АВТОМАТИЧЕСКОЕ ДИФФЕРЕНЦИРОВАНИЕ В ЗАДАЧЕ УПРАВЛЕНИЯ СИСТЕМОЙ ТВЕРДЫХ ТЕЛ СО СВЯЗЯМИ[1]

Шевляков А. А.[2]

(*ФГБУН Институт проблем управления им. В.А. Трапезникова РАН, Москва*)

В настоящее время в робототехнике идет быстрое развитие новых типов шасси и способов передвижения, для которых требуются оригинальные способы управления, отвечающие многим строгим требованиям. Наряду с человекоподобными и четырехногими роботами, исследуется задача управления группами более просто устроенных роботов. В статье рассматривается задача управления системой из двух роботов-кубов как механической системой твердых тел со связями. Для построения траектории и нахождения управления используется оптимизация. В отличие от многих предыдущих методов, мы исследуем возможность нахождения производных оптимизируемой функции ошибки непосредственно по переменным управления путем использования систем автоматического дифференцирования. Дается краткий обзор методов автоматического дифференцирования, а также обучения с подкреплением, в контексте которого в данный момент развивается эта область. Кратко излагается теория динамики систем твердых тел, а также основные детали решения задачи оптимизации. Рассматриваются несколько сценариев, для которых находится управление и траектория совместного движения роботов. Приводятся параметры проведенного численного эксперимента и характеристики использованного оборудования. Результаты исследования представлены в виде графиков и кинограмм.

Ключевые слова: управление, оптимизация, робототехника, задача комплементарности.

1. Введение

1.1. АВТОМАТИЧЕСКОЕ ДИФФЕРЕНЦИРОВАНИЕ

Под автоматическим дифференцированием обычно понимается набор методов, позволяющих одновременно с процедурой для вычисления значения функции в точке автоматически строить процедуру для вычисления ее производной в этой точке. Этот

[1] *Работа выполнена при финансовой поддержке РФФИ, грант №18-31-00032*

[2] *Андрей Анатольевич Шевляков, к.ф.-м.н.* (*aash29@gmail.com*).

способ опирается на то, что любая компьютерная программа при выполнении вычислений исполняет последовательность элементарных операций, таких как арифметические операции и элементарные функции. Применяя правило производной от сложной функции, мы можем вычислить производную с машинной точностью и за линейное время.

Методы автоматического дифференцирования известны уже несколько десятилетий. Их использование позволяет эффективно вычислять производные функций высокой сложности. Данное свойство особенно важно при оптимизации функций большого числа переменных, поэтому автоматическое дифференцирование широко применяется при обучении глубоких нейросетей – сетей с большим количеством слоев и весов. По сути, именно использование алгоритмов обратного распространения ошибки определило большой успех методов машинного обучения в решении многих задач.

Вообще говоря, вычисление производных от функции ошибки при обучении данным методом может быть реализовано классическими способами – одним из двух известных подходов:

– если функция задана явной формулой, можно вычислить ее производную при помощи обычных формул дифференцирования (с привлечением программ символьных вычислений, если эти формулы становятся слишком громоздкими);

– если функция задана в виде процедуры вычисления ее значений, можно использовать формулы конечных разностей для приближенного нахождения ее производной.

Оба этих метода имеют недостатки. Вычисление производных старших порядков затруднительно и вносит большую ошибку, так же как и вычисление производных по нескольким переменным.

Как правило, автоматическое дифференцирование сопряжено с построением графа вычислений, в котором каждая вершина является операцией с несколькими входами-аргументами и одним выходом-значением (рис. 1).

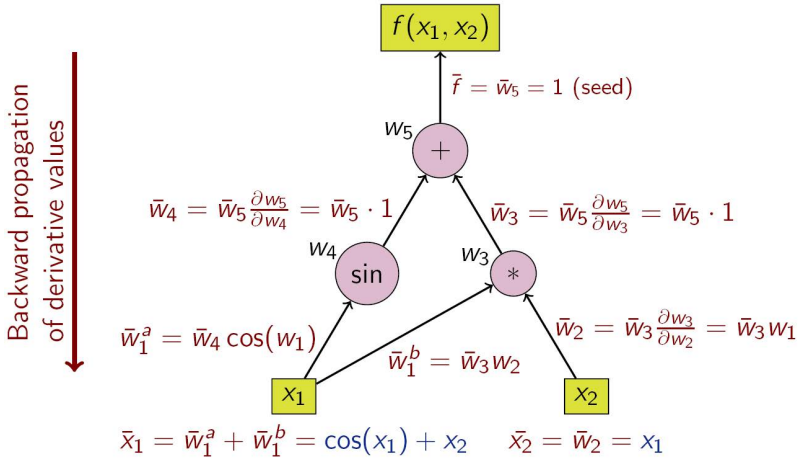

Рис. 1. Пример графа вычислений для функции $z = x_1 x_2 + \sin x_1$

После этого можно провести дифференцирование по правилу сложной функции либо начиная с конечного выражения в корне дерева (обратный ход), либо по ходу выполнения вычисления, начиная с независимых переменных (прямой ход).

На практике прямой ход автоматического дифференцирования может быть реализован с помощью так называемых дуальных чисел. Каждое действительное число x заменяется на $x + x'\varepsilon$, где ε – дуальное число, удовлетворяющее свойству $\varepsilon^2 = 0$. Тогда результаты арифметических операций выглядят следующим образом:

$$
\begin{aligned}
&(x + x'\varepsilon) + (y + y'\varepsilon) &&= x + y + (x' + y')\varepsilon, \\
&(x + x'\varepsilon) \cdot (y + y'\varepsilon) &&= xy + xy'\varepsilon + yx'\varepsilon + x'y'\varepsilon^2 = \\
&= xy + (xy' + yx')\varepsilon.
\end{aligned} \tag{1}
$$

Заметим, что дуальная часть ведет себя как производная сложной функции. Если определить все необходимые функции подобным образом, можно проводить вычисления произвольных выражений, одновременно получая значения их производных в данной точке.

