Оригинальная статья / Original article

УДК 621.865.8

https://doi.org/10.21869/2223-1560-2024-28-2-71-91

Программная реализация иерархического подхода решения обратной задачи кинематики робота– манипулятора

Е. М. Апокин [1], В. А. Хандожко [1] ✉

[1] Брянский государственный технический университет,
бульвар 50 лет Октября, д. 7, г. Брянск 241035, Российская Федерация

✉ e-mail: vichandozhko@gmail.com

Резюме

Цель исследования. *Разработка программной реализации иерархического подхода к решению обратной задачи кинематики робота-манипулятора с произвольным количеством сочленений. Разработанная библиотека может быть использована как в составе САПР машиностроительного назначения с поддержкой функционала инженерного анализа (CAE), так и для разработки программного обеспечения встроенных систем управления многокоординатными манипуляционными машинами.*

Методы. *Идея подхода заключается в итерационном приближении обобщенных координат в последовательности, определенной разработчиком. Количество необходимых итераций определяется требу-емой точностью решения. В качестве исходных данных для решения обратной задачи кинематики требуется «матрица манипулятора» 4 × 4 в символьном виде, начальные значения обобщенных коорди-нат, и «матрица манипулятора» 4 × 4, описывающая финальные положение и ориентацию схвата. Для реализации выбран язык программирования Си стандарта ANSI C, поскольку он обеспечивает доста-точную портативность и близость к аппаратуре.*

Результаты. *Результатом работы является библиотека языка программирования Си для ЭВМ под управлением операционных систем семейства Unix (например, GNU/Linux, freeBSD, openBSD, macOS, Solaris), которая предоставляет функции, необходимые для решения обратной задачи кинематики робота–манипулятора.*

Заключение. *В статье показан пример использования данной библиотеки для трехзвенного робота–манипулятора, а также построена траектория его движения по десяти точкам. Использование предложенной в статье библиотеки libreRGM3 может быть эффективным способом решения ОЗК при моделировании движения робота в САПР (например, OpenSCAD [8]), в учебных или исследовательских целях, в условиях работы с ЭВМ без графического интерфейса или с программируемым логическим контроллером и микроконтроллером.*

Ключевые слова: *робот; иерархический подход; программа; алгоритм; UNIX.*

Конфликт интересов: *Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.*

Для цитирования: Апокин Е. М., Хандожко В. А. Среднегодовые температуры воды в тепловых сетях криолитозоны Программная реализация иерархического подхода решения обратной задачи кинематики робота–манипулятора // Известия Юго-Западного государственного университета. 2024. Т. 28, №2. С. 71-91. https://doi.org/10.21869/2223-1560-2024-28-2-71-91.

Поступила в редакцию 21.02.2024 *Подписана в печать 17.04.2024* *Опубликована 25.06.2024*

Software implementation of a hierarchical approach to solving the inverse problem of kinematics of a robot manipulator

Elisey M. Apokin [1], Viktor A. Khandozhko [1] ✉

[1] Bryansk State Technical University,
50 Let Oktyabrya ave. 7, Bryansk 241035, Russian Federation

✉ e-mail: vichandozhko@gmail.com

Abstract

Purpose of research. *Development of a software implementation of a hierarchical approach to solving the inverse problem of the kinematics of a robot manipulator with an arbitrary number of joints. The developed library can be used both as part of mechanical engineering CAD systems with support for engineering analysis (CAE) functionality, and for the development of software for embedded control systems for multi-axis handling machines.*

Methods. *The idea of the approach is to iteratively approximate generalized coordinates in a sequence defined by the developer. The number of required iterations is determined by the required accuracy of the solution. As initial data for solving the inverse kinematics problem, a 4 × 4 "manipulator matrix" in symbolic form, initial values of generalized coordinates, and a 4 × 4 "manipulator matrix" describing the final position and orientation of the gripper are required. The ANSI C programming language was chosen for implementation because it provides sufficient portability and proximity to the hardware.*

Results. *The result of the work is a library of the C programming language for computers running operating systems of the Unix family (for example, GNU/Linux, freeBSD, openBSD, macOS, Solaris), which provides the functions necessary to solve the inverse problem of the kinematics of a robot manipulator.*

Conclusion. *The article shows an example of using this library for a three-link robotic manipulator, and also plots the trajectory of its movement along ten points. Using the libreRGM3 library proposed in the article can be an effective way to solve OZK when modeling robot movement in CAD (for example, OpenSCAD [8]), for educational or research purposes, when working with a computer without a graphical interface or with a programmable logic controller and microcontroller.*

Keywords: *robot; hierarchical approach; program; algorithm; UNIX.*

Conflict of interest. *The authors declare the absence of obvious and potential conflicts of interest related to the publication of this article.*

For citation*:* Apokin E. M., Khandozhko V. A. Software implementation of a hierarchical approach to solving the inverse problem of kinematics of a robot manipulator. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University*. 2024; 28(2): 71-91 (In Russ.). https://doi.org/ 10.21869/2223-1560-2024-28-2-71-91.

Received 21.02.2024 *Accepted 17.04.2024* *Published 25.06.2024*

***

Введение

Для получения управляющих воздействий для приводов роботов используется решение обратной задачи кинематики (далее по тексту ОЗК) [1]. Данная задача связана с большим количеством сложных расчетов, поэтому для ее решения часто используются ЭВМ.

В настоящее время существует ряд проприетарного программного обеспечения и библиотек, позволяющих решать ОЗК роботов–манипуляторов. Например, Robodk [2], Dyn-Soft RobSim [3], Kuka Load [4], Универсальный механизм [5], Robotics System Toolbox [6] для MatLab и т. д. Стоит отметить свободные программные решения для этой цели, библиотеку OpenRAVE [7] для языка программирования C++ и Robot Operating System [8], представляющую собой операционную систему для решения большого количества задач робототехники. Однако OpenRAVE не поддерживается больше десяти лет, а Robot Operating System, вероятно, является слишком сложным, чтобы быть использованным при решении небольших или узкоспециализированных задач.

В этой работе представлена новая разработка – библиотека языка программирования Си [9], которая позволяет решать обратную задачу кинематики для любых разомкнутых кинематических цепей с произвольным количеством сочленений, поскольку реализует иерархический подход к решению обратной задачи кинематики [10]. Библиотека распространяется под свободной лицензией MIT, что означает, что исходный код программы открыт, все пользователи могут его распространять, вносить в него изменения и делиться своими изменениями. Библиотека использует стандарт ANSI C, стандартную библиотеку языка Си и в функциях ввода-вывода системные вызовы UNIX. То есть, библиотека, скомпилированная из исходного текста, будет работать на любой целевой платформе под управлением операционных системы семейства UNIX, например, GNU/Linux, freeBSD, openBSD, macOS, Solaris, и при небольших изменениях, на машинах под управлением Windows. В статье представлены краткое описание иерархического подхода к решению ОЗК, документация по функциям, которые предоставляет библиотека, показан пример использования данной библиотеки для трехзвенного робота-манипулятора, а также построена траектория его движения по десяти точкам.

Материалы и методы

Основные положения иерархического подхода включают в себя [10]:

1. Выбор ограниченного набора сочленений, необходимых для выполнения движения.

2. Назначение данным сочленениям последовательности элементарных движений и ограничения этого движения.

3. Приближение последовательным итерационным способом к необходимой точности решения.

Исходное положение исполнительного механизма определяют исходными значениями обобщенных координат q_0, q_1, .. , q_n.

