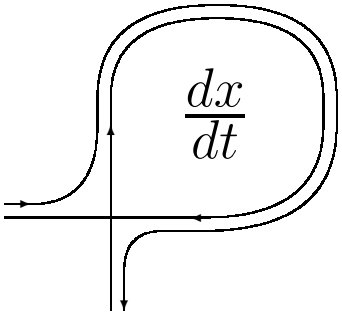

ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ
И
ПРОЦЕССЫ УПРАВЛЕНИЯ
N. 1, 2019
Электронный журнал,
рег. Эл. N ФС77-39410 от 15.04.2010
ISSN 1817-2172

http://diffjournal.spbu.ru/
e-mail: jodiff@mail.ru

Оптимальное управление

Динамическая оптимизация и кусочно постоянные функции на множестве состояний

Орёл Е.Н., Орёл О.Е.
Московский физико-технический институт, г. Долгопрудный
e-mail: enorel@fa.ru

Аннотация

Рассмотрен прямой метод динамической оптимизации для построения приближений к глобальному экстремуму. Метод основан на разбиении множества состояний на классы (ячейки) и построении кусочно-постоянных функций на заданном разбиении. Такой подход приводит к обобщению метода ломаных Эйлера и использованию алгоритмов поиска кратчайшего пути на графе.

В предложенном алгоритме для каждого класса формируется маршрут, ведущий из начальной точки в этот класс. При этом запоминаются только конечная точка маршрута, функционал на маршруте и номер предыдущего класса. Если находится другой маршрут с теми же граничными условиями, но с меньшим (для задачи на минимум) значением функционала, то он становится текущим приближением.

Доказывается, что если разбиение достаточно мелкое, то метод находит оптимальную ломаную. Предложенный подход может быть применен к задачам динамической оптимизации с неполной информацией (дифференциальные игры, управление при неизвестной динамике). Приводятся результаты численного решения некоторых задач оптимального управления и дифференциальных игр.

Ключевые слова: глобальный экстремум, оптимальное управление, ломаные Эйлера, дифференциальные игры, неизвестная динамика, разбиение на классы, кусочно-постоянные функции.

Abstract

We consider a direct method of dynamic optimization for an approximation of global extremum. It is based on splitting the state space into classes (cells) and the construction of piecewise constant functions on the partition. Such an approach leads to a generalization of the Euler polygonal method and using shortest path algorithms on graphs.

In the proposed algorithm for each class (cell) a path from an initial point to this class is formed. In addition, the program remembers only the terminal point of the path, functional value along the path and the number of the previous class. If another path with the same boundary conditions but lesser functional value (for the minimization problem) is found, this path becomes the current approximation.

It is proved that if the partition is sufficiently small, then we obtain the optimal polygon. The suggested approach is applied to problems of dynamic optimization with incomplete information (differential games, control of systems with unknown dynamics). In addition, some results of numerical solutions of optimal control problems and differential games are given.

Keywords: global extremum, optimal control, Euler's polygonal method, differential games, systems with unknown dynamics, splitting into classes, piecewise constant functions.

1 Введение

В работах [1]–[3] для решения задач оптимального управления использовался подход, основанный на построении кусочно постоянных функций, которые представляют собой приближение функции Беллмана $B(x)$ или дуальной к ней функции $A(x)$ (т.е. функции Беллмана при условии обращения времени). В настоящей статье показывается, что подход является универсальным и может быть использован в самых разных задачах динамической оптимизации при наличии неопределённостей разного рода. В достаточно общем виде задачи оптимального управления рассматривались в [4]. Начиная решение конкретной задачи, следует разбить множество состояний на конечное число подмножеств, которые можно считать *классами*, или *ячейками*. Пространство функций, постоянных на классах, конечномерно. Благодаря этому значения

таких функций могут формироваться и храниться в памяти компьютера.

Разбивая пространство на классы, мы не ограничиваем блок принятия решений необходимостью попасть в точки заранее заданной решётки. Данный подход не связан напрямую с необходимыми и/или достаточными условиями экстремума и относится к категории прямых методов. Процесс поиска и формирования кусочно постоянных функций в ряде случаев напоминает имитационное моделирование, при котором многократно проигрывается сценарий будущего поведения системы. В ходе поиска поведение системы улучшается, и в итоге строится траектория, которая может считаться приближением глобального экстремума.

Чтобы показать связь между глобальным экстремумом и его приближением, мы рассматриваем хорошо известные задачи, для которых ранее в литературе (см., напр., [15]) было проведено довольно подробное исследование всевозможных характерных поверхностей переключения, универсальных и экивокальных поверхностей, барьеров и т.д., но не всегда было дано полное решение задачи. В дальнейшем можно будет сравнить результаты работы аналитических и численных методов и установить примерные границы их применения.

Предлагаемый подход позволяет избежать построения "элементарной операции" Н.Н. Моисеева [5] благодаря движению не по заранее намеченным точкам (например, точкам решётки), а по ячейкам.

Истоки метода кусочно постоянных функций обнаруживаются уже в методе ломаных Эйлера, с которого мы начинаем изложение материала. Метод ломаных Эйлера играет важную роль не только в прикладной, но и в чистой математике [6]: он используется при решении дифференциальных уравнений, а также при выводе уравнения Эйлера-Лагранжа и доказательстве теоремы существования в вариационном исчислении.

В данной работе рассматривается применение метода ломаных Эйлера для численного решения задачи вариационного исчисления. Иллюстрация результатов работы алгоритма показывает, что алгоритм строит не одну ломаную, а их семейство, образующее дискретное центральное поле кривых. Каждая ломаная состоит из отрезков, на которых, очевидно, наклон постоянен.

В задачах оптимального управления мы под ломаной будем понимать траекторию с кусочно постоянным управлением. Также рассмотриваются дифференциальные игры и задачи на выживание при неизвестной динамике. Логика

формирования кусочно постоянных функций весьма разнообразна и зависит от конкретной задачи: точки старта на этапе самообучения могут выбираться систематически или по случайному закону, построение можно начинать с миноранты [7] или с мажоранты функции Беллмана, поиск можно осуществлять в ширину или в глубину с возвращениями [8]–[9], и наконец, в задачах с неполной информацией о фазовых ограничениях можно для ускорения процесса использовать эвристическую функцию.

Алгоритмы и программы приводятся на псевдокоде – неполностью формализованном языке PDL (program design language), в котором используются элементы языка программирования Pascal.