$$
\begin{aligned}
&\langle u,u'\rangle + \langle v,v'\rangle && = \langle u+v, u'+v'\rangle\,,\\
&\langle u,u'\rangle - \langle v,v'\rangle && = \langle u-v, u'-v'\rangle\,,\\
&\langle u,u'\rangle * \langle v,v'\rangle && = \langle uv, u'v+uv'\rangle\,,\\
&\langle u,u'\rangle \,/\, \langle v,v'\rangle && = \left\langle \tfrac{u}{v}, \tfrac{u'v-uv'}{v^2}\right\rangle \quad (v\neq 0),\\
&\sin\langle u,u'\rangle && = \langle \sin(u), u'\cos(u)\rangle\,,\\
&\cos\langle u,u'\rangle && = \langle \cos(u), -u'\sin(u)\rangle\,,\\
&\exp\langle u,u'\rangle && = \langle \exp u, u'\exp u\rangle\,,\\
&\log\langle u,u'\rangle && = \langle \log(u), u'/u\rangle \quad (u>0),\\
&\langle u,u'\rangle^k && = \left\langle u^k, ku^{k-1}u'\right\rangle \quad (u\neq 0),\\
&|\langle u,u'\rangle| && = \langle |u|\,, u'\mathrm{sign}u\rangle \quad (u\neq 0).
\end{aligned}
\tag{2}
$$

Данный подход может быть описан и обоснован в рамках дифференциальной геометрии и теории джетов [5].

Существует большое количество библиотек и фреймворков, реализующих автоматическое дифференцирование. Как правило, в пользовательской программе подключается соответствующая библиотека, в которой определены подходящие типа и операции, и в дальнейшем вычисления проводятся с их использованием.

Наиболее распространенными в последние несколько лет были Tensorflow [1] и PyTorch [14], которые имеют заголовки на популярном языке Python и поддерживают использование видеокарт для ускорения матричных вычислений.

Как уже было отмечено, возможность вычисления производных существенно облегчает решение задач оптимизации. К последним могут быть сведены многие задачи моделирования и управления в механических системах. Например, прямая минимизация отклонения от целевой траектории существенно упрощает постановку задачи. Однако в системах с большим количеством разных режимов (например механических системах с односторонними связями и трением) до недавнего времени не был известен способ эффективно продифференцировать целевую функцию. Подробности автоматического вычисления производных для подобных системы изложены в главе 2.2

1.2. ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ

Многие классические методы обучения с подкреплением, например Q-learning [19], не используют искусственные нейросети. Однако применение глубоких нейросетей, начиная с [13], позволило существенно увеличить размерность рассматриваемых систем. Обучение нейросетей происходит путем оптимизации весов с помощью методов обратного распространения ошибки (error backpropagation). Хотя этот метод возможно реализовать вручную, большое распространение получили так называемые алгоритмы автоматического дифференцирования, которые позволяют автоматизировать вычисление градиента функции ошибки с целью ее минимизации.

В настоящий момент большой интерес вызывает применение различных алгоритмов машинного обучения в задачах робототехники. Основное направление таких исследований – использование обучения с подкреплением (Reinforced Learning, RL). Обучение с подкреплением имеет как сходства с другими разделами машинного обучения (использование глубоких нейросетей), так и различия – как правило, обучение производится с целью максимизации функции награды в некоторой моделирующей среде, а не на обучающей выборке. За последние несколько лет с помощью обучения с подкреплением были достигнуты впечатляющие успехи: победы над чемпионами мира в го, DotA 2 и StarCraft 2.

Как правило, такие подходы рассматривают систему как «черный ящик» , основывая обучение только на изменении награды в конце эпизода. В действительности мы имеем доступ к вычислениям в модели и могли бы использовать их для ускорения обучения или оптимизации. Как было упомянуто в 1.1, при выполнении вычисления значения функции в принципе возможно одновременно вычислять и значения производной.

В [6] исследована возможность написания дифференцируемой модели физической системы с помощью фреймворка PyTorch.

В более ранних статьях (например [15]) высказывалась идея о прямой численной оптимизации как методе решения задачи

управления механической системой. Поскольку использование обычных моделей прямого счета связано с трудностями (чрезмерная чувствительность к значениям управления в начале траектории и недостаточная в конце, большое количество вычислений, рассмотрение системы как «черного ящика»), предлагалась переформулировка задачи моделирования как оптимизации с большим количеством ограничений, описывающих уравнения динамики и неравенства связей. Этот подход, элегантный и гибкий со стороны теории, сталкивается однако с существенными трудностями при практической реализации. Даже для не слишком сложных механических систем требуются мощные оптимизационные алгоритмы, регуляризация исходной задачи и не слишком большое количество шагов по времени.

Мы предлагаем объединить эти подходы и использовать прямое вычисление траектории для ее оптимизации и управления системой, сделав оптимизируемую величину дифференцируемой. Для этого нам потребуется реализация математической модели системы с учетом возможности автоматического дифференцирования.

2. Постановка задачи

Отправной точкой для данного исследования были несколько реально существующих моделей робота-куба. Прототип под названием M-Cube был создан в лаборатории CSAIL MIT (рис. 2). Он демонстрирует возможности движения (в том числе прыжков) и стыковки, однако это движение, по большому счету, не является управляемым [18].

Задача управления была в большой мере решена в Institute for Dynamic Systems and Control, ETH Zurich, что потребовало создания сравнительно большого и сложного механизма под названием Cubli [8] (рис. 3).

Обе эти модели роботом используют находящийся внутри электромотор с маховиком как привод управления.

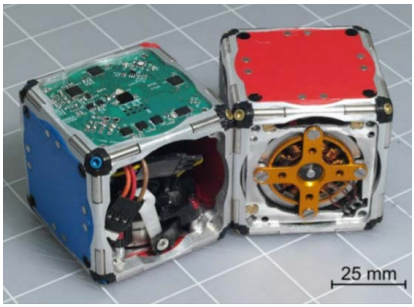

Рис. 2. M-Blocks

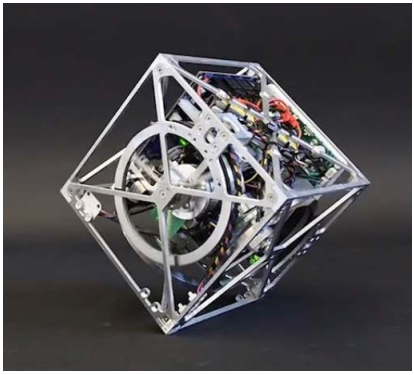

Рис. 3. Cubli

2.1. Математическая модель робота-куба

Будем рассматривать модель куба в виде твердого тела, которое может взаимодействовать с поверхностью пола и с другими твердыми телами с учетом сухого трения и трения покоя.