Конечные положения и ориентация схвата робота–манипулятора в системе координат рабочего пространства представлены матрицей преобразования координат ${}^0A_n=T$, иначе – «матрицей манипулятора» [11].

Для решения ОЗК выполняют итерационное приближение к целевым положению и ориентации схвата путем изменения значений обобщенных координат последовательности движения выбранных сочленений [10].

Каждая обобщенная координата изменяется в соответствии с формулой

$$q_i^{(j)} = q_i^{(j-1)} + (-1)^k \cdot \Delta q_i, \qquad (1)$$

где $q_i^{(j)}$ и $q_i^{(j-1)}$ – значения обобщенных координат на текущей и предыдущей итерациях соответственно; k – коэффициент, который определяет знак изменения обобщенной координаты; Δq_i – величина приращения обобщенной координаты [10].

Величина приращения Δq_i может назначаться произвольно или рассчитываться формуле

$$\Delta q_i = \frac{s_i}{N} = \frac{q_i^{max} - q_i^{min}}{N}, \qquad (2)$$

где s_i – величина диапазона изменения обобщенной координаты q_i; q_i^{max} и q_i^{min} – это соответственно максимальное и минимальное отклонение обобщенной координаты q_i; N – коэффициент, который определяет точность решения [10].

После каждой итерации выполняют ряд проверок:

– влияет ли обобщенная координата на ориентацию схвата или на расстояние от схвата до цели;

– достигнута ли обобщенная координата пределов ее изменения, установленных разработчиком.

При уменьшении расстояния до цели или приближении ориентации схвата к заданному положению корректировка не требуется. Итерационный процесс для текущей обобщенной координаты продолжается.

При удалении обобщенной координаты от цели либо увеличении углового отклонения от заданного положения, знак приращения этой координаты изменяют на противоположный.

При достижении предельных значений обобщенной координаты соответствующее ей приращение также меняет свой знак.

При сохранении удаления от заданной координаты после изменения знака приращения последнего изменение обобщенной координаты отменяют и пере-

ходят к следующей обобщенной координате.

Каждому из q_0, q_1, .., q_n сочленений, отвечающих за решение, разработчик назначает определенную функцию в реализации заданного движения [10]. Например, одно сочленение может отвечать за поворот к заданной позиции, два других – за сближение с заданной позицией, сочленения, находящиеся ближе к схвату – за ориентацию схвата и т. д.

При решении ОЗК функции сочленений необходимо учитывать в процессе проверки соответствующих обобщенных координат [12]. Так, если сочленение участвует только в сближении с целью, то для него проверяют изменение расстояния от схвата до заданной позиции. Если функция сочленения связана с ориентацией схвата – для него проверяют угловые отклонения ортов системы координат схвата от осей системы координат рабочего пространства или других осей, выбранных разработчиком. Если функция сочленения связана с положением и ориентацией, то проводят обе вышеописанные проверки.

Когда решение состоит из последовательности движений, то вычислительный процесс сводится к ряд последовательно выполняемых циклических действий. Решение завершается при условии достижения необходимой точности положения и ориентации схвата.

В состав заголовочного файла rgm3_reverse_kinematic входят следующие группы функции.

1. Функции для определения величины приращения обобщенной координаты.

Функция set_iteration_step вычисляет величину приращения по формуле (2) и возвращает величину приращения для обобщенной координаты q_i.

Функция set_simple_iteration_step позволяет в качестве величины диапазона изменения обобщенных координат задать произвольное число.

Вычисляется величина приращения по данной формуле:

$$\Delta q_i = \frac{\Delta_i}{N},$$

где Δ_i – произвольный диапазон. Функция возвращает величину приращения для обобщенной координаты q_i.

2. Функции для итерационного приближения к требуемым позиции и ориентации.

Функция do_iter_step_position производит итерацию обобщенной координаты с приближением к целевой позиции.

Обобщенная координата изменяется в соответствии с формулой (1).

В случае, если приблизить схват манипулятора к целевой позиции удалось, функция возвращает новое значение итерируемой обобщенной координаты $q_i^{(j)}$. В противном случае, функция возвращает старое значение обобщенной координаты $q_i^{(j-1)}$, т. е. такое же, какое ей было передано.

Функция do_iter_step_orientation иден-тична предыдущей за исключением того, что она производит приближение для сочленений, которые отвечают

за ориентацию схвата, а не за сближение с целевым положением.

3. Функции для проверки достижения обобщенной координатой пределов ее изменения.

Библиотека предоставляет несколько функций для обработки ситуаций, в которых обобщенная координата q_i принимает недопустимое значение, а именно, превышает предел изменения, установленный разработчиком.

Функция is_limit_reached позволяет детектировать этот случай. Функция возвращает 1 в случае $q_i > q_i^{max}$ *or* $q_i < q_i^{min}$ и возвращает 0 в случае $q_i < q_i^{max}$ *or* $q_i > q_i^{min}$.

Следующие функции стоит использовать в связке с функцией is_limit_reached.

Функция avoid_position_limiter возвращает новое значение обобщенной координаты q_i, при котором схват манипулятора принимает наиболее удаленное положение от целевого.

Функция возвращает одно из следующих значений: q_i^{min}, $q_i^{average} = \frac{q_i^{max}+q_i^{min}}{2}$, q_i^{max} при котором схват манипулятора окажется в наиболее удаленном положении от целевого.

Функция avoid_orientation_limiter аналогична avoid_position_limiter, за исключением того, что возвращает положение, наиболее удаленное по ориентации, а не по позиции. При этом, проверяются угловые отклонения между осями *X* и *X'*, *Y* и *Y'*, *Z* и *Z'*, где *OXYZ* – система координат схвата в базовой системе координат, при значениях обобщенных координат q_0, q_1, .., q_n переданных в функцию; а *OX'Y'Z'* – система координат схвата в базовой системе координат, при новых значениях обобщенных координат.

4. Функции для диагностики итерационного приближения.

Библиотека предоставляет несколько функций для диагностики итерационного приближения. Диагностика требуется для расчетов ошибки по позиционированию и ориентации, а также для получения соответствующих матриц.

Ниже описаны три функции, которые используются для получения «матрицы манипулятора» и диагностических матриц.

Функция get_tcp_matrix возвращает «матрицу манипулятора» при текущих значениях обобщенных координат q_0, q_1, .., q_n.

Функция возвращает указатель на одномерный динамический массив на 16 элементов типа double, который содержит «матрицу манипулятора» в численном виде при текущих значениях обобщенных координат q_0, q_1, .., q_n.

Функция get_absolute_error_matrix используется для получения матрицы абсолютных ошибок.

Функция возвращает указатель на одномерный динамический массив на 16 элементов типа double, который содержит матрицу абсолютных ошибок по позиционированию и ориентации.

Абсолютная ошибка вычисляется по формулам:

$\Delta r_u = r_u^{final} - r_u^{(j)}$;

$\Delta\gamma_{uw} = \gamma_{uw}^{final} - \gamma_{uw}^{(j)}$,

где $\vec{r}$ – радиус-вектор, проведенный из начала базовой системы координат в начало системы координат схвата; γ – отклонение между соответствующими осями базовой системы координат и системы координат схвата (может выражаться в разности как и между косинусами углов, так и между углами); индексы u и w – соответствуют перечислениям осей базовой системы координат и системы координат схвата.

