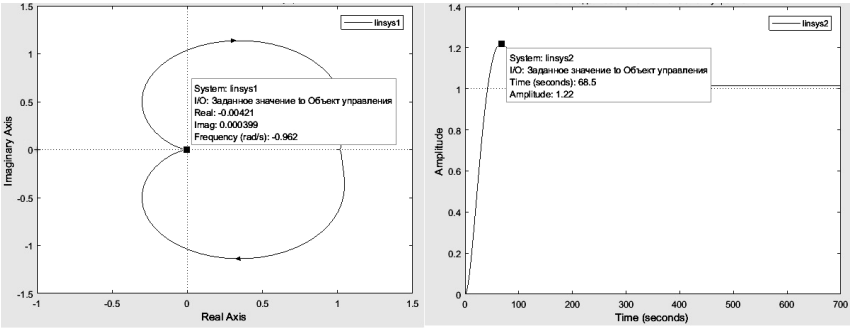

Рис. 6. Анализ качества переходного процесса в Matlab Simulink

ЗАКЛЮЧЕНИЕ

В процессе выполнения работы были достигнуты следующие результаты.

1. В пакете *Matlab Simulink* была собрана модель системы автоматического управления, состоящая из трех апериодических звеньев и звена обратной связи.

2. В программном пакете *Visual Studio* 2017 была разработана программа для моделирования работы исследуемой системы. В качестве алгоритма работы регулятора был использован алгоритм последовательного расчета регулирующего воздействия регулятора по срабатыванию внутреннего таймера программы.

3. Выполнено моделирование установившегося режима и проведено сравнение режимных параметров работы исследуемой системы управления. Произведена оценка правдоподобности полученных результатов.

4. Рассмотрены особенности моделирования замкнутых систем управления при разработке алгоритмов их работы на языке *Visual C#.*

5. Определены дальнейшие задачи по развитию и модернизации полученных результатов.

СПИСОК ЛИТЕРАТУРЫ

1. *Киселев О.Н., Поляк Б.Т.* Синтез регуляторов низкого порядка по критерию H∞ и по критерию максимальной робастности // Автоматика и телемеханика. – 1999. – № 3. – С. 119–130.

2. Программа международной конференции по ПИД-регуляторам // ICREFS 2019: International Conference on Renewable Energy Forecasting and Storage. – New York, 2019. – URL: https://panel.waset.org/conference/2019/10/new-york/program (дата обращения: 06.07.2020).

3. *Ким Д.П.* Теория автоматического управления. Т. 1. Линейные системы. – М.: Физматлит, 2003. – 288 с.

4. Математические основы теории автоматического управления: учебное пособие / В.А. Иванов, В.С. Медведев, Б.К. Чемоданов, А.С. Ющенко. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2009. – 352 с.

5. Open Solving Library for ODEs. – URL: https://www.microsoft.com/en-us/research/project/open-solving-library-for-odes/ (accessed: 06.07.2020).

6. Язык программирования C# / А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд. – 4-е изд. – СПб.: Питер, 2012. – 784 с. – (Классика Computer Science).

7. *Пугачев В.И., Марков Ю.Ф., Подгорный С.А.* Алгоритм предельно высокой интенсивности цифрового управления // Известия высших учебных заведений. Пищевая технология. – 2006. – № 1 (290). – С. 83–86.

8. Анализ точности метода конечных элементов / С.А. Подгорный, В.С. Косачев, Е.П. Кошевой, А.А. Схаляхов // Новые технологии. – 2013. – № 4. – С. 31–38.

9. *Льюис Э.Д.* Математический подход к классическому управлению. – М.: Автоматика и технологии, 2003. – 655 с.

10. *Astrom K.J., Hagglund T.* Advanced PID control. – Research Triangle Park, NC: ISA, 2006. – 460 p.

Пиотровский Дмитрий Леонидович, доктор технических наук, профессор, заведующий кафедрой пилотирования Средиземноморского университета Карпассии. Основное направление научных исследований – автоматизированные системы управления технологическими процессами. Имеет более 200 публикаций. E-mail: Dmitrii.piotrovskii@akun.edu.tr

Подгорный Сергей Александрович, доктор технических наук, профессор кафедры автоматизации производственных процессов Кубанского государственного технологического университета. Основное направление научных исследований – автоматизированные системы управления массообменными процессами. Имеет более 70 публикаций. E-mail: saptich@rambler.ru

Куколев Александр Александрович, аспирант кафедры автоматизации производственных процессов Кубанского государственного технологического университета. E-mail: sashanius@yandex.ru

DOI: 10.17212/2307-6879-2020-1-2-40-54

Software description of the PID-controller algorithm in C# for use in a closed automatic control system*

D.L. Piotrovsky[1], S.A. Podgorny[2], A.A. Kukolev[3]

[1] *University of Mediterranean Karpasia, Sht. Ecvet Yusuf Street, Lefkosa, KKTC, doctor of technical sciences, head of the piloting department. E-mail: Dmitrii.piotrovskii@akun.edu.tr*
[2] *Kuban State Technological University, 2 Moskovskaya Street, Krasnodar, 350072, Russian Federation, doctor of technical sciences, professor of the automation of industrial processes department. E-mail: saptich@rambler.ru*
[3] *Kuban State Technological University, 2 Moskovskaya Street, Krasnodar, 350072, Russian Federation, the post-graduate student of the automation of industrial processes department. E-mail: sashanius@yandex.ru*