Ограничимся рассмотрением движения куба в вертикальной плоскости XY, т.е. в действительности в данной статье речь будет идти не о кубах, а о квадратах. Основанием для данного предположения является то, что многие характерные маневры реальными роботами-кубами выполняются в плоскости (рис. 4). В качестве упрощенной модели маховика примем внешний момент u, приложенный к кубу.

Конкретные рассмотренные сценарии движения приведены в разделе 2.4.

Рис. 4. Запрыгивание одного робота на другой

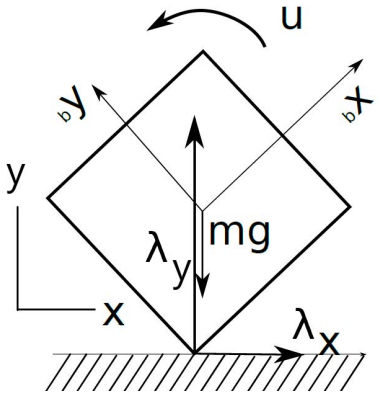

Рис. 5. Схема основных действующих сил

Таким образом, рассматриваемая механическая система будет описываться набором дифференциальных уравнений и неравенств, наложенных на фазовые переменные.

Запишем уравнения динамики движения одного куба:

$$
\begin{aligned}
m\frac{d\bar{V}_0}{dt} &= -m\bar{g} + \sum_{i=1}^{4}(\bar{\lambda}_{n,i} + \bar{\lambda}_{t,i}),\\
I\frac{d\omega}{dt} &= \sum_{i=1}^{4}(\bar{r}_i \times (\bar{\lambda}_{n,i} + \bar{\lambda}_{t,i}))|_z + u.
\end{aligned} \tag{3}
$$

Здесь вектор $\bar{V}_0 = (V_0^x, V_0^y)$ – скорость центра масс куба; $\bar{r}_i = (x_i, y_i)$ – вектор положения i-й вершины куба; $\lambda_{n,i}, \lambda_{t,i}$ – нормальная и тангенциальная составляющая сил реакции в i-й вершине.

При этом реакция связи может иметь значение, отличное от нуля, только при условии контакта, т.е. когда расстояние до поверхности равно 0. То есть из пары неотрицательных значений $y_i, \lambda_{n,i}$ только одно может быть больше 0 в каждый момент времени:

$$
0 \leqslant y_i, 0 \leqslant \lambda_{n,i}, y_i \cdot \lambda_{n,i} = 0. \tag{4}
$$

Такое соотношение обозначают

$$
0 \leqslant y_i \perp \lambda_{n,i} \geqslant 0 \tag{5}
$$

и называют условием комплементарности.

Нормальная и тангенциальная составляющие силы реакции при сухом трении связаны следующим образом:

$$
\begin{aligned}
&\dot{x}_i = 0, -\mu\lambda_{n,i} \leqslant \lambda_{t,i} \leqslant \mu\lambda_{n,i},\\
&\dot{x}_i > 0, \lambda_{t,i} = -\mu\lambda_{n,i},\\
&\dot{x}_i < 0, \lambda_{t,i} = \mu\lambda_{n,i}.
\end{aligned} \tag{6}
$$

Полученная система уравнений и неравенств, описывающая твердое тело с односторонними связями, относится к классу задач комплементарности.

Для описания системы из нескольких роботов-кубов следует записать их уравнения движения совместно и добавить уравнения и неравенства связи описания возможных контактов между ними.

В статье [6] описывается реализация дифференцируемого симулятора динамики твердых тел со связями на языке Python с использованием библиотеки PyTorch (реализация доступна по адресу https://github.com/locuslab/lcp-physics).

Шаг алгоритма выглядит следующим образом:

1. В начале каждого шага t тела находятся в известных положениях p_t со скоростями v_t. Мы предполагаем, что в этот момент все уравнения и неравенства связи выполнены.

2. Все внешние силы, действующие на тела, собираются в один вектор f_t.

3. Формируются матрицы связей

4. Решается задача комплементарности для получения значений скорости на следующем шаге v_{t+dt}.

5. Численное интегрирование используется для получения новых положений тел. Простой метод Эйлера дает следующую формулу: $p_{t+dt} = p_t + dt\Delta v_{t+dt}$. Для новых положений выполняется проверка взаимопроникновения и нарушения связей. Если таковые обнаружены, шаг dt уменьшается вдвое. Процесс продолжается, пока не будет обнаружен достаточно маленький шаг, при котором нарушений не происходит.

6. Если используется пост-стабилизация, то матрицы связей пересчитываются из новых контактов в положении p_{t+dt}, и вычисляется поправка для положений.

2.1.1. Тела

Рассматривается задача в двумерной постановке. Основным объектом моделирования является твердое тело. Твердые тела обладают массой, моментом инерции, положением и скоростью, на них действуют силы. Положение и скорость твердого тела при плоском двумерном движении обладают тремя компонентами: одним угловым и двумя пространственными.

$$p_{body} = \begin{bmatrix} p_a \\ p_x \\ p_y \end{bmatrix} \quad v_{body} = \begin{bmatrix} v_a \\ v_x \\ v_y \end{bmatrix}.$$

Массу и момент инерции тела можно записать в матрицу

$$M_{body} = \begin{bmatrix} I & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix}.$$

Коэффициент упругости k задает эластичность столкновения (при $k = 1$ столкновение полностью упругое, при $k = 0$ – полностью неупругое). Трение задано коэффициентом μ без различия между статическим и динамическим трением. Когда два тела контактируют, сила трения направлена противоположно проскальзыванию одного тела по другому. При этом должно выполнятся соотношение

$$f_{\text{fric}} \leqslant \mu f_{\text{normal}}.$$

И наконец, к каждому телу приложены внешние силы. Их набор может быть заменен равнодействующей, приложенной к центру масс (с компонентами f_x и f_y, и моментом τ).

Для упрощения записи сгруппируем параметры для всех тел в общие структуры. Если в модели есть n тел, пронумерованных от 1 до n, то общие векторы положений и скоростей примут вид

$$p = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix} \quad v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}.$$

Аналогично можно составить вектор из внешних сил, действующих на тела.

$$f = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}.$$

Матрицы масс также могут быть объединены в одну:

$$M = \begin{bmatrix} M_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & M_n \end{bmatrix}.$$

2.1.2. Двусторонние связи

Ограничения типа равенство имеют вид $g(v) = 0$, где g – некоторая функция от скоростей. Такие ограничения могут быть

использованы, к примеру, для моделирования различных шарниров.