Возвращаемая матрица имеет вид

$$\begin{pmatrix} \Delta\gamma_{xx} & \Delta\gamma_{xy} & \Delta\gamma_{xz} & \Delta r_x \\ \Delta\gamma_{yx} & \Delta\gamma_{yy} & \Delta\gamma_{yz} & \Delta r_y \\ \Delta\gamma_{zx} & \Delta\gamma_{zy} & \Delta\gamma_{zz} & \Delta r_z \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Функция get_relative_error_matrix ис-пользуется для получения матрицы относительных ошибок, представленных в процентах.

Относительная ошибка вычисляется по формулам:

$$\Delta^{\%} r_u = \frac{r_u^{final} - r_u^{(j)}}{r_u^{final}} \cdot 100\%,$$

$$\Delta^{\%}\gamma_{uw} = \frac{\gamma_{uw}^{final} - \gamma_{uw}^{(j)}}{\gamma_{uw}^{final}} \cdot 100\%.$$

Возвращаемая матрица имеет вид

$$\begin{pmatrix} \Delta^{\%}\gamma_{xx} & \Delta^{\%}\gamma_{xy} & \Delta^{\%}\gamma_{xz} & \Delta^{\%} r_x \\ \Delta^{\%}\gamma_{yx} & \Delta^{\%}\gamma_{yy} & \Delta^{\%}\gamma_{yz} & \Delta^{\%} r_y \\ \Delta^{\%}\gamma_{zx} & \Delta^{\%}\gamma_{zy} & \Delta^{\%}\gamma_{zz} & \Delta^{\%} r_z \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Следующая функция окажется полезной, если ошибку по ориентации следует искать не как разность косинусов $cos(\alpha) - cos(\beta)$, а как разность углов $\alpha - \beta$.

Функция convert_tcp_matrix_cos2rad преобразует направляющие косинусы «матрицы манипулятора» в углы поворота между соответствующими осями, выраженные в радианах, и возвращает указатель на новый, измененный, динамический массив, который содержит «матрицу манипулятора», идентичную той, что была ему передана в качестве аргумента, с матрицей поворота, выраженной в углах поворота (радианы), а не направляющих косинусах.

Функция get_max_error_of_position возвращает значение максимальной ошибки по позиционированию. Ошибка берется из диагностической матрицы, указатель на которую передается в функцию. Соответственно, ошибка может быть выражена в абсолютных или относительных величинах, в зависимости от того, возвращенное значение какой функции ей передано get_absolute_error_matrix или get_relative_error_matrix.

Функция get_average_error_of_position отличается от предыдущей тем, что возвращает значение средней ошибки по позиционированию.

Функции get_max_error_of_orientation и get_average_error_of_orientation возвращают значения максимальной и средней ошибки по ориентации соответственно.

В состав заголовочного файла rgm3_output входят следующие функции для получения информации от программы.

1. Функции для вывода информации в стандартный поток.

С помощью функции term_print_matrix осуществляется вывод в стандартный поток матриц произвольной размерности.

Фукнция term_print_error выводит в стандартный поток значение максимальных и средних ошибок по ориентации и позиционированию.

Функция term_print_gen_coord выводит в стандартный поток номер итерации и значения обобщенных координат $q_0^{(j)}$, $q_1^{(j)}$,.., $q_n^{(j)}$ на данной итерации.

2. В состав этого же заголовочного файла входят следующие функции для записи в файл.

Данные функции позволяют записать информацию в файл формата csv (comma-separated values). Файлы данного типа удобно обрабатывать электронными таблицами, например, LibreOffice [13], OnlyOffice [14] или Microsoft Excel [15].

Их следует применят последовательно.

a. Функция csv_init_file создает файл в соответствующей директории с соответствующим именем, и открывает связанный с ним поток ввода.

b. Функция csv_create_field создает поле (строку) таблицы, в которую следует заносить соответствующую информацию.

c. Функция csv_put_info записывает информацию в необходимое поле (строку) таблицы.

d. Функция csv_write_file записывает информацию с оперативной памяти компьютера в постоянное запоминающее устройство, т. е. записывает таблицу в файл.

e. Функция csv_close_file закрывает поток ввода–вывода, который ей передан в аргументе.

Результаты и их обсуждение

В качестве объекта исследования выберем манипулятор, кинематическая схема которого представлена на рис. 1.

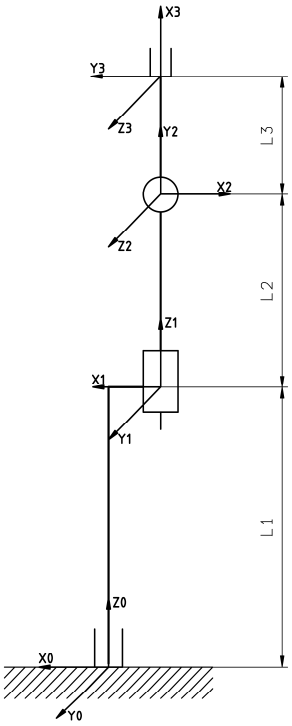

Рис. 1. Кинематическая схема робота–манипулятора

Fig. 1. Kinematic diagram of a robot manipulator

Зададим для робота–манипулятора длины звеньев L_1, L_2, L_3 и укажем их значения (табл. 1, 2).

Как было отмечено в [11], метод Денавита–Хартенберга (ДХ) позволяет сократить количество координат, однозначно определяющих тело (систему координат) в пространстве, с шести до четырех.

Таблица 1. Длины звеньев

Table 1. Link lengths

L_1, мм	L_2, мм	L_3, мм
600	500	250

Таблица 2. ДХ–параметры робота–манипулятора

Table 2. DX–parameters of the robot manipulator

Сочленение i / Joint i	a_i, мм / a_i, mm	α_i, рад / α_i, rad	d_1, мм / d_1, mm	θ_i, рад / θ_i, rad
0	0	0	L_1	θ_0
1	0	$\frac{\pi}{2}$	$L_2 + d_1$	π
2	L_3	0	0	$\theta_2\frac{+\pi}{2}$

Согласно математическому аппарату, сформируем матрицу манипулятора. Вычисления будем производить в системе символьных вычислений Maxima [16]. В итоге имеем

$$ {}^0A_3=T=\begin{pmatrix} \cos\theta_0\sin\theta_2 & \cos\theta_0\cos\theta_2 & -\sin\theta_0 & 25\cos\theta_0\sin\theta_2 \\ \sin\theta_0\sin\theta_2 & \sin\theta_0\cos\theta_2 & \cos\theta_0 & 25\sin\theta_0\sin\theta_2 \\ \cos\theta_2 & -\sin\theta_2 & 0 & 25\cos\theta_2+d_1+L_1+L_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}. $$

Пусть манипулятор занимает положение, представленное на рис. 2. Также зададим ограничения изменений обобщенных координат. Отразим эту информацию в табл. 3.

