RUDN Journal of Engineering researches
Вестник РУДН. Серия: Инженерные исследования
2018 Vol. 19 No. 2 177—189
http://journals.rudn.ru/engineering-researches

DOI 10.22363/2312-8143-2018-19-2-177-189
УДК 62-50, 519-714

БИБЛИОТЕКА PYTHON ДЛЯ СИНТЕЗА ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ УПРАВЛЕНИЯ

А.И. Дивеев[1,2], А.В. Доценко[2]

[1] Федеральный исследовательский центр «Информатика и управление», Российская академия наук
Российская Федерация, 119333, Москва, ул. Вавилова, 44
[2] Российский университет дружбы народов (РУДН)
Российская Федерация, 117198, Москва, ул. Миклухо-Маклая, 6

Статья посвящена описанию библиотеки программ на языке Python для решения задач синтеза систем управления методами символьной регрессии. Задача синтеза становится все более актуальной, приобретая особое значение ввиду стремительного развития робототехники. Как правило, инженеры и просто практики используют регуляторы шаблонного типа при моделировании, а затем подбирают под них параметры. В условиях, когда вычислительная мощность персональных компьютеров достигла своего апогея, а языки программирования стали чрезвычайно выразительны за счет высокого уровня абстрактности и обширности библиотек, целесообразнее реализовать синтез в виде пакета. В качестве языка для реализации синтеза был выбран Python. По мнению авторов статьи, Python является удобным языком для программирования матричных и векторных вычислений благодаря пакету numpy. Более того, доля проектов, написанных на Python, в веб-сервисе для хостинга Github за последнее время неизменно растет, что говорит о поддержке языка со стороны сообщества разработчиков. В данной статье представлено описание применения библиотеки для решения задачи синтеза управления. Приведено описание метода символьной регрессии, метода сетевого оператора и алгоритмов поиска оптимального решения с использованием принципа малых вариаций базисного решения. Рассмотрен пример использования библиотеки для решения задачи синтеза управления мобильным роботом, движущимся на плоскости, в условиях препятствий.

Ключевые слова: синтез управления, библиотека Python, метод символьной регрессии, оптимальное управление, мобильный робот

ВВЕДЕНИЕ

Применение методов символьной регрессии для решения задачи синтеза управления в последнее время становится наиболее популярным из-за быстрого развития вычислительной техники. Для применения методов символьной регрессии необходимо создание специального программного обеспечения, которое должно включать функции кодирования и декодирования математических выражений тем или иным методом символьной регрессии, эволюционные алгоритмы поиска оптимального решения, функции для моделирования объекта управления и др. Несмотря на то, что разработанные программные продукты для решения задачи синтеза системы управления методом символьной регрессии создаются уже в

течение последних десяти лет, универсальная библиотека, которая могла бы быть использована широким кругом пользователей, до сих пор отсутствует.

Язык программирования Python в настоящий момент имеет исключительную популярность и причисляется к языкам искусственного интеллекта или приравнивается к математическим пакетам. Скорее всего, это вызвано не особой лексикой языка или дополнительными операторами, или типам данных, а наличием большого количества библиотек, практически по всем актуальным вычислительным направлениям. Особую популярность приобрел Python у специалистов в области искусственных нейронных сетей. Процесс обучения искусственной нейронной сети методом обратного распространения ошибки доведен в библиотеках Python до утилитарного состояния, когда пользователь уже может использовать искусственную нейронную сеть для решения своих задач, особо не вникая в технологию обучения и теорию искусственных нейронных сетей. Приблизительно так же мы используем функции математических пакетов типа MatLab, например, при вычислении корней полиномов или решения систем дифференциальных уравнений. Одной из целей создания библиотеки Python является расширение круга пользователей, особенно прикладников, занимающихся разработкой систем автоматического управления сложными объектами или, в частности, робототехническими устройствами.