В общем случае для двух тел (обозначим их a и b) уравнения связи определяются двумя матрицами Якоби, по одной на каждое тело, такими, что

$$J_e^{(a)} v^{(a)} + J_e^{(b)} v^{(b)} = 0.$$

Чтобы свести это в единую систему с ранее определенными параметрами, введем общую матрицу Якоби J_e. Для n тел и m связей, J_e формируется из блоков, что позволяет записать следующее условие связи:

$$J_e v = \begin{bmatrix} J_{11} & \dots & J_{1n} \\ \vdots & \ddots & \vdots \\ J_{m1} & 0 \dots & J_{mn} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = 0.$$

2.1.3. Односторонние связи

Односторонние связи задаются ограничениями типа неравенство: $g(v) \geqslant 0$. Они обеспечивают отсутствие проникновения твердых тел друг в друга.

$$J_c^{(a)} v_{t+dt}^{(a)} + J_c^{(b)} v_{t+dt}^{(b)} + c_{ab} \geqslant 0,$$

где член c_{ab} зависит от скоростей до контакта и комбинированного коэффициента упругости k_{ab},

$$c_{ab} = k_{ab} \left[J_c^{(a)} v_t^{(a)} + J_c^{(b)} v_t^{(b)} \right].$$

Комбинированный коэффициент упругости, как правило, выбирается в виде простой функции коэффициентов контактирующих тел, например $k_{ab} = 0{,}5(k_a + k_b)$. На каждом шаге по времени неравенства связи составляются для каждой пары тел, находящихся в контакте. Используя нормальный вектор n и точки контакта p_a и p_b, можно записать матрицы Якоби следующим образом

$$J_c^{(a)} = \begin{bmatrix} (p_a \times n) & n^\top \end{bmatrix}, \quad J_c^{(b)} = \begin{bmatrix} (p_b \times n) & n^\top \end{bmatrix}.$$

Как и в случае с равенствами, мы можем объединить все матрицы Якоби в одну:

$$J_c v = \begin{bmatrix} J_{11} & \dots & J_{1n} \\ \vdots & \ddots & \vdots \\ J_{m1} & 0 \dots & J_{mn} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \geqslant -c,$$

$$c = diag(k) J_c v_t, k = [k_1, \dots, k_m]^\top.$$

2.1.4. Трение

Связи, описывающие трение, также задаются неравенствами вида $g(v) \geqslant 0$. В то время как силы контакта действуют по нормали, силы трения направлены по касательной к плоскости контакта двух тел. Не проводя полного вывода, можно интуитивно предположить, что матрица Якоби для трения устроена примерно так же как и для контакта 2.1.3, с касательным вектором вместо нормального. Поскольку касательный вектор в двумерном случае может быть направлен в две стороны, то для каждого контакта будет две связи для трения:

$$J_f^{(a)} v = \begin{bmatrix} (p_a \times d) & d^\top \\ (p_a \times -d) & -d^\top \end{bmatrix} \text{ и } J_f^{(b)} v = \begin{bmatrix} (p_b \times d) & d^\top \\ (p_b \times -d) & -d^\top \end{bmatrix}.$$

2.1.5. Динамическая задача комплементарности

Назовем $\dot{v}$ вектором ускорений. По второму закону Ньютона имеем

$$M\dot{v} = f^{(c)} + f,$$

где $f^{(c)}$ – силы реакции связей, а f – внешние силы. Предполагаем, что других сил в системе нет. Проблема, однако, состоит в том, что данная система может не иметь решений при наличии трения [4].

Если же мы приблизим ускорение конечной разностью

$$\dot{v}_{t+dt} \approx \frac{v_{t+dt} - v_t}{dt},$$

то можем переписать уравнения динамики в виде

$$M(v_{t+dt} - v_t) = dtf_t^{(c)} + dtf_t,$$

для которого доказано существование решения даже в присутствии трения. Поскольку шаг dt мал, можно считать $dtf_t^{(c)}$ приближением импульсов, возникающих при соударении. Как показано в [9], эти импульсы можно записать в виде

$$dtf^{(c)} = J\lambda,$$

где J – это матрица Якоби связи, а λ – вектор некоторых множителей. Объединяя 2.1.5 и 2.1.5, получим

(7) $$Mv_{t+dt} - J_e\lambda_e - J_c\lambda_c - J_f\lambda_f = Mv_{dt} + dtf_t.$$

Вместе с тем, мы знаем, что в реалистичной модели импульсы $J_c\lambda_c$ не дают телам проникать друг в друга, но не могут их притягивать, поэтому $\lambda_c \geqslant 0$. Дополнительно, если для каждой связи $i, (J_cv)_i + c_i = a_i$ для некоторого строго положительного числа a_i, то тела расходятся и разделяющую силу применять не нужно, т.е. $(\lambda_c)_i = 0$. Наоборот, если это условие не удовлетворено, то необходима сила чтобы остановить проникновение, и $(\lambda_c)_i > 0$. Это приводит к следующему условию комплементарности:

$$\lambda_c \geqslant 0, a := (J_cv) + c \geqslant 0, a^\top\lambda_c = 0.$$

Трение добавляет похожие условия, однако вследствие выражений для кулоновского трения в них входит величина сил реакции. Не приводя здесь вывод (см. [7]), укажем эти условия:

$$\zeta := \mu\lambda_c - E^\top\lambda_f \geqslant 0, \sigma := J_fv + \gamma E \geqslant 0, \sigma^\top\lambda_f = 0, \zeta^\top\gamma = 0,$$

при $\lambda_f \geqslant 0$ и $\gamma \geqslant 0$.