Таблица 3. Начальные значения и ограничения изменения обобщенных координат

Table 3. Initial values and limitations of changes in generalized coordinates

Сочленение i / Joint i	q_i^{min}	q_i	q_i^{max}
0	-2π, рад	0, рад	2π, рад
1	-300, мм	0, мм	300, мм
2	$\frac{-5\pi}{6}$, рад	$\frac{\pi}{4}$, рад	$\frac{5\pi}{6}$, рад

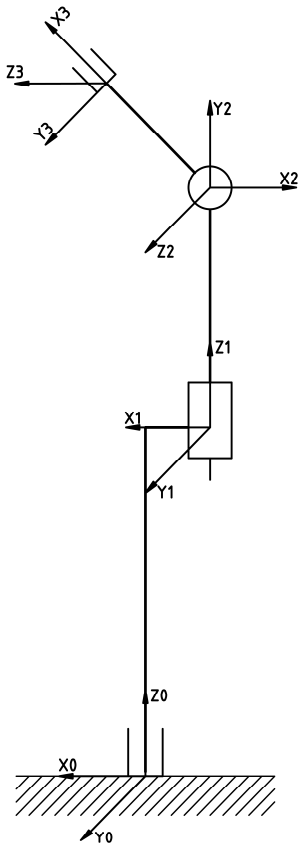

Рис. 2. Исходное положение робота–манипулятора

Fig. 2. The initial position of the robot arm

Необходимо выбрать сочленения для движения. В данном случае это все сочленения манипулятора. Сочленение q_0 отвечает за поворот манипулятора относительно стойки на первом этапе решения, сочленение q_2 – за достижение требуемой ориентации схвата, сочленение q_1 будет отвечать за достижение требуемой позиции на последнем этапе решения.

$q_0 \mapsto q_2 \mapsto q_1$.

Таким образом, имеем три сочленения, для которых необходимо решить ОЗК.

Пусть траектория движения схвата манипулятора (дальше *TCP – Tool Center Point*) будет прямая. Зададим прямую в базовой системе координат $OX_0Y_0Z_0$. Для задания прямой воспользуемся уравнением прямой в пространстве, проходящей через две точки.

$$\frac{x-x_0}{x_1-x_0}=\frac{y-y_0}{y_1-y_0}=\frac{z-z_0}{z_1-z_0}.$$

Зададим произвольные крайние точки прямой в рабочем пространстве робота. Пусть P_0 = (250, 0, 1100) и P_1 = = (25, 225, 920). В таком случае получим

$$\frac{250-x}{5}=\frac{y}{5}=\frac{1100-z}{4}.$$

Разобьем траекторию (прямую) на десять промежутков, и для каждой точки найдем координаты в системе $OX_0Y_0Z_0$. Чтобы получить десять точек, будем изменять z = 1100 ..920 с шагом Δz =20 (табл. 4).

При построении траектории нас не интересует ориентация схвата, так как положение схвата манипулятора в каждом положении определяется только координатами TCP в базовой системе координат.

Для использования библиотеки необходимо описать функции с прототипом `double()(const double*)`, которые будут содержать решение обратной задачи кинематики для каждого элемента матрицы манипулятора в символьном виде. Адреса этих функций передаются в массив. Таким образом, получаем матрицу манипулятора в символьном виде.

Также следует описать начальные значения обобщенных координат, пределы их изменения и коэффициент точности решения.

Таблица 4. Траектория движения TCP

Table 4. TCP trajectory

№	*x*	*y*	*x*
0	250	0	1100
1	225	25	1080
2	200	50	1060
3	175	75	1040
4	150	100	1020
5	125	125	1000
6	100	150	980
7	75	175	960
8	50	200	940
9	25	225	920

Матрица манипулятора целевого положения схвата манипулятора будет задаваться для удобства с помощью аргументов командной строки. При этом подматрицу поворота можно задать нулевой, поскольку все положения траектории задаются только координатами TCP в базовой системе координат.

После чего, в теле главной функции описываем алгоритм решения ОЗК $q_0 \mapsto q_2 \mapsto q_1$. Решение будем производить до тех пор, пока максимальное значение абсолютной ошибки по позиционированию больше 0.01 мм. При этом после последовательного итерирования всех сочленений итерация будет продолжаться с начала алгоритма, но с увеличенным на 25% коэффициентом точности решения по сравнению с предыдущей итерацией.

Вывод информации организован следующим образом. По завершению решения ОЗК в стандартный поток выводят значения обобщенных координат, порядковый номер последней итерации и абсолютные максимальную и среднюю ошибки по позиционированию. В файл будем заносить такую же информацию каждый итерационный шаг.

Блок-схема алгоритма решения ОЗК данного робота-манипулятора представлена на рис. 3.

Результаты решения ОЗК для каждого положения траектории представлены в табл. 5 где № – это номер положения, соответствующий такому же номеру в табл. 4; q_0, q_1, q_2 – значения обобщенных координат для каждого положения; q_{err}^{max} – максимальная ошибка по позиционированию; q_{err}^{aver} – средняя ошибка по позиционированию; *iter* – количество итераций, понадобившихся для решения ОЗК; *time* – время, затраченное на полное решение ОЗК.