ЗАДАЧА СИНТЕЗА УПРАВЛЕНИЯ

Задача синтеза представляет собой задачу, решение которой в общем случае должно привести к нахождению математического выражения, описывающего функционирование системы управления. Заметим, что системы искусственного интеллекта — это системы управления некоторыми объектами. В этом случае задача синтеза управления является общей задачей, при решении которой иногда требуется создание систем искусственного интеллекта.

В задаче синтеза управления [1] предполагается, что известна математическая модель объекта управления. В общем случае модель объекта управления описывается системой обыкновенных дифференциальных уравнений

$$\dot{\mathrm{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{1}$$

где $\mathbf{x}$ — вектор состояния объекта управления; $\mathbf{u}$ — вектор управления; $\mathbf{x} = [x_1 ... x_n]^T$, $\mathbf{u} = [u_1 ... u_m]^T$; $m \le n$.

Компоненты вектора управления ограничены

$$u_i^- \le u_i \le u_i^+, \; i = \overline{1, m}, \tag{2}$$

где u_i^-, u_i^+ — заданные величины; $i = \overline{1, m}$.

Для системы (1) задана область начальных значений

$$\mathbf{x}(0) = \mathbf{x}^0 \in \mathrm{X}_0. \tag{3}$$

Заданы терминальные условия как цель управления

$$\varphi_i(\mathbf{x}(t_f)) = 0,\ i = \overline{1, r},\ r \le n, \tag{4}$$

где t_f — время окончания процесса управления, может быть строго заданным или в общем случае ограниченным и определяемым в процессе решения задачи, например, по достижении терминального многообразия (4)

$$t_f = \begin{cases} t, \text{ если } \| \varphi(\mathbf{x}(t)) \| \le \varepsilon \text{ и } t < t^+ \\ t^+ \text{ — иначе} \end{cases}, \tag{5}$$

где t^+ — предельное время процесса управления; ε — малая положительная величина; $\varphi(\mathbf{x}(t)) = [\varphi_1(\mathbf{x}(t))...\varphi_r(\mathbf{x}(t))]^T$,

$$\| \varphi(\mathbf{x}(t)) \| = \sqrt{\sum_{i=1}^{r} \varphi_i^2(\mathbf{x}(t))}.$$

Задан критерий качества управления

$$J = F(\mathbf{x}(t_f)) + \int_0^{t_f} f_0(\mathbf{x}(t), \mathbf{u}(t))dt \to \min. \tag{6}$$

Необходимо найти управление в виде функции от координат пространства состояний

$$\mathbf{u} = \mathbf{h}(\mathbf{x}). \tag{7}$$

Функция (7) удовлетворяет ограничениям (2) и обеспечивает нахождение такого управления, при котором любое частное решение системы (1) с функцией (7) в правой части вместо вектора управления с начальными условиями из области (3) достигает терминального многообразия за допустимое время (5) с оптимальным значением критерия качества (6).

МЕТОД СЕТЕВОГО ОПЕРАТОРА

Метод сетевого оператора разработан в 2006 г. профессором А.И. Дивеевым специально для решения задачи синтеза управления [2; 3]. Метод использует кодирование математического выражения в виде ориентированного графа, который представляется в компьютере целочисленной матрицей сетевого оператора. Например, математическое выражение

$$y = \ln(q_1)x_1^2 + q_2x_2^2$$

с набором элементарных функций $\rho_1(z) = z$, $\rho_2(z) = z^2$, $\rho_3(z) = \ln(z)$, $\chi_1(z_1, z_2) = z_1 + z_2$, $\chi_2(z_1, z_2) = z_1z_2$ кодируется целочисленной матрицей следующего вида:

$$\Psi = \begin{bmatrix} 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

и графом, представленным на рис. 1.