Все эти условия можно собрать в единую систему

(8)
$$\begin{bmatrix} 0 \\ 0 \\ a \\ \sigma \\ \zeta \end{bmatrix} - \begin{bmatrix} M & -J_e & -J_c & -J_f & 0 \\ J_e & 0 & 0 & 0 & 0 \\ J_c & 0 & 0 & 0 & 0 \\ J_f & 0 & 0 & 0 & E \\ 0 & 0 & \mu & -E^\top & 0 \end{bmatrix} \begin{bmatrix} v_{t+dt} \\ \lambda_e \\ \lambda_c \\ \lambda_f \\ \gamma \end{bmatrix} = \begin{bmatrix} Mv_t + dtf_t \\ 0 \\ c \\ 0 \\ 0 \end{bmatrix},$$

при условии

$$(9)\qquad \begin{bmatrix} a \\ \sigma \\ \zeta \end{bmatrix} \geqslant 0, \begin{bmatrix} \lambda_c \\ \lambda_f \\ \gamma \end{bmatrix} \geqslant 0, \begin{bmatrix} a \\ \sigma \\ \zeta \end{bmatrix}^\top \begin{bmatrix} \lambda_c \\ \lambda_f \\ \gamma \end{bmatrix} = 0.$$

Здесь неравенства преобразованы в равенства введением ослабляющих переменных $[a, \sigma, \zeta]^\top$, определенных выше, а $[v_{t+dt}, \lambda_e, \lambda_c, \lambda_f, \gamma]^\top$– неизвестные. Неравенства с векторами понимаются покомпонентно. Полученная система относится к классу задач о комплементарности. От обычной системы линейных уравнений или задачи линейного программирования ее отличает наличие последнего условия $[a\,\sigma\,\zeta][\lambda_c\,\lambda_f\,\gamma]^\top = 0$, которое делает задачу нелинейной.

Решив систему (8)–(9), получим скорости на новом шаге: v_{t+dt}.

2.2. Решение полученной задачи комплементарности

Решение близко следует методу, описанному в [11]. Система (8)–(9) имеет вид

$$\begin{aligned} Mx + A^\top y + G^\top z + q &= 0, \\ Ax &= 0, \\ Gx + Fz + s &= m, \\ s \geqslant 0, z \geqslant 0, s^\top z &= 0. \end{aligned}$$

Чтобы решить ее, после инициализации мы итеративно минимизируем невязку по переменным x, s, z и y. На каждой итерации, если критерии остановки не выполнены (размер невязки и разрыв двойственности), вычисляем направления аффинного масштабирования, решая систему

$$(10)\quad \begin{bmatrix} M & 0 & G^\top & A^\top \\ 0 & D(z) & D(s) & 0 \\ G & I & F & 0 \\ A & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta s^{aff} \\ \Delta z^{aff} \\ \Delta y^{aff} \end{bmatrix} = \begin{bmatrix} -(Mx + A^\top y + G^\top z + q) \\ -(D(s)z) \\ -(Gx + Fz + s - m) \\ -(Ax) \end{bmatrix}.$$

Затем вычисляем направление центрирование-плюс-корректор:

$$(11)\quad \begin{bmatrix} M & 0 & G^\top & A^\top \\ 0 & D(z) & D(s) & 0 \\ G & I & F & 0 \\ A & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x^{cc} \\ \Delta s^{cc} \\ \Delta z^{cc} \\ \Delta y^{cc} \end{bmatrix} = \begin{bmatrix} 0 \\ \sigma\mu\mathbb{1} - D(\Delta s^{aff})\Delta z^{aff} \\ 0 \\ 0 \end{bmatrix},$$

$$
\begin{aligned}
x &:= x + \alpha(\Delta x^{aff} + \Delta x^{cc}),\\
s &:= s + \alpha(\Delta s^{aff} + \Delta s^{cc}),\\
z &:= z + \alpha(\Delta z^{aff} + \Delta z^{cc}),\\
y &:= y + \alpha(\Delta y^{aff} + \Delta y^{cc}),
\end{aligned}
\tag{12}
$$

параметры σ, μ, α определены в [11], $\mathbf{1}$ – вектор, все компоненты которого равны 1.

Чтобы определить производные, используем уравнение (2.2) в модифицированном виде, так что в точке решения имеем

$$
\begin{aligned}
&Mx^* + A^\top y^* + G^\top z^* + q = 0,\\
&Ax^* = 0,\\
&D(z^*)(Gx^*Fz^* - m) = 0.
\end{aligned}
\tag{13}
$$

Продифференцируем эти уравнения:

$$
\begin{aligned}
dMx^* + Mdx + dA^\top y^* + A^\top dy + dG^\top z^* + G^\top dz + dq = 0,\\
dAx^* + Adx = 0,\\
D(Gx^* + Fz^* - m)dz + D(z^*)(dGx^* + Gdx + dFz^* + Fdz - dm) = 0,
\end{aligned}
$$

что в матричной форме имеет вид

$$
\begin{bmatrix} M & G^\top & A^\top \\ D(z^*)G & D(Gx^* + Fz^* - m) + F & 0 \\ A & 0 & 0 \end{bmatrix}
\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} =
$$

$$
= \begin{bmatrix} -dMx^* - dA^\top y^* - dG^\top z^* - dq \\ -D(z^*)dGx^* - D(z^*)dFz^* + D(z^*)dm \\ -dAx^* \end{bmatrix}.
$$

В этой формулировке заданная частная производная, например $\frac{\partial z^*}{\partial q}$, может быть найдена, если подставить $dq = I$, все остальные дифференциальные члены приравнять нулю, а потом решить относительно dz.

2.3. Параметры оптимизации

Временной интервал $T = 15$ c разбивался на $n = 200$ шагов. Из испытанных алгоритмов оптимизации наилучшим образом показал себя RMSProp, который является вариантом стохастического градиентного спуска [17] (реализован в pyTorch).

Целью оптимизации является выбор вектора u_i^k, $i = 1, ..., n$, $k = 1, ..., 2$, задающего управление (которое является моментом, приложенным к k-му кубу на шаге i).

В качестве начального приближения выбирался случайный вектор с нормальным распределением величин u_i по закону $1000\mathcal{N}(0, 1)$.

2.4. Численный эксперимент

Оптимизация проводилась на компьютерах следующих конфигураций:

1. Характеристики:

- процессор x64 Intel i-3 4150 с тактовой частотой 3500 МГц;
- объем оперативной памяти 8 Гб;
- графическая карта GeForce 650 Ti.

Пакет PyTorch дает возможность выполнять вычисления как на центральном процессоре (CPU), так и на графической карте (GPU). Счет на GPU, однако, связан с некоторыми ограничениями на исполняемый код. В начале исследования обнаружилось, что вычисления на GPU в данной конфигурации выполняются медленнее, поэтому все дальнейшие эксперименты проводились на CPU.

2. На платформе vast.ai был арендован компьютер со следующими характеристиками:

- процессор x64 Intel Core^{TM} i7-7700K с тактовой частотой 4500 МГц;
- объем оперативной памяти 16 Гб;
- 2 графические карты GeForce GTX 1080 Ti.