2 Поле оптимальных ломаных Эйлера

Рассмотрим задачу вариационного исчисления

$$J(x(.)) = \int_{t_0}^{t_1} L(t, x(t), \dot{x}(t)dt \to \min, \quad t_0 < t_1, \quad x(t_0) = x_0, \quad x(t_1) = x_1$$

(для простоты, будет рассматриваться одномерный случай, но всё сказанное легко обобщается на случай $x \in \mathbb{R}^n$). При построении ломаных отрезок времени $[t_0, t_1]$ разбивается на m равных частей, и на оси Ox берутся n точек с постоянным шагом. Возникает регулярная решётка (τ_i, x_j) на плоскости. Затем точки решётки, лежащие на соседних вертикалях, соединяются отрезками, и решается задача поиска пути наименьшей стоимости на полученном графе, содержащем примерно mn вершин. Заполнение меток при вершинах идёт слева направо, поэтому на решение задачи уходит mn^2 макроопераций. Попутно строится функция $A(\tau_i, x_j)$, определённая на точках решётки и равная минимуму стоимости ломаной, ведущей из точки старта в текущую точку решётки. Фактически алгоритм строит не одну траекторию, а дискретное *центральное поле оптимальных ломаных Эйлера*, начинающихся в (t_0, x_0) и оканчивающихся во всех остальных точках решётки, т.е. оптимальное движение имитируется многократно. Последнее обстоятельство очень важно, так как, прежде чем построить одну оптимальную (в глобальном смысле) траекторию, мы должны обойти все точки, чтобы убедиться, что нет таких мест, где стоимость шагов очень низка или даже отрицательна. Переходя к ломаным Эйлера, мы значительно упрощаем задачу и сужаем область поиска. Но, как известно, такими ломаными можно приблизить любую достаточно гладкую кривую. Поэтому если мы хотим точнее решить задачу, нам придётся перейти к более частой решётке.

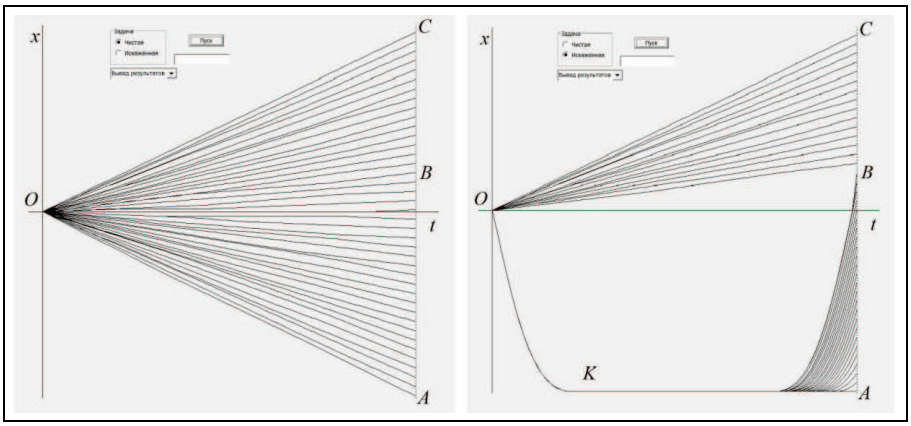

Рис. 1: Оптимальные ломаные на обычной и искажённой евклидовой плоскости

Метод ломаных Эйлера был проверен авторами статьи на известных задачах вариационного исчисления: кратчайшие кривые на плоскостях Евклида и Лобачевского, брахистохроны, минимальные поверхности вращения [10]–[11]. Во всех случаях визуально было трудно отличить поля кривых, построенных аналитически и численно.

На рисунке 1 изображены результаты работы алгоритма для двух родственных задач – кратчайшие кривые на обычной и "искажённой" евклидовой плоскости. Программой построены оптимальные ломаные, соединяющие точку O (начало координат) с точками на прямой AC. На "искажённой" плоскости функция Лагранжа задаётся формулой

$$L = \begin{cases} \sqrt{1+\dot{x}^2}, & x \geq 0, \\ \sqrt{1+\dot{x}^2}\left(1-\left(\frac{x}{a}\right)^2\right), & -a \leq x < 0, \end{cases}$$

где $a > 0$ – константа. Заметим, что на полуплоскости $x \geq 0$ функция Лагранжа для обеих задач одна и та же. Значит, отрезки OM, ведущие в точки M, лежащие над осью Ot, являются экстремалями и для искажённой плоскости. Но если M лежит ниже точки B, то оптимальной будет не стандартная экстремаль, представляющая собой отрезок OM, а ломаная, опускающаяся вниз, в район точки K, идущая далее горизонтально, а потом поднимающаяся вверх.

Возвращаясь к общему случаю, заметим, что, выбирая точки x_j на оси Ox, мы разбиваем ось на $n+1$ множеств: одноэлементные множества $\{x_j\}$ и все остальные точки оси. С этой точки зрения $A(\tau_i, x_j)$ можно рассматривать как тривиальную кусочно постоянную функцию, если считать, что вне точек решётки её значение равно $+\infty$. Нетривиальные функции $A(t, x)$ будут рассмотрены ниже.

3 Задачи оптимального управления с фиксированным временем

3.1 Ломаные в обобщённом смысле

Рассмотрим теперь задачу оптимального управления

$$J\left([t_0,t_1],x(.),u(.)\right)=\int_{t_0}^{t_1}L(t,x(t),u(t))dt\to\min,$$

$$\frac{dx(t)}{dt}=f(t,x(t),u(t)), \tag{1}$$

$$x(t_0)=x_0,\quad x(t_1)=x_1,\quad t_0<t_1,\quad x(t)\in X,\quad u(t)\in U.$$

Для её численного решения, так же как в вариационном исчислении, мы разбиваем отрезок времени $[t_0,t_1]$ на m равных частей точками

$$t_0=\tau_0,\tau_1,\tau_2,\ldots,\tau_i,\ldots,\tau_m=t_1,\quad \tau_i-\tau_{i-1}=\Delta\tau=\frac{t_1-t_0}{m}.$$

При решении задачи вариационного исчисления мы рассматривали кусочно постоянные скорости. Теперь будем рассматривать *управления*, постоянные на промежутках $[\tau_{i-1},\tau_i)$. Для удобства описания ограничимся простым случаем, когда управляющие воздействия принадлежат заранее заданному конечному множеству

$$V\subset U,\quad V=\{v_j\},\quad j=1,2,\ldots l.$$

В общем случае число l и сам набор управлений V могут меняться от точки к точке. Чтобы задать траекторию γ, начинающуюся в (t_0,x_0), достаточно выбрать конечную последовательность управлений $u_i\in V$ на промежутках $[\tau_{i-1},\tau_i)$. Мы будем писать

$$\gamma=\left(\xi_0;u_1,\xi_1;\ldots;u_{\mu-1},\xi_{\mu-1};u_\mu,\xi_\mu\right),\quad u_i\in V,\quad \xi_i\in X,\quad 1\le\mu\le m, \tag{2}$$

где $\xi_0=x_0$, ξ_i – состояние, в котором окажется система в момент τ_i. Такие траектории мы будем тоже называть *ломаными*, а участки с постоянными управлениями – *отрезками*. Пусть Γ – множество ломаных. Значение функционала J на ломаной γ будем называть стоимостью и обозначать $c(\gamma)$.