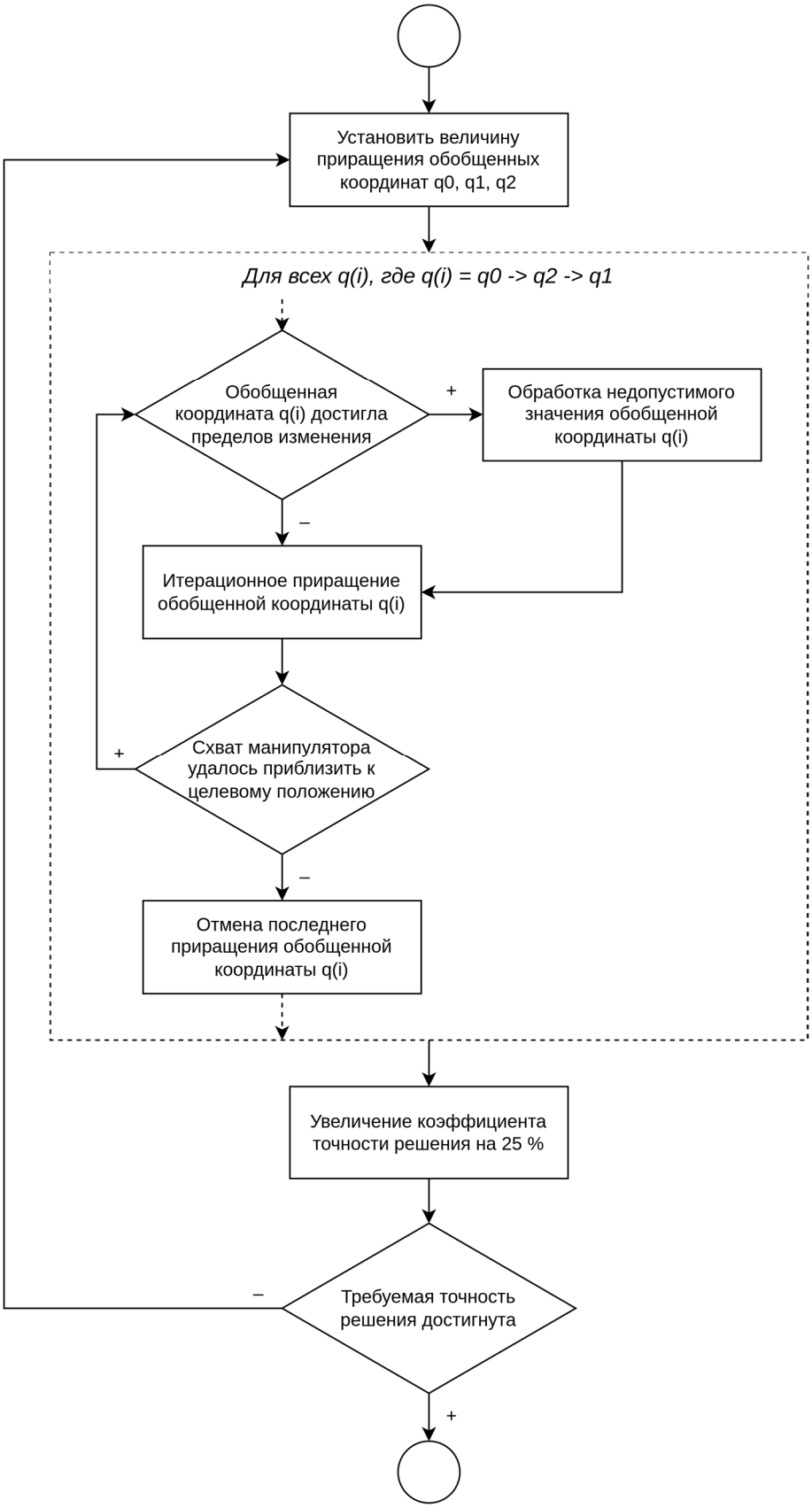

Рис. 3. Блок-схема алгоритма решения ОЗК

Fig. 3. Block diagram of the UGC solution algorithm

Таблица 5. Значение обобщенных координат, точность и скорость решения

Table 5. The value of generalized coordinates, accuracy and speed of the solution

№	q_0, рад / q_0, rad	q_1, мм / q_1, mm	q_2, рад / q_2, rad	q_{err}^{max}, мм/ q_{err}^{max}, mm	q_{err}^{aver}, мм / q_{err}^{aver}, mm	*iter*	*time*, с / *time*, s
0	0.0000	0.0000	1.5708	0.0000	0.0000	30	0.001
1	0.1107	86.0446	2.0089	0.0100	0.0037	8293	0.003
2	0.2450	101.4052	2.1720	0.0090	0.0061	635	0.001
3	0.4050	102.0207	2.2759	0.0100	0.0063	510	0.001
4	0.5881	93.2071	2.3362	0.0080	0.0050	518	0.001
5	0.7854	76.7877	2.3562	0.0091	0.0047	519	0.001
6	0.9827	53.2077	2.3362	0.0082	0.0053	497	0.001
7	1.1660	22.0181	2.2758	0.0094	0.0063	485	0.001
8	1.3258	-18.5746	2.1721	0.0067	0.0052	503	0.001
9	1.4601	-73.9166	2.0090	0.0089	0.0040	560	0.001

Исходя из данных в табл. 5 можно сделать вывод, что все значения обобщенных координат лежат внутри ограничений, приведенных в табл. 3.

Из табл. 5 видно, что полное решение ОЗК трехзвенного робота-манипулятора занимает тысячные доли секунды. Измерение проводилось с помощью утилиты «time», поставляемой в составе многих дистрибутивов ОС GNU/Linux. ЭВМ, на которой производилось вычисление, оснащена ОС Debian 12 GNU\Linux [17], ЦП AMD Ryzen 5 5600H и ОЗУ объемом 15330 МБ. Программа компилировалась посредством компилятора gcc 12.2.0 с использованием реализации стандартной библиотеки glibc 2.36.

Время в значительной степени зависит от алгоритма решения ОЗК и сложности решения прямой задачи кинематики конкретного робота, а также от случайных параметров, например, очереди программы на исполнение центральным процессором [18]. Чтобы снизить влияние случайных факторов на время выполнение программы, из нее были убраны операции вывода информации в стандартный поток и записи в файл [18]. Точность измерения времени обусловлена аппаратными возможностями ЭВМ [18].

Для наглядности работы программы экспортируем для положения №2 значения обобщенных координат, и максимальной и средней ошибки по позиционированию на каждом шаге итерации в csv-файл. Воспользовавшись пакетом электронных таблиц в составе

ПО LibreOffice [13], построим их графики (рис. 4 – 8).

На графиках изменения обобщенных координат q_i во время процесса итерирования (рис. 4 – 6) вертикальными линиями обозначены итерации, на которых ошибка итерирования между текущим итерационным значением и финальным устойчиво достигла 10%, 5%, 1% и 0.5% соответственно.