Данная конфигурация использовалась для проверки возможности ускорения вычислений при использовании нескольких графических карт. Использование встроенных функций pyTorch не дало заметного прироста производительности, поэтому в дальнейшем эта конфигурация не использовалась.

В качестве функции ошибки была выбрана следующая функция:

$$L(p,v) = 0,1\sum_{i=0}^{n}\sqrt{(x_i - x^*)^2 + (y_i - y^*)^2} + |v_n^x| + |v_n^y|,$$

где v_n^x, v_n^y – компоненты скорости в конечный момент времени, x^*, y^* – желаемое положение центра масс робота.

Размеры моделируемого куба 60х60 мм.

Были рассмотрены 3 сценария:

1) запрыгивание с земли на неподвижную ступеньку высотой 90 мм (рис. 6);

2) запрыгивание одного куба на другой (рис. 7);

3) согласованный шаг двух кубов (рис. 8).

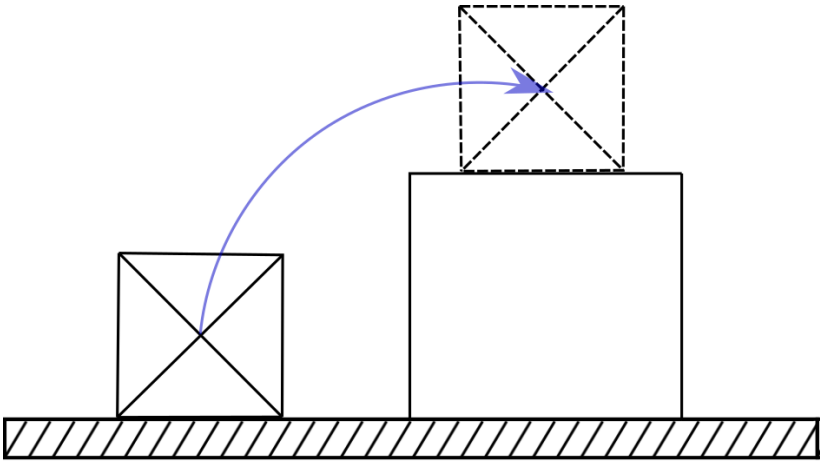

Рис. 6. Запрыгивание с земли на неподвижную ступеньку

2.5. Результаты моделирования

Решение задачи оптимизации занимало от 3 до 20 часов в зависимости от взаимного положения тел, начального приближения искомого вектора и используемой функции ошибки L.

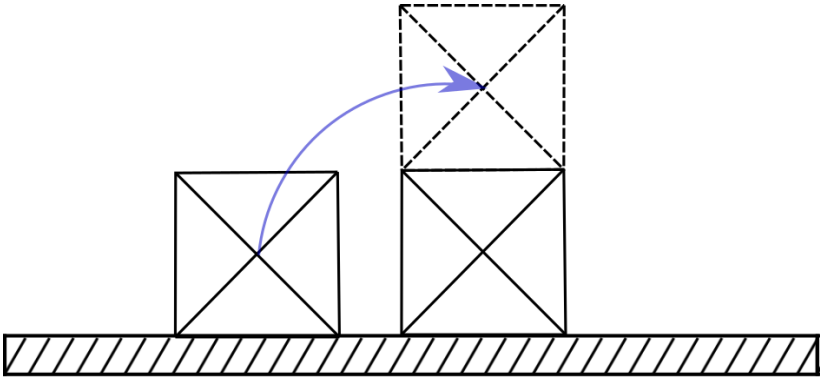

Рис. 7. Запрыгивание с одного куба на другой

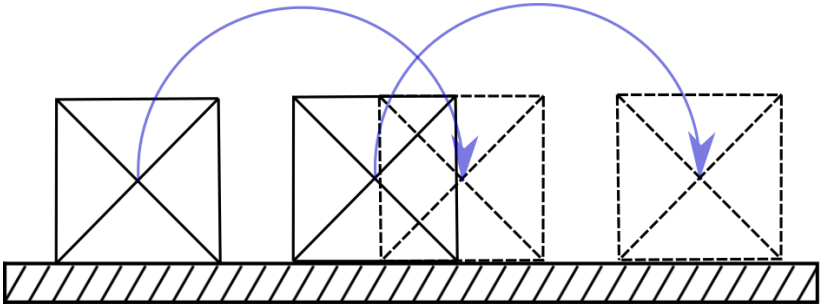

Рис. 8. Согласованный шаг двух кубов

Как и во многих задачах обучения с подкреплением, функция ошибки имеет большое значение, а ее подбор происходит эмпирически.

На многих запусках наблюдалась тенденция «накачивания» энергии в систему и раскрутка кубов до больших скоростей. Это приводило к по сути случайному поведению системы, и для борьбы с данным явлением в функцию L вводился дополнительный член $\sum\limits_{i=0}^{n} \sqrt{(u_i^1)^2 + (u_i^2)^2}$.

Результаты экспериментов можно признать предварительно успешными – рассчитанные траектории приводят систему в положение, близкое к желаемому. Вместе с тем, чувствительность

к весовым коэффициентам функции ошибки оставляет открытым вопрос о том, насколько обобщаем данный метод.