Все ломаные, которые строятся алгоритмом, начинаются в одной точке (t_0,x_0). Здесь, так же как в вариационном исчислении, переходя к ломаным,

мы отказываемся от точного решения задачи, но чем мельче шаг $\Delta\tau$ и шире множество V, тем ближе мы подходим (по функционалу) к глобальному экстремуму.

Отдельно взятый отрезок есть функция $x(t)$, определённая на промежутке времени $[\tau_{i-1}, \tau_i]$, $1 \leq i \leq m$, и удовлетворяющая дифференциальному уравнению

$$\frac{dx(t)}{dt} = f(t, x(t), v_k), \quad v_k \in V, \quad 1 \leq k \leq l \tag{3}$$

при
постоянном управлении v_k. Отрезок имеет начальную точку $(\tau_{i-1}, x(\tau_{i-1}))$, конечную точку $(\tau_i, x(\tau_i))$ и стоимость

$$J([\tau_{i-1}, \tau_i], x(.), v_j) = \int_{\tau_{i-1}}^{\tau_i} L(t, x(t), v_j)dt.$$

Опираясь на теорему о единственности решения задачи Коши, можно считать, что все эти объекты однозначно определяются номером отрезка времени i, номером управления k и начальной точкой $a = x(\tau_{i-1})$. Поэтому для отрезка (3) обозначим

$$b(i, k, a) = x(\tau_i), \quad c(i, k, a) = \int_{\tau_{i-1}}^{\tau_i} L(t, x(t), v_k)dt. \tag{4}$$

Сложность задачи состоит в следующем. Всего при $\mu = m$ существует l^m последовательностей вида (2). Важно, что каждая ломаная проходит по своим собственным точкам, т.е. ломаные, вообще говоря, не имеют общих точек излома, за исключением начальной. Заметим, что число промежуточных вершин теперь растёт в зависимости от m как показательная функция, в отличие от линейной зависимости в вариационном исчислении, где число вершин равно mn. На графе, число вершин которого равно, например, $3^{100} \approx 5 \cdot 10^{47}$, найти оптимальный путь практически невозможно. Если попытаться, как в вариационном исчислении, двигаться только по узлам заранее заданной регулярной решётки (τ_i, x_j), то возникнут трудности при выборе управлений $u \in V$, ведущих из точек (τ_{i-1}, x_j) в соседние точки (τ_i, x_k), а "элементарная операция" Н.Н. Моисеева [5] не всегда приводит к успеху.

3.2 Разбиение на ячейки

Предлагается следующая схема решения задачи. Надо разбить множество X на конечное число (n) подмножеств

$$X = \bigcup_{j=1}^{n} X_j, \quad X_j \bigcap X_k = \varnothing,$$

которые удобно называть *классами*, или *ячейками*. В качестве рекомендации заметим, что диаметры множеств X_j должны быть по возможности небольшими. В процессе работы алгоритма из двух точек, попавших в момент τ_i в одну и ту же ячейку, менее перспективная точка удаляется, так что всегда в памяти хранится не более mn точек, как в вариационном исчислении. Если $x \in X_j$, то будем писать $\bar{x} = j$. Элементы $x, y \in X_j$ одной ячейки X_j будем считать эквивалентными: $x \sim y$. Как уже говорилось, можно сделать так, чтобы каждая ячейка, за исключением одной, состояла только из одной точки, а все остальные точки были бы включены в общую ячейку, и тогда возникла бы регулярная решётка на плоскости, как в вариационном исчислении.

Пусть

$$(\xi_0, \xi_1, \ldots, \xi_\mu) \tag{5}$$

– ординаты точек излома ломаной (2). Полагая $\bar{\xi}_i = \nu_i$, получаем последовательность целых чисел

$$(\nu_0, \nu_1, \ldots, \nu_\mu)\,, \tag{6}$$

представляющих собой номера ячеек, по которым проходит ломаная. Предполагается, что программа содержит процедуры, которые точно или приближённо вычисляют значения функций $b\,(i, j, a)$ и $c\,(i, j, a)$ для произвольных целых чисел $i = 0, 1, \ldots, m$, $j = 1, \ldots, l$ и вещественного числа $a \in X$ (4).

3.3 Алгоритм оптимизации

В процессе работы программа формирует массивы

$$s[i, j], \quad \varphi[i, j], \quad w[i, j], \quad A[i, j], \quad i = 1, \ldots, m, \quad j = 1, \ldots, n,$$

которые интерпретируются следующим образом: $\sigma = s[i, j]$ – номер предыдущей ячейки X_σ; $x = \varphi[i, j] \in X_j$ – "лучшая" точка ячейки X_j, в которую попадает система в момент τ_i; $u = w[i, j] \in V$ – управление, приводящее в точку $(\tau_i, \varphi[i, j])$; $A[i, j]$ – функция действия по Гамильтону (аналог функции Беллмана, если время обращено вспять). По смыслу функция $A(i, j)$

представляет собой оценку стоимости оптимальной ломаной, начинающейся в (t_0, x_0) и оканчивающейся в момент τ_i в ячейке X_j. Программа в целом состоит из четырёх последовательно выполняемых процедур. В фигурных скобках помещаются комментарии.

Программа "Решение задачи оптимального управления с фиксированным временем в классе ломаных"

{*Задать стартовые значения функции действия:*}

$A\,[0, \bar{x}_0] := 0$;

$A\,[i, j] := \infty$ для всех $i, j > 0$;

Заполнить левую вертикаль $t = \tau_1$;

Заполнить остальные вертикали $t = \tau_i, \quad i > 1$, **слева направо**;

Движением вспять построить ломаную;

Конец программы

Теперь опишем процедуры, составляющие программу.