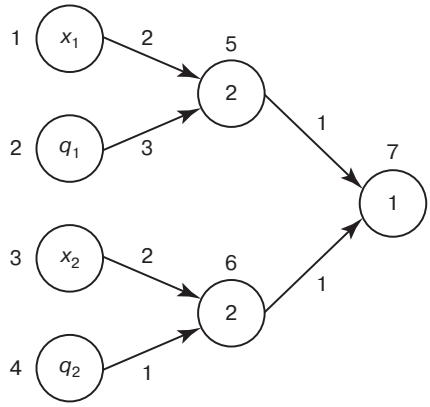

Рис. 1. Граф сетевого оператора
[**Fig. 1.** Network operator graph]

На рисунке 1 рядом с узлами размещены их номера, которые соответствуют номерам строк матрицы сетевого оператора.

Недостатком матрицы сетевого оператора является большое количество не используемых при вычислении ее элементов. В любой матрице сетевого оператора всегда рассматриваются только наддиагональные элементы, а из них в вычислении участвуют только не нулевые элементы.

Для более экономного представления графа сетевого оператора в памяти компьютера используем упорядоченные множества, которые на языке Python описываются встроенным типом list [4]. Сетевой оператор на основе списков представляется в следующем виде:

$$R = (G_1, ..., G_M), \quad (9)$$

где G_i — список операндов бинарной операции в узле $i + N_0$; N_0 — количество узлов источников в графе сетевого оператора.

Список операндов представляет собой множество упорядоченных пар чисел

$$G_i = ((\alpha_1^i, \beta_1^i), ..., (\alpha_{K_i}^i, \beta_{K_i}^i), (i + N_0, \psi_{i+N_0, i+N_0})), \quad (10)$$

где α_j^i — номер узла, из которого выходит дуга, входящая в узел $i + N_0$; β_j^i — номер унарной операции, связанной с дугой, соединяющей узел α_j^i и $i + N_0$; $\psi_{i+N_0, i+N_0}$ —

номер бинарной операции, связанной с узлом $i + N_0$, или диагональный элемент матрицы сетевого оператора, находящийся в строке $i + N_0$.

Заметим, что по правилам построения сетевого оператора все бинарные операции являются коммутативными и ассоциативными, поэтому они могут иметь более двух операндов и выполнять операции над ними в любой последовательности.

Определение 1. Представление сетевого оператора в виде вложенных списков называем **реестром (register) сетевого оператора**.

Рассмотрим граф математического выражения, приведенный на рис. 1. На графе имеем четыре узла источника, $N_0 = 4$. Рассмотрим узел $5 = N_0 + 1$. В узел входят две дуги из узлов источников 1 и 2, поэтому $\alpha_1^1 = 1$, $\alpha_2^1 = 2$. Дуга, выходящая из узла 1, связана с унарной операцией 2, а дуга, выходящая из узла 2, связана с унарной операцией 3, поэтому $\beta_1^1 = 2$, $\beta_2^1 = 3$. Сам узел 5 связан с бинарной операцией 2, $\psi_{5,5} = 2$. Первый список операндов имеет вид

$$G_1 = ((1,2), (2,3), (5,2)).$$

Определяем список операндов для узлов 6 и 7. В результате получаем следующий реестр сетевого оператора:

$$R = (((1,2), (2,3), (5,2)), ((3,2), (4,1), (6,2)), ((5,1), (6,1), (7,1))).$$

Для вычисления математического выражения по реестру сетевого оператора необходимо располагать упорядоченным множеством аргументов математического выражения

$$A = (a_1, ..., a_{N_0}), \tag{11}$$

где a_j — аргумент математического выражения, который в сетевом операторе является либо параметром, либо переменной, $a_N \in \{q_1, ..., q_p, x_1, ..., x_n\}, j = \overline{1, N_0}$.