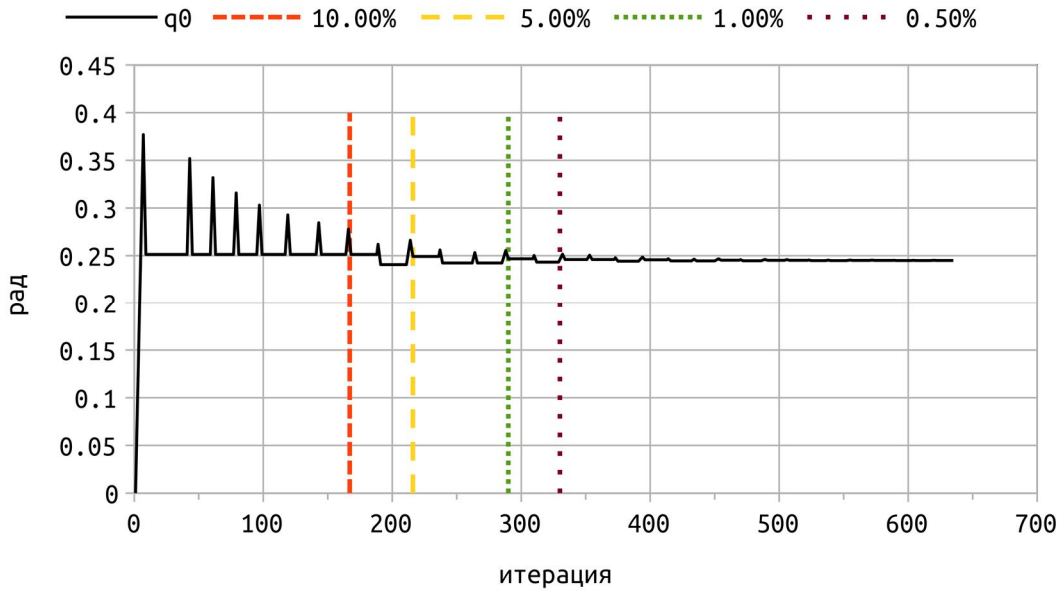

Рис. 4. Процесс итерирования обобщенной координаты q_0

Fig. 4. The process of iterating a generalized coordinate q_0

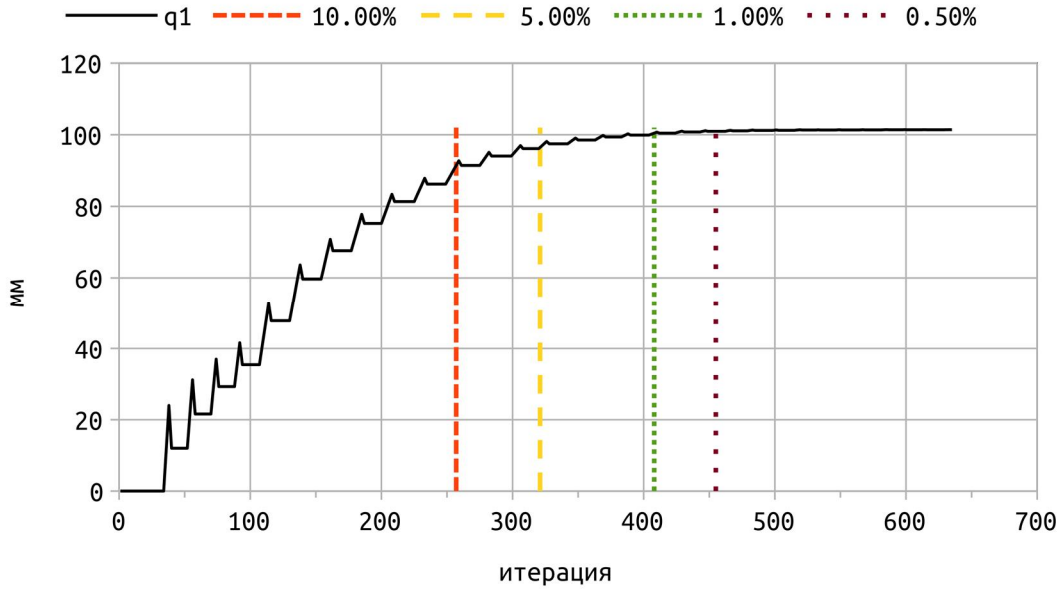

Рис. 5. Процесс итерирования обобщенной координаты q_1

Fig. 5. The process of iterating a generalized coordinate q_1

На рис. 4 представлено изменение значения обобщенной координаты q_0 в процессе приближения к целевому положению №2 из табл. 4. Исходя из графика можно сделать вывод, что обобщенная координата вращательного звена, отвечающего за поворот робота относительно стойки, принимает удовлетворительные значения в пределах 350 итераций для данного положения.

Анализируя график, представленный на рис. 5, можно сделать заключение, что обобщенная координата поступательного звена, отвечающего за достижение требуемой позиции схвата, принимает удовлетворительные значения в пределах 550 итераций для данного положения.

График, представленный на рис. 6, демонстрирует изменение обобщенной координаты вращательного сочленения, отвечающего за достижение требуемой позиции схвата в процессе итерирования. Исходя из графика, можно сделать вывод, что обобщенная координата q_2 принимает удовлетворительные значения в области 350 итераций.

«Пики» на графиках изменения значения обобщенных координат (см. рис. 4 – 6) обусловлены тем, что реализация функций приближения к целевому положению из заголовочного файла rgm3_reverse_kinematic начинает производить итерацию с положительным знаком приращения, и только в случае, если схват приблизить не удалось, меняет знак приращения на противоположный.

На рис. 7. представлено изменение значения максимальной ошибки по позиционированию схвата в процессе итерирования. Из графика видно, что максимальная ошибка принимает значение меньше 1 мм в области 400 итераций.

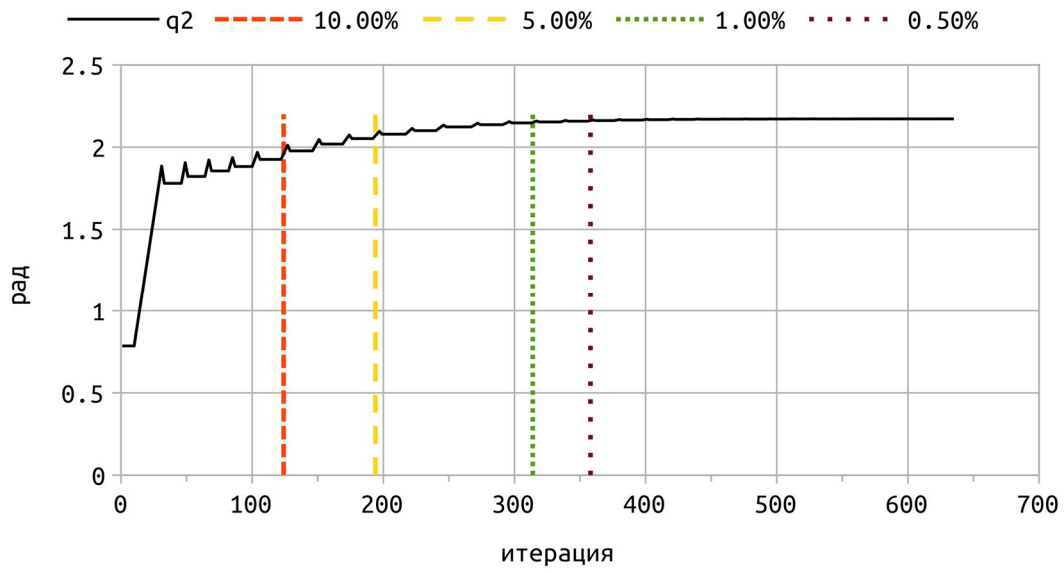

Рис. 6. Процесс итерирования обобщенной координаты q_2

Fig. 6. The process of iterating a generalized coordinate q_2

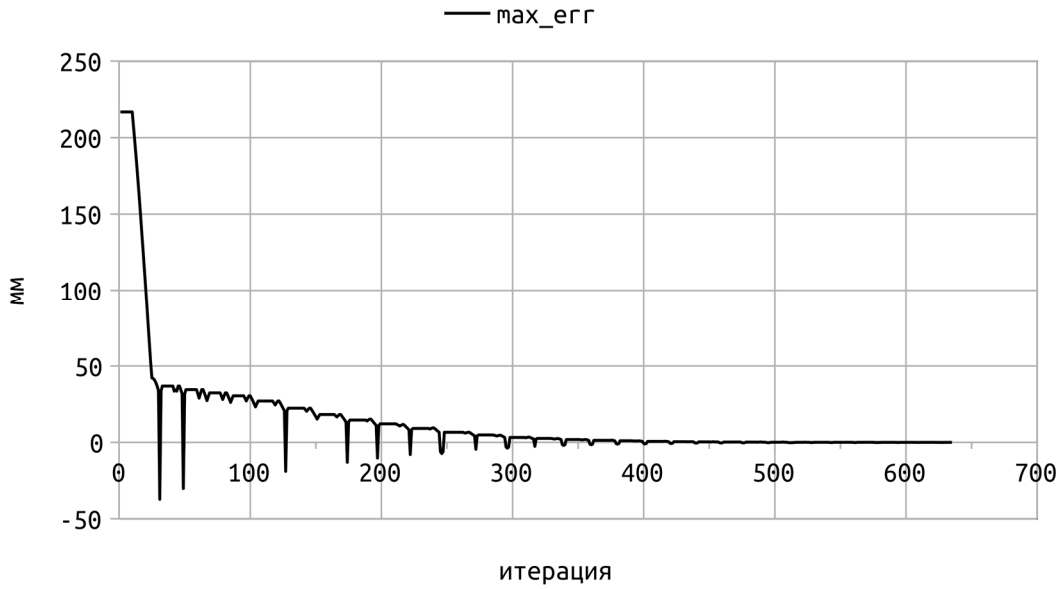

Рис. 7. Изменение значения максимальной ошибки по позиционированию в процессе итерирования

