Нелинейная динамика. 2012. Т. 8. № 4 (Мобильные роботы). С. 689–704.
Полнотекстовая версия в свободном доступе
http://nd.ics.org.ru

УДК: 62-503.56
MSC 2010: 68T40, 93C85

Управление манипулятором с помощью обучения с подкреплением

Н. П. Кошманова, Д. С. Трифонов, В. Е. Павловский

Рассматривается метод построения системы управления роботом-манипулятором с использованием обучения с подкреплением. Система управления будет строиться с помощью обучающегося алгоритма, где информацией для обучения будут совершаемые действия и «награда», — величина, характеризующая качество работы системы управления. Целью обучения является построение алгоритма управления, максимизирующего суммарную награду за некоторый промежуток времени. Алгоритм обучения и построенная в результате его работы система управления протестированы для задачи уклонения манипулятора от летящего в него предмета.

Ключевые слова: обучение с подкреплением, манипулятор, управление

1. Введение

Существует множество алгоритмов управления манипуляторами, базирующихся на теории автоматического управления, для решения задач промышленности, таких как резка листов металла, покраска, перекладывание деталей. В то же время непросто обобщить эти алгоритмы для решения повседневных задач, с которыми сталкивается человек. Во-первых, по причине того, что человеку приходится работать в непредсказуемых условиях. Во-вторых, потому что ему приходится решать множество принципиально разных с точки зрения построения системы управления задач. Алгоритмы машинного обучения могут помочь в решении этих проблем.

Получено 17 июля 2012 года
После доработки 24 августа 2012 года

Кошманова Наталья Павловна
koshmanova.n@gmail.com
Трифонов Дмитрий Сергеевич
slonegg@gmail.com
Павловский Владимир Евгеньевич
vlpavl@mail.ru
Институт прикладной математики им. Келдыша РАН
125047, Россия, г. Москва, Миусская пл., д. 4

В данной статье рассматривается применение алгоритма обучения с покреплением для решения модельной задачи уклонения манипулятора от летящего в него предмета. Стартовое положение манипулятора, стартовое положение объекта и его скорость, топология манипулятора выбирались произвольно. Проведен ряд экспериментов, в ходе которых система управления обучалась избегать столкновения с предметом и восстанавливать исходное положение, прикладывая минимальные усилия в двигателях. При этом для построения системы управления использовалась только информация о состоянии манипулятора, о совершаемых действиях и о событии столкновения. После нескольких повторений системе управления удается избежать столкновения манипулятора с предметом. В результате последующих повторений движения манипулятора становятся плавнее (рис. 1).

Система управления логически состоит из двух модулей: кинематический синтез траектории с помощью обучения с подкреплением и реализация траектории. Для следования по траектории используется классическая система непрерывного управления с линеаризацией с помощью обратной связи [1–3]. Она будет рассмотрена в первой части статьи. Синтез траектории осуществляется разработанным алгоритмом на основе обучения с подкреплением, он будет рассмотрен во второй части статьи.

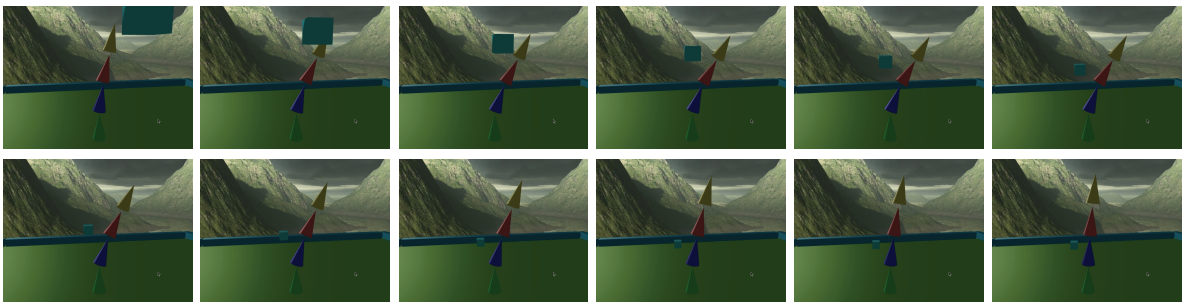


Рис. 1. Кинограмма управления: уклонение манипулятора от летящего предмета.

2. Обзор предметной области

Хорошо исследована сфера построения систем управления для промышленных манипуляторов [1–3]. Но управляемые таким образом манипуляторы предназначены только для использования в заводских условиях для выполнения одной конкретной задачи. Более «гибкие» системы управления могут быть построены с помощью операционного управления [5]. С помощью этого метода можно независимо управлять любыми частями робота (манипулятора), например рабочим органом и мобильной платформой под ним, не задумываясь об их взаимовлиянии. И хотя система управления по-прежнему «жесткая», с помощью этой технологии гораздо проще решить многие задачи, требующие адаптации в реальном времени, такие как податливость [7] или совместное управление [6].

Есть разные методы построения алгоритмов управления манипулятором с помощью машинного обучения. Одним из направлений является использование статистических алгоритмов, основанных на скрытой марковской модели [19, 20]. Принципиально методы из данной группы отличаются способом задания управления и способом обучения.

Для задания управления широко применяются нейронные сети [17], сплайны [18], динамические системы, задающие аттракторы настраиваемой формы для представления движений [11, 16]. Важными достоинствами последнего подхода являются скорость обучения,

простота применения к задачам с отличающимися параметрами (например, координатой аттрактора), устойчивость к небольшим возмущениям.

Для обучения этих моделей могут использоваться алгоритмы регрессии, настраивающие параметры модели согласно данным примерам движений [14], либо обучение с подкреплением [8].

Возможно также совместное применение этих алгоритмов, когда первоначально параметры модели настраиваются с помощью регрессии на основе примеров движений, заданных человеком, а затем они настраиваются с помощью обучения с подкреплением, улучшая характеристики управления [12, 14].