Для множества аргументов математического выражения необходимо также знать номера узлов источников, с которыми связаны аргументы математического выражения

$$I_0 = (l_1, ..., l_{N_0}). \tag{12}$$

Задаем вектор для хранения промежуточных вычислений. Размерность вектора равна количеству списков реестра сетевого оператора

$$\mathbf{z} = [z_1 ... z_M]^T. \tag{13}$$

Для каждого списка вычисляем значение вектора (12)

$$z_i = \begin{cases} \rho_{\beta_1^i}\left(s_1^i\right), \text{ если } k_i = 1 \\ \chi_{\psi_{i+N_0, i+N_0}}\left(\rho_{\beta_1^i}\left(s_1^i\right), ..., \rho_{\beta_{k_i}^i}\left(s_{k_i}^i\right)\right) \text{ — иначе,} \end{cases} \tag{14}$$

где

$$s_j^i = \begin{cases} a_{l_{\alpha_j^i}}, \text{ если } \alpha_j^i \le N_0 \\ z_{\alpha_j^i - N_0} \text{ — иначе} \end{cases}, \; j = \overline{1, k_i}. \tag{15}$$

Для рассматриваемого примера получаем: $N_0 = 4$, $\mathrm{A} = (q_1, q_2, x_1, x_2)$, $\mathrm{I}_0 = (2,4,1,3)$, $\mathbf{z} = [z_1\, z_2\, z_3]^T$,

$$z_1 = \rho_2(x_1)\rho_3(q_1) = x_1^2 \ln(q_1),$$

$$z_2 = \rho_2(x_2)\rho_1(q_2) = x_2^2 q_2,$$

$$z_3 = \rho_1(z_1) + \rho_1(z_2) = x_1^2 \ln(q_1) + x_2^2 q_2.$$

Для поиска оптимального сетевого оператора используем принцип малых вариаций базисного решения [5]. В качестве малых вариаций используем те же вариации, что и для матрицы сетевого оператора [2]: замену унарной операции, замену бинарной операции, вставку унарной операции в список, удаление унарной операции из списка, если при этом в списке остается еще не менее двух элементов. Дополнительно используем вариацию замены компоненты вектора параметров. Все генетические операции скрещивания и мутации выполняем на множествах векторов, описывающих малые вариации сетевого оператора.

БИБЛИОТЕКА PYTHON ДЛЯ СИНТЕЗА СИСТЕМ УПРАВЛЕНИЯ МЕТОДОМ СЕТЕВОГО ОПЕРАТОРА

Библиотека состоит из трех классов, содержащих методы и поля для решения задачи синтеза управления методом символьной регрессии:

1. Base Genetics
2. Network Operator
3. Structure Genetics

Класс Base Genetics предназначен для проведения параметрической оптимизации. При инициализации объекта класса Base Genetics имеется возможность передать объекту вектор, наличие которого означает, что компоненты передаваемого вектора будут являться математическими ожиданиями для нормального распределения, из которого будут генерироваться значения компоненты индивидов популяции. В случае отсутствия данного вектора популяция генерируется согласно равномерному распределению.

Класс Network Operator предназначен для хранения базисной структуры и базисного вектора параметров, а также для отображения структуры в выходной вещественный вектор. При инициализации объекта класса Network Operator на вход объекту необходимо передать:

1) список унарных функций;
2) список бинарных функций;

3) список номеров входных узлов;

4) список номеров выходных узлов.

Класс Structure Genetics предназначен для проведения структурно-параметрической оптимизации. При инициализации объекта класса Structure Genetics на вход объекту необходимо подать объект типа Network Operator и объект-модель. Объект-модель не входит в библиотеку и должна быть реализована отдельно в виде класса, который имел бы интерфейс, согласно которому объект-модель реализовывал численное интегрирование дифференциальных уравнений, жестко запрограммированных в классе. Чтобы объект-модель могла быть интегрирована с объектом класса Network Operator, необходимо, чтобы в ней была реализована функция set_control_function, принимающая на вход объект функции и присваивающая данную функцию внутренней переменной.