Fig. 7. Changing the value of the maximum positioning error during iteration

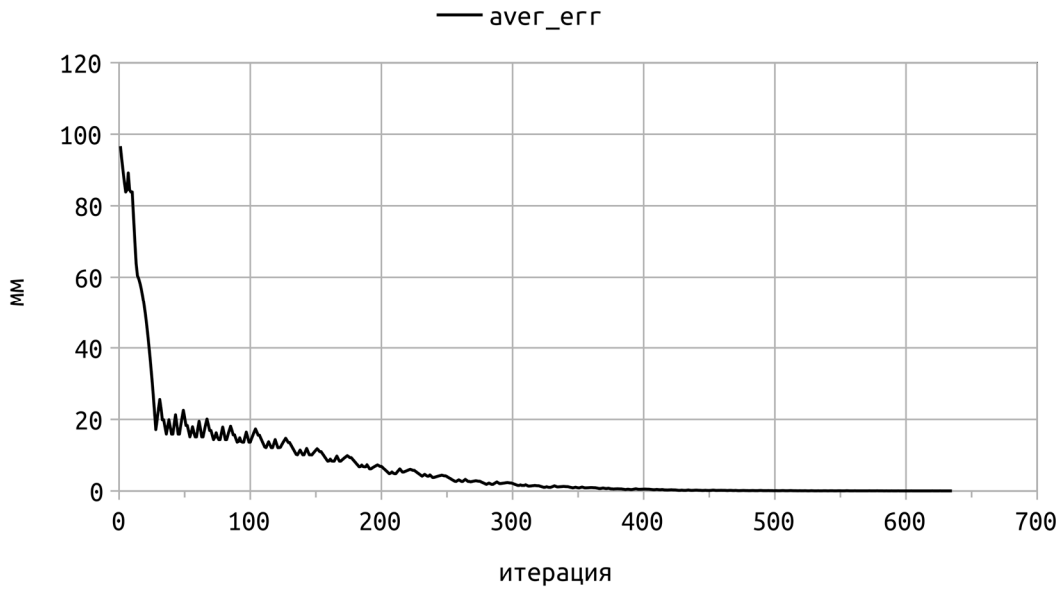

Рис. 8. Изменение значения средней ошибки по позиционированию в процессе итерирования

Fig. 8. Changing the value of the average positioning error in the iteration process

Изменение значения средней ошибки по позиционированию схвата в процессе итерирования представлено на рис. 8. По графику можно сказать, что средняя ошибка принимает значение меньше 1 мм в области 350 итераций.

«Пики» и «волны» на графиках изменения максимальной и средней ошиб-

ки позиционирования (см. рис. 7, 8) обусловлены тем, что приближение производится последовательно $q_0 \mapsto q_2 \mapsto q_1$, следовательно, изменение одной обобщенной координаты требует корректировки другой.

Воспользовавшись САПР OpenSCAD [16], визуализируем модель и траекторию робота (рис. 9). Объединение нескольких изображений в одно произведено с помощью редактора изображений GIMP [17].

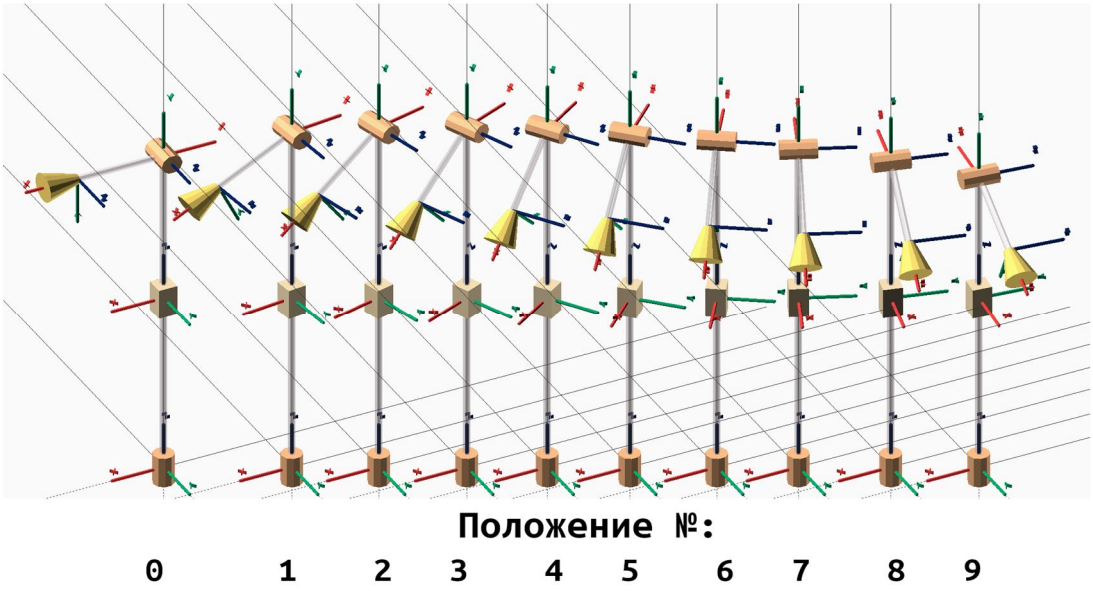

Рис. 9. Визуализация траектории робота

Fig. 9. Visualizing the trajectory of the robot

Выводы

Назовем предложенную в данной статье программную реализацию иерархического подхода к решению ОЗК робота-манипулятора с произвольным количеством сочленений libreRGM3.

libreRGM3 предоставляет функции для определения приращения обобщенной координаты, итерационного приближения к целевым положению и ориентации, избегания пределов изменения обобщенной координаты, диагностики итерационного приближения, а также функции для вывода информации в стандартный поток и записи информации в файл типа csv (comma-separated values).

Работа библиотеки проверена на ЭВМ под управлением операционной системы Debian 12 [17]. Программа, написанная в статье с использованием данной библиотеки, продемонстрировала сходимость за малое количество итераций и быструю скорость решения.

Часть преимуществ и недостатков решения ОЗК посредством написания рабочей программы с использованием библиотеки, представленной в статье, относится к методу решения, который она реализует, другая часть – непосредственно к реализации.

Преимуществами иерархического подхода является однозначность решения, отсутствие необходимости дополнительных проверок полученного решения, отсутствие необходимости дополнительного обучения модели перед решением, общность способа [10].

Недостатками иерархического подхода являются необходимость решения прямой задачи кинематики на каждом шаге итерации, а также то, что алгоритм решения ОЗК и коррекции величин приращения обобщенных координат возлагаются на разработчика [10].

Преимуществами реализации является то, что она не накладывает ограничения на количество сочленений исследуемого манипулятора, число требуемых итераций, а также предлагает ряд функций, которые выходят за рамки иерархического подхода, но могут быть полезны при написании рабочей программы. Например, функции для вывода информации в стандартный поток или записи в файл, функции для обработки ситуаций, когда обобщенная координата принимает недопустимое значение. Помимо этого, реализация иерархического подхода, предложенная в статье, имеет полную документацию, не использует внешних зависимостей и не зависит от конкретного компилятора или реализации стандартной библиотеки, является переносимой между UNIX-подобными ОС, распространяется свободно под лицензией MIT.

Недостатками реализации являются скромность функционала по сравнению с другими проектами [2-8], отсутствие полной поддержки ЭВМ под управлением не-UNIX-подобных ОС (например, семейства Windows) или работающих без ОС, отсутствие собранных бинарных пакетов (распространяется только в виде исходного кода для дальнейшей компиляции).

В данный момент автор работает над устранением существующих недостатков.

Таким образом, использование предложенной в статье библиотеки libreRGM3 может быть эффективным способом решения ОЗК при моделировании движения робота в САПР (например, OpenSCAD [19, 20]), в учебных или исследовательских целях, в условиях работы с ЭВМ без графического интерфейса или с программируемым логическим контроллером и микроконтроллером.

Исходный текст библиотеки доступен для скачивания, использования и изучения по ссылке https://codeberg.org/UlyssesApokin/libreRGM3/releases. Актуальная и подробная документация по функциям библиотеки доступна по ссылке https://codeberg.org/ UlyssesApokin/libreRGM3/wiki.

Список литературы

1. Борисов О. И., Громов В. С., Пыркин А. А. Методы управления робототехническими приложениями. СПб.: Университет ИТМО, 2016. 108 с.