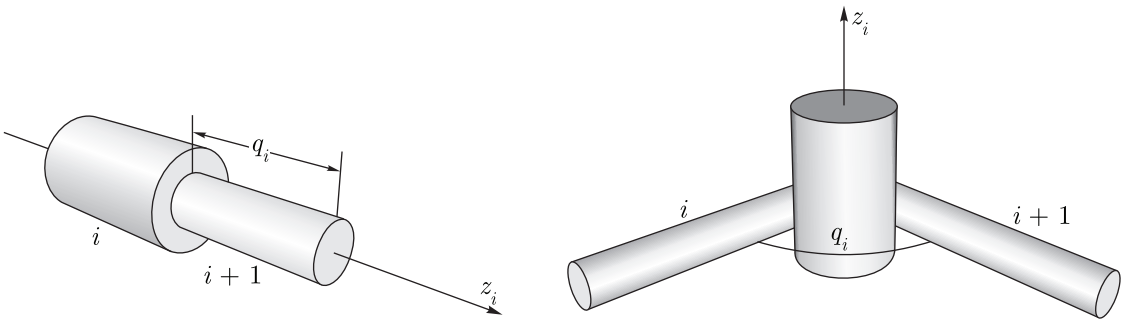
В данной работе рассматривается подход, использующий в качестве модели, описывающей траектории, интерполяцию сплайнами. Параметры модели настраиваются с помощью алгоритма обучения с подкреплением.

3. Кинематика манипулятора

В качестве объекта для системы управления может использоваться произвольный манипулятор, удовлетворяющий следующим ограничениям: используются только вращательные и линейные сочленения, во всех сочленениях находятся моторы. Также предполагается, что устройство манипулятора позволяет решить поставленную задачу.

Будем считать, что всего в манипуляторе n моторов (сочленений) и $n+1$ звеньев, где первое звено закреплено. Звенья могут иметь произвольную, но известную форму, массу и момент инерции. Могут также не иметь массы и размеров. Такие фиктивные звенья будут использоваться для того, чтобы расположить несколько сочленений в одной точке.

Для описания состояния манипулятора зададим вектор обобщенных координат $q = [q_1 \dots q_n]^T$, каждая компонента которого обозначает либо сдвиг в соответствующем сочленении, либо поворот (рис. 2) и вектор обобщенных сил $Q = [Q_1 \dots Q_n]^T$, задающий силу либо момент в соответствующем сочленении.



(a) линейное сочленение (b) вращательное сочленение

Рис. 2. Линейное и вращательное сочленения и соответствующие им обобщенные координаты.

4. Динамика манипулятора

Известны два подхода к выводу уравнений динамики или к получению алгоритма их вычисления [2, 3]: алгоритм Ньютона – Эйлера и уравнения Лагранжа второго рода. Алгоритм Ньютона – Эйлера немного более эффективен с вычислительной точки зрения, уравнения Лагранжа значительно удобнее для формального вывода и построения системы

управления. При построении системы управления будем использовать оба подхода, чтобы сочетать достоинства каждого.

4.1. Алгоритм Ньютона – Эйлера

Алгоритм Ньютона – Эйлера позволяет вычислять скорости и ускорения центров масс звеньев, а также силы и моменты, приложенные к ним. Он состоит из двух этапов: сначала, двигаясь от основания манипулятора, последовательно вычисляются скорости и ускорения звеньев, затем, двигаясь в обратном направлении, вычисляются силы и моменты.

Допустим, задана линейная и угловая скорости центра масс предыдущего звена v_{i-1} и ω_{i-1}. Угловая скорость звена i определяется угловой скоростью звена $i-1$ и скоростью вращательного сочленения. На линейную скорость центра масс звена i влияют угловые скорости звеньев i и $i-1$, линейная скорость звена $i-1$ и оба вида сочленений. Учитывая, что линейная скорость конца вектора r, вращающегося с угловой скоростью ω, равна $v = \omega \times r$ (или $\dot{r} = \omega \times r$), запишем линейную и угловую скорости следующего звена (рис. 3):

$$\begin{aligned} \omega_i &= \omega_{i-1} + \xi_{i-1}\dot{q}_{i-1}z_{i-1}, \\ v_i &= v_{e,i-1} + \omega_i \times r_{c,i} + (1-\xi_{i-1})\dot{q}_{i-1}z_{i-1} = \\ &= (v_{i-1} + \omega_{i-1} \times r_{e,i-1}) + \omega_i \times r_{c,i} + (1-\xi_{i-1})\dot{q}_{i-1}z_{i-1}. \end{aligned} \tag{4.1}$$

В уравнении выше z_i — ось сочленения (рис. 2), а параметр ξ_i отличает вращательное сочленение от линейного:

$$\xi_i = \begin{cases} 1 \text{ — сочленение } i \text{ вращательное}, \\ 0 \text{ — сочленение } i \text{ линейное}. \end{cases} \tag{4.2}$$

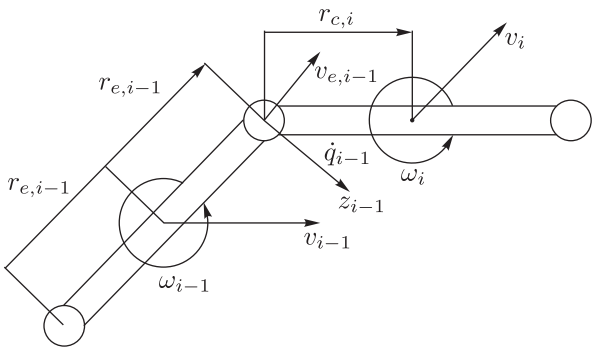


Рис. 3. Последовательный расчет скоростей центров масс звеньев.

Дифференцируя по t формулы выше и учитывая

$$\begin{aligned} \dot{z}_{i-1} &= \omega_{i-1} \times z_{i-1}, \\ \dot{r}_{e,i-1} &= \omega_{i-1} \times r_{e,i-1}, \\ \dot{r}_{c,i} &= \omega_i \times r_{c,i}, \end{aligned}$$

получим выражения для линейного и углового ускорения:

$$\begin{aligned}
\dot{\omega}_i &= \dot{\omega}_{i-1} + \xi_{i-1}(\ddot{q}_i z_{i-1} + \dot{q}_{i-1}\omega_{i-1} \times z_{i-1}),\\
\dot{v}_i &= \dot{v}_{i-1} + \dot{\omega}_{i-1} \times r_{e,i-1} + \omega_{i-1} \times \omega_{i-1} \times r_{e,i-1} +\\
&\quad + \dot{\omega}_i \times r_{c,i} + \omega_i \times \omega_i \times r_{c,i} +\\
&\quad + (1 - \xi_{i-1})(\ddot{q}_{i-1} z_{i-1} + \dot{q}_{i-1} \times \omega_{i-1} \times z_{i-1}).
\end{aligned} \tag{4.3}$$