Семантика кодируемой структуры

Для кодирования математического выражения использовались стандартные типы языка Python — список (*list*) и кортеж (*tuple*) [4]. Одно математическое выражение представляется в виде списка, состоящего из других списков, количество которых фиксировано базисным решением. Каждый из подсписков состоит минимум из трех кортежей, длина которых равна двум. Первая позиция кортежа — это номер узла, вторая позиция — это номер операции. Если кортеж является последним в подсписке (далее диагональный), то операция является бинарной, иначе — унарной. Таким образом, чтобы структура графа математического выражения была неразрывной и ациклической (ссылка на методичку по сетевому оператору), необходимо, чтобы одновременно выполнялись два условия:

1) кортежи подсписка должны содержать только входные узлы либо узлы диагональных кортежей предыдущих подсписков;

2) каждый элемент из множества входных и диагональных узлов должен быть использован в качестве узла, над которым проводится унарная операция, минимум один раз.

Структура вариаций базисного решения

В классе Structure Genetics предусмотрено пять типов вариаций:

0: Замена бинарной операции

1: Замена унарной операции

2: Добавление унарной операции

3: Удаление унарной операции

4: Изменение параметра

Каждый индивид популяции объекта типа Structure Genetics представляет собой матрицу $n \times 4$, где n — число вариаций. При скрещивании двух родителей образуется пул из всех вариаций обоих родителей. К этому пулу добавляются новые вариации четвертого типа. Далее для каждого ребенка из общего пула вариаций случайно выбирается n вариаций, причем одна вариация не может быть выбрана более одного раза.

Начальная популяция генерируется при помощи стандартного типа *dict*. Таким образом, достигается большая разнообразность, так как каждый индивид имеет уникальное значение функционала, не превышающее некоторой заданной границы.

ПРИМЕР ИСПОЛЬЗОВАНИЯ БИБЛИОТЕКИ ДЛЯ СИНТЕЗА УПРАВЛЕНИЯ

Библиотека была протестирована на гусеничном мобильном роботе, модель которого имеет следующий вид:

$$\dot{x} = 0{,}5(u_1 + u_2)\cos(\theta),$$

$$\dot{y} = 0{,}5(u_1 + u_2)\sin(\theta),$$

$$\dot{\theta} = 0{,}5(u_1 - u_2). \qquad (16)$$

На рисунке 2 представлен примерный вид мобильного робота.

Рис. 2. Мобильный робот
[**Fig. 2.** Mobile robot]

Для системы (16) заданы начальные условия: $x(0) = 10$, $y(0) = 10$, $\theta(0) = 0$. Управление объектом ограничено

$$-10 \le u_i \le 10,\ i = 1{,}2. \qquad (17)$$

Заданы терминальные условия

$$x_f = 10,\ y_f = 10,\ \theta_f = 0. \qquad (18)$$

Заданы фазовые ограничения

$$r^* - \sqrt{(x^* - x)^2 + (y^* - y)^2} \le 0, \qquad (19)$$

где $r^* = 2{,}5$, $x^* = 5$, $y^* = 5$.

Задан критерий качества

$$J = t_f + \sqrt{(x_f - x(t_f))^2 + (y_f - y(t_f))^2} + \int_0^{t_f} \vartheta\left(r^* - \sqrt{(x^* - x)^2 + (y^* - y)^2}\right) dt \to \min, \quad (20)$$

где t_f — время окончания процесса управления;

$$t_f = \begin{cases} t, \text{если } t < t^+ \text{ и } \sqrt{(x_f - x(t_f))^2 + (y_f - y(t_f))^2} < \varepsilon \\ t^+ \text{ — иначе} \end{cases}, \quad (21)$$

где $t^+ = 2{,}5$ с, $\varepsilon = 0{,}01$, J(a) — функция Хевисайда,

$$\vartheta(a) = \begin{cases} 1, \text{если } a > 0 \\ 0 \text{ — иначе} \end{cases}.$$

Необходимо найти функцию управления в виде (7), чтобы она удовлетворяла ограничениям (17) и обеспечивала достижение объектом терминального состояния (18) с оптимальным значением критерия качества (20). Для решения задачи используем метод сетевого оператора и реализацию метода с помощью классов Python библиотеки.