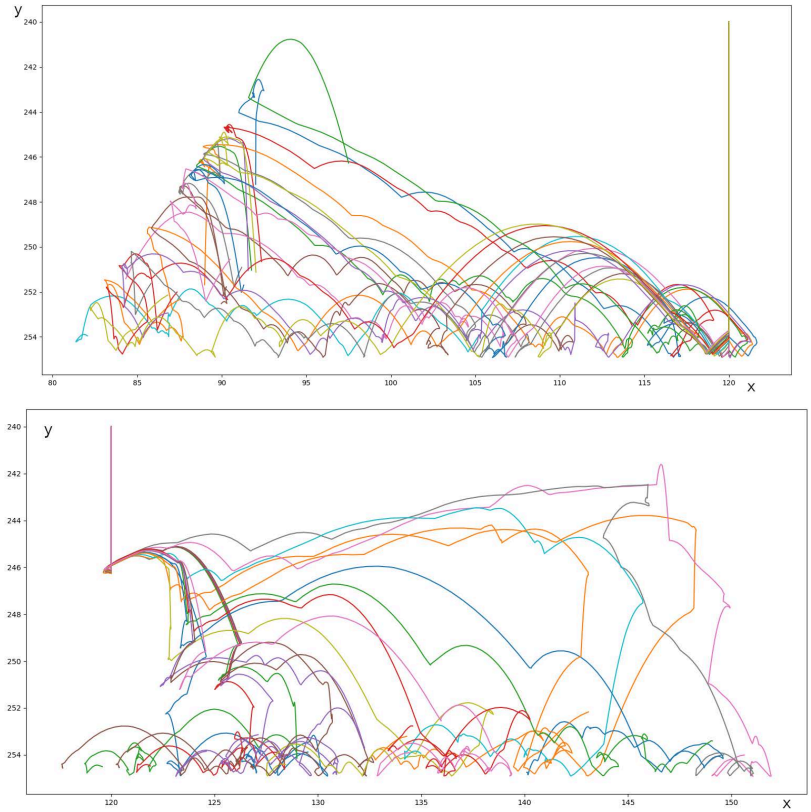

Рис. 9. Траектории центра масс робота-куба в процессе оптимизации для различных задач и начальных условий