Теперь, когда все скорости и ускорения рассчитаны, можно записать уравнения для сил и моментов, действующих на звенья. Введем силу f_i и момент τ_i, с которыми звено $i-1$ действует на звено i. Воспользовавшись вторым законом Ньютона и уравнением Эйлера, получим (рис. 4):

$$\begin{aligned}
f_i + m_i g - f_{i+1} &= m_i \dot{v}_i,\\
\tau_i + f_i \times r_{e,i} - \tau_{i+1} + f_{i+1} \times r_{e,i} &= \dot{\omega} I_i + \omega_i \times I_i \omega_i.
\end{aligned} \tag{4.4}$$

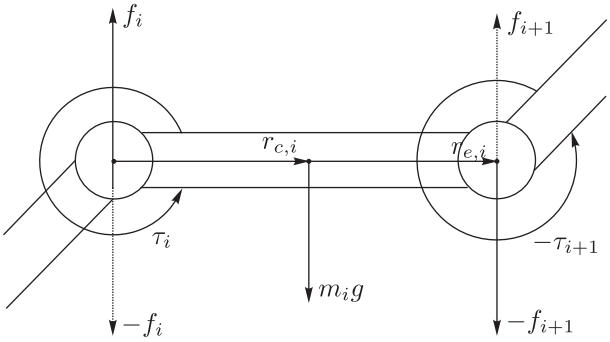


Рис. 4. Силы и моменты, влияющие на звено i.

Уравнение выше верно для тензора инерции I_i, записанного в локальной системе координат звена i, но все скорости и ускорения рассчитывались в глобальной системе координат. Можно перевести все скорости и ускорения в локальную систему координат, но проще будет считать, что I_i — тензор инерции относительно локальной системы координат, переведенный в глобальную систему координат: $I_i = I_i^0 = R_i^0 I_i^i R_i^{0T}$, где R_i^0 — матрица перехода от системы координат звена i в глобальную.

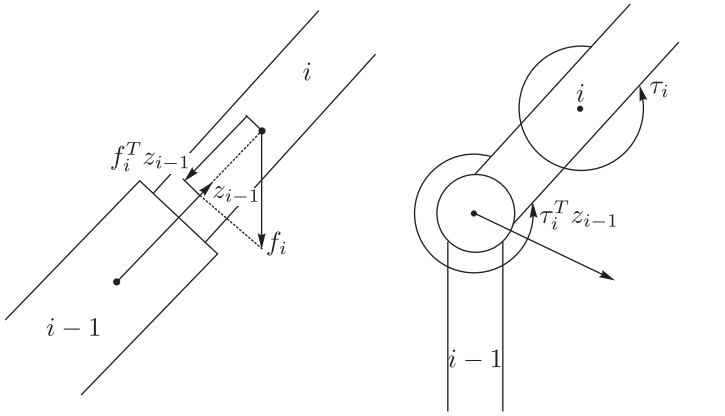
Полагая $f_{n+1} = 0$ и $\tau_{n+1} = 0$, двигаясь от последнего звена к первому, вычисляются все f_i, τ_i по формулам, получающимся из (4.4):

$$\begin{aligned}
f_i &= f_{i+1} + m_i(\dot{v}_i - g),\\
\tau_i &= \tau_{i+1} - f_i \times r_{e,i} - f_{i+1} \times r_{e,i} + \dot{\omega} I_i + \omega_i \times I_i \omega_i.
\end{aligned} \tag{4.5}$$

В таком виде алгоритм можно использовать для симуляции динамики манипулятора: рассчитывая достаточно часто силы и моменты, действующие на центры масс звеньев, проинтегрировав их, можно получить положения и скорости звеньев в определенные моменты времени. С другой стороны, задавая текущий вектор обобщенных координат q, его производную $\dot{q}$ и желаемое ускорение $\ddot{q}_d$, получим силы и моменты, которые нужно приложить к звеньям, чтобы это ускорение создать. Однако управление манипулятором осуществляется при помощи моторов, расположенных в узлах сочленений. Поэтому найдем необходимые

обобщенные силы, которые требуется создать в линейных и вращательных сочленениях, чтобы на звенья действовали вычисленные силы и моменты f_i, τ_i (рис. 5):

$$Q_i = \xi_i \tau_i^T z_{i-1} + (1 - \xi_i) f_i^T z_{i-1}. \tag{4.6}$$



(a) линейное сочленение (b) вращательное сочленение

Рис. 5. Вычисление обобщенной силы.

Нетрудно обобщить алгоритм на случай топологий робота, представляющих собой дерево. Во время прямого прохода, если встретилось разветвление, нужно также разветвить алгоритм (рис. 6), а во время обратного прохода слить ветви алгоритма, складывая все рассчитанные силы и моменты, действующие со стороны разветвленных звеньев на звено, их объединяющее.

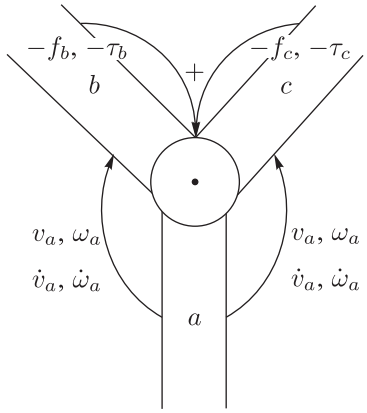


Рис. 6. Разветвление и слияние алгоритма Ньютона – Эйлера.

Таким образом, алгоритм можно записать в виде рекурсивного обхода дерева, описывающего структуру механизма. Во время прямого прохода будут рассчитываться скорости и ускорения звеньев, а во время обратного — силы и моменты. Введем параметры: $\xi_{i,j}$ — тип сочленения между звеньями i и j аналогично (4.2), $q_{i,j}, \dot{q}_{i,j}, \ddot{q}_{i,j}, Q_{i,j}$ — обобщенная координата, ее производная и вторая производная и обобщенная сила, соответствующая сочленению между звеньями i и j. Нулевой индекс будет означать корневое звено. Остальные параметры остаются без изменения. Тогда алгоритм записывается следующим образом:

Algorithm 1 Рекурсивный алгоритм Ньютона – Эйлера NE$(q, \dot{q}, \ddot{q}, g)$

Выход: Q

CalculateForces$(0, 0, 0, 0, 0, 0)$
function CalculateForces$(i, \omega_i, v_i, \dot{\omega}_i, \dot{v}_i)$
  $f_i \leftarrow m_i(\dot{v}_i - g)$
  $\tau_i \leftarrow \dot{\omega} I_i + \omega_i \times I_i \omega_i - f_i \times r_{e,i}$
  for (для) каждого потомка j звена i **do**
    $\omega_j \leftarrow \omega_i + \xi_{i,j} \dot{q}_{i,j} z_{i,j}$
    $\dot{\omega}_j \leftarrow \dot{\omega}_i + \xi_{i,j}(\ddot{q}_{i,j} z_{i,j} + \dot{q}_{i,j} \omega_i \times z_{i,j})$