Процедура "Заполнить левую вертикаль"

for $k := 1$ to l do begin {*перебор всех управлений из* V}

$y := b\,(1, k, x_0)\,; \quad \nu := \bar{y}; \quad r := c\,(1, k, x_0)\,;$ {*следующая точка* (τ_1, y) *и действие* $r = A(1, \bar{y})$}

if $r < A\,[1, \nu]$

then begin {*величину* $A\,[1, \nu]$ *удалось уменьшить*}

$s\,[1, \nu] := \bar{x}_0$; {*начальный класс*} $A\,[1, \nu] := r$; {*действие*}

$\varphi[1, \nu] := y; \quad w[1, \nu] := k$; {*управление* $\bar{x}_0 \to \nu$}

end;

end;

Конец процедуры

Процедура "Заполнить остальные вертикали"

for $i := 2$ to m do {*обработка вертикалей* $i > 1$ *слева направо*}

for $j := 1$ to n do begin {движение по ячейкам при $t = \tau_{i-1}$}

$x := \varphi[i-1, j]; \quad p := A\,[i-1, j]\,;$

for $k := 1$ to l do begin {*перебор всех управлений из* V}

$y := b\,(i,k,x)\,;\quad \nu := \bar{y};\quad r := p + c\,(i,k,x)\,;$ {следующая точка (τ_i, y) и действие $r = A(i,\bar{y})$}

if $r < A\,[i,\nu]$

then begin *{величину $A\,[i,\nu]$ удалось уменьшить}*

$s\,[i,\nu] := j$; {предыдущий класс} $A\,[i,\nu] := r$; *{действие}*

$\varphi[i,\nu] := y;\quad w[i,\nu] := k$; {управление $j \to \nu$}

end;

end;

end;

Конец процедуры

Процедура "Построить γ, начинающуюся в (t_0, x_0) и приводящую систему в момент $t_1 = t_0 + m \cdot \Delta\tau$ в точку x_1"

$j := \bar{x}_1;\quad y := \varphi[i,j];$

if $x_1 \neq y$ then STOP; {*ломаная не существует*}

$u := w[m,j];\quad j := s\,[y,u]\,;$

$\gamma := (y,u)$; {*построение начинаем с конца; ломаная пока состоит из пары* (y,u)}

for $i := m-1$ by -1 to 1 do begin {*движение вспять*}

$y := \varphi[i,j];\quad u := w[i,j];\ \ j := s\,[x,u]\,;$

$\gamma := (y,u) \vee \gamma$; {*конкатенация: слева к построенной траектории* γ *добавляется пара* (y,u)}

end;

$\gamma := x_0 \vee \gamma;$

Конец процедуры

В конце процедуры будет построена последовательность, состоящая из координат и управлений

$$\gamma = (\xi_0; u_1, \xi_1; \ldots; u_{m-1}, \xi_{m-1}; u_m, \xi_m)\,, \tag{7}$$

причём $\xi_0 = x_0$, $\xi_m = x_1$.

3.4 Обоснование алгоритма

Здесь будет показано, что при любом достаточно мелком разбиении X на ячейки алгоритм строит оптимальные ломаные.

Зафиксируем произвольное натуральное число $\mu \le m$ и рассмотрим множество ломаных (6), состоящих из μ отрезков. Всего таких ломаных имеется конечное число, значит, конечно и множество $Y_\mu = \{x_\mu\}$ их крайних правых узлов (точек). Пусть ρ_μ – минимальное расстояние между парами точек множества Y_μ, и пусть $r = \min[r_\mu | 1 \le \mu \le m]$. Ясно, что $r > 0$.

Теорема. Если диаметры всех множеств X_j меньше r, то программа строит оптимальную ломаную, ведущую из (t_0, x_0) в (t_1, x_1), а стоимость этой ломаной равна $A[m, j]$, где $m = \frac{t_1 - t_0}{\Delta\tau}$, $j = \bar{x}_1$.

$\lhd$ Рассмотрим произвольные натуральные $\mu \le m$ и $j \le n$. Поскольку расстояния между точками множества Y_μ больше диаметра множества X_j, то множество $Y_\mu \cap X_j$ состоит не более, чем из одного элемента. Обозначим этот элемент, если он существует, через $z_{\mu j}$. Значит, любая ломаная γ, оканчивающаяся или проходящая в момент τ_μ через ячейку X_j, имеет в этом множестве узел $z_{\mu j}$. Если же $Y_\mu \cap X_j = \varnothing$, то через ячейку X_j в момент τ_μ не проходит ни одна ломаная. Для этой ячейки в конце программы останется $A[\mu, j] = \infty$.

Теперь докажем теорему индукцией по шагу времени m. Для $m = 0$ это очевидно, поскольку ломаная, состоящая из 0 шагов, вырождается в точку (t_0, x_0) и, по построению, $A[0, \bar{x}_0] := 0$.

Предположим, что утверждение верно для $\mu = m - 1$ шагов; докажем его для шага m. Пусть (7) – оптимальная ломаная. Тогда

$$\gamma_1 = (x_0; u_1, \xi_1; \ldots; u_{m-1}, \xi_{m-1})$$

– тоже оптимальная ломаная, но ведущая в (τ_{m-1}, ξ_{m-1}). По индуктивному предположению, она или равная ей по стоимости ломаная была бы построена алгоритмом, если бы точка (τ_{m-1}, ξ_{m-1}) оказалась конечной. При этом $c(\gamma_1) = A[m-1, j]$, где $j = \bar{\xi}_{m-1}$. Пусть $x = z_{m-1,j}$. Посмотрим, что происходит при заполнении вертикали m на этапе, когда программа рассматривает отрезки, начинающиеся в (τ_{m-1}, ξ_{m-1}). При управлении $u_m = v_k \in V$ система попадает в (τ_m, ξ_m). Стоимость этого шага равна $c(m, k, x)$.

Значит, по индуктивному предположению, в алгоритме

$$r = c(\gamma_1) + c(i, k, x) = c(\gamma).$$

Последнее равенство справедливо в силу аддитивности функционала стоимости. В результате заведомо будет $A\left[m, \bar{\xi}_m\right] \leq c(\gamma)$. С другой стороны, строгое неравенство $A\left[m, \bar{\xi}_m\right] < c(\gamma)$ невозможно, так как в противном случае траектория γ была бы неоптимальной. $\triangleright$

3.5 Экономический пример

Рассмотрим задачу оптимального управления с экономическим содержанием [12].

$$J = \int_0^T \left[u^2(t) + x(t)\right] dt \to \min,$$

$$\frac{dx(t)}{dt} = u(t), \quad x(0) = 0, \quad x(T) = H \geq 0, \quad 0 \leq u \leq V, \quad x \geq 0.$$

Содержательный смысл задачи состоит в следующем. Фирма должна выполнить заказ на поставку H единиц продукции к моменту времени T. При этом затраты складываются из затрат на хранение уже изготовленного товара и затрат на интенсивность (скорость u) производства продукции. Для этой задачи в [12] было построено оптимальное центральное поле экстремалей Понтрягина. Кроме того, задача решалась численно с помощью рассмотренного выше разбиения на ячейки. Итоги обоих исследований демонстрируются на рис. 2. Пунктиром на левом рисунке изображена линия перехода на максимальную скорость производства V.