Определим множество аргументов (10) искомой функции управления

$$\text{A} = (q_1, q_2, q_3, x, y, \theta), \text{I}_0 = (2, 4, 6, 1, 3, 5).$$

Определим базисное решение в форме линейной обратной связи по координатам вектора пространства состояний

$$u_i = q_1 x + q_2 y + q_3 \theta, \; i = 1, 2,$$

где $q_i = 1$, $i = 1, 2, 3$.

Закодированное в форме реестра сетевого оператора базисное решение имеет вид

$$\text{R} = (((1,1), (2,1), (6,2)), ((3,1), (4,1), (7,2)), ((5,1), (6,1), (8,2)), \\ ((6,1), (7,1), (8,1), (9,1)), ((6,1), (7,1), (8,1), (10,1))).$$

Было проведено несколько десятков экспериментов. Три наилучших найденных решения имели следующий вид.

Решение 1

$$u_1 = q_1 + \sin(q_2 y \operatorname{arctg}(q_2)) + q_2 \theta x + \ln(x), \quad (22)$$

$$u_2 = \min\{\max(q_2, x), -q_2 y \operatorname{arctg}(q_2), \cos(x)\}, \quad (23)$$

где $q_1 = -3{,}15283$, $q_2 = 9{,}942394$, $q_3 = -7{,}374617$.

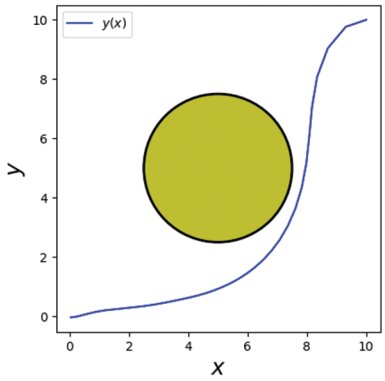

Рис. 3. Траектория движения робота на плоскости для решения (22), (23)
[**Fig. 3.** The robot's trajectory on the plane for the solution (22), (23)]

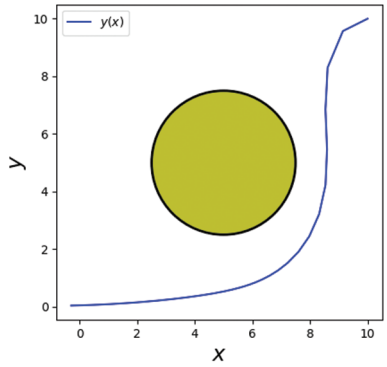

Рис. 4. Траектория движения робота на плоскости для решения (24), (25)
[**Fig. 4.** The robot's trajectory on the plane for the solution (24), (25)]

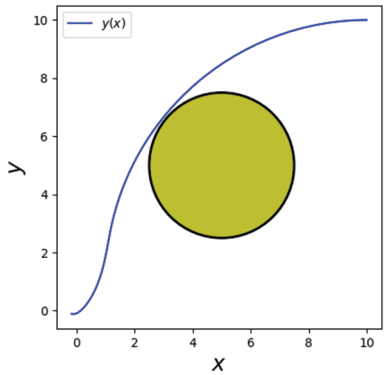

Рис. 5. Траектория движения робота на плоскости для решения (26), (27)
[**Fig. 5.** The robot's trajectory on the plane for the solution (26), (27)]

Для решения (22), (23) функционал имел значение: $J = 2{,}381070$. Результаты моделирования с решением (22), (23) приведены на рис. 3.

Решение 2

$$u_1 = \min\begin{Bmatrix}\ln(q_1), \operatorname{arctg}(q_2), 1+q_3\theta(-x)+\\ +\operatorname{sgn}(q_2 y \ln(q_3))\sqrt{|q_2 y \ln(q_3)|}\end{Bmatrix}, \quad (24)$$

$$u_2 = \min\{\ln(q_1), \operatorname{arctg}(q_2), 1 + q_2 y \ln(q_3)\}, \quad (25)$$

где $q_1 = 1{,}340758$, $q_2 = 8{,}966063$, $q_3 = 9{,}291124$.