    $v_j \leftarrow v_i + \omega_i \times r_{e,i} + \omega_j \times r_{c,j} + (1 - \xi_{i,j}) \dot{q}_{i,j} z_{i,j}$
    $\dot{v}_j \leftarrow \dot{v}_i + \dot{\omega}_i \times r_{e,i} + \omega_i \times \omega_i \times r_{e,i}$
      $+\dot{\omega}_j \times r_{c,j} + \omega_j \times \omega_j \times r_{c,j}$
      $+(1 - \xi_{i,j})(\ddot{q}_{i,j} z_{i,j} + \dot{q}_{i,j} \times \omega_i \times z_{i,j})$

    $f_j, \tau_j = \text{CalculateForces}(j, \omega_j, v_j, \dot{\omega}_j, \dot{v}_j)$
    $Q_{i,j} \leftarrow \xi_{i,j} \tau_k^T z_{i,j} + (1 - \xi_{i,j}) f_k^T z_{i,j}$

    $f_i \leftarrow f_i + f_j$
    $\tau_i \leftarrow \tau_i + \tau_k - f_j \times r_{e,i}$
  return f_i, τ_i

С помощью этого алгоритма можно построить систему управления с открытым циклом. Например, если есть траектория $q_d(t)$, ее можно реализовать, полагая $Q(t) = NE(q_d(t), \dot{q}_d(t), \ddot{q}_d(t))$. Но из-за неточностей вычислений, дискретизации, неточности моторов, на практике результирующая траектория будет сильно отличаться от желаемой. Для построения замкнутой системы управления удобно воспользоваться уравнениями динамики в аналитическом виде — например, в форме Лагранжа.

4.2. Численное решение уравнения Лагранжа для манипулятора

Уравнение Лагранжа для описания динамики манипулятора в матричной форме выглядит следующим образом [1–3]:

$$M(q)\ddot{q} + V(q, \dot{q})\dot{q} + G(q) = Q, \tag{4.7}$$

где $M(q)\ddot{q}$ характеризует силы, зависящие от ускорения, а $M(q)$ — инерцию. $V(q, \dot{q})\dot{q}$ характеризует центробежные и кориолисовы силы, а $G(q)$ — гравитационные. Q — обобщенная сила, имеющая такой же смысл, как и в алгоритме Ньютона – Эйлера из предыдущего раздела. Воспользовавшись определением этого алгоритма, можно записать

$$M(q)\ddot{q} + V(q, \dot{q})\dot{q} + G(q) = NE(q, \dot{q}, \ddot{q}, g).$$

Тогда, полагая $M(q) = [M_1 M_2 \dots M_n]^T$, нетрудно убедиться в следующем:

$$\begin{aligned} G(q) &= NE(q, 0, 0, g), \\ V(q, \dot{q})\dot{q} &= NE(q, \dot{q}, 0, 0), \\ M_i &= NE(q, 0, \psi, 0), \quad \psi_j = \begin{cases} 1, i = j, \\ 0, i \neq j. \end{cases} \end{aligned} \tag{4.8}$$

Таким образом, с помощью алгоритма Ньютона – Эйлера вычисляются все компоненты уравнения Лагранжа в любой момент времени (в любом состоянии), а представление динамики в виде уравнения Лагранжа можно использовать для построения замкнутой системы управления.

5. Управление манипулятором

Для выполнения манипулятором заданной траектории необходима система непрерывного управления. Для ее построения используется классический пропорционально-дифференциальный регулятор и линеаризация уравнения динамики обратной связью. Управляющее воздействие определяется следующим законом:

- $q(t)$ — текущее состояние манипулятора,
- $q_d(t)$ — траектория для выполнения,
- K_p, K_v — скалярные коэффициенты регулятора,
- $\widehat{M}(q)$, $\widehat{V}(q,\dot{q})$, $\widehat{G}(q)$ — приближения соответствующих компонент уравнения Лагранжа, описывающих динамику манипулятора, вычисленные по формулам (4.8):

$$Q = \widehat{M}(q)[\ddot{q}_d + K_p(q - q_d) + K_v(\dot{q} - \dot{q}_d)] + \widehat{V}(q,\dot{q})\dot{q} + \widehat{G}(q). \tag{5.1}$$

Если предположить, что уравнение динамики вычисляется точно, то есть $\widehat{M}(q) = M(q)$, $\widehat{V}(q,\dot{q}) = V(q,\dot{q})$, $\widehat{G}(q) = G(q)$, и подставить управляющее воздействие в уравнение (4.7), получается n независимых уравнений затухающих колебаний для ошибки $e(t) = q(t) - q_d(t)$:

$$\ddot{e} + K_v\dot{e} + K_p e = 0. \tag{5.2}$$

Уравнение имеет три класса решений: апериодичность, граница апериодичности, слабое затухание (см. рис. 7). Кроме того, решения устойчивые, что делает систему управления надежной.

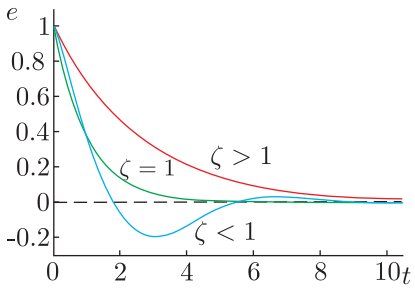


Рис. 7. Решения уравнения (5.2), $K_p = \omega^2, K_v = 2\zeta\omega$.

Для выполнения траектории нами используется решение, соответствующее границе апериодичности, чтобы получить наименьшее время регулирования без перерегулирования (на реальных устройствах может иметь смысл выбирать решение, соответствующее слабому затуханию):

$$\begin{aligned} K_p &= \omega^2, \\ K_v &= 2\omega. \end{aligned} \tag{5.3}$$

Коэффициент ω подбирается нами вручную, чтобы минимизировать время регулирования, но сохранить стабильность симуляции.

6. Обучение с подкреплением

Обучение с подкреплением (*reinforcement learning*) — направление в машинном обучении, в котором осуществляется динамическая настройка параметров стратегии выбора действий, выполняемых агентом.

В каждый момент времени агент находится в некотором состоянии и может совершать действия. Совершая действие, агент переходит в другое состояние и получает награду — вещественное число. Задача агента — максимизировать награду за некоторый длительный промежуток времени.