4 Автономные задачи оптимального управления

Метод разбиения на классы для автономных задач оптимального управления

$$\int_0^T L(t, x(t), u(t))dt \to \min,$$

$$\frac{dx(t)}{dt} = f(x(t), u(t)),$$

$$x(0) = x_0, \quad x(T) = x_1, \quad x(t) \in X, \quad u(t) \in U,$$

(время окончания процесса $T > 0$ не фиксировано) подробно рассматривался в [2]-[3]. Состояния $x(t)$ и управления $u(t)$, вообще говоря, являются векторами.

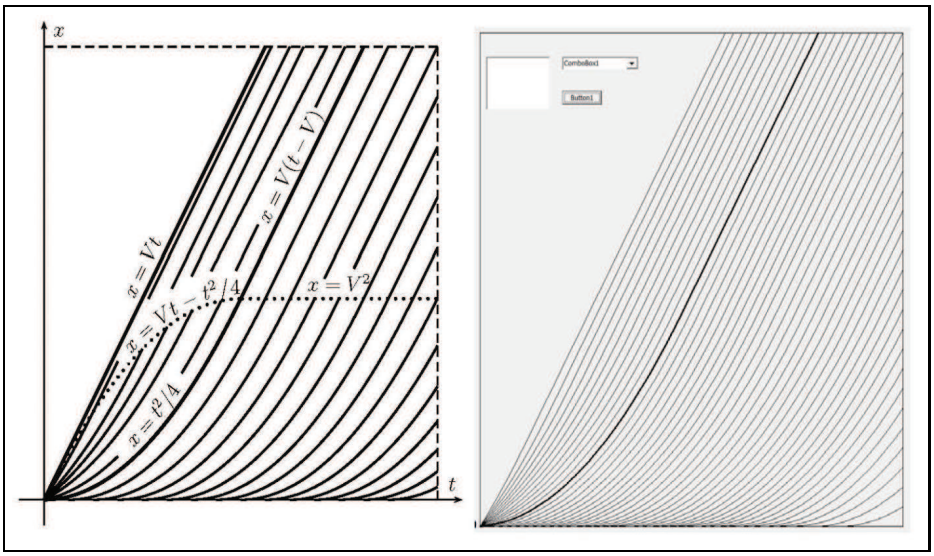

Рис. 2: Оптимальное поле экстремалей и квази-оптимальное поле ломаных в задаче с экономическим содержанием

Для численного решения задачи в простом варианте нужно выбрать конечное множество $V \subset U$ и шаг времени $\Delta\tau$. Тем самым ломаные уже определены. Они состоят из отрезков, т.е. решений дифференциального уравнения $\dot{x} = f(x, u)$ на отрезке $[0, \Delta\tau]$ при постоянном управлении $u \in V$. Теперь надо разбить X на конечное число классов X_i и запустить программу, представляющую собой обобщение какого-либо алгоритма поиска кратчайшего пути на графе [2]-[3]. Это может быть поиск в ширину, например, алгоритм Дейкстры (равных цен) или Нильссона. Но может быть и поиск в глубину с возвращениями [8]-[9]. А может быть и какая-либо комбинация этих алгоритмов. Такой инструментарий может быть очень важным при решении более сложных задач динамической оптимизации, при разного рода дефиците информации.

Рассмотрим несколько примеров на эту тему.

4.1 Задача Бушоу

Данная задача оптимального быстродействия (её называют ещё задачей об управляемой остановке) может быть сформулирована следующим образом [13]–[14]:

$$\frac{dx}{dt} = \dot{x}, \quad \frac{d\dot{x}}{dt} = u, \quad |u| \le 1, \quad x(0) = x_0, \quad x(T) = x_1, \quad T \to \min\,.$$

На рис. 3 изображены центральные поля экстремалей и ломаных при фиксированной точке старта [10].

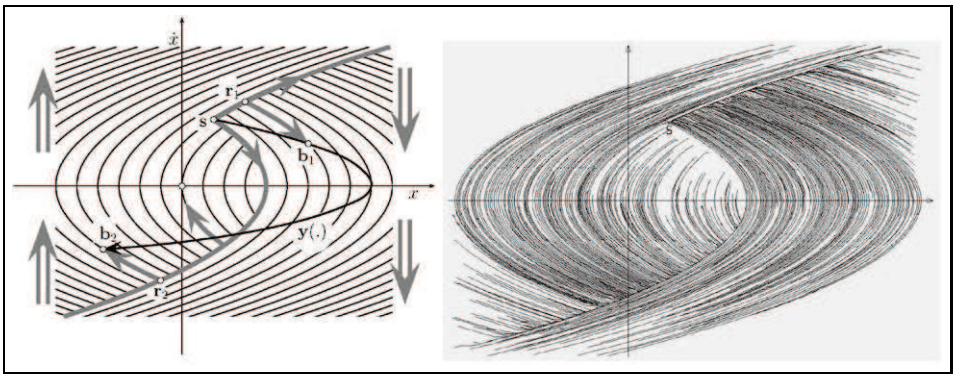

Рис. 3: Центральные поля экстремалей и ломаных в задаче Бушоу

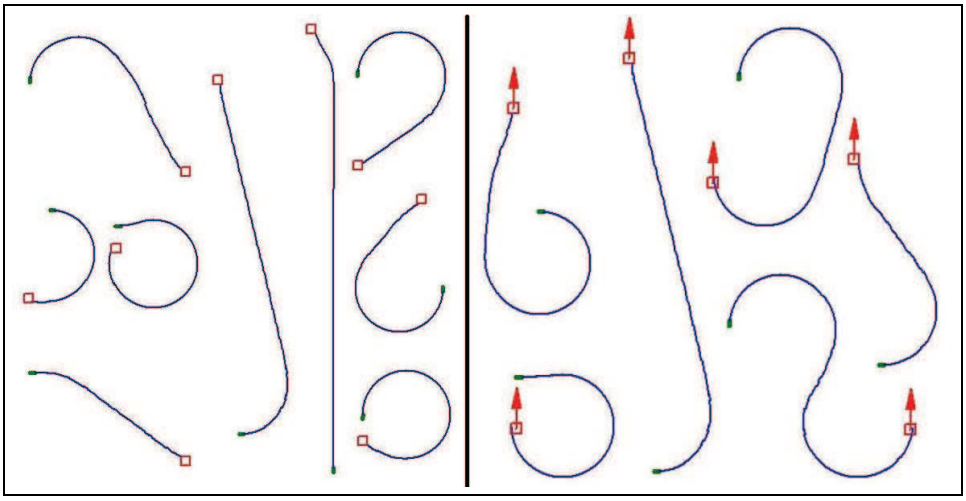

Рис. 4: Движение автомобиля к месту стоянки

4.2 Движение автомобиля к месту стоянки

Эта задача довольно подробно описана в [15]. Внешне кажется, что в каждом конкретном случае задача решается с помощью циркуля и линейки. Тем не менее, точное аналитическое решение авторам статьи неизвестно. Считая линейную скорость материальной точки (x, y) постоянной, а кривизну – ограниченной, задачу можно описать следующим образом:

$$\frac{dx}{dt} = \cos\varphi, \quad \frac{dy}{dt} = \sin\varphi, \quad \frac{d\varphi}{dt} = u, \quad |u| \le 1, \quad x(0) = x_0, \quad x(T) \in G, \quad T \to \min.$$

Под стоянкой можно понимать точку или её окрестность. Заметим, что пространство состояний трёхмерно, поэтому точка на плоскости (x, y) соответствует целому множеству состояний (x, y, φ), где φ – произвольный угол.