2. Simulate Robot Applications Program any Industrial Robot with One Simulation Environment // RoboDK. URL: https://robodk.com/ (дата обращения: 08.12.2023).

3. Программный комплекс для моделирования роботов и робототехнических систем // Dyn-Soft RobSim. URL: http://www.robsim.dynsoft.ru/ (дата обращения: 08.12.2023).

4. Web based free tool to search, compare and immediately analyze particular robots // KUKA Load. URL: https://www.kuka.com/en-de/products/robot-systems/software/cloud-software/kuka-load (дата обращения: 08.12.2023).

5. Универсальный механизм. URL: http://www.robsim.dynsoft.ru/ (дата обращения: 08.12.2023).

6. Design, simulate, test, and deploy robotics applications // Robotics System Toolbox. URL: https://www.mathworks.com/products/robotics.html (дата обращения: 08.12.2023).

7. Latest Official Release: 0.8.2 // Open RAVE. URL: http://www.openrave.org/ (дата обращения: 08.12.2023).

8. The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications // ROS - Robot Operating System. URL: https://www.ros.org/ (дата обращения: 08.12.2023).

9. Кернинган Б. У., Ритчи Д. М. Язык программирования С. 2-е изд. СПб., 2020. 288 с.

10. Каргинов Л. А. Иерархический подход к решению обратной задачи кинематики // Наука и образование. МГТУ им. Н. Э. Баумана. 2016. № 3. С. 37-63.

11. Фу К., Гонсалес Р., Ли К. Робототехника. М.: Мир, 1989. 624 с.

12. Крахмалев О. Н. Моделирование манипуляционных систем роботов. Саратов: Ай Пи Эр Медиа, 2018. 165 с.

13. LibreOffice is a free and powerful office suite, and a successor to OpenOffice.org (commonly known as OpenOffice). Its clean interface and feature-rich tools help you unleash your creativity and enhance your productivity // LibreOffice. URL: https:// www.libreoffice.org/ (дата обращения: 11.12.2023).

14. Run your private office with the ONLYOFFICE // OnlyOffice. URL: https:// www.onlyoffice.com/ (дата обращения: 11.12.2023).

15. Turn data into insights with free and premium spreadsheets // Microsoft Excel. URL: https://www.microsoft.com/en-us/microsoft-365/excel (дата обращения: 11.12.2023).

16. Maxima. A Computer Algebra System // Maxima. URL: https:// maxima.sourceforge.io/ (дата обращения: 13.12.2023).

17. Debian – The Universal Operating System // Debian. URL: https://www.debian.org/ (дата обращения: 25.12.2023).

18. Столяров А. В. Программирование: введение в профессию. Т.2. Системы и сети. 2-е изд. М.: ДМК Пресс, 2021. 656 с.

19. OpenSCAD:The Programmers Solid 3D CAD Modeller // OpenSCAD. URL: https:// openscad.org/ (дата обращения: 25.12.2023).

20. GIMP – GNU Image Manipulation Program // GIMP. URL: https://www.gimp.org/ (дата обращения: 25.12.2023).

References

1. Borisov O. I., Gromov V. S., Pyrkin A. A. Methods for controlling robotic applications. Saint Petersburg: Universitet ITMO; 2016. 108 p. (In Russ.).

2. Simulate Robot Applications Program any Industrial Robot with One Simulation Environment. *RoboDK.* Available at: https://robodk.com/ (accessed 08.12.2023).

3. Software package for modeling robots and robotic systems. *Dyn-Soft RobSim*. Available at: http://www.robsim.dynsoft.ru/ (accessed 08.12.2023).

4. Web based free tool to search, compare and immediately analyze particular robots. KUKA Load. Available at: https://www.kuka.com/en-de/products/robot-systems/software/cloud-software/kuka-load (accessed 08.12.2023).

5. Universal mechanism. (In Russ.). Available at: http://www.robsim.dynsoft.ru/ (accessed 08.12.2023).

6. Design, simulate, test, and deploy robotics applications. *Robotics System Toolbox*. Available at: https://www.mathworks.com/products/robotics.html (accessed 08.12.2023).

7. Latest Official Release: 0.8.2. *Open RAVE.* Available at: http://www.openrave.org/ (accessed 08.12.2023).

8. The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. *ROS - Robot Operating System*. Available at: https:// www.ros.org/ (accessed 08.12.2023).

9. Kerningan B. W., Ritchie D. M. Programming language C. 2nd ed. Saint Petersburg: Dialektika; 2020. 288 p. (In Russ.).

10. Karginov L. A. Hierarchical approach to solving the inverse problem of kinematics. *Nauka i obrazovanie. MGTU im. N. E. Baumana = Science and Education. MSTU im. N. E. Bauman.* 2016; (3): 37-63. (In Russ.).

11. Fu K., Gonzalez R., Li K. Robotics. Moscow: Mir; 1989. 624 p. (In Russ.).

12. Krakhmalev O. N. Modeling of manipulation systems of robots. Saratov: IP Er Media; 2018. 165 p. (In Russ.).

13. LibreOffice is a free and powerful office suite, and a successor to OpenOffice.org (commonly known as OpenOffice). Its clean interface and feature-rich tools help you unleash your creativity and enhance your productivity. *LibreOffice*. Available at: https:// www.libreoffice.org/ (accessed 11.12.2023).

14. Run your private office with the ONLYOFFICE. *OnlyOffice*. Available at: https://www.onlyoffice.com/ (accessed: 11.12.2023).

15. Turn data into insights with free and premium spreadsheets. *Microsoft Excel*. Available at: https://www.microsoft.com/en-us/microsoft-365/excel (accessed 11.12.2023).

16. Maxima. A Computer Algebra System. *Maxima*. Available at: https:// maxima.sourceforge.io/ (accessed 13.12.2023).

17. Debian – The Universal Operating System. *Debian*. Available at: https:// www.debian.org/ (accessed 25.12.2023).

18. Stolyarov A.V. Programming: an introduction to the profession. Vol.2: Sistemy i seti. 2nd ed. Moscow: DMK Press; 2021. 656 p. (In Russ.).

19. OpenSCAD:The Programmers Solid 3D CAD Modeller. *OpenSCAD*. Available at: https://openscad.org/ (accessed 25.12.2023).

20. GIMP – GNU Image Manipulation Program. *GIMP*. Available at: https:// www.gimp.org/ (accessed 25.12.2023).

Информация об авторах / Information about the Authors

Апокин Елисей Михайлович, студент, Брянский государственный технический университет, г. Брянск, Российская Федерация, e-mail: aoipkn@yandex.ru, ORCID: https://orcid.org/0009-0009-3390-3787

Elisey M. Apokin, Student, Bryansk State Technical University, Bryansk, Russian Federation, e-mail: aoipkn@yandex.ru, ORCID: https://orcid.org/0009-0009-3390-3787

Хандожко Виктор Александрович, кандидат технических наук, доцент, заведующий кафедрой «Автоматизированные технологические системы», Брянский государственный технический университет, г. Брянск, Российская Федерация, e-mail: vichandozhko@gmail.com, ORCID: https://orcid.org/0000-0002-5212-0616,

Viktor A. Khandozhko, Cand. of Sci. (Engineering), Associate Professo,, Head of the Automated Technological Systems Department, Bryansk State Technical University, Bryansk, Russian Federation, e-mail: vichandozhko@gmail.com, ORCID: https://orcid.org/0000-0002-5212-0616