В стандартной постановке [8] рассматриваются только задачи с конечным числом действий, однако в данной работе рассматривается расширенная версия, использующая непрерывные пространства состояний и действий.

6.1. Формальная постановка задачи

Опишем общую схему алгоритма обучения. Обучение с подкреплением используется для решения задач, поставленных как марковский процесс принятия решений [8]. Он может быть рассмотрен как четверка (S, A, R, T), где

- S — это набор всех состояний, $s_t \in S$ — это состояние агента в момент времени t,
- A — набор всех возможных действий, $a_t \in A$ — это действие агента, совершенное в момент времени t,
- $R\colon S \times A \times S \to \mathbb{R}$ — это функция награды $R(s_t, a_t, s_{t+1})$; эта награда становится известна агенту после достижения состояния $s_t + 1$,
- $T\colon S \times A \times S \to [0, 1]$ — функция перехода, где $T(s, a, s')$ — вероятность перехода в состояние s' из состояния s после выполнения действия a.

При обучении с подкреплением алгоритм осуществляет случайный выбор действий; согласно правилу, $\pi(s, a) = P(a|s)$ — это вероятность выбора алгоритмом действия a в состоянии s. Цель обучения — найти стратегию выбора действий $\pi\colon S \times A \to [0, 1]$, максимизирующую суммарную награду.

Если задача, в которой происходит обучение, состоит из независимых эпизодов, максимизируется суммарная награда за эпизод.

При отсутствии разделения на эпизоды максимизируется взвешенная сумма, где более близкие действия дают больший вклад в награду:

$$R = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1},$$

где r_{t+1} — награда, полученная при переходе из состояния s_t в состояние s_{t+1}, а $0 \leqslant \gamma \leqslant 1$ — величина, обеспечивающая убывание значимости более далекой награды, также она обеспечивает сходимость ряда. Так как R — случайная величина, максимизируется ее матожидание.

6.2. Алгоритм обучения REINFORCE

В работе рассматривается один из подходов в обучении с подкреплением, использующий непосредственную параметризацию $\pi_\theta(s,a)$. Существуют и другие подходы [8], использующие функции качества состояния и действия. Рассматриваемый же подход отличается лучшей масштабируемостью на непрерывные пространства состояний и действий. Ниже рассмотрим один из алгоритмов данного класса — REINFORCE. Функционал, который необходимо максимизировать,

$$J(\theta) = \mathbb{E}\left\{(1-\gamma)\sum_{t=0}^{\infty}\gamma^t r_t \mid \theta\right\} = \int\limits_S d^\pi(s)\int\limits_A \pi_\theta(s,a) r(s,a)\,da\,ds, \tag{6.1}$$

где $d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p(s_t = s)$, описывает распределение состояний при следовании стратегии выбора действий π. Для обучения используется метод градиентного спуска. Шаг обучения при этом может быть представлен как

$$\theta_{m+1} = \theta_m + \alpha_m \nabla_\theta J|_{\theta=\theta_m},$$

где α_m задает величину шага обучения. Разные алгоритмы обучения с подкреплением различаются разными способами оценки функционала $J(\theta)$ и градиента.

По теореме о градиенте (*policy gradient theorem*) (см., например, [12]),

$$\begin{aligned}\nabla_\theta J(\theta) &= \int\limits_S d^\pi(s)\int\limits_A \nabla_\theta\pi_\theta(s,a)(Q^\pi(s,a) - b^\pi(s))\,da\,ds = \\ &= \mathbb{E}\left\{\int_A \nabla_\theta\pi_\theta(s,a)(Q^\pi(s,a) - b^\pi(s))da\right\},\end{aligned} \tag{6.2}$$

где

$$Q^\pi(s,a) = \mathbb{E}_\pi\{R_t|s_t = s, a_t = a\} = \mathbb{E}_\pi\left\{\sum_{i=0}^{\infty}\gamma^i r_{t+i+1} \mid s_t = s, a_t = a\right\},$$

а $b^\pi(s)$ — базовая функция. Выбор этой функции не влияет на смещенность оценки градиента, но ее использование может позволить существенно ускорить процесс обучения. Распределение $d^\pi(s)$ может быть получено при следовании стратегии π_θ.

Если положить $b(s) \equiv 0$, мы можем получить следующее правило обновления параметров, которое гарантированно сходится к локальному максимуму[15]:

$$\theta_{t+1} = \theta_t + \alpha_t \int\limits_A \widehat{Q}^\pi(s_t,a)\nabla_\theta\pi_\theta(s_t,a)\,da,$$

где $\widehat{Q}^\pi(s,a)$ — оценка $Q^\pi(s,a)$, α_t — скорость обучения, убывающая со временем.

Перепишем его следующим образом:

$$\theta_{t+1} = \theta_t + \alpha_t \int\limits_A \widehat{Q}^\pi(s_t,a)\frac{\nabla_\theta\pi_\theta(s_t,a)}{\pi_\theta(s_t,a)}\pi_\theta(s_t,a)\,da = \theta_t + \alpha_t\mathbb{E}\left\{\widehat{Q}^\pi(s_t,a)\frac{\nabla_\theta\pi_\theta(s_t,a)}{\pi_\theta(s_t,a)}\right\}.$$

Это правило может быть упрощено путем рассмотрения эффекта только одного действия, действительно совершенного в состоянии s_t:

$$\theta_{t+1} = \theta_t + \alpha_t\widehat{Q}^\pi(s_t,a_t)\frac{\nabla_\theta\pi_\theta(s_t,a_t)}{\pi_\theta(s_t,a_t)}.$$

Остается вопрос о том, как формировать оценку $\widehat{Q}^\pi(s,a)$; рассматривая разные оценки, можно получать различные алгоритмы. Алгоритм REINFORCE (см. [15]) может быть получен при использовании следующей оценки:

$$\widehat{Q}^\pi(s_t,a_t)=\sum_{k=t}^{T} r(s_k,a_k),$$

равной сумме наград, полученных начиная с состояния s_t и до конца эпизода. В этом случае суммарное изменение θ к концу эпизода будет равно

$$\sum_{t=0}^{T}\frac{\nabla_\theta\pi_\theta(s_t,a_t)}{\pi_\theta(s_t,a_t)}\sum_{k=t}^{T} r(s_k,a_k).$$

Можно переписать это выражение в виде

$$\sum_{t=0}^{T} r(s_t,a_t)\sum_{k=t}^{T}\frac{\nabla_\theta\pi_\theta(s_k,a_k)}{\pi_\theta(s_k,a_k)},$$