На рисунке 4 показаны результаты численного решения задачи при разных начальных и терминальных условиях. Во всех случаях требуется попасть в небольшой квадрат. С левой стороны рисунка разрешено попадать в квад-

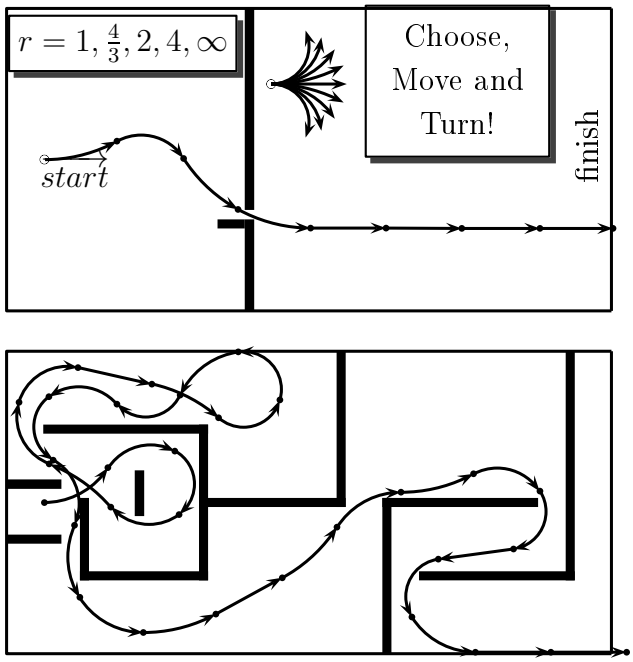

Рис. 5: Выход "автомобиля" из лабиринта

рат под любым углом. С правой – надо попасть под углом, заданным стрелкой, точнее, под достаточно близким к стрелке углом. Таким образом, в первом случае целевым множеством является *двумерная* окрестность точки (x, y), во втором – в трёхмерную окрестность точки (x, y, φ). Конечно, на левой части рисунка задача несколько проще, чем на правой. Это видно и по форме "ломаных".

На рисунке 5 изображено решение задачи выхода автомобиля из лабиринта. Требуется пересечь финишную "ленточку", минуя препятствия. Здесь область движения (прямоугольник 8×16) разбивается на квадраты со стороной $\frac{8}{10}$, а множество направлений – на отрезки длиной $\frac{2\pi}{15}$. Таким образом, получается $10 \times 20 \times 30 = 3000$ ячеек. Эта задача и её решение могут быть использованы в робототехнике и теории искусственного интеллекта. Поэтому в [2]–[3] было предложено использовать данную модель для тестирования методов решения задач динамической оптимизации. Но с тех пор никаких других методов, с помощью которых можно было бы решать задачу выхода автомобиля из лабиринта, так и не появилось.

5 Дифференциальные игры

Здесь будут рассмотрены три дифференциальные игры двух игроков, описанные в [15]. Во всех трёх играх линейные скорости игроков P (преследователь, первый игрок) и E (преследуемый, второй игрок) постоянны, причём скорость игрока P больше скорости игрока E.

5.1 Простое преследование

В этой игре соперники совершают простое движение. Каждый из них движется со своей постоянной линейной скоростью, меняя направление движения по своему усмотрению. Партия заканчивается, как только произойдёт захват, т.е. P окажется в заданной окрестности E. Запишем уравнения движения игроков:

$$\dot{x}_1 = v_1 \cos\varphi_1, \;\; \dot{y}_1 = v_1 \sin\varphi_1, \;\; \dot{x}_2 = v_2 \cos\varphi_2, \;\; \dot{y}_2 = v_2 \sin\varphi_2, \;\; v_1 > v_2 > 0,$$

где $\varphi_1(t)$ и $\varphi_2(t)$ – управления игроков. Оптимальным для каждого из игроков будет движение по прямой PE, причём P движется по направлению к E, а E – в противоположную сторону от P. Эта задача была запрограммирована авторами, но рисунок хода игры не представляет интереса, т.к. кроме отрезков прямой линии PE мы ничего не увидим. Пространство состояний здесь одномерно – состояние полностью определяется расстоянием между партнёрами. В программе было задано максимальное расстояние R (т.е. расстояние, при котором игра заканчивалась победой игрока E), а отрезок $[0, R]$ был, как обычно, разбит на конечное число ячеек.

5.2 Шофёр-убийца

Водитель стремится за минимально возможное время достичь окрестности пешехода при уравнениях движения

$$\begin{aligned} &\dot{x}_1 = v_1 \cos\varphi_1, \;\; \dot{y}_1 = v_1 \sin\varphi_1, \quad \dot{\varphi}_1 = \tfrac{v_1}{r} u, \\ &\dot{x}_2 = v_2 \cos\varphi_2, \;\; \dot{y}_2 = v_2 \sin\varphi_2, \;\; v_1 > v_2 > 0. \end{aligned}$$

Игра изучалась многими авторами [15],[16], но, повидимому, точное аналитическое решение пока не найдено. На рис. 6 изображено несколько партий игры, построенных компьютером. Каждый партнёр после стадии самообучения действует оптимальным для себя образом.

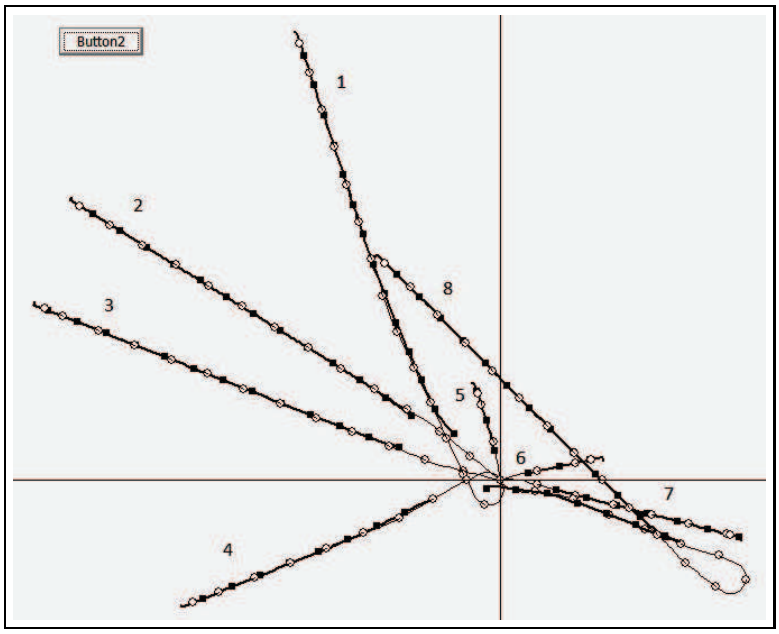

Рис. 6: Несколько партий игры шофёр-убийца

В этой игре пространство состояний двумерно – состояние полностью определяется расстоянием между игроками и углом между линией визирования PE и скоростью игрока P.