Для решения (24), (25) функционал имел значение: $J = 2{,}354040$. Результаты моделирования с решением (24), (25) приведены на рис. 4.

Решение 3

$$u_1 = \min\begin{Bmatrix}\exp(\max\{\ln(q_1), x, \cos(q_1)\}),\\ q_2 y, \cos(q_3)\theta_3^3(-q_3),\\ \operatorname{sgn}(q_3)\sqrt{|q_3|}, \sin(q_1)\end{Bmatrix}, \quad (26)$$

$$u_2 = q_2 + y, \quad (27)$$

где $q_1 = -13{,}161914$, $q_2 = 4{,}006537$, $q_3 = 8{,}425214$.

Для решения (26), (27) функционал имел значение: $J = 2{,}400259$. Результаты моделирования с решением (26), (27) приведены на рис. 5.

ВЫВОДЫ

Описанная в статье Python библиотека классов позволяет решать задачи структурного синтеза, применяя при этом новую, экономную структуру кодирования формулы. Новая структура, названная реестром сетевого оператора, в отличие от матрицы сетевого оператора не включает нулевых элементов, не участвующих в вычислении математического выражения. Проведен вычислительный эксперимент по решению задачи синтеза управления мобильным

роботом, движущимся по плоскости при наличии фазовых ограничений. Результаты моделирования полученных при синтезе различных систем управления показали, что найденная функция управления обеспечивает достижение объектом терминального состояния с близким к оптимальному значению функционала без нарушения фазовых ограничений.

СПИСОК ЛИТЕРАТУРЫ

[1] *Дивеев А.И.* Приближенные методы решения задачи синтеза оптимального управления. М.: ВЦ РАН, 2015. 184 с.
[2] *Дивеев А.И.* Метод сетевого оператора. М.: ВЦ РАН, 2010. 178 с.
[3] *Дивеев А.И.* Численный метод сетевого оператора для синтеза системы управления с неопределенными начальными значениями // Известия РАН Теория и системы управления. 2012. № 2. С. 63—78.
[4] Python 3.5.5 documentation // www.python.org URL: https://docs.python.org/3.5/tutorial/introduction.html#lists (дата обращения: февраль 2018).
[5] *Diveev A.I.* Small Variations of Basic Solution Method for Non-numerical Optimization // Proceedings of 16th IFAC Workshop on Control Applications of Optimization, CAO' 2015. October 6th—9th 2015 Garmisch-Partenkirchen. P. 28—33.

История статьи:
Дата поступления в редакцию: 01 марта 2018
Дата принятия к печати: 10 мая 2018

Для цитирования:
Дивеев А.И., Доценко А.В. Библиотека Python для синтеза интеллектуальных систем управления // Вестник Российского университета дружбы народов. Серия: Инженерные исследования. 2018. Т. 19. № 2. С. 177—189. DOI 10.22363/2312-8143-2018-19-2-177-189

Сведения об авторах:
Дивеев Асхат Ибрагимович — доктор технических наук, профессор, заведующий сектором проблем кибернетики, Федеральный исследовательский центр «Информатика и управление», Российская академия наук, профессор департамента механики и мехатроники Инженерной академии, Российский университет дружбы народов. *Область научных интересов*: вычислительные методы для решения задач управления. *Контактная информация*: e-mail: aidiveev@mail.ru

Доценко Антон Викторович — аспирант департамента механики и мехатроники Инженерной академии, Российский университет дружбы народов. *Область научных интересов*: методы оптимизации, эволюционные алгоритмы, искусственные нейронные сети, машинное обучение, вычислительные методы решения задач оптимального управления. *Контактная информация*: e-mail: anton.dozenko@gmail.com