что означает, что каждый момент времени t вектор параметров может быть обновлен как

$$z_{t+1}=z_t+\frac{\nabla_\theta\pi_\theta(s_t,a_t)}{\pi_\theta(s_t,a_t)},\quad \theta_{t+1}=\theta_t+\alpha_t r(s_t,a_t)z_{t+1}. \tag{6.3}$$

Вектор весов z обнуляется в конце эпизода. В случае задачи с бесконечным горизонтом правило обновления можно записать в следующем виде:

$$z_{t+1}=\gamma z_t+\frac{\nabla_\theta\pi_\theta(s_t,a_t)}{\pi_\theta(s_t,a_t)},\quad \theta_{t+1}=\theta_t+\alpha_t r(s_t,a_t)z_{t+1}. \tag{6.4}$$

Algorithm 2 Алгоритм обучения REINFORCE:

Вход: Параметризованная стратегия выбора действий $\pi_\theta(s,a)$ с начальными параметрами $\theta=\theta_0$
  for (для) эпизода $e=0,1,2,\ldots$ **do**
    Перейти в начальное состояние s_0, обнулить вектор z;
    $z\leftarrow 0$
    for (для) $t=0,1,2,\ldots,N$ **do**
      Выполнить действие $a\sim\pi_\theta(s,a)$;
      Получить новое состояние s_{t+1} и награду r_{t+1};
      $z_{t+1}\leftarrow\gamma z_t+\dfrac{\nabla_\theta\pi_\theta(s_t,a_t)}{\pi_\theta(s_t,a_t)}$
      $\theta_{t+1}\leftarrow\theta_t+\alpha_t r(s_t,a_t)z_{t+1}$

Скорость сходимости может быть существенно увеличена при использовании подходящей ненулевой базовой функции $b^\pi(s)$ (см. [15]). Это может быть достигнуто путем замены в (6.3) и (6.4) $r(s_t,a_t)$ на $r(s_t,a_t)-\widetilde{b}(s)$. Можно рассмотреть варианты базовой функции: $\overline{r}$ — средняя награда за шаг, до момента t, либо $\widehat{V}(s_t)$ — оценка функции качества состояния, которая может быть получена с использованием алгоритма TD(0) либо TD(λ) [8].

7. Генерация траекторий методом обучения с подкреплением

В этом разделе рассматривается применение описанного в предыдущей главе алгоритма для генерации траектории движения манипулятора. Будем представлять траектории в виде сплайнов третьей степени

$$q_{d,k}(t) = \alpha_{0,k} + \alpha_{1,k}t + \alpha_{2,k}t^2 + \alpha_{3,k}t^3, \quad t \in [t_{k-1}, t_k], \tag{7.1}$$

с граничными условиями, обеспечивающими гладкость:

$$q_{d,k-1}(t_k) = q_{d,k}(t_k), \quad \dot{q}_{d,k-1}(t_k) = \dot{q}_{d,k}(t_k). \tag{7.2}$$

При введении зависимости траекторий от начального состояния параметры сплайна станут функциями от начального состояния $\alpha_{i,k}(s_0)$. Учитывая граничные условия, получим, что на каждый участок сплайна приходится две фукнкции $\alpha^{\theta}_{0,k}(s_0), \alpha^{\theta}_{1,k}(s_0)$, зависящие от параметра θ.

Искомая траектория состоит из K участков сплайна. Состоянием будем считать узлы сплайна, действие будет совершаться в узловых точках сплайна, в качестве действия же возьмем пару $(q, \dot{q})$ — координату и скорость в следующей узловой точке сплайна. Для того чтобы получить стохастическое правило выбора действий, добавим небольшое случайное возмущение к действию: $\widehat{a}_d = a_d + \epsilon$. В качестве случайного возмущения будет использоваться нормально распределенная случайная величина с центром в нуле и дисперсией σ. Таким образом, получается стохастическое правило выбора действий:

$$\pi(s, a) = P(a \mid \widehat{a}_\theta(s)) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\widehat{a}_\theta(s) - a)^2}{2\sigma^2}\right). \tag{7.3}$$

Состоянием в этой задаче является тройка $(q_d, \dot{q}_d, k)$, где k — номер узла, $\widehat{a}_\theta(s)$ — параметризованная функция от состояния. Так как k — дискретная величина, можно представить $\widehat{a}_\theta(s)$ в виде K функций:

$$\widehat{a}^1_{\theta_1}(q_d, \dot{q}_d), \widehat{a}^2_{\theta_2}(q_d, \dot{q}_d) \ldots \widehat{a}^K_{\theta_K}(q_d, \dot{q}_d).$$

Учитывая (7.2) и оставляя зависимость только от s_0, получим, что

$$\widehat{a}^k_{\theta_k}(s_0) = [\alpha^{\theta_k}_{0,k+1}(s_0), \alpha^{\theta_k}_{1,k+1}(s_0)]^T.$$

В простейшем случае, если начальное состояние фиксировано:

$$\widehat{a}^k_{\theta_k} = [\alpha_{0,k+1}, \alpha_{1,k+1}]^T = [\theta^1_k, \theta^2_k]^T.$$

Награда за k-е действие $r_k(s, a)$ учитывает характер траектории на всем k-м сегменте, близость к цели и т. п.

В многомерном случае размерности M рассматриваются M независимых задач. Награда для этих задач может иметь как общие, так и независимые части. При этом для каждой из задач используется одинаковое число сегментов сплайнов с узловыми точками в одни и те же моменты времени.

Представленный подход позволяет генерировать траектории для выполнения различных действий. Для обучения некоторому действию необходимо задать награду, которая

будет его описывать. В качестве тестовой задачи рассмотрим уклонение манипулятора от кубика. Задача алгоритма управления манипулятором: стартуя из вертикального положения, уклониться от кубика и вернуться в исходное положение.

Награду в этой задаче будем задавать следующим образом:

$$
\begin{aligned}
&r_k(s,a) = r_{k,t_{i+1}} = -I_1 c_1 \sum_{\tau=t_k}^{t_{k+1}} \left(|q_\tau - q_0| + |\dot{q}_\tau|\right) - c_2 \sum_{\tau=t_k}^{t_{k+1}} |\ddot{q}_\tau| - c_3 I_2,\\
&I_1 = \begin{cases} 1, \ t_k \geqslant 0.7T, \\ 0, \ t_k < 0.7T, \end{cases}\\
&I_2 = \begin{cases} 1, \ \text{если произошло столкновение}, \\ 0, \ \text{иначе}, \end{cases}\\
&c_1 > 0, \ c_2 > 0, \ c_3 > 0,\\
&t_k \text{ — время начала сегмента.}
\end{aligned}
\tag{7.4}
$$

Здесь первый член награды обеспечивает то, что в конце эпизода траектория достигнет целевой позиции с минимальной скоростью. Второй член минимизирует ускорения на протяжении всей траектории. Третий член обеспечивает большую отрицательную награду в случае, если произошло столкновение. Таким образом, c_3 должно быть значительно больше, чем c_1 и c_2.