На рисунке 6 изображено 8 партий, каждая из которых закончилась победой игрока P. Траектория P обозначена обычной линией, а траектория E – жирной. Кроме того, вдоль пути каждого игрока поставлены вехи через одинаковые промежутки времени. Вехи игрока P представляют собой полые окружности, а вехи игрока E – сплошные квадраты. Поэтому в любой партии до точки встречи было одинаковое количество кружков и квадратов (последняя и первая вехи не всегда пропечатывалась). Начальным положение игрока P всегда было начало координат, а начальный угол P и обе начальные координаты E задавались случайным образом. В партиях под номерами 2,3,5,6,7 игрок P легко настигал соперника, так как не приходилось делать сложный манёвр разворота, поскольку начальная скорость P была направлена примерно на E. В партиях 1,4 игрок P сразу делал нужный разворот. В начале партии 8 игрок пристроился в "хвост" игроку P, чтобы тому было трудно развернуться. Но, обладая большей скоростью, игрок P оторвался от преследования и, сделав петлю, нацелился на E. Чтобы продлить время игры, игроку E пришлось поменять направление, но это только отсрочило его поражение. Более отчётливо этим манёвры видны на 7.

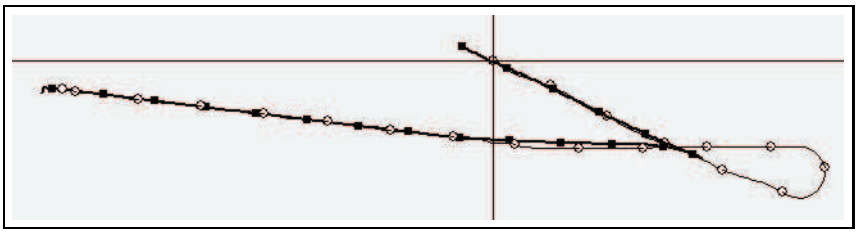

Рис. 7: Игра шофёр-убийца. Игрок E временно заходит в "хвост" игроку P

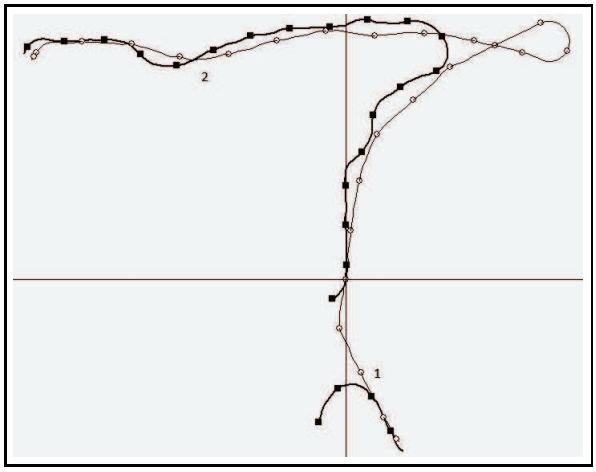

Рис. 8: Две партии игры двух автомобилей

5.3 Игра двух автомобилей

Уравнения движения игроков имеют одинаковый вид: при соотношении линейных скоростей $v_1 > v_2 > 0$ выполнены равенства

$$\dot{x}_1 = v_1 \cos\varphi_1, \quad \dot{y}_1 = v_1 \sin\varphi_1, \quad \dot{\varphi}_1 = \tfrac{v_1}{r_1} u_1,$$
$$\dot{x}_2 = v_2 \cos\varphi_2, \quad \dot{y}_2 = v_2 \sin\varphi_2, \quad \dot{\varphi}_2 = \tfrac{v_2}{r_2} u_2.$$

Пространство состояний трёхмерно. Его координатами являются расстояние между игроками и углы между линией визирования PE и скоростями игроков. Оптимальные действия игроков, найденные программой в двух партиях, показаны на рис. 8. Минимальные радиусы поворота у партнёров были одинаковы. В первой партии показан пример движения в некоторой степени навстречу. Во второй же партии игрок E, оказавшись сзади игрока P, пристроился было ему в "хвост" но потом пришлось сменить направление движения.

Во всех трёх играх работа программы состоит из двух этапов. На первом этапе (имитационное моделирование) методом Монте-Карло проигрывались одноходовые сценарии, на втором этапе происходило тестирование. При этом каждый игрок, не зная реальной стратегии другого, вырабатывал свою соб-

ственную цену игры. Основу логики самообучения игроков составлял принцип априорного оптимизма. А именно, в начальный момент игрок P считал, что цена игры тождественно равна нулю, а игрок E – что цена игры тождественно равна бесконечности. В дальнейшем по результатам обучения цена игры для P увеличивалась, а для E – уменьшалась. В процессе самообучения игрок P вырабатывал стратегию по принципу минимакса, а E – по принципу максимина.

6 Оптимизация при дефиците информации

Задачи динамической оптимизации не исчерпываются моделями оптимального управления, когда решения принимаются в условиях полной информированности, и динамическими играми, когда отсутствуют сведения о том, как будет действовать противник. Возможны и другие формы дефицита информации. Например, поведение среды может описываться случайными законами. Остановимся на модели, в которой блоку управления неизвестны уравнения движения объекта. Именно так обстоит дело в биологических системах. Рассмотрим известный пример удержания шеста в вертикальном положении неустойчивого равновесия [17],[18].

6.1 Задача о плоском перевёрнутом маятнике

Маятник шарнирно прикреплён к тележке, поднят над шарниром и удерживается около положения неустойчивого равновесия. Нужно, чтобы время, в течение которого маятник не падает, было максимально возможным. Считая тележку массивным телом, получаем систему уравнений Лагранжа II рода

$$\ddot{x} = u, \quad l_0\ddot{\theta} + u\cos\theta - g\sin\theta = 0, \tag{8}$$

где x – положение тележки, u – её ускорение, которое является управлением, θ – угол между осью Oy и стержнем, l_0 – приведённая длина маятника, g – ускорение свободного падения. В численных экспериментах были выбраны следующие значения параметров и диапазоны фазовых координат: $g = 980$ см/с^2, $u = \pm 1000$ см/с^2, $l_0 = 100$ см, $|\theta| \le 0.2$ рад., $|x| \le 240$ см, $\Delta\tau = 0.02$ с.