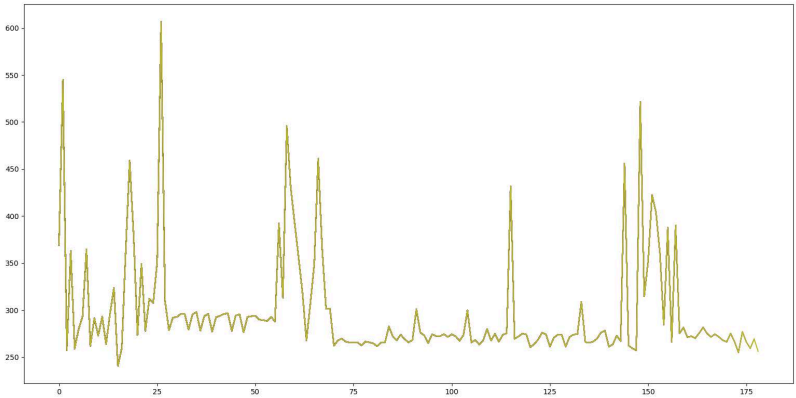

Рис. 10. График функции штрафа в зависимости от номера итерации

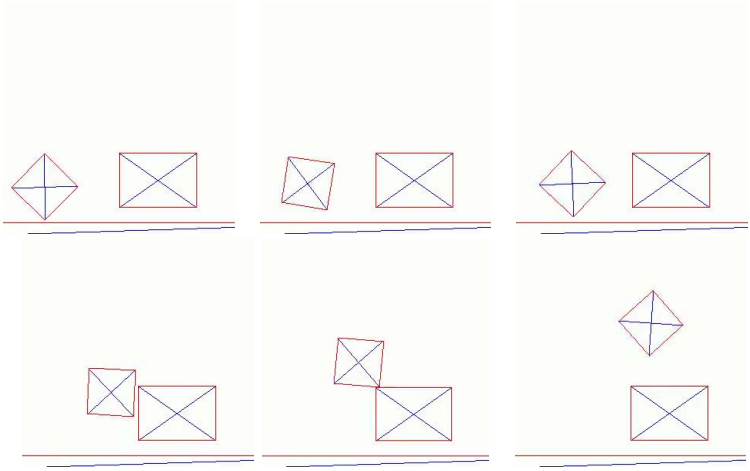

Рис. 11. Кинограмма движения робота-куба при запрыгивании на ступеньку

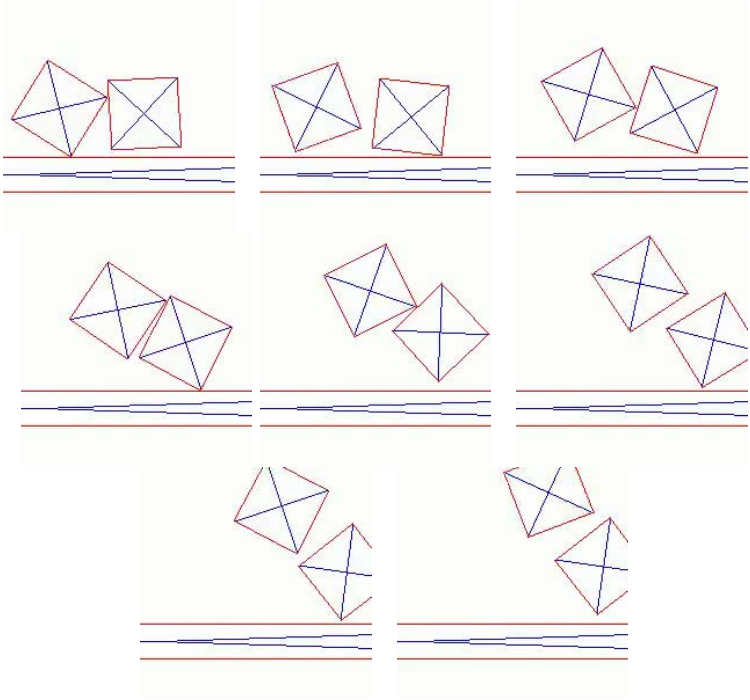

Рис. 12. Киноограмма запрыгивания одного робота-куба на другой

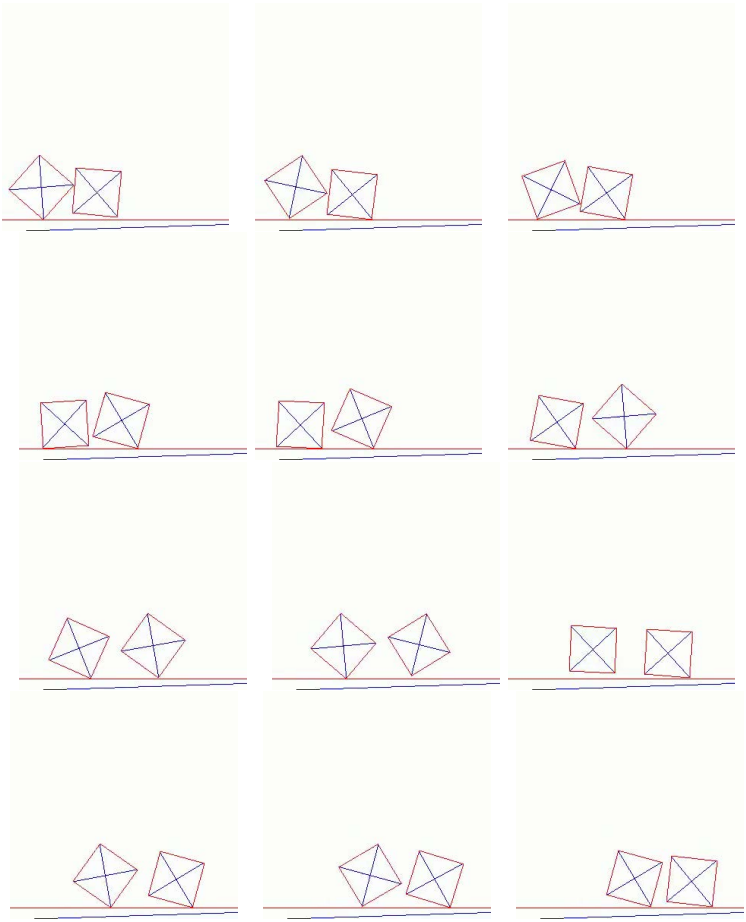

Рис. 13. Кинограмма согласованного движения двух роботов

Литература

1. ABADI M., BARHAM P., CHEN J., CHEN Z., DAVIS A. ET AL. *Tensorflow: A system for large-scale machine learning* // 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). – 2016. – P. 265–283.
2. ANDRYCHOWICZ M., BAKER B., CHOCIEJ M., JOZEFOWICZ R. ET AL. *Learning dexterous in-hand manipulation* // CoRR. – 2018. – URL: http://arxiv.org/abs/1808.00177 (дата обращения: 20.12.19).
3. ANDRYCHOWICZ M., WOLSKI F., RAY A., SCHNEIDER J., FONG R. ET AL. *Hindsight experience replay* // Advances in Neural Information Processing Systems. – 2017. – P. 5048–5058.
4. ANITESCU M., POTRA F.A. *Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems* // Nonlinear Dynamics – 1997. – Vol. 14, No. 3. – P. 231–247.
5. BETANCOURT M. *A Geometric Theory of Higher-Order Automatic Differentiation*. – URL: http://arxiv.org/abs/1812.11592 (дата обращения: 20.12.19).
6. BELBUTE-PERES F. DE AVILA, SMITH K., ALLEN K., TENENBAUM J., KOLTER J.Z. *End-to-end differentiable physics for learning and control* // Advances in Neural Information Processing Systems. – 2018. – No. 31. – P. 7178–7189. – URL: http://papers.nips.cc/paper/7948-end-to-end-differentiable-physics-for-learning-and-control.pdf
7. CLINE M. *Rigid body simulation with contact and constraints*, Ph.D. dissertation. – University of British Columbia, 2002.
8. GAJAMOHAN M., MUEHLEBACH M., WIDMER T., D'ANDREA R. *The Cubli: A reaction wheel based 3D inverted pendulum* // IMU. – 2013. – Vol. 2, No. 2.

9. GARSTENAUER H., KURKA G. *A unified framework for rigid body dynamics*, Ph.D. dissertation. – Jonas Kepler University, Linz, 2006.
10. LILLICRAP T.P., HUNT J.J., PRITZEL A., HEESS N., EREZ T. ET AL. *Continuous control with deep reinforcement learning* // arXiv preprint arXiv:1509.02971, 2015.
11. MATTINGLEY J., BOYD S. *Cvxgen: A code generator for embedded convex optimization* // Optimization and Engineering. – 2012. – Vol. 13, No. 1. – P. 1–27.
12. MAGNUS J. R., NEUDECKER H. *Matrix differential calculus with applications in statistics and econometrics*. – John Wiley & Sons, 2019.
13. MNIH V., KORAY K., SILVER D., GRAVES A., ANTONOGLOU I. ET AL. *Playing Atari with deep reinforcement learning* // NIPS, Workshop on Deep Learning. – 2013.
14. PASZKE A., GROSS S., CHINTALA S., CHANAN G., YANG E. ET AL. *Automatic differentiation in PyTorch* // 31st Conference on Neural Information Processing Systems Long Beach, CA, USA. – 2017.
15. POSA M., CANTU C., TEDRAKE R. *A direct method for trajectory optimization of rigid bodies through contact* // The Int. Journal of Robotics Research. – 2014. – Vol. 33, No. 1. – P. 69–81.
16. POSADA CUERVO D. *Design and control of an inertia wheel cube 2D prototype with a reaction wheel* // B.S. Thesis. – 2017.
17. http://www.cs.toronto.edu/ tijmen/csc321/slides/lecture_slides_lec6.pdf (дата обращения: 20.12.2019).
18. ROMANISHIN J., GILPIN K., RUS D. *M-blocks: Momentum-driven, magnetic modular robots* // Int. Conference on Intelligent Robots and Systems (IROS). – 2013. – P. 4288–4295.
19. SUTTON R.S., BARTO A.G. *Reinforcement learning: An introduction*. – MIT Press, 2018.

AUTOMATIC DIFFERENTIATION IN CONTROL OF CONSTRAINED RIGID-BODY SYSTEMS

Andrey Shevlyakov, V.A. Trapeznikov Institute of Control Sciences of RAS, Moscow, Cand.Sc. (aash29@gmail.com).

Abstract: Currently, many new types of robotic chassis are being developed. These require new approaches to modeling and control, satisfying many hard constraints on performance and quality. The research may be more focused on bipedal and quadrupedal locomotion at the moment, but groups of simple robots are studied as well. In this article, we consider a system of 2 cube robots as a mechanical constrained rigid body system. We use optimization to construct trajectories and find control. Unlike many previous methods, we explore the possibility of differentiating a target function by control variables. This is achieved by using automatic differentiation techniques. We give a brief survey of automatic differentiation, and also of reinforced learning, in the context of which it is now mostly developed. Also, basics of rigid body mechanics are presented, along with most important details of optimization algorithm. We consider several scenarios of robot configurations and goal positions, for which we find control and trajectories. Optimization parameters and hardware characteristics for the numerical experiment are provided. Results are presented as plots and frame sequences.

Keywords: control, optimization, robotics, complementarity problem.

УДК 519.688
ББК 22.193
DOI: 10.25728/ubs.2019.85.5

Статья представлена к публикации членом редакционной коллегии П.С. Щербаковым.

Поступила в редакцию 08.02.2020.
Дата опубликования 31.05.2020.