Начальное состояние здесь определяется только начальным положением и скоростью кубика, так как остальные параметры фиксированы.

8. Тестирование алгоритма управления

Для симуляции физических взаимодействий использовалась библиотека физической симуляции Bullet. В библиотеке есть множество настроек точности симуляции взаимодействий. Для обучения системы управления приходится симулировать много эпизодов, поэтому использовались настройки точности, обеспечивающие высокую производительность (1 секунда симуляции рассчитывалась примерно за 0.1 секунду на персональном компьютере с процессором Core Quad Q6600).

Основным параметром, влияющим на производительность, является частота расчетов физических взаимодействий и работы системы управления: использовались значения в диапазоне 60–200 Гц. Для управления использовалася упрощенный регулятор без члена $V(q,\dot{q})$, потому что ошибки численного дифференцирования делают регулятор с этим членом неустойчивым:

$$
Q = \widehat{M}(q)[\ddot{q}_d + K_p(q - q_d) + K_v(\dot{q} - \dot{q}_d)] + \widehat{G}(q). \tag{8.1}
$$

Графики ниже (рис. 8) показывают характер изменения обобщенных координат в задаче восстановления равновесия манипулятора из крайнего состояния. Все манипуляторы с 2–5 звеньями стабилизируются за одно и то же время — 2.5 секунды, характер изменения обобщенных координат соответствует выбранному регулятору, но при увеличении количества степеней свободы погрешность становится большой.

Данный регулятор использовался для реализации синтезированных траекторий. Ниже (рис. 9) приведен график зависимости награды от номера эпизода в этой задаче для одного из двигателей манипулятора.

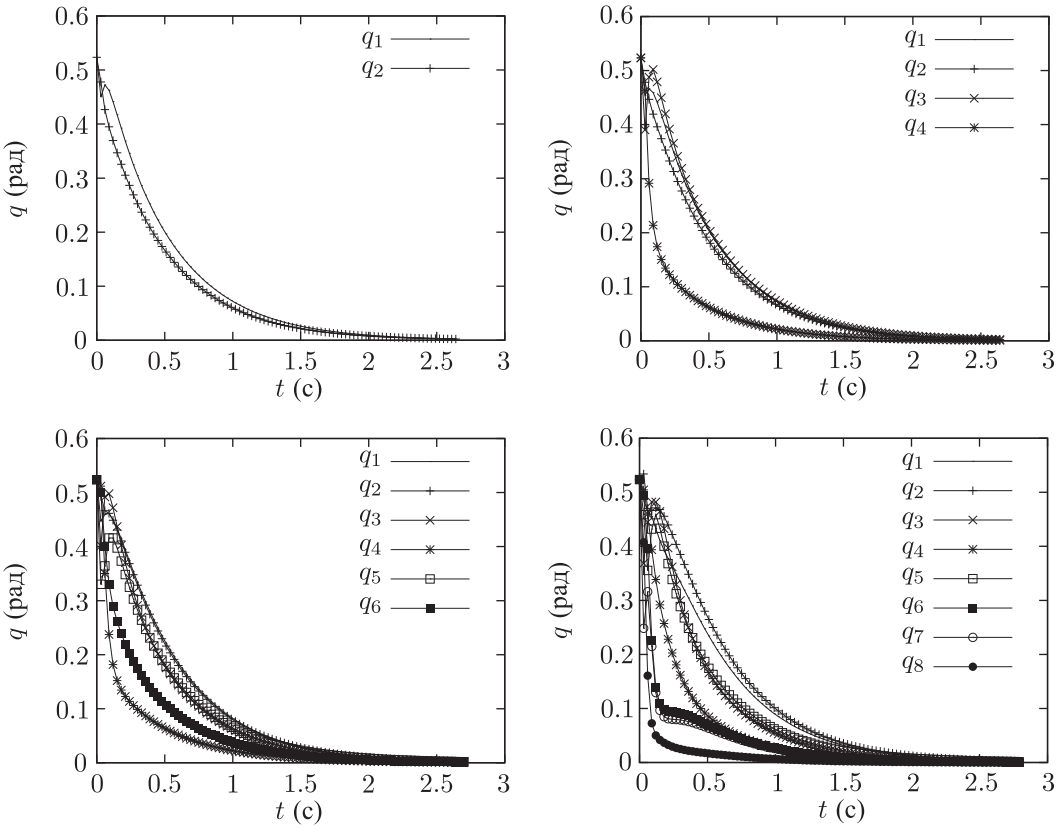


Рис. 8. Стабилизация манипуляторов в точке. Частота обновления 200 Гц.

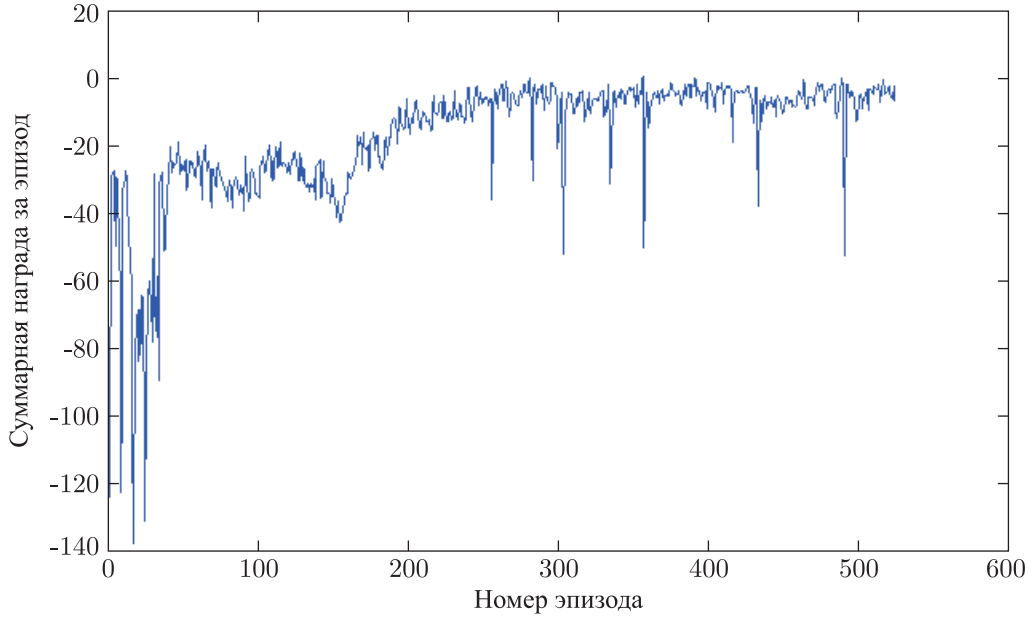


Рис. 9. Награда для одного из двигателей.