Пространство состояний четырёхмерно, координатами состояния являются положение (x) и скорость $(\dot{x})$ тележки, угол отклонения стержня (θ) и его угловая скорость $(\dot{\theta})$. Следуя Д.Мики [18], разобьём пространство состояний на 162 класса с помощью пороговых значений: по углу ± 0.1; ± 0.017; 0, по

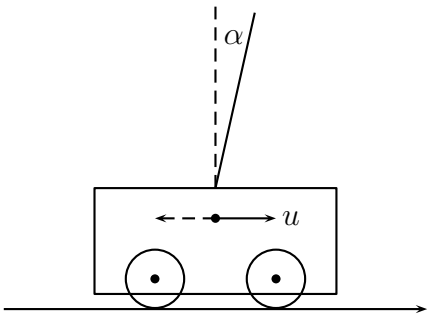

Рис. 9: Перевёрнутый маятник на тележке

угловой скорости ± 0.87, по положению тележки ± 80, по скорости тележки ± 50.

В каждый момент процесса функционирования блоку управления известен только порядковый номер класса, в котором находится объект. На основе только этой регулярно поступающей информации блок должен принимать решение: $u = 1000$ или $u = -1000$. Платой за незнание динамики является удвоение используемой памяти: действие теперь зависит не только от классов состояния, но и от управления ± 1000. Алгоритм принятия решений и самообучения может быть разным. В программе Мики в основе обучения лежал эмпирический подход. После некоторого периода обучения стержень, хотя и долго удерживался в вертикальном положении, но всё-таки в конце концов падал. В программе авторов статьи использовался другой алгоритм, основанный на идее априорного оптимизма и методе встречной коррекции [19]; в итоге по окончании стадии самообучения стержень удерживался около вертикального положения практически бесконечно долго.

7 Заключение

Для применения прямых методов динамической оптимизации, рассмотренных в статье, нужно, во-первых, чтобы в исходной задаче требовалось найти не локальный, а глобальный экстремум, во-вторых, чтобы попытки исследования и построения аналитического решения наталкивались на значительные трудности. Разбивать множество состояний на классы надо так, чтобы соответствующие массивы могли храниться в памяти компьютера. Диаметры классов должны быть по возможности небольшими.

Процесс формирования кусочно постоянных функций во многом напоминает имитационное моделирование. Как известно, оно применяется тогда,

когда трудно получить решение задачи в виде математических уравнений. При выполнении условий теоремы, доказанной в статье, алгоритм строит оптимальную кривую в классе "ломаных" . Если эти условия не выполнены, всё равно можно говорить об оптимизации, понимая её как процесс, в котором появляются дополнительные ресурсы времени и памяти.

Заметим, что в более простой задаче поиска глобального экстремума функции нескольких переменных вполне можно обойтись приравниванием нулю градиента и простым перебором критических точек. Прямые методы здесь не требуются. Это чётко показано в учебниках математического анализа. Картина резко меняется и прямые методы становятся весьма актуальными при переходе к задачам динамической оптимизации, поскольку иначе придётся строить и анализировать центральные поля экстремалей, а общих подходов здесь не разработано. В статье не только показана актуальность и эффективность прямых методов в задачах динамической оптимизации разного вида, но и продемонстрировано внутреннее и внешнее сходство конструкций аналитических и прямых методов. Это наводит на мысль о том, что в конкретных случаях к успеху может привести комбинированное применение обеих групп методов.

Список литературы

[1] *Орёл Е.Н.* Аппроксимация функции Беллмана кусочно-постоянными функциями // Ж. вычисл. матем. и матем. физ., 1978, № 4, с. 916-927

[2] *Орёл Е.Н.* Метод решения задач оптимального управления // ДАН СССР, 1989, № 6, с. 1301-1304

[3] *Орёл Е.Н.* Алгоритмы поиска квазиоптимального управления, использующие разбиение пространства состояний // Ж. вычисл. матем. и матем. физ., 1990, № 9, с. 1283-1294

[4] *Орёл Е.Н., Орёл О.Е.* Условия оптимальности центральных полей траекторий // Дифференциальные уравнения и процессы управления, 2017, № 1, с. 53-77

[5] Моисеев Н.Н. Элементы теории оптимальных систем // М., Наука, 1975, 528 с.

[6] Эльсгольц Л.Э. Дифференциальные уравнения и вариационное исчисление // М., Наука, 1975, 528 с.

[7] Орёл Е.Н. Построение кратчайшего пути на графе по миноранте функции Беллмана // Автоматика и телемеханика, 1977, № 2, с. 88-91

[8] Орёл Е.Н. Коррекция эвристической функции в процессе поиска и принятия решений // ДАН СССР, 1991, № 4, с. 853-855

[9] Орёл Е.Н. Эвристика обучения в задачах поиска // Техническая кибернетика, 1992, № 5, с. 69-81

[10] *Орёл Е.Н., Орёл О.Е.* Центральные поля оптимальных траекторий // Доклады Академии Наук, 2014, № 4, с. 402-405

[11] *Орёл Е.Н., Орёл О.Е.* Центральное поле траекторий в задачах оптимального управления и вариационного исчисления. // Учёт. Анализ. Аудит, 2015, № 3, с. 35-42

[12] *Орёл Е.Н., Орёл О.Е.* Оптимальное управление процессом производства при выполнении заказа к заданному сроку // Экономика и математические методы, 2016, № 2, с. 65-77

[13] *Понтрягин Л.С., Болтянский В.Г., Гамкрелидзе Р.В., Мищенко Е.Ф.* Математическая теория оптимальных процессов // М., Наука, 1969, 391 с.

[14] *Зеликин М.И.* Оптимальное управление и вариационное исчисление // М., Едиториал УРСС, 2004, 160 с.

[15] *Айзекс Р.* Дифференциальные игры М., Мир 1967, 480 с.

[16] *Breakwell J.V., Speyer J.L., Bryson A.E.* Optimization and control of nonlinear systems using the second variations. // SIAM J. Control 1963, № 1, с. 193-223

[17] *Неймарк Ю.И., Коган Н.Я., Савельев В.П.* Динамические модели теории управления // М, Наука, 1985, 400 с.

[18] *Мичи Д., Чемберс Р.* Компьютер-творец // М., Мир, 1987, 225 с.

[19] *Орёл Е.Н.* Дискретные краевые задачи и процессы оптимизации на графах // Искусственный интеллект в технических системах. – М. РАН, ГосИФТП, 1998, с. 3-33