Today, automatic control is increasingly associated with the capabilities of electronic computing. If at the stage of the origin of the control theory, regulators were mainly mechanical devices with the simplest kinematics, now most regulators in production are electronic computing devices containing a Central processor, analog - to - digital converters, amplifiers, auxiliary switching equipment, peripheral devices as part of a human-machine interface. Responsibility for the correct and trouble-free operation and operation of such equipment is assigned to software that is a program code written in a specific programming language (Java, Python, C#, Pascal). Certain languages are adapted for specific purposes that determine their prevalence. Developed in 1998–2001 by Microsoft specialists, the object-oriented C# language has now gained considerable popularity due to its expressive syntax and ease of learning. The syntax of the language aims to level the complexity of C#, providing such impressive features as the use of lambda expressions, delegates, as well as providing direct access to memory. The number of applications written to date written in C# is incalculable due to the cross-platform nature of the .NET Core. For this reason, the authors attempted to consider the possibility of writing a software algorithm for a classical PID controller in the chain of the simplest first-order aperiodic link using this programming language.

Keywords: software, automatic control, electronic computing, programming language, Visual C#, controller, application, stability, transition process

REFERENCES

1 Kiselev O.N., Polyak B.T. Design of low-order controllers by the H∞ and maximal-robustness performance indices. *Automation and Remote Control*, 1999, vol. 60, no. 3, pp. 393–402. Translated from *Avtomatika i telemekhanika*, 1999, no. 3, pp. 119–130.

2. Program for International Conference on PID Controllers. *ICREFS 2019: International Conference on Renewable Energy Forecasting and Storage*, New York,

* *Received 07 May 2020.*

2019. Available at: https://panel.waset.org/conference/2019/10/new-york/program (accessed 06.07.2020).

3. Kim D.P. *Teoriya avtomaticheskogo upravleniya*. T. 1. *Lineinye sistemy* [Automatic control theory. Vol. 1. Linear systems]. Moscow, Fizmatlit Publ., 2003. 288 p.

4. Ivanov V.A., Medvedev V.S., Chemodanov B.K., Yushchenko A.S. *Matematicheskie osnovy teorii avtomaticheskogo upravleniya* [Automatic control theory mathematic aspects]. Moscow, Bauman MSTU Publ., 2009. 352 p.

5. *Open Solving Library for ODEs*. Available at: https://www.microsoft.com/en-us/research/project/open-solving-library-for-odes/ (accessed 06.07.2020).

6. Hejlsberg A., Torgersen M., Wiltamuth S., Golde P. *The C# programming language*. 4th ed. Upper Saddle River, NJ, Addison-Wesley, 2011 (Russ. ed.: Kheilsberg A., Torgersen M., Viltamut S., Gold P. *Yazyk programmirovaniya C#*. 4th ed. St. Petersburg, Piter Publ., 2012. 784 p.).

7. Pugachev V.I., Markov Yu.F., Podgornyi S.A. Algoritm predel'no vysokoi in-tensivnosti tsifrovogo upravleniya [Digital control critically high intensity algorithm]. *Izvestiya vysshikh uchebnykh zavedenii. Pishchevaya tekhnologiya = News of institutes of higher educational. Food technology*, 2006, no. 1 (290), pp. 83–86.

8. Podgornyi S.A., Kosachev V.S., Koshevoi E.P., Skhalyakhov A.A. Analiz tochnosti metoda konechnykh elementov [Ultimate elements method accuracy analysis]. *Novye tekhnologii = New technologies*, 2013, no. 4, pp. 31–38.

9. Lewis A.D. *Matematicheskii podkhod k klassicheskomu upravleniyu* [Mathematical approach to classical control]. Moscow, Avtomatika i tekhnologii Publ., 2003. 655 p.

10. Astrom K.J., Hagglund T. *Advanced PID control*. Research Triangle Park, NC, ISA, 2006. 460 p.

Для цитирования:

Пиотровский Д.Л., Подгорный С.А., Куколев А.А. Программное описание алгоритма работы ПИД-регулятора на языке C# для применения в замкнутой системе автоматического управления // Сборник научных трудов НГТУ. – 2020 – № 1–2 (97). – С. 40–54. – DOI: 10.17212/2307-6879-2020-1-2-40-54.

For citation:

Piotrovsky D.L., Podgorny S.A., Kukolev A.A. Programmnoe opisanie algoritma raboty PID-regulyatora na yazyke C# dlya primeneniya v zamknutoi sisteme avtomaticheskogo upravleniya [Software description of the PID controller algorithm in C# for use in a closed automatic control system]. *Sbornik nauchnykh trudov Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta = Transaction of scientific papers of the Novosibirsk state technical university*, 2020, no. 1–2 (97), pp. 40–54. DOI: 10.17212/2307-6879-2020-1-2-40-54.