В результате первых 50 эпизодов алгоритм обучился избегать столкновения. В течение следующих 150 эпизодов только увеличивалась гладкость траекторий. Награда стабилизируется после 200 эпизодов.

9. Заключение

В работе предложен метод построения системы управления для манипулятора на основе обучения с подкреплением. Разработанный метод пригоден для построения систем управления для решения различных задач, с использованием различных манипуляторов. Этот метод можно использовать при разработке системы управления для решения задач в заранее неизвестном неструктурированном окружении: помощь в работе по дому, работа в экстремальных условиях. Для построения системы управления необходимо только задать кинематические и динамические параметры манипулятора и функционал награды, характеризующий качество управления. Разработанный алгоритм протестирован в задаче уклонения манипулятора от летящего в него предмета, где в качестве подкрепляющей информации выступает событие столкновения и ускорения в двигателях.

Список литературы

[1] Юревич Е. И. Основы робототехники. 2-е изд. СПб.: БХВ-Петербург, 2005. С. 7–207.

[2] Spong M. W., Hutchinson S., Vidyasagar M. Robot modeling and control. New York: Wiley, 2005. P. 1–328.

[3] Craig J. J. Introduction to robotics: Mechanics and control. 3rd ed. Reading, MA: Addison-Wesley, 1986. P. 19–256.

[4] Buss R. Introduction to inverse kinematics with Jacobian transpose, pseudoinverse, and damped least squares methods // IEEE J. Robot. Autom., 2004, vol. 3, pp. 681–685.

[5] Khatib O. A unified approach for motion and force control of robot manipulators: The operational space formulation // IEEE J. Robot. Autom., 1987, vol. 3, pp. 43–53.

[6] Khatib O., Yokoi K., Chang K., Ruspini D., Holmberg R., Casal A., Baader A. Force strategies for cooperative tasks in multiple mobile manipulation systems // Proc. of the Internat. Symposium on Robotics Research (Herrsching, Germany, 1995) / G. Giralt, G. Hirzinger (Eds.). P. 333–342.

[7] Stückler J., Behnke S. Compliant task-space control with back-drivable servo actuators // Proc. of RoboCup International Symposium (Istanbul, Turkey, 2011). P. 78–98.

[8] Sutton R., Barto A. Reinforcement learning: An Introduction. Cambridge, MA: MIT Press, 1998. 322 pp.

[9] Peters J., Schaal S. Natural actor-critic // Neurocomputing, 2008, vol. 71, pp. 1180–1190.

[10] van Hasselt H., Wiering A. Reinforcement learning in continuous action spaces // IEEE Internat. Symp. on Approximate Dynamic Programming and Reinforcement Learning, 2007. P. 272–279.

[11] Peters J., Vijayakumar S., Schaal S. Reinforcement learning for humanoid robotics // 3rd IEEE-RAS Internat. Conf. on Humanoid Robots (Karlsruhe, Germany, 2003). P. 1–20.

[12] Peters J., Schaal S. Reinforcement learning of motor skills with policy gradients // Neural Netw., 2008, vol. 21, pp. 682–697.

[13] Girgin S., Preux P. Basis expansion in natural actor critic methods // Recent Advances in Reinforcement Learning: Proc. of the 8th European Workshop (Villeneuve d'Ascq, France, June 30–July 3, 2008). P. 110–123.

[14] Schaal S. Learning robot control // The handbook of brain theory and neural networks / M. A. Arbib (Ed.). 2nd ed. Cambridge, MA: MIT Press, 2003. P. 983–987.

[15] Vengerov D. A gradient-based reinforcement learning approach to dynamic pricing in partially-observable environments // Future Generation Computer Systems, 2008, vol. 24, no. 7, pp. 687–693.

[16] Schaal S., Peters J., Nakanishi J., Ijspeert A. Learning movement primitives // Robotics Research, 2004, vol. 15, pp. 561–572.

[17] Ito M., Noda K., Hoshino Y., Tani J. Dynamic and interactive generation of object handling behaviours by a small humanoid robot using a dynamic neural network model // Neural Netw., 2006, vol. 9, no. 3, pp. 323–337.

[18] Ude A. Trajectory generation from noisy positions of object features for teaching robot paths // Robotics and Autonomous Systems, 1994, vol. 11, pp. 113–127.

[19] Tso S., Liu K. Hidden Markov model for intelligent extraction of robot trajectory command from demonstrated trajectories // Proc. IEEE Internat. Conf. on Industrial Technology (ICIT), 1996. P. 294–298.

[20] Yang J., Xu Y., Chen C. Human action learning via hidden Markov model // IEEE Trans. Syst. Man Cybernet., 1997, vol. 27, no. 1, pp. 34–44.

Reinforcement learning for manipulator control

Nataly P. Koshmanova[1], Dmitry S. Trifonov[2], Vladimir E. Pavlovsky[3]

[1,2,3]Keldysh Institute of Applied Mathematics RAS
Miusskaya st. 4, Moscow, 125047, Russia
[1]koshmanova.n@gmail.com, [2]slonegg@gmail.com, [3]vlpavl@mail.ru

We present method for constructing manipulator control system with reinforcement learning algorithm. We construct learning algorithm which uses information about performed actions and their quality with respect to desired behaviour called «reward». The goal of the learning algorithm is to construct control system maximizing total reward. Learning algorithm and constructed control system were tested on the manipulator collision avoidance problem.

MSC 2010: 68T40, 93C85
Keywords: reinforcement learning, manipulator, control, Newton – Euler algorithm

Received July 17, 2012, accepted August 24, 2012
Citation: *Rus. J. Nonlin. Dyn.*, 2012, vol. 8, no. 4, pp. 689–704 (Russian)