PYTHON PACKAGE FOR INTELLIGENT CONTROL SYSTEMS SYNTHESIS

A.I. Diveev[1,2], A.V. Dotsenko[2]

[1] Institution of Russian Academy of Sciences Dorodnicyn Computing Centre of RAS
40, Vavilova str., Moscow, 119333, Russian Federation
[2] Peoples' Friendship University of Russia (RUDN University)
6, Miklukho-Maklaya str., Moscow, 117198, Russian Federation

Abstract. This article is devoted to the desription of apython library based on symbolic regression methods for control systems synthesis problem. Control sysnthesis is becoming more and more relevant, gaining particular importance in view of the rapid development of robotics. Usually, practicians and engineers apply template-type regulators when modeling, and then select optimal parameters for them. At a time when the computing power of PC's has reached its peak, and programming languages have become extremely expressive due to the high level of abstraction and the vastness of libraries, it is better to implement the synthesis in the form of a library. Python was chosen as the language for synthesis implementation. According to the authors of the article, Python is a convenient language for programming matrix and vector calculations thanks to the numpy package. Moreover, the share of projects written in Python in the web service for hosting Github has been steadily increasing recently, which indicates the support of the language from the developer community. This article describes how to use the package to solve the problem of control synthesis. The authors provide the description of the symbolic regression method, the network operator and algorithms for finding the optimal solution using the principle of small variations of the basic solution. In the experimental part of the article, an example of how to use the library to solve the problem of synthesis of control of a mobile robot moving on a planewith obstacles is considered.

Key words: control synthesis, python library, symbolic regression method, optimal control, mobile robot

REFERENCES

[1] Diveev A.I. Priblizhennye metody resheniya zadachi sinteza optimal'nogo upravleniya [Approximate methods for solving the optimal control synthesis problem]. Moscow: Dorodnicyn Computing Centre of RAS Publ., 2015. 184 p. (In Russ.)
[2] Diveev A.I. Metod setevogo operatora [Network operator]. Moscow: Dorodnicyn Computing Centre of RAS Publ., 2010. 178 p. (In Russ.)
[3] Diveev A.I. Chislennyi metod setevogo operatora dlya sinteza sistemy upravleniya s neopredelennymi nachal'nymi znacheniyami [Network operator numerical method for the control system synthesis with undefined initial values]. *Journal of Computer and Systems Sciences International*. 2012. (2). P. 63—78. (In Russ.)
[4] Python 3.5.5 documentation // www.python.org URL: https://docs.python.org/3.5/tutorial/introduction.html#lists (access date: Fabuary 2018).
[5] Diveev A.I. Small Variations of Basic Solution Method for Non-numerical Optimization // Proceedings of 16th IFAC Workshop on Control Applications of Optimization, CAO' 2015. October 6th—9th 2015 Garmisch-Partenkirchen. P. 28—33.

Article history:
Received: March 01, 2018
Accepted: May 10, 2018

For citation:
Diveev A.I., Dotsenko A.V. (2018). Python package for intelligent control systems synthesis. *RUDN Journal of Engineering Researches*, 19(2), 177—189. DOI 10.22363/2312-8143-2018-19-2-177-189

Bio Note:
Askhat I. Diveev — Doctor of Technical Sciences, Professor, chief of Sector of Cybernetic Problems, Federal Research Centre "Computer Science and Control" of Russian Academy of Sciences, professor of Department of Mechanics and Mechatronics, Engineering Academy, Peoples' Friendship University of Russia. *Research interests*: Computational methods for problems of control. *Contact information*: e-mail: aidiveev@mail.ru

Anton V. Dotsenko — post-graduate student of Department of Mechanics and Mechatronics, Engineering Academy, Peoples' Friendship University of Russia. *Research interests*: Optimization algorithms, evolutionary algorithms, artificial neural networks, machine learning, computational methods for problems of optimal control. *Contact information*: e-mail: anton.dozenko@gmail.com